

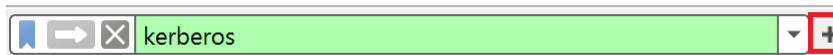
Workbook for the Presentation “Kerberos Deep Dive” at Sharkfest 2024

Task 1: Kerberos Primer

This exercise introduces you to the basic flow of a Kerberos authentication.

You will also create a Wireshark profile that helps with the analysis of Kerberos.

1. Open the trace file `Kerberos_Primer.pcapng`. Verify in the Wireshark status bar that the trace file has 182 packets.
2. You should notice several retransmissions. The retransmissions are an artefact of the recording point: a router connecting network segments 10.1.1.0/24 and 10.1.5.0/24.
3. Create a new profile called Kerberos using the menu or whatever method you prefer **Edit → Configuration Profiles → +**
You can give the profile any name. For this exercise, we assume that the new profile is called Kerberos
4. Apply the display filter `kerberos` and note that the number of display packets in the status bar: 28
5. We want to save this filter for quick access. So, click the plus sign next to the filter bar and label it **Kerberos**



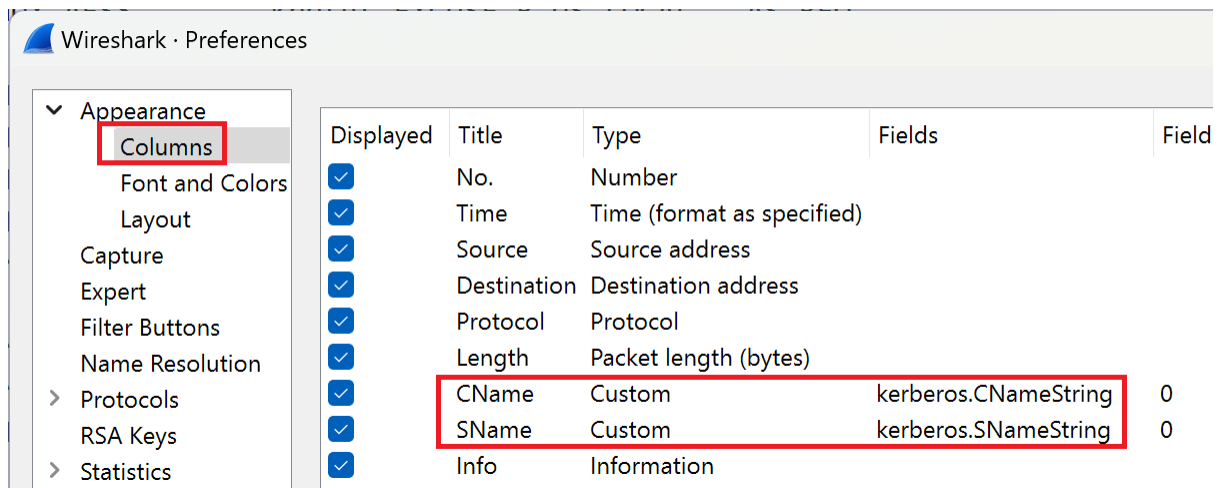
This filter will be updated later.

6. Notice the typical sequence of a Kerberos authentication and authorization in frames 146 to 175:
 - a. An AS-REQ in frame 146 where the client simply asks a TGT without any pre-authentication. Note that the client offers support for the Camellia encryption algorithm. This is one indicator for a Linux-based client.
 - b. The server rejects the request in frame 147, demanding preauthentication. Remarkably, the server indicates support for RC4.
 - ✓ ETYPE - INFO2 - ENTRY

etype: eTYPE-ARCFOUR-HMAC-MD5 (23)

- c. In frames 159 and 160 the client sends an encrypted time stamp and receives a TGT. This exchange uses the AES256.
 - ▼ padata-value: 3041a003020112a23a04388b457d17
 - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
 - cipher: 8b457d17e379cabd423c68cbdc990b58e3
 - d. In frames 170 and 175 the client requests and receives a service ticket for the service principal LINUX-AES\$. This indicates a successful logon to a domain-joined Linux server.
7. It is often helpful, to see the Client or Server Principal in the packet list. Wireshark supports multiple methods to add a new column to the packet list.

Use the menu with **Edit → Preferences ...** and select **Columns** on the left. Now add two columns, choose the type **Custom** and add the fields **kerberos.CNameString** and **kerberos.SNameString**.



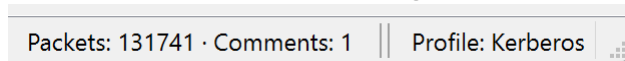
A click on OK will complete the configuration. This new column set is now linked to your Kerberos profile.

Task 2: Kerberos Errors

This exercise shows a few Kerberos errors that can be seen in the trace file

You will also improve our Wireshark profile for Kerberos analysis.

1. Open the trace file Adding_Hosts_to_domain.pcapng. Verify in the Wireshark status bar that the trace file has 131.741 packets.
2. Make sure, that you are still using the Kerberos profile

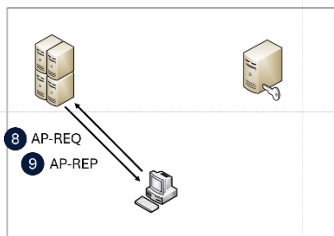


3. The display filter **kerberos** should limit your view to 977 packets. Scrolling through the trace file will show Kerberos messages, that are not transmitted with the Kerberos protocol.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|----------|-------------|----------|--------|------------------------|
| 628 | 22:09:24.756 | 10.1.2.1 | 10.1.1.1 | KRB5 | 185 | TGS-REQ |
| 633 | 22:09:24.756 | 10.1.1.1 | 10.1.2.1 | KRB5 | 1674 | TGS-REP |
| 649 | 22:09:24.757 | 10.1.2.1 | 10.1.1.1 | SMB2 | 591 | Session Setup Request |
| 660 | 22:09:24.757 | 10.1.1.1 | 10.1.2.1 | SMB2 | 314 | Session Setup Response |
| 684 | 22:09:24.761 | 10.1.2.1 | 10.1.1.1 | KRB5 | 298 | AS-REQ |

These packets, like 649 and 660, show in the screenshot contain Kerberos authentication messages. Expanding frame 649 will reveal a Security Blob with a Kerberos message.

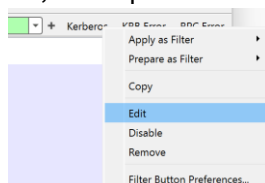
These messages are shown in steps 8 and 9 in the presentation



4. Apply the following display filter focus on just the Kerberos protocol:
kerberos and (tcp.port == 88 or udp.port == 88)

This should bring the number of display packets down to 519

5. You might find the new display filter more helpful than just kerberos. So, let's update the filter. Right-click on the button and select Edit



and change the filter to **kerberos and (tcp.port == 88 or udp.port == 88)**. Save your changes by clicking on OK.

6. Scrolling through your trace file will show several failed Kerberos operations. These can be identified with the following filter:

`kerberos.msg_type == 30`

This filter should reveal 132 packets.

7. The first Kerberos error is “Preauthentication required”. This is not really an error. It is just the client trying to save time by simply asking for a TGT. Since this is the expected behaviour we can filter these errors out. Modify your display filter to `kerberos.msg_type == 30 and not kerberos.error_code == 25`

This should leave 65 packets.

8. Take a few seconds to scroll through the 65 packets. Most errors indicate a more or less serious problem.

“Preauthentication failed” indicates a wrong password and is easy to fix. “Encryption not supported” is more serious and can only be fixed by a configuration change, either for the client principal, the server principal, your GPO or for one of the computers involved.

9. Packet 30839 gives the message “Response too big”. This is another message, that can be ignored. The Kerberos server simply tells the client to switch from UDP to TCP.

To filter out these packets, modify your display filter to

`kerberos.msg_type == 30 and not kerberos.error_code in {25,52}`

This should leave 42 packets indicating real trouble. The filter is worth another button in your profile. Again, click the plus sign next to display filter, enter the label **KRB error** and save the button by clicking OK.

Task 3: Decryption

Next, we generate a Keytab file with the encryption keys and use Wireshark to decrypt the messages.

A Python will generate the keytab file. The handout holds a ready-to-use keytab file, if you don't have a Python interpreter installed.

Option 1: Creating your keytab script from the Github template

1. We use the script `keytab.py`. You will find it on Github at <https://github.com/dirkjanm/forest-trust-tools/blob/master/keytab.py>

The script is a template without keys. For your convenience, the file is part of your downloaded material.

2. In `keytab.py`, lines 112 to 118 hold template keys. This is, where you put the keys for the users, that you want to investigate.

The file `keys_for_keytab.txt` holds all the required lines of code. In `keytab.py`, replace lines 112 to 118 with the content of `keys_for_keytab.txt`.

3. Run the modified script

```
python3 keytab.py sharkfest.keytab
```

Option 2: Use the keytab script from the downloaded material

1. Run the script provided with the tracefiles

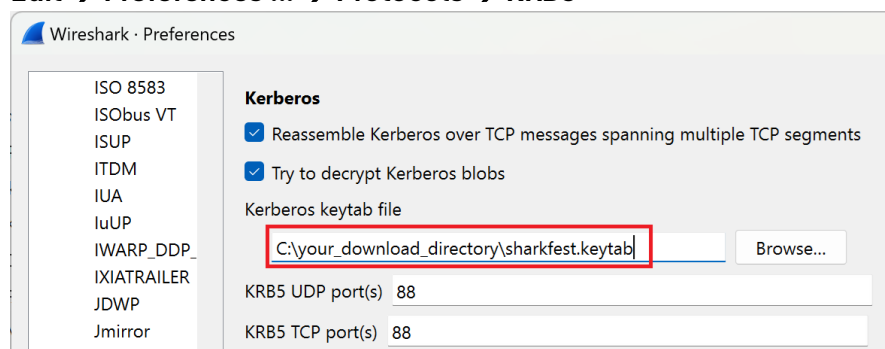
```
python3 keytab.py sharkfest.keytab
```

Option 3: Sit back, relax, enjoy the keytab file provided with trace files.

1. Take a deep breath and continue

Mandatory: Configure your Wireshark profile to support encryption

1. Make sure, that your Kerberos profile is active. Then edit your preferences with **Edit → Preferences ... → Protocols → KRB5**



2. Wireshark should now decrypt the traffic from all the tracefiles that you downloaded.

Optional: Playing with Python:

1. Enter this command in a commandline to see the NTLM Hash for the password Test1234

```
python3 -c 'import hashlib,binascii; print
(binascii.hexlify(hashlib.new("md4", "Test1234".encode("utf-
16le"))).digest())'
```

Note: The command should be entered as a one-liner. It is split for the sake of readability. You might have to configure your OpenSSL, if MD4 is not supported. Check the file /usr/lib/ssl/openssl.cnf. The location might change with your Linux distribution.

```
[provider_sect]
default = default_sect
legacy  = legacy_sect
[legacy_sect]
activate = 1
```

Task 4: Kerberos FAST

In this trace file we check both failed and successful use of Kerberos.

First, let's look at a failed transaction. This error was caused, when the domain controller demanded FAST, but the workstation did not support the configuration.

1. Open the trace file Kerberos_FAST_failed.pcapng and use your filter button **Kerberos**. You should see 84 of 7428 packets.
2. Note that workstation can authenticate to the domain controller
 - a. AS-REQ's will be answered with a TGT, if preauthentication is used.
 - b. Note TGTs issued in frames 569 / 589 and 635 / 639
3. Note that the domain controllers indicates support for Kerberos armoring and claims in a flag found in the AS-REP:

| No. | Time | Source | Destination | Protocol | Length | CName | Info |
|-----|--------------|----------|-------------|----------|--------|-------------|--------|
| 635 | 05:18:08.946 | 10.1.3.1 | 10.1.1.1 | KRB5 | 372 | ws1\$ | AS-REQ |
| 639 | 05:18:08.947 | 10.1.1.1 | 10.1.3.1 | KRB5 | 1781 | WS1\$,WS1\$ | AS-REP |

▼ padata-type: pA-SUPPORTED-ETYPES (165)

▼ padata-value: 1f000500

▼ SupportedEncTypes: 0x0005001f, des-cbc-crc, des-cbc-md5, rc4-hmac, aes128-cts

.....1 = des-cbc-crc: Supported

.....1. = des-cbc-md5: Supported

.....1. = rc4-hmac: Supported

.....1... = aes128-cts-hmac-sha1-96: Supported

.....1... = aes256-cts-hmac-sha1-96: Supported

.....0. = aes256-cts-hmac-sha1-96-sk: Not Supported

.....1 = fast-supported: Supported

.....0. = compound-identity-supported: Not Supported

.....1. = claims-supported: Supported

.....0... = resource-sid-compression-disabled: Not Supported

4. Service Tickets requested in 658 and 673 get a policy error.
5. A user logging in to the workstation (frames 2790, 4042 or 4254) triggers the policy error when requesting a TGT.

Next, let's look at a successful authentication with Kerberos FAST. This walk through assumes, that your Wireshark profile is configured to decrypt Kerberos.

1. Open the trace file Kerberos_FAST_success.pcapng and use your filter button **Kerberos**. You should see 82 of 9465 packets.
2. Notice that the workstation is WS1 is sending multiple requests for the client principal WS1\$. This is the normal behavior for systems running on the computer with the computer account.

3. Starting in frame 4454 the user is logging in. Initially, logons fail due to various reasons
 - a. The account Hella.wahnsinn only supports DES, a detail best detected in the event log.
 - b. The account rainer.wahnsinn does not exist.
 - c. Finally, starting in frame 7372, horst.handshake successfully logs in.
 - d. In frame 8758 we see another successful logon for rainer.unsinn.
4. Let's look at the AS-REQ for horst.handshake. Remember, a few details on the encrypted part: An AES key is derived from the user's password. This key is used to encrypt the current time. So, the security balances on strength of the password.

The armored part of the request uses a key provided in frame 239

| No. | Time | Source | Destination | Protocol | Length | CName | Info |
|---|--------------|----------|-------------|----------|--------|--------------------------|--------|
| 7372 | 06:05:33.390 | 10.1.3.1 | 10.1.1.1 | KRB5 | 700 | WS1\$,WS1\$,horst.han... | AS-REQ |
| 7384 | 06:05:33.391 | 10.1.1.1 | 10.1.3.1 | KRB5 | 2295 | horst.handshake,hor... | AS-REP |
| Used learnt encTicketPart_key in frame 239 keytype 18 (id=239.1 same=24) (699ce5a0...) | | | | | | | |
| [Expert Info (Chat/Security): Used learnt encTicketPart_key in frame 239 keytype 18 (id=239.1 same=24) (699ce5a0...)] | | | | | | | |

Going back to frame 239 we see, that this is the session key provided by the domain controller for the TGT. So, the session key from the workstation's TGT was used to encrypt the users AS-REQ.

Wireshark could only learn the session key, because we extracted the workstations AES key using the DCsync attack. The workstations AES key is derived from a very long and very complex password.