

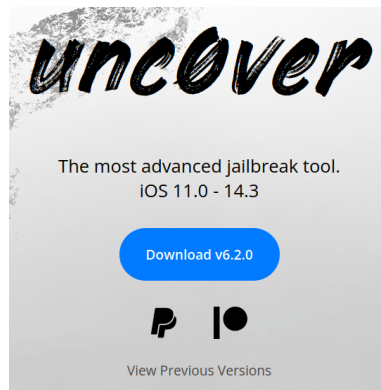
Session 01

Introduction to Systems Security

Security of Information Systems (SIS)

Computer Science and Engineering Department

October 4, 2023



<https://unc0ver.dev/>

History of OWASP Top 10

<https://www.hahwul.com/cullinan/history-of-owasp-top-10/>
2021, 2017, 2013, 2010, 2007, 2003

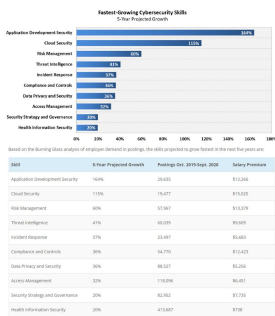
The more things change, the more they stay the same.

- ▶ Injection (3, 1, 1, 1, 2, 6)
- ▶ XSS (-, 7, 3, 2, 1, 4)
- ▶ Broken Authentication and Session Management (7, 2, 2, 3, 7, 3)
- ▶ Security Misconfiguration (5, 6, 5, 6, -, -)

Cybersecurity Jobs

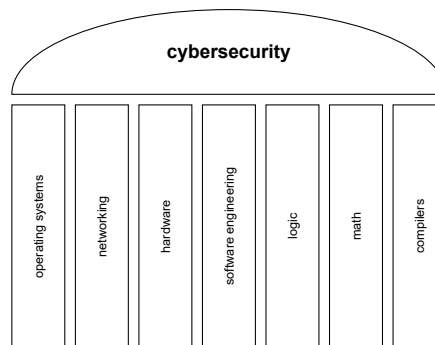
- ▶ <https://cybersecurityventures.com/jobs/>
- ▶ 350 percent growth in open cybersecurity positions from 2013 to 2021
- ▶ 3.5 million unfilled cybersecurity jobs globally by 2021, up from one million positions in 2014

Cybersecurity Skills



<https://www.forbes.com/sites/louiscolumnus/2020/11/01/what-are-the-fastest-growing-cybersecurity-skills-in-2021/>

System / Application Cybersecurity



Papers

- ▶ Reflections on Trusting Trust:
<https://dl.acm.org/doi/10.1145/358198.358210>
- ▶ Understanding the Mirai Botnet: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>

Systems Security

- ▶ IT security, cyber security, computer security
- ▶ applications, operating system, infrastructure, networking
- ▶ hardware, software, information

- ▶ focus on computer, operating system, software stack and infrastructure security
- ▶ little focus on networking, crypto, hardware
- ▶ research focus

- ▶ authentication, authorization
- ▶ attack, mitigation, bypass
- ▶ isolation, sandboxing
- ▶ fuzzing, symbolic execution
- ▶ defensive programming
- ▶ software analysis, verification and validation

10 / 62

11 / 62

Systems Security Research

Systems Security Research Venues

- ▶ research: more on finding the right question, than the right answer
- ▶ not boring, but uncertain
- ▶ high risk, high impact
- ▶ secure build, attack, defend, repeat
- ▶ actual systems: software, hardware, infrastructure

- ▶ big four: IEEE S&P (Oakland), ACM CCS, NDSS, Usenix Security Symposium
- ▶ ESORICS, RAID, ACSAC
- ▶ AsiaCCS, CNS, ACNS
- ▶ http://faculty.cs.tamu.edu/guofei/sec_conf_stat.htm
- ▶ http://s3.eurecom.fr/~balzarot/notes/top4_v1/

12 / 62

13 / 62

Research Results

Resources

- ▶ white papers: company product information
- ▶ workshops: preliminary work, networking
- ▶ conferences: complete work, novel results, networking
- ▶ journals: aggregated work, archive
- ▶ CVEs
- ▶ features in existing products
- ▶ start ups
- ▶ public / open source

- ▶ content website: <http://elf.cs.pub.ro/sis/>
- ▶ discussions, feedback: <https://curs.upb.ro/>
- ▶ MS Teams
- ▶ CTF platform: <https://sis-ctf.security.cs.pub.ro/>
- ▶ VM: <https://ocw.cs.pub.ro/courses/cns/resources/vm>

14 / 62

16 / 62

Logistics

Team

- ▶ lecture: Wednesday, 4pm-6pm, PR706
- ▶ labs
 1. group 1: Wednesday, 6pm-8pm, EG403
 2. group 2: Tuesday, 8pm-10pm, PR706
- ▶ first lecture takes place on Wednesday, October 4, 2023, 4pm-6pm
- ▶ first lab (group 1) takes place on Wednesday, October 4, 2023, 6pm-8pm
- ▶ first lab (group 2) takes place on Tuesday, October 10, 2023, 8pm-10pm

- ▶ Rareș Visalom
- ▶ Răzvan Deaconescu

17 / 62

18 / 62

Typical Lecture Session

- ▶ lecture + discussions
- ▶ research insights, trends
- ▶ demos, highlights
- ▶ DO ask questions, DO take part in discussion, DO challenge ideas, DO rebut ideas, DO open interesting topics (covering systems security)

19 / 62

Typical Lab Session

- ▶ lab/practical focused on problem solving, on approaches, not on tools
- ▶ CTF-like activity in lab: have problem, find solution
- ▶ challenge focused, not tutorial focused
- ▶ need focus, patience and perseverance
- ▶ we expect you to know the basics of: Linux/command line, network protocols and infrastructure, programming, assembly language, operating systems

20 / 62

CNS vs. SIS

- ▶ focus on offensive/exploiting vs focused on a building, attacking, defending
- ▶ narrow, focused, practical vs. diverse, holistic, conceptual
- ▶ binary/runtime application security vs. systems security
- ▶ engineer-centric vs. research-centric
- ▶ well-known steps vs. trends and models
- ▶ skill vs. approach

21 / 62

Grading

- ▶ 2p: lab activity
- ▶ 4p: assignments/projects (bonuses included)
- ▶ 4p: final exam (oral exam)
- ▶ lecture activity as a bonus (DO take part in lectures, see live and offline quizzes and surveys)

22 / 62

Contents

1. Introduction to Systems Security
2. Authentication and Password Management
3. Exploiting. Part 1: Applications
4. Exploiting. Part 2: Web & OS
5. Defense and Mitiation
6. Modern Offensive and Defensive Solutions
7. Application Confinement
8. System Isolation
9. Code Analysis
10. Fuzzing. Symbolic Execution
11. Information Flow Security
12. Software Verification

23 / 62

Storyline

- ▶ we have a system/infrastructure
- ▶ one way to attack it is to crack or abuse authentication
- ▶ assuming valid entry points (i.e. authentication) are secure, the attacker looks for security holes (i.e. vulnerabilities)
- ▶ the attacker exploits vulnerabilities, we use defensive mechanisms, the attacker tries to bypass them
- ▶ we assume attacks are unavoidable, we aim to confine them; we confine applications
- ▶ since applications interact with other applications and system components, we isolate the system
- ▶ proper isolation relies on separation and marking roles and validating the information flow

24 / 62

Storyline (cont.)

- ▶ ideally, our application or system will be secure by specification, design, implementation, validated offline
- ▶ we use static analysis and dynamic analysis (including fuzzing and symbolic execution) to validate an application offline
- ▶ a thorough approach is to use (semi)formal validation on application specifications and build it securely from those

25 / 62

Prerequisites

- ▶ good understanding of operating systems
- ▶ good Unix/Linux command line abilities
- ▶ good C programming skills
- ▶ fair knowledge of C shell scripting
- ▶ fair knowledge of Unix/Linux development tools
- ▶ basic understanding networking
- ▶ basic knowledge of computer architecture and assembly language

26 / 62

- ▶ Linux/command line: USO labs:
<http://ocw.cs.pub.ro/courses/uso>
- ▶ networking: RL labs: <http://ocw.cs.pub.ro/courses/rl>
- ▶ assembly: <http://ocw.cs.pub.ro/courses/iocla>
- ▶ operating systems: <http://ocw.cs.pub.ro/courses/so>

27 / 62

- ▶ attacker goals
- ▶ attacker arsenal: threat model
- ▶ attacker steps: attack vector

29 / 62

System Model

- ▶ application control flow graph
- ▶ applications + configuration + hardware
- ▶ privileges level
- ▶ input (and output)

30 / 62

Scenarios

- ▶ attack iOS Device
- ▶ attack social network
- ▶ attack bank website
- ▶ attack national power grid

31 / 62

Defender Actions

- ▶ prevent
- ▶ isolate
- ▶ react & mitigate

32 / 62

Scope

- ▶ application
- ▶ operating system
- ▶ hardware
- ▶ I/O
- ▶ networking, infrastructure

34 / 62

Security Objectives

- ▶ privacy
- ▶ safety
- ▶ anonymity
- ▶ integrity
- ▶ confidentiality
- ▶ availability

35 / 62

Security vs. Safety

- ▶ safety is state
- ▶ security is process
- ▶ safety is outcome of a secure process/procedure
- ▶ security: umbrella, safety: warm and not getting wet

36 / 62

- ▶ validation among parties
- ▶ chain of trust: bottom-up trust
- ▶ trust anchor
- ▶ certificate chain, digital signature, certification authority

37 / 62

- ▶ authentication: letting an entity be part of the system
- ▶ authorization: providing privileges to a given entity (subject) for resources (object)
- ▶ access control: checking whether subject can access object
- ▶ audit: inspecting action history, validating behavior

38 / 62

A* commands

- ▶ authentication: `login`
- ▶ authorization: `chmod`
- ▶ access control: `ls`
- ▶ audit: `find`, `auditd`

39 / 62

Least Privilege

- ▶ each subject is only provided the permissions it needs
- ▶ if it's not mandatory, ditch it
- ▶ sandboxing, isolation

40 / 62

Privilege Separation/Escalation

- ▶ split privileges among entities
- ▶ reduced actions for super user
- ▶ may require escalation (i.e. one subject may get privileges from another)
 - ▶ be really careful about that
 - ▶ `sudo`, `setuid`, capabilities

41 / 62

Trusted Computing Base

- ▶ security-critical parts of the computer system
- ▶ bugs inside TCP compromise security
- ▶ i.e. privileged parts of the computer system
- ▶ aim to reduce TCB

42 / 62

Attacker Mindset

- ▶ maximize profit
- ▶ time is on your side
- ▶ brute force may be an option
- ▶ you don't care as long as it works
- ▶ target many, get one

44 / 62

Attacker Objectives

- ▶ control
- ▶ cripple
- ▶ steal (information leak)

45 / 62

Threat/Adversarial Model

- ▶ decompose application, identify threats, determine countermeasures
- ▶ classify possible attacker actions and attacker flow

46 / 62

Vulnerability

- ▶ misconfiguration, weakness
- ▶ exposes a risk that may be exploited for unintended outcome
- ▶ i.e. buffer overflow, integer overflow, unsanitized input

47 / 62

Exploit

- ▶ method to use a vulnerability for malicious outcome
- ▶ take advantage
- ▶ i.e. ret-to-libc, ROP, send injection code

48 / 62

Attack Vector

- ▶ steps/path to deliver a malicious outcome
- ▶ composed of attack steps (attack gadgets)
- ▶ usually chaining together exploits towards outcome
- ▶ i.e. SQL injection, actual real world attacks

49 / 62

Attack Surface

- ▶ parts of the system exposed to attacks
- ▶ entry points into system
- ▶ either valid entry points that may be used invalidly
- ▶ or invalid entry points
- ▶ goal is to reduce attack surface (think TCB)

50 / 62

Defender Mindset

- ▶ malicious vs. negligence
- ▶ time is limited
- ▶ proactive before reactive
- ▶ prevent
- ▶ monitor
- ▶ defense in depth

52 / 62

Proactive vs. Reactive

- ▶ similar to medicine, treating disease
- ▶ analyze threats beforehand
- ▶ deploy mitigation solutions
- ▶ react when sh*t happens
- ▶ have solutions in place

53 / 62

Defense in Depth

- ▶ multiple defense layers
- ▶ redundancy in defense
- ▶ assume one layer falls, others take its place

54 / 62

Phases for Defensive Mechanisms

- ▶ harden
- ▶ prevent
- ▶ confine
- ▶ treat

55 / 62

Harden

- ▶ analysis, enhancement
- ▶ off-line
- ▶ static, dynamic analysis
- ▶ verification and validation
- ▶ fuzzing, symbolic execution
- ▶ remove, treat **vulnerabilities**

56 / 62

Prevent

- ▶ make it difficult / impossible for exploits to happen
- ▶ on-line, during runtime
- ▶ i.e. ASLR, DEP, stack guard, ASan

57 / 62

Confine

- ▶ when sh*t happens, reduce damage
- ▶ limit resources use, limit interface
- ▶ as reduced privileges as possible
- ▶ ideally customized (tailored) per app/system
- ▶ sandboxing, virtualization, access control

58 / 62

Treat

- ▶ if sh*t happens, solve it
- ▶ remove threat, remove damaged resources
- ▶ replace components
- ▶ apply lessons learned

59 / 62

Monitor

- ▶ know when sh*t happens ASAP
- ▶ proper monitoring levels; otherwise you ignore it
- ▶ monitor as much as you can
- ▶ key for availability

60 / 62

Keywords

- ▶ systems security
- ▶ Security of Information Systems
- ▶ goals
- ▶ safety
- ▶ trust
- ▶ trusted computing base
- ▶ least privilege
- ▶ threat model
- ▶ vulnerability
- ▶ exploit
- ▶ attack vector
- ▶ attack surface
- ▶ defense in depth
- ▶ harden
- ▶ prevent
- ▶ confine
- ▶ treat
- ▶ monitor

62 / 62