Solutions to Practice Problems

Computer Security

Principles and Practice

Third Edition

William Stallings

TABLE OF CONTENTS

CHAPTER 1 OVERVIEW

1.1 All of these activities could create the right conditions to threaten the network.

- **a.** The regular daily courier is familiar to employees, so they may not notice anything is wrong should that person walk into the server room.
- **b.** Even with good severance packages and benefits, employees who lost their jobs due to downsizing may be disgruntled.
- **c.** An employee's traveling to another location may not create a threat, but if the employee has a laptop computer that contains private information or the Web browser has saved passwords, then if the laptop is stolen, a hacker has gained valuable information.
- **d.** If the sprinkler system went off, it could damage the company's servers and other computing equipment.

CHAPTER 2 CRYPTOGRAPHIC TOOLS

2.1 A function h: X $\rightarrow$Y is collision resistant if it is computationally infeasible to find two different points x_1, $x_2 \in$ X such that h(x_1) = h(x_2)

2.2 We could use triple-DES as a pseudorandom function in order to derive separate keys for the MAC and encryption schemes. Specifically, let *K* be the 168-bit key for triple-DES. Then, compute the MAC-key to be the first 168-bits of (3DES(*K*, 0), 3DES (*K*, 1), 3DES(*K*, 2)) (note that there are 192 bits in the output so just take the first 168). Furthermore, compute the encryption key to be the first 168-bits of (3DES(*K*, 3), 3DES (*K*, 4), 3DES(*K*, 5)). From here on, you can use CBC encryption and a CBC-MAC. (The processor will also need to obtain a random IV for the CBC encryption. We can assume that it has direct access to fresh randomness.) The proof of security works by first replacing the 3DES algorithm with a truly random function. In this case, the derived encryption and MAC keys are uniformly and independently distributed; therefore the MAC and encryption are both secure. The security when using 3DES to derive the keys therefore relies on the assumption that 3DES is a pseudorandom function.

2.3 **a.** Let the `toAccount` of *m* be block M_2 and the adversary's account block M_2'. The adversary then replaces C_2 by

$$C_2' = C_2 \oplus M_2 \oplus M_2' = (M_2 \oplus K_2) \oplus M_2 \oplus M_2') = K_2 \oplus M_2',$$

so when the message $C_1C_2'C_3$ is decrypted it will show the adversary's account as `toAccount`.

b. The message could be authenticated by adding MAC(*KM*, $M_1M_2M_3$) after the message before encryption. If the receiver checks the MAC before accepting the message, the attack will be discovered.

c. We need to use the key to generate a key stream for XORing with the plaintext. One way to do this is to let K_i = H($K \| i$). Then messages are split into blocks of the size of hash values: m = $M_1M_2...M_n$ and c =$C_1C_2...C_n$ where $C_j = M_j \oplus K_j$. Decryption is the same as encryption (so the receiver generates the same key stream, using the common key K).

CHAPTER 3 USER AUTHENTICATION

3.1 Yes, an eavesdropping attacker can do off-line password guessing. The attacker has R and X, and does the following:

```
repeat
{
 choose candidate password cpw;
 compute cJ from cpw;
 compute cX ← encrypt(R) with key J
}
until cX = X;
//cpw = pw
```

3.2 Here we give two solutions.
Solution 1 (detailed handshake at end):
After obtaining L', principal A initiates DH to establish a session key S, where the DH messages are encrypted by L'. Encrypting the DH messages by L' ensures that the attacker cannot hijack the data exchange phase (otherwise the attacker can spoof A in the DH and thus have the session key established between itself and B).
Solution 2:
After obtaining L', principal A initiates DH (with unencrypted messages) to establish a session key S. Then B sends a challenge, say R, encrypted with S, to which A responds with a message M applying S and L to R, for example:
- [encrypt(encrypt(R) with L) with S]
- [encrypt(hash(R | L) with S]
- [encrypt(R+1) with S]

3.3 **a.** 100,000
b. $1000 - (0.09 \times 1000) = 910$
c. $(0.09 \times 1000) = 90$.

Chapter 4 Access Control

4.1 Various problems in this scenario are as follows:

A. A rogue process could modify the address space of other processes or of the kernel.

B. A rogue process could disable interrupts and avoid getting re-scheduled, so that the rogue hogs all the CPU and other process don't get a chance to run.

C. There's no distinction between privileged versus unprivileged instructions, so a rogue process could send I/O commands directly to attach peripherals. For instance, a rogue process could trash the hard disk or snoop on network packets.

CHAPTER 6 MALICIOUS SOFTWARE

6.1 **Bot Net**: A peer-to-peer network of compromised hosts controlled by an owner.
Easter Egg: Pretty unspecified code hidden in a program by developers.
Logic Bomb: Code will delete files or crash a system at a certain time

6.2 The major problem is Buffer overrun. If the input string contains a newline character, then this will write past the end of the input buffer. In the worst case, the size of the string might double. For instance, if the caller allocates a buffer on the stack that is just large enough to hold the string, and passes it to escape(), then a stack-smashing attack would be possible.
Another problem is that memcpy() invokes undefined behavior when invoked on overlapping memory regions.

CHAPTER 7 DENIAL OF SERVICE

7.1 No it is not a good solution because of following reasons:

A. It's too easy to break: with more than 100 zombies, you can flood the victim's network link without any zombie consuming more than 1% of traffic.

B. It's too easy to evade detection with forged source addresses. You just use a different (forged) IP address on every packet.

C. It doesn't protect against attacks that overwhelm resources at the end host (e.g., CPU, memory) without filling the network pipe.

D. Attackers could exploit this to cause collateral damage to innocent third parties. If CNN is using this, an attacker could prevent Joe from being able to reach CNN by sending a large number of packets whose IP address has been forged to look like they came from Joe.

Chapter 8 Intrusion Detection

8.1 Some of the ways by which hackers compromise computers without code breaking are as follows.
- Key Catcher (hw or sw)
- Via email that has an executable file for an attachment.
- A boot CD that has its own Operating System

8.2 A Null session problem is commonly a problem that exists on many Systems especially Microsoft based systems where the system allows a person or other system to connect to it without use of username and/or password such as Shares.

8.3 There are many ways to achieve this.
By alerting administrators via email/pager/phone
By changing firewall configurations to
- increase logging of suspect sessions
- block certain sensitive areas inside or
- block offending areas outside
- throttle offending or suspect traffic
- bring down the Internet connection

There are customer filters that can be configured for signatures that an IDS system looks for and there are standard "out of the box" attack signatures that are known attacks. If the IDS is not configured properly it may send what are known as "false positives" or alerts to an over abundance of traffic, therefore overwhelming people with alerts. While such alerts are active responses, they (as stated above) may become overwhelming.

8.4 **a.** Adversary
- Sees the messages: N, R, X
- Computes Hash(R)
- Computes X XOR Hash(R) = Hash(P)

Later on:
- Adversary Requests Login, submits N.
- Machine generates random number R'.
- Adversary computes Hash(R').
- Adversary computes y = Hash(P) XOR Hash(R').
- Adversary submits y, and logs in as user.

b. To strengthen, simply require the protocol to compute
Hash(R XOR P) instead of Hash(R) XOR Hash(P)

CHAPTER 9 FIREWALLS

9.1 No. The phrase “Make money fast” might be spread across multiple packets (e.g., “Make money” in the first packet, “fast” in the second). A stateless packet filter cannot remember any state from prior packets, so cannot usefully block such email.

9.2 Here are some of the threats and their brief explanation.
(1) Attacks against open ports, such as buffer overrun attacks against unblocked services;
(2) Malicious code or attacks carried in email or web traffic (many firewalls do not scan or examine email and web payloads);
(3) Attacks on the firewall itself (e.g., trying to penetrate the firewall code by exploiting a buffer overflow in the firewall’s packet parsing code);
(4) Internal attacks by malicious insiders;
(5) Attacks from compromised internal machines against other internal machines (e.g., a laptop becomes infected with a worm, which tries to infect other inside hosts)—applies to perimeter firewalls;
(6) Attacks from compromised machines that have a VPN or other tunnel through the firewall—applies to perimeter firewalls;
(7) Denial of service attacks against the network link or the firewall itself.

9.3 The most important component is a web proxy server that performs filtering of URLs. All internal machines must route their requests through it. To enforce this, a packet filter should be used to block all outbound TCP traffic from any machine other than the proxy. By itself, that machine could even be on the inside of a packet filter. However, as noted, it does need to do DNS queries. If no other UDP services run on that machine, outbound UDP access to port 53 from it could be enabled, with all inbound UDP responses permitted. Alternatively, a stateful packet filter could be used, regardless of what other UDP services were enabled.

If there was a threat of unauthorized use of UDP by other inside machines — not mentioned as part of the problem— a DNS server would have to live in a DMZ. Only the Web server — which could live in the DMZ, but doesn’t have to — would be allowed to send packets to it. DNS filtering alone won’t work, since people could browse many sites using just the IP address in the URL.

9.4 Firewall with “Default Deny” policy is better than one with “default allow” policy because inadvertently omitting an item from the list causes a loss of service but does not compromise security.

CHAPTER 10 BUFFER OVERFLOW

10.1 There are two problems in this code snippet:

1. Buffer overrun. If the input string contains a newline character, then this will write past the end of the input buffer. In the worst case, the size of the string might double. For instance, if the caller allocates a buffer on the stack that is just large enough to hold the string, and passes it to escape(), then a stack-smashing attack would be possible.
2. Another problem is that memcpy() invokes undefined behavior when invoked on overlapping memory regions.

10.2 In the sscanf(),the width of string for %s is not specified and it causes the buffer overflow problem. The corrected version is as follows:

```
void main(int argc, char**argv)
{
      charbuf[256];
      sscanf(argv[0],"%255s", buf);
}
```

10.3 The function sprint() is causing the buffer overflow problem. Use snprint() instead of sprint(). The corrected code snippet is as follows:

```
int main(int argc, char *argv[])
{
      char filename[MAXPATHLEN];
      if (argc==1)
            snprintf(filename, "/tmp/xxx%d", getpid());
      else
            snprintf(filename, "/tmp/%s", argv[1]);
if ((fd = open(filename, O_RDWR|O_EXCL|O_CREAT, 0644) )< 0)
{
      perror(filename);
            exit(1);
}
}
```

10.4 The function strcpy() is causing the buffer overflow problem. Use strncpy() instead of strcpy(). The corrected code snippet is as follows:

```
int check_authentication(char *password)
{
      int auth_flag = 0;
      char password_buffer[16];
      strncpy(password_buffer, password);
```

```
     if(strcmp(password_buffer, "brillig") == 0)
          auth_flag = 1;
     if(strcmp(password_buffer, "outgrabe") == 0)
          auth_flag = 1;
     return auth_flag;
}

int main(int argc, char *argv[])
{
     if(argc < 2)
     {
          printf("Usage: %s <password>\n", argv[0]);
          exit(0);
     }
     if(check_authentication(argv[1]))
     {
          printf("\n-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
          printf(" Access Granted.\n");
          printf("-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
     }
     else
     {
          printf("\nAccess Denied.\n");
     }
}
```

CHAPTER 11 OTHER SOFTWARE SECURITY ISSUES

11.1 a. The attacker can exploit lpr by passing in a symbolic link to a file to print. The attacker initially points the link to a file that belongs to the attacker. After the access system call is run, the attacker then switches the symbolic link to point at a file that the attacker normally does not have privileges to access.

b. The difficulty with the attack is that the symbolic link must be changed after the access call starts, but before the open begins. If the access call causes the kernel to make I/O requests to the disk, it will cause the process to sleep until the disk requests complete. This means that the attacker must make sure that the file is not in the file cache in memory. During this time, the attacker can switch the directory links. The process can be made to sleep for longer by providing an initial file with a very deep directory structure that will require multiple disk accesses to traverse.

CHAPTER 20 SYMMETRIC ENCRYPTION AND MESSAGE CONFIDENTIALITY

20.1 The key space has 2^{40} elements, so brute force would take 2^{20} seconds, which is about 12 days. This would be practical if the message revealed the location of enemy missiles in a cold-war situation. It would be impractical if the message's useful life was very short, for example if it was a few frames in a pay-per-view sports video. Doubling the key size would make the brute force decryption time 2^{60} seconds, which is about 3.8×10^{16} years. There is no scenario in which this would be practical.

20.2 **a.** Let c = DESX(K, m). We first XOR both sides with k_2, which gives $c \oplus k_2$= DES(k_1, (m $\oplus$ k_2)). Next, we apply DES decryption, to get $\text{DES}^{-1}(k_1, (c \oplus k_2)) = \text{DES}^{-1}(k_1, [\text{DES}(k_1, (m \oplus k_2)]) = m \oplus k_2$. Finally, we again XOR both sides with $k_2$to get the final result

$$m = \text{DES}^{-1}(k_1, (c \oplus k_2)) \oplus k_2$$

b. DESX' can be attacked using a meet-in-the-middle attack. We assume that the adversary has a few plaintext/ciphertext pairs (m, c). He can then do a brute force attack on the DES part, i.e. compute x = DES(k_1, m) for all possible keys k_1, and store the resulting pairs (x, k_1) in a dictionary. Then he goes through all possible $k_2$values, computes $c \oplus k_2$ and looks it up in the dictionary. When found, he has a potential key pair (k_1, k_2). The complexity of this attack is 2^{64}, which shows that the cipher does not provide 120 bits of security.

Alternatively, with two plaintext/ciphertext pairs (m_1, c_1) and (m_2, c_2), one notes that $c_1 \oplus c_2 = \text{DES}(k_1, m_1) \oplus \text{DES}(k_1, m_2)$, which makes it possible to do a brute force attack with only twice the cost of an attack against DES.

20.3 The state in AES is a 4×4 matrix with entries in the field of 256 elements. The shift row layer shifts each row to the right a certain amount, wrapping the entries around. More precisely, the first row is not shifted, the second row is shifted by one, the third row is shifted by two, and the fourth row is shifted by three.

The Byte Substitution layer can be viewed as a lookup table. Each matrix entry, represented by an 8-bit byte, is broken into two pieces which index the rows and columns of a 16×16 lookup matrix. The byte is replaced by the corresponding entry in the table, which is another 8-bit byte. The Byte Substitution layer is applied entry by entry to the state, with all entries treated in the same way. The Row Shift layer simply moves the bytes around. Thus it doesn't matter in which order we apply these layers: shifting and substituting is the same as first substituting then shifting.

20.4 DES takes a 8-octet (64-bit) plaintext block and yields a 8-octet cipher block. CBC requires a 8-octet initialization vector (IV) to be sent along with the cipher blocks. So X now sends 64 octets of cipher blocks plus 8 octets of IV, for a total of 72 octets.

CHAPTER 21 PUBLIC-KEY CRYPTOGRAPHY AND MESSAGE AUTHENTICATION

21.1 Counter mode is unusable, since here decryption is the same as encryption and anyone can encrypt messages, using the receiver's public key. Hence anyone intercepting a ciphertext can decrypt it. CBC mode does not have this disadvantage, since it uses the decryption function.

21.2 The idea is to split the message to be hashed into blocks of a specified size of, say, *k* bits. Typically messages are padded, using a specified padding scheme, so that the padded message consists of an integral number of full blocks. If the hash function produces digests of size *n* bits, we use a compression function

i. $g : \{0, 1\}^n \text{ X } \{0, 1\}^k \rightarrow \{0, 1\}^n$.

We further need a specified initialization vector $IV \in \{0, 1\}^n$. Then we iterate the compression function:

ii. $D_0 = IV$

iii. $D_i = g(D_{i-1}, M_i)$

The result of the hash function is the last D_i

CHAPTER 22 INTERNET SECURITY PROTOCOLS AND STANDARDS

22.1 We can do so by TLS or SSL. Transport Layer Security (TLS) and Secure Sockets Layer (SSL), are cryptographic protocols that provide security and data integrity for communications over networks such as the Internet. TLS and SSL encrypt the segments of network connections at the Transport Layer end-to-end.

22.2 Proof of submission is a proof that a message is transmitted to electronic mail system. Non-repudiation enables a recipient of a message to prove in a court of law that it was sent by a particular sender. Non-repudiation in e-commerce prevents initiators of a transaction later claiming that the recipient or some party made the transaction in their name.

22.3 There is no definite answers to that but one can form an opinion by first considering what IPsec is and what it does. IPsec refers to a set of standards developed by the Internet Engineering Task Force (IETF). IPsec solves two problems that have plagued the IP protocol suite for years: host-to-host authentication (which will let hosts know that they're talking to the hosts they think they are) and encryption (which will prevent attackers from being able to watch the traffic going between machines).

Neither of these problems is what firewalls were created to solve. Although firewalls can help to mitigate some of the risks present on an Internet without authentication or encryption, there are really two classes of problems here: integrity and privacy of the information flowing between hosts and the limits placed on what kinds of connectivity is allowed between different networks. IPsec addresses the former class and firewalls the latter. Note however from Chapter 19 that IPsec does provide a limited type of firewall capability in that it allows the user to specify traffic processing rules for a variety of classes of traffic. This is a firewall type of service, but IPsec only provides a limited flexibility in this area.

22.4 IPsec would reject the packets and would not pass them to TCP. In SSL, such packets could cause the session to break.

22.5 a. The initial scheme is not any more secure. An eavesdropping attacker need only replay the hash, she does not need to learn the actual password.

b. The scheme can be made more secure with the use of a nonce. A unique nonce is sent with the web page. The JavaScript hashes the nonce along with the user's password and sends this back. Now, an eavesdropper cannot replay the hash because the nonce is different each time.

Chapter 23 Internet Authentication Applications

23.1 **Attack #1:** Wait for Linus to post some other message on his web site. Copy the name, date, and signature, but modify the contents of the message. The viewer will still receive a valid signature and be fooled.
Countermeasure #1: The contents of the web page should also be included in the input to the signature.
Attack #2: When the viewer downloads the public key file, corrupt the response (e.g., send a spoofed response packet) so that it contains a listing for Linus with a public key that is not his. This corruption is possible, since the pubkeys.html is downloaded over insecure HTTP. The attacker can generate his own key pair and list Linus's name next to the attacker's public key. Then, the attacker can create a web page that is validly signed using this key pair, fooling readers.
Countermeasure #2: Secure distribution of pubkeys.html. For instance, it might be distributed over SSL. Or, it might be signed with Kachoo!'s private key, and a copy of Kachoo!'s public key might be embedded in every web browser so that the browser can check that this page has not been corrupted.

CHAPTER 24 WIRELESS NETWORK SECURITY

24.1 It could be done via evil twin. In security, an evil twin is a home-made wireless access point that masquerades as a legitimate hot spot to gather personal or corporate information without the end-user's knowledge. It's fairly easy for an attacker to create an evil twin, simply by using a mobile Internet device such as a laptop or a Smartphone and some readily-available software. The attacker positions himself in the vicinity of a legitimate Wi-Fi access point and lets his Internet device discover what name (SSID) and radio frequency the legitimate access point uses. He then sends out his own radio signal, using the same name. To the end-user, the evil twin looks like a hot spot with a very strong signal; that's because the attacker has not only used the same network name and settings as the "good twin" he is impersonating, he has also physically positioned himself near the end-user so that his signal is likely to be the strongest within range. If the end-user is tempted by the strong signal and connects manually to the evil twin to access the Internet, or if the end-user's computer automatically chooses that connection because it is running in promiscuous mode, the evil twin becomes the end-user's Internet access point, giving the attacker the ability to intercept sensitive data such as passwords or credit card information.

24.2 At the **physical layer**, a device may generate random RF noise, making the medium appear constantly busy.
At the **datalink (MAC) layer**, a device may generate random packets, which violates the ‘‘polite’’ RTS/CTS sequence.
At the **network layer**, any device may masquerade as an access-point and gather and drop packets.
At the **application layer**, an application may naively perform a large file transfer, dominating the medium.

Chapter 27 Trusted Computing

27.1 **a.** TRUE. ACLs are just one representation of an access control matrix—namely, the rows of the matrix.
b. FALSE. In a mandatory access control system, anyone can write to a file marked classified, while only some users (i.e., those who are not cleared for classified data) can write to unclassified files.