Homework 6

CS161 Computer Security, Spring 2008

This homework will not be collected.

Use this to help prepare for the final exam.

1. **Hardware Support for Dual-Mode Operation**

   Early Intel processors (e.g., the 8086) did not provide hardware support for dual-mode operation (i.e., support for a separate user mode and kernel mode). As a result, most of the systems implemented on these processors did not support multi-user operation. List and explain one potential problem associated with supporting multi-user operation without hardware support for dual-mode operation. Be concise: one or two sentences should suffice.

   A few sample solutions (any one is acceptable):

   - *A rogue process could modify the address space of other processes or of the kernel.*
   - *A rogue process could disable interrupts and avoid getting re-scheduled, so that the rogue hogs all the CPU and other process don't get a chance to run.*
   - *There's no distinction between privileged vs unprivileged instructions, so a rogue process could send I/O commands directly to attached peripherals. For instance, a rogue process could trash the hard disk or snoop on network packets.*

2. **Gesundheit**

   Kachoo!, Inc. has just released a new web service that allows people to sign their web pages. The service does this by appending, hidden inside a special HTML tag at the bottom of an otherwise normal web page, the author's name, the date, and a signature (which contains the author's name and date signed by the author's RSA private key). The

web page itself is unencrypted, but the signature can be validated by downloading http://www.kachoo.com/pubkeys.html (which contains a list of all registered Kachoo! users and each user's public key) to retrieve the author's public key. Explain why this gives a completely false sense of security, by outlining two different ways that you could make it appear that Linus Torvalds has posted a web page saying "Open source is for losers; I've decided to go work for SCO". The definition of "different" is that each attack has a unique fix. For each of the attacks you list, give a countermeasure that the author/viewer could take to protect themselves against that attack.

(a) Attack 1:

(b) Countermeasure 1:

(c) Attack 2:

(d) Countermeasure 2:

Attack #1: Wait for Linus to post some other message on his web site. Copy the name, date, and signature, but modify the contents of the message. The viewer will still receive a valid signature and be fooled.

Countermeasure #1: The contents of the web page should also be included in the input to the signature.

Attack #2: When the viewer downloads `http://www.kachoo.com/pubkeys.html`*, corrupt the response (e.g., send a spoofed response packet) so that it contains a listing for Linus with a public key that is not his. This corruption is possible, since the* `pubkeys.html` *is downloaded over insecure HTTP. The attacker can generate his own keypair and list Linus's name next to the attacker's public key. Then, the attacker can create a web page that is validly signed using this keypair, fooling readers.*

Countermeasure #2: Secure distribution of `pubkeys.html`*. For instance, it might be distributed over SSL. Or, it might be signed with Kachoo!'s private key, and a copy of Kachoo!'s public key might be embedded in every web browser so that the browser can check that this page has not been corrupted.*

3. **One is the Loneliest Number**

In this class, we have seen several different mechanisms for isolating untrusted programs, including virtual memory, system call interposition, and virtual machines.

(a) Name one threat that system call interposition protects against but virtual memory does not.

Opening a network connection (e.g., to attack other machines).
Opening files (e.g., to read secret files or modify the user's data).
Sending signals to other processes (e.g., to kill them).

(b) The military runs a multi-user computer that all government employees can log into; programs that require access to top-secret data are run inside a virtual machine. Richard Stallman is given an account on this computer so that he can install emacs. Colonel Greene runs a copy of Stallman's emacs program inside a virtual machine and uses it to edit the top-secret list of UFOs stored in Area 51's warehouses. (Only Greene has an account on the guest OS running inside the virtual machine.) If Richard Stallman were malicious, could he arrange to learn the contents of this list? If yes, explain how; if no, say why not.

Yes. He could embed a Trojan horse in emacs that uses a covert channel to leak out the contents of the UFO list. For instance, emacs might module the system load to communicate the contents of the file it is editing (1 = do heavy computation for one second, 0 = do nothing for one second). Richard could use his account to monitor the system load and thus receive the secret information that is being leaked by his Trojan'ed emacs.

4. **Secure PIN Entry** We want to allow a user to enter a secure PIN (numeric password) into a terminal. We assume that an adversary can monitor any input (such as a keyboard or keypad) but that the channel of the display to the user (such as a screen) is secure: the adversary cannot monitor it. Give a secure way for the user to enter his or her PIN (the adversary should gain no information about the PIN).

   The display shows a random integer. The user increases it or decreases it (cycling around 0 using an UP or DOWN key on the keypad.) and presses ENTER to enter that digit as a PIN entry value. Note that this gives no information to an adversary about the value of the entry.

5. **Firewalls and Reference Monitors** Explain how the requirements of a reference monitor apply specifically to a firewall. Address the feasibility of determining whether a real firewall meets these requirements.

   There are three properties for a reference monitor: non-bypassable, tamper-resistant, and verifiable. A firewall is non-bypassable if you verify that

ALL network traffic from the outside to/from the inside is mediated by the firewall. It is tamper-resistant if it is designed to resist attacks against its hardware and software components (for example, if the software is contained in non-volatile memory). It is verifiable if we can formally verify that the design AND implementation are correct.

In reality, verify the properties can be very difficult (if not impossible). We can test our network to determine if there are external access mechanisms that are not mediated by the firewall (e.g., modems or wireless access points). We can examine the software for potential vulnerabilities, but for a given hardware/software complexity, it may not be possible to determine whether there are bugs and whether they can be used to alter the firewalls behavior.

6. **Intrusion Detection Systems** Explain succinctly the difference between rule-based intrusion detection and statistical anomaly detection. Give one advantage each has over the other.

   Rule-based intrusion detection uses a list of rules describing known attacks. It looks for matches between traffic and the rules, and is only effective at detecting known attacks. It is easier to explicitly block a known exploit with rules, because we don't have to rely on the known exploit being statistically different from normal traffic.

   Statistical anomaly detection looks for differences between normal behavior and attack behavior. It can be used to detect novel attacks. Statistical anomaly detection has the advantage that it can catch attacks that we did not explicitly write rules for.

7. **Buffer Overflow** Why is having a non-executable stack and heap insufficient to protect against buffer overflow code execution attacks?

   The return address can be overwritten to return to any code already loaded. In particular, the attacker may be able to cause a return into the libc `execve()` function with "/bin/sh" as an argument.

8. **Rootkits** Joe wants to protect himself against rootkits, so he runs a virtual Windows XP system on top of Mac OS X. Is Joe vulnerable to Windows XP rootkits? Why or why not?

   Yes, the guest OS can be infected with a rootkit just like a native system can, since the virtual machine simulates the full (or nearly full) hardware interface.

9. **SQL Injection Attacks** SQL's prepared statements add the "?" syntax to the language:

   select * from foo where bar=?

   "?" can then be replaced with a string using a seperate function "setString()". This is more secure than building up queries by concatenating strings, because "setArgument()" understands enough of the SQL language to ensure that its arguments are properly interpreted at the database server. For example, if the "bar" column contains strings, then "setArgument" ensures that its parameter is a string, and the server interprets it as raw string data, instead of as part of a SQL expression.

   setArgument can be applied in various different points in the query syntax. Which of the following can safely interpret untrusted user input? For each case, explain what setArgument would have to verify, or explain why passing such data in from the user is unsafe:

   a) setArgument takes an integer: "select * from foo where num=?"

   Answer: This is safe; the database server simply interprets the passed in value as an integer.

   b) setArgument takes a set of values: "select * from foo where num in ?"

   Answer: This is safe; the database server simply interprets the passed in value as a string. Strange looking strings like "a'; drop students' " will be matched against tuples in the database, just like any other string.

   c) setArgument takes a nested SQL query: "select * from foo where num in ?"

   Answer: This is fundamentally unsafe. The database server is executing code passed in by malicious users. If the SQL passed in via setArgument is restricted to queries that do not update the database, then this is safe from the point of view of data loss. However, the attacker can take up arbitrary amounts of computation (new versions of SQL are turing complete), and access everything that the database connection has permission to read.

10. **Cross Site Scripting and SQL Injection**

    In class, we saw an example of a cross site scripting attack involving javascript. That example enabled the attacker to authenticate as the victim user, to the victim server. This is a two step attack, requiring the attacker to first obtain the user's cookies, and then authenticate to the

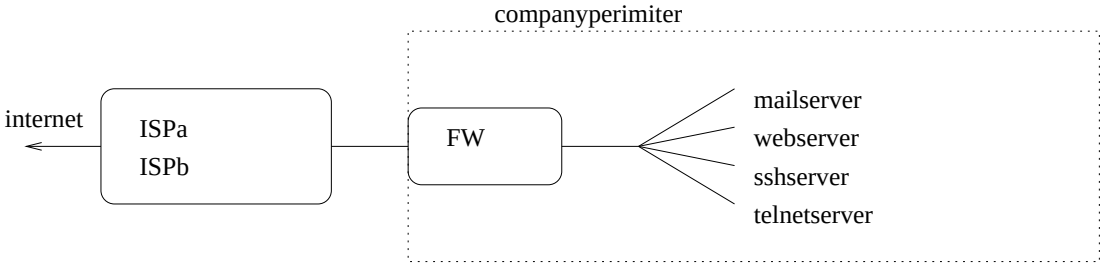
server. Describe how the attacker could develop a more elaborate cross site scripting attack, involving SQL injection along with javascript injection, to eliminate the need for the step where the attacker authenticates as the user. Feel free to make any reasonable assumptions necessary about the victim server, in order to make your attack possible.

Answer:

The malicious server prepares a link that the victim client will click on. This link contains a query to a form that only the victim has access to, with a SQL injection attack. The SQL injection causes the victim server to return sensitive information to the victim client. The malicious link also has javascript code embedded in it, such that the victim client ends up sending the sensitive information to the malicious server, via a javascript command to open a URL pointed at the malicious server.

11. **Firewalls**

The following diagram shows the architecture for your company's network and connection to the internet.



IP addresses:

ISP router	2.2.2.1
Mail server	1.2.3.5
Web server	1.2.3.4
SSH server	1.2.3.3
Telnet server	1.2.3.2

Example rules:

```
allow * *:*/in -> *:*/out
drop * *:* -> *:*
```

Your company is installing a packet filter firewall. Here is the proposed security policy for the firewall:

[I] By default, block all inbound connections.

[II] Allow all inbound TCP connections to SMTP on mail server.

[III] Allow all inbound TCP connections to HTTP and HTTPS on web server.

[IV] Allow all inbound TCP connections to SSH on SSH server.

[V] Allow all outbound connections.

[VI] Telnet access should not be allowed (because it sends passwords in cleartext).

(a) (12 points) Using the syntax from lecture (examples above), write the firewall ruleset for your company's firewall. For each rule, give a brief description of its purpose.

Square brackets around /in or /out mean they may be included but are not strictly necessary.

(i) `allow tcp *:*[/out] -> 1.2.3.5:25[/in]` (allow SMTP)

(ii) `allow tcp *:*[/out] -> 1.2.3.4:80[/in]` (allow HTTP)

(iii) `allow tcp *:*[/out] -> 1.2.3.4:443[/in]` (allow HTTPS)

(iv) `allow tcp *:*[/out] -> 1.2.3.3:22[/in]` (allow SSH)

(v) `drop  tcp *:*[/in]  -> *:23[/out]` (drop telnet)

(vi) `allow tcp *:*/in  -> *:*/out` (allow all outbound)

(vii) `allow tcp *:*/out  -> *:*/in  (if ACK bit set)` (allow TCP responses)

(viii) `drop  *  *:*[/out] -> *:*[/in]` (default deny)

Note that the default deny rule will stop incoming telnet connections.

(b) (8 points) Hackers target your company's network with repeated requests for large images on your company's webserver. The hackers machines are on the `20.1.21.x` subnet. How could you change your firewall ruleset to block these attacks?

Add the rule:

```
drop  tcp 20.1.21.*:*[/out] -> *:*[/in]
```

or:

```
drop  tcp 20.1.21.0/24:*[/out] -> *:*[/in]
```

as the FIRST rule in the ruleset.

(c) (8 points) Employees start downloading lots of movie trailers from the new Pear SlowTime website at `4.3.2.1:80`. How could you change your firewall rules to stop employees from accessing the website?

Add the rule:

```
drop  tcp *:*[/in] -> 4.3.2.1:80[/out]
```

before rule

```
allow tcp *:*[/in] -> *:*[/out]
```