Problem Set 2

This problem set is due on *Thursday, September 30, 1999* at the end of class. Late homeworks will *not* be accepted. You are to work on this problem set in groups of three to four people. (You *must* work in groups of three or four; homeworks turned in by individuals, pairs, pentuples, etc. will not be accepted. Be sure that you all understand the solutions turned in to each problem.)

Mark the top of each sheet with your names, 6.857, the problem set number and question, and the date. **Type up your solutions**[1], and be clear. Points may be deducted if your TA has problems understanding your solution. We strongly recommend you get started on this problem set early.

Problem 2-1. Trusting Trust

Please read over the attached article by Ken Thompson. (This is his 1983 Turing Award Lecture; published in the August 1984 issue of *Communications of the ACM*, volume 27, number 8, pages 761–763.)

The assignment is to discuss Thompson's article in the context of Java applets. What points, if any, would need to be revised or changed? What about byte-code verifiers, interpreted languages, or disassemblers? Etc.

Type up *at most* three pages.

Problem 2-2. Queso con crackers?

There are several open source programs on the Internet that do what is called 'TCP fingerprinting'. By sending malformed packets to the target machine and watching the response, these programs can determine (with high probability) the operating system running on the target. They can do this because the target is responding to conditions unspecified in the TCP specification, so its behavior is up to whoever implemented the TCP stack for that OS.

Assess the security implications of the prevalence of these programs on the Internet. What are the threats and vulnerabilities, for individuals, the general public, and large organizations? What safeguards or countermeasures can be taken to reduce these vulnerabilities? Does having the source code to your OS (and the right to modify it) assist you here? Is hiding the type of your operating system just "security by obscurity"? If so, should we not bother with trying to hide this information?

Some URLs of interest:

- Queso: `http://apostols.org/projectz/queso/`
- Nmap: `http://www.insecure.org/nmap/index.html`
- An IDG article `http://www.idg.net/crd_tcp_9-82837.html`

[1] If you latex your solutions, include a URL where I can find the latex file(s); I will distribute my favorite solution for each problem to the class as the 'official' solution – this is your chance to become famous!

- 'How to Handle and Identify Network Probes':
`http://www.securitysearch.net/search/tips_archive/tips12-5-99.htm`

Problem 2-3. Physical analogues of crypto primitives

Often we use physical security mechanisms as models or metaphors for cryptographic or computer security mechanisms. Let's try the reverse. Describe one plausible implementation (if possible) and one plausible application for each of the following. Would there be any interesting consequences of such techniques?

Physical public-key crypto: A mechanical lock that has one key to unlock it and one key to lock it (such that you can't figure out one key from the other).

Physical hash functions: An easy-to-create concise physical representation of an object such that any change in the object results in a change in the physical hash function, and no two objects have the same physical hash function.

Perfect physical security: A cheap safe that is truly impossible to break into without the appropriate key. (Analogous to the one-time pad, which provides perfect cryptographic security.)