

Angriffe auf RDP – Wie man RDP-Sitzungen abhört

Adrian Vollmer

SySS GmbH
adrian.vollmer@syssec.de

Zusammenfassung

Das Remote Desktop Protocol (RDP) ist ein weit verbreitetes Protokoll, das unter anderem zur Fernadministration von Windows-Rechnern verwendet wird. Dabei wird SSL verwendet, um die Integrität und Vertraulichkeit der übertragenen Daten zu gewährleisten. In der sehr häufig anzutreffenden Standardkonfiguration werden in diesem Kontext selbstsignierte Zertifikate eingesetzt, die fast gar keinen Schutz gegen Abhörangriffe bieten. Wird die oft empfohlene Network Level Authentication (NLA) eingesetzt (aber nicht forciert), kann der Angreifer immer noch die Verbindungssicherheit auf einfaches SSL herabstufen, so dass der Schutz vollständig auf der Annahme beruht, dass der Anwender sorgfältig genug ist, Zertifikatswarnungen nicht zu ignorieren. Dieser Artikel beschreibt, wie eine konkrete Umsetzung eines solchen Abhörangriffs aussieht. Zur Demonstration wurde ein Proof-of-Concept in Form eines Python-Skripts entwickelt, das für einen Angreifer interessante Informationen wie Passwörter und Tastatureingabeereignisse aus einer bestehenden Verbindung extrahiert. Es soll der Sensibilisierung von Systemadministratoren und anderen IT-Verantwortlichen im Rahmen von Penetrationstests und Schulungen dienen. Der Artikel beschreibt die relevanten Details der RDP-Implementierung, die Entwicklung des Proof-of-Concepts und schließt mit Empfehlungen und Gegenmaßnahmen, die den beschriebenen Angriff verhindern. Das Python-Skript steht zur freien Verfügung.

1 Einleitung

Das Remote Desktop Protocol (RDP) wird tagtäglich von Systemadministratoren dazu verwendet, sich über das Netzwerk an Windows-Rechnern anzumelden. Die vielleicht häufigste Anwendung ist die Fernadministration von kritischen Servern wie etwa die Domänencontroller im Kontext von hochprivilegierten Benutzerkonten, dessen Zugangsdaten über RDP übertragen werden. Es ist deshalb entscheidend, eine sichere RDP-Konfiguration zu verwenden.

Im Alltag eines Penetrationstesters werden regelmäßig Fehlkonfigurationen beobachtet, die dazu führen, dass Systemadministratoren und anderen Anwendern in einer Active-Directory-Umgebung routinemäßig Zertifikatswarnungen präsentiert werden (siehe Abbildung 1), die zwangsweise ignoriert werden. Falls diese Warnungen im Firmennetzwerk häufig auftreten, können tatsächliche Man-in-the-Middle-Angriffe nicht erkannt werden.

Das Ziel dieses Artikels ist es, das Bewusstsein für die Bedeutung von Zertifikatswarnungen zu steigern, einen Einblick in die Perspektive eines Angreifers zu geben und zu zeigen, wie man ein Windowsnetzwerk im Unternehmensumfeld in Bezug auf RDP sicher konfiguriert. Die vorgesehene Zielgruppe enthält Systemadministratoren, Penetrationstester und fachlich Interessierte.

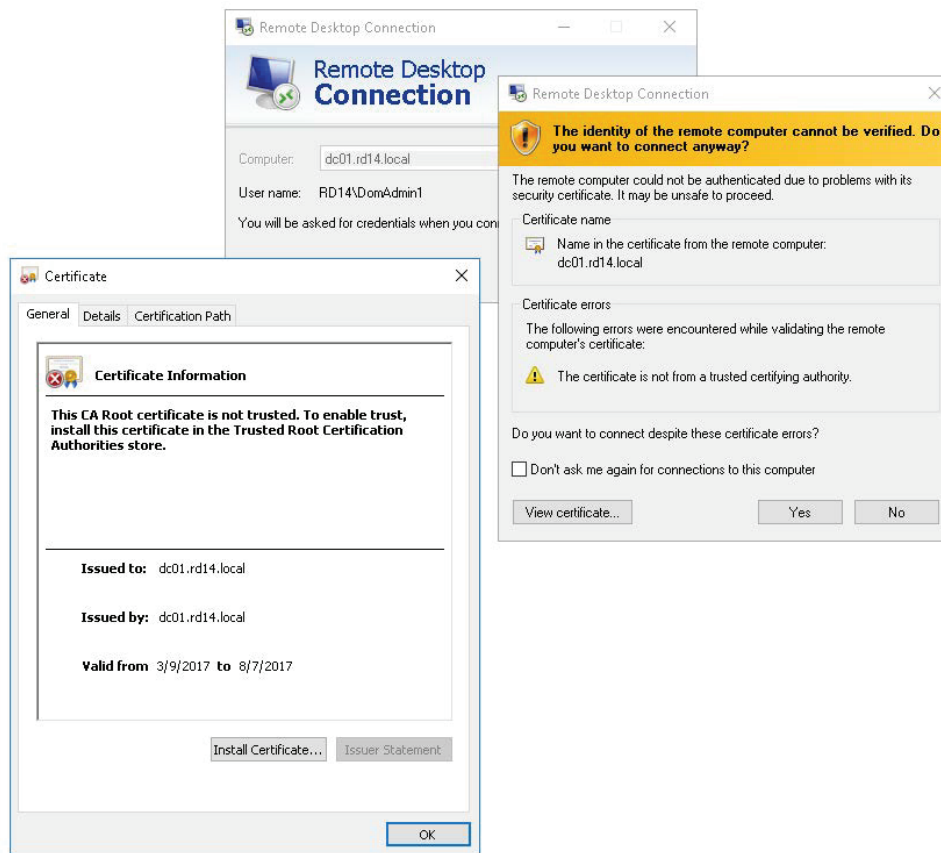


Abb. 1: Eine typische Zertifikatswarnung während des Verbindungsaufbaus einer RDP-Sitzung.

Wir werden sehen, wie ein “Man in the Middle” Zugangsdaten ausspähen kann, wenn der Anwender nicht mit der nötigen Sorgfalt vorgeht. Die hier präsentierten Informationen sind nicht neu, größtenteils bereits frei verfügbar und wurden teilweise schon von anderen ausgearbeitet – zum Beispiel in der Form von Cain, Teil des Frameworks Cain&Abel von [Mont14]. Allerdings wird Cain nicht nur nicht mehr aktiv weiterentwickelt, es ist außerdem nur für Windows verfügbar und der Quelltext steht nicht zur Verfügung. Der Einsatz von Closed-Source-Produkten im Rahmen eines Penetrationstests ist aus verschiedenen Gründen nicht wünschenswert. Davon abgesehen ist der Angriff von Cain – wie wir sehen werden – nicht vollständig transparent und für ein misstrauisches Opfer zu erkennen.

Dieser Umstand gab die Motivation dazu, zum einen die verfügbaren Informationen zusammenzutragen und zum anderen ein neues Proof-of-Concept namens *Seth* zu entwickeln. Wir wollen alle relevanten Details von RDP verstehen und einen Angriff so unerkennbar wie möglich simulieren. Seth ist als Python3-Skript quelloffen und steht unter der MIT-Lizenz frei zur Verfügung [Voll17b].

2 Ein erster Blick auf das Protokoll

Der erste Schritt des Protokolls wird anschaulich, wenn man sich mithilfe von Wireshark den Anfang einer RDP-Verbindung ansieht.

Wie man in Abbildung 2 sehen kann, beginnt der Client damit, Sicherheitsprotokolle für die

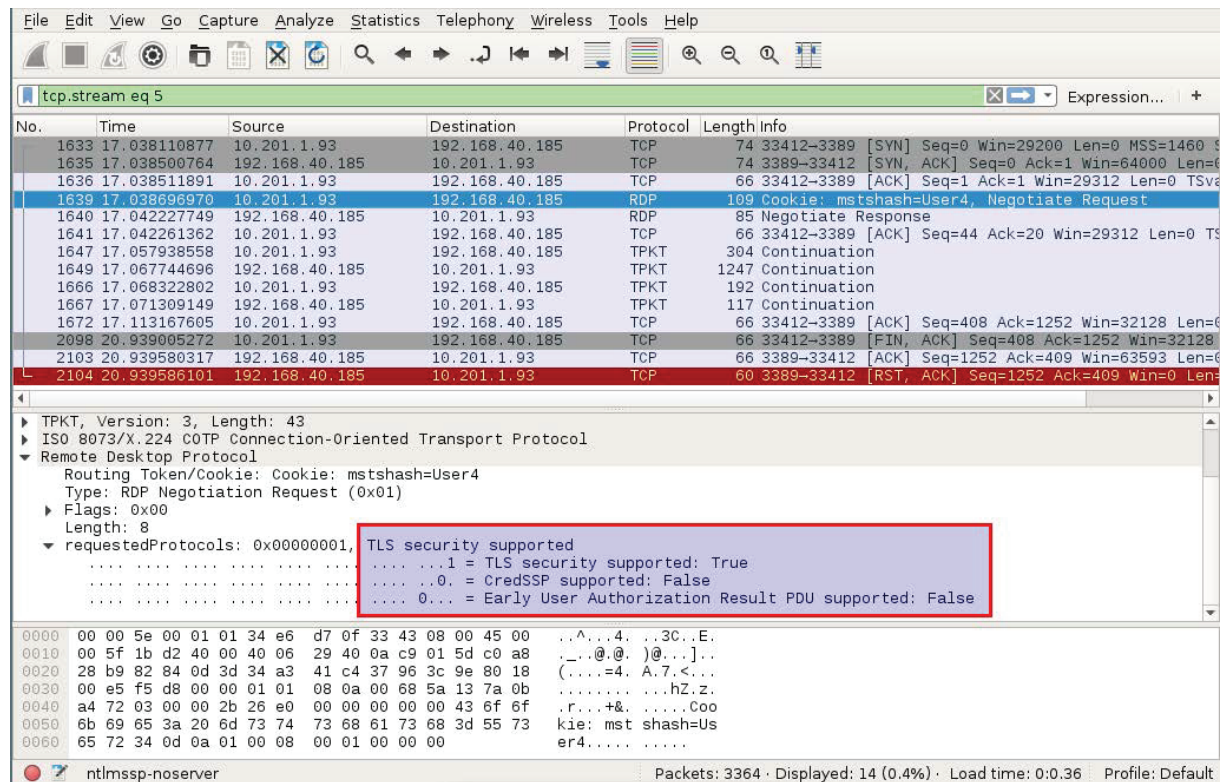


Abb. 2: Der Anfang einer RDP-Sitzung, wie man ihn mit Wireshark beobachten kann. Markiert sind nahe der Mitte des Bildes die vom Client vorgeschlagenen Sicherheitsprotokolle.

Verwendung von RDP vorzuschlagen. Es wird unterschieden zwischen drei Protokollen:

- Standard RDP Security
- Enhanced RDP Security oder TLS Security
- CredSSP

Jeder RDP-Client beherrscht Standard RDP Security. In obigem Fall ist der Client außerdem zum Enhanced-RDP-Security-Protokoll fähig. Dieses Protokoll ist nichts Anderes als eine mit Standard RDP Security “gesicherte” Verbindung, die zusätzlich durch TLS geschützt ist. (Die Begriffe SSL und TLS werden im Folgenden synonym verwendet.)

CredSSP wird auch innerhalb eines TLS-Tunnels verwendet, aber anstatt das Passwort über diesen geschützten Tunnel zu übertragen, wird NetNTLM oder Kerberos verwendet. Dieses Protokoll ist auch unter dem Begriff Network Level Authentication (NLA) bekannt.

Eine weitere vom Client inserierte Fähigkeit ist Early User Authentication. Wenn der Client diesen Modus unterstützt, kann der Server dem Client vor der Authentifizierung mitteilen, ob der sich anmeldende Benutzer überhaupt autorisiert ist, eine Fernanmeldung an diesem System durchzuführen.

In der Wireshark-Sitzung kann man erkennen, dass nach der Protokollaushandlung ein SSL Handshake stattfindet, wenn man den entsprechenden TCP-Stream als SSL dekodiert.

Dies bedeutet, dass man keinen einfachen SSL-Proxy verwenden kann, um die Verbindung abzuhören, da der Proxy sich dem Protokoll bewusst sein muss. Er muss erkennen, wann der

SSL Handshake initiiert werden muss, ähnlich wie das StartTLS-Verfahren bei beispielsweise SMTP oder FTP. Wir werden Python für die Implementierung eines solchen Proxys verwenden. Dafür erzeugen wir einen Server-Socket, zu dem sich der Client des Opfers verbindet, und einen Client-Socket, der sich zum tatsächlichen Server verbindet. Wir leiten die Daten zwischen den beiden Sockets weiter und initiieren den SSL Handshake zum korrekten Zeitpunkt. Natürlich werden wir die weitergereichten Daten genau untersuchen und gegebenenfalls auch modifizieren.

Das erste, was wir modifizieren wollen, sind die vom Client inserierten Fähigkeiten bezüglich des Sicherheitsprotokolls. Der Client wird dem Server üblicherweise mitteilen wollen, dass er CredSSP beherrscht, aber wir werden diese Information auf dem Weg zum Server auf Standard RDP Security ändern. Ist der Server nicht sicher konfiguriert, wird er dem anstandslos Folge leisten.

3 Grundzüge eines RDP-Proxys in Python

Die Hauptschleife des Python-Skripts sieht in den Grundzügen wie folgt aus:

```

1 def run():
2     open_sockets()
3     handle_protocol_negotiation()
4     if not RDP_PROTOCOL == 0:
5         enableSSL()
6     while True:
7         if not forward_data():
8             break
9
10
11 def forward_data():
12     readable, _, _ = select.select([local_conn, remote_socket], [], [])
13     for s_in in readable:
14         if s_in == local_conn:
15             From = "Client"
16             to_socket = remote_socket
17         elif s_in == remote_socket:
18             From = "Server"
19             to_socket = local_conn
20         data = s_in.recv(4096)
21         if len(data) == 4096:
22             while len(data)%4096 == 0:
23                 data += s_in.recv(4096)
24         if data == b"": return close()
25         dump_data(data, From=From)
26         parse_rdp(data, From=From)
27         data = tamper_data(data, From=From)
28         to_socket.send(data)
29     return True
30
31
32 def enableSSL():
33     global local_conn
34     global remote_socket
35     print("Enable SSL")
36     local_conn = ssl.wrap_socket(
37         local_conn,
38         server_side=True,
39         keyfile=args.keyfile,
40         certfile=args.certfile,
41     )
42     remote_socket = ssl.wrap_socket(remote_socket)

```

Die Funktion `run()` öffnet die Sockets, regelt (und manipuliert) die Protokollaushandlung und

initiiert den SSL Handshake, falls nötig. Anschließend werden die Daten einfach zwischen den beiden Sockets weitergereicht. Die Funktion `dump_data()` gibt die Daten zu Debug-Zwecken als Hexdump auf dem Bildschirm aus. Die Funktion `parse_rdp()` extrahiert interessante Informationen aus dem Datenstrom und mit `tamper_data()` können Veränderungen an den Daten durchgeführt werden.

4 Standard RDP Security

Standard RDP Security ist das schwächste aller mit RDP einsetzbaren Sicherheitsprotokolle. Microsoft hat sich dazu entschieden, die gleichen Verfahren wie im SSL-Standard einzusetzen, hat dabei aber einen entscheidenden Fehler gemacht, weshalb das Protokoll als vollständig gebrochen angesehen werden muss.

Wie wir später sehen werden, ist es für einen erfolgreichen Angriff nicht zwingend nötig, die Verbindung auf Standard RDP Security herabzustufen, weshalb wir an dieser Stelle die Details zu diesem Protokoll nur kurz umreißen wollen. Der Ablauf ist wie folgt:

1. Der Client kündigt sein Vorhaben an, Standard RDP Security verwenden zu wollen.
2. Der Server, falls er einverstanden ist, übergibt seinen öffentlichen RSA-Schlüssel, der eigens für ihn erzeugt wurde, zusammen mit dem sogenannten Server Random (einem für die Sitzung erzeugten Zufallswert). Die Gesamtheit des öffentlichen Schlüssels zusammen mit einigen anderen Informationen, wie dem Servernamen, Gültigkeitszeitraum, etc., heißt "Zertifikat". Um seine Authentizität zu gewährleisten ist das Zertifikat unter Verwendung des RSA-Verfahrens kryptographisch signiert – und zwar mit dem privaten Terminal-Services-Schlüssel.
3. Der Client validiert das Zertifikat mithilfe des öffentlichen Terminal-Services-Schlüssels. Falls er erfolgreich ist, erzeugt er ein "Client Random", verschlüsselt es mit dem öffentlichen Schlüssel des Servers und übergibt es ihm.
4. Der Server entschlüsselt das Client Random mit seinem privaten Schlüssel.
5. Sowohl der Server als auch der Client leiten die Sitzungsschlüssel aus dem Client Random und dem Server Random ab. Diese Sitzungsschlüssel werden im Anschluss dazu verwendet, die Daten für den Rest der Sitzung symmetrisch zu verschlüsseln.

Für die symmetrische Verschlüsselung werden je nach Konfiguration der Verschlüsselungsstufe die folgenden Chiffren eingesetzt:

1. 3DES (FIPS)
2. RC4 mit 128 Bit Schlüssellänge (High)
3. RC4 mit 56 Bit Schlüssellänge (Medium)
4. RC4 mit 40 Bit Schlüssellänge (Low)
5. Keine Verschlüsselung (None)

Ein SSL Handshake verläuft vereinfacht gesagt ganz analog dazu ab. Microsoft hat jedoch einen elementaren Fehler begangen: Der Terminal-Services-Schlüssel wird hartkodiert in allen Windows-Instanzen verwendet. Er kann somit mit ein wenig Aufwand aus einer RDP-Host-fähigen Windows-Version extrahiert werden. Deshalb hat Microsoft ihn mittlerweile im Internet veröffentlicht [MSD17a].

Diese Verschlüsselung zu brechen ist also nicht mehr als eine Übung. Dazu muss man lediglich ein eigenes RSA-Schlüsselpaar erstellen und das Zertifikat durch das gefälschte, mit dem Terminal-Services-Schlüssel signierte Zertifikat austauschen, um das Client Random entschlüsseln zu können. Der Rest findet dann rein passiv statt.

Dieses Vorgehensweise ist übrigens genau die von Cain [Mont14].

5 Enhanced RDP Security

Für einen motivierten Angreifer ist die Herabstufung der Verbindung auf Standard RDP Security unbefriedigend, denn die Warnung bezüglich der nicht-validierbaren Identität des Servers sieht anders aus. Es ist kein ungültiges SSL-Zertifikat, das einem präsentiert wird, sondern die Verbindung ist de facto überhaupt nicht verschlüsselt. Doch der Angriff soll für das Opfer völlig unbemerkbar bleiben. Andernfalls könnte der Lerneffekt bei einem Penetrationstest oder einer Schulung ausfallen, da das geübte Auge den Angriff erkennt und der Hinweis, warum Zertifikatswarnungen ernstzunehmend sind könnte weniger fruchtbar sein.

Deshalb wollen wir versuchen, die Verbindung nur auf Enhanced RDP Security herabzustufen. Dafür wird ein SSL-Zertifikat benötigt, was augenscheinlich identisch ist mit dem, das der Benutzer normalerweise präsentiert bekommt. Wir werden also das originale Zertifikat mithilfe von OpenSSL beschaffen, die ASN.1-Struktur parsen und den öffentlichen Schlüssel durch einen eigens erzeugten austauschen. Danach signieren wir das Zertifikat mit dem zugehörigen privaten Schlüssel. Dieses “geklonte” Zertifikat wird für einen Menschen vom Originalzertifikat nahezu ununterscheidbar sein. Es wird (wie eben das Originalzertifikat in den meisten Fällen) nicht validierbar sein, weil es nicht von einer vertrauenswürdigen Zertifikatsautorität signiert wurde. Außerdem wird der Fingerabdruck anders sein, aber es ist äußerst unwahrscheinlich, dass ein Benutzer den Fingerabdruck manuell prüft.

Dieses Zertifikat verwenden wir, wenn SSL in unserem TCP-Socket aktiviert wird. Wie in Abschnitt 4 erwähnt, wird innerhalb des SSL-Tunnels wieder Standard RDP Security gesprochen, aber in diesem Fall wird vom Server als Verschlüsselungsstufe immer “None” gewählt. Das einzige zusätzliche Sicherheitsmerkmal ist eine Nachricht vom Server an den Client, das die ursprünglich vom Client angepriesenen Sicherheitsprotokolle enthält. Stimmen diese nicht mit der ursprünglichen Auswahl überein, trennt der Client die Verbindung. Doch es ist bereits zu spät, denn der Angreifer kann diesen Wert nun ebenfalls manipulieren. Er kann danach also ohne Weiteres Tastatureingabeereignisse mitlesen.

Wie erwartet sieht das Opfer nun eine reguläre Zertifikatswarnung. Doch einen Unterschied gibt es noch: In der Standardkonfiguration ist NLA aktiv, was bedeutet, dass die Zugangsdaten abgefragt werden, bevor die RDP-Sitzung aufgebaut ist. Bei diesem Angriff gelangt das Opfer zuerst zu einer Windows-Anmeldemaske, an dem es seine Zugangsdaten eingeben muss. Auch dies ist eine Unregelmäßigkeit, die vom typischen Arbeitsfluss abweicht und deshalb bemerkt werden könnte.

6 CredSSP

Wie schon erwähnt ist CredSSP als Synonym zu NLA verstehen. Hier kommt wiederum entweder NetNTLMv2 (kurz: NTLM) oder Kerberos zum Einsatz. Es ist also der Domänencontroller, der als vertrauenswürdige dritte Instanz agiert.

Um dieses Verhalten herbeizuführen, können wir die Antwort des Servers, die nach der Übermittlung der NTLM-Antwort übermittelt wird, durch folgende Nachricht ersetzen, die normalerweise gesendet wird, wenn der Domänencontroller nicht verfügbar ist, und die Herabstufung veranlasst:

```
00000000: 300d a003 0201 04a4 0602 04c0 0000 5e 0.....^
```

So kann ein Angreifer die CredSSP-Authentifizierung erfolgreich zu Enhanced RDP Security herabstufen, falls NTLM zur Anwendung kommt.

Was aber, wenn der Client Kerberos statt NTLM einsetzen will? In diesem Fall bekommt der Client vom Domänencontroller ein Service-Ticket, dessen Authentizität der Server prüfen kann ohne den Domänencontroller zu kontaktieren. Ein Angreifer, der bereits den Netzwerkverkehr kontrollieren kann, kann allerdings auch die Verbindung zum Domänencontroller blockieren. Das bedeutet, dass der Client kein Service-Ticket bekommt und deshalb auf NTLM zurückgreifen muss, woraufhin die obige Vorgehensweise wieder greift.

7 Der fertige Angriff

Die bisher aufgeführten Konzepte zusammenzuführen ist eine leichte Übung. Um den RDP-Proxy effektiv einsetzen zu können, muss eine Man-in-the-Middle-Position eingenommen werden. Dafür gibt es zahlreiche Möglichkeiten, wobei eine davon ARP-Spoofing ist. Sie eignet sich besonders für ein kurzes Proof-of-Concept um den Angriff zu demonstrieren. Da es ein Angriff auf Schicht 2 des OSI-Modells ist, muss der Angreifer im gleichen Subnetz wie der Client des Opfers sein. Ein typisches Tool für diesen Angriff trägt den Namen `arp spoof` und ist Teil der Sammlung `dsniff` von [Song00].

Nachdem man ARP-Antworten gespoofed und die Weiterleitung von IPv4-Paketen aktiviert hat, läuft der gesamte Datenstrom zwischen dem Opfer und dem Gateway über die Angreifermaschine. Da der Angreifer die IP-Adresse des RDP-Servers, zu dem sich das Opfer vor hat zu verbinden, noch nicht kennt, kann er den RDP-Proxy noch nicht starten.

Zunächst ist es sinnvoll, mittels `iptables` eine Regel zu erstellen, die SYN-Pakete vom Opfer zu einem RDP-Dienst (TCP-Port 3389) verwirft:

```
$ iptables -A FORWARD -p tcp -s "$VICTIM_IP" --syn --dport 3389 -j REJECT
```

Dies verhindert, dass das Opfer eine RDP-Sitzung direkt mit dem tatsächlichen Server herstellt, ohne bestehende RDP-Sitzungen zu stören.

Als nächstes wartet man auf ein SYN-Paket mit den oben beschriebenen Eigenschaften, um die IP-Adresse des RDP-Servers in Erfahrung zu bringen. Dazu kann beispielsweise `tcpdump` verwendet werden:

```
$ tcpdump -n -c 1 -i "$IFACE" src host "$VICTIM_IP" and \
  "tcp[tcpflags] & tcp-syn != 0" and \
  dst port 3389 2> /dev/null | \
  sed -e 's/.*> \([0-9.]*\)\.3389:.*\/\1/'
```

Die `-c 1` Options weist `tcpdump` an, abzubrechen sobald ein geeignetes Paket eingetroffen ist. Dieses Paket wird dann verloren sein, aber das System des Opfers wird es bald darauf erneut versuchen.

Nun entfernt man die `iptables`-Regeln von vorhin und leitet *alle* TCP-Pakete, die vom Opfer stammen und für Port 3389 bestimmt sind, auf die eigene IP-Adresse um:

Insbesondere dürfen Zertifikatswarnungen nicht ignorierbar sein, sondern müssen Fehlermeldungen darstellen, die einen Verbindungsabbruch erzwingen. Dazu sollten RDP-Server ein validierbares Zertifikat präsentieren, idealerweise im Rahmen einer unternehmensweiten Public-Key-Infrastruktur. Auf den Clients muss dazu das Stammzertifikat der Zertifikatsautorität hinterlegt sein.

Die Frage, ob NLA forciert werden sollte, ist entgegen der landläufigen Meinung nicht ganz eindeutig. Ganz davon abgesehen, dass manche RDP-Clients NLA nicht unterstützen, werden die Zugangsdaten – wie wir gesehen haben – im Speicher des Clientsystems vorgehalten. Damit sind sie (bei den meisten Versionen von Windows) sichtbar für einen Angreifer, der das System des Opfers bereits kompromittiert hat. Er kann sie beispielsweise mithilfe des Tools Mimikatz [Delp17] aus dem Speicher auslesen. Zudem können Benutzer, die ausschließlich über einen Terminalserver arbeiten, sich nicht mehr anmelden, wenn das Attribut “User must change password at next logon” für ihr Konto gesetzt ist.

Hinzu kommt, dass der Mehrwert von NLA begrenzt ist. SSL gilt, solange Zertifikate ordentlich validiert werden, als äußerst robust und sicher. Was durch NLA verhindert wird, ist die Ausnutzung von möglichen Schwachstellen im RDP-Protokoll selbst sowie Denial-of-Service-Angriffe, da ein unauthentifizierter Benutzer eine vollständige, relativ ressourcenintensive RDP-Sitzung aufbauen kann, auch wenn er nur bis zur Windows-Anmeldemaske kommt. Soll NLA doch vom Server forciert werden, kann dies per Gruppenrichtlinie und der folgenden Option getan werden [MST17b]:

```
Computer Configuration\Policies\Administrative Templates\Windows Components\Terminal Services\
Terminal Server\Security\Require user authentication for remote connections by using Network
Level Authentication
```

Des Weiteren empfiehlt sich der Einsatz von Zwei-Faktor-Authentifizierung, insbesondere für kritische Systeme wie die Domänencontroller. Hierfür existieren kommerzielle Produkte von Drittanbietern.

An dieser Stelle ist anzumerken, dass der häufig verwendete RDP-Client für GNU/Linux RDesktop kein NLA beherrscht und nicht einmal in der Lage ist, das SSL-Zertifikat zu validieren. Zumindest ein Alternativprogramm, XFreeRDP, kann immerhin das Zertifikat validieren.

Abschließend soll noch die Empfehlung ausgesprochen werden, Mitarbeiter und Kollegen zu informieren, dass Zertifikatswarnungen ernstzunehmend sind, sei es im Kontext von RDP oder HTTPS oder sonstigem. In einer ordentlich konfigurierten IT-Landschaft sollten derartige Warnungen eine Ausnahme darstellen, die einen Anruf bei der IT-Abteilung rechtfertigt.

Literatur

[Delp17] B. Delpy: Github.com: mimikatz. <https://github.com/gentilkiwi/mimikatz> (2017), Abfragedatum: 2017-06-07.

[Hash] Hashcat. <https://hashcat.net/hashcat/>.

[John] John The Ripper. <http://www.openwall.com/john/>.

[Mont14] M. Montoro: Cain & Abel. <http://www.oxid.it/cain.html> (2014), Abfragedatum: 2017-06-07.

[MSD17a] Microsoft Developer Network: [MS-RDPBCGR]: Terminal Services Signing Key.

- <https://msdn.microsoft.com/en-us/library/cc240776.aspx>, Abfragedatum: 2017-06-07.
- [MSD17b] Microsoft Developer Network: Microsoft NTLM. <https://msdn.microsoft.com/en-us/library/aa378749%28VS.85%29.aspx> (2017), Abfragedatum: 2017-06-07.
- [MSD17c] Microsoft Developer Network: [MS-CSSP]: TSRequest. <https://msdn.microsoft.com/en-us/library/cc226780.aspx> (2017), Abfragedatum: 2017-06-07.
- [MST17a] Microsoft Technet: Remote Desktop Connection Client. [https://technet.microsoft.com/en-us/library/cc753945\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc753945(v=ws.10).aspx) (2017), Abfragedatum: 2017-06-07.
- [MST17b] Microsoft Technet: Security. [https://technet.microsoft.com/en-us/library/cc771869\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc771869(v=ws.10).aspx) (2017), Abfragedatum: 2017-06-07.
- [Song00] D. Song: dsniff. <https://www.monkey.org/~dugsong/dsniff/> (2000), Abfragedatum: 2017-06-07.
- [Voll17a] A. Vollmer: clone-cert.sh. <https://github.com/SySS-Research/clone-cert> (2017), Abfragedatum: 2017-06-07.
- [Voll17b] A. Vollmer: Seth. <https://github.com/SySS-Research/Seth> (2017), Abfragedatum: 2017-06-07.
- [Week15] M. Weeks: Attacking Windows Fallback Authentication. https://www.root9b.com/sites/default/files/whitepapers/R9B_blog_003_whitepaper_01.pdf (2015), Abfragedatum: 2017-06-07.