Technische Hochschule Ingolstadt

Dokumentation

Security Workbench

angefertigt von

Name: Sebastian Schuster, Julian Rieder

Betreuer: Ernst-H. Göldner

Technische Hochschule Ingolstadt: TBD

Ingolstadt, 5. Januar 2016

Inhaltsverzeichnis

1 Einleitung

Mit diesem Dokument werden die Ergebnisse des Netzwerkteams im Masterprojekt (Informatik mit Schwerpunkt Safety & Security) “Security Workbench“ vom Wintersemester 2015/2016 an der Technischen Hochschule Ingolstadt vorgestellt.

2 Anforderungen

Titel: Entwurf und erste Implementierungen einer Security Work Bench an der thi
Betreuer: Prof. Dr. Ernst Göldner
Beschreibung: Ziel dieses Studentenprojekts im WS 15/16 ist die Planung und Erstellung der ersten Teile einer Se- curity Work Bench. Dies soll eine Umgebung werden, in der Praktika bzw. Übungen zu den Security Vorlesungen durchgeführt werden können. Die Security Work Bench könnte auch als Umgebung für weiterführende Bachelor-/ Master-Arbeiten in diesem Gebiet eingesetzt werden.
Für das Projekt im WS 15/16 ist angedacht:

- Entwicklung eines Grobkonzepts für die Security Work Bench (langfristiges Konzept).
- Auswahl der Implementierungen für dieses Semester, Organisation und Projektplanung dafür.
- Umsetzung für die ausgewählten Module (Entwurf, Aufbau und Erprobung, Dokumentation, Erstellen der Versuchsanleitungen)
- Erste Vorschläge für das Projekt im WS 15/16 (abhängig von der Anzahl der Teilnehmer):
  - Entwicklung eines Wettbewerbs zur Computer-/ Netzsicherheit („Capture the Flag“) ähnlich dem iCTF: Die Teilnehmer bekommen zu Beginn des Wettbewerbs Server zugewiesen. Auf diesen sind Programme installiert welche am Laufen gehalten werden müssen. Zunächst gilt es Server und Programme zu analysieren, Schwachstellen zu erkennen und Sicherheitslücken zu schließen. Gleichzeitig sollen genau diese erkannten Schwachstellen dafür ausgenutzt werden, um die Gegner zu attackieren. Für all dies gibt es Punkte. Außerdem erhält man Punkte, wenn eigene Programme trotz Attacken am Laufen gehalten werden.
  - Angriffe auf moderne Netzwerke: Im Rechnernetze-Labor sind zahlreiche Router, Ethernet Switches, Server und auch eine Firewall vorhanden. Mit diesen Geräten sollen Übungen zur Demonstration klassischer Angriffsszenarien entwickelt werden. Im zweiten Schritt sollen dann auch Maßnahmen zum Schutz untersucht und ggfs. implementiert werden.
  - Klassische Schwachstellen in den Betriebssystemen: Entwicklung von Übungen, die bekannte Schwachstellen demonstrieren und deren Behebung
  - Sicherheit von Passwörtern: Entwicklung von Übungen, die bekannte Schwachstellen demonstrieren und deren Behebung

Ziele:

- mit Abschluss dieses Projekts soll eine umfassender Plan für eine Security Work Bench an der thi erarbeitet sein.
- Erste Übungen, die begleitend zu den einschlägigen Vorlesungen durchgeführt werden können, sind entworfen, erprobt und mit einer adäquaten Dokumentation vorhanden, so dass im nächsten Semester diese Übungen durchgeführt werden können.
- Optional können noch weitere Übungen skizziert werden, die eine sinnvolle Ergänzung bzw. Weiterentwicklung dieses Projektes sind und deren Realisierung den Rahmen dieses Semesters übersteigt.

3 Zusammenfassung

Im Projekt *Security-Workbench* vom Wintersemester 2015/2016 wurden durch Sebastian Schuster und Julian Rieder verschiedene (Angriffs-)szenarien auf ISO/OSI-Layer 1-4 entwickelt.

Als Ergebnis können folgende Angriffstechniken vollautomatisiert vorgeführt werden:

- ARP-Spoofing
  - Mitlesen von Datenpakete
  - Manipulation von Inhalten aus Datenpakete
- DNS-Spoofing
- SSL-Strip
- Fake-IPv6-Netz
- Denial of Service

Zur leichteren und schnelleren Demonstration der Angriffstechniken wurde in Python eine Applikation entwickelt, welche alle Szenarios automatisiert ausführt und der Benutzer lediglich wenige notwendige Parameter (z.B. Ziel-IP-Adresse, Domains, Gateway) eingeben muss. Im Hintergrund werden dann alle erforderlichen Programme und Skripte mit der richtigen Konfiguration gestartet.

Um die Wartbarkeit dieses Tools zu erhöhen, wurde eine abstrakte Basisklasse definiert, welche die beiden Methodenrümpfe start() und help() enthält. Alle Angriffsszenarien wurden außerdem in eigene (abgeleitete) Klassen gekapselt.

Damit zukünftige Studenten dieses Projekt weiterentwickeln können, wurde großer Wert auf die Dokumentation gelegt. Alle Angriffsszenarien sind nach diesem Schema aufgebaut:

- Voraussetzungen
- Grundlagen
- Szenario
- Technisches
- Erklärung von erforderlichen Tools
- Benutzung des Python-Skripts
- Gegenmaßnahmen

4 Szenarios

4.1 ARP-Spoofing

ToDo: Julian

4.2 DNS-Spoofing

Vorraussetzungen

- Kali Linux 2.0
- ARP-Spoofing
- dnsspoofing

Grundlagen

DNS

Die Addressierung und der anschließende Verbindungsaufbau zu einem Server erfolgt über eine eindeutige IP-Adresse. Damit der Mensch leichter eine Verbindung zu einem Server aufbauen kann, wurde das DNS (Domain Name System) eingeführt. Dieses verwendet so genannte Domains zur Idenitifzierung von Servern, beispielweise "www.thi.de", da sich diese leichter merken lassen, als eine IP-Adresse (z.B. 194.94.240.179). DNS ähnelt damit der Funktionsweise eines Telefonbuchs. Das Domain Name System ist baumförmig aufgebaut, wie nachfolgende Abbildung 4.1 illustriert:

Szenario

Ein Client (z.B. Windows-Rechner) möchte die Internetseite der Technischen Hochschule Ingolstadt (www.thi.de) aufrufen. Dazu stellt dieser einen DNS-Request an seinen lokalen DNS-Server. Wenn dieser in seinem Cache keinen Eintrag findet, frägt er - beginnend am Root-DNS-Server - iterativ alle Nameserver nach ihren Einträgen ab, um zum Schluss die IP-Adresse von www.thi.de zu erhalten.

Technisches

Um einen DNS-Eintrag für eine Domain, beispielsweise www.thi.de, zu manipulieren, kann mittels DNS Cache Poisoning der lokale DNS-Cache des Clients mit falschen Einträgen

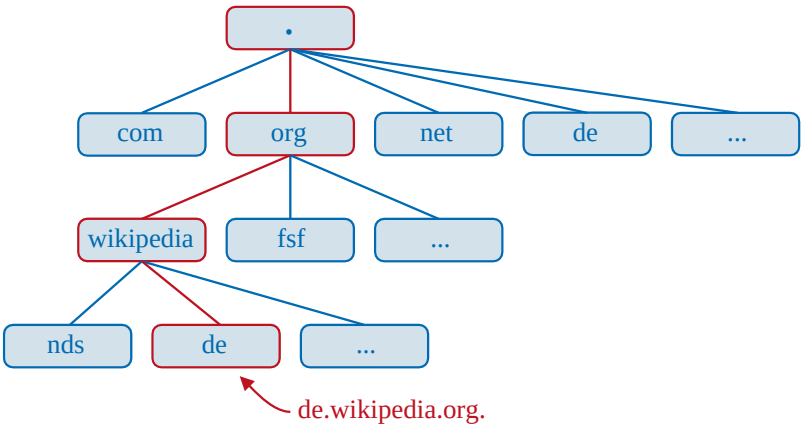

Abbildung 4.1: Aufbau DNS [3]

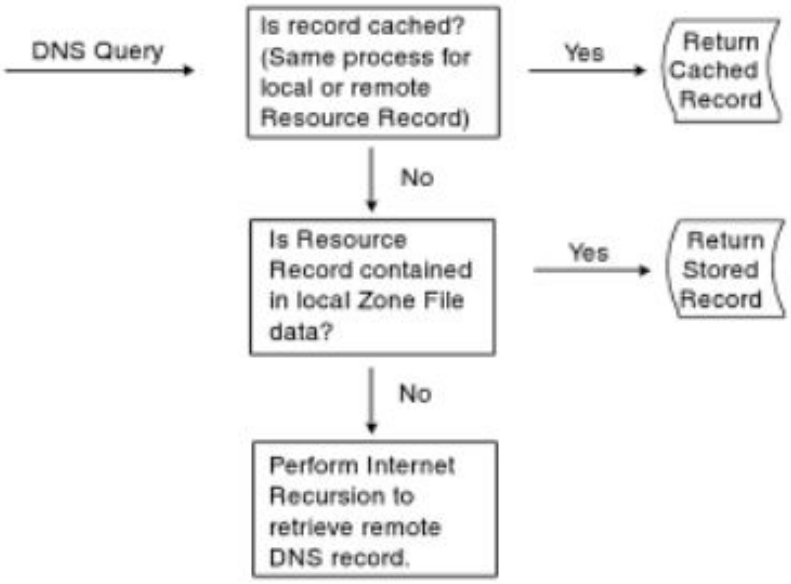

Abbildung 4.2: Ablauf DNS-Anfrage [5]

”vergiftet”werden. Da bei jeder DNS-Anfrage eine zufällig generierte Transaktions-ID mitgeschickt wird, und eine DNS-Antwort nur akzeptiert wird, wenn diese mit der Anfrage übereinstimmt, muss man als Angreifer diese ermitteln, was sich in einem lokalen Netzwerk mit einem Sniffer sehr einfach realisieren lässt. Alternativ kann auch die Transaktions-ID erraten werden, wofür für die 16-Bit lange Transaktions-ID im Durchschnitt 32.768 Versuche notwendig sind.

Tools

DNSSpoofing wurde von Dug Song [1] entwickelt und veröffentlicht. Mit Unterstützung dieses Tools ist ein Manipulation des DNS-Caches eines Clients im lokalen Netzwerk sehr leicht durchzuführen. Das Tool ermittelt die verwendeten Transaktions-ID durch Sniffen der ID, wenn der DNS-Server versucht eine Antwort an den Client zu übermitteln. Sobald er die ID der Anfrage ermittelt hat, muss er eine schnellere Antwort an den anfragenden Client versenden, als der eigentliche DNS-Server. Dies geschieht in mehrfachen Tests und Analysen durch Wireshark regelmäßig.

Benutzung von DNS-Spoofing-Skript

Um dnsspoof einsetzen zu können, muss initial eine hosts-Datei erstellt werden, die die zu manipulierenden Einträge in folgendem Format enthält:

```
<IP-Adresse>            <Domain>
<192.168.20.135>        www.thi.de
(Wichtig ist hierbei die Trennung von IP-Adresse und Domainname durch Tab und keinen Leerzeichen!)
```

Listing 4.1: Beispiel für eine Hosts-Datei

Anschließend wird *dnsspoof* mit folgenden Parameter aufgerufen:

```
-i Interface in dem sich lokales Netzwerk befindet
-f Hosts-File, absoluter Pfad zu Ort der erstellten hosts-Datei
```

Listing 4.2: Parameter für dnsspoof

Gegenmaßnahmen

DNSSEC

Durch DNSSEC kann die Authenzität einer DNS-Antwort verifiziert werden und somit DNS Cache Poisoning vorgebeugt werden. Durch eine asymmetrische Signatur - ähnlich PGP - kann der Absender der DNS-Antwort, also der DNS-Server, seine Antworten signieren, indem er mit dem nur ihm zugänglichen privaten Schlüssel den Record unterschreibt. Die Clientseite kann anschließend im Gegenzug die Antwort mit dem öffentlichen Schlüssel des DNS-Servers überprüfen, ob die Antwort auch von dem richtigen Server war.

[1] Diese und weitere Tools von Dug Song sind unter `www.monkey.org/~dugsong/dsniff` erhältlich.

4.3 Denial of Service (DoS)

Vorraussetzungen

- Kali Linux 2.0
- Python mit Socket- und Thread-Bibliothek

Grundlagen

TCP

TCP (Transmission Control Panel) ist ein verbindungsorientiertes Protokoll zur verlustfreien Übertragung von Daten und Datenströmen. Verschiedene Mechanismen sorgen dafür, dass Datenpakete zuverlässig und verbindungsorientiert übertragen werden.

Szenario

DoS (Denial of Service, zu dt: Dienstblockade) bezeichnet die vorübergehende Nichtverfügbarkeit eines Dienstes, durch Überlastung. Wird die Überlastung von mehreren Systemen verursacht, spricht man von DDoS (Distributed Denial of Service).
Bei einem DoS-Angriff mittels SYN-Flooding wird das Übertragungsprotokoll TCP verwendet, da es zustandsorientiert ist, und somit der angesprochene Server Ressourcen für den Anfragenden reserviert. Das Aufrechterhalten der Ressourcen wird durch eine fehlende ACK-Bestätigung des Clients realisiert, nachdem der Server vorher ein SYN-ACK-Bestätigung übermittelt hat. Durch Versenden von sehr vielen SYN-Paketen auf den selben Zielserver kann es vorkommen, dass auf dem angegriffenen Server keine Ressourcen mehr vorhanden sind, um weitere Anfragen annehmen zu können. Die dann folgenden Pakete werden vom Server umgehend verworfen und es kann keine Verbindung aufgebaut werden. [1]

Technisches

Das selbst geschriebene Python-Skript versendet eine vorgegebene Anzahl von SYN-Paketen an eine Zieladresse. Durch einen Iptables-Eintrag wird verhindert, dass nach Erhalt der SYN-ACK-Bestätigung des Zielserveres eine ACK-Bestätigung zurückgeschickt wird. Dadurch wird für eine bestimmte Zeit Ressourcen reserviert, die in Summe zur Überlastung des Servers führen.

Tools

siehe *Technisches* in Kapitel 4.3

Benutzung von DoS-Skript

Das Skript frägt interaktiv den Benutzer alle erforderlichen Angaben ab. Diese sind die Anzahl der SYN-Pakete und die IP-Adresse des Zielservers.

Gegenmaßnahmen

Netzwerk Monitoring

Mittels eines Intrusion Detecten (IDS) und Prevention System (IPS) kann die Aktivität und der Ursprung eintreffender SYN-Pakete analysiert werden und beispielsweise nur eine bestimmte Anzahl von Paketen pro Minute zugelassen werden. Sollten von der Quell-IP-Adresse dann noch weitere Pakete eintreffen, werden diese bereits an der Firewall verworfen. [2]

SYN-Cookies

Mittels SYN-Cookies kann bei Verbindungsaufbau durch den Server überprüft werden, ob der Client bereits versucht hat, eine Verbindung herzustellen. Bei Implementierung von SYN-Cookies reserviert der Server keine Ressourcen bei Eintreffen eines SYN-Paketes von einem Client, sondern speichert nur einen Hashwert mit Informationen des SYN-ACK-Paketes. Wenn der Client im dritten Schritt ein SYN-Paket mit der Bestätigung des SYN-ACKs an den Server übermittelt hat, wird mittels des gespeicherten Hashwertes überprüft, ob dieser Client bereits vorher mit dem Server kommuniziert hat. Falls diese Überprüfunge positiv ausfällt, wird eine TCP-Verbindung aufgebaut.

[2] Mehr Informationen zu Umfang und Möglichkeiten von IDS und IPS finden Sie unter folgendem Paper:[2]

4.4 SSL-Strip

Vorraussetzungen

- Kali Linux 2.0
- IP Forward
- IPtables
- ARP-Spoofing
- SSLStrip

Grundlagen

HTTP

HTTP (Hypertext Transfer Protocol) ist ein zustandsloses Protokoll zur Übertragung von Dokumenten auf Anwendungsschicht (siehe ISO-OSI-Layer). Der Standard wurde 1991 von der Internet Egnineering Task Force (IETF) und dem World Wide Web Consortium (W3C) eingeführt und ist mittlerweile in Version 2.0 (HTTP/2) veröffentlicht. [1] Nachfolgendes Schema (Abbildung x) verdeutlicht den Ablauf.
Meist wird HTTP verwendet um HTML-Seiten in Webbrowsern darzustellen.

HTTPS

HTTPS (Hypertext Transfer Protocol Secure) wird dazu verwendet um Dokumente auf Anwendungsschicht über ein sicheres Protokoll übertragen zu können. Syntaktisch ist es wie HTTP aufgebaut, wird jedoch um eine Verschlüsselung der Daten umgeben. Zur Verschlüsselung der Daten wird SSL (Secure Socket Layer) bzw. TLS (Transport Layer Security) verwendet.

ARP

siehe Eintrag Address-Resolution-Protocol

Szenario

Eine MITM-Attacke auf eine verschlüsselte HTTPS-Verbindung ist nur mit sehr viel Rechenkapazität zu entschlüsseln. Eine einfachere Möglichkeit des Mitschneiden von übertragenen Datenpaketen ist die Verwendung einer unverschlüsselten HTTP-Verbindung. Da ein Großteil der Benutzer einen Unterschied von *https://www.url.de* zu *http://www.url.de* in der URL-Leiste kaum erkennen würden, ist SSLStrip eine gute Möglichkeit Datenpakete mitlesen und verändern zu können.
Das Auslesen von Passwörtern für Online-Banking oder Webmail wären potentielle Ziele eines solchen Angriffs.

Technisches

SSLStrip [3] wurde von Moxie Marlinspike 2009 entwickelt und ist aktuell in Version 0.9.2 verfügbar. Das Tool durchsucht jeden transparenten HTTP-Verkehr nach https-Links und wandelt diese in http-Links um. Um die Attacke durchführen zu können, wird zusätzlich ARP-Spoofing benötigt. Mittels ARP-Spoofing werden auch die unverschlüsselten HTTP-Links über SSLStrip verschickt. Da mittlerweile viele Webseiten (z.B. Online-Banking, Webmail, ...) nur noch verschlüsselte HTTP(S)-Verbindungen zulassen, baut SSLStrip eine verschlüsselte Verbindung zu diesen Seiten auf, und gibt deren Antwort in einer unverschlüsselten Verbindung an den kompromittierten Client zurück. Folgende Abbildung zeigt den Ablauf der HTTP(S)-Verbindungen zwischen einem Client, Angreifer und dem aufgerufenen (Web-)Server.

[3] Dieses Tool kann über folgende Links abgerufen werden: https://github.com/graingert/sslstrip/, http://www.thoughtcrime.org/software/sslstrip/

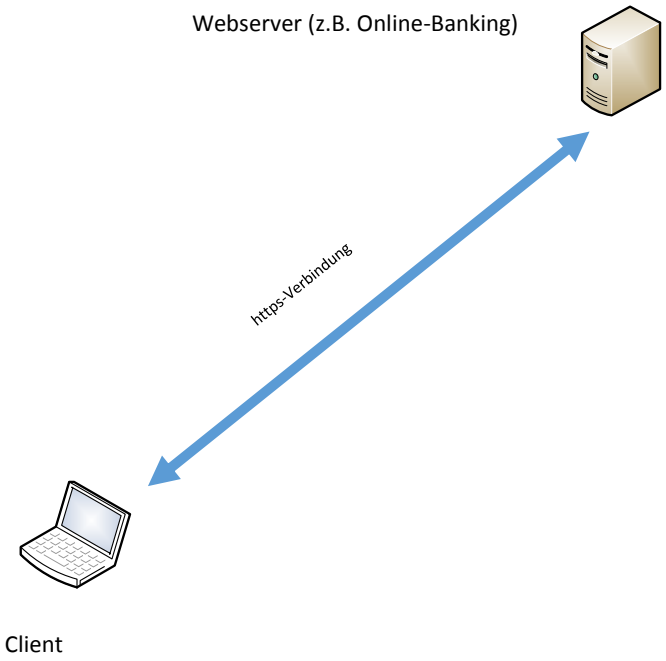

Abbildung 4.3: Reguläre HTTPS-Verbindung zwischen Client und Server

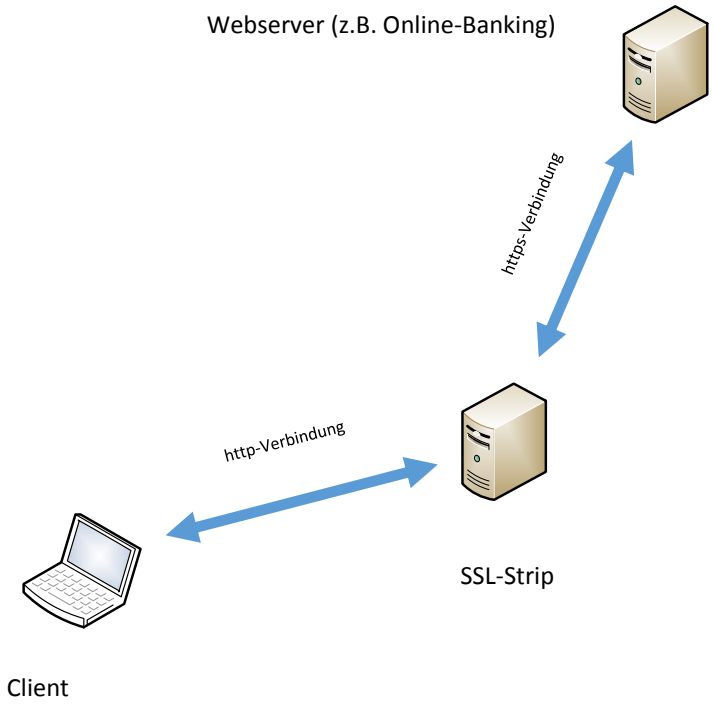

Abbildung 4.4: SSL-Strip Szenario

Tools

Um SSLStrip einsetzen zu können sind mehrere Schritte notwendig. Auf Kali Linux 2.0 sind alle benötigten Tools bereits vorinstalliert.

IP-Forwarding, also das Weiterleiten von IP-Paketen, kann durch folgende Befehle aktiviert werden:

```
sysctl -w net.ipv4.ip_forward=1
alternativ: echo 1 > /proc/sys/net/ipv4/ip_forward
```

Listing 4.3: Aktivieren von IP-Forwarding

Anschließend wird ARP-Spoofing gestartet. Dies geschieht mit folgenden Befehlen:

```
arpspoof -i <interface> -t <targetIP> <gatewayIP>
Parameter:
        -i  <interface>         Angabe des Interfaces, in dem sich Angreifer und Client befinden.
        -t <targetIP>           IP-Adresse des anzugreifenden Clients
        <gatewayIP>             IP-Adresse des Gateways im LAN
```

Listing 4.4: Parameter für ARP-Spoofing

Nachdem nun mittels ARP-Spoofing alle IP-Pakete vom angegriffenen Client über den Angreifer gesendet werden, müssen die umgeleiteten HTTP-Pakete via IPtables an das Tool SSLStrip weitergereicht werden. Dies geschieht mittels folgendem Eintrag:

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port <listenPort>
Parameter:
        -t nat                  : Firewall-Gruppe
        -A PREROUTING           : Regel wird angewandt, BEVOR Paket geroutet wird
        -p tcp                  : Nur TCP-Pakete
        --destination-port 80   : Nur Pakete auf Port 80 (http)
        -j REDIRECT             : Legt Aktion fest, also Weiterleitung
        --to-port <listenPort>  : Port auf dem SSLStrip lauscht.
```

Listing 4.5: Eintrag in IP-Tables damit HTTP-Pakete an sslstrip weitergereicht werden

Nun muss noch SSLStrip selbst gestartet werden. Dies geschieht mittels folgender Eingabe:

```
sslstrip -a -k -l <listenPort> -w <logpath>
Parameter:
        -s : Gesamter SSL Traffic wird gelogged
        -p : Nur SSL POST Traffic wird protokolliert
        -a : SSL- und HTTP-Traffic wird aufgezeichnet
        -k : Bestehende SSL-Verbindungen terminieren, damit diese neu aufgebaut werden
        -l : Port auf dem SSLStrip lauscht. Muss identisch zu --to-port bei iptables-Eintrag sein
        -w : Pfad in dem gehijackter HTTPS-Traffic im Klartext abgespeichert wird
```

Listing 4.6: Erforderliche Parameter für SSLStrip

Benutzung von SSLStrip-Skript

Zur Automatisierung wurden vorangegangene Befehle in einem Skript automatisiert. Nachdem SSLStrip im Auswahlmenü selektiert wurde, wird zuerst nach der Netzwerkschnittstelle gefragt, in der Angreifer und Zielclient sich befinden. Anschließend wird das ausgewählte Netzwerk nach aktiven Hosts gescannt und aufgelistet. Im folgenden Schritt wird die Ziel-IP-Adresse des anzugreifenden Clients eingegeben, gefolgt von der IP-Adresse des Gateways für ARP-Spoofing. Abschließend werden die erforderlichen Konfigurationen für SSLStrip-Attacke im Hintergrund durchgeführt und der mitgeschnittene HTTPS-Verkehr im Klartext in der LOG-Datei abgerufen werden.

Gegenmaßnahmen

HTTP Strict Transport Security

HTTP Strict Transport Security ist ein Mechanismus um einem Client mitzuteilen, dass er für eine bestimmte Zeit nur verschlüsselte Verbindungen verwenden soll. Der Server übermittelt in seiner Antwort im Header, zusätzliche Informationen über die Gültigkeit der Information und ob sämtliche Subdomains ebenfalls ausschließlich verschlüsselte Verbindungen annehmen dürfen. [4]

4.5 Fake IPv6 Netz

ToDo -¿ Julian

5 Ausblick

...

Literatur

[1] Ron Aitchison. „DNSSEC“. English. In: *Pro DNS and BIND 10*. Apress, 2011, S. 317–377. ISBN: 978-1-4302-3048-9. DOI: 10.1007/978-1-4302-3049-6_11.

[2] AsmaaShaker Ashoor und Sharad Gore. „Difference between Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)“. English. In: *Advances in Network Security and Applications*. Hrsg. von DavidC. Wyld u. a. Bd. 196. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2011, S. 497–501. ISBN: 978-3-642-22539-0. DOI: 10.1007/978-3-642-22540-6_48. URL: http://dx.doi.org/10.1007/978-3-642-22540-6_48.

[3] *Meterpreter*. URL: https://commons.wikimedia.org/wiki/File:Dns-raum.svg#/media/File:Dns-raum.svg (besucht am 04. 01. 2015).

[4] Philippe De Ryck u. a. English. In: *Primer on Client-Side Web Security*. SpringerBriefs in Computer Science. Springer International Publishing, 2014, S. 43–55. ISBN: 978-3-319-12225-0.

[5] S. Young und D. Aitel. *The Hacker's Handbook: The Strategy Behind Breaking into and Defending Networks*. CRC Press, 2003. ISBN: 9780203490044. URL: https://books.google.de/books?id=AO2fsAPVC34C.