Netzwerkverkehr schnüffeln

Bernhard Trummer
bernhard.trummer@gmx.at
(B44E B171 890F 85E2 B616 47DC CF17 178E B5D8 5E89)

10. April 2021

Über mich

- Absolvent: Telematik (2001)
- Angestellt bei Beyond (früher: Infonova, BearingPoint) (seit 2000)
- Architekt/Entwickler (Java / Springframework)
- Regelmäßiger Vortragender seit es die GLT gibt
- Spiele gerne mit Software, Netzwerk, Hardware
- Spiele auch Gitarre in einer Hobby-Metal-Band

Inhalt

- Einleitung (was?, warum?, wie?)
- tcpdump
- wireshark
- mitmproxy
- (alles jeweils mit einer kleinen Live-Demo)
- Q & A

Einleitung

Szenario

- Du hast ein Gerät
- . . . oder eine Smartphone App
- Das Ding kommuniziert über das Netzwerk
- Und Du willst mithören (bzw. sniffen)

Warum sniffen?

- Reine Neugierde

Warum sniffen?

- Kontrolle
  - Netzwerkkonfiguration
  - Mit wem kommuniziert ein Gerät eigentlich?
  - Welche Daten sendet eine App eigentlich nach außen?
  - ...

Warum sniffen?

- Reverse Engineering
  - Du willst einen „Ersatzclient“ schreiben
    - Beispiel: LG TV Remote (`https://media.ccc.de/v/glt19-50-smart-tv-per-http-fernsteuern`)
  - Du willst einen „Ersatzserver“ schreiben
    - Beispiel: Game-Server, weil der „offizielle“ abgedreht wurde (`https://media.ccc.de/v/31c3_-_5956_-_en_-_saal_2_-_201412281400_-_cyber_necromancy_-_joseph_tartaro_-_matthew_halchyshak`)

Was braucht man zum sniffen?

- tcpdump und root-Zugriff
- Für genauere Analysen: wireshark
- Bei embedded Dingen (z.B. Smart TV) wird es etwas schwieriger
- Dann brauchst Du einen dezidierten Router mit 2 Netzwerkinterfaces, z.B.:
  - alter PC mit einer zusätzlichen Ethernet-Karte (oder WLAN-Stick)
  - Raspberry Pi (Ethernet + WLAN)
  - ...
- Und etwas Netzwerkkonfiguration (WLAN Access Point, NAT, etc.)
  - Siehe: `https://media.ccc.de/v/GLT18_-_297_-_de_-_g_ap147_006_-_201804281445_-_pimp_up_your_heimnetzwerk_-_bernhard_slash_trummer`

Tools: tcpdump

tcpdump: Überblick

- tcpdump ist ein Commandline-Tool
- es muß als „root“ ausgeführt werden
- die wichtigsten Optionen:
  - -i interface
  - -w file.cap
  - -v
- siehe auch: „man 8 tcpdump“

tcpdump: Expressions

- Zum Filtern von Paketen, die mit -v ausgegeben oder mit -w geschrieben werden.
- Beispiele:
  - proto udp
  - dst port 80
  - expr1 and expr2
  - ...
  - siehe auch: „man 7 pcap-filter“
- Beispiel: tcpdump -i wlan0 -v -w /tmp/wlan0.cap host 192.168.10.5

tcpdump: (p)cap-Files

- Standard-Dateiformat zum Speichern von Netzwerkpaketen
- Es gibt Libraries für alle verbreiteten Programmiersprachen
- Dadurch gibt es auch unzählige Tools, die „irgendwas“ mit pcap-Files machen
- PS: Es gibt auch einen pcap-mode für den Emacs ;-)

tcpdump: Demo

Tools: wireshark

wireshark: Überblick

- Graphisches Tool zur Anzeige von pcap-Files
- Kann praktisch alle Netzwerk-Protokolle parsen und aufbereitet darstellen
- Ähnliche Filter-Expressions wie bei tcpdump:
  - dns
  - ip.addr == 192.168.1.2
  - http.file_data contains “foobar“
  - ...

wireshark: Demo

Tools: mitmproxy

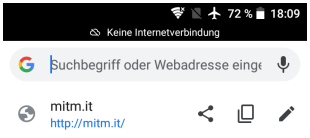
mitmproxy: Überblick

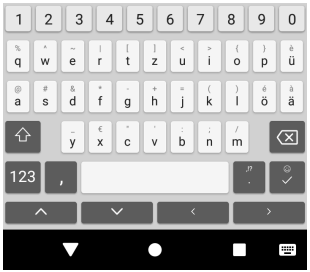
- mitmproxy = Man In The Middle Proxy (für https)
- legt beim ersten Start ein CA-Zertifikat an (self-signed)
- erkennt bei jedem Request zu welchem Server sich der Client verbinden will
- stellt für diesen Server on-the-fly ein eigenes Zertifikat aus
- und macht damit den TLS-Handshake zum Client
- verbindet sich gleichzeitig selbst zum Server (eigener TLS-Handshake)
- und kann dadurch alle Datenpakete mitlesen
- Siehe: `https://docs.mitmproxy.org/stable/`

mitmproxy: CA-Zertifikat

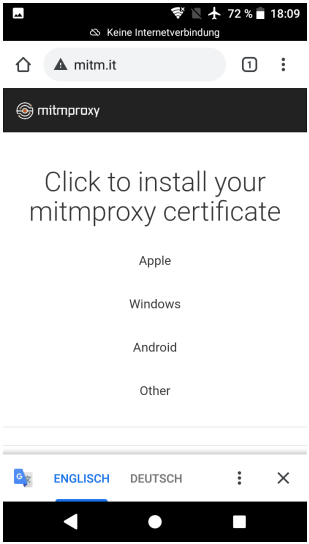
- Moment... Sollte TLS eigentlich nicht MITM verhindern?
- Prinzipiell ja.
- Deswegen muß auch das CA-Zertifikat von mitmproxy im Client importiert werden
- Siehe: `https://docs.mitmproxy.org/stable/concepts-certificates/`

mitmproxy: Zertifikat in Android installieren

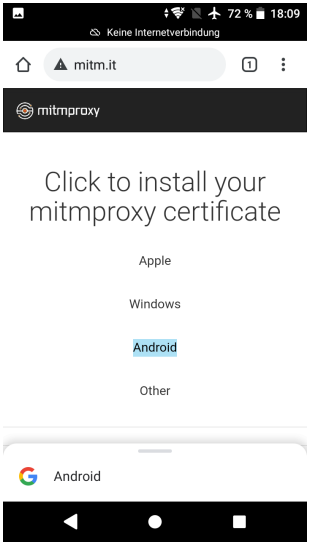




mitmproxy: Zertifikat in Android installieren



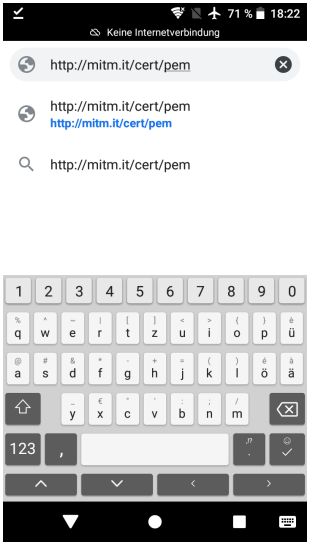
mitmproxy: Zertifikat in Android installieren



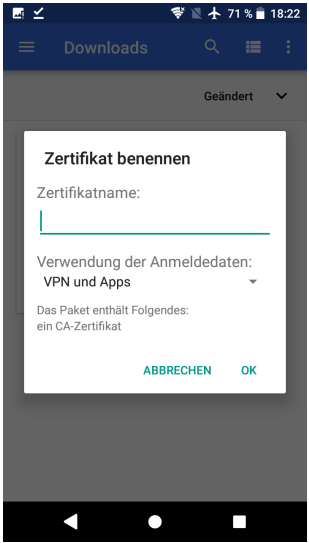
mitmproxy: Zertifikat in Android installieren

- Leider passiert dann nichts weiter...
- (der Download-Button, so wie in der Doku beschrieben, fehlt)
- Workaround: das pem-File via direkter URL herunterladen

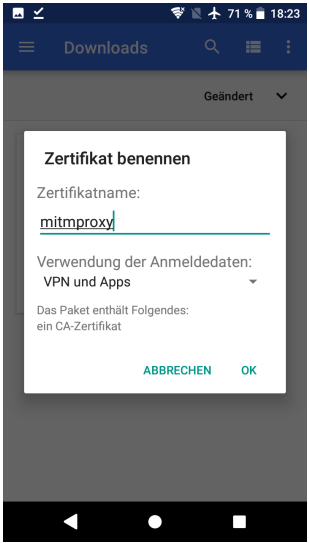
mitmproxy: Zertifikat in Android installieren



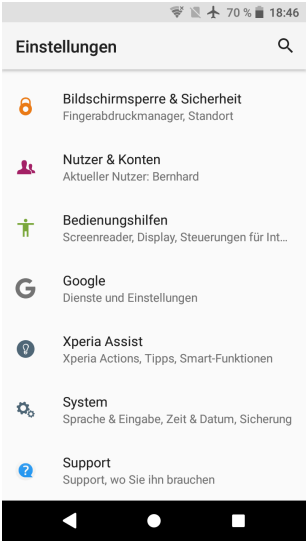
mitmproxy: Zertifikat in Android installieren



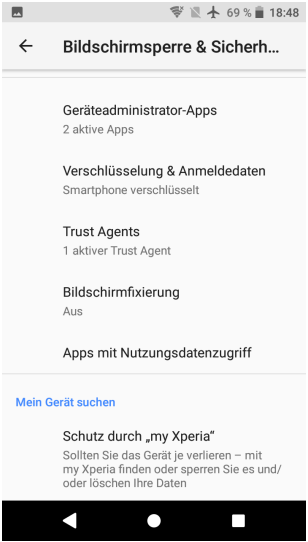
mitmproxy: Zertifikat in Android installieren



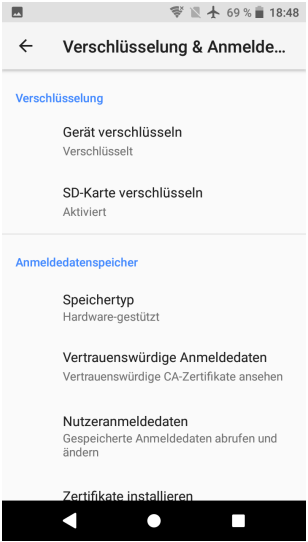
mitmproxy: Zertifikat in Android installieren (Kontrolle)



mitmproxy: Zertifikat in Android installieren (Kontrolle)



mitmproxy: Zertifikat in Android installieren (Kontrolle)



mitmproxy: Zertifikat in Android installieren (Kontrolle)

mitmproxy: Certificate-Pinning

- Funktioniert nicht, wenn der Client „Certificate-Pinning“ macht
- TL;DR:
  - Dann muß der Client modifiziert werden
  - Je nach Client kann der Aufwand dafür sehr groß werden...
  - (abhängig von den implementierten Schutzvorrichtungen gegen Modifikationen)

mitmproxy: Betriebsarten

- Regular, Transparent, Reverse Proxy, Upstream Proxy, SOCKS Proxy
- Welchen nehmen?
- Siehe `https://docs.mitmproxy.org/stable/concepts-modes/`

mitmproxy: Transparent Proxy

- root@net6501:~#
  - iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-port 8443
  - iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 443 -j REDIRECT --to-port 8443
- mitmproxy --listen-host=192.168.10.1 -p 8443 --mode transparent
- (192.168.10.1 ist bei mir die IP vom wlan0)
- Nach Beenden von mitmproxy nicht vergessen, die Redirects wieder zu entfernen:
  - iptables -t nat -D PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-port 8443
  - iptables -t nat -D PREROUTING -i wlan0 -p tcp --dport 443 -j REDIRECT --to-port 8443

mitmproxy: mitmpcap.py

- git clone https://github.com/muzuiget/mitmpcap.git
- mitmproxy . . . -s /path/to/mitmpcap/mitmpcap.py
- -s . . . Einbindung von einem externen Script
- mitmpcap.py schreibt die Klartext-Kommunikation in ein „output.pcap“ File
- Dieses File kann ganz normal in wireshark geöffnet werden

mitmproxy: SSLKEYLOGFILE

- SSLKEYLOGFILE=/tmp/sslkeylogfile.txt mitmproxy ...
- Umgebungsvariable, damit mitmproxy alle Keys in das angegebene File schreibt (funktioniert de facto in allen SSL-fähigen Applikationen)
- wireshark kann mit diesem Logfile „gefüttert“ werden um die TLS-Pakete zu entschlüsseln:
  - Rechte Maustaste auf ein TLS-Paket
  - Protocol Preferences
  - SSL debug file...
  - (Pre)-Master-Secret log filename

mitmproxy: alles kombiniert

- SSLKEYLOGFILE=/tmp/sslkeylogfile.txt mitmproxy --listen-host=192.168.10.1 -p 8443 --mode transparent -s /path/to/mitmpcap/mitmpcap.py
- (iptables REDIRECT nicht vergessen)
- Das Up- und Downlink Interface extra mitsniffen:
  - tcpdump -i wlan0 -v -w /tmp/downlink_wlan0.cap
  - tcpdump -i eth0 -v -w /tmp/uplink_eth0.cap
- Ergebnis: 3 pcap-Files und ein sslkeylogfile.txt
- Zur Erinnerung:
  - output.pcap:
    - kommt von mitmpcap.py
    - enthält nur die HTTP-Kommunikation, die durch mitmproxy geht
  - sslkeylogfile.txt:
    - enthält die Keys für uplink_eth0.cap
    - die Keys funktionieren jedoch nicht für downlink_wlan0.cap!

mitmproxy: Demo

Ende

Ende

- Feedback:
  `https://pretalx.linuxtage.at/glt21/talk/3VARVK/`
- Vollständiges PDF mit den Folien:
  `https://pretalx.linuxtage.at/media/glt21_3VARVK_netzwerkverkehr_schnueffeln.pdf`
- „Capture the Flag“ Übungsbeispiel:
  `https://pretalx.linuxtage.at/media/capture_the_flag.zip`

Q & A