Mobile Hacking VI

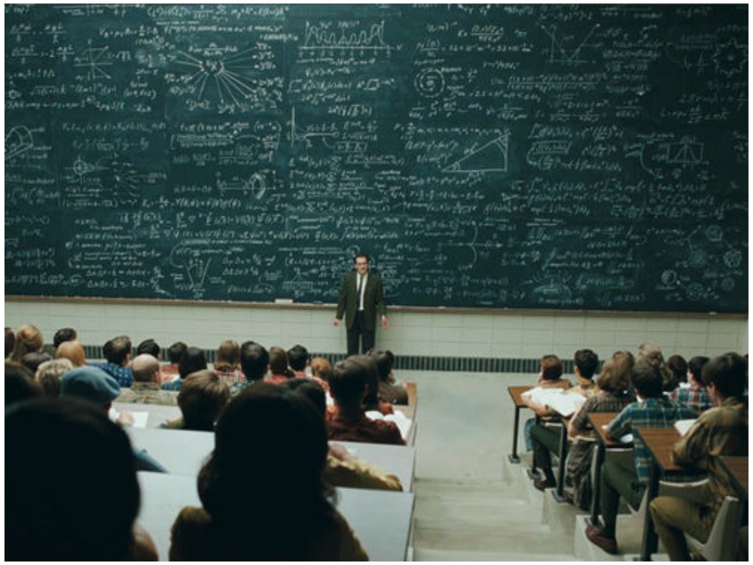
iOS Jailbreak

Crakun
Tues, 10-5-13 Day 3

Mobile Hacking VI
Class Agenda

- Part I:
  - Asserting the Goal
  - Status Update
  - Upcoming Classes
  - POP Quiz (your answers)

- Part II
  - Fuzzing and .mov files



Our Goal

Remember the Goal

The Goal of this class is to build an iOS untethered Jailbreak from scratch by creating a new

Community-based Jailbreak Team

Crakun Contact Info

Email:	crakun@m0bdev.com
Twitter:	crakun
Skype:	m0bdev (m0bdev is spelled with a zero)
IRC:	#openjailbreak on freenode

*-> If I am slow responding / or no response, ping me again because I may have missed it

HOtWAN

Status Updates

Updates: Submittals So Far

Submittals have been so far focus on iPhone5S

Userland: runs as mobile

2 SIGSEGV - 1 local , 1 remote

1 SIGABRT

Kernel:

1 local

Setting Standards: Submittal / Review Process for Bugs

Contact Me:

1. Detail Step-by-Step Method how / what you did so I can reproduce it
2. Tell me what you think it is?
3. Send me Crash Log
4. We will present to Bug Group for further review (Assigns it priority, reproducable, worth persuing
5. Submit to Vuln Group to see if it perhaps exploitable and to what degree
6. Submit to Exploit Group to see if it is exploit

Progress So Far with m0bdev

- Realizing need to focus on teaching Userland Fuzzing for junior m0bdev team
  - I'm stating to understand the levels of expertise from initial Volunteers
  - We will take this in steps with re-enforcement
- Advanced People
  - Sorry this takes time, but there is power in numbers
  - Correct me where I am wrong
  - Contact me so we can get started. Let's take a deep-dive into Kernel Hacking
- Getting a variety of Volunteers
  - Beginners
  - Coders
  - Exploit developers from other platforms

Upcoming Classes

- I am gauging the classes based on current Volunteers skillsets and needs

- I want to attract t0p talent

- So What's Next?
  - Fuzzing (Mutations and Generation based)
  - Fuzzing tools and scripts, monitoring
  - Understanding Crashdumps
  - You review Crashdumps (Bug Group)

Fuzzing .mov Files POP QUIZ

- What is the service that is provides for playing .mov files on the iPhone 4 /5 / 5S?

- Name some Existing Fuzzers that target files that we could use for .mov files?

- Have there been any documented vulnerabilities with .mov files?

- Explain to me about the file format of .mov files?

- How might we test fuzzed .mov files?

- How can we automate the process for testing?

Initial info on Bugs

- Consistent exploit code likely
- Inconsistant exploit code Likely
- Functioning exploit code unlikely

- **<ep0k>** Not all crashes are interesting : aborts, timeouts or out of memory kind of crashes are useless.

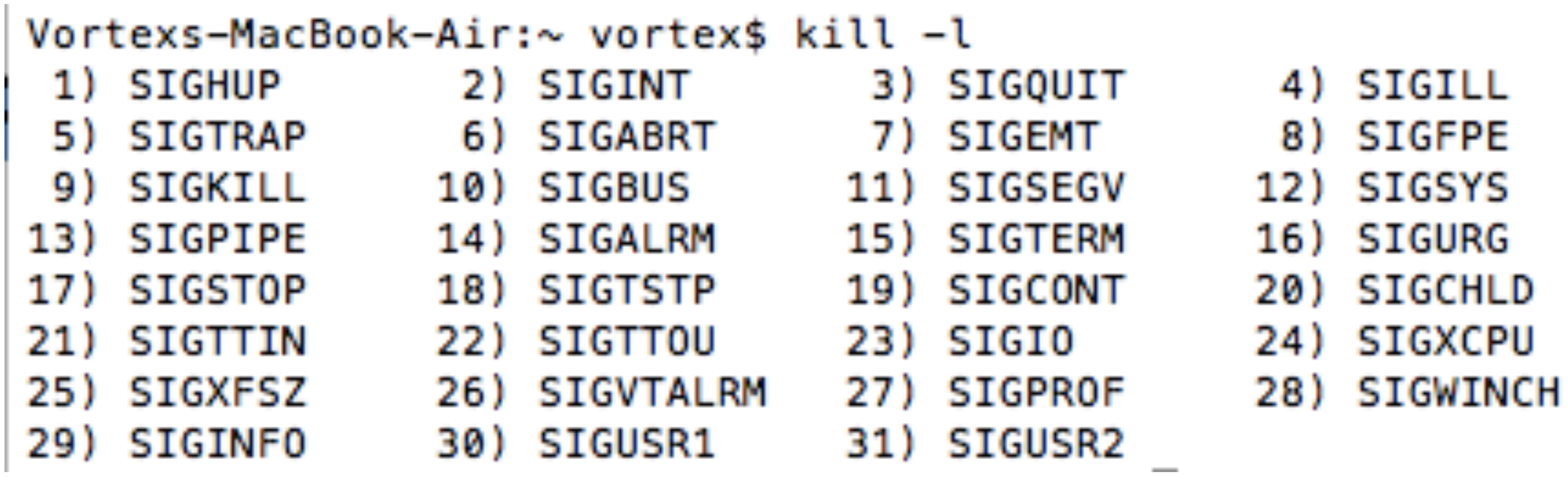
- Verify the crash dump in Settings / General / About / Diagnostics & Usage / Diagnostic & Usage Data that the crash report you created is of Exception Type SIGILL, SIGBUS or SIGSEGV. This are useful types.

Signals

https://developer.apple.com/library/ios/documentation/system/conceptual/manpages_iphoneos/man3/signal.3.html

Signals allow the manipulation of a process from outside its domain, as well as allowing the process to manipulate itself or copies of itself (children). There are two general types of signals: those that cause termination of a process and those that do not. Signals which cause termination of a program might result from an irrecoverable error or might be the result of a user at a terminal typing the `interrupt' character.

Most signals result in the termination of the process receiving them, if no action is taken; some signals instead cause the process receiving them to be stopped, or are simply discarded if the process has not requested otherwise. Except for the SIGKILL and SIGSTOP signals, the **signal**() function allows for a signal to be caught, to be ignored, or to generate an interrupt. These signals are defined in the file <signal.h>:



```
Vortexs-MacBook-Air:~ vortex$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
 9) SIGKILL     10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGURG
17) SIGSTOP     18) SIGTSTP     19) SIGCONT     20) SIGCHLD
21) SIGTTIN     22) SIGTTOU     23) SIGIO       24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGINFO     30) SIGUSR1     31) SIGUSR2 _
```

Signal 11– SIGSEGV

– Always the hottest because it means we're already dealing with a memory read/write issue

- We want a SEGV based on a write operation

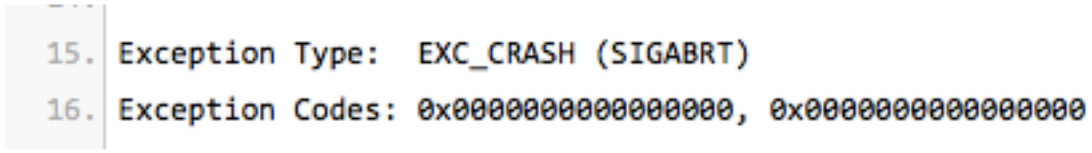
Ideas toward popular heap exploitation techniques

- Clobber function pointer on Heap with overwrite
- Get that function called

SIGILL

The **SIGILL** signal is raised when an attempt is made to execute an invalid, privileged, or ill-formed instruction. **SIGILL** is usually caused by a program error that overlays code with data or by a call to a function that is not linked into the program load module.

SIGABRT



```
Exception Type:   EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000
```

abort() sends the calling process the SIGABRT signal, this is how abort() basically works.

abort() is usually called by library functions which detect an internal error or some seriously broken constraint. For example malloc() will call abort() if its internal structures are damaged by a heap overflow.

Often indicate heap issues when fuzzing

- But that’s why were here, so back to the QuickTime crash from my Mac
  - **SIGFPE** is the signal sent to computer programs that perform erroneous arithmetic operations on POSIX compliant platforms. The symbolic constant for SIGFPE is defined in the header file signal.h. Symbolic signal names are used because signal numbers can vary across platforms.
- So….. is it exploitable or not? Probably not.
  - Why? Because the chance for memory corruption, and thus execution redirection looks minimal. See example:

```
int main() {
  int x = 42/0;
  return 0; /* Never reached */
}
```

- I hear that. You really always need to reverse each exception to gain full understanding of the crash
  - ***GET TO THE ROOT CAUSE VIA REing***
    - If the value is used to influence allocation or similar functions, it could still have a chance of being exploitable
- On the other hand, no invalid memory access is directly happening here, so the chances that it would be exploitable seems very thin

Null pointer derefence

Null-pointer dereference issue can occur through a number of flaws, including race conditions, and simple programming omissions.

They can be exploitable. Ask Mark Dowd

Good coding practice: Before using a pointer, ensure that it is not equal to NULL

- Find/Create Fuzzer http://caca.zoy.org/wiki/zzuf
- Find sample files
- Generate semi-invalid samples
- Fuzz Application
  - Log such that you know which file caused fault
- Review Faults
- AptDiff(GoodFile , Mutated file)
- Change each difference back, to identify the byte that caused the fault
- **Verify/Debug to determine exploitability**

HOtWAN

Homework Assignment

- Review slide deck.
  - Think about what you can do. Let me know.
  - Contact me if you want to share bugs, confirmed vulns, exploits via Email or Skype
- Continue on:
  - Fuzzing .mov files and undestand POP quiz questions
  - What are the vectors for attacking iPhones remotely (SMS, MMS, .pdf, .xls, etc.)
  - What files formats are supported for rendering / parsing from MobileSafari
- Spread the Word. We need Volunteers for this Jailbreak

Open Floor Discussion

APPENDIX

Mobile Hacking VI Class Schedule

- Weekly Class is migrating to Saturdays, 6am PST
- Next meeting (day 4): Sat, Oct 12, 2013 6am PST

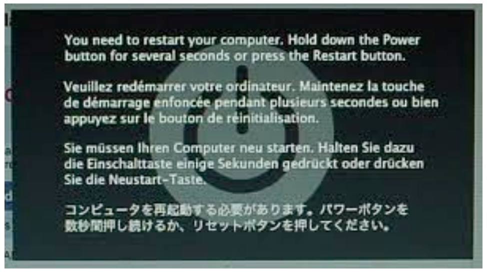
Fuzzing .mov Files

- Any crashes so far? (don't send them to Apple)
- Do we need to build a custom .mov file fuzzer?
  - http://shakacon.org/2009/talks/Exploit_or_Exception__DeMott.pdf
  - http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-sutton.pdf
  - http://www.cert.org/vuls/discovery/bff.html
  - http://peachfuzzer.com/v3/TutorialFileFuzzing.html
  - Chapter 6: iOS Hacker's Handbook
  - https://developer.apple.com/standards/classicquicktime.html

A crash has been detected. Help us improve AngryNerds by reporting crashes?

Bugs, Vulns, Exploits we are looking for in class

- Remote Exploits
- Userland Vulnerabilities(to obtain mobile)
- Privilege escalation (mobile to root)
- Escaping sandbox techniques
- Memory leaks
- Bypassing code signing techniques
- Kernel Vulnerabilities
- Strategies for dealing with KASLR on 64-bit ARM



You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

Sk00l Supplies
Software / Hardware

- Xcode (run latest version)
- IDA Pro
- Gdb
- OxED http://www.suavetech.com/0xed/
- Mac book running Mountain Lion
- Serial Debugging Cable
- WiFi
- Jailbroken iPhone4/5 (6.1.2)
- UnJailbroken iPhone4 (7.0.2)
- UnJailbroken 5/5S (7.0.2)

Great Starter Books

- Mac OSX and iOS Internals
- Hacking and Securing iOS Applications
- iOS Hacker's Handbook
- Mac Hacker's Handbook
- OSX and iOS Kernel Programming
- Cocoa Application Security
- C in a nutshell
- ARM Assembly Language Fundamentals & Techniques
- ARM Assembly Language –An Introduction