

CS 395T - Theory and Practice of Secure Systems
Fall 2006

Homework #2

Due: 3:30pm CDT (in class), November 7, 2006

NO LATE SUBMISSIONS WILL BE ACCEPTED

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Sciences department code of conduct can be found at <http://www.cs.utexas.edu/users/ear/CodeOfConduct.html>

Homework #1 (30 points)

Problem 1 (5 points)

Professor Bugsmash proposes a PointGuard-like method for protecting return addresses on the stack. Each time the process starts, a random 16-bit key is generated and stored in secure memory. The compiler is modified so that whenever a function is called, the return address is encrypted using this key before being pushed onto the stack. When returning from the function, the program decrypts the return address and jumps to it.

Suppose the attacker stages a brute-force buffer overflow attack on the program protected in this way, *i.e.*, he tries to guess the key and overflow the return address with a value that, decrypted with the guessed key, gives the address of the attack code. Assume that if the attacker's guess is wrong, the decrypted address looks "random" and the process crashes. After each crash, the process is re-started, the key is re-generated, and so on.

After how many trials, on average, should the attacker expect his exploit to succeed? What if the key is generated only once, and re-used when the process is re-started?

Explain your calculations for both cases.

Problem 2 (4 points)

XFI is intended to be a comprehensive software protection system that provides both memory safety and control-flow integrity. Describe a remote attack that may enable the attacker to take over an XFI-protected Web server, and explain how the attack works and why it succeeds.

Problem 3 (5 points)

Consider a host-based intrusion detection system (IDS) that combines stack monitoring with path profiling. During the training phase, the IDS observes a large number of program executions and collects a library of all pairs of sequential function calls ever made by the program (*e.g.*, it knows that function `mysuid` often calls function `setuid`).

After the IDS is activated, it monitors the stack during program execution. If the IDS detects that the program made a sequence of function calls that was never observed during the training phase, it raises an alarm.

Give a snippet of C code that the attacker can exploit using a standard return address overflow without triggering an alarm. Your code must include at least two user-defined functions. Explain how the attack manages to evade the IDS.

Problem 4

Acme Security developed a tool for detecting TOCTOU bugs such as `access()/open()` race conditions by static analysis of the program's source code.

Problem 4a (4 points)

Suppose the tool performs data flow analysis, but ignores the program's control flow. Write a short snippet of C code with a TOCTOU bug that will not be detected by the tool.

Problem 4b (4 points)

Suppose the tool performs control flow analysis, but ignores the program's data flow. Write a short snippet of C code with a TOCTOU bug that will not be detected by the tool.

Problem 5

Recall the oblivious transfer protocol between the Sender (S) and the Chooser (C) based on the hard-core predicate of a one-way trapdoor permutation. The Sender chooses a one-way trapdoor permutation F (let T be the trapdoor, and H the hard-core predicate of F). Let $b_{0,1}$ be the Sender's input bits, and let c be the bit indicating the Chooser's choice.

The protocol proceeds as follows:

$$\begin{array}{lcl} \text{S} & \rightarrow & \text{C} \quad F \\ \text{S} & \leftarrow & \text{C} \quad y_0, y_1 \quad \text{where } y_c = F(x_c) \text{ for a random } x_c; y_{\bar{c}} \text{ is random} \\ \text{S} & \rightarrow & \text{C} \quad m_0 = b_0 \oplus H(T(y_0)), m_1 = b_1 \oplus H(T(y_1)) \end{array}$$

The Chooser computes b_c as $m_c \oplus H(x_c) = (b_c \oplus H(T(y_c))) \oplus H(x_c) = (b_c \oplus H(T(F(x_c)))) \oplus H(x_c) = (b_c \oplus H(x_c)) \oplus H(x_c) = b_c$.

This protocol is secure assuming both Sender and Chooser are *semi-honest*.

Problem 4a (4 points)

Suppose the Sender is *malicious* rather than semi-honest. Is the above protocol secure? If not, explain precisely what a malicious Sender can do to make his view of the real-world protocol un-simulatable in the ideal world. Explain why simulation fails (if it fails).

Problem 4b (4 points)

Suppose the Chooser is *malicious* rather than semi-honest. Is the above protocol secure? If not, explain precisely what a malicious Chooser can do to make his view of the real-world protocol un-simulatable in the ideal world. Explain why simulation fails (if it fails).