SSL Downgrade von IOT-Geräten

Seminar "Das Internet der Dinge – Ein Hackerparadies?"

Alexander Korff, Sergej Maul, Yannik Stöcklin
Sebastian Philipp, Daniel Seidinger, Fabian Neumeier
Lukas Stöckli, Manuel Rickli, Désirée Nusch
Samuel Hugger, Clement Francois, Joel Grossenbacher

29. Dezember 2017

1 Lokale Kommunikation mit der Philips Hue

Nach einigen Beobachtungen stellten wir fest, dass in der Zeit zwischen dem Befehl an das Amazon Echo, das Licht anzuschalten, und dem tatsächlichen Aufleuchten der Lampe kein Traffic zwischen der Philips Hue Bridge und dem Internet zu beobachten war. Daher nahmen wir an, dass der Befehl zum Anschalten im lokalen Netzwerk gesendet wird. Lokaler Traffic würde auch nicht in den mitgeschnittenen Paketen enthalten sein, da wir ARP Spoofing nur zwischen der FRITZ!Box und der Hue Bridge betrieben hatten.

Entsprechend änderten wir unser Vorgehen, um den Traffic zwischen dem Echo und der Hue Bridge mitzuschneiden. Dies war sofort erfolgreich; wir konnten feststellen, dass das Echo eine HTTP Request im Klartext an die Hue Bridge sendet. Eine auf das Wesentliche gekürzte Version eines solchen Requests zum Ausschalten einer Lampe ist in Listing **??** dargestellt.

Abgesehen davon, dass der falsche Wert für `Content-Type` angegeben wird (anstatt dem tatsächlichen `application/json`), fielen folgende Punkte auf:

- Die Request enthält eine nicht weiter bekannte Art von Zugriffsberechtigung oder ID in der URL. Dies stellte jedoch kein Hindernis dar; die ID liess sich beliebig oft

```
PUT /api/-UaDNHD5j44y07zYdAOEg0JuIakQpu72ivJXXHVS/lights/2/
   state HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

{"on": false}
```

Listing 1: Ein HTTP Request zum Ausschalten von Lampe Nr. 2.

```
PUT /api/-UaDNHD5j44y07zYdAOEg0JuIakQpu72ivJXXHVS/lights/2/
   state HTTP/1.1
Content-Type: application/json
Content-Length: 42

{"on": true, "bri": 255, "xy": [1.0, 1.0]}
```

Listing 2: Ein HTTP Request für bunte Lampen.

wiederverwenden.

- Die Lampe, die angesprochen wird, wird ebenfalls numerisch in der URL übergeben, in diesem Fall Lampe Nr. 2.
- All dies geschieht komplett unverschlüsselt; ein SSL Downgrade war daher gar nicht notwendig.

Analog konnten wir die Befehle zum Ausschalten, zum Setzen der Helligkeit sowie der Farbe mitschneiden und auch problemlos selbst versenden. Da sich die Farbe mit dem Amazon Echo nur verändern lässt, wenn man einen Philips Hue Account verwendet, schnitten wir hierzu den Datenverkehr zwischen der Hue Bridge und einem Smartphone mit, auf welchem die zugehörige Software verwendet wurde, welche das Setzen der Farbe auch ohne Login erlaubt. Auch hier war wieder keine Verschlüsselung im Spiel.Mit dem HTTP Request in Listing **??** wird die Lampe angeschalten und auf eine gegebene Helligkeit und Farbe gesetzt. Die Helligkeit ist eine einzelne Zahl im Bereich zwischen 0 und 255. Die Farbe wird durch zwei Werte "x" und "y" zwischen 0 und 1 beschrieben; die Bedeutung dieser Werte erschloss sich uns jedoch nicht. Wir stellten hierzu zwei Hypothesen auf:

- Die Werte ensprechen den (normalisierten) Koordinaten der Farbe im Farbauswahldialog der Smartphone-App.
- Die Werte entsprechen dem Farbton und der Sättigung im HSV-Farbmodell; die Helligkeitskomponente wird separat übergeben.

Beide Hypothesen liessen sich durch Versuche falsifizieren. Die API ist zwar vom Hersteller dokumentiert, aber erfordern wiederum einen Login, um darauf zuzugreifen. [**philips:developer**] Aus Zeitgründen wurde diese API nicht weiter untersucht.

2 LED Matrix

Die in einem früheren Projekt der Vorlesung *Computer Architecture and Operating Systems* gebaute LED Matrix wurde im Verlauf des Seminars bereits um einige Funktionalitäten im Bereich des Internets der Dinge erweitert. Unter anderem ist es möglich mittels

Sprachbefehlen (Amazon Alexa) das aktuelle Wetter, eine Notiz oder auch verschiedene Farben anzuzeigen zu lassen. Die Matrix wird mit einem Arduino Mega angesteuert, welcher seine Informationen von einem eigenen Webserver abruft. Bisher fand diese Kommunikation mit dem Internet über eine unverschlüsselte Verbindung mittels einer sogenannten Ethercard, einer Ethernet Erweiterung für den Arduino, statt.

Zum Testen der zahlreichen Angriffsmethoden auf SSL musste also zuerst eine verschlüsselte Verbindung hergestellt werden, um im Anschluss versuchen zu können diese zu knacken. Da der Arduino selbst nicht genug Rechenleistung zum herstellen einer SSL Verbindung besitzt musste das Setup entsprechend verändert werden. Dazu wurde die Ethercard durch einen WiFi-Chip des Typs ATWINC1500 ersetzt, welcher die benötigte SSL Funktionalität mit sich bringt.

Nach Umrüstung der Hardware musste die Software entsprechend angepasst werden, um vom neuen Netzwerkinterface Gebrauch zu machen. Mit der Arduino WiFi101-Bibliothek [**arduino:WiFi**] wird eine SSL Verbindung zum Webserver aufgebaut. Der Webserver selbst war bereits für SSL eingerichtet, womit nur Änderungen beim Arduino-Client nötig waren. Die SSL Zertifikate werden manuell mittels eines Firmware-Updaters auf das WiFi-Modul geladen werden.

```
if (client.connect(server, 443)) {
...
}
```

Listing 3: Port 443 ohne Fallback auf Port 80

Nach Implementierung aller Änderungen wurden diverse Angriffsversuche unternommen. Das blockieren von Port 443, um eine SSL Verbindung zu unterbinden und einen Fallback auf Port 80 zu erzwingen, erwies sich wie vorhergesehen als erfolglos, da schlicht kein Fallback im Code **??** vorgesehen ist und der Arduino ausschliesslich auf Port 443 einen Verbindungsaufbau versucht. Auch die nächste Methode SSL-Strip blieb erfolglos aufgrund der expliziten Nutzung von SSL ohne Fallback. Zuletzt wurde mittels SSL-Sniffing probiert ob gefälschte Zertifikate vom WiFi-Chip akzeptiert werden um die Verschlüsselung zu umgehen, jedoch erwies sich auch diese Methode als nutzlos, da nur die korrekt eingespielten SSL Zertifikate akzeptiert werden.

Die Matrix ist somit gegen allen getesteten Angriffsmethoden immun, aufgrund von korrekter Konfiguration und strikter Verwendung von SSL.

3 Amazon Echo

4 Quellenverzeichnis