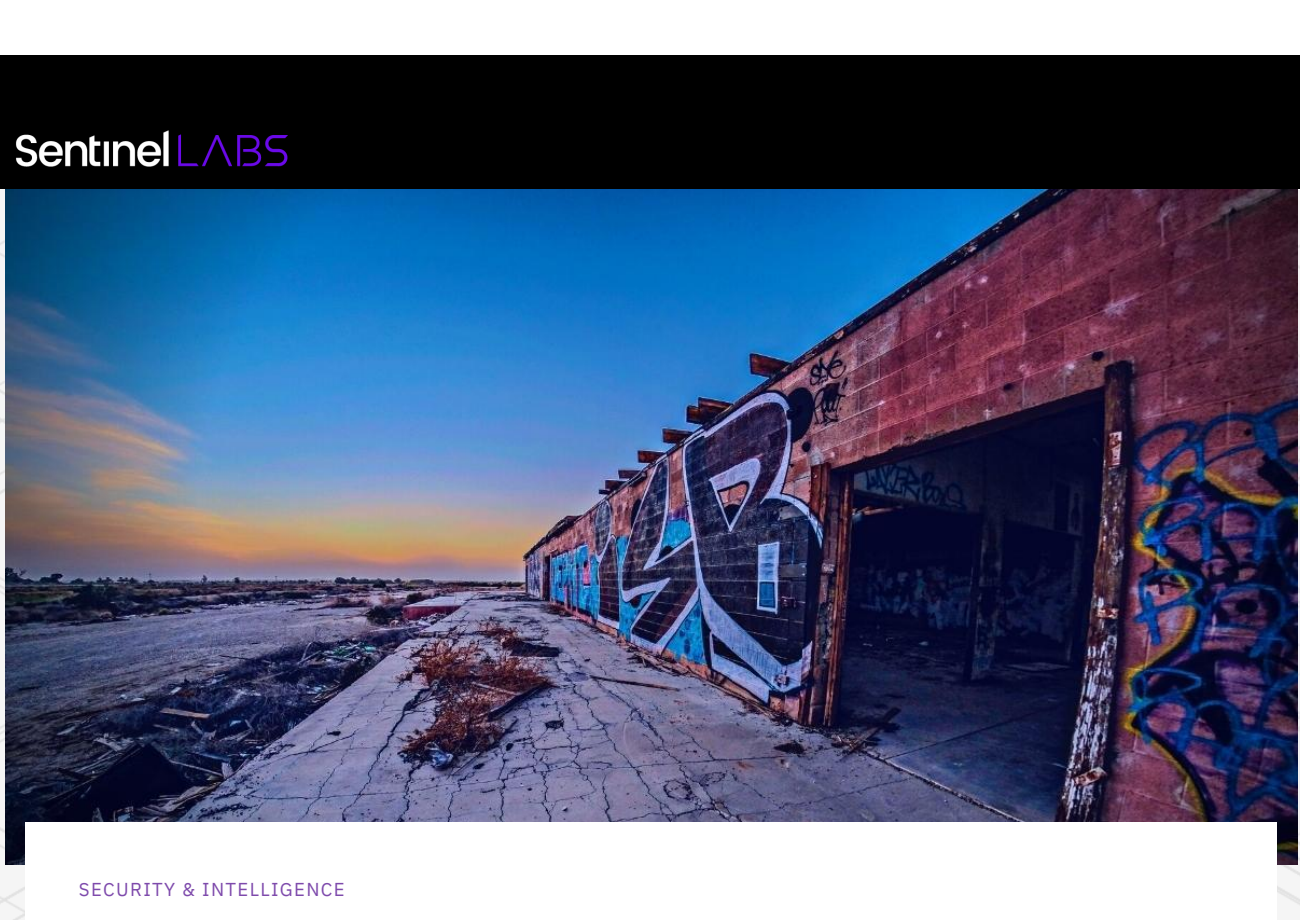
SentinelLABS



SECURITY & INTELLIGENCE

A Threat Hunter's Guide to the Mac's Most Prevalent Adware Infections 2022

PHIL STOKES / JANUARY 4, 2022

Last month, as we closed out 2021, we shared the most recent malware discoveries afflicting the Mac platform, covering spyware, targeted attacks on developers and activists, cryptocurrency theft and cryptomining. As worrisome as those are, the bulk of infections affecting Mac users in and out of enterprise settings revolve around adware.

Once little more than a minor nuisance, adware on all platforms has taken a darker turn in recent years, often emulating malware TTPs and regularly surpassing a lot of malware families in sophistication and rapid evolution. What's driven these developments is simple: adware makes a lot of money. Adware also harvests a lot of data from infections which can be sold off to other actors.

Most importantly from a security team's point of view, however, is that adware infections set up hidden, persistent executables, engage in device and environmental fingerprinting, use anti-removal, anti-analysis and detection avoidance techniques, and reach out to unknown URLs to deliver custom payloads, typically without the knowledge or informed consent of the user or, in the enterprise case, the device owner.

For all these reasons, knowing how to detect an adware infection is no less important than any other malware infection. In this post, we shine a light on the most prevalent adware families affecting the Mac platform over the last 3 months and describe the typical infection patterns for each.

Cataloguing and sharing what we know in this way has two benefits. It enables defenders to improve their immediate detection responses in the short-term, and it represents a cost to threat actors in the mid-term, who are forced to invest in retooling and rethinking their approach.

1. Adload System_Service

Adload has probably been around since 2016 and is the most common family we see in live infections today. We have discussed specific Adload campaigns a few times in the past, here and here and we advise readers to review those posts for earlier Adload indicators. We include in this entry only those that we have not detailed before or which we saw in the last quarter of 2021 and early 2022.

The System_Service campaign remains the most active of current variants that we observe.

These follow a determinate pattern:

Hunting Regex

```
.*/Library\/Application Support\/\.[0-9]{19,}\/(Services|System)/com\.\w+\.(service|system)\/\w+\.(se
```

Examples

```
~/Library/Application Support/.15314127506195013446/Services/com.SkilledObject.service/SkilledObject.s
~/Library/Application Support/.16951906660859967924/Services/com.SkilledUnit.service/SkilledUnit.servi
/Library/Application Support/.2301650498054541179/System/com.ElementaryType.system/ElementaryType.syst
```

A similar, older but still active pattern does not contain the `System` or `Service` terms and does away with the hidden parent folders.

Examples

```
~/Library/Application Support/com.AdvancedRecord/AdvancedRecord
~/Library/Application Support/com.NetDataSearch/NetDataSearch
```

2. Adload Go Variant (Rload/Lador)

An increasingly common pattern we are seeing throughout late 2021 involves Adload variants written in either Go (*aka* Rload/Lador) or Kotlin. The Go variants currently drop a payload with the following file path pattern:

Hunting Regex

```
Library/Application\ Support/com\.\d{19,}\.[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12
```

Examples

```
/Library/Application Support/com.11592658482052096796.D18B18A4-7ED8-434B-B3A1-6F109CA25EB5/_141391364
/Library/Application Support/com.2718493167946217159.4E41C598-9C07-4446-96A4-CE22A41B6BF1/_5214250257
```

Note that the executable file name only contains numerals. Although the underscore prefix is present more often than not in instances we observed, there are cases of this pattern where the underscore is not present.

3. Adload Kotlin Variant

The Kotlin variant of Adload uses a different but still quite distinctive pattern:

Hunting Regex

```
/Library/Application\ Support/\.[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}/\.[0-9a-
```

Examples

```
~/Library/Application Support/.6BA27F8C-697E-4A94-B97C-A0E6AC13F210/.54DA7A17-F1DC-477C-BB31-485FEAC2
~/Library/Application Support/.399CEC38-2BB0-4263-8232-4CCAC933C9E2/.CB273BA5-3138-4274-8C61-B2CBA0F1
```

The Kotlin variants also reach out to a server with the pattern:

Hunting Regex

```
http://m\..*\.com/ktl/uba\?ral=
```

The wildcard part is consistently made up of two-word patterns that mimic the names seen in the Sytem_Service and earlier Adload campaigns.

Examples

```
http://m.originalrotator.com/ktl/uba?ral=
http://m.bufferrecord.com/ktl/uba?ral=
```

4. Other Adload Variants

A pattern seen across a number of different variants involves the Adload installer dropping a Mach-O executable in the `/tmp/` directory with a filename prefixed with the letters "php" followed by 6 alphanumeric characters (a similar pattern is used by MaxOfferDeal/Genieo, which we discuss below)

Hunting Regex

```
/tmp/php\w{6}
```

Examples

```
/tmp/phpO9PLui
/tmp/phpwzmOLI
/tmp/phpZOfJhD
/tmp/php6wmXLq
```

A much older pattern that we still see occasionally appearing in live infections has the form:

Hunting Regex

```
/Library/Application\ Support/com\..*Lookup.*Lookup.*
/Library/Application\ Support/com\..*SearchDaemon.*Search.*
```

Examples

```
/Library/Application Support/com.OdysseusLookupDaemon/OdysseusLookup
/Library/Application Support/com.ExpertLookupEngineDaemon/ExpertLookupEngine
/Library/Application Support/com.ApolloSearchDaemon/ApolloSearch
/Library/Application Support/com.GlobalToolboxSearchDaemon/GlobalToolboxSearch
```

There are other minor variants on this naming convention that will be readily recognizable once you are familiar with the above patterns. For more information on this pattern see here.

5. Bundlore, Shlayer, and ZShlayer

Bundlore has been around since at least 2014 and, after Adload, is the most prevalent family we see in live infections throughout 2021 and into the beginning of 2022.

Bundlore payloads are typically dropped by a Shlayer or ZShlayer DMG installer. Often the Shlayer or ZShlayer installer will have one of the following file patterns:

Hunting Regex

```
/Volumes/(Player|Install)/(Player|Install)\.app/Contents/MacOS/\w+
```

Examples

```
/Volumes/Player/Player.app/Contents/MacOS/overprints
/Volumes/Install/Install.app/Contents/MacOS/isolated
```

Note that in the case of the "Install" pattern, the "I" can appear both as upper and lowercase. We see the "Player" version more often than the "Install" one.

The first-stage Bundlore payload will be dropped in a random folder created in the `/tmp/` directory with a corresponding name:

Hunting Regex

```
/tmp/.{9,}/(player|install)\.app/Contents/MacOS/\w+
```

Examples

```
/tmp/0yuEUAw74/Player.app/Contents/MacOS/overprints
/tmp/zWMp9EpUT/Install.app/Contents/MacOS/isolated
```

Two much older DMG patterns associated with the original Shlayer DMGs, but which we only see on rare occasions now are:

Hunting Regex

```
/Volumes/Player/Player.command
/Volumes/install/.hidden/install.command
```

6. Pirrit

Pirrit is a macOS malware family that was first seen in 2016 and remained relatively active throughout 2017 but had all but disappeared until November 2021. Since then, Pirrit has seen a new burst of activity.

In common with Bundlore, Pirrit will typically drop via a user executed DMG, although the disk image name and application name tend to be as follows:

```
/Volumes/Install Flash Player/Install Flash Player
```

Pirrit's first stage payload drops in the Darwin_User_Temp_Dir (rather than the system `/tmp` dir) and uses an 8 character random directory name with either `tmp` or `Installer` as a prefix.

Hunting Regex

```
/private/var/folders/.*./(tmp|installer)\..{8}$
```

Examples

```
/private/var/folders/7d/7skpstwd7qnctfwpwp7225xw0000gn/T/tmp.kfiBqqFO
/private/var/folders/7d/7skpstwd7qnctfwpwp7225xw0000gn/T/tmp.jNuFmFOE
/private/var/folders/7d/7skpstwd7qnctfwpwp7225xw0000gn/T/Installer.bwlOVmDo
```

The next stage of the infection usually drops in the Application Support folder with a random name:

Hunting Regex

```
~/Library/Application\ Support/com\.[a-z]*/[a-z]*
```

Examples

```
~/Library/Application Support/com.described/described
~/Library/Application Support/com.memberd/memberd
~/Library/Application Support/com.Searchie/Searchie
```

A further component is written to a folder in the User's Library folder or local domain Library folder (depending on available permissions) and contains an application of the same name:

Hunting Regex

```
.*/Library/(.*)/\1.app/Contents/MacOS/\1
```

Examples

```
~/Library/CheckTime/CheckTime.app/Contents/MacOS/CheckTime
~/Library/SysUpdater/SysUpdater.app/Contents/MacOS/SysUpdater
~/Library/UpdateData/UpdateData.app/Contents/MacOS/UpdateData
/Library/TimeCheckDaemon/TimeCheckDaemon.app/Contents/MacOS/TimeCheckDaemon
```

This variant of Pirrit appears to be rapidly evolving. A recent sample installed this application inside the Application Support folder:

```
~/Library/Application Support/com.SearchZen/SearchZen.app/Contents/MacOS/SearchZen
```

Depending on permissions when the infection runs, Pirrit may also install some components into `/var/root/`.

Behaviorally, Pirrit is a good example of adware that attempts evasion techniques that only become apparent upon execution.

VM Detection/Evasion Behavior

```
/usr/bin/grep grep -q VirtualBox\|Oracle\|VMware\|Parallels
```

7. MaxOfferDeal / Genieo

Genieo is another long-standing, common macOS malware family that goes in and out of periods of activity. Late 2021 saw some new variants which we continue to track but we have seen little activity. The most prevalent one on our radar uses a persistent LaunchAgent with the following pattern for its program argument:

Example

```
~/Library/Application Support/.gettime/GetTime
```

Interestingly, the persistence file is copied from a `/tmp/` file that uses a similar naming pattern to Adload, namely "php" followed by 6 characters. This may be coincidence or deliberate, and either way may have caused some vendors to identify one as the other.

The same regex we showed for Adload Mach-Os above, however, will also find these `.plist` files.

Examples

```
/tmp/phpEFab3r
/tmp/phpEkFfeu
/tmp/phpkWZyll
/tmp/phpWLcS4s
/tmp/phpWagty2
/tmp/phpHGpcfX
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>KeepAlive</key>
    <dict>
        <key>SuccessfulExit</key>
        <false/>
    </dict>
    <key>Label</key>
    <string>Label</string>
    <key>ProgramArguments</key>
    <array>
        <string>/Users/User/Library/Application Support/.gettime/GetTime</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>StandardErrorPath</key>
    <string>/dev/null</string>
    <key>StandardOutPath</key>
    <string>/dev/null</string>
</dict>
</plist>
```

However, in the Adload case, these files are always Mach-Os, whereas in the MaxOfferDeal/Genieo case they are always property lists. We see no other indicators or similarities between the executable and known Adload variants.

8. MMInstall/MacUpdater

MMInstall has been around since at least early 2018 and typically installs a LaunchAgent with a program argument with variety of names like "MyShopCoupon", "CouponSmart" and similar. Older forms typically had an executable with the name "mm-install-macos" but we haven't seen those for some time.

Apple recently updated their XProtect malware signatures for a newer version of this adware threat that appears to have been active during the middle of 2021. The following domains are still currently active:

Hunting regex

```
yougotupdated.com
liveforlifetoday.com
```

Examples

```
http://tracker.yougotupdated.com
http://installer.yougotupdated.com/offers/macUpdater/updater
https://installer.yougotupdated.com/offers/macUpdater/updater
http://inst.liveforlifetoday.com/offers/macUpdater/updater
```

The only known installer pattern we have seen to date is as follows.

```
/Volumes/search/Search.app/Contents/MacOS/Search
```

Conclusion

Most adware arrives in the form of trojanized applications that users are persuaded to attempt to install. Free content, cracked apps, and "special deals" are typical vectors. The fact that some – although by no means all – adware installers make a show of obtaining user consent doesn't ameliorate the situation: in the cases where that does happen, the consent mechanism is itself often misleading or aggressive.

Regardless of how it is installed, unless the user has permission from the device owner, then adware will almost certainly be unwanted on company-owned devices. Given the aggressive behavior of adware, it should be of no less concern than any other type of malware.

We hope the information in this post will aid security teams to identify and remove adware infections on Mac devices. We would also encourage analysts to become familiar with other useful behavioral indicators associated with a wide range of macOS threats including adware families that can be found here.

ADWARE

PHIL STOKES

Phil Stokes is a Threat Researcher at SentinelOne, specializing in macOS threat intelligence, platform vulnerabilities and malware analysis. He began his journey into macOS security as a software developer, creating end user troubleshooting and security tools just at the time when macOS adware and commodity malware first began appearing on the platform. Phil has spent the last 7 years closely following the development of macOS threats as well as researching software and OS vulnerabilities.