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

$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


Cb
DISRUPT. DEFEND. UNITE.
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

CARBON BLACK

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

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

CARBON BLACK

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
```

CARBON BLACK

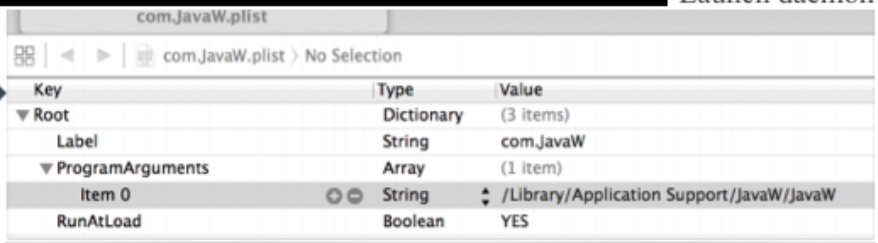
OSX/IWORM

- ‘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

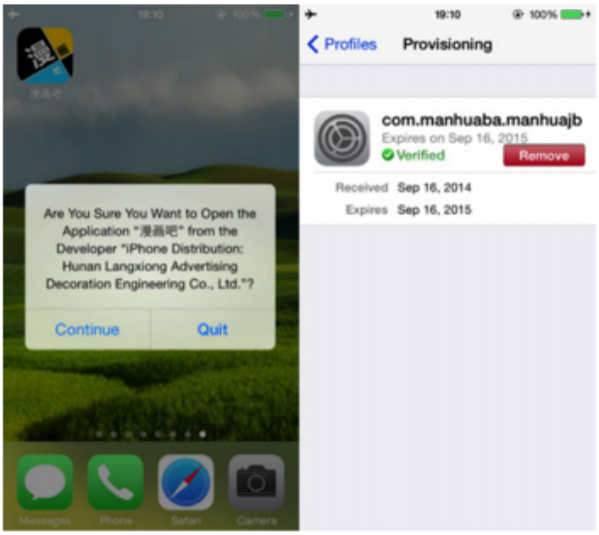
Infection Vector:
Torrents

```
# 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



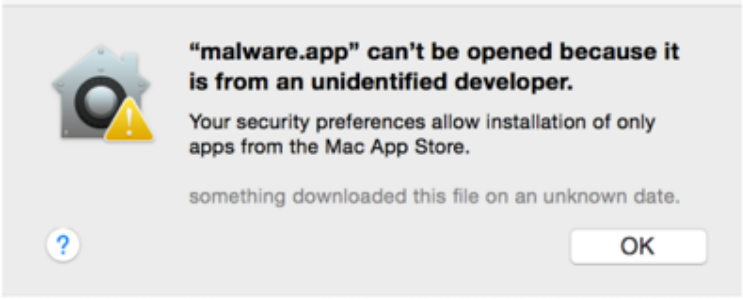
OSX/WIRELURKER



- 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

CARBON BLACK

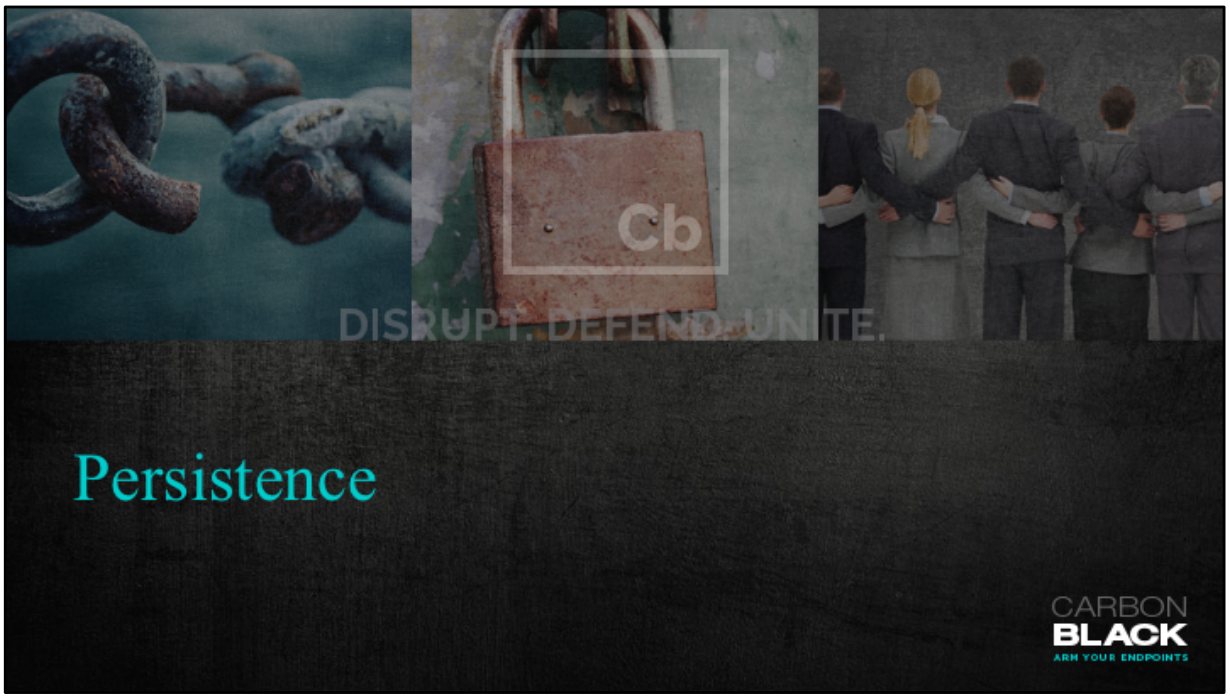
Gatekeeper



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

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

CARBON BLACK


DISRUPT. DEFEND. UNITE.
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

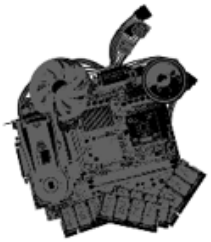
- The issue with launch items and login items is that they are easily visible, easy to detect, and are well known features.
- Consider the Mac equivalent to the run and runonce registry keys on windows or cronjobs in *nix

- 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

CARBON BLACK

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



Hackintosh

Think *really* different

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

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
```

CARBON BLACK


Cb
DISRUPT. DEFEND. UNITE.
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

CARBON BLACK

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/'
]
```

CARBON BLACK

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
```

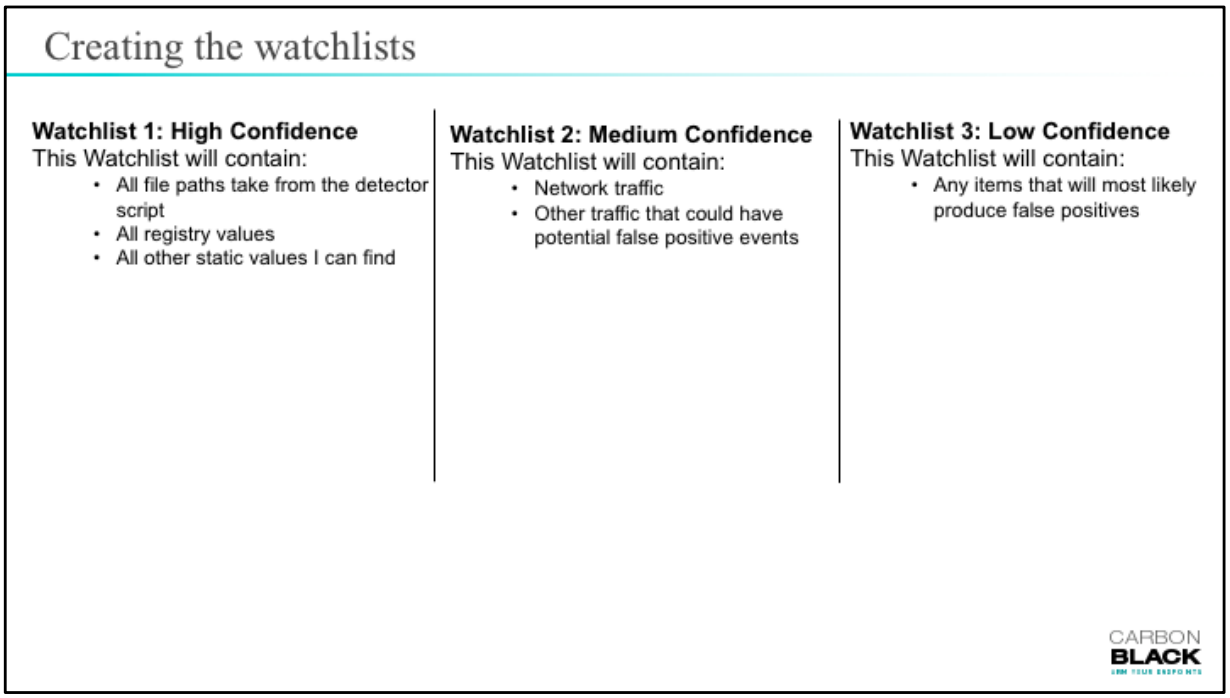
Now, your list may be different than mine. That's OK. The biggest perk of the watchlists, in my opinion, is their flexibility and ease of updating/adapting to incorporate new information. Basically, the more you learn, the more the feed can be refined for efficiency and effectiveness in your environment.

Breaking your findings down into watchlists

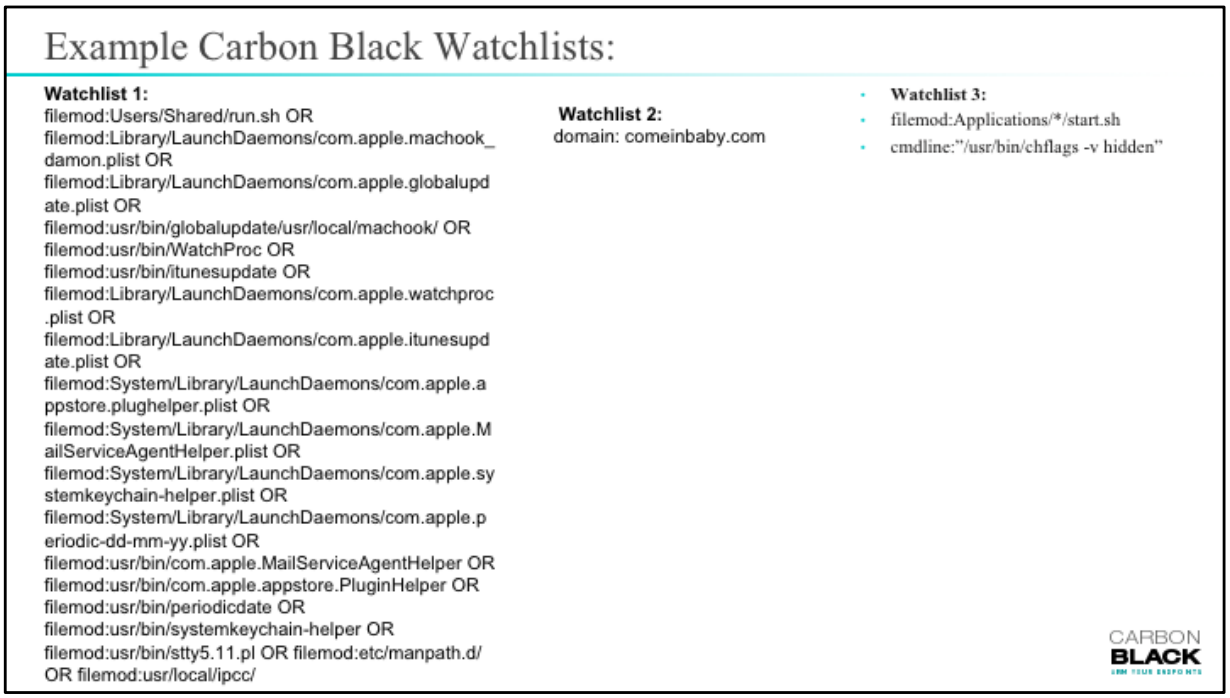
- Now that we have all of this information, we need to break it down in different ways. I suggest one of two ways:
  1. File system artifacts, registry artifacts, memory artifacts, and network artifacts
  2. High confidence, medium confidence, low confidence

- 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.

CARBON BLACK



I chose to go with the three-tiered confidence method. I chose this approach because of my confidence in the data gathered. I think a few of these rules could produce false positive events in my environment and because of that, I have chosen the approach that allows me to separate these possible problem rules to unique watchlists. This approach will allow me to disable any noisy watchlists without turning everything off and keep my environment quiet, secure and functional.



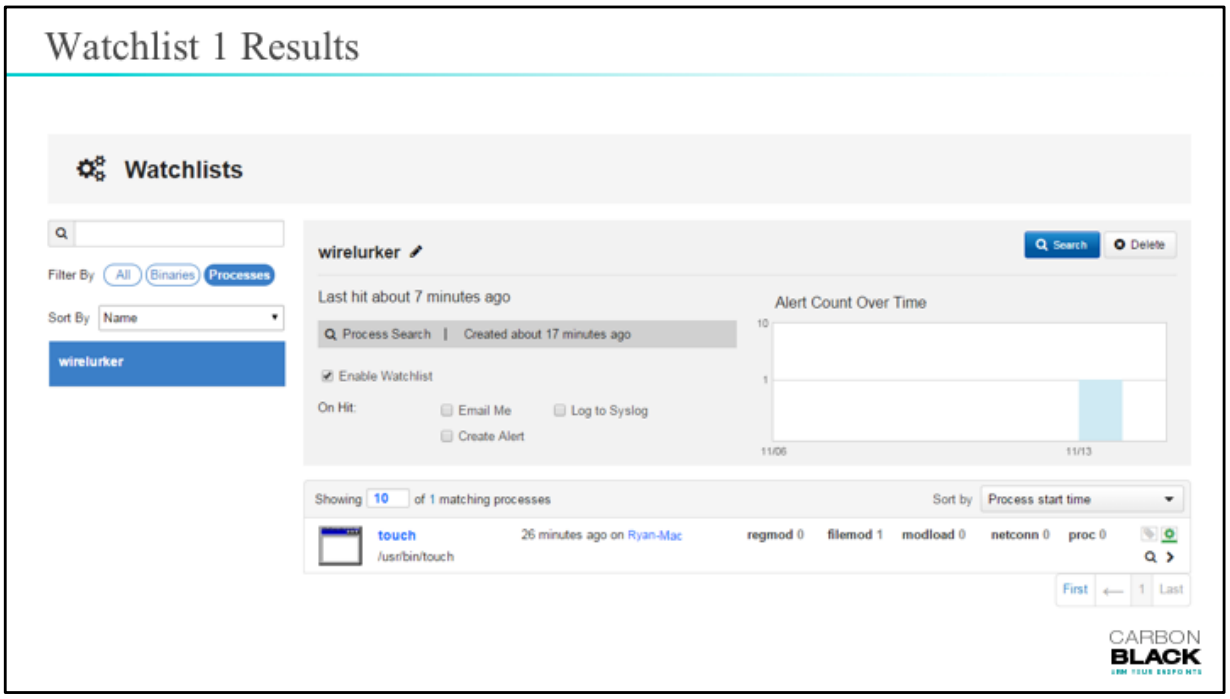
Watchlist 1:
This watchlist contains all of the file artifacts I gathered. These are all indicators that if I see them, I know they are not false positives and that I should immediately take action. I have high confidence in these indicators and am treating them as such.

Watchlist 2:
This watchlist is looking for the known domain that WireLurker connects to. Currently, there is only one known domain. This is uncommon for malware these days but not unheard of. This watchlist is kept uniquely to network traffic only to cut down on editing later on. I have high confidence in this domain being malicious. However, domains change quickly, and I do not expect this watchlist to always give me a true positive result, nor do I expect it to be around for a long time.
Therefore, I keep it separate and can easily disable it when I deem it no longer useful.

Watchlist 3:
This watchlist contains my low-confidence queries. These queries will contain false positives and I know that going into this. The reason they will fire false positives is because of how broad they are. I have high confidence that anything under "/Applications/*/start.sh" will not be legitimate but I have not tested every software

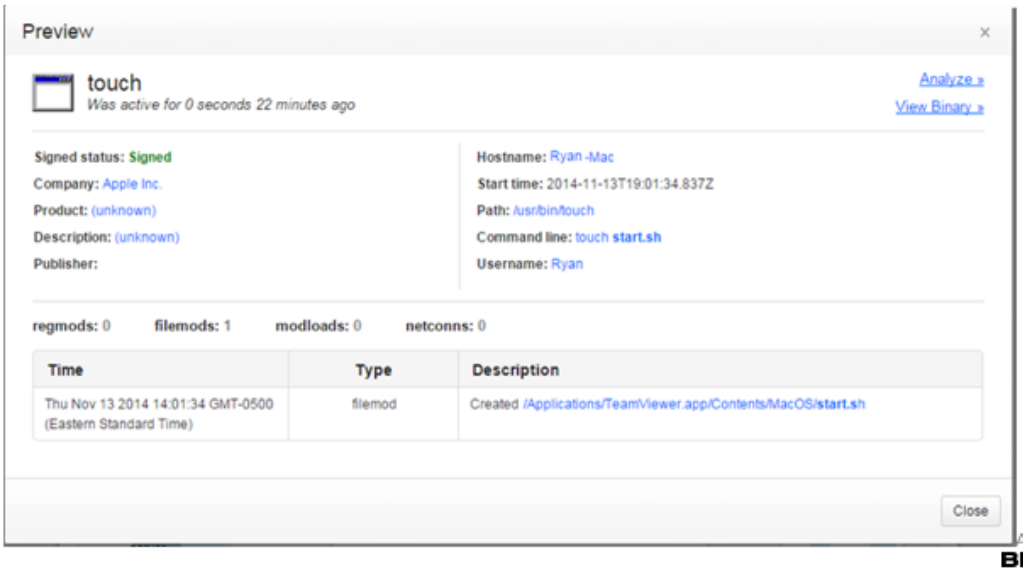
ever in every environment, so I leave room for false positives.

Also, the command for chflags to hidden is not an uncommon command. It is usually not used legitimately because it hides things from finder but not from command line.



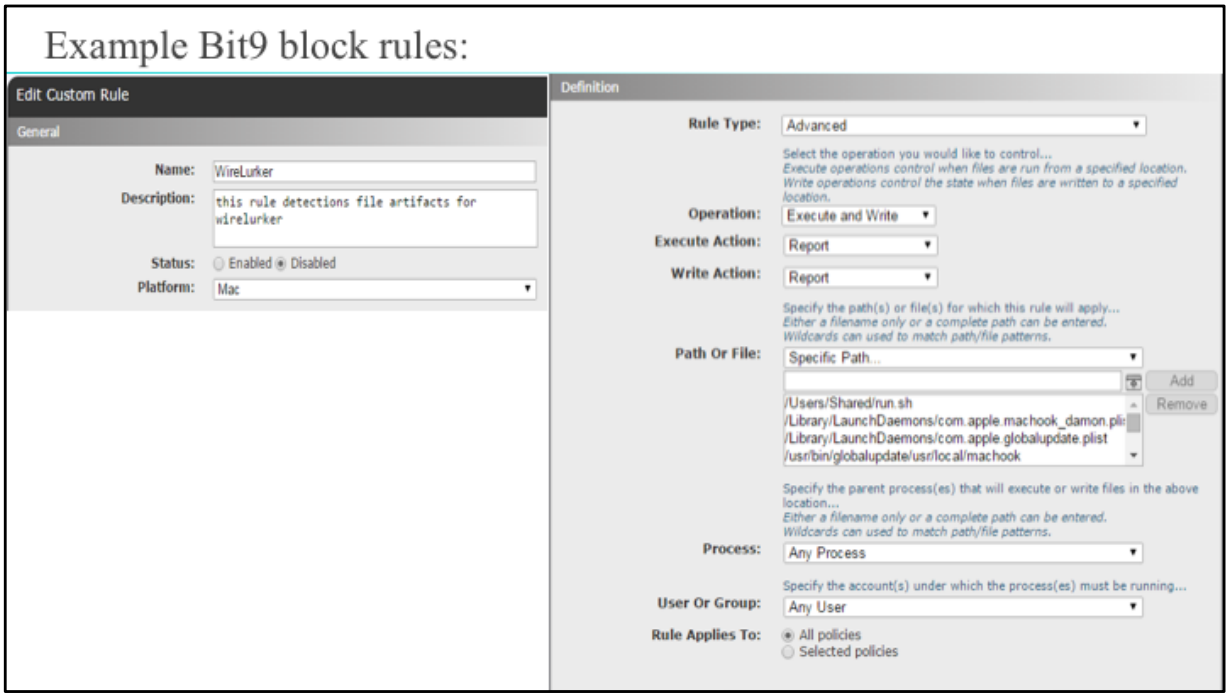
Above you can see an example of the watchlist I created for "filemod:Applications/*/start.sh." As you can see, when I set off the watchlist with the creation of start.sh in the file path of "/Applications/TeamViewer.app/Contents/MacOS/start.sh."

Drill down into Watchlist 1 results

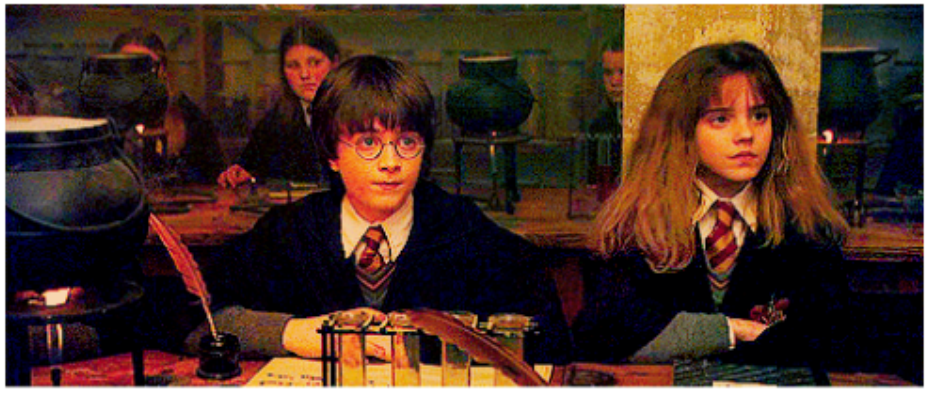


Below, you can see the drill down of the command the script used to create this file (it used the touch command).

Example Bit9 block rules:



Questions



CARBON BLACK

Using these types of techniques, you can find enough information online about pretty much any common threat/malware and create a watchlist for detection and a block rule for protection. In this presentation I used WireLurker as an example, but it could have easily been replaced with Zeus, CryptoLocker, or whatever is currently threatening your environment.