# Seminar IT-Sicherheit
# Capturing Traffic

Eingereicht am:
12.11.2016

Eingereicht von:
Tobias Schwarz
Fachrichtung: Wirtschaftsinformatik
Fachsemester: 7

Betreut von:
Prof. Dr. Gerd Beuster

# Inhalt

# 1. Einführung in die Thematik

## 1.1. Einleitung

Die folgenden Seiten behandeln das Thema rund um das Abfangen, Analysieren und Manipulieren von Traffic. Dabei ist nicht nur der Traffic gemeint, welcher dem eigenen Host-System zugeordnet ist, sondern insbesondere auch der Traffic anderer Systeme, die im selben Netzwerk kommunizieren. Hier werden mit Hilfe einiger Tools die Inhalte der Traffic-Pakete analysiert sowie unter anderem Man-In-The-Middle-Angriffe durchgeführt, durch die man fremden Traffic und damit andere Host-Systeme maßgeblich beeinflussen und sogar geheime Daten abfangen kann.

## 1.2. Was ist Traffic?

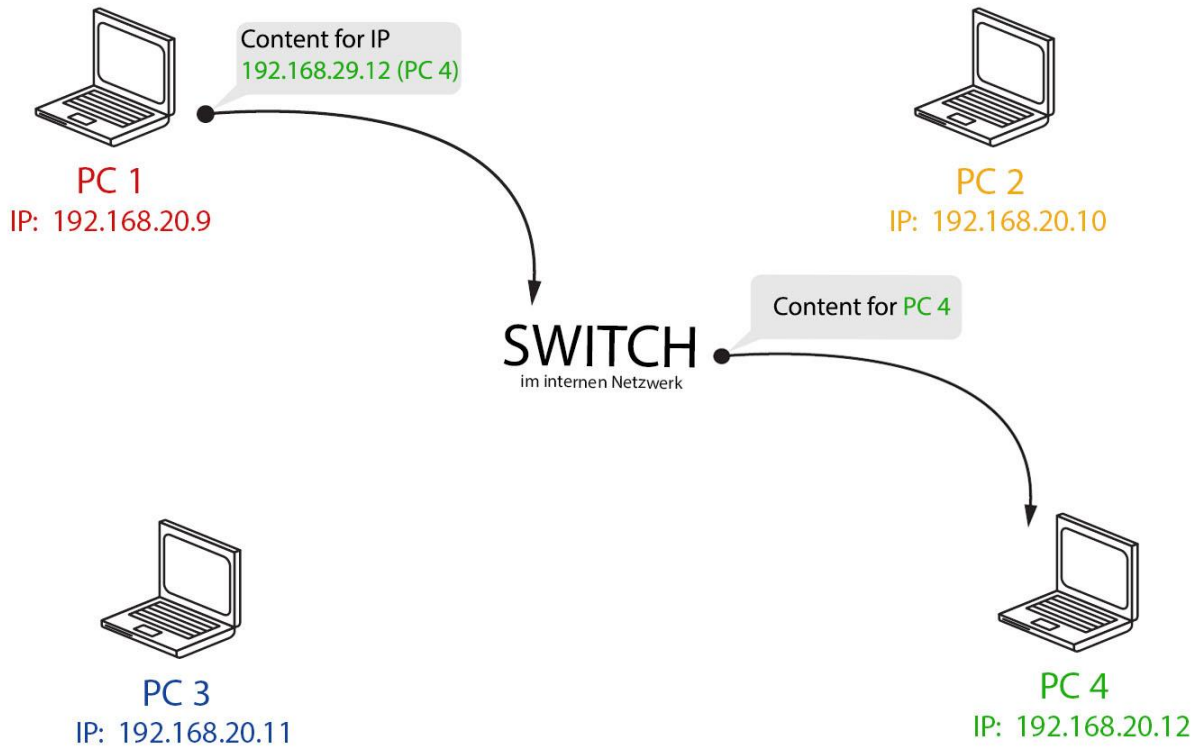
Als Traffic [2] bezeichnet man das Datenaufkommen in Computernetzwerken. Datenpakete werden zwischen Host-Systemen ausgetauscht und es entsteht ein Datenverkehr (engl. traffic). Dieser Vorgang findet sowohl in internen Netzwerken, als auch im Internet statt.
Datenpakete, über die Systeme oder einzelne Dienste miteinander kommunizieren, enthalten sämtliche ausgetauschte Daten dieser Hosts. Hinter diesen Daten können sich zum Teil hochsensible Informationen, wie Benutzernamen und zugehörige Passwörter verbergen, die jedoch fast ausschließlich verschlüsselt übertragen werden und deshalb nicht direkt einsehbar sind.

## 1.3. Traffic in Netzwerken

Bei der Kommunikation zwischen Hosts in Netzwerken muss für unsere Zwecke eine generelle Unterscheidung vorgenommen werden. Für das Abfangen von Traffic spielt es eine wichtige Rolle, ob in einem Netzwerk einzelne Systeme über Switches oder Hubs verbunden sind.

### 1.3.1 Netzwerke, dessen Systeme durch Switches verbunden sind

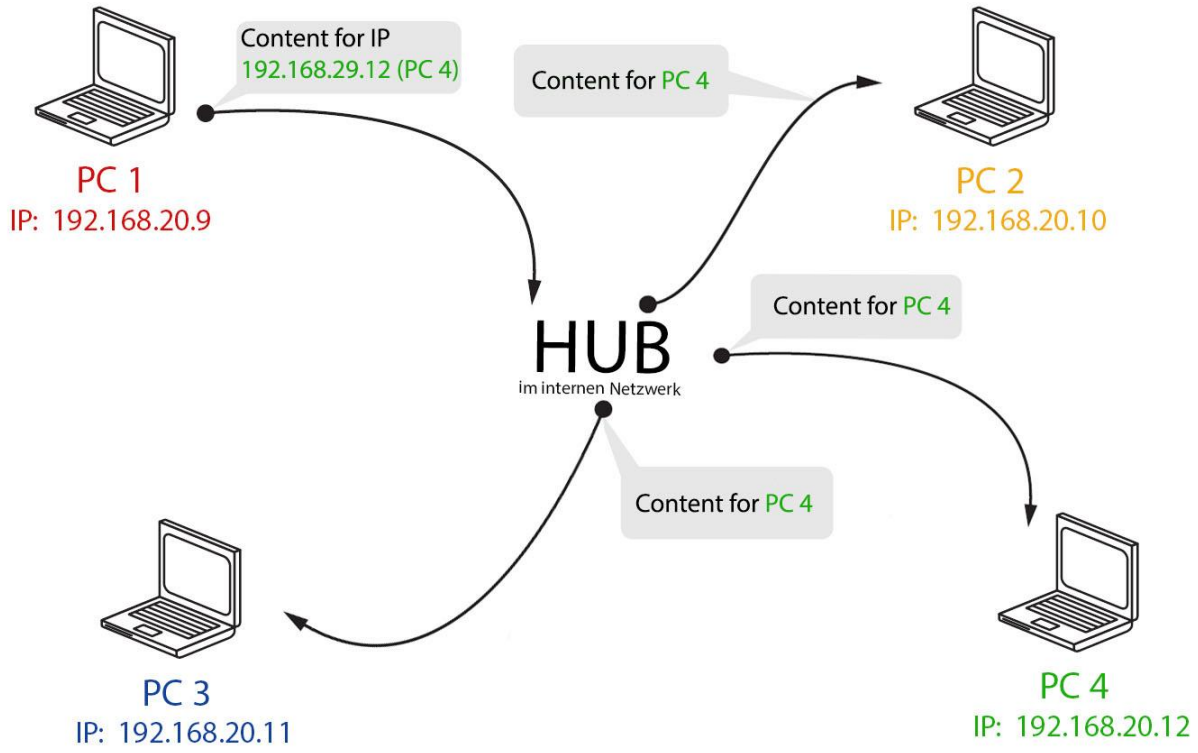
Ist ein Netzwerk über einen Switch [3] zusammengesetzt, so gestaltet sich das Abfangen von fremden Datenpaketen als schwierig. Ein Switch sendet Datenpakete lediglich an die korrekte Zieladresse weiter. Dadurch gelangen keine Datenpakete an der Kommunikation unbeteiligte Host-Systeme (siehe Abb. 1).




(1) Abbildung 1: Austausch von Informationen in einem internen Netzwerk über einen Switch

### 1.3.2 Netzwerke, dessen Systeme durch Hubs verbunden sind

Entstehen die Verbindungen eines Netzwerkes jedoch über einen oder mehrere Hubs [4], so ist es denkbar einfach, sich Informationen aus fremden Datenpaketen anzueignen. Anders als ein Switch verhält sich ein Hub so, dass er empfangene Datenpakete an alle angeschlossenen Systeme broadcastet und den Hosts überlässt, den richtigen Adressaten zu finden.
Auf diese Weise findet eine Datenübermittlung auch mit an der Kommunikation eigentlich unbeteiligten Systemen statt, welchen es so möglich gemacht wird, fremden Traffic abzufangen (siehe Abb. 2).




(2) Abbildung 2: Austausch von Informationen in einem internen Netzwerk über einen Hub

### 1.3.3 Traffic in virtuellen Netzwerken

Häufig stellt jedoch auch ein Hub Funktionalitäten eines Switches bereit, so dass es selbst in einem durch Hubs verbundenen Netzwerk kein fremder Traffic für uns zugänglich ist, ohne dass wir den Switch/Hub austricksen, damit er uns Packages zukommen lässt.

Virtuelle Netzwerke hingegen agieren wie ein klassischer Hub, da sämtliche virtuellen Maschinen auf derselben physikalischen Maschine ausgeführt werden und somit denselben Netzwerkadapter verwenden. In einem solchen Netzwerk ist es mit den richtigen Tools möglich sowohl den Traffic der virtuellen, als auch der Host-Maschine einzusehen. Dies funktioniert unabhängig von dem Umstand, ob in der Netzwerkumgebung einen Switch angeschlossen ist oder nicht.

# 2. Traffic Exploitation

## 2.1. Capturing and Analyzing Traffic

Im Folgenden wird Hilfe des Tools Wireshark der Traffic in einem virtuellen Netzwerk untersucht. Wahlweise lässt sich auch ein nicht-virtuelles Netzwerk simulieren, weshalb sich virtuelle Netzwerke besonders als Testumgebung eignen.

### 2.1.1 Wireshark

Wireshark [5] ist ein Network Protocol Analyzer mit grafischer Oberfläche. Mit Hilfe von Wireshark ist es möglich, sämtlichen auf der Maschine eintreffenden sowie ausgehenden Traffic abzufangen und zu analysieren. Hierzu zählt insbesondere auch Traffic, der von anderen virtuellen Systemen auf der Maschine ausgeht. Sowohl Ethernet, als auch Wireless, Bluetooth und viele andere Arten von Traffic können analysiert werden und zudem sind sogar Rekonstruktionen von Audiodateien eines VoIP-Telefongespräches möglich.

### 2.2.2 Capturing Traffic

Mit Wireshark oder einem vergleichbaren Tool ist es nun möglich, einen oder mehrere Netzwerkadapter zu überwachen und sämtliche darüber ausgetauschten Pakete aufzuzeichnen. Dazu zählt der Datenverkehr der eigenen Maschine sowie sämtlicher Broadcast-Traffic des Netzwerkes. In einem Hub-Netzwerk kann dies bedeuten, dass Wireshark bereits Teile fremden Traffics aufgreift.
In virtuellen Netzwerken hingegen erhalten wir zusätzlich Einsicht in alle Pakete, da diese sich über denselben Netzwerkadapter bewegen. Das Verhalten eines switched Network kann durch Deaktivieren des Promiskuitiven Modus in den Wireshark-Einstellungen simuliert werden.
Die aufgezeichneten Datenpakete werden nun in einer Tabelle aufgelistet. Hierbei werden beinahe alle Übertragungsprotokolle unterstützt. Zu den Wichtigsten zählen:

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- IPv4 (Internet Protocol v. 4)
- IPv6 (Internet Protocol v. 6)
- ARP (Address Resolution Protocol)
- DNS (Domain Name System)
- Anwendungsprotokolle

Anwendungsspezifische Protokolle lassen sich einem Dienst direkt zuordnen, wie beispielsweise das „Dropbox LAN sync Discovery Protocol“. Nach den oben genannten Protokollen lässt sich die Liste problemlos sortieren.

### 2.2.3 Analyzing Packages

Bei der Analyse eines einzelnen Paketes wird als Erstes ein Eintrag in der Tabelle ausgewählt. Bereits im Tabelleneintrag sind das dem Paket zugehörige Protokoll, Source und Destination des Paketes sowie unter anderem eine Paketbeschreibung erkennbar.

In vielen Fällen reicht diese Beschreibung bereits aus, um erste Rückschlüsse auf die Art des Paketes zu ziehen. Weitere Informationen können nach der Paketauswahl auch der Detailansicht entnommen werden (siehe Abb. 3).

```
  4 3.072063    192.168.178.27    192.168.178.255    UDP       305 54915→54915 Len=263
  5 3.754931    192.168.178.29    192.168.178.255    DB-LSP…   186 Dropbox LAN sync Discovery Protocol
  6 3.789811    192.168.178.27    192.168.178.255    DB-LSP…   185 Dropbox LAN sync Discovery Protocol

> Frame 6: 185 bytes on wire (1480 bits), 185 bytes captured (1480 bits) on interface 0
> Ethernet II, Src: AvmGmbh_41:d8:61 (24:65:11:41:d8:61), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.178.27, Dst: 192.168.178.255
> User Datagram Protocol, Src Port: 17500, Dst Port: 17500
> Dropbox LAN sync Discovery Protocol

0000  ff ff ff ff ff ff 24 65  11 41 d8 61 08 00 45 00   ......$e .A.a..E.
0010  00 ab 0f 91 00 00 80 11  44 45 c0 a8 b2 1b c0 a8   ........ DE......
0020  b2 ff 44 5c 44 5c 00 97  85 26 7b 22 68 6f 73 74   ..D\D\.. .&{"host
0030  5f 69 6e 74 22 3a 20 34  33 33 36 34 36 37 32 35   _int": 4 33646725
0040  31 38 32 33 34 30 31 39  32 37 39 31 35 39 33 31   18234019 27915931
0050  35 37 33 30 31 38 34 35  38 37 37 38 38 2c 20 22   57301845 87788, "
0060  76 65 72 73 69 6f 6e 22  3a 20 5b 32 2c 20 30 5d   version" : [2, 0]
0070  2c 20 22 64 69 73 70 6c  61 79 6e 61 6d 65 22 3a   , "displ ayname":
0080  20 22 22 2c 20 22 70 6f  72 74 22 3a 20 31 37 35    "", "po rt": 175
0090  30 30 2c 20 22 6e 61 6d  65 73 70 61 63 65 73 22   00, "nam espaces"
00a0  3a 20 5b 38 38 39 30 30  33 38 30 39 2c 20 31 31   : [88900 3809, 11
00b0  39 32 36 34 37 35 37 5d  7d                        9264757] }
```

(3) Abbildung 3: Detailansicht eines Paketes für Dropbox-Synchronisation

In dem obigen Beispiel werden Synchronisationsinformationen über ein Anwendungsspezifisches Protokoll übertragen. Erkennbar sind unter anderem die folgenden Daten:

- Sender (Source)
- Empfänger (Destination)
- Verwendetes Protokoll
- Package Length
- Port
- Datenfeld

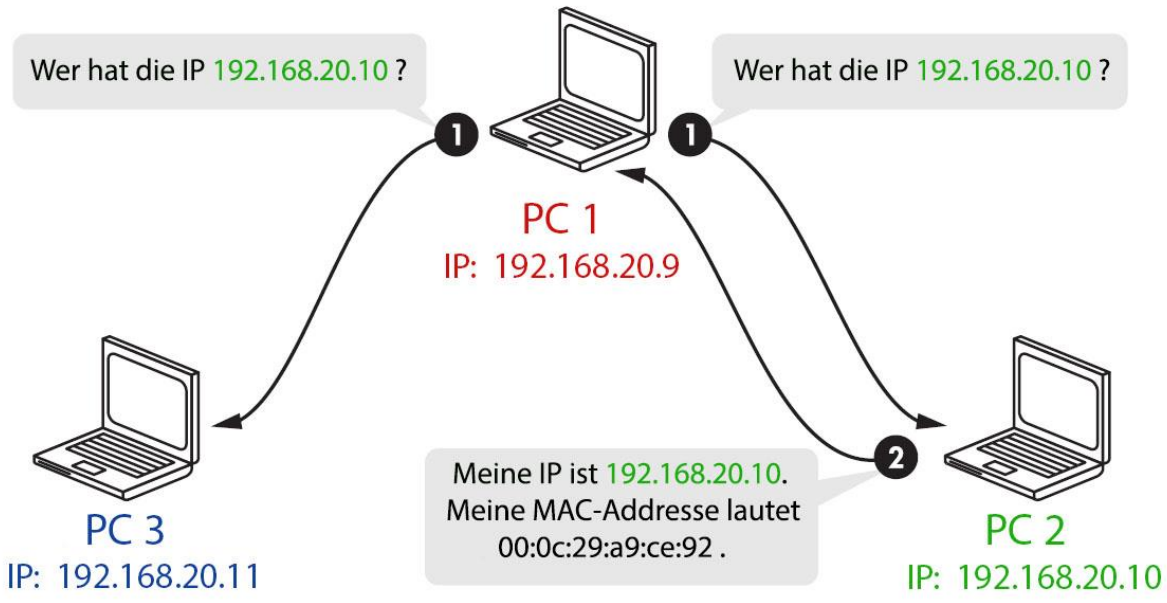
Die Datenfelder der Pakete enthalten dabei die übertragenen Informationen, die nichts mit der Art der Übertragung zu tun haben. Dem Datenfeld der Abb. 3 sind Informationen in Plain-Text zu entnehmen. Meistens sind diese Daten jedoch verschlüsselt und zudem auf mehrere Pakete aufgeteilt. Häufig lassen sich auch zusätzliche Informationen gewinnen, indem man den TCP-Stream eines Paketes verfolgt. So werden weitere Informationen sichtbar, die sich nur durch ein Betrachten mehrerer Pakete im Zusammenhang auslesen lassen.

## 2.1. ARP Cache Poisoning

Das Betrachten des eigenen Traffics fördert durchaus Informationen zu Tage. Allerdings befinden sich darunter keine Daten, deren Informationen nicht ohnehin schon in unserem Besitz sind. Um auch auf fremden Traffic zugreifen zu können, benötigt Wireshark externe Unterstützung. Im Folgenden wird mit Hilfe des ARP Cache Poisoning [1] Traffic von zwei fremden Maschinen zugänglich gemacht.

### 2.1.1. Basics: ARP – Address Resolution Protocol

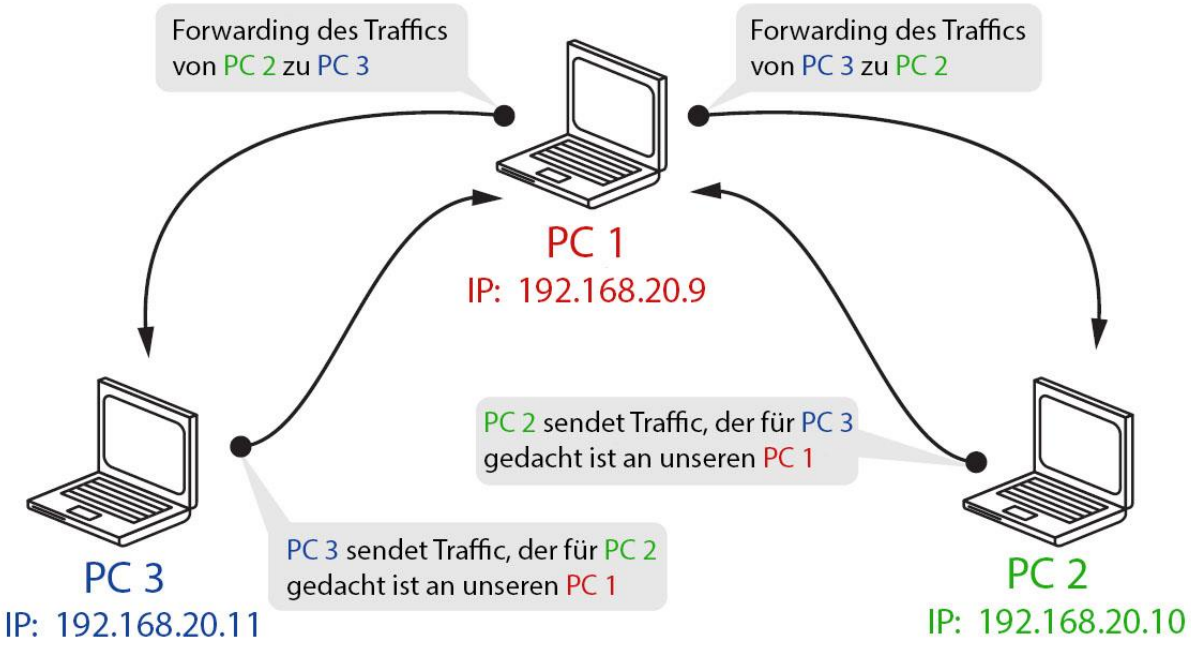
Die folgende Abbildung (Abb. 4) zeigt das Vorgehen des Address Resolution Protocol. Dabei wird ein ARP-Request mit einer aufzulösenden IP-Adresse per Broadcast an alle anderen Systeme im Netzwerk gesendet. Vom angesprochenen System erhält der Requester ein ARP-Reply mit der Zugehörigen MAC-Adresse zurück.




(4) Abbildung 4: Beispielhafter ARP-Request und darauf folgender Reply

### 2.1.2. ARPspoof

Bei einem ARP-Request besteht jedoch keine Garantie, dass ein darauf folgender ARP-Reply korrekt ist. Jedem Hostsystem ist es möglich, auf diesen Request eine Beliebige MAC-Adresse zurückzuliefern, was den folgenden Man-In-The-Middle-Angriff möglich macht.
Hierbei wird der ARP Cache so manipuliert, dass Traffic, der für Dritte bestimmt sind stattdessen auf unserer Maschine eintrifft. Über ein Forwarding des Traffics zur vom manipulierten Host eigentlich adressierten sichergestellt, dass der Traffic auch beim ursprünglichen Ziel eintrifft und die Kommunikation nicht unterbrochen wird. Führt man das ARP Cache Poisoning nun auch bei dem Empfänger des Traffics durch, so wird sämtlicher zwischen den Maschinen ausgetauschter Traffic auch auf unserem System eintreffen und somit einsehbar sein (siehe Abb. 5).




(5) Abbildung 5: Man-In-The-Middle-Attack über ARP Cache Poisioning

Ein solcher Angriff lässt sich leicht mit dem Tool ARPspoof [6] durchführen. Das Tool ist unter anderem Teil der umfangreichen Penetration Testing Umgebung Kali Linux [7]. Hierbei benötigt das Programm jeweils nur ein **Target** und eine IP-Adresse, für die sich unsere Maschine ausgeben soll. Ein beispielhafter Aufruf könnte so aussehen:

*root@kali:~# arpspoof -i eth0 -t **192.168.20.11** 192.168.20.10*

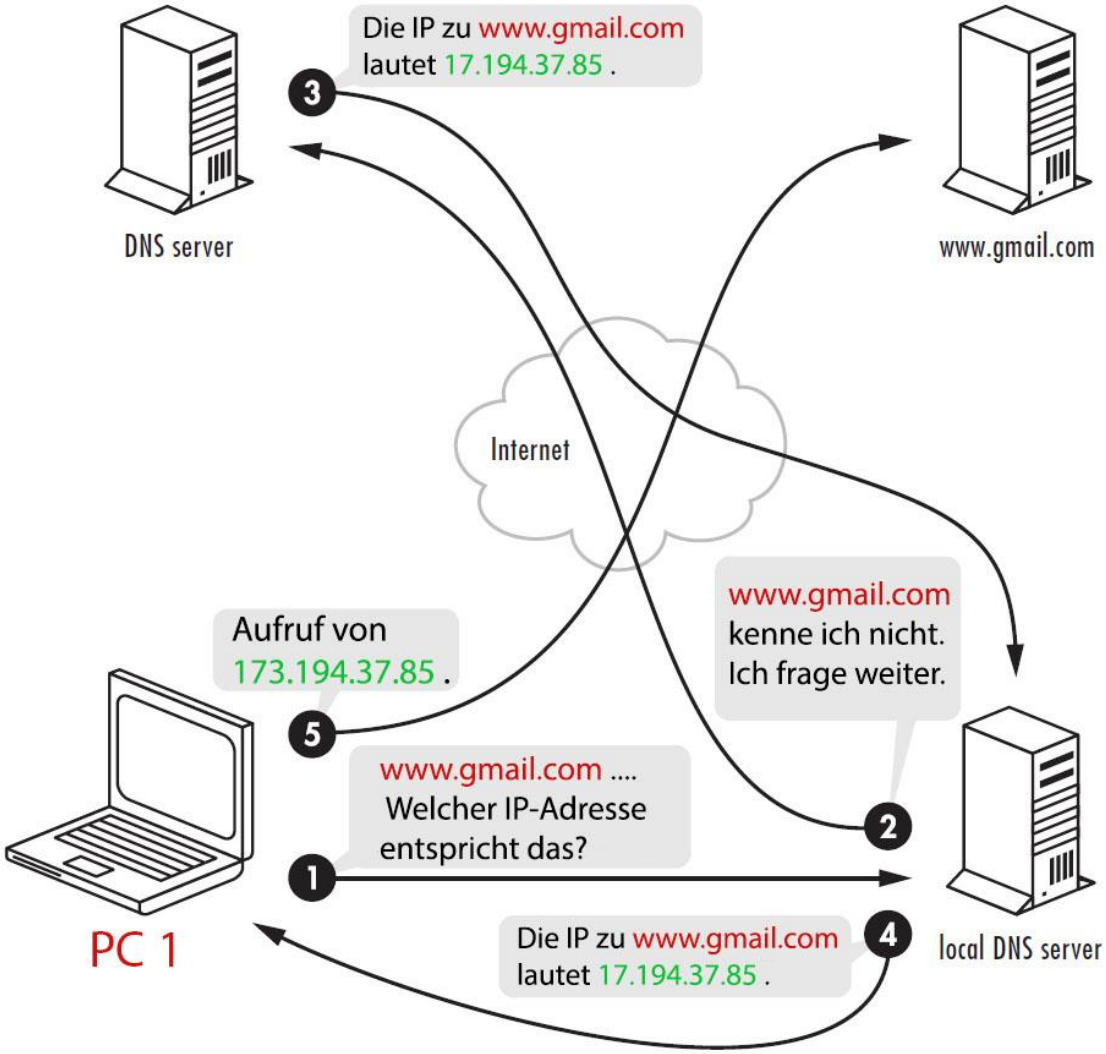
Nachdem über zwei solche Aufrufe sowohl der Sender als auch der ursprüngliche Empfänger ihren Traffic an unser System übertragen, kümmert sich ARPspoof selbstständig um die Weiterleitung des Traffics zu den Ursprungsadressaten.

## 2.3. DNS Poisoning

Das DNS-Poisoning [1] lässt zu, dass wir Maschinen aus dem internen Netzwerk beim Aufruf bestimmter Internet-Seiten auf beliebige andere Sites umlenken können.

### 2.3.1. Basics: DNS – Domain Name System

Das Domain Name System beschäftigt sich mit der Auflösung von Internet-Adressen. Dabei wird die IP-Adresse einer solchen www-Adresse über DNS-Server abgefragt (siehe Abb. 5).




(6) Abbildung 6: DNS-Vorgang beim Aufruf der Internetseite www.gmail.com

### 2.3.2. DNSspoof

Das Vorgehen beim DNS Poisoning ähnelt stark dem der ARP-Man-In-The-Middle-Attack. Zunächst wird ein zu manipulierendes System über ARPspoof davon überzeugt unsere Maschine als Gateway zu betrachten, während der bei uns eintreffende Traffic dorthin weitergeleitet wird. Anschließend wird eine Textdatei erstellt, die eine manipulierte DNS-Auflösung beinhaltet. Ein Aufruf unter Kali könnte wie folgt aussehen:

*root@kali:~# cat hosts.txt*

*192.168.20.9 www.gmail.com*

Der Name der Datei ist hierbei unerheblich. Anschließend wird das Tool DNSspoof [1] aufgerufen, welches lediglich den Netzwerkadapter sowie die vorher erstellte Textdatei benötigt. Ein Aufruf könnte beispielsweise so aussehen:

*root@kali:~# dnsspoof -i eth0 -f hosts.txt*

Anschließend gelangt die mit ARPspoof manipulierte Maschine bei einem Aufruf der Internetseite www.gmail.com auf eine von uns bestimmte Site (siehe Abb. 7).


(7) Abbildung 7: Beispiel einer DNS Umleitung auf einen auf der Kali-Maschine laufenden Apache Server

## 2.4. Capturing Userdata

Wird Traffic umgeleitet, so ist es möglich, sensible Daten aus diesem Datenverkehr zu filtern. Entsprechende Möglichkeiten werden im folgenden Abschnitt erläutert.

### 2.4.1. Basics: TLS – Transport Layer Security

Das Ziel von TLS [8] ist es, sensible Daten wie etwa Login-Informationen oder Kreditkartendaten bei der Übertragung im Internet zu schützen. Um dies zu gewährleisten kommuniziert der Browser mit den aufgerufenen Internetseiten über Zertifikate (Certificates). Bevor der Browser die aufgerufene Seite anzeigt, findet eine Überprüfung des Certificates der Seite statt. Nur wenn ein solches Zertifikat gültig ist, beginnt eine TLS-gesicherte Kommunikation.

### 2.4.2. Capturing Userdata via Man-In-The-Middle-Attacks

Ettercap [9] ist wie Dnsspoof und Arpspoof ein Tool von Kali Linux, welches in der Lage ist, diverse Man-In-The-Middle-Attacken durchzuführen. Beispielswiese lassen sich die mit DNSspoof und ARPspoof realisierten Attacken auch mit Ettercap durchführen. In diesem Fall soll das Tool dazu verwendet werden, Login-Daten in Erfahrung zu bringen. In diesem Fall reicht ein Aufruf von Ettercap mit dem Gateway des Netzwerkes und der IP der Target-Maschine:
*root@kali:~# ettercap -Ti eth0 -M arp:remote /192.168.178.29//192.168.178.1/*
Nach einem solchen Aufruf wartet Ettercap auf beginnende TLS-Übertragungen. Führt die Target-Maschine nun eine TLS-gesicherte Datenübermittlung durch, so gibt Ettercap eingegebenen Informationen in Plain Text aus.

Der große Nachteil an diesem Vorgehen ist im Folgenden zu erkennen:


(8) Abbildung 8: Im Browser auftretende Zertifikats-Warnung

Durch eine solche Attacke über Ettercap tritt in allen Browsern eine Zertifikatswarnung auf, da das durch Ettercap an den Browser übermittelte Zertifikat nicht gültig ist. Userdaten können von Ettercap also nur übermittelt werden, falls eine solche Zertifikatswarnung, falls der Browser dies zuläst, ignoriert wird, und der Benutzer trotzdem einen Login durchführt.

### 2.4.3. SSL-Stripping

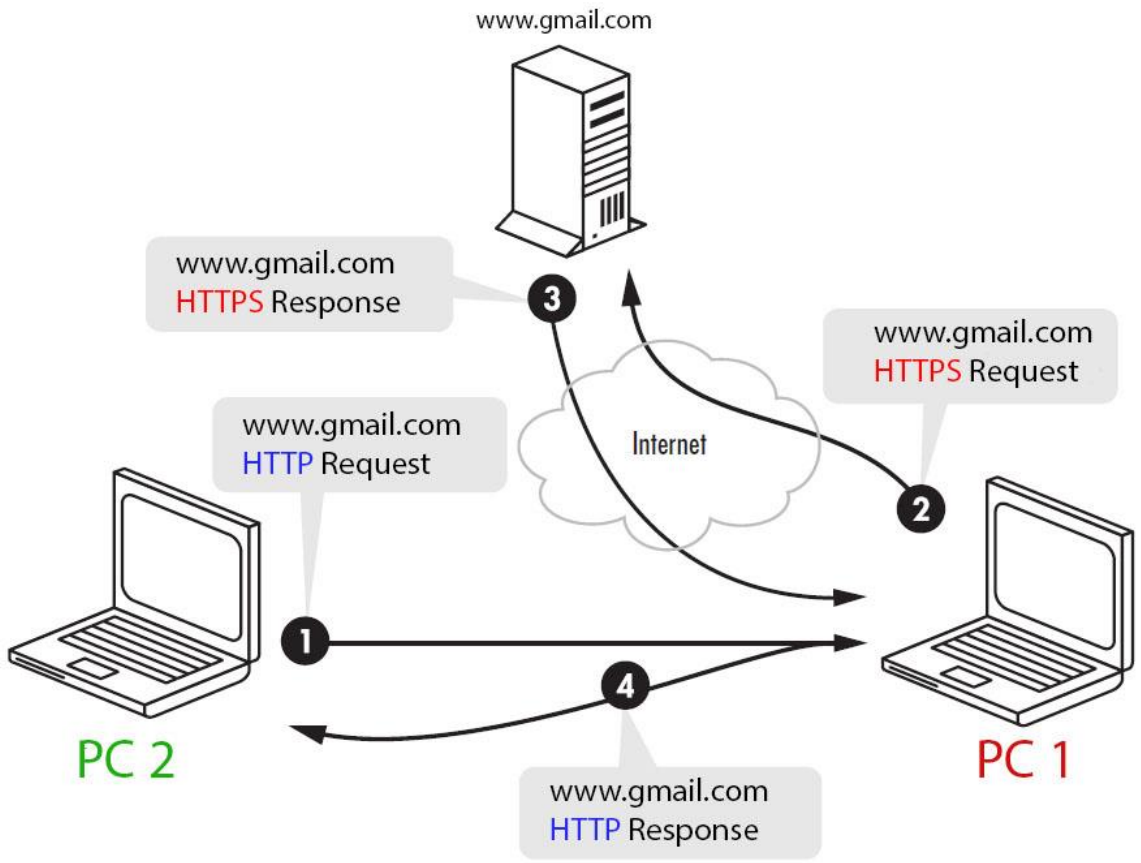
Eine weitere Methode, an Userdaten zu gelangen ist das SSL-Stripping. Hierbei wird zunächst ein Target über das ARPspoofing infiziert. Anschließend wird der auf unserer Maschine auf Port 80 eintreffende Traffic an das Kali Linux Tool SSLstrip [1] umgeleitet. Ein beispielhafter Aufruf würde folgendermaßen aussehen:

*root@kali:# iptables -t nat -A PREROUTING -p tcp -- destination-port 80*
*-j REDIRECT -- to-port 8080*

Nun muss nur noch SSLstrip mit dem vorher ausgewählten Port aktiviert werden:

*root@kali:# sslstrip -l 8080*

Die vom Target aufgerufenen Seiten werden von SSLstrip von einer HTTPS in eine HTTP-Site umgewandelt und zur Verfügung gestellt. Nun wartet das Tool auf eingehende SSL-Übertragung als http. Wird eine solche Übertragung erkannt, beginnt folgender Prozess (siehe Abb. 9).




(9) SSL-Stripping (Man-In-The-Middle-Attack)

Der auf unserer Maschine eintreffende HTTP-Request wird von SSLstrip als HTTPS-Request an den eigentlichen Adressaten weitergeleitet. Aus dem daraus resultierenden HTTPS-Response erstellt das Tool einen HTTP-Response, der an das manipulierte System weitergeleitet wird. Auf diese Weise ist SSLstrip in der Lage, die Userdaten in Plain Text auszulesen und auszugeben während beim Target keine Zertifikatswarnung hervorgerufen wird.

### 2.4.4. Die Zukunft von HTTP und HTTPS

Großkonzerne wie Google sind sehr daran Interessiert, in Zukunft einen Großteil des Datenverkehrs über TLS abzuwickeln.Durch diese Entwicklung würde die Anwendung von SSL-stripping  begünstigt werden. Die Lösung dafür könnte die „HTTP Strict Transport Security“ (HSTS) [12] sein. Dies ist ein Header des HTTP Protokolls, der dem Browser signalisiert, dass eine Seite in Zukunft nur noch über HTTPS abrufbar sein soll.

Dies würde allerdings bedeuten, dass ein Angriff über das SSL-stripping beim allerersten Aufruf einer Seite trotzdem möglich ist, was eine bisher nicht geschlossene Sicherheitslücke darstellt.

# 3. Literatur

**[1]** Georgia Weidman (2014): *Penetration Testing: A Hands-On Introduction To Hacking* - San Francisco, no starch press

**[2]** *Traffic Definition*
https://onlinemarketing.de/lexikon/definition-traffic (Stand 13.02.17)

**[3]** *Switch (Netzwerktechnik)*
http://www.itwissen.info/definition/lexikon/Switch-switch.html (Stand: 13.02.17)

**[4]** *Hub(Netzwerktechnik)*
http://www.itwissen.info/definition/lexikon/Hub-hub.html (Stand: 13.02.2017)

**[5]** Combs, Gerald: *Wireshark – Go Deep*
https://www.wireshark.org/ (Stand: 13.02.17)

**[6]** Uhlmann, Stefan: *Arp Cache Poisioning / Arpspoof*
https://su2.info/doc/arpspoof.php (Stand 13.02.17)

**[7]** Aharoni, Kearns, Hertzog: *Kali Linux - Penetration Testing*
https://www.kali.org/ (Stand 13.02.2017)

**[8]** *TLS – Transport Layer Security*
http://www.itwissen.info/definition/lexikon/transport-layer-security-TLS-TLS-Protokoll.html (Stand 13.02.17)

**[9]** Ornaghi, Alberto: *Ettercap*
https://ettercap.github.io/ettercap/ (Stand 13.02.17)

**[10]** *Die Zukunft von HTTPS*
http://www.golem.de/news/hsts-google-fuehrt-liste-von-reinen-https-seiten-1408-108531.html(Stand 13.02.17)

**[11]** *Penetration Testing - Nostarch*
https://www.nostarch.com/pentesting/ (Stand 13.02.17)

**[12]** *HSTS – HTTP Strict Transport Security*
http://www.elektronik-kompendium.de/sites/net/1902051.htm (Stand 13.02.17)