Internet of Secured Things

Oder: Mein Projekt wird nicht Teil des Problems

Alexander Schreiber <als@thangorodrim.ch>

http://www.thangorodrim.ch/

Chemnitzer Linux-Tage 2018, 2018-03-10

A computer lets you make more mistakes faster than any invention in human history, with the possible exceptions of handguns and tequila.

– Mitch Ratcliffe, Technology Review, April 1992

Inhalt

Über den Autor

- beschäftigt sich seit über 20 Jahren mit Linux
- tätig als Systemingenieur bei YouTube in Zürich
- hat den einen oder anderen Computer im Haus

Internet of Things

In Bildern . . .

image credit: Wikipedia

. . . und Worten

- Internet of Things == Computer in alltäglichen Dingen
- → Computer **überall**:
- Fernsehgeräte (Smart-TVs), mit Kameras, Mikrofon, WiFi,
- Türschloss (Smart Locks), Kühlschrank, Smart Meter,
- Beleuchtung (Lichttechnik per WiFi steuerbar),
- Smart Assistants (Alexa & Co.),
- aber auch: Sensornetzwerke für Industrie & Landwirtschaft,
- vernetzte Computer in Fahrzeugen,
- Expertenschätzung: 30 Billionen Geräte im Jahr 2020

Schöne neue Welt!

- Wohnung/Haus vom Urlaub aus beobachten via Kamera
- Heizung remote steuern
- E-Werk Ableser muss nicht mehr ins Haus
- tolle Lichter mit Smartphone steuern
- Smartlocks: man braucht keine Schlüssel mehr
- auf Zuruf Sachen bestellen (z.B. Alexa)
- und viele andere tolle Sachen

IoT-Probleme, nicht sicherheitsrelevant

- verwendete 3rd party APIs abgeschaltet, Funktionsverlust
- Gerät basiert auf Cloud-Service dieser wird abgeschaltet
- Gerät funktioniert nicht ohne Internet-Zugang
- Funktionsausfall wegen fehlerhaften Updates (Smartlock)

IoT-Probleme, sicherheitsrelevant

- keine oder kaum Softwareupdates
- "Lasst uns Geräte direkt ans Internet hängen, ohne Absicherung & ohne Sicherheitsupdates, was soll schon passieren?"
- $\rightarrow$ 87% IoT Geräte deswegen als angreifbar eingestuft
- IoT-Geräte von Angreifern übernehmbar, Botnetze (z.B. Mirai & Co.)
- Geräte durch Angreifer übernehmbar, als Wanzen missbraucht
- Datenlecks durch ungesicherte Kommunikation zwischen IoT sowie IoT und Cloud

Beispiele aus der Realität

- Lockstate.com Smartlocks: Softwareupdate → zerstört (500 Kunden, Wartezeit auf Ersatz 1-2 Wochen)
- Mirai: IoT botnet, for DDoS Angriffe (u.a. krebs.com) verwendet, 2,5 Mio Geräte maximal, via Telnet (bruteforce), Kollateralschaden: 900000 Telekom-Router (von Arcadyan) gecrasht wegen defektem Exploit
- IoT_Reaper: Mirai-Erbe, aber nutzt remote exploits
- BASHLITE: IoT DDoS Netz, bis 1 Mio Geräte, bis 400 GBps
- Linux.Darlloz: Angriff via PHP, Miner für Mincoin, Dogecoin
- Remaiten: Passwort brute force Angriff per Telnet
- BrickerBot: zerstört Geräte (Flash überschreiben etc.), 2 Millionen zerstörte Geräte: “Killing the oxygen supply of Mirai”

Das eigene Projekt

Beispielprojekt

- einfache Wetterstation mit Sensoren
- und Webcam auf den Garten schauend
- aus dem Internet erreichbar
- “Aus dem Urlaub zu Hause nachschauen.”

Bedrohungsszenarien, Beispiel

- Übernahme & Rekrutierung durch Botnetze
- Übernahme und Deaktivierung (z.B. Firmware löschen, BrickerBot)
- Übernahme und Verwendung als Startpunkt zur Netzwerk-Infiltration
- Klassiker: Ransomware
- neu: Cryptojacking (Übernahme zum Cryptowährung minen)
- Datendiebstahl, Missbrauch als Wanze

Bedrohungsszenarien, Gradient

- Angriffe durch Botnetze
- Angriffe durch Scriptkiddies
- gezielte Angriffe (persönliche Fehde, Stalking, . . .)
- Hardwareangriff mit “Knipex & Schraubendreher”

Worum geht es nicht?

- SELinux
- Tomoyo, RSBAC, ...
- komplexe Sicherheitskonfiguration
- Virenscanner auf Linux
- IDS (Intrusion Detection System)
- komplexe Sicherheitskonfigurationen
- Hardwareabsicherung (tamper resistance)
- (beliebig komplexes Allheilmittel)
- Software für Wetterstation & Webcam

Beispielplattform

- Platform: Raspberry Pi
  - weit verbreitet, billig
  - breite Hardware- und Softwareunterstützung
- Gesagtes gilt analog für andere verbreitete embedded Boards
- handgelötete Exoten: im Prinzip auch, aber selbst schuld

Grundsätzliche Überlegungen

- Minimierung der Angriffsfläche
  - Was nicht am Netz lauscht, kann nicht via Netz angegriffen werden.
  - Dienste die nicht laufen, können nicht angegriffen werden.
  - Software, die nicht installiert ist, kann nicht ausgenutzt werden.
- $\rightarrow$ Minimale Basisplatform
- Vorhandene bekannte Einfallstore schliessen
- zeitnahe Aktualisierung der Software

Antimuster, oder wie man es nicht machen sollte

- Admin-Interface via Telnet. Es ist 2018, warum?
- Generell: plain text Admin-Interface (ja, auch FTP).
- Fest eingebrannte Zugangsdaten.
- Uralte Software verbauen, nicht aktualisieren, “tut ja”.
- Web-Interface: Passwort-Validierung via JavaScript im Client
- Datentransfer zu extern via plain text. Wer liest mit?
- Admin-Zugang via IP(-Bereich) authorisieren.
- Update-Server blind vertrauen (keine Signaturprüfung).

Wie man es besser macht

- Admin-Interface via https oder ssh (mit Schlüsseln).
- Initiale Zugangsdaten normal konfiguriert, werden im Zuge des Setups überschrieben, per Reset wiederherstellbar.
- Regelmässige, zeitnahe Sicherheitsaktualisierungen.
- Updates mit Signaturen verifizieren.
- Web-Interface: Authentisierung serverseitig.
- Datentransfer zu extern via https oder ssh/sftp.

Auswahl der Basis-Distribution

- zahlreiche verschiedene Distros verfügbar:
  - Raspbian & Raspbian Lite
  - Ubuntu Mate & Snappy Core
  - Mediacenter-Distros
  - zahlreiche Spezialdistros/-remixes
- Grundfrage: Was braucht man *wirklich*?
- Spezialdistros: Updates?
- Empfehlung Raspian:
  - Raspbian: minimale Netzwerkdienste (avahi, dhcpd, ggf. sshd)
  - Raspbian Lite: s.o., aber ohne Komplexität von GUI Stack
  - basierend auf Debian, laufend aktualisiert
  - voll unterstützte Distribution für Raspberry Pi

Raspberry, Basisabsicherung

- ssh standardmässig deaktiviert
- Einträge in `/root/.ssh/authorized_keys` vornehmen.
- Raspberry sshd default Konfiguration ist ziemlich brauchbar.
- Non-root User anlegen für Wartung
- `userdel pi`
- Für Wartungszugang ssh aktivieren.
- `dpkg --purge avahi-daemon libnss-mdns`
- Es sei denn, das Projekt *braucht* avahi/ZeroConf.
- weniger Rauschen im Log: sshd auf Port $\neq$ 22 (!23)

Sicherheitsupdates

- Debian & Derivate: SecureApt seit 2005 auf gpg Basis.
- Erfüllt: Updates mit Signaturen verifizieren.
- Regelmässige manuelle updates? Klaaaaaaaaar.
- `apt-get install unattended-upgrades`
- `/etc/apt/apt.conf.d/50unattended-upgrades`:
  - Falls Nachricht erwünscht
  - `Unattended-Upgrade::Mail` = Email-Adresse
  - `Unattended-Upgrade::MailOnlyOnError` = `false`

Verwendete Software

- Vermeiden:
  - Cowboy-Installation: `curl http://foo.baz/dl| bash`
  - fertige Docker-Images
  - Pakete aus obskuren Repositories
  - live aus dem Entwickler-Repo (`git clone ...`
- Empfehlung:
  - Software nur aus der verwendeten Distro
  - $\rightarrow$ erhöht Wahrscheinlichkeit von Paketpflege

Sichere Konfiguration

- Nur Dienste auf externen Interfaces die erreichbar sein sollen.
- "interne" Dienste zum Eigenbedarf (z.B. Datenbank):
  - Nicht an externe Interfaces binden.
  - explizit an localhost `127.0.0.1` binden
  - ggf. mit lokale Packetfilterregeln erzwingen
- Netzwerkverkehr nach Möglichkeit verschlüsseln (https)
- keine nicht zwingend notwendigen Dienste betreiben

Existierende Geräte

Die Start-Situation

- gekaufte IoT-Geräte
- keine Kontrolle über Design & Implementierung
- Standardannahme: Unsicher, muss geschützt werden
- Umfassende Systemdokumentation: Schön wärs . . .
- soll aber eingesetzt werden aus “Gründen” (WAF usw.)
- Was tun?

Bedürfnisse & Anforderungen klären

- Wie kommuniziert das Gerät (Ethernet, WLAN, . . .)?
- Muss es aus dem Internet erreichbar sein?
- Kommunikation nach aussen: push, pull oder beides?
- Welche Protokolle müssen von aussen erreichbar sein?
- Welche Protokolle müssen nach aussen erreichbar sein?

Konkrete Analyse

- Idealfall: alles dokumentiert. (Ihr könnt aufhören zu lachen)
- Realität: eigene Analyse nötig:
  - Portscan: `nmap`
  - Trafficanalyse: `tcpdump` & WireShark
- Nutzerberichte in Internet lesen
- Welche Ports sind offen, welche Traffic geht rein/raus?

Grundannahmen

- Gerät benötigt WLAN.
- Gerät muss aus dem Internet erreichbar sein, z.B.:
  - Fernsteuerung aus der Cloud
  - push-Updates
- Gerät muss aktiv mit dem Internet reden können, z.B.:
  - Software-Updates
  - Datendienste (Feeds)

Gesicherter Betrieb

- Gerät nicht direkt am Internet betreiben.
- Idealfall: IoT-Geräte in eigener Netzumgebung
- Firewall zwischen Gerät und Internet sowie Heimnetz.
- Firewall erlaubt nur notwendigen Traffic, z.B.:
  - eingehend http(s) & ssh
  - ausgehend http(s)
- anderer Traffic wird blockiert
- Firewall liefert angepasste Dienste für Gerät
  - DHCP-Server
  - Nameserver (via DHCP konfiguriert)
  - $\rightarrow$ Schutz des Heimnetzes, verhindert DNS amp attacks
  - http-(Trans-)Proxy und https-Proxy mit Begrenzungen

Fragen?

Vielen Dank für Euer Interesse!