How CrowdStrike Analyzes macOS Malware to Optimize Automated Detection

crowdstrike.com/blog/how-crowdstrike-analyzes-macos-malware-to-optimize-automated-detection-capabilities/

Paul-Danut Urian May 6, 2022

- Ransomware (43% of analyzed threat data), backdoors (35%) and trojans (17%) were the most popular macOS malware categories spotted by CrowdStrike researchers in 2021
- OSX.EvilQuest (ransomware), OSX.FlashBack (backdoor) and OSX.Lador (trojan) were the most prevalent threats in their respective categories
- To strengthen customer protection, CrowdStrike researchers continuously build better automated detection capabilities by analyzing and understanding how macOS threats behave

Understanding the threat landscape and how threats behave is the first step CrowdStrike researchers take toward strengthening customer protection. They based the following threat landscape analysis on internal and open source data, which revealed that in 2021 the most commonly encountered macOS malware types were ransomware (43%), backdoors (35%) and trojans (17%). Each category is powered by a different motive: ransomware by money, backdoors by remote access and trojans by data theft.

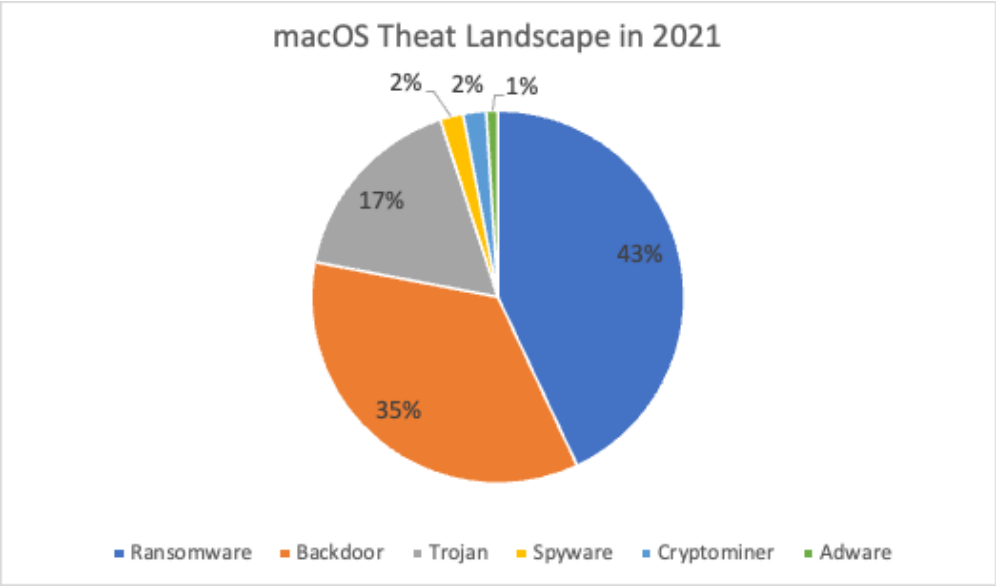

Figure 1. macOS Threat Landscape in 2021

OSX.EvilQuest was the most prevalent macOS ransomware family in 2021, accounting for 98% of ransomware in the researchers' analysis, while OSX.Flashback accounted for 31% of macOS backdoor threats and OSX.Lador accounted for 47% of macOS trojans.

Improving the CrowdStrike Falcon® platform's ability to detect macOS threats is a continuous process. CrowdStrike researchers constantly hunt, analyze and gain understanding of any macOS artifact that looks even remotely suspicious to improve CrowdStrike's automated machine learning and behavior-based protection capabilities.

The fallacies that macOS cannot be harmed by threats or is targeted by less-sophisticated malware still linger. This blog addresses some of the challenges and requirements our researchers must meet when analyzing macOS threats. The deep understanding and knowledge they gain is used both to create new features for structural parsing that augments our machine learning detection capabilities and to improve the proficiency of our behavior-based protection.

Biting Into the Apple

macOS malware research starts with the fundamentals, such as classifying macOS malware by file type; continues with the capabilities, intended targets and general behavior of malware; and ends with obstacles researchers encounter when analyzing macOS malware.

Threats that target macOS systems have the same goals as those targeting any other operating systems; they range from spying and reconnaissance to cryptocurrency mining, file encryption, remote access, and adware-related hijack and injection.

File Type Classification for macOS Threats

Malware developers often try to hide or mask file types in an attempt to trick users into executing them. File-type identification also helps in establishing the tools required in the analysis. Figure 2 offers an overview of macOS malware file types.

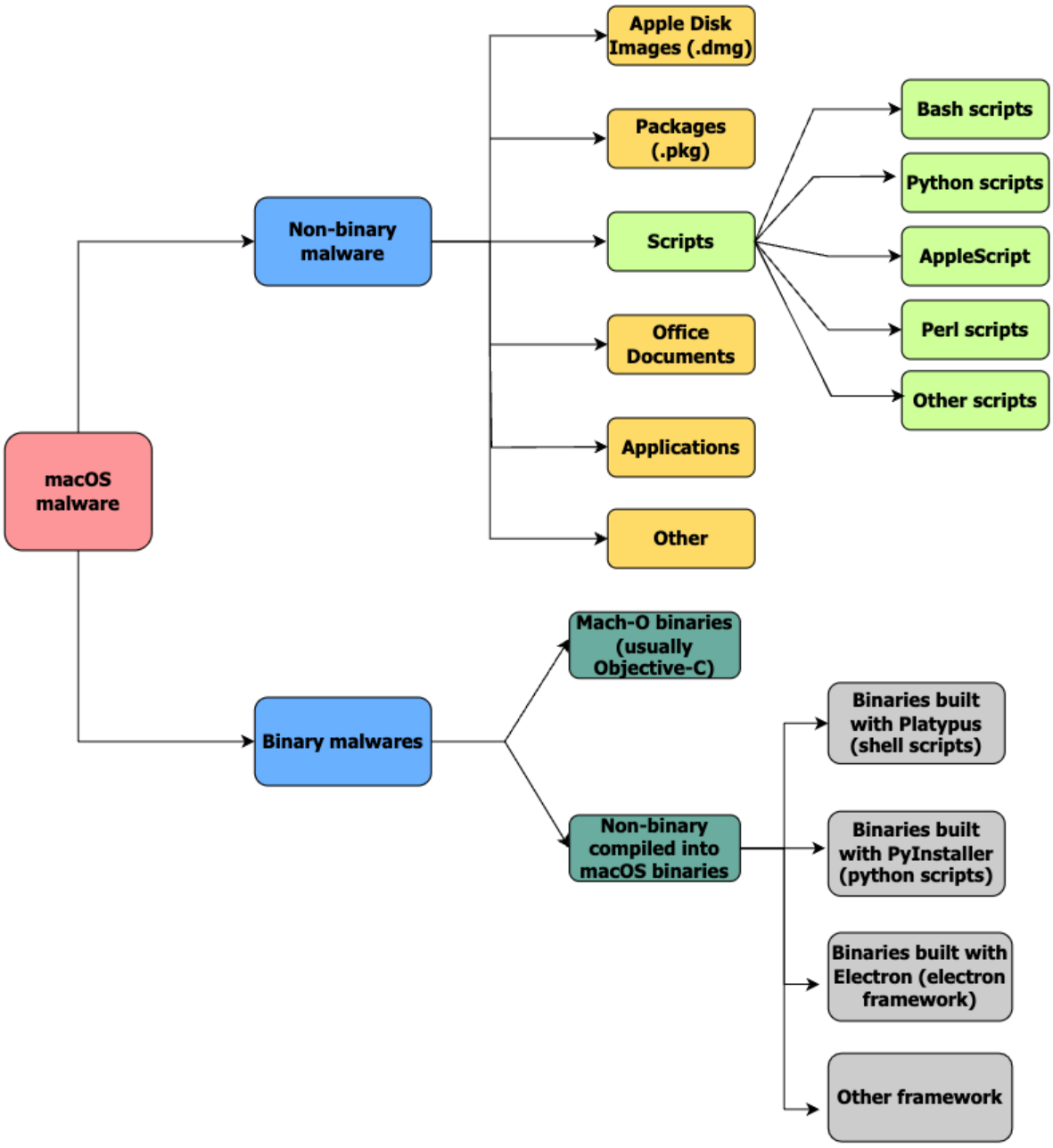

Figure 2. macOS malware by file types

Even though most malware are compiled binaries, many non-binary file types are commonly encountered while analyzing macOS malware; each has its own advantages and disadvantages for the adversaries that use them. Examples include:

- Apple Disk Images (.dmg) are favored because they're automatically mounted on execution; both OSX.EvilQuest (Figure 3) and OSX.Shlayer malware typically use this file type.
- Packages (.pkg, .mpkg) are another common file type abused by malware as they allow malware developers to define preinstall and postinstall scripts that automatically run through the installation process. For example, OSX.EvilQuest uses a malicious package — after mounting the .dmg file — that has a postinstall script that copies the malicious OSX.EvilQuest binary to */Library/mixednkey/* under the name *toolroomd*.

- AppleScripts or AppleScript variants like Run-only that are used for automating repetitive tasks are often abused by macOS threats such as OSX.OSAMiner, a popular cryptocurrency miner.

Delivery and Infection Vectors

One of the most common methods of spreading malware involves using social engineering tactics in an attempt to trick the user into manually infecting their macOS. Fake updates, fake applications, trojanized applications and tainted versions of legitimate applications are the most common methods used to trick users into installing malicious software.

For example, OSX.EvilQuest ransomware has been known to impersonate popular sound mixing applications (as seen in Figure 3), while trojans like OSX.Lador are distributed via spam emails that contain malicious add-ons, cracked applications, free programs and fake updates.

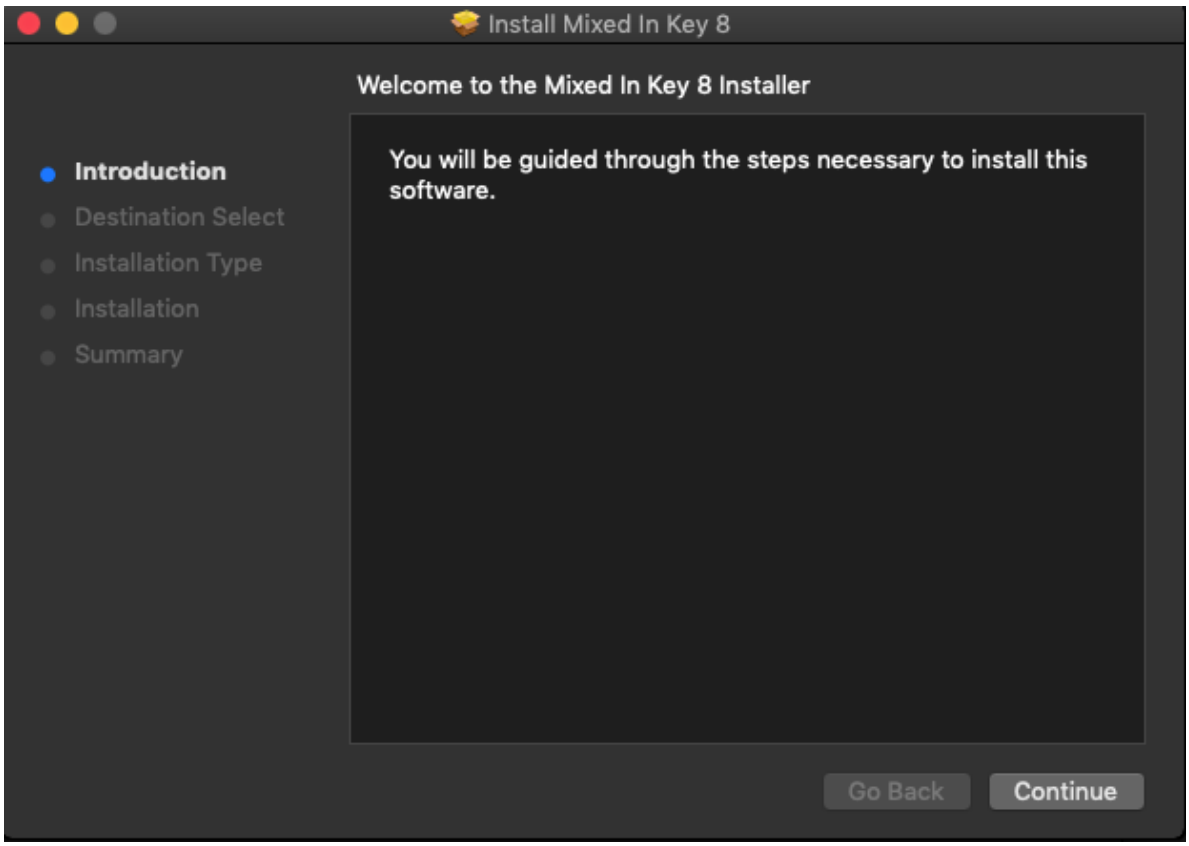

Figure 3. OSX.EvilQuest ransomware installing as fake Mixed In Key DJ application

Other malware variants, such as OSX.XCSSET, are distributed via either malicious documents or supply chain attacks targeting legitimate software development tools such as Xcode, Apple's IDE. More complex attacks use exploits in different applications or in compromised OS kernels or accounts. For example, older OSX.FlashBack backdoor variants were known to use Java exploits to compromise targets.

By understanding delivery and infection vectors, researchers can take a layered approach to security, building protection capabilities to stop breaches.

Persistence and Tactics

Most threats, including macOS malware, attempt to ensure persistence to survive system reboots. Analyzing and understanding persistence tactics enables researchers to build behavior-based detections and train automated machine learning (ML) detections.

While one of the most common persistence mechanisms involves abusing Login Items in macOS, other popular persistence tactics include abusing Launch Items, adding malware to scheduled tasks, or using cronjobs to execute tasks sometime in the future.

The hijacking of dylibs was once one of the stealthiest persistence mechanisms, especially in binaries. For example, some 2012 variants of the OSX.FlashBlack backdoor used malicious libraries injected at load time into a process via the DYLD_INSERT_LIBRARIES environment variable (i.e., at load time the dynamic loader will examine the DYLD_INSERT_LIBRARIES variable and load all specified libraries); others used the dylib hijacking technique of planting a malicious dylib for an application that tries to load dynamic libraries from multiple locations. However, Apple has long since improved security and reduced the number of use cases for abusing DYLD_INSERT_LIBRARIES.

Challenges in Malware Analysis

Most malware, regardless of the targeted platform, make analysis difficult from the start by using anti-static analysis methods, such as string-based obfuscation or code obfuscation and encryption. Scripts usually use obfuscation tools that randomize function and variable names and insert junk and useless code, while binaries make use of packers or encryption.

macOS malware also commonly uses debugger detection tactics, making analysis a challenge for researchers. Such tactics include using the sysctl API to check if the process is under debugging; calling the *ptrace* system call to prevent a debugger from attaching to the process; or even using built-in macOS commands to extract information about the machine.

On a Quest to Understand EvilQuest

Let's take a closer look at a mid-2020 OSX.EvilQuest ransomware sample and see how it implemented various anti-analysis methods to avoid virtual machines and debugging.

Upon executing, OSX.EvilQuest first checked to see if it was running in a virtual machine, in particular a sandboxed environment, by looking at the *is_virtual_mchn* function starting at address *0x0000000100007BC0.* OSX.EvilQuest performed this check by using a sleep function and calling the time function twice; the difference between the two time functions should return the time the malware used to sleep, yet because sandboxes usually patch sleep functions to quicken analysis, the differences between the two timestamps would be different and the malware would know it is running in a sandboxed environment.

Before the malware tries to ensure its persistence — as a launch daemon or a launch agent, depending on the –noroot argument passed to the binary — it implements another two anti-analysis methods. The first one (*is_debugging* function starting at address *0000000100007AA0*)

is to check if the malware is debugged, and the second one (*prevent_trace* function starting at address *0000000100007C20*) is to prevent debugging using a ptrace call with the flag PT_DENY_ATTACH.

Using the ptrace function call, OSX.EvilQuest uses different logics to make it more difficult for the analyst to spot the function call or to bypass the mechanism by patching the binary in the debugger.

CrowdStrike Protection for macOS

Continuous research into the trends and behavior of macOS malware is turned into expert input and knowledge that's used to augment CrowdStrike's automated detection capabilities and build better protection for customers.

Identifying the file type, understanding the behavior, targets and potential persistence mechanisms of possible threats, and knowing the possible obstacles an analyst may encounter in analyzing potential malware is crucial for building a solution that provides comprehensive protection and visibility against threats.

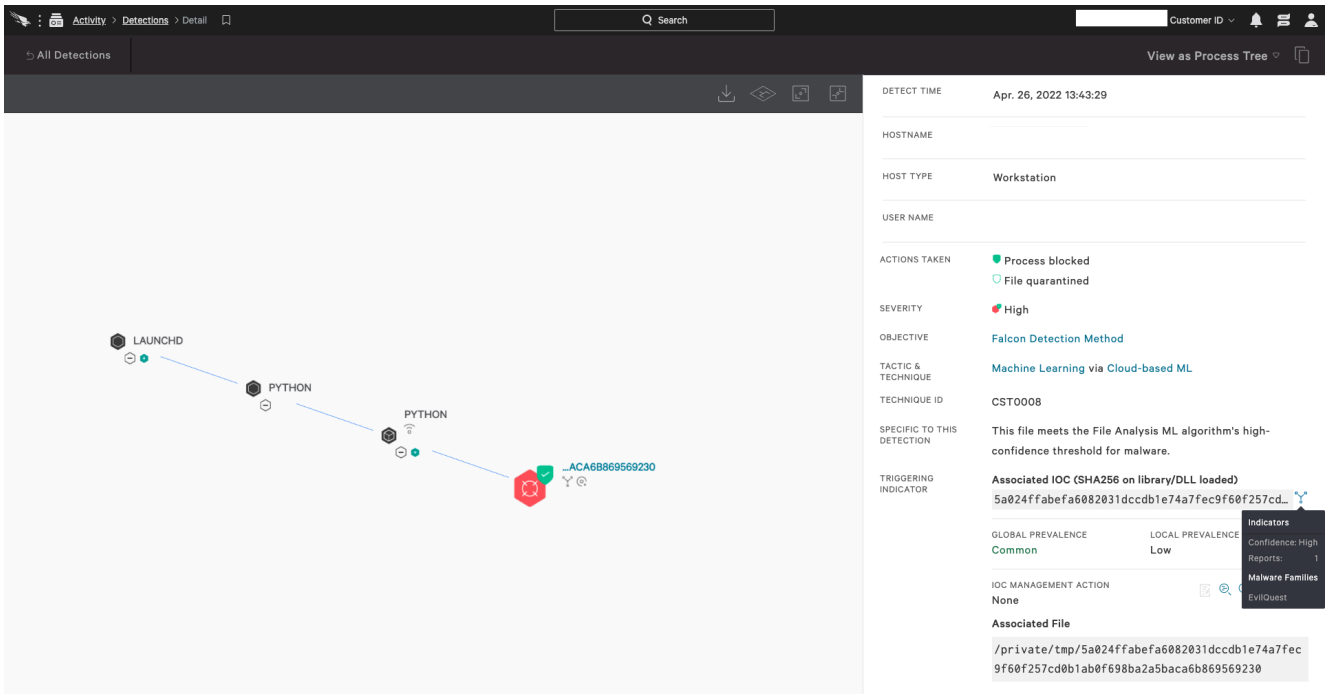

Fig 3. – CrowdStrike Falcon detection for OSX.EvilQuest malware sample (sha256: `5a024ffabefa6082031dccdb1e74a7fec9f60f257cd0b1ab0f698ba2a5baca6b`) using cloud-based machine learning (Click to enlarge)

The CrowdStrike Falcon platform protects macOS workloads using machine learning and behavior-based indicators of attack (IOAs) to defend macOS systems against malware and sophisticated threats, while delivering complete visibility and context into attacks.

Indicators of Compromise (IOCs)

File	SHA256
OSX.EvilQuest	b34738e181a6119f23e930476ae949fc0c7c4ded6efa003019fa946c4e5b287a; 5a024ffabefa6082031dccdb1e74a7fec9f60f257cd0b1ab0f698ba2a5baca6b
OSX.Shlayer	852ff1b97c1155fc28b14f5633a17de02dcace17bdc5aadf42e2f60226479eaf
OSX.Lador	30ca6a13a85ac1ea7858e8163d9c08d8bbd8ed8bc6e97498b5b02d6de042b51e; 33ee40b89ee505bced8caaa4226223a0b9622b944e790fb5a704ffe6fce3eaa6
OSX.XCSSET	a6141dfb0b6a242246d26afecfea00ed04dee24209f7d8d9bfef82042accd0f0; 6614978ab256f922d7b6dbd7cc15c6136819f4bcfb5a0fead480561f0df54ca6; ceb023a95b8ee954c31bc6aa47a8f1461e246fea939a57fc59bc4b457ccb61ff
OSX.FlashBack	8d56d09650ebc019209a788b2d2be7c7c8b865780eee53856bafceffaf71502c

Additional Resources

- *Learn more about how CrowdStrike Falcon extends protection for macOS here.*
- *Download the CrowdStrike Falcon for macOS data sheet here.*
- *Learn what others are saying about CrowdStrike — visit the CrowdStrike Industry Recognition and Technology Validation webpage.*
- *Learn about the powerful, cloud-native CrowdStrike Falcon® platform.*
- *Get a full-featured free trial of CrowdStrike Falcon Prevent™ and see for yourself how true next-gen AV performs against today's most sophisticated threats.*