
ROLL A 20 WITH MSCP
JOHN MAHLMAN
MAC/LINUX PLATFORM LEAD, LEIDOS
MAC ADMINS FONDATION BOARD MEMBER
DUNGEON MASTER FOR THE DAY
2024
MACADMINS
CONFERENCE

SESSION O

Game mechanics (Brief intro about mSCP and JCE)

Use a Pre-Made Character (Pick baseline and build)

Roll Your Own Character (Customize the benchmark)

Play Session 1 (Put this into Jamf)

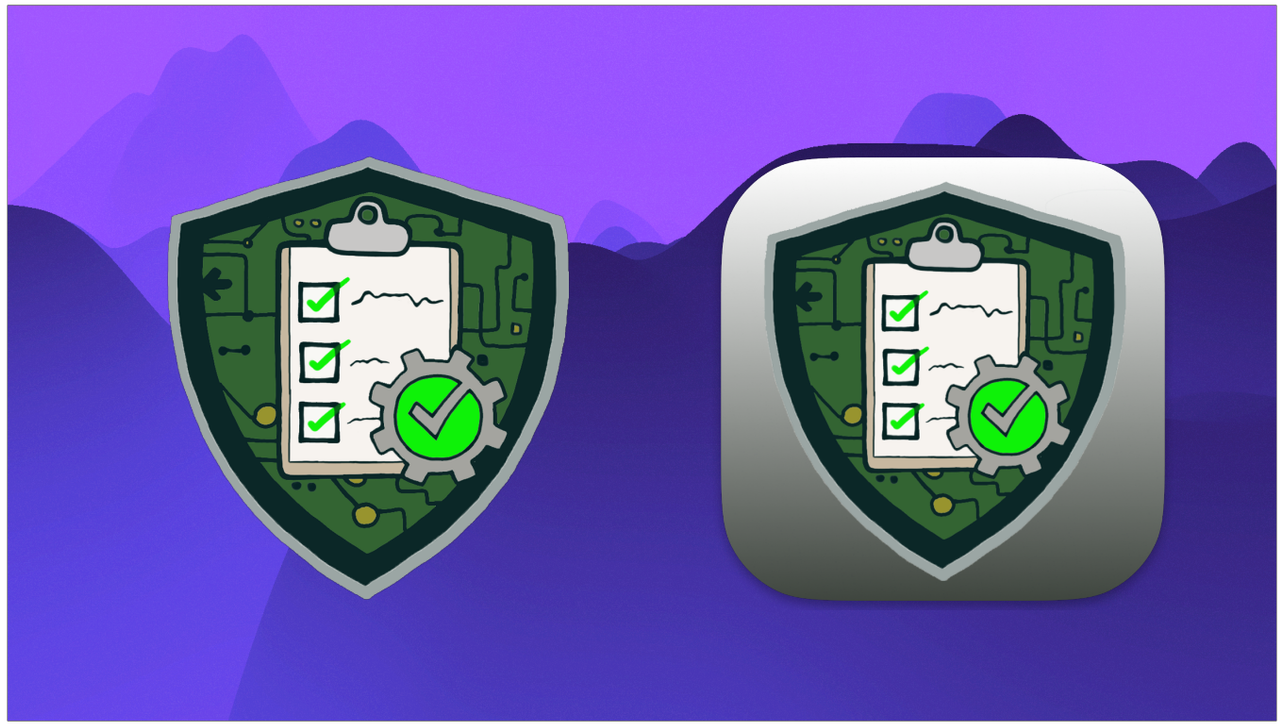
Roll Initiative (Demo!)

Post-game discussion (Q&A)

I want to set the stage a bit with a Session 0…I also want to clarify that while I will use some “tabletop gaming” terms in this talk, I will always back it up with a generalized explanation. So, if you’re not a TTRPG person, fear not! Now, if you are a gamer…I’m sorry, I don’t have a TON of gaming references in this because frankly… I didn’t want to get too in the weeds.

But THIS is one off those gaming references, a Session 0 is a pre-game session, so this is the agenda!

LEt’s talk about the game mechanics…



Today, we’re using two pieces of software…well, one is software and one I code.

The macOS Security Compliance Project on the left, and Jamf Compliance Editor on the right.

MACOS SECURITY COMPLIANCE PROJECT

THIS IS OUR FOUNDATIONAL RULE-SET

Open source framework

Normalize and accelerate annual adoption of OS and guidance documentation

Unify approach in setting controls

Vendor agnostic

Git repository: https://github.com/usnistgov/macos_security

The macOS Security compliance project (or mscp) is an open source framework for normalizing and accelerating annual macOS security guidance, documentation, and adoption.

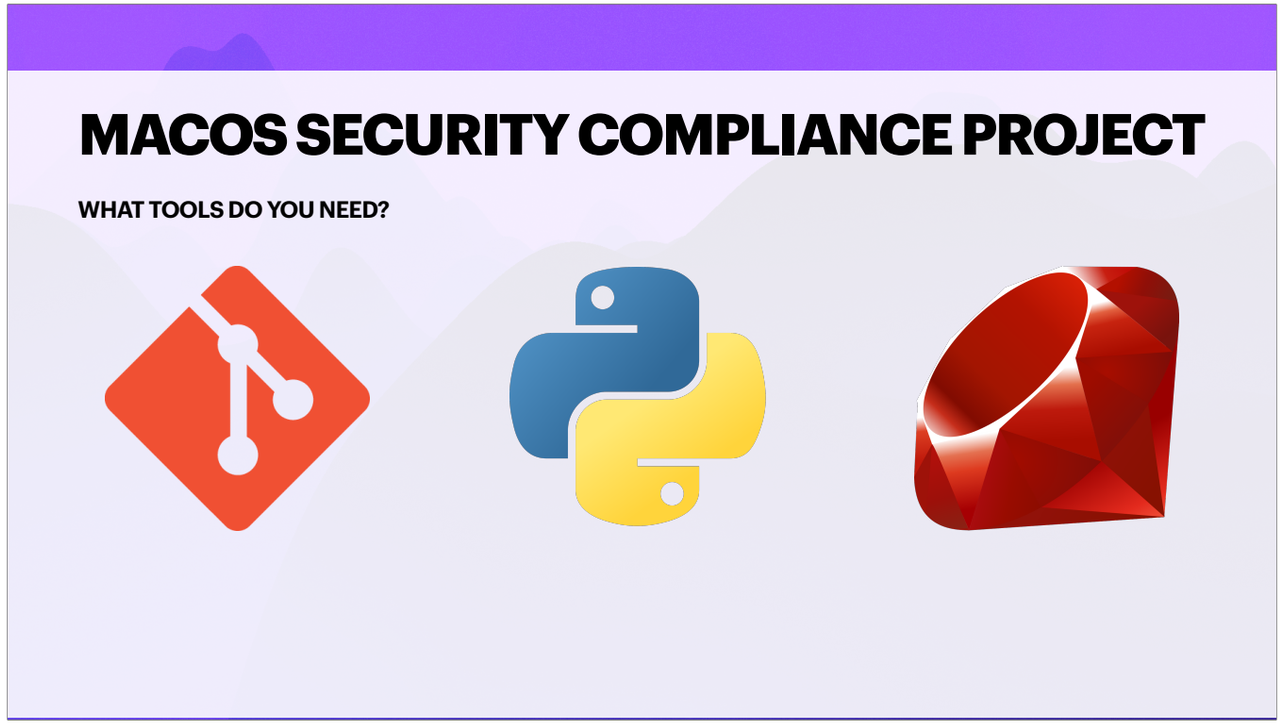
It’s vendor agnostic, and contains several baselines and benchmarks out of the box, more on that later.

I’m calling this the foundational rule-set for our game. Though I have other thoughts about that next…nerdy thoughts.

The macOS security compliance projected started as a joint effort between some individuals from NIST, NASA, DISA, and Los Alamos national laboratory.

Over time, some new orgs got involved with development including my company, Leidos.
So This project isn’t just a joint government agency project, it crosses government and commercial bridges.

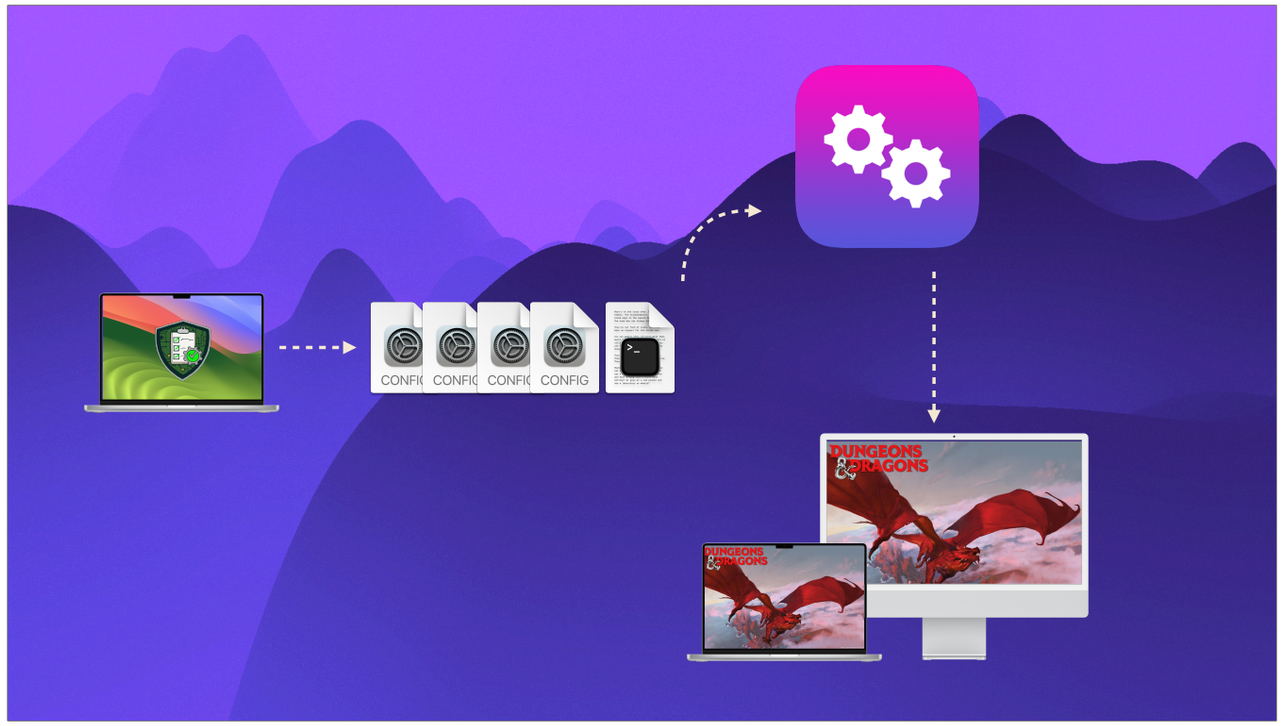
NIST SP 800-53 rev5 Low, Moderate, and High
NIST SP 800-171 Rev2
CNSSI-1253
DISA STIG
CIS Level 1 & 2
CMMC Level 1 & 2 (Ventura Only)
CNSSI (Ventura Only)
CIS Controls Version 8
Custom Baseline creation
All Rules Baseline for everything in this Repository



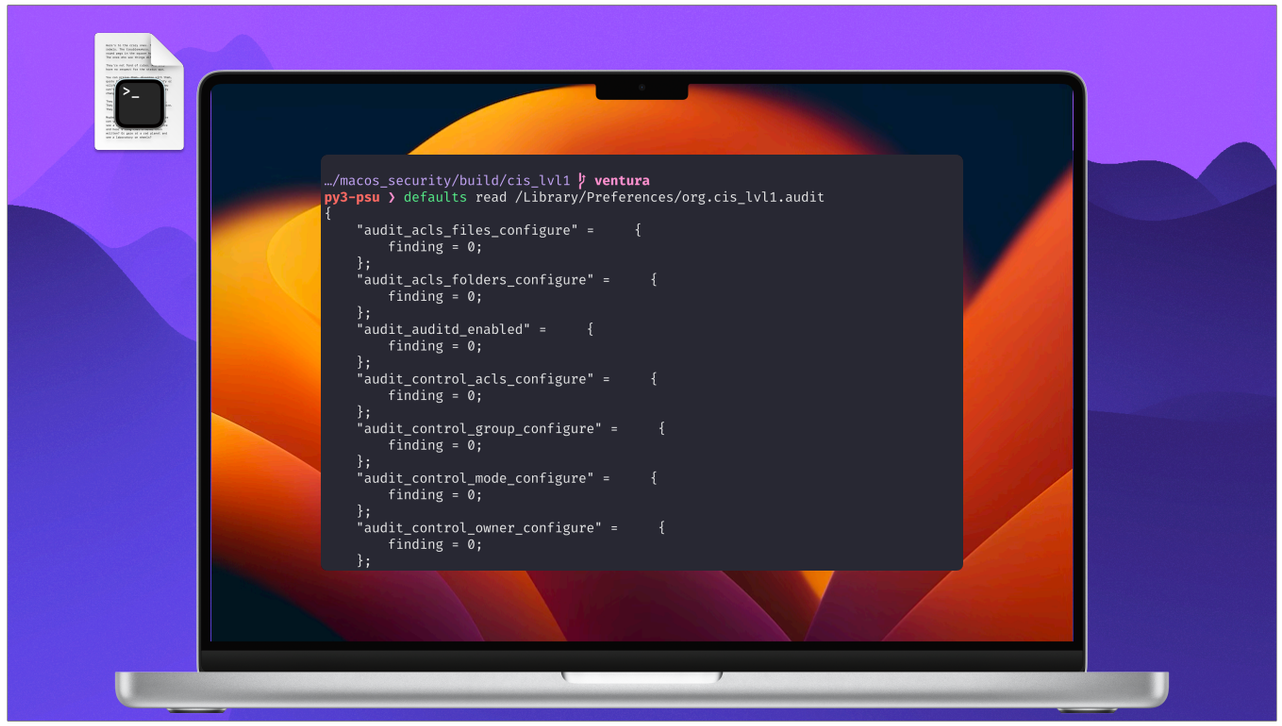
What tools do you need for the MSCP? Well, just three: git, python, and ruby.

When I say that you need these, you only need this is you are using the MSCP git code to make your guidance..and only when making your guidance and scripts. These are **not** needed on your endpoints to run the code. The code is all z-shell and config profiles.

A very very fast explanation of how the MSCP works. You take the built MSCP guidance from your admin system..

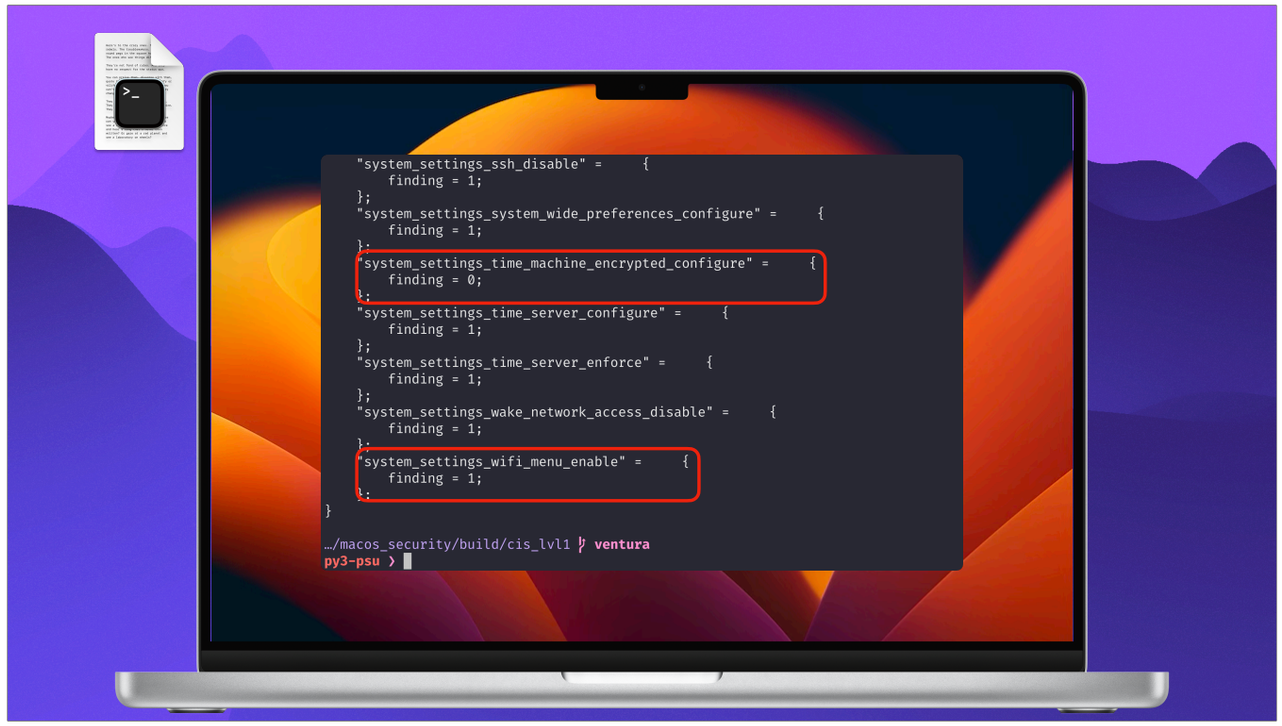


CLICK It creates config profiles and a script..you upload to your MDM of choice, then deploy and run on your fleet.



When the script is run on your fleet…it creates an audit file in /Library/Preferences. The file is usually named “org.baselinename.audit”

If you read the, you’ll see a list of all of the rules in your baseline



Scrolling through, you can see each one shows **CLICK**

Finding = 0 (Passed)
Finding = 1 (Failed)

So that’s how MSCP works in a nutshell…

JAMF COMPLIANCE EDITOR

THIS IS OUR VIRTUAL TABLETOP PROGRAM

GUI Application from Jamf

Can be used with any MDM

Added Bonus if Jamf Pro Customer

No CLI Necessary

Download at Jamf: https://trusted.jamf.com/docs/establishing-compliance-baselines

Next, Jamf compliance editor (or JCE)

JCE is a GUI app developed by Jamf that can be used to generate and customize your MSCP guidance. It is not open source, but it is free to **all** and will work with any MDM. The benefit if you're a Jamf customer is that it has functionality to upload to your Jamf server directly in the application.

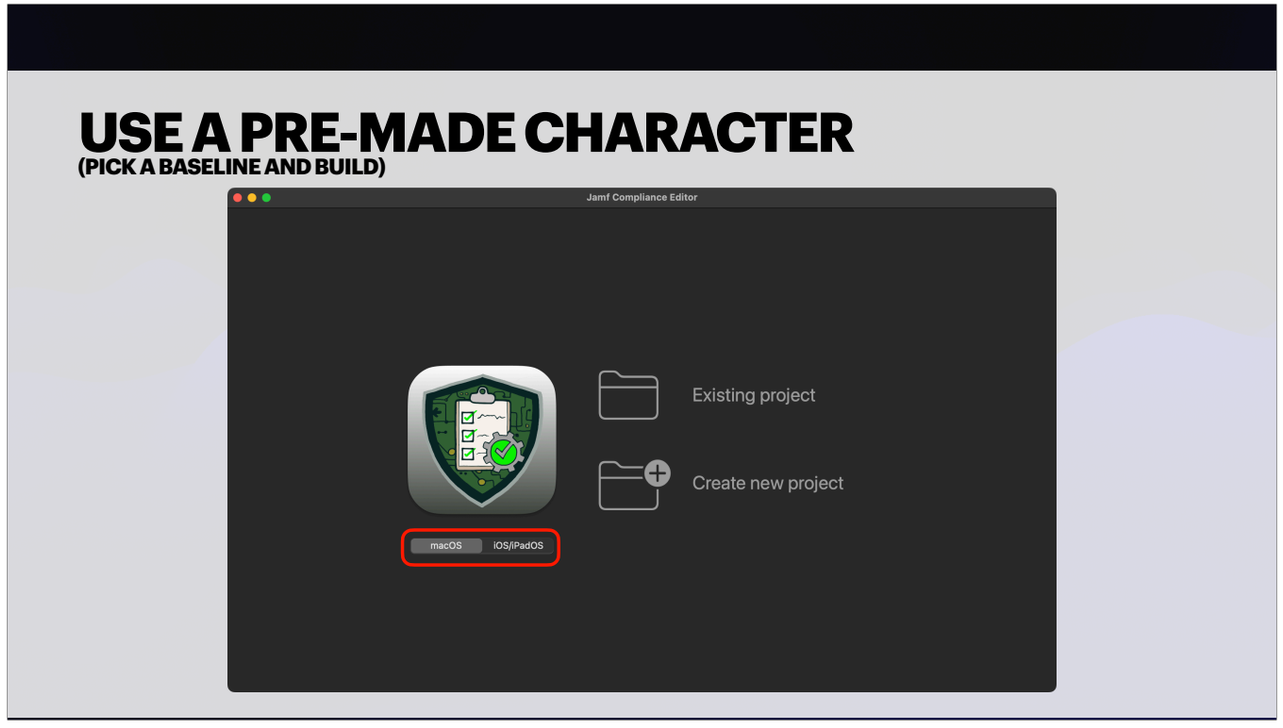
This tool requires no CLI or coding experience, and no pre-reqs to install. This is what we'll be focusing on today but I will suggest that if you're really wanting to know how the project works, try cloning the git and working on it there sometime.

This is our "virtual table top" that manages all of the rules and complexities behind the scenes for us. Though, my other analogy is that this is D&D5e…and MSCP is Pathfinder. Any Math finder fans here? 1e or 2e?

1e…why?

Okay, lets take a look at our pre-made character

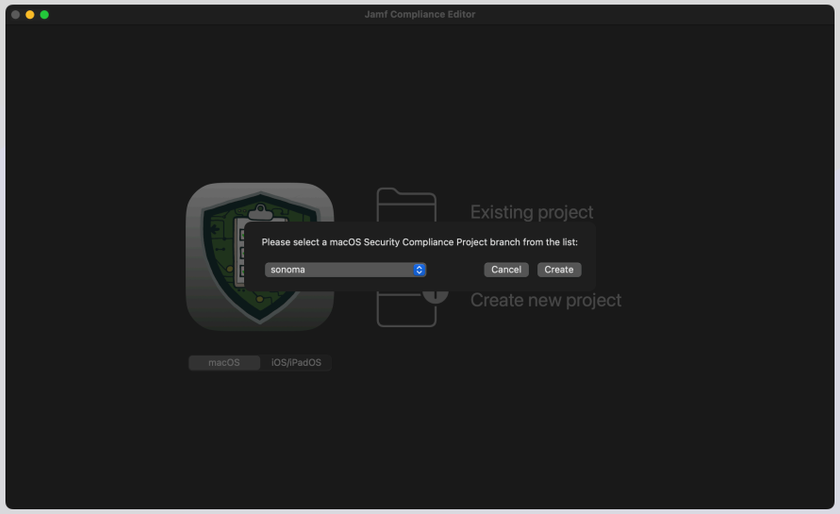
I was told not to do live demos…so I have lots of screenshots, but I will try a live demo one time and see if the works….



Opening JCE we’re greeted with a few selections. **CLICK** We can see we are offered macOS..and iOS/iPadOS. Yes, MSCP does offer guidance creation for iOS and iPadOS. I will not be covering that in this session, but if you security mobile devices, you can get start with MSCP.

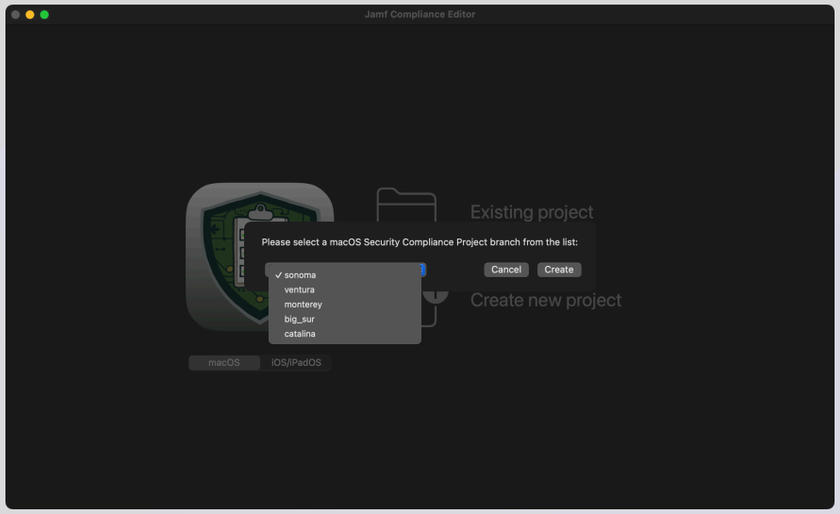
We’re going to keep it macOS then click “Create New Project”

USE A PRE-MADE CHARACTER
(PICK A BASELINE AND BUILD)



Here we are asked which branch we want to use from the list.

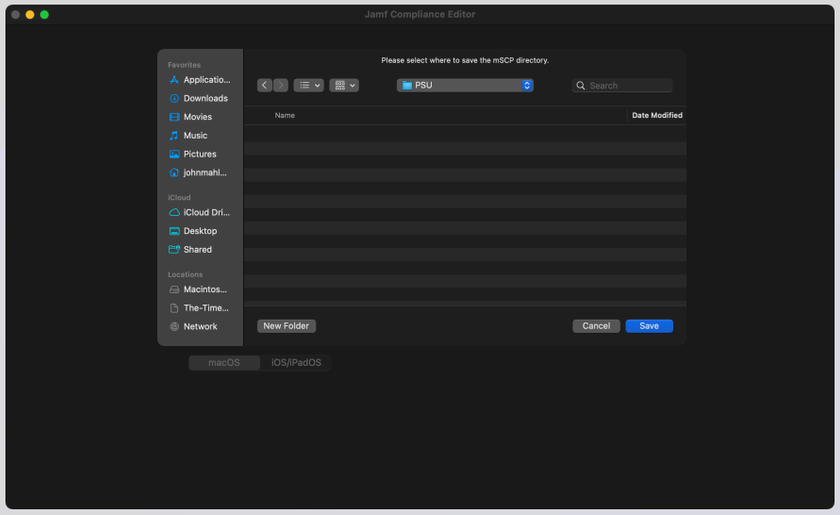
USE A PRE-MADE CHARACTER
(PICK A BASELINE AND BUILD)



Clicking the dropdown we can see all of the OSes that the project supports. Please note that the only safe choices here are N-2 options as previous OSes are no longer maintained by Apple nor are the MSCP branches.

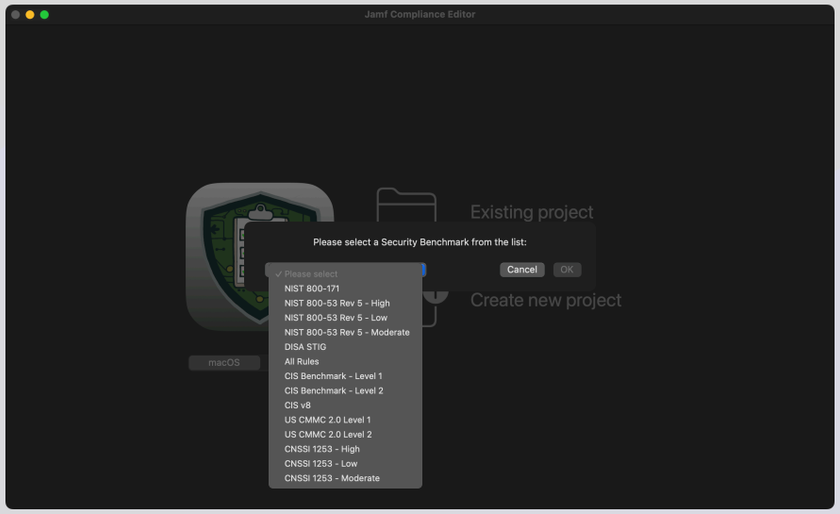
For those of you wondering, this list comes from the GIT branches.

USE A PRE-MADE CHARACTER
(PICK A BASELINE AND BUILD)



After selecting, we'll be asked to save the mscp directory somewhere. After I click save…

USE A PRE-MADE CHARACTER
(PICK A BASELINE AND BUILD)



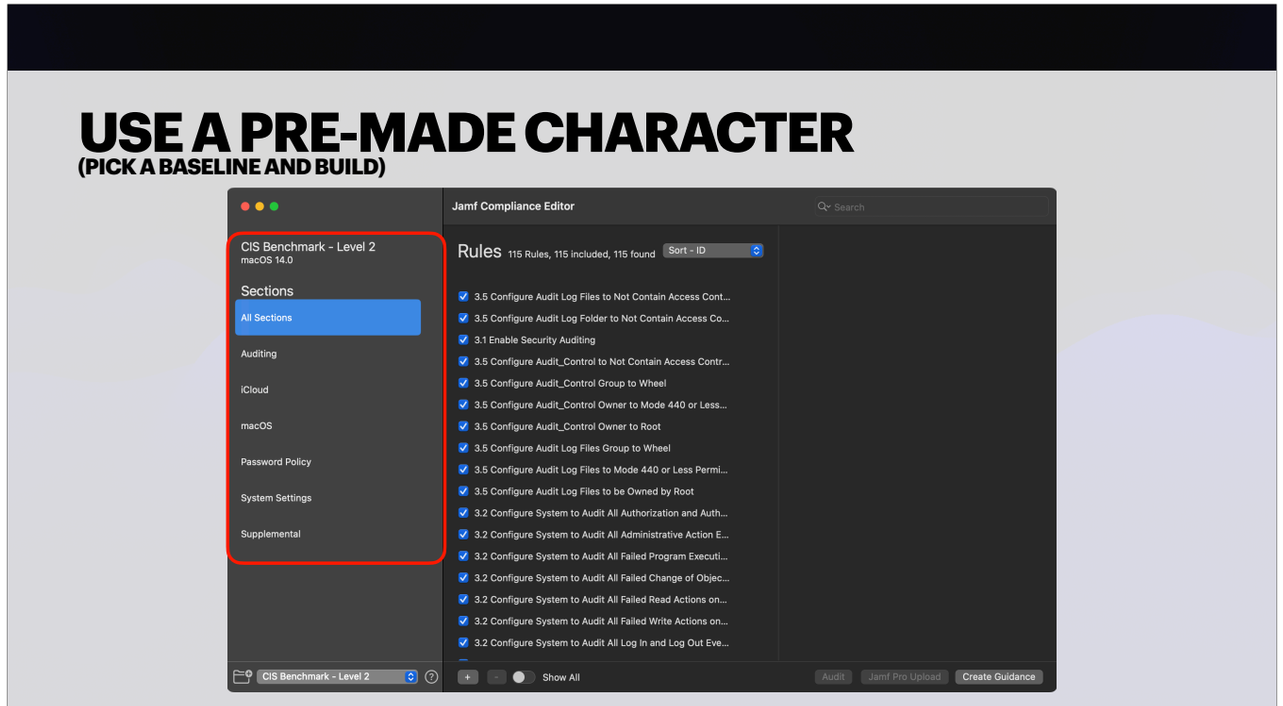
I'm then asked to select a security benchmark. This is where you select your class of character. As mentioned perviously, the MSCP supports a good deal out of the box. There is also an "All Rules" option which you should never select and implement because you will make your Mac worthless and you will regret playing this game just like a ranger in 5e. Just make a fighter…your party will thank you…..

Anyway…we're going to select CIS level 2, the Fighter of benchmarks because its popular and the most versatile and forgiving for many organizations.

When we discuss baselines or benchmarks, we sometimes interchange which word we are using when referring to them.
Baselines provide a starting point for organizations in the security and privacy control selection process. NIST Special Publication 800-53B includes three security control baselines (one for each system impact level: low, moderate, and high), as well as a privacy control baseline that is applied to systems irrespective of impact level.
The rules in the project are mapped to the 800-53 security controls to identify how macOS can be configured to meet the control.

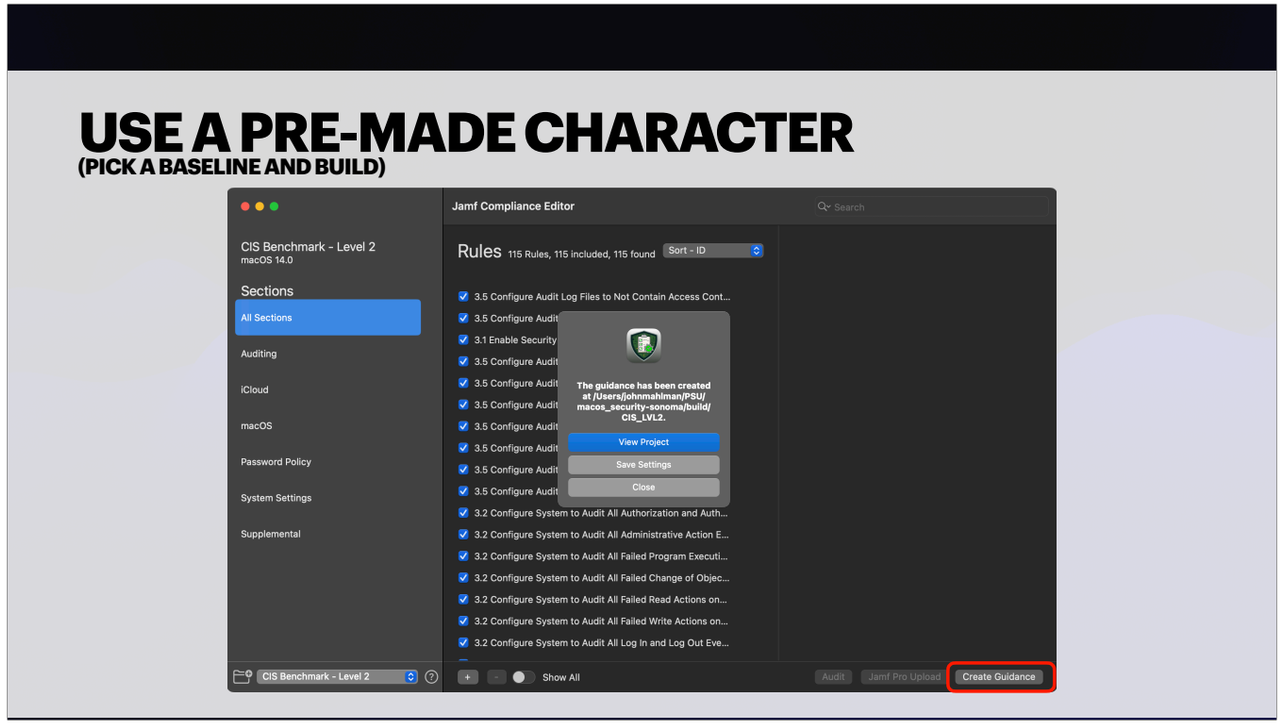
Benchmarks, on the other hand, define which controls are to be included along with the organization defined values (ODVs) recommended by the benchmark's authors.
The DISA STIG, for example, is basically a selection of controls and values outlined by the DOD to meet their security requirements.

As part of tailoring a baseline or benchmark, when you are selecting which rules to include, and defining the values that are to be used… you are, in effect, creating your own organization's benchmark. More on tailoring later!



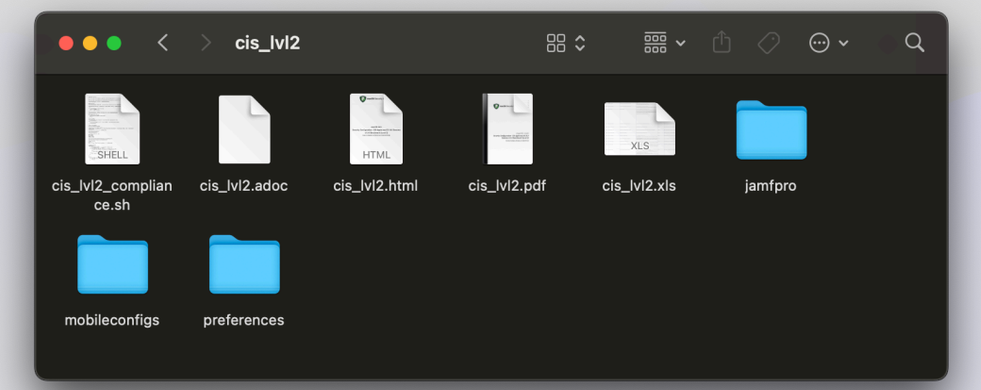
And after a few seconds, we have our benchmark rules! You can see on the **CLICK** left here we have rules broken into sections. The sections are also derived from the Git code and are as such:

Auditing: configuration and enforcement of the OpenBSM settings
Authentication: configuration and enforcement of smartcard authentication (this is not in CIS level 1-2)
iCloud: configuration of Apple’s iCloud/Apple ID service
macOS: rules to configure the operating system that are not defined within other categories of the rules directory
Password Policy: configuration and enforcement of password policy
Supplemental: additional information to support the guidance provided by the baselines
System Settings: configuration and enforcement of settings controlled within the System Settings application

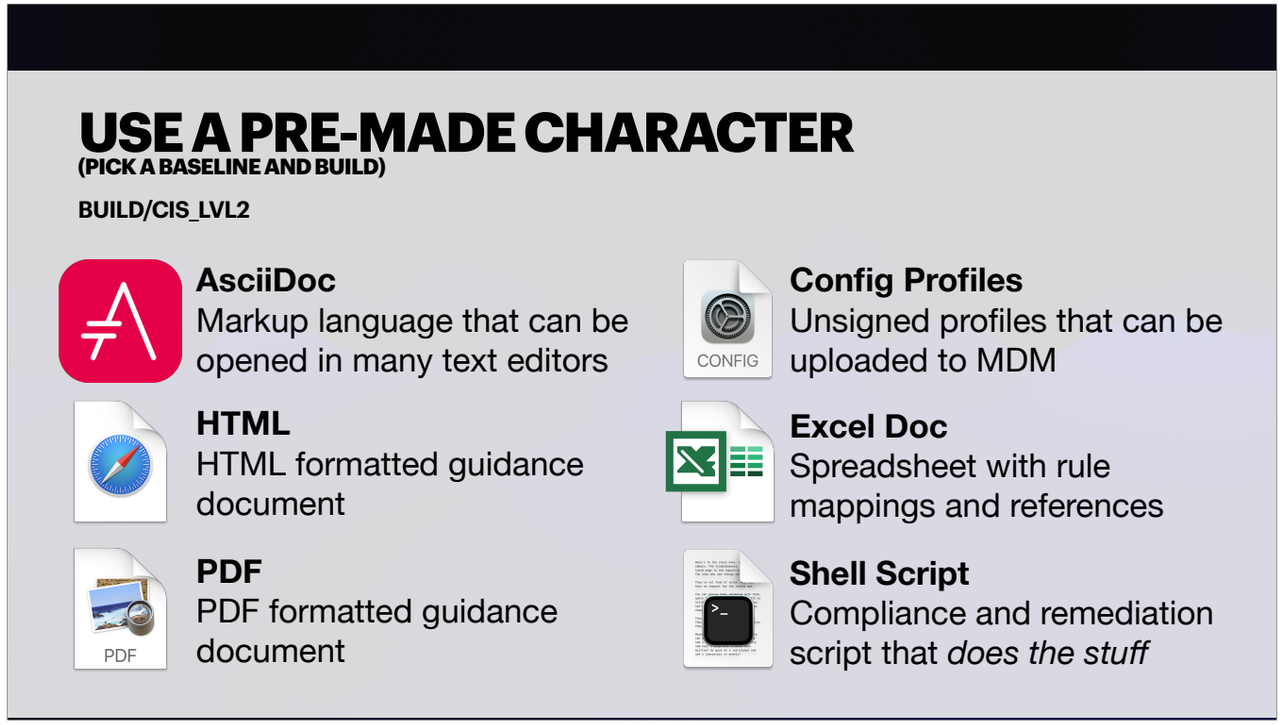


Since we’re using a pre-made character, we’re just going to **CLICK** “Create Guidance” which will do so then show this prompt. We can “Save Settings” which will save a .JCE file that saves our settings, or we can “View Project” which will open the project folder. Lets do that.

USE A PRE-MADE CHARACTER
(PICK A BASELINE AND BUILD)



Here is our project folder….lets take a look a little at what we have in here…



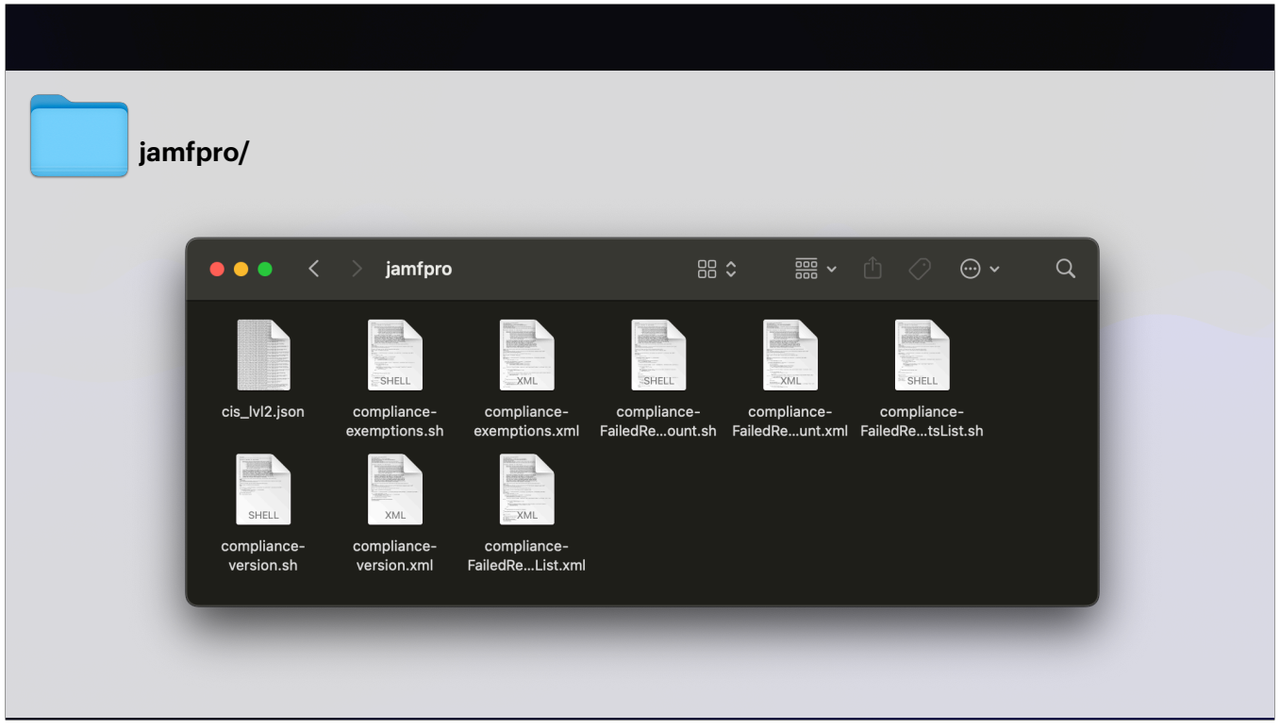
You can see that you get a number of documents built by default when you create guidance. Most of these are really useful for providing to auditors and your security people so they have reference as to how things are implemented. I'll show some of these throughout the presentation…

The most important one we have though is the shell script.

cis_lvl2_compliance.sh

```
#!/bin/zsh --no-rcs

##  This script will attempt to audit all of the settings based on the installed profile.

##  This script is provided as-is and should be fully tested on a system that is not in a production environment.

##################  Variables  ##################

pwpolicy_file=""

##################  DEBUG MODE - hold shift when running the script  ##################

shiftKeyDown=$(osascript -l JavaScript -e "ObjC.import('Cocoa'); ($.NSEvent.modifierFlags & $.NSEventModifierFlagShift) > 1")

if [[ $shiftKeyDown == "true" ]]; then
    echo "~~~~~DEBUG~~~~~"
    set -o xtrace -o verbose
fi

##################  COMMANDS START BELOW THIS LINE  ##################

## Must be run as root
if [[ $EUID -ne 0 ]]; then
    echo "This script must be run as root"
    exit 1
fi

ssh_key_check=0
if /usr/sbin/sshd -T &> /dev/null || /usr/sbin/sshd -G &>/dev/null; then
    ssh_key_check=0
else
    /usr/bin/ssh-keygen -q -N "" -t rsa -b 4096 -f /etc/ssh/ssh_host_rsa_key
    ssh_key_check=1
fi

# path to PlistBuddy
plb="/usr/libexec/PlistBuddy"

# get the currently logged in user
CURRENT_USER=$( /usr/sbin/scutil <<< "show State:/Users/ConsoleUser" | /usr/bin/awk '/Name :/ && ! /loginwindow/ { print $3 }')
CURR_USER_UID=$(/usr/bin/id -u $CURRENT_USER)

# get system architecture
arch=$(/usr/bin/arch)

# configure colors for text
RED='\e[31m'
STD='\e[39m'
GREEN='\e[32m'
YELLOW='\e[33m'

audit_plist="/Library/Preferences/org.cis_lvl2.audit.plist"
audit_log="/Library/Logs/cis_lvl2_baseline.log"

```

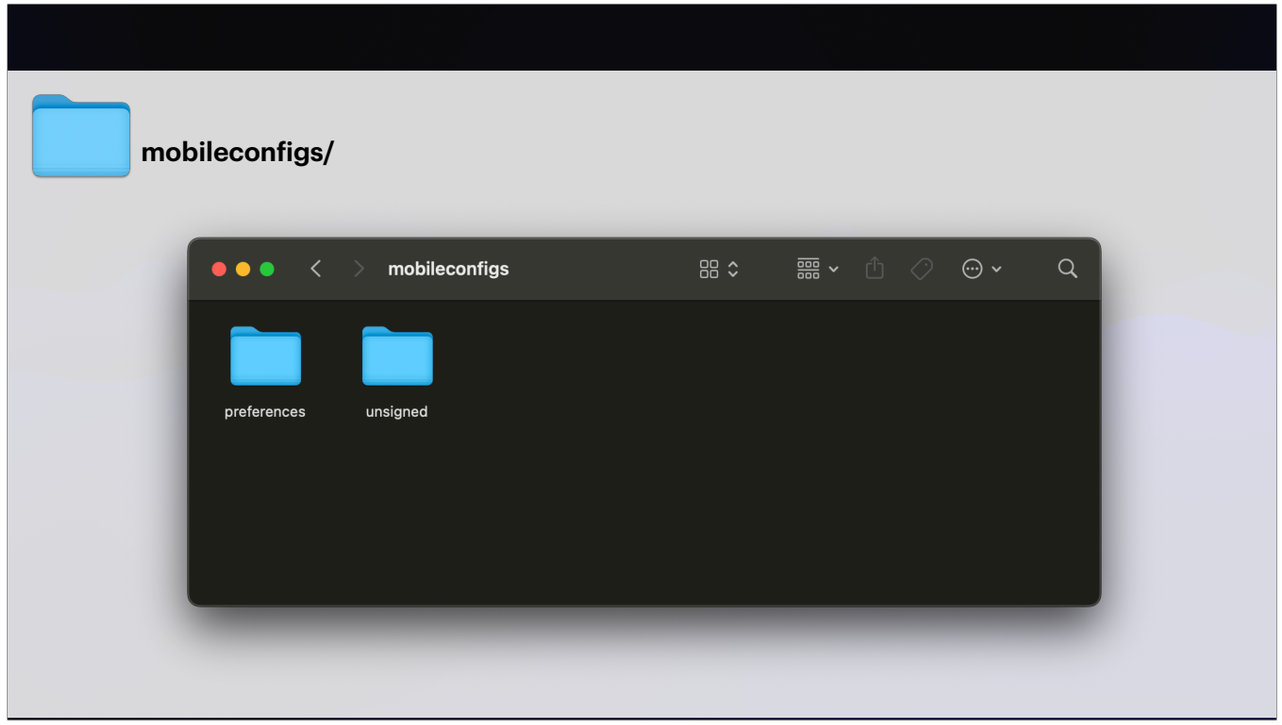
This is a multi-thousand line script that contains all checks and most fixes for the benchmark settings. I say “most fixes” because many are taken care of by a config profile, so the script only runs a check to make sure the config is installed. We’ll be using this script later for policies.



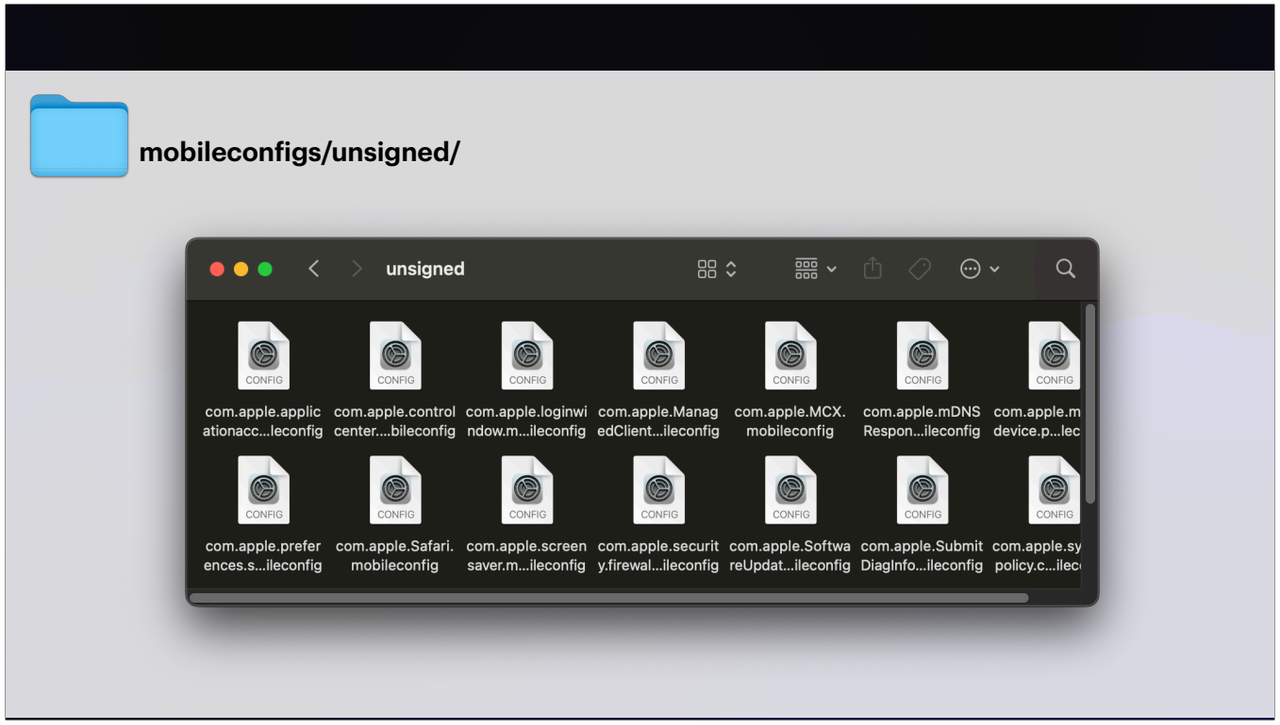
Next lets dig into the folders. This folder, “jamfpro” is one of the items that is created specifically by JCE and not part of the MSCP.

This folder contains shell scripts and XML files for the extension attributes we’ll use later. When you upload a script to Jamf, it wants an XML for all of the additional info (name, category, etc), so thats why there is one of each for each EA.

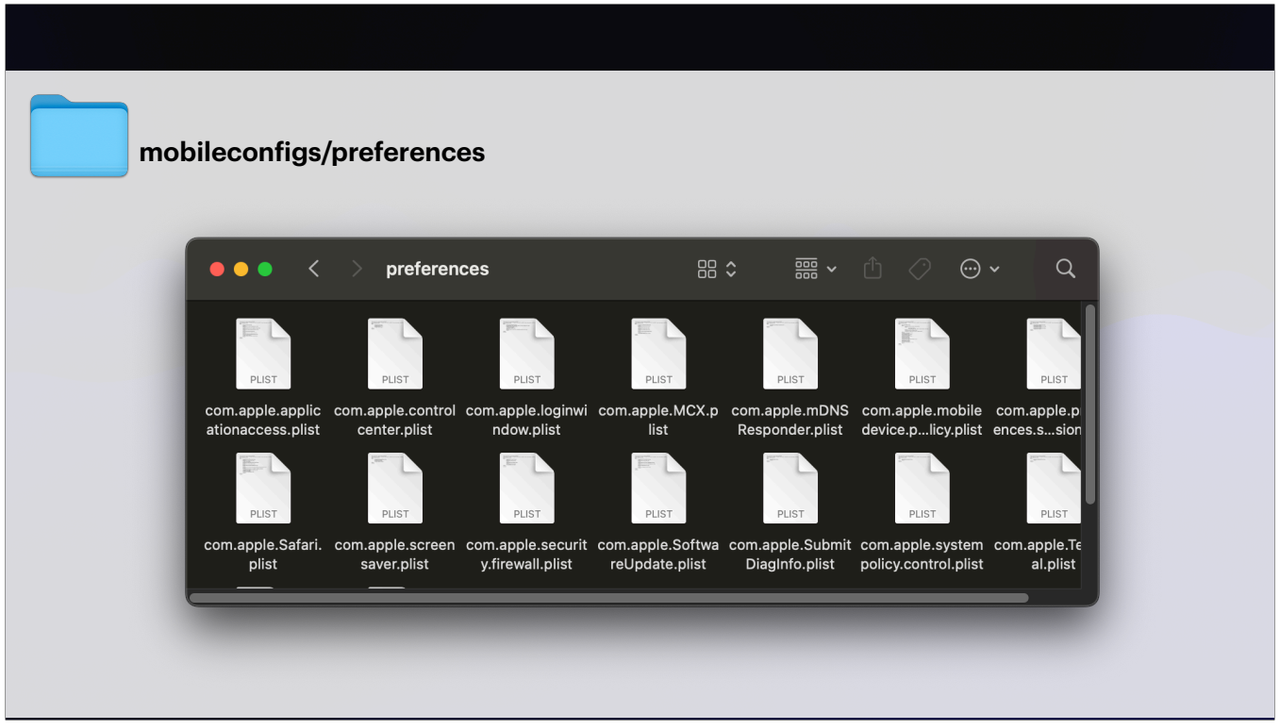
There’s also a JSON in here. That’s a custom schema you can add for **exclusions**…something we’ll show later.



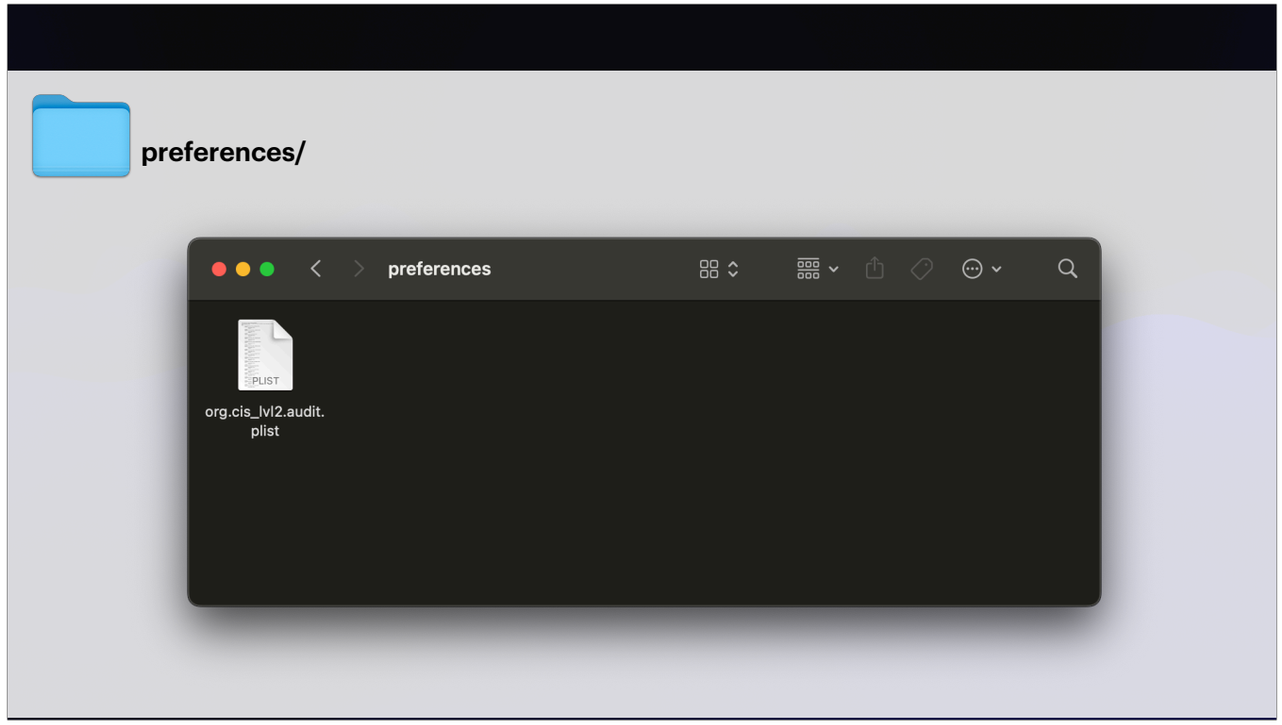
Next is “mobile configs”. This has two folders off, you guessed it, mobile configs.



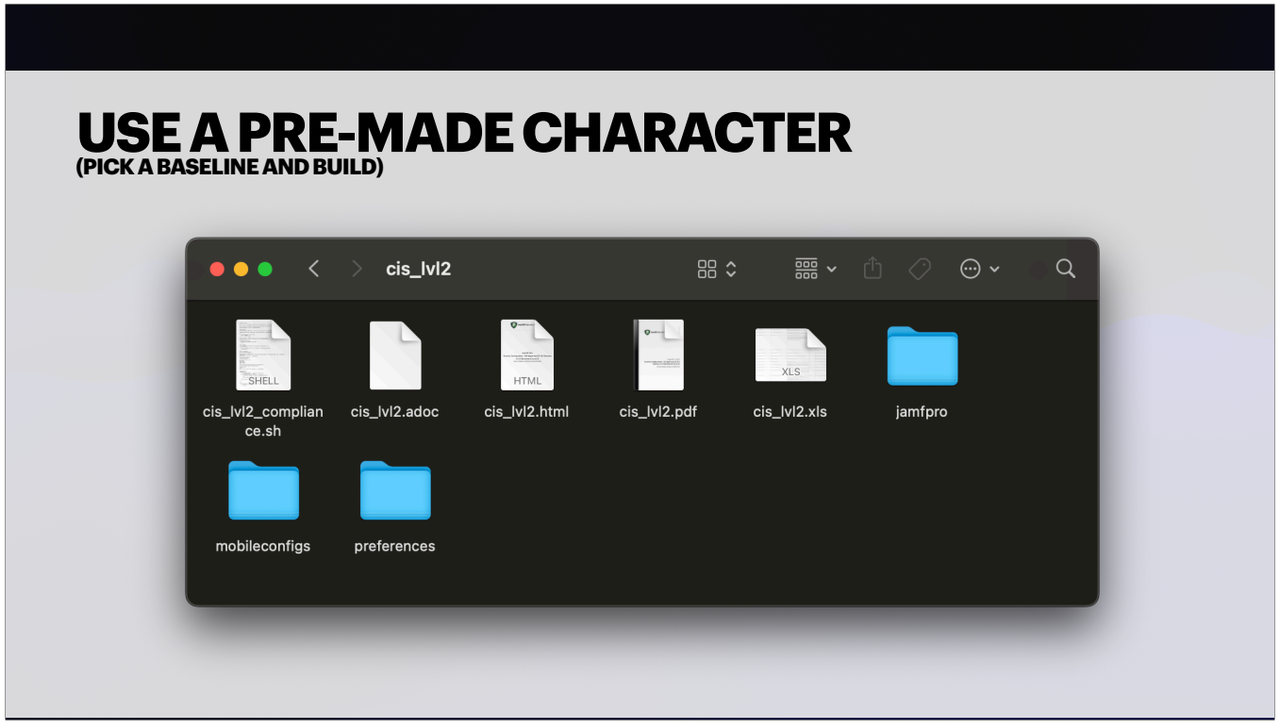
First is the unsigned folder which has unsigned config files you can upload to whatever MDM you use.



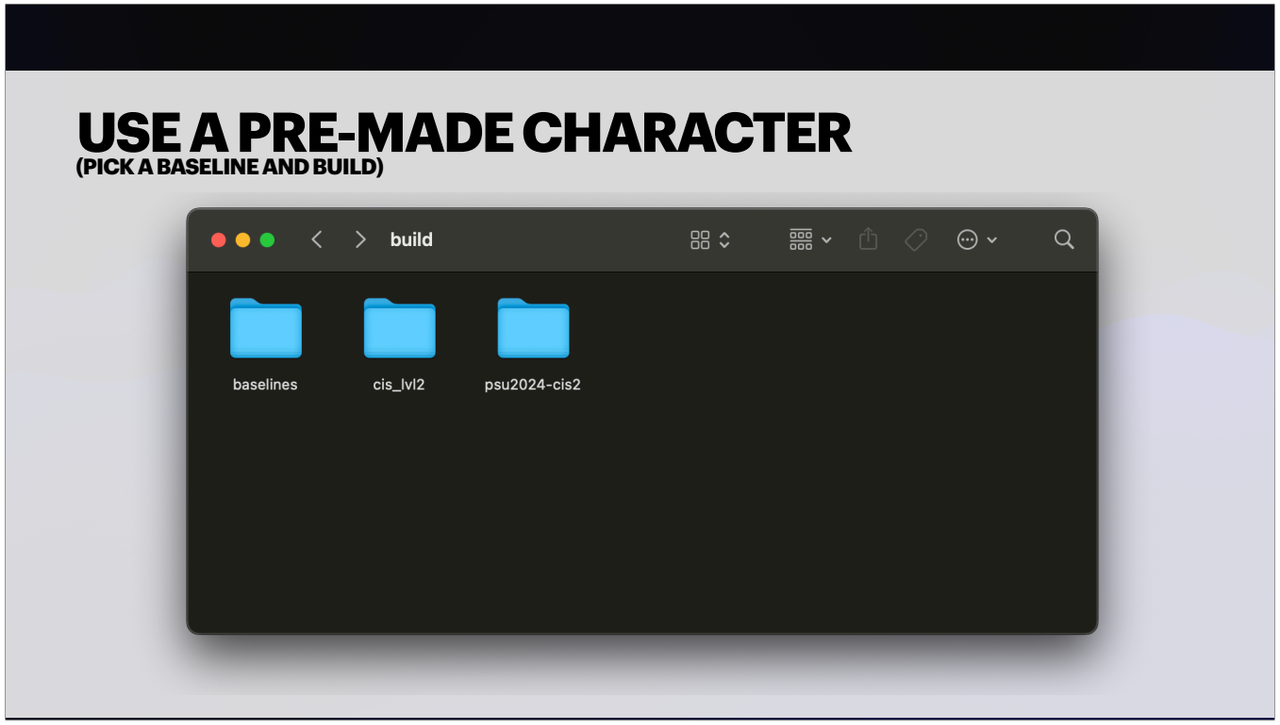
Then the “preferences” folder contains the raw PLISTs for those mobile configs in case you cant or don’t want to upload profiles. These are also useful if you want to inspect everything in VScode or something.



And lastly, we have “preferences” which is just a raw plist for our exclusions (in case you dont want to ue the JSON schema we saw before).

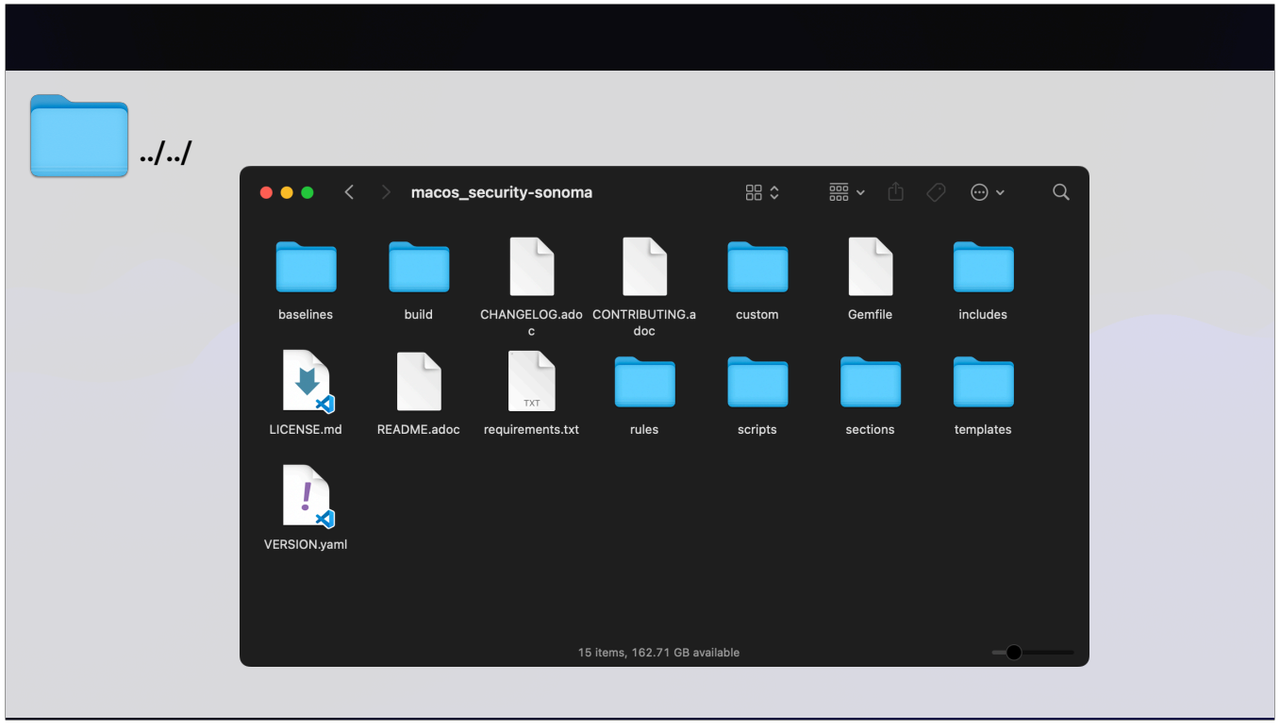


So that’s the project folder. You might be wondering…where is this folder? Well..if you go back in your directory structure…



Well, we go back one…we're in "build"…okay..but where is this?
If you pass a perception check you know where this is going I think…

Going back one more directory…



Oh! Look at that, you’re actually looking at the MSCP repo which was pulled locally!

ROLL YOUR OWN CHARACTER
(CUSTOMIZE THE BENCHMARK)

Now that you've tried a pre-made…let's make your own character sheet!

ROLL YOUR OWN CHARACTER

CUSTOMIZE THE BENCHMARK

Remove rules from the check and fix

Alter/Update rules based on organizational defined values (ODVs)

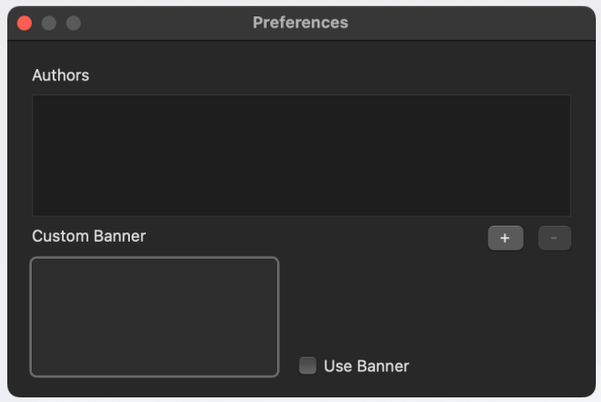
Change checks based on your organization tools

Change verbiage of descriptions

Customizing the benchmark is really where things get fun. When customizing (also known as tailoring in the MSCP), you are selecting which rules to include, and defining the values that are to be used… you are, in effect, creating your own benchmark for your organization

I will also say that you will most likely have to customize/tailor EVERY baseline and benchmark because of how some checks work. We'll go over some of those in this.

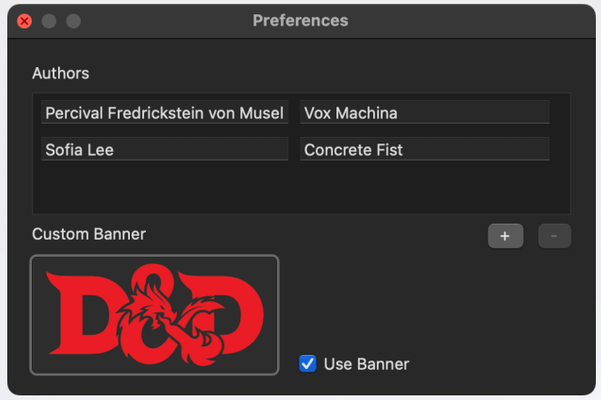
JCE SETTINGS



The first thing we want to do..is configure JCE settings. This also configures items for the baselines! So we'll go to Settings and we'll see the "Authors" and "Custom Banner" items.

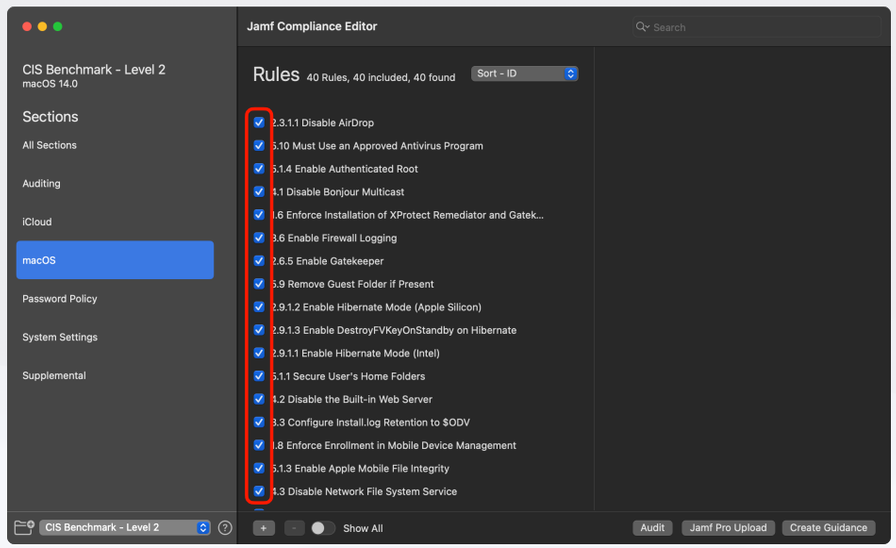
Whenever you customize a baseline, you should add yourself as an author of the baseline. This is only for documentation purposes but is important if you're handing your documentation off to someone else. And the custom banner is where you can add your organization's logo or another banner..so lets add some there.

JCE SETTINGS



There we go…and we'll check "Use Banner" and you'll see where this comes in later.

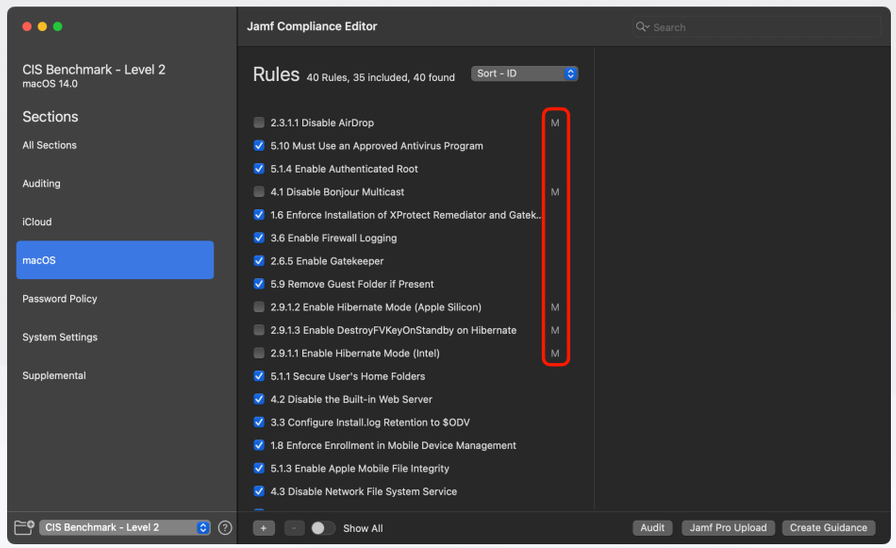
REMOVE RULES



Next, we’re going to remove some rules from our benchmark. We’ll **CLICK** the checkboxes next to…

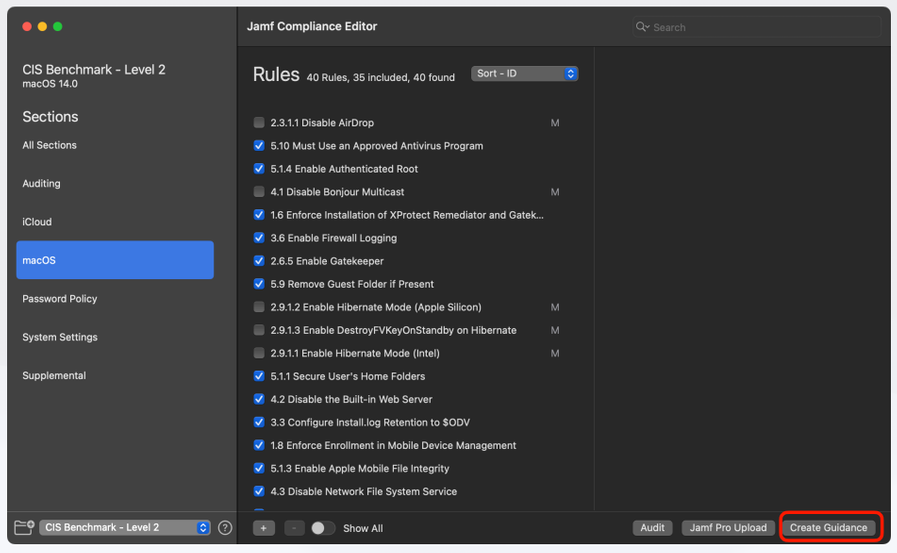
Disable Airdrop, disable bonjour, and all of he hibernate mode ones because they are bad.

REMOVE RULES



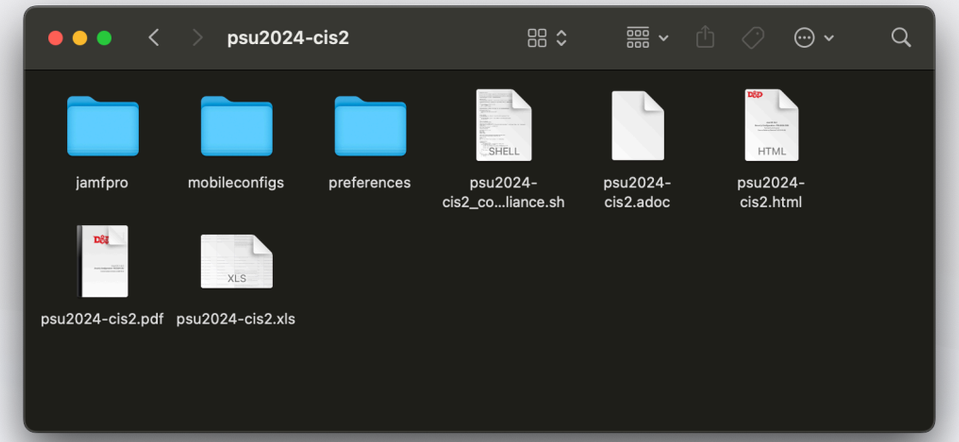
Now that we have done that, you can see the “M” next to those rules. That “M” means the rule was modified in some way. In our case, re removed them!

REMOVE RULES



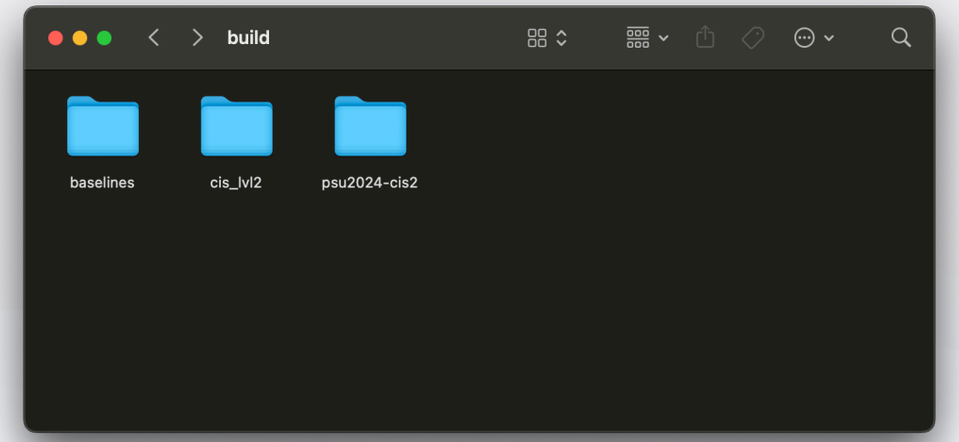
Now we'll hit "Create Guidance" again, it'll ask to save and we'll save as psu2024-cis2 when prompted….and we'll take a look at what that creates..

REMOVE RULES

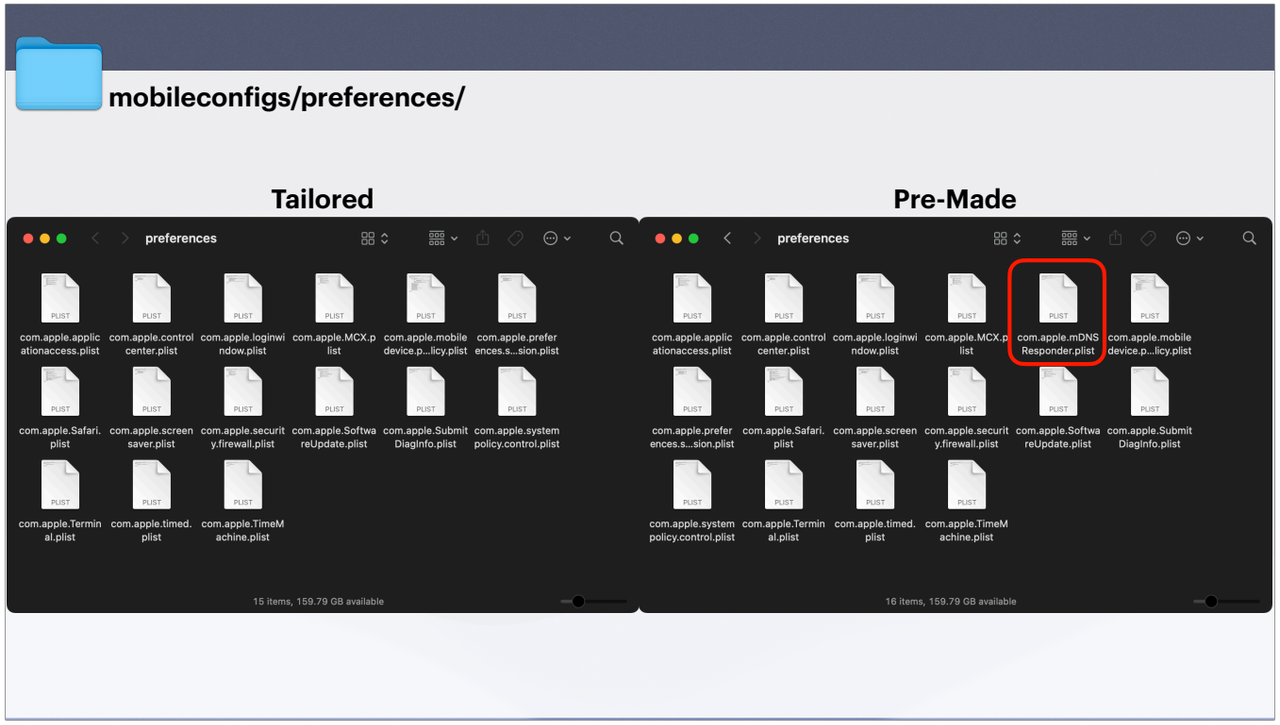


Oh look, the same thing! Except it’s not! It’s now tailored! And renamed…

REMOVE RULES

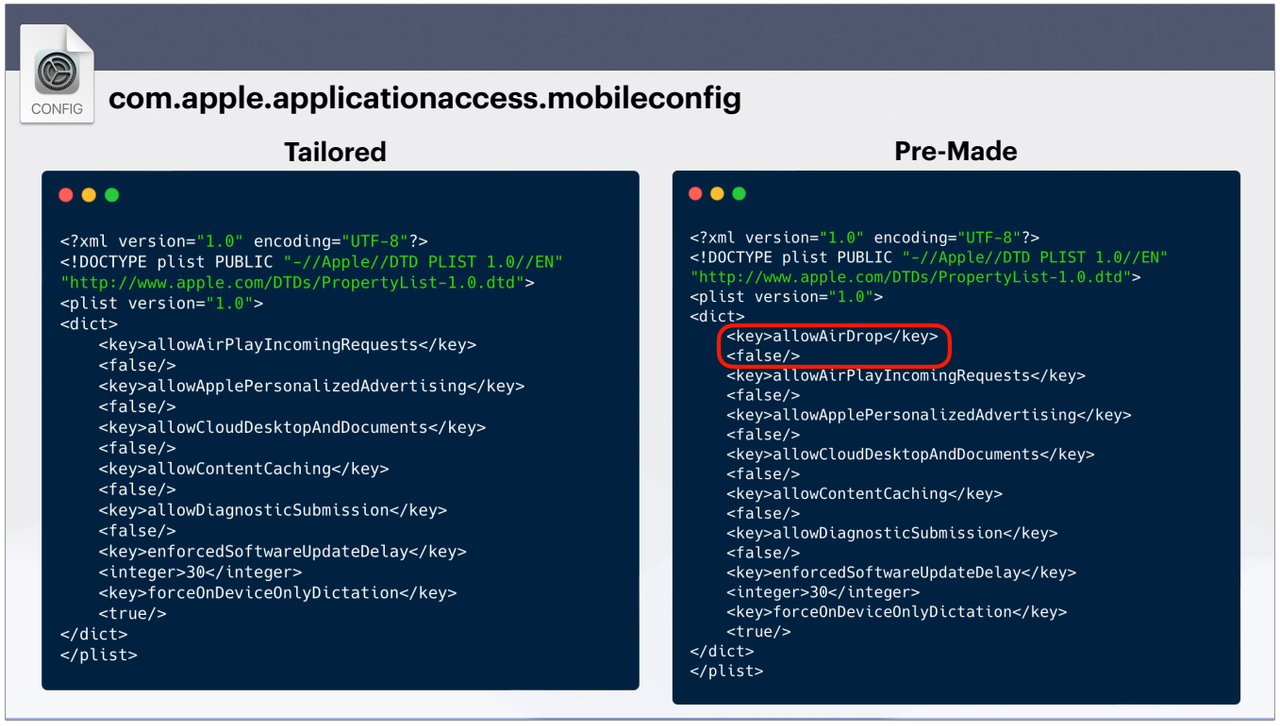


First, just to show you where we are…if I go up one folder you'll see the builds…the first one we made (cis_lvl2) and now this new one… (remember, this build folder is in the root of the git directory)

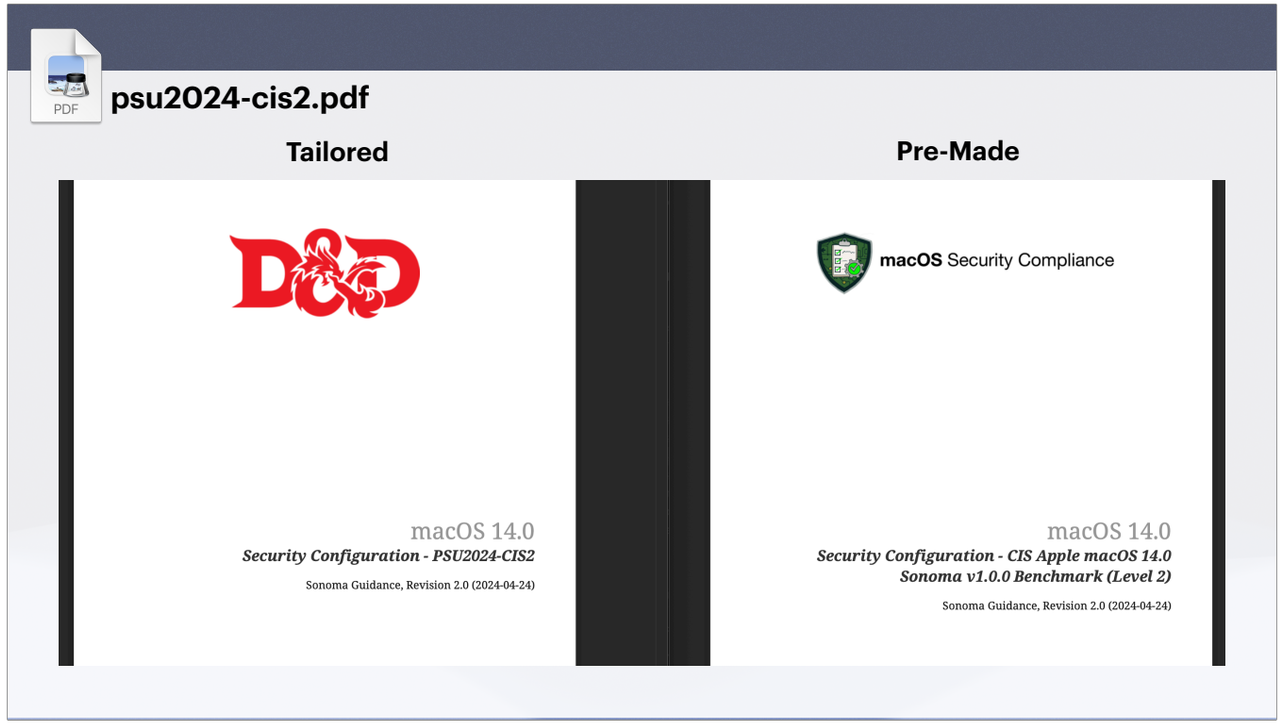


Lets dive into some changes…first, in the mobileconfigs/preferences folder…on the right is our pre-made and you can see compared to our tailored… **CLICK**

We have the MDNS responder plist, which is bonjour. We removed that in the tailored, so it’s not there.

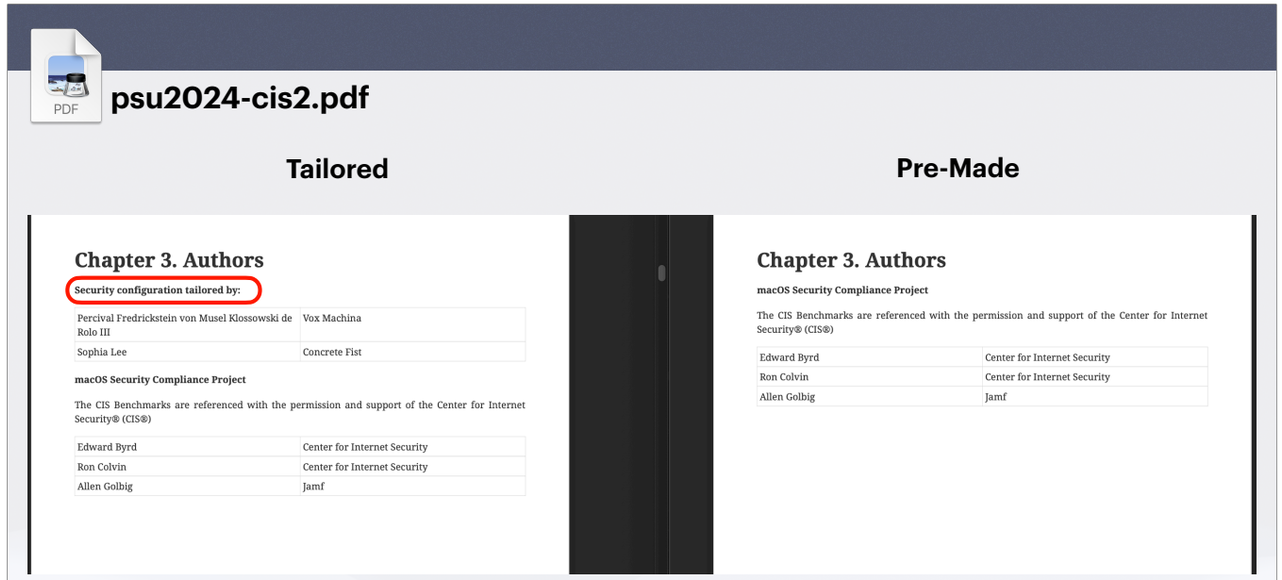


Looking into the com.apple.applicationaccess mobile config (or the preference plist for it) we can see **CLICK** Allow airdrop is in our pre-made on the right but not the tailored one on the left.

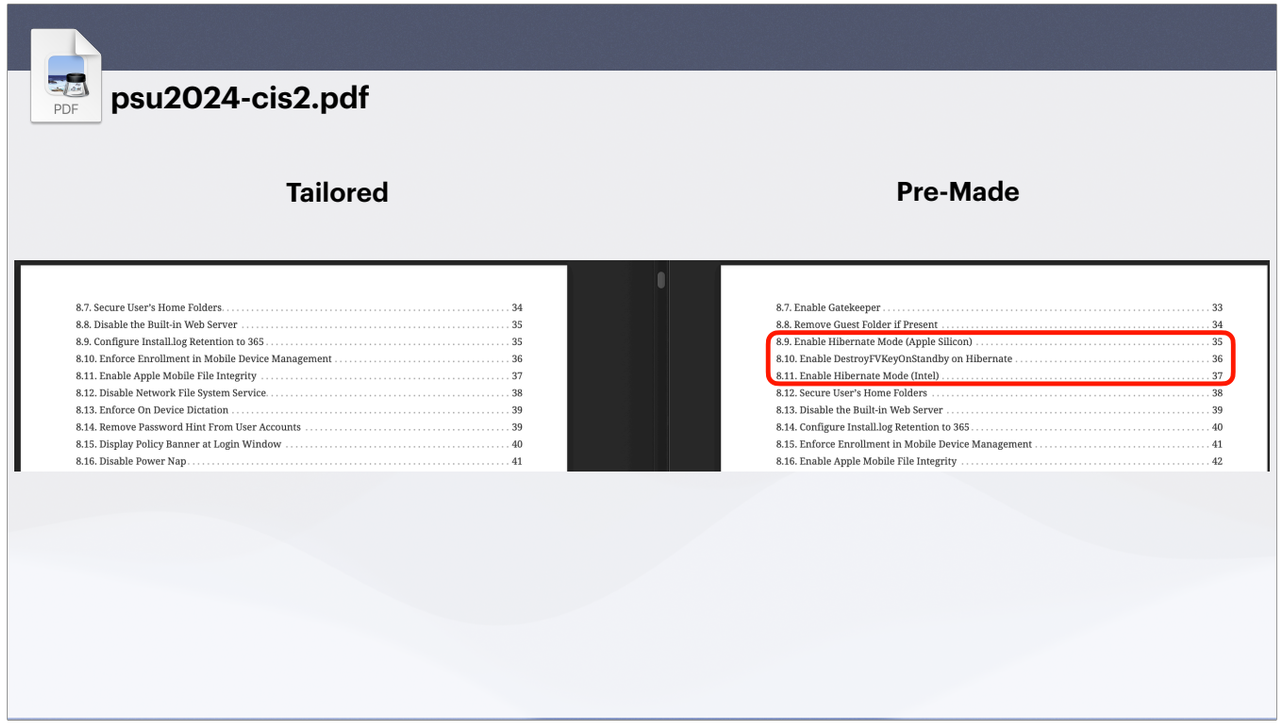


Now lets look at some documents. Here is our PDF…its really great that this gets created automatically and updated every time you change something and generate guidance.

On the left if our tailored one with our company logo and the name of our config.

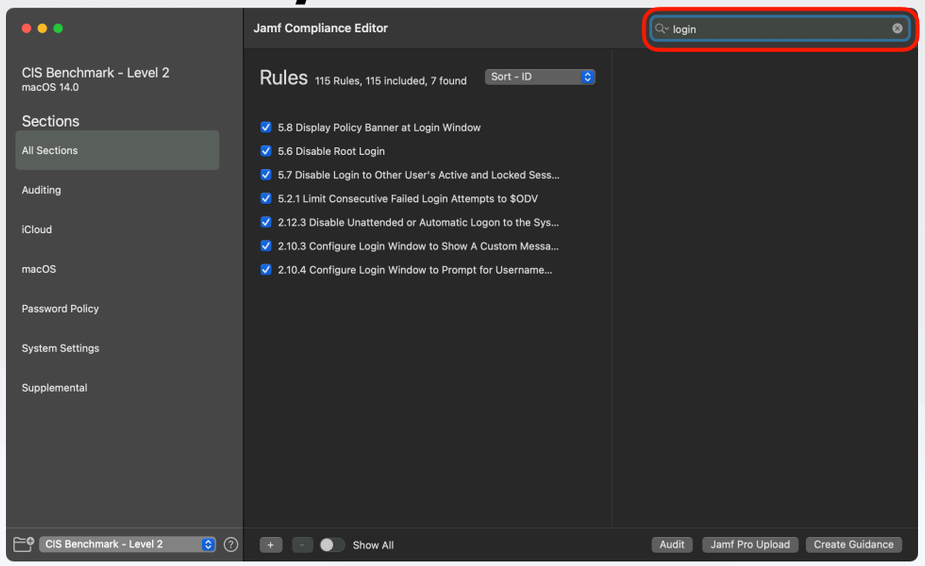


Checking the authors…you can see our two benchmark authors are in there…awesome. You can see the heading states **CLICK** “Security configuration tailored by..”



And finally, you can check the table of contents…and you'll see **CLICK** the pre-made has those hibernate mode settings while our tailored one does not.

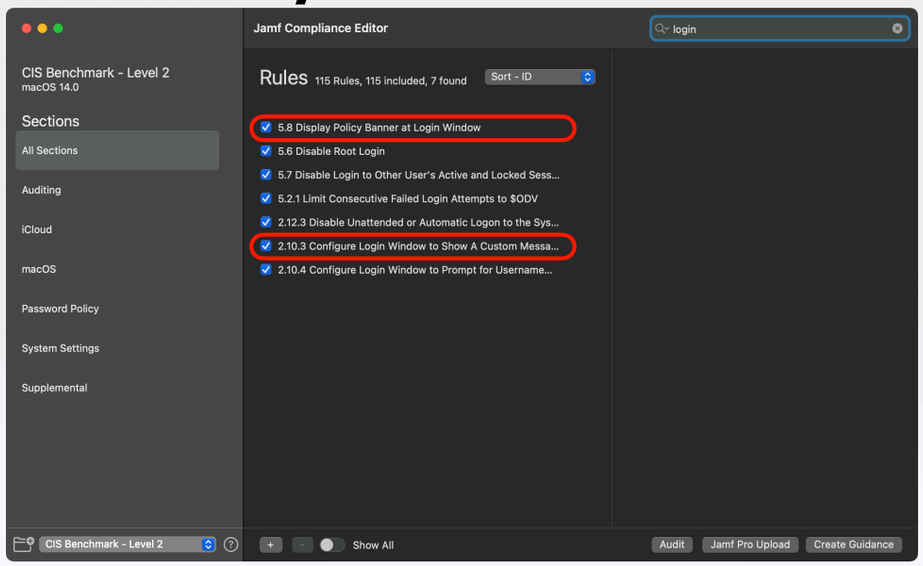
UPDATE RULES/ODV



Now what about updating rules or changing rules to meet your organizational standards.
There are several reasons to do this, and there are some rules that are designed with these in mind. They are expecting something called an organizational defined value or ODV.

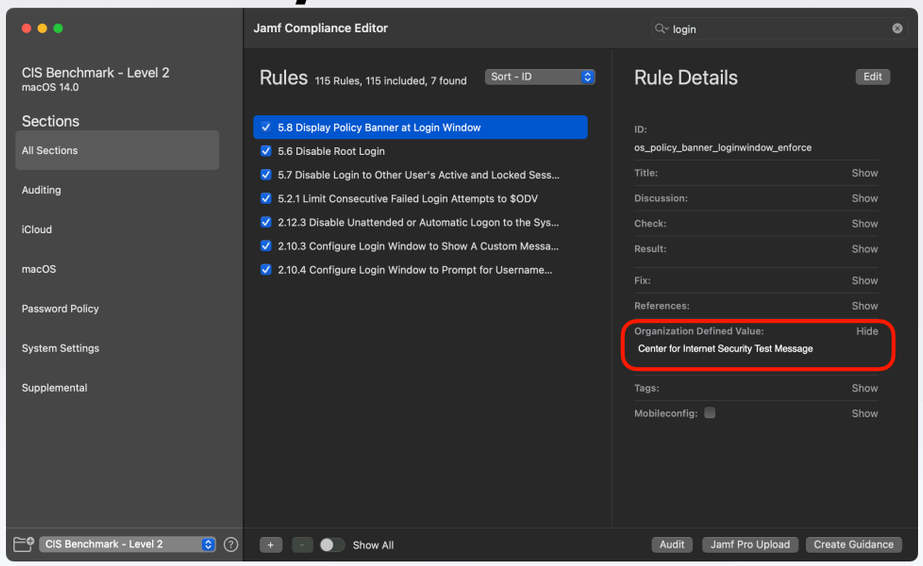
Let’s take a look at two rules that required updates. We’ll search **CLICK** “login” in the search box.

UPDATE RULES/ODV



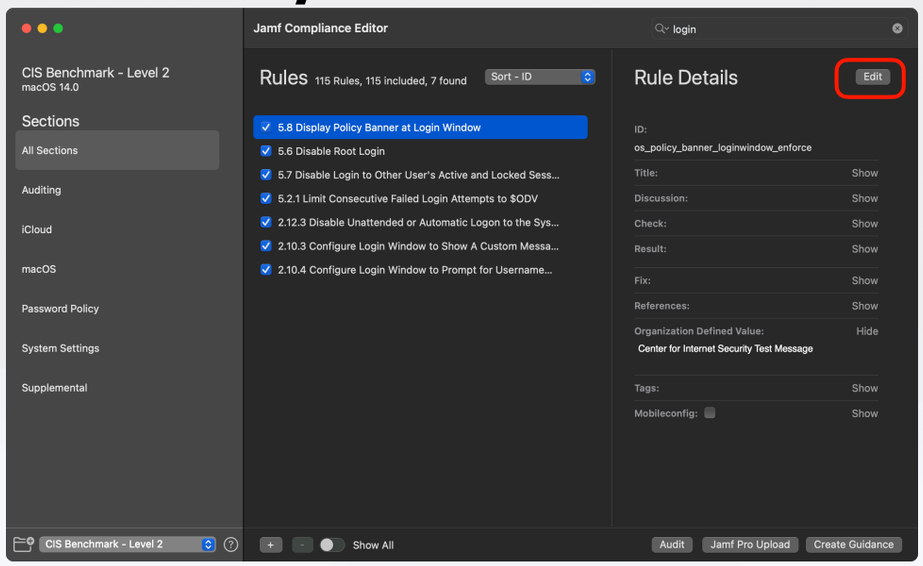
We want to update **CLICK** these two rules….”Display policy banner at login window” and “Configure login windows to show a custom message” We want to update these because they have default settings defined that no organization will meet

UPDATE RULES/ODV



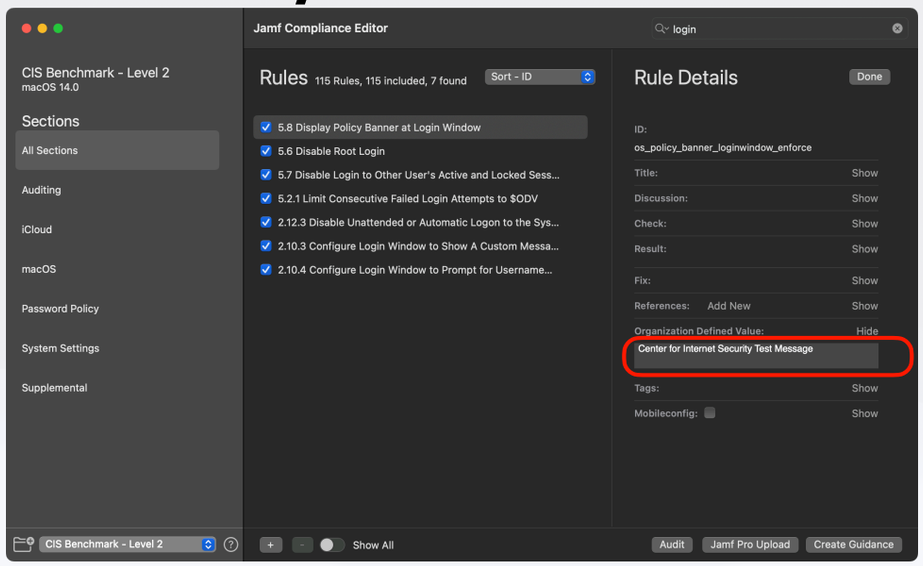
Lets click the first one and then click the “Show” button for the “Organizational Defined Value” and **CLICK** you can see it says “Center for Internet Security Test message”. Not really useful, huh? So now we should update this to our custom text.

UPDATE RULES/ODV



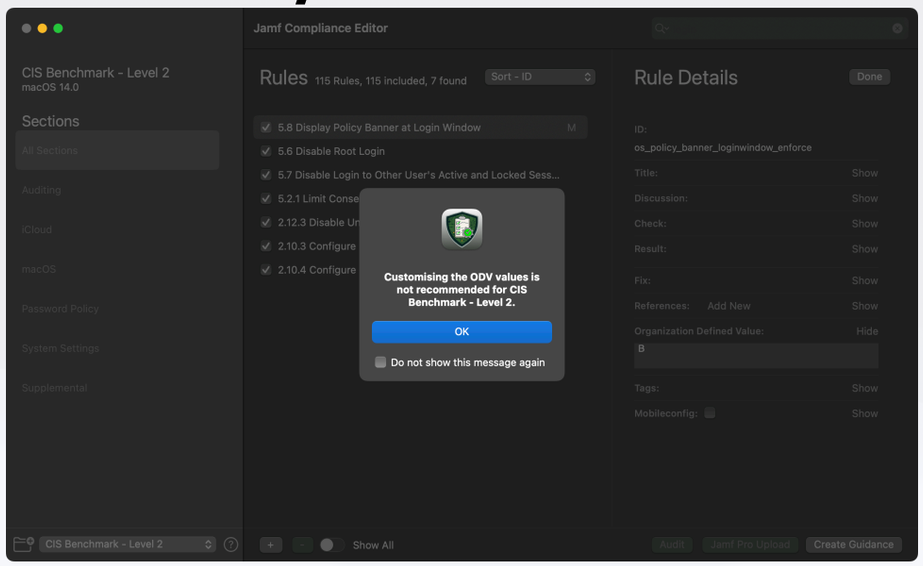
Click edit

UPDATE RULES/ODV



Click the text….and start making your edit..

UPDATE RULES/ODV

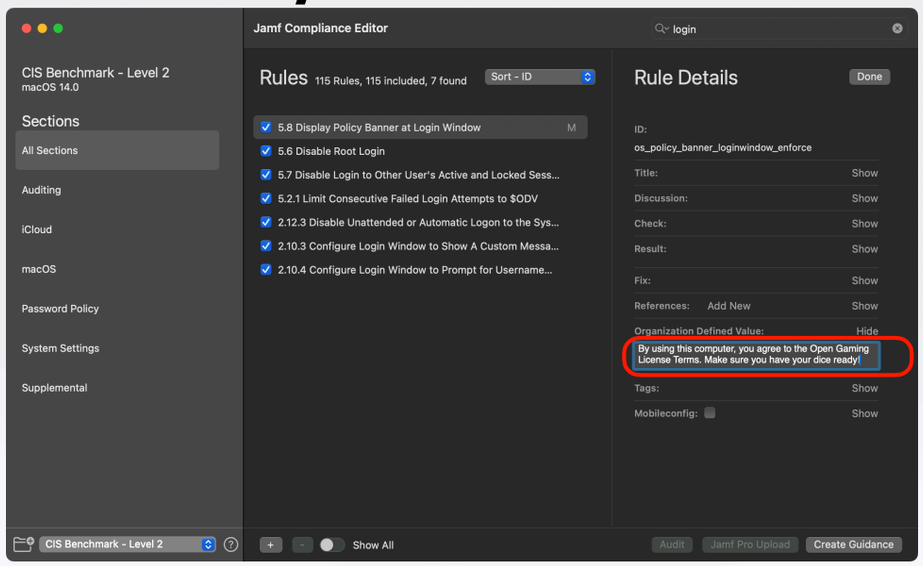


You'll get a prompt when you start editing…
This is telling you that you are editing an already established benchmark and by editing it you may not pass an audit. Remember, this is a benchmark, not a baseline.

Baselines provide a starting point for organizations in the security and privacy control selection process.
Benchmarks define which controls are to be included along with the organization defined values (ODVs) recommended by the benchmark's authors.

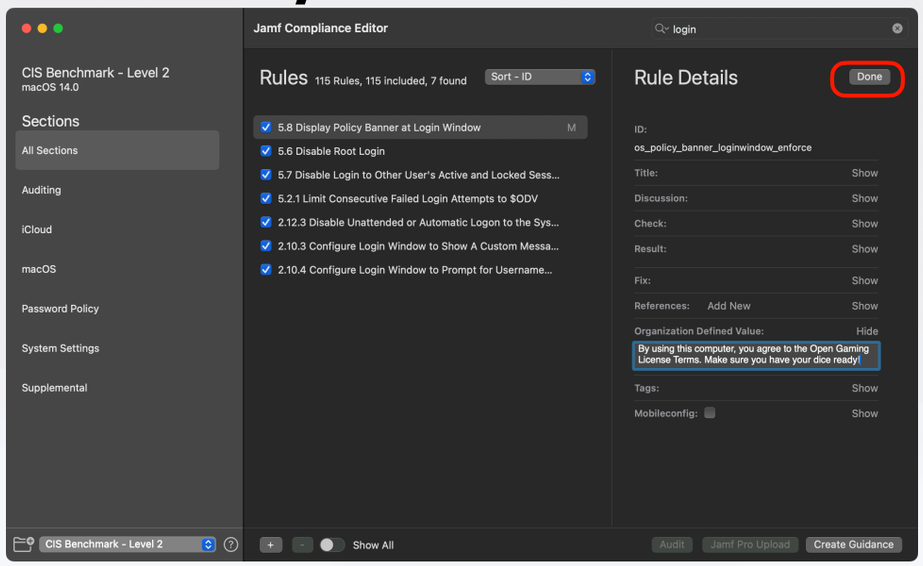
Click OK

UPDATE RULES/ODV



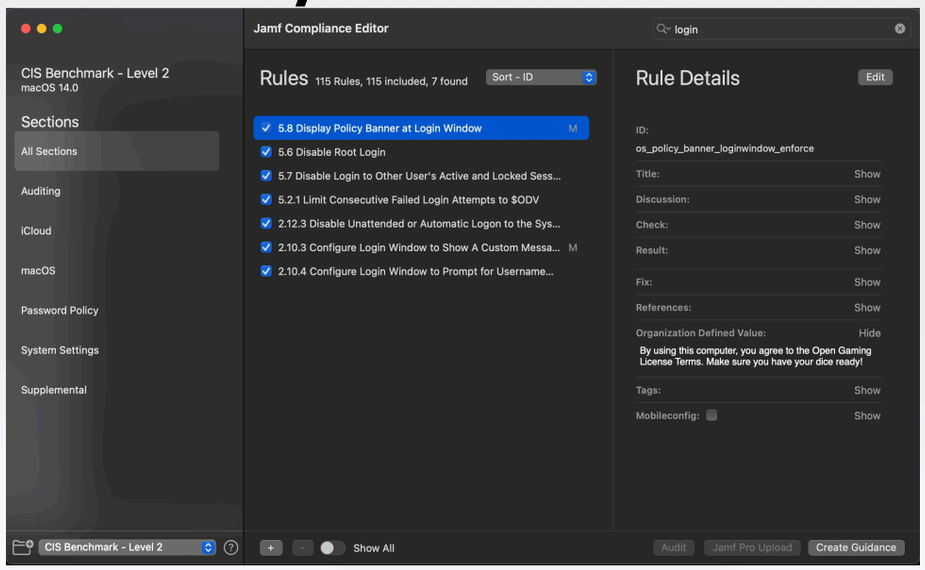
Lets add some custom text here…

UPDATE RULES/ODV

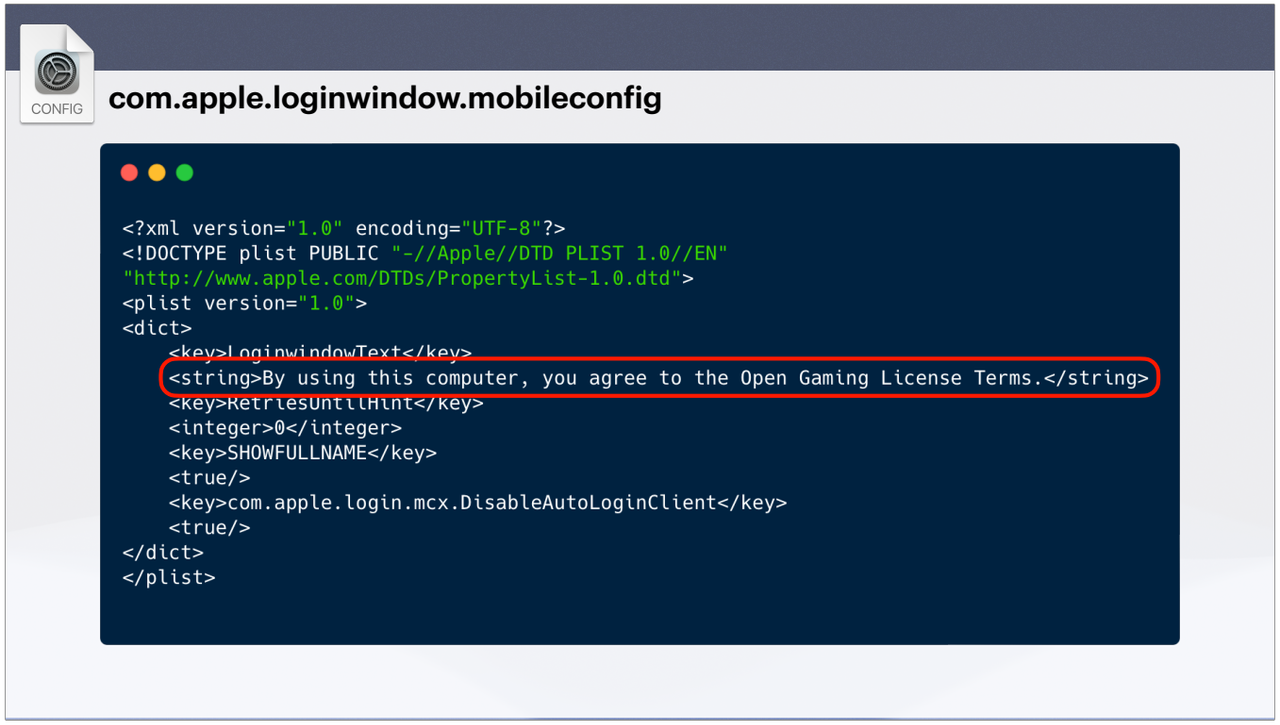


And click done..

UPDATE RULES/ODV



We'll do the same for the next rule
CLICK to start video
Click the rule, click edit, edit the text (we're changing it a little), then hit done.



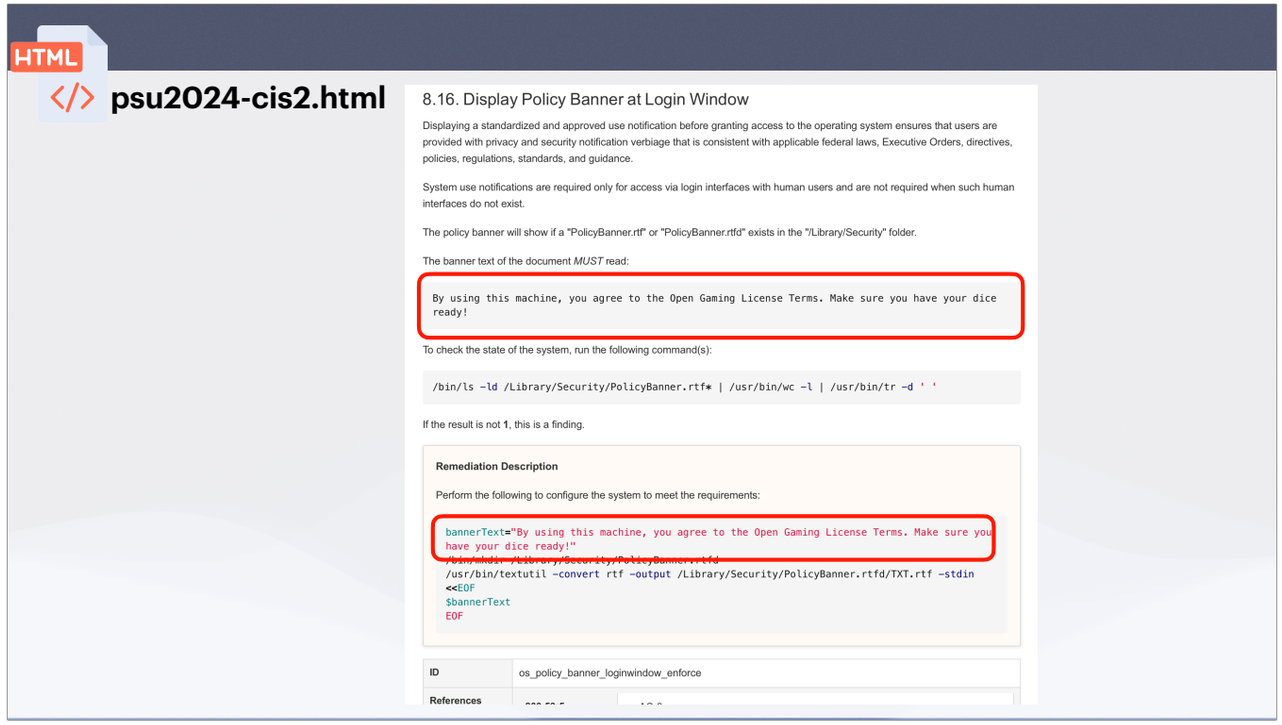
Now let’s see what that looks like in the mobile config for com.apple.loginwindow.

There’s our text..awesome. and now looking at our script for the banner…

psu2024-cis2_compliance.sh

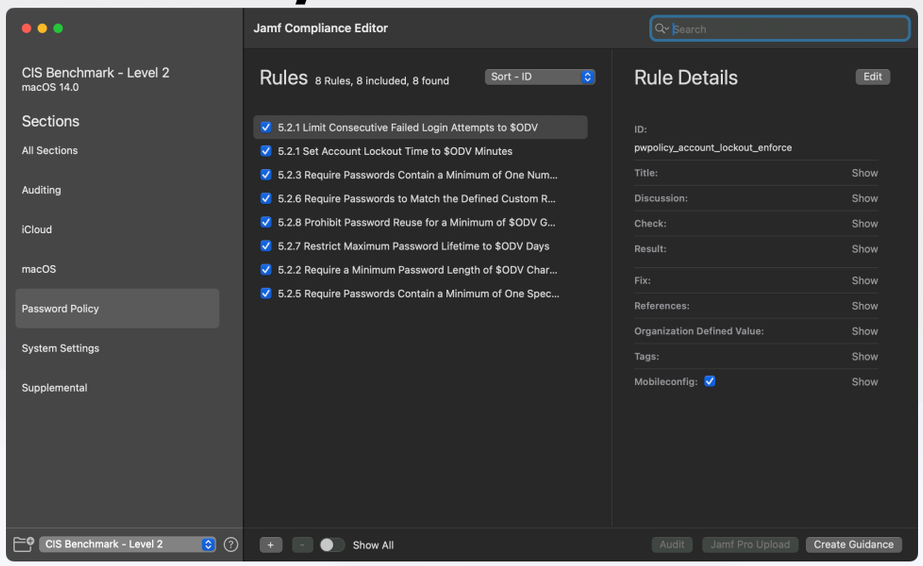
```
#####----- Rule: os_policy_banner_loginwindow_enforce -----#####
## Addresses the following NIST 800-53 controls:
# * AC-8

# check to see if rule is exempt
unset exempt
unset exempt_reason

exempt=$(/usr/bin/osascript -l JavaScript << EOS 2>/dev/null
ObjC.unwrap($.NSUserDefaults.alloc.initWithSuiteName('org.psu2024-
odvs.audit').objectForKey('os_policy_banner_loginwindow_enforce'))["exempt"]
EOS
)

exempt_reason=$(/usr/bin/osascript -l JavaScript << EOS 2>/dev/null
ObjC.unwrap($.NSUserDefaults.alloc.initWithSuiteName('org.psu2024-
odvs.audit').objectForKey('os_policy_banner_loginwindow_enforce'))["exempt_reason"]
EOS
)

os_policy_banner_loginwindow_enforce_audit_score=$($plb -c "print os_policy_banner_loginwindow_enforce:finding" $audit_plist)
if [[ ! $exempt == "1" ]] || [[ -z $exempt ]];then
        ask 'os_policy_banner_loginwindow_enforce - Run the command(s)-> bannerText="By using this computer, you agree to the
Open Gaming License Terms. Make sure you have your dice ready!"
/bin/mkdir /Library/Security/PolicyBanner.rtfd
/usr/bin/textutil -convert rtf -output /Library/Security/PolicyBanner.rtfd/TXT.rtf -stdin <<EOF
$bannerText
EOF '' N
        if [[ $? == 0 ]]; then
                logmessage "Running the command to configure the settings for: os_policy_banner_loginwindow_enforce ..."
                bannerText="By using this computer, you agree to the Open Gaming License Terms. Make sure you have your dice ready!"
/bin/mkdir /Library/Security/PolicyBanner.rtfd
/usr/bin/textutil -convert rtf -output /Library/Security/PolicyBanner.rtfd/TXT.rtf -stdin <<EOF
$bannerText
EOF
        fi
    else
        logmessage "Settings for: os_policy_banner_loginwindow_enforce already configured, continuing..."
    fi
elif [[ ! -z "$exempt_reason" ]];then
    logmessage "os_policy_banner_loginwindow_enforce has an exemption, remediation skipped (Reason: "$exempt_reason")"
fi
```

This is the check for that rule…sorry for the wall of text…but **CLICK** here are the updated ODVs…



And we already saw the PDF that gets generated..here’s the HTML doc and you can see **CLICK** the text was updated here as well!

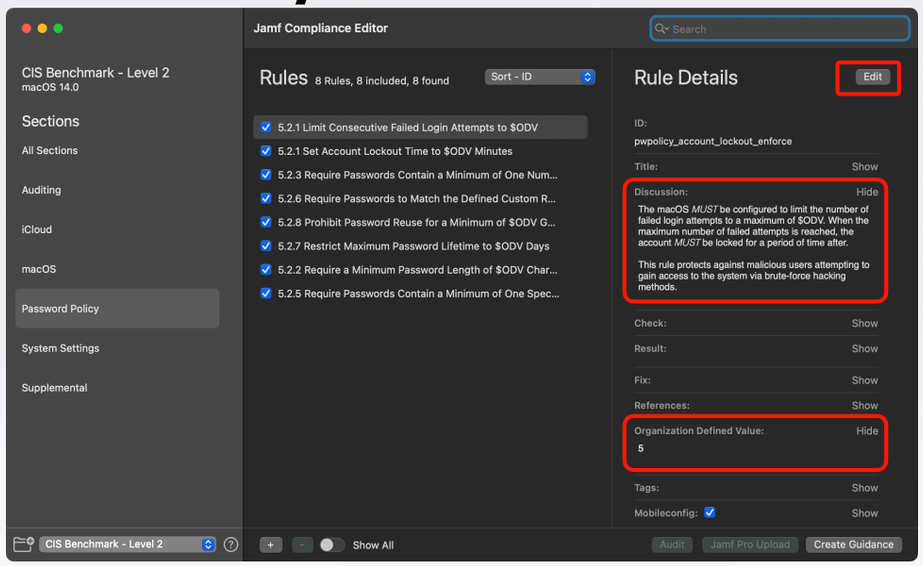
UPDATE RULES/ODV



So what about other rule changes? Well, that's just as simple (though…the checks can get rather complicated).

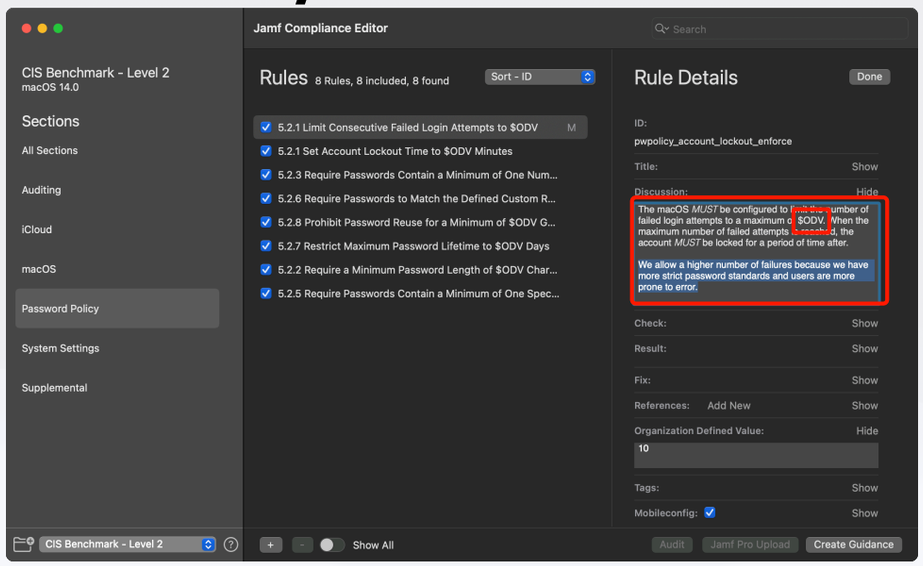
Lets look at password policy and change some items in the "Limit Consecutive failed login attempts"

UPDATE RULES/ODV



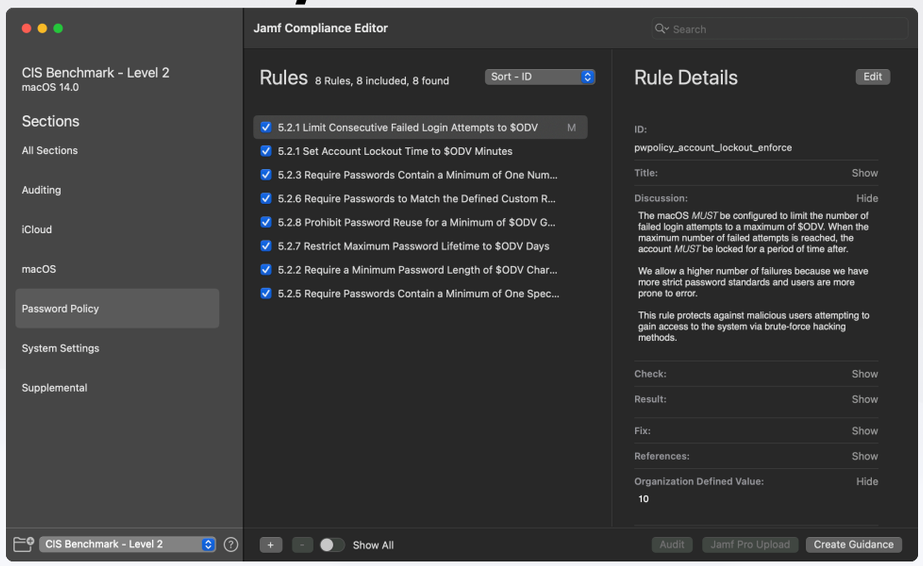
We'll click "show" for the discussion and the ODVs and you can see the info here…let's click edit.

UPDATE RULES/ODV



And now we'll make some changes…first update our ODV to 10..then update our "Discussion" to tell us why.
CLICK
I'm adding "We allow a higher number of failures because we have more strict password standards and users are more prone to error." Note I didn't have to change the value in the discussion, **CLICK** thats because there's an ODV variable that grabs the value we set. Pretty nice.

UPDATE RULES/ODV



Click Done and that’s it! Our updated discussion is now active. When we create guidance again, it’ll show this discussion in the supplemental docs.

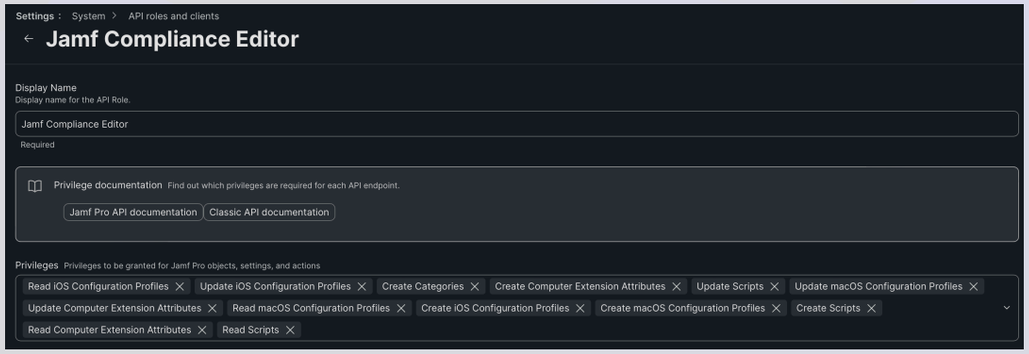
And that’s tailoring..or rolling your own character. Now..

PLAY SESSION 1
(PUT INTO JAMF)

Let’s play!

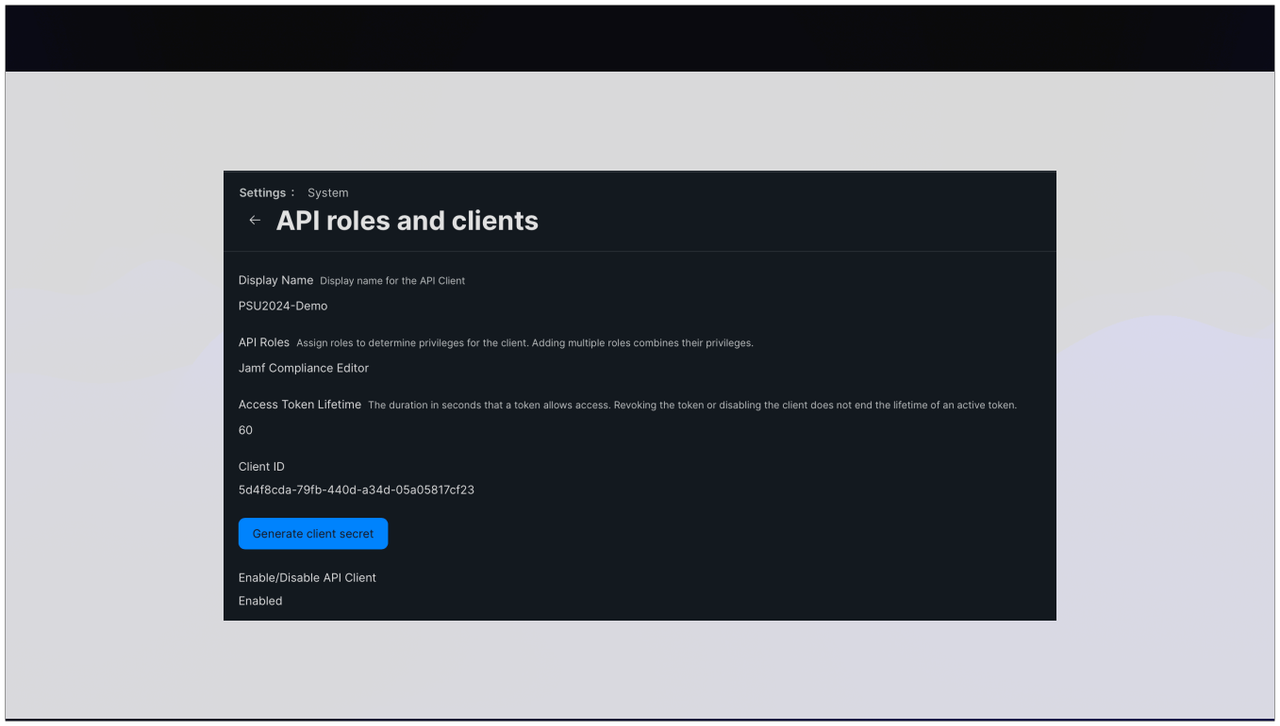
Since we’re focusing on Jamf, I’m going to show you how to configure the API to accept uploads from JCE. If you’re using another MDM, this would help you, sorry.

Jamf Pro Server Object	Create	Read	Update	Delete
Categories	x			
Computer Extension Attributes	x	x	x	
macOS Configuration Profiles	x	x	x	
iOS Configuration Profiles	x	x	x	
Scripts	x	x	x	

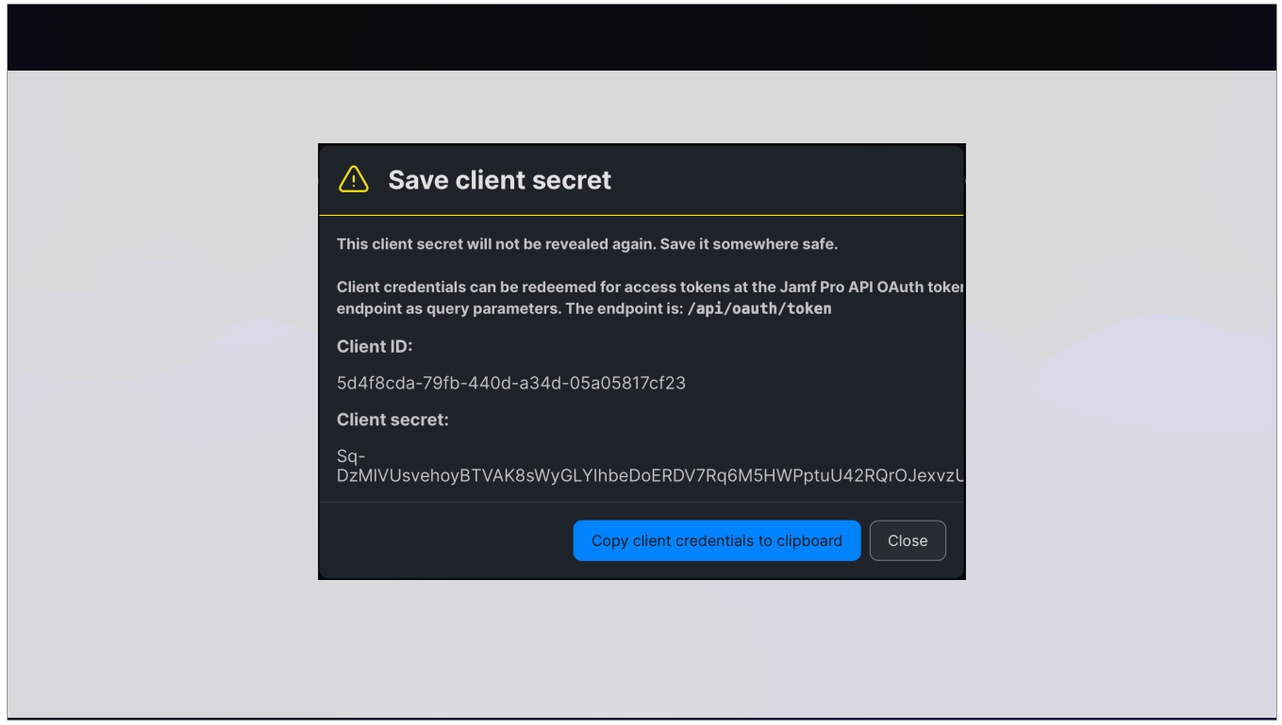


The top image is from the JCE documentation, it shows the required permissions to use the built-in jamf uploading.

The bottom image shows the API Role we are creating. If you’re not using API roles and Clients, you should start..

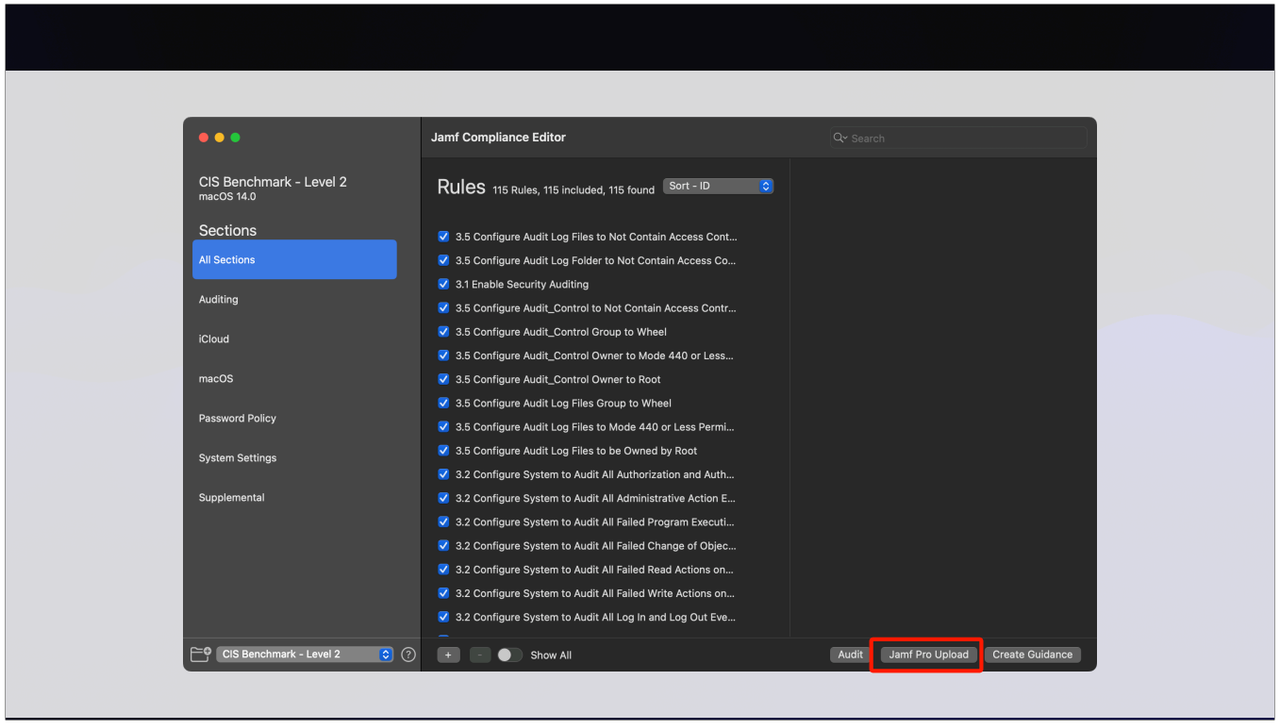


Now, we’ll create our API client..in this case I’m making a new PSU2024 demo client. I then click generate client secret…



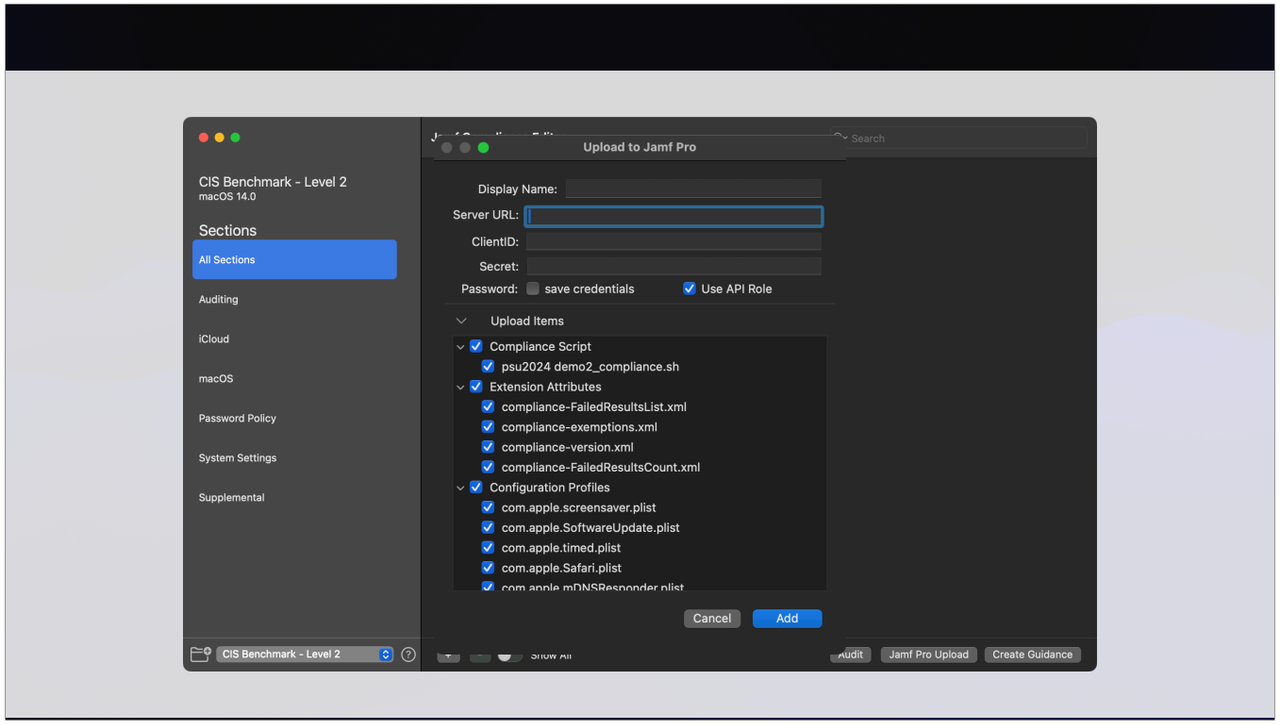
And I’m shown the ID and secret. Hit the copy button there and store this somewhere because if you need to see it again, it’ll change the secret.

You need to add these to Jamf compliance editor when you’re uploading.



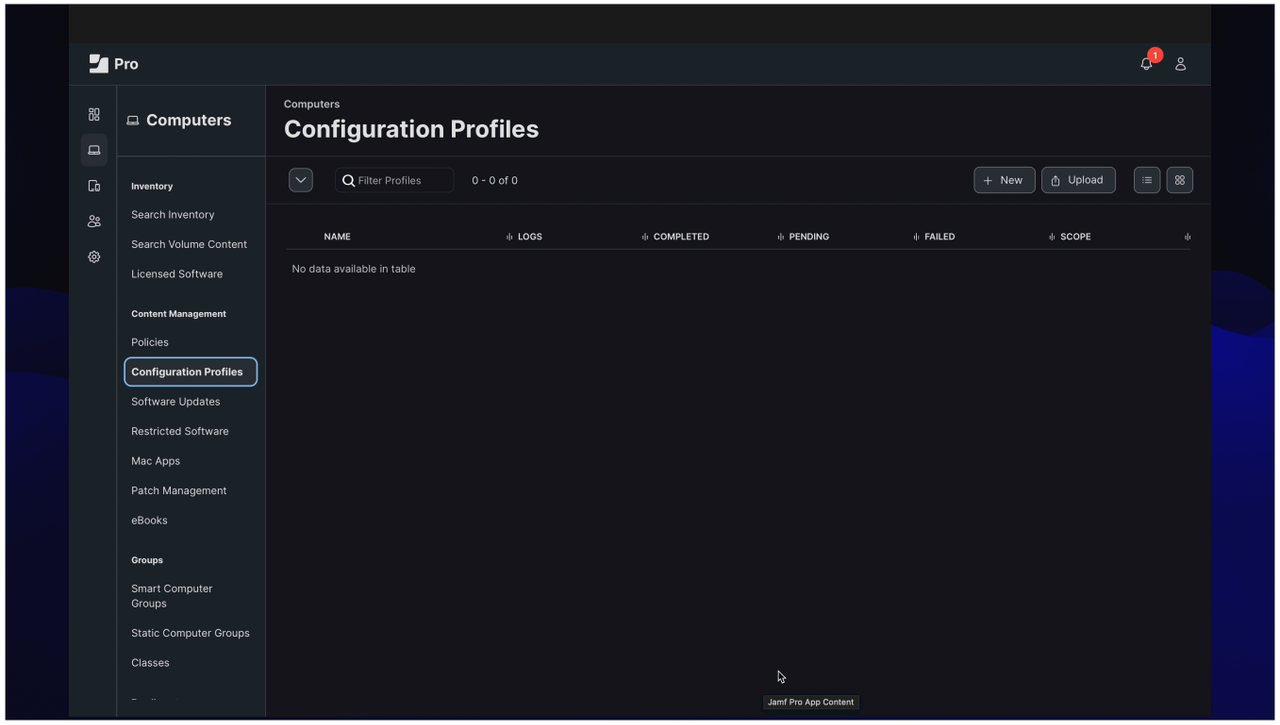
To enter this info into JCE, you will have to generate your guidance, save, then **CLICK** click the Jamf pro upload button…which leads us to…


UPLOAD ALL THE THINGS!



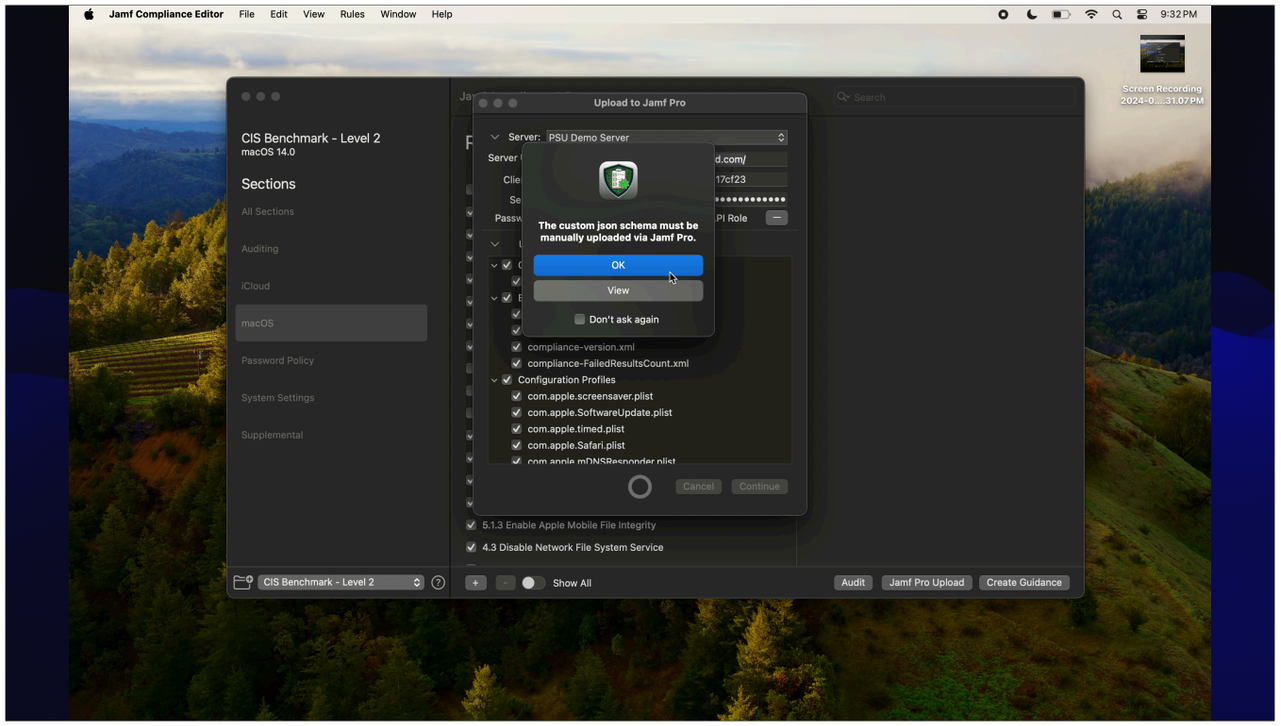
When you click upload…the upload to jamf pro dialog opens. Enter the info requested which is server URL, clientID and secret (those items we copied from the previous step). Suggest saving them and giving a display name.

You can use basic authentication by unchecking the “Use API role” checkbox..but I don’t recommend that.

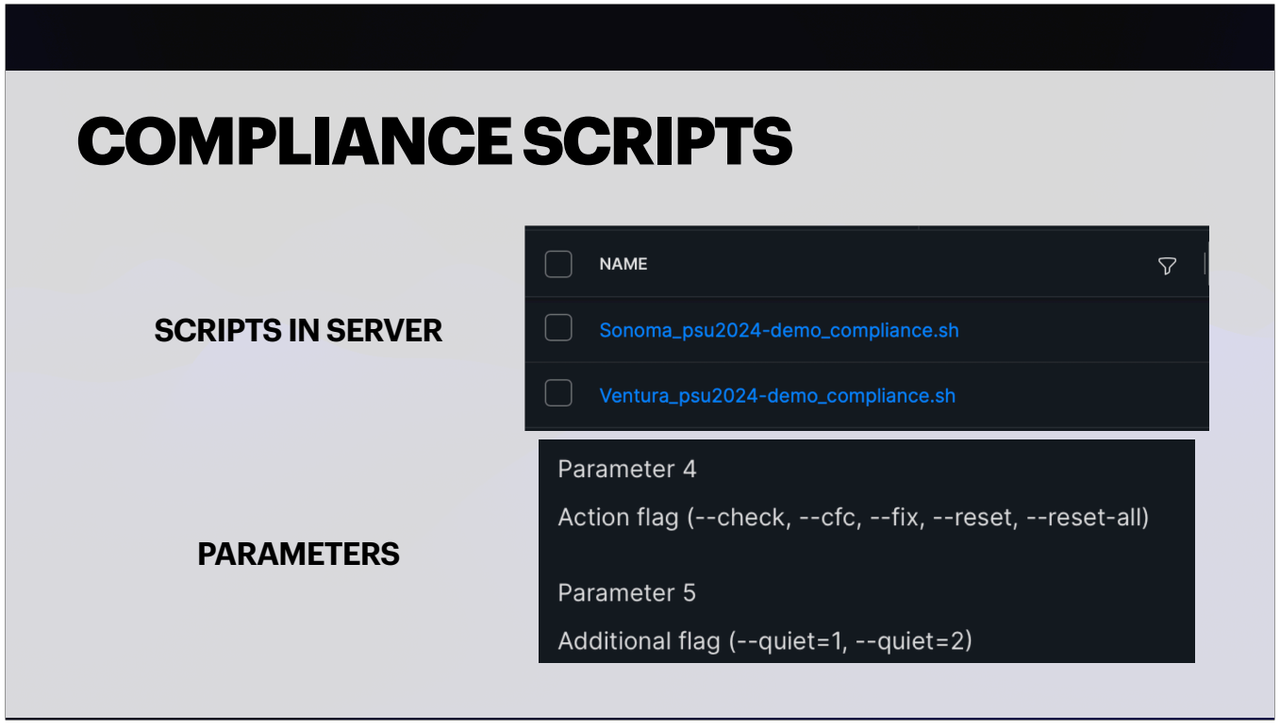


Now lets see that that looks like. **CLICK**

Here’s our jamf server with no config profiles. We’ll switch to JCE and create guidance….save…close the dialog then click “Upload to jamf pro”. Our info is already entered and saved..we’re going to keep everything checked and click continue. It’ll take a few seconds to upload…and when it’s done..you’ll get this prompt about manually adding the JSON schema. We’ll cover that later…



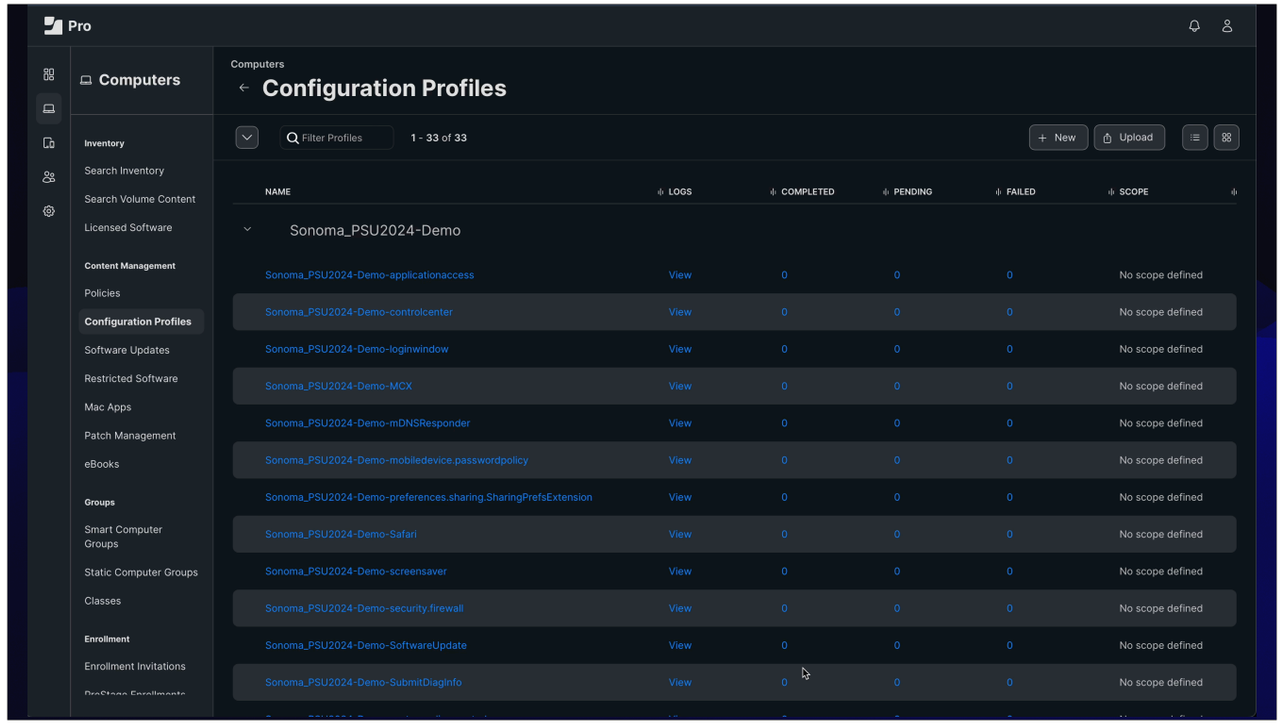
Click OK **CLICK**
And we'll go back to jamf and now we will see our config profiles in the server…note none are scoped. We'll go to our settings and check scripts and we'll see our compliance script is in here..



Just a summary of the scripts. I have one per OS (I didn’t show uploading both) and both have these parameter notes.

Parameter 4 is the most important one to have in place.

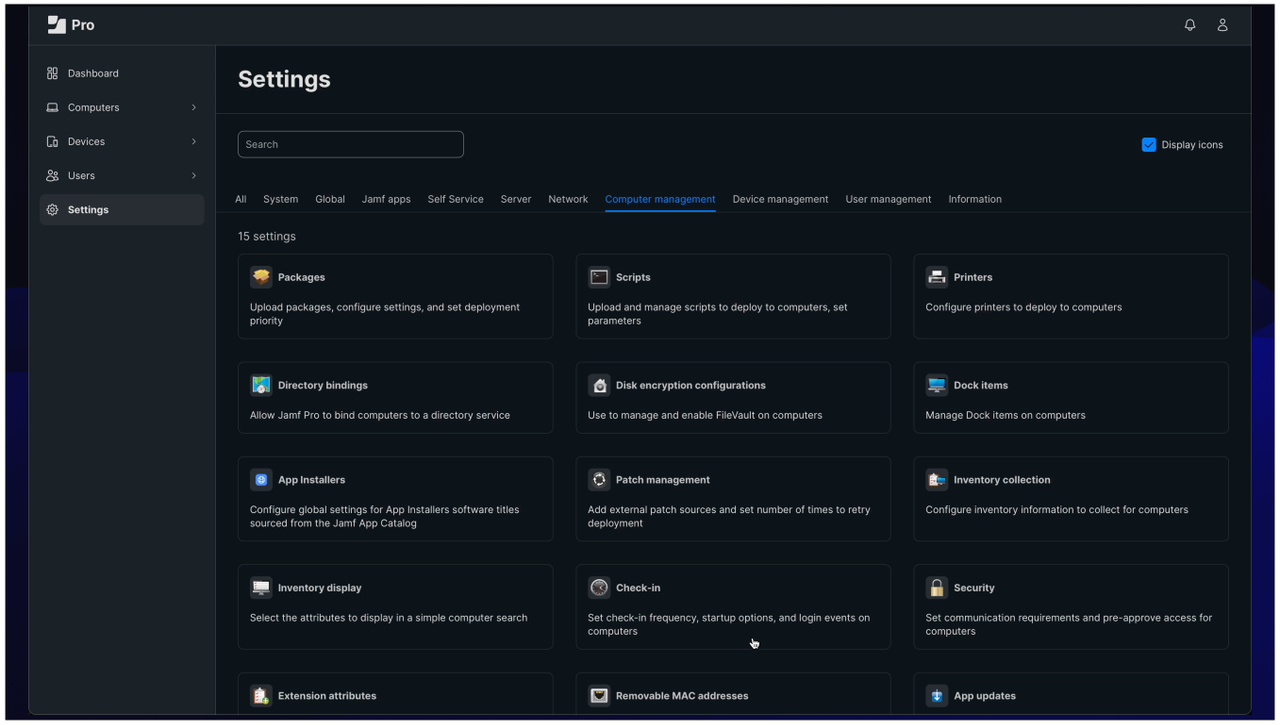
Check will run an audit
CFC is check/fix/check; it’s essentially the “all in one” that you run to remediate
Fix is just that, it fixes the items found to be wrong
Reset will remove the audit file for that script
Reset all will remove ALL audit files (useful when an upgrade happens)



Lets take a look at some of our uploaded items. **CLICK** here's the login window with our text…note that it's added as a custom application and a PLIST.

We'll look at firewall next…you'll note it's using the Jamf UI settings. Similarly, submit diaginfo also uses the UI settings.

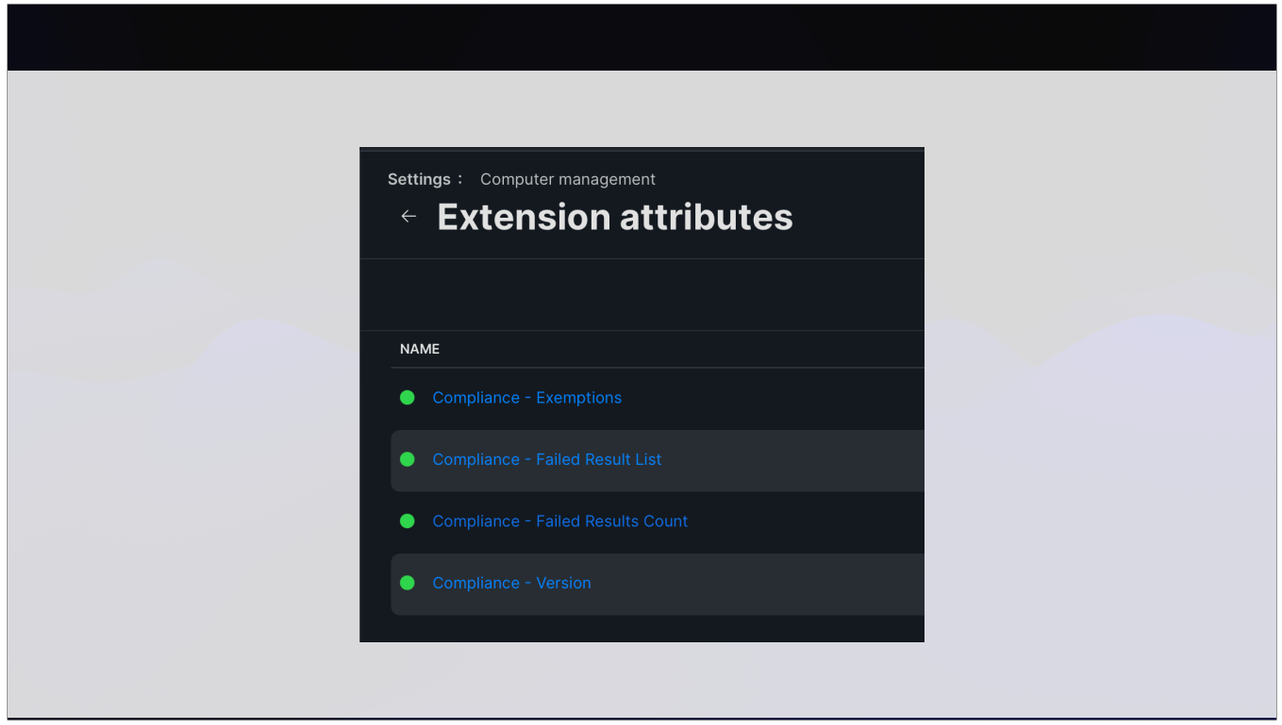
This is because some of the Jamf profiles payloads are already updated to read the profiles correctly..many are not. So you will see that a lot of uploaded items will have a custom PLIST uploaded. You can go in and manually change them to use the Jamf UI settings if you'd like…you can also combine items that use the same Jamf UI elements, but this is up to you.



We also uploaded some Extension attributes. **CLICK**

Four to be exact…

One for showing exemptions, one for a count of found issues, one for a list of the issues, and one for showing the name of the audit version/name it’s reporting.

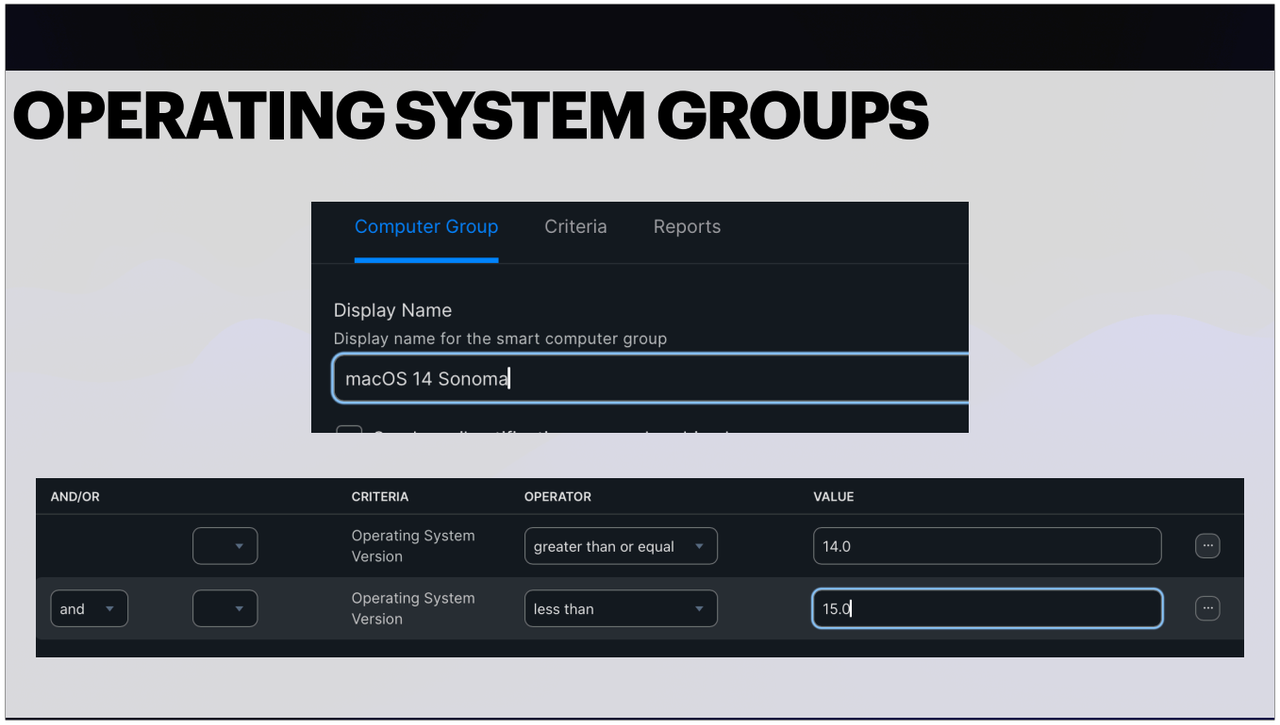


And in case you need to see those again, here are those EAs…

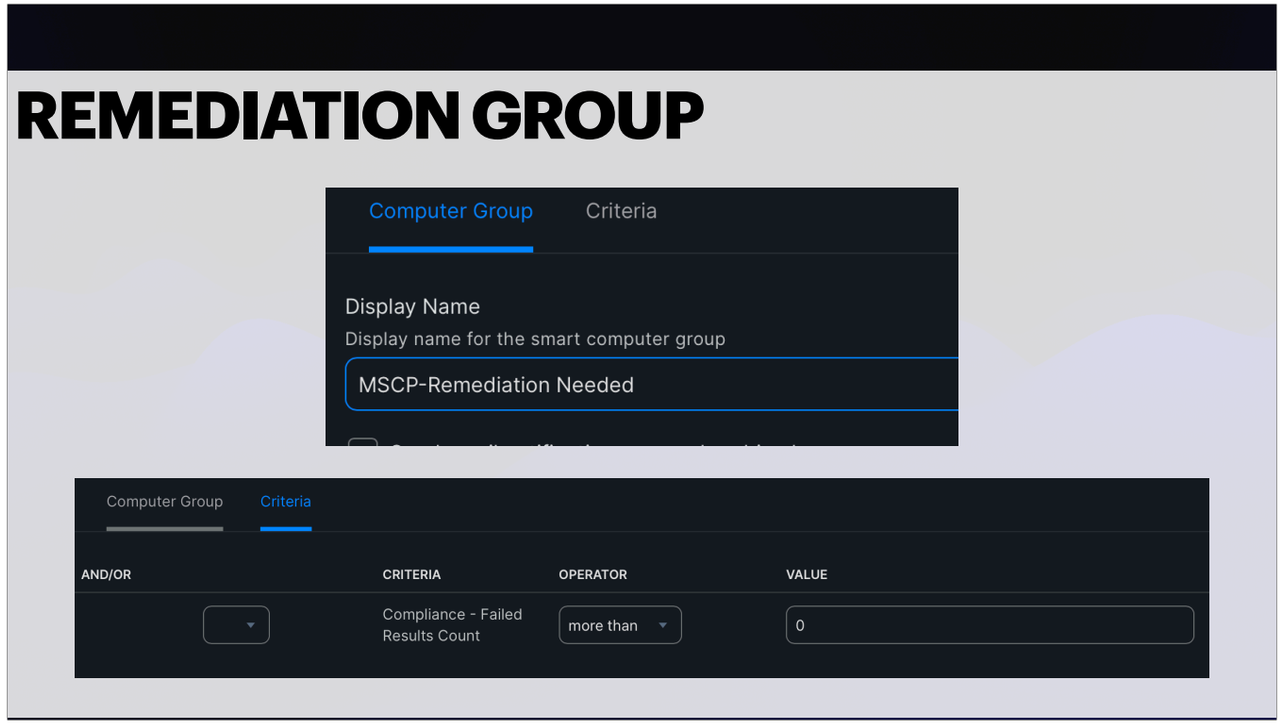
Now that everything is prepped, it’s time to..

GROUPS AND POLICIES

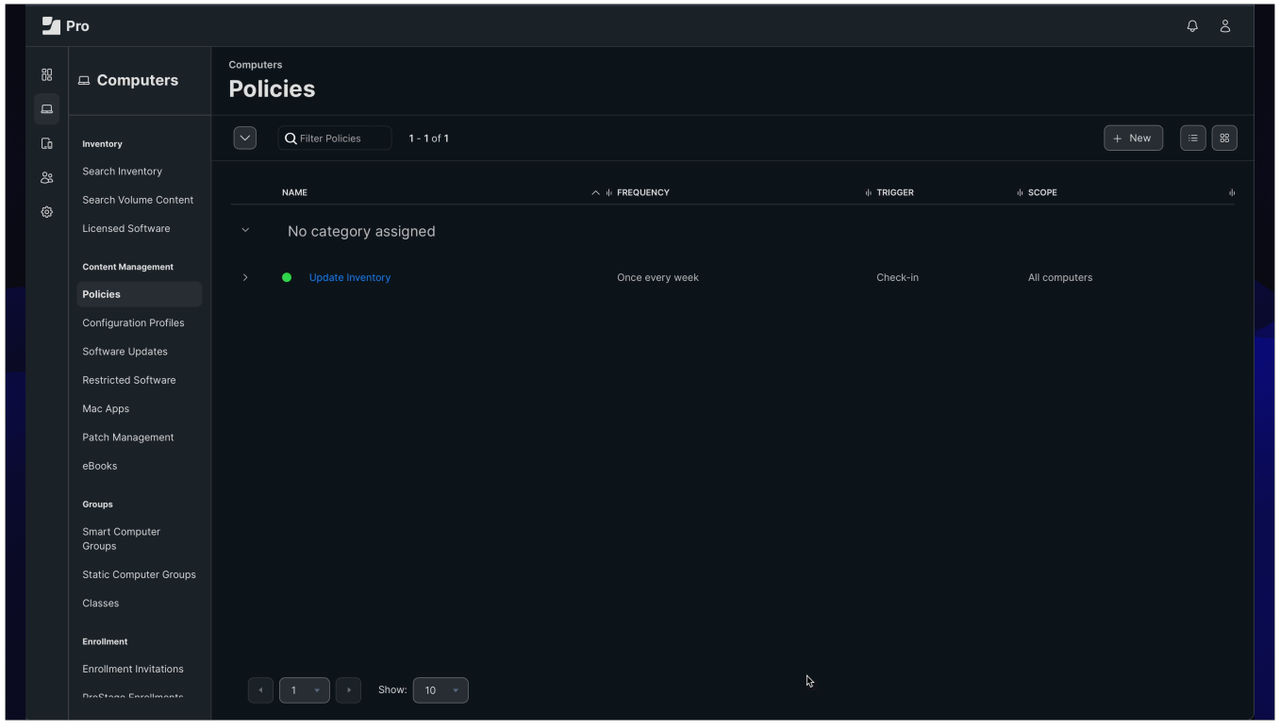
Add our groups and polices.



First, we'll create our Operating system groups. We would do this for all of our supported OSes. In this example we're setting up for Sonoma. Of course, if you're testing Sequoia you want to make sure to avoid running on those, so I created this group for include greater than or equal to 14.0 and less than 15.0.



Next, we'll set up our main remediation group. This uses one of the extension attributes we added, in this case the "failed result count" and we'll check if that is more than 0. Essentially…did it find any failure.



Now that the groups are created, we'll create our first policy. We're going to make an audit policy, the is run to audit the systems then report the audit findings to Jamf using the EAs.

CLICK

Create a policy, let name it "MSCP - Audit" and clarify this is for Sonoma. We'll use recurring check-in, once/day. You may want to set client side limitations about which days to run this but we'll leave is every day.

We'll add a custom trigger, mscp-audit. Then add our compliance script that was created for Sonoma (note we have another in there for Ventura). We'll keep the priority as after then add our flag "—check", add a recon, and our scope is Sonoma since this is our Sonoma audit.

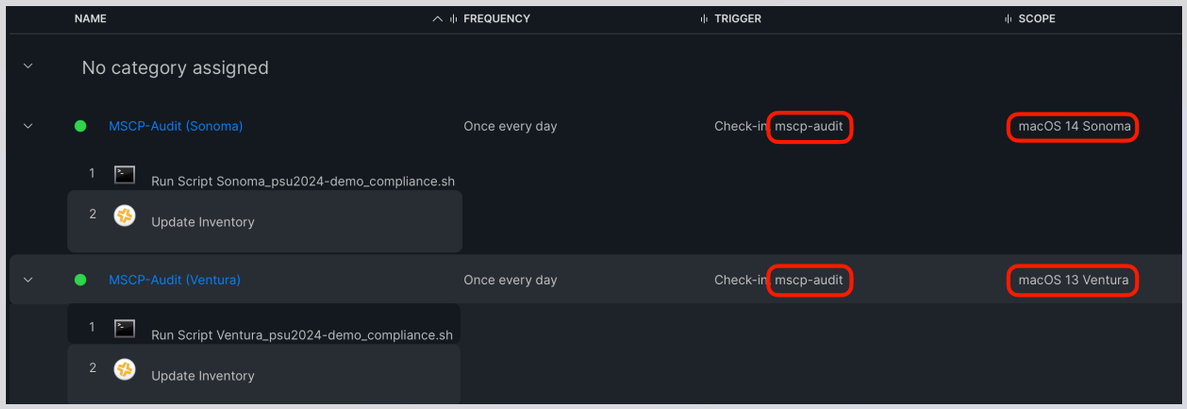
AUDIT POLICIES (PER OS)

SUMMARY FOR SONOMA

- **Title:** MSCP-Audit (Sonoma)
- **Triggers:** Recurring, Custom: *mscp-audit*
- **Frequency:** Once every day (can limit by day if desired)
- **Script:** Sonoma_psu2024-demo_compliance.sh
  - **Parameter 1:** --*check*
- **Maintenance:** *Update Inventory*
- **Scope:** Smart Group: *macOS 14 Sonoma*

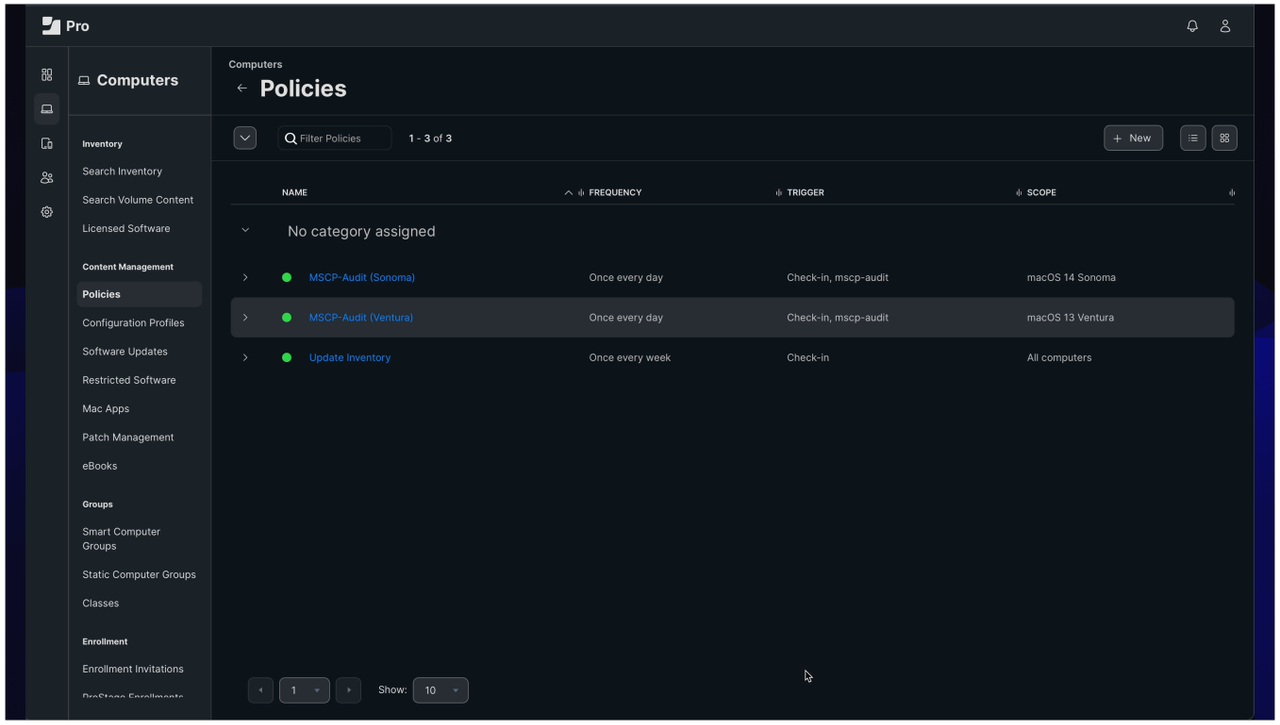
Here’s the summary of that policy. Just repeat this for each operating system.

AUDIT POLICIES



Here are both policies. You can see they both **CLICK** use the same trigger…but they'll only run **CLICK** on the scoped OS. One trigger to take care of every OS, nice!

Okay, so now that we have the auditing, we need a policy for checking our systems and fixing the issues.



So let’s go back **CLICK** Let’s make a new policy, we’ll call it MSCP-Chec and Fix and we’ll note this is for Sonoma.

We’ll set a custom trigger of “mscp-fix” with an ongoing frequency

We’ll add our same Sonoma script but the flag we’ll add is “CFC” and finally we’ll scope it to Sonoma only.

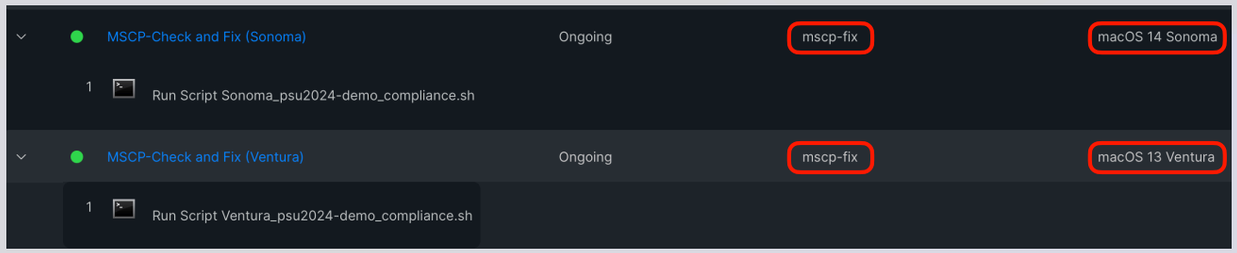
CHECK/FIX POLICIES (PER OS)

SUMMARY FOR SONOMA

- **Title:** MSCP-Check and Fix (Sonoma)
- **Triggers:** Custom: *mscp-fix*
- **Frequency:** Ongoing
- **Script:** Sonoma_psu2024-demo_compliance.sh
  - **Parameter 1:** --*cfc*
- **Scope:** Smart Group: *macOS 14 Sonoma*

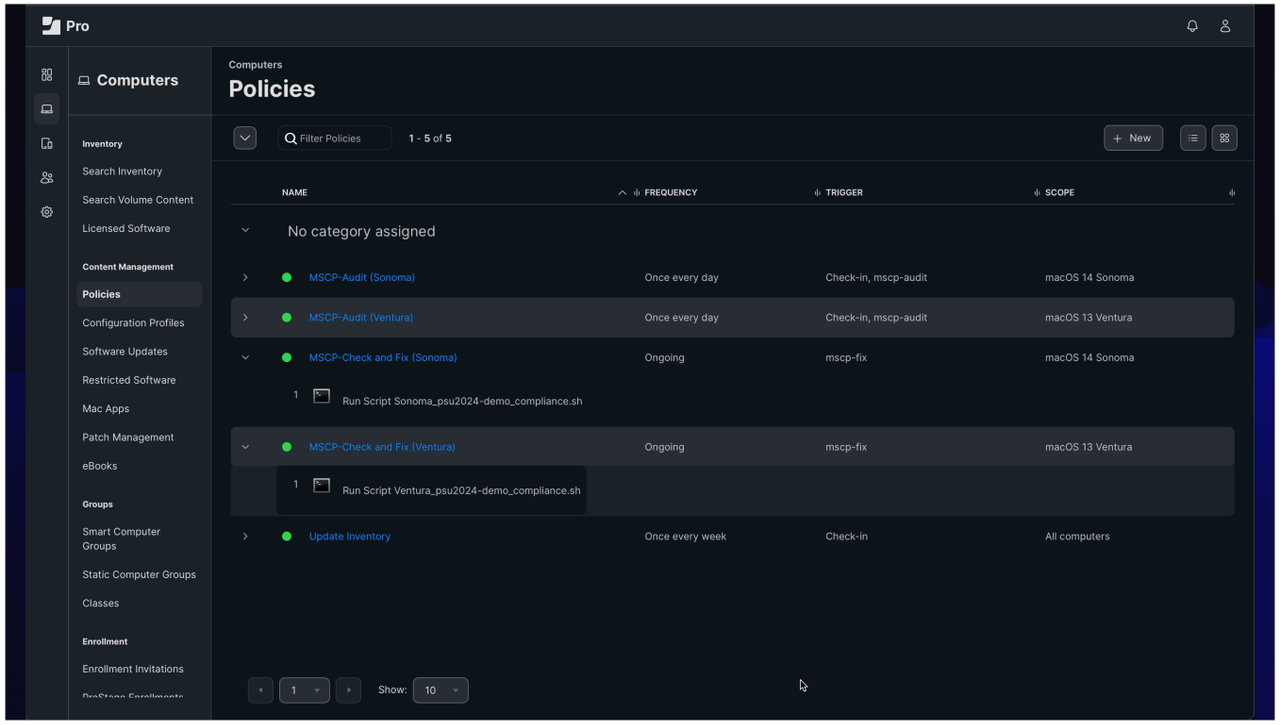
Here’s the summary of that policy. Just repeat this for each operating system.

REMEDIATION POLICIES (FOR EACH OS)



And again, Here are both policies. You can see they both **CLICK** use the same trigger again…but they'll only run **CLICK** on the scoped OS.

Now, if you have a high passive perception, you probably noticed that these are only run via a custom trigger…how do you get your machines to actually run these? Well, we'll make a remediation controller policy.



CLICK Make one more policy, we'll name this MSCP-Remediation (Daily), use a recurring check-in, and execute once/day (edit as needed).

Now instead of adding our script, we'll execute a command under files and processes. We're going to tell this policy to run the trigger "mscp-fix", and we'll update inventory with this one.

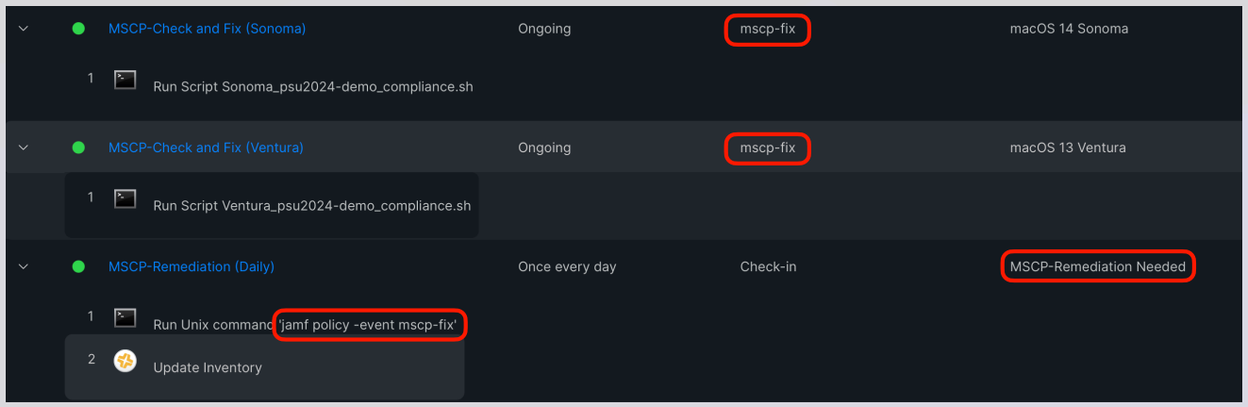
Finally, we'll scope this to our Remediation needed smart group.

REMEDIATION CONTROL POLICY (ONE)

- **Title:** MSCP-Remediation (Daily)
- **Triggers:** Recurring
- **Frequency:** Once every day (can limit by day if desired)
- **Files and Processes:**
  - **Execute Command:** *jamf policy -event mscp-fix*
- **Maintenance:** *Update Inventory*
- **Scope:** Smart Group: *MSCP-Remediation Needed*

Here’s the summary of that policy.

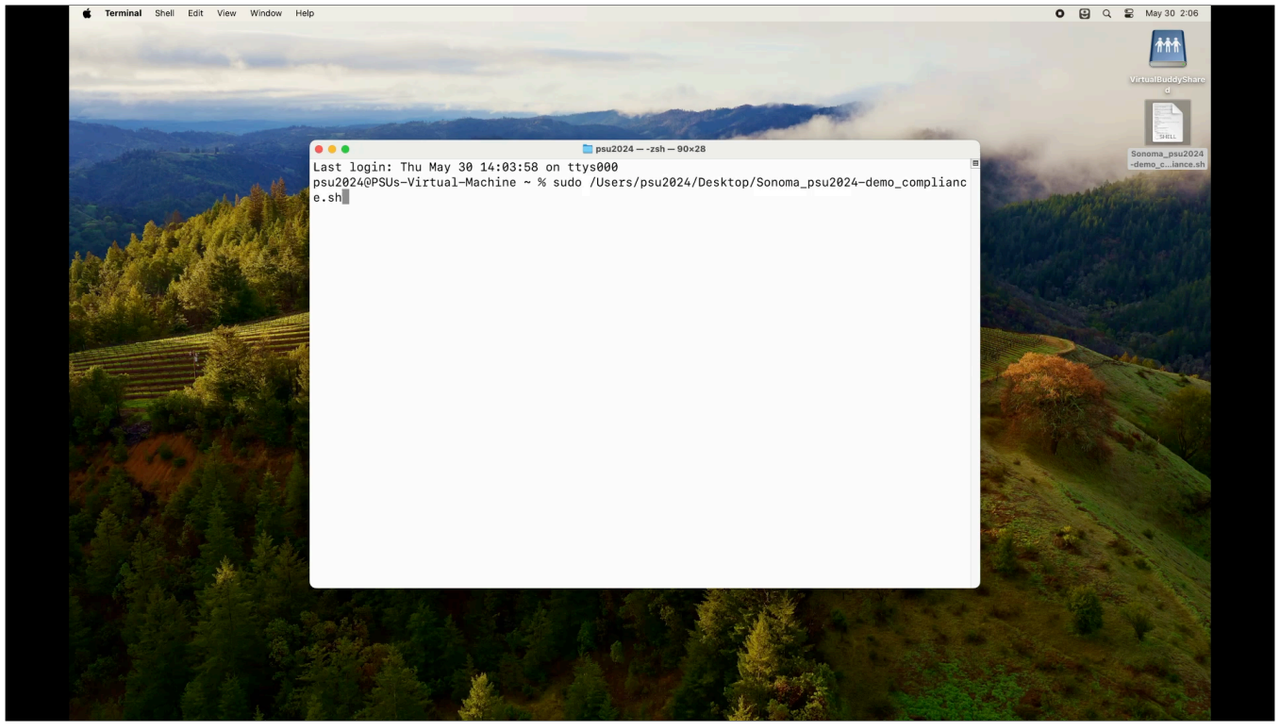
REMEDIATION CONTROL



Finally, these are all three remediation policies. **CLICK** Here's our smart group in the control policy..which will **CLICK** run the jamf trigger . That jamf triggers runs one of our **other policies based on OS.

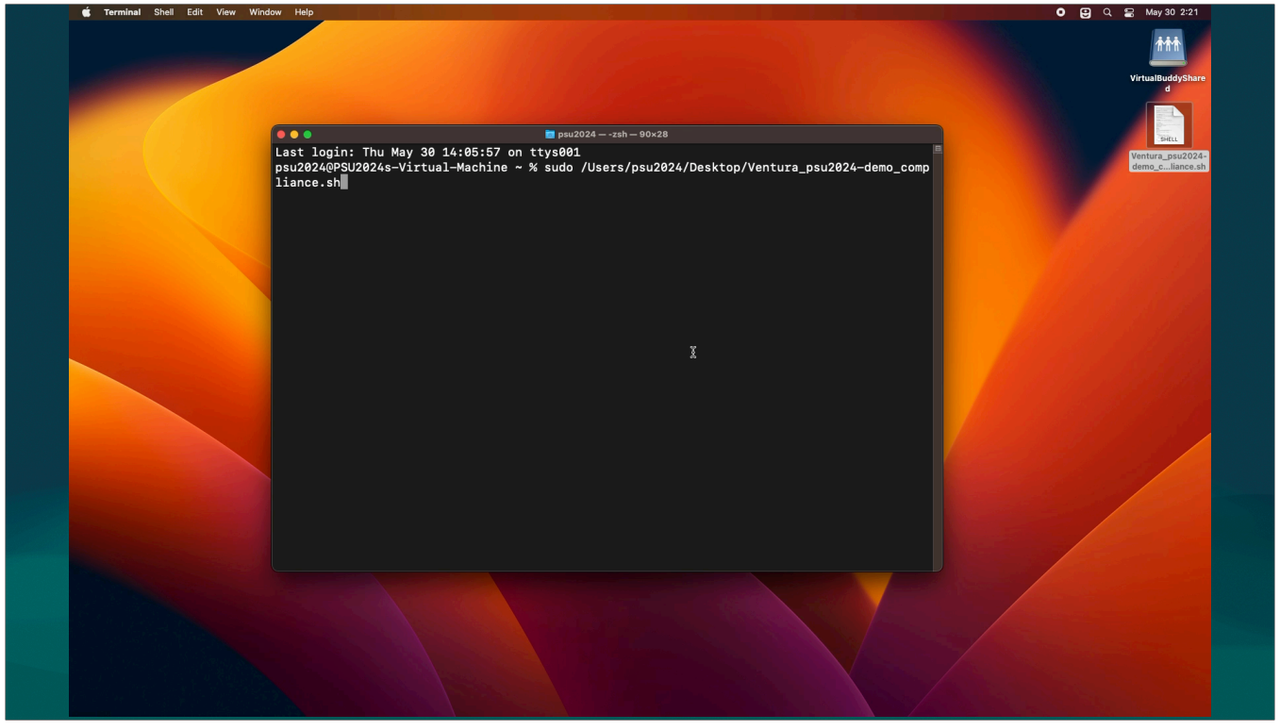
Now, everything is setup…

Let’s roll initiative!

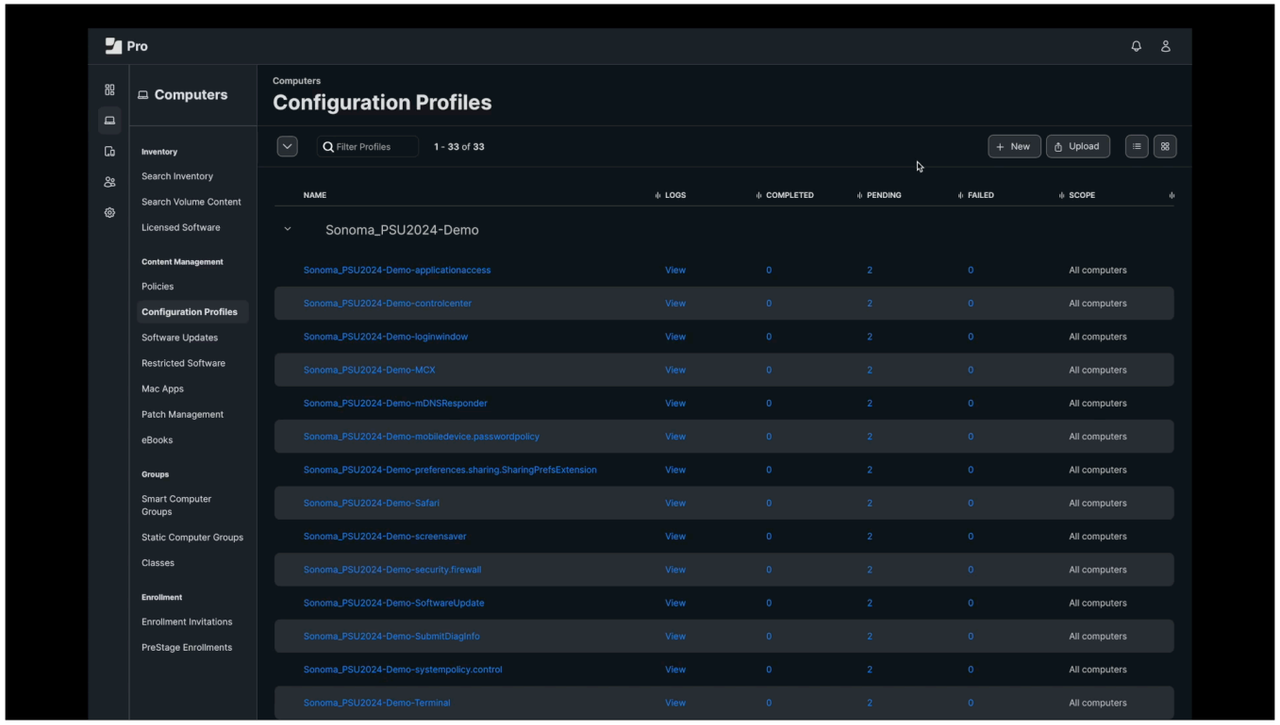


First, let’s just run the script we created locally to see what it does.
You can see it shows a menu and options that we can utilize.

This machine is already enrolled but has no profiles from our project scoped and never ran the script to fix anything…so you can see that from the output here. About 35% compliant. That’s just base install.



You can see the same thing here on Ventura. Different numbers, but nothing “fixed”



Now here, you can see that we have assigned our profiles to our systems, note that not every profile is assigned to all computers.

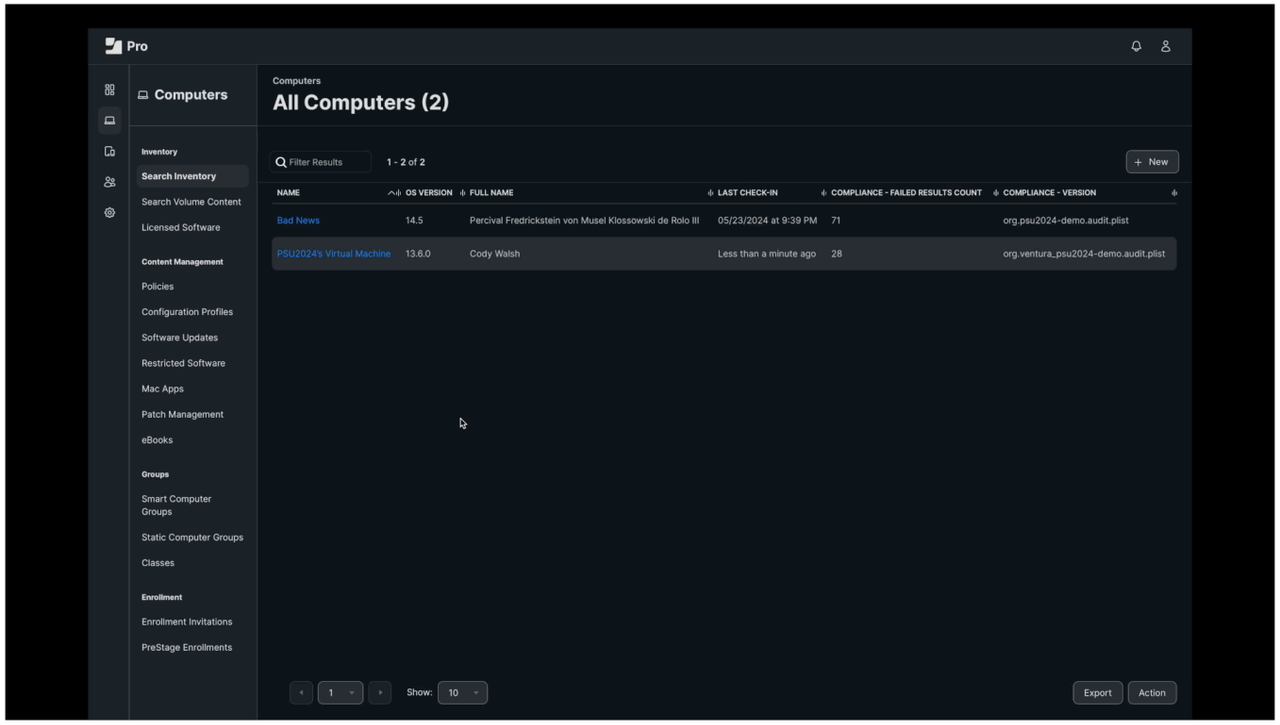
Many of the checks and fixes are the same on Ventura and Sonoma but there are a handful that are Sonoma only. I've adjusted most of them for this demo but I won't cover those here.

Also keep in mind that you **can** combine profiles if you'd like, you can also use the built in Jamf Profiles UI instead of uploading everything. Most of these are custom profiles.

CLICK

Anyway, now that we have them scoped, you can see they have delivered to our Ventura machine. We'll now run a jamf policy to run our audit policy….and refresh and you can now see that the number shows as much lower. Previously it was at 70…

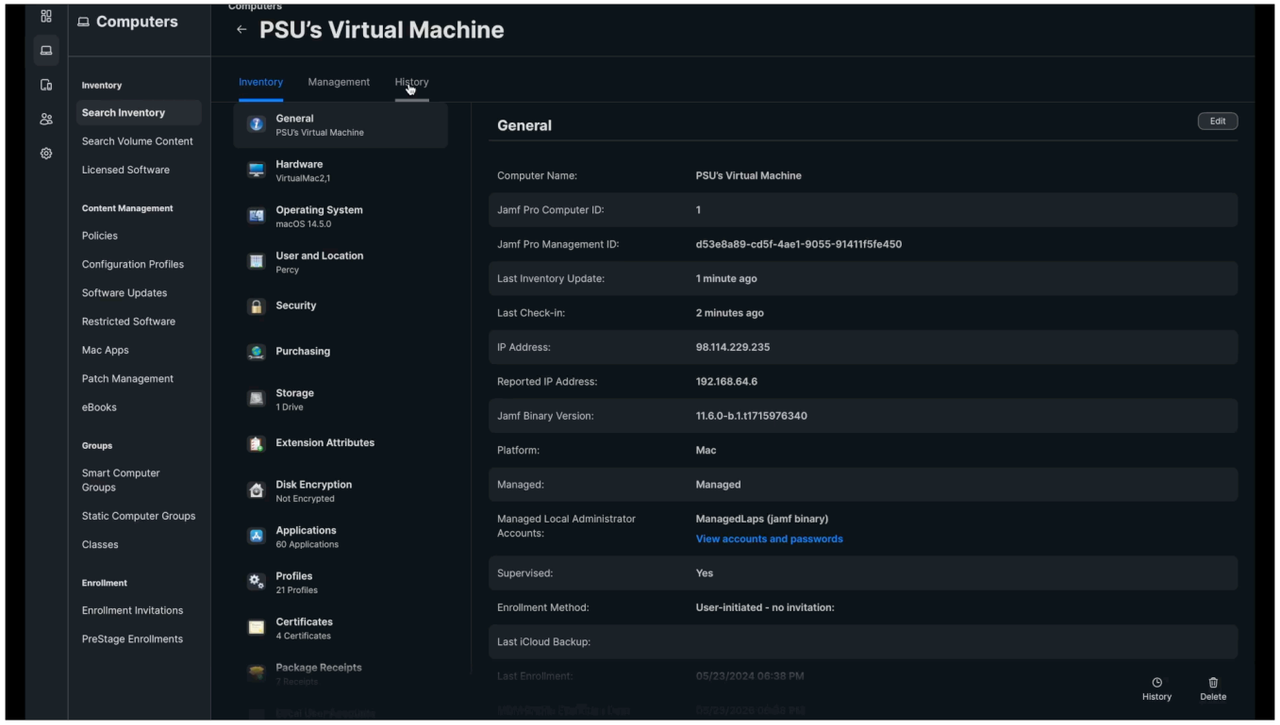
That's just with profiles!



Now I'll do a little reverse look. **CLICK**

You can see our Sonoma machine shows 71 (the number we saw previously), we'll now check for profiles…installed. Now run a policy to get the audit.

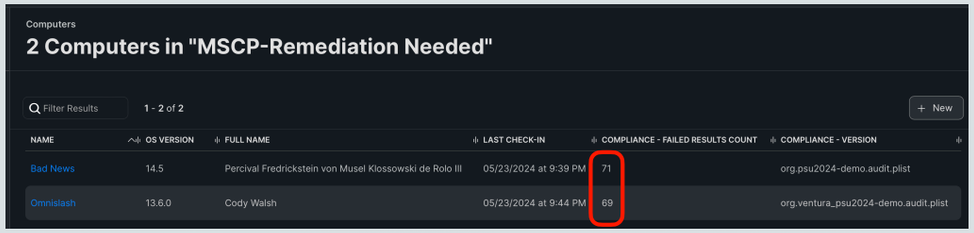
Let's run the script manually again…just to see something different and now you see the number is much lower again. Just from profiles.



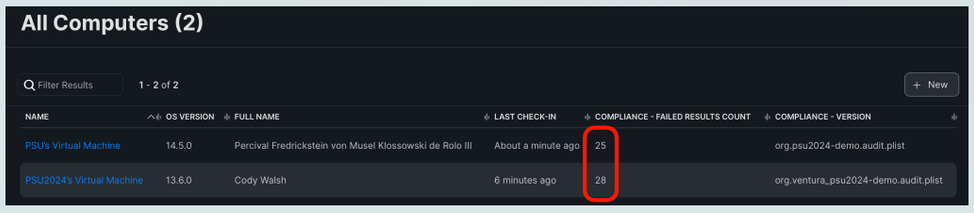
Now lets look at that Sonoma Mac in Jamf
CLICK

You can see the items that are left to fix here…and an quick look at the logs from our audit. These are useful for troubleshooting and finding where the fixes are not working.

BEFORE



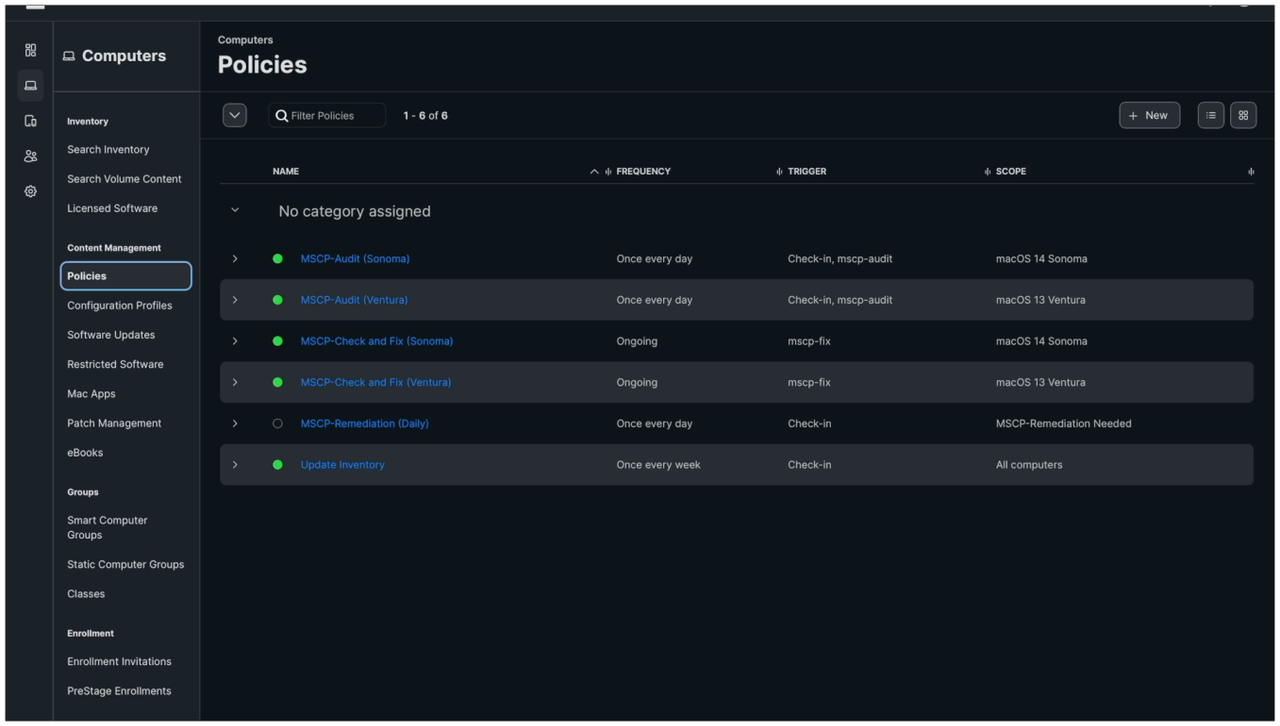
AFTER



So our before numbers….before profiles we have 71 and 69 on Sonoma and Ventura respectively

After profiles installed…25 and 28! Not bad for not even running a script.

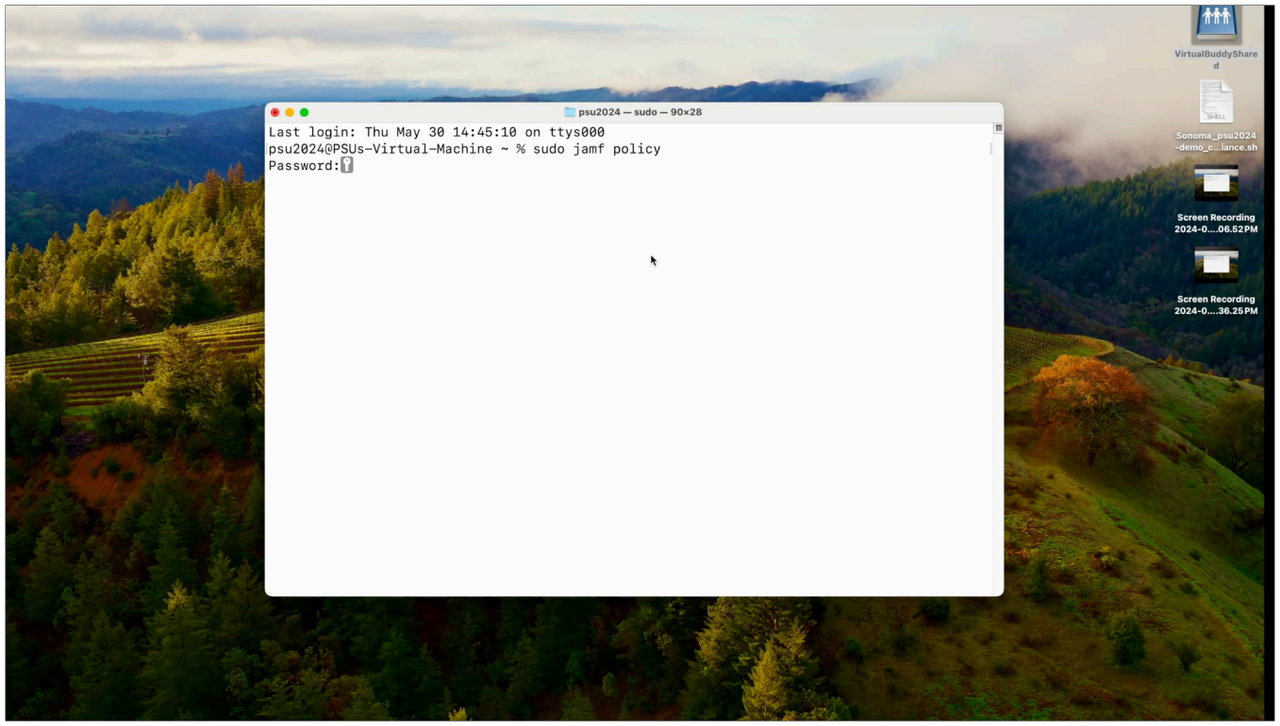
CLICK



Speaking of the script…let’s get that started. **CLICK**

We’ll go to our policies, remember our Remediation policy here.

We’ll edit and activate it and note that it’s running only on the “MSCP-Remediation needed” smart group…we’ll save



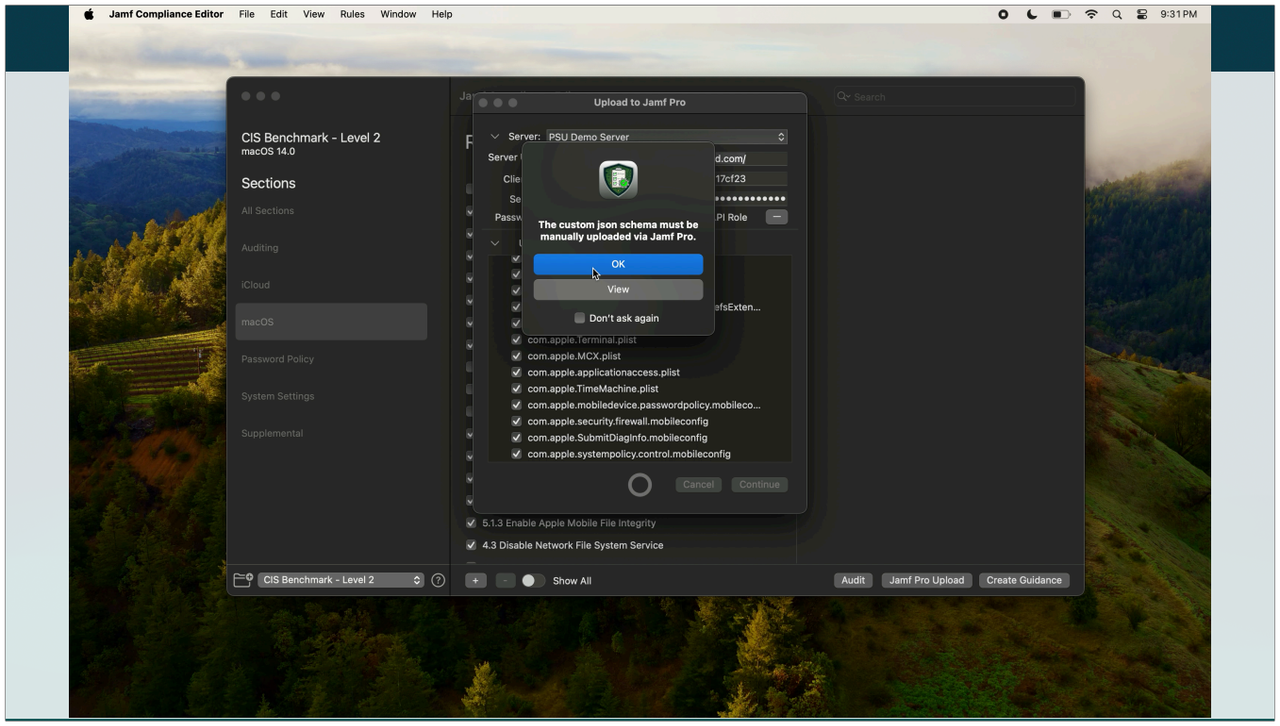
Now…let’s go to Sonoma. **CLICK**

Now that the remediation is active, we’ll run a jamf policy…
You can see it’s running the fix trigger…we’ll let it complete and check our progress in Jamf and WOW. 2 left!

So you may ask, why are there 2 left? Didn’t the script run? Yep! Note the two that are left:

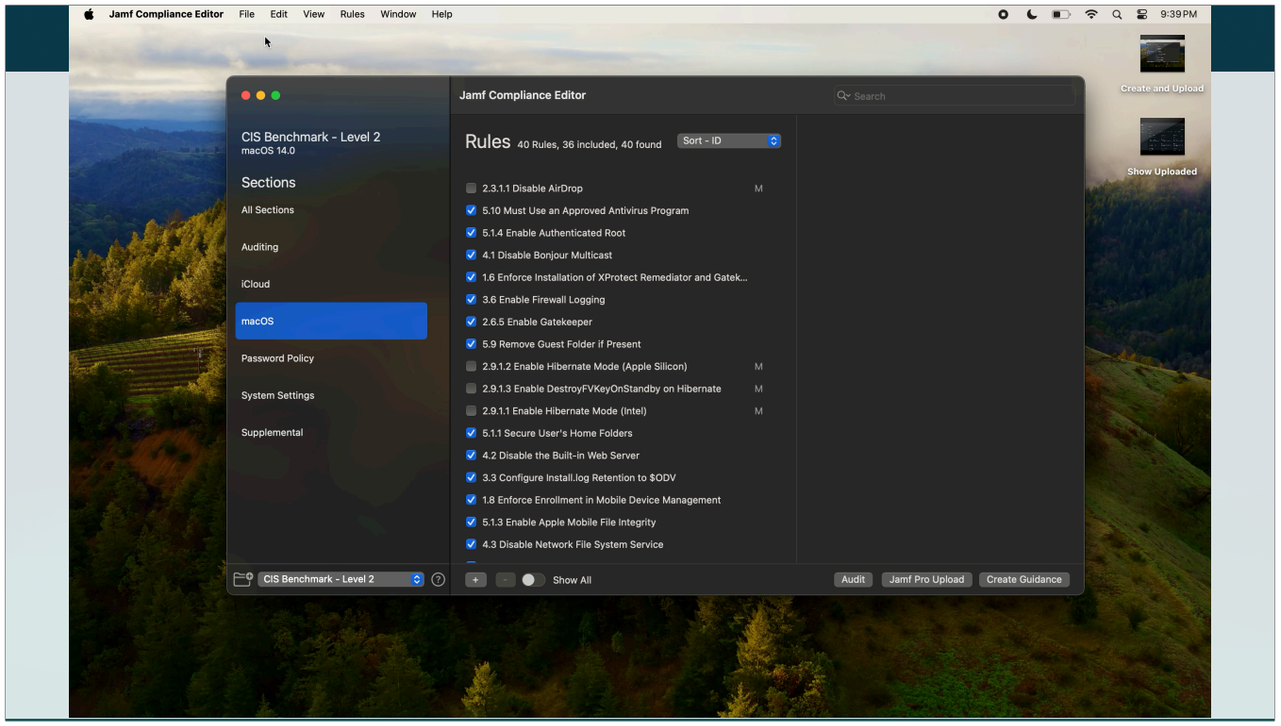
Filevault enforce and wake on network disabled. Both of these show because I’m using virtual machines. I’m not enforcing Filevault on them and the WOL seems to be something odd with virtual buddy. But I know this…and I want to make sure we ignore these errors on some machines.

How do I do that?



If you recall while we were uploading our items to Jamf, we received a prompt that told us we have to manually upload a custom json schema? This is our exemptions schema!

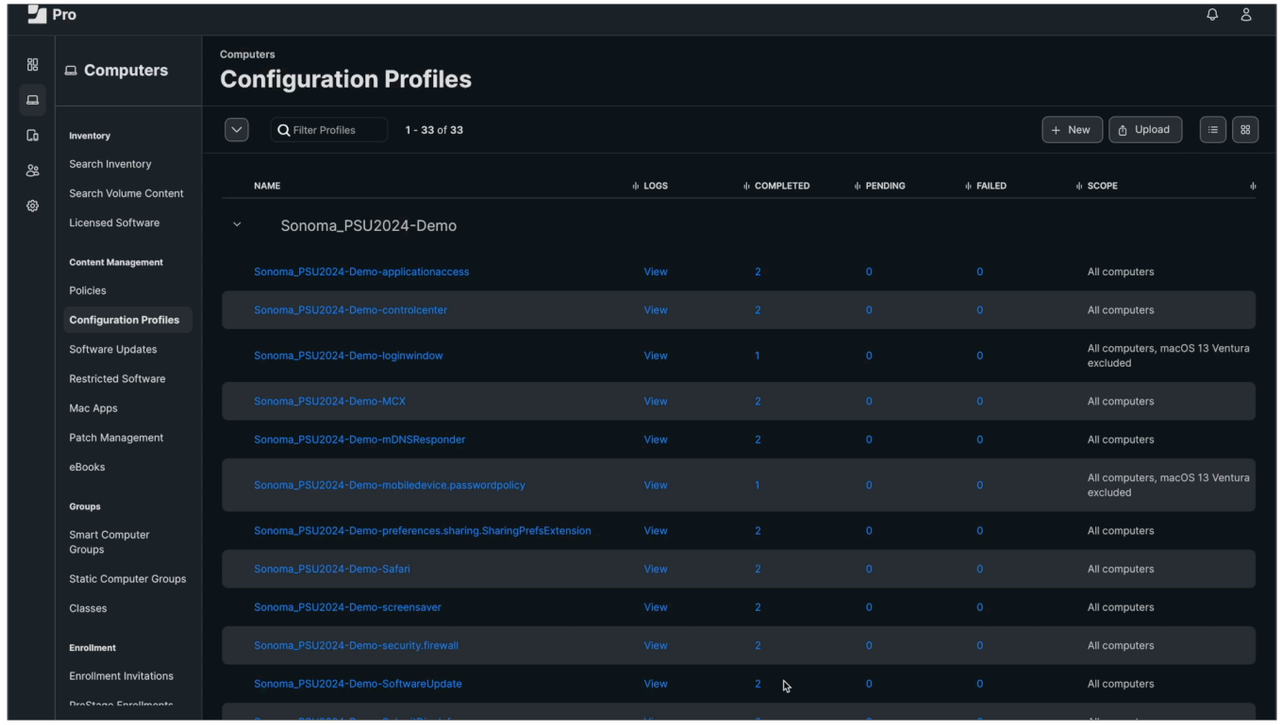
Here's how that works…



First, Let’s get that JSON. **CLICK**

Open the project..file menu, open project folder…build folder, then our build (PSU 2024) and finally, our jamfpro folder.

Inside here we’ll see a jamf JSON schema…let’s get that into jamf and assign it..

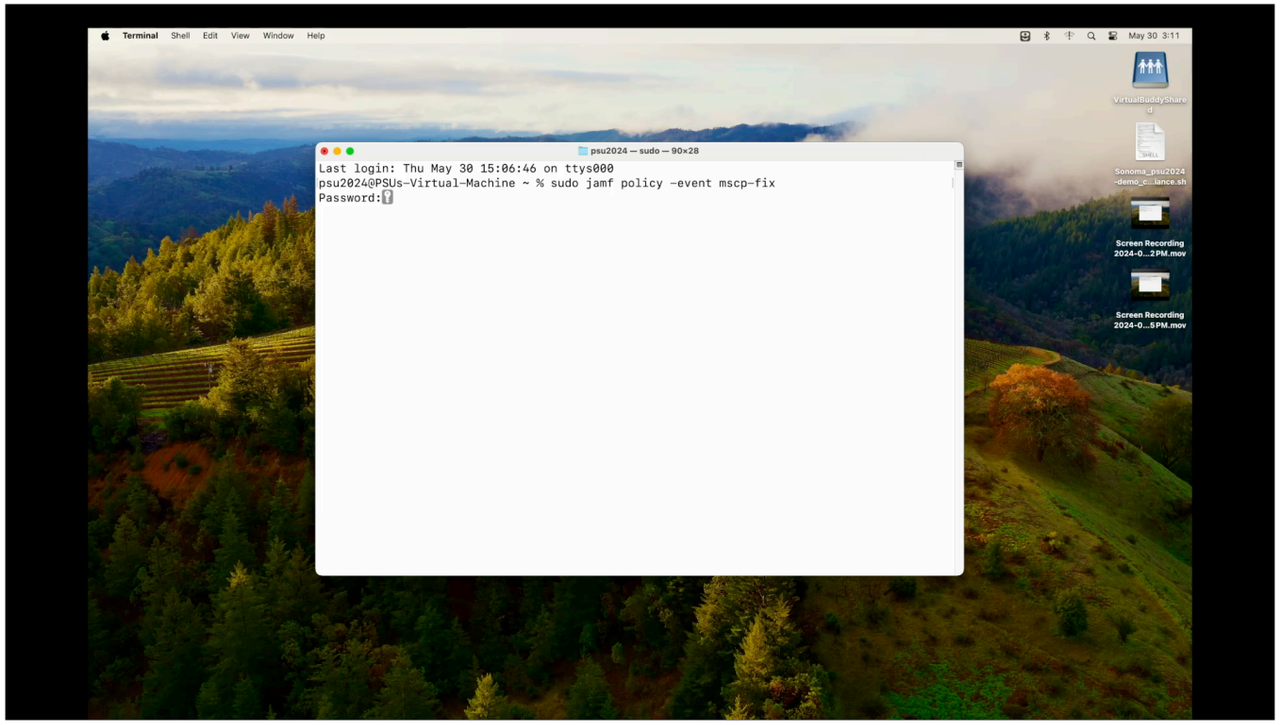


CLICK

Lets make a new config profile. We'll give it a name that describes the exemptions..in our case Virtual Machines.
We'll add a custom application setting, select external applications, then add. Select Custom schema..and we'll now upload our JSON file.

After uploading, et the preference domain to the name of your baseline project .audit.

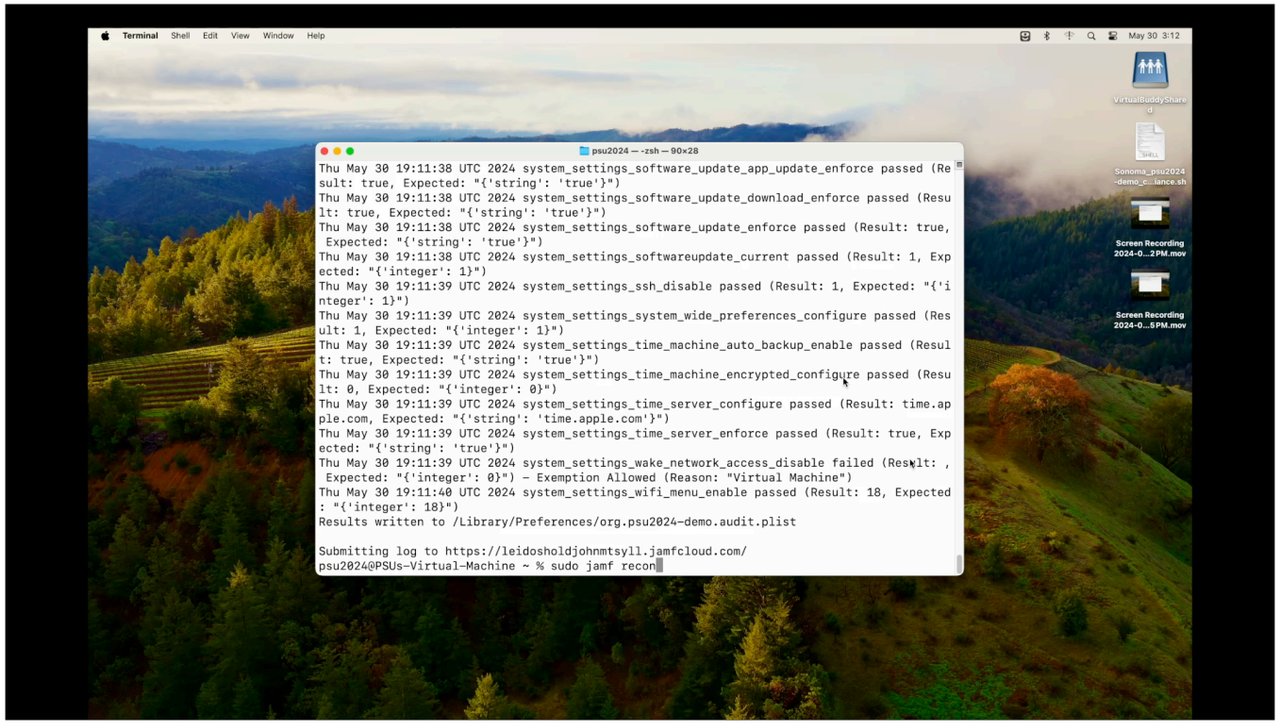
Next, we'll find those two items we want to exempt. Filevault. Configure it…set as true and give a reason. We'll do the same for wake on lan then scope to our machines and lets see what it does.



CLICK
Now we'll re-run the fix on the machine…and check the record…

Oh, it's not updated..why?

Recon! The script wrote the audit, but Jamf didn't collect it. Remember, our policies do that for us when needed…so we'll just..do that manually.



And voila. We’re at 0 with 2 exemptions approved.

EXEMPTIONS != TAILORING

One thing about exemptions. They are not tailoring. You should customize your baselines for your organization, you should not use exemptions to “customize”.

You use exemptions to exclude a small amount of devices from a small amount of rules. If you need extensive customizations, you need to tailor.

AND THAT’S THE GAME!

FOR THE MOST PART

And..that’s really all I have to talk about today!

RESOURCES

My Presentations Git Page

My Blog

macOS Security Compliance Project Git

macOS Security Compliance Project Slack Channel

Jamf Compliance Editor

Apple mSCP Training/Tutorial

POST SESSION DISCUSSION (Q&A)

Feedback

2024 MACADMINS CONFERENCE

WHAT IF YOU WANT TO DO THIS WITH APPLE VISION PRO?

HTTPS://GITHUB.COM/APPLE/DEVICE-MANAGEMENT/TREE/SEED_IOS-18.0_MACOS-15.0

Open JCE

Select iOS

Choose 800-53, option click top to uncheck all

In the GitHub, do a search for the key you want to control and see if it has a vision control

Very manual, but may help