Pivoting Around Obstacles Rpivot

w1j0y

Attack Host

Web Server Pivoting with Rpivot

Rpivot

Rpivot is a reverse SOCKS proxy tool written in Python for SOCKS tunneling. Rpivot binds a machine inside a corporate network to an external server and exposes the client's local port on the server-side. We will take the scenario below, where we have a web server on our internal network (172.16.5.135), and we want to access that using the rpivot proxy.

```
https://github.com/klsecservices/rpivot
```

```
w1j0y@htb[/htb]$ sudo git clone https://github.com/klsecservices/rpivot.git
```

```
w1j0y@htb[/htb]$ sudo apt-get install python2.7
```

We can start our rpivot SOCKS proxy server using the below command to allow the client to connect on port 9999 and listen on port 9050 for proxy pivot connections.

Running server.py from the Attack Host

```
w1j0y@htb[/htb]$ python2.7 server.py --proxy-port 9050 --server-port 9999 --server-ip 0.0.0.0
```

Confirming Connection is Established

Before running client.py we will need to transfer rpivot to the target.

Transferring rpivot to the Target

```
w1j0y@htb[/htb]$ scp -r rpivot ubuntu@<IpaddressOfTarget>:/home/ubuntu/
```

Configure proxychains to pivot over our local server on 127.0.0.1:9050

```
127.0.0.1 9050
```

Browsing to the Target Webserver using Proxychains

```
proxychains firefox-esr 172.16.5.135:80
```

use curl to read the contens of the webpage

Ubuntu Pivot Host

Running client.py from Pivot Target

```
ubuntu@WEB01:~/rpivot$ python2.7 client.py --server-ip 10.10.14.18 --server-port 9999
```

Similar to the pivot proxy above, there could be scenarios when we cannot directly pivot to an external server (attack host) on the cloud. Some organizations have HTTP-proxy with NTLM authentication configured with the Domain Controller. In such cases, we can provide an additional NTLM authentication option to rpivot to authenticate via the NTLM proxy by providing a username and password. In these cases, we could use rpivot's client.py in the following way:

Connecting to a Web Server using HTTP-Proxy & NTLM Auth

```
python client.py --server-ip <IPaddressofTargetWebServer> --server-port 8080 --ntlm-proxy-ip <IPaddressofProxy> --ntlm-proxy-port 8081 --domain <nameofWindowsDomain> --username <username> --password <password>
```

Web Server on port 80