EECE 412, Spring 2007

Final Examination

Your Family name: ______________________________

Your Given name: ______________________________

Your student ID: ______________________________

Name of your left neighbor: ________________________

Name of your right neighbor: ________________________

Attention: If to answer any of the following questions, you need to make additional assumptions, do so but specify these assumptions explicitly writing "Additional assumptions: …"

#	Points	Out of
1		4
2		15
3		10
4		3
5		6
6		8
7		8
8		6
9		10
10		7
11		2
Bonus		3
TOTAL		79

Problems:

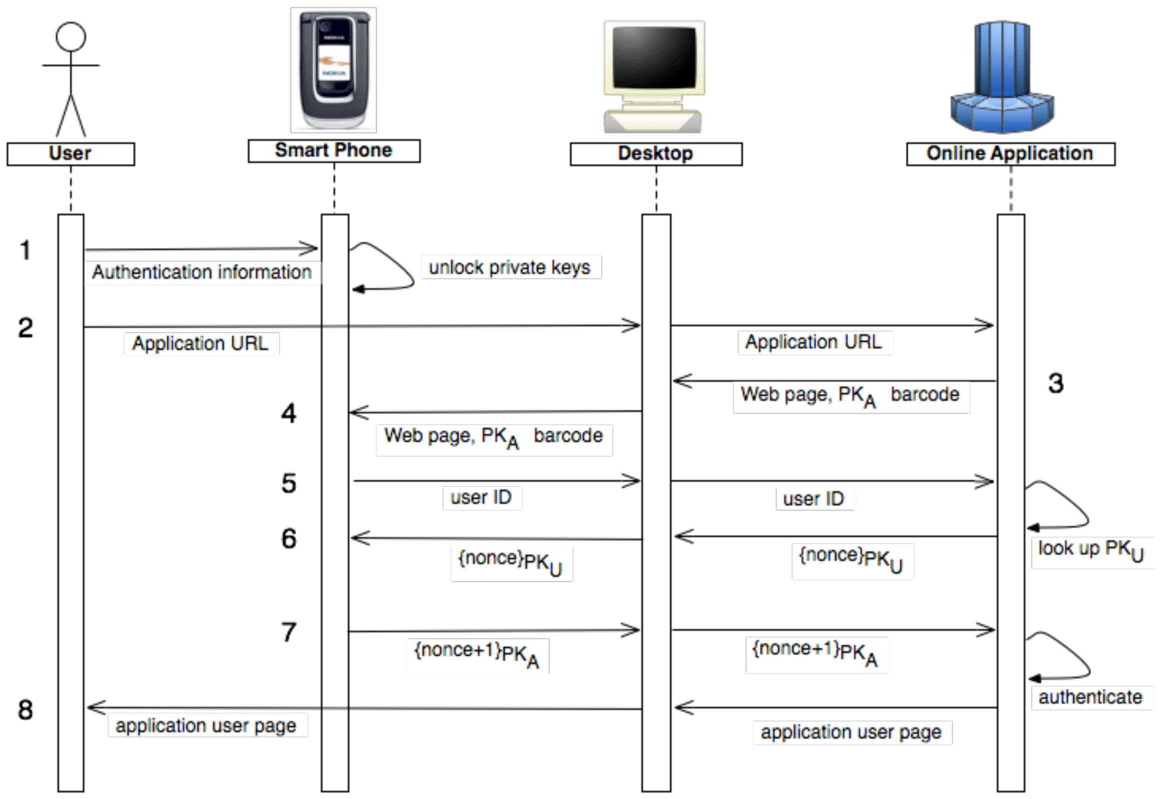
1. (4 points) What are the goals of computer security? Select all applicable.

A. Prevention
B. Investigation
C. Assurance
D. Detection
E. Deterrence
F. Risk transfer
G. Protection
H. Insurance
I. Safety
J. Authorization
K. Recovery
L. Intimidation

Answers: _A, B, D, E, K_____

2. (15 points) You are tasked with the security analysis of the following authentication solution for an online banking application. The application is expected to be very popular among college and university students. Authentication is done as described below and illustrated in the figure.

1. The user first authenticates to their mobile phone using voice-based authentication or fingerprint-based authentication (assume that the phone has fingerprint reader). This step is performed less frequently than the other steps, e.g., once every several hours, subject to the user preferences.
2. Then the user uses a Web browser on a desktop PC to access the web site of the online banking application. The connection is established over SSL with the web site of the application authenticating to the web browser using its X.509 certificate signed by a well-known certificate authority, e.g., Verisign.
3. The returned web page includes the public key of the application (PK_A) in the form of a barcode.
4. The use points the camera of the phone on the barcode, and the phone translates the barcode into the public key, which it uses to lookup the user id for the application and send it back via BlueTooth connection between the phone and the desktop to the application through HTTP POST command.
5. The application looks up the public key (PK_U) that corresponds to the user id, and uses to encrypt the nonce, and send the result back to the phone via the desktop.
6. The phone decrypts the nonce, increments it, and then encrypts the result with PK_A, and sends the result of the encryption back to the application, via the desktop.
7. The application decrypts the incremented nonce, decrements it, and compares it with the value it sent in Step 5.
8. If the comparison is successful, the application logs the user in, and the user starts the session.



What you need to do:

1. **(10 points) Analyze threats and possible vulnerabilities of the above authentication scheme. Explain what can go wrong and why.**

This is an open-ended question. Kosta will mark answers to this question himself.

2. **(5 points) Suggest reasonable improvements to the scheme.**

This is an open-ended question. Kosta will mark answers to this question himself.

3. (10 points)

1. **(3 points) List and explain risk transfer and three other types of risk management.**

   Risk transfer is done commonly through insurance

   Risk acceptance is when the owner of the asset(s) does not do anything about the risk now. Instead, the owner chooses to absorb the corresponding damage when the risk realizes.

   Risk avoidance is when the owner does not conduct the business that is associated with the risk.

   Risk prevention is the use of countermeasures that can mitigate the risk.

2. **(3 points) Give an example—not necessary practical in today settings—of how can the user in the scenario of the previous problem manage risk associated with the use of the online banking application that employs the above authentication scheme.**

This is an open-ended question. Mark answers to this questions to your discretion.

3. **(4 points) If risk transfer allows organizations and users to manage risks associated with the use of computers in general and web applications in particular, are security countermeasures still necessary? Explain why or why not.**

   Security countermeasures allow preventing or reducing the risk in more economically effective ways than risk transfer.

4. (3 points) Give 1-2 examples that illustrate the difference between access control and authentication. Explain your examples.

Login in to a computer by providing user name and password is authentication.

Preventing an unauthorized user, who has logged into the computer, from reading a file is access control.

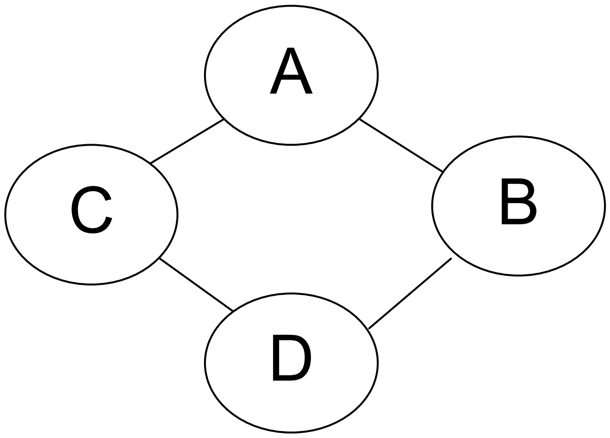
5. (6 points) You are a consultant to the Ministry of National Defence (MND). The ministry is planning to update their physical security system that uses two factors: 1) badge with person's name, picture, and, magnetic strip with the hash value of the 4-digit PIN, and 2) typing in a PIN. They want to keep the authentication system as it is, but increase the strength of the second factor, the PIN. Yes, in military, they can force employees to use random passwords or PINs.

You are asked to evaluate the following two alternatives and offer your opinion which option should be chosen by the MND:
1) upgrading to 6-digit random PINs, or
2) switching to 5-letter password randomly chosen using 26-caharacter English alphabet.

Provide below your recommendation and the rationale for it. Assume that the attacker's goal is to find only the PIN/password for a specific (stolen) badge.

The space of 6-digit PINs is 10**6 , which is 999,999. The average cost is 500,000 attempts. The space of 5-letter password is 26**5 = 11,881,376, making the average case to cost 5,940,688 attempts, which is an increase against a 5-digit PIN by the factor of more than 10. So, the 5-digit random passwords are better and should be recommended.

6. (8 points) For the following BLP lattice



Fill out the psudo-access matrix (follow the example for B x B):

		Objects			
		A	B	C	D
Subjects	A	WR	R	R	R
	B	W	RW	-	R
	C	W	-	WR	R
	D	W	W	W	RW

7. (8 points) The following is a fragment of the analysis of Code Red Worm, damage from which worldwide was estimated to be over $1B. The analysis has been published by eEye Digital Security (www.eeye.com).

ANALYSIS: .ida "Code Red" Worm

Release Date:
July 17, 2001

Severity:
HIGH

Systems Affected:
Unpatched Microsoft IIS Web Servers

Overview:
On Friday, July 13th we received packet logs and information from two network administrators that were experiencing large amounts of attacks targeting the recent .ida vulnerability that eEye Digital Security discovered (http://www.eeye.com/html/Research/Advisories/AD20010618.html) on June 18, 2001.

From the first analysis of the logs that were sent to us we were able to deduce that someone had released a worm for the .ida vulnerability. Within the logs we could see connection attempts from over five thousand IIS 5 web servers targeting various other IIS web servers and sending an .ida exploit to each of them. Evidence also showed that compromised hosts were being used to attack other hosts.

Technical Details:
The following is a detailed analysis of the "Code Red" .ida worm that we reported on July 17th 2001.

This analysis was performed by Ryan Permeh and Marc Maiffret of eEye Digital Security. The disassembly (complete with comments) was done by Ryan "Shellcode Ninja" Permeh.

Table of Contents
=================

Introduction
============

…
The full analysis of the .ida "Code Red" worm provides numerous details as to the functionality and method of propagation of this worm. For instance the worm's purpose ultimately seems to be to perform a denial-of-service attack against www.whitehouse.gov. Also, only US English Windows NT/2000 systems will show the defaced ("Hacked by Chinese!") web page.

We've designated this the .ida "Code Red" worm, first because part of the worm is designed to deface web pages with the text "Hacked by Chinese" and second because "Code Red" Mountain Dew was the only thing that kept us awake while we disassembled this exploit.

Explanation
===========

As stated earlier the .ida "Code Red" worm is spreading throughout IIS web servers on the Internet via the .ida buffer-overflow attack that was published last month.

The following are the steps that the worm takes once it has infected a vulnerable web server:

1. Setup initial worm environment on infected system.
2. Setup 100 threads of the worm.
3. Use the first 99 threads to spread the worm (infect other web servers).
- The worm spreads itself by creating a sequence of random IP addresses. However, the worm's list of IP addresses to attack is not all together random. In fact, there seems to be a static seed (a beginning IP address that is always the same) that the worm uses when generating new IP addresses. Therefore every computer infected by this worm is going to go through the same list of "random" IP addresses. Because of this feature, the worm will end up re-infecting the same systems multiple times, and traffic will cross traffic back and forth between hosts ultimately creating a denial-of-service type effect. The denial-of-service will be due to the amount of data being transferred between all of the IP addresses in the sequence of random IP addresses. The worm could have done truly random IP generation and that would have allowed it to infect many more systems much faster. We are not sure why this was not done, but a friend of ours did pose an interesting idea: If the person who wrote this worm owned an IP address that was one of the first hundred or thousand to be scanned, then they could setup an application to "sniff" the network and anytime and IP address tried to connect to port 80 on their server they would get confirmation that the IP address that connected to them was infected with the worm. With this knowledge, they would be able to create a list of the majority of systems that were infected by this worm.
4. The 100th thread checks to see if it is running on an English (US) Windows NT/2000 system.
- If the infected system is found to be a English (US) system, the worm will proceed to deface the infected system's website. The local web server's web page will be changed to a message that says: "Welcome to http://www.worm.com!, Hacked By Chinese!". This hacked web page message will stay "live" on the web server for 10 hours and then disappear. The message will not appear again unless the system is re-infected by another computer.
- If the system is not an English (US) Windows NT/2000 system, the 100th worm thread is also used to infect other systems.
5. Each worm thread checks for c:\notworm.
- If the file c:\notworm is found, the worm goes dormant.
- If the file is not found, each thread will continue to attempt to infect more systems.
6. Each worm thread checks the infected computer's system time.
- If the date is past the 20th of the month (GMT), the thread will stop searching for systems to infect and will instead attack www.whitehouse.gov. The attack consists of the infected system sending 100k bytes of data (1 byte at a time + 40 bytes overheard for the actually TCP/IP packet) to port 80 of www.whitehouse.gov. This flood of data (410 megabytes of data every 4 and a half hours per instance of the worm) would potentially amount to a denial-of-service attack against www.whitehouse.gov.
- If the date is between the 1st and the 19th of the month, this worm thread will not attack www.whitehouse.gov and will continue to try to find and infect new web servers.

We have calculated that the worm can attempt to infect roughly half a million IP addresses a day. This is a rough estimate generated by testing on a very slow network.

At the time of writing this document (July 19th, 3:00pm), we have had reports from administrators that have been probed by over 196 thousand unique hosts. This leads us to believe that this worm has infected at least 196 thousand computers.

During testing we noticed that sometimes the worm does not execute "normally" and will continue to spawn new threads until the infected machine crashes and has to be rebooted, effectively killing itself. We have not been able to isolate the cause of this behavior.

Deep Analysis
=============

The following is a very detailed analysis of what the worm is doing at each step of its infection. Full disassembled and commented worm code is available on our website at http://www.eeye.com/html/advisories/codered.zip. It is provided as both an assembly text dump and as an IDA (Interactive Disassembler) database file.

Note: We will use CODEREF comments in the following analysis to reference locations in the disassembled code so the reader can follow along while reading.

The tools that we used to perform this detailed analysis are:
- IDA (Interactive Disassembler) from www.datarescue.com. IDA is an advanced disassembler that made this analysis possible.
- MS VC++ debugging environment. We used this to monitor modified pieces of the worm as they interacted with IIS.

We will be heavily referencing the disassembled worm code in the following analysis.

In an attempt to make this easier to understand, we have broken down the functionality of the worm into three parts: the core worm functionality, the worm hack web page functionality, and the attack www.whitehouse.gov functionality.

Core worm functionality

----

1. Initial infection vector (i.e. host is vulnerable to the .ida attack and gets hit with this worm).
The initial infection starts to take place when a web server, vulnerable to the .ida attack, is hit with an HTTP GET request that contains the necessary code to exploit the .ida attack. The worm is used as the attack's payload.
…
EIP is overwritten with 0x7801CBD3 which is an address within msvcrt.dll. The code at 0x7801CBD3 disassembles to *call ebx*
When EIP is overwritten with call ebx, it causes program flow to divert back to the stack. The code on the stack jumps into the worm code that is held in the body of the initial HTTP request.

2. Sets up some initial stack variables
…
3. Load functions (create the "jump table")
The first thing the worm code does is reference the data portion of the exploit code at EBP-198h. The worm then needs to setup its internal function jump table. A function jump table is a stack-based table used to store function addresses. This allows the worm to generate the function addresses at run time (this gives the worm a better chance of executing cleanly on other systems).
…
Finally, the worm stores the base address of w3svc.dll which it will later use to potentially deface the infected website.

4. Check the number of threads the worm has created.
Here the worm appears to perform a WriteClient (Part of the ISAPI Extension API), sending "GET" back to the attacking worm. This could possibly be a way of telling other attacking worms that they have successfully infected a new host.

Next, the worm code will count the number of worm threads already in action. If the number of threads is below 100 then the worm creates a new thread. Each new thread is an exact replica of the worm (using the same code base). If the number of threads is 100 then control is shifted to the worm hack web page functionality.
The worm now continues its path of execution.

6. Checks for the existence of c:\notworm
There seems to be a built in "lysine deficiency" (See Jurassic Park, or Caesar's paper on this at www.rootkit.com). A "lysine deficiency" is a built in check to keep malicious code from spreading further.

In this case, the "lysine deficiency" is a check for the existence of the file c:\notworm. If this file exists then the worm will become dormant. This means it will not attempt to make connections out to other IP addresses to try to infect.

If this file does not exist then the worm continues onto the next step.

7. Check the infected system's time (computer clock).
The worm checks the infected system's local date (in UTC). If the date is greater than 20 UTC, the worm will proceed to the first step of the attack www.whitehouse.gov functionality.

If the date is less than 20 UTC, the worm will continue to try to infect new systems.

8. Infect a new host (send .ida worm to a "random" IP address on port 80).
At this point the worm will resend itself to any IP addresses on which it can find an open port 80 to connect to. It uses multiple SEND()'s so packet traffic may be broken up. On a successful completion of SEND, it closes the socket and goes to step 6...therefore repeating this loop infinitely.

Worm hack web page functionality

This functionality is called after one hundred threads are spawned within the worm.

1. Check if local system default language is English US, then go to step 6 of core worm functionality.
The first thing the worm does is get the local codepage. A codepage specifies the local operating system language (I.E. English (US), Chinese, German etc...). It then compares the local codepage against 0x409. 0x409 is the codepage for English (US) systems. If the infected system is an English (US) system, the worm will proceed to deface the local system's web page. If the local codepage is not English (US), the worm thread will go to step 6 of core worm functionality.

2. Sleep for two hours.
This worm thread now sleeps for two hours. We anticipate that this is built in to allow the other worm threads to attempt to spread the infection before making its presence known via defacing the infected system's web page.

3. Attempt to modify infected systems' web pages in memory.
This worm uses an interesting technique called "hooking" to effectively deface (alter) an infected system's web pages. "Hooking" is modifying code in memory to point to code that the worm provides. In this case, the worm is modifying w3svc.dll to change the normal operation of a function called TcpSockSend. TcpSockSend is what w3svc.dll (IIS core engine) uses to send information back to the client. By modifying this, the worm is able to change data being written back to clients who request web pages of an infected server.

To perform "hooking", the worm first makes the first 4000h bytes of w3svc.dll's memory writeable. In a normal situation, the memory for w3svc.dll (and basically all mapped dll's) is read-only. It uses the function VirtualProtect to change the memory of w3svc.dll to be writeable, saving the old state to a stack variable.

The worm then uses the saved codebase of w3svc.dll (from step 3 of core worm functionality) as a starting point to search the import table (again see PE header documentation) for the address of TcpSockSend. Once the address for TcpSockSend is located, the worm then replaces TcpSockSend's actual address with an address from within the worm.

The address that TcpSockSend points to after the change is a function within the worm that will return the "Hacked by Chinese!" web page.

This thread of the worm now sleeps for ten hours. During this ten hours all web requests to the infected server will return the "Hacked by chinese!" web page.

After the ten hours is up, this thread will return w3svc.dll to its original state, including re-protecting memory.

Execution then proceeds to step 6 of the core worm functionality.

Attack www.whitehouse.gov functionality

Sooner or later every thread within the worm seems to shift its attacking focus to www.whitehouse.gov.

1. Create socket and connect to www.whitehouse.gov on port 80 and send 100k bytes of data (1 byte at a time).
Initially the worm will create a socket and connect to 198.137.240.91 (www.whitehouse.gov/www1.whitehouse.gov) on port 80.

If this connection is made then the worm will create a loop that performs 18000h single byte SEND()'s to www.whitehouse.gov.

After 18000h SEND()'s the worm will sleep for about four and a half hours. It will then repeat the attack against www.whitehouse.gov (go to step 1 of attack www.whitehouse.gov functionality).

What you need to do:

Explain how the worm does the following standard functions: Reconnaissance, Attack, Communication, Command, Intelligence.

Reconnaissance: the worm selects victims with IP addresses that are generated by using a random number generated with each instance of the worm seeding this RNG with the same value.

Attack:

1) make an HTTP GET request from the infected machine to the attacked machine. The request contains payload that
    a. is sufficient to exploit ".ida" vulnerability via buffer overflow, and
    b. contains the worm binary code.
2) Launch 100 replicas of the worm, each as a separate thread.
3) Check for the existence of c:\notworm – "lysine deficiency". Become dormant, if the file exists.
4) If the current date of the month is before 20th, skip this step. Otherwise attack www.whitehouse.gov using the following sub-steps:
    a. Connect to the target on port 80 and send 100KB of data by sending 1 byte at a time, 18000H times.
    b. Sleep for about 4.5 hours, then go back to step a.
5) Infect a new host with a "random" IP address.
6) Flooding attack of www.whitehouse.gov
    a. If the local code page is not English, go back to step 3.
    b. Sleep for 2 hours.
    c. Insert an intercepting function that will deface the infected system's web pages.
    d. Sleep for 10 hours.
    e. Cancel the defacing of the infected system's web pages and go back to Step 3.

Communication: Sends from newly attacked victim HTTP GET request back to the previously infected machine, from which the newly attacked victim has been attacked.

Command: No special-purpose interface for receiving commands has been used. However, because the .ida vulnerability was not patched. Each instance of the worm could have been shutdown by another instance infecting the computer and creating c:\notworm file.

Intelligence: Possibly the attacker used the weak randomness of the IP generation to see which machines were infected because the each victimized machine starts attacking other machines and inevitably attacks machines from not-very-random list of the IP addresses.

8. (6 points) Write logging (i.e., what should be written into the log) and auditing requirements (i.e., on which condition should an analyzer trigger alarm) to detect Code Red attacks. Explain your answer.

Option A:
Logging: HTTP GET requests that contain .ida buffer overflow payload.
Auditing: Trigger the alarm when the machine receives or sends such a request.

Option B:
Logging: HTTP GET requests and process number.
Audit: Trigger the alarm when same process receives and sends HTTP GET requests.

Option C:
Logging: Sending 1 byte of HTTP data to www.whitehouse.gov or www1.whitehouse.gov
Audit: Trigger the alarm on any such event.

9. (10 points) Explain

1) (5 points) Which principles of designing secure systems have been violated by the owners of those computers that were compromised by Code Red Worm.

This question is somewhat open ended. So, use your own judgement for marking answers to this question. Here are some examples from Kosta of right answers:

Principle of least privilege: Web server should not have privilege to send HTTP requests, or even make HTTP connections.

Principle of fail-safe defaults: the buffer overflow should have resulted in system crash, not in the execution of malicious logic.

Defence in depth: the size of the allowed payload for HTTP GET should have been limited. The firewalls should have been configured to stop .ida attacks, and to deny Web servers to make HTTP connections and HTTP GET requests.

Least common mechanism: data on the stack should have not been interpreted as program code.

2) (5 points) How should the above mistakes have been corrected in a practical way.

This is also an open-ended question. Here are some examples from Kosta

1) Either host firewall or the network firewalls should have been configured to not allow machines dedicated as Web servers to make HTTP connections or, even more, HTTP GET requests.
2) The Web server should have been compiled with built-in mechanisms for crashing a process on buffer-overflow.
3) The operating system should have been designed to treat data on the stack as data only, and not as program code.
4) A proxy that limits the size of payload for HTTP GET requests could have been placed in front of the web server.

10. (7 points) You are tasked with offline analysis to collect evidence related to Code Red from the computers of your employer, which is an oil company. The evidence will be used in criminal investigation for the court case against the author(s) of Code Red Worm.

Which of the following are the best practices of handling evidence for Code Red? Select all applicable:

A. disrupt chain of custody
B. shutdown the infected computer and transport it to a secure location
C. reboot the infected computer
D. avoid tools that use command-line interface
E. document the hardware configuration of the system
F. preserve the state of the suspect computer
G. reinstall the Windows on the infected computer

Your answer:____________E, F__________________________

11. (2 points) What is the purpose of making pictures of the victim computer when collecting forensic evidence from it?

Pictures of serial numbers, cable connections, and other details of the computer hardware help 1) to make sure that description written by the investigators are correct and 2) to identify typos and mistakes due to human error.

Bonus question (3 points): You are a network administrator at an oil company. As such, you can only make changes to network routers, switches, and firewalls. Which countermeasures would you employ in order to mitigate the threat of Code Red? Explain your answer.

Private VLANs and protected ports limit the damage a compromised machine can cause to its neighbors on the same VLAN. If either of these mechanisms had been used then the infected machines would not be able to infect other machines of the same organization.