Hack Night

@ Boston University 2016
Week 1

Overview

1. Intro, resources, what we'll learn, and how we'll have fun
2. The life of a binary, and static reverse engineering
3. Abstract machines, and what they mean to security
4. Reversing and cracking demo!

Legend:

Green links - BU Hack Night affiliated

Whoami

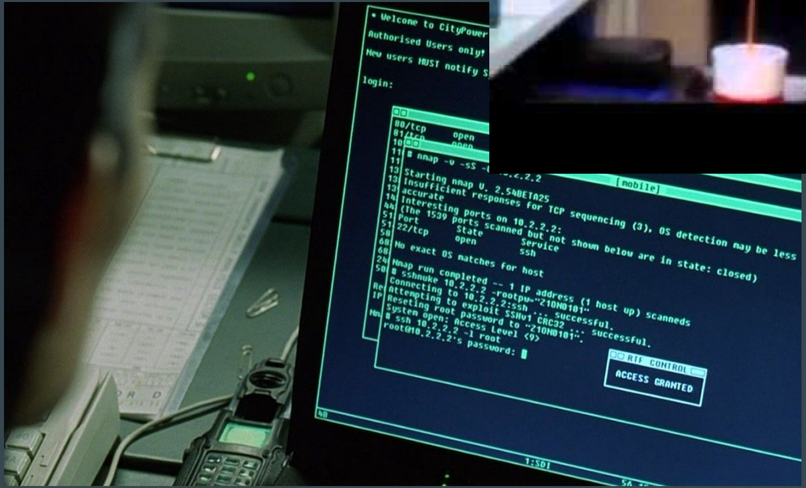
Eugene Kolo.

2000 - 2009	trolling IRC servers, botting and hacking some games, skiddieing off milw0rm, and failing at installing Linux
2009-2012	EE @ BU
2012-2014	SOC Verification on the PS4 at AMD
2015-	Cyber Security Engineer @ MITRE, MS CE @ BU

Jeff Crowell.

2009-2013	CE @ BU
2011-2013	CTF with BUILDS
2013-	CTF with Shellphish

Intro



```
* Welcome to CityPower
Authorised Users only!
New users MUST notify S

login:

80/tcp
81/tcp      open
            open
# nmap -v -sS -          .2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection may be less
accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: closed)
Port        State       Service
22/tcp      open        ssh
No exact OS matches for host
Nmap run completed -- 1 IP address (1 host up) scanneds
# sshnuke 10.2.2.2 -rootpw="Z10N0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z10N0101".
System open: Access Level <9>
# ssh 10.2.2.2 -l root
root@10.2.2.2's password: █
```

Real Intro

What this is

- Learn to hack
- Workshop oriented
- Meet fellow hackers
- Wargame for fame, and glory
- Open community to learn in
- Hands on

Down the rabbit hole...

What this is not

- ~~Unethical~~
- ~~Structured education~~
- ~~Selling to you why you should be here~~

EC521 - Cyber Security
EC700 - Malware, Vulnerability and Defense Analysis
CS558 - Network Security

Goals

- We want to develop all the skills necessary for modern:
  - Vulnerability research
  - Cracking/modifying software, and video games
  - Capture the flag competitions and wargames
  - Reverse engineering
  - Malware analysis
  - Exploit development
  - Hardware hacking
  - Securing systems
  - ....

Some fun stuff

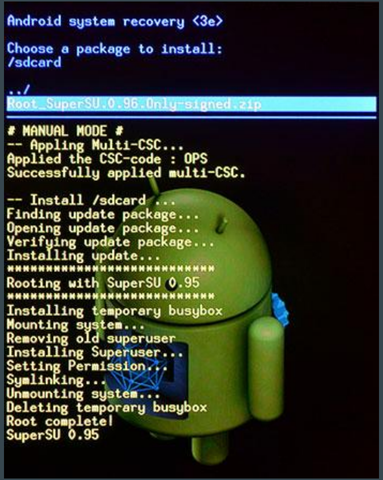
COREL Products Keygen : CORE

Program: Corel Draw Graphics Suite X5

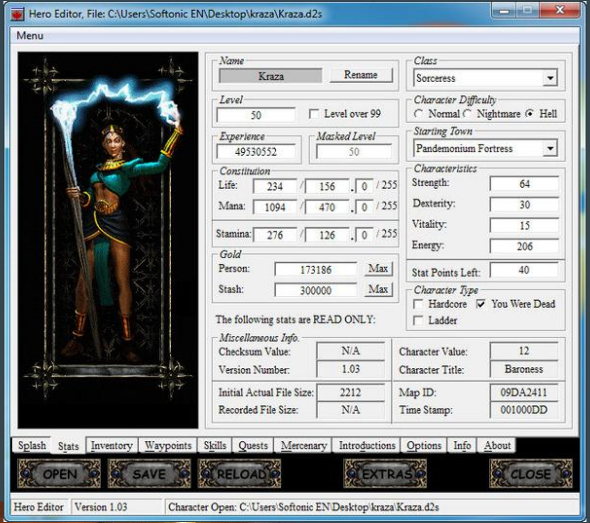
Installation Code: Enter Your Installation Code here !

Serial: [illegible]

Activation Code:



```
Android system recovery <3e>
Choose a package to install:
/sdcard

../
Root_SuperSU.0.96.Only-signed.zip

# MANUAL MODE #
-- Appling Multi-CSC...
Applied the CSC-code : OPS
Successfully applied multi-CSC.

-- Install /sdcard ...
Finding update package...
Opening update package...
Verifying update package...
Installing update...
****************************
Rooting with SuperSU 0.95
****************************
Installing temporary busybox
Mounting system...
Removing old superuser
Installing Superuser...
Setting Permission...
Symlinking...
Unmounting system...
Deleting temporary busybox
Root complete!
SuperSU 0.95
```



Hero Editor, File: C:\Users\Softonic EN\Desktop\kraza\Kraza.d2s

Name: Kraza
Class: Sorceress
Level: 50 — Level over 99 [ ]
Character Difficulty: Normal Nightmare [✓ Hell]
Experience: 49530552
Masked Level: 50
Starting Town: Pandemonium Fortress

Constitution
Life: 234 / 156 . 0 / 255
Mana: 1094 / 470 . 0 / 255
Stamina: 276 / 126 . 0 / 255

Characteristics
Strength: 64
Dexterity: 30
Vitality: 15
Energy: 206

Gold
Person: 173186 Max
Stash: 300000 Max

Stat Points Left: 40

Character Type: Hardcore [ ] — You Were Dead [✓] — Ladder [ ]

The following stats are READ ONLY:

Miscellaneous Info.
Checksum Value: N/A
Version Number: 1.03
Character Value: 12
Character Title: Baroness
Initial Actual File Size: 2212
Recorded File Size: N/A
Map ID: 09DA2411
Time Stamp: 001000DD

Hero Editor | Version 1.03 | Character Open: C:\Users\Softonic EN\Desktop\kraza\Kraza.d2s

Reality

- Hacking isn't easy
- Takes practice, training, learning, exchanging of information
  - Reverse engineering that video game to win can take days - and be so satisfying!
- Rewarding: mentally, financially, ethically

The best teacher is often times a search engine....

https://google.com

Getting started (⅓)

- Linux. Ubuntu prefered. C, assembly
  - Master the Linux command line.
    - http://overthewire.org/wargames/bandit/bandit0.html
  - Read, and write C. Proficiently.
    - https://learnxinyminutes.com/docs/c/
  - Read, and write Asm. Proficiently.
    - http://www.cs.virginia.edu/~evans/cs216/guides/x86.html
  - Know a scripting language. Python or Ruby are generally the two main ones
    - http://inventwithpython.com/chapters/
    - Speak w/ crowell for Ruby

Absolute minimum Linux and programming knowledge: http://security.cs.rpi.edu/courses/binexp-spring2015/lectures/1/01_lecture.pdf slides 27-59

Getting started (⅔)

- Install Ubuntu, on a virtual machine is fine.
    a. http://www.ubuntu.com/download/desktop, https://www.virtualbox.org/wiki/Downloads
- Some of our favourite tools:
    - https://github.com/eugenekolo/sec-tools
    - https://github.com/eugenekolo/win-sec-tools
    - file, md5sum, strings, readelf, objdump, ssh, xxd, echo, printf, ...
- Some good repos:
    - https://github.com/isislab?page=1
    - https://github.com/Gallopsled/pwntools
    - https://github.com/zardus/ctf-tools

Getting started (3/3)

- Resources:
  - https://github.com/buhacknight/main/tree/master/resources
- Books:
  - https://github.com/buhacknight/main/blob/master/resources/book_list.md

Learning with fun

- CTF Writeups
    - https://github.com/ctfs/
- CTF Challenges
    - https://github.com/buhacknight/ctfs
    - http://ctftime.org
- Wargames
    - http://smashthestack.org/
    - http://pwnable.kr/
    - http://overthewire.org/wargames/

Stay up to date and learn

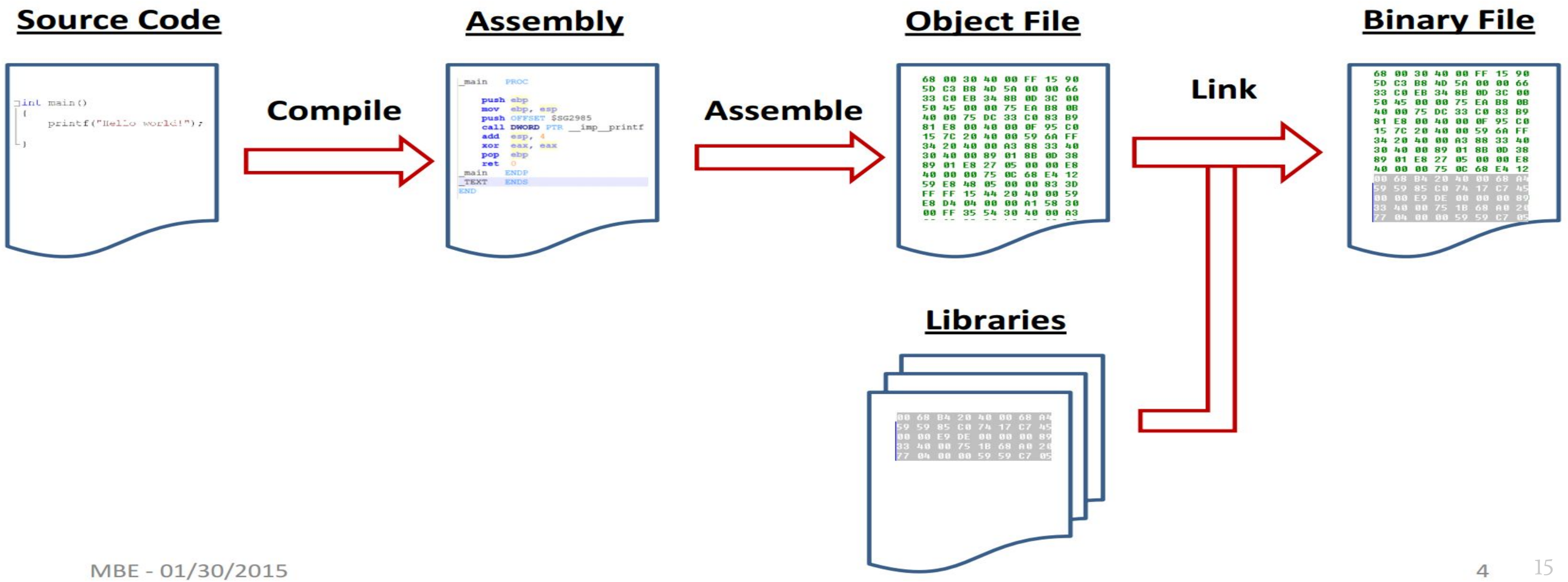
- Twitter, blogs, news sites
  - https://github.com/buhacknight/main/blob/master/resources/blog_roll.md
- Good idea to do write-ups yourself, or chronicle your journey on a blog
  - https://www.eugenekolo.com

Let's begin

1. Intro, resources, what we'll learn, how we'll have fun
2. The life of a binary and static reverse engineering
3. Abstract machines, and what they mean to security
4. Reversing and cracking demo!

Slides and references taken from: http://security.cs.rpi.edu/courses/binexp-spring2015/lectures/2/02_lecture.pdf by Jeremy Blackthorne

Compiling



```
int main()
{
    printf("Hello world!");
}
```

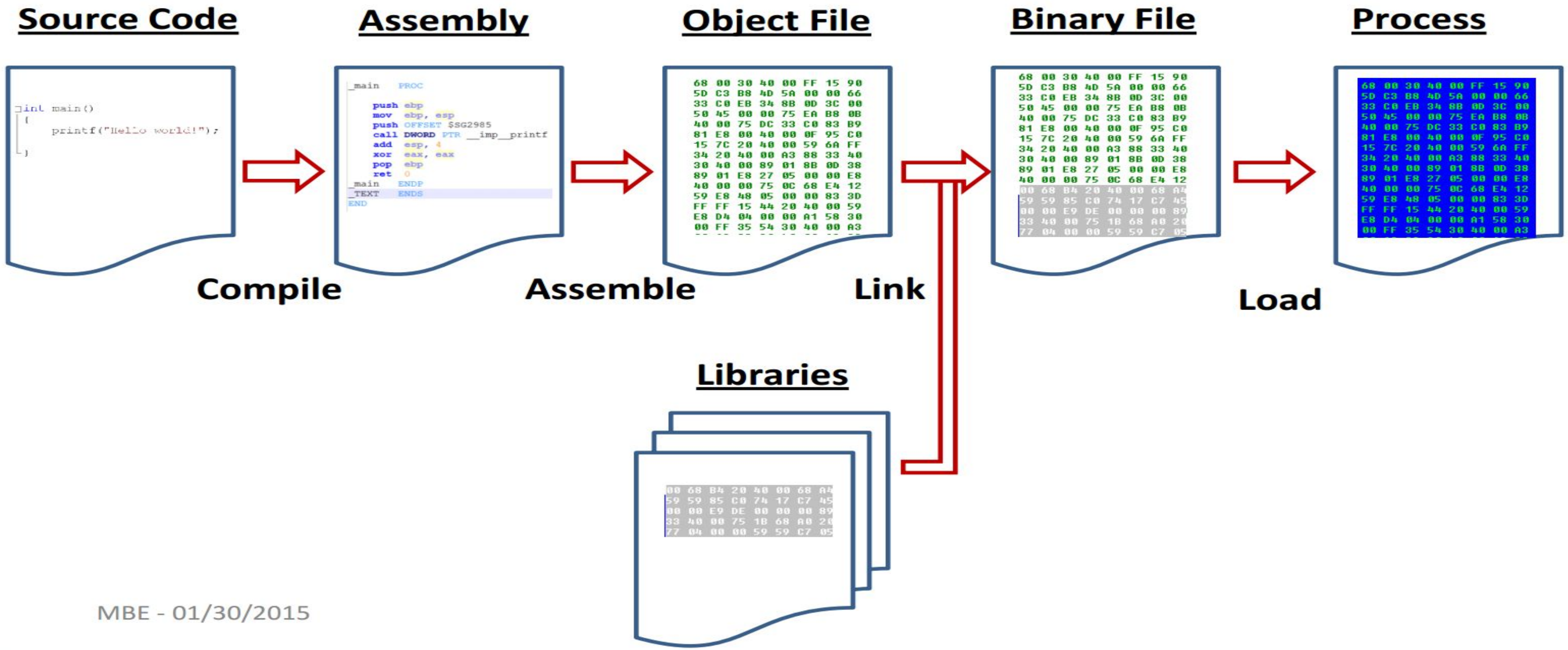
```
_main   PROC

    push ebp
    mov  ebp, esp
    push OFFSET $SG2985
    call DWORD PTR __imp__printf
    add  esp, 4
    xor  eax, eax
    pop  ebp
    ret  0
_main   ENDP
_TEXT   ENDS
END
```

```
68 00 30 40 00 FF 15 90
5D C3 B8 4D 5A 00 00 66
33 C0 EB 34 8B 0D 3C 00
50 45 00 00 75 EA B8 0B
40 00 75 DC 33 C0 83 B9
81 E8 00 40 00 0F 95 C0
15 7C 20 40 00 59 6A FF
34 20 40 00 A3 88 33 40
30 40 00 89 01 8B 0D 38
89 01 E8 27 05 00 00 E8
40 00 00 75 0C 68 E4 12
59 E8 48 05 00 00 83 3D
FF FF 15 44 20 40 00 59
E8 D4 04 00 00 A1 58 30
00 FF 35 54 30 40 00 A3
```

```
68 00 30 40 00 FF 15 90
5D C3 B8 4D 5A 00 00 66
33 C0 EB 34 8B 0D 3C 00
50 45 00 00 75 EA B8 0B
40 00 75 DC 33 C0 83 B9
81 E8 00 40 00 0F 95 C0
15 7C 20 40 00 59 6A FF
34 20 40 00 A3 88 33 40
30 40 00 89 01 8B 0D 38
89 01 E8 27 05 00 00 E8
40 00 00 75 0C 68 E4 12
00 68 B4 20 40 00 68 A4
59 59 85 C0 74 17 C7 45
00 00 E9 DE 00 00 00 89
33 40 00 75 1B 68 A0 20
77 04 00 00 59 59 C7 05
```

```
00 68 B4 20 40 00 68 A4
59 59 85 C0 74 17 C7 45
00 00 E9 DE 00 00 00 89
33 40 00 75 1B 68 A0 20
77 04 00 00 59 59 C7 05
```

Loading



```
int main()
{
    printf("Hello world!");
}
```

```
_main   PROC

    push ebp
    mov  ebp, esp
    push OFFSET $SG2985
    call DWORD PTR __imp__printf
    add  esp, 4
    xor  eax, eax
    pop  ebp
    ret  0
_main   ENDP
_TEXT   ENDS
END
```

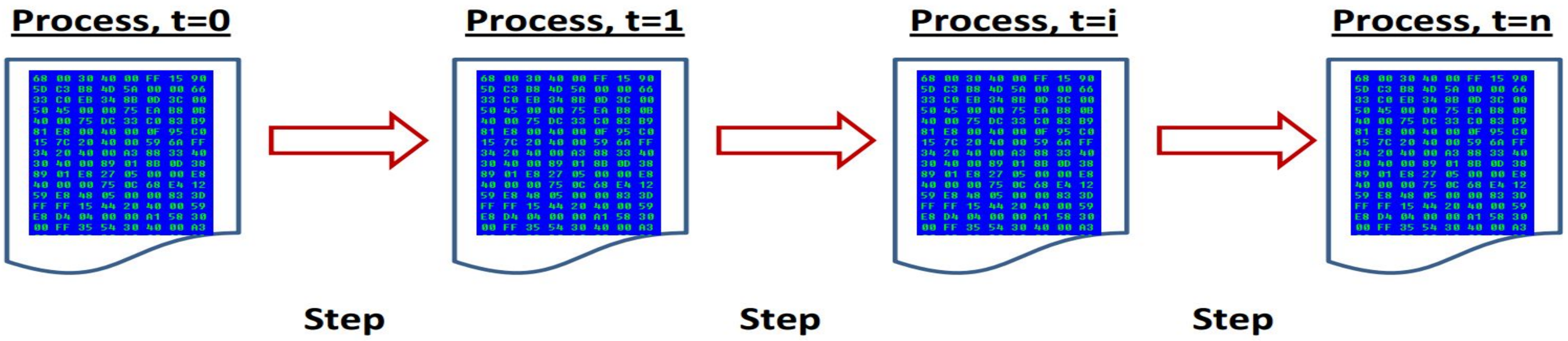
```
68 00 30 40 00 FF 15 90
5D C3 B8 4D 5A 00 00 66
33 C0 EB 34 8B 0D 3C 00
50 45 00 00 75 EA B8 0B
40 00 75 DC 33 C0 83 B9
81 E8 00 40 00 0F 95 C0
15 7C 20 40 00 59 6A FF
34 20 40 00 A3 88 33 40
30 40 00 89 01 8B 0D 38
89 01 E8 27 05 00 00 E8
40 00 00 75 0C 68 E4 12
59 E8 48 05 00 00 83 3D
FF FF 15 44 20 40 00 59
E8 D4 04 00 00 A1 58 30
00 FF 35 54 30 40 00 A3
```

```
68 00 30 40 00 FF 15 90
5D C3 B8 4D 5A 00 00 66
33 C0 EB 34 8B 0D 3C 00
50 45 00 00 75 EA B8 0B
40 00 75 DC 33 C0 83 B9
81 E8 00 40 00 0F 95 C0
15 7C 20 40 00 59 6A FF
34 20 40 00 A3 88 33 40
30 40 00 89 01 8B 0D 38
89 01 E8 27 05 00 00 E8
40 00 00 75 0C 68 E4 12
00 68 B4 20 40 00 68 A4
59 59 85 C0 74 17 C7 45
00 00 E9 DE 00 00 00 89
33 40 00 75 1B 68 A0 20
77 04 00 00 59 59 C7 05
```

```
68 00 30 40 00 FF 15 90
5D C3 B8 4D 5A 00 00 66
33 C0 EB 34 8B 0D 3C 00
50 45 00 00 75 EA B8 0B
40 00 75 DC 33 C0 83 B9
81 E8 00 40 00 0F 95 C0
15 7C 20 40 00 59 6A FF
34 20 40 00 A3 88 33 40
30 40 00 89 01 8B 0D 38
89 01 E8 27 05 00 00 E8
40 00 00 75 0C 68 E4 12
59 E8 48 05 00 00 83 3D
FF FF 15 44 20 40 00 59
E8 D4 04 00 00 A1 58 30
00 FF 35 54 30 40 00 A3
```

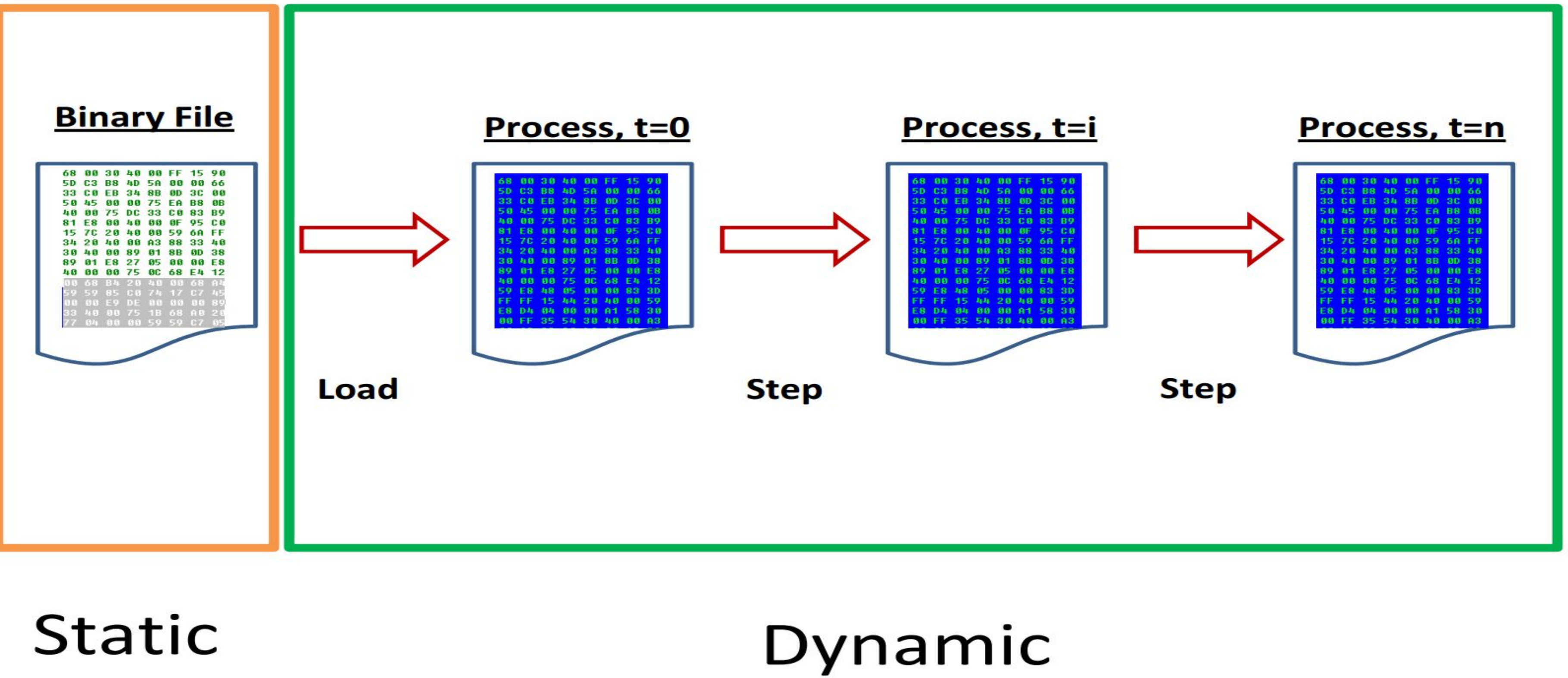
```
00 68 B4 20 40 00 68 A4
59 59 85 C0 74 17 C7 45
00 00 E9 DE 00 00 00 89
33 40 00 75 1B 68 A0 20
77 04 00 00 59 59 C7 05
```

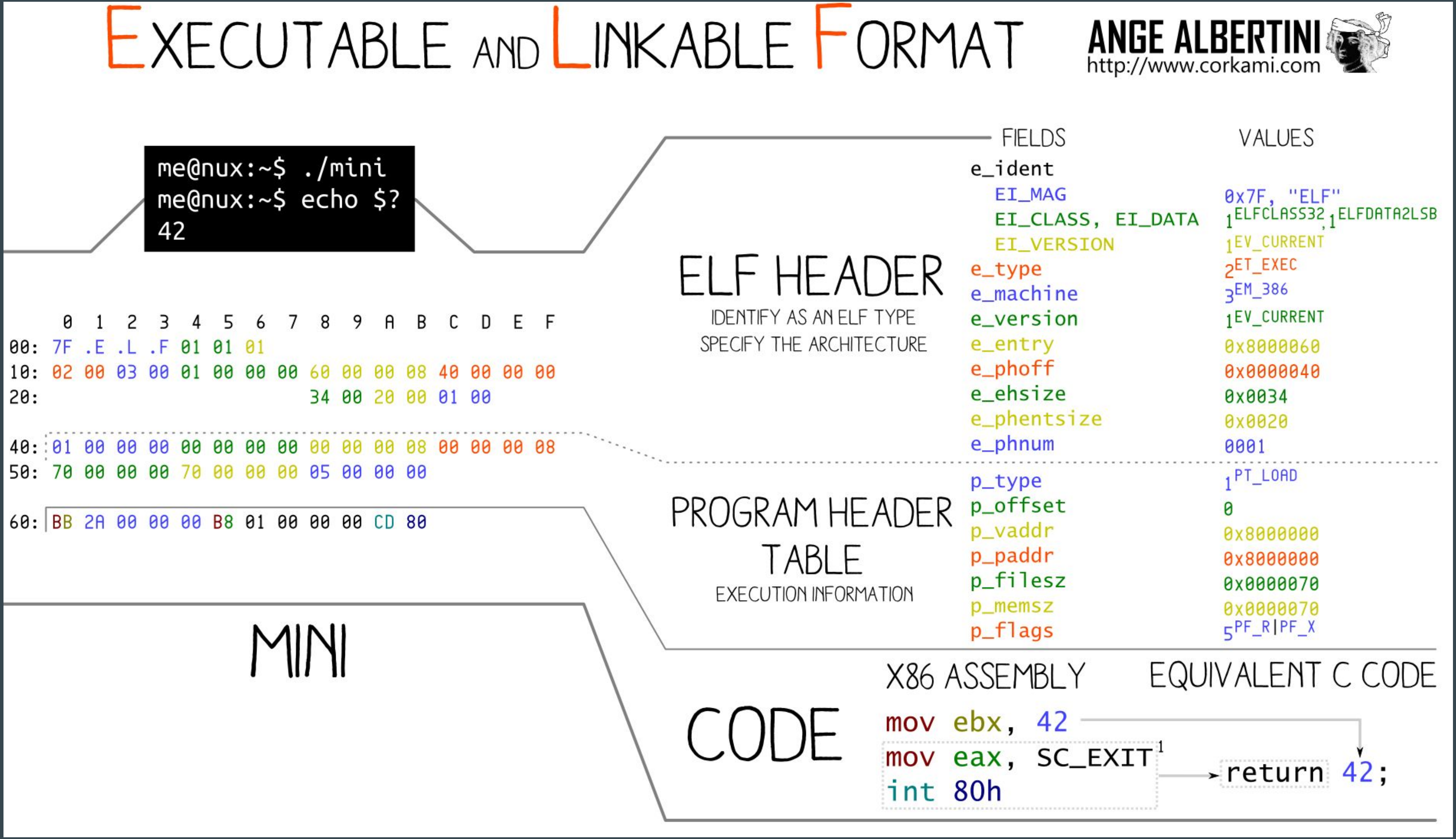
Running



```
68 00 30 40 00 FF 15 90
5D C3 B8 4D 5A 00 00 66
33 C0 EB 34 8B 0D 3C 00
50 45 00 00 75 EA B8 0B
40 00 75 DC 33 C0 83 B9
81 E8 00 40 00 0F 95 C0
15 7C 20 40 00 59 6A FF
34 20 40 00 A3 88 33 40
30 40 00 89 01 8B 0D 38
89 01 E8 27 05 00 00 E8
40 00 00 75 0C 68 E4 12
59 E8 48 05 00 00 83 3D
FF FF 15 44 20 40 00 59
E8 D4 04 00 00 A1 58 30
00 FF 35 54 30 40 00 A3
```

RE Domain


Binary File
Load
Process, t=0
Step
Process, t=i
Step
Process, t=n
Static
Dynamic



EXECUTABLE AND LINKABLE FORMAT

ANGE ALBERTINI
http://www.corkami.com

```
me@nux:~$ ./mini
me@nux:~$ echo $?
42
```

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00: 7F .E .L .F 01 01 01
10: 02 00 03 00 01 00 00 00 60 00 00 08 40 00 00 00
20:                         34 00 20 00 01 00

40: 01 00 00 00 00 00 00 00 00 00 00 08 00 00 00 08
50: 70 00 00 00 70 00 00 00 05 00 00 00

60: BB 2A 00 00 00 B8 01 00 00 00 CD 80
```

MINI

ELF HEADER

IDENTIFY AS AN ELF TYPE
SPECIFY THE ARCHITECTURE

FIELDS	VALUES
e_ident	
EI_MAG	0x7F, "ELF"
EI_CLASS, EI_DATA	1 ELFCLASS32, 1 ELFDATA2LSB
EI_VERSION	1 EV_CURRENT
e_type	2 ET_EXEC
e_machine	3 EM_386
e_version	1 EV_CURRENT
e_entry	0x8000060
e_phoff	0x0000040
e_ehsize	0x0034
e_phentsize	0x0020
e_phnum	0001

PROGRAM HEADER TABLE

EXECUTION INFORMATION

FIELDS	VALUES
p_type	1 PT_LOAD
p_offset	0
p_vaddr	0x8000000
p_paddr	0x8000000
p_filesz	0x0000070
p_memsz	0x0000070
p_flags	5 PF_R|PF_X

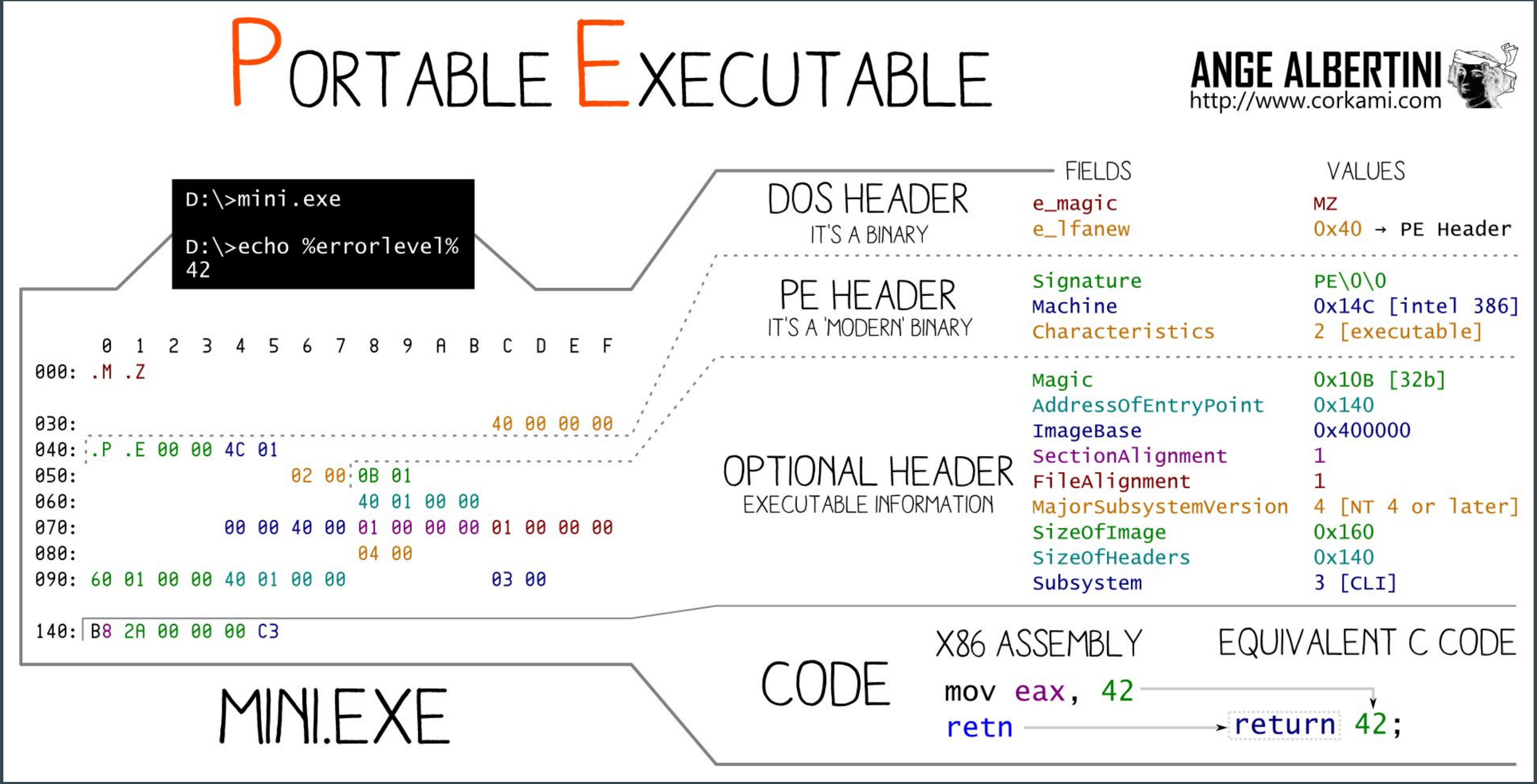
CODE

X86 ASSEMBLY

```
mov ebx, 42
mov eax, SC_EXIT¹
int 80h
```

EQUIVALENT C CODE

```
return 42;
```



Portable Executable

ANGE ALBERTINI
http://www.corkami.com

```
D:\>mini.exe

D:\>echo %errorlevel%
42
```

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000: .M .Z

030:                                     40 00 00 00
040: .P .E 00 00 4C 01
050:                   02 00 0B 01
060:                         40 01 00 00
070:             00 00 40 00 01 00 00 00 01 00 00 00
080:                         04 00
090: 60 01 00 00 40 01 00 00             03 00

140: B8 2A 00 00 00 C3
```

MINI.EXE

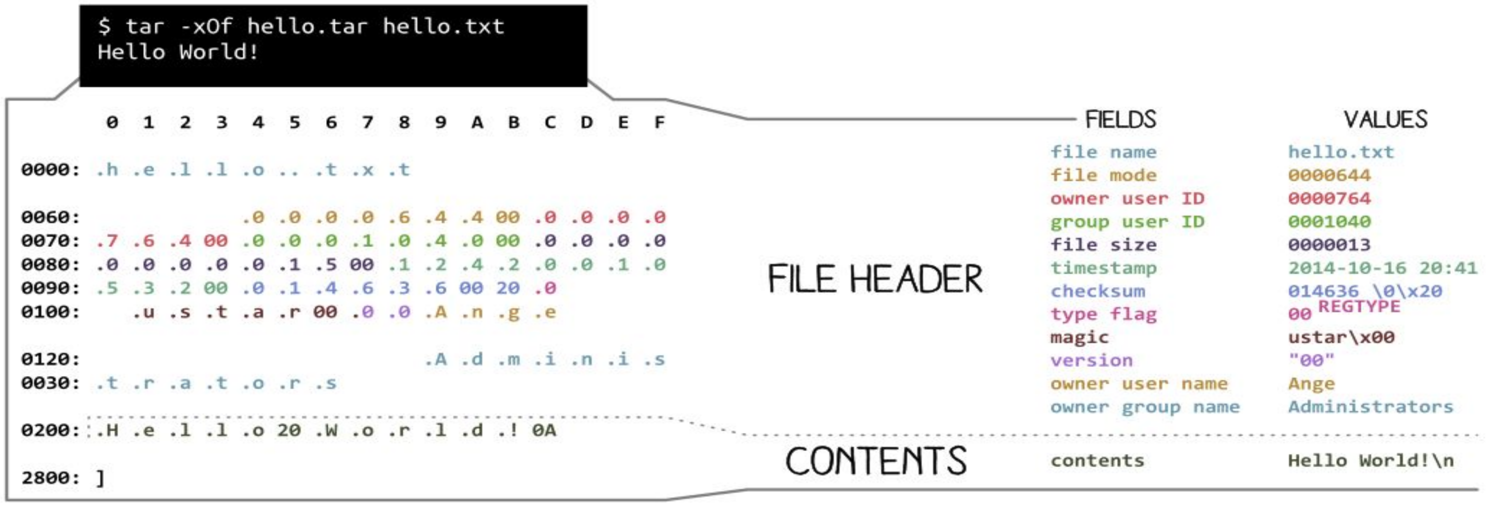
	FIELDS	VALUES
DOS HEADER IT'S A BINARY	e_magic	MZ
	e_lfanew	0x40 → PE Header
PE HEADER IT'S A 'MODERN' BINARY	Signature	PE\0\0
	Machine	0x14C [intel 386]
	Characteristics	2 [executable]
OPTIONAL HEADER EXECUTABLE INFORMATION	Magic	0x10B [32b]
	AddressOfEntryPoint	0x140
	ImageBase	0x400000
	SectionAlignment	1
	FileAlignment	1
	MajorSubsystemVersion	4 [NT 4 or later]
	SizeOfImage	0x160
	SizeOfHeaders	0x140
	Subsystem	3 [CLI]

CODE

X86 ASSEMBLY	EQUIVALENT C CODE
mov eax, 42 retn	return 42;

TAPE ARCHIVE

ANGE ALBERTINI
http://www.corkami.com



```
$ tar -xOf hello.tar hello.txt
Hello World!
```

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

0000: .h .e .l .l .o .. .t .x .t

0060:             .0 .0 .0 .0 .6 .4 .4 00 .0 .0 .0 .0
0070: .7 .6 .4 00 .0 .0 .0 .1 .0 .4 .0 00 .0 .0 .0 .0
0080: .0 .0 .0 .0 .0 .1 .5 00 .1 .2 .4 .2 .0 .0 .1 .0
0090: .5 .3 .2 00 .0 .1 .4 .6 .3 .6 00 20 .0
0100:    .u .s .t .a .r 00 .0 .0 .A .n .g .e

0120:                               .A .d .m .i .n .i .s
0030: .t .r .a .t .o .r .s

0200: .H .e .l .l .o 20 .W .o .r .l .d .! 0A

2800: ]
```

	FIELDS	VALUES
FILE HEADER	file name	hello.txt
	file mode	0000644
	owner user ID	0000764
	group user ID	0001040
	file size	0000013
	timestamp	2014-10-16 20:41
	checksum	014636 \0\x20
	type flag	00 REGTYPE
	magic	ustar\x00
	version	"00"
	owner user name	Ange
	owner group name	Administrators
CONTENTS	contents	Hello World!\n

TAR WAS INITIALLY DESIGNED FOR TAPE DRIVES, IN 1979:
- NO COMPRESSION, BLOCK ALIGNED
- NUMERIC VALUES ARE STORED IN OCTAL, ENCODED IN ASCII

TAR IS OFTEN COMBINED WITH GZIP, BZIP2 OR LZMA.
THE TAR FORMAT EVOLVED:
THIS EXAMPLE IS A "USTAR" FILE, AS DEFINED IN 1988

More on binary formats

- Mach-O (Mac OS X):
  - https://github.com/corkami/pics/blob/master/MachO.png
  - https://raw.githubusercontent.com/corkami/pics/master/Mach-O101.png
- Further PE and ELF in-depth:
  - https://raw.githubusercontent.com/corkami/pics/master/PE101.png
  - https://raw.githubusercontent.com/corkami/pics/master/ELF101.png
- Many non-executable binary formats
  - PDF
  - JPEG
  - etc...

Overview

1. Intro, resources, what we’ll learn, how we’ll have fun
2. The life of a binary and static reverse engineering
3. Abstract machines, and what they mean to security
4. Reversing and cracking demo!

Virtual Machines

- Many executables don't run on the metal, but instead on virtual metal
  - Java, Flash, C#, and more

"The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability. For example, the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine"

- Behaves almost exactly the same as a classic C/x86 executable
  - Load
  - Link
  - Execute

Virtual Machines (2)

- Like C, most VM based languages translate into a form of assembly specific to their virtual machine. Known as Bytecode.
- VM based languages also typically save a lot more information about the original source code. Useful for debugging and reversing.
    - Comments
    - Variable type

Java to Java bytecode

```
outer:
for (int i = 2; i < 1000; i++) {
   for (int j = 2; j < i; j++) {
      if (i % j == 0)
         continue outer;
   }
   System.out.println (i);
}
```

```
0:  iconst_2
1:  istore_1
2:  iload_1
3:  sipush  1000
6:  if_icmpge      44
9:  iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge      31
16:  iload_1
17:  iload_2
18:  irem
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic      #84; // Field
java/lang/System.out:
Ljava/io/PrintStream;
34:  iload_1
35:  invokevirtual   #85; //
Method java/io/PrintStream.
println:(I)V
38:  iinc    1, 1
41:  goto    2
44:  return
```

Info

- Mail list - buhacknight-list@bu.edu
  - How to join: http://www.bu.edu/tech/services/comm/email/mailing-lists/majordomo/commands/
- IRC - #buhacknight on irc.freenode.net
  - http://www.quassel-irc.org/
- Email
  - eugenek [at] [two letter name for this university].edu

Overview

1. Intro, resources, what we'll learn, how we'll have fun
2. The life of a binary and static reverse engineering
3. Abstract machines, and what they mean to security
4. Reversing and cracking demo!

Get inspired

- Live demos and explanations of cracking C, Flash, and C# applications.

- https://www.youtube.com/watch?v=V6hhlvbMFlY - Maximum CTF: Get the Most from Capture the Flag
- https://www.youtube.com/watch?v=hABj_mrP-no - DEFCON 19: Hacking MMORPGs for Fun and Mostly Profit
- https://www.youtube.com/watch?v=U4oB28ksiIo - DEFCON 18 Pwned By the owner What happens when you steal a hackers computer

Stay tuned next week for...

- Introduction to CTFs and Wargames
- Dynamic reverse engineering, debugging, and the stack
- Challenge1: ssh cmd2@pwnable.kr -p2222,
    - Requires flag from ssh cmd1@pwnable.kr -p2222 (pw: guest) first
- Challenge2: Remove pop-ups from Sublime Text

__libc_fini