What's Your Game Plan?

Leveraging Apple's Game Engine to Detect macOS Threats
About Us

cybersecurity solutions for the macOS enterprise

@digita_security
Objective See Tools++

Monitoring Capability + Built-in Detection Logic = New Tool!

Let’s Rethink/Redesign this...
Outline

Today's GamePlan

macOS monitoring

Game Engine-Based Analysis & Detection

macOS threats
macOS monitoring framework
MonitorKit
macOS monitoring framework

Swift/Objective-C framework

- OpenBSM
- FSEvents
- Spotlight Notifications
- Event Tap
- IOKit

- ProcessMonitor.start()
- AuthMonitor.start()
- FileSystemMonitor.start()
- DownloadMonitor.start()
- ScreenShotMonitor.start()
- SyntheticClickMonitor.start()
- USBMonitor.start()

- ProcessEvent: {pid, type, args, path, ppid, ... }
- AuthEvent: {... }
- FSEvent: {type, pid, path, ... }
- DownloadEvent: {path, timestamp, ...}
- ScreenshotEvent: {path, timestamp, ...}
- ClickEvent: {pid, type, targetpid, ... }
- USBEvent: {type, device, address, port, ... }
OpenBSM Parsing
Reading and parsing /dev/auditpipe

Audit Mask Options:

- AUDIT_CLASS_FILE_READ
- AUDIT_CLASS_FILE_WRITE
- AUDIT_CLASS_FILE_ATTR_ACCESS
- AUDIT_CLASS_FILE_ATTR_MODIFY
- AUDIT_CLASS_FILE_CREATE
- AUDIT_CLASS_FILE_DELETE
- AUDIT_CLASS_FILE_CLOSE
- AUDIT_CLASS_PROCESS
- AUDIT_CLASS_NETWORK
- AUDIT_CLASS_IPC
- AUDIT_CLASS_NON_ATTIB
- AUDIT_CLASS_ADMIN
- AUDIT_CLASS_LOGIN_LOGOUT
- AUDIT_CLASS_AUTH
- AUDIT_CLASS_APP
- AUDIT_CLASS_IOCTL
- AUDIT_CLASS_EXEC
- AUDIT_CLASS_MISC
- AUDIT_CLASS_ALL

Audit Record:

Type: AUT_EXEC

- AUT_EXEC_ARGS
- AUT_PATH
- AUT_ATTR32
- AUT_SUBJECT32
- AUT_SUBJECT32
- AUT_TRAILER

Audit Token:

Token AUT_EXEC_ARGS:
- arg : system_profiler
- arg : -xml
- arg : SPMemoryDataType

Token AUT_ATTR32:
- mode : 33261
- uid  : 0
- gid  : 0
- fsid : 16777223
- nid  : 34010260
- dev  : 0

Get Cozy with OpenBSM Auditing - Patrick Wardle
*OS Internals – Chapter 2: Auditing - Jonathan Levin
OpenBSM
Open Source Event Auditing

event coverage

event details

resource impact

complexity

MonitorKit wraps Process and Auth events to simplify usage

---

Raw OpenBSM Monitoring:

```swift
let mask: u_int = AuditConstants.AUDIT_CLASS_PROCESS | AuditConstants.AUDIT_CLASS_EXEC
let filter: [UInt16] = [
    UInt16(AUE_EXEC),
    UInt16(AUE_EXIT),
    UInt16(AUE_FORK),
    UInt16(AUE_EXECVE),
    UInt16(AUE_POSIX_SPAWN)
]
if let monitor = try? BSMMonitor(mask, recordFilter: filter) {
    monitor.start { (record: BSMRecord) in
        for token in record.tokens {
            print("Token \(token.tokenString()):\n\n\(token.stringValue())")
        }
    }
}
```

---

Process Monitoring:

```swift
if let monitor = try? ProcessMonitor() {
    monitor.start { (event: ProcessEvent) in
        if event.type == .CREATE {
            print("Process with pid \(event.pid) created")
        } else if event.type == .EXIT {
            print("Process with pid \(event.pid) exited")
        }
    }
}
```
BSMExplorer - Open source application that utilizes MonitorKit to provide GUI for filtering and viewing BSM records.
Apple’s FSEvents API does not provide the responsible pid for the file i/o, so direct access (/dev/fsevents) is leveraged.

```swift
let monitor = FileMonitorFSE()

public static let filter = [
    FSEventTypes.CREATE_FILE.rawValue,
    FSEventTypes.DELETE.rawValue,
    FSEventTypes.STAT_CHANGED.rawValue,
    FSEventTypes.RENAME.rawValue,
    FSEventTypes.CONTENT_MODIFIED.rawValue,
    FSEventTypes.EXCHANGE.rawValue,
    FSEventTypes.FINDER_INFO_CHANGED.rawValue,
    FSEventTypes.CREATE_DIR.rawValue,
    FSEventTypes.CHOWN.rawValue,
    FSEventTypes.XATTR_MODIFIED.rawValue,
    FSEventTypes.XATTR_REMOVED.rawValue
]

monitor.start(eventFilterInclude: filter) { event in
    print("\(event.pid) \(event.type.string) \(event.path)")
}
```
Spotlight Notifications

Extended Attributes (xattrs) Monitor

Register observers and get notified of user behaviors such as screenshots and downloads.

```
@objcMembers public class ScreenshotEvent: NSObject {
    public let timestamp: Date
    public let path: String
}

let predicate = NSPredicate(fromMetadataQueryString: "kMDItemIsScreenCapture = 1")

public class ScreenshotMonitor: BaseSpotlightMonitor<ScreenshotEvent> {
    public init() {
        super.init(
            predicate: predicate,
            forNotification: NSNotification.Name.NSMetadataQueryDidUpdate)
    }
}

monitor.start() { event in
    print("Screenshot \(event.path)")
}
```

Raw Spotlight Monitoring:

```
let monitor = DownloadMonitor()

monitor.start() { event in
    print("Downloaded File \(event.path)")
}
```

Download Monitoring:
CoreGraphics Event Taps
Programatic Input Monitoring

- event coverage
- event details
- resource impact
- complexity

Monitor more obscure system events such as synthetic/programatic mouse or keyboard clicks

Synthetic Click Monitoring:

```javascript
let monitor = SyntheticClickMonitor()

monitor.start() {
  event in
  print("Detected click from pid \\
         \(event.pid)")
}
```

The Mouse is Mightier than the Sword - Patrick Wardle
IOKit Notifications

Monitoring for New Hardware

- Event coverage
- Event details
- Resource impact
- Complexity

Register observers and get notified of new hardware such as USB Devices.

USB Monitoring:

```swift
let monitor = USBMonitor()

monitor.start() {
    for event in 
        print("Detected usb device \(event.device.serialNumber)")
}
```
monitorKit UI

![System Information](image)

**Name:** Gibson  
**Serial Number:** C02TW08THV2Q  
**Model:** MacBookPro14,2  
**Operating System:** Version 10.14 (Build 18A391) (10.14.0)  
**Memory:** 16 GB  
**Volume:** 357.35 GB / 999.59 GB

<table>
<thead>
<tr>
<th>Type</th>
<th>Time</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Process:** 0.0% **User:** 5.0% **System:** 6.5%
Check out MonitorKit

Future Monitors:

- Do Not Disturb laptop open events
- Oversight microphone and video camera access events
- Keylogger detections

Will be released to our github account (https://github.com/DigitaSecurity)
Real Time Analysis
Using Apple's GameplayKit
Goals: Context + Flexible analysis
Adjust and extend detections without core modification

Provide structured output and configurable logic to allow admins to make better, perhaps automated decisions.

!BlackBox and !Rigid. Allow the system to be adjusted, and extended in the enterprise without core modifications or new code.
Game Logic Engine

the traditional pieces

sensor(s) → modelled events → actuators

logic controller = Σ logic blocks

state

Analyzing System Events

the scheme

logic controller = Σ logic blocks

actions

actuators

monitorKit

cache

analytics

state
GameplayKit
Apple's Game Engine Framework == logic controller

“allows a list of [logic blocks], together with context for evaluating them and interpreting results, for use in constructing data-driven logic or fuzzy logic systems”

"Separate game design from executable code"

GKRueSystem
A simple example

```swift
import Foundation
import GameplayKit

class SimpleLogicSystem {
    let system: GKRueSystem

    init() {
        self.system = GKRuleSystem()
        // Define the game logic
        let hunt = NSPredicate(format: "$invincible == false")
        let retreat = NSPredicate(format: "$invincible == true")
        // Load it into the "rule system"
        self.system.add(GKNSPredicateRule(predicate: hunt, assertingFact: "hunt" as NSObjectProtocol, grade: 1.0))
        self.system.add(GKNSPredicateRule(predicate: retreat, retractingFact: "hunt" as NSObjectProtocol, grade: 1.0))
    }

    // Call update on a timer, or when the event occurs
    func update(powerPellet: Bool) {
        self.system.state["invincible"] = powerPellet
        self.system.reset()
        self.system.evaluate()

        let ghostHunt = self.system.grade(forFact: "hunt" as NSObjectProtocol)
        if ghostHunt > 0.0 {
            print("lets go hunting")
        } else {
            print("run, Forest, run!")
        }
    }
}
```

(1) Define and load the "game" logic

(2) On a timer, or drive-by events, re-evaluate / execute the engine

(3) Respond to the output of the evaluation

```swift
// Simple example
let sys = SimpleLogicSystem()
sys.update(powerPellet: true)
sys.update(powerPellet: false)
run, Forest, run!
```
#### "GamePlans"

Defining predicates across the Mitre ATT&CK Framework

<table>
<thead>
<tr>
<th>Initial Access</th>
<th>Execution</th>
<th>Persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privilege Escalation</td>
<td>Defense Evasion</td>
<td>Credential Access</td>
</tr>
<tr>
<td>Discovery</td>
<td>Lateral Movement</td>
<td>Collection</td>
</tr>
<tr>
<td>Exfiltration</td>
<td>Command &amp; Control</td>
<td></td>
</tr>
</tbody>
</table>

---

**Digita Security**

22
$event.isNewDirectory == 1 AND ( 
$event.path MATCHES[cd] "/[System/Library/Extensions/[^
/]*" OR 
$event.path MATCHES[cd] "/[Library/Extensions/[^
/]*" 
) \rightarrow \text{Persist, Kext}

Labels applied by logic blocks

detecting macOS att&ck persistence techniques
Chaining Logic Blocks

For more accurate and actionable detection

Code Signing (WhatsYourSign inspired) logic blocks

Persist &amp; !AppleSigned &amp; !AppStore

chaining it all together

Apple | AppStore | Dev | AdHoc | Not Signed
Advanced Behavioral Detections

Ransomwhere? Inspired Detections

File I/O Modification of file in /Users/
+ Process responsible for file modification is not Apple Signed or from App Store
+ File contents now encrypted (Chi Square distribution, Monte Carlo pi approximation,...)
+ Process responsible for file modifications has quickly encrypted several files
= Potential Ransomware

\[
$event.isModified = \text{TRUE AND}
\]
\[
$event.path \text{ MATCHES[cd] } "/Users/*/" \text{ AND}
\]
\[
!$event.process.labels.contains("AppleSigned", "AppStore") \text{ AND}
\]
\[
!$event.contents.encrypted = \text{TRUE}
\]

\[
$event.labels.contains("UntrustedEncrypt") \text{ AND}
\]
\[
@\text{SUBQUERY(cache, e, e.date > (X sec ago)) AND}
\]
\[
$e.process.pid = $event.process.pid).@count > 3 \rightarrow \text{Ransomware}
\]

The algorithm (https://objective-see.com/blog/blog_0x0F.html)
MonitorKit + Real Time Analysis vs macOS threats
Another look at FruitFly

A Better Fly Trap?

A chance for detection on install

Generically Detect a Launch Agent being registered

Determine that its a hidden program to be launched


FruitFly Launch & Behaviors

If persisting a hidden file isn't enough cause for alarm

$event.process.path.lastPathComponent.startsWith(".")

A hidden file is being executed

$event.process.path.startsWith("/tmp")

Then extracts and executes a file out of tmp

+ Controls the camera
+ Issue synthetic clicks
+ Synthetic keyboard events

WindShift Execution Vector
Via Phishing, "Safe" Open, CustomURL Handlers, & Redirects

macOS parses and auto-registers handlers on write.

windshift:// launches app

Extracted App Info.plist

Open "Safe" files: Including PDFs & Archives?!?

https://digitasecurity.com/blog/2018/08/30/windshift/
Detecting WindShift Execution Vector
Via "Safe" Open & CustomURL Handlers

- $event.isNewDirectory == 1 AND $event.file.isAppBundle == 1 AND $event.file.bundle.infoDictionary.CFBundleURLTypes != nil

Custom URL Handler

- $event.isNewDirectory == 1 AND $event.process.name != 'SAFARI'

File automatically written by 'Safari'

- $event.signType != 'Apple' OR $event.signType != 'AppStore'

Not distributed by Apple or the App Store

https://digitasecurity.com/blog/2018/08/30/windshift/
My Favorite Adware Detector!? 

Process path contains Adobe or Flash

But not validly signed by Adobe!

$event.process.path.contains("Adobe") OR $event.process.path.contains("Flash")

$event.process.sigtype == Adhoc OR ($event.process.sigtype == Dev AND $event.process.signer != "Adobe Systems, Inc.")
MonitorKit + GamePlan = GamePlan
GamePlan

Purpose built macOS endpoint detection, auditing and threat hunting platform with extensible analytics efficiently executed by a game engine.

Generally available for the enterprise in 2019!
Questions and Answers
Contact us anytime!

https://digitasecurity.com
gameplan@digitasecurity.com