python-krb5ticket

Release 0.0.2

Deric Degagne

Sep 22, 2021

CONTENTS

CHAPTER

ONE

API DOCUMENTATION

1.1 Ktutil

1.2 Ktutil Helpers

The `ktutil_helpers` module, provides easy-to-use functions to create, delete or list Kerberos keytab entries.

1.3 Krb5

Simply Python wrapper to create Kerberos V5 ticket-granting tickets (TGTs), using either password or keytab file. Also, supports the creation of Kerberos keytab files.

CHAPTER

TWO

GETTING STARTED

Install the `python-krb5ticket` library using pip:

Listing 1: bash

```
$ pip install python-krb5ticket
```

2.1 krb5

The Krb5 class provides an interface to aquire Kerberos ticket-granting tickets (TGTs) using either a key table file or password.

Note: SECURITY ADVISORY

Please refrain from acquiring TGTs using the password method as passwords are not encrypted and passed along in plain text.

2.1.1 Examples

Acquires Kerberos ticket-granting ticket (TGT) with keytab file.

Listing 2: Python

```
import krb5

krb = krb5.Krb5("user@EXAMPLE.COM", "/tmp/krb5cc_user")
krb.acquire_with_keytab("/home/user/user.keytab")
```

Acquires Kerberos ticket-granting ticket (TGT) with password.

Listing 3: Python

```
from krb5 import Krb5

krb = Krb5("user@EXAMPLE.COM", "/tmp/krb5cc_user")
krb.acquire_with_password("thisismypassword")
```

2.2 ktutil

The ktutil class provides an interface to manage Kerberos V5 key table files. This class is a wrapper around the MIT Kerberos ktutil command-line interface.

2.2.1 Examples

Reads the Kerberos V5 keytab file keytab into the current keylist, then prints the current keylist.

Listing 4: Python

```
from krb5 import ktutil

KEYTAB = "jsmith.keytab"

kt = ktutil()
kt.read_kt(KEYTAB)
kt.list()
kt.quit()
print(kt.keylist)
```

This would return a list containing dictionary objects with keys: slot, kvno and principal.

```
[
    {
        'slot': 1,
        'kvno': 1,
        'principal': 'jsmith@EXAMPLE.COM'
    },
    {
        'slot': 2,
        'kvno': 1,
        'principal': 'jsmith@EXAMPLE.COM'
    }
]
```

Adds an entry to the current keylist using key or password and writes it to a keytab file.

Listing 5: Python

```
from krb5 import ktutil

PRINCIPAL = "jsmith@EXAMPLE.COM"
PASSWORD = "securepassword"
KVNO = 1
ENCTYPE = "aes128-cts-hmac-sha1-96"
ENTRYTYPE = "password" # if "key", PASSWORD must be a passphrase
KEYTAB = "jsmith.keytab"

kt = ktutil()
kt.add_entry(PRINCIPAL, PASSWORD, KVNO, ENCTYPE, ENTRYTYPE)
kt.write_kt(KEYTAB)
kt.quit()
```

Important: Be aware that the `write_kt` method is confusing as the keylist content is appended to the keytab file if it already exists. This is important to be aware of when using `delete_entry` as this will cause duplication if you do not write the keylist to a new file.

Deletes an entry to the current keylist and writes it to a `NEW` keytab file.

Listing 6: Python

```python
from krb5 import ktutil

KEYTAB = "jsmith.keytab"
NEW_KEYTAB = "jsmith_new.keytab"
SLOT = 2

kt = ktutil()
kt.read_kt(KEYTAB)
kt.delete_entry(SLOT)
kt.write_kt(NEW_KEYTAB)
kt.quit()
```

Important: As indicated above, if you invoke `write_kt` on the original keytab file, the current keylist will be appended to the keytab file causing duplication of all entries in the current keylist. For example, your keytab file has 4 entries, then you delete 1, the current keylist still has 3 entries, and the keytab file still has all 4, therefore when invoke `write_kt` the 3 entries from the keylist are `appended` to the keylist file which would cause the keytab to have 6 total entries (4 - 1 + 3 = 6, which 3 being duplicates).