CSci 5271: Introduction to Computer Security

Exercise Set 4 **due: November 20, 2012**

Ground Rules. You may choose to complete these exercises in a group of up to three students. Each group should turn in **one** copy with the names of all group members on it. You may use any source you can find to help with this assignment but you **must** explicitly reference any source you use besides the lecture notes or textbook.

1. Firewall Schmirewall. Sarah is installing a network firewall for her company. Being familiar with the principle of fail-safe defaults, she has configured the firewall to DENY all packets. Now she needs to identify the minimal access rules that will allow her organization to use its Internet connection. For example, her organization will need to be able to send and receive email through the firewall, and uses a central mail server at IP address 10.1.100.100. So she has added rules to the firewall that look like this:

SRC ADDR	DEST ADDR	SRC PORT	DST PORT	PROTOCOL	ACTION
10.1.100.100	*	*	sendmail	TCP	ALLOW
*	10.1.100.100	*	sendmail	TCP	ALLOW

The organization has determined that it will also require the following kinds of internet access:

- Incoming SSH access to a VPN server, at 10.1.100.200.
- Access to the web, through a proxy that whitelists approved sites, at 10.1.200.200.
- Outgoing SSH access to three client sites: 0.1.2.3, 42.42.42.42, and 3.14.15.9.

List the minimal set of firewall rules necessary to allow these connections. List some potential vulnerabilities associated with this ruleset. Can the firewall and proxy servers defend against these vulnerabilities?

2. False Positive Answer. Anderson's chapter 10 details several ways to defeat physical intrusion detection systems (a.k.a. "burglar alarms"). One of the common ones is to artificially create "false" alarms so that the true alarm is ignored. Let's investigate this idea with respect to logical intrusion detection systems.

(a) An old Snort rule says that any HTTP packet that includes "`/..%c0%af../`" should trigger an alarm, as an attempted IIS exploit. Explain why in "normal" usage this rule would have a low false positive rate.

(b) Suppose Eve discovers a web server, `vulnerable.org` that is vulnerable to the IIS unicode exploit and she wants to exploit the hole without having it noticed. What are a few ways Eve can temporarily increase the false positive rate at `vulnerable.org` for the rule, without getting her IP address noticed?

(c) What can you conclude about "advertised" false positive and false negative rates?

3. Virus Virii Sam has invented a brand-new virus detector, ViruSniff, and he claims it is 100% effective - if executable F is a virus, then ViruSniff(F) will output "`VIRUS!!!`".

(a) Does ViruSniff violate the undecidability of the halting problem? Why or why not? (Hint: is there a simple program that can do exactly what Sam says ViruSniff can do?)

(b) Some hackers reverse engineer ViruSniff and post its algorithm online: it turns out that ViruSniff does processor emulation of the first 10000 instructions of an executable, and then applies a fancy signature matching algorithm (that no one seems to understand) to the sequence of instructions and memory changes to decide if the program is a virus or not. Explain how to change any program that runs for at least 10001 instructions, and does not trigger the `VIRUS!!!` alert, to propagate a virus such that the altered program will also fail to trigger the alert. What does your strategy say about Sam's claim?

(c) Assuming that ViruSniff has some false negatives, how would you go about changing ViruSniff to detect a virus that uses your strategy? What effect would this change have on false positives?

4. Denial of Service Denial. Sly and Carl are really concerned about the possibility of DoS attacks against their web server program. Since one way to defend against DoS attacks is to make the attacker do more work, Sly has developed a new module for his web server that he claims will prevent DoS attacks by slowing them down. In Sly's module, every incoming HTTP request is put into a queue, with a timestamp and a "delayed" bit marked as false. When it is ready to serve a request, the web server takes the first request in the queue. If the "delayed" bit is false and there are no other requests from the same IP address in the queue, it serves the request immediately. If the "delayed" bit is false and there is at least one other request from the same IP address in the queue, the "delayed" bit is set to true and the request is re-inserted at the end of the queue. If the delayed bit is set to "true," then the request is served **if** the current time is at least 1 second greater than the request timestamp, and **otherwise** the request is sent to the end of the queue again. This approach extends the time needed for an attacker to fill the web server's request queue.

Inspired by BitTorrent, Carl has a different suggestion for preventing DoS. In Carl's solution, whenever client C downloads a page, he also downloads an ActiveX control that acts as a mini web server for that page and its contents only. Then when the main server starts to be overloaded, it uses HTTP redirects to point new clients to servers running on old clients. The new clients can then download the pages from old clients directly, without using any more of the main server's bandwidth.

(a) Will Sly's scheme work, or not, and why? Give a detailed explanation.

(b) Will Carl's scheme work? Why or why not? (Note that there are clearly some implementation issues to address, such as avoiding the use of clients that are unreachable due to firewalls or closed browsers, but let's assume these are adequately solved)