MILCOM2015
LEVERAGING TECHNOLOGY—THE JOINT IMPERATIVE
OCTOBER 26–28, 2015•TAMPA, FL

# Tutorial: Practical Program Analysis for Discovering Android Malware

## Module 4: **Novel and Sophisticated Malware**

Suresh Kothari – kothari@iastate.edu
Benjamin Holland – bholland@iastate.edu

Acknowledgment: co-workers and students
DARPA contracts FA8750- 12-2-0126 & FA8750-15-2-0080

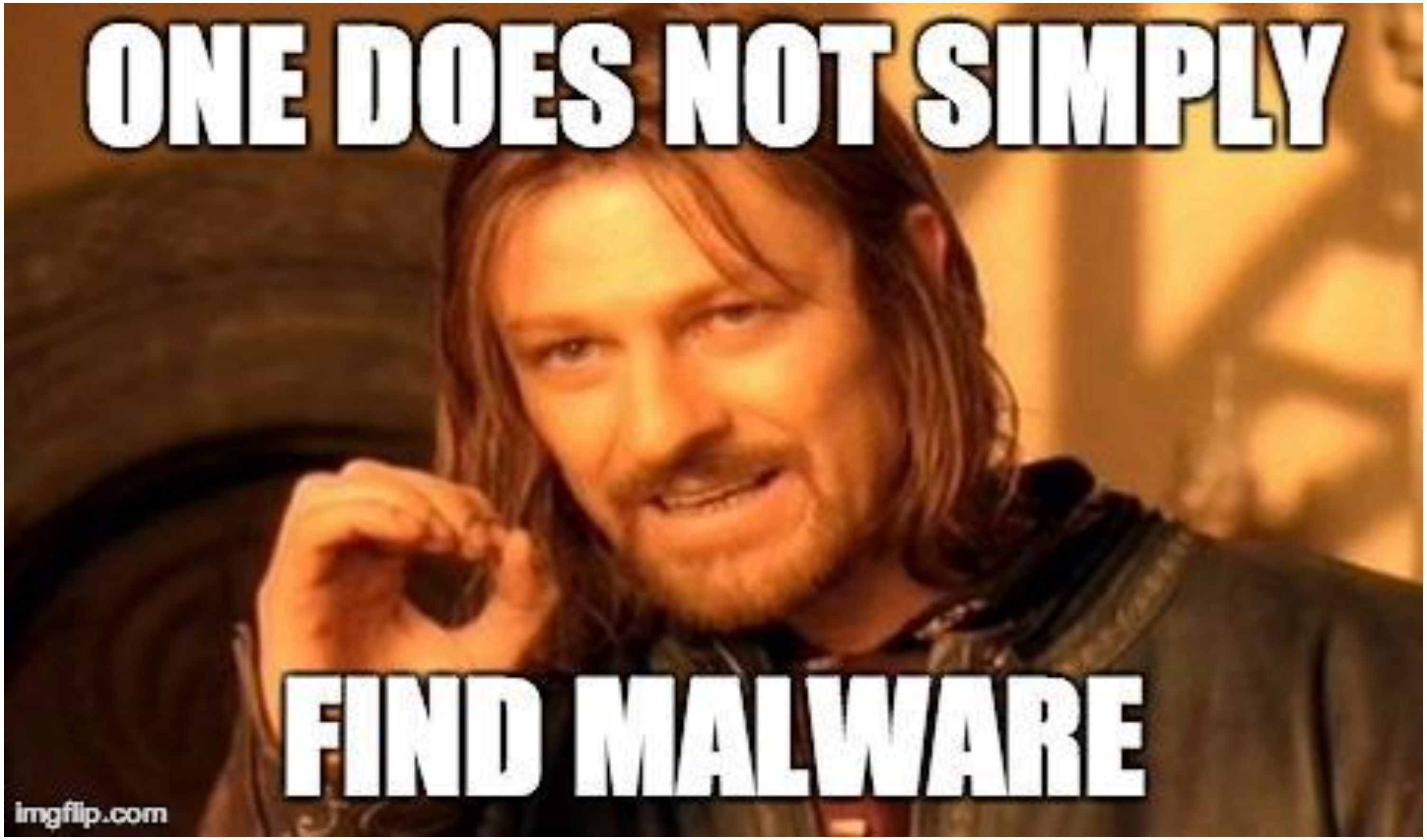
ensoft
conquer complexity

IOWA STATE
UNIVERSITY

# Agenda

- Novel and Sophisticated Malware
- OODA Loops with Atlas

# What have we learned?



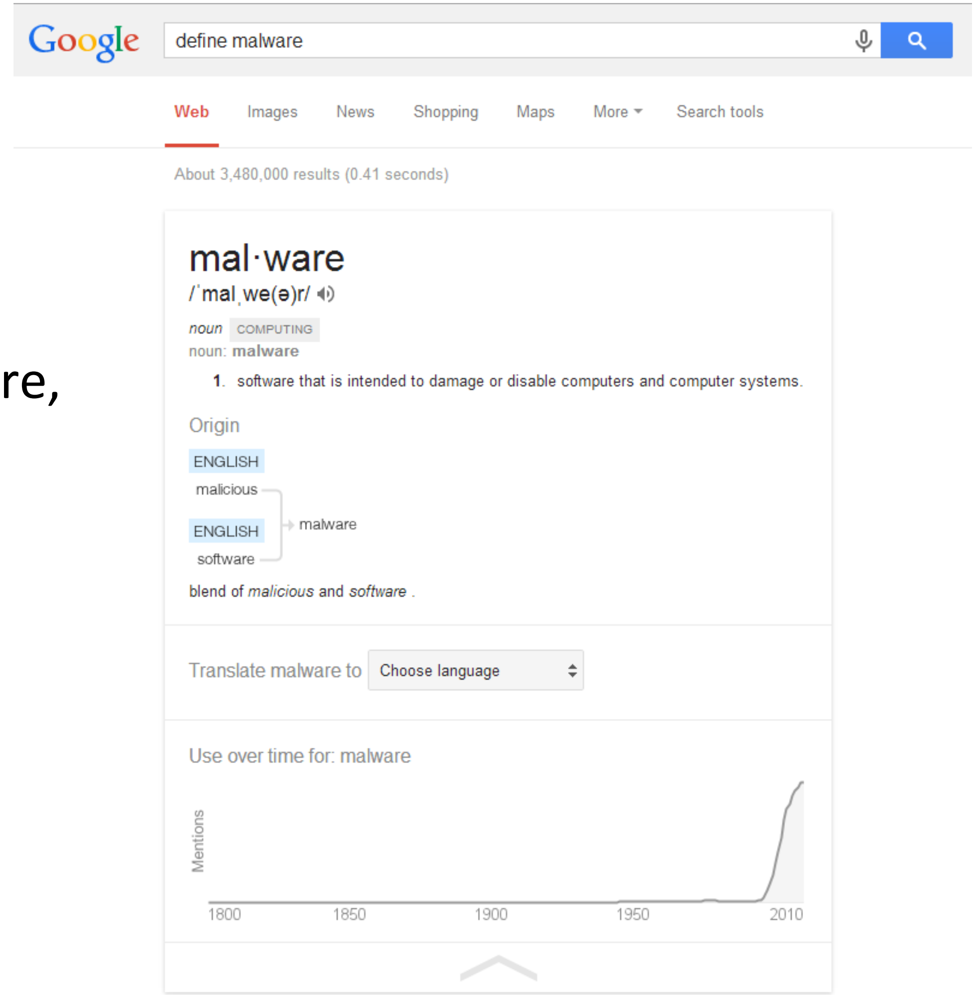
# Simple Refactoring of CVE-2012-4681

- “Allows remote attackers to execute arbitrary code via a crafted applet that bypasses SecurityManager restrictions…”
- CVE Created August 27th 2012
- github.com/benjholla/CVE-2012-4681-Armoring

| Sample | Notes | Score (positive detections) |
|---|---|---|
| Original Sample | http://pastie.org/4594319 | 30/55 |
| Technique A | Changed Class/Method names | 28/55 |
| Techniques A and B | Obfuscate strings | 16/55 |
| Techniques A-C | Change Control Flow | 16/55 |
| Techniques A-D | Reflective invocations (on sensitive APIs) | 3/55 |
| Techniques A-E | Simple XOR Packer | 0/55 |

# Can we define malware?

- Bad (malicious) software
- Examples: Viruses, Worms, Trojan Horses, Rootkits, Backdoors, Adware, Spyware, Keyloggers, Dialers, Ransomware…
- Too broad of a definition?



define malware

About 3,480,000 results (0.41 seconds)

mal·ware

/ˈmalˌwe(ə)r/

*noun* COMPUTING

noun: **malware**

1. software that is intended to damage or disable computers and computer systems.

Origin

ENGLISH malicious → malware

ENGLISH software → malware

blend of *malicious* and *software* .

Translate malware to

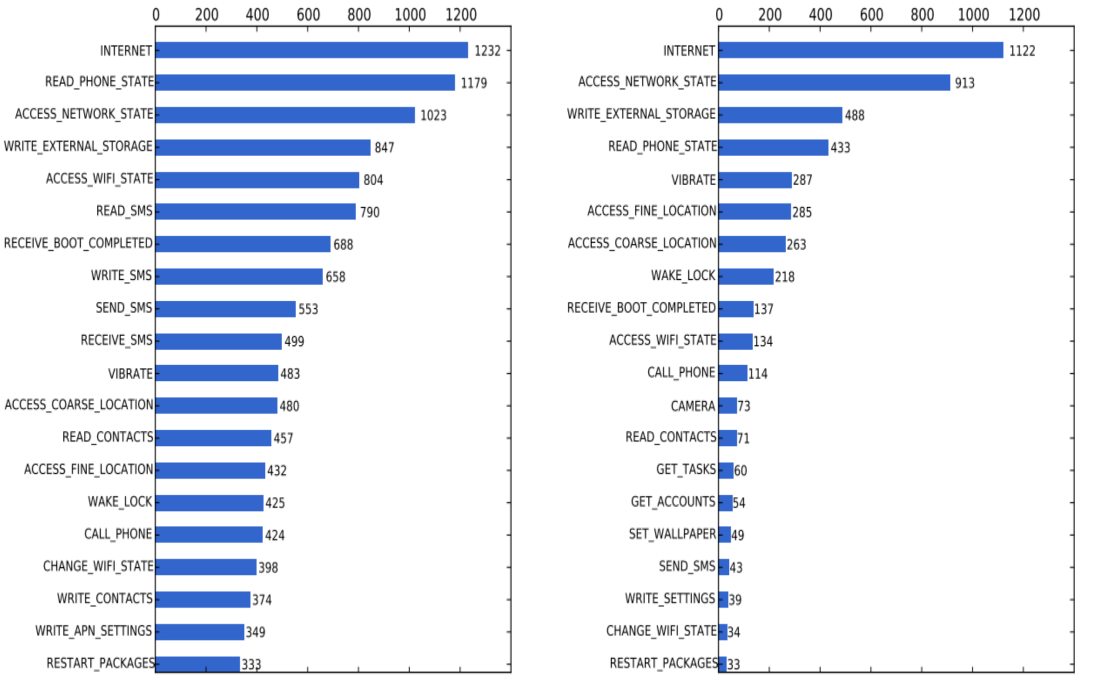
Use over time for: malware

# What properties would ideal malware have?

- Operational goals
  - Effective, adaptable
  - Maintaining ownership
  - Cross platform, cross architecture
  - Persistence (survival, removal, updatable)
- Detection avoidance
  - Resistant to static/dynamic analysis (intractable analysis problems)
  - Difficult to characterize
  - Small footprint (low resource consumption, minimal impact)
  - Blends well with legitimate functionality
- Detection mitigation
  - Plausible deniability
  - Kerckhoffs's principle (ex: untraceable transactions)
- General Software Design Issues
  - Maintainable, deployable, scalable, etc.

# Android Malware in the Wild

- Yajin Zhou, Xuxian Jiang. Dissecting Android Malware: Characterization and Evolution. http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf.

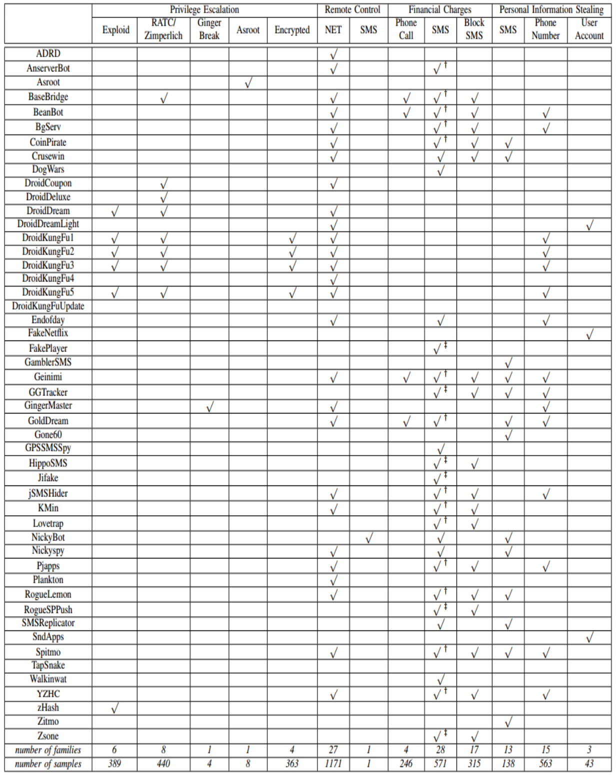


(a) Top 20 Permissions Requested By 1260 Malware Samples

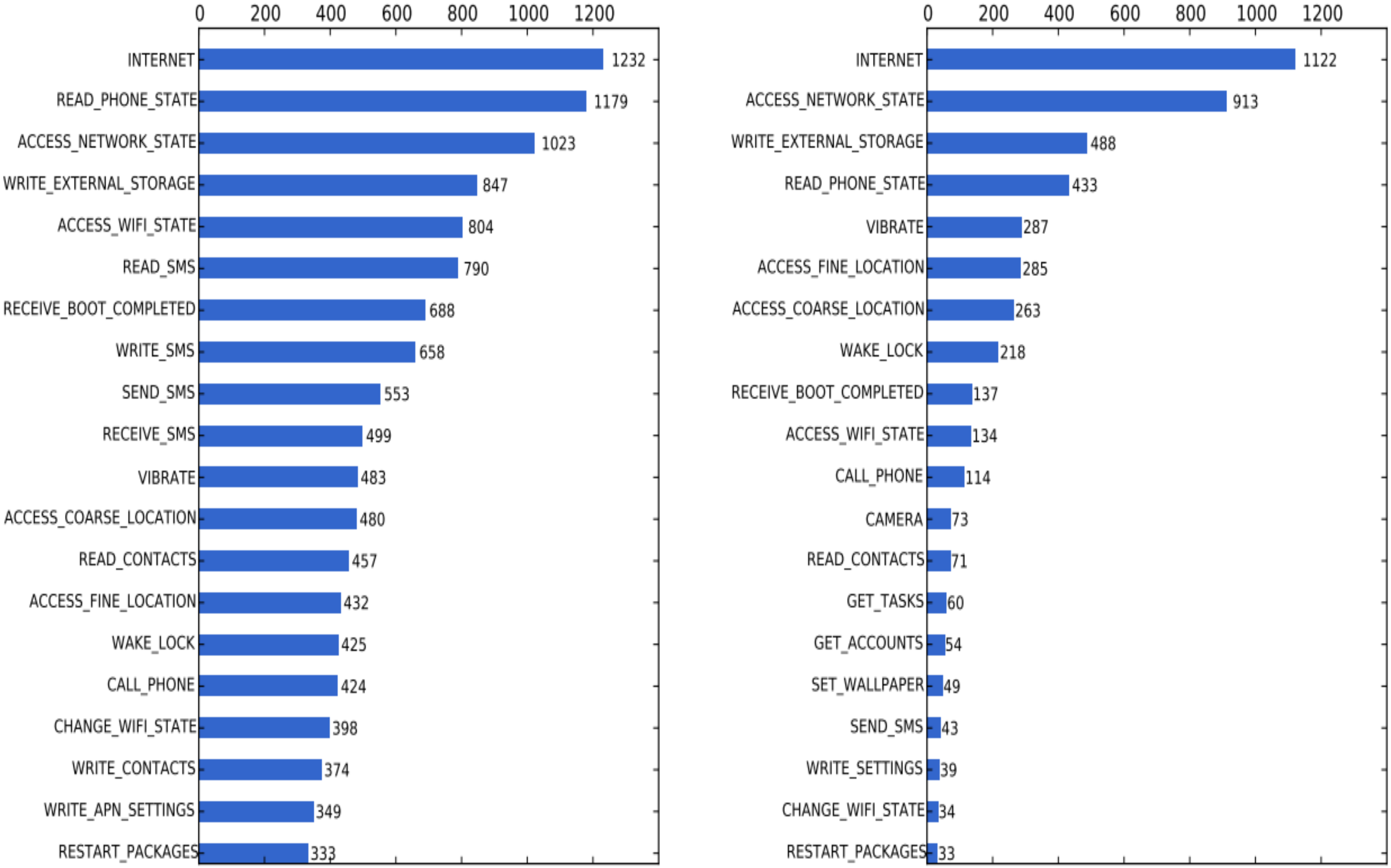
(b) Top 20 Permissions Requested by 1260 Top Free (Benign) Apps on the Offical Android Market

Figure 5. The Comparison of Top 20 Requested Permissions by Malicious and Benign Apps

Table V
AN OVERVIEW OF EXISTING ANDROID MALWARE (PART II: MALICIOUS PAYLOADS)



| | Privilege Escalation | | | | | Remote Control | | Financial Charges | | | Personal Information Stealing | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Exploid | RATC/ Zimperlich | Ginger Break | Asroot | Encrypted | NET | SMS | Phone Call | SMS | Block SMS | SMS | Phone Number | User Account |
| ADRD | | | | | | √ | | | | | | | |
| AnserverBot | | | | | | √ | | | √† | | | | |
| Asroot | | | | √ | | | | | | | | | |
| BaseBridge | | √ | | | | √ | | √ | √† | √ | | | |
| BeanBot | | | | | | √ | | √ | √† | √ | | √ | |
| BgServ | | | | | | √ | | | √† | √ | | √ | |
| CoinPirate | | | | | | √ | | | √† | √ | √ | | |
| Crusewin | | | | | | √ | | | √ | √ | √ | | |
| DogWars | | | | | | | | | √ | | | | |
| DroidCoupon | | √ | | | | √ | | | | | | | |
| DroidDeluxe | | √ | | | | | | | | | | | |
| DroidDream | √ | √ | | | | √ | | | | | | | |
| DroidDreamLight | | | | | | √ | | | | | | | √ |
| DroidKungFu1 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFu2 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFu3 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFu4 | | | | | | √ | | | | | | | |
| DroidKungFu5 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFuUpdate | | | | | | | | | | | | | |
| Endofday | | | | | | √ | | | √ | | | √ | |
| FakeNetflix | | | | | | | | | | | | | √ |
| FakePlayer | | | | | | | | | √‡ | | | | |
| GamblerSMS | | | | | | | | | | | √ | | |
| Geinimi | | | | | | √ | | √ | √† | √ | √ | √ | |
| GGTracker | | | | | | | | | √‡ | √ | √ | √ | |
| GingerMaster | | | √ | | | √ | | | | | | √ | |
| GoldDream | | | | | | √ | | √ | √† | | √ | √ | |
| Gone60 | | | | | | | | | | | √ | | |
| GPSSMSSpy | | | | | | | | | √ | | | | |
| HippoSMS | | | | | | | | | √‡ | √ | | | |
| Jifake | | | | | | | | | √‡ | | | | |
| jSMSHider | | | | | | √ | | | √† | √ | | √ | |
| KMin | | | | | | √ | | | √† | √ | | | |
| Lovetrap | | | | | | | | | √† | √ | | | |
| NickyBot | | | | | | | √ | | √ | | √ | | |
| Nickyspy | | | | | | √ | | | √ | | √ | | |
| Pjapps | | | | | | √ | | | √† | √ | | √ | |
| Plankton | | | | | | √ | | | | | | | |
| RogueLemon | | | | | | √ | | | √† | √ | √ | | |
| RogueSPPush | | | | | | | | | √‡ | √ | | | |
| SMSReplicator | | | | | | | | | √ | | √ | | |
| SndApps | | | | | | | | | | | | | √ |
| Spitmo | | | | | | √ | | | √† | √ | √ | √ | |
| TapSnake | | | | | | | | | | | | | |
| Walkinwat | | | | | | | | | √ | | | | |
| YZHC | | | | | | √ | | | √† | √ | | √ | |
| zHash | √ | | | | | | | | | | | | |
| Zitmo | | | | | | | | | | | √ | | |
| Zsone | | | | | | | | | √‡ | √ | | | |
| *number of families* | *6* | *8* | *1* | *1* | *4* | *27* | *1* | *4* | *28* | *17* | *13* | *15* | *3* |
| *number of samples* | *389* | *440* | *4* | *8* | *363* | *1171* | *1* | *246* | *571* | *315* | *138* | *563* | *43* |



(a) Top 20 Permissions Requested By 1260 Malware Samples

(b) Top 20 Permissions Requested by 1260 Top Free (Benign) Apps on the Offical Android Market

Figure 5. The Comparison of Top 20 Requested Permissions by Malicious and Benign Apps

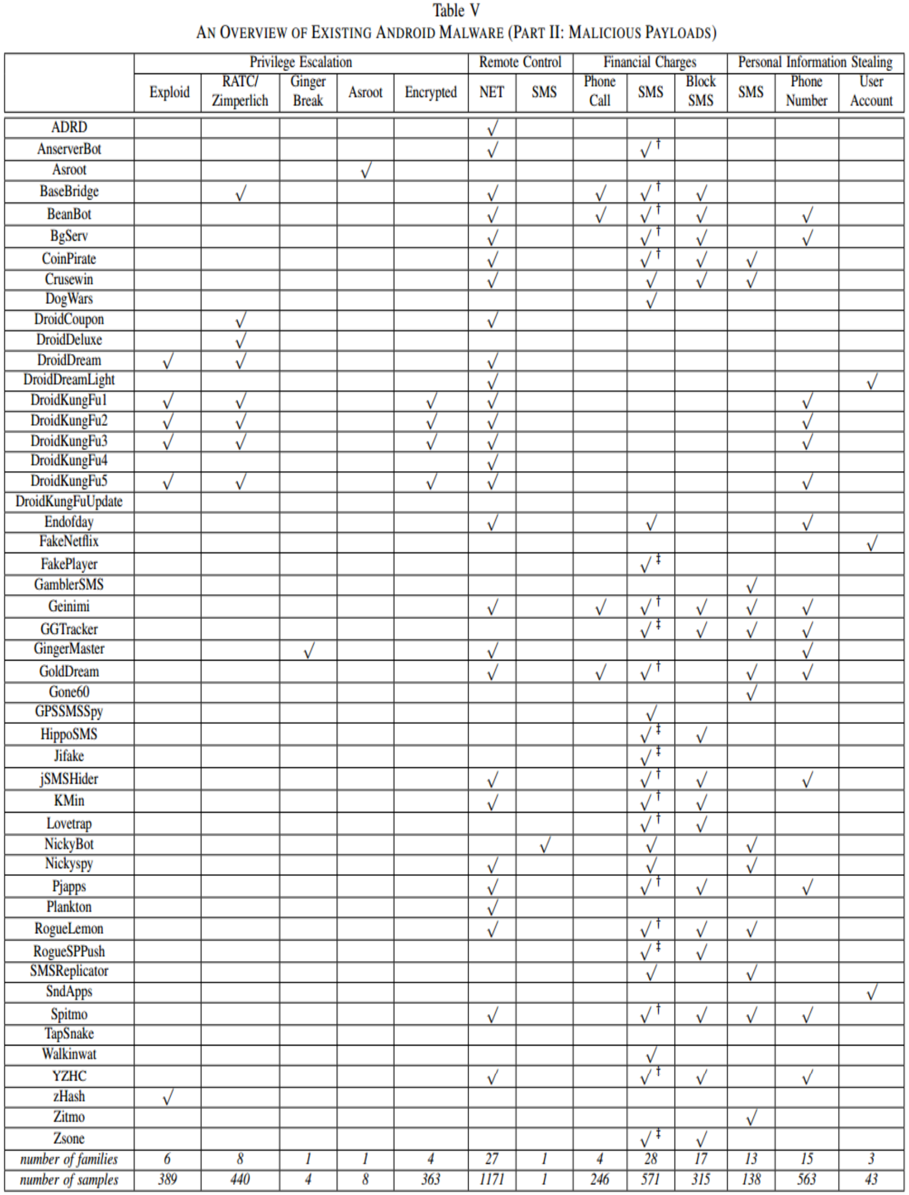


Table V
AN OVERVIEW OF EXISTING ANDROID MALWARE (PART II: MALICIOUS PAYLOADS)

| | Privilege Escalation | | | | | Remote Control | | Financial Charges | | | Personal Information Stealing | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Exploid | RATC/ Zimperlich | Ginger Break | Asroot | Encrypted | NET | SMS | Phone Call | SMS | Block SMS | SMS | Phone Number | User Account |
| ADRD | | | | | | √ | | | | | | | |
| AnserverBot | | | | | | √ | | | √$^{\dagger}$ | | | | |
| Asroot | | | | √ | | | | | | | | | |
| BaseBridge | | √ | | | | √ | | √ | √$^{\dagger}$ | √ | | | |
| BeanBot | | | | | | √ | | √ | √$^{\dagger}$ | √ | | √ | |
| BgServ | | | | | | √ | | | √$^{\dagger}$ | √ | | √ | |
| CoinPirate | | | | | | √ | | | √$^{\dagger}$ | √ | √ | | |
| Crusewin | | | | | | √ | | | √ | √ | √ | | |
| DogWars | | | | | | | | | √ | | | | |
| DroidCoupon | | √ | | | | √ | | | | | | | |
| DroidDeluxe | | √ | | | | | | | | | | | |
| DroidDream | √ | √ | | | | √ | | | | | | | |
| DroidDreamLight | | | | | | √ | | | | | | | √ |
| DroidKungFu1 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFu2 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFu3 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFu4 | | | | | | √ | | | | | | | |
| DroidKungFu5 | √ | √ | | | √ | √ | | | | | | √ | |
| DroidKungFuUpdate | | | | | | | | | | | | | |
| Endofday | | | | | | √ | | | √ | | | √ | |
| FakeNetflix | | | | | | | | | | | | | √ |
| FakePlayer | | | | | | | | | √$^{\ddagger}$ | | | | |
| GamblerSMS | | | | | | | | | | | √ | | |
| Geinimi | | | | | | √ | | √ | √$^{\dagger}$ | √ | √ | √ | |
| GGTracker | | | | | | | | | √$^{\ddagger}$ | √ | √ | √ | |
| GingerMaster | | | √ | | | √ | | | | | | √ | |
| GoldDream | | | | | | √ | | √ | √$^{\dagger}$ | | √ | √ | |
| Gone60 | | | | | | | | | | | √ | | |
| GPSSMSSpy | | | | | | | | | √ | | | | |
| HippoSMS | | | | | | | | | √$^{\ddagger}$ | √ | | | |
| Jifake | | | | | | | | | √$^{\ddagger}$ | | | | |
| jSMSHider | | | | | | √ | | | √$^{\dagger}$ | √ | | √ | |
| KMin | | | | | | √ | | | √$^{\dagger}$ | √ | | | |
| Lovetrap | | | | | | | | | √$^{\dagger}$ | √ | | | |
| NickyBot | | | | | | | √ | | √ | | √ | | |
| Nickyspy | | | | | | √ | | | √ | | √ | | |
| Pjapps | | | | | | √ | | | √$^{\dagger}$ | √ | | √ | |
| Plankton | | | | | | √ | | | | | | | |
| RogueLemon | | | | | | √ | | | √$^{\dagger}$ | √ | √ | | |
| RogueSPPush | | | | | | | | | √$^{\ddagger}$ | √ | | | |
| SMSReplicator | | | | | | | | | √ | | √ | | |
| SndApps | | | | | | | | | | | | | √ |
| Spitmo | | | | | | √ | | | √$^{\dagger}$ | √ | √ | √ | |
| TapSnake | | | | | | | | | | | | | |
| Walkinwat | | | | | | | | | √ | | | | |
| YZHC | | | | | | √ | | | √$^{\dagger}$ | √ | | √ | |
| zHash | √ | | | | | | | | | | | | |
| Zitmo | | | | | | | | | | | √ | | |
| Zsone | | | | | | | | | √$^{\ddagger}$ | √ | | | |
| *number of families* | *6* | *8* | *1* | *1* | *4* | *27* | *1* | *4* | *28* | *17* | *13* | *15* | *3* |
| *number of samples* | *389* | *440* | *4* | *8* | *363* | *1171* | *1* | *246* | *571* | *315* | *138* | *563* | *43* |

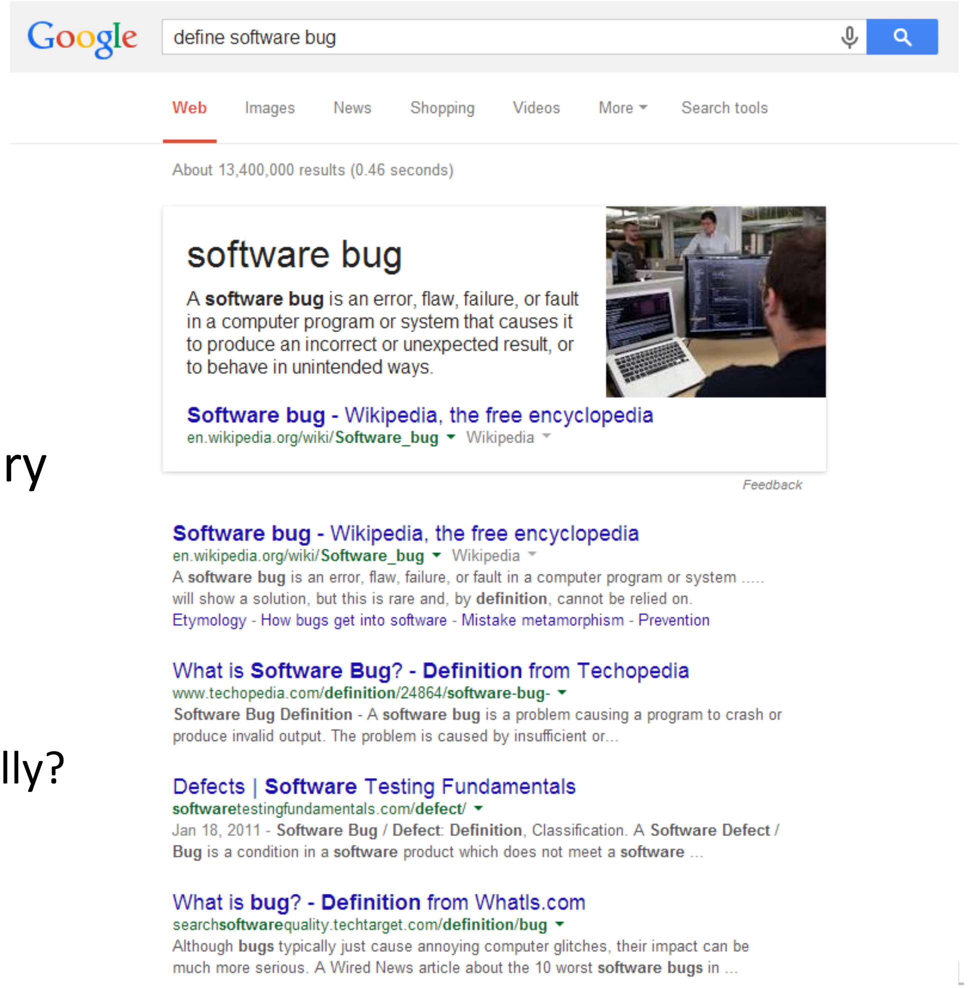
# Permission Abuse

- How do we know if an app is abusing permissions?
- https://developer.android.com/reference/android/Manifest.permission.html
  - 146 (documented) permissions as of Android API 19

| Behavior | App Purpose | Classification |
|---|---|---|
| Send location to Internet | Phone locator | Benign |
| Send location to Internet | Podcast player | Malicious |
| Selectively block SMS messages | Ad blocker | Benign |
| Selectively block SMS messages | Navigation | Malicious |

# Let’s define a “bug”

- Unintentional error, flaw, failure, fault
- Examples: Rounding errors, null pointers, infinite loops, stack overflows, race conditions, memory leaks, business logic flaws…
- Is a software bug malware?
  - What if I added the bug intentionally?



define software bug

About 13,400,000 results (0.46 seconds)

software bug

A **software bug** is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

**Software bug** - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Software_bug** Wikipedia

**Software bug** - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Software_bug** Wikipedia
A **software bug** is an error, flaw, failure, or fault in a computer program or system ..... will show a solution, but this is rare and, by **definition**, cannot be relied on.
Etymology - How bugs get into software - Mistake metamorphism - Prevention

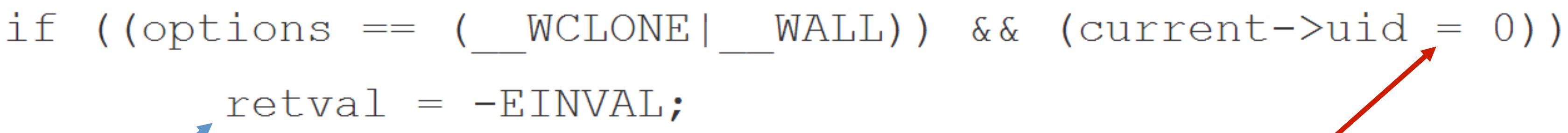
What is **Software Bug**? - **Definition** from Techopedia
www.techopedia.com/**definition**/24864/**software-bug**-
**Software Bug Definition** - A **software bug** is a problem causing a program to crash or produce invalid output. The problem is caused by insufficient or...

Defects | **Software** Testing Fundamentals
**software**testingfundamentals.com/**defect**/
Jan 18, 2011 - **Software Bug** / **Defect**: **Definition**, Classification. A **Software Defect** / **Bug** is a condition in a **software** product which does not meet a **software** ...

What is **bug**? - **Definition** from WhatIs.com
search**software**quality.techtarget.com/**definition**/**bug**
Although **bugs** typically just cause annoying computer glitches, their impact can be much more serious. A Wired News article about the 10 worst **software bugs** in ...

# A bug or malware?

- Context: Found in a CVS commit to the Linux Kernel source



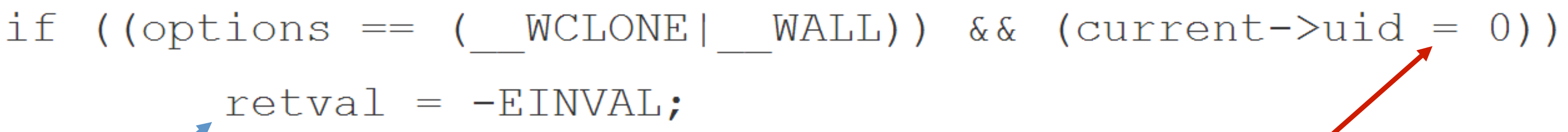
```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
        retval = -EINVAL;
```

Hint: This never executes…

"=" vs. "==“ is a subtle yet important difference!
Would grant root privilege to any user that knew how to trigger this condition.

# Malware: Linux Backdoor Attempt (2003)

- https://freedom-to-tinker.com/blog/felten/the-linux-backdoor-attempt-of-2003/

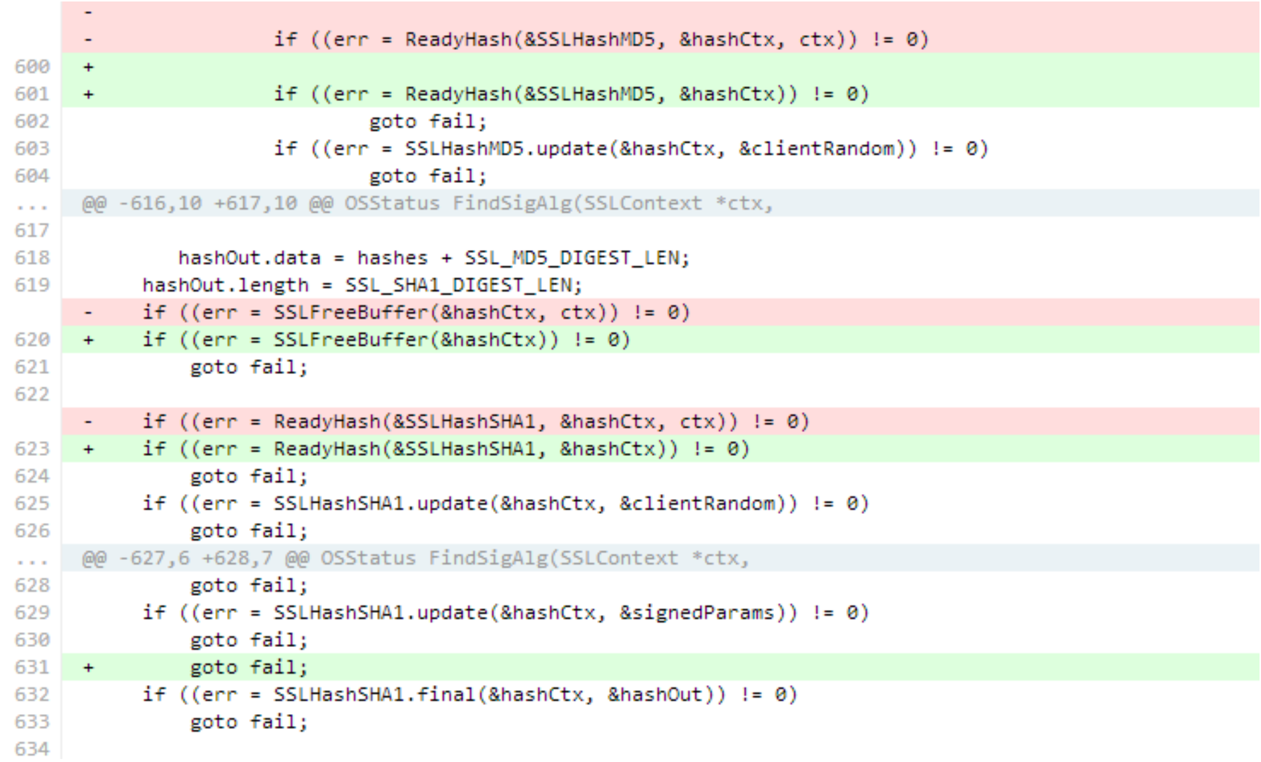


```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
        retval = -EINVAL;
```

Hint: This never executes…

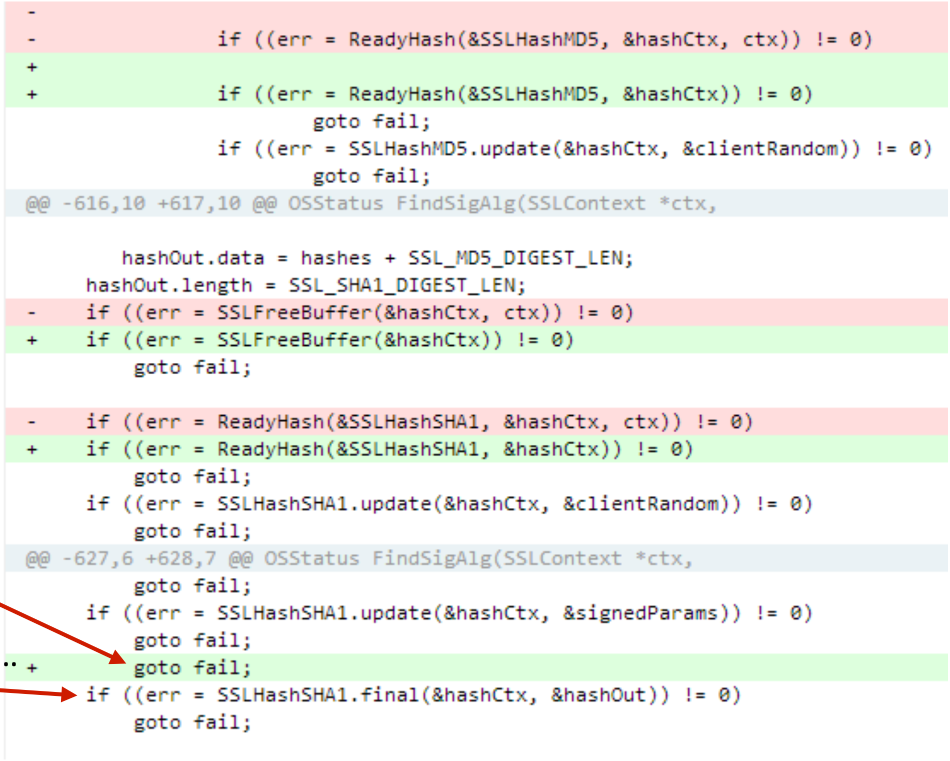
"=" vs. "==" is a subtle yet important difference!
Would grant root privilege to any user that knew how to trigger this condition.

# A bug or malware?



```
-
-               if ((err = ReadyHash(&SSLHashMD5, &hashCtx, ctx)) != 0)
+
+               if ((err = ReadyHash(&SSLHashMD5, &hashCtx)) != 0)
                        goto fail;
                if ((err = SSLHashMD5.update(&hashCtx, &clientRandom)) != 0)
                        goto fail;
@@ -616,10 +617,10 @@ OSStatus FindSigAlg(SSLContext *ctx,

        hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
     hashOut.length = SSL_SHA1_DIGEST_LEN;
-    if ((err = SSLFreeBuffer(&hashCtx, ctx)) != 0)
+    if ((err = SSLFreeBuffer(&hashCtx)) != 0)
         goto fail;

-    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx, ctx)) != 0)
+    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
         goto fail;
     if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
         goto fail;
@@ -627,6 +628,7 @@ OSStatus FindSigAlg(SSLContext *ctx,
         goto fail;
     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
         goto fail;
+        goto fail;
     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
         goto fail;

```

# A bug or malware?

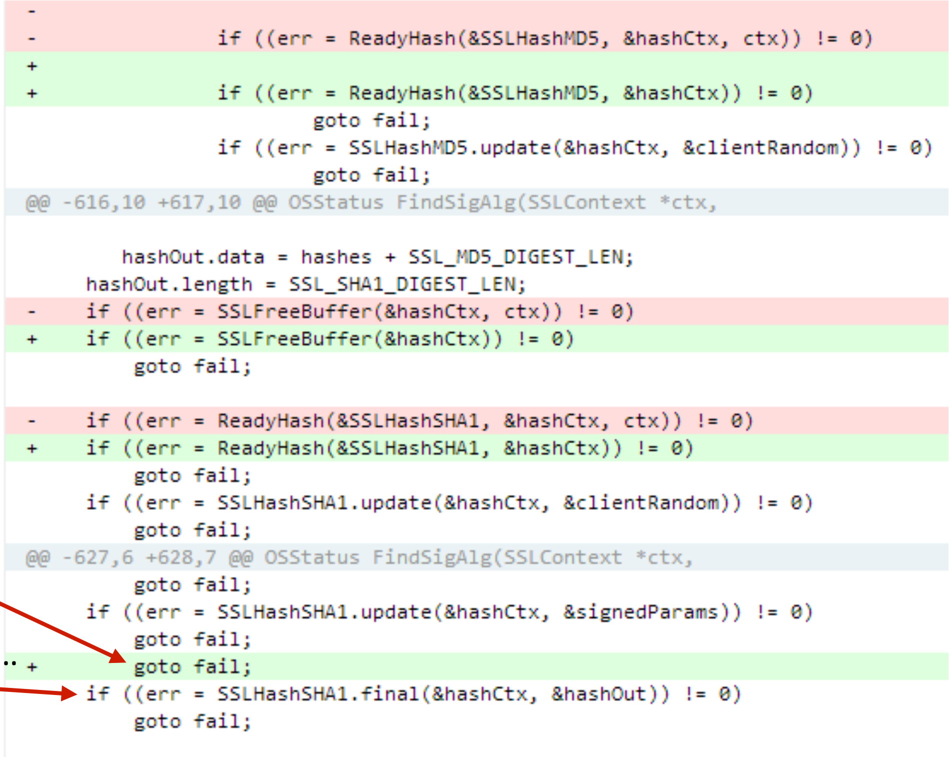


```
-
-               if ((err = ReadyHash(&SSLHashMD5, &hashCtx, ctx)) != 0)
+
+               if ((err = ReadyHash(&SSLHashMD5, &hashCtx)) != 0)
                        goto fail;
                if ((err = SSLHashMD5.update(&hashCtx, &clientRandom)) != 0)
                        goto fail;
@@ -616,10 +617,10 @@ OSStatus FindSigAlg(SSLContext *ctx,

        hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
     hashOut.length = SSL_SHA1_DIGEST_LEN;
-    if ((err = SSLFreeBuffer(&hashCtx, ctx)) != 0)
+    if ((err = SSLFreeBuffer(&hashCtx)) != 0)
         goto fail;

-    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx, ctx)) != 0)
+    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
         goto fail;
     if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
         goto fail;
@@ -627,6 +628,7 @@ OSStatus FindSigAlg(SSLContext *ctx,
         goto fail;
     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
         goto fail;
+        goto fail;
     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
         goto fail;
```

Always goto fail

Never does the check to verify server authenticity…

# Bug?: Apple SSL CVE-2014-1266



```
-
-               if ((err = ReadyHash(&SSLHashMD5, &hashCtx, ctx)) != 0)
+
+               if ((err = ReadyHash(&SSLHashMD5, &hashCtx)) != 0)
                        goto fail;
                if ((err = SSLHashMD5.update(&hashCtx, &clientRandom)) != 0)
                        goto fail;
@@ -616,10 +617,10 @@ OSStatus FindSigAlg(SSLContext *ctx,

         hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
      hashOut.length = SSL_SHA1_DIGEST_LEN;
-     if ((err = SSLFreeBuffer(&hashCtx, ctx)) != 0)
+     if ((err = SSLFreeBuffer(&hashCtx)) != 0)
          goto fail;

-     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx, ctx)) != 0)
+     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
          goto fail;
      if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
          goto fail;
@@ -627,6 +628,7 @@ OSStatus FindSigAlg(SSLContext *ctx,
          goto fail;
      if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
          goto fail;
+        goto fail;
      if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
          goto fail;
```

Always goto fail

Never does the check to verify server authenticity…

- Should have been caught by automated tools
- Survived almost a year
- Affected OSX and iOS

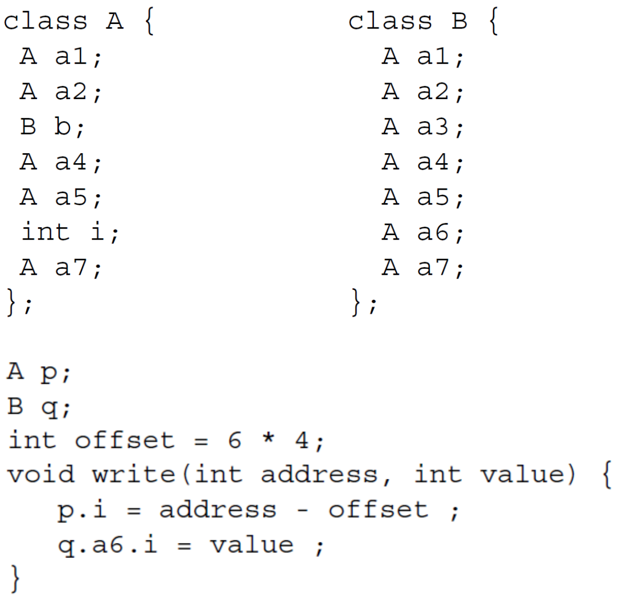
# A bug or malware?

```
class A {
 A a1;
 A a2;
 B b;
 A a4;
 A a5;
 int i;
 A a7;
};
```

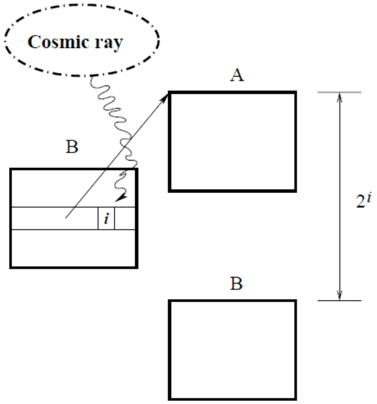
```
class B {
  A a1;
  A a2;
  A a3;
  A a4;
  A a5;
  A a6;
  A a7;
};
```

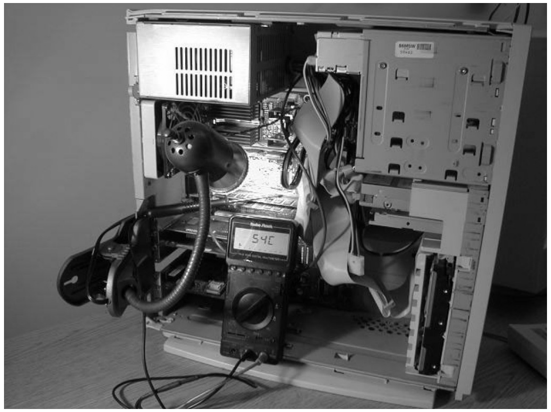
# Malware: VM escape using bit flips

- Govindavajhala, S.; Appel, AW., "Using memory errors to attack a virtual machine," *Proceedings of IEEE Symposium on Security and Privacy,* pp.154-165, May 2003.



```
class A {              class B {
 A a1;                   A a1;
 A a2;                   A a2;
 B b;                    A a3;
 A a4;                   A a4;
 A a5;                   A a5;
 int i;                  A a6;
 A a7;                   A a7;
};                     };

A p;
B q;
int offset = 6 * 4;
void write(int address, int value) {
   p.i = address - offset ;
   q.a6.i = value ;
}
```



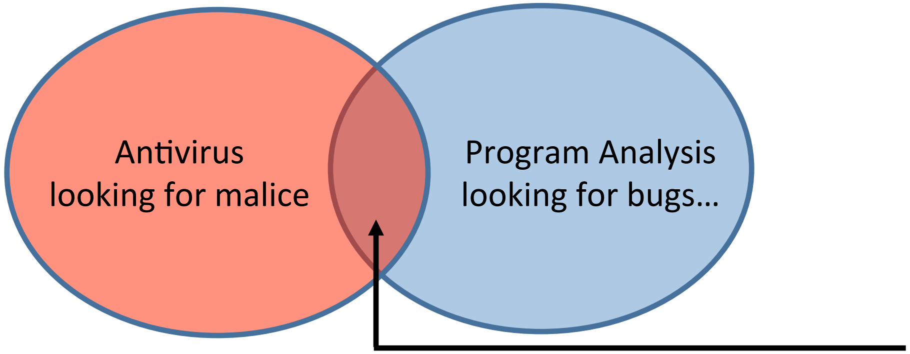


Wait for a bit flip to obtain two pointers of incompatible types that point to the same location to circumvent the type system and execute arbitrary code in the program address space.

# So what’s your point?

- Both bugs and malware have catastrophic consequences
- Some bugs are indistinguishable from malware
  - Plausible deniability, malicious intent cannot be determined from code
- Some issues can be found automatically, but not all
- Novel attacks can be extremely hard to detect

Are we doing ourselves a disservice by labeling these as separate problems?



Next time you own a box try dropping a program with an exploitable “bug”

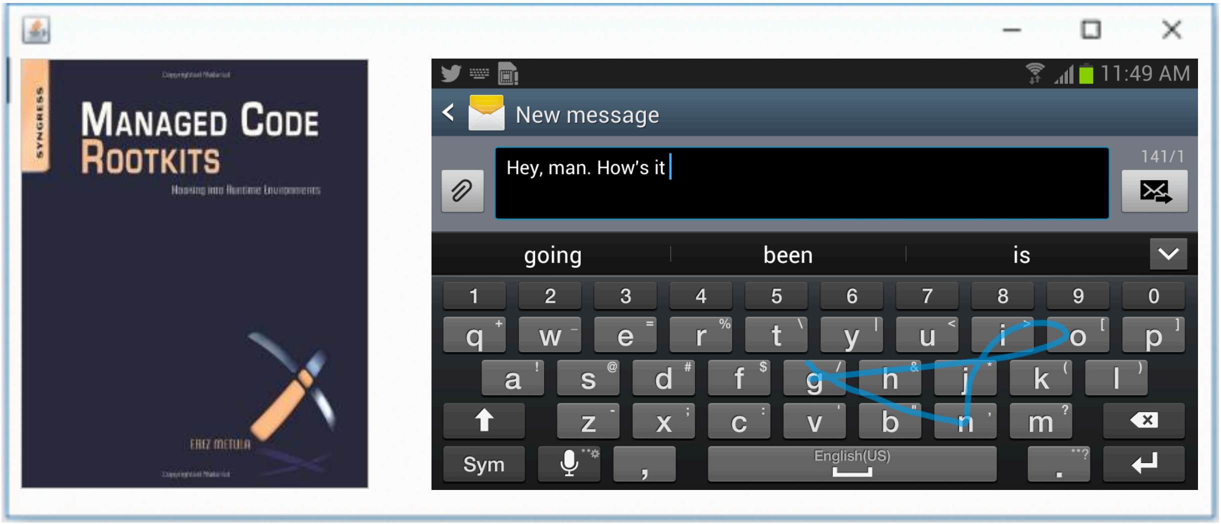
# SpellWrecker

- Consider a spell checker. Invert its logic and what do you get?
- How do we semantically detect the bad one?
- github.com/benjholla/spellwrecker

“Sometimes you have to demo a threat to spark a solution” - Barnaby Jack

# SpellWrecker

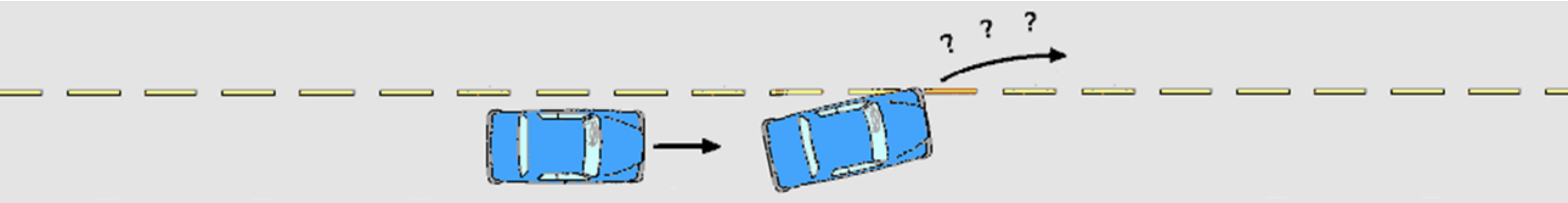
- Consider a spell checker. Invert its logic and what do you get?
- How do we semantically detect the bad one?
- github.com/benjholla/spellwrecker



Hey, man. How's it

# Hypothetical Malware

- Cars are becoming drive-by-wire
- Electronic Stability Controls (ESC) are being added to SUVs for rollover prevention



- Invert logic on roll over prevention systems
- Plenty of evil ways to implement it, e.g. greedy algorithms
  - J. Bang-Jensen, G. Gutin, and A. Yeo, "When the greedy algorithm fails," Discrete Optimizations, vol. 1, no. 2, pp. 121–127, Nov. 2004.
- Legitimate bugs are hard enough, how can we hope to find illegitimate bugs?

# Our Adversary

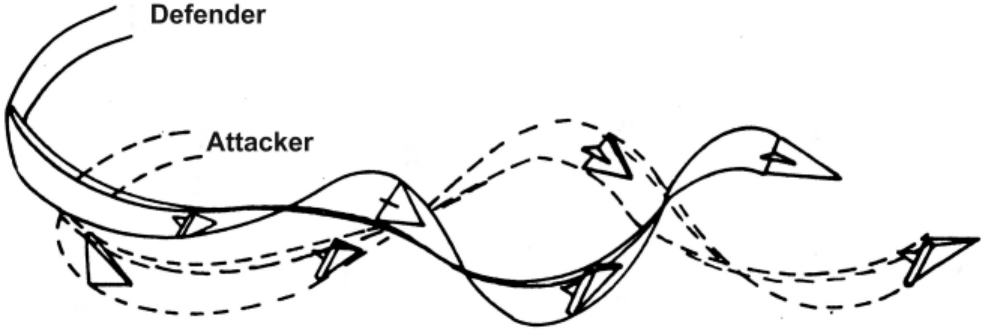
- We are facing formidable adversaries…
  - Capable of sophisticated attacks
  - Familiar with the domain
  - Malware is customized for a specific malicious purpose
  - Well motivated, funded, staffed, etc.
- New attacks demand new analysis techniques
  - Signature detection fails here
  - What does that process look like?

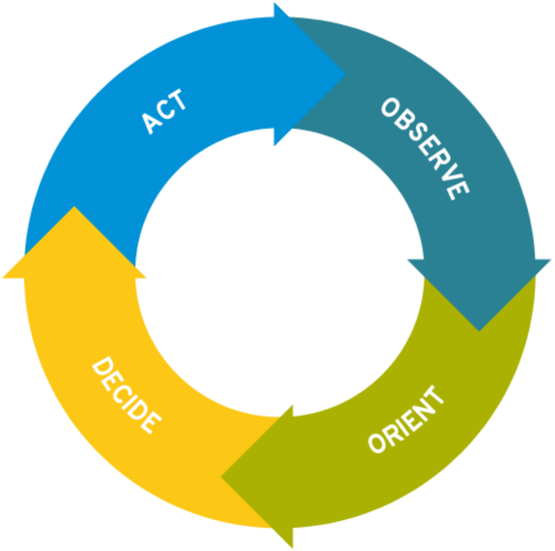
# Agenda

- Novel and Sophisticated Malware
- OODA Loops with Atlas

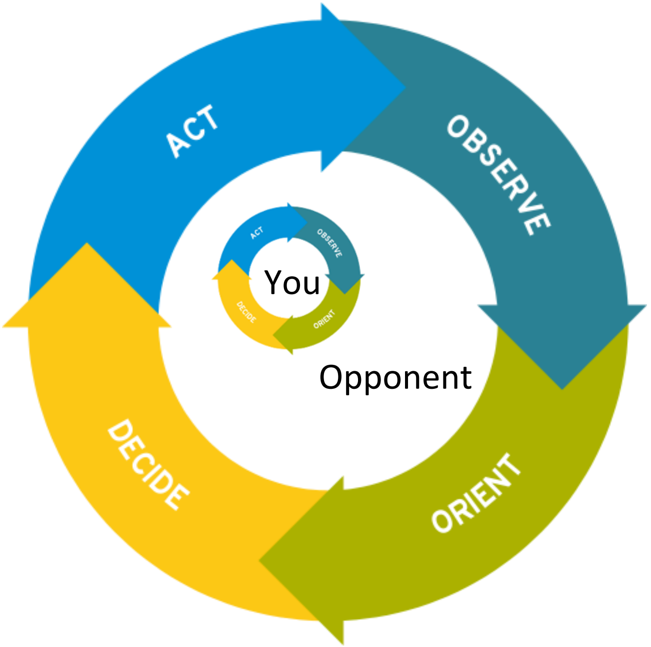
# John Boyd’s OODA Loop

“Security is a process, not a product” – Bruce Schneier





# John Boyd’s OODA Loop



Our opponent

- Time
- Evolution of malware

“…IA > AI, that is, that intelligence amplifying systems can, at any given level of available systems technology, beat AI systems. That is, a machine and a mind can beat a mind-imitating machine working by itself.”
– Fred Brooks

# Timelapse of an Audit

- App Description: A network port scanner with the ability to write custom network scans and generate reports.
- Malware: The application contains a custom programming language and compiler for implementing network scan programs. A malicious binary is assembled in the custom language's VM memory from several binary blobs in the local database and is sent across the network to Windows hosts.