Term Project

Why a Project?

This course covers a very large discipline, and—perhaps more so than many other areas of computer science—the discipline of computer security runs through many other areas. Because the class has a very limited amount of time, we will only touch the surface of many topics. The project is to give you an opportunity to explore one of these topics, or some other area or application of computer security that interests you, in some depth.

The Ground Rules

The project can be a detailed research paper or survey, or a programming project. Please select something that interests you!

You may work individually, or in groups of up to 3 people (if you want to have more than 3, please come see us ***first***). Of course, the larger the group, the more we will expect from it.

What Is Due and When

Please submit the following on the dates indicated:

- *Question* (10% of project grade): due Wednesday, October 9. Submit the names and email addresses of your team members and the question that you want your project to answer, or to contribute to the answer.Your question must be appropriate for a research project, so do some background work to help you select a good question that you can answer. You only need turn in the question, not the research — you can do that at the next step.
- *Progress Report* (20% of project grade): due on Friday, November 1. This is a progress report. Say what you have done, the structure of your report or write-up (for software). If you are developing software, describe what that software is to do in detail, and how far along you are. Experimental procedures and results should be described here.
- *Presentation* (5% of project grade): due on Wednesday. November 27. Each group will be asked to give a ***short*** presentation of their project. The exact times and dates will be decided later in the quarter, based on the number of project teams. This is an opportunity to get feedback on your project, as it is not yet due. If there is not sufficient time, your presentation will be graded on the basis of your slides, so do them well.
- *Completed Project* (65% of project grade): due on Friday, December 13 (the day of the scheduled final exam). Turn in your final project.

In all cases, submit the project to Canvas as described in **All About Homework**. If a team has multiple members, only one need submit the material, and the others need to simply submit a note saying who submitted the final project.

Some Suggestions for Project and Report Topics

Below are some suggestions for project topics. Note *this is not a list of questions — those you must develop on your own*! You are welcome to refine these into questions, but you will have to narrow the topic. You may also choose a topic of your own.

- High assurance programming: design, write, and implement a program for the GEMSOS system (we have one in the Security Lab)
- Create a front-end to LLVM that accepts Pascal, and a back end that produces segments and ring control suitable for GEMSOS (this is a *time-consuming* project — your grade depends on how far you get!)
- Develop a DNS server that replicates data among many different copies, so that the DNS can function properly in the face of a denial of service attack.
- Use a process modeling language such as Little JIL or UML to derive the security requirements for computers used in some workflow, such as elections.
- Analyze some aspect of the insider problem, such as the supply chain problem, to detect potential points of attack, and identify and assess possible remediations.
- Analyze one of the open source SDN implementations for potential vulnerabilities. Define your threat model, your assumptions about the environment, and describe the effects of your potential attacks.
- Examine the question of licensure, certification, and liability in the context of software security. What would be necessary for these to force improvement of software creation and distribution?
- Develop a wrapper that checks for non-secure inputs (from the user, from the network, or as returned values from system calls) and a language in which to express what "non-secure inputs" means. Compare this to a jail and other

restricted environments. What attacks does your wrapper stop that other restrictive environments do not?

- Implement a front-end to a compiler (or modify the compiler) to detect potential race conditions. If such a condition depends upon the execution environment (such as inputs or file names) that cannot be validated at compile time, generate code to check at run time. Compare this with other approaches to solving the race condition.
- Implement a proxy that checks the digital signature of signed executables. If there is no digital signature from a trusted source, embed the executable in a wrapper that will warn the user when the executable tries to access sensitive resources or is given escalated privileges. You will have to define, or otherwise specify, what entities are "trusted sources" and "sensitive resources".
- Choose an Internet protocol and analyze it with respect to specific security requirements. Justify your selection of requirements as well as your analysis.
- Perform a source code analysis of an open-source project. State the conditions under which any vulnerabilities you find are exploitable, and what the effects of such exploits are.
- Develop a "compiler" that will take a policy language (such as Ponder or DTEL) as input, and produce the configuration changes needed for a system to enforce that policy. You should choose a type of system (Linux, FreeBSD, MacOS, Windows) that you know well.
- Given a system, reverse engineer its configuration and other settings into a high-level security policy language, or into a natural language statement of the policy that the system enforces.