SOLUTIONS MANUAL

COMPUTER SECURITY
FOURTH EDITION
GLOBAL EDITION

CHAPTERS 1–12

WILLIAM STALLINGS
LAWRIE BROWN

NOTICE

This manual contains solutions to the review questions and homework problems in *Computer Security, Fourth Edition, Global Edition*. If you spot an error in a solution or in the wording of a problem, I would greatly appreciate it if you would forward the information via email to wllmst@me.net. An errata sheet for this manual, if needed, is available at http://www.box.net/shared/ds8lygu0tjljokf98k85 . File name is S-CompSec4e-mmyy.

TABLE OF CONTENTS

CHAPTER 1 OVERVIEW

ANSWERS TO QUESTIONS

1.1 Confidentiality, Integrity and Availability are three key objectives that form the heart of computer security. These three are often referred to as the CIA triad.

1.2 Data integrity assures that information and programs are changed only in a specified and authorized manner whereas system integrity assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

1.3
i. Unauthorized disclosure: caused by exposure, interception, inference, intrusion
ii. Deception: caused by masquerade, falsification, repudiation
iii. Disruption: caused by incapacitation, corruption, obstruction
iv. Usurpation: caused by misappropriation, misuse

1.4 **Authentication:** The assurance that the communicating entity is the one that it claims to be.
Access control: The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).
Data confidentiality: The protection of data from unauthorized disclosure.
Data integrity: The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
Nonrepudiation: Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
Availability service: The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system (i.e., a system is available if it provides services according to the system design whenever users request them).

1.5 A security policy is a formal statement of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources. Its implementation should cover the following actions: (i) prevention, (ii) detection, (iii) response, and (iv) recovery.

1.6 Network attack surface refers to vulnerabilities over an enterprise network, wide area network or the Internet whereas Software attack surface refers to vulnerabilities in application, utility or operating system code.

ANSWERS TO PROBLEMS

1.1 Apart from the card and USN, if the student needs to enter a pass key to access the information, then the system must keep the pass key confidential, both in the host system and during transmission for a transaction. It must protect the integrity of student records. Availability of the host system is important for maintaining the reputation of the Institution. The availability of SIS machines is of less concern.

1.2 The system has high requirements for integrity on individual data packet, as lasting damage can incur by occasionally losing a data packet. The integrity of routing algorithm and routing tables is also critical. Without these, the routing function would be defeated. A network routing system must also preserve the confidentiality of individual data packets, preventing one from accessing the contents of another.

1.3 **a.** The system will have to assure confidentiality if it is being used to publish corporate proprietary material.
b. The system will have to assure integrity if it is being used to laws or regulations.
c. The system will have to assure availability if it is being used to publish a daily paper.

1.4 **a.** An organization managing public information on its web server determines that there is no potential impact from a loss of confidentiality (i.e., confidentiality requirements are not applicable), a moderate potential impact from a loss of integrity, and a moderate potential impact from a loss of availability.
b. A law enforcement organization managing extremely sensitive investigative information determines that the potential impact from a loss of confidentiality is high, the potential impact from a loss of integrity is moderate, and the potential impact from a loss of availability is moderate.

c. A financial organization managing routine administrative information (not privacy-related information) determines that the potential impact from a loss of confidentiality is low, the potential impact from a loss of integrity is low, and the potential impact from a loss of availability is low.

d. The management within the contracting organization determines that: (i) for the sensitive contract information, the potential impact from a loss of confidentiality is moderate, the potential impact from a loss of integrity is moderate, and the potential impact from a loss of availability is low; and (ii) for the routine administrative information (non-privacy-related information), the potential impact from a loss of confidentiality is low, the potential impact from a loss of integrity is low, and the potential impact from a loss of availability is low.

e. The management at the power plant determines that: (i) for the sensor data being acquired by the SCADA system, there is no potential impact from a loss of confidentiality, a high potential impact from a loss of integrity, and a high potential impact from a loss of availability; and (ii) for the administrative information being processed by the system, there is a low potential impact from a loss of confidentiality, a low potential impact from a loss of integrity, and a low potential impact from a loss of availability. Examples from FIPS 199.

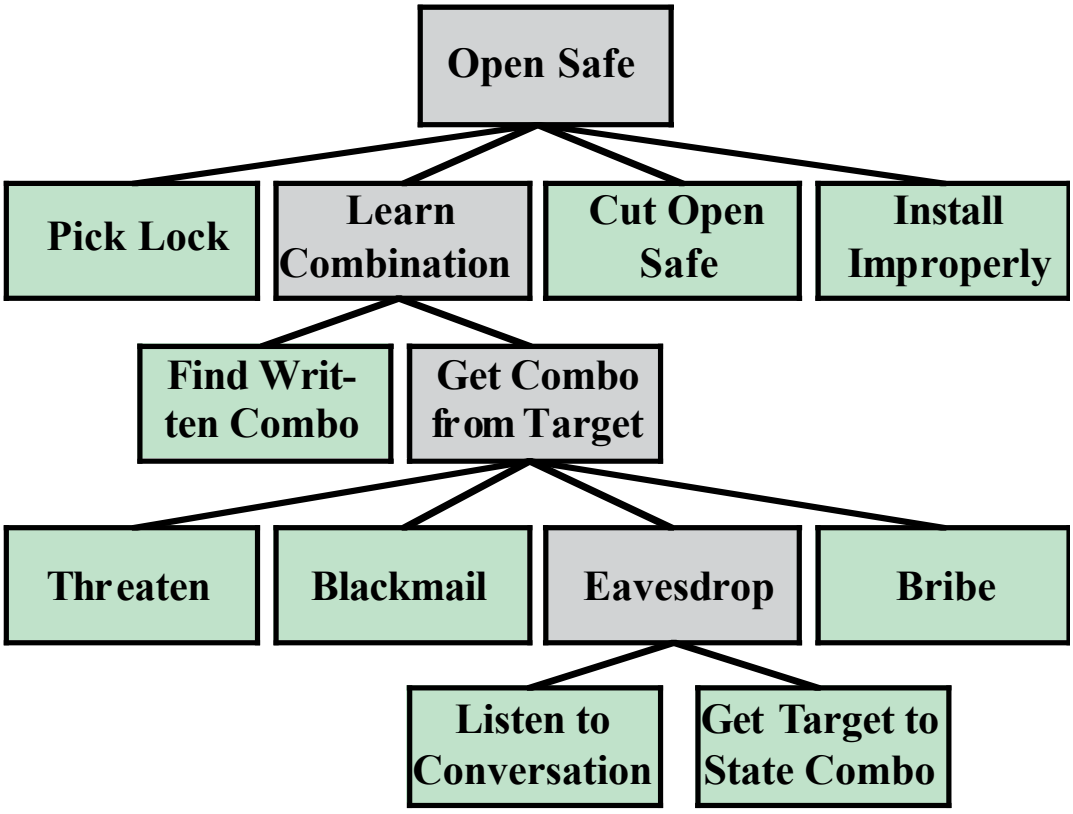
1.5 a. At first glance, this code looks fine, but what happens if IsAccessAllowed fails? For example, what happens if the system runs out of memory, or object handles, when this function is called? The user can execute the privileged task because the function might return an error such as ERROR NOT ENOUGH MEMORY.

b.

```
DWORD dwRet = IsAccessAllowed(...);
if (dwRet == NO_ERROR) {
    // Secure check OK.
    // Perform task.
} else {
    // Security check failed.
    // Inform user that access is denied.
}
```

In this case, if the call to IsAccessAllowed fails for any reason, the user is denied access to the privileged operation.

1.6



1.7 We present the tree in text form; call the company X:

Survivability Compromise: Disclosure of X proprietary secrets

OR 1. Physically scavenge discarded items from X
- OR 1. Inspect dumpster content on-site
- 2. Inspect refuse after removal from site

2. Monitor emanations from X machines
- AND 1. Survey physical perimeter to determine optimal monitoring position
- 2. Acquire necessary monitoring equipment
- 3. Setup monitoring site
- 4. Monitor emanations from site

3. Recruit help of trusted X insider
- OR 1. Plant spy as trusted insider
- 2. Use existing trusted insider

4. Physically access X networks or machines
- OR 1. Get physical, on-site access to Intranet
- 2. Get physical access to external machines

5. Attack X intranet using its connections with Internet
- OR 1. Monitor communications over Internet for leakage
- 2. Get trusted process to send sensitive information to attacker over Internet
- 3. Gain privileged access to Web server

6. Attack X intranet using its connections with public telephone network (PTN)
- OR 1. Monitor communications over PTN for leakage of sensitive information
- 2. Gain privileged access to machines on intranet connected via Internet

CHAPTER 2 CRYPTOGRAPHIC TOOLS

ANSWERS TO QUESTIONS

2.1 Cryptanalysis, one of the approaches to attack symmetric encryption, relies on the nature of the encryption algorithm plus some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs.

Brute-force attack, on the other hand, tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

2.2 In block cipher encryption, the input is processed one block of elements at a time, producing an output block for each input block whereas stream encryption processes the input elements continuously, producing output one element at a time, as it goes along.

2.3 (1) a strong encryption algorithm; (2) Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

2.4 The two important aspects of data authentication are: (i) to verify that the contents of the message have not been altered and (ii) that the source is authentic.

2.5 One-way hash function is an alternative to Message Authentication Code (MAC). Like MAC, oneway hash function too accepts a variable-size message as input and produces a fixed-size message digest as output. It differs from MAC in several aspects, for instance, it does not take a secret key as input like MAC. Moreover, the messages are typically padded out to an integer multiple of some fixed length (e.g., 1024 bits) and the padding includes the value of the length of the original message in bits. The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.

2.6 **(a)** A hash code is computed from the source message, encrypted using symmetric encryption and a secret key, and appended to the message.

At the receiver, the same hash code is computed. The incoming code is decrypted using the same key and compared with the computed hash code. **(b)** This is the same procedure as in (a) except that public-key encryption is used; the sender encrypts the hash code with the sender's private key, and the receiver decrypts the hash code with the sender's public key. **(c)** A secret value is appended to a message and then a hash code is calculated using the message plus secret value as input. Then the message (without the secret value) and the hash code are transmitted. The receiver appends the same secret value to the message and computes the hash value over the message plus secret value. This is then compared to the received hash code.

2.7
1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

2.8 **Plaintext:** This is the readable message or data that is fed into the algorithm as input. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext. **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input. **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts. **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

2.9 **Encryption/decryption:** The sender encrypts a message with the recipient's public key. **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message. **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

2.10 The ECC can offer the same security level as RSA for a significantly smaller sized secret key compared to RSA.

2.11 No, digital signatures do not provide confidentiality, i.e., the message being sent is safe from alteration but not safe from eavesdropping.

2.12 A **pubic-key certificate** consists of a public key plus a User ID of the key owner, with the whole block signed by a trusted third party. Typically, the third party is a certificate authority (CA) that is trusted by the user community, such as a government agency or a financial institution.

2.13 **1.** Generation of secret keys and session keys.
2. Generation of public and private keys.
3. Generation of initialization vector for stream ciphers/ block cipher modes of operation.

ANSWERS TO PROBLEMS

2.1 **a.** Divide the message M into p blocks (m_1, m_2, … , m_p) of n-bit each and apply the same encryption algorithm E with the same key k. The corresponding ciphertext C will be produced by appending the outputs of E, i.e., $C = E_k(m_1)||E_k(m_2)||\ldots||E_k(m_p)$.

b. If the message length is not the multiple of block length the ECB mode cannot be applied.

c. The main flaw of the ECB encryption is that it can reveal information about the plaintext by merely observing ciphertexts. Specifically, the problem with ECB mode is that encrypting the same of plaintext using ECB mode always produces the same block of ciphertext. This can allow an attacker to detect whether two messages are same or different.

2.2 **a.**

2 C	8 R	10 Y	7 P	9 T	6 O	3 G	1 A	4 H	5 I
B	E	A	T	T	H	E	T	H	I
R	D	P	I	L	L	A	R	F	R
O	M	T	H	E	L	E	F	T	O
U	T	S	I	D	E	T	H	E	L
Y	C	E	U	M	T	H	E	A	T
R	E	T	O	N	I	G	H	T	A
T	S	E	V	E	N	I	F	Y	O
U	A	R	E	D	I	S	T	R	U
S	T	F	U	L	B	R	I	N	G
T	W	O	F	R	I	E	N	D	S

4	2	8	10	5	6	3	7	1	9
N	E	T	W	O	R	K	S	C	U
T	R	F	H	E	H	F	T	I	N
B	R	O	U	Y	R	T	U	S	T
E	A	E	T	H	G	I	S	R	E
H	F	T	E	A	T	Y	R	N	D
I	R	O	L	T	A	O	U	G	S
H	L	L	E	T	I	N	I	B	I
T	I	H	I	U	O	V	E	U	F
E	D	M	T	C	E	S	A	T	W
T	L	E	D	M	N	E	D	L	R
A	P	T	S	E	T	E	R	F	O

```
ISRNG  BUTLF  RRAFR  LIDLP  FTIYO  NVSEE  TBEHI  HTETA
EYHAT  TUCME  HRGTA  IOENT  TUSRU  IEADR  FOETO  LHMET
NTEDS  IFWRO  HUTEL  EITDS
```

b. The two matrices are used in reverse order. First, the ciphertext is laid out in columns in the second matrix, taking into account the order dictated by the second memory word. Then, the contents of the second matrix are read left to right, top to bottom and laid out in columns in the first matrix, taking into account the order dictated by the first memory word. The plaintext is then read left to right, top to bottom.

c. Although this is a weak method, it may have use with time-sensitive information and an adversary without immediate access to good cryptanalysis t(e.g., tactical use). Plus it doesn't require anything more than paper and pencil, and can be easily remembered.

2.3 a. Let -X be the additive inverse of X. That is -X $\boxed{+}$ X = 0. Then:

$$P = (C \boxed{+} -K_1) \oplus K_0$$

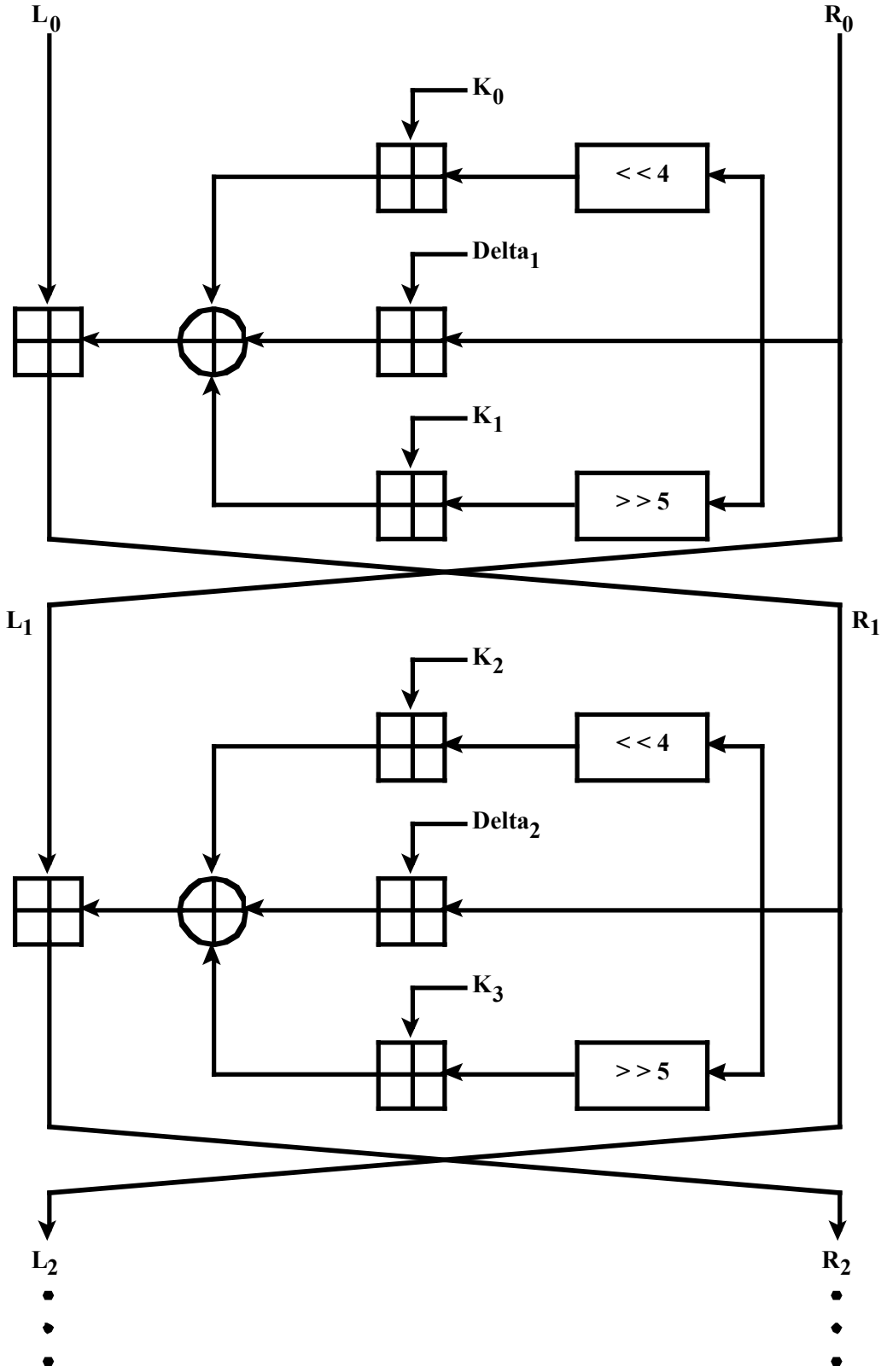
b. First, calculate –C'. Then –C' = (P' $\oplus$ K_0) $\boxed{+}$ (– K_1). We then have:

C $\boxed{+}$ –C' = (P $\oplus$ K_0) $\boxed{+}$ (P' $\oplus$ K_0)

However, the operations $\boxed{+}$ and $\oplus$ are not associative or distributive with one another, so it is not possible to solve this equation for K_0.

2.4 a. The constants ensure that encryption/decryption in each round is different.

b. First two rounds:



c. First, let's define the encryption process:

$L_2 = L_0 \boxed{+} [(R_0 << 4) \boxed{+} K_0] \oplus [R_0 \boxed{+} \delta_1] \oplus [(R_0 >> 5) \boxed{+} K_1]$

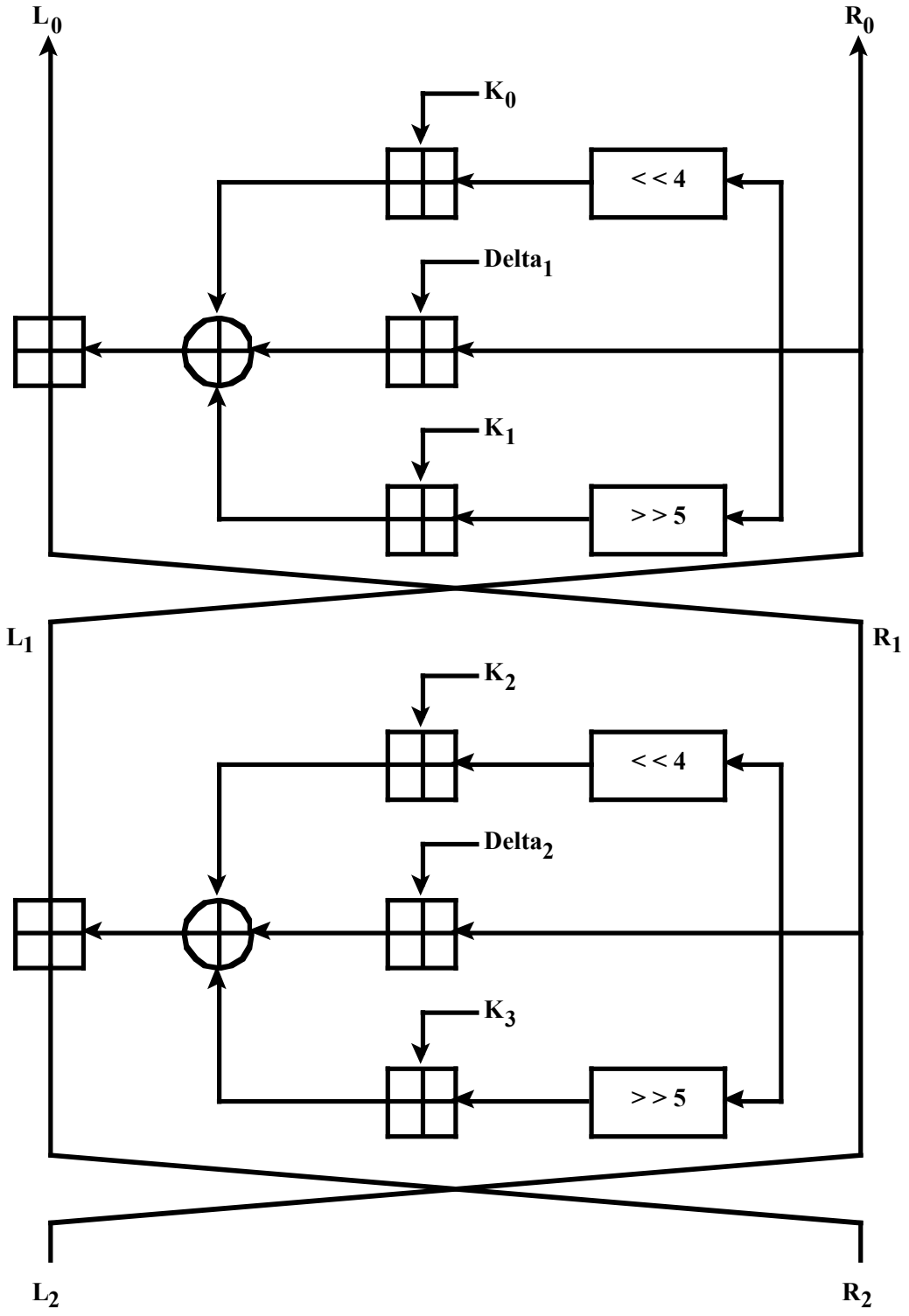
$R_2 = R_0 \boxed{+} [(L_2 << 4) \boxed{+} K_2] \oplus [L_2 \boxed{+} \delta_2] \oplus [(L_2 >> 5) \boxed{+} K_3]$

Now the decryption process. The input is the ciphertext (L_2, R_2), and the output is the plaintext (L_0, R_0). Decryption is essentially the same as encryption, with the subkeys and delta values applied in reverse order. Also note that it is not necessary to use subtraction because there is an even number of additions in each equation.

$R_0 = R_2 \boxed{+} [(L_2 << 4) \boxed{+} K_2] \oplus [L_2 \boxed{+} \delta_2] \oplus [(L_2 >> 5) \boxed{+} K_3]$

$L_0 = L_2 \boxed{+} [(R_0 << 4) \boxed{+} K_0] \oplus [R_0 \boxed{+} \delta_1] \oplus [(R_0 >> 5) \boxed{+} K_1]$

d.



2.5 **a.** Will be detected with both (i) DS and (ii) MAC.
b. Won't be detected by either (Remark: use timestamps).
c. (i) DS: Bob simply has to verify the message with the public key from both. Obviously, only Alice's public key results in a successful verification.

(ii) MAC: Bob has to challenge both, Oscar and Bob, to reveal their secret key to him (which he knows anyway). Only Bob can do that.

d. (i) DS: Alice has to force Bob to prove his claim by sending her a copy of the message in question with the signature. Then Alice can show that message and signature can be verified with Bob's public key) Bob must have generated the message.
(ii) MAC: No, Bob can claim that Alice generated this message.

2.6 **1.** There are three main security requirements of hash functions:

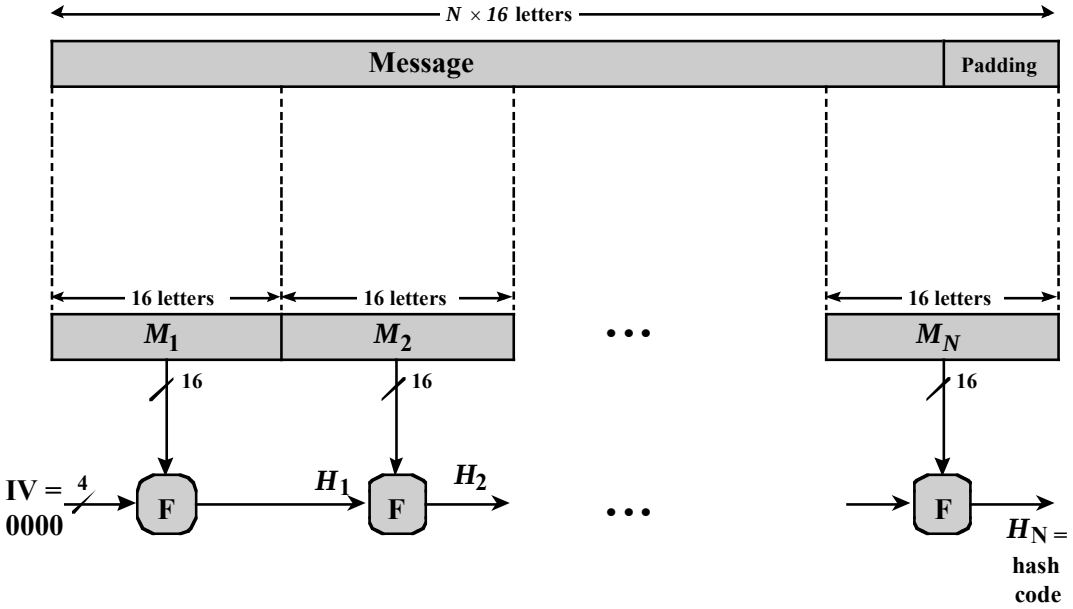
a. Pre-image resistance: For any given hash value h, it is computationally infeasible to find x such that $H(x) = h$.

b. Second pre-image resistance: For any given message x, it is computationally infeasible to find x' such that $x \neq x'$ and $H(x) = H(x')$.

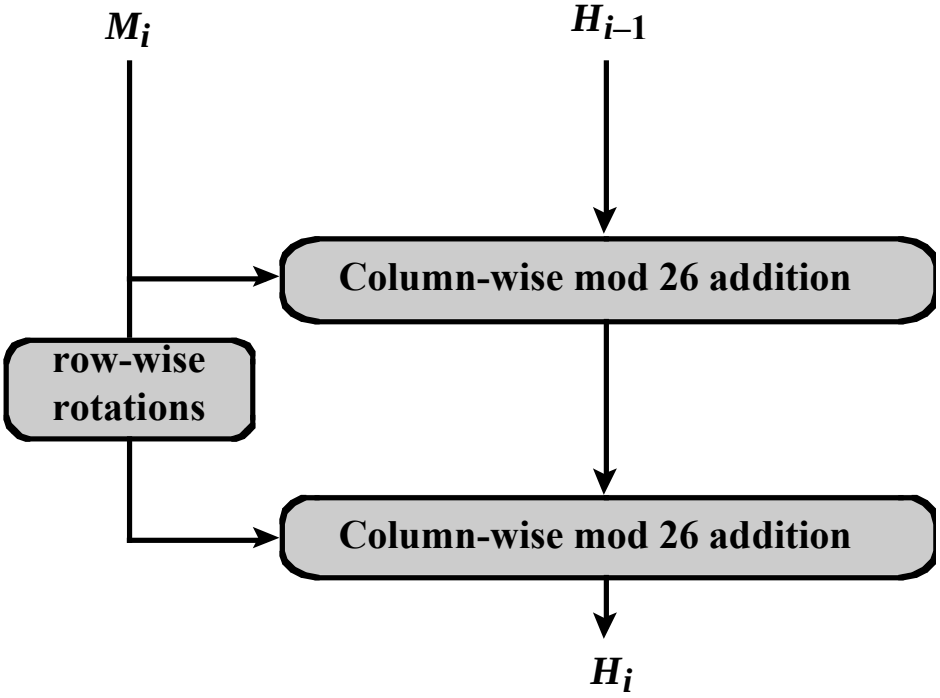
c. Collision resistance: It is computationally infeasible to find any two messages x and x' such that $x \neq x'$ with $H(x) = H(x')$.

2. The number of messages required will be 2^8, with high probability.

2.7 a. Overall structure:



Compression function F:



b. BFQG

c. Simple algebra is all you need to generate a result:

AYHGDAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA

2.8

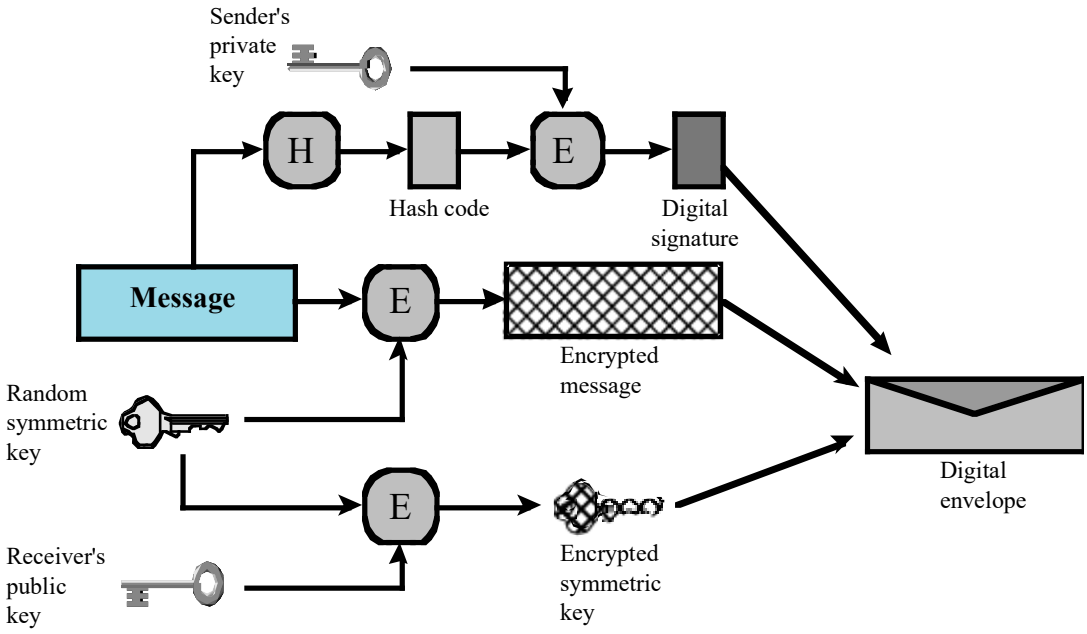
a. $M3 =$

5	2	1	4	5
1	4	3	2	2
3	1	2	5	3
4	3	4	1	4
2	5	5	3	1

b. Assume a plaintext message p is to be encrypted by Alice and sent to Bob. Bob makes use of M1 and M3, and Alice makes use of M2. Bob chooses a random number, k, as his private key, and maps k by M1 to get x, which he sends as his public key to Alice. Alice uses x to encrypt p with M2 to get z, the ciphertext, which she sends to Bob. Bob uses k to decrypt z by means of M3, yielding the plaintext message p.

c. If the numbers are large enough, and M1 and M2 are sufficiently random to make it impractical to work backwards, p cannot be found without knowing k.

2.9 We show the creation of a digital envelope:



CHAPTER 3 USER AUTHENTICATION

ANSWERS TO QUESTIONS

3.1 **Something the individual knows:** Examples includes a password, a personal identification number (PIN), or answers to a prearranged set of questions.
Something the individual possesses: Examples include electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a token.
Something the individual is (static biometrics): Examples include recognition by fingerprint, retina, and face.
Something the individual does (dynamic biometrics): Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

3.2 We can identify the following attack strategies and countermeasures:
Offline dictionary attack: Typically, strong access controls are used to protect the system's password file. However, experience shows that determined hackers can frequently bypass such controls and gain access to the file. The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that ID/password combination.
Specific account attack: The attacker targets a specific account and submits password guesses until the correct password is discovered.
Popular password attack: A variation of the preceding attack is to use a popular password and try it against a wide range of user IDs. A user's tendency is to choose a password that is easily remembered; this unfortunately makes the password easy to guess.
Password guessing against single user: The attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password.
Workstation hijacking: The attacker waits until a logged-in workstation is unattended.
Exploiting user mistakes: If the system assigns a password, then the user is more likely to write it down because it is difficult to remember. This situation creates the potential for an adversary to read the written password. A user may intentionally share a password, to enable a colleague to share files, for example. Also, attackers are frequently

successful in obtaining passwords by using social engineering tactics that trick the user or an account manager into revealing a password. Many computer systems are shipped with preconfigured passwords for system administrators. Unless these preconfigured passwords are changed, they are easily guessed.

Exploiting multiple password use. Attacks can also become much more effective or damaging if different network devices share the same or a similar password for a given user.

Electronic monitoring: If a password is communicated across a network to log on to a remote system, it is vulnerable to eavesdropping. Simple encryption will not fix this problem, because the encrypted password is, in effect, the password and can be observed and reused by an adversary.

3.3 Shadow password file stores the hashed passwords and is separate from the file which stores the user ID. Thus, it is very important to keep shadow password file protected from unauthorized access.

3.4 In proactive password checker approach, a user is allowed to select the desired password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

3.5 We can classify the authentication protocols used with smart tokens into three categories:

- **Static:** With a static protocol, the user authenticates himself or herself to the token and then the token authenticates the user to the computer.
- **Dynamic password generator:** In this case, the token generates a unique password periodically. This password is then entered into the computer system for authentication, either manually by the user or electronically via the token.
- **Challenge-response:** In this case, the computer system generates a challenge, such as a random string of numbers. The smart token generates a response based on the challenge.

3.6 **Facial characteristics:** Facial characteristics are the most common means of human-to-human identification; thus it is natural to consider them for identification by computer. The most common approach is to define characteristics based on relative location and shape of key facial features, such as eyes, eyebrows, nose, lips, and chin shape. An alternative approach is to use an infrared camera to produce a face thermogram that correlates with the underlying vascular system in the human face.

Fingerprints: Fingerprints have been used as a means of identification for centuries, and the process has been systematized and automated particularly for law enforcement purposes. A fingerprint is the pattern of ridges and furrows on the surface of the fingertip. Fingerprints are believed to be unique across the entire human population. In practice, automated fingerprint recognition and matching system extract a number of features from the fingerprint for storage as a numerical surrogate for the full fingerprint pattern.

Hand geometry: Hand geometry systems identify features of the hand, including shape, and lengths and widths of fingers.

Retinal pattern: The pattern formed by veins beneath the retinal surface is unique and therefore suitable for identification. A retinal biometric system obtains a digital image of the retinal pattern by projecting a low-intensity beam of visual or infrared light into the eye.

Iris: Another unique physical characteristic is the detailed structure of the iris.

Signature: Each individual has a unique style of handwriting and this is reflected especially in the signature, which is typically a frequently written sequence. However, multiple signature samples from a single individual will not be identical. This complicates the task of developing a computer representation of the signature that can be matched to future samples.

Voice: Whereas the signature style of an individual reflects not only the unique physical attributes of the writer but also the writing habit that has developed, voice patterns are more closely tied to the physical and anatomical characteristics of the speaker. Nevertheless, there is still a variation from sample to sample over time from the same speaker, complicating the biometric recognition task.

3.7 **Enrollment** is analogous to assigning a password to a user. For a biometric system, the user presents a name and, typically, some type of password or PIN to the system. At the same time the system senses some biometric characteristic of this user (e.g., fingerprint of right index finger). The system digitizes the input and then extracts a set of features that can be stored as a number or set of numbers representing this unique biometric characteristic; this set of numbers is referred to as the user's template. The user is now enrolled in the system, which maintains for the user a name (ID), perhaps a PIN or password, and the biometric value. **Verification** is analogous to a user logging on to a system by using a memory card or smart card coupled with a password or PIN. For biometric verification, the user enters a PIN and also uses a biometric sensor. The system extracts the corresponding feature and compares that to the template stored for this user. If there is a match, then the system authenticates this user.

For an **identification** system, the individual uses the biometric sensor but presents no additional information. The system then compares the

presented template with the set of stored templates. If there is a match, then this user is identified. Otherwise, the user is rejected

3.8 In local authentication, user attempts to access a system that is locally present, such as a stand- alone office PC or an ATM machine. In remote user authentication, user tries to authenticate himself over the Internet, a network, or a communications link. Remote user authentication raises additional security threats, such as an eavesdropper being able to capture a password, or an adversary replaying an authentication sequence that has been observed.

3.9 In this attack, an application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric. The adversary can then use the captured information to masquerade as a legitimate user.

ANSWERS TO PROBLEMS

3.1 **a.** Very unsuitable
b. Easily Guessable
c. Suitable
d. This is "password" in reverse; not suitable
e. Easily guessable if it is a license board
f. Easily guessable
g. Suitable
h. Suitable

3.2 No, a pseudorandom number generator with 216 possible starting values would not suffice because the number of possible character strings of length 9 using a 36-character alphabet is $369 \approx 246$. Thus, a pseudorandom number generator with 246 possible starting values would be required.

3.3 **a.** $T = \frac{10^9}{2 \times 3} =$ 46296.29 hours
b. Expect 5 tries for each digit. $T = 5 \times 9 = 45$ seconds

3.4 **a.** $p = r^k$
b. $p = \frac{r^k - r^p}{r^{k+p}}$
c. $p = r^p$

3.5 **a.** $T = (21 \times 5 \times 21)^2 = 4{,}862{,}025$
b. $p = 1/T \approx 2 \times 10^{-7}$

3.6 There are 1016 possible credit card numbers and the total time taken by an adversary to exhaustively test all these numbers is 31.69 years, which is around 109 seconds.

$$\text{Rate of testing is} = \frac{10^{16}}{10^{9}} = 10 \text{ million numbers per second}$$

3.7 i. 2688 passwords every 2ms resulting in 4,838,400,000 = $2^{32.17}$ passwords per hour

ii. 6 GB / 20 MB = 300 cores. Therefore, only 300 passwords can be tested every 2ms resulting in 2^{28} passwords per hour.

3.8 Without the salt, the attacker can guess a password and encrypt it. If ANY of the users on a system use that password, then there will be a match. With the salt, the attacker must guess a password and then encrypt it once for each user, using the particular salt for each user.

3.9 It depends on the size of the user population, not the size of the salt, since the attacker presumably has access to the salt for each user. The benefit of larger salts is that the larger the salt, the less likely it is that two users will have the same salt. If multiple users have the same salt, then the attacker can do one encryption per password guess to test all of those users.

3.10 a. If there is only one hash function (k = 1), which produces one of N possible hash values, and there is only one word in the dictionary, then the probability that an arbitrary bit b_i is set to 1 is just 1/N. If there are k hash functions, let us assume for simplicity that they produce k distinct hash functions for a given word. This assumption only introduces a small margin of error. Then, the probability that an arbitrary bit b_i is set to 1 is k/N. Therefore, the probability that b_i is equal to 0 is 1 – k/N. The probability that a bit is left unset after D dictionary words are processed is just the probability that each of the D transformations set other bits:

$$\Pr[b_i = 0] = \left(1 - \frac{k}{N}\right)^{D}$$

This can also be interpreted as the expected fraction of bits that are equal to 0.

b. A word not in the dictionary will be falsely accepted if all k bits tested are equal to 1. Now, from part (a), we can say that the expected fraction of bits in the hash table that are equal to one is $1 - \phi$. The probability that a random word will be mapped by a single hash function onto a bit that is already set is the probability that the bit generated by the hash function is in the set of bits equal to one, which is just $1 - \phi$. Therefore, the probability that the k hash functions applied to the word will produce k bits all of which are in the set of bits equal to one is $(1 - \phi)^k$.

c. We use the approximation $(1 - x) \approx e^{-x}$.

3.11 For a static biometric device, the user's biometric is matched to a template at the client; thus the client biometric device must be authenticated. For a dynamic biometric device, the host or server generates a random sequence of numbers, characters, or words. The user then generates a response (handwriting, voice) and the result is analyzed at the host. In this case, the client biometric device need not be authenticated.

CHAPTER 4 ACCESS CONTROL

ANSWERS TO QUESTIONS

4.1 Authentication means verifying that the credentials of a user or other system entities are valid. Authorization, on the other hand, means granting of a right or permission to a system entity to access a system resource.

4.2 **Role-based access control (RBAC)** controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles. RBAC may have a discretionary or mandatory mechanism.

4.3 **Owner:** This may be the creator of a resource, such as a file. For system resources, ownership may belong to a system administrator. For project resources, a project administrator or leader may be assigned ownership.
Group: In addition to the privileges assigned to an owner, a named group of users may also be granted access rights, such that membership in the group is sufficient to exercise these access rights. In most schemes, a user may belong to multiple groups.
World: The least amount of access is granted to users who are able to access the system but are not included in the categories owner and group for this resource.

4.4 Followings are the elements of access control:

1. **Subject:** A subject is an entity capable of accessing objects. A subject is typically held accountable for the actions they have initiated.
2. **Object:** An object is a resource to which access is controlled. In general, an object is an entity used to contain and/or receive information. Examples include records, blocks, pages, segments, files, portions of files, directories.
3. **Access Right:** An access right describes the way in which a subject may access an object.

4.5 ABAC stands for Attribute-based access control which is one of the categories of access control policy. It controls access based on

attributes of the user, the resource to be accessed, and current environmental conditions.

4.6 For each object, an **access control list** lists users and their permitted access rights. A **capability ticket** specifies authorized objects and operations for a user.

4.7 Sticky bit is the final permission bit amongst the 12 protection bits that are associated with each file. When this bit is set on a file, it indicates that the system should retain the file contents in memory following execution. When applied to a directory, it specifies that only the owner of any file in the directory can rename, move, or delete that file. This is useful for managing files in shared temporary directories.

4.8 $RBAC_0$ contains the minimum functionality for an RBAC system. $RBAC_1$ includes the $RBAC_0$ functionality and adds role hierarchies, which enable one role to inherit permissions from another role. $RBAC_2$ includes $RBAC_0$ and adds constraints, which restrict the ways in which the components of a RBAC system may be configured. $RBAC_3$ contains the functionality of $RBAC_0$, $RBAC_1$, and $RBAC_2$.

4.9 Mutually exclusive roles are roles such that a user can be assigned to only one role in the set. This limitation can be either static or dynamic, i.e., a user could be assigned only one of the roles in the set for a session.

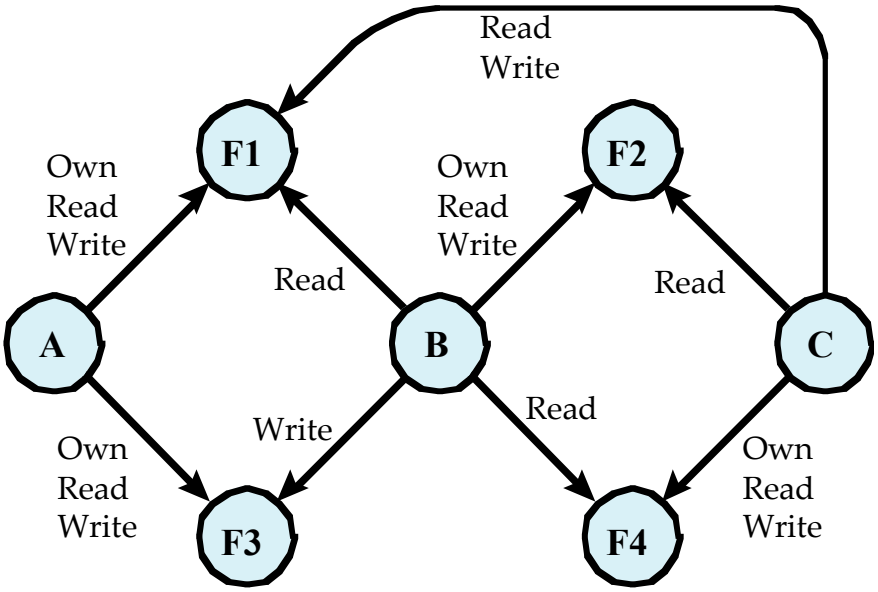
The mutually exclusive constraint supports a separation of duties and capabilities within an organization.

4.10 **Mutually exclusive roles** are roles such that a user can be assigned to only one role in the set. **Cardinality** refers to setting a maximum number with respect to roles. A system might be able to specify a **prerequisite**, which dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role.

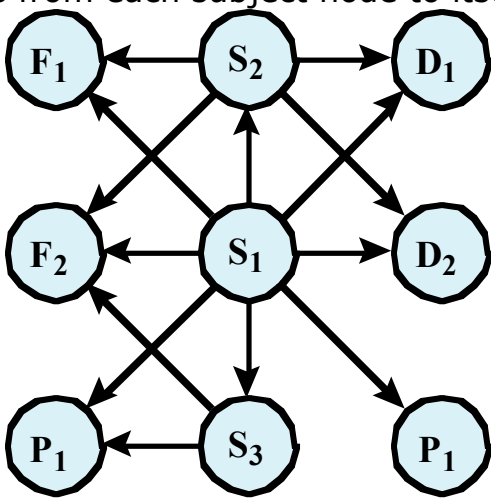
4.11 SSD enables the definition of a set of mutually exclusive roles, such that if a user is assigned to one role in the set, the user may not be assigned to any other role in the set. DSD specifications limit the availability of the permissions by placing constraints on the roles that can be activated within or across a user's sessions.

ANSWERS TO PROBLEMS

4.1 a.



b. For simplicity and clarity, the labels are omitted. Also, there should be arrowed lines from each subject node to itself.



c. A given access matrix generates only one directed graph, and a given directed graph yields only one access matrix, so the correspondence is one-to-one.

4.2 a. Protection domains provide flexibility to the user in terms of assigning the access rights to the objects. For example, each user can have a protection domain, and any process spawned by the user can have access rights defined by the same protection domain. Moreover, a user can also spawn processes with a subset of the access rights of the user which can be then defined as a new protection domain.

b. Most of the modern operating systems protect a certain amount of memory from the user by not permitting them to execute certain instructions (such instructions are also known as privileged instructions and can be executed only by the kernel of the operating system). This is done to enhance system's security against intruders. Thus, to make such a distinction among the instructions, operating systems use the concept of protection domains.

UNIX uses the concept of protection domain by implementing two modes of operation: user mode and kernel mode. Privileged instructions can be executed only in the kernel mode. If a normal user wants to execute these instructions, it makes a system call after which the kernel executes the privileged instructions on user's behalf.

4.3 **a.** The advantage of four modes is that there is more flexibility to control access to memory, allowing finer tuning of memory protection. The disadvantage is complexity and processing overhead. For example, procedures running at each of the access modes require separate stacks with appropriate accessibility.

b. In principle, the more modes, the more flexibility, but it seems difficult to justify going beyond four.

4.4 **a.** With $j < i$, a process running in D_i is prevented from accessing objects in D_j. Thus, if D_j contains information that is more privileged or is to be kept more secure than information in D_i, this restriction is appropriate. However, this security policy can be circumvented in the following way. A process running in D_j could read data in D_j and then copy that data into D_i. Subsequently, a process running in D_i could access the information.

b. An approach to dealing with this problem, known as a trusted system, is discussed in Chapter 10.

4.5 Suppose that the directory **d** and the file **f** have the same owner and group and that **f** contains the text *something*. Disregarding the superuser, no one besides the owner of **f** can change its contents can change its contents, because only the owner has write permission. However, anyone in the owner's group has write permission for **d**, so that any such person can remove **f** from **d** and install a different version, which for most purposes is the equivalent of being able to modify **f**. This example is from Grampp, F., and Morris, R. "UNIX Operating System Security." *AT&T Bell Laboratories Technical Journal*, October 1984.

4.6 A default UNIX file access of full access for the owner combined with no access for group and other means that newly created files

and directories will only be accessible by their owner. Any access for other groups or users must be explicitly granted. This is the most common default, widely used by government and business where the assumption is that a person's work is assumed private and confidential.
A default of full access for the owner combined with read/execute access for group and none for other means newly created files and directories are accessible by all members of the owner's group. This is suitable when there is a team of people working together on a server, and in general most work is shared with the group. However there are also other groups on the server for which this does not apply. An organization with cooperating teams may choose this.
A default of full access for the owner combined with read/execute access for both group and other means newly created files and directories are accessible by all users on the server. This is appropriate for organization's where users trust each other in general, and assume that their work is a shared resource. This used to be the default for University staff, and in some research labs. It is also often the default for small businesses where people need to rely on and trust each other.

4.7 In order to provide the Web server access to a user's 'public_html' directory, then search (execute) access must be provided to the user's home directory (and hence to all directories in the path to it), read/execute access to the actual Web directory, and read access to any Web pages in it, for others (since access cannot easily be granted just to the user that runs the web server). However this access also means that any user on the system (not just the web server) has this same access. Since the contents of the user's web directory are being published on the web, local public access is not unreasonable (since they can always access the files via the web server anyway). However in order to maintain these required permissions, if the system default is one of the more restrictive (and more common) options, then the user must set suitable permissions every time a new directory or file is created in the user's web area. Failure to do this means such directories and files are not accessible by the server, and hence cannot be access over the web. This is a common error. As well the fact that at least search access is granted to the user's home directory means that some information can be gained on its contents by other user's, even if it is not readable, by attempting to access specific names. It also means that if the user accidentally grants too much access to a file, it may then be accessible to other users on the system. If the user's files are sufficiently sensitive, then the risk of accidental leakage due to inappropriate permissions being set may be too serious to allow such a user to have their own web pages.

4.8 a. Assuming that the applicant is indicated as a and grants g, the rule can be as follows (e is ignored here):
R1:can_access(a, g, e) ←
(Age(a) > 35 Grants(g) ∈ {RG})

b. Assuming that the applicant is indicated as a and grants g, the rule can be as follows (e is ignored here):
R2:can_access(a, g, e) ←
(Age(a) ≤ 35 Grants(g) ∈ {RG, TG})

Both rules can be combined into one as shown below:
R:can_access(a, g, e) ←
(Age(a) > 35 Grants(g) ∈ {RG}) V
(Age(a) ≤ 35 Grants(g) ∈ {RG, TG})

4.9 The number of roles and permissions required for an RBAC model would be:

Π_(k=1)^K⟦Range(SA_k)⟧ andΠ_(m=1)^M⟦Range(SA_m)⟧

If additional attributes are added, the number of roles and permissions would grow exponentially. This limits the scalability of the approach.

4.10 a. $r_1 \succ\succ r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1) \wedge authorized\_users(r_2) \subseteq authorized\_users(r_1)$

b. $r \succ\succ r_1 \wedge r \succ\succ r_2 \Rightarrow r1 = r2$

4.11 a. Role(x) > Role(y) ⇔ Role(x).Position > Role(y).Position
Role(x).Function > Role(y).Function

b. Role(x) = Role(y) ⇔ Role(y).Function ^ Role(x).Position = Role(y).Position

4.12 ABAC Policy Rule: A user can access or view a movie can be resolved by evaluating a policy rule such as the following:

Rule 1: A user can access or view a movie, if he is either an adult or a juvenile or a child and the movie rating belongs to either R, or PG-13, or G category respectively.

Rule 2: A user can access or view a movie, if he is premium user, then he can access any type of movies: new-release or old-release; else if he is a regular user then he can access only old-released movies.

Rule 3: A user can access or view a movie, if he satisfies both the rules 1 and 2.

Advantages of ABA:
Eliminates the need of the administrative tasks for user-to-role assignment and permission-to-role assignment.

Easy to enforce a policy that only premium users can view new movies.

This model is more efficient than RBAC model, in the way of dealing additional attributes to accommodate finer-grained policies.

Moreover, it is easy to add more environmental attributes.

CHAPTER 5 DATABASE AND CLOUD SECURITY

ANSWERS TO QUESTIONS

5.1 A **database** is a structured collection of data stored for use by one or more applications. In addition to data, a database contains the relationships between data items and groups of data items. A **database management system (DBMS)**, which is a suite of programs for constructing and maintaining the database and for offering ad hoc query facilities to multiple users and applications. A **query language** provides a uniform interface to the database for users and applications.

5.2 A relational database is a collection of tables (also called relations). Individual values in a table can be used to link one table to another.

5.3 An SQLi attack is designed to send malicious SQL commands to the database server with the aim to extract, modify or delete data, execute arbitrary operating system commands or launch denial-of-service (DoS) attacks.

5.4 SQL injection (SQLi) attack is one of the most prevalent and dangerous network-based security threats. Through SQLi, attackers can dump database tables with hundreds of thousands of customer records. Depending on the environment, SQL injection can also be exploited to modify or delete data, execute arbitrary operating system commands, or launch DoS attacks.

5.5 The following are three main categories for grouping different types of SQLi attacks:

- inband attack
- inferential attack and
- out-of-band attack

5.6 A DBMS usually supports dozens of applications. In such an environment, an individual user may use a variety of applications to perform a variety of tasks, each of which requires its own set of privileges. It would be poor administrative practice to simply grant

users all of the access rights they require for all the tasks they perform. Since RBAC provides a means of easing the administrative burden and improves security, it is considered fit for database access control.

5.7 Encryption can be applied to any of the following levels:

- an entire database
- at the record level (encrypt selected records)
- at the attribute level (encrypt selected columns)
- at the level of the individual field

5.8 **Tier 1**: Basic; 99.671% availability.
Tier 2:: Redundant components; 99.741% availability.
Tier 3 Concurrently maintainable: 99.982% availability.
Tier 4 Fault tolerant; 99.995% availability.

ANSWERS TO PROBLEMS

5.1

Dept Name	Dept ID	Manager	No. of Employees

Manager Name	Dept Name	Dept ID

Employee Name	Dept Name	Dept ID

5.2 The first row can be added because a Student-Id with value 91 does not exist. The second row cannot be added because there already exists an entry with Student-ID having a value of 36. Thus, it violates the uniqueness property of the primary key (i.e., Student-ID). The third row cannot be added because there is no value assigned for a primary key. This violates the integrity constraint.

5.3 a. Following are the problems that may occur with the table:

i. Owner's detail need to be entered in multiple rows. Thus, there are possible chances of inconsistent data i.e., changes made in one row, but not in another.

ii. No place to store owner's (your customer) data unless they own a car.

b.

Card ID	Car Name	Car Model	Car Manufacturer	DOP	Owner Phone

OWNER

Owner Name	Owner Phone	Owner Email

5.4
```
CREATE TABLE employee (
        eid INTEGER PRIMARY KEY,
        efirstname VARCHAR (25),
        elastname VARCHAR (25),
        edept CHAR(25)    )
```

5.5 a. Given the employee name as David and his employee id as 939, this query searches for the "department" to which "David" is associated with.

b. Initially, all records in "employees" table will be selected where firstname would be "david" and then a DROP request would be executed, which deletes the "employees" table. "--", which is a comment indicator, disables the "id" check.

c. Since the OR condition "9=9" will always be true, the query will select all the records irrespective of the firstname entered by the user. Moreover, "--", which is a comment indicator, disables the "id" check.

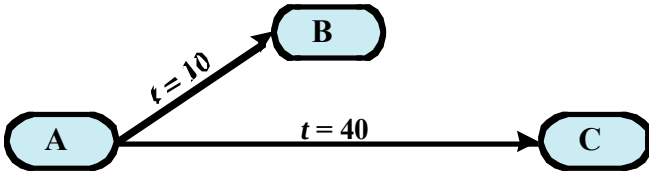
5.6 a. SELECT accounts FROM users WHERE login= 'Mike' AND pass= 'Mike@256' AND pin= '4242'.

b. The following SQL query will be generated:
SELECT accounts FROM users WHERE login= '' or ''= '' AND pass='' or ''= '' AND pin='' or ''= ''.
This is a valid SQL code and will transform the entire WHERE clause into a tautology (as '' = '' is always true). Since the condition is a tautology, the query evaluates to true for each row in the table and returns all rows. The returned set evaluates to a non-null value, which causes the application to conclude that the user authentication was successful.

5.7 This produces the following query:
SELECT accounts FROM users WHERE login= ‘’ OR EXISTS (SELECT * FROM users WHERE name='Mike' AND password LIKE '%t%') -- AND pass= ‘’ AND pin=
Suppose there is no login equal to ‘’ in the database but Mike’s password has a “t” in it. The WHERE clause will be evaluated to TRUE against the row containing Mike’s details (the EXISTS clause will fetch Mike’s details from the users database). The “--“ operator comments out the rest of the part of the query. Thus, the original query will display accounts information of Mike. This will not only reveal Mike’s account data to the attacker but also the fact that Mike’s password has a “t” in it. If Mike’s password has no “t” in it, the whole query will return a null set and authentication error message will be displayed. But this will let the attacker know that Mike’s password has no “t” in it.

5.8 The grant of the DELETE privilege by X at time t = 25 must be revoked because X's earliest remaining DELETE privilege was received at time t = 30. But X’s grants of READ and INSERT are allowed to remain because they are still supported by incoming grants that occurred earlier in time.

5.9



5.10 a. Consider the the concept of a noncascading revocation, defined as follows. Whenever user A revokes a privilege from user B, authorizations granted by B are not revoked; instead they are respecified as if they had been granted by S, the user issuing revocation. The semantics of the revocation without cascade is to produce the authorization state that would have resulted if the revoker (A) had granted the authorizations that were granted by the revokee (B).

b. The disadvantage of the cascading revocation is that it can be disruptive. In many organizations the authorizations a user possesses are related to his or her particular task or function within the organization. If a user changes task or function (e.g., is promoted), it is desirable to remove only the authorizations of this user, without triggering recursive revocation of all authorizations granted by this user. The disadvantage of the noncascading revocation would be if there are instances when we would like to revoke a user's authorization and simultaneously undo any authorizations by that user. This could occur if the user comes under suspicion.

5.11

User	Permission level
Accounts payable clerk	Should be able to access and change all data.
Installation foreman	Needs to access but not change parts information. Probably does not need to have access to any vendor information except perhaps name.
Receiving clerk	Needs to be able to access and change parts information, such as number in stock. Should be able to access but not change vendor information.

5.12 We assume that there is a unique constraint on flight ID and cargo hold (to prevent scheduling two shipments for the same hold). When a user in role 2 sees that nothing is scheduled for hold C on flight 1254, the user might attempt to insert a new record to transport some vegetables on that flight. However, when he or she attempts to insert the record, the insert will fail due to the unique constraint. At this point, the user has all the data needed to infer that there is a secret shipment on flight 1254. The user could then cross-reference the flight information table to find out the source and destination of the secret shipment and various other information.

5.13
```
GRANT select ON inventory TO hulkhogan, undertaker;
GRANT select ON item TO hulkhogan, undertaker;
```

5.14 Another way is to create a separate (unclassified) relation HIRE-DATES (EMP#, START-DATE). Note that EMP # rather than S# must be used as the key for the new relation, otherwise the inference is not removed.

5.15 If the first two queries are answered, then the **max** query is denied whenever its value is exactly equal to the ratio of the sum and the count values (which happens when all the selected rows have the same salary value). Hence the attacker learns the salary values of all the selected rows when denial occurs.

CHAPTER 6 MALICIOUS SOFTWARE

ANSWERS TO QUESTIONS

6.1 The three broad mechanisms malware can use to propagate are:

- infection of existing executable or interpreted content by viruses that is subsequently spread to other systems;
- exploit of software vulnerabilities either locally or over a network by worms or drive-by-downloads to allow the malware to replicate; and
- social engineering attacks that convince users to bypass security mechanisms to install trojans, or to respond to phishing attacks.

6.2 Four broad categories of payloads that malware may carry are:

- corruption of system or data files;
- theft of service in order to make the system a zombie agent of attack as part of a botnet;
- theft of information from the system, especially of logins, passwords or other personal details by keylogging or spyware programs; and
- stealthing where the malware hides it presence on the system from attempts to detect and block it.

6.3 The characteristics of an advanced persistent threat giving its name are:

- **Advanced:** Use of a wide variety of intrusion technologies and malware, including the development of custom malware if required.
- **Persistent:** application of attacks over an extended period against the chosen target in order to maximize the chance of success.
- **Threats:** a result of the organized, capable, and well-funded attackers intent to compromise the specifically chosen targets.

6.4 The typical phases of operation of a virus or worm are:

- a dormant phase (when the virus is idle),
- a propagation phase (where it makes copies of itself elsewhere),
- a triggering phase (when activated), and
- an execution phase (to perform some target function).

6.5 A blended attack uses multiple methods of infection or propagation, to maximize the speed of contagion and the severity of the attack.

6.6 Worm is a computer program that can run independently and can propagate a complete working version of itself onto other hosts on a

network, usually by exploiting software vulnerabilities in the target system whereas zombie is a program activated on an infected machine to launch attacks on other machines.

6.7 The first function in the propagation phase for a network worm is for it to search for other systems to infect, a process known as scanning or fingerprinting. For such worms, which exploit software vulnerabilities in remotely accessible network services, it must identify potential systems running the vulnerable service, and then infect them. Then, typically, the worm code now installed on the infected machines repeats the same scanning process, until a large distributed network of infected machines is created.

6.8 Clickjacking is a vulnerability used by an attacker to collect an infected user's clicks. The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code. Also, by taking advantage of Adobe Flash or JavaScript, an attacker could even place a button under or over a legitimate button, making it difficult for users to detect.

6.9 A **trojan** is an (apparently) useful program or utility containing hidden code that, when invoked, performs some unwanted or harmful function. A Trojan enables malware to propagate as it executes with the privileges of the person running it. Trojans are very common on both computer systems, and increasingly on mobile platforms, though much more on Android than on iOS devices.

6.10 Trojan horse is a computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes it.

6.11 A **backdoor** is a secret entry point into a program or system that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures.
A **bot** subverts the computational and network resources of the infected system for use by the attacker.
A **keylogger** captures keystrokes on the infected machine, to allow an attacker to monitor sensitive information including login and password credentials.
Spyware subverts the compromised machine to allow monitoring of a wide range of activity on the system, including monitoring the history and content of browsing activity, redirecting certain web page requests to fake sites controlled by the attacker, dynamically modifying data exchanged between the browser and certain web sites of interest;

which can result in significant compromise of the user's personal information.
A **rootkit** is a set of programs installed on a system to maintain covert access to that system with administrator (or root) privileges, whilst hiding evidence of its presence to the greatest extent possible.
These **can** all be present in the same malware.

6.12 A **phishing** attack uses a spam e-mail to exploit social engineering to leverage user's trust by masquerading as communications from a trusted source, that may direct a user to a fake Web site, or to complete some enclosed form and return in an e-mail accessible to the attacker. A more dangerous variant of this is the **spear-phishing** attack. This again is an e-mail claiming to be from a trusted source. However, the recipients are carefully researched by the attacker, and each e-mail is carefully crafted to suit its recipient specifically, often quoting a range of information to convince them of its authenticity. This greatly increases the likelihood of the recipient responding as desired by the attacker.

6.13 Clickjacking, also known as a user-interface (UI) redress attack, is a vulnerability used by an attacker to collect an infected user's clicks. The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code. Also, by taking advantage of Adobe Flash or JavaScript, an attacker could even place a button under or over a legitimate button, making it difficult for users to detect. A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is hijacking clicks meant for one page and routing them to another page, most likely owned by another application, domain, or both. Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

6.14 Following are the characteristics which can be used to classify rootkits:

- Persistent
- Memory based
- User mode
- Kernel mode
- Virtual machine based
- External mode

6.15 Following are the elements of a GD scanner:

- **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor.
- **Virus signature scanner:** A module that scans the target code looking for known malware signatures.
- **Emulation control module:** Controls the execution of the target code.

6.16 Countering rootkits requires a variety of network and computer-level security tools. Both network-based and host-based IDSs can look for the code signatures of known rootkit attacks in incoming traffic. Host-based anti-virus software can also be used to recognize the known signatures. In case of new rootkits and modified versions of existing rootkits that display novel signatures, a system needs to look for behaviors that could indicate the presence of a rootkit.

ANSWERS TO PROBLEMS

6.2 D is supposed to examine a program P and return TRUE if P is a computer virus and FALSE if it is not. But CV calls D. If D says that CV is a virus, then CV will not infect an executable. But if D says that CV is not a virus, it infects an executable. D always returns the wrong answer.

6.3 The original code has been altered to disrupt the signature without affecting the semantics of the code. The ineffective instructions in the metamorphic code are the second, third, fifth, sixth, and eighth.

6.4 **a.** The following is from Spafford, E. " The Internet Worm Program: An Analysis." Purdue Technical Report CSD-TR-823.
Common choices for passwords usually include fantasy characters, but this list contains none of the likely choices (e.g., "hobbit", "dwarf", "gandalf", "skywalker", "conan"). Names of relatives and friends are often used, and we see women's names like "jessica", "caroline", and "edwina", but no instance of the common names "jennifer" or "kathy". Further, there are almost no men's names such as "thomas" or either of "stephen" or "steven" (or "eugene"!). Additionally, none of these have the initial letters capitalized, although that is often how they are used in passwords. Also of interest, there are no obscene words in this dictionary, yet many reports of concerted password cracking experiments have revealed that there are a significant number of users who use such words (or phrases) as passwords. The list contains at least one incorrect spelling: "commrades" instead of "comrades"; I also believe that "markus" is a misspelling of "marcus". Some of the words do not appear in standard dictionaries and are non-English names:

"jixian", "vasant", "puneet", etc. There are also some unusual words in this list that I would not expect to be considered common: "anthropogenic", "imbroglio", "umesh", "rochester", "fungible", "cerulean", etc.

b. Again, from Spafford:
I imagine that this list was derived from some data gathering with a limited set of passwords, probably in some known (to the author) computing environment. That is, some dictionary-based or brute-force attack was used to crack a selection of a few hundred passwords taken from a small set of machines. Other approaches to gathering passwords could also have been used: Ethernet monitors, Trojan Horse login programs, etc. However they may have been cracked, the ones that were broken would then have been added to this dictionary. Interestingly enough, many of these words are not in the standard on-line dictionary (in /usr/dict/words). As such, these words are useful as a supplement to the main dictionary-based attack the worm used as strategy #4, but I would suspect them to be of limited use before that time.

6.5 Macro Code.

6.6 Drive-by-downloads.

6.7 A captcha is an image that renders text in a not-so-easy-to-read way, also known as challenge text. By verifying users to type the challenge text into a text field, it verifies some form of human interaction and intelligence. Spam bots, on the other hand, often lack the intelligence to defeat the challenge. This is because the challenge text appears in an image, and not as an html markup, reducing their chances of reading it by an automated program. Secondly, most spam-bots can't process images in an efficient manner to extract the embedded challenge in a short amount of time before the challenge image changes.

6.8 The way captchas are designed, they are often illegible and frustrating to deal with even for a legitimate user. This frustrates the users thus degrading the overall user experience.

A honeypot is a fake resource that is used to detect or divert information security attacks. Honeypots are designed such that they are unlikely to attract legitimate users. In case of email forms, a honeypot is a field added to the form that the users can't see due to the CSS or JavaScript which hides the field. We can understand this technique with the help of an example.

```
<input id="real_email" type="text" name="real_email" size="25" value=""/>
     <input id="test_email" type="text" name="email" size="25" value="" />
# test_email {
     display: none;
     }
```

Here, we have 2 email fields, real_email and test_email. test_email is hidden via the code "display: none". Therefore, this field is not visible to a human user and hence can't be submitted by them. Spam bots, on the other hand, will still see the field in the form's markup, and will attempt to auto-populate it with some value and submit it with the rest of the form. This behavior is sufficient for the process handling the form to distinguish legitimate users from a spam bot. If the hidden field was submitted with a value, the submission can be treated as a spam and rejected.

This is a more user friendly approach, as the field is hidden and out of the view of the users thus demanding no input from the users.

6.9 Yes, there can be a threat to the computer system. It may happen that while installing that software, the system became vulnerable to Adware and hence, instead of completing the desired task, the software keeps pointing to a different website, probably an ecommerce website. Such software should be removed from the system at the earliest because it may also steal other personal information from the system and pass on to the attacker.

6.10 If when you download and start to install some game app, you are asked to approve the access permissions "Send SMS messages" and to "Access your address-book", you should indeed be suspicious that a game wants these types of permissions, as it would not seem needed just for a game. Rather it could be malware that wants to collect details of all your contacts, and either return them to the attacker via SMS, or allow the code to send SMS messages to your contacts, perhaps enticing them to also download and install this malware. Such code is a trojan horse, since it contains covert functions as well as the advertised functionality.

6.11 If you should open the PDF attachment, then it could contain malicious scripting code that could run should you indeed select the 'Open' button. This may be either worm (specifically exploiting a client-side vulnerability), or trojan horse code. You could you check your suspicions without threatening your system by using the scroll bar to examine all the code about to be executed should you select the 'Open' button, and see if it looks suspicious. You could also scan the PDF document with suitable, up-to-date anti-virus software for any

signs of malware – though this will not detect unknown, zero-day exploits. This type of message is associated with a spear-phishing attack, given that the email was clearly crafted to suit the recipient. That particular e-mail would only have been sent to one or a few people for whom the details would seem plausible.

6.12 This email is attempting a general phishing attack, being sent to very large numbers of people, in the hope that a sufficient number of passengers use the named online air ticket reservation, and are fooled into divulging their sensitive login credentials to the attacker. Once the attacker gains login credentials from the passenger, he/she can misuse the account by booking tickets through stored credit card numbers, if any. The most likely mechanism used to distribute this e-mail is via a botnet using large numbers of compromised systems to generate the necessary high volumes of spam emails. You should never ever follow such a link in an email and supply the requested details. You should only ever access sensitive sites (probably the ones that involve financial transactions) by directly entering their known URL into your browser. It may be appropriate to forward a copy of such emails to a relevant contact at the online air ticket reservation system if they ask for this. Otherwise it should just be deleted.

6.13 Such a letter strongly suggests that an attacker has collected sufficient personal details about you, including your old password, in order to create a new password. This was most likely done using either a phishing attack, perhaps persuading you to complete and return some form with the needed personal details such as password; or answer to a secret question through which password can be guessed; or by using spyware installed on your personal computer system by a worm or trojan horse malware, that then collected the necessary details from files on the system, or by monitoring your access to sensitive sites.

6.14 The major limitation of this approach is that it can detect the malware only based on its content and not on its behavior. The reason is that the anti-virus software does not have access to malware's behavior. To overcome this limitation, several monitoring tools can be used, for instance, ingress monitors or egress monitors.

CHAPTER 7 DENIAL-OF-SERVICE ATTACKS

ANSWERS TO QUESTIONS

7.1 Poison packet is a form of system resource attack which uses packets whose structure triggers a bug in the system's network handling software, causing it to crash. Thus, the system can no longer communicate over the network until this software is reloaded.

7.2 SYN flooding attacks the ability of a network server to handle a large number of requests simultaneously whereas SYN spoofing attacks the ability of a network server to respond to TCP connection requests by overflowing the tables used to manage such connections.

7.3 HTTP flood is a DDoS attack whose goal is to consume considerable resources of a Web Server by bombarding it with several HTTP requests.

7.4 Poison packet attack is a system resource attack in which an attacker uses packets whose structure triggers a bug in the system's networking handling. The targeted system cannot communicate over the network until this software is reloaded. Two examples of such attack are ping of death attack and teardrop attack.

7.5 Many DoS attacks use packets with spoofed source addresses so any responses packets that result are no longer be reflected back to the original source system, but rather are scattered across the Internet to all the various forged source addresses. Some of these addresses might correspond to real systems, others may not be used, or not reachable. Any response packets returned as a result only add to the flood of traffic directed at the target system.

7.6 "**backscatter traffic**" are packets generated in response to a DoS attack packet with a forged random source address, e.g. the ICMP echo response from an ICMP echo request being used to flood a link. Monitoring these packets, which are randomly distributed over the Internet, gives valuable information on the type and scale of attacks. This **backscatter traffic** provides information on any DoS attacks that use a forged random source address with the destination address being

the target, including various single and distributed flooding attacks, and syn spoofing attacks. It does not provide information on attacks that do not use randomly forged source addresses, or reflection or amplification attacks where the forged source address is that of the desired target.

7.7 The main difference between DDoS attack and DoS attack is the use of multiple attacking systems controlled by one attacker in the scenario of the DDoS attack. Involvement of multiple systems results in scaling up the volume of the attack. Further, by directing the attack through intermediaries, the attacker is further distanced from the target, and is significantly harder to locate and identify.

7.8 Distributed denial of service (DDoS) attack botnets typically use a control hierarchy, where a small number of systems act as handlers controlling a much larger number of agent systems, as shown in Figure 7.4. These have are a number of advantages, as the attacker can send a single command to a handler, which then automatically forwards it to all the agents under its control. Automated infection tools can also be used to scan for and compromise suitable zombie systems.

7.9 An HTTP flood attack is a type of DDoS attack. It refers to an attack that bombards web servers with HTTP requests coming from many different bots. The requests can be designed to consume considerable resources, e.g., an HTTP request to download a large file from the target. It causes the web server to read the file from the disk, store it in memory, convert it into a packet stream, and then transmit the packets. This process consumes memory, processing, and transmission resources.

7.10 Slowloris is a form of HTTP flood attack. It exploits the common server technique of using multiple threads to support multiple requests to the same server application. It attempts to monopolize all available request handling threads on the Web server by sending HTTP requests that never complete. Since each request consumes a thread, the Slowloris attack eventually consumes all of the Web server's connection capacity, effectively denying access to legitimate users.

7.11 The following are the two disadvantages of the classic ping flood attack from the attacker's perspective:

- the source of the attack is explicitly identified, increasing the chance that the attacker can be identified.
- the targeted system will attempt to respond to the packets being sent. In the case of any ICMP echo request packets received by the server, it would respond to each with an ICMP echo response packet directed back to the attacker. This effectively reflects the attack back at the attacker.

7.12 Non-spoofed flooding attacks are best defended against by the provision of significant excess network bandwidth and replicated distributed servers, particularly when the overload is anticipated. This does have a significant implementation cost though. Rate limits of various types on traffic can also be imposed. However such attacks cannot be entirely prevented, and may occur "accidentally" as a result of very high legitimate traffic loads.

7.13 The SYN Cookies are the best possible method to defend against the SYN spoofing attack.

7.14 Like all the reflection-based attacks, the basic defense against DNS amplification attacks is to prevent the use of spoofed source addresses. This filtering needs to be done as close to the source as possible, by routers or gateways knowing the valid address ranges of incoming packets. Typically this is the ISP providing the network connection for an organization or home user. Otherwise, appropriate configuration of DNS servers, in particular limiting recursive responses to internal client systems only, as described in RFC 5358, can restrict some variants of DNS amplification attacks.

7.15 To prevent an organization's systems being used as intermediaries in a broadcast amplification attack, the best defense is to block the use of IP directed broadcasts. This can be done either by the ISP, or by any organization whose systems could potentially be used as an intermediary.

7.16 The terms **slashdotted** or **flash crowd** refer to very large volumes of legitimate traffic, as result of high publicity about a specific site, often as a result of a posting to the well-known Slashdot or other similar news aggregation site. There is very little that can be done to prevent this type of either accidental or deliberate overload, without also compromising network performance. The provision of significant excess network bandwidth and replicated distributed servers is the usual response as noted in question 7.12.

7.17 In order to successfully respond to a denial of service attack, a good incident response plan is needed to provide guidance. When a denial of service attack is detected, the first step is to identify the type of attack and hence the best approach to defend against it. From this analysis the type of attack is identified, and suitable filters designed to block the flow of attack packets. These have to be installed by the ISP on their routers. If the attack targets a bug on a system or application, rather than high traffic volumes, then this must be identified, and steps taken to correct it to prevent future attacks. In the case of an extended, concerted, flooding attack from a large number of

distributed or reflected systems, it may not be possible to successfully filter enough of the attack packets to restore network connectivity. In such cases the organization needs a contingency strategy to switch to alternate backup servers, or to rapidly commission new servers at a new site with new addresses, in order to restore service.

7.18 The organization may wish to trace the source of various types of packets used in a DoS attack. If non-spoofed addresses are used, this is easy. However if spoofed sources addresses are used, this can be difficult and time-consuming, as their ISP will need to trace the flow of packets back in an attempt to identify their source. This is generally neither easy nor automated, and requires cooperation from the network providers these packets traverse.

ANSWERS TO PROBLEMS

7.1 In a DoS attack using ICMP Echo Request (ping) packets 100 bytes in size, to flood a target organization using a 8 Megabit per second (Mbps) link the attacker needs 8000000/(100 × 8) = 10000 packets per second. If the packet size is 1000 bytes, then the attacker needs 8000000/(1000 × 8) = 1000 packets per second. If the packet size is 1460 bytes, then the attacker needs 80000000/(1460 × 8) ≈ 685 packets per second.

7.2 For a TCP SYN spoofing attack, on a system with a table for 256 connection requests, that will retry 5 times at 30 second intervals, before purging the request from its table, each connection request occupies a table entry for 6 × 30 secs (initial + 5 repeats) = 3 min. In order to ensure that the table remains full, the attacker must continue to send 256/ 3 or about 86 TCP connection requests per minute? Assuming the TCP SYN packet is 40 bytes in size, this consumes about 86 × 40 × 8 / 60, which is about 459 bits per second, a negligible amount.

7.3 In the distributed variant of the attack from Problem 7.1, a single zombie PC can send 256000/(100 × 8) = 32 packets per second. About 32 such zombie systems are needed to flood a target organization using a 8 Megabit per second (Mbps) link, looking either at 8 Mbps/256 kbps, or 10000/320 packets per sec. When packet size is 1000 bytes, each zombie can generate a maximum of 32 packets per second. Similarly, when the packet size is 1460 bytes, each zombie can generate around 22 packets per second.

Even if packet size is 1000 bytes or 1460 bytes, 32 zombie systems would be needed still because the maximum that each zombie system

can generate is 256 kbps, which is independent of the packet size. Given reports of botnets composed of many thousands of zombie systems, clearly multiple such simultaneously DDoS attacks are possible. As is an attack on a major organization with multiple, much larger network links (e.g., 1000 zombies with 256-kbps links can flood 256 Mbps of network link capacity).

7.4 The answers for the DNS amplification attack are the same as in Problem 7.1. On a 8-Mbps link, 10000 packets, each of 100 bytes, are needed per second. 1000 pps, each of 1000 bytes are needed to flood a 8-Mbps link, and 685 pps, each of 1460 bytes are needed to flood a 8-Mbps link. Assuming a 70-byte DNS request packet then $10000 \times 70 \times 8 = 5.6$ Mbps is needed to trigger the flood on a 8-Mbps link.

7.5 TCP services cannot be used because they are connection-oriented and hence, cannot be directed at a broadcast address. Broadcasts are inherently connectionless.

7.6 The answer to this question depends on the type of web server deployed by the organization, but again these features are common on enterprise grade devices.

7.7 In this future idealized more secure network, administrators of server systems still do need to be concerned about, and take further counter-measures against, DoS attacks. Attacks using real addresses from real systems with high bandwidth network connections are still possible, as are "flash-crowd" overloads as a result of, possible fraudulent, publicity. To reduce the impact of such attacks, measures are needed to manage intermittent high traffic volumes, as mentioned in review questions 7.11 and 7.13.

7.8 The results of practical lab experiments such as these depend on the facilities and equipment available.

CHAPTER 8 INTRUSION DETECTION

ANSWERS TO QUESTIONS

8.1

- **Apprentice:** Hackers with minimal technical skill who primarily use existing attack toolkits.
- **Journeyman:** Hackers with sufficient technical skills to modify and extend attack toolkits.
- **Master:** Hackers with high-level technical skills capable of discovering brand new categories of vulnerabilities, or writing new powerful attack toolkits.

8.2 Followings are the example of intrusion:

1. Guessing and cracking the passwords.
2. Defacing a webserver.
3. Using resources without permission.
4. Copying content a sensitive database.
5. Dialling into an unsecured modem and gaining internal network access.

8.3 On the basis of skills level, intruders can be classified into following categories:

1. **Apprentice:** Hackers with very basic technical skills. This type of attackers primarily use existing attack toolkits. They likely comprise the largest number of attackers, including many criminal and activist attackers. Since this class of attackers use existing toolkits they can be identified and defend easily.
2. **Journeyman:** This class of hackers have sufficient technical skills to modify and extend attack toolkits. They are capable in locating new vulnerabilities or to exploit that are similar to some already known. The changes in attack tools make identifying and defending against such attacks harder.
3. **Master:** This class of hackers have high-level technical skills capable of discovering brand new categories of vulnerabilities, or writing new powerful attack toolkits. These attackers are very difficult to defend.

8.4 Security intrusion is an event, or a combination of multiple events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system without having the permission to do so.

8.5 Followings are the classification of IDSs on the basis of source and data analysed:

1. **Host-based IDS:** This type of IDS monitors characteristics of single host and events occurring within the host.
2. **Network-based IDS:** This class of IDS monitors network traffic for particular network segments or devices and analyses network, transport, and application protocols to identify suspicious activity.
3. **Hybrid IDS:** This class of IDS combines information from a number of sensors, often both host and network-based, in a central analyser that is able to better identify and respond to intrusion activity.

8.6 **1.** If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.

2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

8.7 A **false positive**, or false alarm, is where authorized users are identified as intruders by an IDS. A **false negative** is when intruders are not identified as intruders by an IDS, as a result of a tighter interpretation of intruder behavior in an attempt to limit false positives.

8.8 The base-rate fallacy occurs when there is an attempt to detect a phenomenon that occurs rarely. The frequency of occurrence is referred to as the base rate. When the base rate is very low, it is difficult to achieve low levels of both false positives and false negatives.

8.9 •Run continually with minimal human supervision.
•Be fault tolerant in the sense that it must be able to recover from system crashes and reinitializations.
•Resist subversion. The IDS must be able to monitor itself and detect if it has been modified by an attacker.
•Impose a minimal overhead on the system where it is running.
•Be able to be configured according to the security policies of the system that is being monitored.
•Be able to adapt to changes in system and user behavior over time.
•Be able to scale to monitor a large number of hosts.

•Provide graceful degradation of service in the sense that if some components of the IDS stop working for any reason, the rest of them should be affected as little as possible.
•Allow dynamic reconfiguration; that is, the ability to reconfigure the IDS without having to restart it.

8.10 **Anomaly detection** involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior. **Signature or Heuristic detection** uses a set of known malicious data patterns (signatures) or attack rules (heuristics) that are compared with current behavior to decide if is that of an intruder. It is also known as misuse detection. This approach can only identify known attacks for which it has patterns or rules.

8.11 The three broad categories of classification approaches used by anomaly detection systems are:

- **Statistical**: Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics.
- **Knowledge based**: Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior.
- **Machine learning**: Approaches automatically determine a suitable classification model from the training data using data mining techniques.

8.12 The advantages of using machine-learning approaches for anomaly detection include:

- flexibility
- adaptability and
- ability to capture interdependencies between the observed metrics

8.13 **Signature approaches** match a large collection of known patterns of malicious data against data stored on a system or in transit over a network. **Rule-based heuristic identification** involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses. Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage.

8.14 The most important advantage of HIDS is that it can detect both external and internal intrusions, something that is not possible either with NIDSs or firewalls.

8.15 Signature and heuristic HIDS are currently more commonly deployed, particularly on Windows systems, due to the difficulty in gathering suitable data to use in anomaly HIDS, and because of the load placed on the system to gather and classify this data. Signature or heuristic based HIDS are widely used, particularly as seen in anti-virus (A/V), more correctly viewed as anti-malware, products. These are very commonly used on Windows systems, and also incorporated into mail and web application proxies on firewalls and in network based IDSs. These products are quite efficient at detecting known malware, however they are not capable of detecting zero-day attacks that do not correspond to the known signatures or heuristic rules.

8.16 A Distributed HIDS provide a more effective defense by coordination and cooperation among HIDSs across the network.

8.17 An **inline sensor** is inserted into a network segment so that the traffic that it is monitoring must pass through the sensor. A **passive sensor** monitors a copy of network traffic; the actual traffic does not pass through the device.

8.18 Followings are the three advantages:

1. Defence form outside world attacks.
2. Highlights problems with the network firewall policy or performance.
3. Analyse attacks that might target the Web server or ftp server.

8.19 As with host-based intrusion detection, network-based intrusion detection makes use of signature detection and anomaly detection. Unlike the case with HIDS, a number of commercial anomaly NIDS products are available, as well as more traditional signature detection systems.

8.20 A distributed or hybrid IDS combines in a central IDS, the complementary information sources used by HIDS with host-based process and data details, and NIDS with network events and data, to manage and coordinate intrusion detection and response in an organization’s IT infrastructure.

8.21 SNORT is an open source, highly configurable and portable host-based or network-based IDS. SNORT can perform real-time packet capture, protocol analysis, and content searching and matching. It is mainly designed to analyse TCP, UDP, and ICMP network protocols. SNORT can detect a variety of attacks and probes, based on a set of rules configured by a system administrator.

The four logical component of SNORT installation are: 1. Packet decoder 2. Detection engine 3. Logger and 4. Alerter.

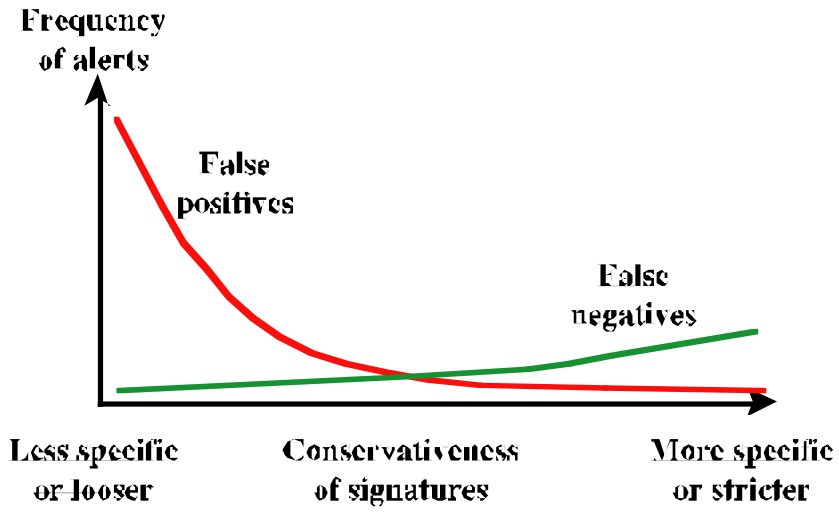
8.22 The following are the four logical components of a Snort architecture:

- Packet decoder
- Detection engine
- Logger and
- Alerter

ANSWERS TO PROBLEMS

8.1 Types of publicly available information that could be used by an attacker include: information required by law such as business registration and contact details or share registration details; contact information in phone books, DNS entries, network registration and WHOIS details; publicity and contact details provided by an organization on their website, or in publications handed out to the public. This suggests that from a security perspective, the content and detail of such information should be minimized. But this may well conflict with the organization's business and legal requirements to make this information available? It can be very difficult to reconcile these conflicting demands, though the appropriate balance may be suggested by the results of a risk assessment of the organization which may identify which types of information may be particularly dangerous. It may be possible to remove details of individual's names and positions, which would be of use in a spear-phishing attack, and use generic position details instead.

8.2 This is a typical example:



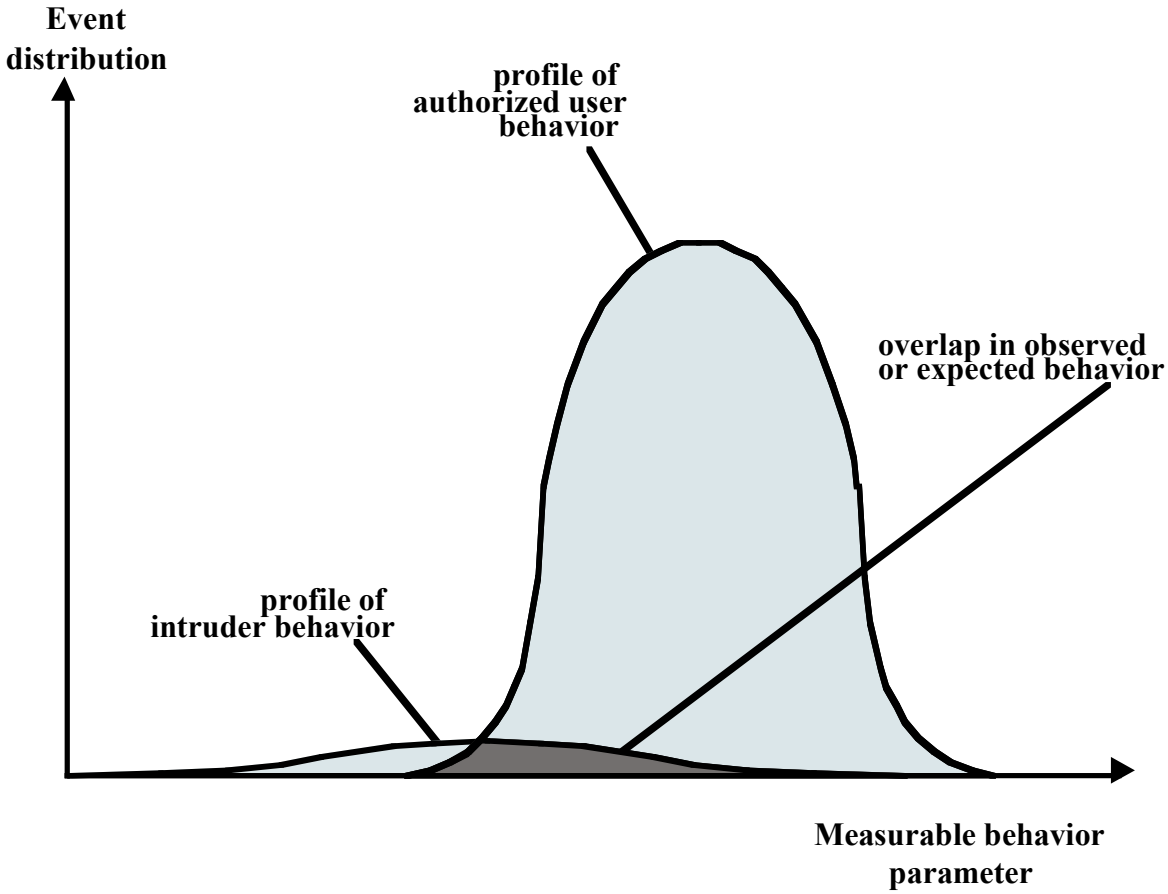
8.3 In case of passive sensors, a copy of the network traffic is monitored by the sensor rather than the actual copy of the network traffic as in case of inline sensors. This feature of passive sensors avoids an extra

handling step that contributes to the packet delay and hence, they are considered much efficient than inline sensors.

8.4 a. This rule wants to catch attempts to drop a database table. Line 1, the rule header, states that interesting packets are flowing from external IP addresses for database servers responding on Oracle ports. Line 2 is the text alert to be reported. Line 3 defines two additional matching conditions: first, packets must be directed to a server and must be part of an already established TCP connection, and second, the case-independent string "drop table_name" must be contained in the packet payload.

b. Typically, a system administrator would configure a system to forbid table deletion from across the Internet. Such attempts would be blocked by the firewall. The external NIDS would simply be a way of segregating out such attacks and alerting on them. If the NIDS is inside the firewall, it would be able to catch a serious deficiency in firewall behavior.

8.5 a. The graph below doesn't look like a correct probability distribution and is instead labeled as *event distribution*. The point here is that even if you have nice, mostly non-overlapping probability distributions for distinguishing intruders and authorized users like Figure 8.1, the problem is for most systems we hope the actual numbers of intruders is dwarfed by the number of authorized users. This means that the long tail of the authorized users distribution that overlaps with the intruders distribution would generate lots of false positives (relative to the number of real intruders detected) even if it is only a few percent of the authorized users.



b. A randomly selected event that in the overlap region is (roughly) 95% likely to be an authorized user, even though the region covers 50% of the intruders probability distribution.

8.6 A file integrity checking tool such as tripwire can be very useful in identifying changed files or directories on a system, particularly when those change should not have occurred. However most computer systems are not static, and significant numbers of files do change constantly. Hence it is necessary to configure tripwire with a list of files and directories to monitor, since otherwise reports to the administrator would be filled with lists of files that are changing as a matter of normal operation of the system. It is not too difficult to monitor a small list of critical system programs, daemons and configuration files. Doing this means attempts to alter these files will likely be detected. However the large areas of the system not being monitored means an attacker changing or adding files in these areas will not be detected. The more of the system that is to be monitored, the more care is needed to identify only files not expected to change. Even then, it is likely that user's home areas, and other shared document areas, cannot be monitored, since they are likely to be creating and changing files in there regularly. As well, there needs to be a process to manage the update of monitored files (as a result of installing patches, upgrades, new services, configuration changes etc). This process has to verify that the changed

files are correct, and then update the cryptographic checksums of these files. Lastly the database of cryptographic checksums must be protected from any attempt by an attacker to corrupt it, ideally by locating on read-only media (except when controlled updates are occurring).

8.7 This is a conditional probability problem. Total possible combinations for threat level Medium are: (P3, P4), (P4, P3) out of a total number of combinations = 16 – [Number of Events with neither of the two nodes generating a P3 signature] = 16 – 9 = 7 All Possibilities:

(P1, P1)(P1, P2), (P1, P3)(P1, P4)(P2, P1)(P2, P2)(P2, P3)(P2, P4)(P3, P1)(P3, P2)(P3, P3)(P3, P4)(P4, P1)(P4, P2)(P4, P3) (P4, P4)

Therefore, the Probability is = 2/7

Or Let A = {1 P3 and 1 P4}, B = {at least one is P3}
Pr[A|B] = Pr[AB]/ P[B]
Pr[AB] is the probability that one outcome is P3 and one outcome is P4 AND that at least one outcome is P3, is 2/16, and the probability of getting at least one P3 is 7/16, therefore, the

Pr[A|B] = [2/16]/[7/16] = 2/7

8.8 Let WB equal the event {witness reports Blue cab}. Then:

$$\Pr[\text{Blue}/\text{WB}] = \frac{\Pr[\text{WB}/\text{Blue}]\Pr[\text{Blue}]}{\Pr[\text{WB}/\text{Blue}]\Pr[\text{Blue}] + \Pr[\text{WB}/\text{Green}]\Pr[\text{Green}]}$$

$$= \frac{(0.8)(0.15)}{(0.8)(0.15)+(0.2)(0.85)} = 0.41$$

This example, or something similar, is referred to as "the juror's fallacy."

CHAPTER 9 FIREWALLS AND INTRUSION PREVENTION SYSTEMS

ANSWERS TO QUESTIONS

9.1 The different types of firewall are:
- Packet filtering firewall
- Stateful inspection firewall
- Application proxy firewall
- Circuit-level proxy firewall

9.2 **IP Address and Protocol Values**: Controls access based on the source or destination addresses and port numbers, direction of flow being inbound or outbound, and other network and transport layer characteristics.
Application Protocol: Controls access on the basis of authorized application protocol data.
User Identity: Controls access based on the users identity, typically for inside users who identify themselves using some form of secure authentication technology, such as IPSec.
Network Activity: Controls access based on considerations such as the time or request, or other activity patterns.

9.3 IP address spoofing, source routing attacks and tiny fragment attacks are possible on packet filtering firewall.

9.4 A traditional packet filter makes filtering decisions on an individual packet basis. It does not take into consideration any higher-layer context.

9.5 A **traditional packet filter** makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context. A **stateful inspection packet filter** tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, as shown in Table 9.2. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory

9.6 A gateway is a device that joins two different networks usually an internal network with the internet. A router is an example of gateway device.

A firewall is a filter that examines a packet against a set of predefined rules in order to decide whether to allow the packets through.

9.7 Circuit-level gateways can be used in a situation where the administrator trusts the internal users. The gateway can be configured to support application-level or proxy service on inbound connections and circuit-level functions for outbound connections.

9.8 **Packet filtering firewall:** Applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet.
Stateful inspection firewall: Tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, as shown in Table 9.2. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory.
Application proxy firewall: Acts as a relay of application-level traffic (Figure 9.1d). The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features
Circuit-level proxy firewall: As with an application gateway, a circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

9.9 • The bastion host hardware platform executes a secure version of its operating system, making it a hardened system.
• Only the services that the network administrator considers essential are installed on the bastion host. These could include proxy applications for DNS, FTP, HTTP, and SMTP.

• The bastion host may require additional authentication before a user is allowed access to the proxy services. In addition, each proxy service may require its own authentication before granting user access.
• Each proxy is configured to support only a subset of the standard application's command set.
• Each proxy is configured to allow access only to specific host systems. This means that the limited command/feature set may be applied only to a subset of systems on the protected network.
• Each proxy maintains detailed audit information by logging all traffic, each connection, and the duration of each connection. The audit log is an essential tool for discovering and terminating intruder attacks.
• Each proxy module is a very small software package specifically designed for network security. Because of its relative simplicity, it is easier to check such modules for security flaws. For example, a typical UNIX mail application may contain over 20,000 lines of code, while a mail proxy may contain fewer than 1000.
• Each proxy is independent of other proxies on the bastion host. If there is a problem with the operation of any proxy, or if a future vulnerability is discovered, it can be uninstalled without affecting the operation of the other proxy applications. Also, if the user population requires support for a new service, the network administrator can easily install the required proxy on the bastion host.
• A proxy generally performs no disk access other than to read its initial configuration file. Hence, the portions of the file system containing executable code can be made read only. This makes it difficult for an intruder to install Trojan horse sniffers or other dangerous files on the bastion host.
• Each proxy runs as a nonprivileged user in a private and secured directory on the bastion host.

9.10 • Filtering rules can be tailored to the host environment. Specific corporate security policies for servers can be implemented, with different filters for servers used for different application.
• Protection is provided independent of topology. Thus both internal and external attacks must pass through the firewall.
• Used in conjunction with stand-alone firewalls, the host-based firewall provides an additional layer of protection. A new type of server can be added to the network, with its own firewall, without the necessity of altering the network firewall configuration.

9.11 Between internal and external firewalls are one or more networked devices in a region referred to as a DMZ (demilitarized zone) network. Systems that are externally accessible but need some protections are usually located on DMZ networks. Typically, the systems in the DMZ require or foster external connectivity, such as a corporate Web site, an e-mail server, or a DNS (domain name system) server.

9.12 **Intrusion Detection System:** A device or application that analyzes whole packets, both header and payload, looking for known events. When a known event is detected a log message is generated detailing the event.
Intrusion Prevention System: A device or application that analyzes whole packets, both header and payload, looking for known events. When a known event is detected the packet is rejected.
Firewall: A device or application that analyzes packet headers and enforces policy based on protocol type, source address, destination address, source port, and/or destination port. Packets that do not match policy are rejected.

9.13 The HIPS addresses the following malicious behavior:
- Modification of system resources
- Privilege-escalation exploits
- Buffer-overflow exploits
- Access to e-mail contact list
- Directory traversal

9.14 Like an IDS, an IPS can be host-based, network-based, or distributed/hybrid combining information from a range of host and network based sensors.

9.15
- Modification of system resources
- Privilege-escalation exploits
- Buffer-overflow exploits
- Access to e-mail contact list

9.16 The following are a few methods used by a NIPS device to identify malicious packets:
- Pattern matching
- Stateful matching
- Protocol anomaly
- Traffic anomaly
- Statistical anomaly

ANSWERS TO PROBLEMS

9.1 Another limitation of this approach is that it violates the end-to-end property which is usually recommended in networks. This is because all the packets are intercepted by the gateway and then forwarded to either of the hosts. End-to-end property states that no intermediate

device should be allowed to gain access to the contents of the packets being transferred.

9.2 When a TCP packet is fragmented so as to force interesting header fields out of the zero-offset fragment, there must exist a fragment with FO equal to 1. If a packet with FO = 1 is seen, conversely, it could indicate the presence, in the fragment set, of a zero-offset fragment with a transport header length of eight octets Discarding this one-offset fragment will block reassembly at the receiving host and be as effective as the direct method described above.

9.3 If the router's filtering module enforces a minimum fragment offset for fragments that have non-zero offsets, it can prevent overlaps in filter parameter regions of the transport headers.

9.4
1. Allow return TCP Connections to internal subnet.
2. Prevent Firewall system itself from directly connecting to anything.
3. Prevent External users from directly accessing the Firewall system.
4. Internal Users can access External servers,
5. Allow External Users to send email in.
6. Allow External Users to access WWW server.
7. Everything not previously allowed is explicitly denied.

9.5
a. Rules A and B allow inbound SMTP connections (incoming email)
Rules C and D allow outbound SMTP connections (outgoing email)
Rule E is the default rule that applies if the other rules do not apply.
b. Packet 1: Permit (A); Packet 2: Permit (B): Packet 3: Permit (C)
Packet 4: Permit (D)
c. The attack could succeed because in the original filter set, rules B and D allow all connections where both ends are using ports above 1023.

9.6
a. A source port is added to the rule set.
b. Packet 1: Permit (A); Packet 2: Permit (B): Packet 3: Permit (C)
Packet 4: Permit (D); Packet 5: Deny €; Packet 6: Deny €

9.7
a. Packet 7 is admitted under rule D. Packet 8 is admitted under rule C.
b. Add a column called ACK Set, with the following values for each rule:
A = Yes;
B = Yes; C = Any; D = Yes; E = Any

9.8 HIPS typically addresses the following types of malicious behavior:

- Modification of system resources such as registry settings, user accounts, etc.
- Privilege-escalation exploits that attempt to provide root access to an ordinary user.
- Buffer-overflow exploits.
- Access to e-mail contact list.
- Directory traversal vulnerability.

The areas for which a HIPS typically offers desktop protection are:

- System calls.
- File system access.
- System registry settings.
- Host input/output.

9.9 The reasons are two folds: (i) SOCKS service is located on TCP port 1080 and (ii) before UDP segments can be forwarded, they need to be authenticated. Thus, they must be authenticated by opening a TCP connection first.

9.10 A possible requirement to manage information leakage requires all external e-mail to be given a sensitivity tag (or classification) in its subject and for external e-mail to have the lowest sensitivity tag. At its simplest a policy can just require user's to always include such a tag in email messages. Alternatively with suitable email agent programs it may be possible to enforce the prompting for and inclusion of such a tag on message creation. Then, when external email is being relayed through the firewall, the mail relay server must check that the correct tag value is present in the Subject header, and refuse to forward the email outside the organization if not, and notify the user of its rejection.

9.11 Suitable packet filter rulesets FOR the "External Firewall" and the "Internal Firewall" respectively, to satisfy the stated "informal firewall policy", could be:

action	src	port	dest	port	flags	comment
permit	DMZ mail gateway	any	any	SMTP (25)		header sanitize
permit	any	any	DMZ mail gateway	SMTP (25)		content filtered
permit	any	any	DMZ mail gateway	POP3S (995)		user auth
permit	DMZ web proxy	any	any	HTTP/S (80,443)		content filtered, user auth
permit	DMZ DNS server	DNS (53)	any	DNS (53)		TCP & UDP
permit	any	DNS (53)	DMZ DNS server	DNS (53)		TCP & UDP
permit	any	any	any DMZ server	any	estab lished	return traffic flow
deny	any	any	any	any		block all else

action	src	port	dest	port	flags	comment
permit	any internal	any	DMZ mail gateway	SMTP (25)		
permit	any internal	any	DMZ mail gateway	POP3/S (110,995)		user auth
permit	any internal	any	DMZ web proxy	HTTP/S (80,443)		content filtered, user auth
permit	any internal	DNS (53)	DMZ DNS server	DNS (53)		UDP lookup
permit	DMZ DNS server	DNS (53)	any internal	DNS (53)		UDP lookup
permit	any internal	any	any DMZ server	SSH (22)		user auth on server
permit	mgmt user hosts	any	any DMZ server	SNMP (161)		
permit	any DMZ server	any	mgmt user hosts	SNMP TRAP (162)		
permit	any DMZ server	any	any internal	any	estab lished	return traffic flow
deny	any	any	any	any		block all else

9.12 Yes. A rule set such as the following will do the trick:

```
drop tcp *:* -> 5.6.7.8:*
```

The following might be a little better, because it does not restrict outbound connections initiated by our internal server:

```
drop tcp *:* -> 5.6.7.8:* (if SYN flag set)
```

9.13 a. Here are the strengths.

(1) It mediates all incoming traffic from external hosts and can protect against many attacks by outsiders;
(2) It is easier to manage and to update policies, because of single central location;
(3) It protects against some kinds of DoS attacks launched from the outside.

Here are the weaknesses.
(1) It has no protection against malicious insiders;
(2) It has no protection for mobile laptops while they are connected to other networks;
(3) It has no protection if laptops get infected while travelling and then spread infection when they re-connect to our internal network

b. Here are the strengths.
(1) It protects against malicious insiders and infected internal machines as well as outside attackers;
(2) It protects laptops even while they are travelling and connected to other networks;
(3) It may be easier to customize firewall protection on a per-machine basis.

Here are the weaknesses.
(1) It is potentially more difficult to manage policies, due to the number of machines whose rulesets must be configured and updated;
(2) Uncooperative users may be able to modify settings or disable firewalls on their own machines, and viruses/worms may be able to do the same to machines they infect;
(3) It is potentially less resistant to DDoS, since DoS attacks can still flood internal network links;
(4) Depending upon firewall configuration, it may block legitimate internal traffic and/or make some internal services harder to use.

c. Here are the strengths.
(1) Layered defense provides redundancy in case one firewall fails;
(2) It can easily update policy against external attacks if a new threat develops, which gives some time to update the rulesets on internal hosts.
(3) Strengths (a)(1) and (b)(1)–(3) also apply.

Here are the weaknesses.
(1) Potential for overblocking of legitimate traffic, since traffic flows only if permitted by both firewalls.
(2) Weaknesses (b)(1), (b)(4) also apply

9.14 Modify the rule action from `Alert` to `Drop` to block these packets entering the home network, and to log the attempt:

```
Drop tcp $EXTERNAL_NET any -> $HOME_NET any\
(msg: “SCAN SYN FIN” flags: SF, 12;\
reference: arachnids, 198; classtype: attempted-recon;)
```

CHAPTER 10 BUFFER OVERFLOW

ANSWERS TO QUESTIONS

10.1 When an attacker tries to launch a buffer overflow attack and if that targeted buffer is stored on a stack, it leads to stack buffer overflow. This kind of attack is also known as stack smashing.

10.2 Buffer overflow attacks typically target buffers located in one of the stack, the heap, or the data section of a process.

10.3 The modern high-level programming languages like Java, ADA, and many others have a very strong notion of the type of variables and what constitutes permissible operations on them. Such languages do not suffer from buffer overflows because they do not permit more data to be saved into a buffer than it has space for. The higher levels of abstraction, and safe usage features of these languages mean programmers can focus more on solving the problem at hand and less on managing details of interactions with variables.

10.4 A stack buffer overflow occurs when the targeted buffer is located on the stack, usually as a local variable in a function's stack frame. This form of attack is also referred to as stack smashing.

10.5 Attacker can identify the vulnerable programs by inspection of program source, tracing the execution of programs as they process oversized input, or using other tools, such as fuzzing.

10.6 To better understand how buffer overflows work, we first take a brief digression into the mechanisms used by program functions to manage their local state on each call. When one function calls another, at the very least it needs somewhere to save the return address so the called function can return control when it finishes. Aside from that, it also needs locations to save the parameters to be passed in to the called function and also possibly to save register values that it wishes to continue using when the called function returns. All of these data are usually saved on the stack in a structure known as a stack frame.

10.7 Off-by-one is one of the categories of buffer overflow attack which can occur in a binary buffer copy when the programmer has included code

to check the number of bytes being transferred, but due to a coding error, allows just one more byte to be copied than there is space available.

10.8 There are several generic restrictions on the content of shellcode. Firstly it has to be "position-independent". That means it cannot contain any absolute address referring to itself, because the attacker generally cannot determine in advance exactly where the targeted buffer will be located in the stack frame of the function in which it is defined. Instead the code is written to determine its location when actually run. Another restriction is that it cannot contain any NULL values. This is a consequence of the common use of C string routines to copy this data into the buffer. To overcome this, any NULL values must be written in when the code actually runs.

10.9 A "NOP sled" is a run of NOP (no operation, do nothing) instructions, which are included before the desired shellcode to help overcome the lack of knowledge by the attacker of its precise location. In a buffer overflow attack, the attacker arranges for the transfer of control (via overwritten return address) to occur somewhere in the NOP Sled (guessing around the middle of the most likely location). Then when control transfers, no matter where in this run it occurs, the CPU executes NOPs until it reaches the actual desired shellcode.

10.10 Apart from just spawning a command-line shell, the attacker may wish to create shellcode to perform somewhat more complex operations. The Packet Storm website includes a large collection of packaged shellcode, including code that can: set up a listening service to launch a remote shell when connected to; create a reverse shell that connects back to the hacker; local exploits that establish a shell or execve a process; flush firewall rules (such as IPTables and IPChains) that currently block other attacks; break out of a chrooted (restricted execution) environment, giving full access to the system.

10.11 Two broad categories of defenses against buffer overflows are: compile-time defenses which aim to harden programs to resist attacks in new programs; and run-time defenses which aim to detect and abort attacks in existing programs.

10.12 Compile-time defenses include: writing programs using a modern high-level programming language that is not vulnerable to buffer overflow attacks; using safe coding techniques to validate buffer use; using language safety extensions and/or safe library implementations; or using stack protection mechanisms.

10.13 Run-time defenses that provide some protection for existing vulnerable programs include: using "Executable Address Space Protection" that blocks execution of code on the stack, heap, or in global data; using "Address Space Randomization" to manipulate the location of key data structures such as the stack and heap in the processes address space; or by placing **guard pages** between critical regions of memory in a processes address space.

10.14 no-execute bit in the x86 processor family has been added in the Memory Management Unit so that the pages of virtual memory can be tagged as nonexecutable. This is mainly required to block attacker from executing the code on the stack.

10.15 Stackguard is one among the most popular stack protection mechanisms that protects programs against stack overflow attacks. It is a GCC compiler extension that inserts additional function entry and exit code.

10.16 A possible approach to overcome global data area overflow attack is to make the global data area nonexecutable, arranging function pointers to be located below any other types of data and using guard pages between the global data area and any other management areas.

ANSWERS TO PROBLEMS

10.1 Safer variants of the unsafe standard C library functions shown in Table 10.2 are:

Original Unsafe Function	Safer Alternative
`gets(char *str)`	`fgets(char *str, int size, FILE * fil)`
`sprintf(char *str, char *fmt, ...)`	`snprintf(char *str, int size, char * fmt, ...)`
`strcat(char *dest, char *src)`	`strncat(char *dest, char *src, int count)` `strlcat(char *dest, char *src, int size)`
`strcpy(char *dest, char *src)`	`strncpy(char *dest, char *src, int count)` `strlcpy(char *dest, char *src, int size)`
`vsprintf(char *str, char *fmt, va list ap)`	`vsnprintf(char *str, int size, char *fmt, va_list ap)`

nb. the `strlXXX` routines are regarded as safer than the `strnXXX` routines, but may not be available on all systems.

10.2 Output of the program:
`./buffer1`

```
SECURITYSECURITY
buffer1: str1(SECURITY), str2(SECURITYSECURITY), valid(1)
```

The comparison in this case would be successful with the given input string `SECURITYSECURITY` because even though the content of `str1` is overwritten, the new content contained in it is similar to the first 8 characters of the content stored in `str2`.

10.3
```
./buffer2
Enter value for name: Computer Engineering
Segmentation fault (core dumped)
```

Since the input string length is more than the size of the buffer, the data extends beyond the end of the buffer and overwrites the saved frame pointer and return address with garbage values. Thus, when the function attempts to transfer control to the return address, it typically jumps to an illegal memory location, resulting in a Segmentation Fault.

10.4
```
./buffer3
Input value:
Computer Security
buffer3 getinp read Computer Securi
read val: Computer Securi
buffer3 done
Segmentation fault (core dumped)
```

Since the destination buffer always stores contents of many other buffers, when the read string is merged with the already present content in it, destination buffer’s overall size increases beyond the available space. This leads to corruption of the stack frame pointer and return address both, depending on the size of the destination buffer.

10.5 The extended shellcode from Figure 10.8b including a call to `exit(0)` is (see bold lines):

```
        jmp    find              // jump to end of code
cont:   pop    %esi              // pop address of sh off stack into %esi
        xor    %eax,%eax         // zero contents of EAX
        mov    %al,0x7(%esi)     // copy zero byte to end of string sh (%esi)
        lea    (%esi),%ebx       // load address of sh (%esi) into %ebx
        mov    %ebx,0x8(%esi)    // save address of sh in args[0] (%esi+8)
        mov    %eax,0xc(%esi)    // copy zero to args[1] (%esi+c)
        mov    $0xb,%al          // copy execve syscall number (11) to AL
        mov    %esi,%ebx         // copy address of sh (%esi) t0 %ebx
        lea    0x8(%esi),%ecx    // copy address of args (%esi+8) to %ecx
        lea    0xc(%esi),%edx    // copy address of args[1] (%esi+c) to %edx
        int    $0x80             // software interrupt to execute syscall
        mov    $0x1,%al          // copy exit syscall number (1) to AL
        xor    %ebx,%ebx         // zero contents of EBX
        int    $0x80             // software interrupt to execute syscall
find:   call   cont              // call cont which saves next address on
stack
sh:     .string "/bin/sh "       // string constant
args:   .long 0                  // space used for args array
        .long 0                  // args[1] and also NULL for env array
```

Note that the syscall numbers are listed in the architecture specific "unistd.h" include file. This code can be tested by changing the encoded shell name to one that does not exist (e.g.. "/bin/xx").

10.6 The details and results from running this experiment depend on the specific UNIX O/S variant and example vulnerable program used. The book's Web site includes a zipfile with the example programs used in this chapter, as run on a Knoppix CD-bootable system. Note that you may well need to use an older O/S release, since recent versions have defenses such as non-executable stack enabled by default.

10.7 The **Packet Storm (**http://www.packetstormsecurity.org/**)** site (shellcode area) includes several examples of PPC code to exec a shell under MacOSX (http://www.packetstormsecurity.org/shellcode/execMacOSX.txt, http://www.packetstormsecurity.org/shellcode/osx72bytes.txt) and Linux/PPC (http://www.packetstormsecurity.org/shellcode/execve-core.c). It also includes a comprehensive paper on this topic http://www.packetstormsecurity.org/shellcode/PPC_OSX_Shellcode_Assembly.pdf.

10.8 The details depend on which safe library alternative is examined. Note the Wikipedia page on "Buffer_overflow" provides links to the websites for many of the alternative safe library implementations.

10.9 No detailed answer available yet, however the Wikipedia page on "Return-to-libc_attack" provides some brief information and links to other resources.

10.10 Corrected version of the functions shown in Figure 10.10 (see bold lines). Note that the function "signatures" have to change, since information on the size of the buffer is needed (as seen in the safer variants of the string copy/cat functions).

```
int safe copy buf(char *to, int size, int pos, char *from, int len)
{
    int i;

    if (len <= 0)         /* invalid negative or zero len */
        return pos;
    if ((pos+len)>size)) /* len exceeds available space in buffer */
        len = size - pos;
    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

```
short safe read chunk(FILE fil, int size, char *to)
{
    short len;
    fread(&len, 2, 1, fil);     /* read length of binary data */
    if (len <= 0)               /* invalid negative or zero len */
        return 0;
    if (len > size))            /* len exceeds space in buffer */
        len = size;
    fread(to, 1, len, fil);     /* read len bytes of binary data
    return len;
}
```

10.11 Corrected version of the program shown in Figure 10.11a (see bold lines):

```
#define INP SIZE 64
/* record type to allocate on heap */
typedef struct chunk {
    char inp[INP SIZE];         /* input buffer */
    void (*process)(char *);    /* pointer to function to process inp */
} chunk t;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    chunk t *next;

    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk t));
    next->process = showlen;
    printf("Enter value: ");
    fgets(next->inp, INP SIZE, stdin);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

10.12 The details depend on the current vulnerability status.

10.13 Format string attacks alter the flow of an application by using string formatting library features to access other memory space. Vulnerabilities occur when user-supplied data are used directly as formatting string input for certain C/C++ format functions (e.g. fprintf, printf, sprintf, ...). If an attacker passes a format string consisting of printf conversion characters (e.g. "%f", "%p", "%n", etc.) as a parameter value to the application, they may: 1) execute arbitrary code on the server, 2) read values off the stack, 3) cause segmentation faults / software crashes.

Examples of format functions, which if not treated, can expose the application to the format string attack are fprint, printf, sprint, snprintf, vfprintf, vprintf, vsprintf and vsnprintf.

To avoid format string vulnerabilities, follow these guidelines: 1) use constant strings for formatting, 2) do not feed variables directly into format strings, 3) where possible, avoid using printf and sprint, 4) if absolutely necessary, consider quoting the "%" specifier before including a user-controlled input into a format string.

10.14 Since an integer is a fixed size, there is a fixed maximum value it can store. When an attempt is made to store a value greater than this maximum value it is known as an integer overflow. An integer overflow is a condition that occurs when the result of an arithmetic operation, such as multiplication or addition, exceeds the maximum size of the integer type used to store it.

Attackers can use these conditions to influence the value of variables in ways that the programmer did not intend. The security impact depends on the actions taken based on those variables. The following examples show the security impacts of the integer overflows.

1. An integer overflow during a buffer length calculation can result in allocating a buffer that is too small to hold the data to be copied into it. A buffer overflow can result when the data is copied.
2. When calculating a purchase order total, an integer overflow could allow the total to shift from a positive value to a negative one. This would, in effect, give money to the customer in addition to their purchases, when the transaction is completed.
3. Withdrawing 1 dollar from an account with a balance of 0 could cause an integer underflow and yield a new balance of 4,294,967,295.
4. A very large positive number in a bank transfer could be cast as a signed integer by a back-end system. In such case, the interpreted value could become a negative number and reverse the flow of money - from a victim's account into the attacker's.

There are several methods of handling the integer overflow:

1. **Avoidance:** by carefully ordering operations, checking operands in advance and selecting the correct data type, it is possible to ensure that the result will never be larger than can be stored.
2. **Handling:** If it is anticipated that overflow may occur and when it happens detected and other processing done. Example: it is possible to add two numbers each two bytes wide using just a byte addition in steps: first add the low bytes then add the high bytes, but if it is necessary to carry out of the low bytes this is arithmetic overflow of the byte addition and it necessary to detect and increment the sum of the high bytes. CPUs generally have a way of detecting this to support addition of numbers larger than their register size, typically using a status bit.
3. **Propagation:** if a value is too large to be stored it can be assigned a special value indicating that overflow has occurred and then have all successive operation return this flag value. This is useful so that the problem can be checked for once at the end of a long calculation rather than after each step. This is often supported in Floating Point Hardware called FPUs.

CHAPTER 11 SOFTWARE SECURITY

ANSWERS TO QUESTIONS

11.1 Software quality and reliability is concerned with the accidental failure of a program as a result of some theoretically random, unanticipated input, system interaction, or use of incorrect code. These failures are expected to follow some form of probability distribution. Software security differs in that the attacker chooses the probability distribution, targeting specific bugs that result in a failure that can be exploited by the attacker. These bugs may often be triggered by inputs that differ dramatically from what is usually expected, and hence are unlikely to be identified by common testing approaches.

11.2 **Defensive programming** is a form of defensive design intended to ensure the continuing function of a piece of software in spite of unforeseeable usage of said software. The idea can be viewed as reducing or eliminating the prospect of Murphy's Law having effect. Defensive programming techniques come into their own when a piece of software could be misused mischievously or inadvertently to catastrophic effect.

11.3 A buffer overflow occurs when a program attempts to write more data to a fixed length block of memory and overwrites adjacent memory locations.

11.4 An **injection attack** refers to a wide variety of program flaws related to invalid handling of input data, particularly when such input data can accidentally or deliberately influence the flow of execution of the program. Examples of injection attacks include: command injection, SQL injection, code injection, and remote code injection. There are a wide variety of mechanisms that can result in injection attacks. These include when input data is passed as a parameter to another helper program (command) or to a database system (SQL), whose output is then processed and used by the original program. Or when the input includes either machine or script code that is then executed/interpreted by the attacked system (code).

11.5 In a **command injection** attack, the unchecked input is used in the construction of a command that is subsequently executed by the

system with the privileges of the attacked program. In an SQL injection attack, the user-supplied input is used to construct a SQL request to retrieve information from a database. In both cases the unchecked input allows the execution of arbitrary programs/SQL queries rather than the program/query specified by the program designer. They differ in the syntax of the respective shell/SQL meta-characters used that allow this to occur.

11.6 Code injection attack is one among the different types of injection attack. In this attack, the input given to the program contains the malicious code and once the input is passed, this malicious code gets executed on the attacked system, as desired by the attacker. Example: The buffer overflow attacks include a code injection component.

11.7 The main technique used by a defensive programmer to validate assumptions about program input is to compare it against a regular expressions, which is a pattern that describes either what is wanted or what is known to be dangerous. The result of the comparison is used to either accept wanted, or reject dangerous, input.

11.8 Canonicalization (standardization or normalization) is a process for transforming the input data that has possibility of multiple encodings into a single, standard, minimal representation. Once canonicalization is done, the input data can be compared with a single representation of acceptable input values. This can be done to count the number of distinct data structures, to improve the efficiency of various algorithm by eliminating repeated calculations, or to make it possible to impose a meaningful soring order.

11.9 If the malicious content supplied by an attacker on a targeted site is displayed to other users without sufficient checking, they will execute the script assuming it is trusted to access any data associated with that site. This kind of an attack is called cross-site scripting (XSS) reflection vulnerability.

11.10 In case of multiple encodings while validating the input, the process of transforming the input data to a single, standard, minimal representation is called canonicalization. The significance of this process is that it replaces alternate, equivalent encodings by one common value.

11.11 A race condition can occur when several processes, or threads within a process, simultaneously access the same shared memory without suitable synchronization. The result can be that the shared memory

values may be corrupted, or changes lost, due to overlapping access, use and replacement of the shared values.

11.12 Environment variables are a collection of dynamic string values inherited by each process from its parent that can affect the way a running process behaves. Well-known environment variables include the variable PATH, which specifies the set of directories to search for any given command; IFS, which specifies the word boundaries in a shell script; and LD_LIBRARY_PATH, which specifies the list of directories to search for dynamically loadable libraries.

11.13 The most important advantage of fuzzing is its simplicity and its freedom from assumptions about the expected input to any program, service, or function. The disadvantage of fuzzing is that it only identifies simple types of faults with handling of input. If a bug exists that is only triggered by a small number of very specific input values, fuzzing is unlikely to locate it.

11.14 Memory leak is a situation which occurs while using dynamic memory allocation. In dynamic memory allocation, if the process of allocating and releasing the memory is not handled correctly, it may lead to a steady reduction in memory available on the heap and completely exhaust it.

Implications: The program will crash once the available memory on the heap is exhausted and hence, this provides an obvious mechanism for an attacker to implement a denial-of-service attack on such a program.

11.15 There are several issues associated with the correct creation and use of a temporary file in a shared directory, as they must be both unique, and not accessed by other processes. An attacker may attempt to guess the temporary filename a privileged program will use, and then attempt to create their own version in the interval between the program checking the file does not exist, and subsequently creating it. Secure temporary file creation and use requires the use of a random temporary filename, and its checking and creation using an atomic system primitive, similar to the creation of a lockfile.

11.16 Problems that may result from a program sending unvalidated input from one user to another user if the output does not conform to the expected form and interpretation by the recipient. A program may accept input from one user, save it, and subsequently display it to another user. If this input contains content that alters the behavior of the program or device displaying this data, and it is not adequately

sanitized by the program, then an attack on the user is possible. Examples include embedding terminal (e.g.. VT100) "escape sequences", or Javascript script code in an XSS attack.

ANSWERS TO PROBLEMS

11.1 A few of the possible ways to defend those types of attacks are:

- **a.** to block the assignment of form field values of global variables. Instead, they can be saved in an array and may be explicitly retrieved by name.
- **b.** to use only constant values in include and require commands. This ensures that the included code indeed originates from the specified files.

11.2 The most popular SQL metacharacters and reserved words in use today are as shown below:

```
; + \ * - ( ) OR AND
```

In particular the '`; \ * - and OR`" characters are used to separate commands, enable comments and check the required conditions respectively. Thus, they would most likely be seen in any attack. Hence, input validation checks to prevent SQL injection attacks for all of these special characters must typically reject the same characters (although as noted in the text, it is much better to write patterns to accept known good input values, which typically are alphanumeric plus limited punctuation).

11.3 Rewritten, extended version of perl script shown in Figure 11.2 (see bold lines):

```
#!/usr/bin/perl
# finger.cgi - finger CGI script using Perl5 CGI module

use CGI;
use CGI::Carp qw(fatalsToBrowser);
$q = new CGI;

# display HTML header
print $q->header,
      $q->start_html('Finger User'),
      $q->h1('Finger User');

# get and validate the name of user
$user = $q->param("user");
showError("The specified user '$user' contains illegal characters!")
    unless (("$user" =~ /^\w[-+%\w\t ]*\w$/) || ("$user" =~ /^\w$/));

# obtain desired finger information (including error messages)
print "<pre>";
print `/usr/bin/finger -sh "$user" 2>&1`;

# display HTML footer
print "</pre>";
print $q->end_html;
exit(0);

# ------------------------------------------------------------
# subroutine showError(reason) - build HTML error response
sub showError
{
    local ($msg) = $ [0];       # description of error
    print "<h1>Error</h1>\n";
    print "$msg";
    print "<p>Unable to safely obtain finger information.\n";
    print "<p>Please go back and correct the input supplied.\n";
    print $q->end_html;
    exit(0);
}
```

11.4 There are a number of deficiencies in the script shown in Figure 11.10a. Despite an attempt to quote the values of the supplied form fields subject, from, body, if any of these include a " character, the shell will assume it's the end of the quoted string, and interpret any shell meta-characters in the remainder. This can allow the execution of arbitrary commands, with out being displayed in the response web form. For example, including a value for "Your Email Address" like:

```
user@some.domain"; whoami; ls -al; echo "
```

will result in the response including the output from the commands: whoami; ls –al

A better version of this script, shown in Figure 11.2, with more stringent input checking (though the email address checking is simplified, and the body cannot contain " characters) is:

```
#!/usr/bin/perl
# comment.cgi - send comment to webmaster
# specify recipient of comment email
$to = "webmaster";

use CGI;
use CGI::Carp qw(fatalsToBrowser);
$q = new CGI;       # create query object

# display HTML header
print $q->header,
$q->start html('Comment Sent'),
$q->h1('Comment Sent');

# retrieve form field values and send comment to webmaster
$subject = $q->param("subject");
$from = $q->param("from");
$body = $q->param("body");

# validate the input information
# subject MUST NOT contain " or multiple lines
showError("The subject '$subject' contains illegal characters!")
    if ("$subject" =~ m:["\r\n]:);
# from MUST only contain characters valid in an email address
showError("The from address '$from' contains illegal characters!")
    unless ("$from" =~ m:^[- .=/\w][-  .=%!/@\w]*[- .=%\w]$:);
# body MUST NOT contain "
showError("The body '$body' contains illegal characters!")
    if ("$body" =~ m:":);

# generate and send comment email
system("export REPLYTO=\"$from\"; echo \"$body\" | mail -s \"$subject\"
$to");

# indicate to user that email was sent
print "Thankyou for your comment on $subject.";
print "This has been sent to $to.";

# display HTML footer
print $q->end html;
exit(0);

# ------------------------------------------------------------
# subroutine showError(reason) - build HTML error response due to reason
sub showError
{
    local ($msg) = $ [0];       # description of error
    print "<h1>Error</h1>\n";
    print "$msg";
    print "<p>Unable to safely send comment.\n";
    print "<p>Please go back and correct the input supplied.\n";
    print $q->end html;
    exit(0);
}
```

To remove the limitation on the contents of the mail body, the script would need to be further rewritten to open a pipeline to the mail program, and explicitly write the body contents into this pipeline, so they are not interpreted by the shell.

11.5 The major issue that may arise while using sequence numbers in TCP/IP implementations as identifiers and authenticators is that an attacker may easily authenticate itself if he/she can predict the next

sequence number. Thus, the root cause of the problem is that algorithms that generate random sequence numbers are highly predictable.

11.6 The goal of an XSS attack is always to execute malicious script in the victim's browser, there are a few different types of XSS attacks explained below.

In a **persistent XSS attack**, the attacker inserts a malicious string into the website's database. The website includes the malicious string from the database in the response and sends it to the victim.

1. The attacker uses one of the website's forms to insert a malicious string into the website's database.
2. The victim requests a page from the website.
3. The website includes the malicious string from the database in the response and sends it to the victim.
4. The victim's browser executes the malicious script inside the response, sending the victim's cookies to the attacker's server.

In a **reflected XSS attack**, the malicious string is part of the victim's request to the website. The website then includes this malicious string in the response sent back to the user.

1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website includes the malicious string from the URL in the response.
4. The victim's browser executes the malicious script inside the response, sending the victim's cookies to the attacker's server.

Finally, **DOM-based XSS attack** is a variant of both persistent and reflected XSS. In this attack, the malicious string is not actually parsed by the victim's browser until the website's legitimate JavaScript is executed.

1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website receives the request, but does not include the malicious string in the response.
4. The victim's browser executes the legitimate script inside the response, causing the malicious script to be inserted into the page.
5. The victim's browser executes the malicious script inserted into the page, sending the victim's cookies to the attacker's server.

Methods of preventing XSS attacks are encoding and validation. Encoding is to escape the user inputs so that the browser interprets it only as data, not as code. Validation is to filter the user input so that the browser interprets it as code without malicious commands.

11.7 No short answer is available, as it requires research to determine the current state of this field. Information on some fuzzing tools is available from the Fuzz Testing of Application Reliability (http://www.cs.wisc.edu/~bart/fuzz/) site.

11.8 No short answer is available, as it requires research to determine the current state of this field.

11.9 No detailed answer is available, as it depends on the system and shell used. The value of all environment variables can be displayed using the "env" command. A variable can be changed temporarily by changing the value of the corresponding shell variable, and then exporting it (details vary depending on the shell used). To change a value permanently for all subsequent logins on the system, the relevant shell startup file, either system-wide, or for a specific user, must be changed. Again the name and location of these files varies depending on the shell and system used.

11.10 No short answer is available, as this question requires experimentation with the supplied scripts.

CHAPTER 12 OPERATING SYSTEM SECURITY

ANSWERS TO QUESTIONS

12.1 The basic steps needed in the process of securing a system are:
- assess risks and plan the system deployment
- secure the underlying operating system and then the key applications
- ensure any critical content is secured
- ensure appropriate network protection mechanisms are used
- ensure appropriate processes are used to maintain security

12.2 **Hardening** process is a feature of systems security that includes planning, installation, configuration, update, and maintenance of the operating system and the key applications in use.

12.3 The basic steps needed to secure the base operating system are:
- install and patch the operating system
- harden and configure the operating system to adequately address the identified security needs of the system by:
  - removing unnecessary services, applications, and protocols
  - configuring users, groups and permissions
  - configuring resource controls
- install and configure additional security controls, such as anti-virus,
- host-based firewalls and IDS, if needed
- test the security of the basic operating system to ensure that the
- steps taken adequately address its security needs

12.4 Keeping all software as up to date as possible so important due to the continuing discovery of software and other vulnerabilities for commonly used operating systems and applications.

12.5 Automated patching:
Pros: minimizes window of opportunity for attackers when new vulnerabilities are found; is convenient, especially if automated.
Cons: patches sometimes introduce instability, especially on change controlled systems.

12.6 Many uninstall scripts fail to completely remove all components of a package. It has also been noted that disabling a service means that while it is not available as an initial point of attack, should an attacker succeed in gaining some access to a system, then disabled software could be re-enabled and used to further compromise a system. It is better for security if untrusted software is not installed, and thus not available for attacking the system at all.

12.7 Additional security controls that may be used to secure the base operating system include: anti-virus software, host-based firewalls, IDS or IPS software, and to white-list applications.

12.8 Backup is the process of making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks whereas Archive is the process of retaining copies of data over extended periods of time, being months or years, in order to meet legal and operational requirements to access past data.

12.9 The steps are used to maintain system security are:

- monitoring and analyzing logging information
- performing regular backups
- recovering from security compromises
- regularly testing system security
- using appropriate software maintenance processes to patch and update all critical software, and to monitor and revise configuration as needed

12.10 Effective logging helps ensure that in the event of a system breach or failure, system administrators can more quickly and accurately identify what happened and thus most effectively focus their remediation and recovery efforts. Logging information generated by the system, network, and applications can generate significant volumes of information. The key is to ensure that the correct data is captured in the logs, and administrators are able to appropriately monitor and analyse this data.

12.11 **Backup** is the process of making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks. Archive is the process of retaining copies of data over extended periods of time, being months or years, in order to meet legal and operational requirements to access past data.

12.12 Information on user accounts and group membership is usually stored in the /etc/passwd and /etc/group files. Modern systems also

have the ability to import these details from external repositories queried using LDAP, etc.

12.13 The "Windows Update" service and the "Windows Server Update Services" assist with the regular maintenance of Microsoft software. Although it can be disabled, it is recommended to be configured and used by the developers of the OS.

12.14 Programs that set user (setuid) to some user (even root, the superuser), or set group (setgid) to some group on Unix and Linux systems execute with the specified user's rights, or with access to resources belonging to the group, no matter which user executes them.

12.15 Linux systems primarily now use the **iptables** program to configure the **netfilter** kernel module. This provides comprehensive, though complex, stateful packet filtering, monitoring and modification capabilities.

12.16 Tripwire is a file integrity checking tool that maintains a database of cryptographic hashes of monitored files, and scans to detect any changes, whether as a result of malicious attack, or simply accidental or incorrectly managed update.

12.17 Unix and Linux systems provide a mechanism to run services in a **chroot jail**, which restricts the servers view of the file system to just a specified portion, and helps contain the effects of a given service being compromised or hijacked.

12.18 Users and groups in Windows systems may be stored and used locally, on a single system, in the Security Account Manager (SAM). It may also be centrally managed for a group of systems belonging to a domain, with the information supplied by a central Active Directory (AD) system using the LDAP protocol. Most organizations with multiple systems will manage them using domains.

12.19 There are major differences between the implementations of the discretionary access control models on Unix and Linux systems verses Windows systems. Unix and Linux systems use a small number of access rights for subjects being owner/group/other across nearly all resources/objects – this means it is a simple model, but because the same rights are used everywhere, their meaning can differ for different objects, and sometimes it is hard or impossible to specify some desired complex requirements. Windows systems use a much larger set of access rights, which differ for different types of objects – a more complex model, which can be harder to master, but

which may allow better specification of some desired complex requirements.

12.20 The mandatory integrity controls used for in Windows systems label all objects, such as processes and files, and all users, as being of low, medium, high or system integrity level. Then whenever data is written to an object, the system first ensures that the subject's integrity is equal or higher than the object's level, implementing a form of the Biba Integrity model.

12.21 On Windows, the privilege that overrides all ACL checks is the ability to backup the computer, which requires over-riding the normal access controls to obtain a complete backup.

12.22 On Windows systems, much of the application and service configuration information is centralized in the Registry, which forms a database of keys and values that may be queried and interpreted by applications on these systems.

12.23 Hypervisor is an entity that coordinates access between each of the guest operating systems and the actual physical hardware resources, such as CPU, memory, disk, network, and other attached devices. The hypervisor provides a similar hardware interface as that seen by operating systems that execute directly on the actual hardware.

12.24 Full virtualization systems are divided into native virtualization systems and hosted virtualization systems. In native virtualization systems, the hypervisor executes directly on the underlying hardware whereas in hosted virtualization systems, the hypervisor executes as just another application on a host OS that is running on the underlying hardware. Native virtualization systems are considered as more secure, as they have fewer additional layers. As hosted virtualization systems add additional layers with the host OS under, and other host applications beside, the hypervisor, they have increased security concerns.

12.25 The major concern with a hypervisor is that the access to the hypervisor should be restricted to authorized administrators only. If an attacker gains access to a hypervisor, he would be capable of accessing and monitoring activity in any of the guest OSs.

12.26 VM escape is one of the vulnerabilities that may exist in a virtualized environment. If such vulnerability exists in the virtualized system, the attacker can gain access of the hypervisor and subsequently access the resources of other guest OSs.

ANSWERS TO PROBLEMS

12.1 Setuid root programs cannot be completely eliminated from the operating systems because superuser privileges are required to access some of the resources on the system. A few examples include: managing user logins, permitting network services to bind to privileged ports, etc. Thus, eliminating the setuid root programs completely from the system would make it even more vulnerable to attacks.

12.2 The 'find' command allows for searches on the permissions, using the 'perm' option, '-perm +4000' will find all files with the SUID bit set. The full command, with output to file, is:

```
find / -perm +4000 > somelogfile
```

To find both SUID and SGID files (and to show the symbolic form of perms) you can use:

```
find / -perm +u+s -o -perm +g+s > somelogfile
```

Once a list of files has been generated, the system operator can check the modification time and other statistics (like size, owner) to ensure the file is the genuine article. However, this is still a manual process, prone to human error. System Security Tools (like "tripwire") could be applied to the files in the list to actively monitor for modifications.

12.3 To ensure secure remote access control, the services (made available for remote access) can be wrapped by using tcpd daemon, which then checks every request whether it is permitted by configured policy before accepting it and invoking the server program to handle it.

If the network servers are heavily loaded, this functionality is included in the connection management code by using the TCP Wrappers library

12.4

```
drwxrwx--- 3 david examiners 0 May 10 07:58 exams
-rwxrwx--- 1 david examiners 0 May 10 08:00 papers
```

12.5 Assets: website availability, system availability, local network integrity (integrity of other systems the attacker may reach via compromised web server), corporate data, customer data, website availability, e-commerce business activity (immediate revenue), company reputation (future revenue)
Vulnerability: buffer-overflow in Apache
Attack-vector: the worm “WorminatorX” (that multiple exploits may target the same vulnerability -- this is just the one we *know* about)
Attackers: competitors, thieves, identity thieves, website defacers (vandals), disgruntled ex-employees, the Byelorusan mob, etc. -- public web sites can be prey to any Internet-connected type of attacker

Likelihood of Occurrence: High (or synonyms thereof) -- Internet worms spread very far very quickly
Likely Impact: High -- complete exposure/loss of any or all affected assets to any potential attacker
Plausible Mitigations: Patch the Apache vulnerability; if no patch is available, protect Apache with SELinux or AppArmor; or run Apache in a chroot jail (Note that Apache already runs as an unprivileged user, by default -- presumably the WorminatorX vulnerability depends on some other privilege-escalation vulnerability)

12.6 It is very important to secure the booting process because attackers can gain control of this process and may change it to install a covert hypervisor.

Yes, it is required to limit which media the system must boot from, otherwise the attackers may just boot a system of their choice from external media in order to bypass the normal system access controls on locally stored data.

12.7 The reasons are mainly two folds:

a. many uninstall scripts fail to completely remove all components of a package.

b. disabling a software means that while it is not available as an initial point of attack, an attacker can still succeed in gaining some access to a system and then reenable the software to further compromise a system.

12.8 No, the attached hard disk will be allocated to only one guest operating system at a time. If the number of network interfaces is not enough, the hypervisor can create "virtual network interfaces" for every guest operating system and then route the traffic as required.

12.9 Unix and Linux systems use a small number of access rights for subjects being owner/group/other across nearly all resources/objects – this means it is a simple model, but because the same rights are used everywhere, their meaning can differ for different objects, and sometimes it is hard or impossible to specify some desired complex requirements.

Windows systems use a much larger set of access rights, which differ for different types of objects – a more complex model, which can be harder to master, but which may allow better specification of some desired complex requirements.

An increase in security features is desirable provided the system administrator is competent and completely understands the purpose and use of each of the security features, how they interact with each other, and which feature is best used in specific circumstances.

If the administrator was not as familiar with all the complex security features then there would be a benefit in having less security features available. This is because having less to understand means less chance of making mistakes.

12.10 Limiting the hypervisor access to authorized administrators only is mandatory to prevent an attacker from gaining access to other guest OS's. Moreover, if the hypervisor supports remote administration also, the impact would be even more catastrophic if the attacker gains an access to the hypervisor