Exploiting and Defense

Dobin Rutishauser

2016, 2017, 2018, 2019, 2020, 2021

Intro

About Me

Dobin Rutishauser

Worked as Security Analyst @ Compass Security for 8 years

- Penetration Tests
- Webapp Checks
- Architecture Reviews
- & lots more

Interested in ~~Hacking~~ Security since a young age (1998+)

I got a bit overboard when I was young

Content

Content

Exploiting & Defense

We will write **exploits** to **exploit buffer-overflows**

We will analyze what **defenses** exist to make writing exploits harder

Lecture

Lecture - Online

https://exploit.courses

- Online exploit development website
- Access to your own Linux via JavaScript terminal
- Uses Hacking-Lab accounts
- Solve **challenges** online
    - Write exploits
    - Debug them
- Slides

(https://www.hacking-lab.com)

- Half-online challenges website
- Uses HLCD (Kali-based Linux Distribution)
- VPN-Based
- Use this if you don't like exploit.courses

Lecture - Online

If you wanna try it by yourself:

https://github.com/dobin/yookiterm-challenges

- The writeup of the challenges

https://github.com/dobin/yookiterm-challenges-files

- Source code of challenges

Lecture - Online

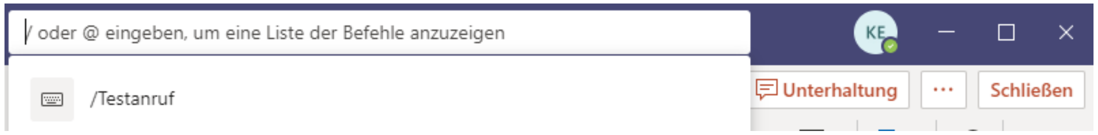
Important slides are marked with in top right corner

Sometimes slides have helpful comments in "notes" section

"Recap" slides at end of chapters point you to which things are important, and should be understood

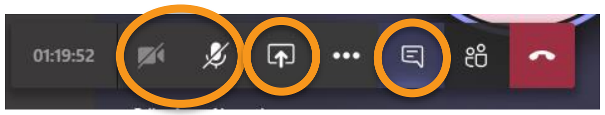
Welcome to the BFH-Teams Meeting

1. Test the audio before the appointment (enter "/test call" in the search bar above)


/ oder @ eingeben, um eine Liste der Befehle anzuzeigen

/Testanruf

2. During the meeting: switch off / mute microphone & camera
   (camera & microphone are crossed out)



3. Turn on the camera before speaking (=message), do not switch on the microphone until called by the teacher

4. Questions and more generally in the chat

Motivation

Motivation for Exploiting & Defense

Motivation

For the hacker:

- Developing exploits
- Debugging of C/C++ code
- Disassembly & reversing of assembler code
- Being 31337

For the Sysadmin

- Judge security level of operating systems, and applications
- Harden and protect servers, clients

For the CISO:

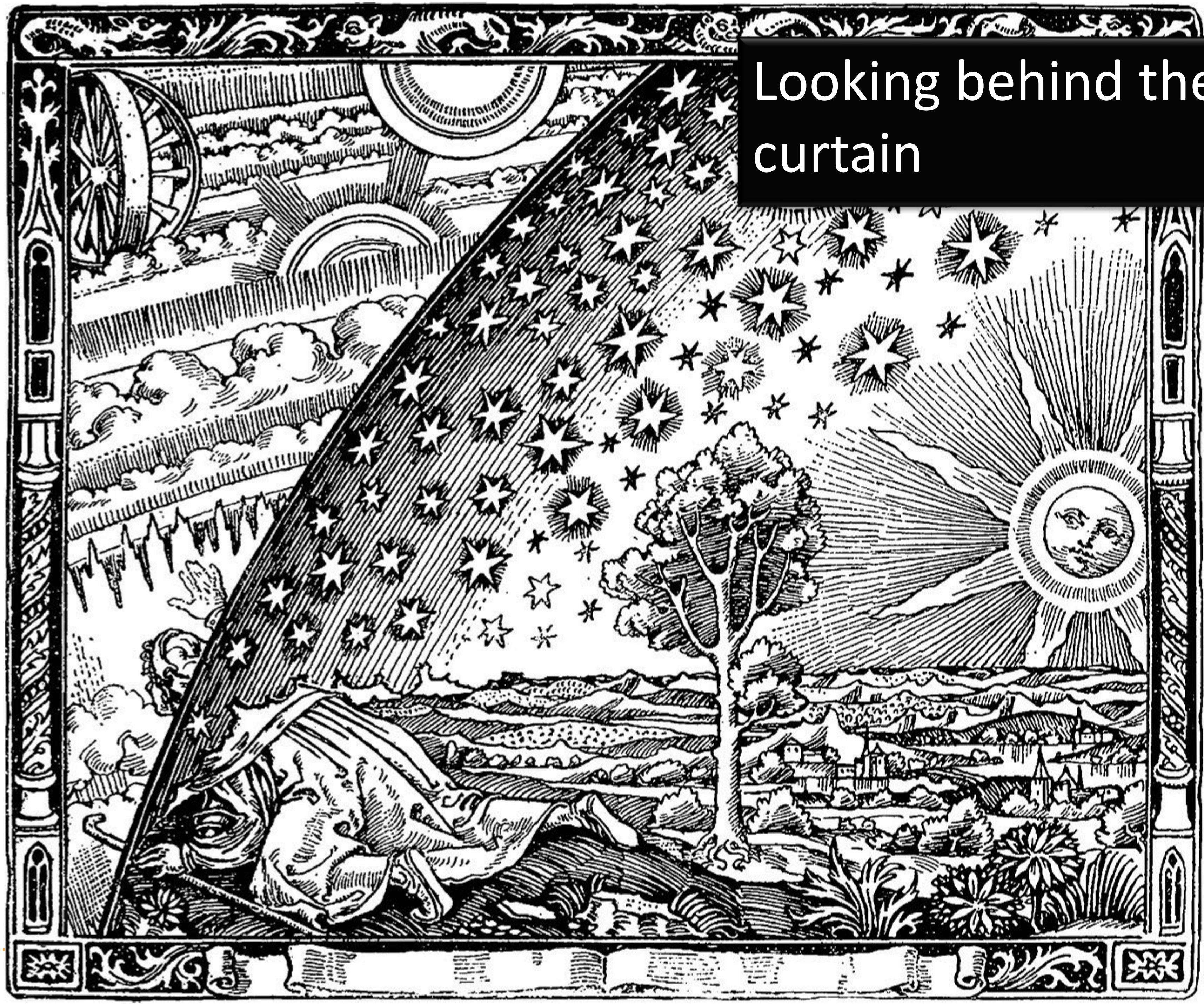
- Assess CVSS scores
- Assess (new) security mitigations
- Better risk analysis

Motivation

For everyone:

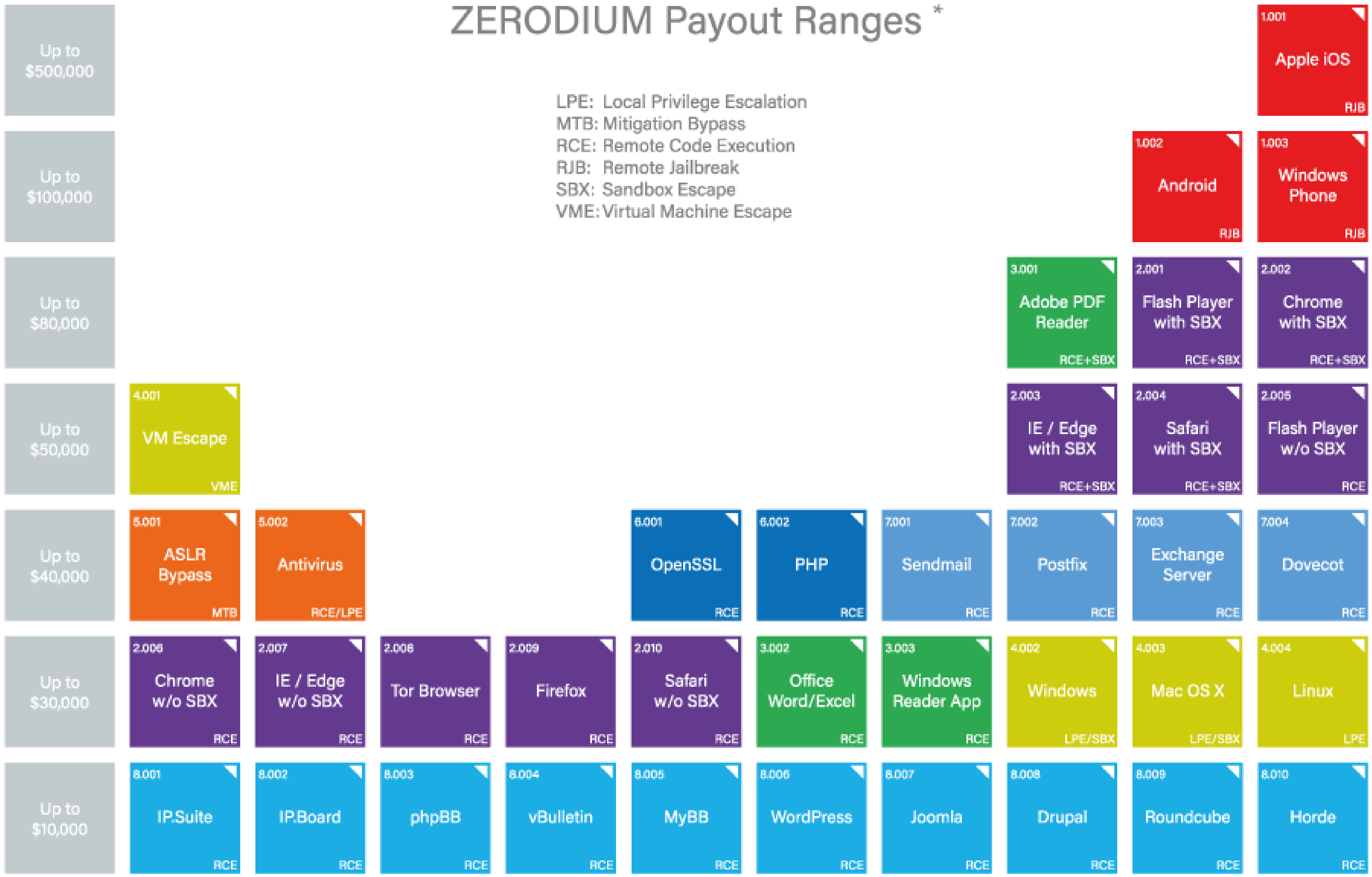
- How do functions work?
- How does the memory allocator work?
- What’s the difference between userspace and kernelspace?
- How does computer work?!

Looking behind the curtain



ZERODIUM Payout Ranges *

LPE:	Local Privilege Escalation
MTB:	Mitigation Bypass
RCE:	Remote Code Execution
RJB:	Remote Jailbreak
SBX:	Sandbox Escape
VME:	Virtual Machine Escape



Up to $500,000										1.001 Apple iOS RJB
Up to $100,000									1.002 Android RJB	1.003 Windows Phone RJB
Up to $80,000								3.001 Adobe PDF Reader RCE+SBX	2.001 Flash Player with SBX RCE+SBX	2.002 Chrome with SBX RCE+SBX
Up to $50,000	4.001 VM Escape VME							2.003 IE / Edge with SBX RCE+SBX	2.004 Safari with SBX RCE+SBX	2.005 Flash Player w/o SBX RCE
Up to $40,000	5.001 ASLR Bypass MTB	5.002 Antivirus RCE/LPE			6.001 OpenSSL RCE	6.002 PHP RCE	7.001 Sendmail RCE	7.002 Postfix RCE	7.003 Exchange Server RCE	7.004 Dovecot RCE
Up to $30,000	2.006 Chrome w/o SBX RCE	2.007 IE / Edge w/o SBX RCE	2.008 Tor Browser RCE	2.009 Firefox RCE	2.010 Safari w/o SBX RCE	3.002 Office Word/Excel RCE	3.003 Windows Reader App RCE	4.002 Windows LPE/SBX	4.003 Mac OS X LPE/SBX	4.004 Linux LPE
Up to $10,000	8.001 IP.Suite RCE	8.002 IP.Board RCE	8.003 phpBB RCE	8.004 vBulletin RCE	8.005 MyBB RCE	8.006 WordPress RCE	8.007 Joomla RCE	8.008 Drupal RCE	8.009 Roundcube RCE	8.010 Horde RCE

* *All payout amounts are chosen at the discretion of ZERODIUM and are subject to change or cancellation without notice.*

ZERODIUM Payouts for Mobiles*

RJB:	Remote Jailbreak with Persistence
RCE:	Remote Code Execution
LPE:	Local Privilege Escalation
SBX:	Sandbox Escape or Bypass

🟥	iOS
🟫	Android
🟦	Any OS

Up to $2,000,000									🟥 1.001 iPhone RJB Zero Click iOS
Up to $1,500,000									🟥 1.002 iPhone RJB iOS
Up to $1,000,000							🟦 2.001 WhatsApp RCE+LPE iOS/Android	🟦 2.002 SMS/MMS RCE+LPE iOS/Android	🟥 2.003 iMessage RCE+LPE iOS
Up to $500,000	🟦 2.004 WeChat RCE+LPE iOS/Android			🟦 2.005 FB Messenger RCE+LPE iOS/Android	🟦 2.006 Signal RCE+LPE iOS/Android	🟦 2.007 Telegram RCE+LPE iOS/Android	🟦 2.008 Email App RCE+LPE iOS/Android	🟫 3.001 Chrome RCE+LPE Android	🟥 3.002 Safari RCE+LPE iOS
Up to $200,000	🟦 4.001 Baseband RCE+LPE iOS/Android		🟦 5.001 LPE to Kernel/Root iOS/Android	🟦 2.009 Media Files RCE+LPE iOS/Android	🟦 2.010 Documents RCE+LPE iOS/Android	🟫 3.003 SBX for Chrome Android	🟫 3.004 Chrome RCE w/o SBX Android	🟥 3.005 SBX for Safari iOS	🟥 3.006 Safari RCE w/o SBX iOS
Up to $100,000	🟦 6.001 Code Signing Bypass iOS/Android	🟦 4.002 WiFi RCE iOS/Android	🟦 4.003 RCE via MitM iOS/Android	🟫 5.002 LPE to System Android	🟦 7.001 Information Disclosure iOS/Android	🟦 7.002 [k]ASLR Bypass iOS/Android	🟫 9.001 PIN Bypass Android	🟥 8.002 Passcode Bypass iOS	🟥 8.003 Touch ID Bypass iOS

* *All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.*

Content of the next 7 Friday afternoons

Content

You want to learn:

- What memory corruptions are
- What buffer overflows are
- What exploits are
- How exploits are being created
- To exploit a local application
- To exploit a remote application
- Learn about anti-exploiting technologies
- To circumvent all common anti-exploiting technologies for Linux
- See how Windows does it
- Use Use-After-Free
- Hack browsers
- ~~Hack facebook “for a friend”~~

Content

You will actually learn:

- Intel x86
  - Architecture
  - CPU
  - Registers
- Linux
  - Userspace memory layout, stacks, heap
  - Syscalls
  - Sockets
  - Networking
- Programming Languages
  - Assembler
  - C
  - Python
  - Bash

Plan

Plan

26.02.2021

Theory:

- 0x01 Intro (this)
- 0x02 Intro Technical
- 0x10 Intel Architecture
- 0x11 Memory Layout

Challenges:

- 0: Introduction to memory layout - basic
- 1: Introduction to memory layout - advanced

Plan

05.03.2021

Theory:

- 0x12 C Array and Data Structures
- 0x30 Assembler Intro
- 0x31 Shellcode
- 0x32 Function Call Convention
- 0x33 Debugging

Challenges:

- 2: C buffer analysis - simple
- 3: Introduction to shellcode development
- 7: Function Call Convention in x86 (32bit)
- 8: C buffer analysis - with debugging
- 9: Simple Buffer overflow - variable overwrite

Plan

12.03.2021

Theory:

- 0x41 Buffer Overflow
- 0x42 Exploit
- 0x44 Remote Exploit

Challenges:

- 11: Development of a buffer overflow exploit - 32 bit
- 12: Development of a buffer overflow exploit - 64 bit
- 13: Development of a remote buffer overflow exploit - 64 bit

Plan

19.03.2021

Theory:

- 0x51 Exploit Mitigation
- 0x52 Defeat Exploit Mitigation
- 0x53 Exploit Mitigation – PIE
- 0x54 Defeat Exploit Mitigation ROP

Challenges:

- 14: Stack canary brute force
- 15: Simple remote buffer overflow exploit - ASLR/DEP/64bit
- 16: Remote buffer overflow with ROP - DEP/64bit
- 17: Remote buffer overflow with ROP - DEP/ASLR/64bit

Plan

26.03.2021

Theory:

- 0x55: Defeat Exploit Mitigation – Heap Intro
- 0x56: Defeat Exploit Mitigation – Heap Attacks

Challenges:

- 31: Heap use-after-free analysis

Plan

09.04.2021

Theory:

- 0x60: Windows Exploiting
- 0x70: Secure Coding

Plan

16.04.2021

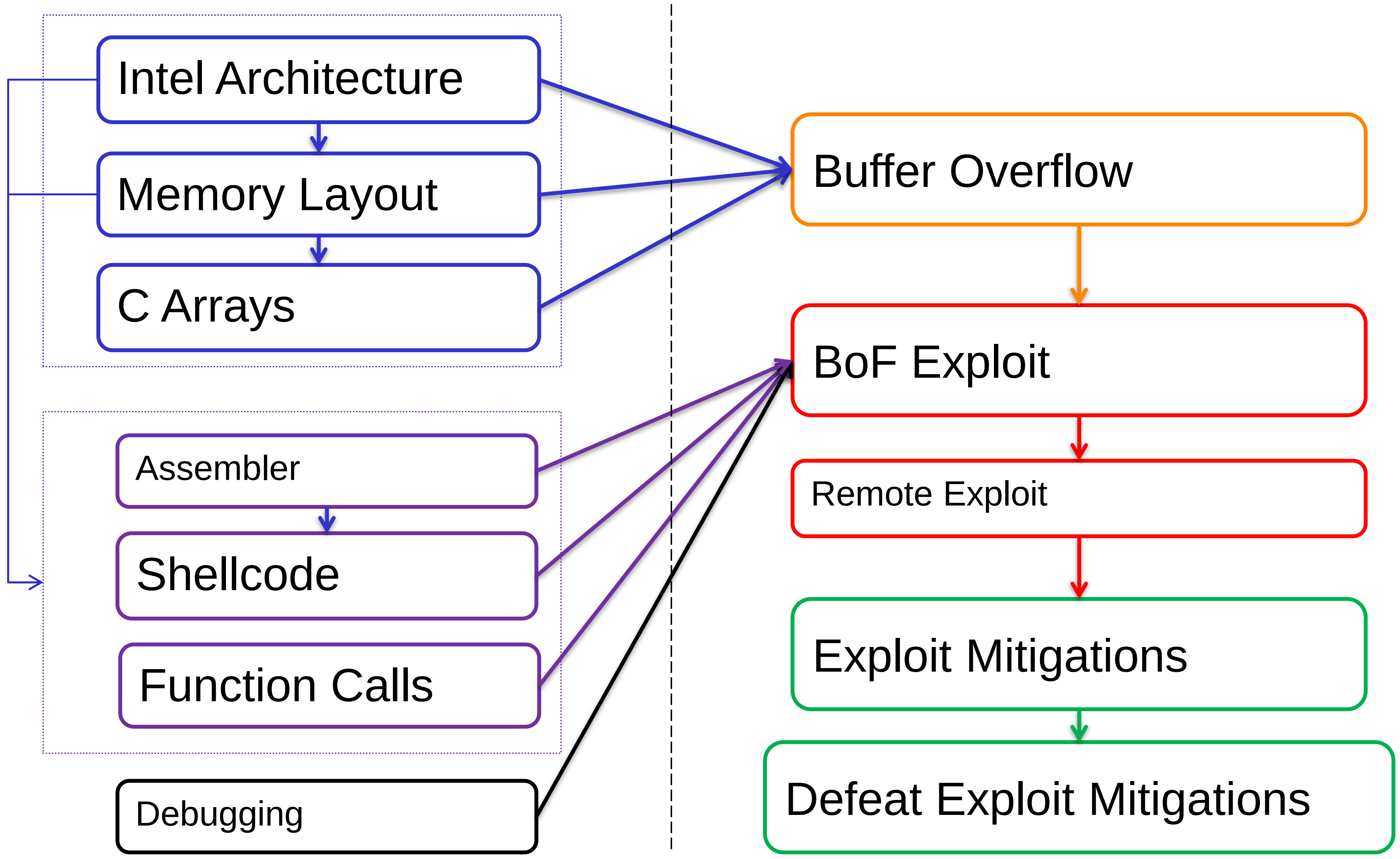
Theory:

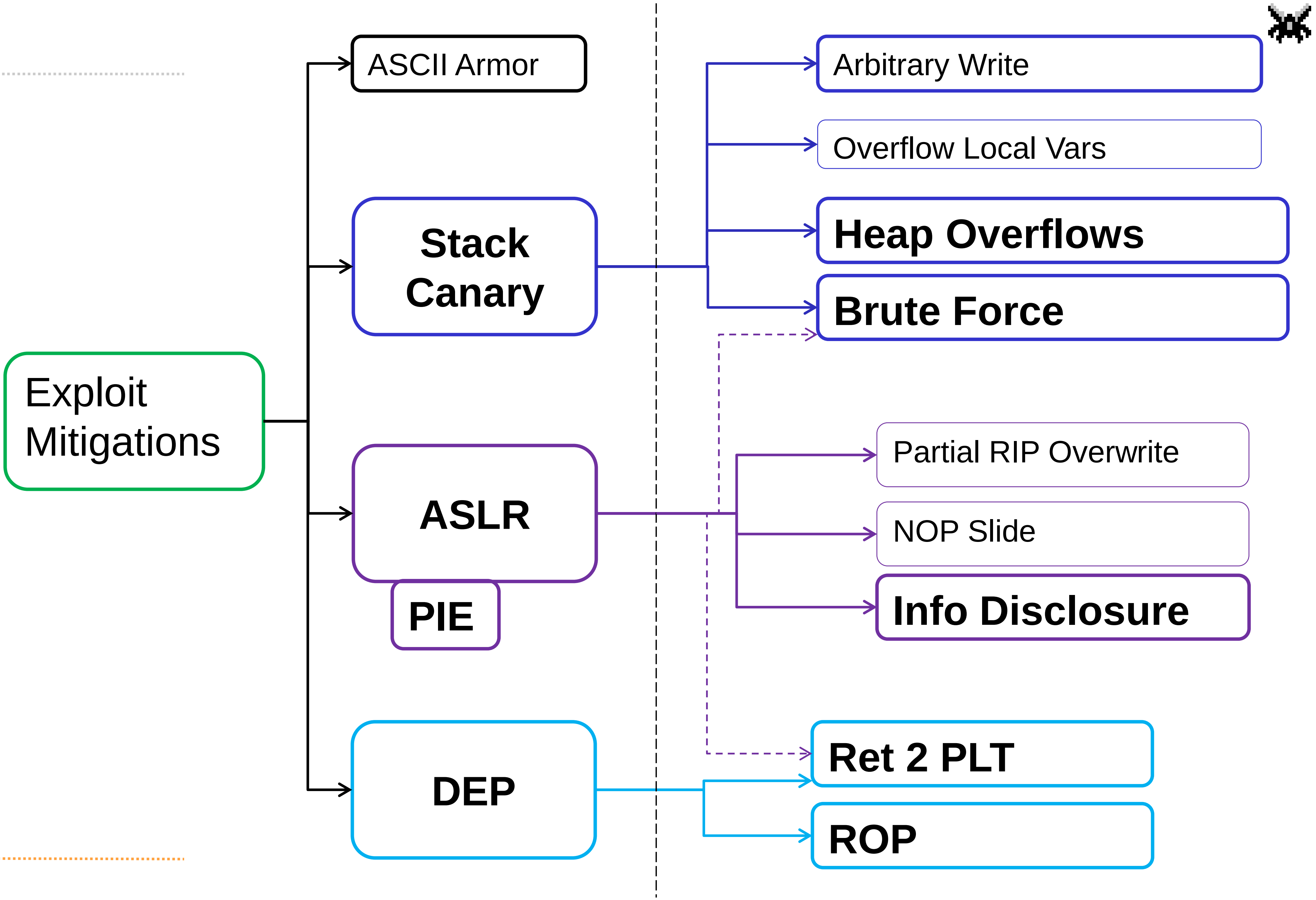
- 0x72: Linux Hardening
- 0x73: Kernel Exploitation

Challenges:

- 60: Linux Hardening

Content


Intel Architecture
Memory Layout
C Arrays
Assembler
Shellcode
Function Calls
Debugging
Buffer Overflow
BoF Exploit
Remote Exploit
Exploit Mitigations
Defeat Exploit Mitigations


Exploit Mitigations
ASCII Armor
Stack Canary
ASLR
PIE
DEP
Arbitrary Write
Overflow Local Vars
Heap Overflows
Brute Force
Partial RIP Overwrite
NOP Slide
Info Disclosure
Ret 2 PLT
ROP

And some…

- Windows Exploiting
- Fuzzing
- Browser Security
- Kernel Exploits
- Secure Coding
- Linux Hardening
- Case Studies

Oral Exam

What is (mainly) relevant for the oral exam?

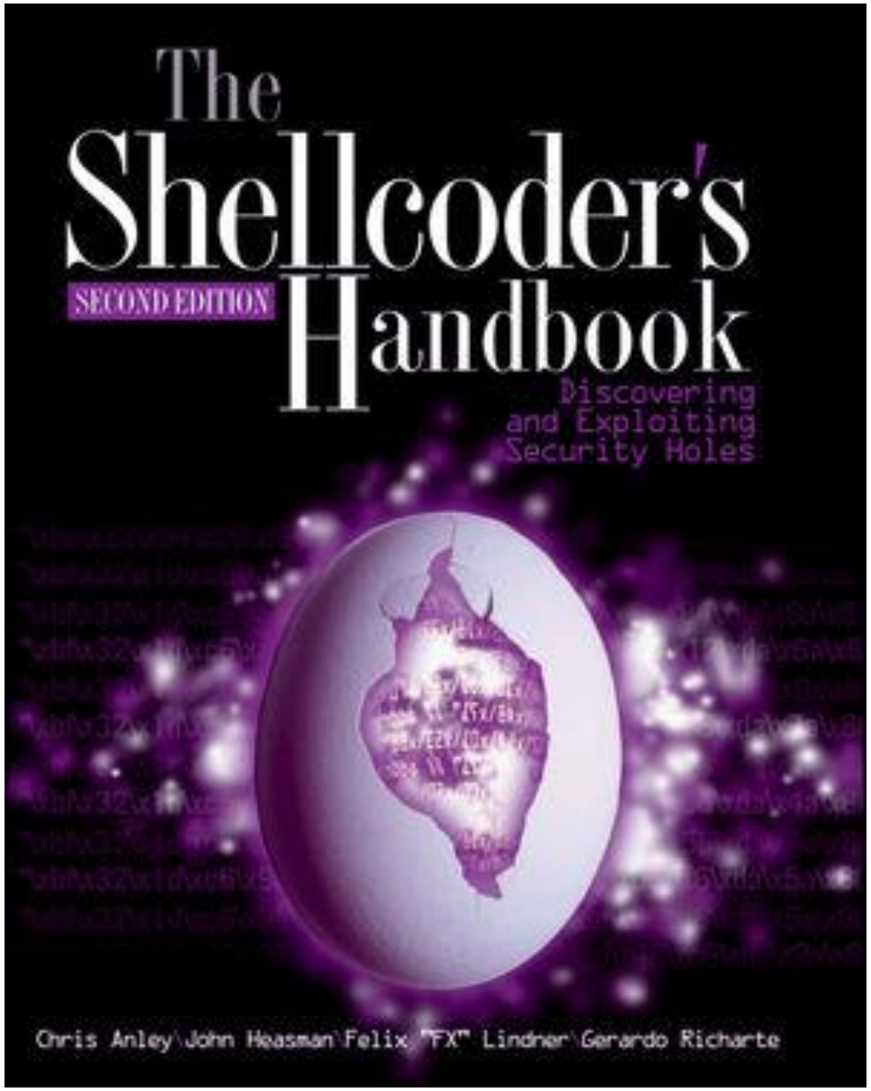
- How does memory corruption work?
- How does an exploit work?
- What exploit mitigations exist?
- How can these exploit mitigations be circumvented?

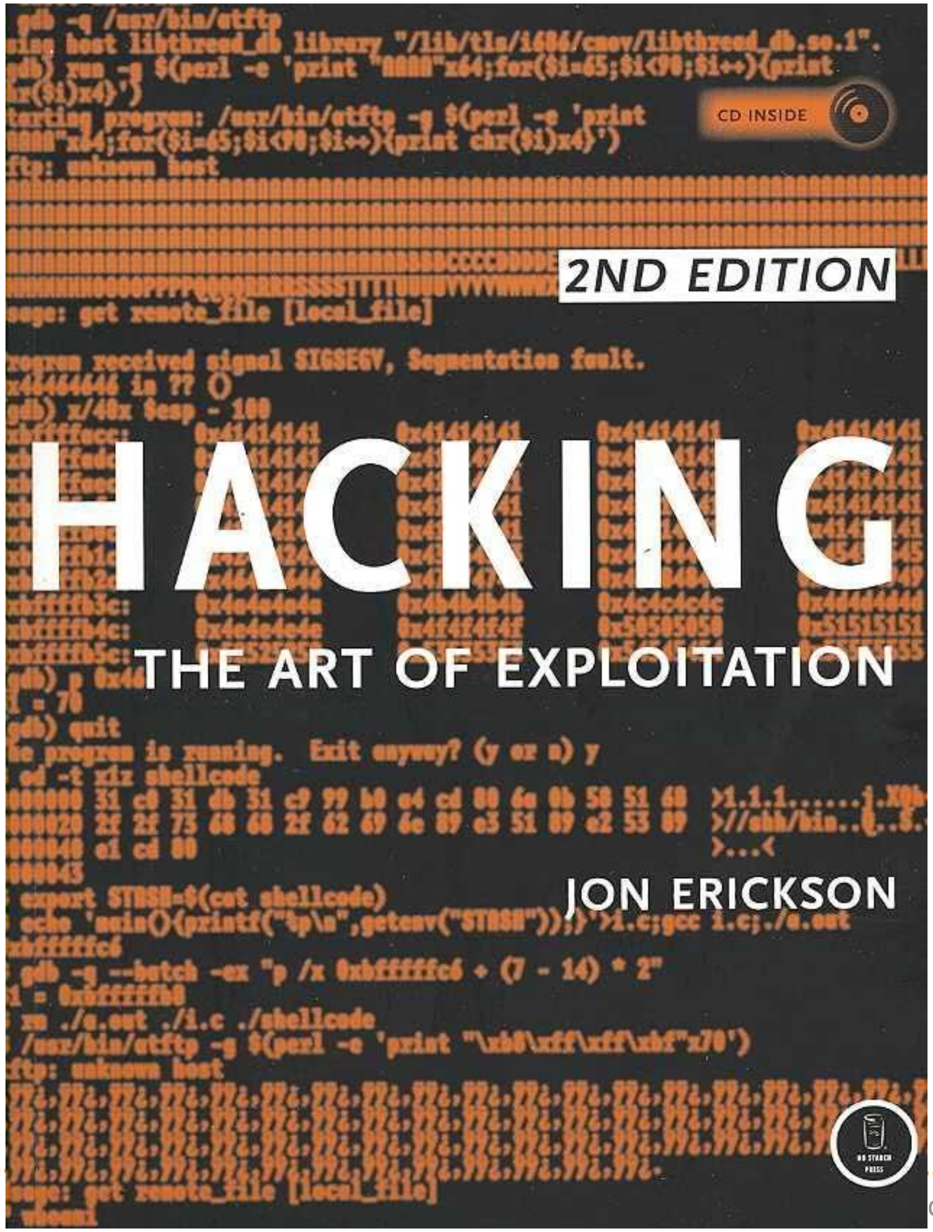
More theoretical, not so much the nitty gritty details

Typical question:

- Explain me how a buffer overflow exploit works
- Now we introduce ASLR. What do you need to change?

Books


The Shellcoder's Handbook
SECOND EDITION
Discovering and Exploiting Security Holes
Chris Anley\John Heasman\Felix "FX" Lindner\Gerardo Richarte


CD INSIDE
2ND EDITION
HACKING
THE ART OF EXPLOITATION
JON ERICKSON

Legal Issues

Legal CH

Don’t hack other people’s systems

«Damit der Tatbestand des **strafbaren Hackens** erfüllt ist, müssen **folgende Voraussetzungen kumulativ** erfüllt sein:

- **Eindringen** in das **Datenverarbeitungssystem**;
- **fremdes Datenverarbeitungssystem**;
- Eindringen auf dem Weg der von **Datenübertragungseinrichtungen;**
- **besondere Sicherung** gegen Zugriff.

https://www.lexwiki.ch/hacken/

Legal International

Wassenaar

- Arms Control Treaty
  - Anti-proliferation of Nukes and stuff
- Includes now (?):
  - Intrusion malware
  - Intrusion exploits
  - IP surveillance
- -> Exploits are now weapons…
  - Not allowed to transport over the border
  - Exception: If they are open source
  - (stop selling 0-days to Chinese gov!)

http://blog.erratasec.com/2015/05/some-notes-about-wassenaar.html