Sergio Caltagirone
CS 523 – Network Security
Midterm

1) I used various tools to gather information on these sites – all the tools were publicly available over the web, I used no tools available on my own machine (other than a web browser).
Websites used: http://www.phaster.com/find_info_net_traffic.html, uptime.netcraft.com

www.cs.uidaho.edu
* IP Address: 129.101.153.126
* Finger connection refused (attempted to finger root, bruce, admin, administrator)
* www.cs.uidaho.edu is a nickname for dalles.cs.uidaho.edu
* Ping is turned on and responding – avg time: 145.484ms
* WHOIS returned University of Idaho, Administrator: Huba Leidenfrost (1-208-885-6721) huba@udiaho.edu
* DNS Server: dns1.uidaho.edu, dns2.uidaho.edu
* ISP: pnw-gigpop.net
* OS: Solaris 9
* Webservices: Apache/1.3.31, PHP/5.02, PHP/4.3.4, mod_perl/1.25
* Average Uptime: 7 days, Longest Uptime: 110 days (seems to restart the machine once-a-week – from uptime statistics)
* Server is also running FTP server (connected using FTP client)

www.eecs.wsu.edu
* IP Address: 199.237.72.126
* Finger connection refused
* www.eecs.wsu.edu is a nickname for moab.eecs.wsu.edu
* Ping is turned OFF – not responding
* WHOIS returned Washington State University, Administrator: Rick Wegner (1-509-335-0464) rick@wsu.edu
* DNS Server: centaur.it.wsu.edu, dns1.eecs.wsu.edu, dns2.eecs.wsu.edu, chiron.it.wsu.edu
* ISP: Verio
* OS: Red Hat
* Webservices: Apache/2.0.46
* Avg Uptime: 30 days

2) The CVE is the Common Vulnerabilities and Exposures. It is a dictionary, not a database, of standard names for all publicly known vulnerabilities and security exposures. Its purpose is to provide a list of standardized names for vulnerabilities so that products, services and individuals can easily reference the same vulnerabilities.

The benefits of conforming and using the CVE are that it is easily identifiable which vulnerability is being referenced and when exploring for information regarding vulnerability, services offering information will provide the CVE reference. Additionally, it is easy to link between sites using the CVE. I do not think there are any drawbacks to using CVE other than it requires the reliance on one organization (Mitre) to organize and assign the names. Other than that, it is good that there is a common reference to vulnerabilities so that disparate knowledge sources can be brought together. http://www.cve.mitre.org/

3) It is difficult to choose a specific operating system that is more vulnerable than the other because each has their own unique flaws. Microsoft has more vulnerability to viruses and a worm, while Linux with loadable kernel modules, is more susceptible to kernel-level Trojans and root-kits. Using the ICAT database from nist.gov, I found 915 matching records for vulnerabilities with 'microsoft' as the keyword. Also, using 'linux' as the keyword, I found 821 matching records. However, this search also includes any applications running on those platforms, such as Apache, PHP, etc. and is not conclusive for only operating systems. However, if the source code itself was examined, as Stanford researchers did, it was found that in Linux 2.6 there were .17 bugs per 1,000 lines of code (http://www.wired.com/news/linux/0,1411,66022,00.html?tw=wn_story_top5) - which is well below the industry standard. Given these two pieces of information, I would say that the most attacked platform is Windows, but the platform with the possibility of more serious problems is Linux (because of LKM).

4) Recently, BugTraq (http://www.securityfocus.com/archive/1/392354) released information as to Windows vulnerability to LAND attacks. In particular, a LAND attack is sending a TCP packet with SYN flag set, and setting the source/destination IP address and source and destination port to that of the destination machine. The attack causes the victim machine to stop processing TCP traffic for 15-30 seconds, resulting in a denial-of-service. The vulnerability is in the way that Windows processes the TCP stack. The results of the attack are that clients connected to a Windows 2003 server will crash, and the CPU on the server will go to 100%. The attack is very easy to launch, you need to create the packet, send it, sniff it (to save it) and then use tcpreplay to send the packet over again. To stop the attack, use a firewall in front of the server to prevent such malformed packets from reaching the server, additionally, use host-based firewalls on each of the workstations connected to the server.

5) Given the scenario, I would be able to tell an application level root-kit through the use of Tripwire or other file hashing utilities. If the root-kit were deployed on a host, then the file hashing would have changed, alerting me to a potential problem. I believe that the given security stance is sufficient for a 50 user network. Although no other information was given as to the value of the work and cost of loss in the network, these preventative measures would be sufficient. A firewall to block malformed traffic, virus protection for worms and viruses, and tripwire for file protection. In addition, I would suggest a network layer IDS, with a sensor at the gateway to look for potential malicious activity – this would have maybe caught the root kit as well when the attacker tried to connect.

To clear the workstation of the root-kit, I would determine which files had changed since the last known-good hash, and then replace those from backup. If I didn't feel confident in my ability to do that, I would clear the machine and reload with the last known-good configuration and backup.

6) Kernel level firewalls, Windows Firewall, and IPTables on linux are very popular. Ipchains is the older version of the kernel firewall for 2.2.x linux kernels and has been superseded by iptables. These are packet filters that look at each packet entering the TCP stack and use rules to determine whether they will pass through or get dropped. The traffic that these kernel level firewalls are protecting from are: only certain port traffic, certain types of traffic (e.g. TCP/UDP), source/destination address filtering – basically filter on any information in a packet. It allows for ingress and egress filtering and therefore protects from denial of service attacks emanating from your network, and other malformed packets coming in. http://people.netfilter.org/~rusty/ipchains/HOWTO-2.html

7) a) Firewalls are still a very valuable security commodity. The only proof necessary is to show that the recent LAND attack on a Window Server can be prevented by easily filtering out malformed TCP packets – a job specifically suited for firewalls. There are still attacks (such as DoS), which good firewall rules can prevent. Although good design and QA can limit the number of attack possibilities, until software is 100% secure, and network traffic can still harm systems, firewalls will be necessary.

b) A firewall can stop a DoS attack by filtering (or dropping) all packets from a specific address that is executing the DoS attack – preventing these packets from getting to the intended destination and filling up the network queue. The bigger question is, whether a DoS attack can actually be detected. There are certain times when a stream of packets may look like a DoS attack, but actually be legitimate traffic from a domain.

c) To stop DoS attacks there are a number of actions that can be implemented. First, ISPs should do more egress filtering to prevent DoS attacks from leaving their network. Second, packets should have a watermark so that they can be traced back to the source and that source can be legitimately prevented from further attacks. Third, software should be created that prevents bots from infecting systems allowing them from becoming zombies – this would include a greater use of intrusion detection systems. Lastly, the cost of sending a packet should increase so that an attacker would not have the resources necessary to send out such a large amount of network traffic.

8) IRC has long been a domain for the easy transfer of worms, viruses, and Trojans. Recently, it has been used to infect computers and turn them into zombies – this allows another IRC application, called a zombie master, to have the zombies do whatever they want, such as launch a DDoS attack or spread a worm. See http://securityresponse.symantec.com/avcenter/venc/data/backdoor.irc.aladinz.n.html, and http://www.antiviruslab.com/e_news_detail.php?lang=gb&news=3268 for more information. The reason IRC is a problem, is that it was designed while security was not very tight in online communities. The IRC applications allow for arbitrary code to be

executed, and the easy sharing and sending of files to users. To chat securely, one should turn off the ability to receive or send files and also choose a chat client that has not seen many buffer overflow attacks.