Intro to Mac Malware

By Ryan Nolette
Security Operations Lead
Senior Security Engineer
Senior Threat Researcher
Incident Response Consultant
Carbon Black

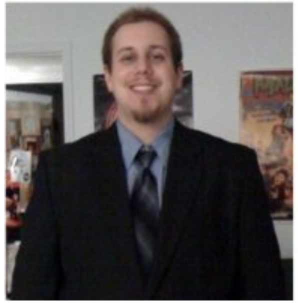
CARBON BLACK
ARM YOUR ENDPOINTS

Agenda

- $whoami
- Overview of Mac malware
  - Infection mechanisms
  - Persistence mechanisms
- Self-defense
- Features
- Bypasses
- Defenses
- Using CBER to detect wirelurker
  - IOCs from detonated sample
  - IOCs from opensource intelligence
  - Translate IOCs to watchlists
- Using CBEP to block wirelurker
  - Example rule from IOCs collected
- Conclusion/Recap/Questions

CARBON BLACK
ARM YOUR ENDPOINTS

$whoami



- **My name is Ryan Nolette**
  - I am currently the **Senior Security Engineer** at Carbon Black
    - **Act as Senior Security Architect for Carbon Black**
  - I am a 10+ year veteran of IT, Incident Response, Threat Intelligence, and Computer Forensics
  - Content I've created
    - https://github.com/sonofagl1tch
    - https://www.carbonblack.com/author/ryan-nolette/
  - **Responsibilities**:
    - Monitor Endpoint Events, Network Based Events, and Physical Security Events
    - User Education and Outreach
    - IT Oversight and Assistance
    - Security Oversight of Enterprise Projects
    - Incident Response
    - System Forensics
    - Vulnerability Scanning
    - Threat Research
    - ETC

CARBON BLACK
ARM YOUR ENDPOINTS

The current state of OS X malware

CARBON BLACK
ARM YOUR ENDPOINTS

Overview

- Macs now make up ~30% of systems in the enterprise
- “It doesn’t get PC viruses. A Mac isn’t susceptible to the thousands of viruses plaguing Windows-based computers.” -apple.com (2012)
- Mac Malware timeline:
  - ‘first’ virus (elk cloner) infected apple II’s
  - "[2014] nearly 1000 unique attacks on Macs; 25 major families" –Kaspersky

The current state of OS X malware

- Infection mechanism
  - Trojans
  - Phishing
  - old bugs
  - occasionally exploits
- Persistence
  - well known techniques
  - majority: launch items
- Self-defense
  - minimal obfuscation
  - trivial to detect & remove
- Stealth
  - 'hide' in plain site
  - stand-alone executables
- Features
  - inelegantly implemented
  - suffice for the job

CARBON BLACK
ARM YOUR ENDPOINTS

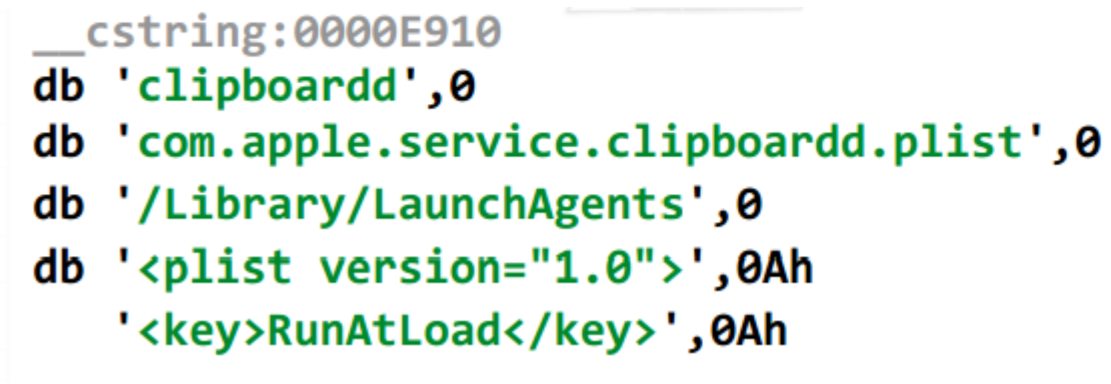
Infection Mechanisms

- Same as PC
- Primary attack vectors are email, drive by downloads, and infected binaries.

- Mac has the unique attack vector of a closed ecosystem which implies a false sense of trust

OSX/XSLCMD

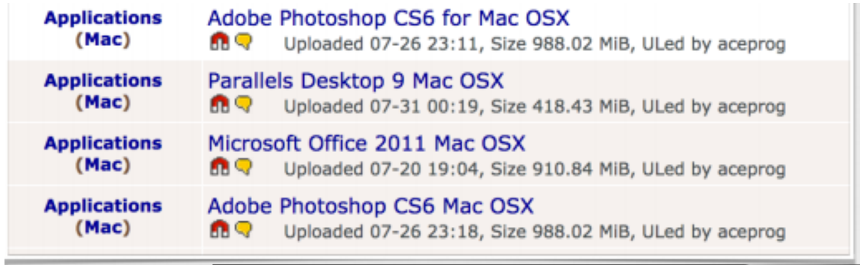
- provides reverse shell, keylogging, & screen capture
- “a previously unknown variant of the APT backdoor XSLCmd which is designed to compromise Apple OS X systems”



```
__cstring:0000E910
db 'clipboardd',0
db 'com.apple.service.clipboardd.plist',0
db '/Library/LaunchAgents',0
db '<plist version="1.0">',0Ah
   '<key>RunAtLoad</key>',0Ah
```

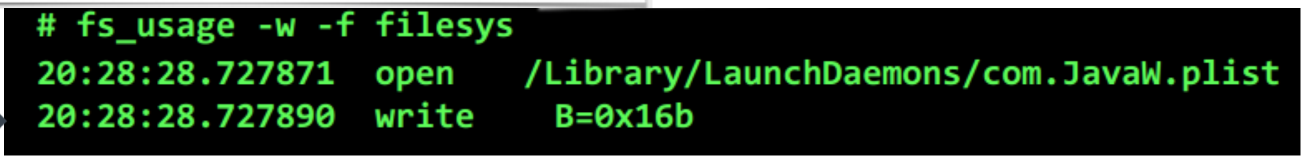
OSX/IWORM

- 'standard' backdoor, providing survey, download/execute, etc.
  - https://www.blackhat.com/docs/us-15/materials/us-15-Wardle-Writing-Bad-A-Malware-For-OS-X.pdf



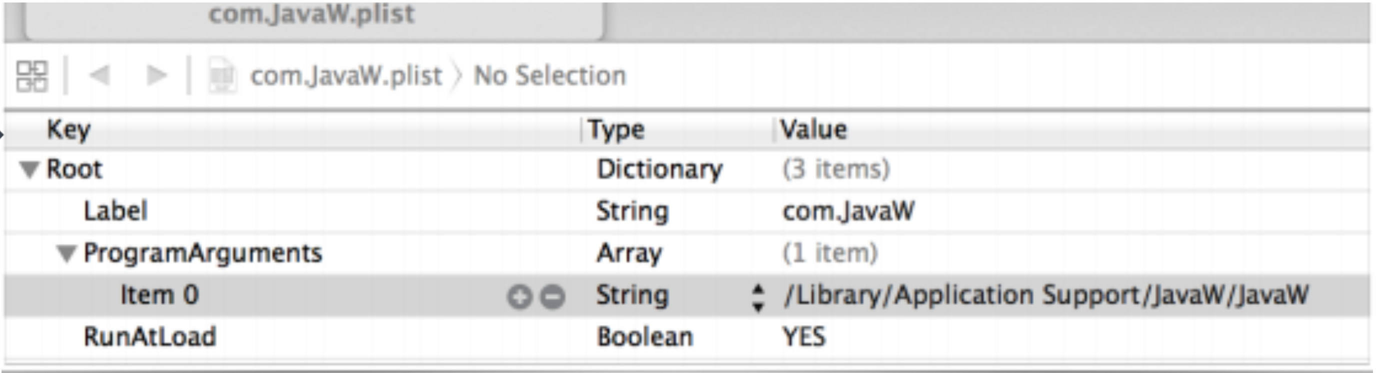
Applications (Mac)	Adobe Photoshop CS6 for Mac OSX Uploaded 07-26 23:11, Size 988.02 MiB, ULed by aceprog
Applications (Mac)	Parallels Desktop 9 Mac OSX Uploaded 07-31 00:19, Size 418.43 MiB, ULed by aceprog
Applications (Mac)	Microsoft Office 2011 Mac OSX Uploaded 07-20 19:04, Size 910.84 MiB, ULed by aceprog
Applications (Mac)	Adobe Photoshop CS6 Mac OSX Uploaded 07-26 23:18, Size 988.02 MiB, ULed by aceprog

Infection Vector:
Torrents



```
# fs_usage -w -f filesys
20:28:28.727871  open   /Library/LaunchDaemons/com.JavaW.plist
20:28:28.727890  write    B=0x16b
```

Persistence Mechanism:
Launch daemon

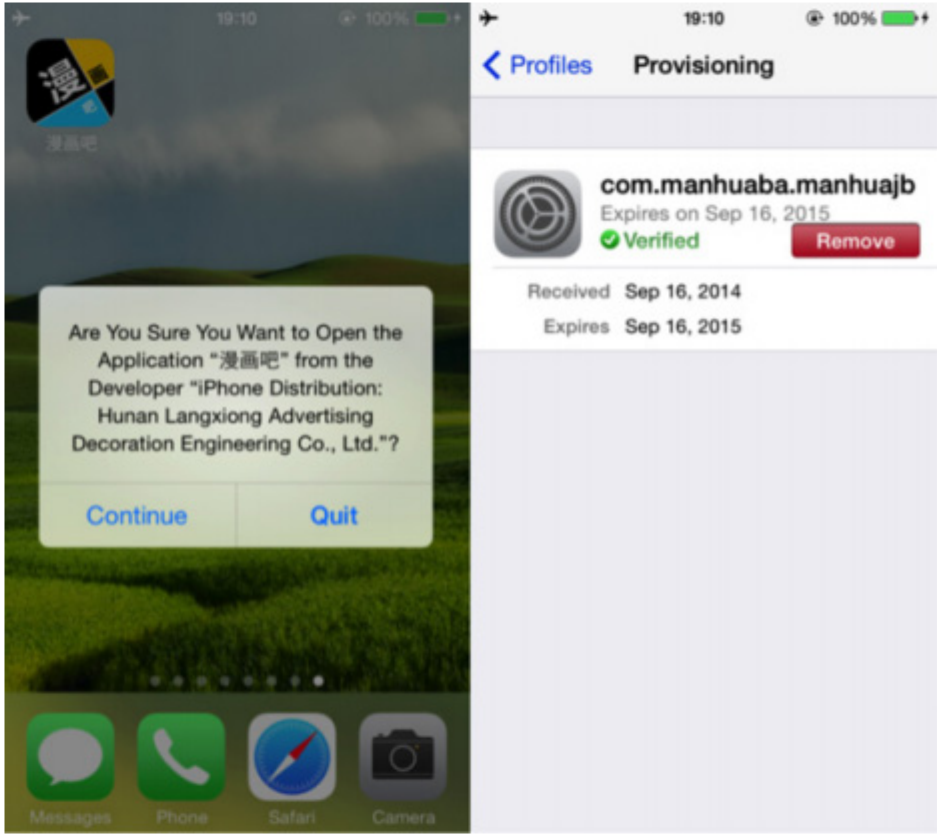


com.JavaW.plist

com.JavaW.plist 〉 No Selection

Key	Type	Value
▼ Root	Dictionary	(3 items)
Label	String	com.JavaW
▼ ProgramArguments	Array	(1 item)
Item 0	String	/Library/Application Support/JavaW/JavaW
RunAtLoad	Boolean	YES

OSX/WIRELURKER


19:10 100%

Are You Sure You Want to Open the Application "漫画吧" from the Developer "iPhone Distribution: Hunan Langxiong Advertising Decoration Engineering Co., Ltd."?

Continue | Quit

19:10 100%

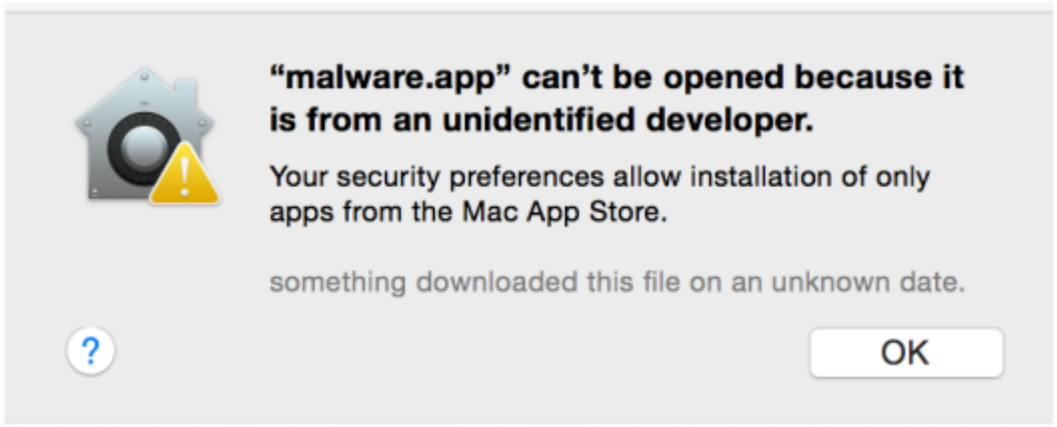
Profiles | Provisioning

com.manhuaba.manhuajb
Expires on Sep 16, 2015
Verified
Remove

Received	Sep 16, 2014
Expires	Sep 16, 2015

- an iOS infector (via USB)
- "a collection of scripts, plists, & binaries all duct-taped together... making it easy to detect." -j zdziarski
- More details later

Gatekeeper


“malware.app” can’t be opened because it is from an unidentified developer.

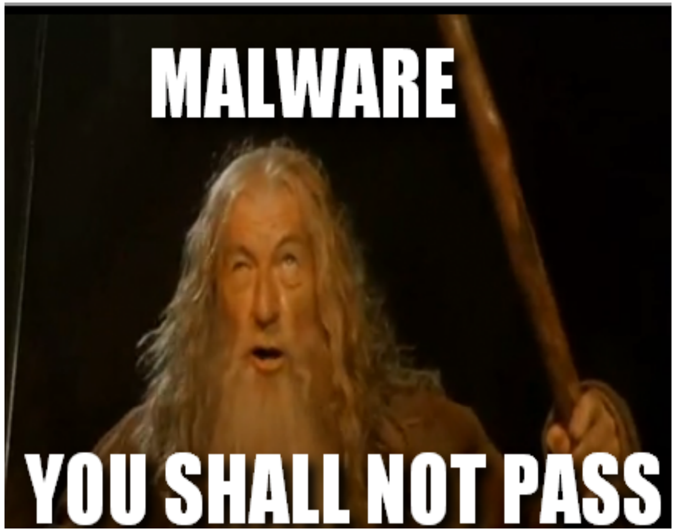
Your security preferences allow installation of only apps from the Mac App Store.

something downloaded this file on an unknown date.

- Bypassing gatekeeper is very easy
- interesting from a defense perspective

- Gatekeeper blocking untrusted code
- somewhat effective, but most users should be ok.

=

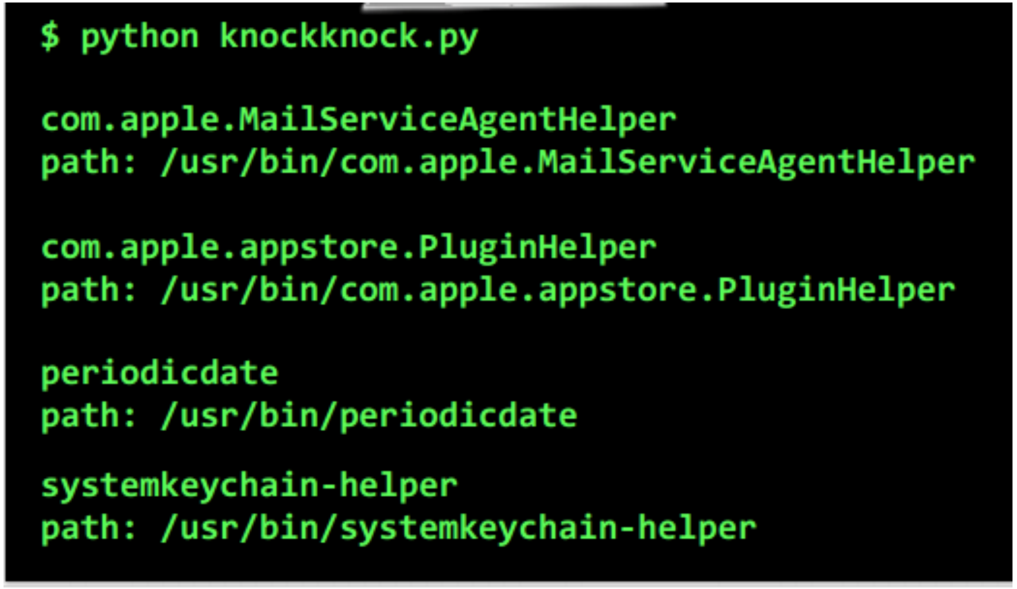


CARBON BLACK
ARM YOUR ENDPOINTS

Persistence

CARBON BLACK
ARM YOUR ENDPOINTS

Persistence



```
$ python knockknock.py

com.apple.MailServiceAgentHelper
path: /usr/bin/com.apple.MailServiceAgentHelper

com.apple.appstore.PluginHelper
path: /usr/bin/com.apple.appstore.PluginHelper

periodicdate
path: /usr/bin/periodicdate

systemkeychain-helper
path: /usr/bin/systemkeychain-helper
```

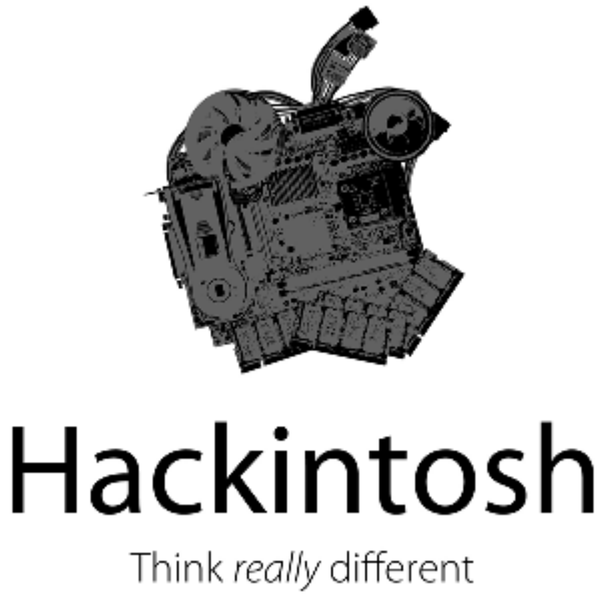
Wirelurker Launch Items

- The issue with launch items and login items is that they are easily visible, easy to detect, and are well known features.
- Consider the Mac equivalent to the run and runonce registry keys on windows or cronjobs in *nix

- Current methods are not advanced
- 2 main persistence mechanisms
  - Launch items
    - Custom start items managed by launchd
  - Login items
    - Start when the user logs into their session
- Alternative methods – old school
  - Cronjobs
    - Similar in function to launch items and can be customized to run every few seconds to every few years
  - Bashrc modifications
    - Similar to login items but only executes at the initiation stage of a CLI session

Persistence

- BINARY INFECTION
  - fairly stealthy, self-contained, difficult to detect, and difficult to disinfect
  - OSX OS loader verifies all signatures
  - Can inject legitimate signature into malware to get around the loader
- DYLIB HIJACKING
  - Easy to do
  - Spawns no new processes
  - No binary or OS modifications required
  - Abuses legitimate functionality of OSX
- Plugin Persistence
  - Abusing system plugins
  - Spawns no new processes
  - Abuses legitimate functionality of OSX



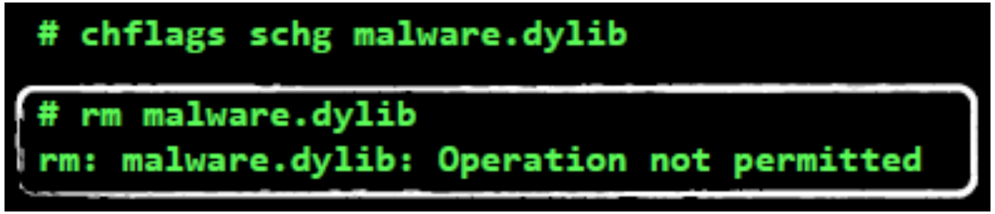
Mac malware SELF-DEFENSE

- Currently, essentially non-existent
- Poor crypto implementations
- Tries to hide in plain sight
- Easy to find
- Easy to analyze
- Easy to disinfect

CARBON BLACK
ARM YOUR ENDPOINTS

Other possible self defense methods

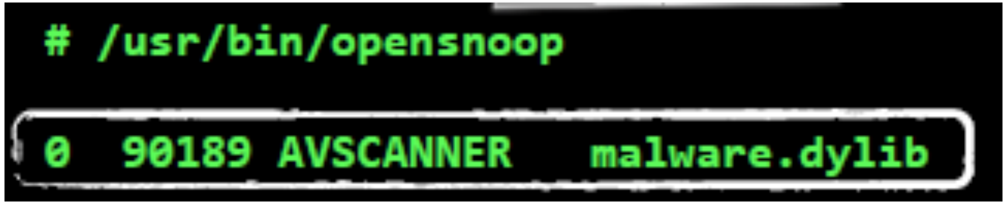
- I haven't seen these in the wild yet but they will be soon enough
- Prevent deletion
  - The schg flag can only be unset in single-user mode



```
# chflags schg malware.dylib

# rm malware.dylib
rm: malware.dylib: Operation not permitted
```

- self-monitoring
  - detect local access (dtrace)
  - Detect detections
    - Uploads to virustotal
    - Google adwords



```
# /usr/bin/opensnoop

0   90189 AVSCANNER    malware.dylib
```

Defending against Wirelurker

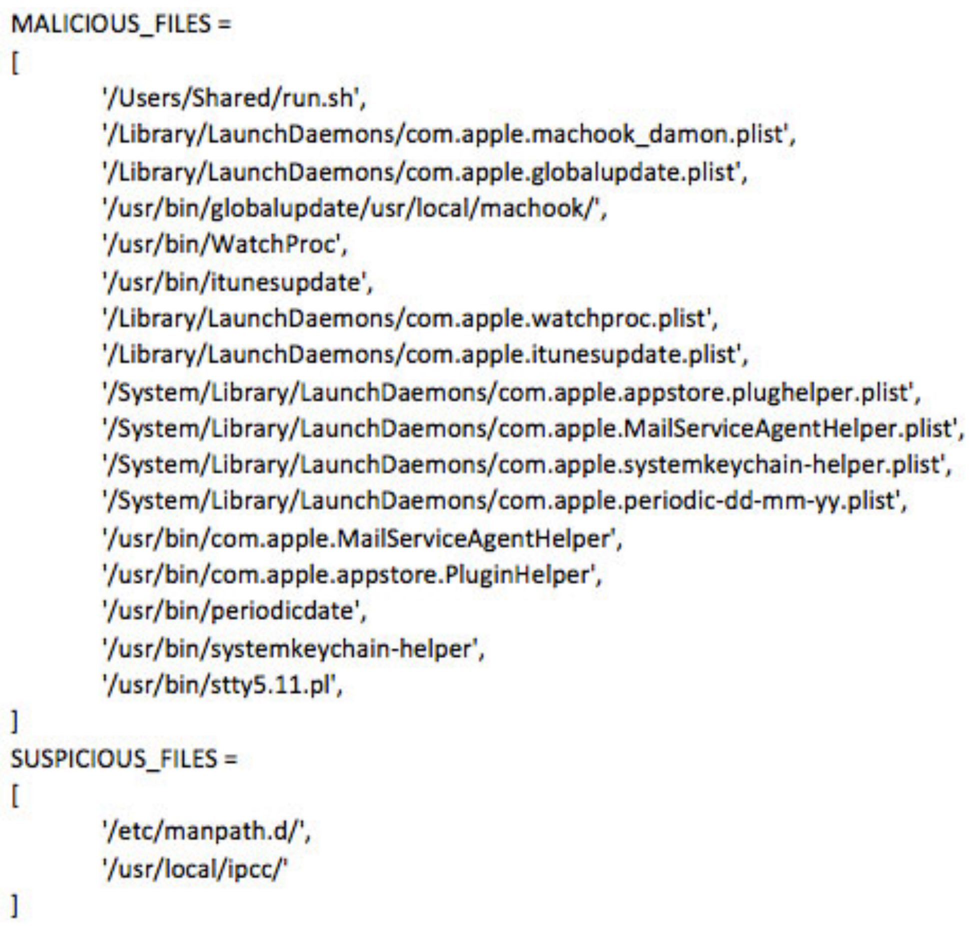
CARBON BLACK
ARM YOUR ENDPOINTS

Recon, research, repeat: gathering data for your watchlist

***NOTE**: assumed you read the WireLurker report, wirelurker detector scripts, a few more blogs on the malware, and have a decent understanding of it.*

- From this research, you should have generated a list of known artifacts about the malware (indicators).
- My list is as follows:
  - Detector script found online
  - IOC’s from blogs
  - IOC’s from manual detonation
  - IOC’s from reverse engineering sample

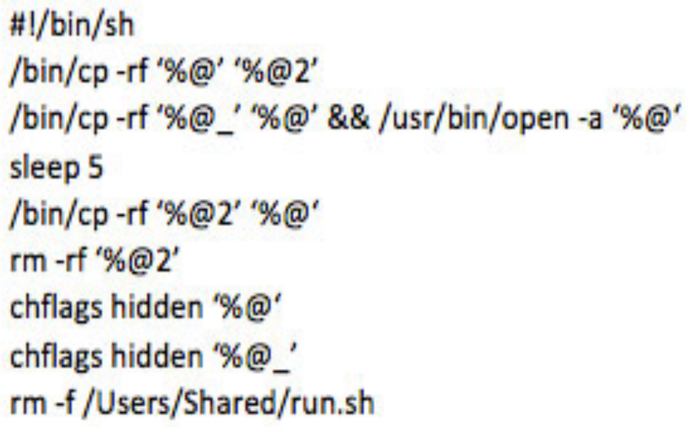
Taken from detector script:



```
MALICIOUS_FILES =
[
        '/Users/Shared/run.sh',
        '/Library/LaunchDaemons/com.apple.machook_damon.plist',
        '/Library/LaunchDaemons/com.apple.globalupdate.plist',
        '/usr/bin/globalupdate/usr/local/machook/',
        '/usr/bin/WatchProc',
        '/usr/bin/itunesupdate',
        '/Library/LaunchDaemons/com.apple.watchproc.plist',
        '/Library/LaunchDaemons/com.apple.itunesupdate.plist',
        '/System/Library/LaunchDaemons/com.apple.appstore.plughelper.plist',
        '/System/Library/LaunchDaemons/com.apple.MailServiceAgentHelper.plist',
        '/System/Library/LaunchDaemons/com.apple.systemkeychain-helper.plist',
        '/System/Library/LaunchDaemons/com.apple.periodic-dd-mm-yy.plist',
        '/usr/bin/com.apple.MailServiceAgentHelper',
        '/usr/bin/com.apple.appstore.PluginHelper',
        '/usr/bin/periodicdate',
        '/usr/bin/systemkeychain-helper',
        '/usr/bin/stty5.11.pl',
]
SUSPICIOUS_FILES =
[
        '/etc/manpath.d/',
        '/usr/local/ipcc/'
]
```

IOCs found through various blogs and forums:

1. Immediately following execution of malicious file
   1. append an underscore to the original bundle executable name
   2. then copy its malicious loader into the bundle to replace the original executable.
2. Adds a shell script, “start.sh”, and archive, “FontMap1.cfg”, to the “Contents/Resources” folder of the bundle.
   1. To me, that means that we should look inside all subdirectories in /applications for start.sh and Fontmap1.cfg.
3. The “hidden” flag is set for these files.
   1. This flag is an Apple-specified file property defined at “/usr/include/sys/stat.h” as “UF_HIDDEN.”
   2. With this flag set, a standard user won’t see the files in the Finder, but can still view them through the Terminal.
   3. Look for change flag on files in /Applications.
4. one of the scripts that the malware drops
   1. loader drops an embedded script file “/Users/Shared/run.sh”.
5. Other IOC:
   1. Known network traffic
   2. com\mac\update.zip
   3. *\mac\getsoft.php



```
#!/bin/sh
/bin/cp -rf '%@' '%@2'
/bin/cp -rf '%@_' '%@' && /usr/bin/open -a '%@'
sleep 5
/bin/cp -rf '%@2' '%@'
rm -rf '%@2'
chflags hidden '%@'
chflags hidden '%@_'
rm -f /Users/Shared/run.sh
```

Breaking your findings down into watchlists

- Now that we have all of this information, we need to break it down in different ways. I suggest one of two ways:
  1. File system artifacts, registry artifacts, memory artifacts, and network artifacts
  2. High confidence, medium confidence, low confidence

- Both of these approaches have their pros and cons and should be chosen based on your findings and your confidence in those finding to not produce false positives.

Creating the watchlists

Watchlist 1: High Confidence
This Watchlist will contain:

- All file paths take from the detector script
- All registry values
- All other static values I can find

Watchlist 2: Medium Confidence
This Watchlist will contain:

- Network traffic
- Other traffic that could have potential false positive events

Watchlist 3: Low Confidence
This Watchlist will contain:

- Any items that will most likely produce false positives

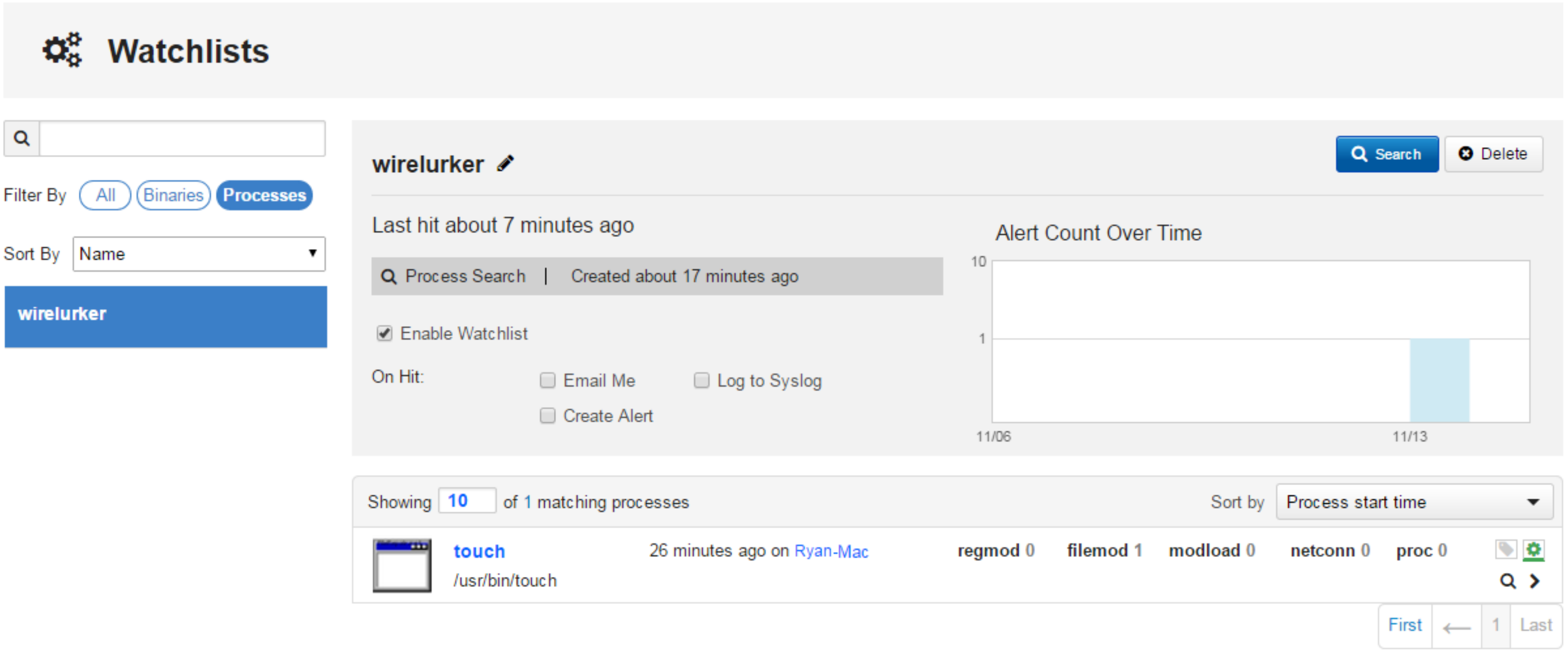
Example Carbon Black Watchlists:

Watchlist 1:
filemod:Users/Shared/run.sh OR
filemod:Library/LaunchDaemons/com.apple.machook_damon.plist OR
filemod:Library/LaunchDaemons/com.apple.globalupdate.plist OR
filemod:usr/bin/globalupdate/usr/local/machook/ OR
filemod:usr/bin/WatchProc OR
filemod:usr/bin/itunesupdate OR
filemod:Library/LaunchDaemons/com.apple.watchproc.plist OR
filemod:Library/LaunchDaemons/com.apple.itunesupdate.plist OR
filemod:System/Library/LaunchDaemons/com.apple.appstore.plughelper.plist OR
filemod:System/Library/LaunchDaemons/com.apple.MailServiceAgentHelper.plist OR
filemod:System/Library/LaunchDaemons/com.apple.systemkeychain-helper.plist OR
filemod:System/Library/LaunchDaemons/com.apple.periodic-dd-mm-yy.plist OR
filemod:usr/bin/com.apple.MailServiceAgentHelper OR
filemod:usr/bin/com.apple.appstore.PluginHelper OR
filemod:usr/bin/periodicdate OR
filemod:usr/bin/systemkeychain-helper OR
filemod:usr/bin/stty5.11.pl OR filemod:etc/manpath.d/
OR filemod:usr/local/ipcc/

Watchlist 2:
domain: comeinbaby.com

- **Watchlist 3:**
- filemod:Applications/*/start.sh
- cmdline:"/usr/bin/chflags -v hidden"

Watchlist 1 Results



Watchlists

Filter By All Binaries [✓ Processes]

Sort By Name

wirelurker

wirelurker

Last hit about 7 minutes ago

Process Search | Created about 17 minutes ago

[✓] Enable Watchlist

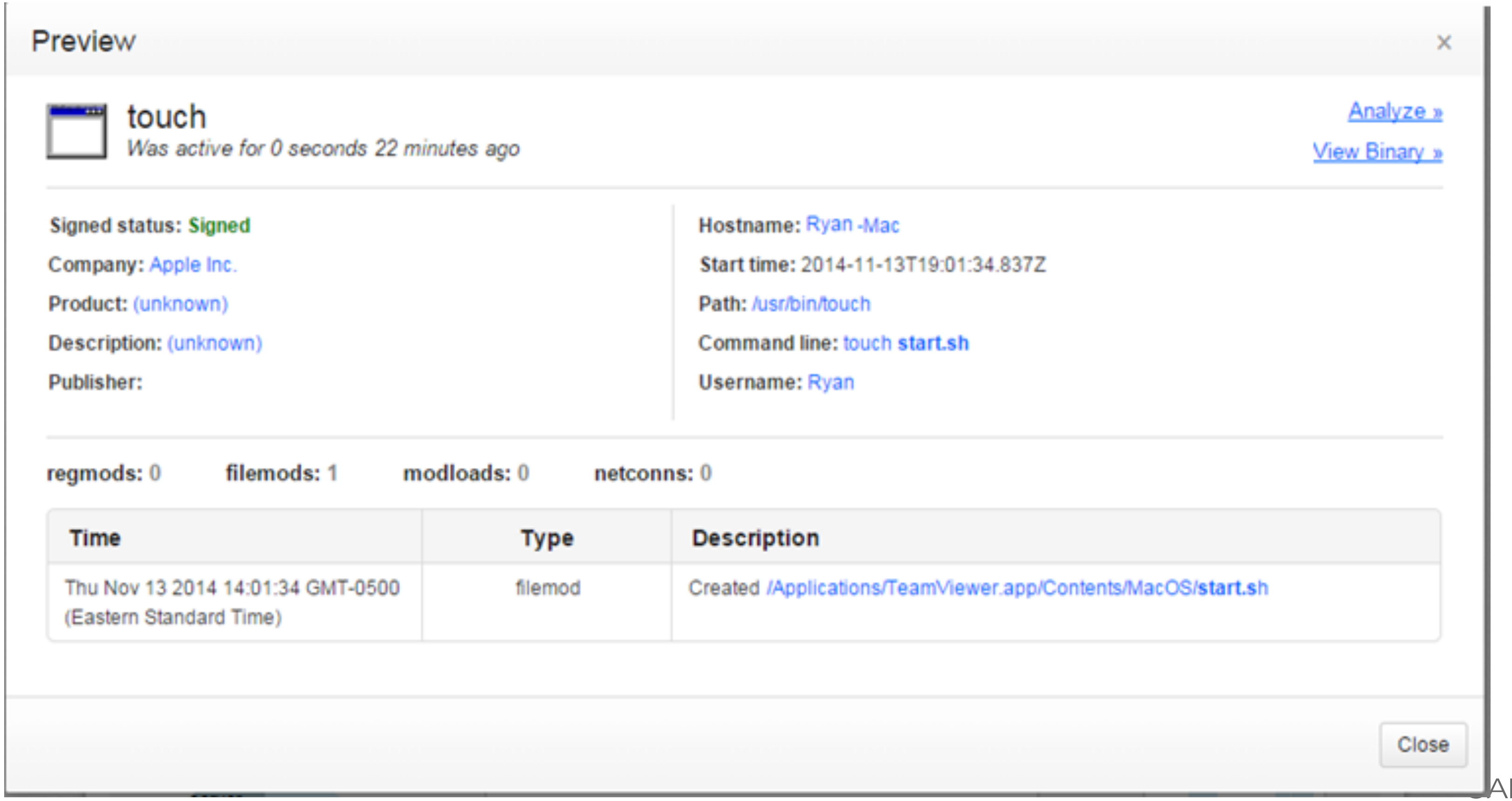
On Hit: [ ] Email Me [ ] Log to Syslog [ ] Create Alert

Showing 10 of 1 matching processes — Sort by Process start time

touch /usr/bin/touch — 26 minutes ago on Ryan-Mac — regmod 0 filemod 1 modload 0 netconn 0 proc 0

CARBON BLACK
ARM YOUR ENDPOINTS

Drill down into Watchlist 1 results



Preview ×

touch
Was active for 0 seconds 22 minutes ago

Analyze »
View Binary »

Signed status: Signed	**Hostname:** Ryan -Mac
Company: Apple Inc.	**Start time:** 2014-11-13T19:01:34.837Z
Product: (unknown)	**Path:** /usr/bin/touch
Description: (unknown)	**Command line:** touch **start.sh**
Publisher:	**Username:** Ryan

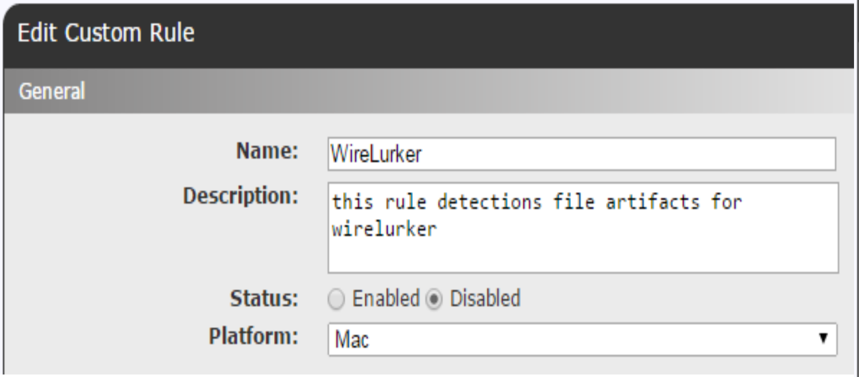
regmods: 0 **filemods:** 1 **modloads:** 0 **netconns:** 0

Time	Type	Description
Thu Nov 13 2014 14:01:34 GMT-0500 (Eastern Standard Time)	filemod	Created /Applications/TeamViewer.app/Contents/MacOS/**start.sh**

Close

CARBON BLACK
ARM YOUR ENDPOINTS

Example Bit9 block rules:



Edit Custom Rule

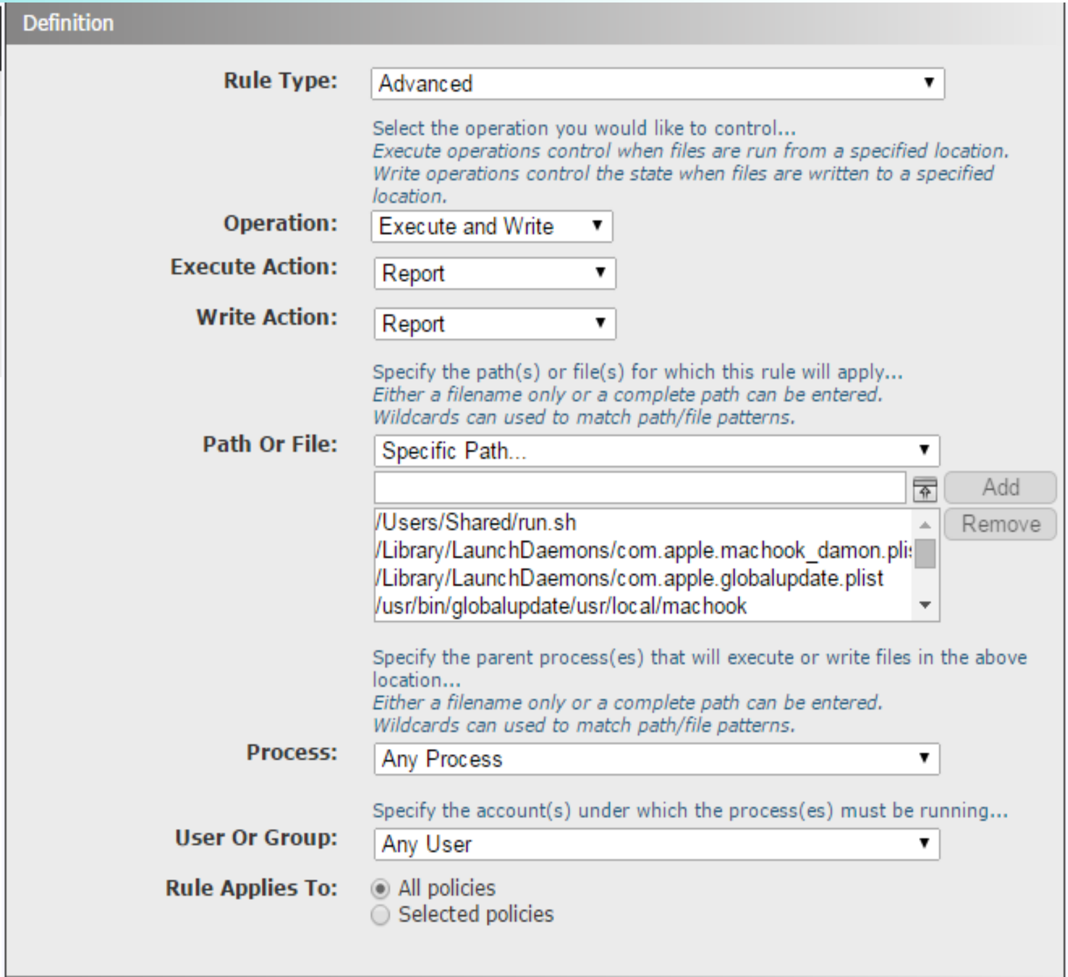
General

Name: WireLurker

Description: this rule detections file artifacts for wirelurker

Status: Enabled [✓ Disabled]

Platform: Mac



Definition

Rule Type: Advanced

Select the operation you would like to control...
Execute operations control when files are run from a specified location.
Write operations control the state when files are written to a specified location.

Operation: Execute and Write

Execute Action: Report

Write Action: Report

Specify the path(s) or file(s) for which this rule will apply...
Either a filename only or a complete path can be entered.
Wildcards can used to match path/file patterns.

Path Or File: Specific Path...

```
/Users/Shared/run.sh
/Library/LaunchDaemons/com.apple.machook_damon.pli
/Library/LaunchDaemons/com.apple.globalupdate.plist
/usr/bin/globalupdate/usr/local/machook
```

Specify the parent process(es) that will execute or write files in the above location...
Either a filename only or a complete path can be entered.
Wildcards can used to match path/file patterns.

Process: Any Process

Specify the account(s) under which the process(es) must be running...

User Or Group: Any User

Rule Applies To: [✓ All policies] Selected policies

Questions