

# macOS Red Teaming on Corporate Scenarios

SPEAKER :

Ricardo Logan

# Agenda

---



## About Me:

Security Specialist with extensive experience in enterprise networks and enthusiastic on malware research, pentest and reverse engineering. I have been focused in the last years in research for vulnerability and malware for macOS environment.

I am part of the staff for some security conferences organizations such as H2HC (Hackers to Hackers Conference), BsidesSP and SlackShow/Slackzine Community.



Brazil

# Agenda

---

- 0x00 Motivation of Research
- 0x01 macOS Security (Default / Corp)
- 0x02 Hacking macOS target
- 0x03 POC Extra (Bypass TCC)
- 0x04 Native macOS Tools
- 0x05 Conclusion
- 0x06 Reference

# 0x00 Motivation of Research

The screenshot shows the official website for Transmission, a BitTorrent client. The page has a red header with the word "TRANSMISSION" in large white letters and a subtitle "A Fast, Easy, and Free BitTorrent Client". To the right of the title is a large, metallic-style download button icon. Below the header is a navigation menu with links for MAIN, ABOUT, DOWNLOAD, DEVELOPMENT, ADD-ONS, CONTENT, and SUPPORT. A "Feature Spotlight" section highlights Transmission 2.90, featuring a list of bullet points about its features: "Uses fewer resources than other clients", "Native Mac, GTK+ and Qt GUI clients", "Daemon ideal for servers, embedded systems, and headless use", "All these can be remote controlled by Web and Terminal clients", "Local Peer Discovery", and "Full encryption, DHT, µTP, PEX and Magnet Link support". There is also a "Learn More..." link. At the bottom of the page is a red footer bar containing a PayPal donation button, a "Donate to Transmission" link, copyright information ("Copyright 2009 - 2016 Transmission Project. All Rights Reserved"), a "Design by Stranded Design, implemented by Etoile" link, and a "Bandwidth provided by Comodo" link.

Empire  
Transfer  
2024

RustDoor  
2024

MetaStealer  
2024

Lockbit  
2023

CriptoMiner  
2023

Silver  
Sparrow  
2021

OSX/Linker  
2019

Mac Auto Fixer  
2018

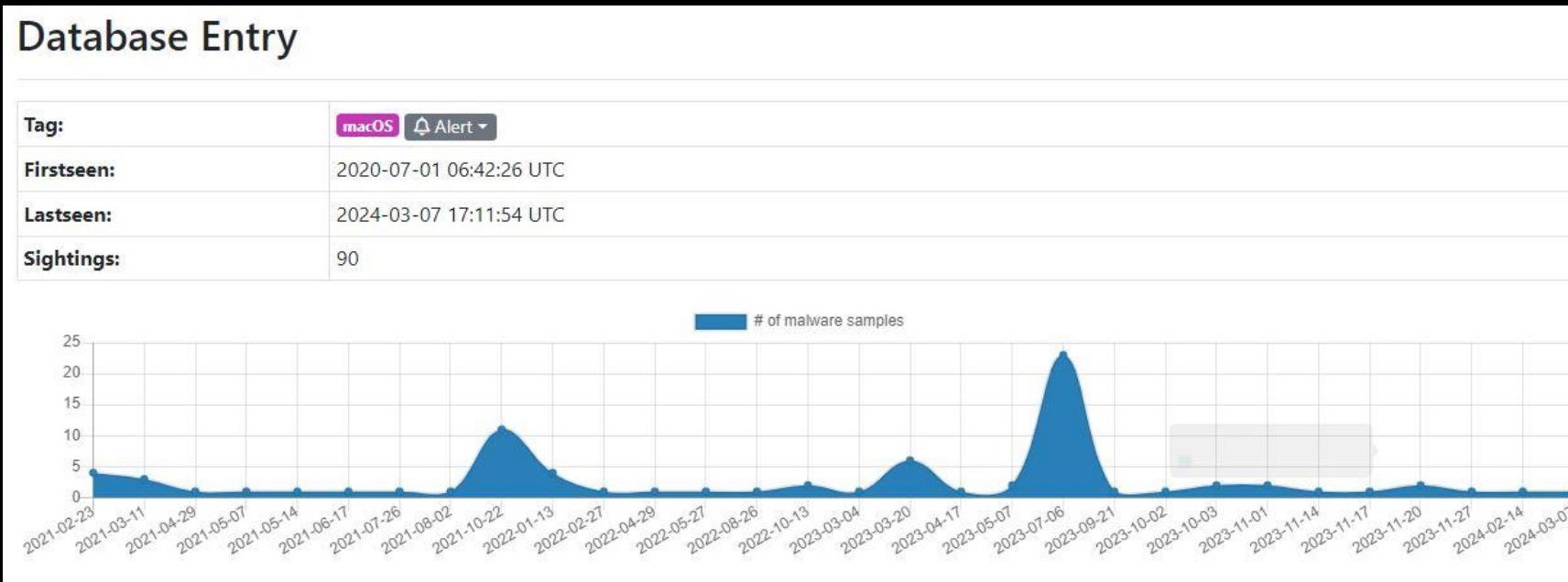
LoundMiner  
2019

Crossrider  
(OSX/Shalyer)  
2018

OSX/MaMi  
2018

Others....

# 0x00 Motivation of Research



Fonte: <https://bazaar.abuse.ch/browse/tag/macos>

### Malware Samples

The table below shows all malware samples that are associated with this particular tag (max 400).

Show 50 entries

Search:

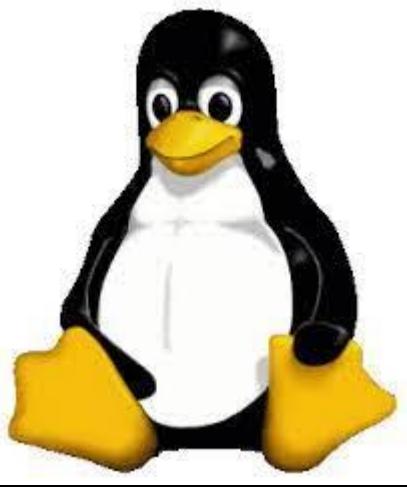
Firstseen (UTC)	SHA256 hash	Tags	Signature
2024-03-07 17:11:55	c802c94d0836039aa986e...	atomic stealer AtomicStealer machO macOS stealer	AtomicStealer
2024-02-14 19:27:12	08ff8a6500d623b062dce...	dmg macOS ViaCrackSite	n/a
2023-11-27 14:21:56	0db57feffa1f92816f9477f...	AMOS AmosStealer machO macOS OSX	AmosStealer
2023-11-20 13:43:25	a7bb346838db73301fe1...	mac Mach-O machO macOS Meterpreter OSX	n/a
2023-11-20 13:34:13	a6fd2f09eb81bf3afc83c4...	mac Mach-O machO macOS Meterpreter OSX	n/a
2023-11-17 07:12:52	a40d65307e67af3f18246...	AMOS AmosStealer ClearFake dmg macOS	AmosStealer
2023-11-14 18:26:31	60792845a1086b5c2ad7...	Adware dmg macOS	n/a
2023-11-01 17:28:39	3ea2ead8f3cec030906dc...	machO macOS	n/a
2023-11-01 17:24:07	2360a69e5fd7217e97712...	HLoader machO macOS	n/a
2023-10-03 11:26:19	b86e245d71f7a1056a7c5...	AMOS AmosStealer AtomicSteal macOS	n/a
2023-10-03 11:26:02	05ee833f167c4afa9e746...	AMOS AmosStealer AtomicSteal macOS	n/a
2023-10-02 10:21:33	135fc266af08a28fc7b2d3...	Mach-O machO macOS Meterpreter	Meterpreter
2023-09-21 19:50:52	6b0bde56810f7c0295d5...	AMOS Atomic macOS stealer xz	n/a
2023-07-06 15:19:57	016a1a4fe3e9d57ab0b2a...	machO macOS RealstStealer	n/a
2023-07-06 15:19:07	4b93ec3fd49c0111e8a11...	macOS pkg RealstStealer	n/a
2023-07-06 15:17:30	2c321b1416fb7226bffd1...	machO macOS RealstStealer	n/a

Fonte: <https://bazaar.abuse.ch/browse/tag/macos>

# 0x00 Motivation of Research

---

Is one of them safer?



?

# 0x01 macOS Security (Default / Corp)

Version	Release Date
Cheetah	2001
Puma	2001
Jaguar	2002
Panther	2003
Tiger	2005
Leopard	2007
Snow Leopard	2009
Lion	2011
Mountain Lion	2012
Mavericks	2013
Yosemite	2014
El Capitan	2015
Sierra	2016
High Sierra	2017
Mojave	2018
Catalina	2019
Big Sur	2020 ARM/Intel
Monterey	2021 (Supported) + ARM/Intel
Ventura	2022 (Supported) + ARM/Intel
Sonoma	2023 (Supported) + ARM/Intel
Sequoia	2024 Is coming...

SIP

FileVault

Xprotect

Secure Boot

Gatekeeper

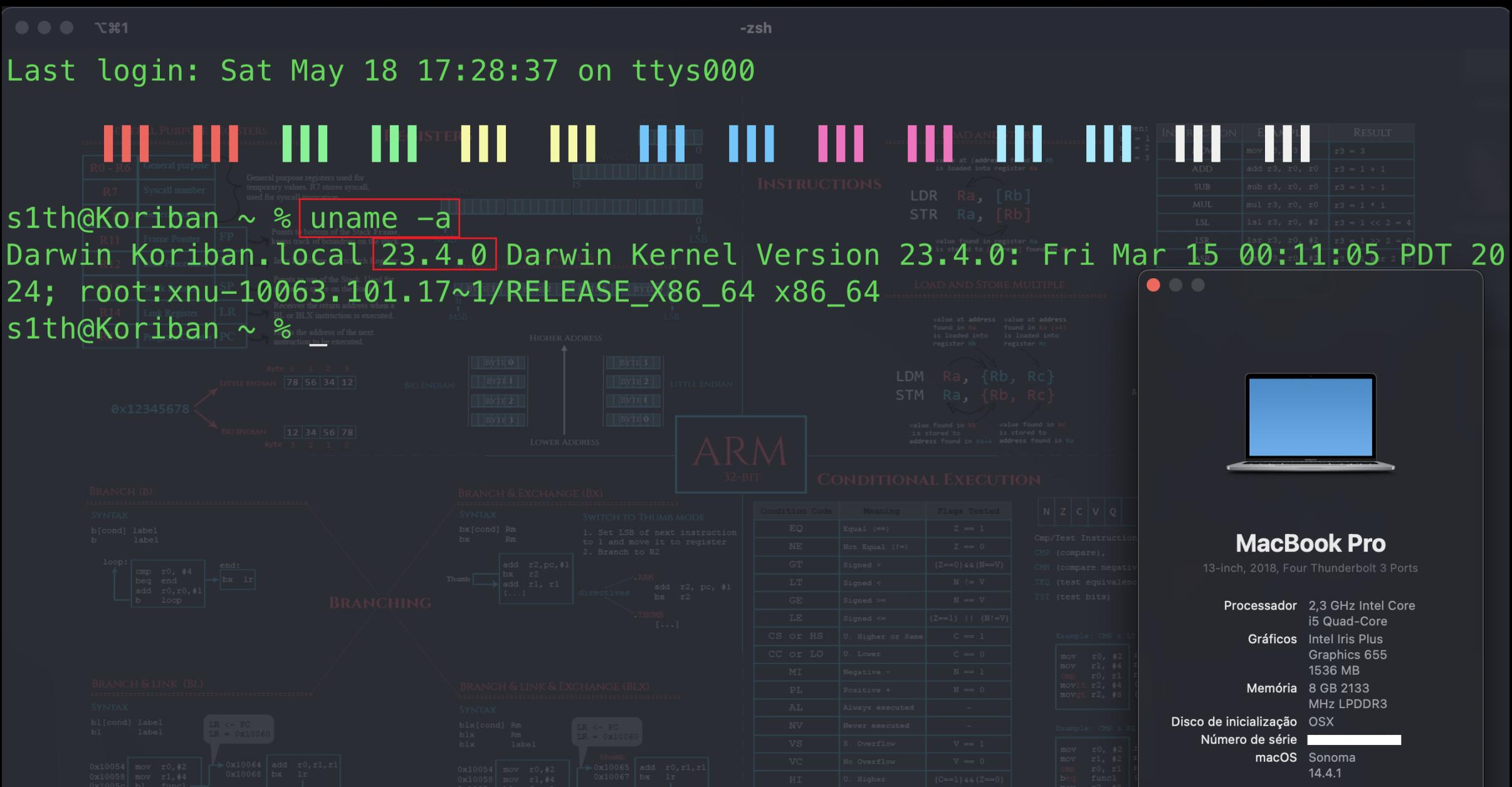
TCC

SSV



# 0x01 macOS Security (Default / Corp)

- A modern Operating system (macOS fanBoy LOL) Unix based;
- Kernel XNU is based on micro-kernel of NeXTSTEP (Mach) and kernel of BSD (FreeBSD);
- Lots of userland applications;
- macOS has grown significantly in market share;
- macOS supports both intel and ARM64(Silicon M1 M2 M3) chips.



Link XNU: <https://github.com/apple/darwin-xnu>

# 0x01 macOS Security (Default / Corp)

KEXT (Kernel Extension) files in macOS are components of the operating system's kernel that allow for the extension and enhancement of kernel functionality.

These files are essential for the operation of macOS and play a critical role in the interaction between hardware and system software.

```
$ sudo kextload payload.kext
```

```
l0gan@Zion-Inf3ct3d:[~][20:58:07]
→ $:>[sudo sqlite3 /var/db/SystemPolicyConfiguration/KextPolicy
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
sqlite> SELECT * from kext_policy;
EG7KH642X6|com.vmware.kext.vmcil1|VMware, Inc.|1
EG7KH642X6|com.vmware.kext.vmnet|1|VMware, Inc.|1
EG7KH642X6|com.vmware.kext.vmx86|1|VMware, Inc.|1
EG7KH642X6|com.vmware.kext.vmioplug.18.1.2|1|VMware, Inc.|1
2Y8XE5CQ94|com.kaspersky.kext.klif|1|Kaspersky Lab UK Limited|1
2Y8XE5CQ94|com.kaspersky.nke|1|Kaspersky Lab UK Limited|1
sqlite>
```

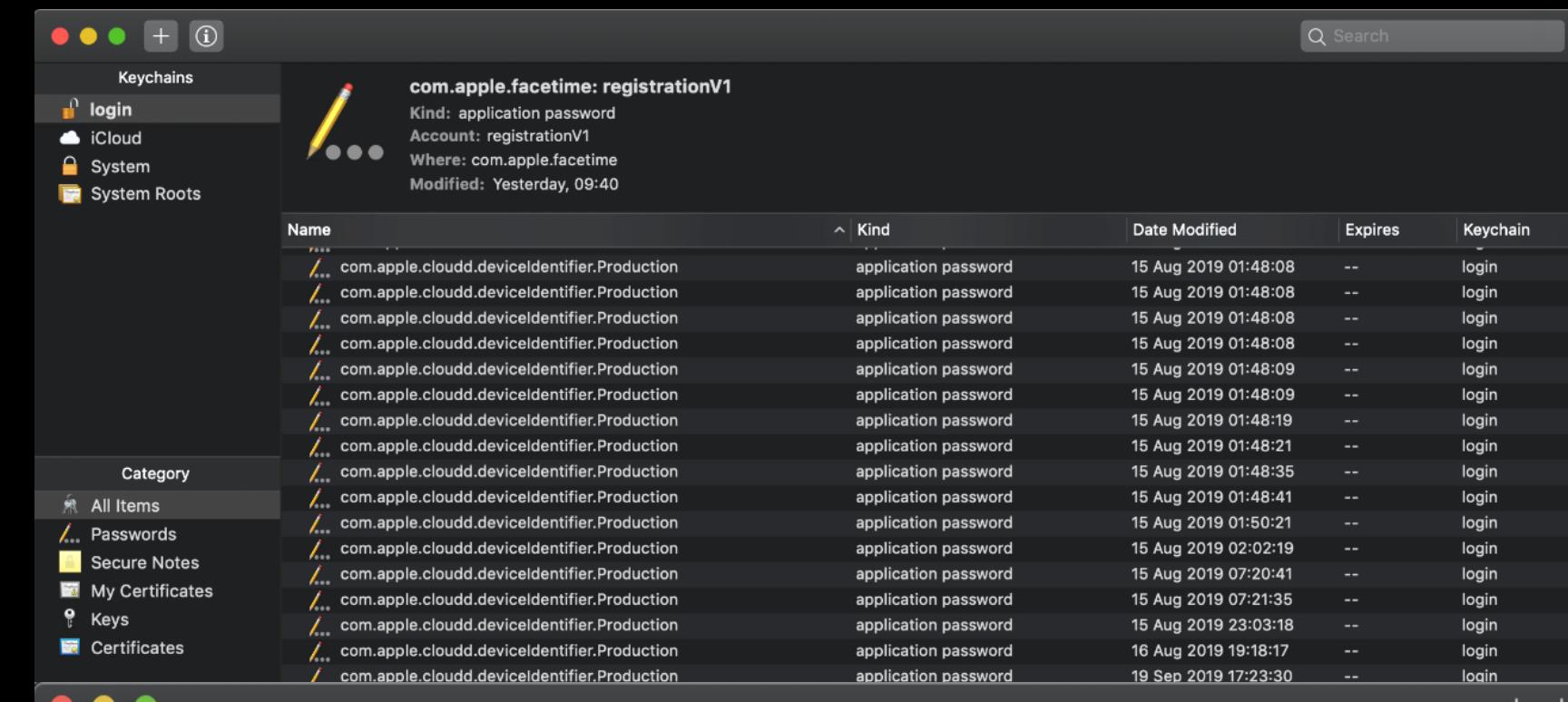
```
-zsh
s1th@Koriban Extensions % pwd
/System/Library/Extensions
s1th@Koriban Extensions %
s1th@Koriban Extensions % ls -l
total 0
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AFKACIPCKext.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AFTK_Kext.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AGXFirmwareKextG13GRTBuddy.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AGXFirmwareKextG13XRTBuddy.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AGXFirmwareKextG14GRTBuddy.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AGXFirmwareKextG14PRTBuddy.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AGXFirmwareKextG14XRTBuddy.kext
drwxr-xr-x@ 3 root wheel 96 16 Set 04:48 AGXFirmwareKextRTBuddy64.kext
```

# 0x01 macOS Security (Default / Corp)

The Keychain in macOS is a password and security key management system that provides secure storage and protection for sensitive information, such as:

- Passwords;
- Encryption Keys;
- Certificates;
- Authentication Information.

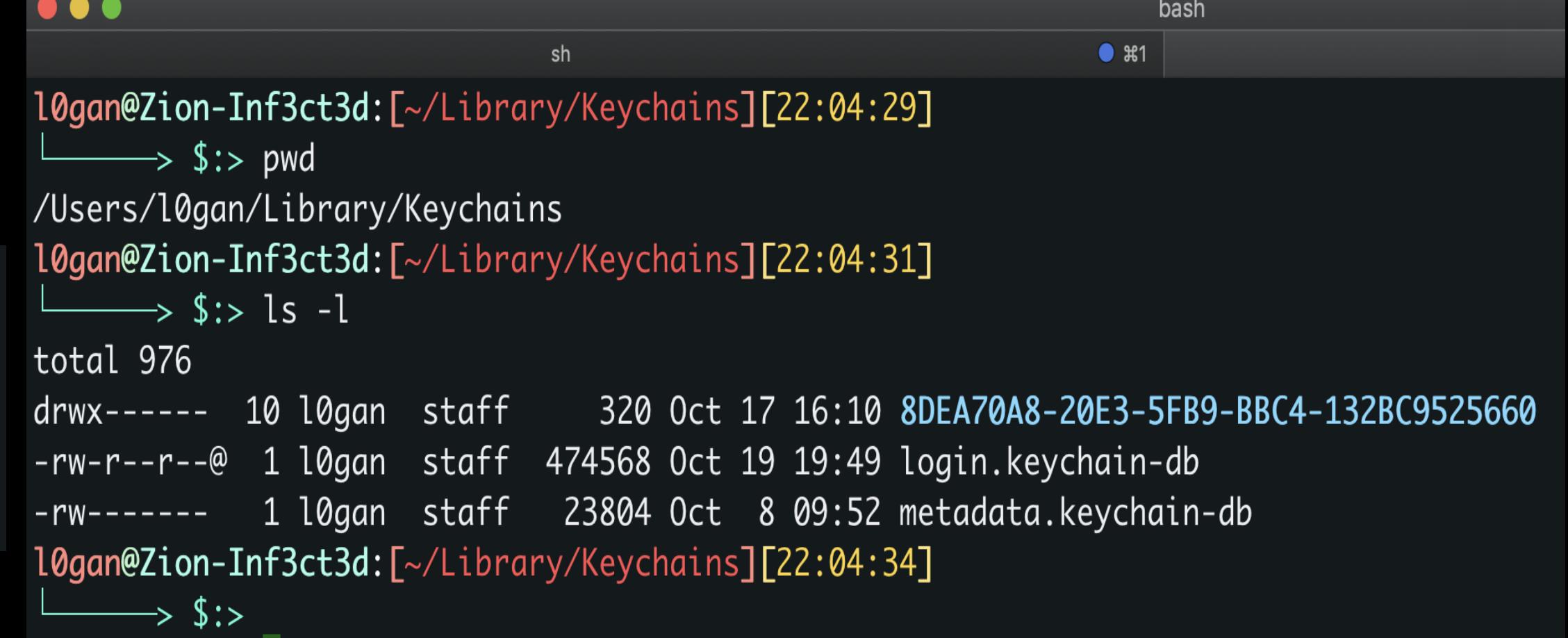
```
l0gan@Zion-Inf3ct3d:[~][16:15:17]
└──→ $:> security list-keychains
    "/Users/l0gan/Library/Keychains/login.keychain-db"
    "/Library/Keychains/System.keychain"
l0gan@Zion-Inf3ct3d:[~][16:15:26]
```



A screenshot of the macOS Keychain Access application. The window title is "com.apple.facetime: registrationV1". The left sidebar shows "Keychains" with "login" selected. Below it is a "Category" section with options: All Items, Passwords, Secure Notes, My Certificates, Keys, and Certificates. The main pane displays a table of items from the selected keychain. The columns are "Name", "Kind", "Date Modified", "Expires", and "Keychain". The table lists numerous entries, all of which are "application password" type items. The "Date Modified" column shows various dates from August 15, 2019, to September 19, 2019. The "Keychain" column consistently shows "login".



Keychain Access

A screenshot of a terminal window titled "bash". The prompt is "sh". The user is in the directory "/Users/l0gan/Library/Keychains". The user runs "ls -l" to list files. The output shows three files: "login.keychain-db" (permissions drwx-----, size 320, modified Oct 17 16:10), "login.keychain-db" (permissions -rw-r--r--@, size 474568, modified Oct 19 19:49), and "metadata.keychain-db" (permissions -rw-----, size 23804, modified Oct 8 09:52). The file "8DEA70A8-20E3-5FB9-BBC4-132BC9525660" is also mentioned in the terminal.

```
l0gan@Zion-Inf3ct3d:[~/Library/Keychains][22:04:29]
└──→ $:> pwd
/Users/l0gan/Library/Keychains
l0gan@Zion-Inf3ct3d:[~/Library/Keychains][22:04:31]
└──→ $:> ls -l
total 976
drwx----- 10 l0gan staff 320 Oct 17 16:10 8DEA70A8-20E3-5FB9-BBC4-132BC9525660
-rw-r--r--@ 1 l0gan staff 474568 Oct 19 19:49 login.keychain-db
-rw----- 1 l0gan staff 23804 Oct 8 09:52 metadata.keychain-db
l0gan@Zion-Inf3ct3d:[~/Library/Keychains][22:04:34]
└──→ $:> _
```

# 0x01 macOS Security (Default / Corp)

## Launchctl (LaunchDaemons / LaunchAgents)

Directories LaunchDaemons and LaunchAgents are used to manage and execute automated tasks and services in the system. These directories are essential for maintaining the persistence of tasks and programs. This can be of interest to an attacker.

```
### Launchctl (System)
/Library/LaunchDaemons/
/Library/LaunchAgents/
```

```
### Launchctl (User)
~/Library/LaunchAgents/
```

```
$ launchctl load file.plist
```

PLIST file is a settings file, also known as a "properties file," used by macOS applications.

```
l0gan@Zion-Inf3ct3d:[/Library/LaunchAgents][23:01:47]
└──> $:> pwd
/Library/LaunchAgents
l0gan@Zion-Inf3ct3d:[/Library/LaunchAgents][23:01:49]
└──> $:> ls -l
total 24
-rw-r--r-- 1 root wheel 674 Oct  7 22:25 com.bjango.istatmenus.agent.plist
-rw-r--r-- 1 root wheel 682 Oct  7 22:25 com.bjango.istatmenus.status.plist
-r-xr-xr-x 1 root wheel 582 Jun 25 14:24 com.kaspersky.kav.gui.plist
l0gan@Zion-Inf3ct3d:[/Library/LaunchAgents][23:01:51]
└──> $:> _
```

```
l0gan@Zion-Inf3ct3d:[/Library/LaunchDaemons][23:02:23]
└──> $:> pwd
/Library/LaunchDaemons
l0gan@Zion-Inf3ct3d:[/Library/LaunchDaemons][23:02:24]
└──> $:> ls -l
total 48
-rw-r--r-- 1 root wheel 632 Sep 19 14:55 com.apple.installer.osmessagetracing.plist
-rw-r--r-- 1 root wheel 584 Oct  7 22:25 com.bjango.istatmenus.daemon.plist
-rw-r--r-- 1 root wheel 557 Oct  7 22:25 com.bjango.istatmenus.fans.plist
-rw-r--r-- 1 root wheel 608 Aug 15 21:58 com.bjango.istatmenus.installerhelper.plist
-r-xr-xr-x 1 root wheel 1080 Jun 25 14:24 com.kaspersky.kav.plist
-rw-r--r-- 1 root wheel 382 Aug 15 07:31 org.wireshark.ChmodBPF.plist
l0gan@Zion-Inf3ct3d:[/Library/LaunchDaemons][23:02:26]
└──> $:> _
```

```
l0gan@Zion-Inf3ct3d:[~/Library/LaunchAgents][23:01:10]
└──> $:> pwd
/Users/l0gan/Library/LaunchAgents
l0gan@Zion-Inf3ct3d:[~/Library/LaunchAgents][23:01:12]
└──> $:> ls -l
total 24
-rw-r--r-- 1 l0gan staff 685 Aug 21 18:31 com.dropbox.DropboxMacUpdate.agent.plist
-rw-r--r--@ 1 l0gan staff 809 Oct  2 22:53 com.google.keystone.agent.plist
-rw-r--r--@ 1 l0gan staff 915 Oct  2 22:53 com.google.keystone.xpcservice.plist
l0gan@Zion-Inf3ct3d:[~/Library/LaunchAgents][23:01:16]
└──> $:> _
```

# 0x01 macOS Security (Default / Corp)

\$ PrivilegedHelperTools (FOLDER)

The `/Library/PrivilegedHelperTools` directory is crucial for executing tasks that require elevated privileges on macOS, ensuring that certain operations can be performed safely and efficiently with the necessary permissions.

```
●●● ~% Last login: Thu May 16 21:15:51 on ttys000
s1th@Koriban ~% cd /Library/PrivilegedHelperTools
s1th@Koriban PrivilegedHelperTools % ls -l
total 32752
drwxr-xr-x 1 root wheel 410496 15 Mar 2023 com.bjango.istatmenus.installerhelper
-rwxr--x-- 1 root wheel 1448016 5 Jun 2023 com.docker.socket
-rxr--r-- 1 root wheel 6025184 5 Jun 2023 com.docker.vmnetd
-rwxr-xr-x 1 root wheel 293200 14 Mai 20:54 com.microsoft.autoupdate.helper
-rwxr-xr-x 1 root wheel 1765328 11 Set 2023 com.microsoft.office.licensingV2.helper
-rxr--r-- 1 root wheel 6597936 12 Jun 2022 com.tristan.fseventstool
drwxr-xr-x@ 3 root staff 96 8 Abr 16:29 com.wacom.DataStoreMgr.app
drwxr-xr-x@ 3 root staff 96 8 Abr 16:29 com.wacom.IOManager.app
drwxr-xr-x@ 3 root staff 96 8 Abr 16:29 com.wacom.UpdateHelper.app
-rxr--r-- 1 root wheel 220576 31 Out 2023 us.zoom.ZoomDaemon
s1th@Koriban PrivilegedHelperTools %
```

Main Functions:

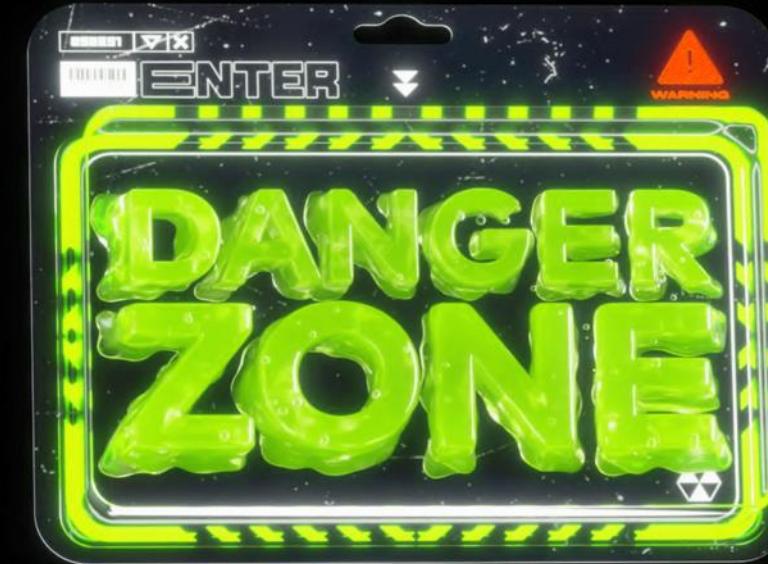
- Execute Tasks with Elevated Privileges
- Support for System and Application Services
- Software Installers, Security Software, and System Updates

# 0x01 macOS Security (Default / Corp)



And now?

Where should I go?



# 0x02 Hacking macOS Target

MITRE | ATT&CK®

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
8 techniques	8 techniques	16 techniques	10 techniques	23 techniques	15 techniques	22 techniques	7 techniques	14 techniques	16 techniques	8 techniques	13 techniques
Drive-by Compromise	Command and Scripting Interpreter (5)	Account Manipulation (1)	Abuse Elevation Control Mechanism (3)	Abuse Elevation Control Mechanism (3)	Adversary-in-the-Middle (2)	Account Discovery (2)	Exploitation of Remote Services	Adversary-in-the-Middle (2)	Application Layer Protocol (4)	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	Exploitation for Client Execution	Boot or Logon Autostart Execution (3)	Debugger Evasion	Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Encrypted for Impact	Data Destruction
External Remote Services	Inter-Process Communication (1)	Boot or Logon Autostart Execution (3)	Deobfuscate/Decode Files or Information	Credentials from Password Stores (4)	Browser Information Discovery	Lateral Tool Transfer	Audio Capture	Data Manipulation (3)	Exfiltration Over Alternative Protocol (3)	Data Manipulation (3)	Data Encrypted for Impact
Hardware Additions	Native API	Boot or Logon Initialization Scripts (3)	Execution Guardrails (1)	Exploitation for Defense Evasion	Device Driver Discovery	Remote Service Session Hijacking (1)	Automated Collection	Defacement (2)	Exfiltration Over C2 Channel	Defacement (2)	Data Manipulation (3)
Phishing (3)	Scheduled Task/Job (2)	Browser Extensions	Exploitation for Defense Evasion	Exploitation for Credential Access	File and Directory Discovery	Clipboard Data	Clipboard Data	Disk Wipe (2)	Exfiltration Over Other Network Medium (1)	Endpoint Denial of Service (4)	Disk Wipe (2)
Supply Chain Compromise (3)	Software Deployment Tools	Compromise Client Software Binary	Create or Modify System Process (2)	File and Directory Permissions Modification (1)	Network Service Discovery	Dynamic Resolution (3)	Data from Information Repositories	Firmware Corruption	Exfiltration Over Physical Medium (1)	Exfiltration Over Web Service (3)	Endpoint Denial of Service (4)
Trusted Relationship	System Services (1)	Create Account (2)	Event Triggered Execution (5)	Forge Web Credentials (1)	Network Share Discovery	Encrypted Channel (2)	Data from Local System	Inhibit System Recovery	Fallback Channels	Network Denial of Service (2)	Firmware Corruption
Valid Accounts (3)	User Execution (2)	Create or Modify System Process (2)	Hijack Execution Flow (2)	Input Capture (3)	Network Sniffing	Ingress Tool Transfer	Data from Network Shared Drive	Resource Hijacking	Exfiltration Over Physical Medium (1)	Non-Application Layer Protocol	Inhibit System Recovery
		Exploitation for Privilege Escalation	Impair Defenses (6)	Modify Authentication Process (2)	Password Policy Discovery	Multi-Stage Channels	Data from Removable Media	Service Stop	Exfiltration Over Web Service (3)	Scheduled Transfer	Resource Hijacking
		Event Triggered Execution (5)	Indicator Removal (7)	Multi-Factor Authentication Interception	Peripheral Device Discovery	Non-Standard Port	Email Collection (1)	System Shutdown/Reboot			System Shutdown/Reboot
		Hijack Execution Flow (2)	Masquerading (7)	Multi-Factor Authentication Request Generation	Permission Groups Discovery (2)						
		External Remote Services	Modifying Authentication Process (2)	Network Sniffing	Process Discovery						
		Process Injection	Obfuscated Files or Information (9)	OS Credential	Remote System Discovery						
		Hijack Execution Flow (2)									
		Scheduled Task/Job (2)									
		Modify									

Fonte: <https://attack.mitre.org/matrices/enterprise/macos/>

# 0x02 Hacking macOS Target

TA = Tactic Analysis

1. Initial Access (TA0001)
2. Execution (TA0002)
3. Persistence (TA0003)
4. Privilege Escalation (TA0004)
5. Defense Evasion (TA0005)
6. Credential Access (TA0006)
7. Discovery (TA0007)
8. Lateral Movement (TA0008)
9. Collection (TA0009)
10. Exfiltration (TA0010)
11. Command and Control (TA0011)

The screenshot shows a GitHub repository page for `redcanaryco / atomic-red-team`. The repository has 5 issues and 3 pull requests. The main content is a file named `macos-index.md` under the `Indexes-Markdown` directory. The file is generated by the Atomic Red Team doc generator from job=generate-docs branch=master [ci skip]. It contains a section titled "macOS Atomic Tests by ATT&CK Tactic & Technique" with a sub-section "defense-evasion". This section lists several AT&CK tactics:

- T1205.002 Socket Filters [CONTRIBUTE A TEST](#)
- T1027.009 Embedded Payloads [CONTRIBUTE A TEST](#)
- T1556.003 Modify Authentication Process: Pluggable Authentication Modules [CONTRIBUTE A TEST](#)
- T1564.012 File/Path Exclusions [CONTRIBUTE A TEST](#)
- T1222.002 File and Directory Permissions Modification: FreeBSD, Linux and Mac File and Directory Permissions Modification
  - Atomic Test #1: chmod - Change file or folder mode (numeric mode) [linux, macos]
  - Atomic Test #2: chmod - Change file or folder mode (symbolic mode) [linux, macos]

Fonte: <https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/Indexes-Markdown/macos-index.md>

## 0x02 Hacking macOS Target



Thinking in a scenario where the company has an employee who is interested in stealing confidential information and/or compromising the environment.



DROP  
MALICIOUS  
PAYLOAD

# 0x02 Hacking macOS Target

```
s1th@Koriban ~ %
s1th@Koriban ~ % system_profiler SPFirewallDataType
Firewall:
GENERAL PURPOSE REGISTERS
Registers
R0 - R6 General purpose
R7 Syscall Register
R8 - R10 General purpose
R11 Frame Pointer
R12 Intra Procedure call scratch Register
R13 Stack Pointer
R14 Link Register
R15 Program Counter PC
Registers
General purpose registers used for temporary values. R7 stores syscall, used for syscall invocation.

Mode: Block all incoming connections
Firewall Logging: Yes
Stealth Mode: No
Firewall Settings:
Points to bottom of the Stack Frame, keeps track of stack pointer.
Intra Procedure call scratch Register
Points to top of the Stack. Used for allocating space on the Stack.
receives the return address when a BL or BLX instruction is executed.
Holds the address of the next instruction to be executed.

ENDIANNES
Byte 0 1 2 3
Byte 0
Byte 1
Byte 2
Byte 3
MSB
LSB
HALFWORD
WORD
BYTE 0
BYTE 1
BYTE 2
BYTE 3
HIGHER ADDRESS
chacal@Hell [?] ~ [?]
chacal@Hell [?] ~ [?] netstat -an | grep "LISTEN"
tcp4      0      0      *.631          *.* LISTEN
tcp6      0      0      *.631          *.* LISTEN
tcp4      0      0      *.3031         *.* LISTEN
tcp6      0      0      *.3031         *.* LISTEN
tcp4      0      0      *.22           *.* LISTEN
tcp6      0      0      *.22           *.* LISTEN
tcp4      0      0      *.88           *.* LISTEN
tcp6      0      0      *.88           *.* LISTEN
tcp4      0      0      *.445          *.* LISTEN
tcp6      0      0      *.445          *.* LISTEN
tcp6      0      0      *.65275        *.* LISTEN
tcp4      0      0      *.65275        *.* LISTEN
chacal@Hell [?] ~ [?]
chacal@Hell [?] ~ [?]
```

## 0x02 Hacking macOS Target

---

```
macadmin in ~  
→ system_profiler SPApplicationsDataType  
Applications:
```

Hopper Disassembler v4:

```
Version: 4.3.11  
Obtained from: Identified Developer  
Last Modified: 28/12/17, 14:59  
Kind: Intel  
Signed by: Developer ID Application: Cryptic Apps (2AMA2753NF), Developer ID Certification Authority, Apple Root CA  
Location: /Applications/Hopper Disassembler v4.app
```

iMovie:

```
Version: 10.4  
Obtained from: App Store  
Last Modified: 16/12/23, 00:16  
Kind: Universal  
Signed by: Apple Mac OS Application Signing, Apple Worldwide Developer Relations Certification Authority, Apple Root CA
```

# 0x02 Hacking macOS Target

```
macadmin in ~
→ dsconfigad -show
Active Directory Forest      = zion.orion
Active Directory Domain      = zion.orion
Computer Account             = hell$
```

```
Advanced Options - User Experience
Create mobile account at login = Disabled
  Require confirmation        = Disabled
Force home to startup disk    = Enabled
  Mount home as sharepoint   = Enabled
Use Windows UNC path for home = Enabled
  Network protocol to be used = smb
Default user Shell            = /bin/bash
```

```
Advanced Options - Mappings
Mapping UID to attribute     = not set
Mapping user GID to attribute = not set
Mapping group GID to attribute = not set
Generate Kerberos authority  = Enabled
```

```
Advanced Options - Administrative
Preferred Domain controller   = not set
Allowed admin groups          = not set
Authentication from any domain = Enabled
Packet signing                 = allow
Packet encryption               = allow
Password change interval       = 14
Restrict Dynamic DNS updates   = not set
Namespace mode                 = domain
```

```
macadmin in ~
→ dscl . list /Users UniqueID | awk '{print $1}' | sort -n | tail -6
chacal
daemon
logan
nobody
root
s1th
macadmin in ~
```

```
-zsh
s1th@Koriban ~ %
s1th@Koriban ~ % whoami
s1th
s1th@Koriban ~ %
s1th@Koriban ~ % dscl . -read /Users/admin hint
dsAttrTypeNative:hint: Test-Lab
s1th@Koriban ~ %
s1th@Koriban ~ %

LOAD AND STORE
value at address: Found in memory
is loaded into register R0
LDR Ra, [Rb]
STR Ra, [Rb]
value found in register Ra
is stored to address Found in memory

INSTRUCTIONS
value at address: Found in memory
is loaded into register R0
LDR Ra, [Rb]
STR Ra, [Rb]
value found in register Ra
is stored to address Found in memory

INDIANS
value at address: Found in memory
is loaded into register R0
LDR Ra, [Rb]
STR Ra, [Rb]
value found in register Ra
is stored to address Found in memory

ARM
32-BIT
value found in R0
is stored to address Found in memory
value found in R0
is stored to address Found in memory

CONDITIONAL EXECUTION
value at address: Found in memory
is loaded into register R0
LDR Ra, [Rb]
STR Ra, [Rb]
value found in register Ra
is stored to address Found in memory
```

## 0x02 Hacking macOS Target

```
macadmin in ~/Desktop
→ ldapsearch -H ldap://zion.orion -x -D "macadmin@zion.orion" -W -b "dc=zion,dc=orion" -s sub "(&(objectClass=user)(objectCategory=person))"
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=zion,dc=orion> with scope subtree
# filter: (&(objectClass=user)(objectCategory=person))
# requesting: ALL
#
# Administrator, Users, zion.orion
dn: CN=Administrator,CN=Users,DC=zion,DC=orion
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Administrator
description: Built-in account for administering the computer/domain
distinguishedName: CN=Administrator,CN=Users,DC=zion,DC=orion
instanceType: 4
whenCreated: 20231026201813.0Z
whenChanged: 20240219234125.0Z
uSNCreated: 8196
memberOf: CN=Group Policy Creator Owners,CN=Users,DC=zion,DC=orion
memberOf: CN=Domain Admins,CN=Users,DC=zion,DC=orion
memberOf: CN=Enterprise Admins,CN=Users,DC=zion,DC=orion
memberOf: CN=Schema Admins,CN=Users,DC=zion,DC=orion
memberOf: CN=Remote Desktop Users,CN=BuiltIn,DC=zion,DC=orion
```

```
macadmin in ~/Desktop
→ ldapsearch -H ldap://zion.orion -x -D "macadmin@zion.orion" -W -b "dc=zion,dc=orion" -s sub "(&(objectClass=user)(objectCategory=person))" sAMAccountName | grep "sAMAccountName" | awk '{print $2}'
Enter LDAP Password:
requesting:
Administrator
Guest
DC01$
krbtgt
admin
DESKTOP-DD3R17L$
macadmin
HELL$
dico
macadmin2
teste02
teste03
teste04
teste05
teste01
macadmin in ~/Desktop took 3.2s
```

## Password Spray

```
### Check groups for user L0gan
```

```
l0gan@macLab % id l0gan
uid=413499013(l0gan) gid=1514433173(ZION\Domain Users) groups=1514433173(ZION\Domain Users),
502(awagent_enrolled),12(everyone),20(staff),62(netaccounts),501(awagent), 81465039(ZION\users teste)
```

## 0x02 Hacking macOS Target

### Disable Protection Solution (like EDR / XDR / DLP / SIEM / Proxy / Etc...)

```
### Turn off Security Solution (System)

sudo launchctl stop /Library/LaunchDaemons/com.X-Protection.plist
sudo launchctl unload /Library/LaunchDaemons/com.X-Protection.plist

sudo launchctl stop /Library/LaunchAgents/com.X-Protection.plist
sudo launchctl unload /Library/LaunchAgents/com.X-Protection.plist

### Turn off Security Solution (User)
launchctl stop ~/Library/LaunchAgents/com.X-Protection.plist
launchctl unload ~/Library/LaunchAgents/com.X-Protection.plist
```

## 0x02 Hacking macOS Target

```
[bash-3.2$  
[bash-3.2$ mdatp exclusion list  
=====  
No exclusions  
=====  
[bash-3.2$  
[bash-3.2$
```

```
[bash-3.2$  
[bash-3.2$ mdatp exclusion list  
=====  
Excluded folder  
Path: "/tmp/exclusion/"  
=====  
[bash-3.2$  
[bash-3.2$
```



Listing the exclusion folders of an antivirus can be dangerous, as it provides an attacker with information about which files and folders are intentionally excluded from antivirus scanning.

\*Note: In this example, we used Microsoft ATP, but the same check can be performed with other vendors.

# 0x02 Hacking macOS Target

-zsh ● #1 -zs

Last login: Thu Nov 2 00:20:33 on ttys000

```
s1th@Koriban ~ % defaults read com.apple.Terminal
{
    "Default Window Settings" = Basic;
    DefaultProfilesVersion = 1;
    HasMigratedDefaults = 1;
    "NSWindow Frame TTAppPreferences" = "387 519 667 565 0 0 1440 875 ";
    "NSWindow Frame TTWindow Basic" = "40 472 570 371 0 0 1440 875 ";
    ProfileCurrentVersion = "2.07";
    SecureKeyboardEntry = 0;
    "Startup Window Settings" = Basic;
    "TTAppPreferences Selected Tab" = 0;
    "Window Settings" = {
        "Basic" = {
            "Font" = {length = 267, bytes = 0x62706c69 73743030 d4010203 04050607 ... 00000000 000000cf };
            "FontAntialias" = 1;
            "FontSize" = 14;
            "FontWidthSpacing" = "1.004032258064516";
            "ProfileCurrentVersion" = "2.07";
            "name" = Basic;
            "type" = "Window Settings";
        };
    };
}
```

GENERAL PURPOSE REGISTERS

REGISTERS

HALF WORD

ENDIANNES

ARM 32-BIT

CONDITIONAL EXECUTION

LOAD AND STORE

LOAD AND STORE MULTIPLE

BRANCH & EXCHANGE (BX)

BRANCHING

INSTRUCTIONS

SWITCH TO THUMB MODE

Cond	Instruction	Description
EQ	BEQ Rm, Rm	Set Z == 1 if Rm == Rm
NE	BNE Rm, Rm	Set Z == 0 if Rm != Rm
GT	BGT Rm, Rm	Set Z == 1 if Rm > Rm
LT	BLT Rm, Rm	Set Z == 1 if Rm < Rm
GE	BGE Rm, Rm	Set Z == 1 if Rm >= Rm
LE	BLE Rm, Rm	Set Z == 1 if Rm <= Rm
CS or HS	BCS Rm, Rm	Set Z == 1 if C == 1
CC or LO	BCS Rm, Rm	Set Z == 0 if C == 0

# 0x02 Hacking macOS Target

```
s1th@Koriban ~ % mdfind xprotect
2023-10-23 23:37:37.597 mdfind[42875:1773091] [UserQueryParser] Loading keywords and predicates for locale "pt_BR"
2023-10-23 23:37:37.600 mdfind[42875:1773091] [UserQueryParser] Loading keywords and predicates for locale "pt"
/System/Library/Sandbox/Profiles/com.apple.XprotectFramework.AnalysisService.sb
/System/Library/FeatureFlags/Domain/XProtect.plist
/System/Library/PrivateFrameworks/XprotectFramework.framework/Versions/A/XPCServices/XProtectBehaviorService.xpc/Contents/Resources/com.apple.Xprotect
s1th@Koriban ~ %
s1th@Koriban ~ % mdfind -name stdlib.h
2023-10-23 23:37:02.270 mdfind[42842:1772471] [UserQueryParser] Loading keywords and predicates for locale "pt_BR"
2023-10-23 23:37:02.271 mdfind[42842:1772471] [UserQueryParser] Loading keywords and predicates for locale "pt"
/Library/Developer/CommandLineTools/SDKs/MacOSX12.3.sdk/usr/include/stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX12.3.sdk/usr/include/xlocale/_stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX13.3.sdk/usr/include/stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX13.3.sdk/usr/include/c++/v1/stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX13.3.sdk/usr/include/xlocale/_stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX13.3.sdk/usr/include/c++/v1/stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX14.0.sdk/usr/include/stdlib.h
/Library/Developer/CommandLineTools/SDKs/MacOSX14.0.sdk/usr/include/xlocale/_stdlib.h
/Users/s1th/Desktop/macadmin in ~
/Library/Frameworks/Pyt → mdfind password
/usr/local/Cellar/gcc/1 2024-02-19 21:58:51.193 mdfind[7407:715014] [UserQueryParser] Loading keywords and predicates for locale "en_US"
/usr/local/Cellar/gcc/1 2024-02-19 21:58:51.193 mdfind[7407:715014] [UserQueryParser] Loading keywords and predicates for locale "en"
/usr/local/Cellar/gcc/1 /Users/macadmin/test-pass.txt
/usr/local/Cellar/gcc/1 /Library/WebServer/share/httpd/manual/mod/quickreference.html.ko.euc-kr
/Library/Developer/Comm /Library/WebServer/share/httpd/manual/mod/directives.html.ko.euc-kr
s1th@Koriban ~ % /Library/WebServer/share/httpd/manual/mod/mod_auth_basic.html.ko.euc-kr
s1th@Koriban ~ % /Library/WebServer/share/httpd/manual/howto/auth.html.ko.euc-kr
/ Library/WebServer/share/httpd/manual/howto/htaccess.html.ko.euc-kr
/ Library/WebServer/share/httpd/manual/programs/ab.html.ko.euc-kr
/ System/Library/LaunchDaemons/com.apple.PasswordService.plist
/ System/Library/Frameworks/PDFKit.framework/Versions/A/Resources/Base.lproj/NewPasswordView.nib
/ System/Library/Frameworks/PDFKit.framework/Versions/A/Resources/Base.lproj/PDFPasswordAlert.nib
```

# 0x02 Hacking macOS Target

The terminal session shows the following commands and output:

```
s1th@Koriban Desktop %
s1th@Koriban Desktop % ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=400<CHANNEL_IO>
    ether 0c:cc:cc:cc:cc:cc
    inet 192.168.15.242 netmask 0xffffffff broadcast 192.168.15.255
        media: autoselect
        status: active
s1th@Koriban Desktop %
s1th@Koriban Desktop % python -m uploadserver
File upload available at /upload
Serving HTTP on 0.0.0.0 port 8000 (http://[::]:8000/) ...

```

Below the terminal is a screenshot of a web browser showing a file upload interface. The URL is 192.168.15.242:8000/upload. The page includes a file input field labeled "File Upload", a "Not Secure" warning, and a "Submit" button.

## File Upload

Choose Files

Token (only needed if server was started with token option):

Uploading

Success:

- + Onedrive / Dropbox / GoogleDrive / Etc...
- + HTTP / HTTPS
  - <https://tmpfiles.org>
  - <https://pastebin.com>
  - <https://transfer.sh>
  - <https://anonfile.net>
  - <https://gofile.io>
  - <https://send.cm>
- + C2
- + E-mail / FTP / FTPS
- + Mass Storage / Bluetooth
- + DNS Exfiltration
- + Others Techniques



```
### Listar pastas compartilhadas de uma maquina em dominio
$ smbutil view //dominio.alvo:10gan.user@dc-server-share
```

```
### Montar um compartilhamento do Server AD (Share)
$ mount_smbfs //10gan.user@dc-server-share/SYSVOL ~/Desktop/Files
```

# 0x03 POC Extra (Bypass TCC)

TCC(Transparency, Consent and Control) - Included in macOS since version 10.11 El Capitan

A bypass in macOS TCC is dangerous because it can compromise privacy, security, and system integrity by allowing apps or process to access sensitive resources without consent.

```
### TCC (Privacy Protections)
```

```
~/Desktop  
~/Documents  
~/Downloads  
iCloud Drive  
etc...
```

```
### TCC (Not Protected)
```

```
/tmp  
~/.ssh
```

```
Last login: Sat Oct 28 17:47:09 on ttys003  
l0gan@HELL ~ % ls -l  
total 0  
drwx-----+ 3 l0gan staff 96 Oct 25 22:54 Desktop  
drwx-----+ 3 l0gan staff 96 Oct 25 22:54 Documents  
drwx-----+ 4 l0gan staff 128 Oct 28 17:46 Downloads  
drwx-----@ 65 l0gan staff 2080 Oct 28 17:53 Library  
drwx----- 3 l0gan staff 96 Oct 25 22:54 Movies  
drwx-----+ 3 l0gan staff 96 Oct 25 22:54 Music  
drwx-----+ 4 l0gan staff 128 Oct 28 17:41 Pictures  
drwxr-xr-x+ 4 l0gan staff 128 Oct 25 22:54 Public  
l0gan@HELL ~ % cd Desktop  
l0gan@HELL Desktop % ls -l
```



# 0x03 POC Extra (Bypass TCC)

The screenshot shows a mobile application interface for Apple's Security Research. At the top, there is a navigation bar with the Apple logo and the text "Security Research". Below it is a blue button labeled "New Report". The main content area has a header "My Reports" with a back arrow and "Vulnerability o...". A message from "Product Security" is displayed, with a timestamp "13/11/2023, 14:44" and a checkmark icon. The message reads: "Thanks! We'll be in touch." Below this, another message from "Product Security" is shown, dated "16/11/2023, 12:30" and timestamped "há 20 horas". The message states: "After further review, we don't see any security implications to the behavior you're reporting as it's working as it was designed." At the bottom, there is a placeholder text "Write a comment.".

## Vulnerability found on TCC (Transparency Consent and Control)

- Access and modify of files protected by the system (TCC+ SSV+SIP).
- Bypass TCC component in macOS does not validate the use of the "open ." command must block the access to the folder from the terminal to the finder.

### Risk:

Drop file with new TCC.db with a malicious entry to disable some security protections that could be explored by another binary (like malware).

### Resolution:

In my opinion, TCC.db should have a flag created in the operating system based on the hardware and the operating system to ensure that it should not be possible to rewrite TCC.db by an installation generated by another machine.

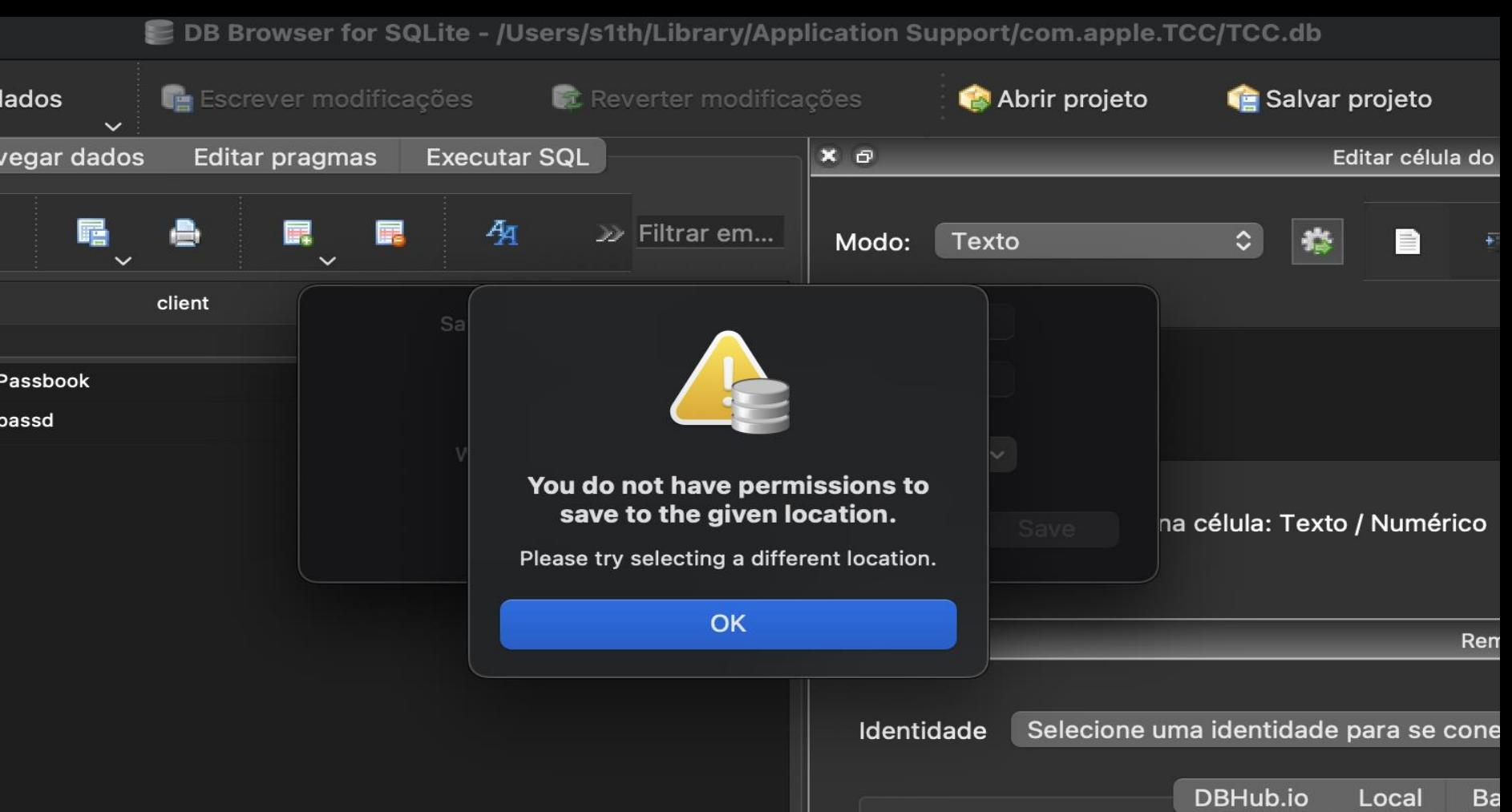
# 0x03 POC Extra (Bypass TCC)

The screenshot shows the DB Browser for SQLite interface. The top menu bar includes File, Edit, View, Tools, and Help. The toolbar has buttons for New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, and Attach Database. The main window has tabs for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. Below these are buttons for Create Table, Create Index, and Print. The Database Structure tab is active, showing a tree view of tables, indices, views, and triggers. Under Tables (6), there are entries for access, access\_overrides, active\_policy, admin, expired, and policies, each with its corresponding CREATE TABLE SQL. Under Indices (1), there is an entry for active\_policy\_id. Under Views (0) and Triggers (0), both are listed as 0. The right side of the screen shows an 'Edit Database Cell' panel with a mode dropdown set to 'Text', a gear icon, and a row of icons. It displays the value '1' and provides options to apply changes. Below this are labels for 'Type of data currently in cell', 'Size of data currently in table', and 'Remote'. The bottom part of the interface shows a detailed view of the 'access' table with columns service, client, client\_type, auth\_value, and auth\_time. The table contains 20 rows of data, all from the 'KTCCServiceLiverpool' service and client 'com.apple.accessibility.heard', with client\_type 0 and auth\_value 2.

Database Structure					
Table: <span>access</span> <span>Filter in any ...</span>					
	service	client	client_type	auth_value	auth_time
1	KTCCServiceLiverpool	com.apple.accessibility.heard	0	2	
2	KTCCServiceLiverpool	com.apple.imagent	0	2	
3	KTCCServiceLiverpool	com.apple.bird	0	2	
4	KTCCServiceLiverpool	com.apple.CloudDocs.iCloudDriveFileProv...	0	2	
5	KTCCServiceLiverpool	com.apple.securityd	0	2	
6	KTCCServiceLiverpool	com.apple.Safari	0	2	
7	KTCCServiceLiverpool	com.apple.upload-request...	0	2	
8	KTCCServiceLiverpool	com.apple.security.cuttlefish	0	2	
9	KTCCServiceLiverpool	com.apple.Passbook	0	2	
10	KTCCServiceLiverpool	com.apple.shortcuts	0	2	
11	KTCCServiceLiverpool	com.apple.findmy.findmylocateagent	0	2	
12	KTCCServiceLiverpool	com.apple.biomesyncd	0	2	
13	KTCCServiceLiverpool	com.apple.amsengagementd	0	2	
14	KTCCServiceUbiquity	com.apple.weather.widget	0	2	
15	KTCCServiceLiverpool	com.apple.stocks	0	2	
16	KTCCServiceLiverpool	com.apple.stocks.widget	0	2	
17	KTCCServiceUbiquity	com.apple.weather	0	2	
18	KTCCServiceLiverpool	com.apple.StatusKitAgent	0	2	
19	KTCCServiceLiverpool	com.apple.voicebankingd	0	2	
20	KTCCServiceLiverpool	com.appleMaps	0	2	

# 0x03 POC Extra (Bypass TCC)

```
s1th@Koriban ~ %
s1th@Koriban ~ % cd Library/Application\ Support/com.apple.TCC
s1th@Koriban com.apple.TCC % ls -l
total 0
ls: .: Operation not permitted
s1th@Koriban com.apple.TCC %
s1th@Koriban com.apple.TCC % csrutil status
System Integrity Protection status: enabled.
s1th@Koriban com.apple.TCC %
```



```
Last login: Thu May 9 15:09:53 on console
s1th@Koriban ~ %
s1th@Koriban com.apple.TCC %
s1th@Koriban com.apple.TCC % ls -l
total 160
drwxr-xr-x 6 s1th staff 192 24 Abr 02:52 AdhocSignatureCache
-rw-r--r--@ 1 s1th staff 81920 4 Mai 22:35 TCC.db
s1th@Koriban com.apple.TCC % csrutil status
System Integrity Protection status: disabled.
s1th@Koriban com.apple.TCC %
s1th@Koriban com.apple.TCC %
```

```
-zsh
Last login: Thu May 9 20:54:54 on ttys000
s1th@Koriban ~ %
s1th@Koriban com.apple.TCC %
s1th@Koriban com.apple.TCC % ls -l
total 168
drwxr-xr-x 25 root wheel 800 24 Abr 02:52 AdhocSignatureCache
-rw-r--r-- 1 root wheel 20480 9 Mai 15:09 REG.db
-rw-r--r-- 1 root wheel 65536 2 Mai 22:25 TCC.db
s1th@Koriban com.apple.TCC %
```

## 0x03 POC Extra (Bypass TCC)

POC ->



### Bypass

Replace the TCC.db file located in a protected folder: ~/Library/Application Support/com.apple.TCC with a new modified TCC.db.



Automator is an application developed by Apple Inc. for macOS, which can be used to automate repetitive tasks through point-and-click or drag and drop. Automator enables the repetition of tasks across a wide variety of programs, including Finder, Safari, Calendar, Contacts and others.

## 0x04 Native macOS Tools

---

csrutil -> Configure system security policies (SIP)  
codesign -> create and manipulate code signatures  
dsconfigad-> retrieves/changes configuration for Active Directory  
dscl -> Directory Service command line utility  
dtrace -> generic front-end to the DTrace facility  
fs\_usage -> report system calls and page faults related to filesystem activity in real-time  
Launchctl -> Manage and Inspect daemons, agents and XPC Services (PLIST)  
Lipo -> create or operate on universal files  
mdfind -> finds files matching a given query  
nettop -> Display updated information about the network  
networksetup -> configuration tool for network settings in System Preferences.  
nm -> display name list (symbol table)  
osalang -> information about installed OSA languages  
otool -> object file displaying tool like a objdump and ldd  
plutil -> property list utility (Convert plist files)  
security -> Command line interface to keychains and Security framework  
spctl -> SecAssessment system policy security (Gatekeeper)  
sysctl -> get or set kernel state

.....

## 0x05 Conclusion

---

- Are your SOC and Blue Team monitoring and protecting the company from attacks?
- Are the controls really effectives and well implemented?
- Are your systems updated and with last security patches?
- Do you make security tests (Pentest) recurrent in your macOS endpoints?
- Do you have a well oriented team or update service with the last published vulnerabilities?

**"A motivated attacker achieves his/her goals regardless of time"**

## 0x06 Reference

---

### Hacking is a way of life

**My First Publication of this research (Nullbyte 2019)**

<https://github.com/loganbr/Presentations/blob/main/2019%20-%20Nullbyte%20-%20macOS%20Hacking%20Tricks.pdf>

**Research about malwares (mach-o files) for macOS(H2HC 2016)**

<https://github.com/loganbr/Presentations/blob/main/2016%20-%20H2HC%20-%20R3v3rs1ng%20on%20Mach-0%20File%20%20Version%200.pdf>

**Cedric Owens - Gone Apple Pickin: Red Teaming MacOS Environments in 2021**

<https://www.youtube.com/watch?v=IiMladUbL6E&t=93s>

Objective-See / Objective ByTheSea

<https://objective-see.org>

<https://objectivebythesea.org>

# Research Group Focus on Apple Devices #Attackium



<https://discord.gg/tSpGtcUHVJ>

Thanks a Lot

Any Questions ?

[ricardologanbr@gmail.com](mailto:ricardologanbr@gmail.com)

Twitter: [@10ganbr](https://twitter.com/10ganbr)