Man in the Middle Attack

Version 2.0

Lukas Müller

03. Mai 2010

Allgemeines

Der Man-in-the-middle-Angriff ist ein beliebtes Angriffsszenario, wobei der Angreifer logisch zwischen den beiden Kommunikationspartnern steht und dabei die vollständige Kontrolle über den Datenverkehr zwischen zwei oder mehreren Netzwerkteilnehmern hat. Dadurch können sämtliche Informationen mittels sogenannten Sniffern, wie zum Beispiel Wireshark[1], ausgelesen und sogar manipuliert werden. Wichtig dabei ist, dass keines der beiden Opfer etwas vom Angriff mitbekommt und dass man dem jeweiligen Kommunikationspartner vortäuscht, der jeweils andere zu sein. (vgl.[WIKI2009a], vgl.[LEWI2008] S. 403)

Solch ein Angriff kann sehr einfach im LAN bzw. WLAN durchgeführt werden. Wenn der Angreifer beispielsweise physikalischen Zugriff auf die Datenleitungen hat. Andernfalls kann ARP-Spoofing[2] eingesetzt werden, um den Datenverkehr über den Angreifer laufen zu lassen. Eine weitere Angriffsmethode dieser Art ist das Manipulieren des DHCP-Servers[3] oder wenn der Angreifer selbst den DHCP-Server spielt, sodass die Hosts ein falsches Default-Gateway erhalten.

Weitaus schwieriger ist es, einen Man-in-the-middle-Angriff im WAN durchzuführen, da wir nicht die Routing-Entscheidungen beeinflussen können, um so den Datenverkehr auf die IP-Adresse des Angreifers leiten zu können. Außer der Angreifer hat Kontrolle über einen Router, durch den der Datenverkehr geschleust wird. Da jedoch kein Benutzer direkt IP-Adressen anspricht, sondern mittels Namen bzw. URL arbeitet, benötigt man einen DNS-Server, um eine URL auflösen zu können. Daher kann man mittels DNS Cache Poisoning[4] die DNS-Einträge verändern, sodass die URL mit der Zieladresse des Angreifers aufgelöst wird. Da jedoch zuerst die lokale Host-Datei auf dem Rechner abgefragt wird, um eine URL aufzulösen, kann man auch die Einträge in dieser Datei manipulieren. Dadurch kann trotz Eingabe der echten URL, die gefälschte IP-Adresse des Angreifers aufgelöst werden. (vgl.[WIKI2009a])

[1] http://www.wireshark.org [Okt2009]
[2] siehe 3.3 ARP Spoofing
[3] siehe 3.4 DHCP Spoofing
[4] siehe 3.11 DNS Cache Poisoning

Wenn ein Angreifer durch einen mitm (Man-in-the-middle)-Angriff unverschlüsselten Datenverkehr (HTTP, FTP, SMTP, TELNET usw.) mitsnifft, kann er jede Information im Klartext lesen und dazu zählen auch Benutzernamen und Passwörter. Abhilfe schafft dagegen das Verschlüsseln[5] der Daten, indem man den Einsatz von Protokollen wie SSL/TLS, SSH usw. forciert. Jedoch können auch solche Protokolle dem mitm-Angriff zum Opfer fallen. Zum leichteren Verständnis wird der Angriff anhand eines Beispiels erklärt. Der Angreifer H möchte die Kommunikation zwischen den beiden Kommunikationspartnern A und B abhören. Der erste Schritt ist, dass H den öffentlichen Schlüssel von B abfängt und an A seinen eigenen öffentlichen Schlüssel schickt. A denkt nun in Besitz des öffentlichen Schlüssels von B zu sein und beginnt die Nachricht zu verschlüsseln und an B zu senden. Der Angreifer E fängt diese Nachricht ab und ist nun in der Lage die Nachricht, welche mit seinem öffentlichen Schlüssel verschlüsselt wurde, zu entschlüsseln. Als Nächstes verschlüsselt E die Nachricht mit dem öffentlichen Schlüssel von B, den er zu Beginn abgefangen und mit seinem Schlüssel ausgetauscht hat. Dann schickt E die verschlüsselte Nachricht an B. Der Kommunikationspartner B entschlüsselt diese Nachricht mit seinem privaten Schlüssel und weder A noch B bemerken, dass E die Informationen mitgelesen oder sogar manipuliert hat. Wichtig hierbei ist, dass die verschlüsselte Nachricht von A den Empfänger B nicht erreichen darf, da dieser sonst Verdacht schöpfen könnte, wenn er die Nachricht mit seinem privaten Schlüssel nicht entschlüsseln kann, da B die Nachricht mit dem öffentlichen Schlüssel des Angreifers verschlüsselt hat. (vgl.[SDO2008])

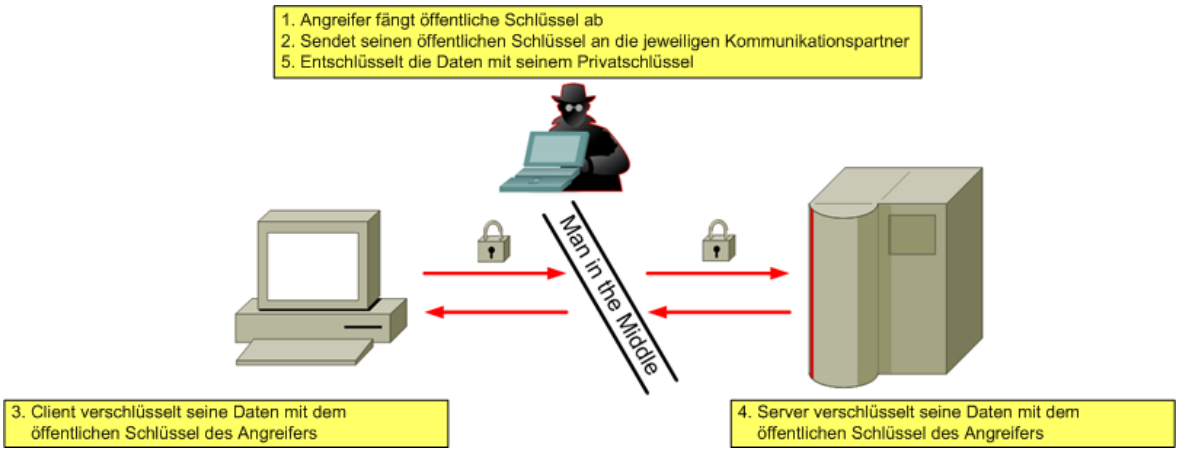

Abbildung 0.1: Man in the middle Attack

Um sicher gehen zu können, dass der Kommunikationspartner gegenüber wirklich der Partner ist, mit dem man kommunizieren will, also um die Authentizität zu gewährleisten, verwendet man digitale Signaturen bzw. Zertifikate. Diese Zertifikate werden von einer Certification Authority (CA), welche eine meist staatlich beglaubigte Zertifizierungsstelle ist, technisch bereitgestellt. Für ein effizientes Schlüsselmanagementsystem wurde das PKI (Public Key Infrastructur) System konzipiert und dies wurde im ITU-T-Standard X.509 realisiert. (vgl.[SDO2008])

[5]siehe 2.8 Kyoptologie

Man-in-the-middle-Angriffe auf SSL/TLS

Null Prefix Attack

Die heutige Version des X.509 Zertifikats ist Version 3 (X.509v3) und dieses Zertifikat identifiziert beispielsweise eindeutig einen Server bei einer SSL/TLS Kommunikation. Genauergesagt ist bei allen SSL/TLS Implementation der Common Name essentiell, denn anhand dieses Feldes wird ein Server identifiziert. Beispielsweise würde im Falle von PayPall im Feld „common name“ www.paypall.com stehen. Um es der Certification Authority zu erleichtern, prüft die nur den Besitzer solch einer Domain mittels einer WHOIS-Abfrage und überprüft nur die Root Domain (also in diesem Beispiel paypall.com), wobei Subdomains in den meisten Fällen ignoriert werden. Nun muss man unterscheiden zwischen Pascal Strings und C Strings. Denn bei einem Pascal String wird in den ersten Bytes die Länge des Strings angegeben, wobei bei einem C String der Wert NULL den String beendet.

Pascal String:

0x04 (Länge)	0x44 ('D')	0x41 ('A')	0x54 ('T')	0x41 ('A')

Tabelle 0.1: Pascal String

C String:

0x44 ('D')	0x41 ('A')	0x54 ('T')	0x41 ('A')	0x00 (NULL)

Tabelle 0.2: C String

Wenn ich nun Beispielsweise www.paypal.com\0.hsm-pro.at in das „common name“ Feld eintrage, ignoriert die Certification Authority weiterhin alle Subdomains und würde nur abfragen, ob hsm-pro.at wirklich in meinem Besitz ist. Da dies der Fall ist, wird mein X.509 Zertifikat beglaubigt und bestätigt, dass ich wirklich der gewünschte Kommunikationspartner bin. Viele SSL/TLS Implementationen jedoch lesen den Common Name als C String und würden daher www.paypal.com\0.hsm-pro.at nicht unterscheiden können zu www.paypal.com. Daher würde nun eine Verbindung mit www.paypal.com aufgebaut werden, die Certification Authority würde das Zertifikat für hsm-pro.at bestätigen und eine verschlüsselte Verbindung mit einem falschen Kommunikationspartner aufbauen, wo wir wieder beim Man-in-the-middle Angriff wären und sämtliche Informationen ausgelesen bzw. manipuliert werden können. Diesen Angriff kann man mit dem Programm sslsniff[6] problemlos durchführen und so zu essentiellen Informationen kommen. (vgl.[MOXIE2009a], vgl.[HEISE2009a], vgl.[HEISE2009b], vgl.[HEISE2009c], vgl.[NOIS2009])

Beispiel-sslsniff

In diesem Beispiel wird eine SSL Verbindung mittels sslsniff mit der oben beschriebenen Null-Prefix-Attack gehackt.

[6] http://www.thoughtcrime.org/software/sslsniff [Okt2009]

```
# Routing aktivieren
echo 1 > /proc/sys/net/ipv4/ip_forward

#NAT + Weiterleitung aller Pakete
iptables -t nat -A PREROUTING -p tcp -destination-port 80 -j
   REDIRECT -to-port 10000

# ARP-Spoofing mittels ettercap um Traffic umzuleiten
arpspoof -i wlan0 -t 192.168.0.1 192.168.0.254

#SSL Strip starten
sslsniff -t -p -s <$listenPort> -w <$logFile> -m zertifikat.
   crt \            -c <$certDir>

# Sniffen mittels ettercap bzw. Wireshark
ettercap -T -q -i wlan0
```

Listing 1: SSL/TLS Verbindung mittels Null-Prefix-Attack hacken

Links & Redirects

Da diese oben beschriebene Null-Prefix-Attack relativ komplex ist und einige SSL/TLS Implementationen daraufhin verbessert wurden, gibt es trivialere Lösungen, HTTPS Verbindungen mittels eines mitm-Angriffs zu belauschen. Viel einfacher ist es nämlich, nicht das eigentliche SSL/TLS-Protokoll zu knacken, sondern einfach HTTP-Verbindungen abzuhören. Der Trick dabei ist, dass kein Benutzer in den Browser https:// und dann die weitere URL eingibt, sondern einfach die Domain angibt, wie zum Beispiel www.psk.at. Nun ist die Startseite unverschlüsselt, da noch keine essentiellen Informationen übertragen wurden. Wenn man sich jedoch nun mit seinem Benutzernamen bzw. Verfüger und seinem Passwort bzw. Identifikationsnummer anmeldet, leitet uns der Link, nachdem wir den Button betätigt haben, zu einer HTTPS-Verbindung und somit ist die gesamte Kommunikation verschlüsselt. Wenn man nun jedoch, als man-in-the-middle, sämtliche Links dieser Seite von https:// auf http:// umändert, so werden die Benutzerdaten im Klartext an den Angreifer gesendet. Der wiederum muss sich sämtliche HTTPS-Verbindungen merken, sodass er sie wieder korrekt an den Server weiterleiten kann, um die Kommunikation nicht zu unterbrechen. Eine zweite Möglichkeit, wie ein Benutzer eine verschlüsselte Verbindung mittels SSL/TLS zu einem Server aufbaut, ist mittels Weiterleitung, im speziellen HTTP 302 redirect. Dadurch wird die eingegebene URL des Benutzers von einer HTTP-Verbindung zu einer HTTPS-Verbindung weitergeleitet. Jedoch funktioniert der mitm-Angriff in diesem Fall genauso wie oben beschrieben. Somit denkt der Kommunikationspartner A eine gesicherte Verbindung zum Server (Kommunikationspartner B) aufgebaut zu haben, jedoch sendet A alle Informationen im Klartext an den Angreifer. Die Schwierigkeit ist nur, sämtliche Sicherheitsvorkehrungen des Browsers, um den Benutzer darauf hin zu weisen, dass die Verbindung verschlüsselt ist, nachzubilden. Das Programm sslstrip[7] hat dafür auch

[7] http://www.thoughtcrime.org/software/sslstrip [Okt2009]

wunderbare Funktionen wie zum Beispiel das Einbinden eines Sicherheitsschlosses im Browser, um eine verschlüsselte Verbindung anzuzeigen, obwohl die Kommunikation nur über HTTP läuft. (vgl.[MOXIE2009a])

Beispiel-sslstrip

In diesem Beispiel wird eine SSL-Verbindung mittels sslstrip mit dem oben beschriebenen Man-in-the-middle-Angriff gehackt.

```
# Routing aktivieren
echo 1 > /proc/sys/net/ipv4/ip_forward

#NAT + Weiterleitung aller Pakete
iptables -t nat -A PREROUTING -p tcp -destination-port 80 -j
   REDIRECT -to-port 10000

# ARP-Spoofing mittels ettercap um Traffic umzuleiten
arpspoof -i wlan0 -t 192.168.0.1 192.168.0.254

#SSL Strip starten
sslstrip -p -f

# Sniffen mittels ettercap bzw. Wireshark
ettercap -T -q -i wlan0
```

Listing 2: SSL/TLS Verbindungen mittels sslstrip hacken

Zunächst muss der Angreifer volle Kontrolle über den Datenverkehr erhalten, indem sämtlicher Traffic über ihn läuft. Dies gelingt ihm mit den angesprochenen Methoden wie beispielsweise DNS-Cache-Poisioning[8], Route Poisoning oder wie in diesem Fall mittels ARP-Spoofing[9]. Damit der Angreifer als Proxy fungieren kann, muss das Routing aktiviert werden, sodass er sämtlichen vom Client erhaltenen Traffic zum eigentlichen Server weiterleitet. Damit der Angreifer nur den gewünschten Traffic, also in unserem Fall nur normale HTTP-Verbindungen, weiterleitet, muss man festlegen, auf welchem Port gelauscht wird und auf welchem Port die Weiterleitung erfolgt. SSL Strip achtet auf HTTPS-Links und Redirects und legt eine Tabelle mit den jeweiligen HTTPS-Verbindungen und den jeweils dazu extra aufgebauten HTTP-Verbindungen an. Zusätzlich wird ein Favicon mit einem Schloss in den Browser eingebunden, sodass der Client keinen Unterschied zu einer verschlüsselten Verbindung merkt. Daraus resultiert eine hohe Anzahl an gelungenen Angriffen.

Gegenmaßnahmen

Als Benutzer kann man sich gegen solch einen Angriff nur wehren, indem man in den Browser direkt das Protokoll https:// und dann die weitere URL eingibt, oder wenn man sich beispielsweise https://www.psk.at als Lesezeichen in seinem Browser

[8] siehe 3.11 DNS Chache Poisoning
[9] siehe 3.3 ARP Spoofing

einspeichert. Zusätzlich muss man genau auf die Sicherheitswarnungen des Browsers achten, wie zum Beispiel das Sicherheitsschloss.

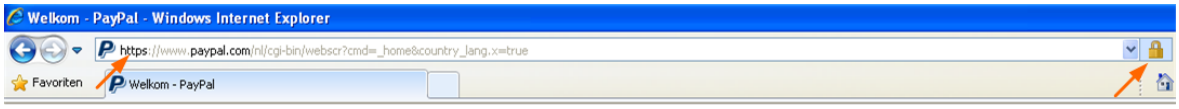

Abbildung 0.2: HTTPS-Verbindung im Internet Explorer

Die roten Pfeile zeigen an, wie der Browser gesicherte HTTPS-Verbindungen anzeigt, um zu signalisieren, dass diese Verbindung zum Server verschlüsselt und gesichert ist. Man kann sich aber auch direkt das Zertifikat anzeigen lassen, wie in Abbildung 3.13: SSL Zertifikat zu sehen, um zu überprüfen, wer das Zertifikat ausgestellt hat und auf welchen Namen das Zertifikat läuft. Zusätzliche Informationen wie das Ablaufdatum des Zertifikates sind auch aus der Abbildung zu entnehmen.

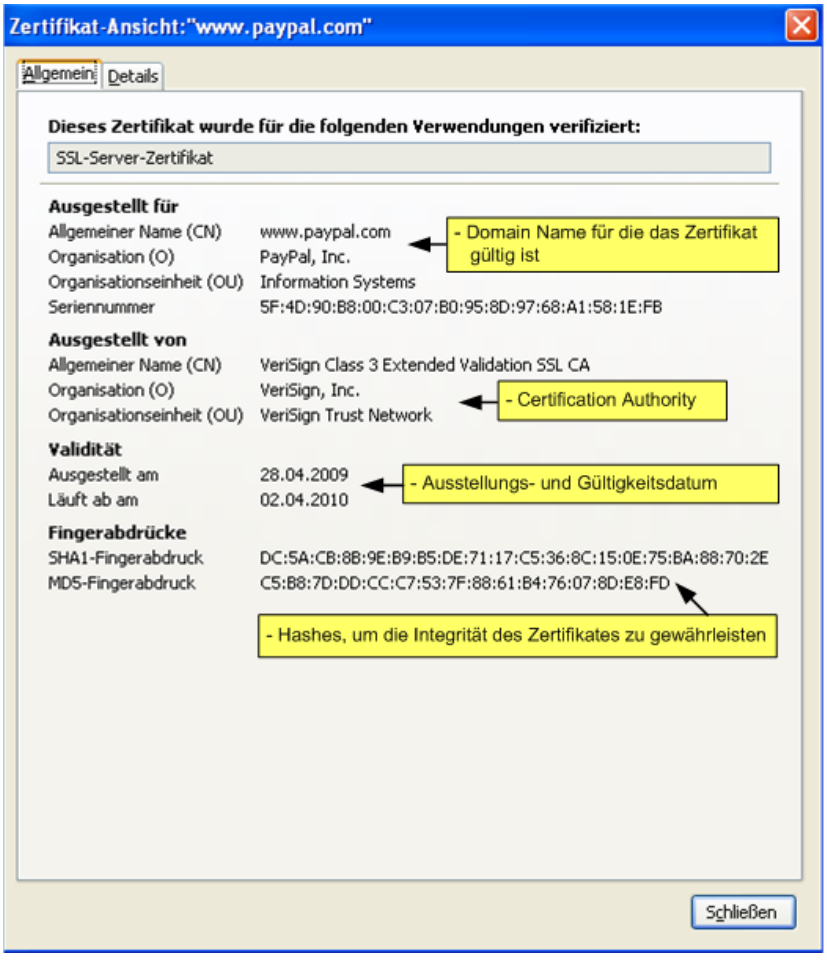

Abbildung 0.3: SSL Zertifikat

Wichtig ist auch, keine unbekannten, selbst ausgestellten Zertifikate zu akzeptieren, da diese nicht von einer Certification Authority ausgestellt wurden und daher nicht die Integrität bestätigt werden kann.

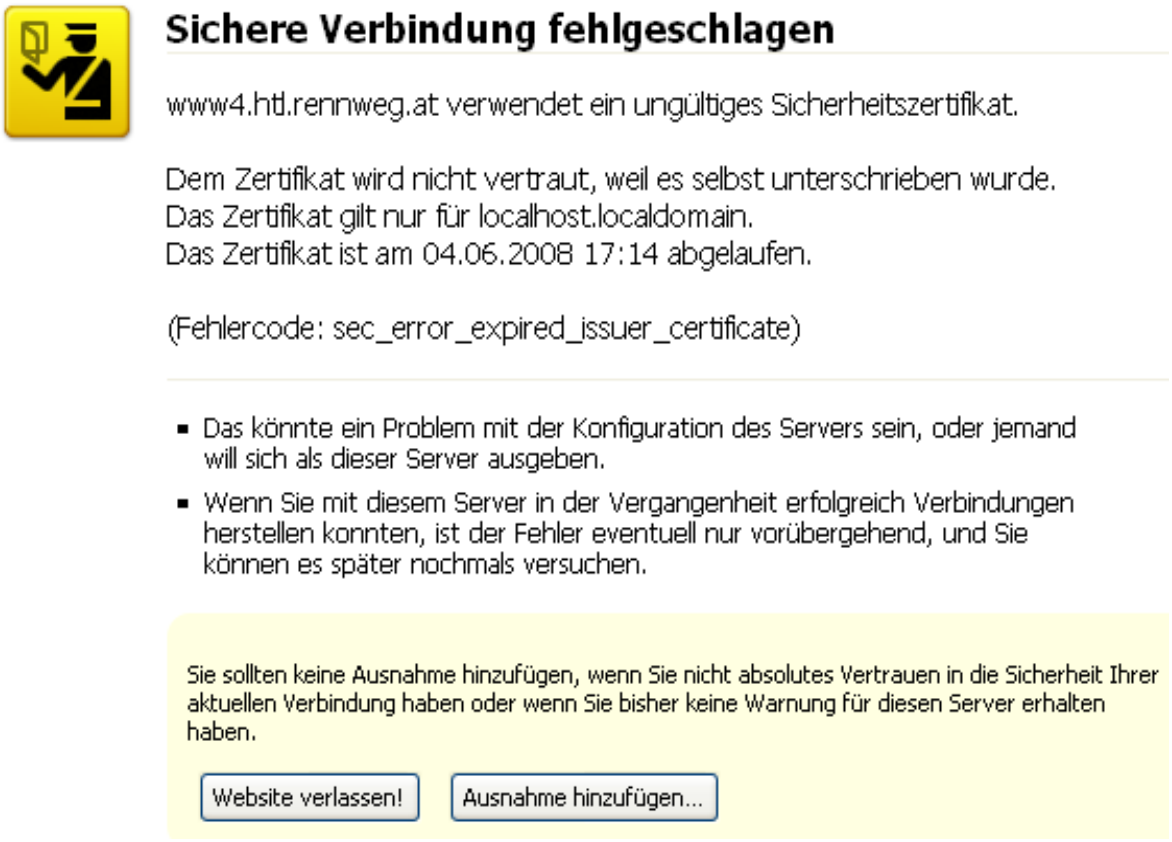

Abbildung 0.4: Warnung eines unbekannten Zertifikates

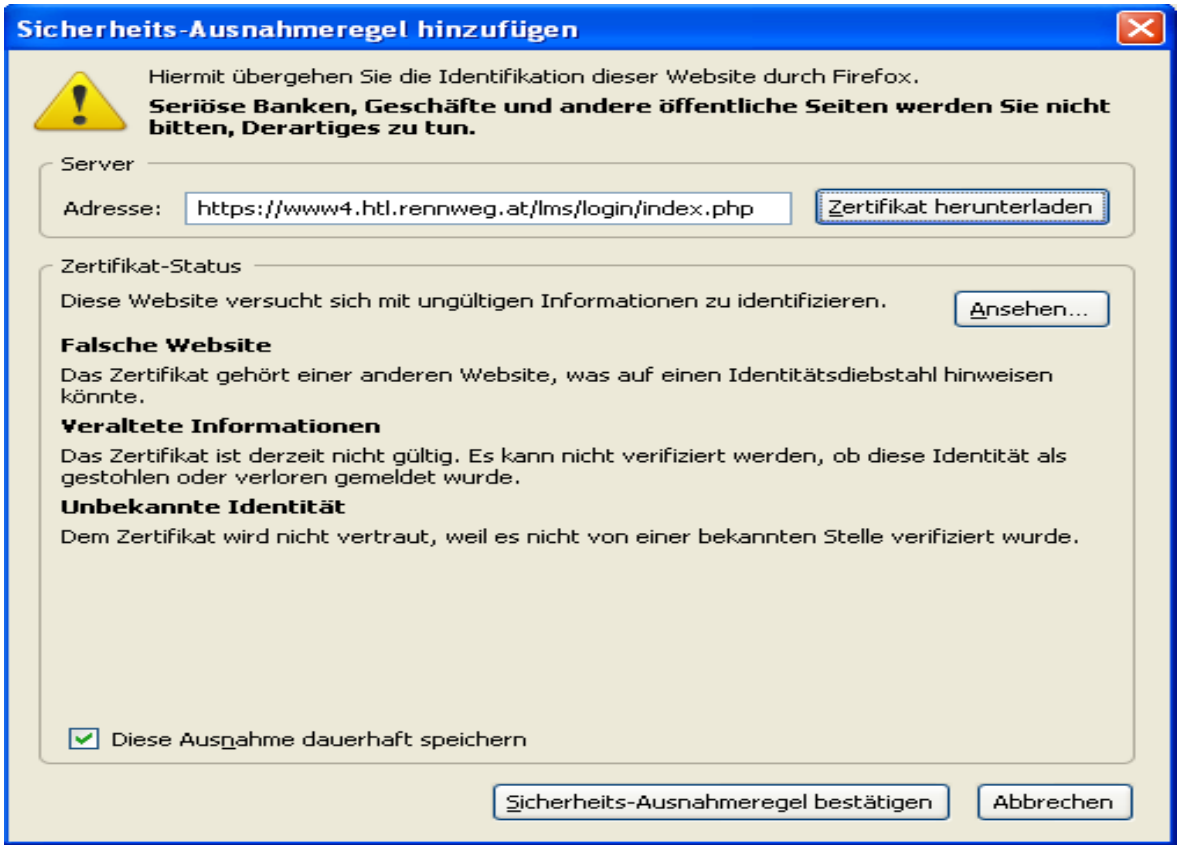

Abbildung 0.5: unbekanntes Zertifikat herunterladen

Authentication Gap

Durch Fehlimplementationen von SSL/TLS können mitm-Angriffe wie bei der Null-Prefix-Attack durchgeführt werden. Aber es gibt auch Designfehler im TLS-Protokoll (SSL 3.0+ & TLS 1.0+) und zwar bei der Neuaushandlung der Parameter einer schon bereits bestehenden HTTPS-Verbindung, dies ist auch als TLS Renegotiation bekannt. Beispielsweise nimmt ein Client eine gesicherte Verbindung mit einem Webserver auf. Der Angreifer hört den gesamten Datenverkehr ab und stellt selber eine neue HTTPS-Verbindung mit dem Server her. Die Verbindung zum Client wird währenddessen für kurze Zeit in einem unvollendeten Zustand gehalten. Als Nächstes sendet der Server einen HELLO-Request und möchte einen TLS-Handshare mit dem Angreifer durchführen, um sein Client-Zertifikat zu überprüfen. Nun leitet der Angreifer wieder den gesamten Traffic von Server zum Client weiter und die beiden tauschen ihre Zertifikate aus. Dadurch kann die gesicherte Verbindung vom Angreifer übernommen werden und wird als Authentication Gap bezeichnet. Dieses Problem tritt bei aktuellen Versionen des Apache Servers, des IIS Servers und auch bei OpenSSL auf. Die Überarbeitung dieses Designfehlers ist in Arbeit. (vgl.[EXTE2009], vgl.[LINK2009], vgl.[IETF2009])

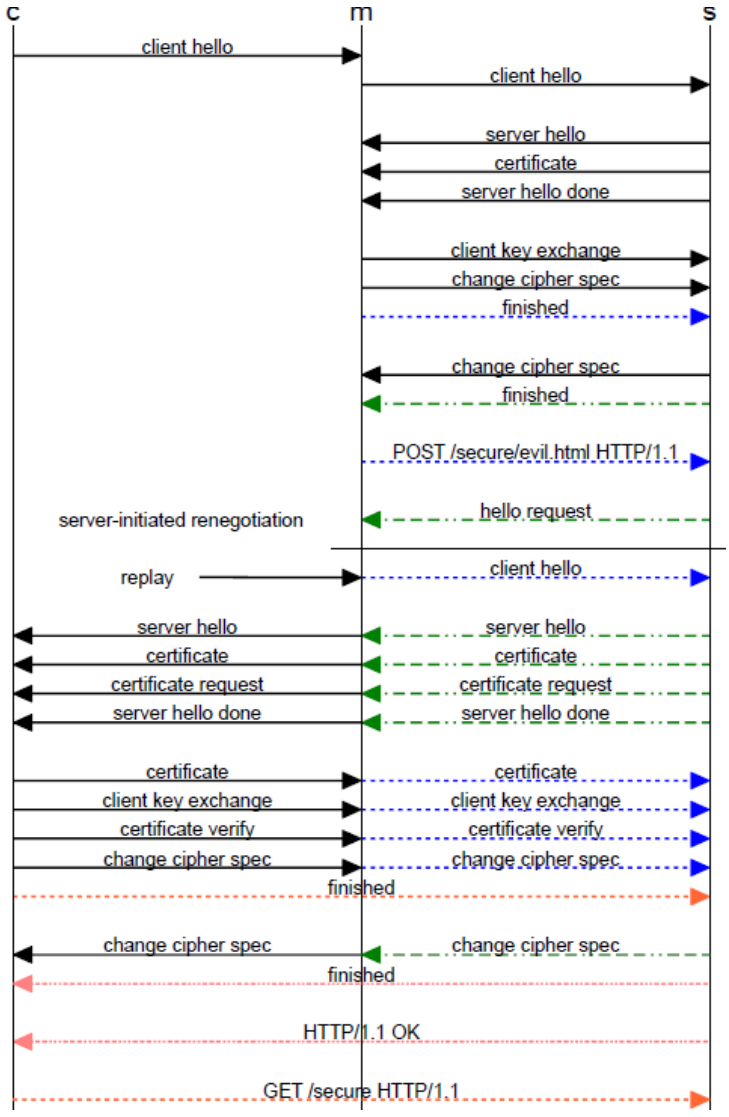

Abbildung 0.6: TLS Handshake

Man-in-the-middle-Angriffe auf RDP & VNC

Das Remote Desktop Protocol (RDP) ist ein Netzwerkprotokoll von Microsoft zur Steuerung von Desktops auf fernen Computern. Bei RDP fungiert eines der beiden Systeme als Terminalserver. Dieser Terminalserver erzeugt Bildschirmausgaben auf dem Terminal-Client. Außerdem können Maus- und Tastatureingaben vom Terminal-Client entgegengenommen werden. Dieses Protokoll wird häufig verwendet und ist der De-facto-Standard für Fernwartung in vielen Rechenzentren. Daher wird auch ein großer Wert auf Sicherheit gelegt, deshalb verwendet jede RDP-Version den RC4-Chiffrieralgorithmus, der für die Verschlüsselung von Datenströmen in Netzwerken konzipiert ist. Als Standardeinstellungen wird eine 128 Bit Verschlüsselung verwendet. Dennoch gibt es Schwachstellen im Remote Desktop Protocol. Standardmäßig sind die Zertifikate, welche für die Verschlüsselung verwendet werden, in der Registry vom Server gespeichert unter

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermService\Parameters\Certificate

Dieses Zertifikat wird für den Schlüsselaustausch verwendet und beinhaltet einen öffentlichen RSA-Schlüssel und eine digitale Signatur. RDP verwendet einen privaten RSA-Schlüssel um den öffentlichen RSA-Schlüssel des Servers zu signieren. Dieser private RSA-Key ist jedoch unter jeder Windows-Version, sowohl bei Clients als auch bei Servern in der Datei „mstlsapi.dll“ gespeichert. Das bedeutet, dass man den öffentlichen RSA-Schlüssel manipulieren kann und so verschlüsselte RDP-Information bei Standardeinstellungen mittels eines mitm-Angriffs mitlesen kann. Grundsätzlich kann man auch irgendein SSL-Zertifikat verwenden, denn die Standardinstallation würde diesen Fehler nicht melden, da beim Terminal Services Client unter dem Reiter „Erweitert“ als Verification Policy steht, dass Verbindungen hergestellt werden und keine Warnungen angezeigt werden sollen. Das bedeutet, dass bei Standardeinstellungen der Angreifer mittels eines MITM Angriffs sämtliche Informationen, wie zum Beispiel Benutzername und Passwort erhält. Dies kann man leicht mit dem Programm CAIN[10] durchführen. (vgl.[SANS2009a], vgl.[OXID2005], vgl.[MICR2007a])

[10] http://www.oxid.it/cain.html

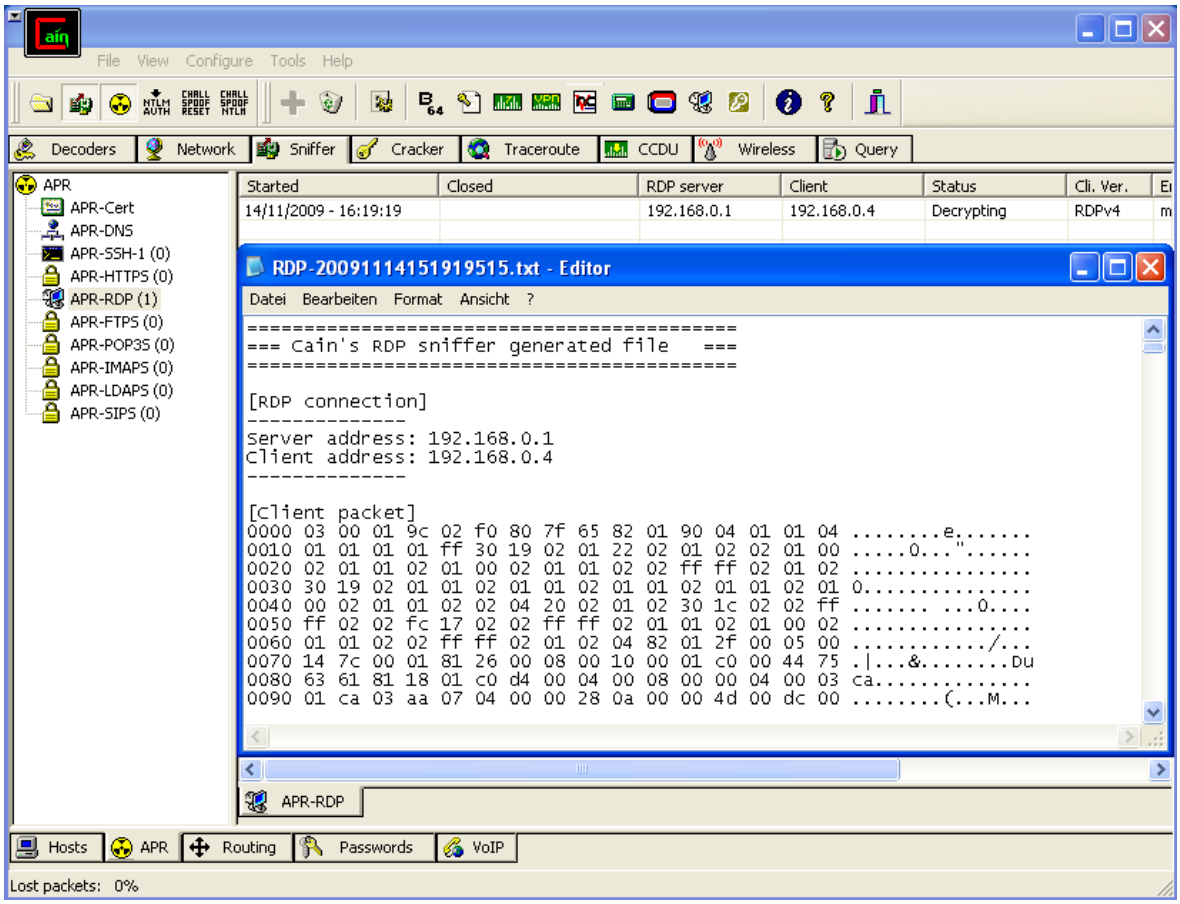

Abbildung 0.7: RDP Hack mittels dem Programm Cain

Man erhält eine Datei mit dem mitgesnifften Benutzernamen und dem eingegebenen Passwort. Die durchsucht man am besten mit der Konsole und erhält so das eingegebene Passwort.

```
C:\>type RDP-20091114151919515.txt | find "press"
Key pressed client-side:0x1e-'a'
Key pressed client-side:0x20-'d'
Key pressed client-side:0x32-'m'
Key pressed client-side:0x17-'i'
Key pressed client-side:0x31-'n'
```

Listing 3: Mitgesnifftes RDP-Passwort

Allerdings besteht die Möglichkeit, RDP-Verbindungen mittels Transport Layer Security (TLS) zusätzlich abzusichern, womit eine sichere Authentifizierung gewährleistet ist, wobei man hierbei wieder auf die oben erwähnten Schwachstellen von SSL/TLS achten muss.

Eine weitere Option wäre, eine andere Remote-Administration-Lösung zu verwenden. Denn man kann nicht nur das Microsoft eigene Protokoll RDP, sondern beispielsweise VNC verwenden. Dies ist eine weitere Software, die den Bildschirminhalt eines entfernten Rechners auf einem lokalen Rechner anzeigt und im Gegenzug Tastatur- und Mausbewegungen des lokalen Rechners an den entfernten Rechner sendet. So ist ein entferntes Arbeiten möglich. Ein weit verbreitetes Software-Produkt ist zum Beispiel

RealVNC, von dem es auch eine abgespeckte Open-Source-Lösung gibt.

VNC arbeitet auch mit unterschiedlichen Authentifizierungsmethoden und kann verschlüsselte Verbindungen aufbauen. Jedoch ist auch dieses Protokoll nicht vor Implementierungsfehlern gefeit. Denn in einigen Versionen funktioniert die Client-Authentifizierung nicht einwandfrei. Der Client verbindet sich zum Server und dieser bietet ihm eine Liste von unterstützten Authentifizierungsmöglichkeiten an. Grundsätzlich wählt der Client dann eine Methode aus der Liste aus. Jedoch, dank eines Implementierungsfehlers, kann der Client angeben, keine Authentifizierung zu verwenden, auch wenn der Server diese Möglichkeit gar nicht anbietet. Der Server akzeptiert diese Vorgehensweise und der Angreifer würde unauthentifiziert Zugriff auf den Server erhalten. Wenn nun dieser Server mit administrativen Rechten ausgestattet ist, hat der Angreifer vollen Zugriff und Kontrolle über das System.

In den neuesten Versionen von RealVNC ist dieses Problem gelöst worden. Die Passwörter werden aber immer noch in der Registry unter Windows gespeichert. Bei Lokalem Zugriff, kann man diese Hashes auslesen und gelangt so zu den Passwörtern. Jedoch ist es zudem sinnvoll bei Remote-Administration über das Internet immer über einen verschlüsselten Tunnel sich zum Server zu verbinden. Denn solch eine Remote-Administration stellt eine große Angriffsfläche dar, weil ein Angreifer vollen Zugriff über ein System erlangen kann, sobald er solch eine Session übernommen hat.

Man-in-the-middle-Angriffe auf SSH

Angriffe auf SSHv1

Diese erste Version von SSH hat bekannte Schwächen in der Integritätsprüfung, denn SSHv1 verwendet noch Checksummen zur Sicherstellung der Daten-Integrität. Dies hilft zwar bei Übertragungsfehlern, jedoch nicht bei einem mitm-Angriff bzw. bei der Manipulation von Daten, da der Angreifer die Checksumme entsprechend korrigieren kann. Außerdem wird sowohl der Host- als auch der Server-Key an den Client gesendet und daraufhin sendet er den Session-Key verschlüsselt mit den soeben erhaltenen Schlüsseln zurück. Dieser Prozess ist wieder einem mitm-Angriff ausgeliefert. Deshalb kann man dieses Protokoll leicht mit Tools wie zum Beispiel sshmitm[11], oder dem schon unter 3.7.2 Man-in-the-middle-Angriffe auf RDP vorgestellten Programm CAIN angreifen.

Angriffe auf SSHv2

Schwieriger ist es jedoch, SSH Version 2 mittels solcher Tools anzugreifen, da sämtliche Schwachstellen die unter Angriffe auf SSHv1 beschrieben wurden, ausgebessert wurden. Die Ausbesserung der Designfehler solch eines Protokolls bedeutet nicht, dass das Protokol sicher implementiert wurde. Viele Implementierungen erlauben eine SSHv1/SSHv2 Kompatibilität, um flexibler zu sein und dies führt zu gewissen Schwächen. Das SSH-Protokoll sieht vor, dass sowohl Client als auch Server einen sogenannten „banner“ austauschen mit den Informationen der SSH-Version, bevor der Schlüsselaustausch durchgeführt wird. Solch ein Banner sieht zum Beispiel folgendermaßen aus:

[11] http://www.monkey.org/ dugsong/dsniff/

```
SSH-1.99-OpenSSH_2.2.0p1
```

SSH-1.99 bedeutet, dass der Client sowohl mittels SSHv1 als auch mittels SSHv2 mit dem Server kommunizieren kann. Abhängig von der Client-Konfiguration bevorzugt er entweder Version 1 oder Version 2. Daraufhin kann der Angreifer als Antwort auf diese Protokollanfrage jeweils nur die nicht bevorzugte Protokollversion anbieten. Wenn der Client Version 1 bevorzugt, sieht das zum Beispiel so aus:

```
SSH-2.00-TESO-SSH
```

Da Der Client normalerweise aber nur die Server-Authentifikation der anderen Protokollversion kennt, kommt statt einer Angriffswarnung nur eine Standardmeldung zum Akzeptieren des angeblich unbekannten RSA Schlüssels.

```
Enabling compatibility mode for protocol 2.0
The authenticity of host 'klein-lukas (192.168.0.1)' can't be
   established.
DSA key fingerprint is ab:8a:18:15:67:04:18:34:ec:c9:ee:9b:89:
   b0:da:e6.
Are you sure you want to continue connecting (yes/no)?
```

Listing 4: Hinweis auf den angeblich unbekannten RSA Schlüssel

Nun ist es viel einfach für den User „yes“ einzugeben. Nach der „yes“-Eingabe kann der SSH Server des Angreifers den Benutzernamen und das Passwort erfassen und leitet die SSH-Verbindung zum eigentlichen Server weiter, damit der Benutzer nichts von dem Angriff mitbekommt. Sobald der Client nur ein Protokoll (im besten Fall SSHv2) unterstützt, kann dieser Angriff nicht durchgeführt werden. Denn sobald der Trick mit dem „Banner“ nicht durchgeführt werden kann, erhält der Benutzer eine, aus der Sicht des Angreifers, viel problematischere Warnung.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!       @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-
   middle attack)!
It is also possible that the RSA1 host key has just been
   changed.
The fingerprint for the RSA1 key sent by the remote host is
f3:cd:d9:fa:c4:c8:b2:3b:68:c5:38:4e:d4:b1:42:4f.
Please contact your system administrator.
```

Listing 5: Angriffswarnung eines möglichen Angriffs

Bei dem oben erklärten Angriff, wo die Versionskompatibilität des SSH-Protokolls ausgenutzt wird kommt man als Angreifer relativ leicht ans Ziel. Jedoch wäre ein weiterer Angriff auf SSHv2, wo Client und Server nur eine Version unterstützen aus der Sicht des Angreifers wünschenswert. SSHv2 verwendet den Host-Key nicht zum Verschlüsseln, so wie bei Version 1, sondern SSH2 verwendet den Host-Key zum Prüfen, ob die ausgetauschten Pakete manipuliert wurden. Dies macht es, indem es den „Message Authentication Code-MAC“ des Servers und den Hash des Clients vergleicht. Der

„MAC“ wird erstellt, indem der Server einen Hash aus den ausgetauschten Paketen erstellt und signiert diesen mit dem privaten Schlüssel des Verschlüsselungsalgorithmus. SSHv2 ist flexibel mit der Auswahl des Algorithmus und die beiden Kommunikationspartner machen sich den zu verwendenden Algorithmus aus. Das sieht beispielsweise so aus:

```
Lukas@klein-lukas:~> telnet 192.168.0.1 22
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '^]'.
SSH-1.99-OpenSSH_2.2.0p1
SSH-2.0-client
'$es??%9?2?4D=?)??ydiffie-hellman-group1-sha1ssh-dss ...
```

Listing 6: Aushandlung des Verschlüsselungsalgorithmus

Hierbei ist das “ssh-dss” am Schluss sehr essentiell, da dies das bevorzugte Protokoll ist. Daraufhin sendet der Angreifer sein bevorzugtes Protokoll, und zwar immer das jeweils andere, in diesem Fall RSA. Nun erscheint wieder eine Standardmeldung zum Akzeptieren des unbekannten RSA-Schlüssels (da der Client den RSA-Schlüssel nicht kennt) und keine Warnung eines Angriffes. Diese beiden Angriffsformen (Banner-Hack & Key-Hack) auf SSHv2 können mit dem Programm ssharp[12] durchgeführt werden. (vgl.[THEP2002], vgl.[THC2003])

Es gibt noch zwei weitere Schwachstellen im SSH Protokoll. Zum Ersten werden die zu sendenden Pakete nur bis zu einer Größe von acht Byte mit Zufallsdaten aufgestockt. Dies ermöglicht es, die ungefähre Größe der eigentlichen Daten zu ermitteln. Solch ein Angriff wird auch Padding-Attack bezeichnet. Zum Zweiten bietet das SSH Protokoll den sogenannten „interactive mode“ als Übertragungsmodus an und dabei wird jeder einzelne Tastendruck sofort verschlüsselt und gesendet. Dabei kann man wiederum die Zeiträume zwischen zwei Tasteninteraktionen des Benutzers ermitteln, dies ist bekannt als Timing-Attack. Diese beiden Schwächen können ausgenutzt werden, sodass gezielte Pakete mit sensitiven Daten (z.B. Benutzername und Passwort) aus dem Strom der gesendeten Informationen herausgefiltert werden und durch Analyse das Passwort entschlüsselt werden kann. (vgl.[STAN2003], vgl.[OPEN2001])

[12] http://stealth.7350.org/7350ssharp.tgz

Literaturverzeichnis

[LEWI2008] Lewis, Wayne, Cisco Lan Switching and Wireless – CCNA Exploration Companion Guide, 2008, Cisco Press, Indianapolis

[SDO2008] Schöndorfer, Christian, Theoretische & Praktische Grundlagen der Netzwerksicherheit, 2008, Version 1.5, HTL Rennweg

Internet Quellen

[WIKI2009a] Man-in-the-middle-Angriff [online], aktualisiert am 16.10.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: http://de.wikipedia.org/wiki/Man-in-the-middle-Angriff

[MOXIE2009a] Moxie, Marlinspike, Null Prefix Attacks Against SSL/TLS Certificates [online], aktualisiert am 29.07.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: http://www.thoughtcrime.org/papers/null-prefix-attacks.pdf

[MOXIE2009b] Moxie, Marlinspike, Defeating SSL [online], aktualisiert am 30.07.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf

[HEISE2009a] Black Hat: Neue Angriffsmethode auf SSL vorgestellt [online], aktualisiert am 29.02.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: http://www.heise.de/security/meldung/Black-Hat-Neue-Angriffsmethoden-auf-SSL-vorgestellt-198285.html

[HEISE2009b] Neue SSL–Attacken demonstriert [online], aktualisiert am 30.07.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: http://www.heise.de/security/meldung/Neue-SSL-Attacken-demonstriert-748883.html

[HEISE2009c] Trickzertifikat für SSL veröffentlicht [online], aktualisiert am 30.09.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: http://www.heise.de/security/meldung/Trickzertifikat-fuer-SSL-veroeffentlicht-Update-798273.html

[NOIS2009] Merry Certmas! CN=*\x00thoughtcrime.noisebridge.net [online], aktualisiert am 29.09.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: https://www.noisebridge.net/pipermail/noisebridge-discuss/2009-September/008400.html

[EXTE2009] Authentication Gap in TLS Renegotiation [online], aktualisiert am 04.11.2009 [zitiert am 05.11.2009], Auszug verfügbar im Internet: http://extendedsubset.com/?p=8

[LINK2009] Another Protocol Bites The Dust [online], aktualisiert am 05.11.2009 [zitiert am 05.11.2009], Auszug verfügbar im Internet: http://www.links.org/?p=780

[IETF2009] MITM attack on delayed TLS-client auth through renegotiation [online], aktualisiert am 04.11.2009 [zitiert am 05.11.2009], Auszug verfügbar im Internet: `http://www.ietf.org/mail-archive/web/tls/current/msg03928.html`

[SANS2009a] Cyber Security Awareness Month - Day 9 - Port 3389/tcp (RDP) [online], aktualisiert am 09.10.2009 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://isc.sans.org/diary.html?storyid=7303`

[OXID2005] Remote Desktop Protocol, the Good the Bad and the Ugly [online], aktualisiert am 28.05.2005 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://www.oxid.it/downloads/rdp-gbu.pdf`

[MICR2007a] Grundlegendes zu Remote Desktop Protocol (RDP) [online], aktualisiert am 27.03.2007 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://support.microsoft.com/kb/186607`

[THEP2002] It cuts like a knife [online], aktualisiert am 28.07.2002 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://www.phrack.org/issues.html?id=11&issue=59`

[THC2003] Attacking Vulnerabilities in the Human Brain [online], aktualisiert am 25.10.2003 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://freeworld.thc.org/papers/ffp.pdf`

[STAN2003] Remote Timing Attacks are Practical [online], aktualisiert am 12.05.2003 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf`

[OPEN2001] Passive Analysis of SSH (Secure Shell) Traffic [online], aktualisiert am 06.08.2001 [zitiert am 30.10.2009], Auszug verfügbar im Internet: `http://www.openwall.com/advisories/OW-003-ssh-traffic-analysis/`