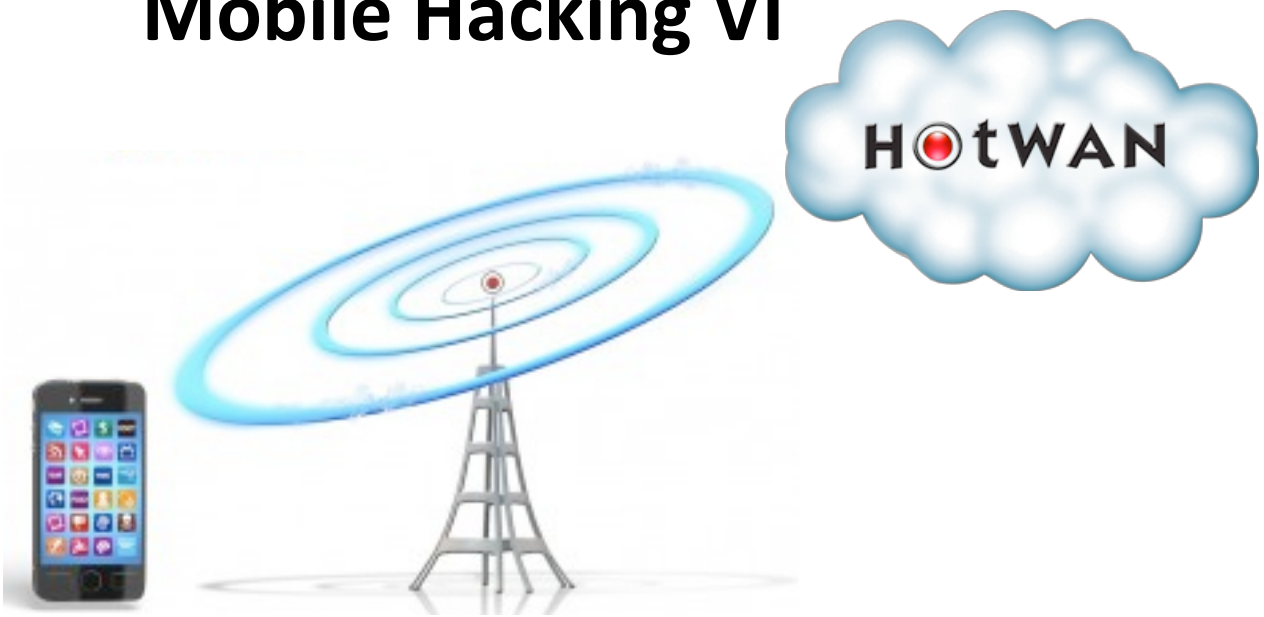
Mobile Hacking VI



iOS Jailbreak

Crakun
Sat, 10-12-13 Day 4

Crakun Contact Info

Email: crakun@m0bdev.com

Twitter: crakun

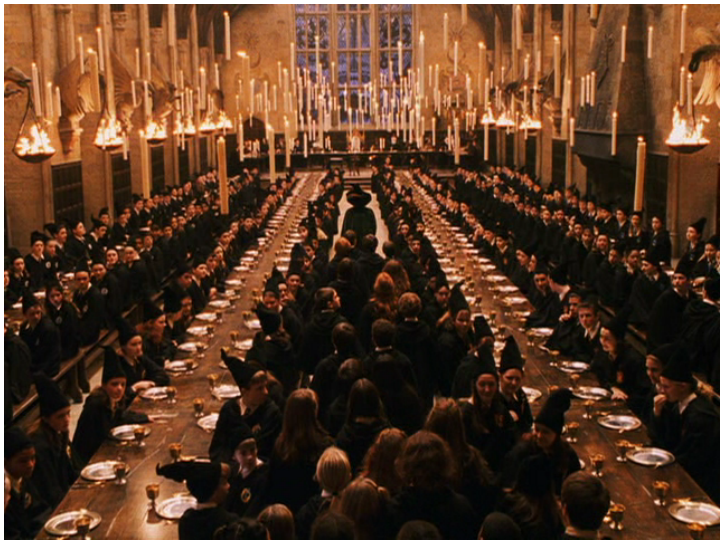
Skype: m0bdev (m0bdev is spelled with a zero)

IRC: #openjailbreak on freenode

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

Mobile Hacking VI
Class Agenda

- Part I:
  - Back to the Goal
  - Status Update
  - Submittal Process
  - Homework (your answers)



- Part II
  - Fuzzing

I need You to Focus

On the Goal

The “Winning” Goal

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

Status Updates

1. Submittals so Far

2. Your Progress

3. Fuzzing Automation

Class review of .mov Crashes

Ideally, we are looking for a .mov file that is exploitable for

- Jailbroken iPhone4/5 (6.1.2)
- UnJailbroken iPhone4 (7.0.2)
- UnJailbroken 5/5S (7.0.2)

Starting on Oct 26, Based on Submittals we will go over a few .mov files to determine if they are exploitable as a class

Submittal for .mov Crashes

Email Me with a zip:

1. Which phones and firmware were the .mov file tested on

2. Include Crash Report

3. Include original, unfuzzed .mov file and fuzzed .mov file

4. Tell me what you think it is?

Submittal for other Bugs/ Vulns / Exploits

Email Me with a zip:

1. Detail Step-by-Step Method how / what you did so I can reproduce it
2. What devices and firmware does this occur on?
3. Tell me what you think it is?
4. Include Crash Report
5. Include Proof-of-Concept source code

Review of Oct 5 Homework Assignment

1. What are the vectors for attacking iPhones remotely (SMS, MMS, .pdf, .xls, etc.)

2. What files formats are supported for rendering / parsing from MobileSafari

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
- Prepare a list of the best IRC channels and websites for iOS Jailbreak research (we will distribute this to class for everyone)
- Continue on:
  - Fuzz .mov files / Submit
- Spread the Word. We need Volunteers for this Jailbreak

Part II
What is Fuzzing

- Exploring Code Paths

- Sending invalid data to an interface in hopes of triggering a error condition or fault condition.

- These errors can lead to exploitable vulnerabilities

And when applied to network

VALID CASE(S) ARE USED TO VALIDATE CONNECTIVITY

FUZZING PLATFORM

SYSTEM UNDER TEST

VALID SERVICE QUERY

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap,
image/jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
```

VALID response

```
HTTP/1.1 200 OK
Date: Wed, 07 Nov 2007 09:44:49
GMT
Server: MyWebServer/2.1 (Linux)
Last-Modified: Wed, 07 Nov 2007
09:44:36 GMT
Accept-Ranges: bytes
Content-Length: 100
Connection: close
Content-Type: text/html;
charset=UTF-8
```

HOtWAN

And when applied to network

ANOMALY CRASHES SUT

FUZZING PLATFORM

SYSTEM UNDER TEST

ANOMALY sent

```
GET http://[?
aAaAaAaAaAaAaAaAaaAaAaAaAaAaAaA::0]
HTTP/1.1
Accept: image/gif, image/x-xbitmap, image
jpeg, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
```

CRASH (no response)

```
<========[ NO RESPONSE ]=========
```

HOtWAN

Types of fuzzing

```
random / bit blasting: {samples} -> G
template based:        {samples, attacks} -> G
model based:           {samples, attacks, specs} -> G
```

- Random bit blasting: you take samples and give them to your test generator. None or limited semantics (= recalculation of lengths, checksums, etc).
- Template based: you have defined attacks you apply to analyzed sample(s) and give to test generator. Up to full message semantics, but limited inter-message semantics. Testing is still limited to observed samples.
- Model based: In addition to samples and attacks, a formal protocol specification is given to test generator. Test coverage goes beyond observed samples as one can exercise full specification. Up to complete semantics, which allows covering more protocol states.

Not all inputs are born equal

Minimizing Infinity

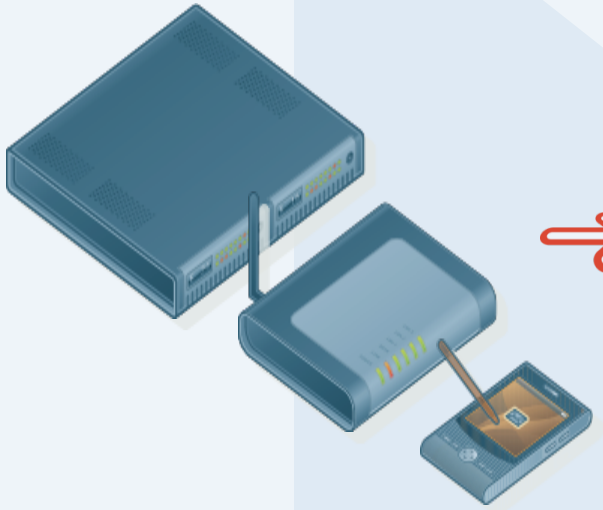
INPUT ANOMALIES

FIELD LEVEL
overflows
integer anomalies

STRUCTURAL
underflows
repetition of elements
unexpected elements

SEQUENCE
out of sequence
omissions
unexpected messages
repetition of messages

system under test



- SYSTEMATIC
- REPEATABLE
- INTELLIGENTLY TARGETED

EXPOSE VULNERABILITIES

- crashes
- denial of service
- security exposures
- degradation of service
- thrashing
- anomalous behavior

HOtWAN

What can be fuzzed, what are failures

[Images from images.google.com]

SMS

- Smartphone owners are exposed to threats from text messaging as well, our research shows that 0.02% of SMS sent are malicious, this seems like a low percentage and this might suggest a low chance to get infected BUT, in the USA alone, during Dec 2010, 187.7 Billion Text SMS were sent (source: www.ctia.org) which means that 3.75 Million messages are potentially malicious.

Smart phone – [typical] attack surfaces

WIRELESS: GPRS, EDGE/3G
GSM, SMS, MMS, SMIL, OTA updates,...

WIRELESS: 802.11:
802.11a/b/g/n, WPA, WPA2,..

WIRELESS: Bluetooth:
L2CAP, RFCOMM, SDP, OPP, A2DP, AVRCP, PBAP, DUN,...

IP CONNECTIVITY:
IPv4 (ARP, ICMP, IGMP, IP, UDP, TCP), IPv6 (IP, ICMP, ND, RD, SEND, MLD, TCP, UDP), HTTP, TLS/SSL, OCSP, RTSP, SIP/IMS, RTP/RTCP, SigComp, DNS, MDNS, DHCP, NTP , SOAP, REST/JSON, SMTP, POP3, IMAP4, WAP/WMLC,..

PHYSICAL CONNECTIVITY:
USB, SERIAL, MEMORY CARD, SIM,..

MEDIA:
AUDIO (AAC, MP3, MP4, 3GP, WAV, ...), IMAGES (JPG, GIF, PNG, TIFF, ...), VIDEO (MPG1, MPG2, MP4/H.264, WEBM,...), ARCHIVES (ZIP, JAR, CAB, ...), DOCUMENTS (PDF, DOC, PPT,..), X509, EMAIL (MIME, calendar, vcards,...), DRM, Flash, Java classes , Application installers,...

[WEB] APPLICATIONS:
XML, DRM, HTML5 (CSS, HTML, Javascript) , AT commands, inter process APIs/RPCs,

[typical] Attack delivery channels

3G / GPRS / EDGE (active & passive attacks)

Wi-Fi (active & passive attacks)

Bluetooth (active attacks)

USB (active & passive attacks)

Memory card reader (passive attacks)

HOtWAN

[typical] Failure modes

- OS kernel panic
- Application crash
- Application busy loop (stuck application, UI)
- Loss of functionality
- Draining of the battery
- [exploitation]

HOtWAN

Direct Attack interfaces

Tested applications and protocol
Entities can be actively contacted

Fuzzing tool

Bluetooth active tests

IPv4, IPv6, WiFi,
SIP, RTP, etc. active tests.

HOtWAN

Assisted attack interfaces

Fuzzing tool acting as a server

- TLS/SSL enabled web server
- Web server
- Mail server
- ...

Either use protocol capabilities to automate tests or use purpose build apps.

Media tests

Media, HTTP, TLS/SSL,
Email (MIME), SMTP, POP3, IMAP4, tests,
Application tests

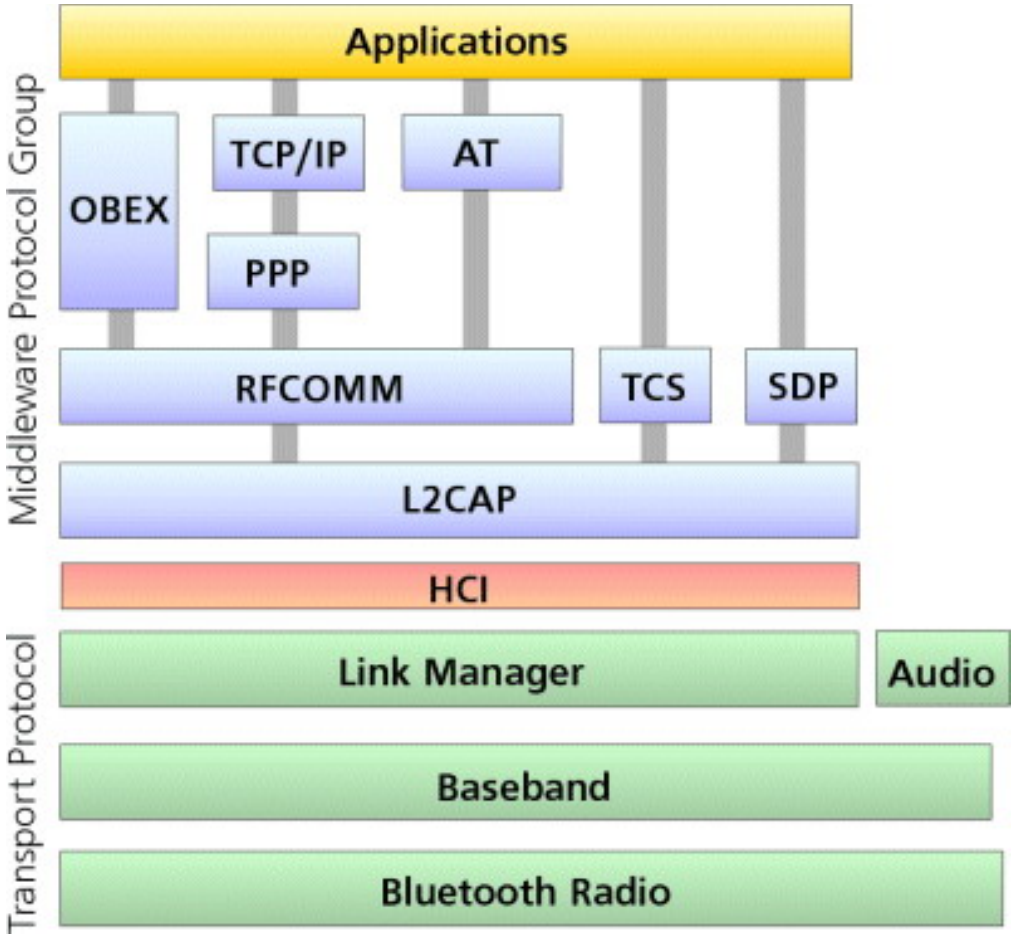
HOtWAN

Mobile fuzz testing challenges

- Every mobile OS different
  - Automation may not be portable
  - Restarting applications / phone often hard
  - Runtime analysis/debugging tools often not available
- Multiple ”stacks” for same protocol
  - E.g. MMS viewer, Web browser and image gallery may use different code to decode .jpeg file
  - Sometimes transport matters (IP over WiFi vs. 3G vs. 2G)
  - Carrier modifications add to this
- Most applications are ”clients”
- Vendors not always responsive, reactive or informed

HOtWAN

Bluetooth Fuzzing



File Fuzzing

These tools are useful for testing any program which processes binary file inputs such as archivers and image file viewers.

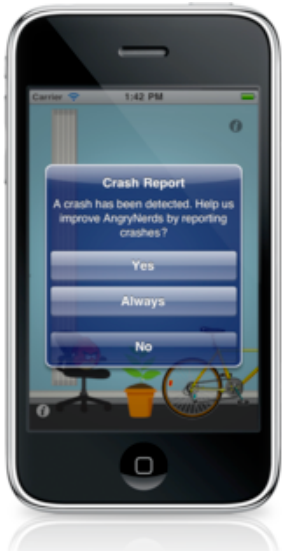
FileP is a python-based file fuzzer. It generates mutated files from a list of source files and feeds them to an external program in batches.

Prerequisites: Python 2.4

FileH is a haskell-based file fuzzer. It generates mutated files from a list of source files and feeds them to an external program in batches.

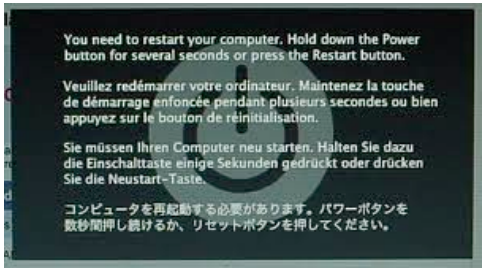
Fuzzing .mov Files

- 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



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



Next Meeting:
(Day 4): Sat, Oct 12, 2013 6am PST

- Fuzzing Automation
- Intro to CrashDump

Open Floor Discussion

APPENDIX

- Find/Create Fuzzer
- 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**

Mobile Hacking VI
Class Schedule

- Weekly Class is migrating to Saturdays, 6am PST
- Next meeting (day 5): Sat, Oct 19, 2013 6am PST

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
- Learn C on the Mac: For OSX and iOS
- ARM Assembly Language Fundamentals & Techniques
- ARM Assembly Language –An Introduction