Lecture 1
Introduction. Basic Exploration Tools

Computer and Network Security
30th of September 2019
Computer Science and Engineering Department
Introduction

Things You Need to Know

Tools of the Trade (That You May or May Now Know)

Basic Tools for Exploration

Demo

Conclusion
Computer and Network Security
offensive security, hacking, reverse engineering, runtime application security
programming/practical oriented
focus on binary exploitation (pwn levels in CTFs)
lecture: Monday, 6pm-8pm, room PR002, Răzvan
labs:
  Monday, 4pm-6pm, room PR706, Ștefania
  Monday, 8pm-10pm, room PR706, Mihai
  Monday, 8pm-10pm, room EG106, Adrian
  Monday, 8pm-10pm, room EG306, Dennis
http://ocw.cs.pub.ro/cns/
but first:
https://ocw.cs.pub.ro/courses/cns/need-to-know
labs start on Monday, 30th of September 2019, 8pm
last instance of the first lab on Monday, 7th of October 2019, 4pm, room PR706
The Team

- Răzvan Deaconescu: lectures, lecture tests, exam
- Mihai Dumitru: labs, infrastructure
- Ștefania Popescu: labs
- Dennis Plosceanu: labs
- Adrian Șendroiu: labs, assignments, lectures
▶ wiki (content): http://ocw.cs.pub.ro/cns/
▶ Moodle (news, deadlines, exam, dicussions, links to content, feedback)
   ▶ SRIC (enrolable)
   ▶ SCPD (enrolable)
   ▶ common/meta (not-enrolable, actually used)
▶ Facebook (news, trivia): http://facebook.com/cns.upb
▶ mailing list (news, dicussions): https://ocw.cs.pub.ro/courses/cns/resources/mailing-list
▶ assignment write-only mailing-list (assignments): http://cursuri.cs.pub.ro/cgi-bin/mailman/listinfo/oss-support
▶ calendar & planning: https://ocw.cs.pub.ro/courses/cns/calendar
▶ virtual machines (labs, assignments, CTFs): https://ocw.cs.pub.ro/courses/cns/resources/vm
▶ CTF platform (assignments, labs): https://cns-ctf.security.cs.pub.ro/home
▶ lab rooms: PR706, EG106, EG306
▶ team: to yell at
Lab Split

- happens on the acs.curs.pub.ro Discussion forum
- threads for each of the 4 lab slots
- almost complete
- you need to be enrolled
- you can enrol by yourself by accessing the CNS acs.curs.pub.ro instance
- limit is 16 students per lab slot
Class Keywords

- reverse engineering
- binary inspection
- stack overflow
- buffer overflow
- shellcode

- shell execution
- exploiting
- runtime application security
- return oriented programming
- CTF (Capture the Flag)
planning: https://ocw.cs.pub.ro/courses/cns/calendar

1. Introduction. Basic Exploration Tools
2. Program Analysis
3. The Stack. Buffer Management
4. Exploiting. Shellcodes
5. Exploiting. Shellcodes (part 2)
6. Exploit Protection Mechanisms
7. Strings. Information Leaks
8. Return Oriented Programming
9. Return Oriented Programming (part 2)
10. Use After Free
11. Practical Attacks (part 1)
12. Practical Attacks (part 2)
13. Guest Lecture
Bibliography

- Robert Seacord – Secure Coding in C and C++, Addison Wesley Professional, 2005
- Anton Chuvakin, Cyrus Peikari – Security Warrior, O’Reilly, 2004
- Enrico Perla, Massimiliano Oldani – A Guide to Kernel Exploitation, Syngress, 2011
- Bruce Schneier – Applied Cryptography, John Wiley & Sons, 1996
Grading

- **2 points** – lab involvement
- **4.5 points** – 3 assignments
- **2 points** – lecture tests
- **2.5 points** – final exam
Tests during Lectures

- at the beginning of lectures 3, 5, 7, 9, 11, from the past two lectures
- start at 6:05pm; please don’t be late
- 10 minutes, 4 short questions
- 0.4 points each
Final Exam

- one part is a multiple answer questions test (22 questions, 20 minutes)
- the other part is an on paper test (60 minutes)
computer security competition
educational, practice
attack/defense vs. jeopardy
web, stegano/forensics, crypto, binary/reverse, pwn/exploit, protocol, misc
wargames
may equate assignment points
Introduction

Things You Need to Know

Tools of the Trade (That You May or May Not Know)

Basic Tools for Exploration

Demo

Conclusion
C Programming Language

- lingua franca of low-level programming
- powerful enough to build amazing software and flexible enough to shoot yourself in the foot
- close to hardware, everything is at some point coming from C code
- direct access to memory management (buffers, strings, arrays, pointers): mixed blessing
move around quickly
investigate, analyze system
quickly develop, build, debug, analyze applications
automate tasks
Assembly Language

- everything turns to machine code
- one may not have access to the source code, but it can be disassembled
- hardware specific – the “guts” of the computer
- required knowledge to fully be able to exploit and protect the system
- Binary, octal, hexadecimal
- ASCII
- Signed / unsigned integers: size, range, 2's complement representation
- Endianness
- There are 10 types of people in the world...
- Disassembled code, addresses and hardware instructions are shown in hexadecimal
- One is required to easily convert hexadecimal to decimal and the other way around
- system and application inner workings
- process virtual address space
- application run time: CPU, memory, I/O usage
- system calls, kernel space
Process Investigation

- processes and resource usage: ps, pstree, pgrep, procfs
- filesystem
- memory mappings: pmap
- open file descriptors: lsof
this is a master class, you need to be on the level
work, work, work
C programming:
https://ocw.cs.pub.ro/courses/programare
Linux / Unix CLI, shell scripting:
https://ocw.cs.pub.ro/courses/uso
assembly language + hexadecimal:
https://ocw.cs.pub.ro/courses/iocla
operating systems + process investigation:
https://ocw.cs.pub.ro/courses/so
Introduction

Things You Need to Know

Tools of the Trade (That You May or May Not Know)

Basic Tools for Exploration

Demo

Conclusion
Scripting Languages

- Python, Perl
- automation
- generate/print binary data and feed it to an executable
- generate strings, generate varying integers & addresses
- do redirects, make conversions, process strings
Python

- quick’n’dirty scripting language
- more powerful than shell scripting
- create binary payloads (use struct package)
- convert data
- work with strings
- work with files
- work with processes (use subprocess package)
- advanced exploit techniques (use pwn package)
- use Python3, FTW!!!
▶ dump and edit data in binary files (object files, executables, encrypted files)
▶ `hexdump`, `xxd`, `od`: make hexdumps
▶ `hte`: terminal hex editor
▶ `ghex`, `Bless`: GUI hex editor
▶ dynamic analysis
▶ default debugger on Unix systems
▶ may be used to trace programs, check variables and return values
Python Exploit Development Assistance for GDB

- enhance GDB for exploit development
- improved commands
- improved views
- search for ROP gadgets
- generate shellcodes
- generate buffer cyclic patterns
- http://ropshell.com/peda/
Binary Code Analysis

- inspect object and executable files
- disassembling: **objdump**
- forensics: **strings**
- executable parsing: **readelf, nm**
- dependencies: **ldd**
Call Tracing

- dynamic analysis
- capture system calls, function calls of program
- check out system call arguments
- check out system call return values
- see whether process blocks in a system call
- `strace`, `ltrace`
Advanced Disassemblers

- **IDA**
  - IDA 7.0 freeware
  - different executable formats for different processors
  - debugger
  - decompiler
  - interactive
  - plugins

- **Ghidra**
  - open source
  - similar to IDA

- **radare2**
  - disassemble, debug
  - static and dynamic analysis
  - CLI

- **capstone**
  - “lightweight multi-platform, multi-architecture disassembly framework”
  - open source
Other Binary-related Tools

- **Binary Ninja**: https://binary.ninja
- **BinNavi**: http://www.zynamics.com/binnavi.html
- **Hopper**: http://www.hopperapp.com/
Emulators

- run executables for different architectures
- QEMU: emulates MIPS, ARM, PowerPC, SPARC
- Unicorn Engine, based on QEMU
- CTF framework and exploit development library
- Python
- Connections to local and remote processes
- Packing / unpacking
- Assembly and disassembly
- ELF manipulation
- Shellcode generation
- Return Oriented Programming

https://github.com/Gallopsled/pwntools
- brain
- will
- perseverance
- will
- perseverance
- perseverance
- Did we mention perseverance?
Introduction

Things You Need to Know

Tools of the Trade (That You May or May Not Know)

Basic Tools for Exploration

Demo

Conclusion
- search for ASCII strings in binary data
- `strings /path/to/binary/file`
- `man ascii` to show ASCII table
Let's print shellcode from http://www.shell-storm.org/shellcode/files/shellcode-827.php:

Shellcode Sample

```c
char *shellcode = "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69" "\x6e\x89\xe3\x50\x53\x89\xe1\xb0\x0b\xcd\x80";
```

Do it in several scripting languages:

Print Shellcode in Bash, Python, Perl

(Bash) `echo -e '\x31\xc0\x50\x68\x2f...'`
(Python) `python -c 'print "\x31\xc0\x50\x68\x2f..."'`
(Perl) `perl -e 'print "\x31\xc0\x50\x68\x2f..."'`
Dump binary data in hex and binary:

### Using `xxd`

```bash
$ echo -en '\x31\xc0\x50\x68...' | xxd
0000000: 31c0 5068 2f2f 7368 682f 6269 6e89 e3 1.Ph//shh/bin..P
0000010: e1b0 0bcd 80 S......

$ echo -en '\x31\xc0\x50\x68...' | xxd -g 4
0000000: 31c05068 2f2f7368 682f6269 6e89e350 1.Ph//shh/bin..P
0000010: 5389e1b0 0bcd80 S......

$ echo -en '\x31\xc0\x50\x68...' | xxd -g 1
0000000: 31 c0 50 68 2f 2f 73 68 68 2f 62 69 6e 89 e3 50 1.Ph//shh/bin..P
0000010: 53 89 e1 b0 0b cd 80 S......

$ echo -en '\x31\xc0\x50\x68...' | xxd -b
0000000: 00110001 11000000 01010000 01101000 00101111 00101111 1.P.....
0000006: 01110011 01101000 01101000 00101111 01100010 01101001 shh/bi
000000c: 01101110 10001001 11100011 01010000 01010111 10001001 n..PS.
0000012: 11100001 10110000 00001011 11001101 10000000 ......
Using strace

- `strace ./executable`
- `strace -e write ./executable` – print write syscalls
- `strace -e trace=file ./executable` – print syscall taking a filename as argument
- `strace -f ./executable` – trace child processes
- `strace -p PID` – trace existing process by PID
- `strace -s strsize` – trace using a different size for strings
▶ see library calls
▶ `ltrace -p PID` — trace process
▶ `ltrace -t` — show timestamp
command $(python -c 'print ...')$
Python -c 'print ...' | command

cat file - | command

cat <(python -c 'print ...') - | command
List Open Files

- `lsof`
- `lsof -p PID` – show open files for process
- shows file descriptors: standard input/output, sockets, pipes
Process Address Space

- **pmap**
- **pmap PID** – show address space mappings for process
- shows permissions and addresses
Show Library Dependencies

- `ldd /path/to/executable`
- Useful to check if an executable may run on a given system, what library version is it using
Installing 32bit Development Libraries

# dpkg --add-architecture i386
# apt update
# apt install gcc-multilib g++-multilib libc6:i386 libc6-dev:i386 \
  libstdc++6:i386 libstdc++6-dev:i386
Introduction

Things You Need to Know

Tools of the Trade (That You May or May Not Know)

Basic Tools for Exploration

Demo

Conclusion
use objdump to disassemble binary

use man ascii or hex printing to print password
Outline

Introduction

Things You Need to Know

Tools of the Trade (That You May or May Not Know)

Basic Tools for Exploration

Demo

Conclusion
Keywords

- offensive security
- runtime application security
- table of contents
- grading
- CTF (*Capture the Flag*)
- tools of the trade
- hex editors
- scripting language
- disassemblers
- exploration

- hex/binary data
- Python
- strings
- objdump
- strace, ltrace
- ldd, lsof, pmap
- IDA
- Ghidra, radare2
- GDB, PEDAS
- pwntools
Useful Links

▶ http://reverseengineering.stackexchange.com/
▶ http://security.cs.pub.ro/hexcellents/wiki/
▶ http://web.cecs.pdx.edu/~jrb/cs201/lectures/handouts/gdbcomm.txt
▶ http://ctftime.org/
▶ https://picoctf.com/
▶ http://captf.com/practice-ctf/
▶ https://io.netgarage.org/
▶ http://www.overthewire.org/wargames/
▶ http://ctf365.com/
▶ PEDA: https://github.com/longld/peda
▶ IDA: https://www.hex-rays.com/products/ida/
▶ Ghidra: https://ghidra-sre.org
▶ Radare: http://rada.re/r/
References

- **Security Warrior**
  - Chapter 1. Assembly Language
  - Chapter 2. Windows Reverse Engineering
  - Chapter 3. Linux Reverse Engineering

- **The Ethical Hacker’s Handbook, 3rd Edition**
  - Chapter 10: Programming Survival Skills
  - Chapter 20: Passive Analysis
  - Chapter 21: Advanced Static Analysis with IDA Pro