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

- Refer to CourSys webpage for dates

1. Project proposal
   - Team formation
   - Initial project idea
   - Unsuitable project ideas may require resubmission in one week

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

Checkpoints – Hard deadlines

3. Second project progress report
   - Each group will send their progress and initial results

4. Project demo/presentation session
   - Use the feedback to finalize your project

5. Project code and report

Progress proposal 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?
  - This is very important!
- Proposed solution
  - Overview, objectives
  - Technical details: technical stack
  - Challenges
- Proposed timeline
  - What are your own objectives? When will they be achieved?
  - Task breakdown

Progress report structure

- Expand on related work as you learn more about the field and the state of the art
- Improve your solution
- Current progress on solution
  - Describe new challenges encountered
  - Describe changes in your approach/plans
  - Show that you have been working on the project
  - Tackle team issues **now**
- Evaluation Plan
  - How will you know if your solution succeeded?

Final report structure

- Similar to progress report structure, but should also have:
- Abstract
- Evaluation of proposed solution
  - Define metrics
  - Present results
  - Discussion of limitations; criticize your own work
- 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
  - Shows good progress, good/evolving understanding of the field you're tackling
- Presentation: 30%
  - Graded on communication, clarity, organization, and content
  - Quality of work at this stage does matter as it is shown in the demo
- Final report: 20%
  - Graded on quality of writing, organization, and content
- 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

My idea is to create a network mapping tool, can I use `nmap`?

No

My idea is to reproduce “Paper X”. I found its source code online, can I use it?

No

Examples

My idea is to improve “Paper X”. I found its source code online, can I use it?

Check with me first

My idea is to create a ML-based anomaly detection for IDS, can I use `pytorch`?

Okay

Reproducing Attacks and Defenses

- Examples:
- Anomaly detection
- 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
- Network traffic analysis

Reproducing research papers

- Search for papers at top security conferences:
  - IEEE Security & Privacy
  - USENIX Security
  - NDSS
  - ACM CCS

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 from previous years

- Deep packet inspection implementation
- Explainable AI for anomaly detection
- DNS rebinding attacks
- Kubernetes automatic remediation
- Reproducing sandbox escape vulnerability
- GitHub Actions vulnerabilities
- Detecting and repairing control flow hijacking attacks
- Analyzing software source code for vulnerability detection