Trustwave®
SpiderLabs®

DEF CON 18
"This is not the droid you're looking for…"

Nicholas J. Percoco & Christian Papathanasiou

Agenda

- About Us / Introduction
- Introduction to Android
- Motivations Behind this Work
- Building a Linux Kernel Rootkit
  - Overcoming Hurdles
- Introducing Mindtrick – The Android rootkit
- Live Demo
- Current Prevention
- Conclusions

About Us

Nicholas J. Percoco / Senior Vice President at Trustwave

- 15 Years in InfoSec / BS in CompSci
- Built and Lead the SpiderLabs team at Trustwave
- Interests:
  - Targeted Malware, Attack Prevention, Mobile Devices
    - Business / Social Impact Standpoint

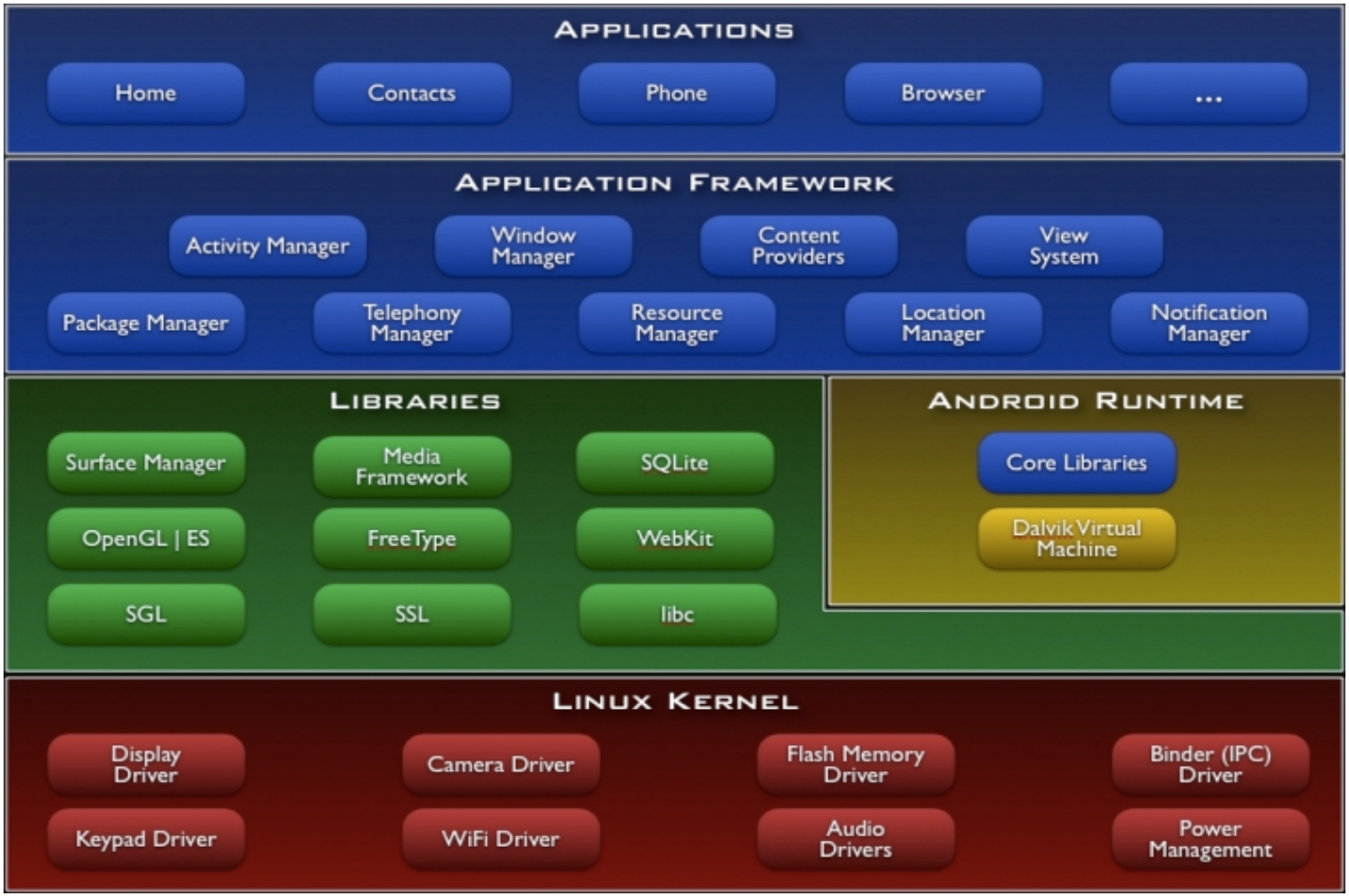
Christian Papathanasiou / Security Consultant at Trustwave

- 8 Years in InfoSec / MSc in InfoSec / MEng in ChemEng
- Interests:
  - Rootkits/Anti-Rootkit detection, Algorithmic Trading, and Web Application Security

Introduction

- **Android is a software stack for mobile devices**
  - 60,000 phones running Android ship every day
  - Ranks 4th most popular smart phone device platform
- **Not much research around rootkits on mobile devices**
  - Android == Linux == almost 20 yr old Open Source OS
  - Very established body of knowledge in Linux Rootkits
- **We created a kernel-level Android rootkit**
  - Loadable Kernel Module
  - Activated via a Trigger number

Introduction to Android – The Model



Source: Google

Introduction to Android – Linux Kernel

- **Based upon the Linux 2.6.x kernel**
- Hardware Abstraction Layer
- **Offers:**
  - Memory Management
  - Process Management
  - Security
  - Networking
- Android Platform sits atop of the Kernel
- **This is where our rootkit lives (more later…)**

Introduction to Android – Libraries

- **Libraries == Most of Android's Core Functionality**
- **Libraries of most Interest:**
  - SQLite – main storage/retrieval (calls/SMS records)
  - Webkit – browser functionality
  - SSL – crypto
- **Ideas/Hints:**
  - What if you can read SMS messages?
  - How about intercepting browser sessions?
  - Can you hook the PRNG with static low numbers?

Introduction to Android – Runtime

- **Android's Runtime Environment == Dalvik VM**
- **What is Dalvik?**
  - Virtual Machine on Android Devices
  - Runs applications converted into .dex format
  - The "Dalvik Executable" is for systems that have low:
    - Memory
    - Processor Speed
- **We didn't spend much time here…**

Introduction to Android – Application

- **Application Framework**
  - Core User Functionality
  - Used by the Applications
- **Applications**
  - This is where the User Applications live
  - Either come installed with the Phone, Downloaded from Android Market or self-installed
- **Again, we didn't spend much time here…**

Introduction to Android – Other Notes

- **All Applications and User Activity Utilizes Linux**
  - I/O with Hardware
- **By hijacking Linux Kernel, you "own" all other layers**
  - Modify phone behavior at will
- **Complete end-user abstraction is a Usability Advantage**
- **Complete end-user abstraction is a Security Disadvantage**
  - A successful attack just needs to subvert to Application Layer, since the end-user doen't look below it
  - Even if the attack causes a performance issues, the end-user will just call it a "bug" and reboot the phone.

Motivations Behind this Work

- **As of Q4 2009, 485 million devices on 3G networks**
- **By 2020, there will be 10 billion devices**
- **60% of all users carry their devices with them at ALL times**
  - For high-profile and business folks that is near 100%
- **A typical smartphone today, has the same processing power as a PC from 8 years ago, plus:**
  - Always-on network connectivity
  - Locations aware thanks to GPS

Motivations Behind this Work (cont'd)

- **Users accessing highly sensitive information via smartphones is the norm**
- **Users trust a smartphone over a public computer or kiosk**
  - Never question their smartphones integrity
- **Communication Services Providers (CSPs) must allow for governments to access subscribers communications**
  - Case: In the UAE, Etisalat pushed a "performance update" to all their Blackberry subscribers.
  - Reality: Malware was intentionally pushed down to allow interception of data communications.

Motivations Behind this Work (cont'd)

- **What we are NOT doing here:**
  - Developing a new attack vector to get our payload on the phone
    - Just wait a few weeks/months and there will be one…
    - *cough* Adobe Flash / Acrobat Reader *cough*
    - Malicious App
- **We chose Android, because it runs Linux**
  - Everyone *can* access the source code
- **No personal issues with Google or Android**
  - Great OS, Great Phones, Great Apps

Building a Linux Rootkit

- **Loadable Kernel Modules (LKMs) allow OS kernel to be extended dynamically.**
- **LKMs has the same capabilities as code in the kernel**
- **System Calls are used e.g., for file, process, and network operations**
- **Systems Calls are listed in sys_call_table**
  - An array of pointers / Indexed by system call number

Building a Linux Rootkit (cont'd)

- **Traditional "rootkits" are software packages**
  - Often replace system binaries like ls, ps, netstat
    - Used to hide attacker's files, processes and connections
- **Traditional "rootkits" can be easily be detected by:**
  - Comparing "known good" files with suspect ones
  - Comparing checksums (RPM database or FIM utility)
- **A "kernel rootkit" can subvert the kernel itself using "hooks"**
  - Hide specific processes from /proc so ps can't see it
  - Hide itself from LKM listings
    - Subvert calls made by lsmod command

Building a Linux Rootkit (cont'd)

What is a "hook"?

- A hook is a redirection of a system call
- Modifies the flow of execution
- A hook registers its address as the location for a specific function
  - When the function is called the hook is executed instead

By Creating a LKM in Android, we not only subvert the layers above the kernel, but the *End-User Himself!*

Building a Linux Rootkit – Hurdles

- **There were a few hurdles to overcome:**
  - Retrieve the sys_call_table address
  - Compile against the device kernel source code
  - Enable System Call Debugging

Building a Linux Rootkit – Hurdles

Retrieve the sys_call_table address

- **Problem:**
  - Linux Kernel 2.5 or greater no longer export sys_call_table structure
  - **`extern void *system_call_table[];` DOES NOT WORK!**

- **Solution:**
  - It can be found in the System.map
  - Find it in the device's kernel source code

```
root@argon:~/android/legend-kernel# grep sys_call_table System.map
C0029fa4 T sys_call_table
root@argon:~/android/legend-kernel#
```

These addresses are STATIC all devices with the same hardware/firmware/kernel!

Building a Linux Rootkit – Hurdles

Compile against the device kernel source code

- **Problem:**
  - The kernel refused to accept our LKM because version magics didn't match
- **Solution:**
  - We found version magics are stored in the form of a static string
  - We need modify kernel source code in `include/linux/utsrelease.h`

OLD

```
root@argon:~/android/legend-kernel# cat utsrelease.h
#define UTS_RELEASE "2.6.29"
root@argon:~/android/legend-kernel#
```

NEW

```
root@argon:~/android/legend-kernel# cat utsrelease.h
#define UTS_RELEASE "2.6.29-9a3026a7"
root@argon:~/android/legend-kernel#
```

After re-compiling our LKM against the HTC Legend source, the module loaded!

Building a Linux Rootkit – Hurdles

Enable System Call Debugging

- **Problem:**
  - We need to map out the system calls we were interested in in order to discover high layer phone functions which we would later intercept

- **Solution:**
  - We wrote a debug LKM that incepted the following calls:
    - sys_write
    - sys_read
    - sys_open
    - sys_close

Building a Linux Rootkit – Hurdles

Enable System Call Debugging

- **What did we learn?**
  - We can discover phone routines by parsing **`dmesg`** for specific actions (or data we input).

- Example:
  - Placing/Receiving a call to/from the "rootkitted" phone and parsing for the phone number reveals commands used by the phone.
  - Our debug LKM captures all browsing activity and social networking activity being conducted on the phone as well. This could be used as an additional C&C channel.

Introducing Mindtrick – The Android Rootkit

What does it do (today)?

- **Sends an attacker a reverse shell over 3G/WiFi**
- **Triggered by a pre-defined phone number**
- **Attacker than have access to the phone's OS as ROOT**
  - **See Demo for other FUN!**
- **The rootkit is hidden from the kernel**

```
# lsmod
# insmod mindtrick.ko
# lsmod
#
```

Note: The source for Mindtrick is on the DEFCON 18 CD.

Live Demo

What are we going to do?

- Install the rootkit
- Activate the rootkit via a phone call
- View the reverse shell connect
- View SMS messages
- View Contacts
- Retrieve GPS coordinates
- Make phantom phone call
- Shutdown the phone

Current Prevention

What did we test?

- Neither *Lookout Mobile Security* nor *Norton Smart Phone Security* detect LKM Rootkits

What can be done?

- Manufactures should ensure all device drivers / LKM / are centrally signed.

Conclusions

- **It is possible to write a rootkit for the Android platform.**
- **We didn't include automated functionality (by design).**
  - This can easily be done.
- **Little attention is being paid to smartphone security, while everyone trusts their device to perform critical tasks.**
- **In the next 10 years, we will see an explosive growth in the number of attacks against smartphones and other mobile computing device platforms. Will we be prepared?**

Trustwave®
SpiderLabs®

Contact Us:

Nicholas J. Percoco / npercoco@trustwave.com / @c7five

Christian Papathanasiou / cpapathanasiou@trustwave.com / @h0h0_