

Cyber@UC Meeting 77

Magical Goats

If You're New!

- Join our Slack: cyberatuc.slack.com
- Check out our website: cyberatuc.org
- **SIGN IN!** (*Slackbot will post the link in #general every Wed@6:30*)
- Feel free to get involved with one of our committees:
Content Finance Public Affairs Outreach Recruitment Lab
- Ongoing work in our research lab!



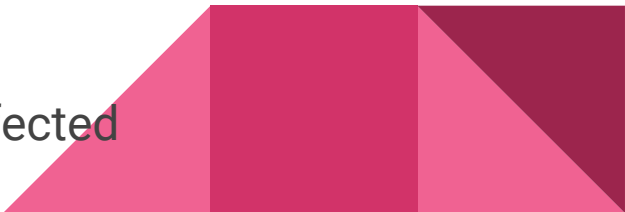
Announcements

- **Battelle Visit** Nov. 20th from Aaron McCanty!
- Lab committee volunteers!
- Merchandise on the way, Online Shop
- Cyclones game social event Nov 17th!
- CTF team training Nov 17th 11am-6pm



Weekly News

StatCounter Hijacked Leads To Bitcoin Theft

- Gate.io crypto exchange compromised by their web analytics service, StatCounter
 - Malicious code found on >700k websites, bundled with traffic tracking code
 - Replaced tracking script with code target Gate.io customers
 - StatCounter is a bit old, but very popular real-time web analytics platform
 - Reported as being used in >2 million websites and >10 billion page views/month
 - Malicious code made to target a gate.io specific URI
 - Code replaced destination of bitcoin address with that of attacker's
 - Generated a new address everytime
 - Gate.io no longer uses StatCounter
 - Gate.io has not released stats on how many were affected
- 

VirtualBox Flaw, Escaping The Sandbox

- Vulnerability for Intel PRO 1000 MT Desktop network card when network mode is set to NAT, memory corruption
- OS type does not matter
- Poc published to GitHub, link in article
- Allows a malicious attacker with root privs in guest OS to escape and run arbitrary code in the application layer (ring 3, low privs) on the host
- Could leave host open to other vulnerabilities, like privilege escalation
- Not yet patched



Bleeding Bit

- Two new zero day vulns found by Armis, the guys who caught BlueBorne
- Allow arbitrary code execution and full C&C w/o auth
 - Ex. Insulin pumps, pacemakers, credit card readers, routers
- Vulns in bluetooth chips made by Texas Instruments
- Sending more traffic to the BLE chip causes a buffer overflow, allow malicious code execution, requires physical proximity
- Firmware update feature, Over the Air firmware Download (OAD)
- All Aruba devices share OAD password, obtainable by sniffing legitimate packets or reverse-engineering the firmware
 - Attacker can send a malicious firmware update
- Patches released last Thursday



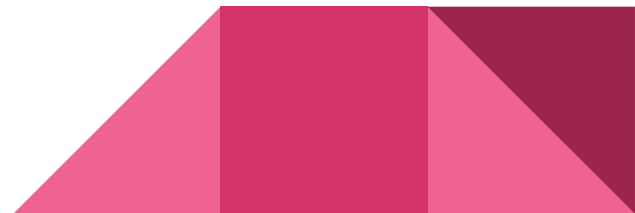
Recommended Reading

<https://thehackernews.com/2018/11/android-in-app-updates-api.html>

<https://krebsonsecurity.com/2018/11/u-s-secret-service-warns-id-thieves-are-abusing-usps-mail-scanning-service/>

<https://www.darkreading.com/vulnerabilities---threats/the-morris-worm-turns-30-/d/d-id/1333225>

<https://www.welivesecurity.com/2018/11/05/malware-1980s-brain-virus-morris-worm/>



Recommended Reading (continued)

<https://www.welivesecurity.com/2018/11/09/us-air-force-hackable-bug-bounty-program/>

<https://krebsonsecurity.com/2018/11/bug-bounty-hunter-ran-isp-doxing-service/>

<https://thehackernews.com/2018/11/gaming-server-ddos-attack.html>

<https://www.welivesecurity.com/2018/11/08/cyber-insurance-question/>

<https://www.welivesecurity.com/2018/11/09/emotet-launches-major-new-spam-campaign/>

Recommended Reading (continued)

<https://thehackernews.com/2018/11/portsmash-intel-vulnerability.html>

<https://thehackernews.com/2018/11/self-encrypting-ssd-hacking.html>

<https://thehackernews.com/2018/11/woocommerce-wordpress-hacking.html>





**OUR FEATURE
PRESENTATION**

Workshop: Goat Disassembly

“I can’t wait to be in this goat” - You, right now.

The Topics Today Go Something Exactly Like This

- Quick touch on Assembly & Disassembly
- The RE tools in Kali and IDA
- Battelle's *Feed the Magical Goat* CTF



Assembly?!

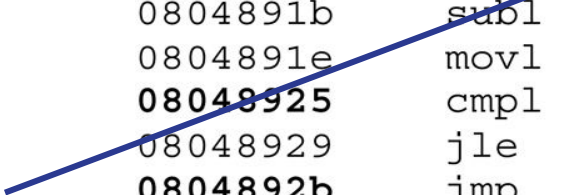
- Nearest possible human readable version of machine code
- Everything is either stored in registers, which can be compared to variables, or in literals values (ints/strings)
- Functions are called subprocesses
- First years take note

```
08048918    pushl   %ebp
08048919    movl   %esp,%ebp
0804891b    subl   $0x4,%esp
0804891e    movl   $0x0,0xffffffffc(%ebp)
08048925    cmpl   $0x63,0xffffffffc(%ebp)
08048929    jle    08048930
0804892b    jmp    08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl   0xffffffffc(%ebp),%eax
08048933    pushl   %eax
08048934    pushl   $0x8049418
08048939    call   080487c0 <printf>
0804893e    addl   $0x8,%esp
08048941    incl   0xffffffffc(%ebp)
08048944    jmp    08048925
08048946    nop
08048947    nop
08048948    xorl   %eax,%eax
0804894a    jmp    0804894c
0804894c    leave
0804894d    ret
```

Registers?!

- Usually prefixed with a “%”
- You only have 8 that you should really be looking at / using
- Basically 32 bit pointers / ints
 - Pointers are ints
- Google the names for x64, there's plenty of tables

```
08048918    pushl    %ebp
08048919    movl    %esp,%ebp
0804891b    subl    $0x4,%esp
0804891e    movl    $0x0,0xffffffffc(%ebp)
08048925    cmpl    $0x63,0xffffffffc(%ebp)
08048929    jle     08048930
0804892b    jmp     08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl    0xffffffffc(%ebp),%eax
08048933    pushl   %eax
08048934    pushl   $0x8049418
08048939    call    080487c0 <printf>
0804893e    addl    $0x8,%esp
08048941    incl    0xffffffffc(%ebp)
08048944    jmp     08048925
08048946    nop
08048947    nop
08048948    xorl    %eax,%eax
0804894a    jmp     0804894c
0804894c    leave
0804894d    ret
```



Subprocesses

- Equivalent of functions
- Functions arguments are **pushed** onto the stack
- The subprocess is **called**
- Subprocess **return** as functions do


```
08048918    pushl   %ebp
08048919    movl   %esp,%ebp
0804891b    subl   $0x4,%esp
0804891e    movl   $0x0,0xffffffffc(%ebp)
08048925    cmpl   $0x63,0xffffffffc(%ebp)
08048929    jle    08048930
0804892b    jmp    08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl   0xffffffffc(%ebp),%eax
08048933    pushl  %eax
08048934    pushl  $0x8049418
08048939    call   080487c0 <printf>
0804893e    addl   $0x8,%esp
08048941    incl   0xffffffffc(%ebp)
08048944    jmp    08048925
08048946    nop
08048947    nop
08048948    xorl   %eax,%eax
0804894a    jmp    0804894c
0804894c    leave
0804894d    ret
```


Conditionals

- Variables can be **compared**
- **Jumps** in execution can be made depending on comparisons
- Jumps can also be unconditional (like goto & break)
- C if statements are typically compares and jumps sequentially executed

```
08048918    pushl   %ebp
08048919    movl   %esp,%ebp
0804891b    subl   $0x4,%esp
0804891e    movl   $0x0,0xffffffff(%ebp)
08048925    cmpl   $0x63,0xffffffff(%ebp)
08048929    jle   08048930
0804892b    jmp   08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl   0xffffffff(%ebp),%eax
08048933    pushl   %eax
08048934    pushl   $0x8049418
08048939    call   080487c0 <printf>
0804893e    addl   $0x8,%esp
08048941    incl   0xffffffff(%ebp)
08048944    jmp   08048925
08048946    nop
08048947    nop
08048948    xorl   %eax,%eax
0804894a    jmp   0804894c
0804894c    leave
0804894d    ret
```

Other Notes

- Strings are typically stored as static character arrays then copied later when they are used
- This is basically just C with harder syntax and heavy use of goto
- Every instruction has a position offset value compared to where the program's base memory address is 

```
08048918    pushl    %ebp
08048919    movl    %esp,%ebp
0804891b    subl    $0x4,%esp
0804891e    movl    $0x0,0xffffffffc(%ebp)
08048925    cmpl    $0x63,0xffffffffc(%ebp)
08048929    jle     08048930
0804892b    jmp     08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl    0xffffffffc(%ebp),%eax
08048933    pushl   %eax
08048934    pushl   $0x8049418
08048939    call    080487c0 <printf>
0804893e    addl    $0x8,%esp
08048941    incl    0xffffffffc(%ebp)
08048944    jmp     08048925
08048946    nop
08048947    nop
08048948    xorl    %eax,%eax
0804894a    jmp     0804894c
0804894c    leave
0804894d    ret
```

Other Notes Cont.

- AT&T vs Intel Format
- **Move** operations just copy paste a register value into another register

```
08048918    pushl    %ebp
08048919    movl    %esp,%ebp
0804891b    subl    $0x4,%esp
0804891e    movl    $0x0,0xffffffffc(%ebp)
08048925    cmpl    $0x63,0xffffffffc(%ebp)
08048929    jle     08048930
0804892b    jmp     08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl    0xffffffffc(%ebp),%eax
08048933    pushl    %eax
08048934    pushl    $0x8049418
08048939    call    080487c0 <printf>
0804893e    addl    $0x8,%esp
08048941    incl    0xffffffffc(%ebp)
08048944    jmp     08048925
08048946    nop
08048947    nop
08048948    xorl    %eax,%eax
0804894a    jmp     0804894c
0804894c    leave
0804894d    ret
```

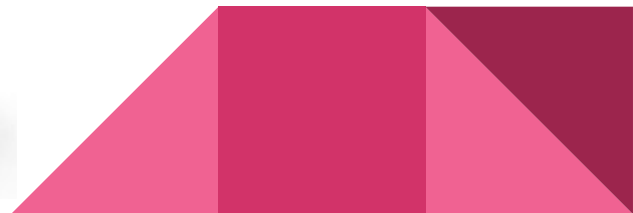
Disassembly

- All the 1337 HaX0rs do it
- You should too
- Process of taking apart binary programs, which are typically compiled from C/C++
- **Static analysis** - Just reading assembly code
- **Dynamic analysis** - running and debugging the program
- Basically just feed a binary in and assembly code comes out



Disassembly Tools in Kali Linux (and IDA)

Binary Tools (ELF / PE)	Android / Java Tools
diStorm3 IDA edb-debugger OllyDbg Valgrind YARA strings	apktool dex2jar jad jvasnoop jd-gui smali



Interactive Disassembler (IDA)

- Download the **free** version from <https://www.hex-rays.com/>
- Radare2 looks like a good alternative but I'm not familiar with it yet
- Grab the Magical Goat zip file from <https://www.battelle.org/cyber-challenge>
- I don't have any slides for IDA itself so we'll just go into it with the binary



A night sky with the Milky Way galaxy visible in the upper left. A dark flag is planted on a rocky mountain peak in the upper right. The background is a dark, starry sky.

BATTELLE

Aaron McCanty, who helped prepare the goat challenge, will be here next week and will be at CEAS tomorrow