CS547: Term Project

Deadline:

Topic of Interest: 10th October 2019 (through Googledoc)

Final report and Slides: Monday, 10th November 19

The final report

You are expected to write a technical report, in the style of a conference submission, on the research you have done.

State the problem you're solving, motivate why it is an important or interesting problem, present your research thoroughly and clearly, compare to any related work that may exist, summarize your research contributions, and draw whatever conclusions may be appropriate.

I expect the project reports will be between 8--10 pages long.

Project topics:

Note: Some examples are very specific. Others are quite generic; for the generic suggestions, be sure to narrow down the topic substantially and propose something concrete and focused. **If you wish to choose the topics of your own**, please discuss with me (Stating the problem and why it is an interesting problem)

Fill the **title** (in Googledoc) as of yours after **narrowing down the topics**, before the deadline. **Area** may be **as specified here**.

1. Security auditing

Audit a widely-used and under-scrutinized open-source package that is security-critical. Report on your experiences and lessons. How would you re-structure/re-implement the system to make it more robust? What tools would have made your auditing task easier? How effective are existing tools?

2. Tools for vulnerability detection

Tools to automate the process of reviewing security-critical source code. You can look for the suitable mechanism like runtime testing, static analysis, model checking, formal verification, or other techniques to detect any interesting classes of common security holes?

3. Validation bugs in Linux kernel

Investigate the bugs in Linux kernel, explain those. Elaborate those bugs and validate them. You can see if the tools available for validating the source code.

4. Virtual machines for security

Evaluate the security of, VMWare virtual machine against malicious attempt to harm the host OS?

Is there any way to structure the virtual machine implementation to isolate the security-critical functionality and thereby make the TCB simpler and easier-to-verify?

5. Security of Whatsapp.

Investigate the security holes in Whatsapp.

6. Privilege separation

We discussed in class, sandbox can be used to prevent untrusted code from affecting the rest of the system in any way. This is a useful primitive, but in practice we often want to allow some amount of controlled sharing or limited interaction. How to securely allow this limited interaction in some application context of interest, such as dividing programs into privileged and unprivileged pieces.

7. Distributed firewalls

Make an analysis of distributed firewalls. Looking at the enforcement mechanism found on a firewall and replicating it on all inside machines. (This would require installation of new software on all internal machines, but suppose we can handle that administrative burden.) How do we maintain centralized control over security policy? How do we specify policies? How do we maximize assurance in such an environment? How do we handle multi-organization scenarios, where a machine is a member of multiple organizations and thus multiple parties would like to add security restrictions?

8. Distributed IDS

Make an analysis of distributed IDS. Discuss the challenging issues and define a methodology to address these.

9. New attacks

Find new security weaknesses in any widely-deployed system.

10. Secure coding

    Improve (somehow) the state of the art in implementation of security-critical projects. You might explore the relevance of various techniques from software engineering or programming languages, for instance.

11. Encrypted databases

    Suppose we want to store our data on a remote server (e.g., so that we can take advantage of the computational power of the server) without requiring full trust in the remote server. What types of database semantics can we support?