CS 161 Computer Security
Spring 2010 Paxson/Wagner

MT1 Solns

Midterm 1 exam solutions

Please— do not read or discuss these solutions in the exam room while others are still taking the exam.

Problem 1. [True or false] (14 points)

(a) TRUE or [FALSE]: SQL injection vulnerabilities can be avoided by applying the following algorithm to every untrusted input: replace every instance of & with `&`, replace every instance of < with `<`, replace every instance of > with `>`, and replace every instance of " with `&quot;`.

Comment: That might be part of a defense against XSS, but it doesn't help with SQL injection.

(b) [TRUE] or FALSE: The same-origin policy generally allows JavaScript from `berkeley.edu` to read cookies associated with `berkeley.edu`.

(c) TRUE or [FALSE]: The same-origin policy generally allows JavaScript from `berkeley.edu` to read cookies associated with `stanford.edu`.

(d) [TRUE] or FALSE: A benefit of privilege separation is that it can provide an opportunity to reduce the size of the TCB.

(e) TRUE or [FALSE]: If we ensure that the attacker does not have permission to read the value of session cookies stored by the victim's browser, session fixation attacks become impossible.

Comment: Session fixation attacks rely upon the attacker's ability to *write* the value of session cookies; reading is irrelevant.

(f) TRUE or [FALSE]: It is easier to passively eavesdrop on UDP traffic than on TCP traffic.

(g) TRUE or [FALSE]: Ingress filtering refers to carefully escaping meta-characters in URLs.

Comment: Ingress filtering refers to filtering of network packets to avoid forged source addresses.

Problem 2. [Short answer] (22 points)

(a) BankOBits is a local bank that offers its customers access to a number of conveniently located ATMs. Normally, when a customer inserts his/her ATM card into a BankOBits ATM, the ATM will contact the BankOBits central server to validate the ATM card inserted into it and check that the corresponding account has sufficient funds before allowing the user to withdraw money. However, if the server does not respond, the network connection is down, or something else goes wrong with this query, the BankOBits ATM will assume all is well, allow the customer to withdraw up to $300, keep a record of the transaction, and upload that information to the BankOBits server whenever connectivity is restored. As a result of this design decision, a gang of criminals are able to steal from the bank by cutting the network connection on BankOBits ATMs and withdrawing $300 from them using a fake ATM card.

In this story, which security principle was violated? Circle one (the best answer), and briefly explain.

(i) Least privilege. (ii) [Fail-safe defaults.]
(iii) Separation of responsibilities. (iv) Human factors matter.

Justification: The default is to give the customer $300 in case of doubt, which is unsafe because it does not check whether the card is valid or the account has sufficient funds.

(b) David Wagner once heard about a kiosk at one airport that let you access the web, for a fee. To use the kiosk, you had to enter your credit card information at a welcome screen before the kiosk would give you access to a web browser. However, some hacker discovered that if you press F1 to invoke the "help" screen, the Windows help subsystem would pop up a window with generic help information

about the login screen. The help text happened to contain a link to an external web site with more help information, and if you click on that link, the kiosk would open the Internet Explorer web browser to display that web page. At that point, one could change the URL in the Internet Explorer address bar and gain full access to the web, without paying.

In this story, which security principle was violated? Circle one (the best answer), and briefly explain.

(i) Fail-safe defaults. (ii) Separation of responsibilities.
(iii) Complete mediation. (iv) Psychological acceptability.

Justification: The system does not control all ways to obtain access to the web.

(c) The Fortune 500 company FooCorp has an internal web application that its employees can use to fill out travel vouchers. Unfortunately, FooCorp's system administrators have recently discovered that the voucher web application has cross-site request forgery (CSRF) vulnerabilities. FooCorp has a firewall that blocks all inbound connections from the external world to FooCorp's internal network, but allows all web connections initiated from machines on FooCorp's internal network.

Does FooCorp's firewall prevent exploitation of the CSRF vulnerabilities in its travel voucher application? Circle yes or no, then briefly explain (in one line or less).

(i) Yes. (ii) No.

Justification: If an internal employee visits a malicious external website, that website can exploit the CSRF vulnerability. The request to the voucher system will come from the employee's internal machine and therefore will not be blocked (or even seen) by the firewall.

(d) Suppose we are building a web application that asks the user for their email address and stores it in a variable `m`. We want to invoke the shell to send an email message to the email address `m`, like this:

```
void sendemail(char *m) {
  char cmd[1024];
  snprintf(cmd, sizeof(cmd), "mail %s", m);
  f = popen(cmd, "w");
  ...
}
```

However before we invoke `sendemail(m)`, we want to ensure that `m` is safe to use with this code. Which of the following would be the best way to do that? Circle the best (safest) answer.

(a) Check that `m` does not contain any of the following characters: `*|‘()`.

(b) Remove all instances of the following characters from `m`: `*|‘()`.

(c) Check that `m` starts with a letter (a-z or A-Z) and is composed solely of the following characters: `abc...zABC...Z0123...9@+-_..`

(d) None of the above: This code cannot be made safe, no matter what checks you do on `m`.

Explain why briefly (one line or less):
Whitelisting is safer than blacklisting. The whitelist doesn't contain any shell metacharacters, so any string that passes the check will be interpreted by the shell as a single argument to mail and will not affect the flow of execution.

Problem 3. [Printer discovery] (16 points)

A consortium of printer vendors have come up with a great new protocol to help users automatically discover the set of printers on their local network. In this protocol, when the user wants to print something, the user's computer automatically broadcasts a Printer Discovery packet. A Printer Discovery packet is a UDP packet whose destination address is the broadcast address, and whose source and destination port is 56184. Because this is a broadcast packet, every host on the local network will receive it.

Printers constantly listen for Printer Discovery packets. Any time that they receive one, they immediately respond with a Printer Announcement packet. A Printer Announcement packet is a UDP packet whose destination address is the broadcast address, and whose source and destination port is 56185; its payload identifies the name of the printer, the printer's IP address, and any special options supported by the printer (e.g., 2-sided printing, color printing). The Printer Announcement packet is broadcast to the entire network, so that other hosts on the local network can also learn about this printer.

Whenever a machine receives a Printer Announcement packet, it checks that the source address of the packet matches the printer's IP address found in the payload. In case of a mismatch, it ignores the packet. Otherwise, it accepts the packet and adds this printer to its list of known printers. If the machine's list of known printers already contains a printer with the same name, the machine overwrites the previous entry in its list with the information found in the newly received packet.

Vicky the Victim is about to connect her laptop to a local switched Ethernet network. Her laptop will use this printer discovery protocol to look for a printer, and then Vicky will connect to one of the printers found in this way and send it a sensitive corporate document to be printed. Meanwhile, Attila the Attacker's computer is attached to this same network. Attila has the ability to inject packets onto this network and to receive all broadcast packets, but he cannot eavesdrop on other traffic. The printers are in locked rooms that Attila does not have access to, and Attila has not been able to hack or access any of the machines or printers attached to this network, so his only hope is to attack the printer discovery protocol.

(a) Can Attila arrange to learn the contents of Vicky's document, without physically accessing any of the printers? Circle either "yes" or "no", then briefly justify your answer. If you circle "yes", describe the attack; if you circle "no", explain why this kind of attack is not possible.

(i) Yes. (ii) No.

Justification: Attila can observe Vicky's Printer Discovery packet and the real printers' Printer Announcement packets, then (before Vicky prints the document) broadcast Printer Announcement packets containing Attila's IP address but the name of the other printers. When Vicky prints her document, she will send it to Attila, and Attila can see the contents of the document. Attila can then optionally forward the document on to the printer so Vicky doesn't notice anything amiss.

(b) Can Attila modify what is printed on the printer? In other words, Attila wants to *replace* Vicky's chosen document with something else Attila has chosen, hopefully without Vicky noticing. It's not acceptable if Vicky's original document gets printed in addition to Attila's replacement, because then Vicky might notice and get suspicious; Attila is only interested in an attack that causes his document to be printed *instead of* Vicky's. Can Attila mount such an attack, without physically accessing any of the printer? Circle either "yes" or "no", then briefly justify your answer. If you circle "yes", describe the attack; if you circle "no", explain why this kind of attack is not possible.

(i) Yes. (ii) No.

Justification: Do the same as in (a), except modify the document before forwarding it on to the printer.

Problem 4. [DDoS] (19 points)

This question asks you to consider a (hypothetical) anti-spam company called Turquoise Security Inc. Turquoise Security uses a vigilante approach to fighting spam: when one of Turquoise Security's users identifies an email they've received as spam, Turquoise Security's servers automatically visits all the web-sites advertised in the spam message and leaves generic complaints on those websites. Turquoise Security operates on the assumption that as their user base grows, the flow of complaints from hundreds of thousands of computers will apply enough pressure on spammers and their clients to convince them to stop spamming.

Yesterday, Turquoise Security's web site came under a massive DDoS attack using a variety of techniques. The attackers are using DNS amplification: the attackers identified several third-party DNS servers that will respond to any DNS query, and are sending many spoofed DNS queries to those DNS servers with a forged source address. In particular, each query is sent in a spoofed UDP packet, where the source address on each of these DNS queries is forged to be the IP address of Turquoise Security's web server. Also, each query has been chosen so that it will trigger a response that is much larger than the query itself, amplifying the effect of the attack. This attack has overloaded Turquoise Security's web server with huge amounts of traffic.

(a) Consider the packets that Turquoise Security's web server received as a result of this DNS amplification attack. For each of the following fields in the IP header, state whether you expect that field to be the same for all of these packets or to differ from packet to packet (circle one choice). If you select "same", describe the value of that field, in the space to the right. If you select "differs", briefly justify your answer, in the space to the right.

Source address:	(i) same:	
	[(ii) differs:]	varies among addresses of third-party DNS servers
Destination address:	[(i) same:]	address of Turquoise web server
	(ii) differs:	
Source port:	[(i) same:]	port 53 (the DNS port)
	(ii) differs:	
Destination port:	(i) same:	
	[(ii) differs:]	DNS queries likely use source port randomization

(b) Suppose that Turquoise Security's ISP, the company who provides Turquoise Security with their Internet connection, would like to help Turquoise Security survive this attack. Name one thing that their ISP could do in response to this attack, to relieve the load on Turquoise Security's servers for now.

Answer: They could block all incoming UDP packets, or all incoming UDP packets with source port 53 (assuming the Turquoise web server does not perform any DNS queries itself).

Comment: There are probably other reasonable answers.

(c) Today, DNS servers accept queries via the UDP protocol. But imagine that DNS had been designed differently, so that DNS used only TCP (not UDP) and DNS servers accepted queries only via TCP (ignoring all UDP packets). Would this make the DNS amplification attack described above easier, harder, or have no effect? Circle one answer, then briefly explain your answer (in one line or less).

(i) The attack would be easier. (ii) No effect. [(iii) The attack would be harder.]

Justification: The attackers would have to guess TCP Initial Sequence Numbers to complete the three-way handshake. These days TCP ISNs are usually random unguessable 32-bit numbers, so guessing them is hard.

Comment: It's true that if the attackers sent SYN packets to the third-party DNS servers with the source address forged to be that of Turquoise Security's web server, those DNS servers would respond with a SYN|ACK packet to Turquoise's web server. However, this would not provide any amplification, so it's not a DNS amplification attack (and it's not clear it has any benefit over just sending packets directly to Turquoise with a spoofed source address). Also, such an attack is not specific to DNS—it's an attack that can be mounted against any server that is listening on any known TCP port.

(d) Setting aside the current DDoS attack on Turquoise Security, how could the Turquoise Security service itself be used to mount a DoS attack on others?

Answer 1: An attacker could send millions of spam messages, containing links to a target web site; when the recipients mark those as spam, Turquoise will mount a DoS attack against the target.

Answer 2: An attacker could send an email containing thousands of links to the same web site to a conspirator who is a Turquoise customer, and ask the conspirator to mark that email as spam. Then Turquoise servers will send one complaint per link in the email, providing an amplification effect.

Problem 5. [Firewalls and NATs] (14 points)

(a) Can a stateless firewall (such as a packet filter) enforce the following policy?

Policy: Block TCP connection initiation requests from any external host to any internal host. Allow TCP connection initation requests from any internal host to any external host, and also allow returning traffic on these connections initiated by internal hosts.

You may assume that the internal hosts (those on the inside of the firewall) all have IP addresses of the form 128.32.153.x, where the x can be anything in the range 0–255, and no external host has an IP address of this form. You may assume that the TCP/IP stack on every internal host operates correctly.

Circle "yes" or "no", depending on whether you think a stateless firewall (such as a packet filter) can enforce the policy above or not, then briefly explain your answer (in one line or less).

(i) Yes, it can enforce the policy. (ii) No, it cannot.

Justification: Block any inbound packet that does not have the ACK bit set. Allow everything else.

(b) Name one security benefit that NAT provides.

Answer 1: A NAT prevents external hosts from initiating connections to internal hosts.

Answer 2: A NAT prevents external hosts from scanning of the internal network.

Problem 6. [Secure coding] (15 points)

Consider the following C code:

```
/* Information about the current CD. */
struct cd {
  int numtracks; /* The number of tracks on this disc. */
  int tracklen[16]; /* The length of each track on the disc, in seconds. */
  void (*notify)(struct cd *); /* Call this whenever the CD info changes. */
};
```

```
struct cd *curcd = makestructcd();

/* Update the length of track number 'track'. */
void update_cdinfo(int track, int newtracklen) {
  if (track > 16)
    return;
  curcd->tracklen[track] = newtracklen;
  (curcd->notify)(curcd);
}
```

(Don't worry about `makestructcd()`; it just allocates and initializes a `struct cd`.) Assume the adversary can arrange for `update_cdinfo()` to be called with whatever values of `track` and `newtracklen` he likes (those values may have been read directly off the CD, for instance). Answer the following questions about this code, *concisely*:

(a) What is the security vulnerability in this code?

Answer 1: Buffer overrun (or array out-of-bounds error): if `track=16`, then this writes one past the end of the `curcd->tracklen` array.

Answer 2: Buffer overrun (or array out-of-bounds error): if `track` is negative, the array dereference `curcd->tracklen[track]` writes outside the bounds of the `curcd->tracklen` array (it writes before the start of the array).

Either answer is acceptable.

(b) How could an attacker exploit this vulnerability to trigger the execution of malicious code? Describe how the attacker should choose the values of `track` and `newtracklen`.

Answer 1: Set `track=16` and make `newtracklen` the address of malicious code loaded somewhere in the address space of this program. This will overwrite `curcd->notify` with a pointer to malicious code.

Answer 2: Set `track` to some large negative number chosen so that `curcd->tracklen[track]` references, e.g., a return address stored on the stack somewhere, and make `newtracklen` the address of malicious code loaded somewhere in the address space of this program. (There are many possible variations on this answer.)