macOS Control Bypasses

Offensive Security
macOS Control Bypasses

Copyright © 2021 Offensive Security Ltd.

All rights reserved. No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from the author.
# Table of Contents

1. **Introduction to macOS**
   1.1 macOS System Overview
   1.2 High-Level OS Architecture
      1.2.1 Apple Proprietary File System (APFS)
      1.2.2 System Volume Protections
      1.2.3 Firmlinks
      1.2.4 Important Directories
      1.2.5 Property List Files
      1.2.6 Bundles
      1.2.7 The Application Bundle
      1.2.8 Other Bundles
      1.2.9 The dyld Shared Cache
   1.3 The Mach-O File Format
      1.3.1 Universal Binaries
      1.3.2 Mach-O Structure
      1.3.3 Mach-O Header
      1.3.4 Load Commands
      1.3.5 Mach-O Data
   1.4 Objective-C Primer
      1.4.1 Defining Classes, Objects, and Calling Methods
      1.4.2 Setter and Getter Methods
      1.4.3 Instance Variables
      1.4.4 Putting it Together
      1.4.5 Protocols
      1.4.6 Basic Types, Classes
      1.4.7 Blocks
      1.4.8 Working with Files
   1.5 Wrapping Up

2. **macOS Binary Analysis Tools**
   2.1 Command Line Static Analysis Tools
      2.1.1 codesign
      2.1.2 objdump
      2.1.3 jtool2
   2.2 Static Analysis with Hopper
2.2.1 Views in Hopper
2.2.2 Navigating the Code
2.2.3 External C Function Resolution

2.3 Dynamic Analysis
2.3.1 macOS Debugging Rules

2.4 The LLDB Debugger
2.4.1 Setting Breakpoints
2.4.2 Disassembling with LLDB
2.4.3 Reading and Writing Memory, and Registers
2.4.4 Modifying Code During Debugging

2.5 Debugging with Hopper
2.5.1 Setting Breakpoints
2.5.2 Starting the Debugger
2.5.3 Basic Controls and Functionality
2.5.4 Inspecting External Function Resolution

2.6 Tracing Applications with DTrace
2.6.1 Basic Terms
2.6.2 DTrace Example - Monitoring System Calls
2.6.3 DTrace Example - Monitoring Write Calls
2.6.4 DTrace Example - Creating Aggregation Info
2.6.5 DTrace Probes
2.6.6 System DTrace Scripts

2.7 Wrapping Up

3 The Art of Crafting Shellcodes
3.1 Writing Shellcode in ASM
3.1.1 Calling Conventions and Registers
3.1.2 System Call Numbering
3.1.3 Making Syscalls from Shellcode

3.2 Custom Shell Command Execution in Assembly
3.2.1 Planned Memory Layout
3.2.2 Putting Arguments on the Stack
3.2.3 Setting up the SySCALL
3.2.4 Putting it Together
3.2.5 Analyzing the Shellcode with dtrace
3.2.6 Analyzing the Shellcode in a Debugger
3.3 Making a Bind Shell in Assembly
  3.3.1 Creating a Socket
  3.3.2 In the Darkness Bind Them
  3.3.3 Listening on the Socket
  3.3.4 Accepting Incoming Connections
  3.3.5 Duplicating File Descriptors
  3.3.6 Executing /bin/zsh
  3.3.7 Putting the Bind Shell Together

3.4 Writing Shellcode in C
  3.4.1 Writing execv Shellcode in C
  3.4.2 Eliminating RIP Relative Addressing
  3.4.3 Eliminating Calls into the __stub Section
  3.4.4 Locating execv Pointer and Running the Code

3.5 Wrapping Up

4 Dylib Injection
  4.1 DYLD_INSERT_LIBRARIES Injection in macOS
    4.1.1 Performing an Injection
    4.1.2 Restrictions of DYLD_INSERT_LIBRARIES Injection
    4.1.3 Verifying Restrictions
  4.2 DYLIB Hijacking
    4.2.1 Dylib LOAD Commands
    4.2.2 Dylib Loading Process and Hijacking Scenarios
    4.2.3 Finding Vulnerable Applications
    4.2.4 Performing Dylib Hijacking
    4.2.5 Hijacking Dlopen

4.3 Wrapping Up

5 The Mach Microkernel
  5.1 Mach Inter Process Communication (IPC) Concepts
  5.2 Mach Special Ports
  5.3 Injection via Mach Task Ports
    5.3.1 Getting the SEND Right
    5.3.2 Writing to Remote Process Memory
    5.3.3 Starting a Remote Thread
  5.4 BlockBlock Case Study - Injecting execv Shellcode
    5.4.1 The Vulnerability
5.4.2 The BlockBlock Shellcode
5.4.3 Finding the Process ID
5.4.4 Putting it Together
5.5 Injecting a Dylib
  5.5.1 Promoting Mach Thread to POSIX Thread
  5.5.2 The Shellcode
5.6 Wrapping Up
6 Function Hooking on macOS
  6.1 Function Interposing
    6.1.1 Interposing printf
    6.1.2 Interposing ioctl Calls
  6.2 Objective-C Method Swizzling
    6.2.1 The Objective-C Runtime
    6.2.2 Objective-C Message Sending
    6.2.3 Using the Runtime API
    6.2.4 Hooking Objective-C Methods
    6.2.5 Sniffing a KeePass Master Password
  6.3 Wrapping Up
7 XPC Attacks
  7.1 About XPC
  7.2 The Low Level C API: XPC Services
  7.3 The Foundation Framework API
  7.4 Attacking XPC Services
    7.4.1 Typical Issues in XPC Services
    7.4.2 The API to Verify Client Signature Information
  7.5 Apple’s EvenBetterAuthorizationSample
    7.5.1 Authorization Concepts
    7.5.2 Authorization in EvenBetterAuthorizationSample
  7.6 CVE-2019-20057 - Proxyman Change Proxy Privileged Action Vulnerability
    7.6.1 CVE-2019-20057 - Root Cause Analysis
    7.6.2 CVE-2019-20057 - Exploitation
  7.7 CVE-2020-0984 - Microsoft Auto Update Privilege Escalation Vulnerability
    7.7.1 CVE-2020-0984 - Root Cause Analysis
    7.7.2 CVE-2020-0984 - Exploitation
  7.8 CVE-2019-8805 - Apple EndpointSecurity Framework Local Privilege Escalation
7.8.1 CVE-2019-8805 - Root Cause Analysis
7.8.2 CVE-2019-8805 - Exploitation
7.9 CVE-2020-9714 - Adobe Reader Update Local Privilege Escalation
  7.9.1 The Original Vulnerability and Exploit
  7.9.2 Analyzing the Patch
  7.9.3 CVE-2020-9714 - Exploitation
7.10 Wrapping Up

8 The macOS Sandbox
  8.1 Sandbox Internals
    8.1.1 Sandbox Containers
    8.1.2 Entering the Sandbox
    8.1.3 Disable Sandbox Through Interposing
  8.2 The Sandbox Profile Language (SBPL)
    8.2.1 SBPL Syntax
    8.2.2 Writing Custom SBPL Profiles
    8.2.3 System Sandbox Profiles
  8.3 Sandbox Escapes
  8.4 Case Study: QuickLook Plugin SB Escape
    8.4.1 The QuickLook Vulnerability
    8.4.2 Creating QuickLook Plugins
    8.4.3 Escaping the Sandbox - QuickLook
  8.5 Case Study: Microsoft Word Sandbox Escape
    8.5.1 The Word Vulnerability
    8.5.2 Escaping the Sandbox - Word
    8.5.3 The Patch
  8.6 Wrapping Up

9 Bypassing Transparency, Consent, and Control (Privacy)
  9.1 TCC Internals
    9.1.1 The Consent Databases
    9.1.2 User Intent
    9.1.3 Managing TCC
    9.1.4 TCC Summary
  9.2 CVE-2020-29621 - Full TCC Bypass via coreaudiod
    9.2.1 CVE-2020-29621 Vulnerability Analysis
    9.2.2 The Private TCC API
9.2.3 CVE-2020-29621 Exploitation

9.3 Bypass TCC via Spotlight Importer Plugins
9.3.1 The Spotlight Service
9.3.2 Vulnerability Analysis
9.3.3 Exploitation

9.4 CVE-2020-24259 - Bypass TCC with Signal to Access Microphone
9.4.1 CVE-2020-24259 Vulnerability Analysis
9.4.2 CVE-2020-24259 Exploitation

9.5 Gain Full Disk Access via Terminal
9.5.1 Exercises

9.6 Wrapping Up

10 Symlink and Hardlink Attacks
10.1 The Filesystem Permission Model
10.1.1 The POSIX Model
10.1.2 Flag Modifiers
10.1.3 The Sticky Bit
10.1.4 Access Control Lists
10.1.5 The macOS Sandbox

10.2 Finding Bugs
10.2.1 Static Analysis
10.2.2 Dynamic Analysis
10.2.3 Exploitable Conditions

10.3 CVE-2020-3855 - macOS DiagnosticMessages File Overwrite Vulnerability
10.4 CVE-2020-3762 - Adobe Reader macOS Installer Local Privilege Escalation
10.5 CVE-2019-8802 - macOS Manpages Local Privilege Escalation

10.6 Wrapping Up

11 Getting Kernel Code Execution
11.1 KEXT Loading Restrictions
11.2 Sample KEXT
11.3 The KEXT Loading Process
11.3.1 Initiating KEXT Load Requests
11.3.2 Entering kextd
11.3.3 KEXT Staging
11.3.4 KEXT Authentication and syspolicyd
11.3.5 Loading the KEXT, Entering XNU
11.4  CVE-2020-9939 - Unsigned KEXT Load Vulnerability
      11.4.1  The Vulnerability and the Exploit Plan
      11.4.2  Staging a KEXT with Symlink
      11.4.3  The Insecure Location Problem
      11.4.4  The Race to the Kernel
      11.4.5  Disabling SIP

11.5  CVE-2021-1779 - Unsigned KEXT Load Vulnerability
      11.5.1  The Patch
      11.5.2  Bypassing Code Signing
      11.5.3  Forget the Race, Meet Interactive Mode

11.6  Changes in Big Sur

11.7  Wrapping Up

12  macOS Penetration Testing

12.1  Small Step For Man

12.2  The Jail
      12.2.1  Prison Break
      12.2.2  Let’s Persist

12.3  I am (g)root
      12.3.1  Searching for Low-Hanging Fruit?

12.4  CVE-2020-26893 - I Like To Move It, Move It
      12.4.1  Periodic Scripts
      12.4.2  PAM Modules
      12.4.3  This is the Way

12.5  Private Documents - We Wants It, We Needs It
      12.5.1  CVE-2020-9934 - HOME Relocation

12.6  The Core

12.7  Wrapping Up