1 Analyse mit Wireshark

Nun werden wir heutigen Datenverkehr aufnehmen und analysieren. Via Youtube catchen wir einige Datenpackete welche wir anschließend begutachten, gemäß folgender Fragen:

1. Versuchen Sie die Verbindung über unterschiedliche Zugangsnetzwerke (z.B. LAN, WLAN, 3/4G sofern möglich) herzustellen und dokumentieren Sie allfällige Unterschiede.
2. Welche Objekte werden vom Client via HTTP angefordert? Hinweis: nur jene beim Videostreaming, andere Objekte (z.B. HTML, Text, Bilder) können vernachlässigt werden. Wir setzen Wireshark auf unser Netzwerk an und starten den Stream. Wir beenden die Aufnahme alsbald und analysieren unsere .pcapng's mithilfe von

```
- Analyze > Follow > TCP Stream / UDP Stream
- Statistics > Conversations
- Filter wie 'tcp.ack == 1 && tcp.seq == 1'
- u.s.w.
```

1.1 Stream eines Youtube-Videos via Ethernetverbindung

Wenn wir Info betrachten, so können wir direkt den TCP-Handshake des Klienten und des Servers verifizieren, ebenso dass der Server uns nicht via RST abgewiesen hat. Ebenso können wir mittels Filter die genutzten Protokolle identifizieren: ARP, SNMP, TCP und TSLv1.2, ICMP, DNS

TSLv1.2 beschreibt eine SSL-Enkryption vom TCP-Austausch über Secure Sockets (Spezielle Ports) SNMP als das Zuweisungsprotokoll innerhalb unseres Heimnetzwerkes

Weiters können wir feststellen, dass es beim Verbindungsaufbau zur URL bis hin zur Beendigung des Videos insgesamt

- 6 unterschiedliche IPs miteinander kommunizierten und im Rahmen dieser
  - 19 unterschiedliche IPv4 Konversationen erfolgten
  - 16 Konversationen über TCP verliefen
  - 4 UDP Transfers stattfanden

Ein Objektzugriff über HTTP erfolgte hier erstaunlicherweise nicht, was jedoch mit der Implementierung seitens Youtube zusammenhängt.

Wir starten einen neuen Sniff-Versuch, nun jedoch über das WLan unseres Heimnetzwerkes.

1.2 Stream eines Youtube-Videos via WLan-Verbindung

Der Stream über WLan erfolgte über weniger Packeten als wie beim Stream durch direkte Ethernetverbindung. Die Protokolltypen blieben die gleichen, während es einen Umschwung bei den Konversationen gab. Diesmal kommunizierten lediglich

- 2 unterschiedliche IPs miteinander, samt
  - 12 unterschiedlichen IPv4 Konversationen
  - 26 TCP Konversationen

- und 1 UDP Transfer

Bei beiden Versuchen handelte es sich um denselben Anfangszustand (Videoauswahl Youtube), dasselbe Video und dieselbe Videoqualität.

Wir versuchen uns diesmal an einem anderen Anbieter:

1.3 Stream von anilinkz.io[1] über Ethernet Wir umgehen hierbei triviale Daten und starten direkt mit dem Stream selbst, welcher jedoch wiederum Werbung hervorruft (dementsprechend die abgebrochenen TCP-Packete am Start).

Über die Zeitdauer von 5 Minuten ergaben sich ein gutes Dutzend an Konversationen. Verteilt über

- 12 unterschiedliche IPs ergaben sich
  - 138 IPv4 und 5 IPv6 Konversationen
  - 187 TCP Datentransfers und
  - 9 UDP Übertragungen

Der Hauptverbindungsträger ist www11.mp4upload.com, eine klassifizierte Subdomain eines Hosts, ein Hinweis auf implementierte Server Load Balancing.

Es erfolgten hierbei Konversationen über Protokolle wie TLS1.2, TCP, UDP, DNS, MDNS, ARP, IGMPv2 ...

- MDNS oder Multicast DNS dient der IP-Adressierungen in kleineren Netzwerken.

[1] http://anilinkz.to/no-game-no-life-episode-1?src=8