Leveraging Emond on macOS For Persistence

Christopher Ross · Follow
Published in Posts By SpecterOps Team Members · 7 min read · Jan 18, 2018

NOTE: This binary was described in the recently released, "*OS Internals, Volume I, User Space" textbook by Jonathan Levin. This book has already proven to be a great resource and I would highly recommend it if you have an interest in macOS security research.

The event monitor daemon (emond), according to Apple, "accepts events from various services, runs them through a simple rules engine, and takes an action. The action can run commands; send email, or SMS messages". Sounds interesting right? Emond has been available since OS X 10.7, so the details discussed in this post are applicable to the most recent version of macOS (10.13.2).

This binary functions as a normal daemon and is executed by launchd every time the OS starts up. There are a few on-disk components to emond as well. The launchd config file is located where other system daemons reside:

```
xorrior in ~(ruby-2.4.1) λ cat /etc/emond.d/emond.plist
<?x
<!D
<pl
<di
        <dict>
                <key>additionalRulesPaths</key>
                <array/>
                <key>debugLevel</key>
                <integer>0</integer>
                <key>filterByUID</key>
                <string>0</string>
                <key>filterByGID</key>
                <string></string>
                <key>periodicEvents</key>
                <array>
                        <dict>
                                <key>eventType</key>
                                <string>periodic.daily.midnight</string>
                                <key>startTime</key>
                                <string>0</string>
                        </dict>
                </array>
                <key>errorLogPath</key>
                <string>/Library/Logs/EventMonitor/EventMonitor.error.log</string>
                <key>eventLogPath</key>
                <string>/Library/Logs/EventMonitor/EventMonitor.event.log</string>
                <key>logEvents</key>
                <false/>
                <key>saveState</key>
                <true/>
        </dict>
        <key>initialGlobals</key>
        <dict>
                <key>notificationContacts</key>
                <array/>
        </dict>
</dict>
</plist>
xorrior in ~(ruby-2.4.1) λ
```


For rules, they're stored in the */etc/emond.d/rules/* directory and they should be in plist format. There is an example rules file already present in this directory (SampleRules.plist). The example defines the name, eventType, and the action once the event triggers. There are several event types (*startup, periodic, auth.success, auth.failure, etc.*) but for this demonstration we will only use *startup*. The *startup* event type will trigger the rule once it has been loaded by emond. The *periodic* event type will only trigger once the defined 'startTime' has elapsed. The *auth.success* event type will only trigger once a user successfully authenticates, and *auth.failure* will trigger on

action. Thus, any commands that require network access will not work.

To craft a rule file, we will utilize the SampleRule.plist file that already exists and modify it as necessary.

Figure 2: SampleRules.plist

```
<?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">
<array>
    <dict>
        <key>name</key>
        <string>sample rule</string>
        <key>enabled</key>
        <true/>
        <key>eventTypes</key>
        <array>
            <string>startup</string>
        </array>
        <key>allowPartialCriterionMatch</key>
        <false/>
        <key>criterion</key>
        <array>
            <dict>
                <key>operator</key>
                <string>True</string>
            </dict>
        </array>
        <key>actions</key>
        <array>
            <dict>
                <key>message</key>
                <string>Event Monitor started at ${builtin:now}</string>
                <key>type</key>
                <string>Log</string>
                <key>logLevel</key>
                <string>Notice</string>
                <key>logType</key>
                <string>syslog</string>
            </dict>
        </array>
```

The sample contains some of the values that we need for our rules file. Specifically, we can remove the “allowPartialCriterionMatch” key and change the name if desired. The defined actions need to be modified for the *run*

Notice that the first action is to sleep for 10 seconds in order to wait with the hope that network access will become available. The amount of time is just a rough estimate and may vary across hosts. The second action will just curl a hosted Empire stager and pipe it to Python. The ampersand symbol is an

file specifically but emond will stop complaining about not finding any
ru

Figure 5: emond error log after starting the service

Once the service has started, if you have defined a startup event type, your event will immediately fire and trigger any actions. Now, we should see the request for an Empire stager and then a new agent.

Figure 6: Hosted stager and web request from emond

Figure 7: New agent

time of writing. It's certainly possible that this is being used in the wild and ju

Detection

This method of persistence is predicated on several changes to the file system. Fortunately, macOS offers the fsevents API to capture file system events. In essence, fsevents records all events for the file system in each volume. Initially, events are stored in memory and then written to disk once the memory buffer becomes full or before the volume is unmounted. FSEvent log files are stored in a gzip compressed format and follow a hexadecimal naming scheme. All log files are stored in a hidden directory: */.fseventsd/*. Root privileges are required to access this directory. Another caveat for fsevents are that timestamps are not included with entries in the log file. With access to the API, we can use Python or Objective-C to sift through all received events and alert once an event for file creation/modification occurs in the rules directory or the QueueDirectory.

For a simple example, we can use the fswatch open-source project to monitor for changes. It offers support for multiple platforms, thorough documentation, and the project is actively maintained. You can review the project here. So, after installing the application via homebrew, we can setup a monitor for the emond rules directory.

Figure 8:Example fswatch rule for emond rules directory

This is a very rudimentary example and may not be the best solution in a large macOS environment with multiple deployments. A more applicable alternative would be osquery. Osquery offers File Integrity Monitoring which uses the fsevents api to log file system changes to specific directories and/or files. Additional information can be found here. After installing osquery, you will need to provide a configuration file to monitor file system events. You can see a simple example to monitor all file system events within the rules directory below. All events will be queried at 60 second intervals.

Figure 10: Example osquery config

The log entry shown above is also available here. These methods of de ut th

References

Levin, J. (2017) *OS Internals, Volume I: User Space*. North Castle, NY: Technologeeks.com

Reynolds, J. (2016, April). *What is emond?*. Retrieved from: http://www.magnusviri.com/Mac/what-is-emond.html

. . .

Originally published at www.xorrior.com.

Programming | Mac Os X | Persistence | Empires | Security

114

Jul

Hope Walk... in Posts By SpecterOps Team Memb...

An Introduction to Manual Active Directory Querying with Dsquery...

Introduction

Jun 2, 2021 91 3

Christopher R... in Posts By SpecterOps Team Me...

No Place Like Chrome

Chrome extensions were first introduced to the public in December of 2009 and use...

Feb 8, 2019 17

See all from Christopher Ross

See all from Posts By SpecterOps Team Members

Recommended from Medium

F. Perry Wilson, MD MSCE

How Old Is Your Body? Stand On One Leg and Find Out

According to new research, the time you can stand on one leg is the best marker of...

Oct 23 6.3K 146

Austin Starks in DataDrivenInvestor

I used OpenAI's o1 model to develop a trading strategy. It is...

It literally took one try. I was shocked.

Sep 15 5.3K 138

Thomas Ricouard

Why you should use Xcode 16 buildable folders instead of groups

I've recently migrated Ice Cubes, my open-source SwiftUI Mastodon client to use file...

Oct 24 355 3

Nidhi Jain 👩‍💻 in Code Like A Girl

7 Productivity Hacks I Stole From a Principal Software Engineer

Golden tips and tricks that can make you unstoppable

Oct 15 3.4K 64