CS342 Computer Security **Handout # 13**
Profs. Daniel Bilar and Lyn Turbak **October 17th, 2006**
Wellesley College

Problem Set 4

Due: In class, Friday, October 27th

Overview:

Over the last two weeks of classes, we introduced you to network protocols, network attacks, and gave a malware overview.

One exercise asks you to analyze network traces provided to you. A network trace is simply a recording of network activity passing through a network interface card. This will serve to familiarize yourself with *Ethereal*, the free network protocol analyzer. Ethereal offers abundant display, analysis and filtering options. We would like you to become comfortable reading network traces, also as a preparation for more involved case studies.

Another exercise deals with the three way TCP handshake and asks you to evaluate a scheme to thwart the SYN flood attacks (please refer to lecture handouts for a review of a SYN flood attacks).

The last exercise prompts you to do some research on selected malware. The goal here is to familiarize yourself with concepts and processes involving malware and (hopefully!) pique your interest when we delve into our unit on software vulnerabilities in the coming weeks.

For each problem in the group problems, **you need to write at most one or two paragraphs**. Please answer the questions concisely; remember that clarity of writing, clarity of speech, and clarity of code usually follows clarity of thought.

Group Problem 1 [40]: Network trace: Starting up and going to a website

In these exercises, you are provided with traces of network activity that was captured with Ethereal. First, install Ethereal (or its successor Wireshark) from `www.ethereal.com` or `www.wireshark.org` on the machine you administer. Then, start Ethereal and open `http://cs.wellesley.edu/~cs342/ps4website.cap`. This is a trace of network activity on a Linux machine.

a. After starting the trace capture, the first thing I did was to issue an `ifup eth0`. This is similar to the `ipconfig //renew` you may be familiar with on a Windows machine.

Which packets do you think are direct results of this command? Can you say specifically what these packets accomplish?

b. The second thing I did was to open a browser window and go to the URL `http://www.mit.edu`. What is the purpose of the `DNS A` query for `www.mit.edu` in packet 6? Does this imply anything about how long it has been since traffic was exchanged with `www.mit.edu`?

c. What are the IP address of the machine that was being used to capture this trace? What is the IP address of the DNS server?

d. What is the difference between `DNS A` requests and `DNS PTR` requests?

e. Packets 8-10 show the three way handshake that establishes the TCP connection between the local machine and `www.mit.edu`. What is the value in the sequence number field and the ACK field for each of packets 8-10? Can you explain the purpose of these numbers?

f. Packet 11 takes up 494 bytes. What accounts for the bulk of this space? Look in the detailed view of the packet and describe some of the information sent with the GET request.

g. Go to *Conversations* under the *Statistics* Menu, and click on the TCP tab. How many TCP connections are established to `www.mit.edu`? Which packets show the opening of the other connections? Why might a web browser establish several simultaneous connections?

h. How many bytes total are transferred from www.mit.edu to the local machine? How many bytes total are transferred in the other direction? How do you know? (Hint: Try the *FollowTCPStream* option under the *Tools* menu and use the beginning and ending sequence numbers)

i. There is a Google©toolbar in my browser that displays the "page rank" for each page I visit. Open http://cs.wellesley.edu/~cs342/ps4websiteGoogle.cap and describe what evidence can you find for this in the trace.

Group Problem 2 [15]: Network trace: Remote login connection

In these exercise, you'll examine a trace from an SSHv2 remote login connection. Start Ethereal and open http://cs.wellesley.edu/~cs342/ps4SSH.cap.

Secure Shell (SSH) is a set of standards and an associated network protocol that allows establishing a secure channel between a local and a remote computer. It uses public-key cryptography to authenticate the remote computer. SSH provides confidentiality and integrity of data exchanged between the two computers using encryption and message authentication codes (MACs). For more (harrowing) details, you may refer to http://cs.wellesley.edu/~cs342/SSH2Protocol.html.

The SSHv2 protocol is quite complex, and consists of the following sub-protocols:

- SSH Transport Layer Protocol: This layer is a secure low level transport protocol which provides strong encryption, server authentication and integrity protection.
- SSH Authentication Protocol: This protocol authenticates the client to the server using a set of authentication methods such as password authentication, RSA public key authentication, and host-based client authentication methods defined in the server.
- SSH Connection Protocol: This protocol provides the interactive login sessions, remote execution of commands, forwarded TCP/IP connections, and forwarded X11 connections multiplexed in a single encrypted tunnel.

In our trace, we are going to focus on the Transport Layer protocol, which entails mutually agreeing on the encryption, integrity, and compression algorithms to use and to agree on the keys (and seed values, so-called IVs) for those encryption and integrity algorithms. Simplified, these are the first steps of an SSH connection:

1. Server and client exchange banners. A 'banner' is an identification message, announcing the program version.
2. Client and server both send so-called `MSGKEXINIT` ("Key EXchange INITialization") packets. These packets consists of a nonce, supported algorithms to use for key exchange, server host key formats, symmetric algorithms (both ways), MAC and compression algorithms, supported languages, and more.
3. Upon receipt of `MSGKEXINIT` packet, both client and server decide on a session key exchange algorithm (normally Diffie-Hellman). Diffie-Hellman is a key agreement protocol, and was developed by Diffie and Hellman in 1976. The purpose of Diffie-Hellman is to allow two entities to exchange a secret over a public medium without having anything shared beforehand.
4. Using the D-H exchanged secret, the client and the server build and attain consensus for the two keys that are going to be used for traffic encryption and message integrity checks.

a. In the trace provided, identify the packets that are associated with each the steps above.

b. Where exactly does the encryption start (i.e. which packet is the first encrypted packet)?

Group Problem 3 [20]: Thwarting SYN Flood Alice has the following idea for preventing SYN flood attacks. After receiving a SYN message from a TCP client requesting for connection, the TCP server creates the outgoing sequence number as a function of the incoming information (IP addresses, port number, and sequence number), followed by a secret key, followed by a counter that changes every minute. This scheme is a simplified variant of SYN cookies, a countermeasure of SYN flood attacks.

Alice claims that using this idea, a TCP server doesnt have to drop connections when its SYN queue fills up. Instead it sends back a SYN-ACK, exactly as if the SYN queue had been larger.

a. How does the server handles a received ACK message? (Hint: An attacker may send massive numbers of ACK messages with random sequence numbers. How would the server decide whether an ACK message is valid, thus acceptable, or not?)

b. Can Alice's solution protect a server from flooding attacks? Why or why not? If you think her idea can work: What kind of requirement(s) should the function for computing the outgoing sequence number have? (Hint: having a sequence number, should anyone be able to compute the secret key used?) If not, explain your objections.

c. Why does the secret key have to kept secret by the server from the public?

Individual Problem [25]: Malware analysis New malware pops up every day, and a few instances inflict (world-wide) damage on networks and hosts. Research a corresponding virus/worm from table 1below. A good place to start looking is the CERT advisory database, located at www.cert.org, the security portal SecurityFocus wwww.securityfocus.com, and AntiVirus vendors such as Kaspersky Labs (http://www.kaspersky.com/) and Symantec (www.symantec.com).

We recommend researching several these malware specimens, but you need only submit a writeup for one. The write up should be at most 2 pages for this problem.

Winword/Concept
Win32/Nimda
Win32/Lirva
Win32/Code Red
Win32/Mytob
Win32/Bagle

Table 1: Malware

a. What is the payload, if any?

b. What is/are the specific mechanism(s), if any, by which the malware reproduces itself?

c. What systems are vulnerable to infection?

d. What kind of damage does the malware cause to computers and networks?

e. What preventative measures can one take to avoid being infected?

f. If one becomes infected, what measures should one take to recover from the compromise?

g. What type of assumptions (of the vulnerable host, protocols, programs, services etc.) are being violated?