Get started

SPECTEROPS

Attacking FreeIPA — Part I Authentication

n0pe_sled
Nov 25 · 7 min read

Recently I had the opportunity to operate inside of an environment managed by FreeIPA. I wanted to take the time to share some of the lessons I learned about FreeIPA, how it works, and my methodology behind attacking it.

First things first, what is FreeIPA, and why should I care about it?

Well to be honest I had no idea until I actually ran into it in the wild. After a bit of research, I discovered that it is an open source alternative to Microsoft Windows Active Directory, primarily used as an integrated management solution for Unix environments. Similar to Active Directory, FreeIPA implements a full LDAP directory infrastructure backed by an MIT Kerberos Key Distribution Center. It uses the Dogtag Certificate System for CA & RA certificate management, giving it the ability to handle multi-factor authentication, including smartcards. SSSD is used to integrate FreeIPA into the standard Unix authentication process.

So all together we have a Unix host management system, complete with LDAP and Kerberos that allows for multi-factor authentication.

. . .

Due to the sheer amount of content, and in an attempt to make these blog posts more readable I will be splitting them into a series. This post is aimed to cover the following:

files, and CCACHE Tickets stored in memory.

Situational Awareness

Linux hosts enrolled in FreeIPA domains have a few indicators that operators can triage to gain information about the host and the domain. Let's briefly review a few files, environmental variables, and binaries that indicate the host has been enrolled into an FreeIPA domain.

The following files should be on each host enrolled in a Kerberos domain:

/etc/krb5.conf

- The krb5.conf file contains the Kerberos client information required to be enrolled in the domain. This includes the locations of KDCs and admin servers for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of hostnames onto Kerberos realms.

/etc/ipa/default.conf

- This is the default configuration file for IPA servers, it is used to set system-wide defaults to be applied when running IPA clients and servers.

/etc/krb5.keytab

- The krb5.keytab file is required on all hosts inside of the domain. It is required as part of the authentication process to the KDC. By default it allows unrestricted access to its host, and is only readable by root.

. . .

There are also several environment variables that, if set, may indicate the host is enrolled in a Kerberos domain:

KRB5CCNAME

- If set, this variable points to the location of the Keytab file to be used for authentication.

KRB5_CONFIG

- If set, this variable points to the location of the Kerberos configuration file.

KRB5_KDC_PROFILE

- If set, this variable points to the location of the KDC configuration file, which contains additional configuration directives for the Key Distribution Center daemon.

KRB5RCACHETYPE

- This variable specifies the default type of replay cache to use for servers.

KRB5RCACHEDIR

- This variable specifies the default directory for replay caches used by servers.

KRB5_TRACE

- This variable specifies a filename to write trace log output to. Trace logs can help illuminate decisions made internally by the Kerberos libraries.

KRB5_CLIENT_KTNAME

- This variable sets the default client keytab file name.

KPROP_PORT

- This variable sets the default port for kprop to use.

. . .

- This binary is the standard for managing a FreeIPA domain. It can be used to manage hosts, users, sudo rules, and much more.

kdestroy

- The kdestroy binary is used to destroy any current Kerberos tickets in the users session.

kinit

- The kinit binary is used to establish, or renew Kerberos tickets.

klist

- The klist binary lists any current Kerberos tickets in use, and which principals the tickets provide access to.

kpasswd

- The kpasswd command is used to change a Kerberos principal's password. kpasswd first prompts for the current Kerberos password, then prompts the user twice for the new password, and the password is changed.

ksu

- Ksu can be used as an alternative to the su binary, to switch the current user context.

kswitch

- The kswitch command will switch the current credential cache in use.

kvno

- The kvno binary acquires a service ticket for the specified Kerberos principals and prints out the key version numbers of each.

Understanding the underlying technologies and processes that are utilized for authentication and authorization are vital for an attacker. Without understanding each authentication mechanism, an attacker may miss valid credentials that may have enabled lateral movement, or another attack path throughout the environment. Let's briefly review how the authentication process works in an environment managed by FreeIPA.

Since FreeIPA uses Kerberos for authentication, this process is very similar to authentication in Active Directory. In order to access resources on the domain, a user must have a valid Kerberos ticket for that resource. These tickets can be stored in a number of different locations based on the configuration of the FreeIPA domain. I will briefly review each authentication mechanism, covering how to parse and re-use that material from an attackers prospective.

. . .

CCACHE Ticket Files

When tickets are set to be stored as a file on disk, the standard format and type is a CCACHE file. This is a simple binary file format to store Kerberos credentials. These files are typically stored in /tmp and scoped with 600 permissions. From an attackers perspective this is important for the following reasons:

1. Valid tickets can be utilized to authenticate, without the need of the respective users plaintext password.
2. CCACHE tickets are highly portable. They can be downloaded and loaded onto another host without the need to renew, or validate the ticket.

Parsing a CCACHE Ticket is easily accomplished a number of different ways. The simplest method is parsing it with the klist binary.

For an attacker re-using a CCACHE Ticket is very easy. To re-use a valid CCACHE Ticket, export KRB5CCNAME to the path of the valid ticket file. The system should recognize the environment variable and will attempt to use that credential material when interacting with the domain.

Re-using a valid CCACHE Ticket File Found on Disk.

. . .

Unix Keyring

Another option for FreeIPA administrators is to store the CCACHE Tickets inside of the Linux keyring. The keyring lives inside of the kernel, and gives administrators more control over the retrieval and use of stored tickets. Tickets can be scoped in the following different ways:

KEYRING:name

- Tickets are scoped to a specific named Keyring.

KEYRING:process:name

- Tickets are scoped to a specific process id.

KEYRING:thread:name

- Tickets are scoped to a specific user session.

KEYRING:persistent:uidnumber

- Tickets are scoped to a specific user regardless of session.

Depending on how the administrator scoped the ticket stored inside of the Unix keyring, parsing it out may be difficult. However, the default scope for CCACHE Tickets in the Unix keyring is KEYRING:persistent:uidnumber. Fortunately if you are in the context of the user, klist can parse this information for us.

Parsing a CCACHE Ticket Stored in the Keyring.

As an attacker, re-using a CCACHE Ticket stored in the Unix keyring is fairly difficult depending on how the ticket is scoped. Fortunately @Zer1t0 from @Tarlogic has built a tool that can extract Kerberos tickets from the Unix keyring. The tool is called Tickey and can be found here.

. . .

Keytab

Keytabs are another form of credential material that is utilized in FreeIPA, and Kerberos in general. It consists of pairs of Kerberos principals and encrypted keys that are derived from the Kerberos password associated with the principal. Since these keys are derived from the principal's password, if that password changes the keytab will be invalidated.

Keytab files can be used to obtain a valid ticket granting ticket (TGT) for the principal it is scoped to. This authentication process does not require the password, as it contains keys derived from the password.

Parsing a Keytab file is very easy, and can be accomplished a few ways. The easiest way to parse a keytab file is with klist. The second way utilizes a great python utility that Cody Thomas has created. His KeytabParser project will parse out the principal and its relevant encrypted keys.

Parsing KeyTab Files with Klist and KeytabParser

Utilizing a Keytab File to Obtain a Valid CCACHE Ticket.

. . .

Conclusion

It is fundamental for an attacker to understand the operating environment, including various technologies in use and how they are applied. Hopefully this series can serve as a reference for operating inside of environments managed by FreeIPA. Specifically, providing a rudimentary understanding of the underlying technology and how to abuse it from an attackers prospective.

This is the first post in a series of posts documenting what I have learned operating against managed FreeIPA environments. In the following posts I will aim to cover the following:

- A brief overview of the various objects inside of FreeIPA, and how they interact with each other. How to enumerate information about these objects in a FreeIPA environment from the IPA server. Specifically information enabling lateral movement.
- Exploiting an entire Attack Chain in a custom lab environment.
- Finally, an overview of some misconfigurations and unique scenarios that attackers can abuse inside of a FreeIPA environment.

- Kerberos Credential Thievery (GNU/Linux), Ronan Loftus and Arne Zismer, https://www.delaat.net/rp/2016-2017/p97/report.pdf
- Tickey, TarlogicSecurity, https://github.com/TarlogicSecurity/tickey
- KeytabParser, Cody Thomas, https://github.com/its-a-feature/KeytabParser
- Using a Keytab, Indiana University, https://kb.iu.edu/d/aumh

Thanks to Andy Robbins.

Security Freeipa Kerberos Unix Red Team

123 claps

WRITTEN BY

n0pe_sled

Follow

Posts By SpecterOps Team Members

Posts from SpecterOps team members on various topics relating information security

Follow


SPECTEROPS