

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

MIDTERM

October 26, 2006

DO NOT OPEN UNTIL INSTRUCTED

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this midterm. 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>

Midterm (45 points)

Problem 1a (3 points)

Give **three** reasons why it is better to search for buffer overflow bugs in a C program when the program is being compiled and not when it is being executed.

Problem 1b (3 points)

Give **three** reasons why it is better to search for buffer overflow bugs in a C program when the program is being executed and not when it is being compiled.

Problem 2 (6 points)

Consider a system call monitor enforcing the following policy. When a thread tries to open a file for writing, the monitor checks whether the file name is on the “forbidden” list (*e.g.*, the name is `/etc/passwd`). If the name is on the list, the monitor blocks the call. Otherwise, the monitor executes the call and returns the file descriptor to the calling thread.

Sketch how a malicious multi-threaded program can stage a TOCTOU attack against this system call monitor in order to open `/etc/passwd` for writing. You can write each thread separately in pseudo-code.

Problem 3 (8 points)

Acme Security Inc. developed a new host-based intrusion detection system designed to detect rootkits and sniffers. For the purposes of this problem, assume that these three possibilities are mutually exclusive: either the host is normal, or rootkit-infected, or sniffer-infected. Acme computed the following accuracy rates for its product:

Type of host	How this host is classified		
	Rootkit	Sniffer	Normal
Rootkit	88%	4%	8%
Sniffer	10%	80%	10%
Normal	3%	7%	90%

For example, when the Acme system analyzes a rootkit-infected host, it correctly classifies the host as rootkit-infected with probability 88%, misclassifies it as sniffer-infected with probability 4%, and misclassifies it as normal with probability 8%. **(Continued on the next page)**

For the purposes of this problem, assume that 1% of all hosts are infected with rootkits, 3% are infected with sniffers, and the remaining 96% are normal.

What is the false positive rate of sniffer detection? Give your calculations.

Problem 4 (7 points)

For the following snippet of C code, write the constraints on `len` and `alloc` of all string variables that might be inferred by the BOON tool (described in the Wagner *et al.*'s paper "A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities"). Write each constraint next to the line where it is inferred.

Will BOON indicate that this code is potentially vulnerable to a buffer overflow attack? If you think it will, which line in the code will BOON consider potentially vulnerable? Why?

```
int debug=0;

char buf[16] = "HELLOWORLD";

char msg[8] = "DEBUG";

if (debug) {
    strcpy(buf, msg); }

else {
    buf[0] = '\0'; }

if (!debug) {
    strcpy(msg, buf); }
```


Problem 6 (5 points)

Trusted Computing architecture relies on a tamper-proof TPM chip on the computer's motherboard. Suppose the user manages to extract the secret signing key from the TPM chip. How can the user use this key to subvert the remote attestation process, *i.e.*, to lie to a remote server about the software running on the machine?

Problem 7 (4 points)

Recall that Kocher's timing attack on RSA works, roughly, by guessing some bits of the secret exponent d and predicting how long the standard RSA decryption $c^d \pmod n$ will take (here $n = pq$ is the RSA modulus). Why doesn't the exact same attack work against the CRT optimization of RSA decryption, which involves $c^{d_1} \pmod p$ and $c^{d_2} \pmod q$ modular exponentiations?