black hat BRIEFINGS

AUGUST 6-7, 2025

MANDALAY BAY / LAS VEGAS

Leveraging Jamf for Red Teaming in Enterprise Environments

By

Lance Cain and Daniel Mayer

Lance and Dan

Lance Cain

- Service Architect at SpecterOps Inc.
- macOS Security Researcher
- Red Teaming and Pentest Lead
- Jamf Exploitation Enthusiast

Daniel Mayer

- Senior Consultant at SpecterOps Inc.
- Ex-Senior Security Researcher at CrowdStrike
- Hobbyist free-to-play game cheat maker
- Blogs about it and other topics at mayer.cool

SPECTEROPS

Overview

- Introduction
  - MacOS in the Modern Enterprise
  - Jamf Management and Permissions
  - Pros and Cons of Jamf Abuse
  - Tool References
- Privilege Escalation
  - Accounts
  - Api Integrations
- Code Execution
  - Policies and Scripts
  - Policies
  - Computer Extension Attributes
- Defensive Recommendations
  - Local vs. Cloud Deployments
- Credits and Kudos
- Questions

Introduction – MacOS in Modern Enterprises

- macOS is popular with developers, cloud admins, IT engineers, and users with privileged technical access

Introduction – MacOS in Modern Enterprises

- macOS is popular with developers, cloud admins, IT engineers, and users with privileged technical access
- Often macOS devices are initially setup with a Jamf Pro enrollment and integrated with a cloud provider like Azure, then not monitored as much afterwards

Introduction – MacOS in Modern Enterprises

- macOS is popular with developers, cloud admins, IT engineers, and users with privileged technical access
- Often macOS devices are initially setup with a Jamf Pro enrollment and integrated with a cloud provider like Azure, then not monitored as much afterwards
- Sharing some of the most dangerous attack paths we have discovered in client environments regarding Jamf Pro

Introduction – Jamf Management and Permissions

- Many capabilities across Jamf Pro, Jamf Connect, Jamf Protect, Jamf Account:
  - Mobile Device Management (MDM)
  - Software Licensing
  - Device Compliance Checks
  - Initial Provisioning Setups
  - SSO Integrations
  - Device Hardening and Protection
  - More...

Introduction – Jamf Management and Permissions

- Jamf Pro offers a couple different permission assignment interfaces:
  - CRUD access for JSSObjects
  - Allow action for JSSActions
  - Read and Update for JSSSettings

Introduction – Jamf Management and Permissions

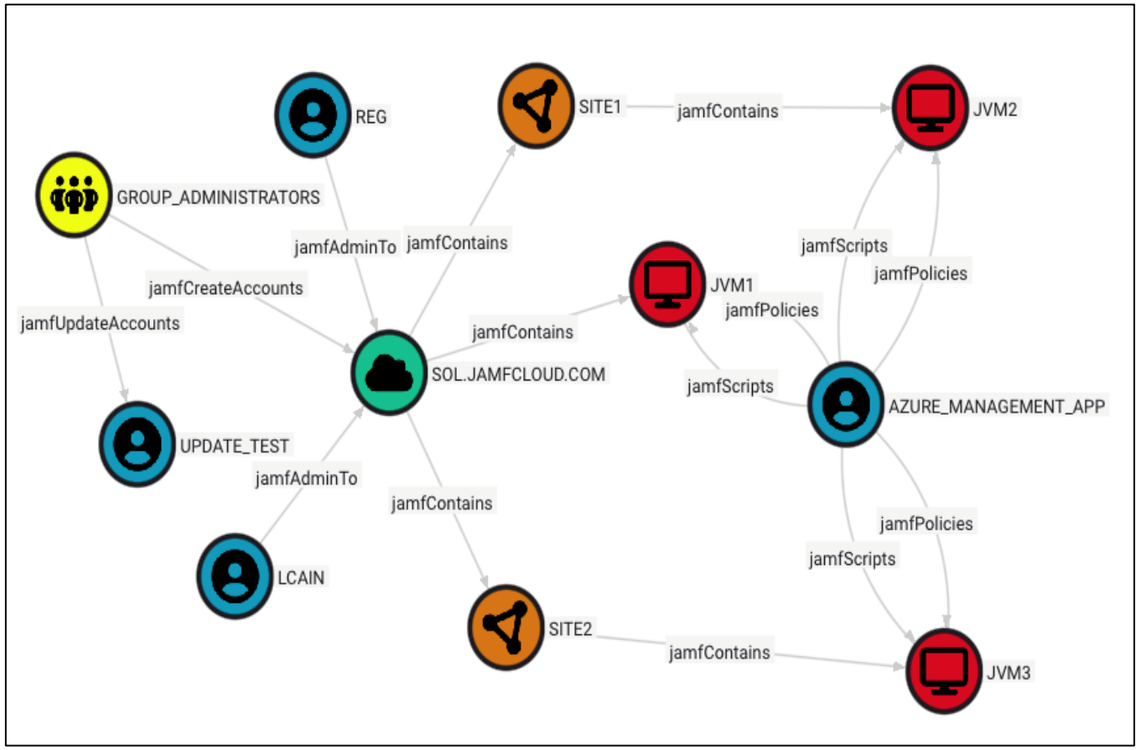
- Jamf Pro offers a couple different permission assignment interfaces:
  - CRUD access for JSSObjects
  - Allow action for JSSActions
  - Read and Update for JSSSettings
- Each permission object commonly has an API endpoint :
  - xxx.jamfcloud.com/JSSObjects/computers

Introduction – Jamf Management and Permissions

- Jamf Pro offers a couple different permission assignment interfaces:
  - CRUD access for JSSObjects
  - Allow action for JSSActions
  - Read and Update for JSSSettings
- Each permission object commonly has an API endpoint :
  - xxx.jamfcloud.com/JSSObjects/computers
- With Jamf credentials we leverage API access to:
  - Escalate Privileges
  - Perform Reconaissance
  - Laterally Move to Managed Devices

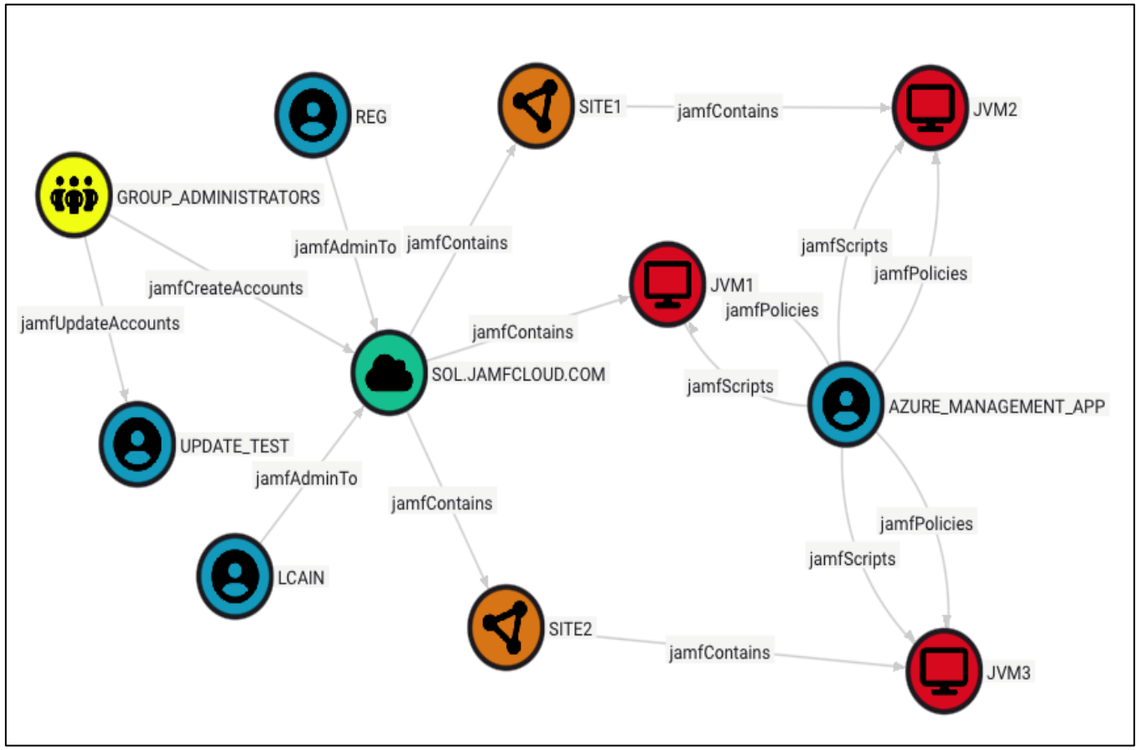
Introduction – Expectations vs. Reality

What Admins Document

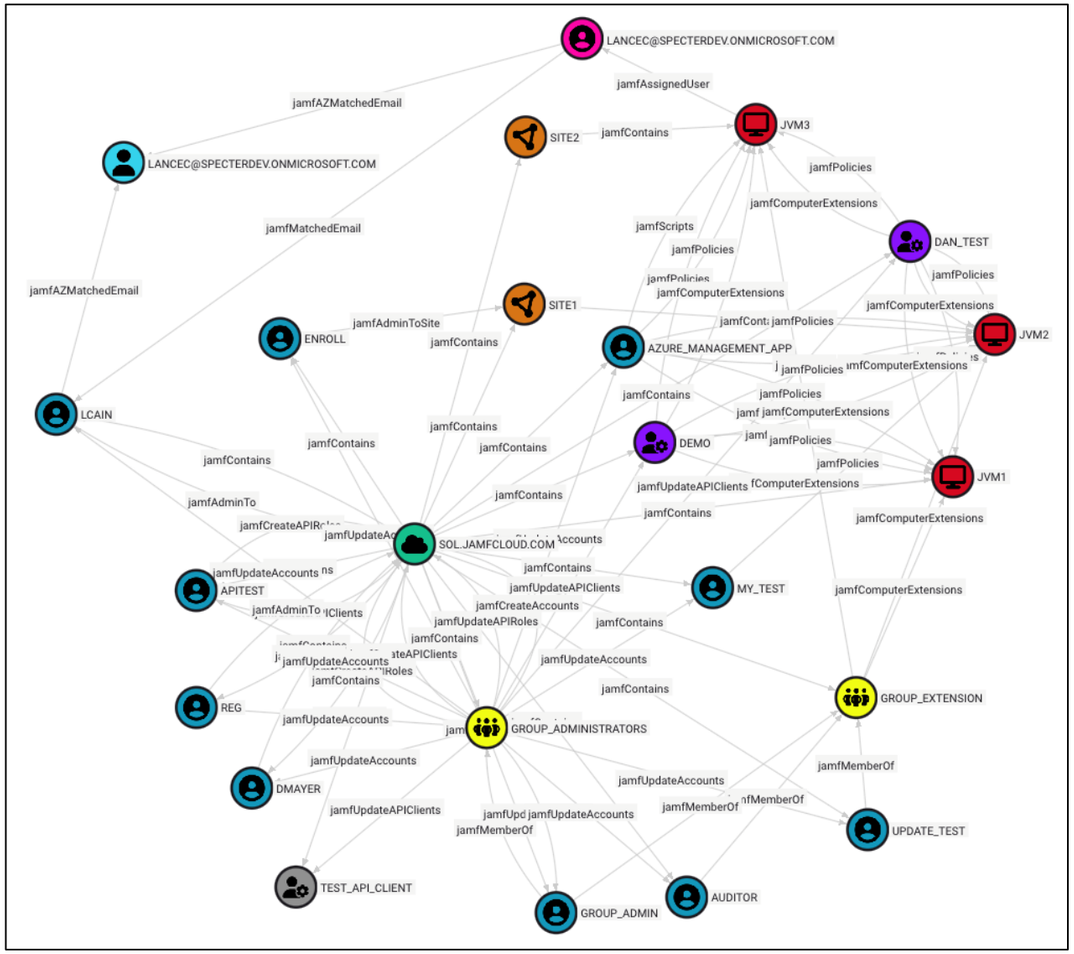


Introduction – Expectations vs. Reality

What Admins Document



What We Have Found



Introduction – Pros of Jamf Abuse

- Jamf performs multiple administrative actions that EDRs filter to avoid false positives

Introduction – Pros of Jamf Abuse

- Jamf performs multiple administrative actions that EDRs filter to avoid false positives
- Jamf offers the option to set up self-signing for software deployments

Introduction – Pros of Jamf Abuse

- Jamf performs multiple administrative actions that EDRs filter to avoid false positives
- Jamf offers the option to set up self-signing for software deployments
- Most organizations aren’t monitoring their Jamf environments for change

Introduction – Cons of Jamf Abuse

- If log forwarding is configured, then defenders have a path to follow

Introduction – Tools Eve and JamfHound

- Eve is an open-source python3 post-exploitation toolkit that automates many of the attacks we will be discussing and more

README

Eve

Overview

Eve is a Jamf exploitation toolkit used to interact with either cloud hosted Jamf tenants or locally hosted Jamf servers using various API calls. To use this toolkit credentials for an account registered with the Jamf instance that has API access will be required. This tooling automates attacks that my team and I have performed successfully to exploit Jamf access to enumerate Apple devices, escalate privileges, as well as execute code to laterally move to different systems. The intended user for this toolkit should already have some awareness about Jamf API permissions to know how to best leverage their access. For those trying to discover what can be done I recommend starting here.

Requirements

1. Python 3.12 or newer
2. requests Python Module
3. dateutil Python Module
4. flask Python Module
5. flask_cors Python Module

Introduction – Tools Eve and JamfHound

- JamfHound is an open-source python3 solution that integrates with BloodHound to visualize attack paths and audit the security of Jamf environments

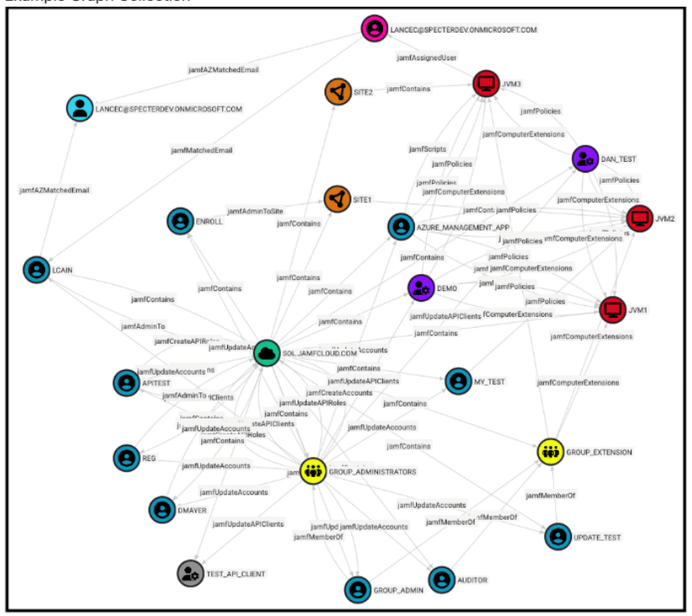
README License

JamfHound

About

JamfHound is a python3 project designed to collect and identify attack paths in Jamf Pro tenants based on existing object permissions. When run the scripts create JSON object files that can be imported into BloodHound instances to populate custom nodes such as Jamf accounts or computers and display edges between those nodes to reveal different methods of control or code execution available. JamfHound can perform collections against both cloud-hosted and on-site Jamf Pro instances using known credentials. Users are recommended to provision auditor accounts to perform collections which is a pre-defined role that will have the necessary permissions to read all resources to be collected.

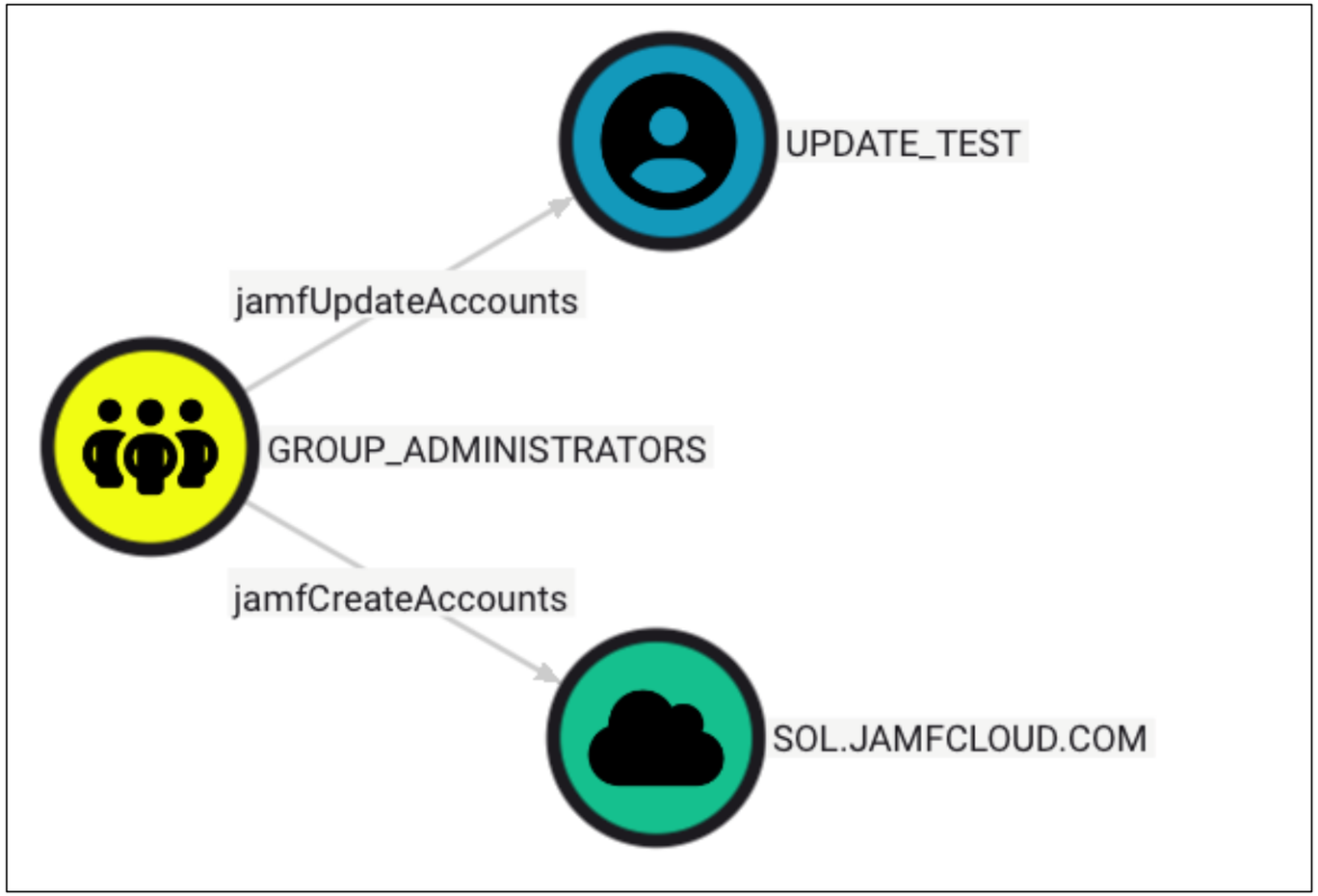
Example Graph Collection



Privilege Escalation

Privilege Escalation – Accounts

Jamf Pro Account Creation and Update Edges



Privilege Escalation – Accounts

- JSSObject Permission ‘Create Accounts’ – Allows creating new local Jamf accounts
- JSSObject Permission ‘Update Accounts’ – Allows updating any existing local Jamf accounts

Privilege Escalation – Accounts

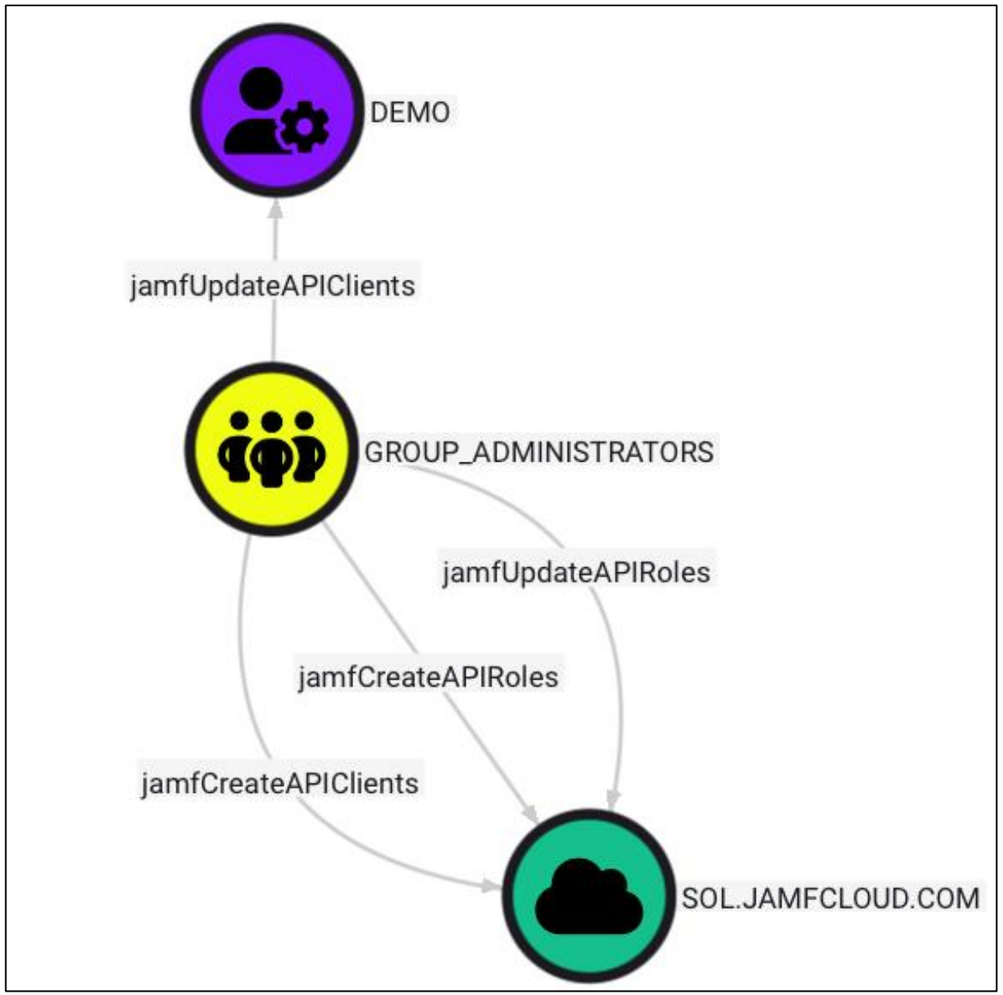
- Permissions are scoped to the entire tenant when given to an account or API client

Privilege Escalation – Accounts

- Permissions are scoped to the entire tenant when given to an account or API client
- Create or Update Account permissions are linked with the Jamf Pro Create or Update Group permissions

Privilege Escalation – API Integrations

JamfHound Create and Update API Clients and Roles Edges



Privilege Escalation – API Integrations

- JSSObject Permission ‘Create API Integrations’ – Allows creating new API clients and retrieving new passwords
- JSSObject Permission ‘Update API Integrations’ – Allows updating any existing clients
- JSSObject Permission ‘Create API Roles’ – Allows creating API client permission sets
- JSSObject Permission ‘Update API Roles’ – Allows updating existing permission sets

Privilege Escalation – API Integrations

- JSSObject Permission ‘Create API Integrations’ – Allows creating new API clients and retrieving new passwords
- JSSObject Permission ‘Update API Integrations’ – Allows updating any existing clients
- JSSObject Permission ‘Create API Roles’ – Allows creating API client permission sets
- JSSObject Permission ‘Update API Roles’ – Allows updating existing permission sets
- **Control of an API Client + Control of API Role Assignments = Jamf Pro Admin**

Code Execution

Code Execution – Recon

- Computer objects contain any information you could ever want about a particular host
  - Real-world information about the user
  - All installed software
  - Hardware information such as storage info
  - All running services
  - What Jamf policies and groups effect it
  - User accounts
  - Much more

```
curl \
--request GET \
--url https://jamf.corp.local:443/JSSResource/computers/id/1 \
--header 'Authorization: Bearer <...snip...>'

{
  "computer": {
    "general": {
      "id": 1,
      "name": "mytarget-mac",
      <...snip...>
    "location": {
      "username": "mytarget",
      "realname": "My Target",
      "phone": "555-555-5555"
    "software": {
      <...snip...>
      "running_services": [
      <...snip...>
      "applications": [
       {
         "name": "Activity Monitor.app",
         "path": "/System/Applications/Utilities/Activity Monitor.app",
         "version": "10.14",
         "bundle_id": "com.apple.ActivityMonitor"
       },
       <...snip...>
      <AND SO MUCH MORE>
```

Code Execution – Recon

- Computer objects contain any information you could ever want about a particular host
  - Real-world information about the user
  - All installed software
  - Hardware information such as storage info
  - All running services
  - What Jamf policies and groups effect it
  - User accounts
  - Much more

```
curl \
--request GET \
--url https://jamf.corp.local:443/JSSResource/computers/id/1 \
--header 'Authorization: Bearer <...snip...>'

{
  "computer": {
    "general": {
      "id": 1,
      "name": "mytarget-mac",
      <...snip...>
    "location": {
      "username": "mytarget",
      "realname": "My Target",
      "phone": "555-555-5555"
    "software": {
      <...snip...>
      "running_services": [
      <...snip...>
      "applications": [
       {
         "name": "Activity Monitor.app",
         "path": "/System/Applications/Utilities/Activity Monitor.app",
         "version": "10.14",
         "bundle_id": "com.apple.ActivityMonitor"
       },
       <...snip...>
      <AND SO MUCH MORE>
```

Code Execution – Recon

- Computer objects contain any information you could ever want about a particular host
  - Real-world information about the user
  - All installed software
  - Hardware information such as storage info
  - All running services
  - What Jamf policies and groups effect it
  - User accounts
  - Much more

```
curl \
--request GET \
--url https://jamf.corp.local:443/JSSResource/computers/id/1 \
--header 'Authorization: Bearer <...snip...>'

{
  "computer": {
    "general": {
      "id": 1,
      "name": "mytarget-mac",
      <...snip...>
    "location": {
      "username": "mytarget",
      "realname": "My Target",
      "phone": "555-555-5555"
    "software": {
      <...snip...>
      "running_services": [
      <...snip...>
      "applications": [
       {
         "name": "Activity Monitor.app",
         "path": "/System/Applications/Utilities/Activity Monitor.app",
         "version": "10.14",
         "bundle_id": "com.apple.ActivityMonitor"
       },
       <...snip...>
      <AND SO MUCH MORE>
```

Code Execution – Recon

- Computer objects contain any information you could ever want about a particular host
  - Real-world information about the user
  - All installed software
  - Hardware information such as storage info
  - All running services
  - What Jamf policies and groups effect it
  - User accounts
  - Much more

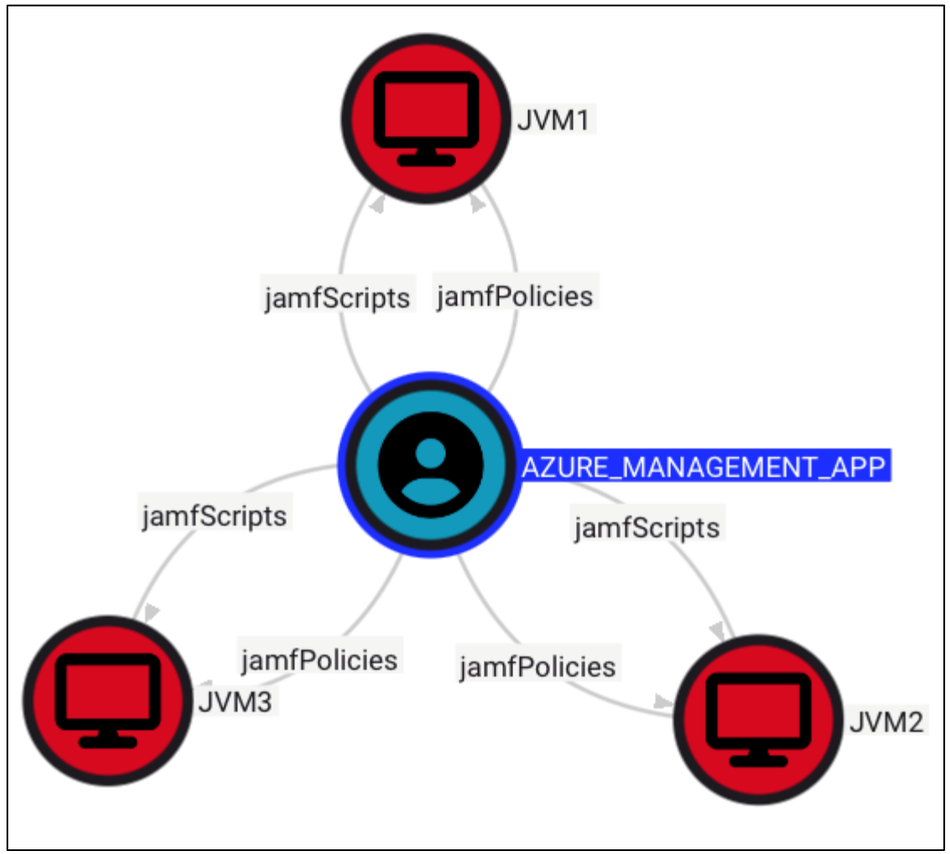
```
curl \
--request GET \
--url https://jamf.corp.local:443/JSSResource/computers/id/1 \
--header 'Authorization: Bearer <...snip...>'

{
  "computer": {
    "general": {
      "id": 1,
      "name": "mytarget-mac",
      <...snip...>
    "location": {
      "username": "mytarget",
      "realname": "My Target",
      "phone": "555-555-5555"
    "software": {
      <...snip...>
      "running_services": [
      <...snip...>
      "applications": [
       {
         "name": "Activity Monitor.app",
         "path": "/System/Applications/Utilities/Activity Monitor.app",
         "version": "10.14",
         "bundle_id": "com.apple.ActivityMonitor"
       },
       <...snip...>
      <AND SO MUCH MORE>
```

Code Execution – Recon

- JSSObject Permission ‘Computers Read’ – Allows reading computer objects

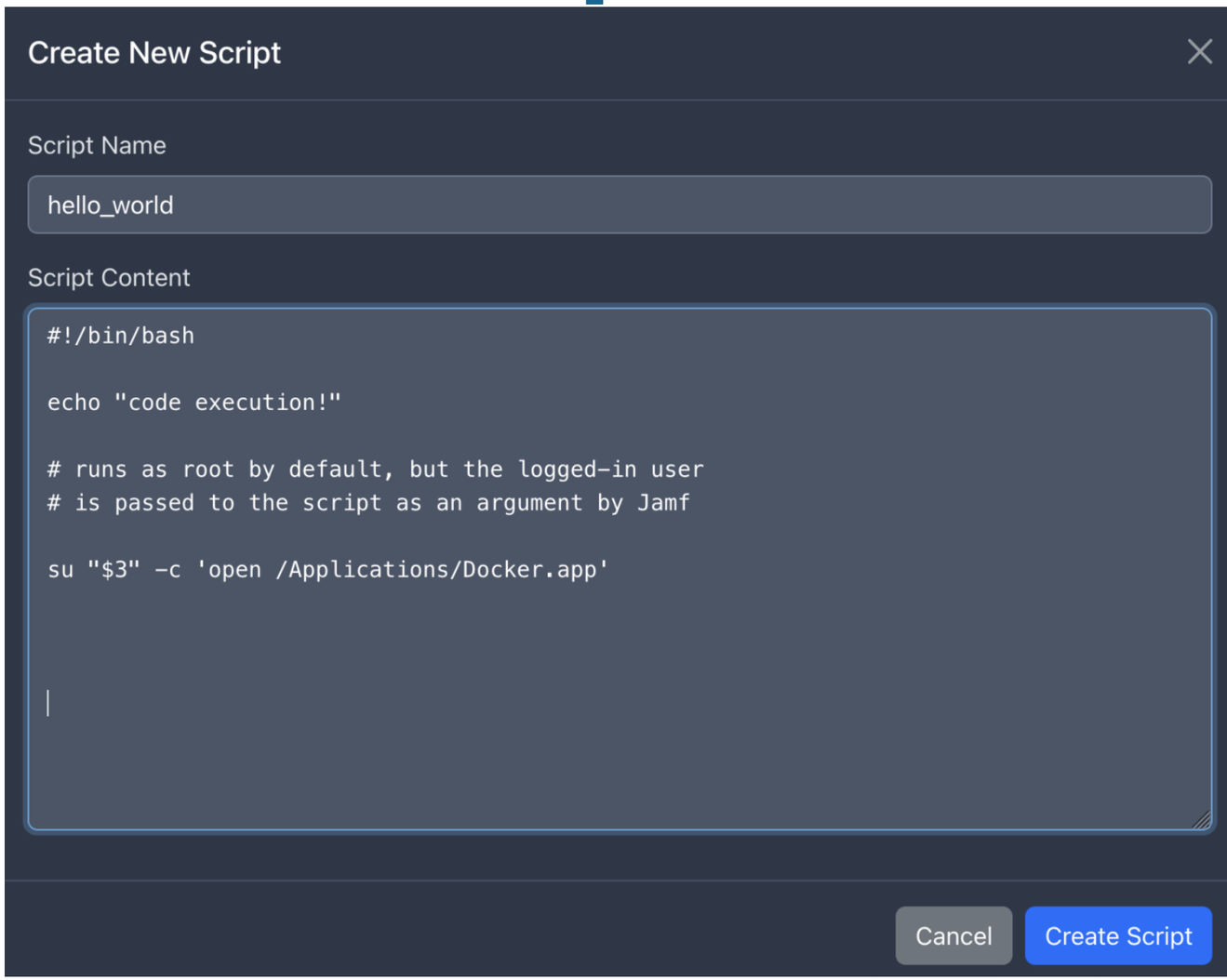
Code Execution – Policies and Scripts

JamfHound Policies and Scripts Edges



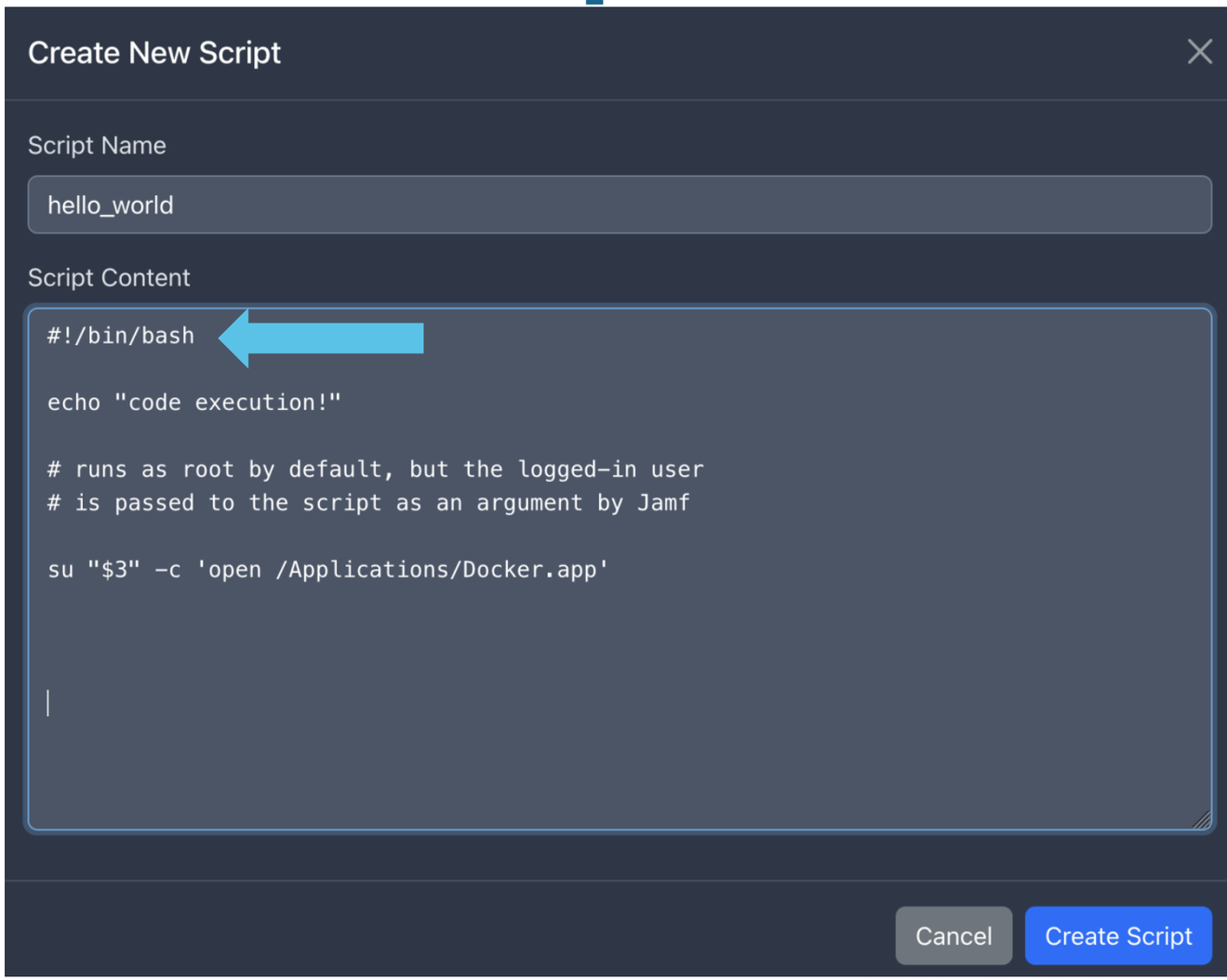
Code Execution – Policies and Scripts

- Scripts can be bash, perl, python3, or anything else you can shebang (#!)
- Run as root by default
- We have leveraged this capability extensively to execute malicious scripts on macOS



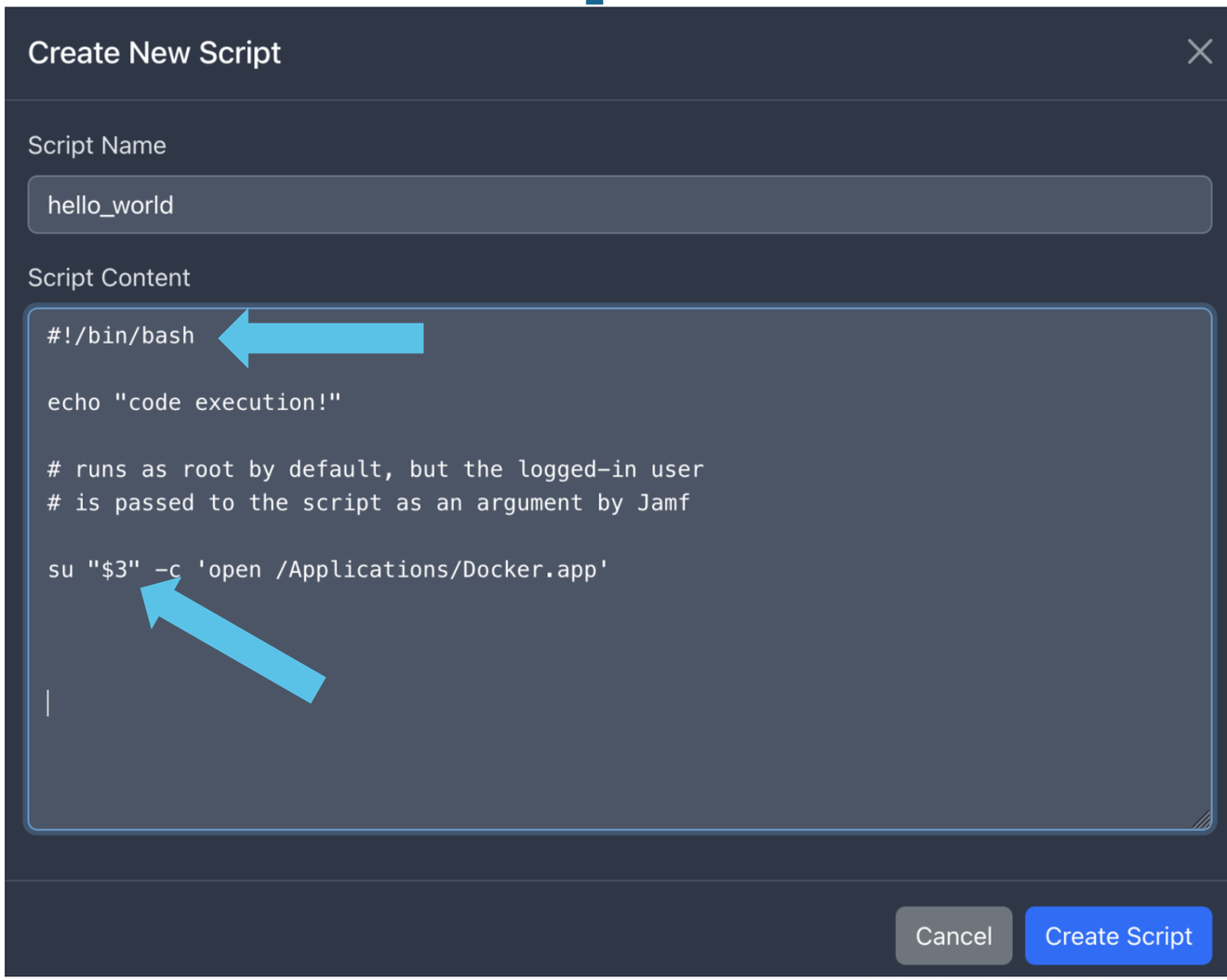
Code Execution – Policies and Scripts

- Scripts can be bash, perl, python3, or anything else you can shebang (#!)
- Run as root by default
- We have leveraged this capability extensively to execute malicious scripts on macOS



Code Execution – Policies and Scripts

- Scripts can be bash, perl, python3, or anything else you can shebang (#!)
- Run as root by default
- We have leveraged this capability extensively to execute malicious scripts on macOS



Code Execution – Policies and Scripts

- Policies are used to handle configuring macOS devices and a major application.
- These can include scripts executed by different trigger events
- You can configure policies to run pre-defined scripts at regular intervals, once per computer, whenever a computer initially joins a Jamf tenant
- You can specify target computers and even specific users of computers

Policy XML:

```
<policy>
    <general>
        <name>My Policy Updated</name> <!-- Custom Policy Name -->
        <enabled>true</enabled>
        <target_drive>/</target_drive>
        <trigger>EVENT</trigger>  <!-- Recurring Check-in trigger -->
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
    </general>
    <scope>
        <computers>
            <computer>
                <id>1</id>
            </computer>
            <computer>
                <id>2</id>
            </computer>
        </computers>
    </scope>
    <scripts>
        <script>
            <id>1</id>
            <priority>After</priority>
        </script>
    </scripts>
</policy>
```

Code Execution – Policies and Scripts

- Policies are used to handle configuring macOS devices and a major application.
- These can include scripts executed by different trigger events
- You can configure policies to run pre-defined scripts at regular intervals, once per computer, whenever a computer initially joins a Jamf tenant
- You can specify target computers and even specific users of computers

Policy XML:

```
<policy>
    <general>
        <name>My Policy Updated</name> <!-- Custom Policy Name -->
        <enabled>true</enabled>
        <target_drive>/</target_drive>
        <trigger>EVENT</trigger>  <!-- Recurring Check-in trigger -->
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
    </general>
    <scope>
        <computers>
            <computer>
                <id>1</id>
            </computer>
            <computer>
                <id>2</id>
            </computer>
        </computers>
    </scope>
    <scripts>
        <script>
            <id>1</id>
            <priority>After</priority>
        </script>
    </scripts>
</policy>
```

Code Execution – Policies and Scripts

- Policies are used to handle configuring macOS devices and a major application.
- These can include scripts executed by different trigger events
- You can configure policies to run pre-defined scripts at regular intervals, once per computer, whenever a computer initially joins a Jamf tenant
- You can specify target computers and even specific users of computers

Policy XML:

```
<policy>
    <general>
        <name>My Policy Updated</name> <!-- Custom Policy Name -->
        <enabled>true</enabled>
        <target_drive>/</target_drive>
        <trigger>EVENT</trigger>  <!-- Recurring Check-in trigger -->
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
    </general>
    <scope>
        <computers>
            <computer>
                <id>1</id>
            </computer>
            <computer>
                <id>2</id>
            </computer>
        </computers>
    </scope>
    <scripts>
        <script>
            <id>1</id>
            <priority>After</priority>
        </script>
    </scripts>
</policy>
```

Code Execution – Policies and Scripts

- Policies are used to handle configuring macOS devices and a major application.
- These can include scripts executed by different trigger events
- You can configure policies to run pre-defined scripts at regular intervals, once per computer, whenever a computer initially joins a Jamf tenant
- You can specify target computers and even specific users of computers

Policy XML:

```
<policy>
    <general>
        <name>My Policy Updated</name> <!-- Custom Policy Name -->
        <enabled>true</enabled>
        <target_drive>/</target_drive>
        <trigger>EVENT</trigger>  <!-- Recurring Check-in trigger -->
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
    </general>
    <scope>
        <computers>
            <computer>
                <id>1</id>
            </computer>
            <computer>
                <id>2</id>
            </computer>
        </computers>
    </scope>
    <scripts>
        <script>
            <id>1</id>
            <priority>After</priority>
        </script>
    </scripts>
</policy>
```

Code Execution – Policies and Scripts

- JSSObject Permission ‘Create Policies’ – Allows creating new management polices
- JSSObject Permission ‘Update Policies’ – Allows updating existing management policies
- JSSObject Permission ‘Create Scripts’ – Allows creating scripts run by policies
- JSSObject Permission ‘Update Scripts’ – Allows updating scripts run by policies

Code Execution – Policies

- Later discovered policies can be used alone to execute commands on managed macOS devices
- Configured via a separate XML tag
- Avoids uploading scripts

Policy XML:

```
<policy>
    <general>
        <name>Execute Policy</name> <!-- Custom Policy Name -->
        <enabled>true</enabled>
        <target_drive>/</target_drive>
        <trigger>EVENT</trigger>  <!-- Recurring Check-in trigger -->
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
    </general>
    <scope>
        <computers>
            <computer>
                <id>2</id>
            </computer>
            <computer>
                <id>3</id>
            </computer>
        </computers>
    </scope>
    <files_processes>
        <kill_process>false</kill_process>
        <run_command>echo hello world</run_command>
    </files_processes>
</policy>
```

Code Execution – Policies

- Later discovered policies can be used alone to execute commands on managed macOS devices
- Configured via a separate XML tag
- Avoids uploading scripts

Policy XML:

```
<policy>
    <general>
        <name>Execute Policy</name> <!-- Custom Policy Name -->
        <enabled>true</enabled>
        <target_drive>/</target_drive>
        <trigger>EVENT</trigger>  <!-- Recurring Check-in trigger -->
        <trigger_checkin>true</trigger_checkin>
        <frequency>Once per computer</frequency>
    </general>
    <scope>
        <computers>
            <computer>
                <id>2</id>
            </computer>
            <computer>
                <id>3</id>
            </computer>
        </computers>
    </scope>
    <files_processes>
        <kill_process>false</kill_process>
        <run_command>echo hello world</run_command>
    </files_processes>
</policy>
```

Code Execution – Policies

- JSSObject Permission ‘Create Policies’ – Allows creating new management polices
- JSSObject Permission ‘Update Policies’ – Allows updating existing management policies

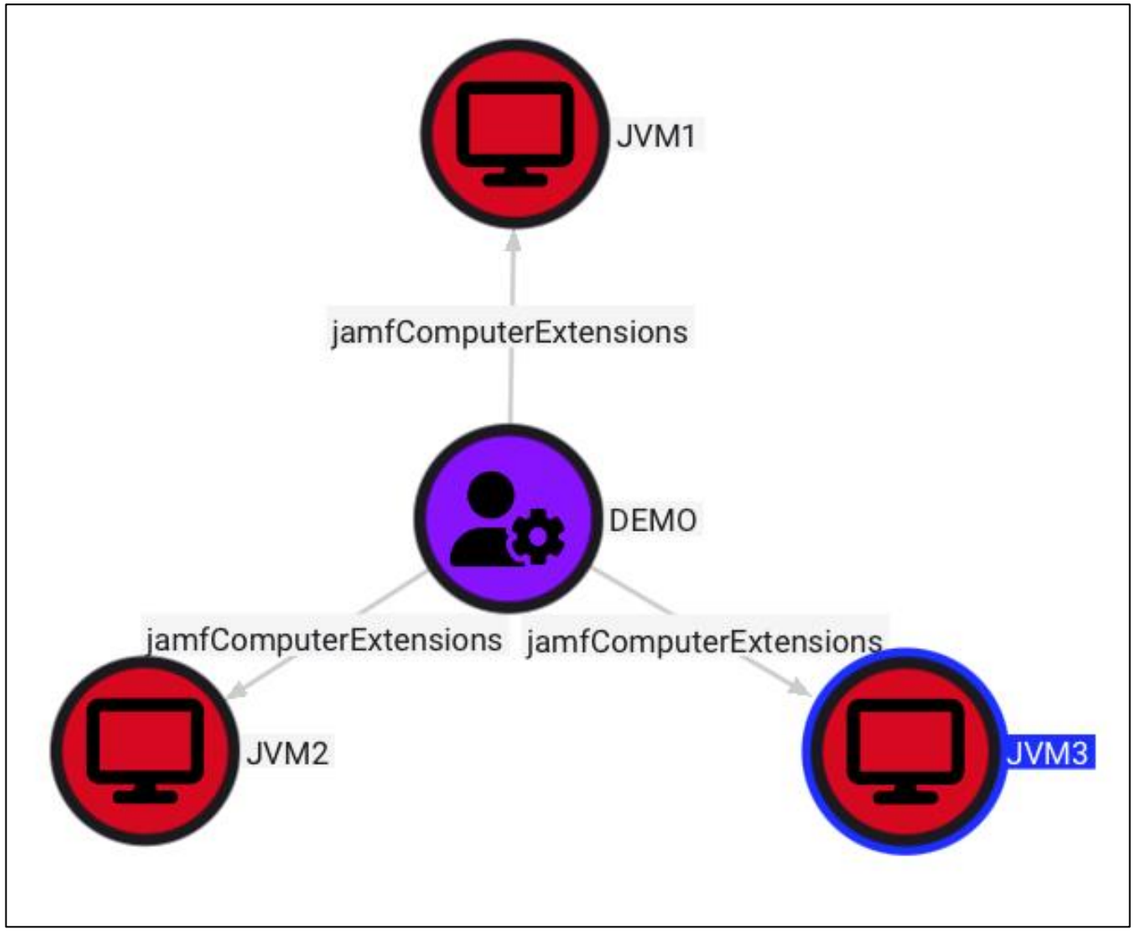
Code Execution – Policies

- JSSObject Permission ‘Create Policies’ – Allows creating new management polices
- JSSObject Permission ‘Update Policies’ – Allows updating existing management policies

No Script Permissions Needed!

Code Execution – Computer Extension Attributes

JamfHound Computer Extension Edges



Code Execution – Computer Extension Attributes

- Used for populating custom information in the Jamf Pro interface
- Allows supplying scripts to generate data
- Runs whenever ‘jamf recon’ is executed, by default once every 24 hours
- Applied to **all** Jamf controlled macOS devices, filtering must be done in the script

XML Content

```
<computer_extension_attribute>
    <name>Eve Test</name>
    <description>Creates /tmp/test.txt on the client</description>
    <data_type>String</data_type>
    <input_type>
        <type>script</type>
        <platform>Mac</platform>
        <script>#!/bin/bash
touch /tmp/test.txt
echo "&lt;result&gt;Created /tmp/test.txt&lt;/result&gt;"</script>
<!-- Must escape <> characters -->
    </input_type>
    <inventory_display>General</inventory_display>
    <enabled>true</enabled>
</computer_extension_attribute>
```

Code Execution – Computer Extension Attributes

- Used for populating custom information in the Jamf Pro interface
- Allows supplying scripts to generate data
- Runs whenever ‘jamf recon’ is executed, by default once every 24 hours
- Applied to **all** Jamf controlled macOS devices, filtering must be done in the script

XML Content

```
<computer_extension_attribute>
    <name>Eve Test</name>
    <description>Creates /tmp/test.txt on the client</description>
    <data_type>String</data_type>
    <input_type>
        <type>script</type>
        <platform>Mac</platform>
        <script>#!/bin/bash
touch /tmp/test.txt
echo "&lt;result&gt;Created /tmp/test.txt&lt;/result&gt;"</script>
<!-- Must escape <> characters -->
    </input_type>
    <inventory_display>General</inventory_display>
    <enabled>true</enabled>
</computer_extension_attribute>
```

Code Execution – Computer Extension Attributes

- Used for populating custom information in the Jamf Pro interface
- Allows supplying scripts to generate data
- Runs whenever ‘jamf recon’ is executed, by default once every 24 hours
- Applied to **all** Jamf controlled macOS devices, filtering must be done in the script

XML Content

```
<computer_extension_attribute>
    <name>Eve Test</name>
    <description>Creates /tmp/test.txt on the client</description>
    <data_type>String</data_type>
    <input_type>
        <type>script</type>
        <platform>Mac</platform>
        <script>#!/bin/bash
touch /tmp/test.txt
echo "&lt;result&gt;Created /tmp/test.txt&lt;/result&gt;"</script>
<!-- Must escape <> characters -->
    </input_type>
    <inventory_display>General</inventory_display>
    <enabled>true</enabled>
</computer_extension_attribute>
```

Code Execution – Computer Extension Attributes

- JSSObject Permission ‘Create Computer Extension Attributes’ – Allows creating new Computer Extension Attribute objects
- JSSObject Permission ‘Update Computer Extension Attributes’ – Allows updating existing Computer Extension Attributes

Defensive Recommendations

Defensive Recommendations

- JSS access and Tomcat access logs can be used to monitor for API access
- Change Management logs are generated on modifications to Jamf objects, but do not show the delta
- API Credentials can be time-gated to small timespans and created just-in-time
- Firewalls can be configured to allowlist access to API endpoints

Cloud Caveats:

- If you are a cloud customer, support will engage their internal IR team to investigate logs you do not have access to
    - There is also a paid log forwarding service for JSS access and Change Management logs
- If you are a cloud customer, you can also request allowlisting for API endpoints

Credits and Recognitions

- Defensive Recommendations
  - Jamf Team Members for collaborating and providing best practice recommendations and answering questions
    - Dino Minutolo
    - Adam Rozmus
    - Chris McMacken
    - Michael Paul
- JamfHound
  - Craig Wright, JD Crandell, West Shepherd for helping implement the v0.0.1 of JamfHound to demonstrate the POC
  - Elad Shamir and the SpecterOps Research Team for helping shape the tool for use with OpenGraph
  - Thank you to SpecterOps clients and partners that tested JamfHound enterprise collection via early access

Questions?

- … What's a Jamf?
- Where Are the Links to Eve and JamfHound?
    - Eve - https://github.com/RobotOperator/Eve
    - JamfHound - https://github.com/SpecterOps/JamfHound

Black Hat Sound Bytes

1. Compromised Jamf principals can lead to privilege escalation and undetected code execution in multiple ways – this can be tested with Eve
2. Organizations need to monitor changes across their Jamf tenant to detect compromise – configure local and cloud logging
3. Organizations should regularly audit permissions of their Jamf accounts, groups, and API clients – JamfHound can help