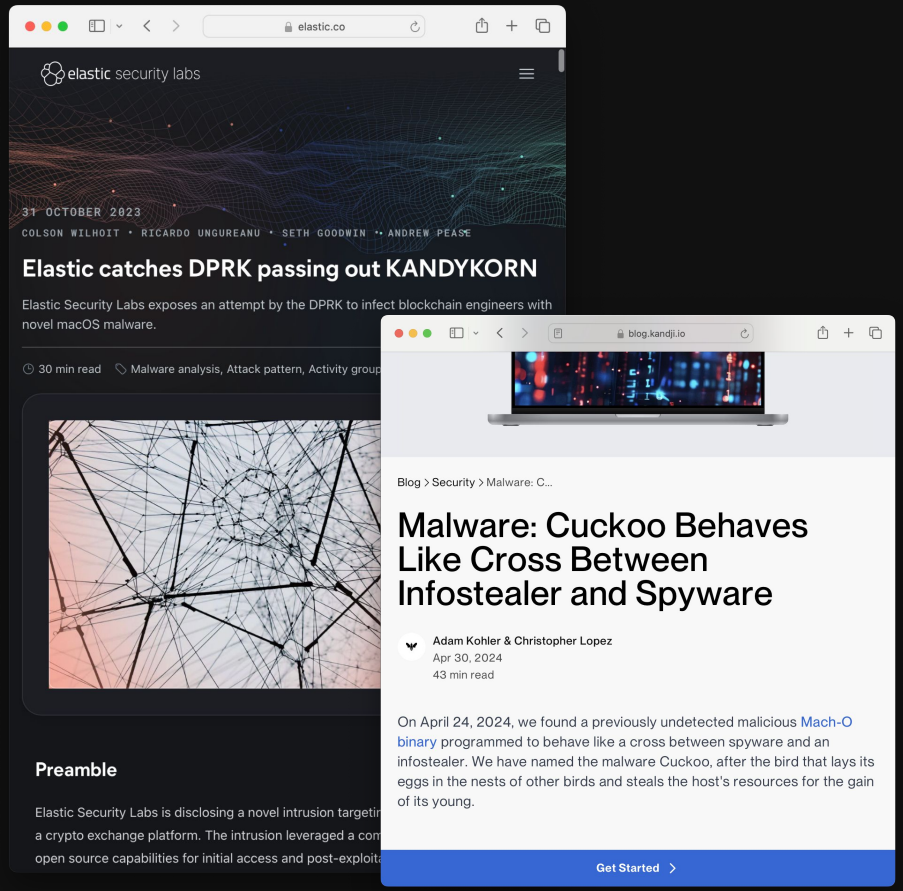
Your Mac's Immune System

Resilience through Endpoint Security

05.10.2024

Brandon Dalton
Staff Security Researcher

🏠 How can we detect what we've never seen?



- macOS threat and internals researcher
- Security tools developer
  - Red Canary Mac Monitor
  - POSIX AtomicTestHarnesses
  - Threat emulation
- Behavioral threat detection

Brandon Dalton

Staff Security Researcher @ Kandji

Overview

01 **The threats**

02 **Telemetry and eventing**

03 **Understanding Endpoint Security**

04 **AMOS: A case study**

05 **Building a detector**

06 **Demo!**

What’s out there?

AppleScript / OSA abuse

IDENTIFIED	VARIANT	CLASSIFICATION
2017	OSX.Pirrit	Adware
2017	Snake	Trojan
2017	OSX/Dok	RAT
2018	OSX.DarthMiner	RAT
2019	OSX.Pirrit (second variant)	Adware
2020	OSX.EvilQuest / ThiefQuest	Stealer
2020	XCSSET	RAT
2021	OSX.OSAMiner	Cryptojacking
2023	RustBucket	Stager / RAT
2023	Atomic Stealer	Stealer
2023	MacStealer	Stealer
2023	Geacon stager	Stager
2023	MetaStealer	Stealer

2024 Threat Detection Report (Red Canary)
AppleScript

blog.kandji.io

Blog > Security > How AMOS m...

How AMOS macOS Stealer Avoids Detection

Sam Mayers & Christopher Lopez
Mar 2, 2024

10 min rea

Atomic macOS Stealer (AMOS) was first spo
powerful piece of malware that targets Apple
installing the software on their computers. Th
Telegram; as of January 20, 2024, the price

Once installed, Atomic Stealer can exfiltrate
including keychain passwords, user documer
browser data, credit card information, crypto

```
ATOMIC MACOS STEALER (AMOS)
(The first and most famous stealer on

SYSTEM:
- Collect notes from Notes
- Keychain (Dump all saved user passw
- SystemInfo (Complete system informa
- MacOS Password
- Hidden console when launching softw

BROWSERS:
```

TELEGRAM

AMOS (MacOS Stealer)
ATOMIC MACOS STEALER

SYSTEM:
- Keychain (Export all passwords from apple keychain)
- SystemInfo (Full information about system)
- FileGrabber (Desktop, Documents)
- MacOS Password

BROWSERS:
- Chrome (Autofills, Passwords, Cookies, Wallets, Cards)
- Firefox (Autofills, Cookies)
- Brave (Cookies,Passwords,Autofills, Wallets, Cards)
- Edge (Cookies,Passwords,Autofills, Wallets, Cards)
- Vivaldi (Cookies,Passwords,Autofills, Wallets, Cards)
- Yandex (Cookies,Autofills, Wallets, Cards)
- Opera (Cookies,Passwords,Autofills,Wallets,Cards)
- OperaGX (Cookies,Passwords,Autofills,Wallets,Cards)

Kandji and SentinelOne

virustotal.com

URL, IP address, domain or file hash

Sign in | Sign up

15 / 44

Community Score

15/44 security vendors and no sandboxes flagged this file as malicious

Reanalyze | Similar | More

0afc208dedd1fafd63df51b4c344383bca867f1e2067d65b4...
CrackInstall

macho | 64bits | checks-hostname | arm | multi-arch

Size 335.52 KB
Last Modification Date 6 hours ago

DETECTION | DETAILS | RELATIONS | BEHAVIOR | COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

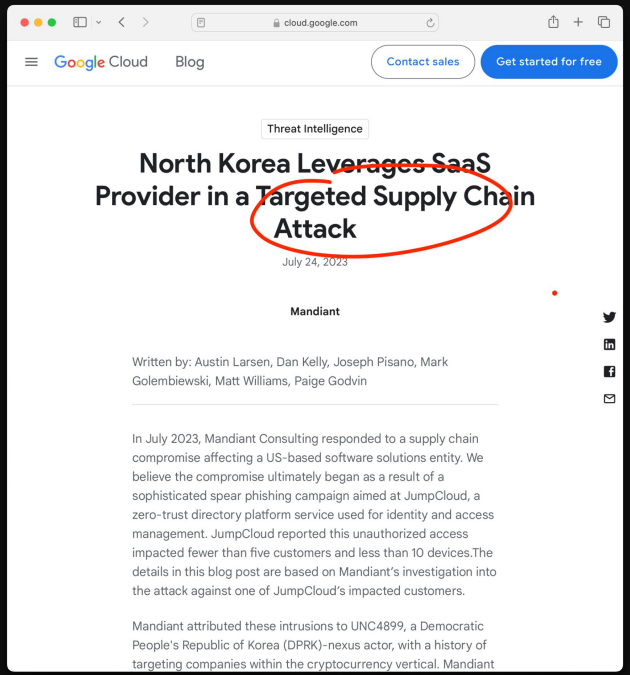
Popular threat label trojan.
Threat categories trojan

Security vendors' analysis — Do you want to automate checks?

Avira (no cloud)	OSX/GM.Agent.QD	ClamAV	Unix.Malware.Macos-10027468-0
Cynet	Malicious (score: 99)	Emsisoft	Trojan.MAC.Generic.118365 (B)
eScan	Trojan.MAC.Generic.118365	ESET-NOD32	A Variant Of OSX/PSW.Agent.AN
Fortinet	MAC/Stealer.20!tr	Google	Detected
Ikarus	Trojan.OSX.Psw	K7GW	Trojan (0040f46e1)
Lionic	Trojan.OSX.Amos.i!c	MAX	Malware (ai Score=82)
	OSX.Trojan.Gen.2	Tencent	Osx.Trojan-QQPass.QQRob.Zchl
	Trojan.MAC.Generic.118365	Acronis (Static ML)	Undetected

VirusTotal April 30, 2024

Further still… supply chain horror…


cloud.google.com

Google Cloud Blog Contact sales Get started for free

Threat Intelligence

North Korea Leverages SaaS Provider in a Targeted Supply Chain Attack

July 24, 2023

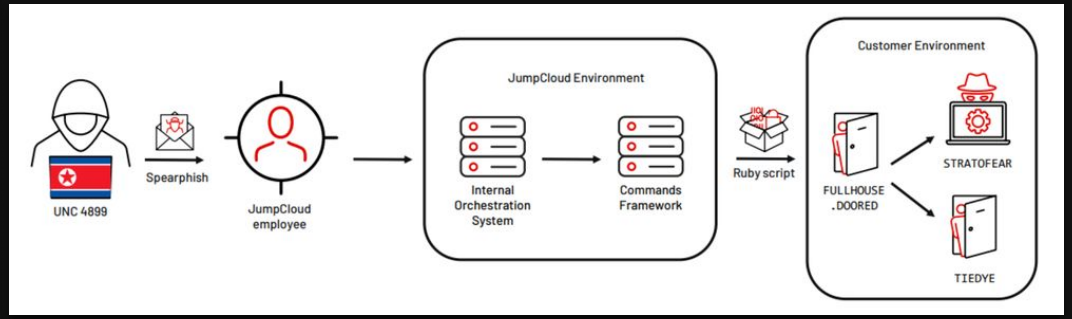
Mandiant

Written by: Austin Larsen, Dan Kelly, Joseph Pisano, Mark Golembiewski, Matt Williams, Paige Godvin

In July 2023, Mandiant Consulting responded to a supply chain compromise affecting a US-based software solutions entity. We believe the compromise ultimately began as a result of a sophisticated spear phishing campaign aimed at JumpCloud, a zero-trust directory platform service used for identity and access management. JumpCloud reported this unauthorized access impacted fewer than five customers and less than 10 devices.The details in this blog post are based on Mandiant's investigation into the attack against one of JumpCloud's impacted customers.

Mandiant attributed these intrusions to UNC4899, a Democratic People's Republic of Korea (DPRK)-nexus actor, with a history of targeting companies within the cryptocurrency vertical. Mandiant

- Stager execution (*Ruby*)
- Persistence (*launchd*)
- Collection (*proc, file, netconn, sleep*)

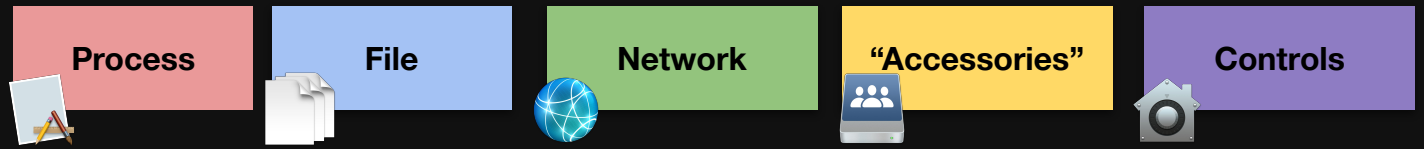


Mandiant / Google Cloud

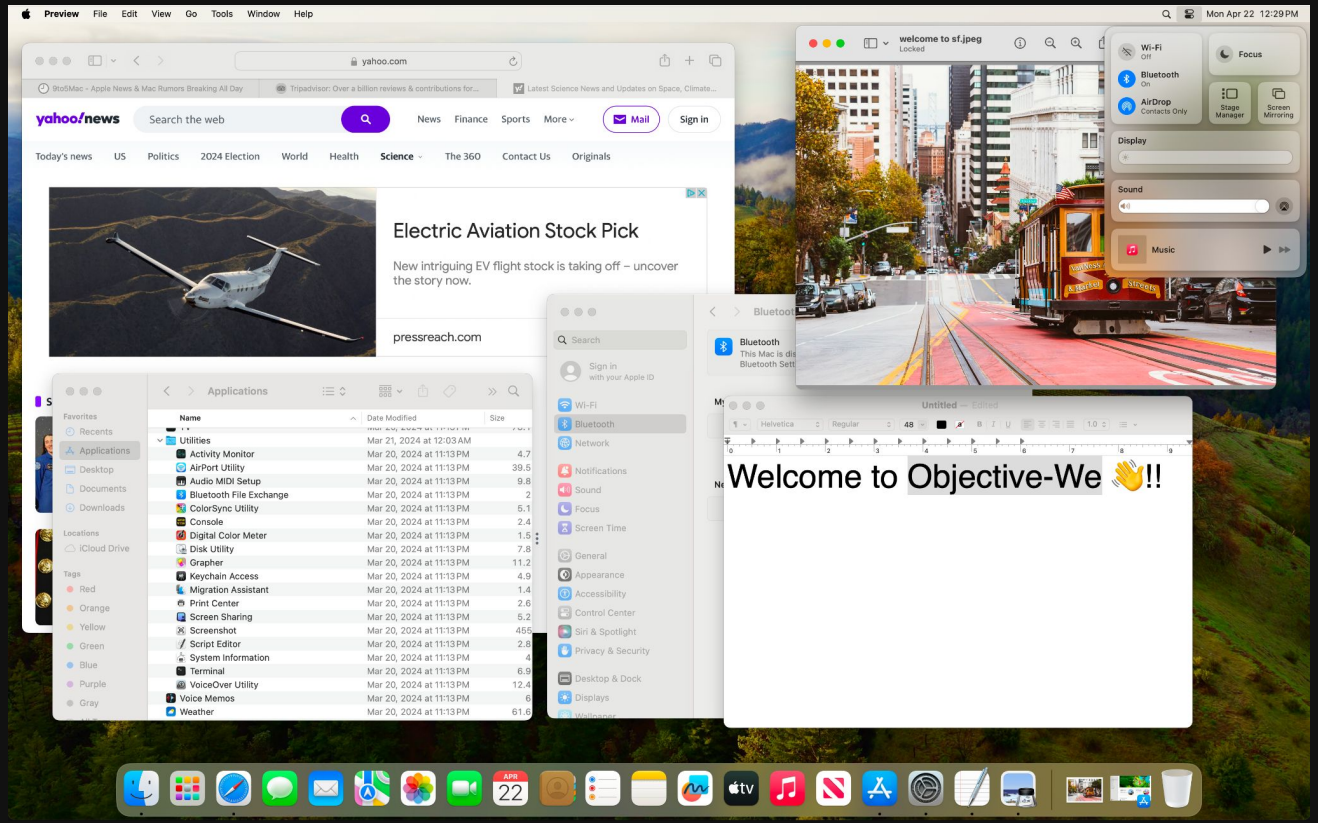
Let’s make sense of what we have…

What does a Mac do?

- **Runs software** and manages services
- Creates and **modifies files**
- Communicates over **networks** like WiFi and bluetooth
- Talks to **“accessories”** like the camera, trackpad, etc
- Enforces **privacy, management, and security** controls



Process | File | Network | “Accessories” | Controls



Telemetry

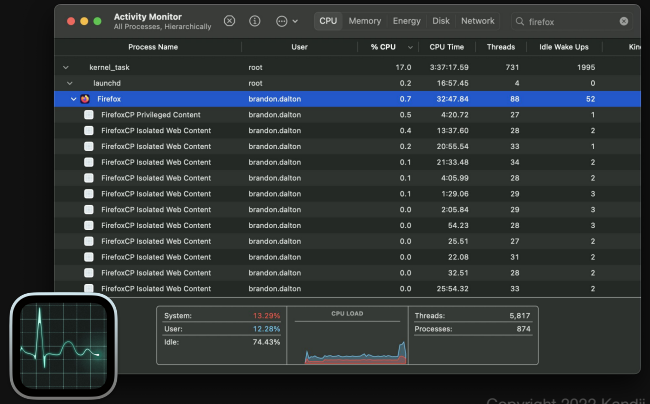
A set of events…

System roles and responsibilities

User space

Desktop → Aqua (vs. the iOS SpringBoard)

- Application services and frameworks
- Talks to the kernel w/“**System Calls** / Mach Traps”



Kernel space

At this level macOS can stop an action before it occurs

Kernel → XNU (XNU is Not Unix)

- Available, *secure*, performant
- Largely trusted code written by Apple
- Hybrid kernel – two *personalities*:
  - ***BSD*** (`execve()`, `posix_spawn()`, *etc*)
  - and ***Mach*** *(IPC and memory)*

Detecting unknown threats…

Behavioral detection

Events are… a notification that something *will or has* happened.

- Logging into to your Mac with Touch ID
- Talking over iMessage
- Launching an app
- Saving a PDF
- Sharing your screen

In other words?

Standard set of types minted *by Apple*. Events are *"messages"* containing info about what happened.

- ES_EVENT_TYPE_NOTIFY_AUTHENTICATION
- ES_EVENT_TYPE_AUTH_RENAME
- ES_EVENT_TYPE_AUTH_EXEC
- ES_EVENT_TYPE_AUTH_CREATE
- ES_EVENT_TYPE_NOTIFY_SCREENSHARING_ATTACH

They come from… Endpoint Security

- **APIs** offered by Apple that allow us to develop **userland** security **agents**
- Real time **eventing**
  - *Kernel and userspace*
  - Authorize and notification of system activity
- **Performance**
  - Powerful event muting capabilities
- Available to ***trusted* entities** (restricted entitlement)

Classes of events?

ES_EVENT_TYPE_NOTIFY_AUTHENTICATION

ES_EVENT_TYPE_AUTH_EXEC

Kernel Space (Mach / BSD)

ES_EVENT_TYPE_AUTH_*

User Space

ES_EVENT_TYPE_NOTIFY_*

What do they cover?

ES_EVENT_TYPE_AUTH_*

44 events (**30%**)

- AUTH_EXEC
- AUTH_OPEN
- AUTH_MMAP
- AUTH_SIGNAL
- AUTH_CREATE
- AUTH_RENAME

ES_EVENT_TYPE_NOTIFY_*

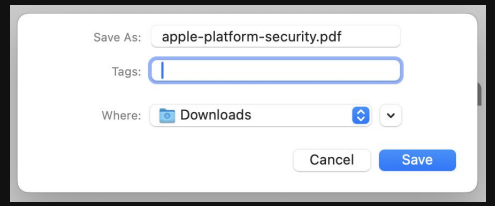
99 events (**70%**)

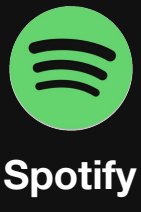
- NOTIFY_BTM_LAUNCH_ITEM_ADD
- NOTIFY_XPC_CONNECT
- NOTIFY_SCREENSHARING_ATTACH
- NOTIFY_XP_MALWARE_DETECTED

Event correlation

Every event was **“initiated”** by a process

Google Chrome





Spotify

`https://api.spotify.com/v1/albums/`

Terminal

`/usr/bin/curl`

Collection (a few examples)

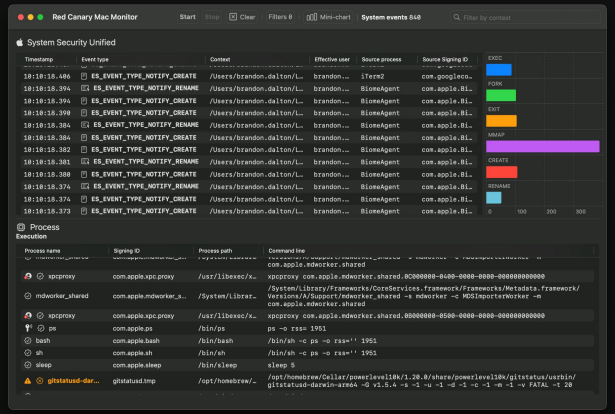
Depends on use case and workflow

/usr/bin/eslogger

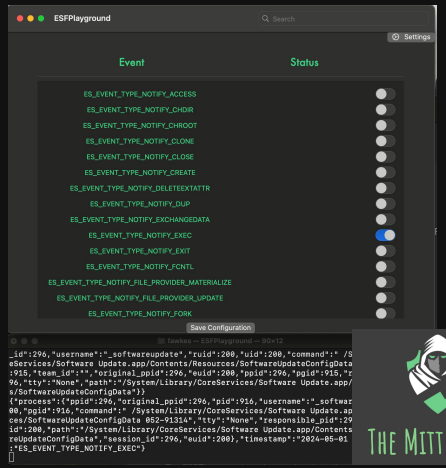
Process, file, and DNS monitor

Mac Monitor

ESFPlayground




```
% DNSMonitor.app/Contents/MacOS/DNSMonitor -json -pretty
[{
  "Process" : {
    "processPath" : "\/usr\/bin\/nslookup",
    "processSigningID" : "com.apple.nslookup",
    "processID" : 17493
  },
  "Packet" : {
    "Opcode" : "Standard",
    "QR" : "Query",
    "Questions" : [
      {
        "Question Name" : "objective-see.org",
        "Question Class" : "IN",
        "Question Type" : "A    "
      }
    ],
    "RA" : "No recursion available",
    "Rcode" : "No error",
    "RD" : "Recursion desired",
    "XID" : 63159,
    "TC" : "Non-Truncated",
    "AA" : "Non-Authoritative"
  }
},
...
```



THE MITTEN MAC

Building a detector

first… some questions to answer…

- What data do we have access to?
- Given a "threat"…
- What're the visible end behaviors?
- Using those behaviors write a "heuristic"
- *Test* it
- *Tune* it at scale

What's our data source?

Endpoint Security!

- What data do we have access to?
- Given a "threat"...
- What're the visible end behaviors?
- Using those behaviors write a "heuristic"
- *Test* it
- *Tune* it at scale

AMOS

A case study - Stealer

blog.kandji.io

Blog > Security > How AMOS m...

How AMOS macOS Stealer Avoids Detection

Sam Mayers & Christopher Lopez
Mar 2, 2024
10 min read

Atomic macOS Stealer (AMOS) was first spotted in early 2023. It's a powerful piece of malware that targets Apple users and tricks them into installing the software on their computers. The malware is sold via Telegram; as of January 20, 2024, the price was $3,000 a month.

Once installed, Atomic Stealer can exfiltrate an extensive amount of data, including keychain passwords, user documents, system info, cookies, browser data, credit card information, cryptocurrency wallets, and more.

```
ATOMIC MACOS STEALER (AMOS)
(The first and most famous stealer on MacOS)

SYSTEM:
- Collect notes from Notes
- Keychain (Dump all saved user passwords)
- SystemInfo (Complete system information)
- MacOS Password
- Hidden console when launching software

BROWSERS:
```

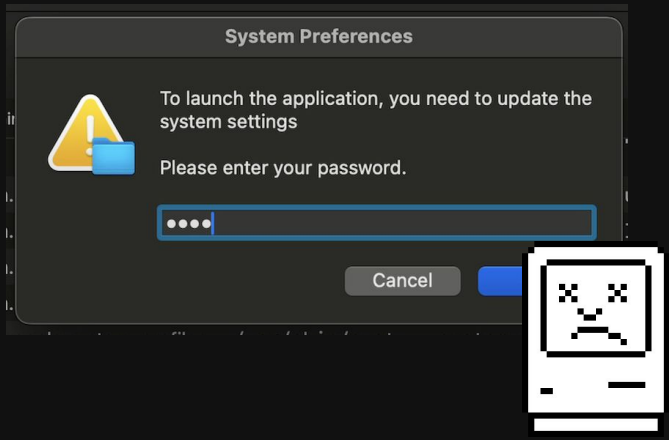
TELEGRAM

AMOS (MacOS Stealer)

ATOMIC MACOS STEALER

SYSTEM:
- Keychain (Export all passwords from apple keychain)
- SystemInfo (Full information about system)
- FileGrabber (Desktop, Documents)
- MacOS Password

BROWSERS:
- Chrome (Autofills, Passwords, Cookies, Wallets, Cards)
- Firefox (Autofills, Cookies)
- Brave (Cookies,Passwords,Autofills, Wallets, Cards)
- Edge (Cookies,Passwords,Autofills, Wallets, Cards)
- Vivaldi (Cookies,Passwords,Autofills, Wallets, Cards)
- Yandex (Cookies,Autofills, Wallets, Cards)
- Opera (Cookies,Passwords,Autofills,Wallets,Cards)
- OperaGX (Cookies,Passwords,Autofills,Wallets,Cards)

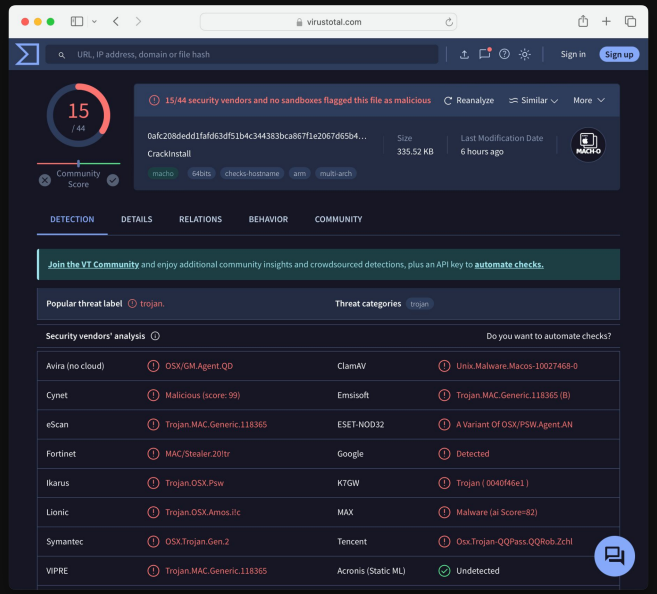


Kandji and SentinelOne

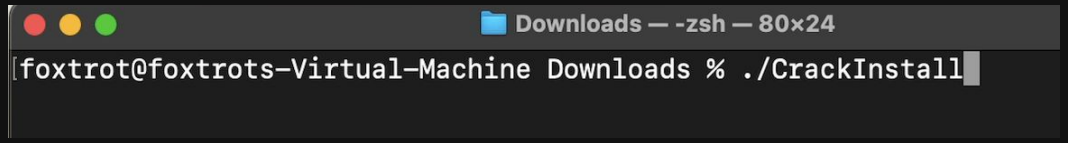
What's the threat?

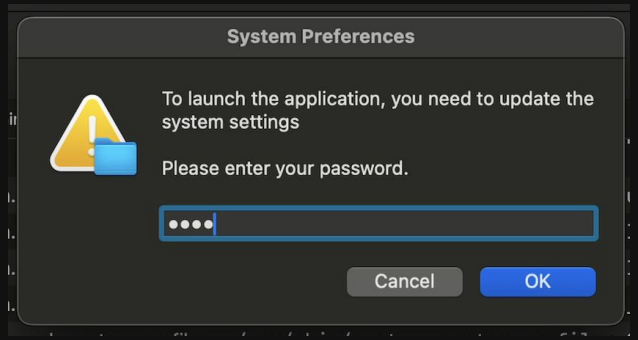
- What data do we have access to?
- Given a "threat"...
- What're the visible end behaviors?
- Using those behaviors write a "heuristic"
- *Test* it
- *Tune* it at scale

⚠️ Grab a sample



VirusTotal April 30, 2024





AMOS end behaviors

Event correlation

Timestamp	Event type	Context
12:10:06.409	ES_EVENT_TYPE_NOTIFY_EXIT	CrackInstall
12:10:02.386	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:10:02.381	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:10:02.346	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:47.320	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:44.507	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:44.449	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:44.447	ES_EVENT_TYPE_NOTIFY_CREATE	/Users/foxtrot/735085359/login-keychain
12:08:44.446	ES_EVENT_TYPE_NOTIFY_CREATE	/Users/foxtrot/735085359/password-entered
12:08:44.335	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:36.054	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.944	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.864	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.764	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.756	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.756	ES_EVENT_TYPE_NOTIFY_CREATE	/Users/foxtrot/735085359/Sysinfo.txt
12:08:35.752	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.673	ES_EVENT_TYPE_NOTIFY_FORK	CrackInstall
12:08:35.670	ES_EVENT_TYPE_NOTIFY_MMAP	CrackInstall
12:08:35.668	ES_EVENT_TYPE_NOTIFY_EXEC	./CrackInstall

Some correlated events include:

- **Adhoc** signed execution
- Executed for ~2 minutes
- Created 3 files in the user's home directory:

  `Sysinfo.txt`

  `password-entered`

  `login-keychain`

exec_target.group_leader.codesigning_type == .adhoc

AMOS end behaviors

Process group #1

Event Facts

Subtree: 6

zsh
- zsh
- CrackInst…
- CrackInst…
- sh
- bash

Metadata | Event correlation 4 | Process group 61 | Initiating process | JSON

Process group (61)

Process execution events in the same group as **bash** will show in this unified table.

Timestamp	Process name	Signing ID	Process p…	Command line
				"System Preferences" with icon caution default answer "" giving up after 30 with hidden answer'
12:08:36.054	sh	com.apple.sh	/bin/sh	sh -c osascript -e 'display dialog "To launch the application, you need to update the system settings \n\nPlease enter your password." with title "System Preferences" with icon caution default answer "" giving up after 30 with hidden answer'
12:08:35.946	dscl	com.apple.dscl	/usr/bi…	dscl /Local/Default -authonly foxtrot
12:08:35.945	bash	com.apple.bash	/bin/ba…	sh -c dscl /Local/Default -authonly foxtrot ""
12:08:35.944	sh	com.apple.sh	/bin/sh	sh -c dscl /Local/Default -authonly foxtrot ""
12:08:35.870	system_pro…	com.apple.syst…	/usr/sb…	system_profiler SPDisplaysDataType
12:08:35.868	bash	com.apple.bash	/bin/ba…	sh -c system_profiler SPDisplaysDataType
12:08:35.864	sh	com.apple.sh	/bin/sh	sh -c system_profiler SPDisplaysDataType
12:08:35.767	system_pro…	com.apple.syst…	/usr/sb…	system_profiler SPHardwareDataType
12:08:35.765	bash	com.apple.bash	/bin/ba…	sh -c system_profiler SPHardwareDataType
12:08:35.764	sh	com.apple.sh	/bin/sh	sh -c system_profiler SPHardwareDataType
12:08:35.758	sw_vers	com.apple.sw_…	/usr/bi…	sw_vers
12:08:35.757	bash	com.apple.bash	/bin/ba…	sh -c sw_vers
12:08:35.756	sh	com.apple.sh	/bin/sh	sh -c sw_vers
12:08:35.755	mkdir	com.apple.mkdir	/bin/mk…	mkdir /Users/foxtrot/735085359
12:08:35.753	bash	com.apple.bash	/bin/ba…	sh -c mkdir /Users/foxtrot/735085359
12:08:35.752	sh	com.apple.sh	/bin/sh	sh -c mkdir /Users/foxtrot/735085359
12:08:35.678	osascript	com.apple.osas…	/usr/bi…	osascript -e tell application "Terminal" to set visible of front window to false
12:08:35.675	bash	com.apple.bash	/bin/ba…	sh -c osascript -e 'tell application "Terminal" to set visible of front window to false'
12:08:35.673	sh	com.apple.sh	/bin/sh	sh -c osascript -e 'tell application "Terminal" to set visible of front window to false'
12:08:35.668	Crackl…	setup-555549…	/Users/…	./CrackInstall

Photos

Some behaviors include:

- Hides the terminal w/`osascript`
- Makes a directory w/`mkdir`
- Grabs the OS version w/`sw_vers`
- Grabs hardware information w/`system_profiler`
- Displays a dialog w/`osascript`

exec_target.correlated_events.has_exec_target_path ('/usr/bin/osascript')

AMOS end behaviors

Process group #2 (10s later)

Event Facts

bash

Subtree: 3

zsh

zsh

CrackInst…

Metadata | Event correlation 20 | Process group 61 | Initiating process | JSON

Process group (61)

Process execution events in the same group as **CrackInstall** will show in this unified table.

Timestamp	Process name	Signing ID	Process path	Command line
12:08:44.501	basename	com.apple.basena…	/usr/bin/…	basename /Library/Application Support/Script Editor/Templates/Droplets/Recursive Image File Processing Droplet.app
12:08:44.500	basename	com.apple.basena…	/usr/bin/…	basename /Library/Application Support/Script Editor/Templates/Droplets/Droplet with Settable Properties.app
12:08:44.499	basename	com.apple.basena…	/usr/bin/…	basename /Library/Application Support/Script Editor/Templates/Droplets/Recursive File Processing Droplet.app
12:08:44.498	basename	com.apple.basena…	/usr/bin/…	basename /Library/Image Capture/Support/LegacyDeviceDiscoveryHelpers/AirScanLegacyDiscovery.app
12:08:44.496	basename	com.apple.basena…	/usr/bin/…	basename /Applications/Safari.app
12:08:44.495	basename	com.apple.basena…	/usr/bin/…	basename /Library/Kandji/Kandji Agent.app
12:08:44.494	basename	com.apple.basena…	/usr/bin/…	basename /Library/Application Support/Kandji/Kandji Menu/Kandji Menu.app
12:08:44.493	basename	com.apple.basena…	/usr/bin/…	basename /Applications/Red Canary Mac Monitor.app
12:08:44.491	basename	com.apple.basena…	/usr/bin/…	basename /Applications/The Legend of Pirates Online.app
12:08:44.490	basename	com.apple.basena…	/usr/bin/…	basename /Applications/Suspicious Package.app
12:08:44.489	basename	com.apple.basena…	/usr/bin/…	basename /Applications/Utilities/Kandji Extension Manager.app
12:08:44.487	basename	com.apple.basena…	/usr/bin/…	basename /Applications/Kandji Self Service.app
12:08:44.486	basename	com.apple.basena…	/usr/bin/…	basename /Applications/Visual Studio Code.app
12:08:44.454	grep	com.apple.bzfgrep	/usr/bin/…	grep -v /System
12:08:44.454	mdfind	com.apple.mdfind	/usr/bin/…	mdfind kMDItemContentType == 'com.apple.application-bundle' && kMDItemKind != 'System'
12:08:44.454	xargs	com.apple.xargs	/usr/bin/…	xargs -I % basename %
12:08:44.451	bash	com.apple.bash	/bin/bash	sh -c echo ' Installed Apps: ' >> /Users/foxtrot/735085359/Sysinfo.txt && mdfind "kMDItemContentType == 'com.apple.application-bundle' && kMDItemKind != 'System'" | grep -v '/System' | xargs -I % basename "%" >> /Users/foxtrot/735085359/Sysinfo.txt
12:08:44.450	sh	com.apple.sh	/bin/sh	sh -c echo ' Installed Apps: ' >> /Users/foxtrot/735085359/Sysinfo.txt && mdfind "kMDItemContentType == 'com.apple.application-bundle' && kMDItemKind != 'System'" | grep -v '/System'

Searches for installed apps with:

`mdfind kMDItemContentType`

and `xargs` + `basename`

exec_target.process_path == "/usr/bin/xargs"

What're the behaviors?

- What data do we have access to?
- Given a "threat"...
- What're the visible end behaviors?
- Using those behaviors write a "heuristic"
- *Test* it
- *Tune* it at scale

Detecting AMOS

ES_EVENT_TYPE_AUTH_EXEC

```
exec_target.process_path == “/usr/bin/xargs”
&&
exec_target.group_leader.codesigning_type == .adhoc
&&
(
    exec_target.process_group.has_exec(‘/usr/bin/osascript’)
    ||
    exec_target.group_leader.correlated_events.has_mmap(‘/System/…/AppleScript.component/’)
    ||
    exec_target.group_leader.correlated_events.has_mmap(‘/System/…/JavaScript.component/’)
)
```

Testing!

- What **data** do we have access to?
- Given a **“threat”**...
- What’re the visible **end behaviors**?
- Using those behaviors write a “heuristic”
- ***Test*** it
- ***Tune*** it at scale

and lastly…maintain

Our options for testing?

(a) (**Ideally**) With the sample used to build the detector

(b) Engineer ***emulation*** and use that focusing on core behaviors

Process group (61)

Process execution events in the same group as **`CrackInstall`** will show in this unified table.

Timestamp	Process name	Signing ID	Process path	Command line
12:08:44.501	basename	com.apple.basena...	/usr/bin/...	basename /Library/Application Support/Script Editor/Templates/Droplets/Recursive Image File Processing Droplet.app
12:08:44.500	basename	com.apple.basena...	/usr/bin/...	basename /Library/Application Support/Script Editor/Templates/Droplets/Droplet with Settable Properties.app
12:08:44.499	basename	com.apple.basena...	/usr/bin/...	basename /Library/Application Support/Script Editor/Templates/Droplets/Recursive File Processing Droplet.app
12:08:44.498	basename	com.apple.basena...	/usr/bin/...	basename /Library/Image Capture/Support/LegacyDeviceDiscoveryHelpers/AirScanLegacyDiscovery.app
12:08:44.496	basename	com.apple.basena...	/usr/bin/...	basename /Applications/Safari.app
12:08:44.495	basename	com.apple.basena...	/usr/bin/...	basename /Library/Kandji/Kandji Agent.app
12:08:44.494	basename	com.apple.basena...	/usr/bin/...	basename /Library/Application Support/Kandji/Kandji Menu/Kandji Menu.app
12:08:44.493	basename	com.apple.basena...	/usr/bin/...	basename /Applications/Red Canary Mac Monitor.app
12:08:44.491	basename	com.apple.basena...	/usr/bin/...	basename /Applications/The Legend of Pirates Online.app
12:08:44.490	basename	com.apple.basena...	/usr/bin/...	basename /Applications/Suspicious Package.app
12:08:44.489	basename	com.apple.basena...	/usr/bin/...	basename /Applications/Utilities/Kandji Extension Manager.app
12:08:44.487	basename	com.apple.basena...	/usr/bin/...	basename /Applications/Kandji Self Service.app
12:08:44.486	basename	com.apple.basena...	/usr/bin/...	basename /Applications/Visual Studio Code.app
12:08:44.454	grep	com.apple.bzfgrep	/usr/bin/...	grep -v /System
12:08:44.454	mdfind	com.apple.mdfind	/usr/bin/...	mdfind kMDItemContentType == 'com.apple.application-bundle' && kMDItemKind != 'System'
12:08:44.454	xargs	com.apple.xargs	/usr/bin/...	xargs -I % basename %
12:08:44.451	bash	com.apple.bash	/bin/bash	sh -c echo ' Installed Apps: ' >> /Users/foxtrot/735085359/Sysinfo.txt && mdfind "kMDItemContentType == 'com.apple.application-bundle' && kMDItemKind != 'System'" | grep -v '/System' | xargs -I % basename "%" >> /Users/foxtrot/735085359/Sysinfo.txt

Emulation

Building the threat

Requirements

1. **Adhoc** signed
2. Calls `xargs` in same proc group
3. Leverages **OSA**

Game plan

1. Just compile locally!
2. Use something like `popen(...)`
3. Two variations
   a. API (`NSAppleScript` / `OSAKit`) and
   b. Command line

```
❯ ./amos_poc
Executing AppleScript via NSAppleScript API
Dialog displayed successfully.
Output: An example string to xargs!
File created successfully at /tmp/danger.txt
```

⚠️ Demo time

The result?

```
❯ ./amos_poc
Executing AppleScript via NSAppleScript API
Dialog displayed successfully.
zsh: killed     ./amos_poc
```

✅ AMOS resiliency!

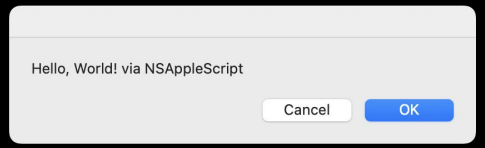
```
foxtrot — ObjectiveDefens ◂ sudo — 80×24
foxtrot@foxtrots-Virtual-Machine ~ % sudo ObjectiveDefense
✅ Terminated GID: 748
⚠️ Detected: AMOS-STEALER on event type: ES_EVENT_TYPE_AUTH_EXEC
```

Building a detector

- What data do we have access to?
- Given a "threat"...
- What're the visible end behaviors?
- Using those behaviors write a "heuristic"
- *Test* it
- *Tune* it at scale

and lastly…maintain

```
❯ sudo ./ObjectiveDefense
⚠️ Detected: AMOS-STEALER on event type: ES_EVENT_TYPE_AUTH_EXEC
⚠️ Detected: AMOS-STEALER on event type: ES_EVENT_TYPE_AUTH_EXEC
```




```
❯ ./amos_poc
Executing AppleScript via NSAppleScript API
Dialog displayed successfully.
zsh: killed     ./amos_poc
```

Thank you

Brandon Dalton / brandon.dalton@kandji.io