
Problem Set 6

Submit this problem set in PostScript, PDF, or MS Word format to 6857-submit@csail.mit.edu before lecture on the due date. We have provided templates for L^AT_EX and Microsoft Word on the course website. Each solution must appear on separate sheets of paper. Mark the top of each sheet with your name(s), the course number (6.857), the problem set number and question, and the date.

You are to work in groups of three or four people and should submit a single set of solutions for all problems parts designated **[Group]**. You should turn in a separate, individual solution to any problems designated **[Individual]**.

Problem 6-1. Dig'em Stacks **[Group]**

Bob has hired you as a consultant to analyze his code for possible security flaws. Unfortunately, Bob is supremely confident in his coding skills and will not believe a security flaw exists unless he can reproducibly demonstrate it.

Bob provides you a snippet of code to analyze that is available at:

<http://crypto.csail.mit.edu/classes/6.857/bob.c> and printed below. Bob's code simply takes an input `char*`, prints out a greeting, then returns the address of its local buffer as a long.

```
#include <stdio.h>
#include <string.h>

/* Bob's code snippet displays a greeting message then
   returns the address of the stack pointer as a long */
unsigned long bob(char* input)
{
    char buf[64];
    bzero(buf,64);
    strcpy(buf, input);
    printf("Hello %s, my name is Bob.\n",buf);
    return (unsigned long)&buf;
}
```

- (a) Demonstrate a buffer overflow attack that allows you to execute `/bin/sh` by calling `bob(char *)` with an appropriately formed input. Provide the source code of your attack. Please ensure that your code is adequately commented, understandable, and formatted for printing. A basic attack can be implemented in less than 20 lines of code.
- (b) Explain how you would conduct the attack in part (A) if Bob did not cheerfully provide his stack pointer address for you.
- (c) Fix Bob's code so that it is not susceptible to this attack.
- (d) Considering Bob's flaw in this code, what other C commands could potentially be abused in buffer overflow attacks?
- (e) Find a buffer-overflow prevention or detection tool and use it on Bob's code. For example, you may want to try StackGuard, StackShield, or ProPolice. Explain what tool you tried, whether it detected Bob's bug, and *briefly* comment on its usability.

Problem 6-2. Polymorphic Pseudoquines **[Individual]**

A *quine* is a program P that generates a copy of its own source text as its complete output, i.e. P is a quine if $\text{execute}(P) \rightarrow P$. Define a *pseudoquine* as a program that outputs another pseudoquine of the same length as the original. Thus, quines are instances of pseudoquines that happen to output themselves.

Write a “polymorphic pseudoquine” P that selects a randomly chosen 6-digit integer $r \in [100000, 999999]$, then outputs another polymorphic pseudoquine P' such that r appears in P' . Turn in the source code for P , and the output P' of one of its executions. Please ensure that your source code is formatted to be clear and easy to read.

You may use an existing quine as a starting point, as long as you cite your source. We may give special recognition to especially simple, clever, or elegant polymorphic pseudoquines.

Problem 6-3. Tagged and Released into the Wild [Group]

Choose three of the following six situations. For each of those three, discuss the potential security benefits and risks of using RFID technology. Based on these benefits and risks, give your opinion of whether RFID use is acceptable in each of these three settings.

You should consider costs, convenience, privacy, security vulnerabilities, and how RFID compares to alternate or existing systems. If you consider RFID to be unacceptable, briefly discuss any countermeasures or policies which could be used to make it acceptable in your view.

State any assumptions you make for your answer. If there are minor tweaks that would in your opinion make the system more acceptable, explain those as well. Limit your answer to a maximum of two pages for the entire problem.

1. A grocery store keeps RFID tags on individual items. They are used to facilitate checkout, to keep track of inventory on shelves, and as an anti-theft measure.
2. People may elect to have an RFID tag implanted under their skin. In the event of a medical emergency, emergency personnel can scan the tag and look up its entry in a database; the database contains not a full medical history but only certain critical information (e.g. “allergic to penicillin”).
3. A university puts RFID tags in the ID cards it gives to students, faculty, and staff, for use in building and lab access.
4. A large retailer puts RFID tags on pallets and crates of products in their warehouses for supply-chain management. The tags track the boxes from the factory to the warehouse to the store, but are not placed on the individual items.
5. A country considers embedding RFID chips in its large-denomination bills, as an anti-counterfeiting measure and also a potential anti-theft measure.
6. RFID tags are used on highways in order to pay tolls. A car need only slow down and drive through a special lane in order to pay a toll—a reader reads the tag on the windshield and deducts the toll from the associated credit/debit account.

Problem 6-4. Program Shepherding [Group]

Describe to what extent, if any, the “program shepherding” technique of Kiriansky, Bruening, and Amarasinghe may be useful in defeating the following kinds of attacks. Justify your answers briefly. Indicate what security policies, settings, and usage mode for program shepherding might be used, if any.

1. The user downloads and executes on his PC a game that has been infected with a polymorphic virus.
2. A sysadmin has modified the system login and compiler binaries in accordance with Ken Thompson’s “Reflections on Trusting Trust”.
3. The emulator within a virus detector contains a bug allowing the (emulated) virus to cause the emulator to be “off-by-one” in certain conditions, causing the virus detection emulator to start emulating code from the wrong location (wrong means not the same as what the hardware execution unit would do when executing this code in native mode).

4. The malware detects the presence of the RIO program shepherding sandbox, and pops up a window to the user asking him to “please turn programing shepherding off, as it interferences the the correct operation of your new browser plug-in”.
5. A browser contains a URL encoding vulnerability, as described in Chien and Szor, page 15 etc.