Secure Network Authentication with Kerberos and OpenLDAP

Franklin E. Diaz
fdiaz@paloaltonetworks.com[1,2]

[1]Palo Alto Networks
[2]Professional Services - Extended Expertise

March 23, 2025

Abstract

The goal of this project paper is to detail my use of MIT Kerberos and OpenLDAP as an authentication mechanism for private cloud computing environments. In addition to how the authentication mechanism is installed and configured, this paper will illustrate ways of using the mechanism to provision user accounts and show some examples of how the framework might be employed. The intent is to create an implementation that can be easily reproduced.

1 Secure Network Authentication with Kerberos and OpenLDAP

Secure authentication is a primary concern in multi-user networks. Simply put, network administrators are tasked with ensuring valid users are able to access network resources, and the rest of the world is not. For this project, MIT Kerberos[?][?] provides a secure authentication mechanism, using OpenLDAP schemas for defining user accounts.

2 Project Goals

There are several goals related to this project. These goals overlap and interconnect at times. The overarching goal is to understand the composition, management, and weaknesses of the items listed here.

1. Normalize the installation and use of secure authentication methodology using Open Source software.
2. Move away from SSH keys and complicated "key management" schemes.
3. Provide a working example to make this authentication paradigm easier to reproduce.

There are websites and documentation that detail installation and some configuration tasks related to this authentication paradigm. It can be difficult to find practical examples and detailed demonstration of usage and usability.

3 The Lab Environment

This section is a brief description of the "Private Cloud" lab environment in which the project was built and deployed.

1. Single Board Compute (SBC) devices were chosen for their low power consumption/heat output and silent operation when possible.
2. Cluster nodes are meant to be added and rebuilt quickly and easily. Use Ansible and other automation whenever possible.

The core of the Kerberos authentication implementation is the Key Distribution Center (KDC). In this lab configuration, the KDC is running on an O-droid C1 running Ubuntu.

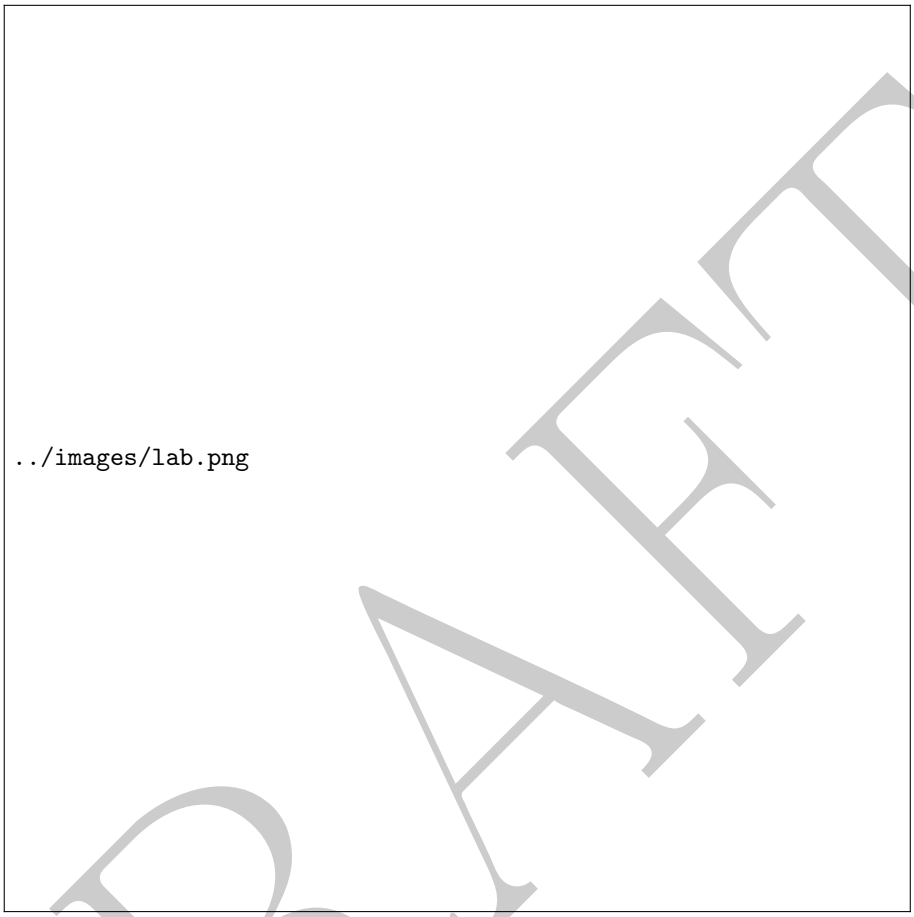

Figure 1: The project environment used for testing.

A Palo Alto Networks VM-220 is used to pass traffic in and out of the cluster. The firewall has also been configured to serve DHCP to clients, though this service could be placed elsewhere in the network with minimal effort.

3.1 [DNS Configuration]

It is possible for clients to locate a KDC through URI records in DNS. While the MIT Kerberos Documentation does provide a good explanation and an example configuration, some changes were needed to make the feature behave correctly with Debian 10 Buster and BIND9 with private/local DNS servers.

3.1: Configuring DNS for Kerberos

```
;KERBEROS Records
$ORIGIN lab.bitsmasher.net.
_kerberos                TXT      "LAB.BITSMASHER.NET"
_kerberos._udp           SRV      0 0 88 odroid-c1
_kerberos-master._udp    SRV      0 0 88 odroid-c1
_kerberos-adm._tcp       SRV      0 0 749 odroid-c1
_kpasswd._udp            SRV      0 0 464 odroid-c1
_kerberos                URI     10 1 "krb5srv:m:tcp:odroid-c1.lab.bitsmasher.net"

;LDAP Records
_ldap._tcp               IN SRV 10 50 389 server3
```

Validation of the proper DNS configuration can be performed via command line.

3.2: Verifying DNS for Kerberos

```
franklin@server1:~ $ dig srv _kerberos._udp.lab.bitsmasher.net | grep -v ';;'

; <<>> DiG 9.11.5-P4-5.1+deb10u5-Raspbian <<>> srv _kerberos._udp.lab.bitsmasher.net
;_kerberos._udp.lab.bitsmasher.net. IN  SRV
_kerberos._udp.lab.bitsmasher.net. 10800 IN SRV 0 0 88 odroid-c1.lab.bitsmasher.net.
odroid-c1.lab.bitsmasher.net. 10800 IN  A       10.10.12.254

franklin@server3:/etc/bind $ dig ANY _ldap._tcp.lab.bitsmasher.net @10.10.13.1 | grep net

; <<>> DiG 9.11.5-P4-5.1+deb10u5-Raspbian <<>> ANY _ldap._tcp.lab.bitsmasher.net @10.10.13.1
;_ldap._tcp.lab.bitsmasher.net. IN      ANY
_ldap._tcp.lab.bitsmasher.net. 10800 IN SRV     10 50 389 server3.lab.bitsmasher.net.
lab.bitsmasher.net.     10800   IN      NS      odroid-c1.lab.bitsmasher.net.
server3.lab.bitsmasher.net. 10800 IN    A       10.10.13.1
odroid-c1.lab.bitsmasher.net. 10800 IN  A       10.10.12.254
```

3.2 [NTP Configuration]

A reliable source of time for the network is critical for this project. An example NTP configuration via Ansible playbook has been included for reference. While it is possible to synchronize time via the HTTP network time protocol, it is highly discouraged for this authentication methodology.

4 Kerberos

4.1 [Kerberos Server]

Kerberos KDC servers are configured using the files listed in the following table. Examples are provided in the "ansible/" folder in source repository of this paper.

1. /etc/krb5.conf
2. /etc/krb5kdc/kadm5.acl
3. /etc/krb5kdc/kdc.conf

4.2 [Kerberos Client]

Client configurations tend to be a bit more complex than configuring the servers simply because there are more touch points.

4.3 [Principals and Keytabs]

A Kerberos principal is a user or service upon with certain access related entitlements are bestowed. Administrators use the "kadmin" command to define the user and service principals. Consider the following typical provisioning session.

4.1: Keytab Creation

```
sudo -i
kinit root/admin
kadmin
addprinc -randkey host/head1.lab.bitsmasher.net
addprinc -randkey ldap/head1.lab.bitsmasher.net
ktadd host/grimoire.lab.bitsmasher.net ldap/grimoire.lab.bitsmasher.net
(quit)
klist -ke /etc/krb5.keytab
```

The keytabs present on any given host in the network will look similar to the following code block. Note that the "kadmin" command is run on the host where the keytab file should ultimately reside. This allows for the "klist" command to successfully show the "host" and "ldap" principals in the output.

4.2: Viewing Keytabs

```
franklin ~: sudo klist -ke /etc/krb5.keytab

Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   2 host/thelio.lab.bitsmasher.net@LAB.BITSMASHER.NET (aes256-cts-hmac-sha1-96)
   2 host/thelio.lab.bitsmasher.net@LAB.BITSMASHER.NET (aes128-cts-hmac-sha1-96)
   2 ldap/thelio.lab.bitsmasher.net@LAB.BITSMASHER.NET (aes256-cts-hmac-sha1-96)
   2 ldap/thelio.lab.bitsmasher.net@LAB.BITSMASHER.NET (aes128-cts-hmac-sha1-96)
```

5 OpenLDAP

The OpenLDAP portion of the configuration serves as a backend storage for details about user accounts. User accounts are stored as objects with unique "distinguished names" (DNs).

5.1 [Supported SASL Mechanisms]

5.1: Viewing the Support SASL Mechanisms

```
franklin@server3:~ $ ldapsearch -x -b '' -s base supportedSASLMechanisms -LLL
dn:
supportedSASLMechanisms: SCRAM-SHA-1
supportedSASLMechanisms: SCRAM-SHA-256
supportedSASLMechanisms: GS2-IAKERB
supportedSASLMechanisms: GS2-KRB5
supportedSASLMechanisms: GSS-SPNEGO
supportedSASLMechanisms: GSSAPI
supportedSASLMechanisms: DIGEST-MD5
supportedSASLMechanisms: NTLM
supportedSASLMechanisms: CRAM-MD5
```

The "/etc/nsswitch.conf" file is used to define the type and order of the methods that will be used for lookups on things like the password, group, etc for a user. One the OpenLDAP server is installed and configured properly, the nsswitch file is updated so the host will perform lookups against the LDAP database.

5.2: The /etc/nsswitch.conf file

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch
#  functionality.

passwd:         compat systemd ldap
group:          compat systemd ldap
shadow:         compat
gshadow:        files

hosts:          files mdns4_minimal [NOTFOUND=return] dns
networks:       files

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

netgroup:       nis
```

6 Integrations

An application that is configured to work with the Kerberos authentication framework is said to be "Kerberized".

6.1 [Kerberized SSH]

Integration of Secure Shell (SSH) with Kerberos is a common use case. Attackers are actively looking for SSH keys that they can exploit[?]. SSH keys and other sensitive credentials are often inadvertently committed to revision control systems by developers. These can be quite difficult to remove from the history of a repository and may even necessitate deleting and recreating the repository. Leaving it up to the end user to opt-in to the SSH key management initiatives outlined by IT or Security arm of the organization brings risk. Centralization of user and password management means reducing this risk.

Updates to the "/etc/ssh/ssh_config" and "/etc/ssh/sshd_config" files are made to allow Kerberos interactions with SSH on the incoming and outgoing sessions respectively. Modification of the user configuration can be done by the user or enforced via Ansible playbook to ensure compliance. Settings in the aforementioned server level configuration may be overridden if permitted.

6.1: User's ${HOME}/.ssh/config file

```
Host *.lab.bitsmasher.net
  User "franklin"
  PreferredAuthentications gssapi-keyex,gssapi-with-mic
  GSSAPIAuthentication yes
  GSSAPIDelegateCredentials yes
  CanonicalizeHostname yes
  CanonicalDomains lab.bitsmasher.net
  CanonicalizeMaxDots 3
  CanonicalizeFallbackLocal yes
```

Once a user has successfully authenticated to the KDC using the kinit command, hosts can be configured to allow logins with no password.

6.2: Obtaining a ticket from the KDC.

```
franklin ~: klist
klist: No credentials cache found (filename: /tmp/krb5cc_1000)
franklin ~: kinit
Password for franklin@LAB.BITSMASHER.NET:
franklin ~: klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: franklin@LAB.BITSMASHER.NET

Valid starting       Expires              Service principal
10/21/2021 08:25:16  10/21/2021 18:25:16  krbtgt/LAB.BITSMASHER.NET@LAB.BITSMASHER.NET
        renew until 10/22/2021 08:25:05
franklin ~:
```

The project support authentication using GSS-API user authentication as well as authentication using GSS-API key exchange as described in RFC 4462[?].

The next figure shows the debug from a typical SSH session with the "gssapi-with-mic" authentication method enabled. Note that the "-K" flag is typically not required since we've specified our setup in "${HOME}/.ssh/config".

6.3: SSH Connection with Kerberos

```
franklin ~: ssh -K server1 -v
(0.012475 hrs)
OpenSSH_8.4p1 Ubuntu-5ubuntu1.1, OpenSSL 1.1.1j  16 Feb 2021
debug1: Connecting to server1 [10.10.12.12] port 22.
debug1: Skipping ssh-dss key /home/franklin/.ssh/id_dsa - not in PubkeyAcceptedKeyTypes
debug1: Authentications that can continue: gssapi-keyex,gssapi-with-mic
debug1: Next authentication method: gssapi-with-mic
debug1: Delegating credentials
debug1: Authentication succeeded (gssapi-with-mic).
Authenticated to server1 ([10.10.12.12]:22).
Linux server1.lab.bitsmasher.net 5.10.73-v7+ #1470 SMP Mon Oct 18 11:25:58 BST 2021 armv7l

Welcome to the lab cluster: server1

Last login: Thu Oct 21 12:13:30 2021 from 10.10.12.17

franklin@server1:~ $
```

A similar method could be used to enable/validate the GSS-API key exchange method as described in the reference document[?].

6.2 [Ansible Integration]

Ansible playbooks can be used to deploy and manage the entire authentication ecosystem, be it in public/private networks or cloud environments. The Ansible inventory is used to group and manage your hosts in ways that make sense for your network.

Once the KDC and OpenLDAP servers are running, you can authenticate your Ansible plays against the KDC. Note the specification of the Kerberos Realm in the Ansible hosts file. The realm is appended to each host in all capital letters.

6.4: Authorizing Ansible via Kerberos

```
[servers]
odroid-c1.LAB.BITSMASHER.NET ansible_user=root
server1.LAB.BITSMASHER.NET
server2.LAB.BITSMASHER.NET
server3.LAB.BITSMASHER.NET

[storage]
storage1.LAB.BITSMASHER.NET

[raspi_cluster]
head1.LAB.BITSMASHER.NET
node0.LAB.BITSMASHER.NET
node1.LAB.BITSMASHER.NET
node2.LAB.BITSMASHER.NET
node3.LAB.BITSMASHER.NET

[nvidia_cluster]
node900.LAB.BITSMASHER.NET
node901.LAB.BITSMASHER.NET
```

6.3 [Pan OS Integration]

Once the KDC is up and the user has been provisioned with an LDAP account, it is trivial to integrate with Palo Alto Networks security devices running PANOS. The user accounts are available on the device after entering the details about the KDC.

6.4 [Container Integration]

Users are able to authenticate against the KDC from a container by simply copying in the "/etc/krb5.conf" file and then running "kinit" as usual.

A full working example Dockerfile has been included for testing and proof of concept. Consider the following execution provided for demonstration purposes.

6.5: Authenticating against KDC from a Container

```
franklin : docker build -t franklin:krb-ldap \
             --build-arg BUILD_DATE=(date -u +'%Y-%m-%dT%H:%M:%SZ') .
franklin : docker image ls | grep krb-ldap
franklin : docker run --rm -it franklin:krb-ldap /bin/ash

franklin : docker run --rm -it franklin:krb-ldap /bin/ash
/go # su - franklin
24e52ca4c260:~$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1000)
24e52ca4c260:~$ kinit
Password for franklin@LAB.BITSMASHER.NET:
24e52ca4c260:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: franklin@LAB.BITSMASHER.NET

Valid starting     Expires            Service principal
10/21/21 20:13:47  10/22/21 06:13:47  krbtgt/LAB.BITSMASHER.NET@LAB.BITSMASHER.NET
        renew until 10/22/21 20:13:39
24e52ca4c260:~$
```

7 Summary

This project paper demonstrates the configuration and capabilities of Kerberos and OpenLDAP, and outlines the reasoning behind the need for adopting a centralized mechanism for authentication.

7.1 [Future Direction]

1. Containerize the KDC and LDAP servers and move the entire ecosystem into Kubernetes.
2. Integration of Kerberos with NFS and SSSD

Revision History

Revision	Date	Author(s)	Description
v0.1	March 18th, 2014	Franklin Diaz	Initial Draft
v0.2	Sept. 29th, 2021	Franklin Diaz	Updates