CS 161 Spring 2010 — Computer Security — Paxson/Wagner — Final Exam

PRINT your name: ______________________________, ______________________________
(last) (first)

SIGN your name: ____

PRINT your class account login: `cs161-`____________

Your TA's name: ____

Your section time: ____

Name of the person sitting to your left: ________________________ Name of the person sitting to your right: ________________________

You may consult one sheet of paper (double-sided) of notes written for this final exam, plus the two sheets of paper you brought to midterm 2 (one of which was from midterm 1). You may not consult other notes, textbooks, etc. Calculators and computers are not permitted. Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there.

You have 180 minutes. There are 9 questions, of varying credit (200 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Problem 1		Problem 6	
Problem 2		Problem 7	
Problem 3		Problem 8	
Problem 4		Problem 9	
Problem 5		Total	

Problem 1. [Spoofing attacks] (18 points)

Usually, the DNS protocol runs over UDP. However, it is also possible for DNS to use TCP.

(a) Suppose you are using your laptop on an open wireless network and an attacker is within range of the wireless network, so the attacker can eavesdrop on all your traffic and inject forged packets. **Circle one** of the following that best describes the threat the attacker poses:

1. The attacker can successfully inject a spoofed DNS response if your laptop uses UDP for all of its DNS queries, but not if it uses TCP for all of its queries.
2. The attacker can successfully inject a spoofed DNS response if your laptop uses TCP for all of its DNS queries, but not if it uses UDP for all of its queries.
3. The attacker can successfully inject a spoofed DNS response if your laptop uses either TCP or UDP for its DNS queries.
4. The attacker cannot successfully inject spoofed DNS responses.

(b) Suppose you access the Internet over a secured Ethernet network, so that the attacker cannot eavesdrop on your traffic, but the attacker can still inject forged packets. You can use either TCP or UDP for your DNS queries. Assume that the relevant TCP implementations choose Initial Sequence Number (ISNs) uniformly at random, and that the relevant DNS implementations do not implement source port randomization. Regarding Kaminsky-style "blind spoofing" of DNS replies, **circle one** of the following that best describes the threat the attacker poses:

1. When you use TCP for your queries you are safer (harder to attack) than when using UDP.
2. When you use UDP for your queries you are safer (harder to attack) than when using TCP.
3. You are equally vulnerable to the attack whether you use UDP or TCP.
4. In this scenario, you are not vulnerable to the attacker regardless of whether you use UDP or TCP.

(continued on next page)

(c) Suppose we could deploy a mechanism that would ensure IP source addresses always correspond to the actual sender of a packet—in other words, suppose it is impossible for an attacker to spoof source addresses. **Circle all** of the following threats that this mechanism would *completely eliminate*. By "eliminate a threat", we mean that the anti-spoofing mechanism would suffice to prevent exploitation of the threat, without any additional mechanisms or assumptions.

1. Buffer overflow attacks
2. Cross-site request forgery (CSRF) attacks
3. TCP SYN flooding
4. TCP RST injection
5. Spam
6. None of the above

(d) Again suppose we could deploy a mechanism that would ensure IP source addresses always correspond to the actual sender of a packet. **Circle all** of the following threats for which this mechanism would eliminate at least *some* common instances of the attack but not *all* instances. "Eliminate an attack instance" refers to preventing that attack instance from succeeding, without any additional mechanisms or assumptions.

1. Buffer overflow attacks
2. Cross-site request forgery (CSRF) attacks
3. TCP SYN flooding
4. TCP RST injection
5. Spam
6. None of the above

Problem 2. [Reasoning about memory-safety] (24 points)

Consider the following C code:

```
void delescapes(char s[], int n) {
    int i=0, j=0;
    while (j < n) {
        if (s[j] == '%') {
            j=j+3;
        } else {
            s[i] = s[j];
            i=i+1; j=j+1;
        }
    }
}
```

We'd like to know the conditions under which `delescapes()` is memory-safe, and then prove it.

On the next page, you can find the same code again, but with blank spaces that you need to fill in. Find the blank space labelled `requires:` on the next page, and fill it in with the precondition that's required for `delescapes()` to be memory-safe. (If several preconditions are valid, you should list the most general precondition under which it is memory-safe.)

Also, on the next page fill in the three blanks inside `delescapes()` with invariants, so that (1) each invariant is guaranteed to be true whenever that point in the code is reached, assuming that all of `delescapes()`'s callers respect the precondition that you identified, and (2) your invariants suffice to prove that `delescapes()` is memory-safe, assuming that it is called with an argument that satisfies the precondition that you identified.

Keep in mind that, as emphasized in the last homework, invariants should be self-contained and state all facts that are needed for a proof of memory-safety.

You may ignore the possibility of NULL dereference errors for this question.

You may use the notation `size(s)` to refer to the amount of memory allocated for `s`.

Here is the same C code again, this time with space for you to fill in the precondition and three invariants. Remember, fill in all blanks.

```
/* requires: ________________________________________________________ */
void delescapes(char s[], int n) {
    int i=0, j=0;
    while (j < n) {

        /* ___________________________________________________________ */
        if (s[j] == '%') {
            j=j+3;

            /* _______________________________________________________ */
        } else {
            s[i] = s[j];
            i=i+1; j=j+1;

            /* _______________________________________________________ */
        }
    }
}
```

Problem 3. [Espionage] (25 points)

For this problem assume the existence of a super-top-secret *Incredibly Valuable Document* (IVD) belonging to a business competitor. You are desperate to read the IVD and are prepared to undertake dubious measures to do so. The IVD is 100 KB (kilobytes) in size and you already have a copy of it encrypted using a 128-bit AES key. You have 3 avenues available for reading the document:

- **Buy bots**, at a cost of \$1 per 2^{10} bots. Each bot can brute-force 2^{40} keys per week. (FYI, this is about 1.8 million keys per second.) You can buy up to 2^{24} ($\approx$ 17 million) bots and then, alas, the underground market has no more to offer you.
- **Bribe** an employee of the competitor who has access to the key used to encrypt the document. The employee is willing to leak out to you, via a covert channel, 4 bits of the key each week. Because you blew it and let on to the employee how desperate you are, each set of 4 bits will cost you \$100,000.
- Employ an incredibly sophisticated **side channel attack** that can remotely monitor the power lines going to the competitor's machine room and from their tiny fluctuations infer 1 KB of the document each week. (Each week you obtain a new 1 KB chunk.) This scheme, however, costs \$1,000,000 up front for lasers and superconducting magnets and cool touch-screen technology.

You can employ any or all of these as you see fit. You may use a combination of these methods.

(a) Suppose that what's most important is recovering the document as quickly as possible, in terms of *guaranteed running time*. Which approach will do so, roughly how long do you expect it to take (in weeks), and how much will it wind up costing?

Quickest approach:

Approximately how many weeks it will take:

Approximately how much it will cost:

(b) Suppose that what's most important is recovering the document as cheaply as possible, providing you get your hands on it within 5 years (260 weeks). Which approach will do so, roughly how much do you expect it to cost, and about how long will it take (in weeks)?

Cheapest approach:

Approximately how much it will cost:

Approximately how many weeks it will take:

Problem 4. [Detecting attacks] (30 points)

You work for a company that sells intrusion detection systems. When you start at the company, the core technology is a signature-based scheme we'll call *S*. A little later the company acquires a competitor whose core technology is an anomaly-based scheme we'll call *A*.

Suppose that *S* is a network-based scheme that works by passively analyzing individual UDP and TCP packets. Suppose that *A* is a host-based scheme that is a component of the browser that processes and analyzes individual URLs before they're loaded by the browser. Scheme *S* operates in a stateless fashion and scheme *A* maintains state regarding URLs it has previously analyzed.

(a) For each of the following, circle your answer or answers.

1. With regard to the general properties of different types of detectors, **circle all** of the following that are correct:

   i. To use a signature-based approach like scheme *S* at a new site first requires access to logs of the site's historical activity in order to train the detector.

   ii. It is possible to design a signature-based detector that has no false negatives.

   iii. Specification-based approaches work well for detecting known attacks but do not work well for detecting novel attacks.

   iv. One appealing property of network-based detectors is that they can provide easier management than host-based detectors.

   v. An attraction of behavioral approaches is that they are especially well-suited to prevent initial system compromises.

   vi. None of the above.

2. Your company wants to deploy a new product that provides intrusion prevention functionality, building upon either *S* or *A* as a foundation. The best candidate scheme for this is (**circle just one**):

   i. Scheme *S*, because signature-based approaches have lower false negative rates than anomaly-based approaches.

   ii. Scheme *S*, because signature-based approaches work in real time and anomaly-based approaches do not.

   iii. Scheme *A*, because anomaly-based approaches have lower false positive rates than signature-based approaches.

   iv. Scheme *A*, because anomaly-based approaches require less state than signature-based approaches.

   v. It is not clear without additional information which of schemes *S* or *A* would work better for intrusion prevention.

(part (a) is continued on the next page)

3. As your company becomes more successful you grow concerned that attackers will try to evade your detectors. **Circle all** of the following that are correct:

   i. Scheme S is vulnerable to evasion by attackers who can manipulate the order and timing of the packets they send.

   ii. Scheme A is vulnerable to evasion by attackers who can force their traffic to be sent using fragmented packets.

   iii. An attacker can more easily try to exhaust the memory used by scheme S than the memory used by scheme A.

   iv. For analyzing unencrypted HTTP traffic, scheme A is more vulnerable to evasion by attackers that employ URL hex-escape encodings than is scheme S.

   v. Given how the different schemes operate, scheme A is likely better suited for resisting evasion by imposing a canonical form ("normalization") on the traffic it analyzes than is scheme S.

   vi. None of the above.

(b) Your company decides to build a hybrid scheme for detecting malicious URLs. The hybrid scheme works by combining scheme S and scheme A, running both in parallel on the same traffic. The combination could be done in one of two ways. Scheme H_E would generate an alert if for a given network connection *either* scheme S or scheme A generates an alert. Scheme H_B would generate an alert only if *both* scheme S and scheme A generate an alert for the same connection. (Assume that there is only one URL in each network connection.)

Assuming that decisions made by S and A are well-modeled as independent processes, and ignoring any concerns regarding evasion, which of the following statements regarding the hybrid scheme are correct? **Circle all** that apply.

1. H_E will result in a lower false positive rate than scheme H_B.
2. H_E will result in a lower false negative rate than scheme H_B.
3. H_E will be likely to detect a larger number of known attacks than H_B.
4. H_E will be likely to detect a larger number of novel attacks than H_B.
5. None of the above.

(c) If deploying the hybrid scheme in a new environment, and under the same assumptions as in part (c), is one of H_E and H_B clearly better? If yes, explain which would be your choice and why. If not, explain why there is no clear choice.

Problem 5. [Viruses and Worms] (25 points)

(a) Which of the following is true of viruses/worms? **Circle all that apply**.

1. Polymorphic viruses are harder to detect with signature-based techniques than metamorphic viruses.
2. You can prevent metamorphic viruses from infecting your systems by making your disks read-only after bootup.
3. Metamorphic viruses require public-key cryptography.
4. You can write a worm that uses multiple different techniques to spread.
5. None of the above.

(b) Which of the following is true of worms? **Circle all that apply**.

1. Worms were first invented less than 10 years ago, with the appearance of Code Red.
2. Recovering from a worm infection that compromises a Linux administrator account requires rebuilding the system from its original source code.
3. There have been Internet worm outbreaks that infected more than 1,000,000 systems.
4. There have been Internet worm outbreaks that infected more than 250,000 systems in under 5 seconds.
5. For a worm that randomly scans Internet addresses and has a constant contact rate β, the larger the vulnerable population the more quickly the worm spreads.
6. None of the above.

(c) Suppose you have a technology available that will prevent any buffer overflow attack. If you deploy it everywhere, which of the following best describes its effectiveness? **Circle just one.**

1. It would prevent all types of worms from propagating.
2. It would slow down the propagation of all types of worms but not fully prevent the propagation of any.
3. It would prevent the propagation of some types of worms, but not all types.
4. It would not help in preventing worms from propagating, but would slow down the propagation of some types of worms.
5. It would not help in preventing worms from propagating nor in slowing down their propagation.

(d) A random-scanning worm spreads (**circle just one**):

1. with linear speed until it reaches critical mass, after which it propagates at an exponentially increasing rate.
2. exponentially quickly at the beginning, but with the rate decreasing as more and more systems are infected.
3. at a quadratically increasing rate throughout the worm's propagation.
4. at an exponentially increasing rate until all susceptibles are infected.

Problem 6. [Crypto] (32 points)

This problem tests your understanding of how to use cryptographic algorithms. Alice and Bob conduct business extensively over the Internet, and they would like to be able to enter into binding contracts with other parties located around the world. To save the environment, they'd like to do it entirely electronically, over the Internet. Therefore, we'd like some cryptographic way that each party can agree to the terms of the contract. If Alice and Bob are contemplating a contract, Alice should be able to tell whether Bob has agreed to the contract, and similarly Bob should be able to tell whether Alice has agreed to the contract.

Once Alice and Bob have agreed to the terms of the contract, neither should be able to back out. If one party fails to live up to his/her obligations, or there is any dispute, we assume that the injured party will bring a lawsuit. In case of a lawsuit, it should be possible for Alice to convince the judge that Bob agreed to the terms of the contract (by showing the judge the messages she received from Bob). Similarly, it should be possible for Bob to convince the judge that Alice agreed to the contract. It should never be necessary for any party to give their private key to the judge, but it is OK to give the judge any session keys that are relevant.

The scheme should be secure against attackers with the ability to intercept and/or modify packets sent electronically, and with the ability to inject forged packets. There should be no way for an attacker to fool Alice into thinking that Bob has accepted any contract that he did not actually accept, and no way to fool Bob into thinking that Alice has accepted any contract that she did not actually accept. Also, in the event of a lawsuit, there should be no way for Alice to fool the judge into concluding that Bob accepted the contract when he actually didn't (even if Alice colludes with the attacker), and no way for Bob to fool the judge into concluding that Alice accepted the contract when she actually didn't (even if he colludes with the attacker). Let's look at several possible cryptographic protocols for this.

Assumptions: You can assume that Alice has a public key K_A and a matching private key K_A^{-1}; Bob has a public key K_B and private key K_B^{-1}; and the court has a public key K_C and a private key K_C^{-1}. Assume that everyone knows everyone else's public key, with no possibility of spoofed public keys (e.g., Alice knows Bob's public key and the court's public key, and so on). The parties do not share their private key with anyone. Each party has his/her own personal computer, which is not shared, is adequately protected from compromise, and can be assumed free of malware. You may assume that all encryption, MAC, and digital signature algorithms are secure against chosen-plaintext attack and are implemented properly. Let T denote the text of the contract. T will include the identities of the parties and all the terms of the contract. Alice and Bob already know T. There is no need to prevent the attacker from learning T. At least one of the schemes below is OK.

Instructions: For each scheme listed below, circle "OK" if the scheme meets the requirements above, or "NOT" if the scheme is does not meet the requirements above (e.g., it is insecure). Each part is worth 4 points; you will receive 4 points for a correct answer, 2 points if it is left blank, and 0 points for an incorrect answer.

(a) OK or NOT: Alice sends $E_{K_B}(T)$ to Bob. Bob sends $E_{K_A}(T)$ to Alice.

(b) OK or NOT: Alice sends $E_{K_C}(T)$ to Bob. Bob sends $E_{K_C}(T)$ to Alice.

(c) OK or NOT: Alice sends $E_{K_B}(T), E_{K_C}(T)$ to Bob. Bob sends $E_{K_A}(T), E_{K_C}(T)$ to Alice.

(d) OK or NOT: Alice sends $\text{Sign}_{K_A^{-1}}(T)$ to Bob. Bob sends $\text{Sign}_{K_B^{-1}}(T)$ to Alice.

(e) OK or NOT: Alice sends $\text{Sign}_{K_A^{-1}}(H(T))$ to Bob, where H is a cryptographic hash function. Bob sends $\text{Sign}_{K_B^{-1}}(H(T))$ to Alice.

(f) OK or NOT: Alice picks a random session key k and sends $k, E_k(T), \text{Sign}_{K_A^{-1}}(k), \text{Sign}_{K_A^{-1}}(E_k(T))$ to Bob. Bob picks a random session key k' and sends $k', E_{k'}(T), \text{Sign}_{K_B^{-1}}(k'), \text{Sign}_{K_B^{-1}}(E_{k'}(T))$ to Alice.

(g) OK or NOT: Alice picks a random session key k and sends $U, \text{Sign}_{K_A^{-1}}(U)$ to Bob, where $U = (k, E_k(T))$ (U is the concatenation of k and $E_k(T)$). Bob picks a random session key k' and sends $U', \text{Sign}_{K_B^{-1}}(U')$ to Alice, where $U' = (k', E_{k'}(T))$.

(h) OK or NOT: Alice picks a random session key k and sends $\text{MAC}_k(T), E_{K_B}(V)$ to Bob, where $V = (k, \text{Sign}_{K_A^{-1}}(k))$. Bob picks a random session key k' and sends $\text{MAC}_{k'}(T), E_{K_A}(V')$ to Alice, where $V' = (k', \text{Sign}_{K_B^{-1}}(k'))$.

Problem 7. [Password hashing] (6 points)

Suppose you are reviewing email encryption software used by millions of people. You discover it encrypts every message under a 128-bit key K derived as $K = F(P)$, where P is a password chosen and entered by the user. The encrypted message is then broadcast over an insecure network. However, you have not reverse-engineered the algorithm for F yet, and all you know is that F is a deterministic unkeyed function. What can you conclude about the security of this key generation method? **Circle just one** (the best answer).

1. If F is not publicly documented anywhere, this is an excellent design, because it ensures that no one will ever be able to learn the encryption key K.
2. This will be secure as long as F has the property that all 2^{128} possible keys can occur as the output of F (i.e., for every $K \in \{0,1\}^{128}$ there exists P such that $K = F(P)$). F does not need to be secret; it is OK if F is publicly documented.
3. This will be secure as long as F is a cryptographic hash function. F does not need to be secret; it is OK if F is publicly documented. However, it is not sufficient to meet the conditions of answer 2: i.e., it is not sufficient for F to merely have the property that all 2^{128} possible keys can occur as the output of F.
4. This design is problematic no matter how F is chosen, because the key K will most likely not have enough entropy. It will be problematic even if F is a cryptographic hash function.

Problem 8. [Denial of service] (20 points)

`BrowserTest.com` is a useful new web service that helps you compare how a web site looks when viewed with Firefox vs. how it looks when viewed with Internet Explorer.

Here's how it works. If you visit a URL like `http://browsertest.com/?u=http://cnn.com/`, the BrowserTest server starts a Firefox process, loads `http://cnn.com/` in Firefox, and takes a screenshot of the resulting Firefox window. In parallel, it starts Internet Explorer, loads `http://cnn.com/` in Internet Explorer, and takes a screenshot of the resulting Internet Explorer window. When both of these two parallel steps complete, the BrowserTest server responds to the original request with a HTML document containing both screenshots. (Note that the process of handling this HTTP request from the user causes the BrowserTest server to issue two separate HTTP requests to `cnn.com`, one for each browser.) The BrowserTest server is willing to accept any URL you specify; it treats everything after the `?u=` as a URL and loads that URL in each browser. Thus, the BrowserTest site is very general and useful for testing how portable a website is.

Explain how an attacker could mount a serious denial-of-service attack against the BrowserTest server simply by visiting a single carefully chosen URL. Show the malicious URL that can cause so much havoc. You may assume the BrowserTest folks haven't taken any special precautions against denial-of-service.

Problem 9. [Terminology] (20 points)

For each of the numbered concepts given below, list the term that best applies:

Application proxy	Leap-of-faith authentication	Same origin policy
Browser-in-browser	Least privilege	Security perimeter
Bulletproof hosting	Logic bomb	Security-by-obscurity
Click-jacking	Man-in-the-middle	Separation of responsibility
Command injection	Onion router	Side channel
Complete mediation	Opportunistic ack'ing	Stored XSS
CSRF	Ransomware	Third-party cookie
Default deny	Redaction	TOCTTOU vulnerability
Defense in depth	Reference monitor	Trusted path
Directory/path traversal	Reflected XSS	Tunneling
Dongle	Remanence	VPN
Homograph attack	RST injection	Warez
Inband signaling	Rubber hose cryptanalysis	Watermarking

Not all terms are used. No term is used twice.

1. An approach to copy protection based on requiring the user to plug into their computer a specialized hardware device

2. A component in some anonymity systems that works by forwarding traffic that has been multiply encrypted using different keys

3. An alternative to PKI whereby the first time a user connects to a service they simply accept whatever public key the server offers and assume it is likely to be correct

4. A web attack that exploits the generality of the file system of a server hosting web content in order to access files that the operator did not intend to make accessible

5. A web attack based on embedding a script in a URL such that when a web server processes the URL, its reply includes the script within it

6. An attack whereby an adversary inserts themselves into communication between two parties without their knowledge, enabling the adversary to inspect and modify the communication

7. A service that supports keeping Internet servers operational in the presence of complaints and "take-down" demands

8. An attack whereby a TCP receiver spurs the host sending to it to transmit more quickly

9. Overcoming the use of cryptography by threatening the person who holds the secret key rather than attempting to break the cryptographic mechanisms

Application proxy	Leap-of-faith authentication	Same origin policy
Browser-in-browser	Least privilege	Security perimeter
Bulletproof hosting	Logic bomb	Security-by-obscurity
Click-jacking	Man-in-the-middle	Separation of responsibility
Command injection	Onion router	Side channel
Complete mediation	Opportunistic ack'ing	Stored XSS
CSRF	Ransomware	Third-party cookie
Default deny	Redaction	TOCTTOU vulnerability
Defense in depth	Reference monitor	Trusted path
Directory/path traversal	Reflected XSS	Tunneling
Dongle	Remanence	VPN
Homograph attack	RST injection	Warez
Inband signaling	Rubber hose cryptanalysis	Watermarking

10. The problem of data persisting even though the user instructed their system to delete it

11. A web attack based on confusing users regarding what URL their browser displays by employing characters that look the same, such as the digit one ("1") for an an ell ("l")

12. The notion of gaining more robust security by employing multiple mechanisms of different types

13. The notion that one can gain more robust security by ensuring that system components cannot acquire capabilities they do not need for their operation

14. A system component that securely mediates all access to a given object

15. An attack based on confusing the user about which browser window or frame is receiving their mouse events

16. The notion that when enforcing access control policies, one must ensure that every access to every object is checked

17. Communicating control information using the same channel as is used for communication data

18. A web attack based on (a) predicting a URL that when sent to a server instructs the server to take actions on behalf of the user whose browser sends the URL, and (b) causing the victim's browser to send that URL to the server

19. A scheme to enable detection of copying of digital content by robustly embedding a distinct pattern in the content

20. A software flaw that occurs when a program tests for permission to access an object separately from later actually accessing the object