

“Think Different” About Compliance: Is Effective, Automated macOS Configuration Achievable with NIST’s macOS Security Compliance Project?

Author: T. Boone Berlin
www.linkedin.com/in/tbooneberlin

Advisor: Clay Risenhoover

Accepted: 21 November 2022

Abstract

Information security compliance within the Apple macOS ecosystem is an especially challenging problem for IT practitioners. Apple’s Mac computers continue to grow in enterprise deployment market share (Evans J., 2021). Simultaneously, compliance audit reporting and management is a growing concern for IT teams, managers, and executives as the threat of ransomware and other financially motivated attacks have become more prevalent in recent years. Compared to Windows, there remains a relative need for configuration management and compliance tools natively available for macOS. NIST recently released SP800-219, titled "Automated Secure Configuration Guidance from the macOS Security Compliance Project (mSCP)." It aims to provide 1) a new compliance/configuration solution for the macOS ecosystem and 2) an automated configuration tool built around mitigating the compliance challenges from Apple’s annual macOS release cycle.

This paper reports findings between four systems: macOS 10.15 Catalina, 11 Big Sur, 12 Monterey, and 13 Ventura. Nessus, Lynis, and mSCP tools audited each system in a “reference” new installation configuration. Then each was configured by the mSCP tool according to the CIS Level 2 Benchmark for each macOS version. Finally, each system was audited again with the trio of tools to determine the effectiveness of the mSCP tool. Results show that mSCP is an effective configuration management and audit tool. However, due to limitations within macOS and design decisions within mSCP, there is still a notable amount of manual configuration required. mSCP is a flexible command line interface (CLI) based tool and could be easily integrated into custom scripts, further automating the configuration and auditing process. However, documentation is somewhat limited and presents a learning curve.

1. Corraling the Mac

Configuration management can be overwhelming due to numerous individual computers, innumerable settings, and users who want to customize and tweak a tool that reflects their preferences. Compliance is simply the process of auditing the configuration management process. It generates concise reporting metrics to determine if a network, and its assets, are meeting a compliance standard, benchmark, or baseline.¹ Automated tools are necessary to successfully manage enterprise configuration and continuously audit the environment for compliance.

Steve Jobs once said, “you always had to be a little different to buy an Apple computer” (Renesi, 2018). That sentiment has been a hallmark of Apple products for decades. However, in information security, being different is not so lauded. Congruence is necessary to successfully streamline compliance scanning, reporting, and management when talking about compliance. The goal of compliance, enforced congruence to a standard, is seemingly at odds with everything Apple, its products, and its users stand for. As a niche product, tools specific to Mac’s peculiarities and unique software design decisions are few. However, these tools are also required to integrate an increasing number of Macs into enterprise environments.

1.1. Are Macs Worth the Time?

Many of the young people targeted by Apple’s “Think Different” marketing of the 1980s have matured; presumably, some have become decision-makers in organizations worldwide. Additionally, 2020 brought the COVID-19 pandemic, which has ushered in a new wave of work-from-home opportunities in various forms.

Mac has long been relegated to home-use computers before its recent uptick in enterprise acceptance. However, increasing numbers of remote workers suggest that those

¹ Standard, benchmark, and baseline could all be used interchangeably in the discussion of compliance configuration and audit. For this paper, "standard" will refer to the generic discussion of a set of controls designed to enforce a compliance program, "benchmark" will refer specifically to CIS-published benchmarks, and "baseline" will refer to mSCP baselines used to audit and configure a target system.

who chose Mac as their home computer are now more likely to access enterprise networks with those devices.

Reasons aside, the simple fact is that Macs and macOS have been steadily growing in market share over the last ten years (StatCounter, 2022), and this trend seems to be accelerating in the post-pandemic work-from-home shift (Warwick, 2022). IT teams need to spend more time thinking about and planning for the implications of macOS on their enterprise networks.

1.2. Macs are Inherently Secure, Right?

It is crucial to avoid eschewing a deliberate macOS configuration management plan under the assumption that Macs are simply more secure. Much of Mac’s perceived security has been derived from its relative obscurity. Windows simply presents a higher return on investment for attackers. As macOS market share increases, this argument will become increasingly less valid.

No matter how inherently secure or insecure an operating system (OS) is, configuration management and compliance auditing are critical to securing an environment against attacks. Standards-setting bodies for cybersecurity, such as the Center for Internet Security (CIS) or the National Institute of Standards and Technology (NIST), emphasize configuration and compliance in their guidance. The CIS Critical Security Controls prioritize identifying hardware and software assets in an environment and then configuring those assets to a secure standard (CIS, 2022). The NIST Cybersecurity Framework mirrors a similar priority list. After the “Identify” phase, the “Protect” phase follows and once again emphasizes the importance of asset configuration (NIST, 2022a).

1.3. Making Different the Same

Apple’s “Think Different” culture has almost certainly led to decisions that have put its products at odds with configuration management and compliance goals. In contrast to the Windows ecosystem, Macs present an average user with far fewer settings, configuration panes, and dialogs. Windows has long provided enormous, if not

overwhelming, opportunities for configuration through its graphical user interface (GUI). Macs have sought the “it just works” quality by seeking to allow users to *use* their computers versus manage their computers.

Opinions aside on these contrasting design decisions, options for deep configuration management in macOS can be less accessible than in Windows. Couple this with the far smaller enterprise market share held by Mac, and only a few companies offer solutions for automated, centralized macOS configuration management or compliance auditing. Finally, as a function of market share, IT teams are more familiar with Windows configuration management, and macOS represents a new learning curve for many teams to climb.

It is important to note that Apple has been making a concerted effort since macOS 10.15 Catalina to improve compatibility with Mobile Device Management (MDM)² solutions, provide more tools for centralized device management, and provide more tools for compliance audit. Some of these improvements include configuration profiles (.mobileconfig files), property lists (.plist files), Terminal commands and scripts, MDM protocol APIs, and the new Declarative Management process (Faas, 2022). More details are available at Apple’s Developer site for Device Management (Apple, 2022a) and Apple’s Support site for Platform Deployment (Apple, 2022b).

1.4. A Moving Target

Further complicating the issue is Apple’s divergence from what IT teams might consider “normal” software development cycles as established by Microsoft. The Windows operating system typically releases new major versions on various development timelines and then provides security update support for around ten years following the release date.³ Apple, in contrast, has established an annual major version release cycle

² MDM will be used in this paper to refer collectively to MDM, EMM, and UEM. Elaborating further between these concepts is beyond the scope of the paper (Evans A., 2019).

³ It is noted that Microsoft’s development cycle seems to be changing with Windows 11 and future versions being on the “Modern Lifecycle Policy.” However, this still represents a departure from what many IT teams might consider “normal” (Microsoft, 2022).

with ambiguous support timelines (Hoffman, 2018). Apple has also designed planned obsolescence into their software and hardware development cycle by strictly defining what major macOS versions can run on a given Mac computer model (Haslam, 2022). In an industry that favors picking a solution and adhering to it for a while, Apple expects its users to keep their environments updated with the most recent macOS releases and hardware improvements.

Apple’s macOS development and support lifecycle mean IT teams must be prepared to continuously update their environments to keep up with new macOS versions. This presents a unique challenge since OS security compliance standards are organized around major versions. Instead of organizations choosing an OS, selecting an appropriate configuration standard, tailoring it to an organization’s risk tolerance, and then expecting to work to a known standard for up to a decade, macOS requires standards to be created, vetted, and implemented on an annual basis.

2. A NISTy solution

In June of 2022, NIST released a new Special Publication (SP) aimed at macOS configuration and compliance: SP800-219 Automated Secure Configuration Guidance from the macOS Security Compliance Project (mSCP). This document superseded SP800-179, NIST’s series of security configuration guides and checklists for macOS (NIST, 2016). Much like the current CIS library of benchmarks, NIST would update and republish SP800-179 for each new major macOS version. SP800-219 and SP800-179 aimed to provide macOS-specific guidance for implementing the general NIST cybersecurity controls found in SP800-53 Security and Privacy Controls for Information Systems and Organizations. However, SP800-219 is a significant departure from the paradigm of publishing a new checklist and secure configuration guide for each major macOS version release (NIST, 2022b).

SP800-219 introduces the mSCP tool, an open-source, CLI-based tool hosted on GitHub. After providing a high-level overview of the project’s goals and structure, SP800-219 provides introductory documentation for the tool. However, it then points to

the GitHub repository at [usnistgov/macos-security](https://github.com/usnistgov/macos-security) for more detailed guidance and information (M. Trapnell, 2022). SP800-219 will no longer be updated with each new macOS release since the mSCP repository will be updated as required.

SP800-219 states, "The macOS Security Compliance Project (mSCP) provides resources that system administrators, security professionals, security policy authors, information security officers, and auditors can leverage to secure and assess macOS desktop and laptop system security in an automated way" (M. Trapnell, 2022).

Human and machine-readable rules and baselines in the YAML file format define the project's structure. The project's scripts compile the rules directed by the baseline being implemented into a set of custom compliance scripts deployed locally or remotely (through Secure Shell or SSH) to audit or configure a system. The project's scripts also produce custom configuration profiles and property lists that are provided to configure a system to baseline requirements or allow integration with Mac-specific MDM solutions. Finally, mSCP provides a password policy extensible markup language (.xml) file that is used in conjunction with the compliance script to configure the system password policy.

Current mSCP repository branches are available for macOS 10.15 Catalina, 11 Big Sur, 12 Monterey, and 13 Ventura. Each branch includes common standards for the respective macOS version but also allows for the creation of custom baselines. Baselines built around NIST SP800-171, SP800-53 (high, medium, low), CIS Controls v8, CIS Level 1 & Level 2 Benchmarks, and CNSSI-1253 are immediately available in the project. However, with hundreds of individual rules available, it is simple to develop a custom baseline around a specific organization's risk decisions or even write custom rules by following the documented YAML format (mSCP, 2022a).

It is important to note that mSCP is not a fully automatic solution. While the custom compliance scripts can automate most auditing, the configuration of a macOS system still requires manual steps. Manually installing configuration profiles, saving files to the file system, and Terminal commands are required to fully implement a baseline due to mSCP design decisions and macOS limitations.

By moving to an open-source repository, mSCP can quickly implement updated guidance published by standards bodies. Therefore, IT teams can take advantage of all the improvements, security or otherwise, found in major macOS version releases more quickly. During this research, macOS 13 Ventura was released to the public on 24 October 2022. mSCP committed an update to its repository four days before the release and incorporated CIS Level 1 and 2 draft Benchmarks for Ventura. Instead of forcing IT teams to wait for standards, configuration tools, and compliance tools to be updated with lengthy review and publishing delays, mSCP provided a fully functional tool to implement in an environment four days *before* the public Ventura release.

Even during the Beta period of macOS 13 Ventura, a Ventura branch was available in mSCP and leveraged iterative improvements by using CIS macOS 12 Monterey baselines as a starting point for configuring a Ventura Beta system before published standards were available. The authors of mSCP are also quick to respond to questions posted on the project’s GitHub discussion board or bugs reported on the project’s issue reporting tool.

3. Research Method

The intent and design for this research are to evaluate the mSCP’s effectiveness quantitatively and objectively as an automated configuration tool and as a compliance scanner in the four most recent versions of macOS: 10.15 Catalina, 11 Big Sur, 12 Monterey, and 13 Ventura.

3.1. Testing Methodology

To accomplish these goals, the test design followed this methodology:

- a. Create a clean install of each macOS version from an installer obtained from official Apple channels on a VMware Virtual Machine (VM).
- b. Follow a documented and repeatable installation and initialization of macOS on each VM.
- c. Perform minimal configuration required to allow remote scanning with Nessus from a second VM on the same host system.

- d. Perform a Nessus CIS Level 1 and 2 Benchmark compliance scan on each VM to establish an initial reference for each macOS version.
- e. Install and configure mSCP and Lynis tools on each VM.
- f. Perform compliance scans with mSCP, Lynis, and Nessus to establish reference points for each tool and validate that installing the two additional tools did not change Nessus results.⁴ This system state will be referred to as “Reference” throughout the remainder of the paper.
- g. Configure each VM using the automated tools and files produced by mSCP to meet a CIS Level 2 Benchmark for each respective macOS version.
- h. Perform a final compliance scan with all three tools to measure the change in compliance score against the previously established “Reference” in Step f. This system state will be referred to as “Configured” throughout the remainder of the paper.

3.2. Reference System Configuration

macOS installers for each version were downloaded directly from the App Store using links on Apple's Support Site (Apple, 2022c). The installers were converted to .iso images and used to create new VMware VMs on a macOS host system.⁵ Each VM was initialized and set up through macOS's installer. The settings were standardized as much as possible between the four macOS versions, with only minor differences due to dialog changes between the macOS installers. Initialization settings that are relevant to the topic of this research were as follows:

- Decline:
 - Accessibility Setup
 - Apple ID Setup
 - Migration Assistant data migration
 - Screen Time Setup

⁴ Note that mSCP and Nessus compliance scans were against the CIS Level 1 and 2 benchmarks for the respective macOS version, while Lynis compliance scans were against its proprietary standard.

⁵ Ventura was only available through Apple's Beta program when this research began. Therefore, the Ventura VM was created using a cloned Monterey VM that was then enrolled in Apple's Beta program and upgraded to Ventura Beta. As new builds of Ventura Beta were released, the VM was also updated. Near the end of the research period, the public Ventura RTM (Release to Manufacturing) version was announced and used to establish data reported for macOS 13 Ventura (BetaWiki, 2022).

- Sharing crash and usage data with App Developers
- Enable:
 - Location Services
 - Share Analytics with Apple
 - Ask Siri

Following the installation and initialization of each VM, VMware tools were installed to allow for convenience while operating the VMs in parallel with the host system. Next, each VM was updated to the most recent sub-version or security update for the respective macOS version. This step resulted in the following versions for the test VMs:

- macOS Catalina 10.15.7 with Security Update 2022-005
- macOS Big Sur 11.7
- macOS Monterey 12.6
- macOS Ventura 13.0 RTM⁵

To facilitate remote Nessus scans from a different VM, SSH was enabled on each VM, which also required Whole Disk Access to be enabled for Terminal. The Root user was enabled in each VM to allow privileges required for all three tools to perform their scans. Finally, Apple Developer CLI tools were installed on each VM to provide additional Terminal functionality (git command, for example). Before installing mSCP or Lynis, the first Nessus compliance scans were performed corresponding to Step d. in Section 3.1. Finally, mSCP and Lynis were installed and configured with the following set of Terminal commands:

```
% git clone https://github.com/CISOfy/lynis
% sudo chown -R 0:0 lynis
% cd lynis
% git pull
```

```
% git clone https://github.com/usnistgov/macos_security.git
% cd macos_security
% pip3 install -r requirements.txt --user
% bundle install --binstubs --path mscp_gems
% git checkout [appropriate branch name]
% git pull
```

Step f. in Section 3.1 was then performed to establish the reference against which post-configuration scans will be measured.

3.3. Automatic and Manual Functions within mSCP

To ensure the test evaluated only the automated aspects of the mSCP tool, most manual configuration changes and guidance provided by mSCP were excluded from the test. However, as mentioned in Section 2, mSCP automatically produces configuration scripts and files (configuration profiles, property lists, and a password policy .xml file) that configure a system to baseline requirements. Without a centralized MDM solution, the configuration profiles and the password policy .xml file must be manually installed and implemented into the target system. The effect those files had on the system was relevant to the research, and the steps required to utilize these files were performed during the configuration portion of the test. On the other hand, specific individual rules in each branch of the mSCP tool require manual entry of Terminal commands or additional steps to implement a control. These steps were excluded since they need to be manually performed or integrated into a custom Terminal script.

3.4. Exceptions and Inconsistencies

Inconsistencies that could lead to skewed results depending on the version of macOS or the benchmarks available within the two tools are documented in this section. Further, required modifications to mSCP baselines to enable the selected testing methods are also reported.

3.4.1. CIS Benchmark Inconsistencies Between mSCP and Nessus

At the time of testing, inconsistencies between the CIS Baselines available from mSCP and Nessus were noted and are documented in Table 1.

| macOS | CIS ⁶ | Nessus ⁷ | mSCP ⁸ |
|----------------|------------------|---------------------|-------------------|
| 10.15 Catalina | 2.1.0 | 2.0.0 | N/A |
| 11 Big Sur | 2.1.0 | 2.0.0 | 2.1.0 |
| 12 Monterey | 1.1.0 | 1.0.0 | 1.1.0 |
| 13 Ventura | 1.0.0 Draft | N/A | 1.0.0 Draft |

Table 1. CIS Level 1 & 2 Benchmark Versions

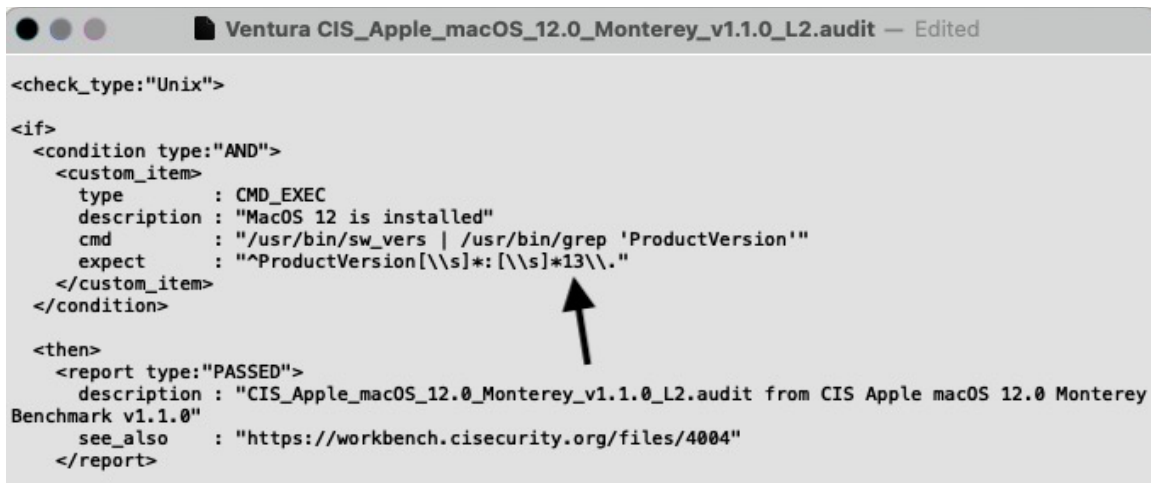
In the case of macOS 10.15 Catalina, the authors of mSCP intentionally decided not to include CIS Level 1 and 2 Benchmarks in the Catalina branch of the project.⁸ Instead, they provided a baseline named "cisv8," which reflected the CIS Critical Security Controls v8 as interpreted by the authors and NIST.⁸

As discussed in Section 2, mSCP committed a new repository promulgating the CIS draft benchmarks for macOS 13 Ventura four days before the public release of Ventura from Apple. In the case of Nessus, CIS benchmarks for Ventura were unavailable, so the CIS macOS 12 Monterey Level 2 v1.1.0 Benchmark published on Tenable’s Audit files page was used instead (Tenable, 2022a). To bypass OS checks built into Nessus .audit files and allow the scan to run, the product version check in the .audit file was modified to expect "13" instead of "12," as shown in Figure 1. The modified audit file was then added to the Nessus Scanner using instructions provided by Tenable (Tenable, 2022b).

⁶ Based on published current benchmarks found at <https://downloads.cisecurity.org/#/> and draft benchmarks found at <https://workbench.cisecurity.org/communities/20>

⁷ Based on the list of available benchmarks in Nessus Professional Scanner at the time of testing.

⁸ Based on comments in baseline files included in mSCP and discussion posts found on the mSCP GitHub page.



```
<check_type:"Unix">
<if>
  <condition type:"AND">
    <custom_item>
      type      : CMD_EXEC
      description : "MacOS 12 is installed"
      cmd       : "/usr/bin/sw_vers | /usr/bin/grep 'ProductVersion'"
      expect    : "^ProductVersion[\s]*:[\s]*13\."
```

Figure 1. Monterey Nessus .audit File Modification for Ventura Scans

3.4.2. Baseline Modification for the Testing Environment

Nessus and Lynis compliance scanners rely on the `sudo` or `su` Terminal commands to execute checks as the Root user. Nessus also relies on SSH access to remotely scan each system.

Since the CIS Level 1 Benchmark for macOS prescribes disabling the Root account and SSH access, the mSCP baselines for macOS 11 Big Sur, 12 Monterey, and 13 Ventura were modified to remove these rules so that post-configuration compliance scans could still be performed.

3.4.3. macOS 10.15 Catalina Specific Configuration Modification

For macOS 10.15 Catalina, disabling the Root account was removed from the "ciscv8" baseline provided by mSCP. However, this baseline also contained six rules which enabled and enforced multi-factor authentication (MFA) for system login, SSH, `sudo`, and `su`. Since setting up and testing MFA was beyond the scope of this research, these rules were removed to allow uninhibited access to the macOS 10.15 Catalina VM.

4. Findings and Discussion

This section will report raw data and scan results with comparisons between system states before and after automated configuration with the mSCP tool. These data

points will be followed by a discussion of the results and an investigation into contributing factors.

4.1. Effect of Installing Tools

Section 3.1 discusses validating no change in Nessus compliance scores following the installation of mSCP and Lynis. Comparing scan results from Step d. against results from Step f. showed no change in scores. Based on these findings, all future comparisons will be between the “Reference” state (Step f.) and the “Configured” state (Step h.).

4.2. Data Reporting Format by Tool

Nessus compliance reporting metrics provide pass, fail, or warning results. Pass and fail result from a command returning the expected result or not, respectively. Warnings represent either an inconclusive result, a control that requires additional evaluation, or a control that requires manual auditing. Nessus also calculates the percentage of the pass, fail, and warning results against the total number of controls checked.

mSCP reports simple statistics following a scan with passed or failed checks and a compliance percentage. Failed checks include both inconclusive and failed results. Percent compliance is simply the passed checks divided by the total number of checks.

Lynis (non-enterprise version) was selected as one of the few actively supported compliance audit tools available for macOS without a license. It is also a host-based tool, like mSCP, which contrasts the remote scanning of Nessus. However, it is important to recognize that Lynis is scanning against its proprietary standard, which is not easily compared to the CIS Level 1 and 2 Benchmarks used in mSCP and Nessus. Lynis does not provide an apples-to-apples comparison against mSCP and Nessus results. However, it is a recognized tool for auditing macOS system configurations and provides valuable context to the results from Nessus and mSCP. Lynis reports several metrics, but only the “Hardening Index” will be used for this research since the rest are not especially helpful without a common standard.

4.3. Reported Compliance Data

This section consists of three data tables. In Table 2, "Reference" corresponds to the system state denoted by Step f., and "Configured" corresponds to Step h. in Section 3.1. Table 3 provides a simple statistical analysis of the results. Table 4 reports the number of rules used by Nessus and mSCP.

| macOS | | Nessus | | | mSCP | | Lynis |
|----------------|------------|--------|---------|------|------|------|-------|
| 10.15 Catalina | Reference | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 51 | 18 | 54 | 10 | 79 | 62 |
| | Percentage | 41% | 15% | 44% | 11% | 89% | N/A |
| | Configured | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 66 | 18 | 39 | 83 | 6 | 72 |
| | Percentage | 54% | 15% | 32% | 93% | 7% | N/A |
| 11 Big Sur | Reference | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 48 | 19 | 58 | 34 | 65 | 62 |
| | Percentage | 38% | 15% | 46% | 34% | 66% | N/A |
| | Configured | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 68 | 19 | 38 | 94 | 5 | 72 |
| | Percentage | 54% | 15% | 30% | 95% | 5% | N/A |
| 12 Monterey | Reference | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 45 | 17 | 57 | 35 | 65 | 59 |
| | Percentage | 38% | 14% | 48% | 35% | 65% | N/A |
| | Configured | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 63 | 17 | 39 | 96 | 4 | 69 |
| | Percentage | 53% | 14% | 33% | 96% | 4% | N/A |
| 13 Ventura | Reference | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 52 | 13 | 47 | 38 | 61 | 59 |
| | Percentage | 46% | 12% | 42% | 38% | 62% | N/A |
| | Configured | Pass | Warning | Fail | Pass | Fail | Index |
| | Numerical | 68 | 13 | 31 | 93 | 6 | 68 |
| | Percentage | 61% | 12% | 28% | 94% | 6% | N/A |

Table 2. Compliance Scores by macOS Version and Scanner

| macOS | Improvement Reported ⁹ | | | Difference in Reported Improvement ¹⁰ | |
|----------------|-----------------------------------|------|-------|--|------------|
| | Nessus | mSCP | Lynis | Reference | Configured |
| 10.15 Catalina | 12% | 82% | 10 | -30% | 40% |
| 11 Big Sur | 16% | 61% | 10 | -4% | 41% |
| 12 Monterey | 15% | 61% | 10 | -3% | 43% |
| 13 Ventura | 14% | 56% | 9 | -8% | 33% |

Table 3. Compliance Score Reporting Comparison

| macOS | Number of Rules | | | | |
|----------------|-----------------|----------------|--------------------------------|-----------------|------------------------------------|
| | Nessus | mSCP as Tested | mSCP Rules Removed for Testing | mSCP unmodified | Latest CIS Benchmark ¹¹ |
| 10.15 Catalina | 123 | 89 | 7 | 96 | 92 |
| 11 Big Sur | 125 | 99 | 2 | 101 | 94 |
| 12 Monterey | 119 | 100 | 2 | 102 | 97 |
| 13 Ventura | 112 | 99 | 2 | 101 | 104 |

Table 4. Rules Checked by macOS Version and Scanner

4.4. Notable trends

Table 2 above and Figure 2 below show that reference system compliance scores may indicate that macOS is shipping as an increasingly secure OS. The data in Table 4 indicates that the number of controls audited by mSCP and Nessus may be converging on the number of controls published in CIS benchmarks.

⁹ Calculated by (Configured Pass % - Reference Pass %) from Table 2.

¹⁰ Calculated by (mSCP Pass % - Nessus Pass %) from Table 2.

¹¹ See Table 1.

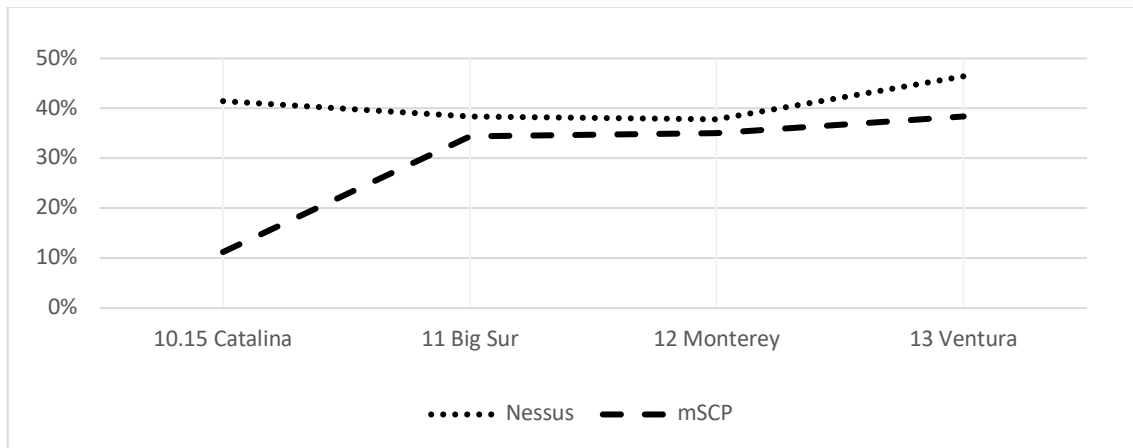


Figure 2. Reference System Compliance Scores by macOS Version

4.5. Framing for Discussion and Investigation

Attempting to draw meaningful conclusions from some of the data presented in Tables 2 and 3 is challenging due to inconsistencies documented in Section 3.4.

Lynis data shows little change from one macOS version to the next, although its improvement reflects the more modest improvements reported by Nessus when compared to mSCP. Lynis data included in the results provide a valuable data point, but delving into the reasons for the differences between Lynis and the other two tools in the test is beyond the scope of the research.

For macOS 10.15 Catalina, Nessus is reporting based on CIS Catalina Level 2 Benchmark v2.0.0, while mSCP is reporting based on an interpretation of the CIS Critical Security Controls v8 created by the authors of mSCP and NIST. The seven rules were also removed from the mSCP “cisv8” baseline to facilitate testing. The inconsistencies created by these issues are evident in the data.

Nessus reporting for macOS 13 Ventura is based on the modified CIS Monterey Level 2 Benchmark v1.1.0, as discussed in Section 3.4.1. In contrast, mSCP reporting is based on a draft version of CIS Ventura Level 2 Benchmark v1.0.0. Once again, this inconsistency does not lend itself to deeper analysis.

The most consistent data is provided by comparing results from Nessus and mSCP in macOS 11 Big Sur and 12 Monterey. Since Monterey is the newest macOS version providing consistent data, all further analysis will be based on the Monterey data. These findings are expected to be the most applicable for current and future macOS versions. The analysis will also highlight important details about the mSCP structure, which applies to any project branch for investigating and comparing audit findings.

4.6. Discussion and Investigation of macOS 12 Monterey Data

4.6.1. mSCP Baseline File Supplemental Rules Section

The mSCP baseline YAML file structure is organized around descriptive headings that provide information about the baseline and then define the rules included in the baseline. The individual rules are further organized into section headings to group rules by type. One of these sections is "Supplemental," which refers to a set of rules that provide additional guidance on implementing specific controls within macOS. The mSCP `cis_lvl2` baseline in the Monterey branch includes the supplemental rules documented in Table 5. It is essential to investigate these rules to understand which ones do not automatically correct failed compliance checks or which controls are not automatically audited.

Row E of Table 5 deals with the `pwpolicy.xml` file provided by mSCP and is briefly discussed in Sections 2 and 3.3. The supplemental rule provides instructions for saving the file to an appropriate location in the target's file system and running a command to configure the system's password policy. Additionally, the configuration script generated by the mSCP has a variable field that must be updated with the location of the `pwpolicy.xml` file. The `KNOWN_ISSUES.txt` file provides more detail and is located in the mSCP repository at: `macos_security/scripts`.

The manual steps for implementing the `pwpolicy.xml` file were performed after the scans in the reference state were complete but before the configured state scans were conducted.

| | Rule | Applicable | Notes |
|---|------------------------------|------------|--|
| A | supplemental_cis_manual | Yes | Provides a list of CIS controls denoted as "manual" according to the Monterey v1.1.0 Benchmark. Four controls listed as "manual" in the CIS benchmark are not included in this rule but are covered by the password policy rules in the cis_lvl2 baseline and guidance on row E of this table. |
| B | supplemental_controls | No | Provides a list of "out of scope" NIST SP800-52 requirements for administrative and procedural processes that cannot be addressed with technical configurations. These do <u>not</u> apply to meeting a CIS benchmark. |
| C | supplemental_filevault | Yes | Provides guidance on manually enabling FileVault in conjunction with the syspref_filevault_enforce rule, which checks for correct configuration but does not automatically configure an out-of-compliance system. This rule <u>is</u> included in the cis_lvl2 baseline. |
| D | supplemental_firewall_pf | No | Provides guidance on manually enabling and configuring the firewall in conjunction with a rule <u>not</u> included in the cis_lvl2 baseline. |
| E | supplemental_password_policy | Yes | Provides guidance on using the pwpolicy.xml file provided by mSCP to achieve a compliant password policy in conjunction with the pwpolicy_lower_case_character_enforce and the pwpolicy_upper_case_character_enforce rules, which check for compliance but cannot automatically configure an out-of-compliance system. These rules <u>are</u> included in the cis_lvl2 baseline. |
| F | supplemental_smartcard | No | Provides guidance on enabling and enforcing smart cards. These rules are <u>not</u> included in the cis_lvl2 baseline. |

Table 5. mSCP Supplemental Rules Matrix

4.6.2. mSCP Rule Files Tagged as Manual

The mSCP rule YAML file structure is organized around descriptive headings that provide information about the rule and then define functional aspects of the rule.

Fields include the automated check, expected result, automated fix, references, applicable macOS version, variables, tags, and if a configuration profile should be generated. The tags field defines what baselines the rule is included in and may also carry a "manual" tag. If a rule is tagged as “manual,” mSCP will not automatically configure a system for that specific control but usually provides information under the "fix" heading of the rule file to achieve a compliant configuration manually. It is important to recognize that rules tagged as “manual” are based on mSCP design decisions, while controls listed in the supplemental_cis_manual rule (Table 5, row A) are manual controls based on CIS guidance.

In the case of the Monterey branch, the following command yields the rules tagged as “manual.” None of these rules are included in the cis_lvl2 baseline and do not affect this analysis. However, in other mSCP branches, some “manual” rules were used in the respective cis_lvl2 baseline and contributed to controls reported as failed.

```
% grep -nR "\- manual" rules
rules/os/os_ess_installed.yaml:28:  - manual
rules/os/os_filevault_authorized_users.yaml:33:  - manual
rules/os/os_certificate_authority_trust.yaml:31:  - manual
rules/pwpolicy/pwpolicy_temporary_or_emergency_accounts_disable
.yaml:79:  - manual
```

4.6.3. Failed mSCP Controls in Configured State

Four controls were returned as failed when mSCP audited the configured state of macOS 12 Monterey. Table 6 documents these failed controls while listing the two rules intentionally removed from the cis_lvl2 baseline to support the testing environment.

The two removed rules were not reported as failed controls because of how mSCP builds the compliance script from a selected baseline. By removing the rules from the baseline prior to generating the compliance script, those two rules do not exist in the script for configuration or audit.

An alternative method would be to generate a compliance script for an out-of-compliance configuration that excludes the two rules. Then a second compliance script, including the two rules, could be generated but only used for auditing. This would have

allowed the system to be configured out of compliance to support testing but would have also reported the out-of-compliance controls during the audit. mSCP also includes a method for listing out-of-compliance controls as “allowed exceptions” per organizational policy. Further information is available on the mSCP GitHub Wiki Compliance Script Page (mSCP, 2022b).

| | Rule | Notes |
|---|----------------------------------|---|
| A | os_efi_integrity_validated | This check is related to firmware checks and is expected due to running macOS on a VM. |
| B | pwpolicy_account_lockout_enforce | This control is implemented by a configuration profile which was manually verified with Terminal commands as correctly configured. However, the automated check did not return the expected result yielding a failed control. This seems to be a bug in the Terminal command for this rule. |
| C | pwpolicy_max_lifetime_enforce | This control is implemented by a configuration profile which was manually verified with Terminal commands as correctly configured. However, the automated check did not return the expected result yielding a failed control. This seems to be a bug in the Terminal command for this rule. |
| D | sysprefs_filevault_enforce | Expected based on findings of Table 5, row C, which requires manual configuration to enable FileVault. |
| E | os_root_disable | Rule removed from cis_lvl2 baseline to support the testing environment and not reported by mSCP per discussion in Section 4.6.3. |
| F | sysprefs_ssh_disable | Rule removed from cis_lvl2 baseline to support the testing environment and not reported by mSCP per discussion in Section 4.6.3. |

Table 6. Failed Checks in Configured State

4.6.4. Failed or Warning Nessus Controls in Configured State

Nessus scans of the configured macOS Monterey system returned seventeen warnings and thirty-nine failed results. These Nessus findings were investigated by cross-checking results against the CIS benchmarks (v1.0.0 & v.1.1.0), evaluating the system manually, and comparing them to mSCP results. The goal was to determine how accurately these results reflected the automated system configuration performed by mSCP.

Nine warnings and three failed findings match CIS controls marked as “manual” in v1.1.0 of the benchmark. Per Table 5, row A, these controls are not automatically configured by mSCP and are not representative of a failure to properly configure the system. However, six of these twelve controls have automated audit methods listed in the CIS benchmark. This represents a missed opportunity for mSCP to provide automated audit feedback to IT practitioners either by providing a warning finding (this functionality is not currently programmed into mSCP) or listing these findings as failures to alert users to pursue manual compliance checks and configuration. These twelve controls will be treated as warnings in the Nessus results because they represent valid findings but findings that mSCP is not currently programmed to audit or report.

Six warnings and four failed findings were removed in the latest v1.1.0 of the CIS benchmark. mSCP configured the system based on v1.1.0, while Nessus audited the system based on v1.0.0 of the benchmark. Therefore, these findings do not represent a failure to properly configure or audit the system with mSCP. These ten findings will be removed from the Nessus results.

Two warnings and twenty-three failed findings were verified to be compliant by manual system checks. This seems to be due to updated audit methods in v1.1.0 of the benchmark or errors in the Nessus audit method. These twenty-five findings will be treated as pass findings in the Nessus results.

Four failed findings agreed with mSCP findings on rows A, D, E, and F of Table 6. The remaining five failed findings were valid and represented missed configuration and audit by mSCP. Further, Nessus validated that mSCP findings in rows B and C of Table 6 were erroneous by finding the configured system compliant with those controls.

| | Rule | Notes |
|---|---|--|
| A | 2.7.1 Ensure Backup Up Automatically is Enabled | mSCP enabled this control with a configuration profile that turned on Automatic Backups. However, in the absence of a configured backup volume, mSCP failed to meet the intent of the control. This should be tagged as a manual rule in mSCP. |

| | | |
|---|--|---|
| B | 2.7.2 Ensure Time Machine Volumes Are Encrypted | Related to the above finding, no encryption is enabled in the absence of a configured volume, and the control should have failed in mSCP. |
| C | 3.5 Ensure Access to Audit Records Is Controlled - /var/audit | mSCP audit rules do not seem to perform checks on /var/audit files where out-of-compliance files exist. |
| D | 5.1.1 Ensure Home Folders Are Secure | mSCP check returns the expected result, but the check provided in the CIS benchmark shows out-of-compliance folders. This indicates an error in the mSCP check. |
| E | 5.1.7 Ensure No World Writable Files Exist in the Library Folder | Nessus check finds one world writable file. mSCP check finds none. However, Nessus also returns an error. Inconclusive result and indicates errors in both tools' checks. |
| F | 2.11 Ensure EFI Version Is Valid and Checked Regularly - valid | Corresponds with mSCP finding on Table 6, row A |
| G | 2.5.1.1 Ensure FileVault Is Enabled | Corresponds with mSCP finding on Table 6, row D |
| H | 5.6 Ensure the 'root' Account Is Disabled | Corresponds with mSCP finding on Table 6, row E |
| I | 2.4.5 Ensure Remote Login Is Disabled | Corresponds with mSCP finding on Table 6, row F |

Table 7. Valid Nessus Findings

4.6.5. Normalized Results

| Configured | Nessus | | | mSCP | |
|------------|--------|---------|------|------|------|
| | Pass | Warning | Fail | Pass | Fail |
| Numerical | 91 | 12 | 6 | 99 | 3 |
| Percentage | 83% | 11% | 6% | 97% | 3% |
| Rules | 109 | | | 102 | |

Table 8. Normalized macOS 12 Monterey Configured System Results

Per Section 3, this research aims to evaluate the mSCP's effectiveness quantitatively and objectively as an automated configuration tool and compliance scanner. To accomplish this goal, data normalization was used to better compare Nessus and mSCP results.

The original data reported for the configured macOS 12 Monterey system in Table 2 were normalized and are presented in Table 8. The findings and adjustments discussed throughout Section 4.6 were used to modify the data in Table 2 and produce the data in Table 8.

The three failed controls directly attributable to the testing environment (Table 6, rows A, E, and F and Table 7, rows F, H, and I) were moved to the pass columns of both the Nessus and mSCP findings. This better reflects real-world results had the configured system been in a production environment.

When the normalized data is compared to the original data presented in Table 2, it is evident that mSCP provides somewhat optimistic scores. This is attributed to 1) five missed findings reported by Nessus but not reported by mSCP (Table 7, rows A-E) and 2) mSCP’s design decision not to include CIS benchmark “manual” controls in audit reporting. However, mSCP also artificially depressed its scores by erroneously reporting two failed controls that were compliant.

On the other hand, Nessus' results required heavy normalization, as discussed in Section 4.6.4. The findings were adjusted by correcting twenty-five controls erroneously reported as failed or warnings and then removing the findings reporting on deleted controls per v1.1.0 of the CIS benchmark. After these adjustments, a more realistic score emerges, which reflects the five controls missed by mSCP and reporting on CIS “manual” controls that mSCP ignores in its audit reporting.

One additional note: The CIS v1.1.0 benchmark added two new controls relative to the v1.0.0 benchmark. These controls were not validated by Nessus but were validated manually to be compliant in the configured system and included in the mSCP cis_lvl2 baseline for macOS 12 Monterey.

5. Recommendations

5.1. Recommendations for Improvement

As highlighted in Section 4.6.5, mSCP compliance score reporting ignores the CIS controls notes as "manual" in the benchmark and may skew results to be more

optimistic than reality. It is recommended that the mSCP incorporates a mechanism for raising these controls to the user’s attention through the compliance score reporting system.

5.2. Recommendations for Use

Mac administrators will find the mSCP is a highly effective and flexible automated configuration management tool for macOS which leverages the latest improvements in the OS to incorporate MDM technologies and aids in centralized configuration management. Further, the project's open-source and human-readable rule files also provide a wealth of information about configuration settings, Terminal commands, and configuration profiles available to the Mac administrator manually configuring their environment.

As an audit tool, the mSCP is also an effective and valuable tool. However, it is crucial to carefully evaluate the entire rule set implemented by a selected baseline to ensure reported compliance scores reflect reality before reporting results to management or auditors. A particular emphasis on the supplemental rules section of a baseline and any rules tagged as "manual" is warranted. Additionally, a second compliance scanner is recommended to validate mSCP results since errors may be present in any compliance auditing tool.

As a CLI-based and non-proprietary tool, mSCP produces simple, human-readable, and easily parsed reports and findings. Incorporating mSCP CLI into an enterprise’s custom scripts could further extend its utility, and its reports may lend significant visibility into analyzing audit findings to correct compliance issues.

Finally, make use of mSCP’s GitHub issue reporting and discussion boards. The authors are active participants in the project and are quick to resolve issues or answer questions.

5.3. Recommendations for Future Research

Examine batch installation of configuration profiles to multiple hosts. For this research, the configuration profiles produced by mSCP were manually installed one by

one into the target system. This is unsustainable for any enterprise environment. Research on how to script configuration profile installation or incorporate MDM solutions would be valuable.

Automate the mSCP configuration, audit, and configuration process with custom scripting. Using the mSCP is still somewhat user intensive. Research on incorporating the mSCP into scripts that would allow fully automatic deployment, auditing, and reporting would be valuable to enterprise users.

Investigate and evaluate other mSCP features. Several features written into mSCP have not been discussed in this paper. A deeper dive into other features could be valuable: generating and using Security Content Automation Protocol (SCAP) profiles, working with the automatically generated .plist files, generating and deploying signed configuration profiles, using and evaluating the effectiveness of other baselines included in the project, or mapping rules to baselines not included in the project with the mapping script.

6. Conclusion

This paper sought to introduce and evaluate the effectiveness of a new tool available to IT teams seeking configuration management and compliance auditing for Macs in their environments. Mac administration is characterized by working with machines and software that are different from the Windows environments most organizations use. This paper highlighted two challenges 1) Apple’s annual major macOS version release cycle creating high compliance standard turnover and 2) the relatively few tools available to administrators to automatically configure and audit Mac environments. Despite some minor faults, mSCP’s biggest strength is a transparent, open-source code base with ready support and regular updates from its authors. As an open-source project, its agility and adaptiveness were observed to be uniquely well-suited to address the challenges of an annual macOS development cycle. As a configuration management tool, the data collected and analyzed in this paper demonstrate that mSCP is an effective and valuable configuration management tool and compliance auditing tool.

References

- Apple. (2022a, October). *Device Management*. Retrieved from Apple Developer Documentation: <https://developer.apple.com/documentation/devicemanagement>
- Apple. (2022b, October). *Platform Deployment*. Retrieved from Apple Support: <https://support.apple.com/en-asia/guide/deployment/welcome/web>
- Apple. (2022c, October). *How to download macOS*. Retrieved from Apple Support: <https://support.apple.com/en-us/HT211683>
- BetaWiki. (2022, October). *macOS Ventura*. Retrieved from BetaWiki: https://betawiki.net/wiki/MacOS_Ventura
- CIS. (2022, October). *The 18 CIS Critical Security Controls*. Retrieved from CIS: <https://www.cisecurity.org/controls/cis-controls-list>
- Evans, A. (2019, May 10). *Mastering mobility management: MDM vs EMM vs UEM*. Retrieved from Hexnode: <https://www.hexnode.com/blogs/mastering-mobility-management-mdm-vs-emm-vs-uem/>
- Evans, J. (2021, January 27). *Macs reach 23% share in US enterprises, IDC confirms*. Retrieved from ComputerWorld: <https://www.computerworld.com/article/3604601/mac-reach-23-share-in-us-enterprises-idc-confirms.html>
- Faas, R. (2022, June 20). *How Apple is updating mobile device management*. Retrieved from ComputerWorld: <https://www.computerworld.com/article/3664129/how-apple-is-updating-mobile-device-management.html>
- Haslam, K. (2022, October 10). *Find out what versions of macOS you can run on your Mac*. Retrieved from Macworld:
- T. Boone Berlin, www.linkedin.com/in/tbooneberlin

- <https://www.macworld.com/article/673697/what-version-of-macos-can-my-mac-run.html>
- Hoffman, C. (2018, October 2018). *Which Releases of macOS Are Supported With Security Updates?* Retrieved from How-To Geek:
<https://www.howtogeek.com/350901/which-releases-of-macos-are-supported-with-security-updates/>
- M. Trapnell, E. T. (2022, June). *Automated Secure Configuration Guidance from the macOS Security Compliance Project (mSCP)*. Retrieved from NIST:
<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-219.pdf>
- Microsoft. (2022, October). *Microsoft Lifecycle Policy*. Retrieved from Microsoft Documentation: <https://learn.microsoft.com/en-us/lifecycle/>
- mSCP. (2022a, October). *Rules YAML Format*. Retrieved from GitHub:
https://github.com/usnistgov/macos_security/wiki/Rules
- mSCP. (2022b, October). *Compliance Script*. Retrieved from GitHub:
https://github.com/usnistgov/macos_security/wiki/Compliance-Script
- NIST. (2016, December). *SP800-179*. Retrieved from NIST:
<https://csrc.nist.gov/publications/detail/sp/800-179/archive/2016-12-05>
- NIST. (2022a, October). *Cybersecurity Framework*. Retrieved from NIST:
<https://www.nist.gov/cyberframework>
- NIST. (2022b, October). *macOS Security*. Retrieved from NIST:
<https://csrc.nist.gov/Projects/macos-security>

Renesi, M. (2018, March 25). *Think Different*. Retrieved from Medium:

<https://medium.com/ad-discovery-and-creativity-lab/think-different-b566c2e6117f>

StatCounter. (2022, Oct). *Desktop Operating System Market Share Worldwide*. Retrieved

from StatCounter: <https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-200901-202209>

Tenable. (2022a, October). *Audits*. Retrieved from Tenable:

<https://www.tenable.com/audits>

Tenable. (2022b, October). *Documentation*. Retrieved from Add a Custom Audit File:

<https://docs.tenable.com/tenable-sc/Content/AddCustomAuditFile.htm>

Warwick, S. (2022, October 21). *Apple's Mac sales surge 40% in declining PC market*.

Retrieved from iMore: <https://www.imore.com/mac/apples-mac-sales-surge-40-in-declining-pc-market>

Appendix A

mSCP Resources and Further Reading

- NIST macOS Security - <https://csrc.nist.gov/Projects/macOS-security>
- NIST SP800-219 - <https://csrc.nist.gov/publications/detail/sp/800-219/final>
- mSCP GitHub - https://github.com/usnistgov/macOS_security
- Apple Support mSCP Page - <https://support.apple.com/en-asia/guide/sccc/sccc22685bb2/web>
- Apple macOS User Guide: Configuration Profiles - <https://support.apple.com/en-asia/guide/mac-help/mh35561/13.0/mac/13.0>
- Getting to Know: mSCP Part 1 - <https://golbiga.medium.com/getting-to-know-macos-security-compliance-project-part-1-31d0689b7c30>
- Getting to Know: mSCP Part 2 - <https://golbiga.medium.com/getting-to-know-macos-security-compliance-project-part-2-24131b60cdfb>
- MacAdmins Conference mSCP Session - <https://youtu.be/mpEBEelSWII>
- Incorporating mSCP into Jamf Pro - <https://github.com/Honestpuck/NIST-macos-security-HOWTO>
- The Mac Observer - <https://www.macobserver.com/news/product-news/macOS-security-compliance-project/>
- Grove Technologies - <https://grovetech.co/blog/nist-compliance-and-systems-hardening-in-an-all-mac-environment/>
- dataJAR - <https://datajar.co.uk/the-macos-security-compliance-project/>
- JumpCloud - <https://community.jumpcloud.com/t5/security/mac-security-compliance-project-mscp-and-more/m-p/671>

Appendix B mSCP Usage Notes

Getting started with mSCP is somewhat convoluted. Ample reading and guidance are provided in App A. However, these notes provide a concise set of steps required to build a compliance script and use it for auditing or configuring a system the first time.

```
% git clone https://github.com/usnistgov/macos_security.git
% cd macos_security
% pip3 install -r requirements.txt --user
% bundle install --binstubs --path mscp_gems

# List all available branches
% git branch -a
# Work from the appropriate branch, not the main branch.
% git checkout [appropriate branch name]

# Lists available baselines in the selected branch. The -h
# flag is available to list other options for the script.
% ./scripts/generate_baseline.py -l

# This command will produce a YAML file at
# macos_security/build/baselines/[selected baseline].yaml.
% ./scripts/generate_baseline.py -k [selected baseline]

# This command will produce a directory at
# macos_security/build/[selected baseline]. If a customized
# baseline is desired, the simplest method is to modify the
# baseline produced in the last step before running this
# command. The -h flag is available to list other options for
# the script.
% ./scripts/generate_guidance.py -s -p
build/baselines/[selected baseline].yaml

# After looking through the new directory invoking the
# following command without options will run the compliance
# script interactively. Options for non-interactive use
# include: --check, --stats, --fix, and --reset
% sudo ./build/[selected baseline]/[selected
baseline]_compliance.sh [options]

# After running the compliance script, results are written to:
# /Library/Preferences/org.[selected baseline].audit.plist
# /Library/Logs/[selected baseline]_baseline.log
# Although these locations can be easily modified in the
# first few lines of the script.
```

Appendix C

Presentation and Raw Data

Raw testing data and presentation slides are available at
https://github.com/paparooky/macOS_security_evaluation