

# Acknowledgments Contents at a glance

## Part I: Defensive Techniques and Technologies

### The missing documentation for Apple's proprietary security mechanisms

---

1. Authentication
2. MacOS Auditing
3. Authorization - KAuth
4. MACF - The Mandatory Access Control Framework
5. Code Signing
6. MacOS Software Restrictions
7. AppleMobileFileIntegrity
8. Sandboxing
9. MacOS System Integrity Protection
10. Privacy
11. Data Protection

## Part II: E pur si rompe

### A detailed exploration of vulnerabilities and their exploits

---

- @p0sixninja - For reaching out to me one sunny day, and inviting me to his IRC channel where other notable jailbreakers reside. Without him, I wouldn't have made such good and
12. MacOS: Classic vulnerabilities in 10.10.x and 10.11.x
  13. iOS: Jailbreaking
  14. evasi0n (6.x)
  15. evasi0n 7 (7.0.x)
  16. Pangu Axe (7.1.x)
  17. XuanYuan Sword (8.0-8.1)
  18. TaiG (8.0-8.1.2)
  19. TaiG (8.1.3-8.4)
  20. Pangu 9 (9.0.x)
  21. Pangu 9.3 (9.2-9.3.3)
  22. Pegasus (9.0-9.3.4)
  - 22½. Phoenix (9.0-9.3.5)
  23. I t mach\_portal (10.1.1)
  24. Yalu (10.0-10.2)
  25. async\_wake (11.0-11.1.2) and the QiLin Toolkit
- 

## Appendix A: MacOS Hardening Guide

## Appendix B: Darwin 18 security changes

# Table of Contents

## Part I: Defensive Techniques and Technologies

### The missing documentation for Apple's proprietary security mechanisms

---

0. About This Book.....	vii
1. Authentication.....	1
o Password files (*OS)	
o SetUID and SetGID (MacOS)	
o The Pluggable Authentication Module (MacOS)	
o opendirectoryd (MacOS)	
o LocalAuthentication.framework	
o Apple IDs	
o External Accounts	
2. MacOS Auditing .....	21
o Design	
o Audit Sessions	
o Implementation	
o System Call Interface	
o OpenBSM APIs	
o Auditing Considerations	
3. Authorization - KAuth.....	35
o Design	
o Implementation	
o KAuth Identity Resolvers (MacOS)	
o Debugging KAuth	
4. MACF - The Mandatory Access Control Framework.....	47
o Background	
o MACF Policies	
o MACF Callouts	
o MACF System Calls	
5. Code Signing.....	63
o The Code Signature Format	
o Code Signature Requirements	
o Code Signature Enforcement	
o Code Signing Weaknesses	
o Code Signing APIs	

6. macOS Software Restrictions .....	91
o authd	
o Gatekeeper	
o libquarantine	
o Quarantine.kext	
o Quarantine in Action	
o Managed Clients	
7. AppleMobileFileIntegrity .....	119
o AppleMobileFileIntegrity.kext	
o amfid	
o Provisioning Profiles	
o SIP Integration (MacOS)	
o AMFI Trust Cache	
o AMFI UserClient	
8. Sandboxing.....	151
o Evolution of the Sandbox	
o App Sandbox (MacOS)	
o Mobile Containers (*OS)	
o Sandbox Profiles	
o User Mode APIs	
o mac_syscall	
o Sandbox.kext	
o sandboxd (MacOS)	
o ContainerManagerd (*OS)	
9. macOS System Integrity Protection.....	183
o Design	
o Implementation	
o APIs	
10. Privacy.....	191
o Transparency, Consent and Control	
o Unique Device Identifiers	
o Differential Privacy (MacOS 12/iOS 10)	
11. Data Protection.....	205
o Partition Level Encryption (MacOS)	
o File Level Encryption (*OS)	
o Obliteration (*OS)	
o KeyBags (*OS)	
o The AppleKeyStore.kext	
o Keychains	

## Part II: Vulnerabilities and Exploitation

### A detailed exploration of both the bugs and their exploits

- 
- 12. MacOS: Classic vulnerabilities in 10.10.x and 10.11.x..... 233
    - o 10.10.1: ntpd
    - o 10.10.2: rootpipe
    - o 10.10.3: Racing kextd
    - o 10.10.4: DYLD\_PRINT\_TO\_FILE
    - o 10.10.5: DYLD\_ROOT\_PATH (muymacho)
    - o 10.11.0: tpwn
    - o 10.11.3: Mach race
    - o 10.11.4: Lokihardt's Trifecta (pwn2own 2016)
  - 13. iOS: Jailbreaking..... 255
    - o The Jailbreaking Process
    - o Kernel Patches
    - o Kernel Patch Protection
    - o Evolution of iOS Jailbreaks
  - 14. evasi0n (6.x)..... 283
    - o The Loader
    - o The Untether
    - o Kernel Mode Exploits
    - o Apple Fixes
  - 15. evasi0n 7 (7.0.x)..... 305
    - o The Loader
    - o The Untether
    - o Kernel Mode Exploits
    - o Apple Fixes
  - 16. Pangu Axe (7.1.x) ..... 323
    - o The Loader
    - o The Jailbreak Payload
    - o The Untether
    - o Kernel Mode Exploits
    - o Apple Fixes
  - 17. XuanYuan Sword (8.0-8.1)..... 339
    - o The Loader
    - o User Mode Exploits
    - o The Untether
    - o Apple Fixes

18. TaiG (8.0-8.1.2).....	349
o The Loader	
o User Mode Exploits	
o The Untether	
o Kernel Mode Exploits	
o Apple Fixes	
19. TaiG (8.1.3-8.4).....	373
o The Loader	
o User Mode Exploits	
o The Untether	
o Kernel Mode Exploits	
o Apple Fixes	
20. Pangu 9 (9.0.x).....	387
o The Loader	
o The Jailbreak Payload	
o Kernel Mode Exploit	
o Code Signing Bypass	
o The Untether	
o Pangu 9.1	
o Apple Fixes	
21. Pangu 9.3 (9.2-9.3.3).....	409
o Kernel Mode Exploit	
22. Pegasus (9.0-9.3.4).....	415
o Exploit Flow	
o Kernel Memory Read and KASLR Bypass	
o Kernel Arbitrary Memory Write	
o Persistence	
22½. Phoenix (9.0-9.3.5).....	427
o The Info Leak	
o mach_ports_register	
23. mach_portal (10.1.1).....	437
o Exploit Flow	
o Mach port name urefs handling	
o Crashing powerd	
o XNU UaF in set_dp_control_port	
o Disabling Protections	
o Apple Fixes	
o The AppleKeyStore.kext	
o Keychains	

24. Yalu (10.0-10.2).....	451
o Primitives	
o KPP Bypass	
o Post Exploitation	
o A deadly trap and recipe for disaster (10.2)	
▪ The exploit (Beer)	
▪ The exploit (Todesco & Grassi)	
25. async_wake (11.0-11.1.2) and the QiLin Toolkit .....	471
o Bypassing KASLR	
o Kernel Memory Corruption	
o Post-Exploitation: The QiLin Toolkit	

## Appendix

A. MacOS Hardening Guide .....	497
B. Darwin 18 security changes .....	511

But now, I am my own publisher. I had to buy back my own publishing rights (Katharsis, know), but that was a small price to break the shackles. The only restrictions I have at this point - if any would be self-imposed. I thus look for greater liberties. The first restriction - page count - is what forced the split of this work into three volumes. Each about or more than the word count of the first work. It follows, therefore, that security-minded individuals will be first to complain. For Part I, this means MacOS Administrators and Power Users, as well as system administrators and auditors. The parts where I've exposed and documented internal Apple vulnerabilities, certainly not what you'd find at [open-source.apple.com](http://open-source.apple.com). Indeed, most of the topics covered here - especially in the "OS internals" - are for code whizzes who will never venture outside Apple's networks and VPNs. As such, an amount of work has been spared in reverse engineering the daemon, framework and kernel extensions which provide security. It gets very low-level and technical, and may be unsuitable for young readers. Faint of heart, or with a deep aversion of Intel and/or ARMv8 assembly, Reverse Engineers and hacker-types, however, will hopefully find it *very much* to their liking. It provides not only a detailed of *why* the exploits and jailbreaks work, but also a step by step walkthrough with register sequences, and plenty of disassembly.