

# Security Lab – Network Attacks

Dieses Lab besteht aus zwei Teilen, welcher jeder 2 Punkte gibt.

Teil 1: Kapitel 1-5

Teil 2: Kapitel 6-7

## Setup

Das Lab wird in unserem «cloud-lab» in Gruppen gelöst:

<https://sec.cloudlab.zhaw.ch>

Die Infrastruktur ist so aufgesetzt, dass Sie nach dem login direkt zu den VMs, welche Ihrer Gruppe zugeordnet sind, gelangen.

Pro Gruppe sind dies die folgenden VMs (wobei ID der Gruppennummer entspricht):

- Attacker      (**its-group-ID-attacker**)
- Client        (**its-group-ID-client**)
- Server        (**its-group-ID-server**)

Achtung: alle Mitglieder der Gruppe arbeiten auf denselben VMs.

Die VMs befinden sich alle im gleichen Subnet (10.51.ID.0/24). Die IP-Adressvergabe ist wie folgt:

**10.51.ID.1 Gateway**

**10.51.ID.2 DHCP**

**10.51.ID.10 Attacker**

**10.51.ID.11 Client**

**10.51.ID.12 Server**

## 1 Einleitung

Dieses Lab soll Ihnen aufzeigen, wie die gesamte Netzwerkkommunikation in einem LAN mit frei erhältlichen Tools relativ einfach mitgelesen werden kann, um sensitive Informationen wie zum Beispiel Passwörter zu extrahieren. Darauf hinaus werden Sie auch einige Man-in-the-Middle-Attacken kennen lernen, die es möglich machen, aktiv in die Kommunikation einzugreifen. Damit können sogar scheinbar sichere Protokolle wie HTTPS angegriffen werden.

**Wichtig:** Da es prinzipiell möglich ist, mit den im Lab eingesetzten Tools Teile des Netzwerks in die Knie zu zwingen, möchten wir Sie bitten, vorsichtig bei Ihren Netzwerkattacken vorzugehen und sich nur auf die virtuellen Systeme zu beschränken, die gemäss Praktikumsanleitung im Zusammenhang mit diesem Lab verwendet werden sollen.

## 2 Wireshark<sup>1</sup>

Wireshark ist ein Netzwerk-Protocol-Analyzer mit einem grafischen Interface, der eine Vielzahl von Protokollen bis ins letzte Detail zu interpretieren weiss. Ebenfalls können gelesene Daten als *libpcap/winpcap*-Rohdaten gespeichert und zur Offline-Analyse auch eingelesen werden. Wireshark ist für diverse Systeme (Linux, Windows, macOS) frei verfügbar und ist auf dem Kali Linux-Image vorinstalliert. Starten Sie Wireshark auf dem Attacker über den Menü-Button oben links und Auswahl von *Sniffing & Spoofing* → *wireshark*.

Über den Menüpunkt *Capture* → *Options* wird das Sniffen gestartet. Dabei wird das *Capture Interfaces*-Fenster geöffnet, wo Sie Optionen zum Sniffen angeben können. Markieren Sie die Zeile mit

---

<sup>1</sup> <https://www.wireshark.org>

dem Netzwerkinterface, von dem Sie sniffen wollen. Meist wird dies das externe Netzwerkinterface sein

**Kali:**           **eth0**

**Ubuntu:**       **ens3**

da eine IP-Adresse des LANs aufweist, in dem sich der Host (in diesem Fall der Attacker) befindet. Da Sie in diesem Praktikum jeweils alle Pakete aufzeichnen werden und nicht nur die, die eine Sender- oder Empfänger-MAC-Adresse des eigenen Hosts aufweisen, müssen Sie den *Promiscuous mode* verwenden. Dies konfigurieren Sie entweder pro Interface in der Spalte *Promiscuous* oder für alle Interfaces mit der Checkbox *Enable promiscuous mode on all interfaces*. Mit dem *Start*-Button wird das Sniffen gestartet. Es dauert so lange, bis Sie den *Stop*-Button in der Toolbar klicken.

Die gesnifften Pakete werden im *Wireshark*-Hauptfenster aufgelistet. Wenn Sie im oberen Teil ein Paket auswählen, werden unten die Detailinformationen zum Paket angezeigt, aufgeschlüsselt nach den einzelnen Protokollsichten.

Im *Capture Interface*-Fenster können auch Capture-Filterregeln definiert werden. Dadurch werden nur die Pakete aufgezeichnet, die diesen Regeln entsprechen. Für die Definition von Capture-Filterregeln wird die *tcpdump*-Syntax verwendet. Neben Capture-Filterregeln gibt es auch Display-Filterregeln. Diese können Sie im Hauptfenster von *Wireshark* spezifizieren, um die angezeigten Pakete einzuschränken. Die Syntax für die Display-Filterregeln ist anders als die für die Capture-Filterregeln. Wenn Sie in «**Apply a display filter...**» zu tippen beginnen, werden Ihnen Vorschläge mit der korrekten Syntax angezeigt. Einfache Filterregeln sind z.B. *arp* oder *http*, um nach Paketen eines Protokolls zu filtern oder *ip.addr == 192.0.2.1*, um Pakete mit einer bestimmten IP-Adresse anzulegen.

Machen Sie sich kurz mit der Grundfunktionalität von *Wireshark* vertraut, damit Sie das Tool in den folgenden Szenarien sinnvoll einsetzen können. Analysieren Sie dazu den Netzwerkverkehr beim externen Interface. Generieren Sie dazu auch eigenen Traffic, zum Beispiel indem Sie mit dem Webbrowser eine Webseite herunterladen (Kali beinhaltet Firefox, starten können Sie diesen über die Menübar links). Versuchen Sie herauszufinden, was die aufgezeichneten Pakete bedeuten. Wenden Sie ebenfalls Display-Filterregeln an, um nur gewisse Protokolle oder nur die Pakete gewisser Hosts aufzunehmen.

### 3 Ettercap<sup>2</sup>

*Ettercap* ist ein mächtiges Cracker-Tool und unterstützt Sniffing, Content-Filtering, Man-in-the-Middle-Attacken auf diverse Protokolle und mehr. *Ettercap* bietet zudem einen Plugin-Mechanismus, mit welchem *Ettercap* von jedem durch weitere Funktionalität erweitert werden kann.

Öffnen Sie auf dem Attacker ein neues Konsolenfenster und starten Sie *Ettercap* wie folgt:

```
$ sudo ettercap -Tzqu
```

Dies startet *Ettercap* in einem rein «passiven» Modus. *Ettercap* wird mithören, was auf dem Netzwerkinterface des Attackers alles für Verkehr fliesst und dazu interessante Informationen anzeigen, z.B. im Verkehr entdeckte Benutzernamen und Passwörter. Lassen Sie diese als *Listener* bezeichnete Instanz von *Ettercap* während des ganzen restlichen Labs laufen. Sollte diese Crashen oder erwartete Informationen dort nicht erscheinen, beenden Sie die Instanz (in der Konsole Ctrl+C drücken) und starten Sie diese neu.

Loggen Sie sich nun vom Attacker her via FTP (in einem Terminal) auf dem Server mit

---

<sup>2</sup> <https://www.ettercap-project.org>

```
ftp 10.51.x.12
```

**username: user**

**password: securityzhaw**

ein. Was beobachten Sie beim *Listener*?

```
FTP : 10.51.5.12:21 → USER: user PASS: securityzhaw
```

Der Listener sieht alle Angaben, die der User eingegeben hat.

Öffnen Sie nun den Browser auf dem Attacker und kontaktieren Sie den Server (via HTTP). Der Webserver verlangt eine HTTP-Authentisierung, melden Sie sich auch hier mit *user/securityzhaw* an. Was beobachten Sie im *Listener*?

```
HTTP : 10.51.5.12:80 → USER: user PASS: securityzhaw INFO: 10.51.5.12/
```

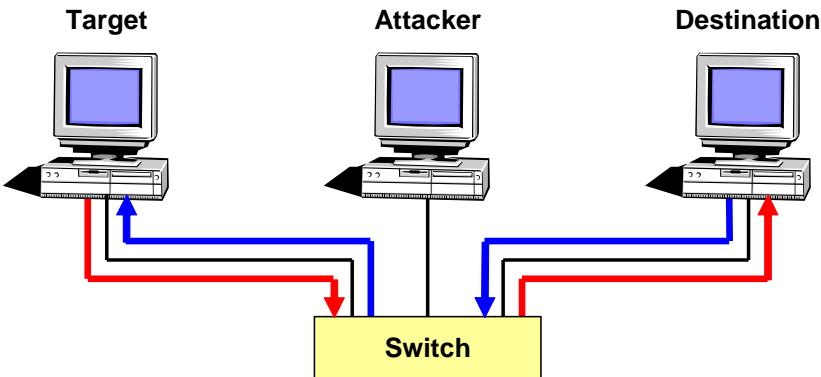
Sie sehen also, dass *Ettercap* die Passwörter im beobachteten Netzwerkverkehr «snifft» und ausgibt. Bei HTTP funktioniert das Sniffen deshalb, weil hier nur die *Basic Authentication* des HTTP-Protokolls verwendet wird, bei der Username und Passwort im Klartext übermittelt werden.

## 4 ARP-Spoofing

In Abschnitt 3 haben Sie die Passwörter ausspioniert, die auf Ihrem eigenen System eingegeben wurden. Interessanter ist es natürlich, den Verkehr und damit Passwörter von anderen Benutzern im gleichen LAN zu sniffen. Im Folgenden werden wir uns damit beschäftigen.

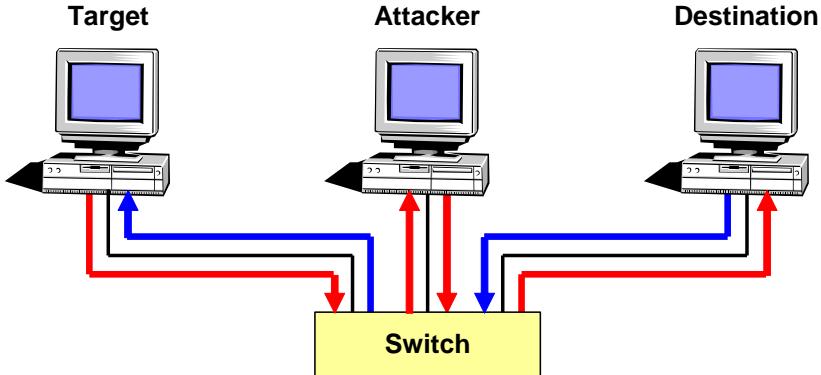
Als noch primär Hubs verwendet wurden, um Ethernet-basierte LANs aufzubauen, hat jeder Host im LAN den Verkehr aller anderen Hosts gesehen, weil ein Hub ankommende Ethernet-Frames immer über alle Ports hinaus sendet. Heute werden Switches statt Hubs verwendet, welche die Frames nur über denjenigen Port hinaus senden, an welchem der Empfänger des Frames erreichbar ist. Die entsprechenden LANs werden deshalb auch als geswitchte LANs bezeichnet.

In einem geswitchten LAN kann ein Attacker die Ethernet-Frames zwischen einem Target und seiner Destination (ein anderer Host im LAN oder der Default-Gateway des LANs, wenn der Zielhost in einem anderen Netz liegt) folglich nicht „einfach so“ sniffen. Die Pakete werden ja nicht an den Host des Attackers gesendet, wie die untenstehende Abbildung zeigt.



ARP-Spoofing kann hier abhelfen, indem der korrekte MAC-Adresseintrag im ARP-Cache des Targets durch die MAC-Adresse des Attackers ausgetauscht wird. Weil man dabei den ARP-Cache des Targets „vergiftet“, wird ARP-Spoofing auch ARP-Poisoning genannt. Der Target-Host „denkt“ dann, zu der IP-Adresse der Destination gehört die MAC-Adresse des Attackers. Als Folge werden alle Ethernet-Frames, die für die Destination bestimmt sind, auf Layer 2 zum Attacker gesendet, der sie dann wiederum an die Destination weiterleitet. Das Target merkt nichts davon, weil die Kommunikation mit der Destination problemlos funktioniert. Die Situation nach erfolgreichem ARP-Spoofing zeigt die nachfolgende Abbildung.

**Man in the middle**



Wenn man auch den Rückkanal von der Destination zum Target spoofen will, muss man zusätzlich auch den ARP-Cache der Destination vergiften.

Technisch ist ARP-Spoofing einfach. Es funktioniert, weil Hosts nicht nur diejenigen ARP-Replies beachten, die auf eigene ARP-Requests folgen, sondern sämtliche ARP-Replies, die an den Host gesendet werden. Aufgrund dieser ARP-Replies wird dann der ARP-Cache aktualisiert. Deshalb muss der Attacker einfach eine ARP-Reply an das Target senden, in welchem die MAC-Adresse des Attackers mit der IP-Adresse der Destination assoziiert wird.

Im Folgenden werden Sie selbst ARP-Spoofing mit *Ettercap* durchführen. Für die Rolle des Attackers verwenden Sie das eingangs als «Attacker» bezeichnete Kali Linux-System, für die Rolle des Targets verwenden Sie das eingangs als «Client» bezeichnete eine Ubuntu-System, und als Destination verwenden Sie das als «Server» bezeichnete andere Ubuntu-System.

#### 4.1 Ermitteln der spoofbaren Hosts

Starten Sie nun die GUI-Version *Ettercap* über den Menü-Button oben links und Auswahl von *Sniffing & Spoofing → ettercap-graphical*. *Ettercap* verlangt root-Rechte, weshalb Sie nach dem Start das Passwort *securityzhaw* eingeben müssen. Klicken Sie nach dem Start auf den *Accept* Button oben rechts. Während diese Instanz ebenfalls per Default Passwörter im Datenstrom erkennt und anzeigt, funktioniert dies z.T. nach dem Aktivieren von Plugins nicht mehr, weshalb hierfür weiterhin die zuvor gestartete *Listener* Instanz genutzt wird.

Damit ein ARP-Spoofing durchgeführt werden kann, müssen zuerst die Hosts im lokalen Netz ermittelt werden. Klicken Sie dazu auf dem Attacker in *Etercap* den *Scan for Hosts* Button oben links, worauf die Hosts im lokalen Netz ermittelt werden. *Etercap* gibt dabei im unteren Fenster aus, wie viele Hosts gefunden wurden. Mit dem *Hosts List* Button oben rechts werden die gefundenen Hosts mit IP- und MAC-Adressen angezeigt.

## 4.2 Aktivieren des ARP-Spoofing

You can also use ip addr show

Bestimmen Sie zuerst die IP- und MAC-Adressen von Client und Server (zur Erinnerung: Der Server wird als Destination verwendet). Dies können Sie jeweils in einem Terminal mit *ifconfig* machen. Mit der Kenntnis der Adressen der involvierten Systeme können Sie Client und Server nun in der *Hosts list* identifizieren. Um Client und Server für das ARP-Spoofing auszuwählen, dienen die Buttons *Add to Target 1* und *Add to Target 2*. Dabei spielt es keine Rolle, welcher der beiden Hosts als Target 1 bzw. 2 ausgewählt wird, denn das ARP-Spoofing wird in beiden Richtungen durchgeführt, d.h. sämtliche Daten zwischen Target und Server werden über den Attacker geleitet. Markieren Sie also den Client oder den Server und klicken Sie auf den Button *Add to Target 1*. Markieren Sie dann den anderen Host und klicken Sie auf *Add to Target 2*.

Über den Menü-Button mit den drei Punkten und Auswahl von *Targets* → *Current targets* können Sie überprüfen, ob Sie die beiden richtigen Hosts zugewiesen haben und mit einem Klick auf den *MITM menu* Button oben rechts und Auswahl von *ARP poisoning* wird das Spoofing gestartet. Markieren Sie im sich öffnenden Fenster die Checkbox bei *Sniff remote connections* Und klicken Sie auf *OK*. Das untere Fenster von *Etercap* wird Sie darüber informieren, welche Hosts gespoofed werden. Überprüfen Sie auf dem Client in einem Terminal mit dem Befehl

```
$ arp -an
```

ob das ARP-Spoofing erfolgreich war. Wie sieht der relevante ARP-Eintrag bei Ihnen aus und wie erkennen Sie, dass das Spoofing erfolgreich war?

```
user@its-group-5-client:~/Desktop$ arp -an
? (10.51.5.12) at fa:16:3e:dc:e6:f6 [ether] on ens3
? (10.51.5.1) at fa:16:3e:44:65:27 [ether] on ens3
? (10.51.5.10) at fa:16:3e:dc:e6:f6 [ether] on ens3
```

**Server hat die Mac Adresse des Angreifers**

Verwenden Sie nun wie oben FTP und HTTP und loggen Sie sich vom Client aus auf dem Server ein. Was beobachten Sie im *Listener*? Welche Credentials werden im Vergleich zu den lokalen Versuchen in Abschnitt 3 angezeigt?

**Der Listener hat die User und Password Werte. Die Credentials sind die von dem Client.**

```
FTP : 10.51.5.12:21 → USER: user PASS: securityzhaw
HTTP : 10.51.5.12:80 → USER: user PASS: securityzhaw INFO: 10.51.5.12/
HTTP : 10.51.5.12:80 → USER: user PASS: securityzhaw INFO: 10.51.5.12/
```

Setzen Sie ebenfalls *Wireshark* auf dem Attacker ein, um den Network-Traffic genau zu analysieren. Beachten Sie dabei insbesondere die ARP-Replies für das ARP-Spoofing, die der Attacker regelmäßig an Client und Server sendet und wie die normalen Datenpakete zwischen Client und Server vom Attacker empfangen und weitergeleitet werden. Schauen Sie dabei speziell auf die MAC- und IP-Adressen der Pakete, um im Detail zu verstehen, warum die Attacke tatsächlich funktioniert. In welchem Intervall werden die ARP-Spoofing Pakete gesendet und an wen?

10841 89.773647157 fa:16:3e:dc:e6:f6	fa:16:3e:6c:09:9c	ARP	42 10.51.5.12 is at fa:16:3e:dc:e6:f6
10842 89.773665631 fa:16:3e:dc:e6:f6	fa:16:3e:e8:05:8b	ARP	42 10.51.5.11 is at fa:16:3e:dc:e6:f6 (duplicate use of 10.51.5.12 detected!)
11761 99.783841167 fa:16:3e:dc:e6:f6	fa:16:3e:6c:09:9c	ARP	42 10.51.5.12 is at fa:16:3e:dc:e6:f6
11762 99.783869861 fa:16:3e:dc:e6:f6	fa:16:3e:e8:05:8b	ARP	42 10.51.5.11 is at fa:16:3e:dc:e6:f6 (duplicate use of 10.51.5.12 detected!)
12681 109.793997857 fa:16:3e:dc:e6:f6	fa:16:3e:6c:09:9c	ARP	42 10.51.5.12 is at fa:16:3e:dc:e6:f6
12682 109.794025369 fa:16:3e:dc:e6:f6	fa:16:3e:e8:05:8b	ARP	42 10.51.5.11 is at fa:16:3e:dc:e6:f6 (duplicate use of 10.51.5.12 detected!)

**Alle 10 Sekunden wir ein ARP-Spoofing Paket je an den Client und Server geschickt! Beide IP-Adressen(Server und Client) haben die gleiche Mac-Adresse!**

Grundsätzlich ist es schwierig, sich gegen ARP-Spoofing zu verteidigen; dies geht allenfalls mit Netzwerkmonitoring-Lösungen, die Anomalien in ARP-Responses erkennen können. Deshalb können Sie meistens kaum verhindern, dass Ihr Traffic über den Host von anderen Benutzern im gleichen LAN umgeleitet werden kann. «Ein anderer Benutzer im LAN» kann dabei natürlich auch ein externer Angreifer sein, der sich irgendwie Zugang verschafft hat, z.B. indem er den Rechner eines internen Benutzers kompromittiert hat (dieser Schritt ist nicht trivial, kommt in der Realität aber natürlich immer wieder vor). Deshalb ist es wichtig, dass sensitive Daten immer nur verschlüsselt übermittelt werden, denn dann ist es eigentlich unerheblich, ob die (verschlüsselten) Daten von anderen Benutzern mitgelesen werden können.

**Hinweise zur weiteren Verwendung von Ettercap:** In den weiteren Attacken werden Sie weiter mit *Ettercap* arbeiten. Beachten Sie dazu die folgenden Hinweise:

- Wenn Sie ein ARP-Spoofing ändern wollen (zwischen zwei anderen Hosts), dann wählen Sie zuerst über den Menü-Button *MITM menu* den Eintrag *Stop mitm attack(s)* und dann über den Menü-Button mit den drei Punkten *Targets → Wipe targets*. Darauf können Sie über den Menü-Button *Hosts List* wieder neue Hosts für Target 1 und 2 wählen.
- *Ettercap* ist nicht sehr stabil und kann auch mal crashen. Wenn Sie das Gefühl haben, es funktioniert etwas nicht richtig, dann ist ein Beenden und Neustart von *Ettercap* meist eine gute Idee. Teilweise ist sogar ein Neustart des virtuellen Hosts notwendig.

## 5 DNS-Spoofing

Bisher haben wir nur passiv gesnifft. Nun wollen wir aktiv den Netzwerkverkehr beeinflussen und durch eine Man-in-the-Middle-Attacke Verbindungen zu einem anderen Host umleiten. In diesem Szenario möchten wir einen Benutzer auf dem Target, der via FTP von [ftp.gnu.org](http://ftp.gnu.org) Software herunterladen möchte (auf diesem Server befinden sich eine Vielzahl von Open Source Softwarepaketen), auf einen eigenen FTP-Server umleiten. Ein solches Szenario stellt durchaus eine Bedrohung dar, denn es ermöglicht dem Angreifer, dem Benutzer manipulierte Software anzubieten – zum Beispiel eine modifizierte Version des GNU C-Compilers (gcc), der beim Kompilieren Malware in die Software einfügt.

Eine Möglichkeit, um Verbindungen umzuleiten ist über DNS-Spoofing. Dabei werden die DNS-Requests, die vom Client abgesetzt werden, direkt vom Attacker (als Man-in-the-Middle) und nicht vom richtigen DNS-Server beantwortet. Verlangt der Client also die IP-Adresse von [ftp.gnu.org](http://ftp.gnu.org), so liefert der Attacker nicht die echte IP-Adresse von [ftp.gnu.org](http://ftp.gnu.org), sondern die des eigenen FTP-Servers, auf den der Client umgeleitet werden soll. Für den eigenen FTP-Server verwenden wir dabei den Server. Falls der DNS-eintrag bereits gecached wurde, kann dieser mit dem Kommando

**resolvectl flush-caches**

gelöscht werden.

*Ettercap* unterstützt DNS-Spoofing und es wird im File */etc/ettercap/etter.dns* konfiguriert. Das File enthält standardmäßig bereits einige Einträge. Fügen Sie auf dem Attacker folgenden Eintrag am Ende des Files hinzu (editieren Sie das File mit *root*-Rechten, d.h., mit Verwendung von *sudo*):

ftp.gnu.org A 10.51.ID.12
---------------------------

Damit werden DNS-Anfragen für die IP-Adresse von [ftp.gnu.org](http://ftp.gnu.org) mit der IP-Adresse des Servers beantwortet – genau was wir wollen. Um die neue Konfiguration zu verwenden, müssen Sie die GUI-

Version von *Ettercap* beenden und neu starten. Damit nun DNS-Spoofing funktionieren kann, muss die DNS-Abfrage beim Attacker «vorbeikommen». Der verwendete DNS-Server (z.B. der DNS-Server der ZHAW) befindet sich dabei ausserhalb des LANs, d.h. wir müssen den Verkehr zwischen dem Client, dem Default-Gateway, sowie dem DHCP Server des lokalen Netzes über den Attacker umleiten (in diesem Fall entspricht also der Default-Gateway der Destination).

Dies erreichen Sie mit einem ARP-Spoofing zwischen dem Client und dem Default-Gateway sowie den DHCP Servern. Konfigurieren Sie nun also das ARP-Spoofing zwischen dem Client und dem Default-Gateway.

Fügen Sie dem Target 1 also folgende IP-Adressen zu:

Gateway : **10.51.ID.1**  
DHCP : **10.51.ID.2**

Und dem Target 2 den Client

**Client: 10.51.ID.11**

Starten Sie nun das eigentliche ARP-Spoofing (Menü-Button *MITM menu* → *ARP poisoning*).

Melden Sie sich auf dem Attacker (nicht auf dem Client!) zuerst (mittels Anonymous-FTP-Login in einem Terminal (username: **anonymous**)) beim richtigen Server *ftp.gnu.org* an, damit Sie sehen, wie das richtige FTP-Login aussieht. Aktivieren Sie anschliessend in *Ettercap* das DNS-Spoofing mit einem Klick auf den Button mit den drei Punkten oben rechts und dann Auswahl von *Plugins* → *Manage plugins* und Doppelklick auf *dns\_spoof*.

Gehen Sie nun zum Client und öffnen ein Terminal. Sie werden feststellen, dass der Client sehr langsam ist. Warum ist dies so?

**Jede Bildschirmzeile wird neu geladen (Laggt). Das System des Attackers ist überlastet und kann die ganzen Netzwerddaten, nicht gut an den Client weitergeben.**

Führen Sie nun auf dem Client ein FTP-Login auf *ftp.gnu.org* durch. Was beobachten Sie? Können Sie sich mit *user/securityzhaw* anmelden? Was schliessen Sie daraus? Betrachten Sie auch hier den relevanten Traffic mit *Wireshark*, um die von *Ettercap* gespoofte DNS-Response zu finden.

**Ja, es hat funktioniert. Meldung auf Ettercap, dass jemand versucht hat sich anzumelden.**

2246... 685.446280050 8.8.8.8	10.51.5.10	DNS	103 Standard query response 0xf587 A ...
-------------------------------	------------	-----	--

Natürlich würde sich der Angreifer in der Realität Mühe geben den gespoofeten Server möglichst identisch nachzubauen, damit der angegriffene Benutzer nichts merkt. Darauf haben wir hier noch verzichtet, die folgende Attacke zeigt aber, dass dies mit frei verfügbaren Tools komfortabel machbar ist. Unser Ziel ist es, DNS-Abfragen für *ctf.cloudlab.zshaw.ch* so zu spoofen, dass die IP-Adresse des Attackers zurückgeliefert wird. Das funktioniert grundsätzlich identisch wie eben mit *ftp.gnu.org* gezeigt. Zusätzlich möchten wir auf dem Attacker-Host mit möglichst wenig Aufwand die Login-Seite von *ctf.cloudlab.zshaw.ch/login* nachbauen, mit dem Ziel Credentials zu sammeln.

Rufen Sie zuerst mit dem Browser auf dem Attacker (nicht auf dem Client!) <https://ctf.cloudlab.zhaw.ch/login> auf. Sie erhalten die Login-Seite. Dies ist die Seite, die Sie weiter unten nachbauen werden. Schliessen Sie als nächstes *Ettercap* auf dem Attacker und erweitern Sie */etc/ettercap/etter.dns* um folgendes:

```
ctf.cloudlab.zhaw.ch A 10.51.ID.10
```

Beachten Sie, dass hier die IP-Adresse des Attackers und nicht wie oben die des Servers zu verwenden ist. Starten Sie dann in gewohnter Manier *Ettercap*, führen Sie ein ARP-Spoofing zwischen Client und Default-Gateway sowie den DHCP Servern durch und aktivieren Sie DNS-Spoofing.

Zusätzlich verwenden Sie auf dem Attacker noch das *Social Engineering Toolkit*<sup>3</sup>, um möglichst einfach die Login-Seite von *ctf.cloudlab.zhaw.ch/login* nachzubauen (zu klonen) zwecks Sammelns der Benutzercredentials. Sie können es über den Menü-Button oben links und Auswahl von *Social Engineering Tools* → *social engineering toolkit* starten. Im sich öffnenden Terminal geben Sie nun folgendes ein:

- 1 (Social-Engineering Attacks)
- 2 (Website Attack Vectors)
- 3 (Credential Harvester Attack Method)
- 2 (Site Cloner)
- IP-Adresse des Attackers (an diesen Host werden die Requests, welche die nachgebaute Webseite erhält (wie z.B. der Login-Request), gesendet)
- <https://ctf.cloudlab.zhaw.ch/login> (die URL der Login-Seite, die geklont werden soll)

Nach kurzer Zeit meldet *Social Engineering Toolkit*, dass alles bereit ist; die geklonte Seite ist nun auf einem lokalen Webserver vorhanden. Sie können das prüfen, indem Sie auf dem Attacker mit dem Browser *localhost* kontaktieren – Sie sollten die geklonte Login-Seite von *ctf.cloudlab.zhaw.ch/login* sehen.

Schliessen Sie nun auf dem Client den Browser und entfernen Sie das vorhandene Profil. Damit werden allfällig gecachte Informationen von *ctf.cloudlab.zhaw.ch/login*, die der Attacke «in die Quere» kommen könnten, gelöscht. Geben Sie dazu in einem Terminal folgendes ein:

```
$ rm -rf /home/user/snap/firefox/*
```

Starten Sie dann den Browser und geben Sie in der Adresszeile <http://ctf.cloudlab.zhaw.ch> ein. Nun werden Sie via DNS-Spoofing auf die geklonte Website auf dem Attacker-Host umgeleitet.

Ihnen werden wahrscheinlich zwei Dinge auffallen, respektive aufgefallen sein: Erstens, dass wir hier <http://ctf.cloudlab.zhaw.ch> nutzen und nicht <http://ctf.cloudlab.zhaw.ch/login>. Das hat damit zu tun, dass die Klon- und Webserver-Funktionalität des *Social Engineering Toolkit* relativ primitiv und beschränkt ist. Zweitens, dass die Webseite diverse Schönheitsfehler hat. So werden z.B. Bilder beim Klonen nicht heruntergeladen, um diese dann durch den Webserver vom *Social Engineering Toolkit* über http auszuliefern. Dasselbe trifft auf andere Ressourcen zu, insbesondere solche, die mittels JavaScript dynamisch geladen werden. Diese Unzulänglichkeiten können mittels besseren Klonmethoden und Nutzung eines Webservers wie Apache für deren Auslieferung, grundsätzlich einfach behoben werden. Ein robusterer Klon einer Webseite kann z.B. mittels der Module *website-scrapers* und *website-scrapers-puppeteer* für Node wie folgt erstellt werden:

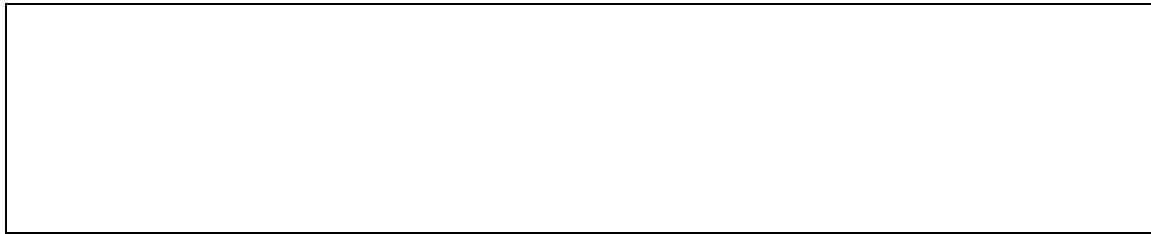
```
const scrape = require('website-scrapers');
const PuppeteerPlugin = require('website-scrapers-puppeteer');
```

<sup>3</sup> <https://github.com/trustedsec/social-engineer-toolkit>

```
scrape({
  urls: ['https://ctf.cloudlab.zhaw.ch/login'],
  directory: '/var/www/html/tpada',
  plugins: [ new PuppeteerPlugin() ]
});
```

Bei Interesse können Sie dies gerne weiterverfolgen, für die Punkte ist dies jedoch nicht erforderlich.

Geben Sie nun als Username die konkatinierten ZHAW-Kürzel Ihrer Gruppe ein und als Passwort das aktuelle Datum. Betrachten Sie dann den Output des *Social Engineering Toolkit*. Was sehen Sie dort?



Machen Sie hiervon einen Screenshot für die Abgabe.

Hat es ein Angreifer erst mal geschafft, im Netzwerk DNS-Requests mitzulesen und DNS-Responses zu erzeugen, dann kann man sich gegen DNS-Spoofing in den meisten Fällen nicht wirklich verteidigen, denn das DNS-Protokoll wird (meist) im PlainText verwendet. Auch hier muss man deshalb mit anderen Massnahmen prüfen, ob man wirklich mit dem richtigen Zielsystem kommuniziert. Im Web bedeutet dies z.B., dass man grundsätzlich immer prüfen sollte, ob man wirklich mit HTTPS auf die Website zugreift, und zudem sollten Zertifikatswarnungen (siehe auch Abschnitt 7) niemals ignoriert werden.

Beenden Sie nun das Spoofing, indem Sie die GUI-Version von *Ettercap* schliessen. Stellen Sie zudem sicher, dass der Client nun wieder auf die korrekte Adresse geleitet wird, indem Sie dort den DNS-Cache mit folgendem Befehl löschen:

```
$ resolvectl flush-caches
```

## 6 Man-in-the-Middle-Attacke auf TLS-Verbindungen – TLS-Stripping

*Ettercap* bietet auch gegen scheinbar sichere Protokolle wie HTTPS (HTTP über TLS) Möglichkeiten für Man-in-the-Middle-Attacken (MITM-Attacken) an. Damit diese Attacken funktionieren, müssen Sie eine kleine Änderung bei der *Ettercap*-Konfiguration vornehmen.

Editieren Sie das File `/etc/ettercap/etter.conf` (mit Verwendung von `sudo`) und entfernen Sie die vier fett markierten Kommentarzeichen (#) in den folgenden Zeilen (jeweils die vor dem Befehlen `redir_command_on/off` und `redir6_command_on/off`):

```
# if you use iptables:
#redir_command_on = "iptables -t nat -A PREROUTING -i %iface
#                    -p tcp --dport %port -j REDIRECT --to-port %rport"
#redir_command_off = "iptables -t nat -D PREROUTING -i %iface
#                    -p tcp --dport %port -j REDIRECT --to-port %rport"

# pendant for IPv6 - Note that you need iptables v1.4.16 or newer to
# use IPv6 redirect
#redir6_command_on = "ip6tables -t nat -A PREROUTING -i %iface -p
#                      tcp -s %source -d %destination --dport %port -j REDIRECT --to-port
#                      %rport"
```

```
#redir6_command_off = "ip6tables -t nat -D PREROUTING -i %iface -p
tcp -s %source -d %destination --dport %port -j REDIRECT --to-port
%rport"
```

Wie wir später in der Vorlesung noch zeigen werden, ist TLS in den meisten Konfigurationen sicher gegen MITM-Attacken. Wir nützen hier aber aus, dass ein Benutzer typischerweise gar nicht bewusst realisiert, dass HTTPS verwendet werden sollte (1. Angriffsvariante) oder dass er ein erhaltenes Zertifikat eventuell nicht genau untersucht (2. Angriffsvariante).

Achten Sie im Folgenden jeweils darauf, dass Sie immer das richtige Protokoll (HTTP oder HTTPS) in der Adresszeile des Browsers verwenden, so wie in der Praktikumsanleitung angegeben!

Die erste Variante, um TLS-Verbindungen anzugreifen nennt sich TLS-Stripping und um das Prinzip zu erklären, betrachten wir eine fiktive Shop-Website, deren Login-Seite über <https://www.turboshop.com/login.html> erreicht wird. Wird im Browser diese URL mit *http* statt *https* eingegeben (oder auf einen entsprechenden *http*-Link geklickt) oder wird das Protokoll bei der Eingabe weggelassen (d.h. [www.turboshop.com/login.html](http://www.turboshop.com/login.html) wird verwendet), so wird der Browser den HTTP Request direkt über TCP und damit ungesichert (d.h. ohne TLS) versenden – er verwendet also <http://www.turboshop.com/login.html>. Um damit korrekt umzugehen, erfolgt als Antwort durch den Webserver in der Regel sogleich ein Redirect auf HTTPS, in diesem Fall also auf <https://www.turboshop.com/login.html>. Gibt der Benutzer anschliessend seine Credentials ein, so werden dieses ebenfalls über HTTPS und damit sicher zur Webapplikation gesendet.

Befindet sich nun ein Angreifer als MITM zwischen Benutzer und Webserver, so wird der MITM den Request an <http://www.turboshop.com/login.html> an den Webserver weiterleiten, worauf der MITM auf HTTPS redirected wird und die Login-Seite über <https://www.turboshop.com/login.html> erhält. Diese Login-Seite wird der MITM dann zum Browser zurücksenden, allerdings nach wie vor über HTTP. Der MITM kommuniziert nun also mit dem Browser über HTTP und mit dem Webserver über HTTPS. Damit kann der Angreifer alle weiteren Daten vom Benutzer problemlos lesen (und ggfs. auch manipulieren). Gibt der Benutzer also nun seine Credentials ein, so werden diese ungesichert zum MITM gesendet und der Angreifer kann damit auf diese zugreifen.

Um dies in der Praxis auszuprobieren, verwenden wir wiederum die Website für ‘*Capture the Flag*’ Herausforderungen an der ZHAW: [ctf.cloudlab.zhang.ch/login](http://ctf.cloudlab.zhang.ch/login). Um es gleich vorwegzunehmen: Das für diese Art der Attacke verwendete Plugin von *Ettercap* funktioniert bei den wenigsten Websites ganz korrekt. Oft funktioniert es gar nicht und teilweise werden die HTML-Dokumente bei der Anpassung der Links verunstaltet. Falls es für die angegebene Seite gar nicht mehr funktionieren sollte, so melden Sie dies bitte umgehend. Vom Prinzip her funktioniert die Attacke immer, man müsste einfach ein Tool schreiben, das mit der gewünschten Website umgehen kann.

Öffnen Sie zuerst den Browser auf dem Attacker-Host und navigieren Sie zu <http://ctf.cloudlab.zhang.ch/login>. Es erfolgt sogleich ein Redirect auf <https://ctf.cloudlab.zhang.ch/login>. Werden hier nun Credentials eingegeben, so werden diese gesichert übertragen und sind für einen Angreifer auf dem Kommunikationskanal nicht einsehbar.

Um nun eine TLS-Stripping Attacke durchzuführen, gehen Sie wie folgt vor: Starten Sie in gewohnter Manier *Ettercap* auf dem Attacker, führen Sie ein ARP-Spoofing zwischen Client und Default-Gateway sowie DHCP Server durch (der Webserver befindet sich ja ausserhalb des LANs) und aktivieren Sie das *sslstrip*<sup>4</sup>-Plugin.

Greifen Sie auf dem Client nun mit dem Browser auf <http://ctf.cloudlab.zhang.ch/login> zu. Falls dies nicht der erste Zugriff darauf ist, entfernen Sie zuerst das vorhandene Profil damit allfällige gecachte Informationen nicht in die Quere kommen:

```
$ rm -rf /home/user/snap/firefox/*
```

<sup>4</sup> <https://github.com/moxie0/sslstrip>

Der Redirect auf HTTPS wird nun nicht erfolgen und der Browser wird über HTTP kommunizieren, was Sie einfach an der dargestellten Adresszeile erkennen können (ebenfalls fehlen Sicherheitsmerkmale wie das Schlosssymbol). Eventuell klappts nicht im ersten Anlauf, laden Sie die Seite dann einfach noch einmal. Die Seite wird im Browser etwas „kümmерlich“ dargestellt, was mit den erwähnten Unzulänglichkeiten des Plugins zu tun hat. Sie sollten aber den Login-Link finden können. Wenn Sie ihn nun klicken, so gelangen Sie wiederum zum Login-Formular, nach wie vor über HTTP.

Geben Sie nun als Username die konkatinierten ZHAW-Kürzel Ihrer Gruppe ein und als Passwort das aktuelle Datum. Was sehen Sie im *Listener*?

```
HTTP : 160.85.255.225:80 → USER: PASS: 12042024 INFO: http://ct
f.cloudlab.zhaw.ch/login
CONTENT: name=stojial1thavajuvmuellt13mafferic&password=12042024&_s
ubmit=Submit&nonce=f3fd480c8b0856132d02784d5363c1a1e5b216764898537b
e7b95bfacd0cc304
```

**Content hat den Usernamen und Password. Pass hat nur den Passwort**

Machen Sie davon einen Screenshot für die Abgabe.

Wie oben erwähnt ist diese Attacke mit einem gewissen technischen Aufwand grundsätzlich bei fast jeder Website durchführbar. Hätten Sie eine solche Attacke als Benutzer bemerkt – unter der Annahme, die Webseite wird inhaltlich korrekt dargestellt? Vermutlich ja, da heutige Browser Verbindungen über HTTP deutlicher als problematisch kennzeichnen (z.B. rot durchgestrichenes Schloss-Symbol) und auch bei Eingabefelder Warnhinweise anzeigen, dass die eingegebenen Daten über eine unsichere Verbindung übertragen werden. Ob dies jedoch auch unter Stress oder bei weniger versierten Personen der Fall wäre, kann zumindest angezweifelt werden.

Man könnte diese Attacke verhindern, wenn man den Browser dazu bringt, eine Website überhaupt nicht mehr über HTTP und nur noch über HTTPS anzusprechen. Genau dies kann mit *HTTP Strict Transport Security (HSTS)* erreicht werden (RFC 6797). HSTS wird heute von allen namhaften Browsern unterstützt und wird tendenziell auch immer häufiger von Websites eingesetzt.

Um die Funktionsweise von HSTS zu verstehen, verwenden wir <http://www.microspot.ch>. Beenden Sie zuerst die GUI-Version von *Ettercap* auf dem Attacker und verwenden Sie für die folgenden Schritte der Client. Schliessen Sie den Browser und führen Sie

```
$ rm -rf /home/user/snap/firefox/*
```

aus, damit allfällige gecachte Informationen nicht in die Quere kommen. Starten Sie nun den Browser und öffnen Sie im Browser den *Network Analyzer* mit *Menü-Button oben rechts* → *More tools* → *Web Developer Tools* → *Network*. Geben Sie im Browser dann <http://www.microspot.ch> (HTTP verwenden!) ein. Die Hauptseite wird darauf über HTTPS geladen. Untersuchen Sie nun die im *Network-Tab* aufgezeichneten Requests und Responses. Sie sollten folgendes erkennen:

- Auf den ersten Request von <http://www.microspot.ch> antwortet der Server mit einer Redirect-Antwort auf <https://www.microspot.ch> (spezifiziert im *Location-Header* der Antwort). Dies ist das typische Verhalten einer Webapplikation, die nur über HTTPS verwendet werden soll.
- Anschliessend führt der Browser den Request von <https://www.microspot.ch> aus. Interessant ist nun der dabei erhaltene Response-Header. Dieser hat einen Eintrag der Form  
*Strict-Transport-Security: "max-age=31536000"; includeSubDomains*

Mit diesem Header weist die Website den Browser an, dass die Website in Zukunft nur noch über HTTPS kontaktiert werden soll. *max-age* gibt dabei an, wie lange das gelten soll (in Sekunden). Der obige Wert 31'536'000 entspricht einem Jahr in Sekunden und ist ein typischer Wert, der oft verwendet wird.

Verifizieren Sie nun das neue Verhalten des Browsers. Schliessen und öffnen Sie den Browser, öffnen Sie wiederum den *Network Analyzer* und geben Sie erneut <http://www.microspot.ch> (HTTP verwenden!) ein. Im *Network*-Tab sollten Sie nun sehen, dass der erste Request zu <https://www.microspot.ch> ging, obwohl Sie <http://> eingegeben haben. Der Browser führt also keine HTTP-Requests mehr aus, wenn mit [www.microspot.ch](http://www.microspot.ch) kommuniziert wird und verwendet stattdessen konsequent HTTPS – weil er von der Website zuvor (siehe oben) mit dem *Strict-Transport-Security*-Header dazu angewiesen wurde. Damit sind auch keine TLS-Stripping Attacken mehr möglich.

Das stimmt allerdings nicht ganz. Es gibt nach wie vor eine Möglichkeit, wie der Angreifer (als MITM) auch bei der Verwendung von HSTS eine TLS-Stripping Attacke durchführen kann. Wie könnte das funktionieren?

**Beim First-Contact wird der Response-Header vom MITM abgeändert, so dass der Browser nicht die Anweisung bekommt, für ein Jahr HTTPS zu verwenden.**

Trotz dieser Unzulänglichkeit ist HSTS ein sehr sinnvolles Feature, das bei HTTPS-only-Websites genutzt werden sollte.

## 7 Man-in-the-Middle-Attacke auf TLS-Verbindungen – Certificate-Spoofing

Die zweite Variante, um TLS-Verbindungen anzugreifen führt eine effektive MITM-Attacke auf TLS durch. Als Beispiel verwenden wir wiederum <https://ctf.cloudlab.zhaw.ch/login>.

Starten Sie *Ettercap* noch nicht und öffnen Sie dann auf dem Attacker <https://ctf.cloudlab.zhaw.ch/login>, worauf Sie die Login-Seite erhalten. Die TLS-Verbindung wurde dabei bereits aufgebaut. TLS-funktioniert prinzipiell so, dass zu Beginn ein sog. TLS-Handshake zwischen Client und Server durchgeführt wird. Dabei authentisiert sich der Server beim Client (dem Browser) mit einem Zertifikat.

Das Zertifikat des Servers können Sie sich anzeigen lassen (*Schlosssymbol links neben der Adresszeile* → *More Information* → *View Certificate*). Da noch keine Attacke durchgeführt wurde, wird das korrekte Zertifikat des Servers angezeigt, das von Let's Encrypt ausgestellt und signiert worden ist (oder auch von einer anderen Certification-Authority, sollte sich dies in der Zwischenzeit geändert haben).

zhaw.ch	R3	ISRG Root X1
Subject Name	-	
Common Name	zhaw.ch	
Issuer Name	-	
Country	US	
Organization	Let's Encrypt	
Common Name	R3	
Validity	-	
Not Before	9/28/2021, 1:02:29 AM (Central European Standard Time)	
Not After	12/27/2021, 12:02:28 AM (Central European Standard Time)	
Subject Alt Names	-	
DNS Name	*.archbau.zshaw.ch	
DNS Name	*.engineering.zshaw.ch	
DNS Name	*.gesundheit.zshaw.ch	
DNS Name	*.ilgi.zshaw.ch	
DNS Name	*.linguistik.zshaw.ch	
DNS Name	*.lsfm.zshaw.ch	
DNS Name	*.psychologie.zshaw.ch	
DNS Name	*.sml.zshaw.ch	
DNS Name	*.sozialarbeit.zshaw.ch	
DNS Name	*.studiportal.gesundheit.zshaw.ch	
DNS Name	*.zshaw.ch	
DNS Name	zshaw.ch	
Public Key Info	-	
Algorithm	RSA	
Key Size	2048	
Exponent	65537	
Modulus	F8:22:30:25:F3:CE:6C:BF:30:81:3C:C9:CA:E3:40:4A:D1:7D:0A:36:30:50:6C:46:BA:31:83:A8:A7:01:D8:19:F6:C...	

Durch die zu Beginn von Abschnitt 6 gemachte Konfigurationsänderung ist *Ettercap* nun so konfiguriert, dass MITM-Attacken auf TLS-Handshakes automatisch ausgeführt werden. Ist also ARP-Spoofing aktiviert und werden die Daten zwischen Client und Destination über den Attacker-Host gesendet, dann wird *Ettercap* bei jedem TLS-Handshake zwischen Target und Destination (bzw. einem dahinterliegenden Server) auch gleich eine Man-in-the-Middle-Attacke durchführen.

Die MITM-Attacke funktioniert im Wesentlichen so, dass *Ettercap* einerseits einen TLS-Handshake mit dem Client (z.B. dem Browser) durchführt und sich dabei gegenüber dem Client als Server ausgibt. Zudem führt *Ettercap* einen zweiten TLS-Handshake mit dem Server durch. Für *Ettercap* gibt es damit also zwei TLS-Verbindungen, eine erste zwischen Client und *Ettercap* und eine zweite zwischen *Ettercap* und dem Server. Dadurch hat *Ettercap* Zugriff auf die unverschlüsselten Daten.

Um sich gegenüber dem Client als Server auszugeben, erzeugt *Ettercap* auf der Basis des vom Server erhaltenen Zertifikats «on the fly» ein gefälschtes Zertifikat. In diesem gefälschten Zertifikat wird ein Public Key verwendet, von welchem *Ettercap* den Private Key kennt. Damit das gefälschte Zertifikat möglichst echt aussieht, wird so viel Information wie möglich vom echten Server-Zertifikat im gefälschten Zertifikat eingefügt. Im Unterschied zum echten Zertifikat wird das gefälschte Zertifikat jedoch von *Ettercap* selbst und nicht von einer offiziellen Certification-Authority signiert. Deshalb wird im Browser auf dem Client während des TLS-Handshake eine Warnung angezeigt, dass mit dem Zertifikat etwas nicht korrekt ist. Ignoriert der Benutzer diese Warnung, so wird der TLS-Handshake komplettiert, worauf *Ettercap* Zugriff (lesend und schreibend) auf die unverschlüsselten Daten erhält, die zwischen Target und Server ausgetauscht werden.

Starten Sie nun in gewohnter Manier *Ettercap* auf dem Attacker und führen Sie ein ARP-Spoofing zwischen Client und Default-Gateway sowie den DHCP Servern durch (auch hier befindet sich der Server ja ausserhalb des LANs). Schliessen Sie auf dem Client den Browser und führen Sie

```
$ rm -rf /home/user/snap/firefox/*
```

aus. Damit „vergisst“ der Browser allfällig gespeicherte Zertifikate im Zusammenhang mit *ctf.cloudlab.zshaw.ch*, welche die nachfolgende Attacke beeinflussen können. Starten Sie den Browser neu und geben Sie die URL

```
https://ctf.cloudlab.zshaw.ch/login
```

ein. *Ettercap* führt nun die MITM-Attacke auf den TLS-Handshake durch. Was sehen Sie im Browser als erstes? Wieso?

The screenshot shows a Firefox browser window with a yellow warning icon. The main message reads: "Das Zertifikat wurde verändert und es gibt eine Warnung aus." Below this, a larger section says: "Warning: Potential Security Risk Ahead". It states: "Firefox detected a potential security threat and did not continue to [www.mozilla.org](#). If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details." It also includes sections for "What can you do about it?", "The issue is most likely with the website, and there is nothing you can do to resolve it.", and "If you are on a corporate network or using antivirus software, you can reach out to the support teams for assistance. You can also notify the website's administrator about the problem." At the bottom right are buttons for "Go Back (Recommended)" and "Advanced...".

Der Browser bemerkt, dass mit dem Server-Zertifikat etwas nicht stimmt und gibt eine Warnung aus, die je nach verwendetem Browser unterschiedlich dargestellt wird. Der Benutzer kann diese Warnung jedoch ignorieren (was in der Realität häufig der Fall sein wird) und die „sichere“ Verbindung trotzdem komplettieren. Ignorieren Sie nun die Warnung, indem Sie beginnend bei *Advanced* die dazu notwendigen Klicks ausführen, worauf Sie „ganz normal“ die Login-Seite erhalten. Geben Sie nun als Username die konkatenierten ZHAW-Kürzel Ihrer Gruppe ein und als Passwort das aktuelle Datum.

Was zeigt Ettercap im unteren Fenster an?

```
HTTP : 160.85.255.225:443 -> USER: PASS: 12042024 INFO: https://ctf.cloudlab.zshaw.ch/login  
CONTENT: name=stojial1maffericmuellti3thavajuv&password=12042024&_submit=Submit&nonce=f5b4f1ff1d3feb98ba6d490f083e1f5d7d8278e83edaf207653c4b409f9255c
```

Machen Sie hiervon einen Screenshot für die Abgabe.

Schauen Sie sich das Zertifikat im Browser an. Inwiefern unterscheidet es sich vom echten Zertifikat des Servers?

```
Original: 3 Zertifikate  
Fälschung: 1 Zertifikat (Webseite) und Webseite hat weniger  
Properties(Kein Authority Info) und der Authority Key ist nicht derselbe
```

Dieser Fall mit *ctf.cloudlab.zshaw.ch/login* ist natürlich noch relativ harmlos. Denken Sie an interessantere Varianten, zum Beispiel aus dem E-Banking-Umfeld. Deshalb ist es enorm wichtig, dass Zertifikatwarnungen nicht einfach ignoriert werden. Leider wird dies in der Realität oft nicht beachtet.

Eine potenzielle Abhilfe existiert mit *HTTP Public Key Pinning (HPKP)*, gemäss RFC 7469). Die Idee dabei ist, dass die Website dem Browser explizit mitteilt, welche Zertifikate bzw. ausstellenden Certification-Authorities für diese Website als akzeptabel betrachtet werden sollen.

Betrachten wir dies gleich an einem konkreten Beispiel. Beenden Sie zuerst die GUI-Version von *Etercap* auf dem Attacker. Schliessen Sie dann auf dem Client den Browser und führen Sie

```
$ rm -rf /home/user/snap/firefox/*
```

aus, damit allfällige gecachte Informationen nicht in die Quere kommen. Starten Sie auf dem Client nun den Browser und öffnen Sie im Browser den *Network Analyzer*. Geben Sie im Browser <https://projects.dm.id.lv> ein, worauf die Website über HTTPS geladen wird. Untersuchen Sie dann den ersten aufgezeichneten Request und die zugehörige Response. Dabei sollten Sie einen Response-Header in der folgenden Form sehen:

```
Public-Key-Pins:  
"pin-sha256="8MFHQC9XAUF/XBmQ/mZ8S/XEc5aSYz0lj0EHTj870+s=";  
pin-sha256="tpFbv65QoYvcWNV17gAEd1FAWln/pjL8Fo2+f1pTrC8=";  
pin-sha256="WKBsAc1TiYDM7EEJ5yUmrWmp9DxWm/hG+D+wCCLA24=";  
pin-sha256="nxpEakAMgSw92zksspA8LdZyrdW/MGGr70VfcIT7DBU=";  
max-age=31536000; includeSubDomains"
```

Mit diesem Response-Header instruiert die Website den Browser, dass im Zusammenhang mit dieser Website in Zukunft nur noch Zertifikate akzeptiert werden sollen, bei denen der Hashwert des Public Keys einen der obigen Hashwerte aufweist oder die von einer Certification-Authority ausgestellt wurden, deren Public Key einen solchen Hashwert aufweist. Mit dem Header werden also die zulässigen Public Keys «festgenagelt» – daher auch der Begriff *Public Key Pinning*. Genau wie bei HSTS gibt es zudem auch hier einen *max-age*-Wert, der spezifiziert, wie lange sich der Browser «dies merken soll».

Wird für das Pinning nun der Hashwert des Public Keys des Zertifikats der Website verwendet, so sind Attacken mit gefälschten Zertifikaten nicht mehr möglich, denn diese weisen mangels Kenntnis des Private Keys der Website andere Public Keys auf. Und beim Pinning auf Hashwerte der Public Keys der ausstellenden Certification-Authorities können Zertifikate für die Website nur noch bei diesen beschafft (bzw. ergaunert) werden.

Um zu verifizieren, dass dies auch funktioniert, gehen Sie wie folgt vor: Starten Sie *Etercap* auf dem Attacker und führen Sie ein ARP-Spoofing zwischen Client und Default-Gateway sowie den DHCP Servern durch. Schliessen Sie dann auf dem Client den Browser, starten Sie ihn erneut und geben Sie <https://projects.dm.id.lv> ein. Wie verhält sich der Browser? Entspricht dies Ihren Erwartungen?

#### Secure Connection Failed

An error occurred during a connection to projects.dm.id.lv. Peer's certificate has an invalid signature.

Error code: SEC\_ERROR\_BAD\_SIGNATURE

- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the website owners to inform them of this problem.

[Learn more...](#)

Try Again

Das klingt nach einer sehr guten Lösung, leider gibt es in der Praxis aber einige Problemfelder beim Einsatz von HPKP, die dazu führen, dass HPKP erst recht selten verwendet wird und dass es unklar ist, ob es je grössere Verbreitung findet.

Ein erstes Problem ist dasselbe wie bei HSTS: Wenn es der Angreifer schafft, bei der ersten Kontakt-aufnahme eine MITM-Attacke durchzuführen, dann ist HPKP wirkungslos, denn der Angreifer kann sein gefälschtes Zertifikat verwenden und den *Public-Key-Pins*-Response-Header einfach unterdrücken. Dies wäre aber kein Argument gegen die Verwendung von HPKP.

Sehr viel stärker wiegen andere Probleme. Nehmen Sie an, eine Website verwendet im Response-Header genau einen Hashwert, und zwar den des Public Keys im eigenen Zertifikat. Nehmen Sie nun an, die Website «verliert» ihren Private Key (nicht kompromittiert, sondern einfach z.B. bei einer Fehlkonfiguration gelöscht). Die Website braucht damit ein neues Zertifikat mit einem neuen Public Key.

Wenn nun ein Browser, der die Website bereits früher besucht hat, eine Verbindung zur Website aufbaut, wie wird sich der Browser dann verhalten? Und was bedeutet das für die Website?

Kunden mit dem alten Public Key, können nicht auf die Webseite zugreifen.

Mit HPKP ist es also möglich, eine Denial-of-Service-Attacke (DoS-Attacke) „gegen sich selbst“ durchzuführen. Durch die Verwendung von mehreren Hashwerten, insb. auch von den Public Keys mehrere Certification-Authorities, kann dieses Risiko zwar reduziert werden, aber dies zeigt, dass man sich bei der Verwendung von HPKP einiges überlegen muss. Um das Risiko etwas zu reduzieren, sollen gemäss HPKP-Standard übrigens immer die Hashwerte von zwei verschiedenen Public Keys verwendet werden. Dies wird von den Browsern zumeist aber nicht erzwungen und die Browser akzeptieren den Header auch dann, wenn er nur einen Hashwert beinhaltet.

Ein noch grösseres Problem ist, dass mit HPKP ein externer Angreifer eine DoS-Attacke gegen eine Website durchführen kann. Nehmen Sie an, ein Angreifer hat sich Zugang zu einer HTTPS-Website verschafft, so dass er die Konfiguration der verwendeten Response-Header manipulieren kann. Zudem kann er auch das auf dem Server vorhandene Schlüsselmaterial löschen. Wie kann er in diesem Zusammenhang HPKP ausnutzen, um eine DoS-Attacke gegen die Website durchzuführen?

Er kann den Public Key verändern oder den Private Key löschen, dann wird der Server nicht verifiziert.

Darum wird HPKP heute nicht mehr empfohlen und von keinem aktuellen Browser per default unterstützt. So muss es auch bei Firefox extra enabled werden.

Weitere Informationen zur HPKP-Problematik erhalten Sie unter  
<https://blog.qualys.com/product-tech/2016/09/06/is-http-public-key-pinning-dead>.

[https://en.wikipedia.org/wiki/HTTP\\_Public\\_Key\\_Pinning](https://en.wikipedia.org/wiki/HTTP_Public_Key_Pinning)

## Praktikumspunkte

In diesem Praktikum können Sie **4 Praktikumspunkte** erreichen:

- 2 Punkte erhalten Sie für die Aufgaben in den Kapitel 1-5
- 2 Punkte erhalten Sie für die Aufgaben in den Kapitel 6 und 7

## Anhang: Trouble-Shooting

Nach dem Deaktivieren des DNS Spoofing führt ein Zugriff auf die zuvor gespoofte Domain (z.B. `ftp.gnu.org`) immer noch auf die gespoofte Zieladresse.

Stellen Sie jeweils sicher, dass auch effektiv eine DNS-Anfrage ausgelöst wird und die Antwort nicht aus dem DNS-Cache kommt. *Ettercap* schickt je nach Version DNS-Antworten mit einer langen *Time to live* (v0.8.3.1 z.B. 1 Stunde für Antworten mit IPv4 und 5 Sekunden für Antworten mit IPv6 Adressen). Löschen Sie den DNS-Cache wie folgt:

```
$ resolvectl flush-caches
```