# Building Defensible Systems (Advanced)

by Daniel Griggs of 


dan — Hogwarts
Last login: Sat Apr 23 14:18:05 on ttys000
dan@Hogwarts: ~
$ whoami
• Daniel Griggs, CEO and founding partner: cmdSecurity
• Worked for the US DoD and other government agencies on security for Apple devices
• Specializing in security and management at scale for more than 10 years
• Currently working on multiple security standards

Security and management should
NEVER destroy the user's experience

# Continuous Diagnostics and Mitigation (CDM)

## How to win friends & influence ~~people~~ machines

# General Problems

- Security is complex
- Users are focused on ease of use
- Threats and vulnerabilities change daily
- Compromise is nearly inevitable

# Goals of security

- Know when something bad happens
- Facilitate easy workflows that are secure
- Train users to recognize security incidents
- Protect the data on the device

# Attack Lifecycle

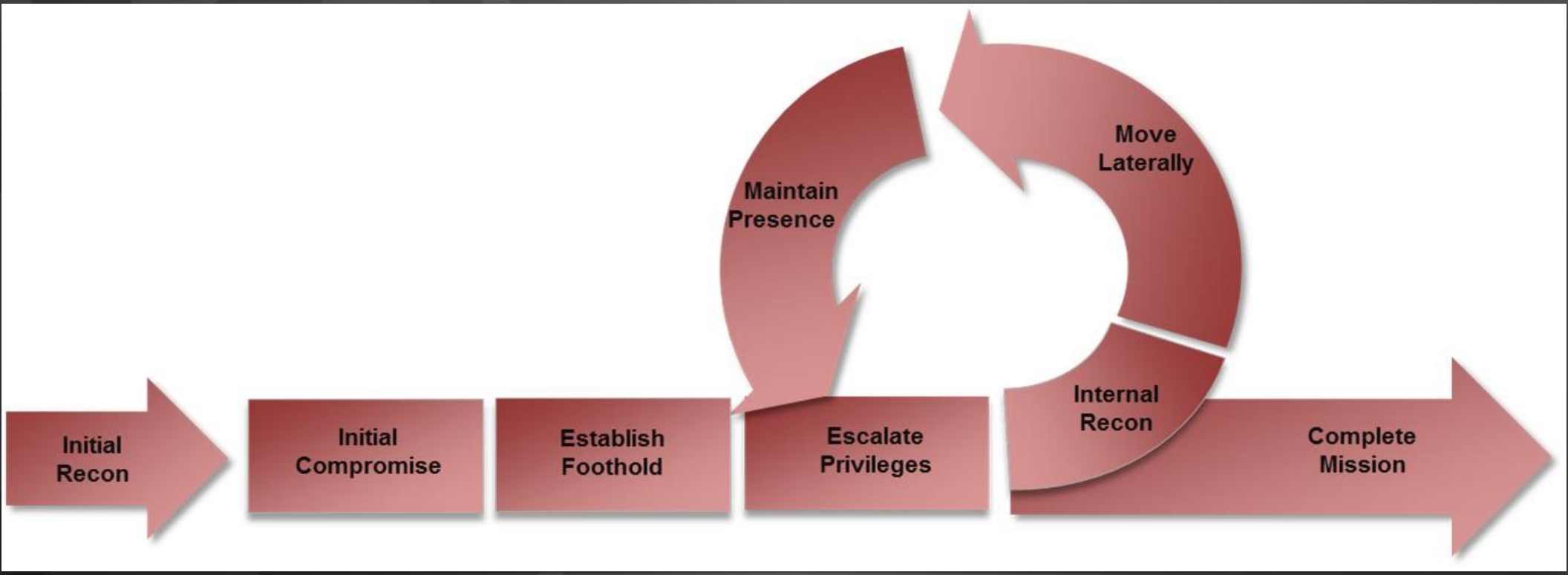



Image Source: Mandiant Consulting, see https://www.fireeye.com/services.html

# Definitions


- **Vulnerability:** A mistake in software that can be directly used by a hacker to gain access to a system or network
- **CVE:** Common Vulnerabilities and Exposures
  - Gives a common serial number to publicly known cybersecurity vulnerabilities
  - https://nvd.nist.gov/

# Definitions

Vulnerability -> Exploit -> Malware/Virus

- **Exploit:** A sequence of commands that takes advantage of a bug or vulnerability in order to cause unintended or unanticipated behavior to occur on computer software
- **HOW** an attacker uses a vulnerability to gain access to a computer

# Definitions

**Vulnerability -> Exploit -> Malware/Virus**

- **Malware/Virus:** A malicious program that, when installed, performs some form of harmful activity. These activities can be
  - Data corruption or exfiltration
  - Movement to other, more important systems
  - Denial of service(s)
- **WHAT** an attacker is using vulnerabilities and exploits to place on your computer

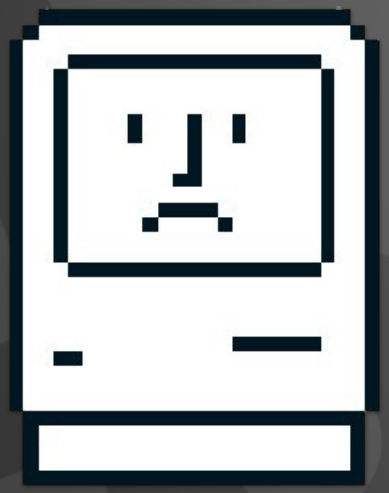
# Macs are immune though, right?

CVE(s) in the past 3 Months

- Mac CVE: 283
- Windows CVE: 261
- Linux CVE: 183

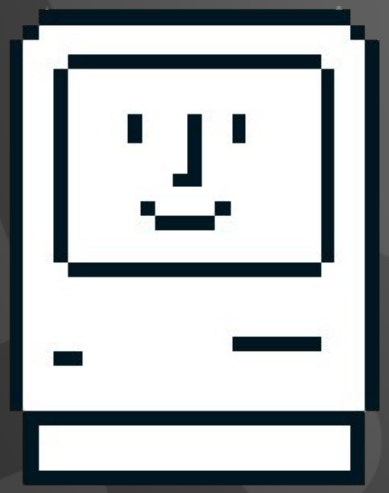
https://nvd.nist.gov

# What bad security looks like



- Machines are barely operational because of all the restrictions on them
- No record of software, services, or open ports
- Inconsistent configurations
- Users finding less-secure ways to do work

# What a properly secured device looks like



- Attackers face obstacles at every stage of the attack
- Most of the system is open for the user to do work
- Computers continuously and completely inventoried
- Scan for known vulnerabilities

# Config Profiles vs Scripts

## Profiles

Always configured, never change

(Password Policy)

## Scripts

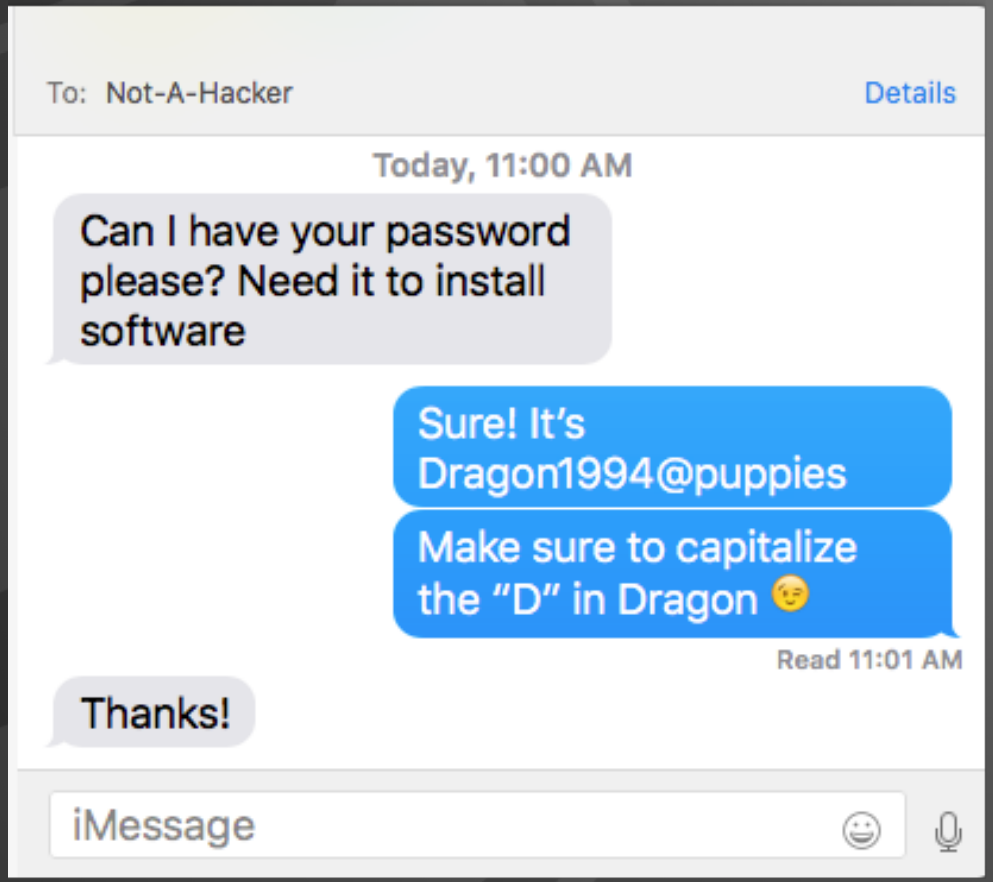
Controls that have some flexibility

(File Shares)

# Why you need to watch everything

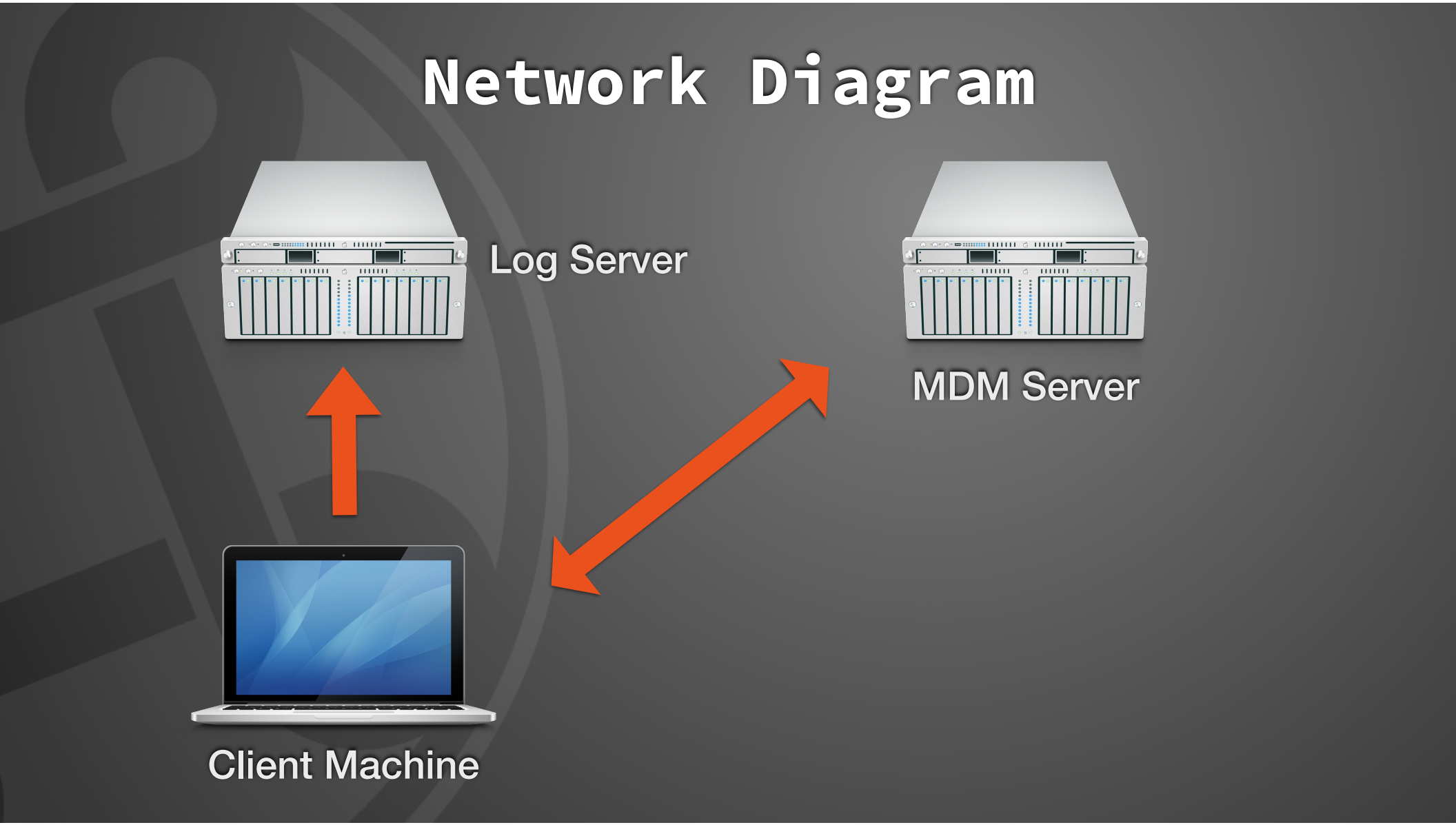
- As much meaningful signal as possible
- The best chance of detecting malicious activity
- Can allow more leeway with your user’s settings if you track everything

# Phishing




- All hope is lost
- This is the most common and most successful attack

# Network Diagram



# Use a scalpel, not an axe

## Anatomy of a Security Decision

**Define What To Protect**

Stop users from running low-level systems configuration commands

**Lock Down Access**

Only allow designated management users to ‘sudo’

**Apply Moderate Security**

Users are still able to be admins and install software

**Monitor Everything**

Know when malicious activity is happening by attempted ‘sudo’-ing

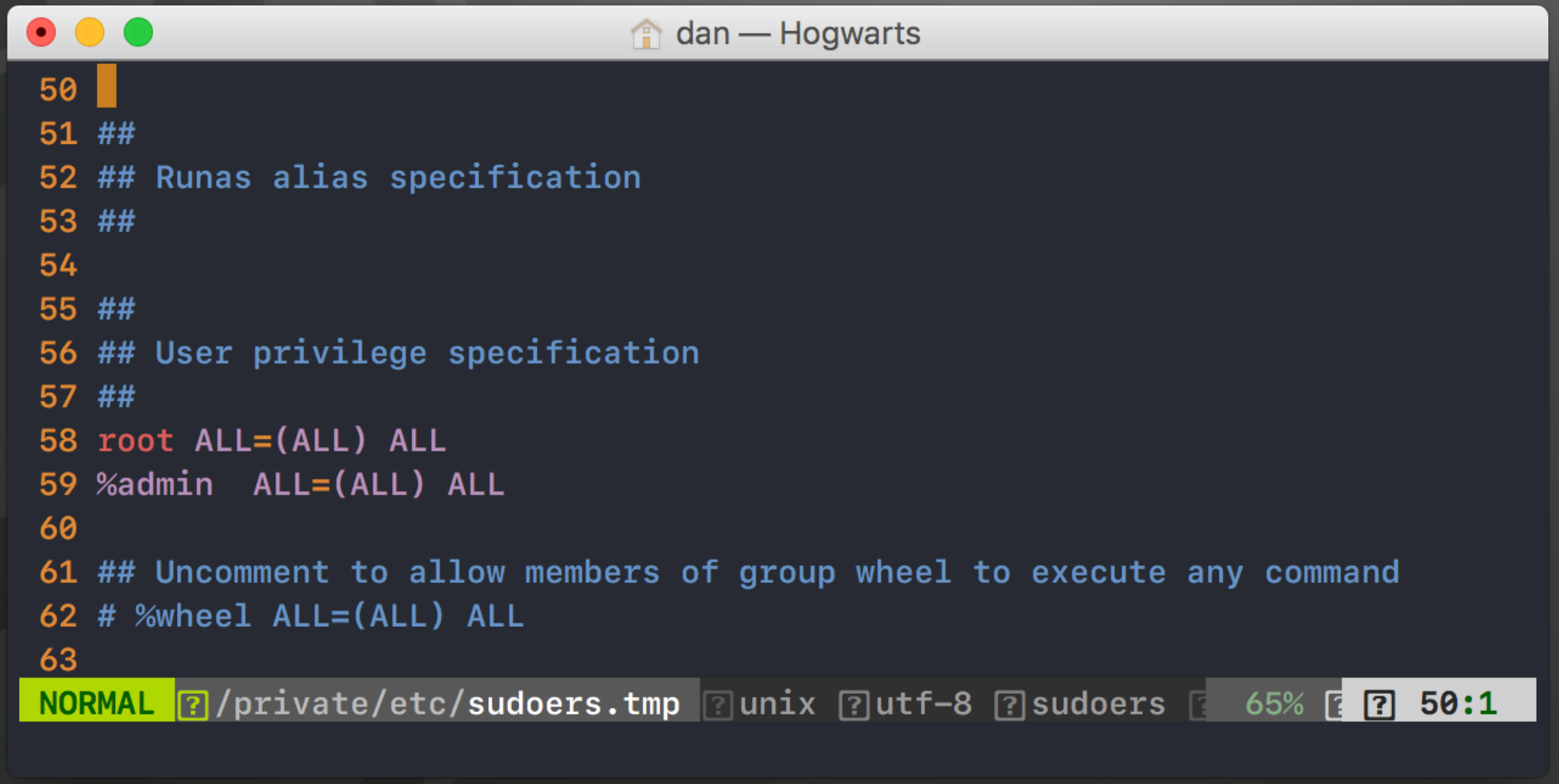
# Remove user's sudo rights

SUPER IMPORTANT!!

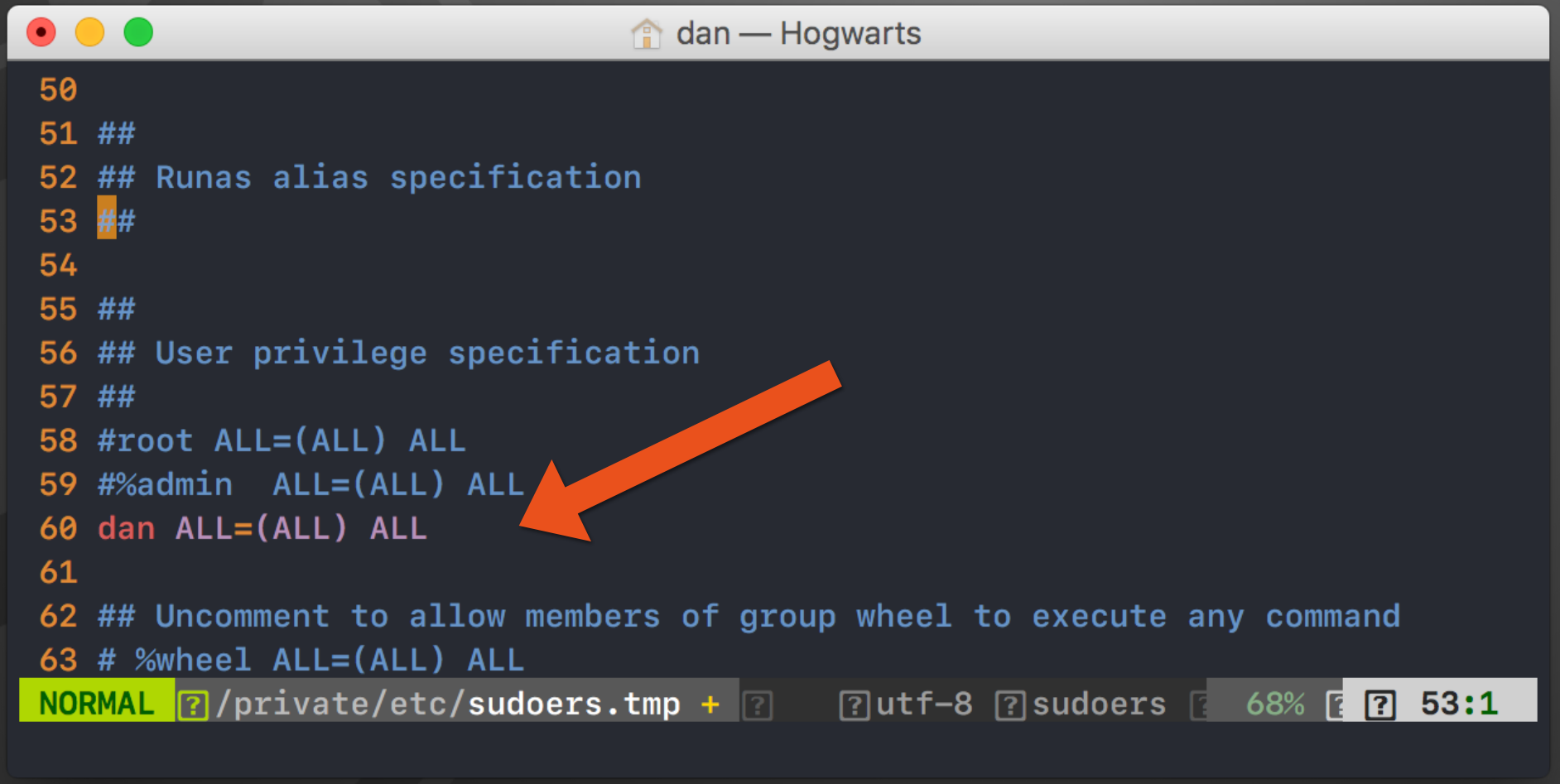
```
sudo cp /etc/sudoers ~/tempSudoers.txt
```

ONLY TEST ON THIS tempSudoers FILE

# Remove user's sudo rights



# Remove user's sudo rights



# Remove user's sudo rights

```
# This is SUPER BAD
%staff ALL=(ALL) NOPASSWD: ALL
```

# Remove user's sudo rights

```
# This is SUPER BAD
%staff ALL=(ALL) NOPASSWD: ALL
```

**# Let’s find that kind of thing**

```
grep '^%' /etc/sudoers
```

# Remove user's sudo rights

```
for user in $allUsers
do
  sudoFind=$(grep "^$user" /etc/sudoers)
  if [[ -n "$sudoFind" ]]; then
    checkOutput=("${checkOutput[@]}" "NOT SECURE $user allowed to sudo")
  fi
done
```

# BONUS! Fancy JSS Formatting!

```
if [[ ${#checkOutput[@]} -gt 0 ]]; then
  printf '%s\n' "<result>${checkOutput[@]}</result>"
  exit 0
fi
```

```
<result>NOT SECURE jssUser not in sudoers
NOT SECURE dan allowed to sudo
NOT SECURE presentation allowed to sudo
NOT SECURE Admins allowed to sudo</result>
```

# Remove user's sudo rights

```
# Find every group in the sudoers file and comment it out
sed -i -backup '/^%*/s/^/#/g' /etc/sudoers
### OR ###
# Find the admin group in sudoers file and comment it out
sed -i -backup '/^%admin/s/^/#/g' /etc/sudoers
```

# Remove user's sudo rights

```
cat /etc/pam.d/sudo

# sudo: auth account password session
auth       required       pam_opendirectory.so
account    required       pam_permit.so
password   required       pam_deny.so
session    required       pam_permit.so
```

# Side Track:
# how to get active users

DON'T `ls /Users/`

DO:

```
dscl . list /Users UserShell |
grep -v '/usr/bin/false\|jssAdmin' |
awk '{print $1}'
```

# Remove user's sudo rights

```
terminal = tty
```

```
echo "## custom settings ##" >> /etc/sudoers
echo "Defaults requiretty" >> /etc/sudoers
echo "Defaults tty_tickets" >> /etc/sudoers
echo "Defaults timestamp_timeout=5" >> /etc/sudoers
```

# JSS SG example: File Shares

In your Extension Attribute:

```
sharing -l | grep '^name:'
```

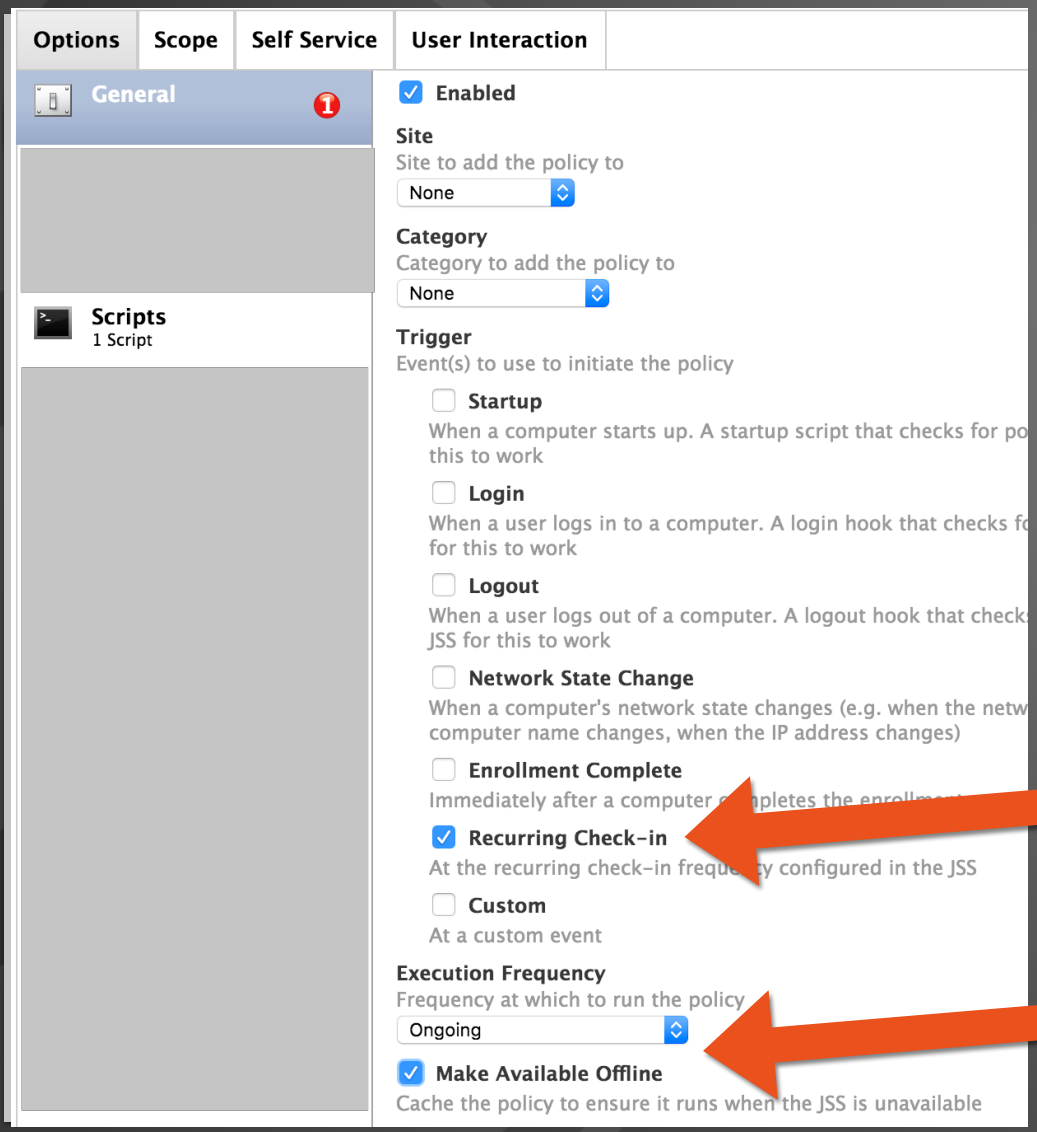
```
sudo launchctl list |
grep ‘com.apple.AppleFileServer\|com.apple.smbd'
```

Write your recon results to the syslog!

```
logger “something I want to log”
```

# JSS SG example: File Shares

- Smart group checking for extension attribute is anything BUT “secured”
- When recon find a NOT SECURE setting it writes the UNIX time to a special file only we control
  - **echo $(date +%s) > /var/.fileShareTimeStamp**




- Scope is only “NOT SECURE” computers for file sharing
- Script runs every 15 mins, forever
- Even if computer isn’t online
- Script will turn off file sharing

# JSS SG example: File Shares

First lines of that script

```
fileShareTimeStamp=$(cat /var/.fileShareTimeStamp)
currentTimeStamp=$(date +%s)
timeDiff=$(expr $currentTimeStamp - $fileShareTimeStamp)
if [[ $timeDiff -lt 3600 ]]; then
  echo $timeDiff
  exit 0
fi
```

# Time for some loggin’

# Different log levels

| Log level | Suggested Usage |
|---|---|
| Emergency (Level 0) | The highest priority, usually reserved for catastrophic failures and reboot notices. |
| Alert (Level 1) | A serious failure in a key system. |
| Critical (Level 2) | A failure in a key system. |
| Error (Level 3) | Something has failed. |
| Warning (Level 4) | Something is amiss and might fail if not corrected. |
| **Notice (Level 5)** | **Things of moderate interest to the user or administrator. (Default Level)** |
| Info (Level 6) | The lowest priority that you would normally log, and purely informational in nature. |
| Debug (Level 7) | The lowest priority, and normally not logged except for messages from the kernel. |

# Log File Attribute & Behavior: Files Involved

Top Level Configs

/etc/asl.conf

Service Specific Configs

/etc/asl/com.apple.something

# Log File Attribute & Behavior: Basic Syntax

```
> system.log mode=0640 format=bsd rotate=seq compress file_max=5M all_max=50M
```

```
> system.log mode=0640 format=bsd rotate=seq compress file_max=5M all_max=50M
```

- Name of the log file that will be created
- Default path is /var/log
- This line defines the attributes of /var/log/system.log

```
> system.log mode=0640 format=bsd rotate=seq compress file_max=5M all_max=50M
```

- Mode of the log file that will be created
- Default users are root/wheel can set different ones with uid= and gid=
- The owner can read/write, the group can read only, everyone else has no permissions (640)

```
> system.log mode=0640 format=bsd rotate=seq compress file_max=5M all_max=50M
```

- Format the log lines are written in, std is the default for almost every log (except system.log)
- bsd looks like the default system.log if you read the file itself NOT in console
- The other options of raw,xml,asl are available
- Default in macOS is std log format.

```
> system.log mode=0640 format=bsd rotate=seq compress file_max=5M all_max=50M
```

This both enables log rotation and defines a naming scheme for archived logs

```
> system.log mode=0640 format=bsd rotate=seq compress file_max=5M all_max=50M
```

- Compress = Compress the archived log files. The only format available is gzip
- file_max = The ‘live’ log file is allowed to be 5mb in size before it is rotated
- all_max = once the total size of live and archived logs reaches this point, the oldest logs will be deleted to allow new ones. Works like time machine backups

```
> batman-journal.log mode=0600 UID=501 GID=0
rotate=seq compress file_max=2M all_max=41M
```

```
> batman-journal.log mode=0600 UID=501 GID=0 rotate=seq compress file_max=2M all_max=41M
```

- /var/log/batman-journal.log
- Only Batman and Group wheel can access
- Std log format (with error level in plaintext)
- Rotate and compress after 2M
- Keep up to 41mb of Batman journals

# Basic Syntax: Log Routing

## /etc/asl/ and friends

# 3 Major Parts to a log handler entry

```
? [= Sender example] [<= Level error] notify com.cmdsec.example
```

?

This line is a query rule that will be tried against incoming logs

The previous examples we were using a > to define file attributes

# 3 Major Parts to a log handler entry

```
? [= Sender example] [<= Level error] notify com.cmdsec.example
```

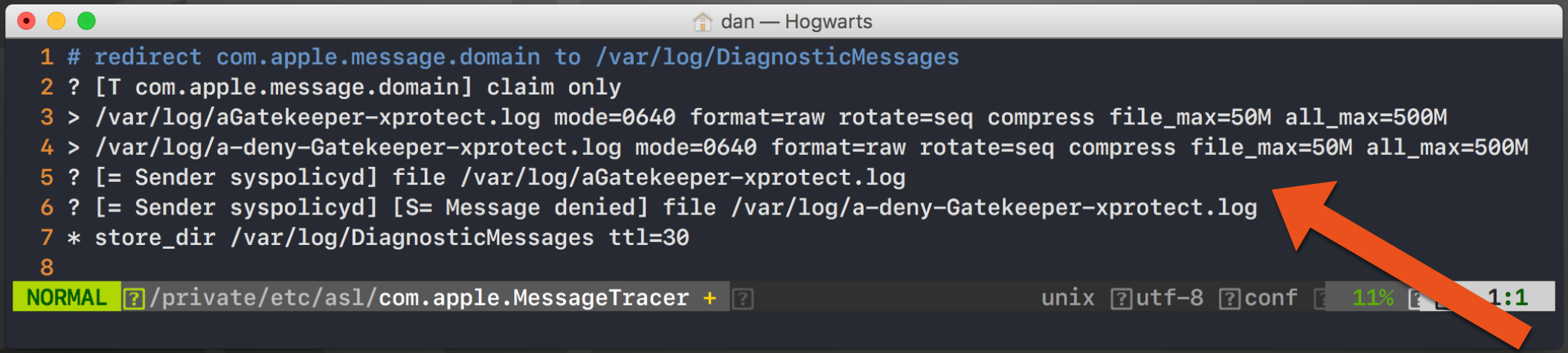
- Queries are enclosed in [] brackets
- The above example line is looking for two things
  - The sending application is named “example”
  - The log level of that log line is worse or equal to error
- The logic is always **AND** when specifying two query parameters

# 3 Major Parts to a log handler entry

`? [= Sender example] [<= Level error] notify com.cmdsec.example`

- Send a notify command through notifyd to a listening service or script
- **notifyutil -w [optional number] com.cmdsec.example**
  - This log line would alert the above command every time a match occurred
  - The script would continue once it had one or [opt number] calls

# Parse all XProtect and Gatekeeper



# Parse all X-Protect and Gatekeeper

```
> /var/log/aGatekeeper-xprotect.log mode=0640 format=raw rotate=seq
compress file_max=50M all_max=500M

? [= Sender syspolicyd] file /var/log/aGatekeeper-xprotect.log
```

# Parse all X-Protect and Gatekeeper

[Level 5]

[Time 1467161044]

[TimeNanoSec 899451000]

**[Message assessment granted for Test-download.jpg by _XProtect] [PID 64413]**

[Sender syspolicyd]

[Facility daemon]

[com.apple.message.domain com.apple.security.assessment.outcome2] [com.apple.message.signature2 bundle:UNBUNDLED] [com.apple.message.signature3 Test-download.jpg] [com.apple.message.signature5 UNKNOWN]

[com.apple.message.signature granted:_XProtect] [com.apple.message.signature4 3] [UID 0] [GID 0][Host Hogwarts] [ReadGID 80]

# Parse all X-Protect and Gatekeeper DENY Events

```
> /var/log/a-deny-Gatekeeper-xprotect.log mode=0640 format=raw
rotate=seq compress file_max=50M all_max=500M

? [= Sender syspolicyd] [S= Message denied] file /var/log/a-deny-
Gatekeeper-xprotect.log
```

# Parse all X-Protect and Gatekeeper DENY Events

[Level 5] [Time 1467167544] [TimeNanoSec 480538000]

**[Message assessment denied for .pkg] [PID 64413]**

[Sender syspolicyd]

[Facility daemon]

[com.apple.message.domain com.apple.security.assessment.outcome2]

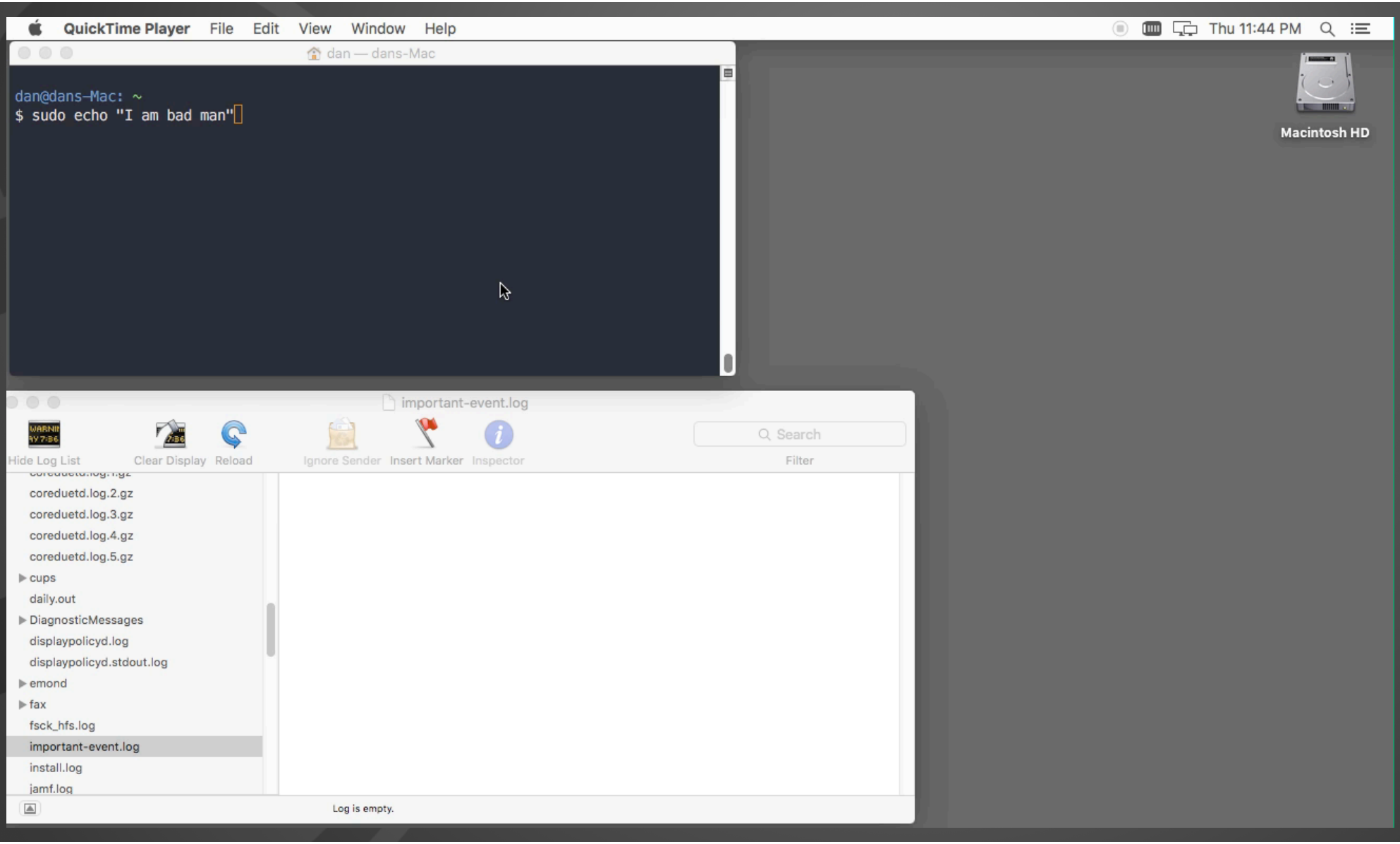
[com.apple.message.signature2 bundle:com.mygreatcompany.pkg.UpdatePackage]

[com.apple.message.signature3 .pkg] [com.apple.message.signature5 1.0] [UID 0] [GID 0] [Host Hogwarts] [ReadGID 80] [com.apple.message.signature4 2] [com.apple.message.signature denied:no usable signature]

# On-Device Defenses

# Sudo Events!

- Attacker runs a sudo command
- Writes to special log file as well as syslog
- logs alert a script
- Fixes the problem
- Notifies the user (Just for this demo)


QuickTime Player File Edit View Window Help
Thu 11:44 PM
dan — dans-Mac
dan@dans-Mac: ~
$ sudo echo "I am bad man"
Macintosh HD
important-event.log
Hide Log List
Clear Display
Reload
Ignore Sender
Insert Marker
Inspector
Search
Filter
coreduetd.log.2.gz
coreduetd.log.3.gz
coreduetd.log.4.gz
coreduetd.log.5.gz
cups
daily.out
DiagnosticMessages
displaypolicyd.log
displaypolicyd.stdout.log
emond
fax
fsck_hfs.log
important-event.log
install.log
jamf.log
Log is empty.

# Settings involved in that demo

**/etc/asl.conf**

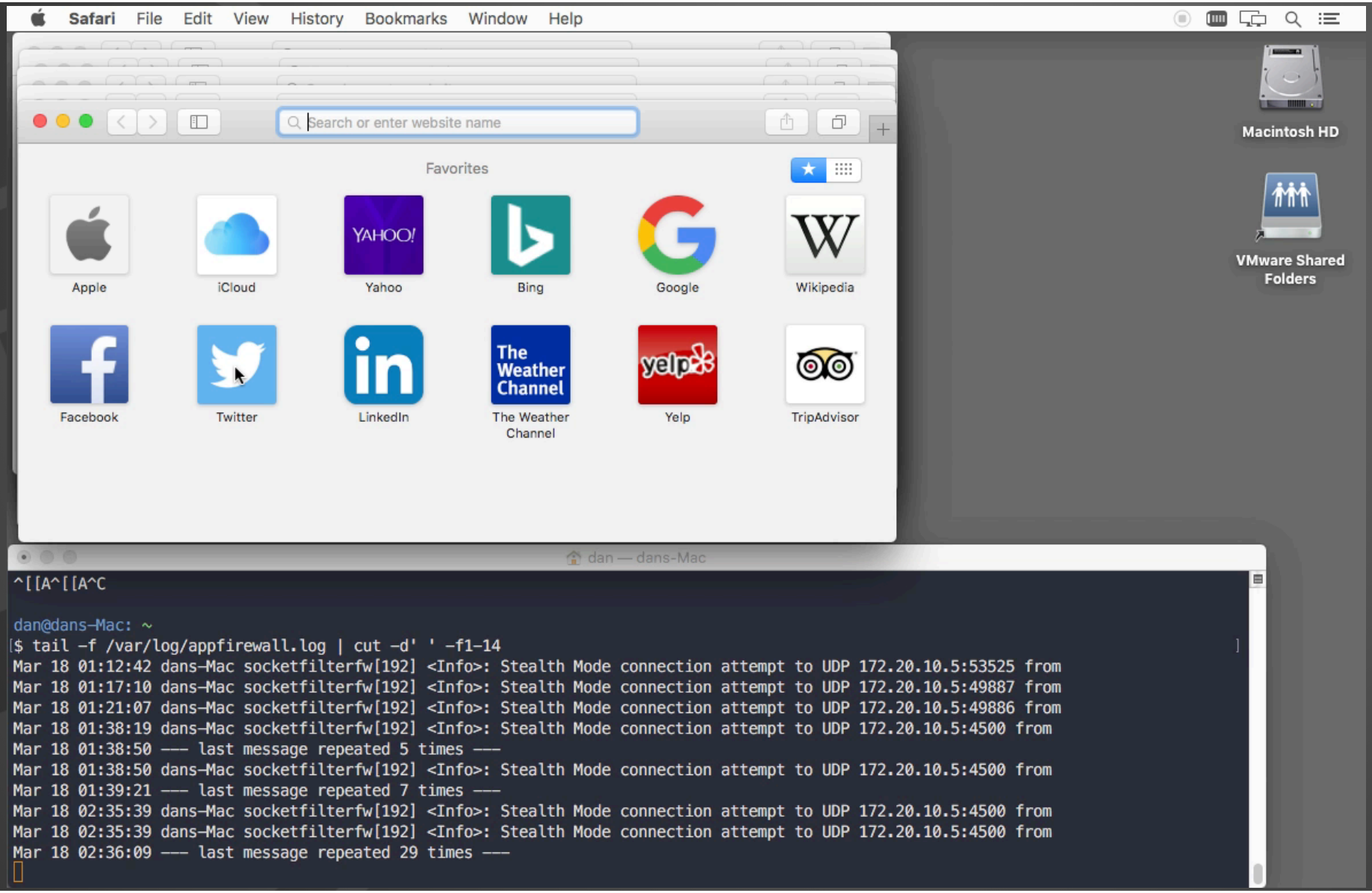
```
? [= Facility authpriv] [<= Level notice] notify com.cmdsec.test
```

LaunchDaemon running a script listening with notifyutil -w com.cmdsec.test

Script patches the problem and notifies the user

# Firewall Events!

- Attacker starts scanning the VM
- VM forces all networking off
- Only network allowed is OUT to my JSS and to VPN server
- Networking enabled again after connected to VPN


Safari File Edit View History Bookmarks Window Help
Search or enter website name
Favorites
Apple
iCloud
Yahoo
Bing
Google
Wikipedia
Facebook
Twitter
LinkedIn
The Weather Channel
Yelp
TripAdvisor
Macintosh HD
VMware Shared Folders
dan — dans-Mac
^[[A^[[A^C
dan@dans-Mac: ~
$ tail -f /var/log/appfirewall.log | cut -d' ' -f1-14
Mar 18 01:12:42 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:53525 from
Mar 18 01:17:10 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:49887 from
Mar 18 01:21:07 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:49886 from
Mar 18 01:38:19 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:4500 from
Mar 18 01:38:50 --- last message repeated 5 times ---
Mar 18 01:38:50 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:4500 from
Mar 18 01:39:21 --- last message repeated 7 times ---
Mar 18 02:35:39 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:4500 from
Mar 18 02:35:39 dans-Mac socketfilterfw[192] <Info>: Stealth Mode connection attempt to UDP 172.20.10.5:4500 from
Mar 18 02:36:09 --- last message repeated 29 times ---

# Settings involved in that demo

/etc/asl.conf

```
? [= Facility com.apple.alf.logging] [S= Message connection attempt] notify com.cmdsec.test
```

LaunchDaemon keeping a script alive

Script starts with: `notifyutil -w 40 com.cmdsec.test`

Once the count of notifications to com.cmdsec.example reach 40 the Firewall and VPN script will run

It will append this to /etc/pf.conf

```
block all
pass out inet proto tcp from any to jss.company.com keep state
pass out inet proto tcp from any to vpn.company.com keep state
```

Then Run

```
pfctl -d
pfctl -e -f /etc/pf.conf
```