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 that focuses on validating or working with some formalism or implements a model so others can explore it thoroughly. It can be a formalism, a model, or something else theoretical that we do not cover in class. In any case, check with me before beginning to be sure it is a reasonable project and no-one else has chosen it. 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.

Some Suggestions for Project and Report Topics

Below are some suggestions for projects. If you pick one of these, you will need to refine it or limit the scope of your project. You may also think of a project on your own.

- 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 comile 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.

What Is Due and When

Please submit the following on the dates indicated:

1. *Project selection*: due on Friday, January 22; 10% of project score. Submit a write-up with your team members consisting of a one-line title of your project, a one-paragraph description, and the names of all team members. If

you're doing a programming project, state the problem you want to solve and the requirements for a solution.
2. *Progress report*: due on Monday, February 8; 20% of project score. Submit a one-page progress report, and a bibliography of references that you have used or plan to use.
3. *Completed project*: due on Monday, March 14 (this is the last date of instruction); 70% of your project score. Turn in your final project.

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