Course Project

Final Project: Objectives

- Learning new concepts
- Gaining hand-on experience
- Making an impact

Final Project

- This is your opportunity to explore or dig deeper in a specific security-related topic.
- Related to **systems** and **networking** topics
- Has to have an implementation component
- **Highly recommended** to discuss with me and/or in the discussion board

Checkpoints – Hard deadlines

- Proposal:
  - Team formation
  - Initial project idea
  - Unsuitable project ideas may require resubmission in one week

- First report:
  - First project progress report
  - Focused on helping you with feedback on project direction

Checkpoints – Hard deadlines

- Second report:
  - Second progress report
  - Each group will send their progress and initial results
- Presentation:
  - Project demo/presentation session
  - Use the feedback to finalize your project
- Final report:
  - Project code and report

Project Proposal (2—4 pages)

- **Group members** and **Project title**
- **Problem** definition
  - Objective
  - Scope
  - Importance
  - Challenges
- Initial **idea/solution** (or at least the approach)
  - Precisely describe the **outcome** (or software artifacts).
- How you're going to **implement** your solution
- Tech. stack: potential **libraries** and **software** to be used
- Detailed **evaluation** plan (e.g., setup, datasets, VMs, etc.)
- High-level **plan** (timeline, task breakdown, task assignment)

Progress report structure

- Introduction: explain project direction, motivation and challenges
- Related work: what is the current state of the art on this problem? What will you compare your work with?
- Proposed solution
  - Overview
  - Details
  - Analysis
  - Limitations
- Current progress on solution

Final Report Structure

- Similar to progress report structure, but should also have:
- Abstract
- Evaluation of proposed solution
  - Define metrics
  - Present results that cover possible counter-arguments
- Conclusion
  - Future work/learned lessons

Project Ideas – Examples

- Reproducing (complex) Attacks and Defenses
- Reproducing research papers (related to security)
- Implementing security-related tools
- New research ideas
  - New attack/defense
  - New architecture or component

Grading

- Two progress reports: 5% each
  - Progress reports are loosely graded; main point is to get you feedback
- Presentation: 30%
  - Graded on quality, delivery, communication
  - Quality of work at this stage does matter as it is shown in the demo
- Final report: 20%
- Code deliverable: 40%, of which:
  - Implementation: 25%. Produce working code that is well organized and documented, and easily reproducible
  - Novelty: 15%. Produce an interesting result that surpasses prior work.

Open Source Code: Guidelines

- If your project idea is implemented somewhere else:
  - You cannot use that code; you need to implement it by yourself
- What type of libraries can I use?
  - A library that doesn't directly implement your main/code idea
  - Helper utilities
- If in doubt, ask me.

Examples

- Not suitable:
  - Reproducing a paper, but you are using its source code
  - Creating a security tool, but you are using the source code of a tool that achieves the same purpose
- Suitable:
  - Using a known library to create your tool with new features

Reproducing Attacks and Defenses

- DNS Rebinding Attacks
- SDN-related Attacks
  - The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links
  - Attacking the Brain: Races in the SDN Control Plane
- Bypassing Virtualization/Sandboxing
- Side-channel Attacks

Reproducing research papers

- Examples:
  - BlindBox: Deep Packet Inspection over Encrypted Traffic
  - Embark: Securely outsourcing middleboxes to the cloud
  - The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links
  - Attacking the Brain: Races in the SDN Control Plane
  - …

Implement/improve security-related tools

- One metric if you're improving an existing tool:
  - your code is merged to a popular open source tool
- Security-related dev tools:
  - Static and dynamic code analysis: Discover bugs and vulnerabilities
  - Compiler instrumentation
- Attack-based/enumeration tools:
  - nmap
  - ROPGadget

Implement/improve security-related tools

- Defense-based tools:
  - IDS, IPS, Firewall
- End user tools:
  - Tor (privacy)
  - VPN (and Wireguard)
- Security-related protocols

Project Ideas [A sample from Spring'20—'21]

- ROP gadget finder
- Virtual Private Networks
- DNS rebinding attacks
- Detecting and repairing control flow hijacking attacks
- Analyzing software source code for vulnerability detection
- Reproducing sandbox escape vulnerability
- eBPF-based intrusion detection engine

Other ideas

- New attack/defense
- Security issues in serverless/container platforms
- Are vNICs secure?
- Detecting malicious IoT behavior (large-scale, distributed env.)
- Attacks based on traffic analysis, e.g., Website fingerprinting
- Detecting caching policies in web/video servers
  - Find the worst-case scenario → launch DoS attack
  - Recall the PHP hash collision attack!
- Security of self-driving vehicles
  - Example