More-zero-trust-ish SSH

Duane Waddle

@duckfez

BSides Nashville 2022

So you've got Linux servers and...

- Need to authenticate interactive logins (ssh)
- You'd like to have:
  - centralized user management
  - strong authentication
  - Reasonably secure
  - Compliance Requirements Met
  - "zero trust" (whatever that is!)

Before we go there, some background

- 1970s Unix heritage
- Evolution of "User Management"
  - /etc/passwd
  - /etc/shadow
  - PAM
- Evolution of "logging in"
  - Serial terminals
  - Telnet
  - SSH
- Evolution of "Privilege Management"
  - su
  - sudo

/etc/{passwd,shadow,group}

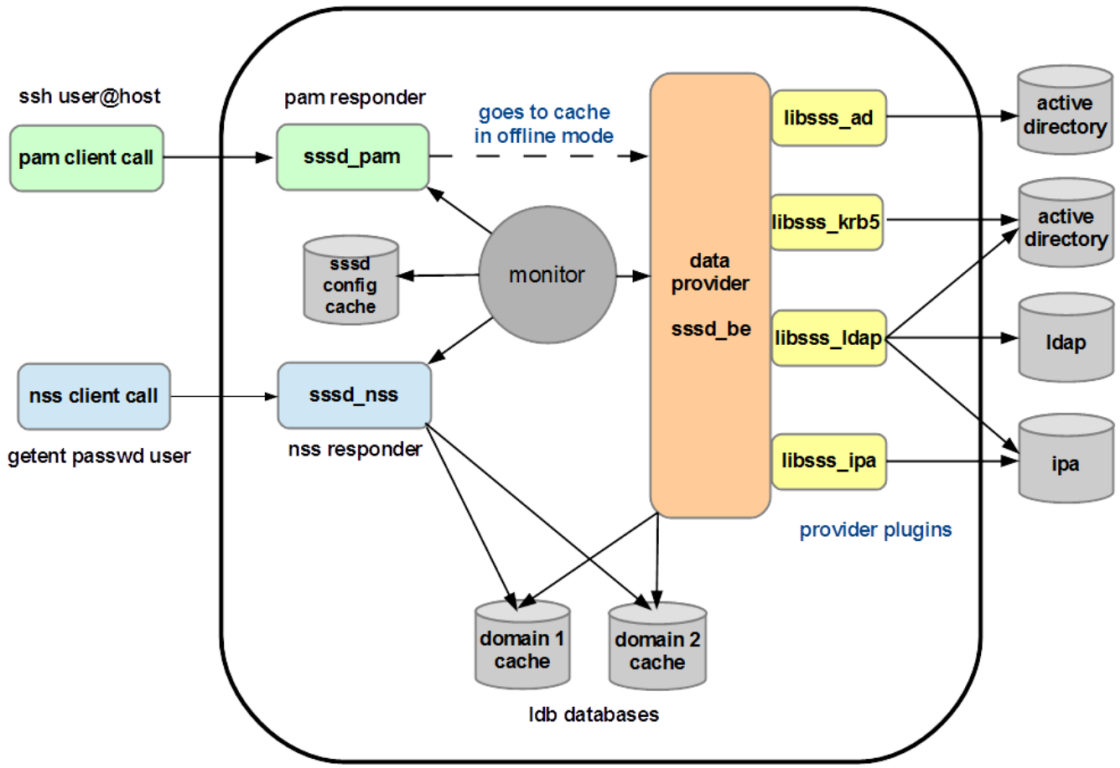
- Text files delimited by ":"
- "x" in passwd means "password is in shadow"
- 6 = Salted SHA512 password hash
- "Local only" - not very modular

```
[root@plaindemo ~]# grep duane /etc/passwd
duane:x:1000:1000:Cloud User:/home/duane:/bin/bash
[root@plaindemo ~]#
[root@plaindemo ~]# grep duane /etc/shadow
duane:$6$y63UXOQKAexvVep4$cvhqOx6LB94QZXqlk1xBHKPjjbyJ5cEEhAfPOz/awbYYAMEyTkTtVtbY1L4iSnUszNLLl.Ip/8B1kG9q3AM.t0:19022:0:99999:7:::
[root@plaindemo ~]#
[root@plaindemo ~]# grep duane /etc/group
adm:x:4:duane
systemd-journal:x:190:duane
duane:x:1000:
[root@plaindemo ~]#
```

NSS, PAM, and SSSD (oh my!)

- Approaches to modularizing /etc/{passwd,shadow,group}
- Dynamically Linked APIs and daemon processes (sssd)
- NSS – controls lookups in databases
  - "does a user named duane exist? What groups is duane a member of?"
- PAM – handles authentication / authentication-time things
- SSSD – daemon to hook NSS & PAM APIs in a "modern" way

SSSD Architecture



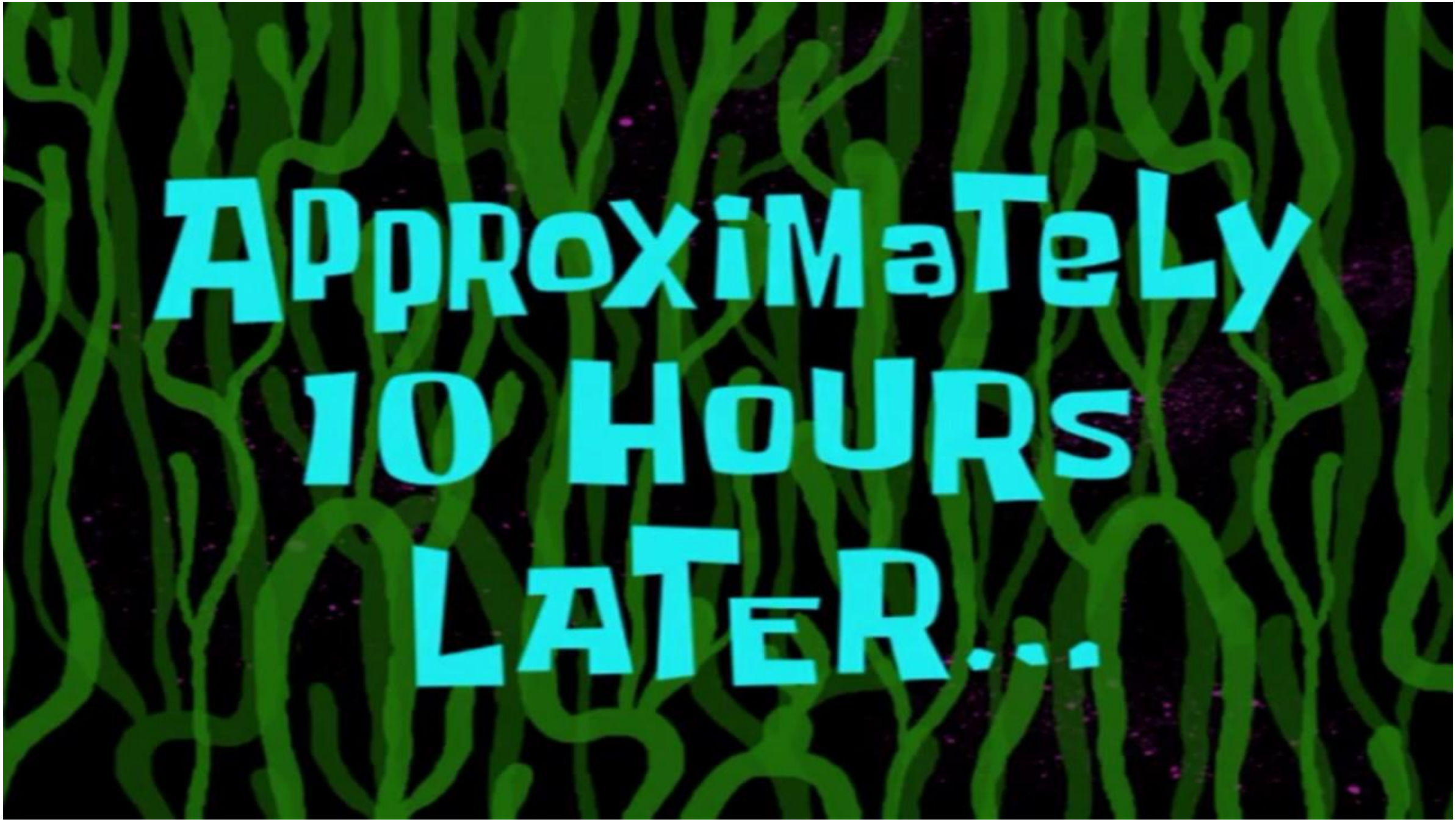
https://community.cloudera.com/t5/Community-Articles/How-PAM-NSS-SSD-work-together-on-Linux-OS/ta-p/247879

Terminal access

- 70s and 80s – hardwired serial terminal (teletype!)
- TCP/IP brought telnet (RFC 855 in 1983)
  - "virtual terminal" over arbitrary IP network (yay!)
  - Everything is cleartext across the network (boo)
- SSH "secure shell" (mid 90s)
  - Encrypted transport
  - More flexible authentication
    - Username / password
    - Username / private key
    - Kerberos and others

su and sudo

- Escalate non-root user to root
- "wheel" group
- su – know root's password, get a root shell
- sudo – run some commands as root flexibly
  - Policy based execution
  - Passwords optional
  - Your password, not root's


APPROXIMATELY
10 HOURS
LATER...

Tie into existing AD?

- Assuming you have one
- Dozens of ways of doing it
- Per-server authorization is hard
- MFA can be messy
- Painful Smart Card (PIV) integration
- Not very "zero trust"
  - Passwords easily stolen
  - Many AD attacks may work (PTH, etc)
  - Power users will make static RSA keys

About them static RSA keys

- Pros:
  - pretty strong authentication
  - supports passwordless logins
- Cons:
  - Hard to enforce key lifecycle
  - Attackers like to steal them
  - Hard to enforce compliance (users don't have to set a passphrase and you can't make them)
  - ssh-agent can be a big security risk

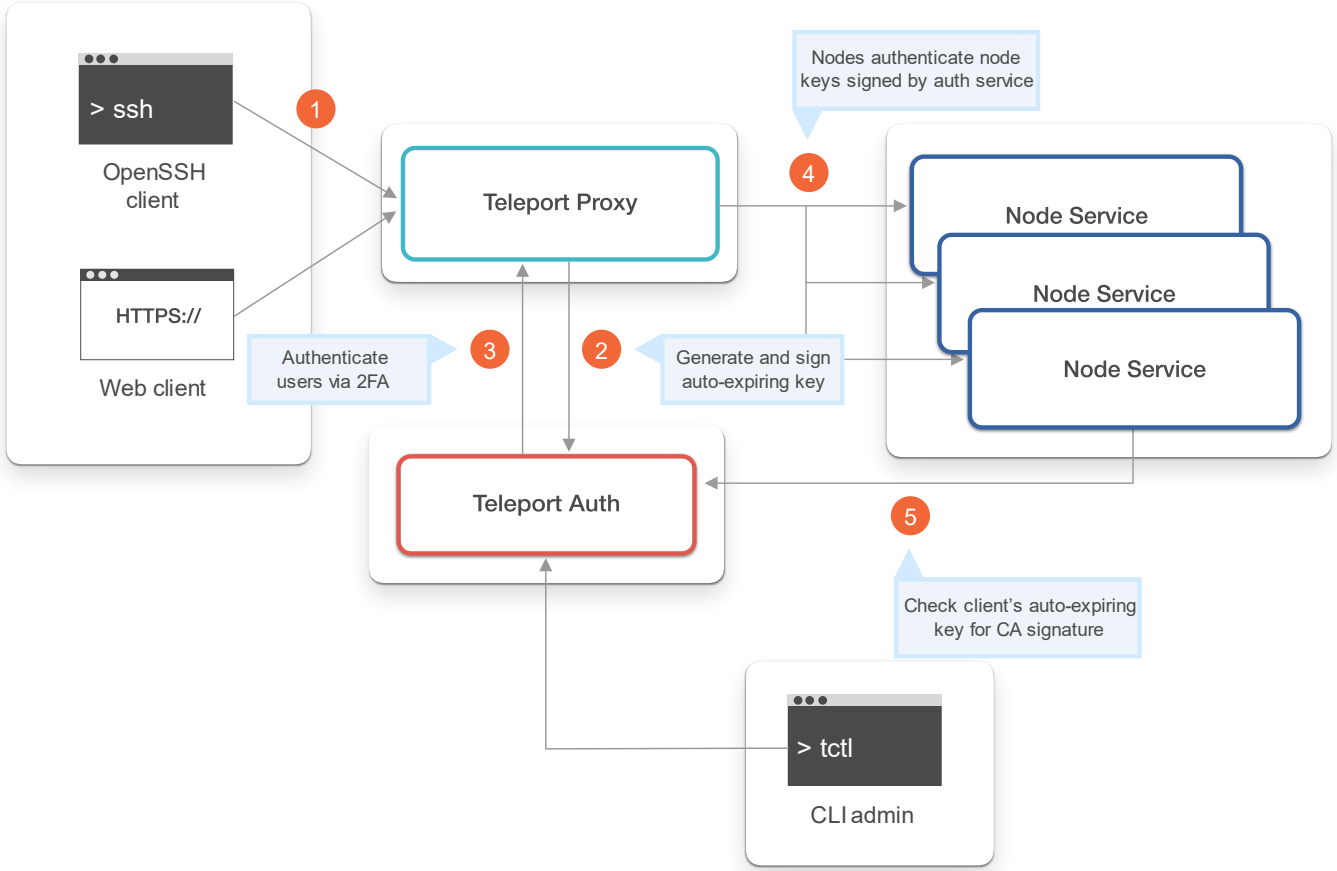
Goals / Requirements

- Strong authentication
- No static RSA keys
- Tied to central directory
- Strong MFA
- Lots of compliance
- No listening ports

How we did it

- No Active Directory (all Linux environment)
- SAML IDP (Okta / Azure AD)
- Teleport (www.goteleport.com)
- (working on) Yubikeys for Smart Card/PIV

Teleport Architecture



https://goteleport.com/docs/architecture/overview/#detailed-architecture-overview

How it works

- Stand up a teleport proxy / auth server
- Nodes connect to proxy (optional reverse tunnel mode)
- Define users & roles in teleport
- Users install tsh on their clients
- tsh commands replace ssh commands
  - tsh login
  - tsh ssh
  - tsh scp
  - … others

Nodes

- Nodes run teleport instead of sshd (or alongside for both?)
- Nodes are enrolled with the auth server / proxy
- Nodes have labels, used for RBAC
  - Static
  - Dynamic (run command periodically, use output as labels)
- Standard mode vs tunnel is less than obvious

```
[duane@targetserver1 ~]$ sudo cat /etc/teleport.yaml
version: v2
teleport:
  auth_servers:
    - teleport.duanewaddle.com:443
auth_service:
  enabled: no
proxy_service:
  enabled: no
ssh_service:
  enabled: "yes"
  pam:
    enabled: yes
    service_name: "teleport"
  labels:
    env: example
  commands:
  - name: hostname
    command: [hostname]
    period: 1m0s

[duane@targetserver1 ~]$
```

Enrolling nodes

- Auth server makes a token
- Client uses that token to auth itself to the auth server / proxy

```
[root@teleport-master ~]# tctl tokens add --type=node --ttl=24h
The invite token: 9570e69ed5ef1a8ebcd27f1c11694fdc.
This token will expire in 1440 minutes.

Run this on the new node to join the cluster:

> teleport start \
   --roles=node \
   --token=9570e69ed5ef1a8ebcd27f1c11694fdc \
   --ca-pin=sha256:3dd6e54cd6f45748098edcf5ed30942aa1b0387cbb379b1691a45404b78e732e \
   --auth-server=10.3.0.4:3025

Please note:

  - This invitation token will expire in 1440 minutes
  - 10.3.0.4:3025 must be reachable from the new node
[root@teleport-master ~]#
```

```
[root@target2 ~]#  /usr/local/bin/teleport start    --roles=node    --token=9570e69ed5ef1a8ebcd27f1c1
1694fdc    --ca-pin=sha256:3dd6e54cd6f45748098edcf5ed30942aa1b0387cbb379b1691a45404b78e732e    --auth
-server=teleport.duanewaddle.com:443
2022-01-28T17:00:44Z INFO             Generating new host UUID: 879ce945-1842-4378-8491-66dae86c1432.
 service/service.go:659
2022-01-28T17:00:44Z INFO [PROC:1]    Joining the cluster with a secure token. service/connect.go:362
2022-01-28T17:00:44Z INFO [AUTH]      Attempting registration via proxy server. auth/register.go:160
2022-01-28T17:00:44Z INFO [AUTH]      Successfully registered via proxy server. auth/register.go:167
2022-01-28T17:00:44Z INFO [PROC:1]    Node has obtained credentials to connect to the cluster. servic
e/connect.go:391
2022-01-28T17:00:44Z INFO [PROC:1]    The process successfully wrote the credentials and state of Nod
e to the disk. service/connect.go:432
2022-01-28T17:00:44Z INFO [PROC:1]    Node: features loaded from auth server: Kubernetes:true App:tru
e DB:true  service/connect.go:59
2022-01-28T17:00:44Z INFO [NODE:1:CA] Cache "node" first init succeeded. cache/cache.go:676
2022-01-28T17:00:44Z INFO [AUDIT:1]   Creating directory /var/lib/teleport/log. service/service.go:21
05
```

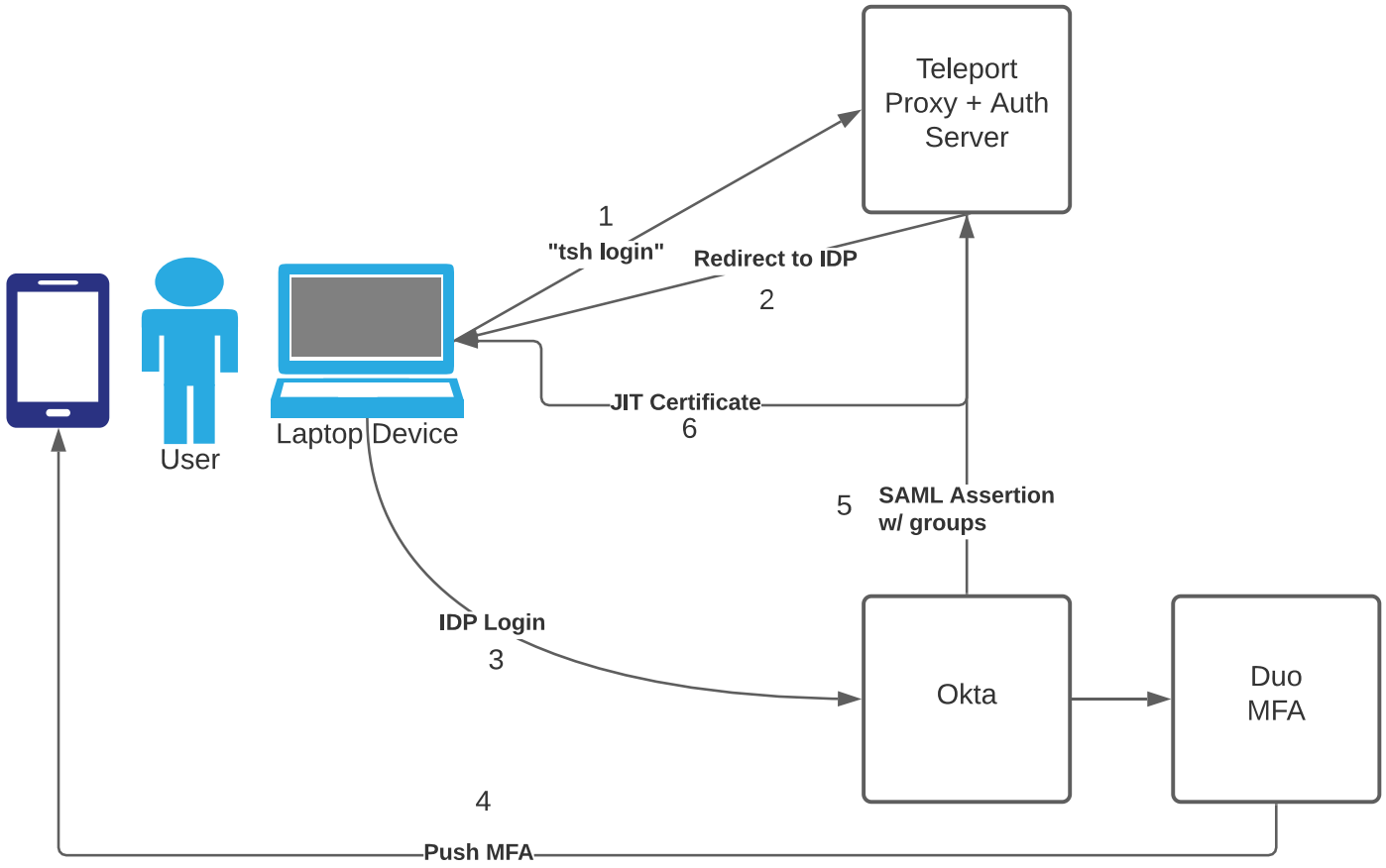
All nodes register to auth server

```
[root@teleport-master ~]# tctl nodes ls
Nodename        UUID                                 Address           Labels
--------------- ------------------------------------ ----------------- ----------------------------------------------
teleport-master 38693225-1e78-45ad-a24a-9d0220d9fd85 127.0.0.1:3022    env=example,hostname=teleport-master
targetserver1   5ee7ab29-55ff-46f7-8ae7-d4bed337e302 52.188.67.65:3022 cloud=azure,env=example,hostname=targetserver1
target2         879ce945-1842-4378-8491-66dae86c1432                   cloud=azure,env=example,hostname=
[root@teleport-master ~]#
```

tsh login

- Logs a user into a teleport proxy – not into a specific host
- Auth server mints a user-specific short-lived certificate
- Certificate has user authorization attributes:
  - Roles
  - Logins
  - SSH extensions
- Proxy authentication can be robust
  - User+Password w/ MFA (TOTP or FIDO2)
  - Federated Login (Github Org/SAML/OpenID)

"tsh login" visualized



tsh login

```
dwaddle@rick:~$ tsh login
Enter password for Teleport user duane.waddle:<hunter2>
Tap any security key
> Profile URL:        https://teleport.duanewaddle.com:443
  Logged in as:       duane.waddle
  Cluster:            teleport.duanewaddle.com
  Roles:              access, group_wheel
  Logins:             duane.waddle, root
  Kubernetes:         enabled
  Valid until:        2022-01-26 09:59:33 -0600 CST [valid for 12h0m0s]
  Extensions:         permit-agent-forwarding, permit-port-forwarding, permit-pty
```

Also in the cert!

```
dwaddle@rick:~/.tsh/keys/teleport.duanewaddle.com$ openssl x509 -text -in duane.waddle-x509.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5a:12:12:b9:fe:35:92:f0:e1:c7:53:bc:6a:83:73:05
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O = teleport.duanewaddle.com, CN = teleport.duanewaddle.com, serialNumber =
  44034826603267495139198678937413329882
        Validity
            Not Before: Jan 10 02:27:01 2022 GMT
            Not After : Jan 10 14:28:01 2022 GMT
        Subject: L = root + L = duane.waddle, street = teleport.duanewaddle.com, postalCode =
  "{\"kubernetes_groups\":[\"\"],\"kubernetes_users\
":[\"\"],\"logins\":[\"duane.waddle\"],\"windows_logins\":null}", O = access + O = group_wheel,
  CN = duane.waddle, 1.3.9999.1.7 = teleport.duanewaddle.com
```

Roles for per-server authorization

- It's YAML!
- logins: who you can log in as
- node_labels: where you can log into

  label not *hostname*

access-targetserver1

SPEC

```yaml
kind: role
metadata:
  description: Access to targetserver1
  id: 1643254618575932221
  name: access-targetserver1
spec:
  allow:
    logins:
    - '{{internal.logins}}'
    - root
    node_labels:
      hostname: targetserver1
    rules:
    - resources:
      - event
      verbs:
      - list
      - read
  deny: {}
  options:
    cert_format: standard
    enhanced_recording:
```

tsh ls

- Once I'm logged in, I can see my nodes
- Address is important:
  - IP = node is listening, proxy connects to the node
  - <- Tunnel = node connected outbound to proxy, no listening port at all!

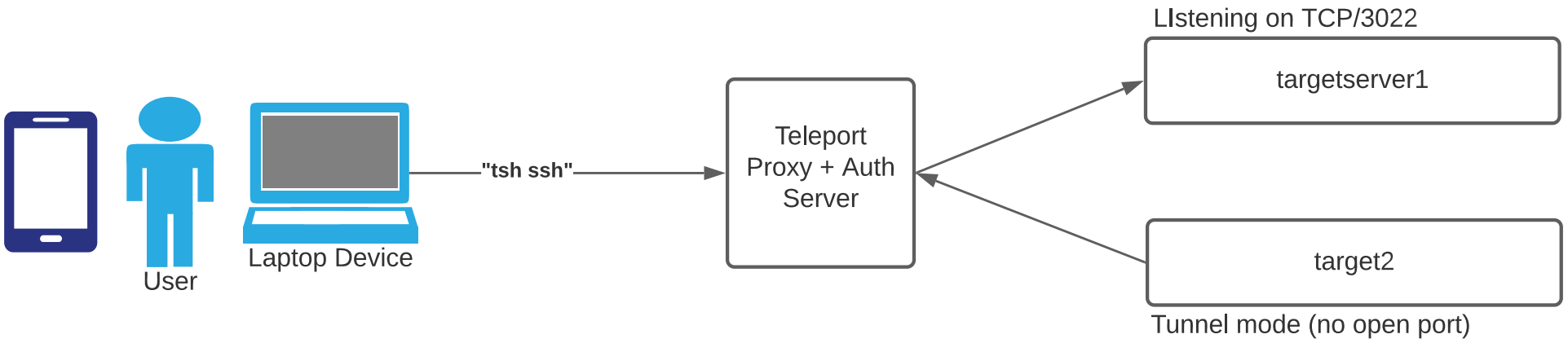
```
dwaddle@rick:~$ tsh ls -f text
Node Name     Address            Labels
------------- ------------------ ------------------------
target2       ← Tunnel           cloud=azure, env=example
                                 hostname=target2
targetserver1 52.188.67.65:3022  cloud=azure, env=example
                                 hostname=targetserver1

dwaddle@rick:~$
```


```
[root@targetserver1 ~]# netstat -pant | egrep -i LISTEN
tcp        0      0 0.0.0.0:5355            0.0.0.0:*               LISTEN      875/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1284/sshd
tcp6       0      0 :::5355                 :::*                    LISTEN      875/systemd-resolve
tcp6       0      0 :::3022                 :::*                    LISTEN      7302/teleport
tcp6       0      0 :::22                   :::*                    LISTEN      1284/sshd
[root@targetserver1 ~]#
```

```
[root@target2 ~]# netstat -pant | egrep -i LISTEN
tcp        0      0 0.0.0.0:5355            0.0.0.0:*               LISTEN      876/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1249/sshd
tcp6       0      0 :::5355                 :::*                    LISTEN      876/systemd-resolve
tcp6       0      0 :::22                   :::*                    LISTEN      1249/sshd
[root@target2 ~]#
```

tsh ssh

- tsh ssh root@targetserver1
- Workstation -> proxy, proxy -> targetserver1
- Teleport validates session is allowed
- And records session data in fulltext
- Only proxy needs to be "exposed"



Intermission … how is this ZT?

- Note: I am not a ZT expert...

- Strong authentication
- Granular access control
- Time-boxed credentials (Just in time access)
- Allows ssh access w/o VPN, without network trust
- Ties into existing enterprise IAM w/o legacy protocols
- Minimal exposed listening ports / smaller attack surface

New users, new problems

- tsh ssh root@<server> works great
- tsh ssh duane.waddle@<server> not so good
- No synchronization of user directory
- Workarounds?

No, not that Pam .. PAM

- Linux Pluggable Authentication Modules
- Use pam_exec.so to call a script during "account" phase.
- Script handles:
  - Creating the user if they do not exist
  - Mapping teleport role assignments to unix groups
  - Synchronizing user's unix groups to mapped roles (add and remove)
  - Disabling (Unix) password expiration for the user (WHY??)

```
[duane.waddle@targetserver1 ~]$ cat /etc/pam.d/teleport
account   required   pam_exec.so /etc/pam-exec.d/teleport_acct
account   required   pam_permit.so
session   required   pam_motd.so motd=/etc/issue.net
session   required   pam_loginuid.so
session   required   pam_permit.so
[duane.waddle@targetserver1 ~]$
```

Results

```
# Users that this script should ignore
USERS_TO_SKIP = [ 'root', 'centos', 'ec2-user', 'ubuntu' ]

# Teleport roles are mapped to unix groups
ROLES_TO_GROUPS_MAP = {
    "access"     : [ "teleport" ],
    "group_wheel" : [ "teleport", "wheel" ],
}
```

```
[root@targetserver1 ~]# grep duane.waddle /etc/passwd
duane.waddle:x:1004:1004::/home/duane.waddle:/bin/bash
[root@targetserver1 ~]# grep duane.waddle /etc/shadow
duane.waddle:!!:19020:0:99999:7:::
[root@targetserver1 ~]# id duane.waddle
uid=1004(duane.waddle) gid=1004(duane.waddle) groups=1004(duane.waddle),10(wheel),984(teleport)
[root@targetserver1 ~]# chage -l duane.waddle
Last password change                                    : Jan 28, 2022
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
[root@targetserver1 ~]#
```

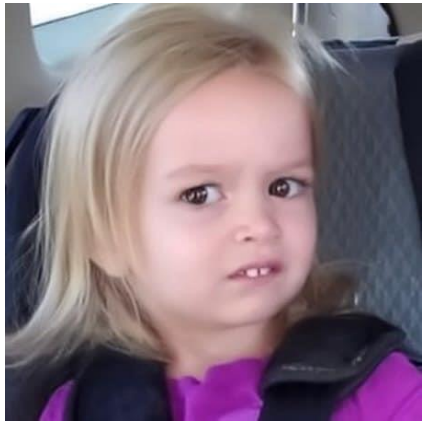
Make me a sandwich

- New problem – sudo

```
[duane.waddle@targetserver1 ~]$ sudo whoami

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for duane.waddle:
```

- One "Solution?" NOPASSWD

```
## Allows people in group wheel to run all commands
#%wheel ALL=(ALL)       ALL

## Same thing without a password
%wheel  ALL=(ALL)       NOPASSWD: ALL
```



Sudo without a password without NOPASSWD

- Privilege Escalation problem here
- Need a way to challenge the user running sudo
  - Make sure it's actually them, not an attacker
  - Make sure they "meant to do that"
- This sounds like … MFA?
- Like most things in Linux, sudo uses PAM
- So I need a PAM module … that can authenticate someone … without a password

pam_duo to the rescue

- From the docs:

```
PAM_DUO(8)                    BSD System Manager's Manual                    PAM_DUO(8)

NAME
     pam_duo – PAM module for Duo authentication

SYNOPSIS
     pam_duo.so [conf=⟨FILENAME⟩]

DESCRIPTION
     pam_duo provides secondary authentication (typically after successful pass-
     word-based authentication) through the Duo authentication service.
```

- Let's configure sudo to use pam_duo

Sudo via pam_duo

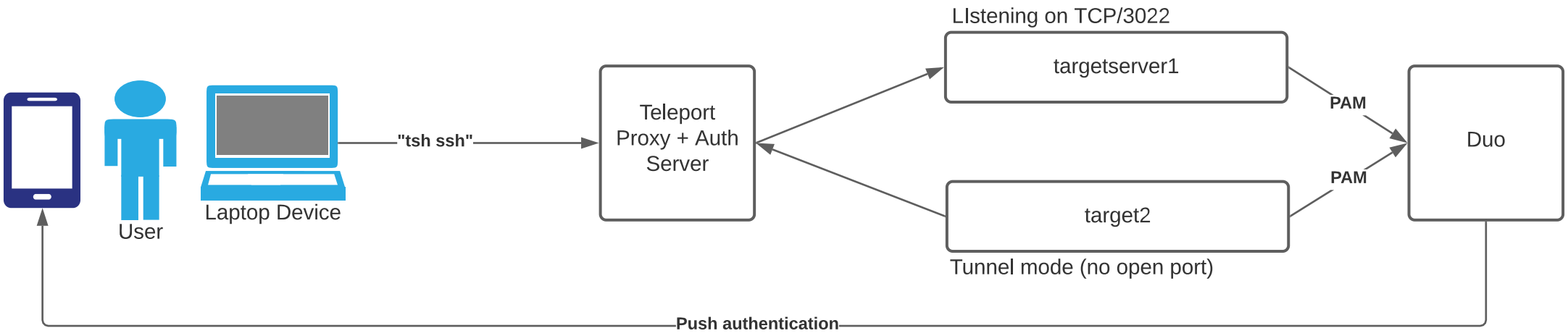
- Install pam_duo
- Configure pam_duo to talk to duo backend
- Configure /etc/pam.d/sudo to use pam_duo instead of pam_unix
- Update sudoers to require a "password"

```
[root@targetserver1 duo]# cat /etc/duo/pam_sudo.conf

[duo]
ikey = DI[redacted]
skey = [redacted]
host = api-39dfd334.duosecurity.com

failmode = secure
pushinfo = yes
autopush = yes
prompts = 1
fallback_local_ip = yes
[root@targetserver1 duo]#
```

```
[root@targetserver1 duo]# cat /etc/pam.d/sudo
#%PAM-1.0
auth        required      pam_env.so
auth        sufficient    pam_duo.so conf=/etc/duo/pam_sudo.conf
auth        required      pam_deny.so

account    include      system-auth
password   include      system-auth
session    optional     pam_keyinit.so revoke
session    required     pam_limits.so

[root@targetserver1 duo]#
```

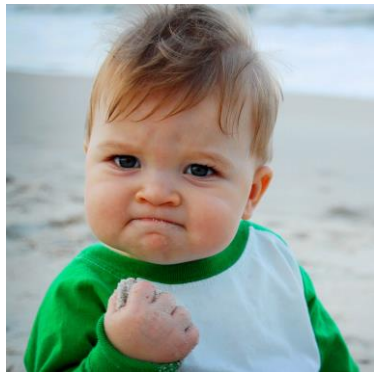
```
Defaults    passwd_tries = 1
## Allows people in group wheel to run all commands
## "PASSWD" is now duo in pam config
%wheel  ALL=(ALL)       ALL

## Same thing without a password
#%wheel ALL=(ALL)       NOPASSWD: ALL
```

Duo for sudo


LIstening on TCP/3022
targetserver1
PAM
Duo
"tsh ssh"
Teleport
Proxy + Auth
Server
PAM
target2
Tunnel mode (no open port)
User
Laptop Device
Push authentication

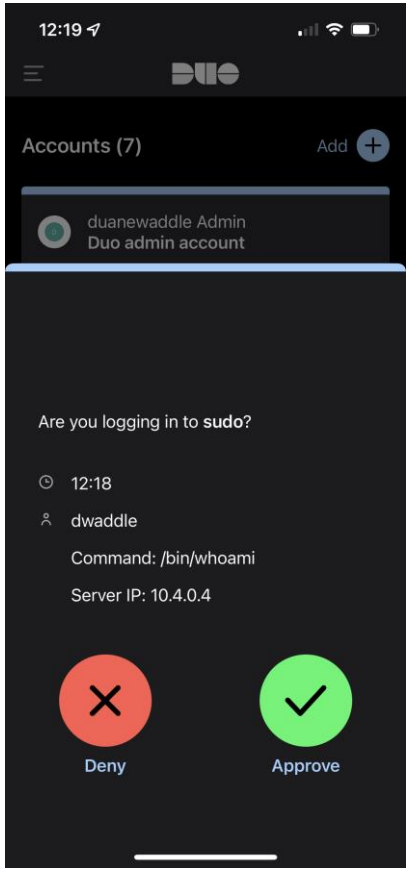
Check it out

- We do a sudo command – and get a push notification to phone!!
- Click the green check and it runs!!
- Press deny and it fails



```
[duane.waddle@targetserver1 ~]$ sudo whoami
Autopushing login request to phone...

Pushed a login request to your device...
Success. Logging you in...
root
[duane.waddle@targetserver1 ~]$
```

```
[duane.waddle@targetserver1 ~]$ sudo whoami
Autopushing login request to phone...

Pushed a login request to your device...
Login request denied.

sudo: 1 incorrect password attempt
[duane.waddle@targetserver1 ~]$
```



How'd we do?

- Strong authentication ☑
- No static RSA keys ☑
- Tied to central directory ☑
- Strong MFA ☑
- Lots of compliance ☑
- No listening ports ☑

Teleport Editions – should I spend $$?

Open Source

- Local Auth
- Federated with Github
- FIDO2 Hardware MFA
- Roles
- Session Recording
- All the k8s, desktop, database, etc stuff we don't use

Enterprise

- SAML/OIDC Federation
- Access Requests
- FIPS crypto

FIN