Android Based Penetration Testing Framework

Ron Munitz

Founder & CEO - The PSCG
ron@thepscg.com

Android Builders Summit 2015

@ronubo
about://Ron Munitz

- Founder and CEO of the PSCG
  - The Premium Embedded/Android consulting and Training firm
- Founder and (former) CTO of Nubo Software
  - The first Remote Android Workspace
- Instructor at NewCircle
- Senior Lecturer at Afeka College of Engineering
- Working on an awesome stealth startup
- Building up on diverse engineering experience:
  - Distributed Fault Tolerant Avionic Systems
  - Highly distributed video routers
  - Real Time, Embedded, Server bringups
  - Operating Systems, very esoteric libraries, 0’s, 1’s and lots of them.
    - Linux, Android, VxWorks, Windows, iOS, devices, BSPs, DSPs,...
Agenda

● What is Penetration Testing?
● Pentesting Android devices
● Using Android as a Pentest module
● Future work
Introduction
Penetration Testing

- Penetration Testing ("Pentest-ing") is the act of attempting to find weaknesses ("exploits") in the system of test.
- It is essentially **attacking** the system of choice, with permission for the purposes of:
  - Evaluating system security
  - Finding weaknesses in the tested system
  - Analyzing the performance of a system in extreme conditions
  - More
Penetration Testing

- Usually done to prove the security of the product before unethical hackers take advantage of such *exploits*.
- Or as a mean of preparing to *auditing*
- Or for research purposes.
  - Academic
  - Personal
    - Some time triggers “personal” undesired earning
  - Bug-Bounty
Typical life of an exploit

- An exploit is being found
  - hopefully by “the good guys”
    - Otherwise: Being used, until some good guys discovers it, and the following loop is closed
- It is being reported to the corresponding entities (depending on the products affected etc.)
- A patch (usually defines a “Security Update” is being published)
Typical life of an exploit

- The exploit is being published
  - e.g. the CVE List (https://cve.mitre.org/)

- It is extremely important not to publish an exploit before a fix is available.

- Sometimes, a fix cannot be available for all affected platform (e.g. old phones that do not receive OTA updates anymore).
  - Beware that.
  - That is extremely relevant to Android
Android zero-day exploits

- Many companies ship versions of Android
- Most of them never ship the latest or greatest version
  - When the phone/tablet/whatever comes out
  - Or at all
- At the mean time, some exploits that affect known versions of Android have been discovered and disclosed
- And users may be suspect to them forever…
- (Good/Bad?) Example: Rootkits
Rootkits

- Rootkits enable to gain superuser permissions on your Android Phone
- Which is really a Linux Phone
- Do the math...
Penetration Testing tools

- There are many pentest tools that maintain an open database of known exploits, and allow you to exploit a target.
- Those tools can help you test your device for vulnerabilities.
- Or exploit it or other devices...
- Metasploit is such a tool that allows you to
  - “Exploit yourself” - via adb/et.-al
  - Exploit others via malicious software...
Pentesting Android Targets
Remote exploitation scenario

- Assume S, a metaspliot server containing an enormous database of Android exploits
- One can pack a module that connects to S within an APK
  - e.g. a Service within a game that may have been granted some permissions by the user
  - Especially if the phone is rooted...
- And opens a shell on the target side to connect to S
- Then Metaspliot can work on the user’s phone without them ever knowing about it!
Pentesting Android devices

● An Android ROM cooker (e.g. an OEM) may use those techniques for good causes
● To see whether the devices they are going to ship have no [disclosed] 0-day vulnerabilities
● Or to verify their injected 0-day malware is undisclosed by “current” public knowledge tools…
Pentesting Android devices

- Android penetration testing, just as any other pentest deals with diverse “victims” such as:
  - Apps [behavior under some conditions, fuzzing]
  - Resources [exhausting some resources, e.g. network, memory]
  - Permissions [shellcodes/privilege escalation…]
  - User data [applies to each]
  - Forensics [also applies to each]
  - Information “over the wire” [e.g. redirect everything via a proxy / MITM etc.]
  - And much, much more…
Pentesting Android devices

- There are many tools that allow fuzzing and testing Android apps / network etc. Some of them are:
  - The monkey* suite
  - drozer
  - dSploit
  - And many many more.
Pentesting Android devices

- If you are able to build a ROM / have root access - you can port your own “legacy tools easily”
- If you just care for testing the software - an Android Emulator (or equivalent) would be just fine.
Using Android to Pentest other targets
Android as a Pentest Suite

- Since Android is essentially Linux, it can benefit from a lot of work that has been done on Linux hosts, to **pentest other hosts/modules**.
- For many of such cases, root access is essential (e.g. USB MTM attacks etc.)
Android as a Pentest Suite

- Kali Linux for Android is a very nice example for such system
- It is just another “Android/Linux distro”
- Using chroot on your Android device to start another Linux
- And connecting to it’s GUI via VNC
- Can be ran natively [if you want to port it]
Pentesting Mobile Ecosystems

- On these days, there are nearly no significant “self contained” applications.
- Not for the mobile app market
- **Always two, there are:**
  - A mobile app (x deployments)
  - A mobile backend

⇒ If the mobile backend (Servers etc.) is broken - so are the clients.
Pentesting Mobile Ecosystems

There are many ways to test, or try to break an app:

- From within the app itself [e.g. fuzzing the app, monkey* etc.]
- By denying a service from the app [e.g. overloading the servers]
- By cooperating with other apps to achieve some sort of DOS
- By breaking the rewarding activities / identification etc

In one line: Advertising, Monetization
Pentesting Mobile Ecosystems

(monetized) Mobile ecosystems are affected by many parameters:

- A user ID (AndroidID, AdvertisingID, etc.)
- User location
- IP, used networks
- User-Agent
- Particular device they are using
- Accounts
- And more…
Pentesting Mobile Ecosystems

WHAT IF I TOLD YOU

ONE COULD MODIFY THOSE PARAMETERS ON A LARGE, DISTRIBUTED SCALE?
Pentesting Mobile Ecosystems

Well, one can:

● Leveraging on techniques done in remote application testing (e.g. Perfecto Mobile, Applause and similar solutions)
● But without the human or 1-1 interaction…
● Rather doing the same on a much, much, much larger scale
● Reducing operation costs by leveraging on:
  ○ Android Virtualization
  ○ Containers
  ○ Hypervisors
Pentesting, Scaling, Achieving

Management
- Scripts
- ADB host
- ssh
- Web interface
- ...

A1
Linux or any OS/
Vbox/AX86
Germany

A2
<any OS>/
Android Emulator
United States

A3
GCE (Google Compute
Engine)/
AX86
Israel

A4
AWS (Amazon Web
Services)/
AX86
Ireland

A5
Some Android
Some Geo

NFS
System.img
userdata.img

NFS
NFS/other mount
sdcard
Why Morpheus?

● Because (Thank God) it’s a Linux Conference, so there is no way you’ve never been asked “what is this matrix?!?” after an endless make.

● In a next session I might use a Thor picture, for having them used “whois” in “Blackhat”.

● For those who don’t remember the morpheus slide - you chose the wrong pill.
  ○ It’s not like I “sed-ed” the text with the image before uploading.
Thank You

Questions/Consulting/Training requests:
ron@thepscg.com