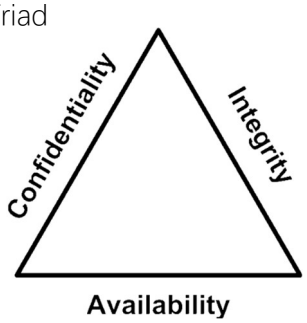
CH01 Overview

ZJU 2020

The CIA Triad



The CIA Triad

- Confidentiality: this term covers two related concepts:
  - Data confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.
  - Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
- Integrity: this term covers two related concepts:
  - Data integrity: Assures that information and programs are changed only in a specified and authorized manner.
  - System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.
- Availability: Assures that systems work promptly and service is not denied to authorized users.

CIA Triad Examples

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.	An unencrypted CD-ROM or DVD is stolen.	
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines and Networks	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Security Concepts

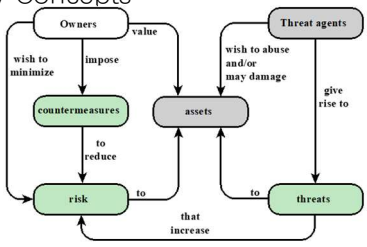


Figure 1.2 Security Concepts and Relationships

Vulnerabilities, Threats and Attacks

- Categories of vulnerabilities
  - Corrupted (loss of integrity)
  - Leaky (loss of confidentiality)
  - Unavailable or very slow (loss of availability)
- Threats
  - Capable of exploiting vulnerabilities
  - Represent potential security harm to an asset
- Attacks (threats carried out)
  - Insider vs. outsider
  - Passive vs. active

Passive Attacks

- In passive attacks, attacker attempts to learn or make use of information from the system but does not affect system resources
  - Eavesdropping on, or monitoring of, transmissions
  - Goal of attacker is to obtain information that is being transmitted
- Two categories :
  - Leaking of message content
  - Traffic analysis: attacker infers information from network traffic patterns, **even though message content is not leaked (e.g., with encryption)**

Active Attacks

- In active attacks, attacker attempts to alter system resources or affect their operation.
  - Involve some modification of the data stream or the creation of a false stream.
- Four categories:
  - Replay: attacker captures a message and subsequent retransmits it to produce an unauthorized effect.
  - Masquerade: one entity pretends to be a different entity.
  - Modification of messages: some portion of a message is altered, or messages are delayed or reordered.
  - Denial of Service: **prevents or inhibits the normal use of the target system.**

Countermeasures

- Means used to deal with security attacks
  - Prevent, Detect, Recover
  - Goal is to minimize residual level of risk to the assets

Attack Surface

- An attack surface consists of the reachable and exploitable vulnerabilities in a system, including:
  - Network Attack Surface
    - Vulnerabilities over an enterprise network, wide-area network, or the Internet
    - Including network protocol vulnerabilities, such as those used for DoS attacks
  - Software Attack Surface
    - Vulnerabilities in application, utility, or operating system code
  - Human Attack Surface
    - Social engineering, human error, and trusted insiders

Security Risk

- Defense in depth is a concept used in information security in which multiple layers of security controls (defense) are placed throughout the system.
- Security risk can be determined by defense in depth layering (shallow or deep) and attack Surface (small or large).

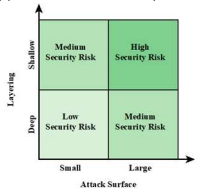


Figure 1.4 Defense in Depth and Attack Surface

CH02 Cryptographic Tools

ZJU 2020

Outline

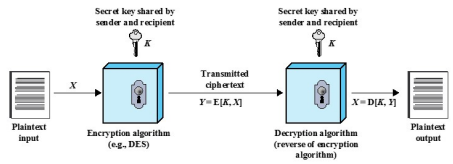
- Symmetric encryption
- Public-key cryptography
- Message authentication and hash functions
- Digital signatures
- Random and pseudorandom numbers

Terminology

- Plaintext
  - Also called cleartext
- Ciphertext
  - Scrambled message produced as output
- Encryption algorithm
  - Transforms plaintext to ciphertext
- Decryption algorithm
  - Transforms ciphertext to plaintext

Symmetric Encryption

- Also called secret-key cryptography, for protecting confidentiality
  - Sender and receiver must share the same secret key
  - Need a strong encryption algorithm



Attacking Symmetric Encryption

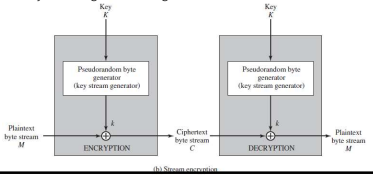
- Cryptanalytic Attacks
  - Exploits the characteristics of the algorithm to attempt to recover plaintext or secret key
  - Rely on:
    - Nature of the algorithm
    - Some sample plaintext-ciphertext pairs
    - Some knowledge of the general characteristics of the plaintext, e.g., letter “e” and word “the” are common in English texts
- Brute-Force Attacks
  - Try all possible keys on some ciphertext until an intelligible plaintext is obtained

Block & Stream Ciphers

- Block Cipher
  - Processes input data one block at a time
  - Produces an output block for each input block
- Stream Cipher
  - Processes the input elements continuously, producing output one element at a time
  - One element may be 1 bit, 1 Byte, or more than 1 Byte
  - Faster than block ciphers

An Example Stream Cypher

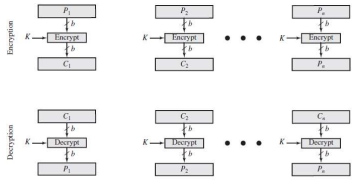
- A stream cypher that stream cipher that operates one bit at a time:
  - Sender and receiver share a secrete key K, which can be input to a pseudorandom byte generator that produces a pseudorandom stream of bytes, called a keystream $KS(k$ in the figure).
  - The plaintext is XOR'ed with KS bit-by-bit to produce the cyphertext: $C_i = M_i \; XOR \; KS_i$
  - The cyphertext is XOR'ed with the same keystream KS to recover the plaintext $M_i = C_i \; XOR \; KS_i$
  - This relies on sender and receiver sharing a secrete key K and using the same key stream generator algorithm.



(b) Stream encryption

Block Cipher: Electronic CodeBook (ECB)

- Each plaintext block P_i (e.g., 64 or 128 bits) is encoded independently using the same key K
  - Attacker may exploit regularities in the plaintext to perform cryptoanalysis, since same plaintext block generates same cyphertext block. If it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with.

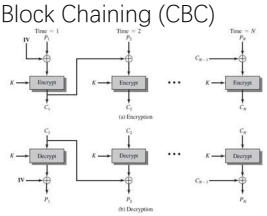


(a) Block cipher encryption (electronic codebook mode)

Block Cipher: Cipher Block Chaining (CBC)

- Input to the encryption algorithm is the XOR of the next plaintext block P_i and the preceding ciphertext block C_{i-1}.
  - P_i and C_{i-1} have no fixed relationship, removing the regularities in the plaintext for ECB, hence more secure.
  - Uses the properties
    - $C_{i-1}\ XOR\ C_{i-1} = 0$ (any string XOR'ed with itself is all 0's).
    - $0\ XOR\ P_i = P_i$ (0 XOR'ed with any string is the same string).

```
  0101
XOR 0011
  = 0110
```



(a) Encryption

(b) Decryption

Figure 20.7 **Cipher Block Chaining (CBC) Mode**

For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block. To see that this works, we can write

$$C_i = E(K, [C_{i-1} \oplus P_i])$$

where $E[K, X]$ is the encryption of plaintext X using key K, and $\oplus$ is the exclusive-OR operation. Then

$$D(K, C_i) = D(K, E(K, [C_{i-1} \oplus P_i]))$$
$$D(K, C_i) = C_{i-1} \oplus P_i$$
$$C_{i-1} \oplus D(K, C_i) = C_{i-1} \oplus C_{i-1} \oplus P_i = P_i$$

which verifies Figure 20.7b.

Comparison of 3 Symmetric Encryption Standards (Block Ciphers)

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard
AES = Advanced Encryption Standard

Data Encryption Standard (DES)

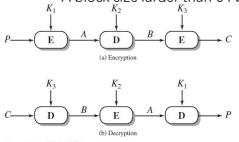
- DES Uses 64-bit plaintext block and 56-bit key to produce a 64-bit ciphertext block
  - Key length of 56 bits is too short, and subject to brute-force attacks.
  - Should no longer be used in production systems.

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard
AES = Advanced Encryption Standard

Triple DES (3DES)

- Repeats DES three times using either 2 or 3 unique keys, with total key length of 112 or 168 bits
- Pros:
  - Key length of 168 bits removes the vulnerability to brute-force attacks
- Cons:
  - Performance is slow: three times as many calculations as DES.
  - A block size larger than 64 bits is desirable for efficiency and



(a) Encryption

(b) Decryption

Figure 20.2 **Triple DES**

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard
AES = Advanced Encryption Standard

Advanced Encryption Standard (AES)

- Due to performance and block size issues with 3DES, NIST called for proposals for a new **Advanced Encryption Standard** (AES) in 1997, and made a selection in 2001.
  - 128-bit block size and key length of 128, 192 or 256 bits
- **AES is now widely deployed commercially.**

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard
AES = Advanced Encryption Standard

Time Required for Brute-Force Attack

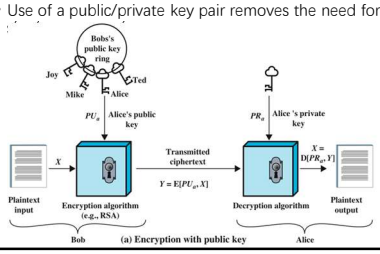
Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Public-Key Cryptography

- Proposed by Diffie and Hellman in 1976
  - Based on modular arithmetic functions
- Asymmetric: uses two separate keys
  - Public key and private key

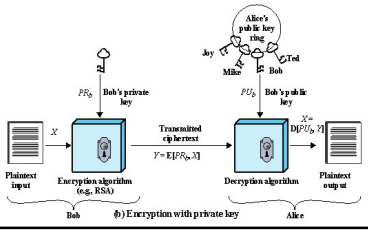
Public-Key Crypto for Confidentiality

- Sender encrypts data using the receiver's public key
- Receiver decrypts data using his own private key
  - Use of a public/private key pair removes the need for



Public-Key Crypto for Integrity

- Sender encrypts data using his or her private key
- Receiver, or anyone else, can decrypt the message using sender's public key.
- There are more efficient methods based on MAC or crypto hash function (discussed later).



Public-Key Crypto Algorithms and Protocols

- RSA (Rivest-Shamir-Adelman)
  - Key generation, encryption, decryption
- Elliptic Curve Cryptography (ECC)
  - Lightweight public key algorithm for embedded and IoT devices
- Diffie-Hellman
  - Key exchange protocol used to establish a shared secret between two parties, e.g., a secret key for symmetric encryption

Diffie-Hellman

- Alice and Bob agree on:
  - A sufficiently large prime number p
  - A base number $g < p$
- Alice chooses a secret number a and sends to Bob
  - $A = g^a \bmod p$
- Bob chooses a secret number b and sends to Alice
  - $B = g^b \bmod p$
- Alice computes $S_1 = B^a \bmod p$
- Bob also computes $S_2 = A^b \bmod p$
- We can easily show that $S_1 = S_2$, and this is their shared secret key
  - $S_1 = B^a \bmod p = g^{ab} \bmod p = A^b \bmod p = S_2$
- p, g, A, B are all public; only Alice knows secret a; only Bob knows secret b. At the end, Alice and Bob have a shared secret $S_1 = S_2$.

Requirements for Public-Key Crypto

- Computationally easy to
  - create key pairs
  - encrypt/decrypt messages using either public or private key
- Computationally infeasible to
  - determine private key from public key
  - recover cleartext without key
- Either key can be used for each role (public/private key)

Message Authentication

- Verifies received message is authentic
  - Its content has not been altered (authentic content), and it is from the alleged sender (authentic source)
- Can use encryption (symmetric or public-key)
  - Sender encrypts the message, receiver decrypts it
  - But inefficient for large messages

Message Authentication without Encryption

- Two approaches:
  - Message Authentication Code (MAC)
    - Computes $MAC_M = F(K, M)$ for input message M, with secret key K, as a fixed-length output (e.g., 16 or 32 bits)
  - Cryptography Hash Function
    - Computes a hash $H(M)$ for input message M of any size, with fixed-length output (e.g., 128-512 bits)
    - Also called one-way hash
- MAC needs a shared secret key K; crypto hash does not.

Message Authentication Code (MAC)

- Sender and receiver share a secret key K. Sender calculates MAC as a complex function of message and key: $MAC_M = F(K, M)$.
- Receiver recomputes the MAC with $F(K, receivedM)$, and compares it with the received MAC_M. If they match, then message is authenticated: $receivedM = M$, and it is from the alleged sender with secrete key K.

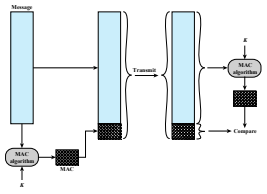


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

MAC Explanations

- If only the sender and receiver know the secret key K, and if the received MAC matches the receiver-computed MAC, then
- 1. The receiver is assured that the message has not been altered.
  - If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker does not know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
- 2. The receiver is assured that the message is from the alleged sender.
  - Because no one else knows the secret key K, no one else could prepare a message with a proper MAC.
- 3. If the message includes a sequence number (such as in TCP)
  - Then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Crypto Hash Function

- The crypto hash function accepts a variable-size message M as input and produces a fixed-size hash value, also called message digest, $H(M)$ as output.

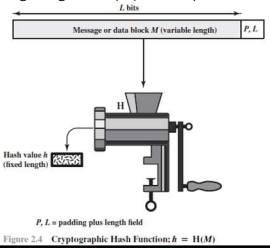


Figure 2.4 **Cryptographic Hash Function: $h = H(M)$**

Apply crypto hash function to generate hash value $H(M)$, which is encrypted using either (a) symmetric encryption with shared secret key K, or (b) public-key encryption with sender's private key PR_a. Receiver recomputes $H(receivedM)$, and compares it with the received and decrypted $H(M)$. If they match, then message is authenticated.

(c) Keyed hash: sender and receiver share a secret key K. Apply hash function $H()$ on the concatenation of secret key K and the message to generate Message Digest $MD_M = H(K||M||K)$. Receiver recomputes $H(K||receivedM||K)$ and compares it with received MD_M.

(a)(b) are more efficient than MAC approach, since encryption/decryption is applied to the hash, not the entire message. (c) avoids encryption/decryption altogether.

In (b), the encrypted hash is called the digital signature.

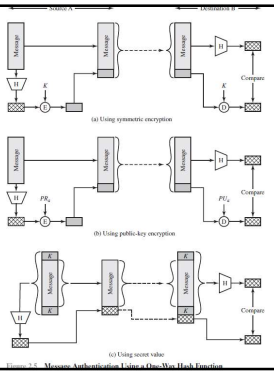


Figure 2.5 Message Authentication Using a One-Way Hash Function

Crypto Hash Function Requirements

- One-way function: Easy to compute $H(x)$ given x, but computationally infeasible to find x given $H(x)$.
- Weak collision resistance: given x, computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$
  - Otherwise, attacker can substitute a fake message y for a given authentic message x
- Strong collision resistance: computationally infeasible to find any pair $(x, y), x \neq y$, such that $H(x) = H(y)$
  - Otherwise, attacker Bob can generate two messages x and y with same $H(x) = H(y)$. Msg x is an IOU (欠条) for $10, and Msg y is an IOU for $100. Bob sends x to Alice, who computes $H(x)$ and encrypts it with her private key as signature of message x. Bob can use Msg y in conjunction with $H(x)$ as proof that Alice owes him $100 instead of $10.

Additional Hash Function Applications

- UNIX password checking
  - password hash values are stored in the file /etc/passwd. When user tries to log in, system computes hash of user-entered password and compares with stored password hash (discussed in CH03)
- Intrusion detection
  - Hash value $H(F)$ for a file F is stored in a secure location to detect any alteration of file contents

Random Numbers

- Used to generate:
  - Keys for public-key algorithms
  - Stream key for symmetric stream cipher
  - Symmetric key for use as a temporary session key or in creating a digital envelope
  - Handshaking to prevent replay attacks
  - Session key

Random Number Requirements

- Randomness
  - Uniform distribution
    - Frequency of occurrence of each of the numbers should be approximately the same
  - Independence
    - No one value in the sequence can be inferred from the others
- Unpredictability
  - Each number is statistically independent of others in the sequence, so future elements of the sequence cannot be predicted based on past elements

Pseudorandom vs. Random Numbers

- Pseudorandom numbers are generated with deterministic algorithms with random seeds:
  - The same seed results in the same sequence of random numbers
  - The sequence produced are not truly statistically random, but may pass **many reasonable tests of randomness**
- True random number generator (TRNG):
  - Uses a nondeterministic source to produce randomness
  - Most operate by measuring unpredictable natural processes
    - e.g. radiation, gas discharge, leaky capacitors

CH03 User Authentication

ZJU 2020

Outline

- Introduction
- Password-based authentication
- Token-based authentication
- Biometric authentication
- Remote user authentication

User Authentication

- NIST definition: "The process of establishing confidence in user identities that are presented electronically to an information system."
- It is the fundamental building block and first line of defense, and the basis for access control and user accountability.

Four Means of Authentication

- Something the individual knows
  - Password, PIN, answers to prearranged questions
- Something the individual possesses (token)
  - Smartcard, electronic keycard, physical key
- Something the individual is (static biometrics)
  - Fingerprint, retina, face
- Something the individual does (dynamic biometrics)
  - Voice pattern, handwriting, typing rhythm, modern face recognition

Multifactor Authentication

- Uses multiple methods (factors) in combination
  - Bank card in combination with a PIN
  - Finger print in combination with a PIN
  - Password in combination with a code via SMS to your mobile phone (验证码)
- May be triggered by unusual activities
  - e.g., if system detects (via IP address or GPS) that you are not at your usual location, then use multi-factor authentication

Risk Assessment for User Authentication

- Potential impact
  - Three levels of potential impact on organizations or individuals should there be a breach of security: low, moderate, high.
- Assurance Level determined by potential impact
  - Describes an organization's degree of certainty that a user has presented a credential that refers to his or her identity.
  - e.g., If the potential impact is low, an assurance level of 1 is adequate; if the potential impact is high, an assurance level of 4 is needed.

	Assurance Level Impact Profiles			
Potential Impact Categories for Authentication Errors	1	2	3	4
Inconvenience, distress, or damage to standing or reputation	Low	Mod	Mod	High
Financial loss or organization liability	Low	Mod	Mod	High
Harm to organization programs or interests	None	Low	Mod	High
Unauthorized release of sensitive information	None	None	Low	Mod/ High
Personal safety				
Civil or criminal violations	None	Low	Mod	High

Outline

- Introduction
- Password-based authentication
- Token-based authentication
- Biometric authentication
- Remote user authentication

Password Authentication

- User provides user ID/password, and system compares password with the one stored for the specified user ID. How to store passwords?
- Method 1: store a list of passwords, one for each user, in a file readable only by the root/admin user. Drawbacks:
  - The admin should not know the user's passwords
  - If permissions are set incorrectly, or an attacker gets in, and the password file is exposed
- Method 2: store a list of hash values, in **/etc/password file**, computed by a one-way hash function H applied to each password pwd concatenated with a salt H(pwd || salt).
  - Used in almost all systems today
  - Example: user1 and user2 both have password "password123"

Username	Salt value	String to be hashed	Hashed value = SHA256 (Salt value + Password)
user1	E1F53135E559C253	**password123**E1F53135E559C253	72AE25495A7981C40622D49F9A52E4F1565C90F048F59027BD9C8C8900D5C3D8
user2	84B03D034B409D4E	**password123**84B03D034B409D4E	B4B6603ABC670967E99C7E7F1389E40CD16E78AD38EB1468EC2AA1E62B8BED3A

The UNIX Password Scheme

- When user Bob creates a new account or sets a new password, his user ID Bob, the salt and the hash value H(pwd[Bob] || salt[Bob]) are stored in /etc/password file.
- When Bob attempts to log in, he enters his user ID Bob and Password. The system uses the user ID Bob to retrieve salt[Bob], and computes H(Password || salt[Bob]). If the result matches the stored value H(pwd[Bob] || salt[Bob]), then Bob is authenticated.
- The hashing algorithm is designed to be relatively slow to make it difficult for the attacker to try many passwords, but not perceptible to a legitimate user who logins.

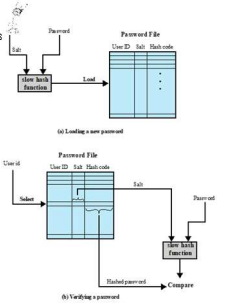


Figure 3.3 UNIX Password Scheme

Password Cracking

- Dictionary attacks
  - Develop a large dictionary of possible passwords and try each against the password file
    - Numerous leaked plaintext passwords are available online.
  - Each password is hashed using each user's salt value, and then compared to his stored hash value in the password file.
    - Try popular passwords first.
- Rainbow table attacks
  - To trade off space for time, attacker pre-computes a huge tables of hash values for all password/salt combinations, to avoid computing hash values during attack.

Purpose of the Salt

- It prevents duplicate passwords from being visible in the password file. Even if two users have the same password, they have different salt values, so their password hashes are different.
- It makes it difficult to find out whether a person with accounts on two or more systems has used the same password on all of them.
- It greatly increases the difficulty of dictionary or rainbow table attacks. Assuming attacker obtains a copy of the password file.
  - For dictionary attack: to crack a single password of any user.
    - Without the salt: for each password guess attempt[i] in the dictionary, he computes H(attempt[i]), then checks whether that hash appears anywhere in the password file. The likelihood of a match, i.e. cracking one of the passwords, increases with the number of passwords in the file.
    - With the salt: each password is hashed and compared separately for each salt, i.e., for each password attempt[i], he computes H(attempt[i] || salt[A]) for user A, compares against password hash of user A; computes H(attempt[i] || salt[B]) for user B, compares against password hash of user B, and so on.
  - For rainbow table attack: size of the rainbow table is increased by a factor of 2^b for a salt of length b bits.

Salt Quiz

- Q: Assuming attacker obtains a copy of the password file, and he wants to use dictionary attack to crack the password of a *specific* user Bob. Does the salt increase the difficulty of this attack?
- ANS: No.
  - For each password guess attempt[i] in the dictionary, he computes H(attempt[i] || salt[Bob]) for a single salt[Bob], then compares it to Bob's stored hash value in the password file. Computing the hash adds a little computation delay, but it is not significant.
  - The salt greatly increases difficulty of cracking password for *any* user in the password file with dictionary or rainbow table attack, which is more likely to find a match than cracking the password of any specific user.

The Hash Function

- Recommended hash function is based on MD5
  - Password length is unlimited
  - Uses 48-bit salt to create 128-bit hash value
  - Uses an inner loop with 1000 iterations to achieve slowdown
- OpenBSD uses Blowfish block cipher based hash algorithm called Bcrypt
  - Most secure version of Unix hash/salt scheme
  - Uses 128-bit salt to create 192-bit hash value

Shadow Password Scheme

- For added security, password hashes are typically not stored in /etc/passwd, but in a separate file /etc/shadow, which is only readable by privileged users, e.g., root.
  - /etc/passwd: row format "User Name : x : User ID : Group ID : Home Dir : Shell"
    - 2nd entry is "x", indicating that the password hash is stored elsewhere.
  - /etc/shadow: 2nd entry in /etc/shadow is either password hash for the User Name, or *, indicating that this user has no password.
    - Command "sudo head -5 /etc/shadow": run as root, and display the top 5 lines in /etc/shadow.

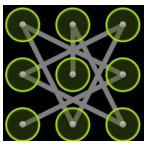
```
Terminal
seed@VM$ head -5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
seed@VM$
seed@VM$ sudo head -5 /etc/shadow
root:$6$NrF4601p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEEO
5wj8aU3GRHW2BaodUn4K3vgyEjwPspr/kqzAqtcu.:17400:0:99999:7:::
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
sync:*:17212:0:99999:7:::
```


Password Selection Strategies

- User education
  - Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords
- Computer generated passwords
  - Users have trouble remembering them, often used for resetting password, and user has to change it immediately.
- Reactive password checking
  - System periodically runs its own password cracker to find guessable passwords
- Complex password policy
  - Forcing users to pick stronger passwords

Unlock Patterns as Passwords

- Attackers can exploit user biases in unlock patterns
  - There is a bias in starting near the top left of the screen
  - The ease of moving from current to next point introduces bias
  - Would anyone choose such a secure, but complex pattern?



Outline

- Introduction
- Password-based authentication
- Token-based authentication
- Biometric authentication
- Remote user authentication

Magnetic Stripe Card

- A magnetic stripe can store only a simple security code, which can be read (and reprogrammed/cloned) by an inexpensive card reader.
- "2017年5月1日起，银行将全面关闭芯片磁条复合卡的磁条交易。"
  - From 百度百科
- Used in low-impact use cases, e.g., hotel room key

Smart Cards

- Contains an embedded microprocessor with memory; cannot be easily cloned like a magnetic stripe card
- Interface:
  - Contact vs. contactless
- Challenge-response authentication protocol:
  - Computer system generates a challenge, e.g., a nonce; the smart card generates a response, e.g., by encrypting the nonce with its private key.

Smart Card/Reader Exchange

- Figure illustrates the typical interaction between a smart card and a card reader. When the card is inserted into the reader:
- Card sends a reset to the reader
- Reader sends Answer To Reset (ATR) message to card, which defines the parameters and protocols that the card can use and the functions it can perform.
- Protocol Type Selection (PTS) messages are used to negotiate a protocol
- Reader and card can now exchange data (APDU) to perform the desired function.

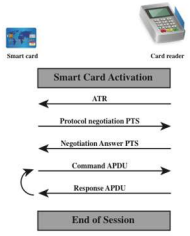


Figure 3.6 Smart Card/Reader Exchange

Outline

- Introduction
- Password-based authentication
- Token-based authentication
- Biometric authentication
- Remote user authentication

Biometric Authentication

- Authentication based on unique physical characteristics
  - Face, fingerprints, retina, voice…
  - Face recognition today is very accurate (Fig. 3.8 is out of date)
- Possible attacks
  - Copied fingerprints, fake face, fake voice…

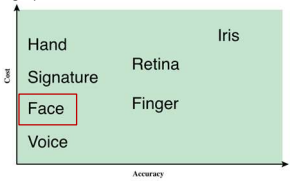


Figure 3.8 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes

- Enrollment followed by either
  - Verification: the user enters a name (PIN), and uses a biometric sensor. The system identifies the user through name (PIN), and compares her biometric feature to the template stored for this user.
  - Identification: the user uses a biometric sensor, but does not enter name (PIN). The system compares her biometric feature to the set of stored templates for all users, and if there is a match, then this user is authenticated.

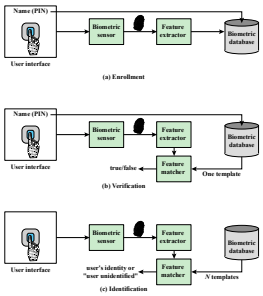


Figure 3.9 A Generic Biometric System. Enrollment creates an association between a user and the user's biometric characteristics. Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

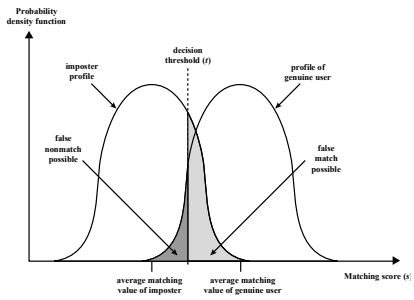


Figure 3.10 Profiles of a Biometric Characteristic of an Imposter and an Authorized Users In this depiction, the comparison between presented feature and a reference feature is reduced to a single numeric value. If the input value (s) is greater than a preassigned threshold (t), a match is declared.

Fig. 3.10

- The matching score s is a random variable with Probability Density Function (PDF) typically forming a bell curve.
  - For example, in the case of a fingerprint, results may vary due to sensor noise; changes in the print due to swelling, dryness, and so on; finger placement; and so on.
- The imposter should have a much lower matching score but again will exhibit a bell-shaped PDF.
- The range of matching scores produced by two individuals, one genuine and one an imposter, compared to a given reference template, are likely to overlap.
- A threshold value is selected thus that if the presented value s ≥ t a match is assumed, and for *s <t, a mismatch is assumed.*
- The area of each shaded area to the right of *t* indicates probability for which *a false match (*accepting an imposter) *is* possible; the shaded part to the left indicates probability for which a *false non-match* (rejecting he genuine user) is possible.
- Designer can set the threshold to tradeoff between security and convenience
  - Higher threshold: decrease in false match rate; increase in false non-match rate; more secure and less convenient
  - Lower threshold: increase in false match rate; decrease in false non-match rate; less secure and more convenient

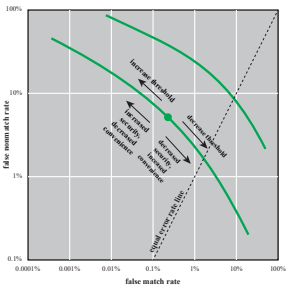


Figure 3.11 Idealized Biometric Measurement Operating Characteristic Curves (log-log scale)

Outline

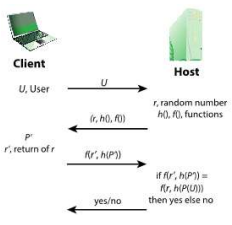
- Introduction
- Password-based authentication
- Token-based authentication
- Biometric authentication
- Remote user authentication

Remote User Authentication

- Authentication over the Internet
- Security threats include:
  - Eavesdropping, capturing a password, replaying an authentication sequence that has been observed
- Generally rely on some form of a challenge-response protocol to counter threats

Challenge-Response Protocol for Password Authentication

- User first transmits his or her identity U to the remote host.
- Host generates a random number r, *called a* ***nonce,*** *and* returns it to the user. In addition, the host specifies two functions, $h()$ and $f()$, to be used in the response. This transmission from host to user is the challenge.
- User sends the response $f(r', h(P'))$, *where* $r' = r$, and P' is user's password. The function $h()$ is a hash function, so that the response consists of the hash function of user's password combined with the random number using the function $f()$.
- Host stores the hash of each user's password, $h(P(U))$ for user U. When the response arrives, the host compares the incoming $f(r', h(P'))$ to the calculated $f(r, h(P(U)))$. If the quantities match, the user is authenticated $P' = P$.

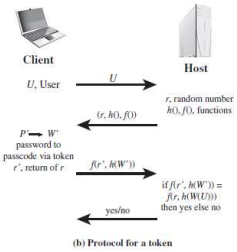


Benefits of Challenge-Response Protocol

- This scheme defends against several forms of attack.
  - Host stores a hash code of the password, not the clear text. This secures the password from intruders into the host system. (Same as UNIX password scheme)
  - Password hash is not transmitted directly, but rather a function $f()$ in which the password hash is one of the arguments. Thus the password hash cannot be captured during transmission.
  - Use of a random number (nonce) as one of the arguments of $f()$ defends against a replay attack, in which an adversary captures the user's transmission and attempts to log into a system by retransmitting the user's messages.

Other Challenge-Response Protocols: Token

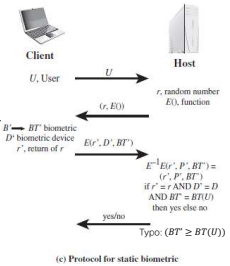
- User side: the token (e.g., U盾) provides a passcode W. The token either stores a static passcode or generates a one-time random passcode. The user activates the passcode by entering a password P. This password is shared only between the user and the token and does not involve the remote host. The token responds to the host with the quantity $f(r, h(W))$.
- Host side: For a static passcode, the host stores the hashed value $h(W(U))$; for a dynamic passcode, the host generates a one-time passcode (identical to that generated by the token) and takes its hash.



(b) Protocol for a token

Other Challenge-Response Protocols: Static Biometric

- Host sends nonce r and identifier for an encryption function $E()$
- User controls a biometric device, which generates a biometric template BT', **a digital representation of** the user's biometric B', and returns the ciphertext $E(r', D', BT')$, where D' is ID for this particular biometric device.
- Host decrypts the message.
  - Nonce must match ($r' = r$).
  - Device ID must be in the host database ($D' = D$)
  - Matching score between BT' and the stored template $BT(U)$ must exceed a predefined threshold ($BT' \geq BT(U)$)



(c) Protocol for static biometric

Authentication Security Issues

- Eavesdropping
  - Adversary attempts to learn the password by, say, looking over the shoulder of victim, or sniffing network packets
- Host Attacks
  - Directed at the user file at the host where passwords, token passcodes, or biometric templates are stored
- Replay
  - Adversary repeats a previously captured user response
- Client Attacks
  - Adversary attempts to achieve user authentication without access to the remote host or the intervening communications path
- Trojan Horse
  - An application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric
- Denial-of-Service
  - Attempts to disable a user authentication service by trying multiple authentication attempts, e.g., the account may be locked after N unsuccessful attempts.

CH04 Access Control

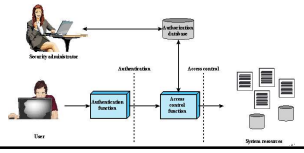
ZJU 2020

Outline

- Introduction
- Discretionary Access Control (DAC)
  - UNIX file access control
- Role-based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)
- Bank RBAC system

Authentication and Access Control

- Authentication: verifies the credentials of a user to determine if the user is permitted to access the system at all.
  - e.g., is the current user trying to log in with John's user ID really John himself?
- Access control: mediates between a user and system resources (files and databases), and determines if the specific requested access is permitted.
  - e.g., should John's process making a request to read a certain file be allowed to do so?



Access Control Policies

- Discretionary Access Control (DAC)
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
  - A user may grant access rights to another user
- Mandatory Access Control (MAC)
  - Controls access based on comparing security labels with security clearances
  - A user may not grant access rights to another user
  - Covered in L5-CH27-Trusted Computing.
- 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
- Attribute-Based Access Control (ABAC)
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

Subjects, Objects, and Access Rights

- Subject
  - An active entity capable of accessing objects
- Object
  - A resource to which access is controlled
- Access right
  - The way in which a subject may access an object
  - Could include: Read, Write, Execute, Delete, Create, Search

Outline

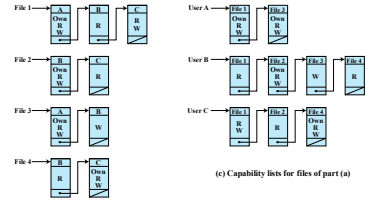
- Introduction
- Discretionary Access Control (DAC)
  - UNIX file access control
- Role-based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)
- Bank RBAC system

Discretionary Access Control (DAC)

- The Access Control Matrix
  - Each entry indicates the access rights of a subject for an object
  - A user may enable another user to access some resource

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

- An access matrix is usually sparse and is implemented by decomposition in one of two ways: access control list (used by Linux), or capability list (used by L4).



(b) Access control lists for files of part (a)

(c) Capability lists for files of part (a)

Figure 4.2 Example of Access Control Structures

Extended Access Control Matrix

- Objects may be files, processes, disk drives, or other subjects

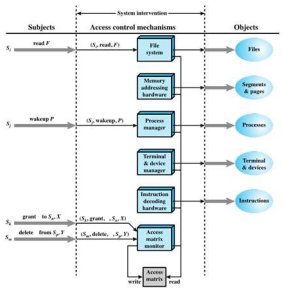
		OBJECTS								
		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

Figure 4.3 Extended Access Control Matrix

Architecture of Access Control

- Every access by a subject to an object is mediated by the controller for that object, and that the controller's decision is based on the current contents of the matrix.
- Certain subjects can modify the access matrix. A request to modify the access matrix is treated as an access to the matrix, with the individual entries in the matrix treated as objects. Such accesses are mediated by an access matrix controller, which controls updates to the matrix.



Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\begin{Bmatrix}\alpha*\\ \alpha\end{Bmatrix}$ **to** S, X	'α*' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha*\\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	**grant** $\begin{Bmatrix}\alpha*\\ \alpha\end{Bmatrix}$ **to** S, X	'owner' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha*\\ \alpha\end{Bmatrix}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	delete α from $A[S, X]$
R4	$w \leftarrow$ **read** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	copy $A[S, X]$ into w
R5	**create object** X	None	add column for X to A; store 'owner' in $A[S_0, X]$
R6	**destroy object** X	'owner' in $A[S_0, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_0, S]$	delete row for S from A; execute **destroy object** S

Table 4.2

Access Control System Commands

Table 4.2 Explanations

- The first three rules deal with transferring, granting, and deleting access rights.
- R1: If the entry α* exists in A[S_0, X], then S_0 has access right α to subject X and, because of the presence of the copy flag, can transfer this right, with or without copy flag, to another subject. A subject would transfer the access right without the copy flag if there were a concern that the new subject would maliciously transfer the right to another subject that should not have that access right.
- R2 states that if S_0 is designated as the owner of object X, then S_0 can grant an access right to that object X for any other subject S.
- R3 permits S_0 to delete any access right from any matrix entry in a row for which S_0 controls the subject and for any matrix entry in a column for which S_0 owns the object.
- R4 permits a subject to read that portion of the matrix that it owns or controls.
- The remaining rules govern the creation and deletion of subjects and objects.
- R5: any subject can create a new object, which it owns, and can then grant and delete access to the object.
- R6: the owner of an object can destroy the object, resulting in the deletion of the corresponding column of the access matrix.
- R7 enables any subject to create a new subject; the creator owns the new subject and the new subject has control access to itself.
- R8 permits the owner of a subject to delete the row and column (if there are subject columns) of the access matrix designated by that subject.

DAC Quiz

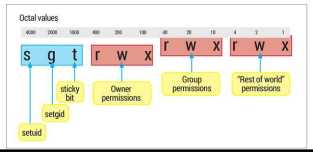
- Assume user Alice is the owner of file foo, and can choose to grant read access right to foo to another user Bob, but can prevent Bob from further granting read access right to other users
  - "grant r to {Bob, foo}" instead of "grant r* to {Bob, foo}"
- Q: Does this ensure that a third user, Charlie, can never read data from file foo?
- ANS: No. Bob cannot issue "grant r to {Charlie, foo}", but he can make a copy of foo, foo2, so now he is the owner of foo2 and can issue "grant r to {Charlie, foo2}"
  - If Alice issues "grant w to {Bob, foo}", the Charlie cannot possibly write to foo.

DAC Example: UNIX File Access Control

- For a given file, there are 3 classes of subjects, with different access rights for each class:
  - Owner: the file's owner
  - Group: a user may belong to multiple groups
  - Other: all other users, who are neither the owner, nor members of any group relevant to the file
  - Each class of subjects has 3 types of permission: read (r), write (w), execute (x)

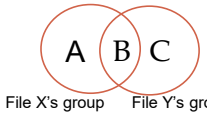
UNIX File Permission Bits

- Each UNIX user is assigned a unique user ID (root has ID 0). A user is also a member of a primary group, and possibly a member of other groups, each identified by a group ID.
- When a file is created, its owner is user ID of its creator, and its group ID is its creator's primary group, but they can be changed later.
  - e.g., command "chown root passwd" sets owner of file "passwd" to root.
- 12 permission bits; 9 bits specify read, write, and execute permission for user (owner), group, and all other users
  - e.g., command "chmod 640 file1" sets file1's permissions to be readable and writable by owner, readable by group. (Decimal number 640 is 110 100 000 in binary)
- The other 3 bits are Set-UID, Set-GID (used in Lab1) and Sticky bit (used in Lab2)



Drawbacks of Permission Bits

- The permission bits scheme becomes unwieldy when a large number of user groups are needed to assign access rights to different files
- For example, suppose a user wants to give read access for file X to users A and B and read access for file Y to users B and C. At least two user groups are needed, and user B would need to belong to both groups in order to access the two files.



Access Control Lists (ACLs) in UNIX

- Modern UNIX systems support ACL, called "facl (full access control list)", which is more expressive than permission bits.
- "getfacl file1"
  - Display file1's facl
- "setfacl –m user:bob:r file1"
  - Grant user:bob read permission to file1.
  - Can associate any number of users or groups with a file, each with 3 protection bits (rwx).

```
Terminal
seed@VM:$ getfacl file1
# file: file1
# owner: seed
# group: seed
user::rw-
group::rw-
other::r--

seed@VM:$ setfacl -m user:bob:r file1
seed@VM:$ getfacl file1
# file: file1
# owner: seed
# group: seed
user::rw-
user:bob:r--
group::rw-
mask::rw-
other::r--
```

Outline

- Introduction
- Discretionary Access Control (DAC)
  - UNIX file access control
- Role-based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)
- Bank RBAC system

- RBAC is based on the roles that users assume in a system rather than the user's identity; Access rights are assigned to roles instead of users
  - e.g., a user in the Accounts Payable clerk position would be assigned the AP Role in the accounting system
- The relationship of users to roles is many-to-many
- The set of users may change frequently, but the set of roles is likely to be relatively static, with only occasional additions or deletions

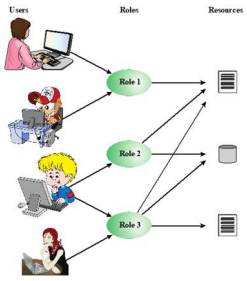


Figure 4.6 Users, Roles, and Resources

- The upper matrix relates users to roles; The lower matrix relates roles to objects.

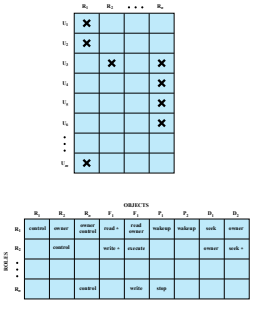


Figure 4.7 Access Control Matrix Representation of RBAC

RBAC Models

Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

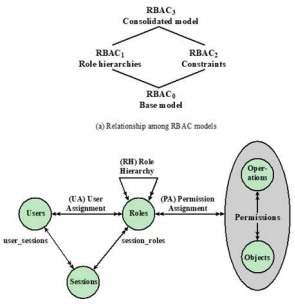


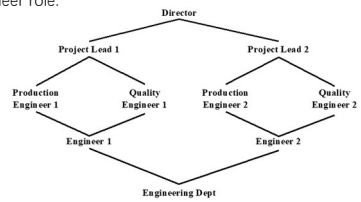
Figure 4.8 A Family of Role-Based Access Control Models.

Fig. 4.8(b) Explanations

- Figure 4.8b illustrates $RBAC_1$. It contains the four types of entities:
- **User**: An individual with a user ID.
- **Role:** A named job function within the organization.
  - Many-to-many relationship between users and roles: one user may have multiple roles, and multiple users may be assigned to a single role.
- **Permission:** Equivalent terms are *access right, privilege, and authorization.*
  - Many-to-many relationship between roles and permissions.
- **Session**: A mapping between a user and an activated subset of the set of roles to which the user is assigned.
  - Principle of least privilege: the user establishes a session with only the roles needed for a particular task.
- The arrowed lines indicate relationships, or mappings, with a single arrowhead indicating one and a double arrowhead indicating many.

Example of Role Hierarchy

- A line between two roles implies that the upper role includes all of the access rights of the lower role, as well as other access rights not available to the lower role.
- One role can inherit access rights from multiple subordinate roles, e.g., the Project Lead role includes all of the access rights of the Production Engineer role and of the Quality Engineer role.



Constraints

- A defined relationship among roles or a condition related to roles
- Mutually exclusive roles
  - A user can only be assigned to one role in the set (either during a session or statically).
    - e.g., the set (student, teacher) is mutually exclusive; the set (student, TA, teacher) is not mutually exclusive (if a student may work as a TA).
  - Any permission (access right) can be granted to only one role in the set
- Cardinality
  - Setting a maximum number of users that can be assigned to a given role, or the maximum number of roles that a user is assigned to, or a user can activate in a single session
- Prerequisite roles
  - Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role (used with role hierarchies)

Outline

- Introduction
- Discretionary Access Control (DAC)
  - UNIX file access control
- Role-based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)
- Bank RBAC system

Attribute-Based Access Control (ABAC)

- Can define authorizations that express conditions on properties of both the resource and the subject
- Strength is its flexibility and expressive power
- Main obstacle to its adoption in real systems is slow performance of evaluating predicates on both resource and user properties for each access
- There is considerable interest in applying the model to web services.

ABAC Model: Attributes

- Subject attributes
  - Define identity and characteristics of the subject
- Object attributes
  - Define identity and characteristics of the object
- Environment attributes
  - Describe the operational, technical, and situational environment or context in which the information access occurs
  - e.g., a stock brokerage system may specify that transactions are permitted only during workdays 9am–5pm, so time is part of env attributes

ABAC

- Controls access to objects by evaluating rules against the attributes of entities, operations, and the environment relevant to a request.
- Allows an unlimited number of attributes to be combined to satisfy any access control rule.
- Most general. ABAC can be used to implement DAC, MAC and RBAC

ABAC Scenario

- ABAC allows very flexible policies to be specified, e.g. only users with affiliation of ZJU, security clearance of high, age older than 40 can access files with owner ZJU, classification "secret" or below, during workdays 9am-5pm.

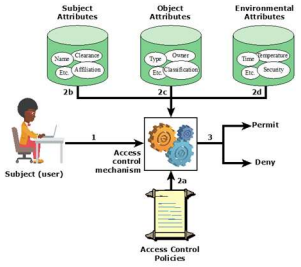


Figure 4.10 ABAC Scenario

ABAC Policies

- A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions
- Typically written from the perspective of the object that needs protecting and the privileges available to subjects (similar to ACL)

RBAC vs ABAC

- RBAC is simpler and more efficient at runtime
- RBAC is for coarse-grained access control; ABAC is for fine-grained access control
  - e.g., RBAC can be used to give all teachers (not students) access to a database; ABAC can be used to give teachers access to a database if they are at School X and teach Grade Y, during 9am to 5pm workdays
- RBAC and ABAC can be used together hierarchally
  - e.g., RBAC controls who can see what modules; ABAC controls access to what a user can see (or can do) inside of a module.

Outline

- Introduction
- Discretionary Access Control (DAC)
  - UNIX file access control
- Role-based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)
- Bank RBAC system

Case Study: RBAC System For A Bank

- Each role is defined by an official position performing a function.

(a) Functions and Official Positions

Role	Function	Official Position
A	financial analyst	Clerk
B	financial analyst	Group Manager
C	financial analyst	Head of Division
D	financial analyst	Junior
E	financial analyst	Senior
F	financial analyst	Specialist
G	financial analyst	Assistant
•••	•••	•••
X	share technician	Clerk
Y	support e-commerce	Junior
Z	office banking	Head of Division

Functions and Roles

(b) Permission Assignments

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments	1, 2, 4, 7
• • •	• • •	• • •

(c) PA with Inheritance

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
• • •	• • •	• • •

Functions and Roles Explanations

- Role B has more access rights than Role A.
  - Role B has as many or more access rights than role A in three applications and has access rights to a fourth application.
- We can define a role hierarchy in which Role B is superior to Role A, making it possible to simplify access rights definitions, as in (c).

Access Control Administration

- Each user may be statically assigned 1-4 roles (constraint) by the HR Department, and selects a given role for use in invoking a particular application.
- The user/role information is provided to the Authorization Administration, which creates a security profile for each user that associates the User ID and role with a set of access rights.
- When a user invokes an application, the application consults the security profile for that user to determine what subset of the application's access rights are in force for this user in this role.

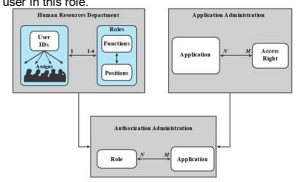


Figure 4.14 Example of Access Control Administration

CH27 Trusted Computing and MILS

ZJU 2020

Outline

- Bell-LaPadula (BLP), Biba and Chinese wall model
- Reference monitors for trusted systems
- Application of multilevel security to databases
- TCB and TPM
- Common Criteria

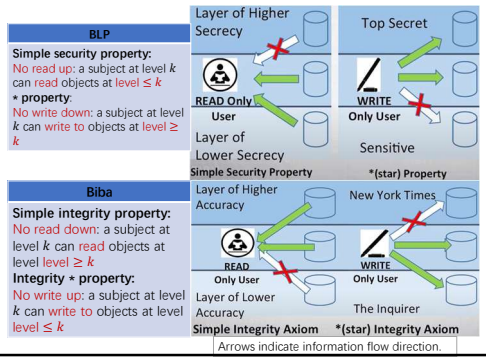
Multilevel Security (MLS)

RFC 4949 defines multilevel security as follows:

"A mode of system operation wherein (a) two or more security levels of information are allowed to be handled concurrently within the same system when some users having access to the system have neither a security clearance nor need-to-know for some of the data handled by the system and (b) separation of the users and the classified material on the basis, respectively, of clearance and classification level are dependent on operating system control."

BLP and Biba Models

- Security levels
  - A subject has a security clearance; an object has a security classification
  - Both are security levels forming a total order: top secret > secret > confidential > restricted > unclassified
- Bell-LaPadula (BLP) model: designed for protecting *confidentiality* of high-level objects from low-level subjects
  - Information flow from high to low-level is disallowed.
- Biba model: designed for protecting *integrity* of high-level objects from low-level subjects
  - Information flow from low to high-level is disallowed.
- BLP and Biba are examples of Mandatory Access Control (MAC).



BLP Quiz

- True or False: under BLP, an unclassified document can be read or written by anyone.
- ANS: False
  - Since "unclassified" is the lowest security level, any subject can read it (no read up), but only subjects at the unclassified level can write to it (no write down)
- True or False: under BLP, an unclassified user can create a classified document
- ANS: True (no write down)

*-property

- If no *-property, the following may occur:
- A malicious subject passes classified information along by putting it into an object labeled at a lower security classification than the information itself. This will allow a subsequent read access to this object by a subject at the lower clearance level.

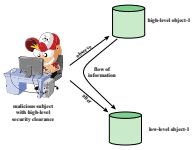


Figure 13.1 Information Flow Showing the Need for the *-property

BLP Model Example

- Assuming a Role-Based Access Control system. Carla is a student (s) in course c1. Dirk is a teacher (t) in course c1 but may also access the system as a student; thus two roles are assigned to Dirk:
  - Carla: (c1-s)
  - Dirk: (c1-t), (c1-s)
- The student role is assigned a lower, and the teacher role a higher security clearance level

- Figure 13.2a: Dirk creates a new file f1 as c1-t; Carla creates file f2 as c1-s (). Carla can read and write to f2, but cannot read f1, because it is at a higher classification level (teacher level). In the c1-t role, Dirk can read and write f1 and can read f2 if Carla grants access to f2. However, in this role, Dirk cannot write f2 because of the *-property; neither Dirk nor a Trojan horse on his behalf can downgrade data from the teacher level to the student level. Only if Dirk logs in as a student can he create a c1-s file or write to an existing c1-s file, such as f2. In the student role, Dirk can also read f2.
- Figure 13.2b: Dirk reads f2 and wants to create a new file with comments to Carla as feedback. Dirk must sign in student role c1-s to create f3 so that it can be accessed by Carla. In a teacher role, Dirk cannot create a file at a student classification level.

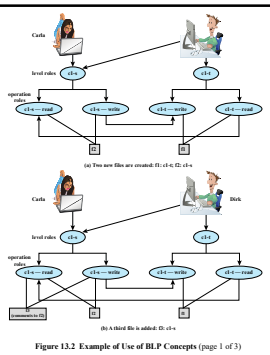


Figure 13.2 Example of Use of BLP Concepts (page 1 of 3)

- Figure 13.2c: Dirk creates an exam based on an existing exam template file at level c1-t. Dirk must log in as c1-t to read the template and the file he creates (f4) must also be at the teacher level.
- Figure 13.2d: Dirk wants Carla to take the exam and so must provide her with read access. However, such access would violate the ss-property. Dirk must downgrade the classification of f4 from c1-t to c1-s. Dirk cannot do this in the c1-t role because this would violate the *-property. Therefore, a security administrator must have downgrade authority and must be able to perform the downgrade *outside the BLP model*. The dotted line connecting f4 with c1-s-read indicates that this connection has not been generated by the default BLP rules but by a system operation.

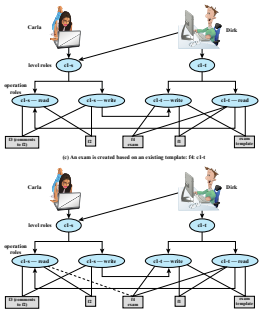


Figure 13.2 Example of Use of BLP Concepts (page 2 of 3)

BLP Model Example cont'd

- Carla writes the answers to the exam into a file f5. She creates the file at level c1-t so that only Dirk can read the file. Carla can still see her answers at her workstation but cannot access f5 for reading.

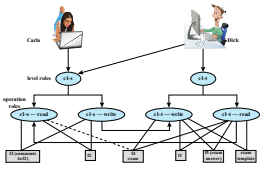


Figure 13.2 Example of Use of BLP Concepts (page 3 of 3)

Biba: Integrity *

- By simple integrity property, a low-integrity process may read low-integrity data but is prevented from contaminating a high integrity file with that data (lower part of Fig. 13.4).
- But this is not enough! A high-integrity process could conceivably copy low-integrity data into a high-integrity file (higher part of Fig. 13.4).
- Hence we need integrity * property, to disallow the high integrity process reading low-integrity file.

Figure 13.4 Contamination With Simple Integrity Controls [GASS88]

Chinese Wall (CW) Model

- Designed to prevent *Conflict-of-Interest (CI)*. Elements include:
  - Dataset (DS): All objects that concern the same company
  - Conflict-of-interest (CI) class: All datasets whose companies are in competition
- A subject can access any object as long as it has not accessed an object from another company in the same CI class
- Once a subject accesses information from one dataset, a wall is set up to protect information in other datasets in the same CI class.
- CW model does not protect confidentiality (like BLP) or integrity (like Biba).

CW Simple Security Rule

- **CW Simple Security rule:** A subject S can read object O only if
  - O is in the same DS as an object already accessed by S, OR
  - O belongs to a CI class from which S has not yet accessed any information
- Figures 13.6b and c: Assume that at some point, John has made his first read request to any object in this set for an object in the Bank A DS. Because John has not previously accessed an object in any other DS in CI 1, the access is granted. Any subsequent request for access to an object in the Bank B DS will be denied. Any request for access to other objects in the Bank A DS is granted.
- At a later time, John requests access to an object in the Oil A DS. Because there is no conflict, this access is granted, but a wall is set up prohibiting subsequent access to the Oil B DS.
- Similarly, Figure 13.6c reflects the access history of Jane.
- (Bank A DS should contain a,b,c here. In next slide, John changes a to g in violation of *-property.)

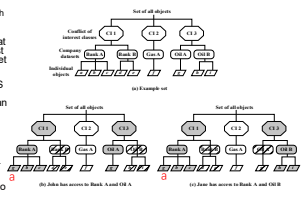


Figure 13.6 Potential Flow of Information Between Two CIs

CW *-Property Rule

- **CW *-property rule**: A subject S can write an object O only if
  - (1) S can read O according to the simple security rule, and
  - (2) All objects that S can read are in the same DS as O.
- Either subject cannot write to any DS, or a subject's access (both read and write) is limited to a single DS. (In Fig. 13.6(b)(c), John or Jane should not have write access to any DS.)
  - In our example, John has access to Oil A DS and Bank A DS; Jane has access to Oil B DS and Bank A DS. If John is allowed to read from Oil A DS and write into Bank A DS, John may transfer information about Oil A into Bank A DS by changing the value of the first object under the Bank A DS from a to g. *The data can then subsequently be read by Jane.* Thus, Jane would have access to information about both Oil A and Oil B, creating a CI.

Outline

- Bell-LaPadula (BLP), Biba and Chinese wall model
- Reference monitors for trusted systems
- Application of multilevel security to databases
- TCB and TPM
- Common Criteria

Reference Monitor

- The reference monitor regulates the access of subjects to objects on the basis of their security parameters, based on the security kernel database that lists access privilege (security clearance) of each subject and the protection attribute (classification level) of each object.

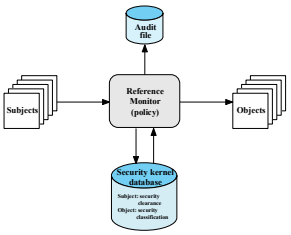


Figure 13.7 Reference Monitor Concept

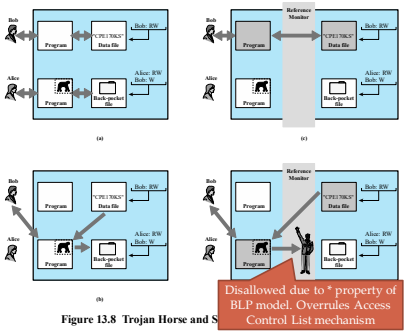


Figure 13.8 Trojan Horse and S

Fig. 13.8: Trojan Horse Defeated by Reference Monitor

- Fig. 13.8 (a): User Bob interacts through a program with a data file containing the critically sensitive character string "CPE170KS.". He has created the file with read/write permission provided only to programs executing on his own behalf, that is, only processes that are owned by Bob may access the file. A hostile user Alice installs both a Trojan horse program and a private file to be used in the attack as a "back pocket." Alice gives *read/write (RW)* permission to herself for this file and gives Bob *write (W)* permission, in the file's Access Control List (ACL).
- Fig. 13.8 (b): Alice tricks Bob into invoking the Trojan horse program, perhaps by advertising it as a useful utility. When the program detects that it is being executed by Bob, it reads the sensitive character string from Bob's file and copies it into Alice's back-pocket file. Both the read and write operations satisfy the constraints imposed by ACL. Alice then can access Bob's file at a later time to learn the value of the string.
- Fig. 13.8 (c): A secure OS implements BLP model with two security levels: sensitive (high) and public (low). Bob's files and processes are assigned security level sensitive; Alice's file and processes are assigned public.
- Fig. 13.8 (d): Bob invokes the Trojan horse program, which acquires Bob's security level and reads the sensitive character string. When the program attempts to store the string in a public file (the back-pocket file), however, BLP's *-property is violated and the attempt is disallowed by the reference monitor.
- Thus, the attempt to write into the back-pocket file is denied even though the ACL permits it: *Reference monitor's security policy (BLP) takes precedence over the ACL mechanism.*

Outline

- Bell-LaPadula (BLP), Biba and Chinese wall model
- Reference monitors for trusted systems
- Application of multilevel security to databases
- TCB and TPM
- Common Criteria

Fig. 13.9: Multi-Level Security for Databases

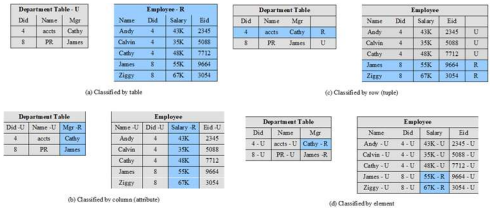


Fig. 13.9: Multi-Level Security for Databases Explanations

- Security classification can be assigned at different levels of granularity. Finer granularity classification may be vulnerable to inference attacks.
- Entire database
- Fig. 27.9(a): Individual tables
  - Two security levels of are defined: low (unrestricted), and high (restricted). "Employee" table contains sensitive salary information and is classified as restricted, while "Department" table is unrestricted.
- Fig. 27.9(b): Individual attributes:
  - Attribute "Salary" in "Employee" table, and attribute "Manager" in "Department" table are restricted.
- Fig. 27.9(c): Individual tuples:
  - All tuples in "Department" table that contain information relating to the Accounts Department (Did = 4), and all tuples in "Employee" table with Salary ≥ $50K$ are restricted.
- Fig. 27.9(d): Individual elements:
  - Attribute "Salary" in "Employee" table for tuples with Salary ≥ $50K$, and attribute "Manager" in "Department" table for tuples with Did = 4 are restricted.

Database Security: Read Access

- Assuming BLP model. **For read access, the database needs to enforce the *simple security* property (no read up).**
- Consider SQL query:
  - SELECT Ename FROM Employee WHERE Salary ≥ $50K$
- Inference problem for classification by column:
  - Fig. 13.9(b): Attribute "Salary" in "Employee" table is restricted.
  - The SQL query returns only unrestricted Attribute (Ename), but reveals restricted Attribute "Salary", namely whether any employees with Salary ≥ $50K$ and, if so, which employees.
  - Solution is to check not only the attribute returned by SELECT (Ename), but also any other attributes must be accessed.
- Inference problem for classification by row:
  - Fig. 13.9(c): all tuples in "Employee" table with Salary ≥ $50K$ are restricted.
  - The SQL query returns null, so the previous inference is prevented.
  - But a weaker inference occurs: one can infer that either tuples in "Employee" table with Salary ≥ $50K$ is restricted, or no employee has Salary ≥ $50K$.
- These problems are avoided if we use classification by database or table.

Employee

Name -U	Did -U	Salary -R	Eid -U
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

Fig. 13.9(b)

Employee

Name	Did	Salary	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
Ziggy	8	67K	3054	R

Fig. 13.9(c)

Database Security: Write Access

- Assuming BLP model. **For write access, the database needs to enforce the *- property (no write down).**
- Consider SQL statement:
  - INSERT INTO Employee VALUES (James,8,35K,9664)
- Consider Fig. 27.9(c): all tuples in "Employee" table with Salary ≥ $50K$ are restricted.
- An unrestricted user wants to insert a tuple with primary key (Eid = 9664) that already exists in a restricted tuple (for James):
  - Can reject, but then user can infer that the row exists
  - Can replace, but then it compromises data integrity
  - Solution is *polyinstantiation*: insert a new *unrestricted* row with the same primary key, but with salary 35K to create conflicting entries.
- **This also helps prevent inference problems for read access.**
  - **If an unrestricted user queries James' salary in the original table, the user's request is rejected and the user may infer that his salary ≥ 50K. Inclusion of the "false" row provides a form of "cover" for the true salary of James.**
- These problems are avoided if we use classification by database or table.

Employee

Name	Did	Salary	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
Ziggy	8	67K	3054	R

Before insertion

Employee

Name	Did	Salary	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
James	8	35K	9664	U
Ziggy	8	67K	3054	R

After insertion with polyinstantiation

Outline

- Bell-LaPadula (BLP), Biba and Chinese wall model
- Reference monitors for trusted systems
- Application of multilevel security to databases
- TCB and TPM
- Common Criteria

Trusted Computing Base (TCB)and Trusted Platform Module (TPM)

- TCB: the set of all hardware, firmware, and/or software components that are critical to its security. It provides 3 basic services
  - Authenticated boot
  - Certification
  - Encryption
- TPM: a secure hardware crypto-processor with integrated cryptographic keys.

Authenticated Boot Service

- Responsible for booting entire OS in stages and ensuring each is valid and approved for use
  - At each stage digital signature associated with code is verified
  - TPM keeps a tamper-evident log of the loading process
- Log records versions of all code (OS and application) that is running
  - Can then expand trust boundary to include additional hardware and application and utility software
  - Confirms component is on the approved list, is digitally signed, and that serial number hasn't been revoked.
- Result is a configuration that is well-defined with approved components

Certification Service

- Once a configuration is achieved and logged the TPM can certify configuration to others
  - Can produce a digital certificate
- Confidence that configuration is unaltered because:
  - TPM is considered trustworthy
  - Only the TPM possesses this TPM's private key
- To assure that the certificate is both valid and up to date, a requester issues a "challenge" in the form of a random number when requesting a signed certificate from the TPM. The TPM signs a block of data consisting of the configuration information with the random number appended to it.
- Provides a hierarchical certification approach
  - Hardware/OS configuration
  - OS certifies application programs
  - User has confidence in application configuration

Encryption Service

- Encrypts data so that it can only be decrypted by a machine with a certain configuration
- TPM maintains a master secret key unique to machine
  - Used to generate secret encryption key for every possible configuration of that machine
- Can extend scheme upward to application-level
  - Provide encryption key to application so that decryption can only be done by desired version of application running on desired version of the desired OS
  - Encrypted data can be stored locally or transmitted to a peer application on a remote machine

TPM for File Protection

- A TPM protected file can be "sealed" to a particular software configuration (OS + applications), and only permits access if the current software config matches.
- 1. The symmetric key that was used to encrypt the file is stored with the file. The key itself is protected (encrypted) with another key to which the TPM has access. The protected key is submitted to the TPM with a request to reveal the key to the application.
- 2. Associated with the protected key is a specification of the software config that may have access to the key. The TPM verifies that the current config matches the config required for revealing the key. In addition, the requesting application must be authorized to access the key.
- 3. If the current config is permitted access to the protected key, then the TPM decrypts the key and passes it on to the application.
- 4. The application uses the key to decrypt the file.

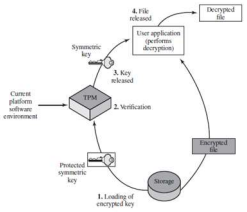


Figure 27.12 **Decrypting a File Using a Protected Key**

TPM Component Architecture (Typical)

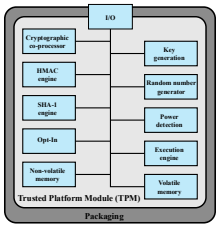
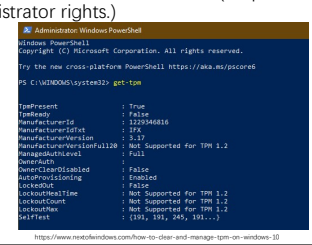


Figure 13.12 TPM Component Architecture

TPM on Windows

- To check the status of TPM on your computer, you can either use TPM.msc management console or get-tpm in a PowerShell session. (Requires administrator rights.)



https://www.nextofwindows.com/how-to-clear-and-manage-tpm-on-windows-10

Outline

- Bell-LaPadula (BLP), Biba and Chinese Wall model
- Reference monitors for trusted systems
- Application of multilevel security to databases
- TCB and TPM
- Common Criteria

Common Criteria

- ISO standard for security evaluation
- Aim is to provide greater confidence in IT product security
  - Development using secure requirements
  - Evaluation confirming meets requirements
  - Operation in accordance with requirements
- Following successful evaluation a product may be listed as CC certified
  - 7 Evaluation Assurance Levels (EALs)
- EAL 1: Functionally tested
- EAL 2: Structurally tested
- EAL 3: Methodically tested and checked
- EAL 4: Methodically designed, tested, and reviewed
- EAL 5: Semi-formally designed and tested
- EAL 6: Semi-formally verified design and tested
- EAL 7: Formally verified design and tested

CH05 Database Security

ZJU 2020

Outline

- Relational databases
- SQL injection attacks
- Access control
- Inference
- Database encryption

Relational Database

- A relational database consists of tables
- A table is defined by a schema and consists of tuples
  - Each tuple (row) stores attribute values as define by schema
  - Typically, one column contains the primary key, each uniquely identifies a tuple
- Enables the creation of multiple tables linked together by keys
- Structured Query Language (SQL)
  - Provides a uniform interface to the database

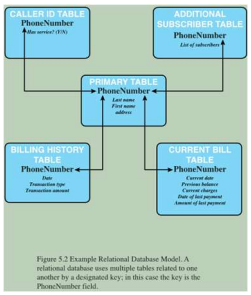


Figure 5.2 Example Relational Database Model. A relational database uses multiple tables related to one another by a designated key; in this case the key is the PhoneNumber field.

Relational Database Elements

- Primary key
  - Uniquely identifies a row
  - Consists of one or more column names
- Foreign key
  - Links one table to attributes in another table
- View/virtual table
  - Result of a query that returns selected rows and columns from one or more tables

Formal Name	Common Name	Also Known As
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

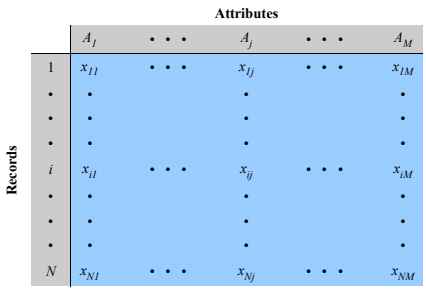


Figure 5.3 Abstract Model of a Relational Database

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

primary key

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

foreign key

primary key

(a) Two tables in a relational database

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

Figure 5.4 Relational Database Example

Structured Query Language (SQL)

- Standardized language to define schema, manipulate, and query data in a relational database.
- Operations include:
  - Create tables
  - Insert and delete data in tables
  - Create views
  - Retrieve data with query statements
- Example: SELECT * FROM EMLPOYEE WHERE DID='15'
  - return all tuples in EMPLOYEE table with attribute DID='15'

OS vs. Database Security

- OS security mechanisms typically control read and write access to entire files, but not to limit access to specific records or fields in that file.
- A database typically allows this type of fine-grained access control, expressed with SQL commands, such as to select, insert, update, or delete specified items in the database.
- Thus, security services and mechanisms are needed that are designed specifically for database systems.

Outline

- Relational databases
- SQL injection attacks
- Access control
- Inference
- Database encryption

SQL Injection Attacks (SQLi)

- Sends malicious SQL commands to the database
- One of the most prevalent and dangerous network-based security threats
- Designed to exploit the nature of Web application pages
- Can be used to
  - Modify or delete data
  - Execute arbitrary OS commands
  - Launch DoS attacks

SQLi Example

- The following web form's intention is to let the user provide some data for the blank areas:

```
SELECT Name, Salary, SSN
FROM employee
WHERE eid=' [          ] ' and password=' [          ] '
```

One Attack

- User inputs a random string in the password entry, and EID5002'# in the eid entry.
- Since everything from the # sign to the end of line is considered as comment, the SQL statement will be equivalent to the right statement, with no password checking

```
SELECT Name, Salary, SSN
FROM employee
WHERE eid='EID5002' #' and password='xyz'
```

⇨

```
SELECT Name, Salary, SSN
FROM employee
WHERE eid='EID5002'
```

157

More Attacks

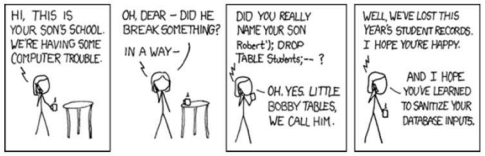
- Left: user enters a' OR 1=1 to create a predicate for WHERE clause, which is always true, so it returns all the records from the database, even if he does not know all the eid's in the database.
- Right: user enters a'; DROP table employee. **The database views this SQL statement as 2 separate statements separated by ;**
  - First select all entries with eid='a', then delete the entire employee table.
  - As countermeasure, use mysql_real_escape_string() to prepend backslashes to some special characters, including \n, \r, \, ', ". Now we have: WHERE name = 'a\'\; drop table suppliers'. The database looks for a tuple with name matching the long string a'; drop table employee, and returns null.

```
SELECT Name, Salary, SSN
FROM employee
WHERE eid='a' OR 1=1
```

```
SELECT Name, Salary, SSN
FROM employee
WHERE eid='a'; DROP table employee
```

158

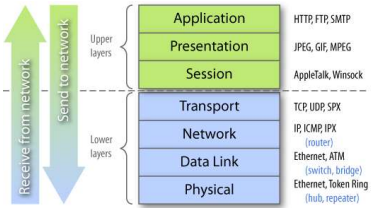
SQLi Cartoon



159

SQLi is Application-Level Attack

- Not detectable by lower-layer defense mechanisms such as firewalls



SQLi Countermeasures

- Defensive coding
  - Parameterized query insertion
  - SQL DOM
- Detection
  - Signature based
  - Anomaly based
  - Code analysis
- Run-time prevention
  - Check queries at runtime to see if they conform to a model of expected queries

Outline

- Relational databases
- SQL injection attacks
- Access control
- Inference
- Database encryption

162

Database Access Control

- Access control determines:
  - If the user has access to the entire database or just portions of it
  - What access rights the user has (create, insert, delete, update, read, write)
- Can support a range of administrative policies:
  - Centralized administration
    - Small number of privileged users may grant and revoke access rights
  - Ownership-based administration
    - The creator of a table may grant and revoke access rights to the table
  - Decentralized administration
    - The owner of the table may grant and revoke authorization rights to other users, allowing them to grant and revoke access rights to the table

SQL Access Controls

- Two commands for managing access rights:
  - GRANT and REVOKE
- Typical access rights are:
  - Select
  - Insert
  - Update
  - Delete
  - References

```
GRANT { privileges | role }
[ON table]
TO { user | role | PUBLIC }
[IDENTIFIED BY password]
[WITH GRANT OPTION]
```

```
REVOKE { privileges | role }
[ON table]
FROM { user | role |
PUBLIC }
```

- Examples:
  - GRANT SELECT on ANY TABLE to Alice
    - It enables user Alice to query any table in the database; but Alice cannot further propagate the access to other users, due to lack of "WITH GRANT OPTION" (similar to * in Access Control Matrix in CH04)
  - REVOKE SELECT on ANY TABLE from Alice
    - Revokes the SELECT right from Alice
  - (This is Discretionary Access Control)

Cascaded Grants

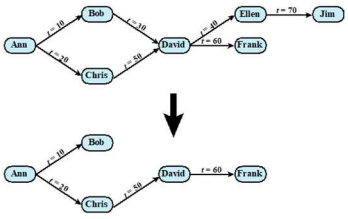


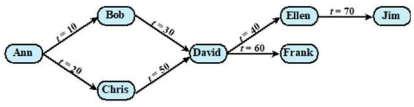
Figure 5.6 Bob Revokes Privilege from David

Cascaded Grants Explanations

- The revocation of privileges is cascaded: when user A revokes an access right, any cascaded access right is also revoked, unless that access right would exist even if the original grant from A had never occurred, i.e., it does not depend on the original grant from A.
- The figure indicates that Ann grants the access right to Bob at time t = 10 and to Chris at time t = 20. Assume that the *grant option* is always used. Thus, Bob is able to grant the access right to David at t = 30. Chris redundantly grants the access right to David at t = 50. Meanwhile, David grants the right to Ellen, who in turn grants it to Jim; and subsequently David grants the right to Frank.
  - if Ann revokes the access right to Bob and Chris, then the access right is also revoked to David, Ellen, Jim, and Frank.
- A complication arises when a user (David) receives the same access right multiple times.
  - If the grant from Bob to David at t=30 is revoked, the grants from David to Ellen at time t=40, and from Ellen to Jim at t=70, are also revoked, since they depend on the grant from Bob to David at t=30.
  - But the grant from David to Frank at t=60 remains, since it depends on the grant from Chris to David at t=50.

Cascaded Grants Quiz

- Q: If the grant from Chris to David at t=50 is revoked, what happens to downstream grants?
- ANS: None of the downstream grants will be revoked, since they depend on the grant from Bob to David at t=30, not the grant from Chris to David at t=50
- Q: If the grant from Ann to Chris at t=20 is revoked, what happens to downstream grants?
- ANS: The grant from Chris to David at t=50 will be revoked; all other grants remain
- Q: If the grant from Ann to Bob at t=10 is revoked, what happens to downstream grants?
- ANS: The grant from Bob to David, from David to Ellen, and from Ellen to Jim will be revoked; all other grants remain



Role-Based Access Control (RBAC)

- RBAC is a natural fit for database access control.
  - A database system supports multiple of applications. An individual user may use a variety of applications to perform a variety of tasks, each of which requires its own set of privileges
- RBAC eases administrative burden and improves security
- A database RBAC needs to provide the following capabilities:
  - Create and delete roles
  - Define permissions for a role
  - Assign and cancel assignment of users to roles

Table 5.2 Fixed Roles in Microsoft SQL Server

Role	Permissions
Fixed Server Roles	
sysadmin	Can perform any activity in SQL Server and have complete control over all database functions
serveradmin	Can set server-wide configuration options, shut down the server
setupadmin	Can manage linked servers and startup procedures
securityadmin	Can manage logins and CREATE DATABASE permissions, also read error logs and change passwords
processadmin	Can manage processes running in SQL Server
dbcreator	Can create, alter, and drop databases
diskadmin	Can manage disk files
bulkadmin	Can execute BULK INSERT statements
Fixed Database Roles	
db_owner	Has all permissions in the database
db_accessadmin	Can add or remove user IDs
db_datareader	Can select all data from any user table in the database
db_datawriter	Can modify any data in any user table in the database
db_ddladmin	Can issue all Data Definition Language (DDL) statements
db_securityadmin	Can manage all permissions, object ownerships, roles and role memberships
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements
db_denydatareader	Can deny permission to select data in the database
db_denydatawriter	Can deny permission to change data in the database

169

Outline

- Relational databases
- SQL injection attacks
- Access control
- Inference
- Database encryption

170

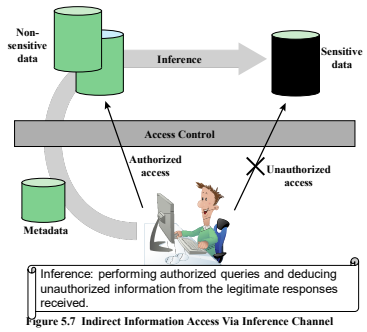


Figure 5.7 Indirect Information Access Via Inference Channel

Inference Attack Examples

- Consider a database of student grades with schema (StudentID, Standing (junior or senior), Exam Score). Attacker wants to find exam score of some student. Any student should be able to query for the average exam score
- Example 1:
  - The target student Alice takes the exam late
  - Attacker queries for the average scores before and after Alice takes the exam, then calculate Alice's score
- Example 2:
  - Only one student Bob has junior standing in a class full of seniors
  - Attacker queries for the average score of all students with junior standing, which is equal to Bob's score.

Inference Attack Examples cont'd

- Each employee's salary should be confidential
- RBAC constraint: Name and Salary cannot be accessed together in the same query by the Clerk role.
- But a user who knows (1) the structure of the Employee table and (2) the views maintain the same row order as the Employee table is able to merge the two views in (b) to construct the table shown in (c)

Name	Position	Salary ($)	Department	Dept. Manager
Andy	senior	43,000	strip	Cathy
Calvin	junior	35,000	strip	Cathy
Cathy	senior	48,000	strip	Cathy
Dennis	junior	38,000	panel	Herman
Herman	senior	55,000	panel	Herman
Ziggy	senior	67,000	panel	Herman

(a) Employee table

Position	Salary ($)
senior	43, 000
junior	35,000
senior	48,000

Name	Department
Andy	strip
Calvin	strip
Cathy	strip

(b) Two views

Name	Position	Salary ($)	Department
Andy	senior	43,000	strip
Calvin	junior	35,000	strip
Cathy	senior	48,000	strip

(c) Table derived from combining query answers

Figure 5.8 Inference Example

Solution

- Construct three tables, which include the following information:
  - Employee (Emp#, Name, Address)
  - Salary (S#, Salary)
  - Emp-Salary (Emp#, S#)
- Each line consists of the table name followed by a list of column names for that table. In this case, each employee is assigned a unique employee number (Emp#) and a unique salary number (S#). The Employee and Salary tables are accessible to the Clerk role, but the Emp-Salary table is only available to the Administrator role. In this structure, the sensitive relationship between employees and salaries is protected from users assigned the Clerk role.
- **Another inference channel**: suppose that we want to add a new attribute, employee start date, which is not sensitive. This could be added to the Salaries table as follows:
  - Employees (Emp#, Name, Address)
  - Salaries (S#, Salary, Start-Date)
  - Emp-Salary (Emp#, S#)
- However, an employee's start date is an easily observable or discoverable attribute (e.g. through human/personal connections). Thus a user in the Clerk role may be able to infer the employee's name. This would compromise the relationship between employee and salary.
- A straightforward way to remove the inference channel is to add the start-date column to the Employees table instead of the Salaries table.

Inference Detection

- The first inference problem, that it was possible to infer the relationship between employee and salary, can be detected through analysis of the data structures and security constraints.
- However, the second inference problem, in which the start-date column was added to the Salaries table, cannot be detected using only the information stored in the database. In particular, the database does not indicate that the employee name can be inferred from the start date.

Two Approaches to Inference Detection

- During database design
  - Remove an inference channel by altering the database structure or by changing the access control regime to prevent inference
    - Examples include removing data dependencies by splitting a table into multiple tables or using more fine-grained access control roles in an RBAC scheme.
  - Techniques in this category often result in unnecessarily stricter access controls that reduce availability
- Inference detection at query time
  - Seek to eliminate an inference channel violation during a query or series of queries
  - If an inference channel is detected, the query is denied or altered

Outline

- Relational databases
- SQL injection attacks
- Access control
- Inference
- Database encryption

Database Encryption

- The database is typically the most valuable information resource for any organization, protected by multiple layers of security (defense-in-depth)
  - Firewalls, authentication, general access control, database access control
  - Encryption becomes the last line of defense in database security. Can be applied to the entire database, at the record level, the attribute level, or level of the individual field
- Disadvantages to encryption:
  - Key management
    - Authorized users must have access to the decryption key for the data for which they have access
  - Inflexibility
    - When part or all of the database is encrypted it becomes more difficult to perform record searching

A Database Encryption Scheme

Data owner: organization that produces data to be made available for controlled release

User: human entity that presents queries to the system

Client: frontend that transforms user queries into queries on the encrypted data stored on the server

Server: an organization that receives the encrypted data from a data owner and make them available for distribution to clients

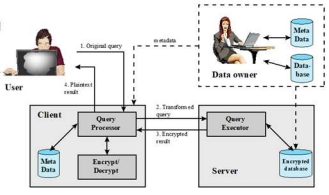


Figure 5.9 A Database Encryption Scheme

- Each row R_i is treated as a contiguous binary block $B_i = (x_{i1} \| x_{i2} \| \dots \| x_{iM})$
- The entire row is encrypted as $E(k, B_i)$.
- To assist in data retrieval, attribute indexes are created for some or all of the attributes. For each row, the mapping from unencrypted to encrypted database is as follows:
- $(x_{i1} \| x_{i2} \| \dots \| x_{iM}) \rightarrow [E(k, B_i), I_{i1}, I_{i1}, \dots, I_{iM}]$

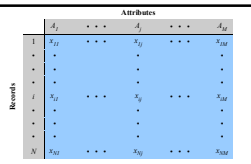


Figure 5.3 Abstract Model of a Relational Database

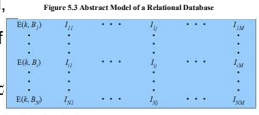


Figure 5.10 Encryption Scheme for Database of Figure 5.3

Table 5.3 Encrypted Database Example

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

E(*k*, *B*)	I(eid)	I(ename)	I(salary)	I(addr)	I(did)
1100110011001011…	1	10	3	7	4
0111000111001010…	5	7	2	7	8
1100010010001101…	2	5	1	9	5
0011010011111101…	5	5	2	4	9

181

Table 5.3 Explanations

- Suppose employee ID (eid) values lie in the range [1, 1000]. We can divide these values into five partitions: [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000], assigned index values 1, 2, 3, 4, and 5, respectively. For a text field, we can derive an index from the first letter of the attribute value. For the attribute ename , let us assign index 1 to values starting with A or B, index 2 to values starting with C or D, and so on. Similar partitioning schemes can be used for each of the attributes.
- In Table 5.3b, the values in the first column represent the encrypted values for each row. The remaining columns show index values for the corresponding attribute values. The mapping functions between attribute values and index values constitute metadata that are stored at the client and data owner locations but not at the server.
- This arrangement provides for more efficient data retrieval. Suppose, for example, a user requests records for all employees with eid < 300. The query processor requests all records with I(eid) ≤ 2. These are returned by the server. Or, a user requests records for all employees with eid=205. The query processor requests all records with I(eid) = 2. The query processor decrypts all rows returned, discards those that do not match the original query, and returns the requested unencrypted data to the user.
- The indexing scheme provides some information to an attacker, namely a rough relative ordering of rows by a given attribute. To obscure such information, the ordering of indexes can be randomized. For example, the eid values could be partitioned by mapping [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000] into 2, 3, 5, 1, and 4, respectively. Because the metadata are not stored at the server, an attacker could not gain this information from the server.

182

CH06 Malicious Software

ZJU 2020

183

Outline

- Types of Malicious Software (Malware)
- Advanced Persistent Threat (APT)
- Propagation
  - Infected Content – Viruses
  - Vulnerability Exploit -- Worms
  - Social Engineering -- Spam E-Mail, Trojans
- Payload
  - System Corruption
  - Attack Agent -- Zombie, Bots
  - Information Theft -- Keyloggers, Phishing, Spyware
  - Stealthing -- Backdoors, Rootkits
- Countermeasures

184

Malware Definition

- A program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or otherwise annoying or disrupting the victim.

185

Classification of Malware

- Malware can be classified by how it spreads or propagates to reach the desired targets
- Need for host program:
  - Those that need a host program (parasitic code such as viruses)
  - Those that are independent, self-contained programs (worms, trojans, and bots)
- Self-replication
  - Those that do not replicate themselves (trojans and spam e-mail)
  - Those that replicate themselves (viruses and worms)

186

Types of Malware

- Propagation mechanisms:
  - Infection of existing files by viruses that is subsequently spread to other systems
  - Exploit of software vulnerabilities by worms or drive-by-downloads to allow the malware to replicate
  - Social engineering attacks that convince users to bypass security mechanisms to install Trojans or to respond to phishing attacks
- Payload actions performed by malware once it reaches a target system:
  - Corruption of system or data files
  - Make the system a zombie agent of attack as part of a botnet
  - Theft of information from the system/keylogging
  - Stealthing/hiding its presence in the system

187

Types of Attackers

- Attackers may range from individuals demonstrating their technical competence to their peers, to more organized and dangerous attack sources such as:
  - Politically motivated attackers
  - Criminals
  - Organized crime
  - Organizations that sell their services to companies and nations
  - National government agencies
- A large underground economy involving the sale of attack kits, access to compromised hosts, and to stolen information

Outline

- Types of Malicious Software (Malware)
- Advanced Persistent Threat (APT)
- Propagation
  - Infected Content – Viruses
  - Vulnerability Exploit -- Worms
  - Social Engineering -- Spam E-Mail, Trojans
- Payload
  - System Corruption
  - Attack Agent -- Zombie, Bots
  - Information Theft -- Keyloggers, Phishing, Spyware
  - Stealthing -- Backdoors, Rootkits
- Countermeasures

189

Advanced Persistent Threats (APTs)

- Well-resourced, persistent application of a wide variety of intrusion technologies and malware to selected targets (usually business or political)
- Typically attributed to state-sponsored organizations and criminal enterprises
- Differ from other types of attack by their careful target selection and stealthy intrusion efforts over extended periods

190

APT Characteristics

- Advanced
  - Using a wide variety of intrusion technologies
- Persistent
  - Determined application of the attacks over an extended period against the chosen target in order to maximize the chance of success
  - A variety of attacks may be progressively applied until the target is compromised
- Threats
  - Serious threats to the selected targets as a result of the organized, capable, and well-funded attackers

191

APT Attacks

- Aim:
  - Varies from theft of intellectual property or security and infrastructure related data to the physical disruption of infrastructure
- Techniques used:
  - Social engineering
  - Spear-phishing attack: phishing attack targeting specific individuals
  - Drive-by-downloads from selected compromised websites likely to be visited by personnel in the target organization
- Intent:
  - To infect the target with sophisticated malware with multiple propagation mechanisms and payloads
  - Once they have gained initial access, a range of attack tools are used to maintain and extend their access

192

Outline

- Types of Malicious Software (Malware)
- Advanced Persistent Threat (APT)
- Propagation
  - Infected Content – Viruses
  - Vulnerability Exploit -- Worms
  - Social Engineering -- Spam E-Mail, Trojans
- Payload
  - System Corruption
  - Attack Agent -- Zombie, Bots
  - Information Theft -- Keyloggers, Phishing, Spyware
  - Stealthing -- Backdoors, Rootkits
- Countermeasures

193

Propagation

- Infected Content
  - Viruses
- Vulnerability Exploit
  - Worms
- Social Engineering
  - Spam E-Mail, Trojans

Viruses

- Piece of software that infects a host program
  - Modifies it to include a copy of the virus
  - Replicates and goes on to infect other content
  - Easily spread through network environments
- When attached to an executable program a virus can do anything that the program is permitted to do
  - Executes secretly when the host program is run
- May be specific to OS and hardware
  - Takes advantage of their details and weaknesses

195

Virus Components

- **Infection mechanism**
  - Means by which a virus spreads or propagates
  - Also referred to as the *infection vector*
- **Trigger**
  - Event or condition that determines when the payload is activated or delivered
- **Payload**
  - What the virus does (besides spreading)
  - May involve damage or benign but annoying activity

196

Virus Phases

- Dormant phase
  - Virus is idle
  - Will eventually be activated by some event
- Triggering phase
  - Virus is activated by a variety of system events
- Propagation phase
  - Virus places a copy of itself into other programs or into certain system areas on the disk
  - May not be identical to the propagating version
  - Each infected program will now contain a clone of the virus which will itself enter a propagation phase
- Execution phase
  - Payload function is performed

197

Virus Structure

```
program V
1234567;

procedure attach-to-program;
begin
  repeat
    file := get-random-program;
  until first-program-line ≠ 1234567;
  prepend V to file;
end;

procedure execute-payload;
begin
  (* perform payload actions *)
end;

procedure trigger-condition;
begin
  (* return true if trigger condition is true *)
end;

begin (* main action block *)
  attach-to-program;
  if trigger-condition then execute-payload;
  goto main;
end;
```

(a) A simple virus

```
program CV
1234567;

procedure attach-to-program;
begin
  repeat
    file := get-random-program;
  until first-program-line ≠ 1234567;
  compress file;  (* t1 *)
  prepend CV to file;  (* t2 *)
end;

begin (* main action block *)
  attach-to-program;
  uncompress rest of this file into tempfile;  (* t3 *)
  execute tempfile;  (* t4 *)
end;
```

(b) A compression virus

Figure 6.1 Example Virus Logic

198

Explanations for "Virus Structure"

- The string 1234567 is the virus' starting bit pattern. If first-program-line == 1234567, then the program is already infected with the virus, and should be skipped in the procedure "attach-to-program"
- The virus should (typically) be prepended to the program instead of postpended, since the program may have multiple exit points and may not execute to the end of its main(), so the postpended virus may not be executed

Compression Virus

- Fig. 6.1(a): Simple virus is easily detected because the virus-infected program is larger than the corresponding uninfected one.
- Figs. 6.1(b), 6.2: One way to thwart detection is to compress the executable file so that both the infected program (CV+P_2') and uninfected program (P_2) have identical size.

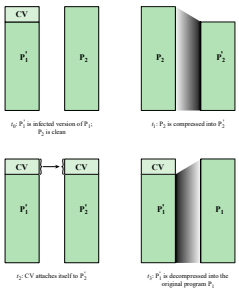


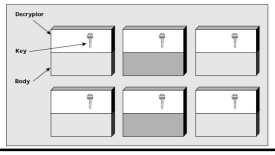
Figure 6.2 A Compression Virus

Virus Classification

- Classification by target
- Parasitic virus
  - Infects executable files (programs)
- Boot sector virus
  - Infects master boot record and spreads when system is booted from the disk containing the virus
- Macro virus
  - Infects files with macro or scripting code that is interpreted by an application
- Multipartite virus
  - Infects files in multiple ways
- Classification by concealment strategy
- Encrypted virus
  - A portion of the virus creates a random encryption key and encrypts the remainder of the virus
- Stealth virus
  - A form of virus explicitly designed to hide itself from detection by anti-virus software
  - may use code mutation, compression, or rootkit techniques.
- Polymorphic virus
  - A virus that mutates with every infection, changing its bit pattern, but different copies are functionally equivalent
- Metamorphic virus
  - Rewrites itself completely at each iteration, may change their function/behavior as well as their bit patterns.

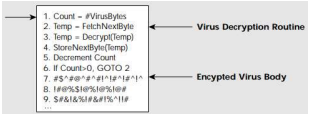
Encrypting Virus

- An encrypting virus always propagates using the same decryption routine. However, the key value within the decryption routine changes from infection to infection. Consequently, the encrypted body of the virus also varies, depending on the key value.
- It is a type of polymorphic virus.

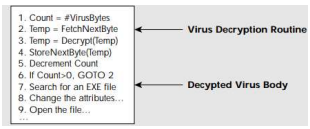


Encrypting Virus Example

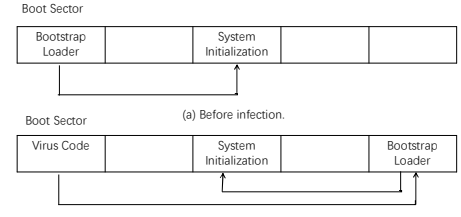
Before decryption



After decryption



Boot Sector Virus



(b) After infection. Virus code performs malicious functions before transferring control to bootstrap loader.

Macro and Scripting Viruses

- Exploit macro capabilities of MS Office files (e.g., Word, Excel documents) to infect these documents (not executable programs like word.exe, excel.exe)

Worms

- A program that actively seeks out more machines to infect, and each infected machine serves as an automated launching pad for attacks on other machines
  - Can use network connections to spread from system to system
  - Or spread through shared media (USB drives, CD, DVD···)
  - Or spread in macro or script code included in attachments and instant messenger file transfers
- Upon activation, the worm may replicate and propagate again

Worm Propagation Model

- Here is a classic epidemic model of worm (or virus) propagation

$$\frac{dI(t)}{dt} = \beta I(t)\,S(t)$$

where

$I(t)$ = number of individuals infected as of time t

$S(t)$ = number of susceptible individuals (susceptible to infection but not yet infected) at time t

β = infection rate

N = size of the population, $N = I(t) + S(t)$

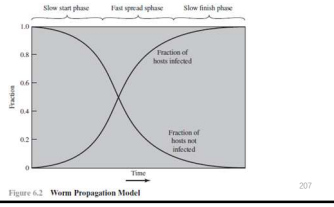


Figure 6.2 **Worm Propagation Model**

Worm Replication Mechanisms

- E-mail or instant messenger
  - Sends itself as an attachment to email, or via an instant messenger (e.g., QQ)
- File sharing
  - Creates a copy of itself on removable media (USB stick), then executes on the target system using the autorun mechanism or when a user opens it
- Remote execution
  - Executes a copy of itself on another system
- Remote file access or transfer
  - Uses remote file access or transfer service to copy itself from one system to the other
- Remote login
  - Logs onto a remote system as a user and then uses commands to copy itself from one system to the other

Target Discovery

- **Scanning** is the first function in the propagation phase of a worm
  - Searches for other systems to infect
- Scanning strategies:
  - **Random**: Each compromised host probes random addresses in the IP address space using a different seed. This produces a high volume of Internet traffic which may be detected even before the actual attack is launched
  - **Hit-list**: Compile and infect a list of potential vulnerable machines. Each infected machine is provided with a portion of the list to scan. This results in a very short scanning period which may help evade detection
  - **Topological**: Use information in an infected victim machine to find more hosts to scan
  - **Local subnet**: If a host can be infected behind a firewall, that host then looks for targets in its own local network using the subnet address structure, bypassing firewall protection

Morris Worm

- Earliest significant worm infection developed by Robert Morris in 1988.
- The worm achieved communication with the UNIX command interpreter. It then sent this interpreter a short bootstrap program, issued a command to execute that program, and then logged off.
- The bootstrap program then called back the parent program and downloaded the remainder of the worm. The new worm was then executed.
- (Robert Morris now a professor at MIT)

State-of-the-Art Worm Technology

- **Multiplatform:** attack a variety of platforms, including Windows, Linux, MacOS; or exploit macro supported in popular document types.
- **Multi-exploit:** penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications; or via shared media (USB sticks).
- **Fast spreading:** optimize the rate of spread of a worm to locate as many vulnerable machines as possible in a short time period.
- **Polymorphic:** Each copy of the worm has new code generated on-the-fly using functionally-equivalent instructions and encryption techniques.
- **Metamorphic:** In addition to changing their appearance, change their behavior at different iterations.
- **Transport vehicles:** spread a wide variety of malicious payloads, such as distributed denial-of-service bots, rootkits, and spyware.
- **Zero-day exploit**: an unknown vulnerability is exploited that is only discovered when the worm is launched.

Mobile Code

- Programs that can be shipped unchanged to a variety of platforms
  - e.g., Java applets, ActiveX, and JavaScript
- Transmitted from a remote system to a local system and then executed on the local system
- Often acts as an enabling mechanism for a virus, worm, or Trojan horse

212

Drive-By-Downloads (DbD)

- Exploits browser vulnerabilities to download and install malware on the system when the user views a web page controlled by the attacker
  - aimed at anyone who visits a compromised site
  - does not actively propagate

Watering-Hole Attacks

- A variant of DbD attack that is more targeted
- The attacker researches their intended victims to identify websites they are likely to visit, and then scans these sites to identify those with vulnerabilities that allow their compromise with a drive-by-download attack. Then wait for one of their intended victims to visit one of the compromised sites
- Their attack code may only infect systems belonging to the target organization, not for other visitors to the site. This helps evade detection

Malvertising

- Places malware on websites without actually compromising them
- The attacker pays for advertisements that to be placed on their intended target websites and incorporate malware in them
- Using these malicious ads, attackers can infect visitors to sites displaying them
- The malware code may be dynamically generated to either reduce the chance of detection or to only infect specific systems
- Attackers can place these ads for as little as a few hours, when they expect their intended victims could be browsing the targeted websites, greatly reducing their visibility

215

Clickjacking

- Also known as a User-Interface (UI) redress attack, used to collect an infected user's clicks or keystrokes
  - 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 websites that might have malicious code
  - By taking advantage of Adobe Flash or JavaScript an attacker could place a button under or over a legitimate button making it difficult for users to detect
  - A user can be led to believe that he/she is typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

216

Spam

- Spam is a significant carrier of malware.
  - The e-mail may have an attached document, which, if opened, may exploit a software vulnerability to install malware on the user's system.
  - Or, it may have an attached trojan horse program
- It may be used in a phishing attack
  - Directing the user either to a fake web site that mirrors some legitimate service, such as an online banking site, where it attempts to capture the user's login and password;
  - Or to complete some form with personal details to allow the attacker to impersonate the user in an identity theft.
- Requires the user's active choice to click on email attachments, or to permit the installation of some program.

Trojan Horses

- A Trojan horse is a useful program containing hidden code that, when invoked, performs some unwanted or harmful function.
  - e.g., it may scan local files for sensitive information and send a copy of it to the attacker.
  - It may be bundled with a game or useful program, and made available on a software distribution site or app store.
    - Be careful with "cracked software" (破解软件)
- Three models:
  - Continuing to perform the function of the original program and additionally performing a separate malicious activity
  - Continuing to perform the function of the original program but modifying the function to perform malicious activity (e.g., a Trojan version of a login program that collects passwords) or to hides other malicious activity (e.g., a Trojan version of a Linux "ps" program that does not display certain processes that are malicious)
  - Performing a malicious function that completely replaces the function of the original program

Outline

- Types of Malicious Software (Malware)
- Advanced Persistent Threat (APT)
- Propagation
  - Infected Content – Viruses
  - Vulnerability Exploit -- Worms
  - Social Engineering -- Spam E-Mail, Trojans
- Payload
  - System Corruption
  - Attack Agent -- Zombie, Bots
  - Information Theft -- Keyloggers, Phishing, Spyware
  - Stealthing -- Backdoors, Rootkits
- Countermeasures

Payload

- System Corruption
  - Data Destruction, Real-World Damage, Logic Bomb
- Attack Agent
  - Zombie, Bot
- Information theft
  - Keyloggers, Phishing, Spyware
- Stealthing
  - Backdoors, rootkits

Payload -- System Corruption

- Data destruction and ransom ware
  - Corrupts or encrypts the file system
  - Demands a ransom (in bitcoin) to recover
- Real-world damage
  - Causes damage to physical equipment, e.g., the Stuxnet worm that targeted Industrial Control System in Iran's nuclear facility
- Logic bomb
  - Code embedded in the malware that is set to "explode" when certain trigger conditions are met
  - Example trigger conditions include: presence or absence of certain files or devices on the system, a particular day of the week or date, a particular version or configuration of some software, or a particular user running the application, e.g., if I am deleted from the Payroll Table, encrypt hard disk.

Payload – Attack Agents: Zombies, Bots

- Takes over another computer and uses that computer to launch or manage attacks
- Botnet - collection of bots capable of acting in a coordinated manner for
  - Distributed Denial-of-Service (DDoS) attacks
  - Spamming
  - Sniffing traffic
  - Keylogging
  - Spreading new malware
  - Installing browser add-ons for advertisement
  - Manipulating online games
- Differences between bots and worms
  - Worms propagate and activate themselves
  - Bots are controlled from some central command-and-control facility

Payload – Information Theft Keyloggers and Spyware

- Keylogger
  - Captures keystrokes to allow attacker to monitor sensitive information
  - Typically uses some form of filtering mechanism that only returns information close to keywords ("login", "password")
- Spyware
  - Subverts the compromised machine to allow monitoring of a wide range of activity on the system
  - Monitoring history and content of browsing activity
  - Redirecting certain Web page requests to fake sites
  - Dynamically modifying data exchanged between the browser and certain Web sites of interest

Payload – Information Theft Phishing

- Exploits social engineering to leverage the user's trust by masquerading as communication from a trusted source
  - Include a URL in a spam e-mail that links to a fake Web site that mimics the login page of a banking site
- Spear-phishing
  - Recipients are carefully researched by the attacker
  - E-mail is crafted to specifically suit its recipient, often quoting a range of information to convince him of its authenticity

Payload – Stealthing – Backdoor

- Secret entry point into a program allowing the attacker to gain access and bypass the security access procedures
  - *Maintenance hook* is a backdoor used by Programmers to debug and test programs
    - e.g., if username is "133t h4ck0r" return LOGIN_SUCCESS;
  - Can also be implemented as a network service listening on some non-standard port that the attacker can connect to and issue commands to be run on the system.

Payload – Stealthing – Rootkit

- A rootkit is a set of hidden programs installed on a system to maintain covert access to that system
  - Stealthing means hiding itself from detection (stealthing)
- Gives root privileges to attacker
  - Can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand

Rootkit Classification

- Early rootkits worked in user mode, modifying utility programs and libraries in order to hide their presence.
  - Can be easily detected by code in the kernel, as this operated in the layer below the user
- Later rootkits operate at the level of OS kernel, or Virtual Machine Monitor (hypervisor), if virtualization is used
  - The lower the level, the harder it is to detect

Example Kernel-Mode Rootkit

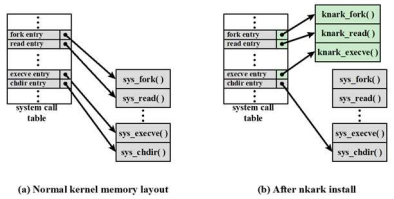


Figure 6.3 System Call Table Modification by Rootkit

The rootkit modifies the system call table, or redirects it to a new table in another kernel memory address, to invoke malicious system calls.

Outline

- Types of Malicious Software (Malware)
- Advanced Persistent Threat (APT)
- Propagation
  - Infected Content – Viruses
  - Vulnerability Exploit -- Worms
  - Social Engineering -- Spam E-Mail, Trojans
- Payload
  - System Corruption
  - Attack Agent -- Zombie, Bots
  - Information Theft -- Keyloggers, Phishing, Spyware
  - Stealthing -- Backdoors, Rootkits
- Countermeasures

Malware Countermeasures

- Prevention:
  - Harden systems and users in preventing infection
- Detection:
  - Once the infection has occurred, determine that it has occurred and locate the malware.
- Identification:
  - Once detection has been achieved, identify the specific malware that has infected the system.
- Removal:
  - Once the specific malware has been identified, remove all traces of malware virus from all infected systems so it cannot spread further.

Malware Countermeasure Approaches

- Host-based anti-virus scanners
- Sandbox analysis
- Host-based dynamic malware analysis
- Perimeter scanning approaches
  - Ingress & egress monitors

Host-Based Anti-Virus Scanners

- First generation: simple scanners
  - Requires a malware signature to identify the malware
  - Limited to the detection of known malware
- Second generation: heuristic scanners
  - Uses heuristic rules to search for probable malware instances
  - Another approach is integrity checking
- Third generation: activity traps
  - Memory-resident programs that identify malware by its actions rather than its structure in an infected program
- Fourth generation: full-featured protection
  - Packages consisting of a variety of anti-virus techniques used in conjunction
  - Include scanning and activity trap components and access control capability

Sandbox Analysis

- Running potentially malicious code in an emulated sandbox or on a virtual machine
- Allows the code to execute in a controlled environment where its behavior can be closely monitored without threatening the security of a real system
- This enables the detection of complex encrypted, polymorphic, or metamorphic malware
- The most difficult design issue with sandbox analysis is to determine how long to run each interpretation

Host-Based Dynamic Malware Analysis

- Monitors program behavior in real time for malicious action, and blocks potentially malicious actions before they have a chance to affect the system, e.g.,
  - Attempts to open, view, delete, and/or modify files;
  - Attempts to format disk drives;
  - Modifications to executable files or macros;
  - Modifications to critical system settings;
  - Scripting of e-mail and instant messaging clients to send executable content;
  - Initiation of network connections.
- Pros: real-time protection, as opposed to offline anti-virus scanning
- Cons: malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked

Perimeter Scanning Approaches

- Anti-virus software can run on an organization's firewall and IDS (Intrusion Detection Service)
- May also include intrusion prevention measures, blocking the flow of any suspicious traffic
- Ingress monitors
  - Located at the border between the enterprise network and the Internet
  - One technique is to look for incoming traffic to unused local IP addresses
- Egress monitors
  - Located at the egress point of individual LANs as well as at the border between the enterprise network and the Internet
  - Monitors outgoing traffic for signs of scanning or other suspicious behavior

CH07 DoS Attacks

ZJU 2020

Outline

- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

Denial-of-Service (DoS)

- A form of attack on the availability of some service
- NIST definition: "An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space."

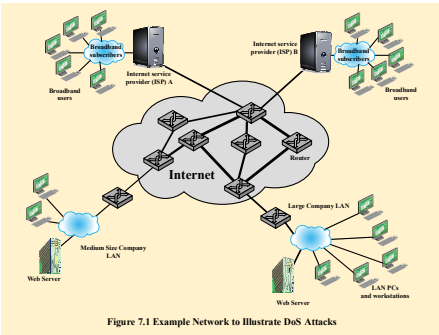


Figure 7.1 Example Network to Illustrate DoS Attacks

Resources that May be Attacked

- Network bandwidth
  - Overloads the capacity of the network links connecting to the Internet through an Internet Service Provider (ISP)
- System resources
  - Overloads and crashes the network handling software
- Application resources
  - Issues valid requests to a specific application, each of which consumes significant resources

Outline

- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

Ping Flood

- The UNIX ping utility is implemented using "echo request" and "echo reply" messages in ICMP (Internet Control Message Protocol).
- Attacker generates large volumes of ICMP packets with the target system as the destination address, in order to overwhelm the capacity of the network connection to the target organization and causing network congestion
  - Attacker needs access to higher-capacity network connection than the target system
- If the real source address is used in echo request packets, then source of the packets is clearly identified
  - The attacker can be easily identified by its IP address
  - The targeted system responds to each ICMP echo request packet with an ICMP echo response packet directed back to the sender, thus reflects the attack back at the source system
- Hence flooding ping attack packets need to use a spoofed source address

Source Address Spoofing

- Use forged source addresses
  - Usually via the raw socket interface on operating systems
    - Raw sockets allow direct sending and receiving of IP packets without any protocol-specific transport layer formatting
  - Makes attacking systems harder to identify
- Backscatter traffic
  - Response packets to spoofed-source packets
  - Monitoring the type of packets gives information on the type and scale of attacks
  - Security researchers take blocks of unused IP addresses, called a honeypot, advertised routes to them, and then collected details of any packets sent to these addresses.

TCP SYN Flooding

- An attack on system resources, specifically the network handling code in the operating system
  - Attacks the ability of a server to respond to future connection requests by overflowing the tables used to manage them, thus legitimate users are denied access to the server
  - Attacker does not need access to a high-capacity network connection
  - Also called SYN Spoofing attack

TCP Three-Way Handshake

- Client initiates the request for a TCP connection by sending a SYN packet to the server. This identifies the client's address and port number and supplies an initial sequence number x
- Upon receiving the SYN packet, the server records all the details about this request in a table of known TCP connections. It then responds to the client with a SYN-ACK packet. This includes a sequence number for the server y, and increments the client's sequence number x+1.
- Upon receiving the SYN-ACK packet, the client sends an ACK packet to the server with an incremented server sequence number y+1 and marks the connection as established.
- Upon receiving the ACK packet, the server also marks the connection as established

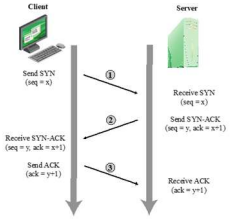


Figure 7.2 TCP Three-Way Connection Handshake

TCP SYN Flooding

- The attacker generates a large number of SYN connection request packets with forged source addresses.
- The server records the details of the TCP connection request and sends the SYN-ACK packet to the claimed source address
  - If there is a system with that address, it will respond with a RST (reset) packet, and the server deletes the TCP connection
  - If the address is non-existent, then no reply. The server will resend the SYN-ACK packet a number of times before finally assuming the connection request has failed and deleting the TCP connection
- If the server's TCP connection table is kept full, then no further TCP connections can be established

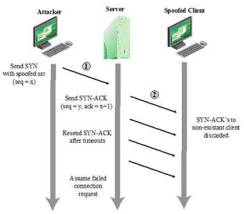


Figure7.3 TCP SYN Spoofing Attack

Outline

- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

Flooding Attacks

- Intent is to overload the network capacity on some link to a server
- Any type of network packet can be used
- ICMP flood
  - Classic Ping flood attack
- UDP flood
  - Sends UDP packets directed to some port number on the target system, e.g., diagnostic echo service

TCP Flooding Quiz

- Can you do TCP flooding with regular TCP packets instead of SYN packets?
- ANS: No. After a TCP connection is established, both client and server would know each other's IP address due to the three-way handshake mechanism, so it is not possible for the attacker to have source address spoofing, and any flooding attack is easily detected.

Outline

- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

Distributed Denial-of-Service (DDoS)

- Use of multiple computers to generate attacks
- Attacker gains access and installs their program on it, turning it into a zombie
- Large collections of such computers under the control of one attacker's control can be created, forming a botnet

DDoS Attack Architecture

- A small number of handler zombies control a large number of agent zombies, forming a control hierarchy

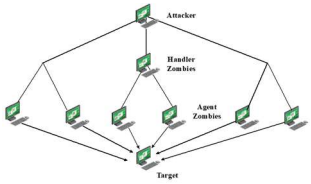


Figure 7.4 DDoS Attack Architecture

Outline

- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

SIP Attack

- Session Initiation Protocol (SIP) for Voice over IP (VoIP): Alice's user agent runs on a computer, and Bob's user agent runs on a cell phone. Alice's user agent sends an INVITE SIP request to the proxy server indicating its desire to invite Bob's user agent into a session. The proxy server uses a DNS server to get IP address of Bob's proxy server, and forwards the INVITE request to it, which then forwards the request to Bob's user agent, causing Bob's phone to ring.
- Since a single INVITE request triggers considerable resource consumption, the attacker can issue numerous INVITE requests with spoofed IP addresses, puts a load on the computation and network resources of SIP proxy servers.
- Call receivers are also victims of this attack. Bob's phone will be flooded with numerous VoIP calls.

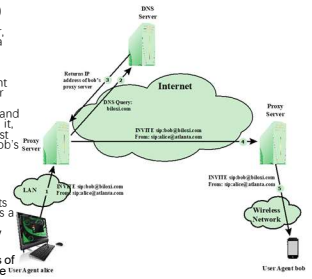


Figure 7.5 SIP INVITE Scenario

HTTP Based Attacks: HTTP flood

- Bombard Web servers with HTTP requests that consume a lot of resources
  - E.g., an HTTP request to download a large file from the target causes the Web server to read the file from hard disk, store it in memory, convert it into a packet stream, and then transmit the packets. This process consumes memory, processing, and network resources.
  - Since HTTP is based on TCP, HTTP requests themselves cannot be used as attack packets (refer to "TCP Flooding Quiz").
- **Recursive HTTP flood** (spidering)
  - Bots starting from a given HTTP link and following all links on the provided Web site in a recursive way

HTTP Based Attacks: Slowloris

- HTTP protocol specification (RFC2616) states that a blank line must be used to indicate the end of the request header and the beginning of the payload.
- Slowloris attack sends HTTP requests that does not include the terminating newline, so web server keeps connection open waiting for the newline that never arrives. Each connection occupies a thread, so many connections complete, eventually using up all threads in web server's thread pool size
- Existing IDS that relies on signatures to detect attacks will not recognize Slowloris since it sends legitimate HTTP traffic
- Countermeasure: delayed binding
  - The load balancer performs an HTTP request header completeness check, which means that the HTTP request will not be sent to the appropriate Web server until the final newlines are sent by the HTTP client.

Outline

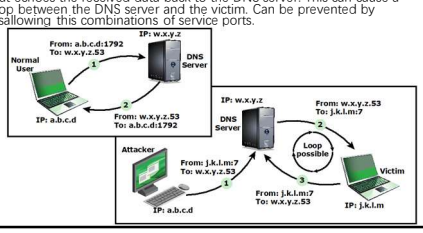
- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

Reflection Attacks

- Attacker sends packets to a known service on the intermediary server with a spoofed source address pointing to the target system
- Goal is to generate enough volumes of packets to flood the link to the target system without alerting the intermediary server
  - e.g. attacker sends a number of TCP SYN packets with spoofed source address as the target system's IP address to an intermediary server, which responds with a SYN-ACK packet to the target system
  - Target system will respond with a RST packet for each SYN-ACK packet, so intermediary server's TCP connection table will not be filled.
  - Can use a large number of intermediary servers
- Difference from "Fig. 7.3: SYN spoofing attack"
  - Target system is spoofed source address, not the server
  - Goal is to flood the network link to the target, not to fill the TCP connection table of the server

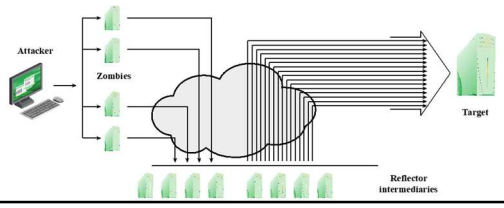
DNS Reflection Attack (Fig. 7.6)

- A variant of reflector attack.
- The attacker sends a query to the DNS server with a spoofed IP source address of j.k.l.m; this is the IP address of the target; and port 7, associated with echo service. The DNS server then sends a response to the victim of the attack, j.k.l.m:7. If the victim offers the echo service, it creates a packet that echoes the received data back to the DNS server. This can cause a loop between the DNS server and the victim. Can be prevented by disallowing this combinations of service ports.



Amplification Attack (Fig. 7.7)

- Another variant of reflector attack.
- The attacker sends packets with spoofed source IP addresses to the broadcast IP address of a subnet (IP Directed Broadcast), where the target system is. Each original packet triggers multiple broadcast response packets from all hosts on the subnet.
- Defense is to disallow IP Directed Broadcast, e.g., Cisco router command "no ip directed-broadcast"



DNS Amplification Attacks

- Leverage legitimate DNS servers as the intermediary system.
- Attacker issues DNS queries containing the spoofed source IP address of the target system
  - Using the classic DNS protocol, a 60-byte UDP request packet can easily result in a 512-byte UDP response, the maximum traditionally allowed.
  - The extended DNS protocol allows much larger responses of over 4 KB to support extended DNS features such as IPv6, security, etc.
- Different from DNS Reflection Attack, this attack exploits DNS behavior to convert a small request to a much larger response (amplification) to flood the target with responses.

Outline

- Introduction
- Classic DoS attacks
  - Ping flood
  - SYN spoofing
- Flooding attacks
  - ICMP (Ping), UDP, TCP SYN flood
- DDoS attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflection and amplification attacks
  - Reflection attacks
  - Amplification attacks
- Defenses and responses

DoS Attack Defenses

- DoS attacks cannot be prevented entirely
  - High traffic volumes may be legitimate
  - High publicity about a specific site
  - Activity on a very popular site
  - Described as slashdotted, flash crowd, or flash event
- Four lines of defense against DoS attacks
  - Before attack: attack prevention and preemption
  - During the attack: attack detection and filtering
  - During and after the attack: attack source traceback and identification
  - After the attack: attack reaction

DoS Attack Prevention

- Block spoofed source addresses
  - On routers as close to source as possible, by routers with knowledge of the valid address ranges of its local subnet
- Can be implemented with
  - An ISP (Internet Service Provider) knows which addresses are allocated to all its customers. The ISP can install access control rules in its routers to ensure that the source address of any packet from a customer is valid
  - Check the validity of the claimed source IP address, e.g., Cisco router command "ip verify unicast reverse-path" verifies the reachability of the source IP address in packets being forwarded. If the source IP address is not valid, the packet is discarded

SYN Spoofing Attack Prevention

- Random drop
  - Use modified TCP connection handling code that drop an entry for a random incomplete connection from the TCP connections table when it overflows
  - Based on the assumption that the majority of the entries in an overflowing table result from the attack, it is more likely that the dropped entry will correspond to an attack packet.
  - If a legitimate connection is dropped, the connection is reset, and the client will retry to re-establish the connection.
- Rate-limiting on select packet types,
  - e.g. ICMP packets or UDP packets to certain services (echo service)
- Modify TCP/IP network parameters
  - Increase size of the TCP connections table or reduce timeout period used to remove TCP connections from this table when no response is received.

Other Prevention Techniques

- To prevent reflection attacks, block suspicious services and combinations of source and destination ports
- To prevent broadcast amplification attacks, block IP directed broadcasts
- To prevent attacks on application resources, use a graphical puzzles to distinguish humans from bots
  - captcha, Completely Automated Public Turing Test to Tell Computers and Humans Apart
- Use mirrored and replicated servers when high-performance and reliability is required

Responding to DoS Attacks

- Identify type of attack
  - Capture and analyze packets
  - Design filters to block attack traffic upstream
  - Or identify and correct system/application bug
- Have ISP trace packet flow back to source
  - May be difficult and time consuming if spoofed IP source addresses are used
  - Necessary if planning legal action
- Implement contingency plan
  - Switch to alternate backup servers
  - Commission new servers at a new site with new addresses
- Update incident response plan
  - Analyze the attack and make a plan of response for the future

CH08 Intrusion Detection

ZJU 2020

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

Definition

- Security Intrusion:
  - Unauthorized act of bypassing the security mechanisms of a system
- Intrusion Detection:
  - A hardware or software function that gathers and analyzes information from various areas within a computer or a network to identify possible security intrusions

Examples of Intrusion

- Remote root compromise
- Web server defacement
- Guessing/cracking passwords
- Copying databases containing credit card numbers
- Viewing sensitive data without authorization
- Running a packet sniffer
- Distributing pirated software
- Using an unsecured modem to access internal network
- Impersonating an executive to get information
- Using an unattended workstation

Intruder Behavior

- Target acquisition and information gathering
- Initial access
- Privilege escalation
- Information gathering or system exploit
- Maintaining access
- Covering tracks

272

Intrusion Detection System (IDS)

- Three types of IDS
  - Host-based IDS (HIDS)
    - Monitors the characteristics of a single host for suspicious activity
  - Network-based IDS (NIDS)
    - Monitors network traffic and analyzes network, transport, and application protocols to identify suspicious activity
  - Distributed or hybrid IDS
    - Combines information from a number of sensors, often both host and network based, in a central analyzer that is able to better identify and respond to intrusion activity
- Three components of IDS
  - Sensors - collect data
  - Analyzers - determine if intrusion has occurred
  - User interface - view output or control system behavior

273

Detecting Intruders

- Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors.
  - False positives: false alarms, where authorized users are identified as intruders.
  - False negatives: intruders not identified as intruders.
- Moving detection threshold to the left will lead to more false negatives, since user is LESS likely to be identified as intruders

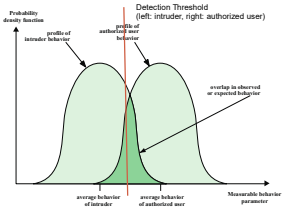


Figure 8.1 Profiles of Behavior of Intruders and Authorized Users

274

IDS Requirements

- Run continually with minimal human supervision.
- Be fault tolerant: recover from system crashes and reinitializations.
- Resist subversion: 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 up to monitor a large number of hosts.
- Provide graceful degradation of service: if some components of the IDS stop working, the rest of them should be affected as little as possible.
- Allow dynamic reconfiguration: the ability to reconfigure the IDS without having to restart it.

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

276

Analysis Approaches

- Anomaly detection
  - Defines normal, or expected, behavior by collecting data relating to the behavior of legitimate users over a period of time
  - Identify intruders whose behavior deviates from normal behavior
  - Can detect unknown attacks
- Signature/Heuristic detection
  - Defines malicious or unauthorized behavior, using a set of known malicious data patterns or attack rules that are compared with current behavior
  - Can only detect known attacks seen before

Anomaly Detection Techniques

- Three approaches to anomaly detection
- Statistical
  - Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics
- Knowledge based
  - A set of rules that model legitimate behavior developed by experts
- Machine-learning
  - Approaches automatically determine a suitable classification model that can classify data as either normal or anomalous

Signature/Heuristic Detection Techniques

- Signature approaches
  - Match a large collection of known patterns of malicious data against data stored on a system or in transit over a network
  - The signatures need to be large enough to minimize the false positive rate, while still detecting a sufficiently large fraction of malicious data
  - Widely used in anti-virus products, network traffic scanning proxies, and in NIDS
- 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
  - SNORT is an example of a rule-based NIDS

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

Host-based IDS

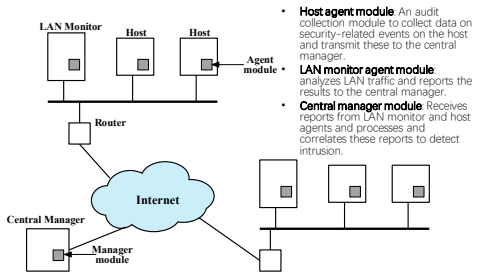
- Adds a specialized layer of security software to vulnerable or sensitive systems
- Can use either anomaly or signature/heuristic detection techniques
- Monitors activity to detect suspicious behavior
  - To detect intrusions, log suspicious events, and send alerts
  - Can detect both external and internal intrusions (outside or inside the firewall)
- Common data sources include:
  - System call traces
  - Audit (log file) records
  - File integrity checksums
  - Registry access

System Calls

(a) Ubuntu Linux System Calls

accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon, auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve, exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat, fstatfs, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getdopt, getdtablesize, getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize, getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt, gettimeofday, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mcfl, mincore, mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap, nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace, putmsg, quota, quotactl, read, readlink, readv, reboot, recv, recvfrom, recvmsg, rename, resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname, setdopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp, setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid, shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec, socket, socketaddr, socketpair, sstk, stat, stat, statfs, stime, stty, swapon, symlink, sync, sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustat, utime, utimes, vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4, write, writev

- System calls are the means by which user programs access core kernel functions
- System call traces can be analyzed to detect anomalies



- **Host agent module**: An audit collection module to collect data on security-related events on the host and transmit these to the central manager.
- **LAN monitor agent module**: analyzes LAN traffic and reports the results to the central manager.
- **Central manager module**: Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

Figure 8.2 Architecture for Distributed Intrusion Detection

283

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

284

Network-based IDS (NIDS)

- Monitors traffic at selected points on a network
- Examines traffic packet by packet in real or close to real time
- May examine network, transport, and/or application-level protocol activity
- Comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human operator
- Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two

Inline vs. Passive Sensor

- Sensors can be deployed in one of two modes: inline and passive.
- Inline sensors: the traffic being monitored must pass through the sensor. Can be implemented on firewall machines or standalone NIDS machine
- Passive sensors: monitors a copy of network traffic, and does not cause delays to the actual traffic

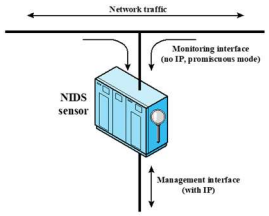


Figure 8.4 Passive NIDS Sensor

286

NIDS Sensor Deployment Locations

- Location 1:just inside the external firewall
- Location 2: just outside the external firewall
- Location 3: Just inside an internal firewall to protect major backbone networks for internal servers and database resources
- Location 4: Just inside an internal firewall to protect LANs that support user workstations and servers s

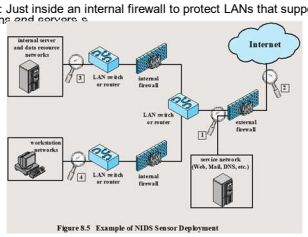


Figure 8.5 Example of NIDS Sensor Deployment

287

Intrusion Detection Techniques

- Signature detection for:
  - Reconnaissance and attacks at application, transport, or network layer
  - Unexpected application services
  - Policy violations (forbidden websites or protocols)
- Anomaly detection for:
  - Denial-of-service (DoS) attacks
  - Scanning
  - Worms

288

Logging of Alerts

- Typical information logged by a NIDS sensor includes:
  - Timestamp
  - Connection or session ID
  - Event or alert type
  - Rating (e.g., priority, severity, impact, confidence)
  - Network, transport, and application layer protocols
  - Source and destination IP addresses
  - Source and destination TCP or UDP ports, or ICMP types and codes
  - Number of bytes transmitted over the connection
  - Decoded payload data, such as application requests and responses
  - State-related information (e.g., authenticated username)

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

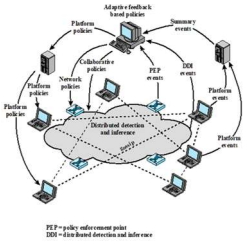


Figure 8.6 Overall Architecture of an Autonomic Enterprise Security System

Autonomic Enterprise Security System, developed by Intel, does not rely solely on perimeter defense mechanisms, such as firewalls, or on individual host-based defenses. Instead, each end host and each network device (e.g., routers) is considered to be a potential sensor and may have the sensor software module installed. The sensors exchange information to corroborate the state of the network (i.e., whether an attack is under way).

An Analogy to Illustrate AESS

- Suppose a single host is subject to a prolonged attack and that the host is configured to minimize false positives. Early on in the attack, no alert is sounded because the risk of false positive is high.
- If the attack persists, the evidence that an attack is under way becomes stronger and the risk of false positive decreases. However, much time has passed.
- Now consider many local sensors, each of which suspect the onset of an attack and all of which collaborate. Because numerous systems see the same evidence, an alert can be issued with a low false positive risk.
- Thus, instead of a long period of time, we use a large number of sensors to reduce false positives and still detect attacks. A number of vendors now offer this type of product.

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

IETF Intrusion Detection Working Group

- Purpose is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to management systems that may need to interact with them
- The working group issued the following RFCs in 2007:

Intrusion Detection Message Exchange Requirements (RFC 4766)
- Document defines requirements for the Intrusion Detection Message Exchange Format (IDMEF)
- Also specifies requirements for a communication protocol for communicating IDMEF

The Intrusion Detection Message Exchange Format (RFC 4765)
- Document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model
- An implementation of the data model in the Extensible Markup Language (XML) is presented, and XML Document Type Definition is developed, and examples are provided

The Intrusion Detection Exchange Protocol (RFC 4767)
- Document describes the Intrusion Detection Exchange Protocol (IDXP), an application level protocol for exchanging data between intrusion detection entities
- IDXP supports mutual authentication, integrity, and confidentiality over a connection oriented protocol

IETF Intrusion Detection Message Exchange

- **The sensor** monitors data sources looking for suspicious activity.
- **The sensor** communicates suspicious activity to **the analyzer** as an event, which characterizes an activity within a given period of time.
- If **the analyzer** determines that the event is of interest, it sends an alert to the **manager component** that contains information about the unusual activity that was detected, as well as the specifics of the occurrence.
- The **manager component** issues a notification to **the human operator**.
- A response can be initiated automatically by the **manager component** or by the **human operator**. Examples include logging the activity; recording the raw data (from the data source) that characterized the event; terminating a network, user, or application session; or altering network or system access controls.
- **The security policy** defines what activities are allowed to take place on an organization's network. This includes, but is not limited to, which hosts are to be denied external network access.

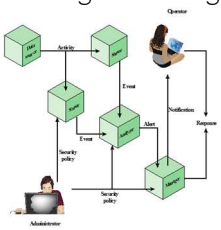


Figure 8.7 Model For Intrusion Detection Message Exchange

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

296

Honeypots

- Decoy systems designed to:
  - Collect information about the attacker's activity
  - Encourage the attacker to stay on the system long enough for administrators to respond
- Systems are filled with fabricated information that a legitimate user wouldn't access
- Resources that have no production value
  - Incoming communication is most likely a probe, scan, or attack
  - Initiated outbound communication suggests that the system is likely compromised

297

Honeypot Classifications

- Low interaction honeypot
  - Consists of a software package that emulates particular IT services or systems well enough to provide a realistic initial interaction, but does not execute a full version of those services or systems
  - Provides a less realistic target
  - Often sufficient for use as a component of a distributed IDS to warn of imminent attack
- High interaction honeypot
  - A real system, with a full operating system, services and applications, which are instrumented and deployed where they can be accessed by attackers
  - Is a more realistic target that may occupy an attacker for an extended period
  - However, it requires significantly more resources
  - If compromised could be used to initiate attacks on other systems

298

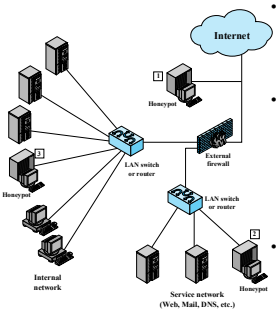


Figure 8.8 Example of Honeypot Deployment

- Location 1: outside the external firewall
  - does not increase the risk for the internal network
  - cannot detect internal attacks
- Location 2: within the externally available services, such as Web and mail, called the DMZ (demilitarized zone)
  - must assure that the other systems in the DMZ are secure against any activity generated by the honeypot
- Location 3: within the LAN
  - can detect internal attacks
  - if the honeypot is compromised, it can attack other internal systems

Outline

- Intrusion detection basics
- Analysis approaches
  - Anomaly detection
  - Signature/heuristic detection
- Host-based intrusion detection
- Network-based intrusion detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Example system: Snort IDS

300

Snort IDS

- Snort is an open source host-based or network-based lightweight IDS.
  - **Packet decoder**: processes each captured packet to identify and isolate protocol headers
  - **Logger**: For each packet that matches a rule, the rule specifies what logging and alerting options are to be taken.
  - **Alert**: For each detected packet, an alert can be sent. The alert option in the matching rule determines what information is included in the event notification, and where to send it to.

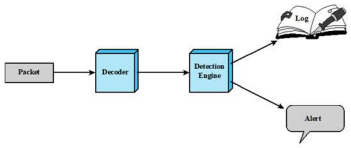


Figure 8.9 Snort Architecture

Snort Rule Actions

Action	Description
alert	Generate an alert using the selected alert method, and then log the packet.
log	Log the packet.
pass	Ignore the packet.
activate	Alert and then turn on another dynamic rule.
dynamic	Remain idle until activated by an activate rule , then act as a log rule.
drop	Make iptables drop the packet and log the packet.
reject	Make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
sdrop	Make iptables drop the packet but does not log it.

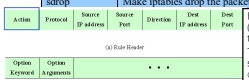


Figure 8.10 Snort Rule Formats

Example: alert tcp any any -> 192.168.1.0/24 (content: "mail from: root"; msg: "root user attempts to send an email")

This rule monitors TCP traffic on any host on the /24 subnet, and sends an alert if a mail from root is detected)

/24 means that the first 24 bits of the IP address denotes the subnet with mask of 255.255.255.0 , so the subnet address is 192.168.1; the last part is the host address (1-254). (255 is the broadcast addr.)

302

CH09 Firewalls and Intrusion Prevention

ZJU 2020

303

Outline

- Introduction
- Types of firewalls
  - Packet filtering firewall
  - Stateful inspection firewalls
  - Application-level gateway
- Firewall basing
  - Bastion host
  - Host-based firewalls
  - Personal firewall
- Firewall location and configurations
  - DMZ networks
  - VPNs
  - Distributed firewalls
- Intrusion prevention systems
  - Host-based IPS
  - Network-based IPS
  - Snort inline
- Unified Threat Management Products

304

Firewalls

- Effective means of protecting LANs
- Inserted between the premises network and the Internet to establish a controlled link
- Used as a perimeter defense
  - Single choke point that insulates the internal systems from external networks
  - Only authorized traffic as defined by the local security policy will be allowed to pass
- Direction
  - Ingress: for incoming traffic
    - Block outside users from accessing certain intranet computers
  - Egress: for outgoing traffic
    - Block inside users from accessing certain outside websites like Facebook

305

Firewall Access Policy

- A critical component in the planning and implementation of a firewall is specifying a suitable access policy
  - This lists the types of traffic authorized to pass through the firewall
  - Includes IP address ranges, protocols, applications and content types
- This policy should be developed from the organization's information security risk assessment and policy, and which traffic types the organization needs to support

Firewall Filter Characteristics

- IP address and protocol values
  - Used by packet filter and stateful inspection firewalls to limit access to specific services
- Application protocol
  - Used by an application-level gateway that relays and monitors the exchange of information for specific applications, e.g., checking SMTP email for spam
- User identity
  - Controls access based on user identity, for users who identify themselves using some form of secure authentication (IPSec)
- Network activity
  - Controls access based on considerations such as the time of request, e.g., only in business hours; rate of requests, e.g., to detect scanning attempts.

Firewall Capabilities And Limits

- Capabilities:
  - Defines a single choke point
  - Provides a location for monitoring security events
  - Can serve as the platform for IPSec
- Limitations:
  - Cannot protect against attacks bypassing firewall
  - May not protect against internal threats
  - Improperly secured wireless LAN can be accessed from outside the organization
  - Mobile devices may be infected outside the corporate network then used internally

308

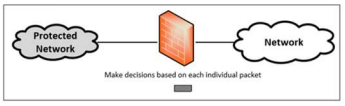
Outline

- Introduction
- Types of firewalls
  - Packet filtering firewall
  - Stateful inspection firewalls
  - Application-level gateway
- Firewall basing
  - Bastion host
  - Host-based firewalls
  - Personal firewall
- Firewall location and configurations
  - DMZ networks
  - VPNs
  - Distributed firewalls
- Intrusion prevention systems
  - Host-based IPS
  - Network-based IPS
  - Snort inline
- Unified Threat Management Products

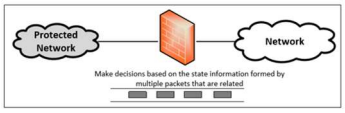
309

Types of Firewalls

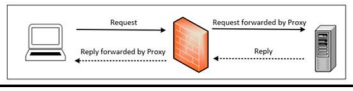
Packet Filtering Firewall



Stateful Firewall



Application Proxy



Packet Filtering Firewall

- Applies rules to each incoming and outgoing IP packet to decide to forward or discard it
- Two default policies:
  - Discard - prohibit unless expressly permitted
    - More conservative, controlled, visible to users
  - Forward - permit unless expressly prohibited
    - Easier to manage and use but less secure

Filtering rules are based on information contained in a network packet

- Source IP address
- Destination IP address
- Source and destination transport-level address
- IP protocol info
- Which interface of the firewall the packet comes from or is destined for

TCP & SMTP

- Port numbers lower than 1024 are assigned permanently to particular applications (e.g., 25 for server SMTP). Port numbers between 1024 and 65535 are generated dynamically and have temporary significance only for the lifetime of a TCP connection.
- When a TCP client creates a session with a remote host, it creates a TCP connection in which the TCP port number for the remote (server) application is a number less than 1024 and the TCP port number for the local (client) application is a number between 1024 and 65535.
  - A simple packet filtering firewall must permit inbound network traffic on all these high-numbered ports for TCP-based traffic to occur. This creates a vulnerability that can be exploited by unauthorized users.
- Simple Mail Transfer Protocol (SMTP)
  - SMTP operates by setting up a TCP connection between client and server for sending emails. The SMTP server's TCP port number is 25; The SMTP client's TCP port number is any number between 1024 and 65535 that is generated dynamically by the SMTP client, and only valid during the email session.

Packet-Filtering Rules Example

- This firewall only allows SMTP traffic through it (inbound or outbound).
- Rules A and B allow incoming email:
  - A. Allow inbound SMTP connection to local SMTP server: Inbound TCP packets from an external source to port 25.
  - B. Allow outbound response to inbound SMTP connection: TCP packets to a destination port above 1023.
- Rules C and D allow outgoing email:
  - C. Allow outbound SMTP connection to remote SMTP server: Outbound TCP packets from an internal source to port 25.
  - D. Allow inbound response to outbound SMTP connection: Inbound TCP packets to a destination port above 1023.
- Rule E. Any packets that do not match rules A-D are disallowed.

Rule	Direction	Src Addr	Dest Addr	Protocol	Dest Port	Action
A	In	External	Internal	TCP	25	Permit
B	Out	Internal	External	TCP	>1023	Permit
C	Out	Internal	External	TCP	25	Permit
D	In	External	Internal	TCP	>1023	Permit
E	Either	Any	Any	Any	Any	Deny

Possible Attacks

- The original Rule D allows inbound traffic to any destination port above 1023. An external attacker can open a connection from his port 5150 to an internal Web proxy server on port 8080 to attack the server.
- To counter this attack, the firewall rule set can be configured with a source port field for each row. For rules B and D, the source port is set to 25; for rules A and C, the source port is set to >1023.
- This assumes the attacker cannot send attack packets from Port 25. Is it true?

Rule	Direction	Src Addr	Dest Addr	Protocol	Src Port	Dest Port	Action
A	In	External	Internal	TCP	>1023	25	Permit
B	Out	Internal	External	TCP	25	>1023	Permit
C	Out	Internal	External	TCP	>1023	25	Permit
D	In	External	Internal	TCP	25	>1023	Permit
E	Either	Any	Any	Any	Any	Any	Deny

Adding ACK Flag

- Using port 25 for SMTP receipt is only a default; the attacker's machine could be configured to have some other application linked to port 25 and send attack packets from it.
- **To counter this threat, we can add an ACK flag field to each row. For rule 4, the field would indicate that the ACK flag must be set on the incoming packet with a source port number of 25.**
- **The rule takes advantage of a feature of TCP connections. Once a connection is set up, the ACK flag of a TCP segment is set to acknowledge segments sent from the other side.**

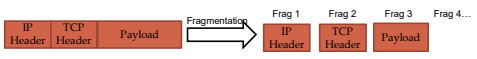
Rule	Direction	Src address	Src port	Dest address	Protocol	Dest port	Flag	Action
4	In	External	25	Internal	TCP	>1023	ACK	Permit

Packet Filter
Advantages And Weaknesses

- Advantages
  - Simple and efficient
  - Transparent to users
- Weaknesses
  - Cannot prevent attacks that employ application-level vulnerabilities, e.g., SQL injection attacks
    - e.g., cannot block specific application commands; if a packet filter firewall allows a given application, all functions available within that application will be permitted.
  - Limited logging functionality
    - Only source address, destination address, and traffic type.
  - Do not support advanced user authentication
    - Due to the lack of upper-layer functionality
  - Vulnerable to attacks on TCP/IP protocol bugs, e.g., IP address spoofing
  - Improper configuration of firewall policy can lead to breaches

Packet Filter
Attacks and Countermeasures

- IP address spoofing
  - Intruder transmits packets from the outside with a source IP address field containing an address of an internal host to attack systems that employ simple source address security, in which packets from trusted internal hosts are accepted.
  - Countermeasure is to discard packets with an inside source address if the packet arrives on an external interface.
- Source routing attack
  - Source routing allows a sender of a packet to partially or completely specify the route the packet takes through the network. In contrast, in non-source routing protocols, routers in the network determine the path based on the packet's destination. This can be used to bypass certain security measures on routers that do not analyze the source routing information.
  - Countermeasure is to discard all packets that use this option.
- Tiny fragment attack
  - Intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment in order to circumvent filtering rules that depend on TCP header information (port numbers). The attacker hopes that the filtering firewall examines only the first fragment (IP addresses without port numbers) and that the remaining fragments are passed through.
  - Countermeasure is to enforce a rule that the first fragment of a packet must contain a minimum amount of the TCP header. If the first fragment is rejected, the filter can remember the packet and discard all subsequent fragments.



Stateful Firewall

- Tightens rules for TCP traffic by creating a directory of TCP connections
  - There is an entry for each currently established connection
  - Incoming traffic to high numbered ports is allowed only for those packets that fit the profile of one of the entries in this directory
- Reviews packet information but also records information about TCP connections
  - Keeps track of TCP sequence numbers to prevent attacks that depend on the sequence number
  - Inspects data for protocols like FTP commands

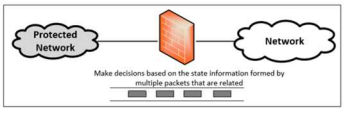


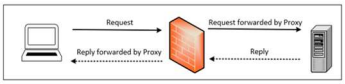
Table 9.2 Example Stateful Firewall Connection State Table

- Some stateful firewalls also keep track of TCP sequence numbers.

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

Application-Level Gateway

- Also called an application proxy. Acts as a relay of application-level traffic
  - User contacts the gateway using a TCP/IP application, such as Telnet or FTP
  - Gateway asks the user for the name of the remote host to be accessed.
  - User responds and provides a valid user ID and authentication information.
  - Gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.
- Tend to be more secure than packet filters
  - Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowed applications.
- Disadvantage is the additional processing overhead on each connection
  - Needs to remove original TCP/IP headers and add its own headers before relay.



Outline

- Introduction
- Types of firewalls
  - Packet filtering firewall
  - Stateful inspection firewalls
  - Application-level gateway
- Firewall basing
  - Bastion host
  - Host-based firewalls
  - Personal firewall
- Firewall location and configurations
  - DMZ networks
  - VPNs
  - Distributed firewalls
- Intrusion prevention systems
  - Host-based IPS
  - Network-based IPS
  - Snort inline
- Unified Threat Management Products

321

Bastion Host

- Standalone system identified as a critical strong point in the network's security; it serves as a platform for application-level gateways, or to support other services such as IPSec.
- Common characteristics:
  - Runs secure OS, only essential services, e.g., proxy applications for DNS, FTP, HTTP, and SMTP
  - Each proxy can restrict features, hosts accessed
  - Each proxy is small, simple, checked for security flaws
  - Each proxy is independent, non-privileged
  - 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 read only. This makes it difficult for an intruder to install Trojan horse sniffers or other dangerous files on the bastion host.

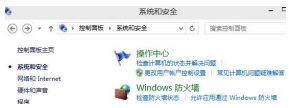
322

Host-Based Firewall

- Software module used to secure an individual host, typically a server
  - In contrast to network firewalls at the network perimeter
- Filter and restrict packet flows
- Advantages:
  - Filtering rules can be tailored to the host environment
  - Protection is provided independent of network topology
  - Provides an additional layer of protection

Personal Firewall

- Controls traffic between a personal computer or workstation and the Internet or enterprise network
- For both home or corporate use
- Typically is a software module on a personal computer
- Can be housed in a router that connects all of the home computers to a DSL, cable modem, or other Internet interface
- Typically much less complex than server-based or stand-alone (bastion host) firewalls
- Primary role is to deny unauthorized remote access
- May also monitor outgoing traffic to detect and block worms and malware activity



324

Outline

- Introduction
- Types of firewalls
  - Packet filtering firewall
  - Stateful inspection firewalls
  - Application-level gateway
- Firewall basing
  - Bastion host
  - Host-based firewalls
  - Personal firewall
- Firewall location and configurations
  - DMZ networks
  - VPNs
  - Distributed firewalls
- Intrusion prevention systems
  - Host-based IPS
  - Network-based IPS
  - Snort inline
- Unified Threat Management Products

Example Firewall Configuration

- DMZ (DeMilitarized Zone) (also called perimeter network) is a subnetwork that contains an organization's external-facing services, web, email, DNS servers.

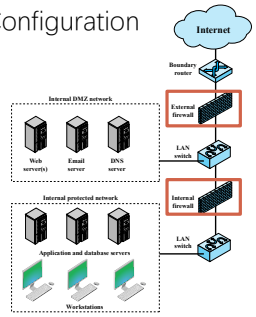


Figure 9.2 Example Firewall Configuration

External & Internal Firewalls

- The external firewall provides a measure of access control and protection for the DMZ systems consistent with their need for external connectivity, and provides a basic level of protection for the remainder of the enterprise network.
  - One example is the firewall for SMTP traffic.
- Internal firewalls serve 3 purposes:
  - 1. Have more stringent filtering rules in order to protect enterprise servers and workstations from external attack.
  - 2. Provides two-way protection with respect to the DMZ. First, it protects the remainder of the network from attacks launched from DMZ systems. Such attacks might originate from worms, rootkits, bots, or other malware lodged in a DMZ system. Second, it protects the DMZ systems from attack from the internal protected network.
  - 3. Multiple internal firewalls can be used to partition and protect portions of the internal network from each other.

VPN

- A Virtual Private Network (VPN) consists of a set of LANs interconnect by the public internet.

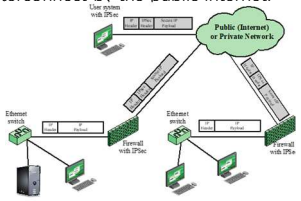
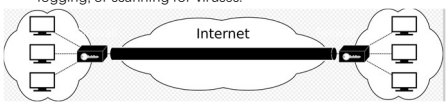


Figure 9.3 A VPN Security Scenario

VPN and IPSec

- IPSec has 2 modes:
  - Transport mode: only the payload of the IP packet is usually encrypted or authenticated.
  - Tunnel mode (for VPN): the entire IP packet is encrypted and authenticated. It is then encapsulated into a new IP packet with a new IP header.
- IPSec (tunnel mode) should be implemented within the firewall.
  - If IPSec is implemented in a separate box different from the firewall, then VPN traffic passing through the firewall in both directions is encrypted, so the firewall is unable to perform filtering or other security functions, such as access control, logging, or scanning for viruses.



IPSec tunnel mode

Distributed Configuration

- A *distributed firewall* configuration involves stand-alone firewall devices plus host-based firewalls working together under a central administrative control
- There may be both an internal DMZ and an external DMZ. Web servers with less critical information on them can be placed in an external DMZ, outside the external firewall.

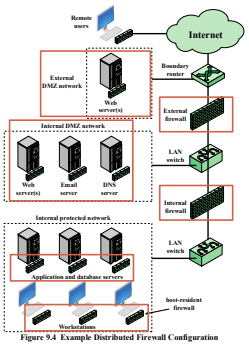


Figure 9.4 Example Distributed Firewall Configuration

Outline

- Introduction
- Types of firewalls
  - Packet filtering firewall
  - Stateful inspection firewalls
  - Application-level gateway
- Firewall basing
  - Bastion host
  - Host-based firewalls
  - Personal firewall
- Firewall location and configurations
  - DMZ networks
  - VPNs
  - Distributed firewalls
- Intrusion prevention systems
  - Host-based IPS
  - Network-based IPS
  - Snort inline
- Unified Threat Management Products

331

Intrusion Prevention Systems (IPS)

- Also known as Intrusion Detection and Prevention System (IDPS), combining IDS with firewall functionality
- Is an extension of an IDS that includes the capability to attempt to block or prevent detected malicious activity
- Can be host-based, network-based, or distributed/hybrid
- Can block traffic as a firewall does, but makes use of the types of algorithms developed for IDS to determine when to do so

Host-based IPS (HIPS)

- Can make use of either signature/heuristic or anomaly detection techniques to identify attacks
  - Signature: look for malicious patterns in the content of application network traffic, or in sequences of system calls, to identify known malicious behavior .
  - Anomaly: detect anomalies that deviate from normal behavior
  - c.f. L9-CH08-Intrusion Detection p. 10.
- Examples types of malicious behavior addressed by HIPS:
  - Modification of system resources
  - Privilege-escalation exploits (e.g., set-UID)
  - Buffer-overflow exploits
  - Access to e-mail contact list
  - Directory traversal

333

Protection Areas of HIPS

- System calls: The kernel controls access to system resources such as memory, I/O devices, and processor. To use these resources, user applications invoke system calls to the kernel. Any exploit code will execute at least one system call. The HIPS can be configured to examine each system call for malicious characteristics.
  - File system access: The HIPS can ensure that file access system calls are not malicious and meet established policy.
- System registry settings: The Windows registry maintains persistent configuration information about programs and is often maliciously modified to extend the life of an exploit. The HIPS can ensure that the system registry maintains its integrity.
- Host input/output: I/O communications, whether local or network based, can propagate exploit code and malware. The HIPS can examine and enforce proper client interaction with the network and its interaction with other devices.

The Role of HIPS

- Traditionally, endpoint security has been provided by a collection of distinct products, such as antivirus, antispyware, antispam, and personal firewalls
- The integrated HIPS approach provides an integrated, single-product suite of functions that work closely together
- HIPS may be used as the last line-of-defense in a defense-in-depth strategy, in addition to firewalls and/or network-based IPS

Network-Based IPS (NIPS)

- Inline NIDS with the authority to modify or discard packets and tear down TCP connections
- Makes use of signature/heuristic detection and anomaly detection
- Methods used to identify malicious packets:
  - Pattern matching: Scans incoming packets for specific byte sequences (signature) stored in a database of known attacks
  - Stateful matching: Scans for attack signatures in the context of a traffic stream (TCP connection) rather than individual packets
  - Protocol anomaly: Looks for deviation from standards set forth in RFCs
  - Traffic anomaly: Watches for unusual traffic activities, such as a flood of UDP packets or a new service appearing on the network
  - Statistical anomaly: Develops baselines of normal traffic activity and throughput, and alerts on deviations from those baselines

Snort Inline

- Snort Inline adds three new rule types for intrusion prevention, to enable Snort IDS to function as an IPS.
  - Drop
    - Packet is rejected and result is logged
  - Reject
    - Packet is rejected and result is logged, and an error message is returned e.g., TCP connection reset
  - Sdrop
    - Packet is rejected but not logged

Outline

- Introduction
- Types of firewalls
  - Packet filtering firewall
  - Stateful inspection firewalls
  - Application-level gateway
- Firewall basing
  - Bastion host
  - Host-based firewalls
  - Personal firewall
- Firewall location and configurations
  - DMZ networks
  - VPNs
  - Distributed firewalls
- Intrusion prevention systems
  - Host-based IPS
  - Network-based IPS
  - Snort inline
- Unified Threat Management Products

Unified Threat Management (UTM)

- One approach to reducing the administrative and performance burden is to replace all inline network products (firewall, IPS, IDS, VPN, antispam, anti-spyware…) with a single device that integrates a variety of approaches to dealing with network-based attacks.

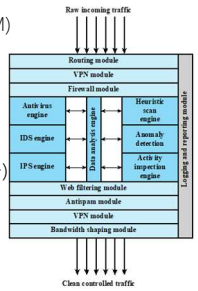


Figure 9.6 Unified Threat Management Appliance (based on [JAME06])

CH10 Buffer Overflow

ZJU 2020

Outline

- Introduction
- Stack overflows
- Defending against buffer overflows
  - Compile-time defenses
  - Run-time defenses
- Other forms of overflow attacks
  - Heap overflows
  - Global data area overflows
  - Integer overflows

Introduction to Buffer Overflow

- NIST Definition:
  - "A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system."
- A very common attack mechanism
  - First used by the Morris Worm in 1988
- Still of major concern
  - Buggy legacy code in widely deployed OSes and applications
  - Careless programming practices

Buffer Overflow Basics

- When a process attempts to store data beyond the limits of a fixed-sized buffer, it may overwrite adjacent memory locations that hold data or instruction.
  - Buffer could be located on the stack, in the heap, or in the global data section of the process
- Consequences:
  - Program data corruption
  - Unexpected transfer of control
  - Memory access violations
  - Execution of code chosen by attacker

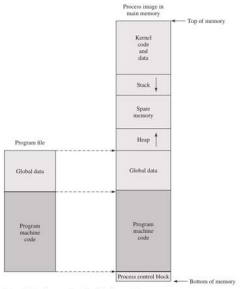


Figure 10.4 **Program Loading into Process Memory**

Buffer Overflow Attacks

- To exploit a buffer overflow an attacker needs:
  - To identify a buffer overflow vulnerability in some program that can be triggered using externally sourced data under the attacker's control
  - To understand how that buffer is stored in memory and determine potential for corruption
- Identifying vulnerable programs can be done by:
  - Inspection of program source
  - Tracing the execution of programs as they process oversized input
  - Using tools such as fuzzing to automatically identify potentially vulnerable programs

Outline

- Introduction
- Stack overflows
- Defending against buffer overflows
  - Compile-time defenses
  - Run-time defenses
- Other forms of overflow attacks
  - Heap overflows
  - Global data area overflows
  - Integer overflows

Stack Overflow

- Occurs when buffer is located on the stack
  - Also referred to as *stack smashing*
- Stacks are used
  - In function calls for allocation of memory in the stack frame for
    - Local variables
    - Parameters
    - Return address
- Stack overflow consequences:
  - Corruption of program data
  - Memory access violations
  - Transfer of control to execute malicious code

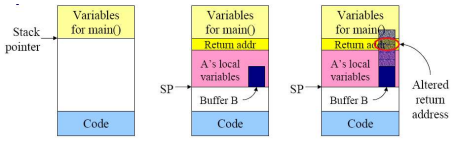
A Vulnerable Function

- Copies from its argument string `argstr` (of type `char *`) to its local variable `buffer[5]` on the stack
- What if `argstr[]` contains more than 5 chars?

```
int A(char *argstr)
{char buffer[5];
strcpy(buffer,argstr);
return 0;
}
```

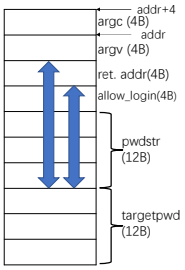

Stack Overflow Attack

- 1. Program starts running in main()
- 2. After procedure *A()* called
- 3. Buffer overflow alters the return address from A()
  - Can be garbage that causes program crash, or can be address of a malicious program, e.g., shellcode



Stack Overflow Attack Example

```
int main(int argc, char *argv[]) {
   int allow_login = 0;
   char pwdstr[12];
   char targetpwd[12] = "MyPwd123";
   gets(pwdstr); //No bounds check!
   if (strncmp(pwdstr,targetpwd, 12) == 0)
         allow_login = 1;
         if (allow_login == 0)
            printf("Login request rejected");
            else
            printf("Login request allowed");
}
```



Input pwdstr larger than 12 B will cause stack overflow and may overwrite allow_login and/or return address.

If allow_login is overwritten to be 1, then login is successful.

If return address is overwritten to point to malicious code, then control flow jumps to it after main() finishes.

(This example shows attack on function main(), which is similar to attack on function A() shown earlier.)

System stack grows downward (assuming a 32-bit CPU, so each word is 4 Bytes).

More Vulnerable Programs

- In both (a) and (b), **there is no bounds check on destination buffer size to[], hence the to[] array bound may be exceeded.**

```
int copy_buf(char *to, int pos, char *from, int len)
{
   int i;

   for (i=0; i<len; i++) {
      to[pos] = from[i];
      pos++;
   }
   return pos;
}
```

(a) Unsafe byte copy

```
short read_chunk(FILE fil, char *to)
{
   short len;
   fread(&len, 2, 1, fil); ..... /* read length of binary data */
   fread(to, 1, len, fil); ..... /* read len bytes of binary data
   return len;
}
```

(b) Unsafe byte input

Figure 10.10 Examples of Unsafe C Code

Unsafe C Standard Library Routines

- They should not be used before checking the total size of data being transferred.

`gets(char *str)`	read line from standard input into str
`sprintf(char *str, char *format, ...)`	create str according to supplied format and variables
`strcat(char *dest, char *src)`	append contents of string src to string dest
`strcpy(char *dest, char *src)`	copy contents of string src to string dest
`vsprintf(char *str, char *fmt, va_list ap)`	create str according to supplied format and variables

shellcode

- shellcode is code supplied by the attacker that creates a shell (command-line interpreter) that allows the attacker to execute any code he wants
  - Return address is overwritten to jump to shellcode, which is typically put on the stack by attacker.
  - On Linux: call execve ("/bin/sh") to replace the current program code with the Bourne shell
  - On Windows: call system("cmd.exe") to run the command shell

Variants of Buffer Overflow Attacks

- Target program can be:
  - Trusted system utility
  - Network service daemon
  - Commonly-used library code
- Shellcode functions may be:
  - Set up a listening service to launch a remote shell when connected to
  - Create a reverse shell that connects back to the attacker
  - Use local exploits that establish a shell
  - Change firewall rules that block other attacks
  - Break out of a chroot jail (restricted execution environment), giving full access to the file system

Outline

- Introduction
- Stack overflows
- Defending against buffer overflows
  - Compile-time defenses
  - Run-time defenses
- Other forms of overflow attacks
  - Heap overflows
  - Global data area overflows
  - Integer overflows

Buffer Overflow Defenses

- Compile-time defenses: harden programs to resist attacks in new programs
  - Choose a high-level language (e.g., Java), encouraging safe coding standards, using safe standard libraries, or including additional code to detect stack frame corruption
- Run-time defenses: detect and thwart attacks in existing programs
  - Executable Address Space Protection, Address Space Layout Randomization, Guard Pages.

Compile-Time Defenses: Programming Languages

- Low-level languages such as C allow direct access to memory addresses.
  - Are vulnerable to buffer overflow
  - Large legacy code base that is unsafe
- Modern high-level languages (Java, C#…) have a strong notion of type and valid operations
  - Compiler enforces array bounds checks and permissible operations on variables
  - Incur more runtime overhead; not good for writing low-level code like device drivers

Compile-Time Defenses: Safe Coding Techniques

- C was designed with more emphasis on space efficiency and performance than on type safety
  - Programmers must take responsibility for ensuring safe use of all data structures and variables.
  - For security hardening, programmers need to inspect the code and rewrite any unsafe code

Compile-Time Defenses: Language Extensions/Safe Libraries

- The compiler can be augmented to automatically insert range checks on references to arrays and pointers. While easy for statically allocated arrays, dynamically memory allocation is harder as size information is not available at compile time
  - Requires an extension to the semantics of a pointer to include bounds information and the use of library routines to ensure these values are set correctly
    - Programs and libraries need to be recompiled
    - Likely to have problems with third-party applications
- Replace unsafe standard C library routines with safe variants
  - Example: libsafe adds bounds checks to vulnerable libc functions

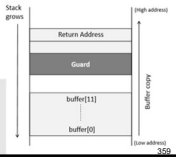
Compile-Time Defenses: Stack Guard

- Compiler stores a random secret value (`secret`) in a memory location (in the heap, not on stack), which is assigned to a local variable of the function (`guard`) which gets stored in the stack.
- Before jumping to the return address, check if the value of the local variable is changed. If yes, buffer overflow has taken place and the program exits without returning.

```
void foo (char *str)
{
   int guard;
   guard = secret;

   char buffer[12];
   strcpy (buffer, str);

   if (guard == secret)
      return;
   else
      exit(1);
}
```




```
seed@ubuntu:~$ gcc -o prog prog.c
seed@ubuntu:~$ ./prog hello
Returned Properly

seed@ubuntu:~$ ./prog hello00000000000
*** stack smashing detected ***:  ./prog terminated
```


Run-Time Defenses: Executable Address Space Protection

- Use virtual memory support to make some regions of memory non-executable
  - Requires support from memory management unit (MMU)
  - NX bit (No-eXecute) in modern CPUs
    - Put in page table; says “don’t execute code in this page”

Run-Time Defenses: Address Space Layout Randomization (ASLR)

- ASLR randomizes memory addresses of key data structures, incl. stack, heap, global data, standard library functions (libc) to make it harder for attacker to find important addresses.
  - For a 32-bit CPU, the virtual memory address range has size 2^{32}; typically the user space address range has size 2^{19}. This is not large enough against brute-force attacks that try all possible addresses.
  - For a 64-bit CPU, the virtual memory address range has size 2^{64}; This is large enough against brute-force attacks.

361

ASLR Example

- Array x[12] is statically allocated; hence it is on the stack;
- Array y[12] is allocated dynamically with malloc(), hence it is on the heap.

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   char x[12];
   char *y = malloc(sizeof(char)*12);

   printf("Address of buffer x (on stack): 0x%x\n", x);
   printf("Address of buffer y (on heap) : 0x%x\n", y);
}
```

362

When set to 0, address space is not randomized.

```
$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
$ a.out
Address of buffer x (on stack): 0xbffff370
Address of buffer y (on heap) : 0x804b008
$ a.out
Address of buffer x (on stack): 0xbffff370
Address of buffer y (on heap) : 0x804b008
```

When set to 1, only stack memory address is randomized.

```
$ sudo sysctl -w kernel.randomize_va_space=1
kernel.randomize_va_space = 1
$ a.out
Address of buffer x (on stack): 0xbf9deb10
Address of buffer y (on heap) : 0x804b008
$ a.out
Address of buffer x (on stack): 0xbf8c49d0
Address of buffer y (on heap) : 0x804b008
```

When set to 2, both stack and heap memory addresses are randomized.

```
$ sudo sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
$ a.out
Address of buffer x (on stack): 0xbf9c76f0
Address of buffer y (on heap) : 0x87e6008
$ a.out
Address of buffer x (on stack): 0xbfe69700
Address of buffer y (on heap) : 0xa020008
```

363

Run-Time Defenses: Guard Pages

- Place guard pages between critical regions of memory
  - Flagged in MMU as illegal addresses
  - Any attempted access aborts process
- Further extension places guard pages between multiple stack frames and heap buffers

364

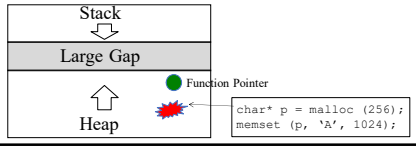
Outline

- Introduction
- Stack overflows
- Defending against buffer overflows
  - Compile-time defenses
  - Run-time defenses
- Other forms of overflow attacks
  - Heap overflows
  - Global data area overflows
  - Integer overflows

365

Heap Overflow

- Dynamic memory allocation on the heap
  - Malloc() in C, and new() in C++
- No function return address on the heap
  - But attacker may overwrite function pointers to point to shellcode
- Defenses
  - Making the heap non-executable
  - Randomizing the allocation of memory on the heap



366

Global Data Overflow

- Can attack buffer located in global data area
  - If buffer overflow occurs, data may overflow global buffer and change adjacent memory locations, including perhaps one with a function pointer.
- Defenses
  - Making the global data area non-executable,
  - Arranging function pointers to be located below any other types of data
  - Using guard pages between global data area and other parts

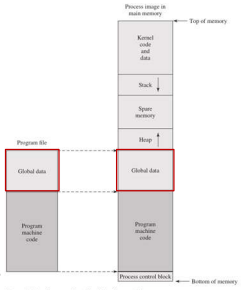


Figure 10.4 **Program Loading into Process Memory**

Integer Overflow

- Integer overflow occurs value assigned to an integer exceeds its maximum size.
- Feed a program large params to cause integer overflow on variable, then program may allocate a too-small buffer based on the variable's value, hence enabling buffer overflow attack
- This convert a dataflow attack into a control flow attack

```
int f() {!
unsigned short x = 65535;!
x++; // overflows to become 0!
printf("%d\n",x); // memory safe!
char *p = malloc(x); // size-0 buffer!!
p[1] = 'a'; // violation!
}
```


CH12 OS Security

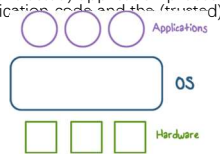
ZJU 2020

Outline

- Understand the important role an Operating System (OS) plays in computer security
- Learn about the need for hardware support for isolating OS from untrusted user/application code
- Understand key Trusted Computing Base concepts

Operating System (OS)

- An OS interacts with both applications and hardware.
- It provides easier to use and high level abstractions for resources such as address space for memory and files for disk blocks.
- Provides controlled access to hardware resources.
- Provides isolation between different application processes , and between (untrusted) application processes running untrusted/application code and the (trusted) OS.



Need for Trusting an OS

- The OS is a Trusted Computing Base (TCB). To be trusted, it must meet the following requirements:
  - Tamper-proof
    - Untrusted code in user space cannot tamper with it
  - Complete mediation
    - Every access to protected resources must go through and be mediated by it
  - Correct
    - Protected resources should be used in a correct way

TCB and Resource Protection

- TCB controls access to protected resources
  - Must establish the source of a request for a resource through authentication
  - Authorization
  - Mechanisms that allow various access control policies to be supported

373

Secure OS Quiz

- A system call allows application code to gain access to functionality implemented by the OS. A system call is often called a protected procedure call. Is the cost of a system call:
  A. the same as a user-level function call
  B. higher than a user-level function call
- ANS: B
  - A system call crosses the user-kernel boundary, hence is more expensive than a user-level function call

374

TCB Requirement 1

Tamper Proof

375

Isolating OS from Untrusted User Code

How do we meet the first requirement of a TCB (e.g., isolation or tamper-proof)?

- Hardware support for memory protection (MMU)
- Processor execution modes
  - system vs. user modes
  - some processors support more than two execution rings
- Privileged instructions which can only be executed in system mode
  - e.g., those that access hardware directly
- System calls used to transfer control between user and OS code

System Calls: Going from User to OS Code

System calls used to transfer control between user and system code

- Such calls come through **"call gates"** and return back to user code. The processor execution mode or privilege ring changes when call and return happen.
- x86 *sysenter/sysexit* instructions

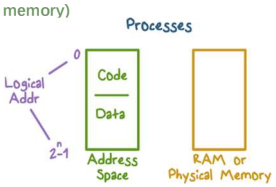
Isolating User Processes from Each Other

How do we meet the user/user isolation and separation?
OS uses **memory protection** to ensure this.

Address Space: Unit of Isolation

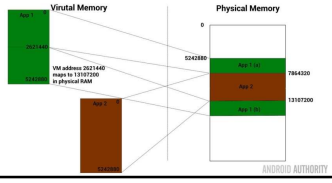
Each process views memory as a **contiguous memory address space (often larger than available physical memory)**

- Each process has its own address space, with size 2^{32} (for 32-bit address) or 2^{64} (for 64-bit address).



Process Data/Code Protection

- The OS Page Table maps logical virtual memory addresses (pages) into physical memory addresses (pages)
- Memory spaces of different processes are mapped to different parts of the physical memory
- OS will not map a virtual page of one process to a physical page of another process (unless explicit sharing is desired)
  - App1 cannot access App2's memory because it has no way to reach its memory



Hardware Support for Memory Management

- The Page Table mechanism requires hardware support, the Processor's Memory Management Unit (MMU)
- RWX bits on physical memory pages limit **type of access** to addressable memory
  - R: Read; W: Write; X: eXecute
  - e.g., Malicious code injection and execution on the stack can be prevented by making the stack memory non-executable

Revisiting Stack Overflow Quiz

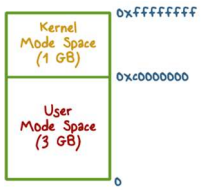
- The stack can be exploited through:
  - A. Overflowing the buffer to change the return address to alter program execution
  - B. Pushing data onto the stack to overflow the stack into the heap
  - C. Popping data off the stack to gain access to application code.
- ANS: A

OS Isolation from Application Code

- OS (Kernel) resides in a portion of each process's address space.
- True for each process, **processes can cross the fence only in controlled/limited ways.**

OS Isolation from Application Code

- 32-bit Linux: lower 3GB for user space, top 1GB for kernel space
- Corresponds to x86 privilege ring transitions (ring0 for kernel space, ring3 for user space)
- Windows and OS X similar
- DOS, and small Real-Time Operating Systems, have no such fence, **any process could alter the kernel**



Execution Privilege Level Quiz

- Which of the following functions should be executed in kernel space?
  A. Switching CPU from one process to another when a process blocks. (scheduler)
  B. Page fault handling
  C. Changing who can access a protected resource such as a file
  D. Setting up a new stack frame when an application program calls one of its functions
- ANS: A, B, C

TCB Requirement 2

Complete Mediation

Complete Mediation: The TCB

- Make sure that no protected resource (e.g., memory page or file) could be accessed without going through the TCB
- TCB acts as a reference monitor that cannot be bypassed
- Privileged instructions

Complete Mediation: User Code

- User code cannot access kernel space without issuing a system call and changing to system mode
- User code cannot access physical resources directly because they require privileged instructions (e.g. servicing interrupts) which can only be executed in system mode

Secure OS Quiz #2

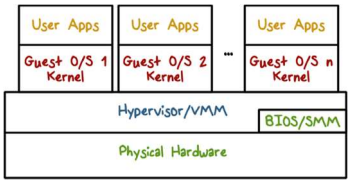
- Complete mediation ensures that the OS cannot be bypassed when accessing a protected resource. How does the OS know who is making the request for the resource?
  A. Process runs on behalf of a user who must have previously logged in,
  B. Requested resource allows us to find out who must be requesting it.
- ANS: A

Complete Mediation: OS

- To ensure complete mediation:
  - OS virtualizes physical resources and provides an API for virtualized resources, e.g., files for storing persistent data on disk
  - Virtual resource must be translated to physical resource handle (e.g., disk blocks), which can only be done by OS

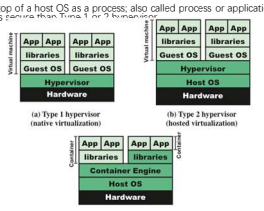
Virtualization

- OS is large and complex; compromise of an OS impacts all applications
- With virtualization: a Hypervisor (Virtual Machine Monitor) enforces isolation between Virtual Machines (VMs) that run guest OSes and applications
  - Compromise of OS in VM1 only impacts applications running on VM1
- The TCB is the Hypervisor



Types of Virtualization

- Type 1 hypervisor runs directly on top of hardware; also called bare-metal hypervisor or native virtualization.
  - XEN
- Type 2 hypervisor runs on top of a host OS; also called hosted hypervisor or hosted virtualization.
  - KVM, VMWare, VirtualBox
- Container runs on top of a host OS as a process; also called process or application virtualization. Lightweight but less secure than Type 1 or 2 hypervisor
  - Docker



(a) Type 1 hypervisor (native virtualization)

(b) Type 2 hypervisor (hosted virtualization)

(c) Container (application virtualization)

Figure 12.2 Comparison of Virtual Machines and Containers

TCB Requirement 3

Correctness

Correctness

- Compromise of TCB means an attacker has access to everything. Hence correctness of TCB is extremely important
  - Secure coding is important,
    - OS or hypervisor is typically written in languages that are not type safe (e.g., C)
  - Hypervisor is smaller and simpler than OS, hence easier to ensure correctness
    - Hypervisor is only responsible for partitioning physical resources among VMs

TCB Requirements Quiz

- What TCB requirement is violated as a result of an attack that exploits a vulnerability in an OS turns off the check that is performed before access to a protected resource is granted.
  - A. Complete mediation
  - B. Correctness
  - C. Tamper-proof
- ANS: A, C

Summary

- Understand the important role an OS plays in protecting resources and applications
- Understand how OS is isolated from untrusted code with hardware support for memory management
- Understand how complete mediation is provided.

CH16 Security Protocols

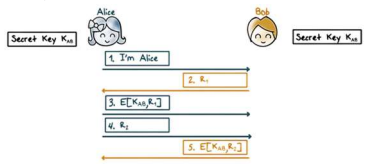
ZJU 2020

Why Security Protocols

- Alice and Bob want to communicate securely over the Internet, they need to:
  - (Mutually) authenticate
  - Establish and exchange keys
  - Agree to cryptographic operations and algorithms
- Building blocks:
  - Public-key (asymmetric) and secret-key (symmetric) algorithms, hash functions

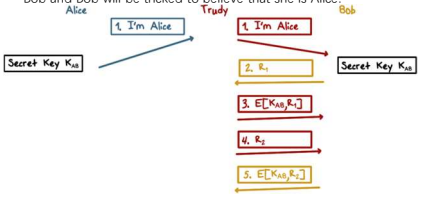
Mutual Authentication: Shared Secret

- Suppose that Alice and Bob share a secret key K_{AB}, that is, only Bob and Alice know this key. Here is an authentication protocol using symmetric cryptography:
- 1. Alice says "I'm Alice";
- 2. Bob says to Alice: prove it, by sending a random value R_1, called a nonce or a challenge;
- 3. Alice encrypts R_1 using the shared key K_{AB} and sends Bob the ciphertext $E[K_{AB}, R_1]$, called the response; when Bob receives the response, he decrypts it and see if it matches R_1 that he sent to Alice, and if it matches, he knows that the party he is communicating with must be Alice because other than himself, only Alice knows the shared key K_{AB}, and without K_{AB}, R_1 cannot be encrypted properly, hence the ciphertext he receives won't be decrypted back to R_1.
- 4. Alice can authenticate Bob by sending another nonce R_2 to Bob, followed by the same process.
- Sometimes only one-way authentication is required (Steps 1-3), for example, Alice is a client and needs to authenticate to Bob, which is a server. In this case, if Alice is a user with an account on the server, K_{AB} can be derived from her password hash, which is known to Bob.



Record-and-Replay Attacks

- It is important for challenges R_1, R_2 to be not easily repeatable or predictable. Otherwise, an eavesdropper Trudy can record the challenge and response between Alice and Bob and replay them to impersonate Alice or Bob.
  - e.g., Trudy tries to impersonate Alice and Bob happens to send R_1 as the challenge, which was used previously and recorded by Trudy, then Trudy can just send the recorded response ciphertext $E[K_{AB}, R_1]$ to Bob and Bob will be tricked to believe that she is Alice.



Record-and-Replay Attacks cont'd

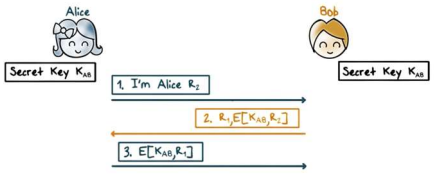
- If the challenges always increase in value, then the above attack no longer works.
- But Trudy can first impersonate Bob and send a larger challenge $R_1 = 1000$ and record the response from Alice. Meanwhile, the real Bob is using a smaller $R_1 = 950$. Then Trudy can impersonate Alice sometime in the future when the real Bob finally sends $R_1 = 1000$.
  - In practice, R_1, R_2 should be large random values so that they are not predictable and the chance of repeats is very small.
- Another attack is when Trudy steals the shared secret key K_{AB} from either Alice or Bob, then she can impersonate either Alice or Bob.

Mutual Authentication Quiz

- Which ones are true?
  - A. The challenge values used in an authentication protocol can be repeatedly used in multiple sessions
  - B. The authentication messages can be captured and replayed by an adversary
  - C. Authentication can be one-way, e.g., only authenticating Alice to Bob
- ANS: B, C.

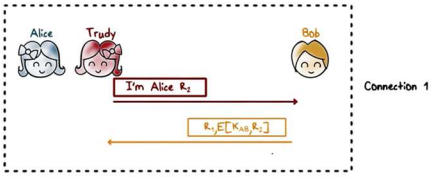
Mutual Authentication: Simplified

- 1, Alice says Hi to Bob and sends along a challenge R_2;
- 2. Bob sends Alice the ciphertext $E[K_{AB}, R_2]$ and his own challenge R_1; Upon receiving this response Alice decrypts the ciphertext and sees if it matches the plaintext R_2 that she had sent to Bob, if it matches, then she knows that she is communicating with Bob;
- 3. Alice then sends Bob a ciphertext of R_1; and Bob decrypts it and matches with the plaintext R_1 to authenticate Alice.



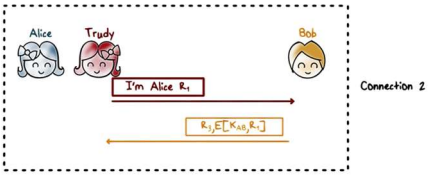
Reflection Attack

- This protocol is subject to a reflection attack, a man-in-the-middle scheme.
- Trudy tries to impersonate Alice. By following the protocol, Trudy will be stuck at Step 3 because she cannot respond to challenge R_1 sent from Bob, since she does not know the shared secret key K_{AB}.



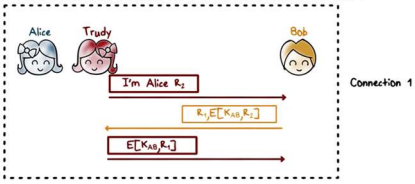
Reflection Attack

- Then Trudy opens another connection with Bob, again impersonating Alice. This time, Trudy sends Bob the challenge R_1 that Bob had sent her in the first connection, that one that she is stuck in. According to the protocol, Bob responds with the ciphertext of R_1, and another challenge R_3.

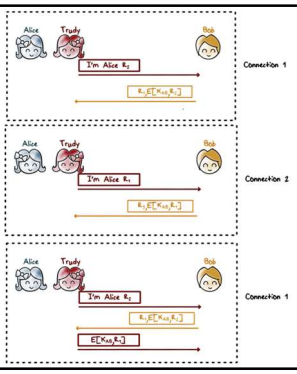


Reflection Attack

- Trudy can then take the ciphertext $E[K_{AB}, R_1]$ to complete the step 3 of the first connection, by just sending the ciphertext of R_1 back to Bob. At this point, the first connection successfully concludes and Trudy has successfully impersonated Alice.
- This is called a reflection attack because Trudy sends back to Bob what Bob had just sent her from another conn

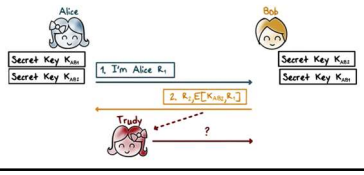


Reflection Attack



Preventing Reflection Attack

- One way is to let Alice and Bob share two different shared keys, one key K_{AB1} for the initiator of the connection, Alice, and the other key K_{AB2} for the responder, Bob. Then Trudy cannot get Bob to encrypt using Alice's key.
- When Trudy gets the ciphertext $E[K_{AB}, R_1]$ from Bob, she can not just send it back to Bob because Bob is expecting a ciphertext produced using Alice's key $E[K_{AB1}, R_1]$.

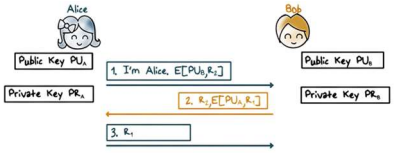


Mutual Authentication: Reflection Attack

- Another way is the use different challenges for the initiator and responder, e.g., even number challenge for Alice and odd number for Bob.
- Trudy cannot send the challenge R_1 she received from Bob, which is an odd number, as a new challenge for Bob because Bob is expecting an even number from Alice.

Mutual Authentication w Public Keys

- 1. Alice sends Bob a challenge ciphertext $E[PU_B, R_2]$, as plaintext R_2 encrypted using Bob's public key PU_B.
- 2. Bob decrypts $E[PU_B, R_2]$ using his private key PR_B, and send back R_2, along with his own challenge ciphertext $E[PU_A, R_1]$, as plain R_1 encrypted using Alice's public key PU_A.
- 3. The plaintext R_2 in Bob's response to Alice tells her that she is communicating with Bob because only Bob has his private key PR_B that is paired the public key PU_B that encrypted R_2; Alice decrypts $E[PU_A, R_1]$ using her private key PR_A and sends the plaintext R_1 to Bob; Bob will then know he is communicating with Alice because only Alice has her private key PR_A that is paired with her public key PU_A used to encrypt R_1.
- Another variant to use signing with private keys instead of encryption with public keys.



Reflection Attack Quiz

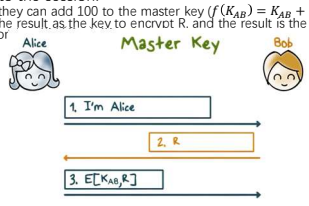
- A reflection attack is a form of man-in-the-middle attack
  - A. To defeat a reflection attack, we can use an odd number as challenge from the initiator and even number from the responder.
  - B. We can use signing with public keys to achieve mutual authentication.
- ANS: A, B

Session Keys

- Typically, Alice and Bob share a long-term secret key called the master key. For example, the master key is derived from a password. For each session, Alice and Bob use the master secret key to authenticate and establish a temporary session key to protect message security during the session.
- The main benefit of using a session key is that if the key is leaked or broken, the impact is limited to the current session. Intuitively, the more a secret is used, the higher the chance that the secret can be leaked. Therefore, we should limit the use of the long-term, master secret, and only use it at the beginning of a session for authentication and establishing the session key.

Session Key Example

- Steps 1-3 are for Alice to authenticate to Bob, say, Bob is a server. (It is a one-way authentication)
- 4. Both Alice and Bob compute the same session key that is based on the shared master key and something about the current session, so that the session key is both a secret and unique to the session.
  - e.g., they can add 100 to the master key ($f(K_{AB}) = K_{AB} + 100$), and use the result as the key to encrypt R, and the result is the shared session



Exchanging Session Key w. Public Keys

- Alice sends Bob a session key K, encrypted using Bob's public key PU_B, then signed using Alice's private key PR_A. Bob can decrypt and recover key K with PU_A and PR_B.
  - Alice → Bob: $E(PR_A, E(PU_B, K))$
- Or, they can use their private keys to sign the public messages in a Diffie-Hellman key exchange, and this can prevent the man-in-the-middle attack. Diffie-Hellman with signing:
  - Alice → Bob: $E(PR_A, A)$
  - Bob → Alice: $E(PR_B, B)$
  - Contrast this with plain Diffie-Hellman (next page), where A and B are sent as plaintext.

Recall: Diffie-Hellman

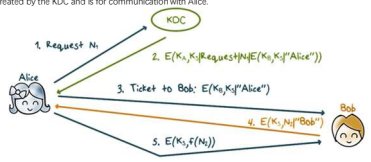
- Alice and Bob agree on:
  - A sufficiently large prime number p
  - A base number $g < p$
- Alice chooses a secret number a and sends to Bob
  - $A = g^a \bmod p$
- Bob chooses a secret number b and sends to Alice
  - $B = g^b \bmod p$
- Alice computes $S_1 = B^a \bmod p$
- Bob also computes $S_2 = A^b \bmod p$
- We can easily show that $S_1 = S_2$, and this is their shared secret key
  - $S_1 = B^a \bmod p = g^{ab} \bmod p = A^b \bmod p = S_2$

Key Distribution Center (KDC)

- Pairwise key-exchange based on shared secret master key does not scale to large number of parties, since each communication pair needs to share a master key
  - Alice needs to share a master key with Bob, another master key with Carol, and so on.
- This motivates KDC.

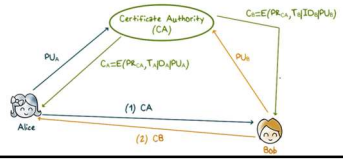
Key Distribution Center (KDC)

- KDC has many master keys, one for each user; each user only keeps one master key that is shared with the KDC. Suppose Alice and Bob need to establish a session key K_s. They each have a **master key** (K_a,K_b) shared with KDC.
- 1. Alice sends a request "need a key to talk to Bob", to KDC along with a nonce N_1.
- 2. KDC sends a message encrypted using the master key K_a shared between Alice and the KDC. This message contains the session key K_s that the KDC just created for Alice and Bob to share. The same request, nonce N_1, and a ticket $E(K_b, K_s||"Alice")$. It is encrypted using the master K_b shared between Bob and the KDC, and it contains the session key K_s and the ID of Alice.
- 3. Alice decrypts the message from KDC with master key K_a to get the session key K_s. She knows that the message is from the KDC and is fresh (not a replay), because only the KDC can use K_a to encrypt properly a message that contains that request and nonce that she just sent.
- 4. Alice then sends the ticket $E(K_b, K_s||"Alice")$ to Bob. Only Bob can decrypt the ticket because it is encrypted using K_b, the master key shared between Bob and the KDC. When Bob decrypts the ticket, he knows that ticket is created by the KDC because only the KDC can encrypt the ID of Alice properly, and he knows that session key K_s was created by the KDC and is for communication with Alice.



Public Key Certificates

- An imposter Eve may post her public key, and claim that it is someone else's public key. To prevent this, the distribution of public keys is done through a Certificate Authority (CA).
- Alice sends her public key PU_A to the CA. The CA verifies Alice's identification, and creates a certificate of Alice's public key $C = E(PR_{CA}, T_A||ID_A||PU_A)$ and sends it to Alice. The certificate contains time of creation and validity period T_A, and ID of Alice ID_A, and her public key PU_A, and is signed by the CA's private key PR_{CA}. Alice can then sends this certificate to any user, or publish it so that any user can get it. It is assumed all users have the CA's public key PU_{CA}, and therefore, any user, say, Bob, can use the CA's public key to verify the certificate and obtain Alice's public key PU_A.
- YouTube: How does HTTPS work? What's a CA? What's a self-signed Certificate?
  - https://www.youtube.com/watch?v=T4DfS_cojAs



Session Key Quiz

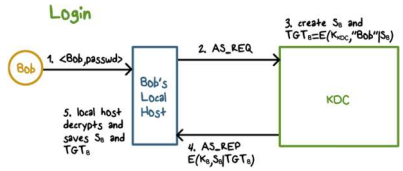
- Which items are true?
  - A. A session key should be a secret and unique to the session.
  - B. Authentication should be accomplished before key exchange.
  - C. A key benefit of using of KDC is for scalability.
  - D. In order for Bob to verify Alice's public key, the Certificate Authority must be on-line.
  - E. In order for Bob and Alice to establish a session key, the KDC must be on-line.
  - F. Signing the message exchanges in Diffie-Hellman eliminates the man-in-the-middle attack.
- ANS: A, B, C, E, F.

Kerberos

- A standard protocol for authentication and key distribution in a network environment
- Every principal (a human user or a network resource such as workstation and printer) has a master (secret) key shared with the Kerberos server (referred to as KDC)
  - A user's master key is derived from password
  - Other resources must have their keys configured in by the device manufacturer
- All master keys are stored at the KDC, protected/encrypted

Kerberos

- 1-2. When Bob logs in, his workstation (local host) contacts the KDC with authentication service request (AS_REQ).
- 3. The KDC generates a per-day session key S_B and a Ticket-Granting-Ticket $TGT_B = E(K_{KDC}, "Bob"|S_B)$ that contains Bob's ID and S_B, encrypted using the KDC's own key K_{KDC}.
- 4. The KDC then sends a message to Bob that includes S_B and TGT_B, encrypted using K_B, the master key shared between Bob and KDC.
- 5. Bob's local host decrypts this message, and stores S_B and TGT_B, used for subsequent messages with KDC.
- 6.Bob must include TGT_B in subsequent requests to KDC, so KDC can decrypt it with K_{KDC} to get Bob's ID and S_B. Bob must decrypt subsequent replies from KDC with S_B.

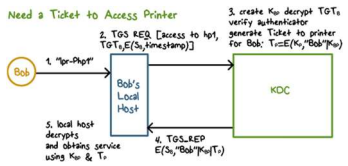


Kerberos Benefits

- Localhost does not need to store Bob's password once a user (Bob) has logged in and it has obtained S_B from the KDC
- The master key K_B that the user shares with the KDC is only used once every day. This limits exposure of the master key, which is derived from Bob's password and is subject to password-guessing attacks

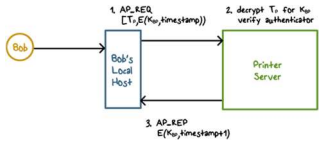
Getting a Ticket to Access the Printer

- 1-2. Bob's local host sends a Ticket Granting Service Request (TGS_REQ) to the KDC. The request contains "access to hp1", $TGT_B = E(K_{KDC}, "Bob"|S_B)$, and an authenticator $E(S_B, timestamp)$, the current timestamp encrypted using Bob's per-day key S_B.
- 3. KDC decrypts the TGT_B, and it knows it is valid because only it had the key K_{KDC} to properly encrypt Bob's ID contained in TGT_B. It extracts S_B from TGT_B and uses it to verify the authenticator $E(S_B, timestamp)$ by decrypting it and checking the timestamp is current. This proves that the sender is Bob because only Bob has the key S_B for encrypting the current timestamp. The KDC then generates a ticket $T_P = E(K_P, "Bob"|K_{BP})$ for Bob to communicate with the printer, which contains Bob's ID and a new session key K_{BP}, encrypted using the printer's master key K_P. (A network device such as a printer has a long and random master key K_P that is hard to guess. Therefore, the ticket for these devices is encrypted using their master keys.)
- 4. KDC sends the response TGS_REP $E(S_B, Bob|K_{BP}|T_P)$ to Bob's local host. It contains Bob's ID, the session key K_{BP}, and ticket to the printer T_P, encrypted using Bob's per-day key S_B.
- 5. Bob's local host decrypts it and verifies that it is from the KDC because only the KDC can encrypt Bob's ID correctly with the key S_B. It now has the session key K_{BP}, and ticket to the printer T_P.



Accessing the Printer

- Bob's local host authenticates itself to the printer.
- 1. It sends an authentication request AP_REQ containing the ticket $T_P = E(K_P, "Bob"|K_{BP})$ and the authenticator $E(K_{BP}, timestamp)$, the current timestamp encrypted using K_{BP}, the new session key shared between Bob's local host and the printer.
- 2. The printer decrypts the ticket using its master key K_P shared with the KDC. The printer now has the session key K_{BP}, shared with Bob's Local Host. The printer then uses K_{BP} to verify the authenticator $E(K_{BP}, timestamp)$,by decrypting it and checking the timestamp is current.
- 3. The printer sends a response AP_REP containing the authenticator $E(K_{BP}, timestamp + 1)$ to authenticate itself.
- 4. Bob's Local Host uses K_{BP} to verify the authenticator by decrypting the server text and verifying that the result matches with the previous timestamp+1. It can now send print jobs to the printer.



Kerberos Quiz

- Which ones are true?
  - A. Kerberos provides authentication and access control
  - B. Kerberos distributes session keys
  - C. To avoid over-exposure of a user's master key, Kerberos uses a per-day key and a ticket-granting-ticket
  - D. The authenticators used in requests to KDC and application servers can be omitted
  - E. Access to any network resource requires a ticket issued by the KDC
- ANS: A,B,C,E

Summary

- One-way and mutual authentication protocols using secret key or public keys. It is important that the challenges are randomly generated to prevent record-and-replay attacks.
- Protocols to establish session key, which can use pre-shared key or public keys and the random challenge used in authentication. Key Distribution Center (KDC) is often used to establishing and distributing session keys, and certificate authority is used to generate and sign public key certificate that can be distributed and verified.
- Kerberos is used for authenticating users and devices on a network and enforce access control. Each access requires a ticket, and a Ticket-Granting Ticket (TGT) after user login is used to request such tickets to access devices.

CH22 IPSec and TLS

ZJU 2020

Outline

- IPSec
  - Tunnel mode, transport mode
  - ESP and AH for confidentiality and authenticity
- Transport Layer Security (TLS)
  - Record protocol
  - Handshake protocol

IP Spoofing

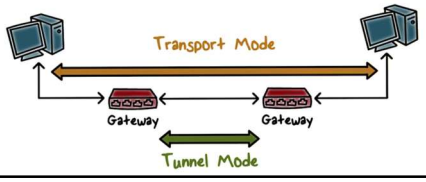
- IP spoofing is a common technique in cyber attacks, e.g., DNS Amplification DoS Attack:
  - Bots spoof IP address of a target system
  - Then send DNS queries to DNS servers
  - The DNS servers respond, sending large amounts of data to the target system

Goals of IPSec

- Verify sources of IP packets
  - Provide Authentication that is lacking in IPv4
- Protect integrity and/or confidentiality of packets
- Prevent replaying of old packets
- Provide security for upper layer protocols and applications

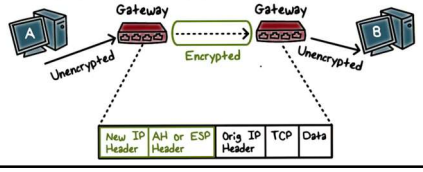
Two Modes of IPSec

- IPSec has 2 modes:
  - Transport mode: only the payload of the IP packet is encrypted or authenticated, not the header. Security protection is provided to traffic from one end host to another, i.e., end-to-end.
  - Tunnel mode (for VPN): the entire IP packet (header+payload) is encrypted and authenticated. It is then encapsulated into a new IP packet with a new IP header. Security protection is provided to traffic from gateway of a network to the gateway of another network. (Gateway is often also the firewall)



Tunnel Mode

- Suppose we have two end hosts, A and B, belonging to the same company but in two different Local Area Networks (LAN) over the Internet (e.g., two campuses of a university).
- Before A's packets leave its LAN, the gateway encrypts them and sends encrypted packets to the gateway to B's LAN, which then decrypts the packets. An encrypted packet contains:
  - New IP header that specifies the gateway as the source IP and B's gateway as its destination IP.
  - IPSec header that contains information about the protection provided using AH or ESP
  - The original packet as the data or payload.

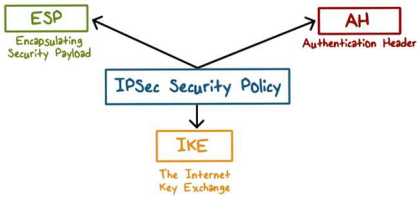


IPSec Quiz

- IPsec can assure that
  - A. a router advertisement comes from an authorized router
  - B. a routing update is not forged
  - C. a redirect message comes from the router to which the initial packet was sent
  - D. all of the above
- ANS: D

IPSec Architecture

- IPSec includes
  - Internet Key Exchange (IKE) protocol for negotiating protection parameters, including cryptographic algorithms and keys
  - Two types of protections: ESP and AH.



Encapsulated Security Payload (ESP)

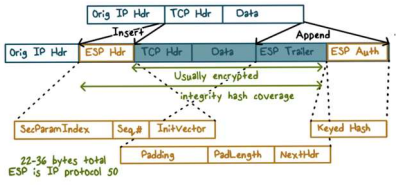
- Encrypt and authenticate each packet
- **Encryption is applied to packet payload**
- Authentication is applied to data in the **IPSec header as well as the data contained as payload**, after encryption is applied

ESP Modes Quiz

- ESP can be securely used in...
  - A. encryption only mode
  - B. authentication only mode
  - C. encryption and authentication mode
- ANS: A,B,C (but only C is fully secure.)

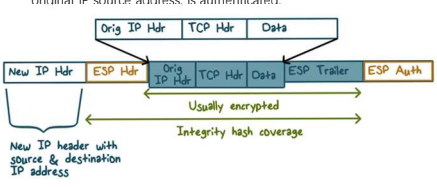
ESP in Transport Mode

- ESP header is inserted after the original IP header. It includes the security parameter index, a sequence number, the InitVector (IV) for encryption.
- ESP trailer had the padding information and pointer to next header, such as the TCP or UDP header.
- Packet payload and the ESP trailer are both encrypted but the ESP header is not because it provides information (e.g., the security parameter index) that tells the receiving end how to decrypt the payload, for example, which algorithm and shared secret key to use.
- The ESP header and the encrypted payload are then hashed together with a secret key, and the hash value is stored in ESP Auth as the Message Authentication Code (MAC) for the receiver to ve



ESP Tunnel Mode

- ESP header is added after the new IP header.
- Packet payload, which contains the entire original packet plus the ESP trailer, is encrypted.
  - The original IP header data, including the original source and destination IP addresses are encrypted.
  - MAC is computed over the entire original packet plus the ESP header and trailer, therefore, header of the original IP, for example, the original IP source address, is authenticated.

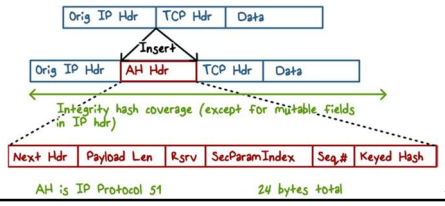


Authentication Header (AH)

- In ESP, the IP header is not authenticated. So what if we want to authenticate the entire packet? We can use Authentication Header (AH).
- Authentication is applied to the entire packet
  - Certain fields in the IP header, e.g., time-to-live, or, TTL, that may change in transmission are not included, or, zeroed out, when MAC is computed.
- If both ESP and AH are applied to a packet, AH follows ESP
  - Use ESP to encrypt the payload and then apply AH to authenticate the entire packet.

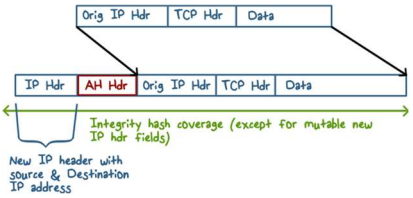
Authentication Header in Transport Mode

- If AH is used with transport mode, the AH header is inserted after the original IP header, and contains the authentication code.



Authentication Header in Tunnel Mode

- If AH is used with tunnel mode, the AH header is inserted after the new IP header.



ESP and AH Quiz

- Which are true:
  - A. ESP can provide both confidentiality and integrity protection
  - B. If the authentication option of ESP is chosen, message integrity code is computed before encryption
  - C. To protect the confidentiality and integrity of the whole original IP packet, we can use ESP with authentication option in tunnel mode.
  - D. In AH, the integrity hash covers the IP header
- ANS: A,C,D
  - B is false. If the authentication option of ESP is chosen, message integrity code is computed after encryption.

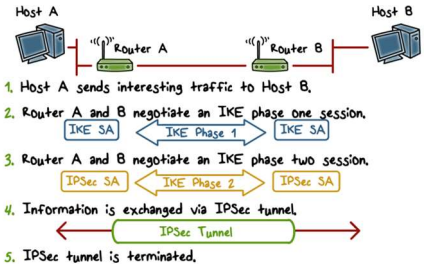
Internet Key Exchange (IKE)

- IKE is the security protocol for two end-hosts or two gateways using IPSec for secure connections. It allows the two parties to:
  - Exchange and negotiate security policies.
  - Establish security parameters, or security associations
    - e.g., ESP or AH, and which cryptographic algorithms for encryption or hashing.
  - Key exchange
    - Establish shared keys between two parties.

Security Association (SA)

- An SA describes security parameters for a type of traffic, e.g., all http connections from host A to B.
- One-way relationship between a sender and a receiver, defined by IPSec parameters
  - One SA for inbound traffic, another SA for outbound
- Security Association Database (SADB)
- Security Parameter Index (SPI)
  - A unique index for each entry in the SADB
  - Receiver identifies the SA associated with a packet by table lookup, in order to unprocess (decrypt) it
- Security Policy Database (SPD)
  - Stores policies used to establish SAs

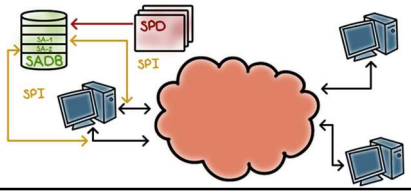
IPSec Summary



- (Details of IKE is omitted.)

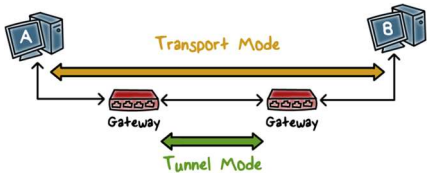
SPD and SADB Fit Together

- An SPD entry describes the security policy, which decides the security parameters stored in an SA in SADB.
- Each SA has a unique index SPI, which is included in the IPSec packet header.



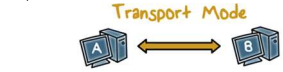
SPD and SADB Example:

- Computers A and B wish to communicate with each other.



SPD and SADB for Transport mode

- A's SPD entry specifies:
  - Traffic from A to B for any (protocol, port) combination needs to be authenticated with AH, using HMAC with MD5 as the embedded hash function.
  - The negotiated parameters are stored in an SA in both A's and B's SADB.
- A's SADB stores:
  - the secret key for HMAC
  - SPI for looking for the SA in B's SADB. When A sends out traffic to B, it includes this SPI in the IPSec header so that B can use it to look up the SA and unprocess the traffic.



A's SPD

From	To	Protocol	Port	Policy
A	B	Any	Any	AH[HMAC-MD5]

A's SADB

From	To	Protocol	SPI	SA Record
A	B	AH	12	HMAC-MD5 key

SPD and SADB for Tunnel Mode:

- SPD entry of gateway C for A's subnet specifies:
  - Traffic from A's subnet to B's subnet has tunnel destination D, gateway of B's subnet, and it should use ESP with 3DES.
- C's SADB stores:
  - The SA that has the 3DES key, and the SPI for looking up the SA in D's SADB.

From	To	Protocol	Port	Policy	Tunnel Dest
A_{sub}	B_{sub}	Any	Any	ESP[3DES]	D

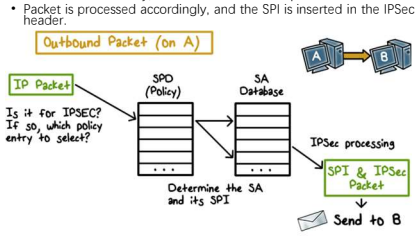
C's SPD

From	To	Protocol	SPI	SA Record
A_{sub}	B_{sub}	ESP	14	3DES key

C's SADB

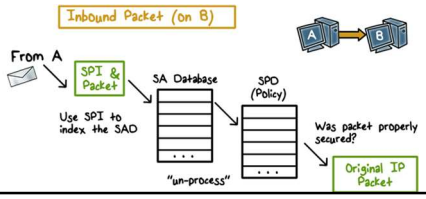
Outbound Processing

- When A sends an outbound packet:
  - First the SPD is looked up to see if traffic, e.g., http traffic from A to B, needs to be protected with IPSec.
  - If there is a SPD entry, then the SA is looked up in the SADB
  - Packet is processed accordingly, and the SPI is inserted in the IPSec header.



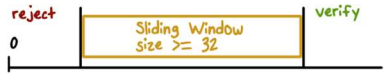
Inbound Processing

- When B receives an inbound packet:
  - The SPI in the IPSec header is used to look up the SA in the SADB, and the packet is unprocessed accordingly.
  - Then the SPD is looked up to make sure that the packet had the proper security measures according to the policy.
  - If yet, the packet is delivered to the upper layer.



Anti-Replay w. Sliding Window

- To prevent replay, the IPSec header has an IPSec sequence number, used only if authentication (AH, or ESP's authentication option) is used.
- Although packets may arrive out of order, their sequence numbers should be within a sliding window of size ≥ 32.



452

Anti-Replay w. Sliding Window

- When a packet arrives with sequence number N and has been authenticated to be valid:
- if N is smaller than the smallest number of the window, it is rejected.
- If N is greater than the largest sequence number of the sliding window, the packet is accepted and the window advances to just cover N.
- If N falls in the window, check to see if N has been received before:
  - If yes, it is rejected since it is a duplicate.
  - If not, it is accepted and N is marked as having been received.

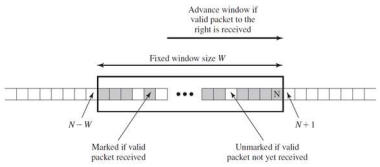


Figure 22.9 **Antireplay Mechanism**

453

IPSec Quiz

- Which are true:
  - A. Security Association, SA, specifies a two-way security arrangements between the sender and receiver.
  - B. SPI is used to help receiver identify the SA to un-process the IPSec packet.
  - C. If the sequence number in the IPSec header is greater than the largest number of the current anti-replay window the packet is rejected.
  - D. If the sequence number in the IPSec header is smaller than the smallest number of the current anti-replay window the packet is rejected.
- ANS: B, D
  - A. SA is "One-way relationship between a sender and a receiver, defined by IPSec parameters. One SA for inbound traffic, another SA for outbound."
  - C. "the packet is accepted and the window advances to just cover N."

Outline

- IPSec
  - Tunnel mode, transport mode
  - ESP and AH for confidentiality and authenticity
- Transport Layer Security (TLS)
  - Record protocol
  - Handshake protocol

455

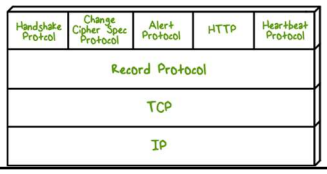
SSL and TLS

- Secure Socket Layer (SSL) is a general-purpose service implemented as a set of protocols that rely on TCP.
- Transport Layer Security (TLS) is an updated, more secure, version of SSL, and an Internet standard.
- Two implementation choices.
  - TLS could be provided as part of the underlying protocol suite and therefore be transparent to applications.
  - Alternatively, TLS can be embedded in specific packages.

456

TLS Layers

- TLS has two layers
  - Record Protocol provides basic security services to higher-layer protocols, e.g. HTTPS.
  - Three higher-layer protocols used in the management of TLS exchanges.
    - Handshake Protocol, Change Cipher Spec Protocol, Alert Protocol.



TLS Concepts

TLS Session

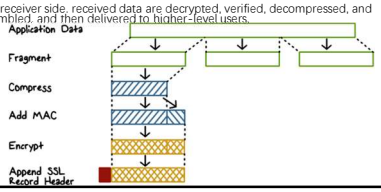
- An association between a client and a server
- Created by the Handshake Protocol
- Defines a set of cryptographic security parameters
- Used to avoid the expensive negotiation of new security parameters for each connection

TLS Connection

- A transport-layer connection that provides a suitable type of service
- Peer-to-peer relationships
- Transient
- Every connection is associated with one session

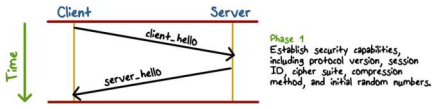
SSL Record Protocol

- The content types that have been defined are change_cipher_spec, alert, handshake, and application_data.
- Record Protocol steps:
  - 1. Fragmentation, where each upper-layer message is fragmented into blocks.
  - 2. Compression (optional).
  - 3. Compute a MAC over the compressed data.
  - 4. Encrypt the compressed message plus MAC using symmetric encryption.
  - 5. Prepend an SSL record header, which includes version and length fields.
  - 6. Transmit the resulting unit in a TCP segment.
- At the receiver side, received data are decrypted, verified, decompressed, and reassembled, and then delivered to higher-level users.



Handshake Protocol Phase 1

- Handshake Protocol provides two services for SSL connections:
  - Confidentiality: define a shared secret key that is used for symmetric encryption of SSL payloads.
  - Integrity: define a shared secret key that is used to form MAC.
- It consists of a series of messages exchanged between client and server, with four phases.
- Phase 1 is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message with the following parameters:

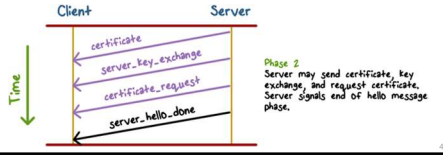


Handshake Protocol Parameters

- **Version:** the highest TLS version understood by the client
- **Random:** a 32-bit timestamp and 28 bytes generated by a secure random number generator
- **Session ID:** a variable-length session identifier
- **CipherSuite:** a list containing the combinations of cryptographic algorithms supported by the client
- **Compression Method:** a list of compression methods supported by the client

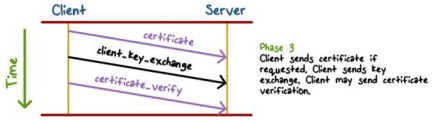
The Handshake Protocol Phase 2

- The details of phase 2 depend on the underlying public-key encryption scheme. In some cases, the server passes a certificate to the client, possibly additional key information, and a request for a certificate from the client.
- The final message in phase 2 is the server_done message, sent by the server to indicate the end of the server hello and associated messages. After sending this message, the server will wait for a client response.



The Handshake Protocol Phase 3

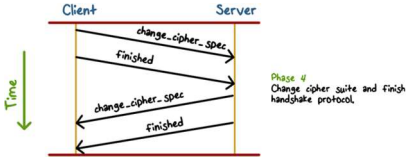
- In phase 3, upon receipt of the server_done message, the client should verify that the server provided a valid certificate (e.g., ZJU's certificate), and check that the server_hello parameters are acceptable.
- If all is satisfactory, the client sends one or more messages back to the server, depending on the underlying public-key scheme.



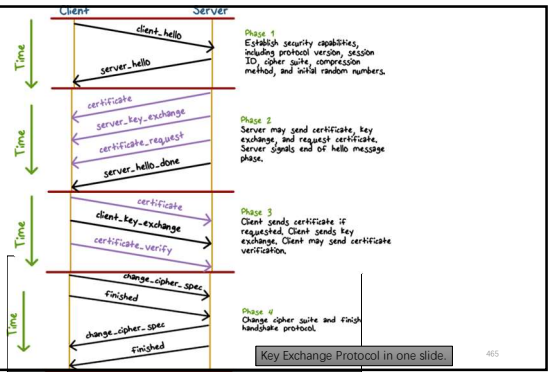
463

The Handshake Protocol Phase 4

- Phase 4 completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending CipherSpec into the current CipherSpec. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.
- In response to these two messages, the server sends its own change_cipher_spec message, transfers the pending to the current CipherSpec, and sends its finished message.
- At this point, the handshake is complete. The client and server now agree on the security parameters, so they may begin to exchange application data.



464



Key Exchange Protocol in one slide.

465

SSL & TLS Quiz

- A. Most browsers come equipped with SSL or TLS and most Web servers have implemented the protocol.
- B. Since TLS is for the transport layer, it relies on IPsec which is for the IP layer.
- C. In most applications of TLS or SSL, public keys are used for authentication and key exchange.
- ANS: A, C.
  - B is false. TLS does not rely on IPSec.

466

Summary

- IPSec can operate in tunnel or transport mode
- Confidentiality and authenticity protection provided through ESP and AH
- The one-way security association stores security parameters.
- SSL/TLS has two layers:
  - record protocol
  - handshake, change cipher spec and alert protocols

467

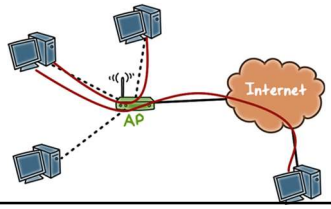
CH24 Wireless and Mobile Security

Outline

- WiFi security
- iOS security
- Android security

Introduction to WiFi

- WiFi enables devices to access the Internet through the Access Point (AP)..
- Devices in a locale can also connect to each other wirelessly through the AP.



Introduction to WiFi

- No inherent physical protection since data is transmitted in the air, an open medium.
- Broadcast communication between one sender and multiple receivers.

WiFi Quiz

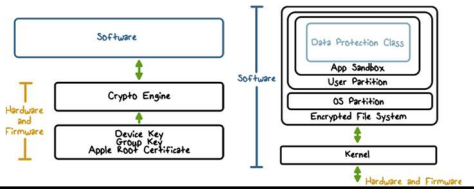
- Early solution was based on Wired Equivalent Privacy (WEP)
  - Too weak, and no longer used
- New security standard for WiFi is 802.11i, implemented as WiFi Protected Access II (WPA2)

Overview of 802.11i

- Access control model is based on 802.1X, with Extensible Authentication Protocol (EAP), a carrier protocol designed to transport the messages of different authentication protocols (e.g., TLS)
- Mutual authentication process (server-to-client and client-to-server) results in a shared session key (which prevents session hijacking)
- Different functions (encryption, integrity) use different keys derived from the session key using a one-way function
- Different keys for encryption vs. integrity protection
- Uses AES, a strong encryption method

iOS Security Architecture

- It combines SW and HW features to provide security.
- Second, it has built-in crypto capabilities to support data protection such as confidentiality and integrity.
- Third, it provides strong isolation mechanisms, e.g., app sandbox that enables apps to run securely and without compromising platform integrity.



Hardware Security Feature

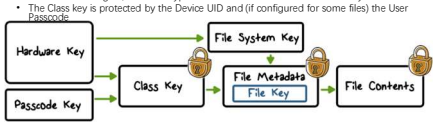
- Each iOS device has a dedicated AES-256 HW crypto engine
- Manufacturer keys: Apple provides the Device ID (UID) and the device group ID (GID) as AES 256-Bit keys
  - UID is unique to each device
  - GID represents a processor class (e.g., Apple A5 processor), and is used for non security-critical tasks such as when delivering system software during installation and restore.
- The UID and GID keys are burned into the HW and can only be accessed by the Crypto Engine

iOS Trusted Bootchain

- The security of a device is established when the device is turned on initially, by a chain of trust where each step ensures that the next is signed by Apple.
  - When an iOS device is turned on, its application processor immediately executes code from the Boot ROM. The code is immutable and serves as hardware root-of-trust. It is laid down during chip fabrication, and is implicitly trusted. It contains the Apple Root CA public key, which is used to verify that the Low-Level Bootloader (LLB) is signed by Apple before allowing it to load.
  - When the LLB finishes its tasks, it verifies and runs the next-stage bootloader, iBoot, which in turn verifies and runs the iOS kernel.
- This secure boot chain helps ensure that the lowest levels of software are not tampered with and allows iOS to run only on validated Apple devices.

Data Protection

- Data Protection protects data stored in flash memory, by constructing and managing a hierarchy of keys, and builds on the hardware encryption technologies built into each iOS device. It is controlled on a per-file basis by assigning each file to a class; accessibility is determined by whether the class keys have been unlocked.
- When a file is created:
  - It is encrypted with a unique Per-File Key (called File Key)
  - The file key is encrypted with a Class Key and stored in the file's metadata
  - The metadata of all files is encrypted with the File System Key, which is created when iOS is first installed or when the device is wiped by a user.
- When a file is opened
  - Its metadata is decrypted with the file system key, revealing the encrypted per-file key and which class protects it. The file key is decrypted with the class key, then supplied to the hardware AES engine, which decrypts the file as it is read from flash memory.
  - The Class key is protected by the Device UID and (if configured for some files) the User Passcode



iOS Security Quiz

- Which are true?
  - A. All cryptographic keys are stored in flash memory
  - B. Trusted boot can verify the kernel before it is run
  - C. All files of an app are encrypted using the same key
- ANS: B
  - A is false, since UID and GID are burned into the HW, not in flash memory
  - C is false, since each file is encrypted with a unique File Key

Mandatory Code Signing

- To ensure that all apps come from a known and approved source and have not been tampered with, iOS requires that all executable code be signed using an Apple-issued certificate.
  - Apps provided with the device, like Mail and Safari, are signed by Apple.
  - Third-party apps must also be validated and signed using an Apple-issued certificate.
- Mandatory code signing extends the concept of chain of trust from the OS to apps, and prevents third-party apps from loading unsigned code resources or using self-modifying code.

Code Signing Check

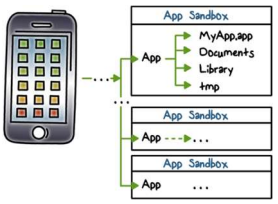
- At runtime, check all executable memory pages as they are loaded for code signatures to ensure that an app has not been modified since it was installed or last updated.
  - Enforced by kernel, handled by a user-space daemon

Restricted App Distribution Model

- iOS devices are only allowed to download apps through the AppStore.
- iOS developers must register with Apple and join the iOS Developer Program. The real-world identity of each developer, whether an individual or a business, is verified by Apple before their certificate is issued. This certificate enables developers to sign apps and submit them to the App Store for distribution. As a result, all apps in the AppStore have been submitted by an identifiable person or organization, serving as a deterrent to the creation of malicious apps. They have also been reviewed by Apple to ensure they operate as described and don't contain obvious bugs or other problems.

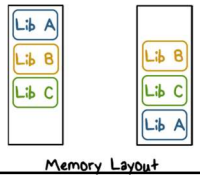
Sandboxing

- All third-party apps are "sandboxed". Each app has a unique home directory for its files, and it cannot access files stored by other apps or from making changes to the device. This prevents an app from gathering or modifying information stored by other apps.



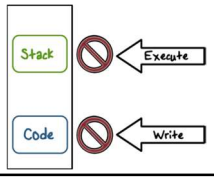
Address Space Layout Randomization (ASLR)

- ASLR protects against the exploitation of memory corruption bugs by randomly arranging the memory addresses of stack, heap, main executable, and dynamic libraries when the application is launched. This reduces the likelihood of many sophisticated memory exploits.



Data Execution Prevention (DEP)

- DEP prevents code-injection attacks, e.g., buffer-overflow is used to write malicious code on stack or in memory and force execution to jump to the malicious code.
  - Data memory pages (stack, heap) are writable but non-executable,
  - Instruction (code) memory pages are executable but non-writable
  - Needs hardware support, e.g., ARM's Execute Never (XN) feature.
- Memory pages marked as both writable and executable can be used only by apps under tightly controlled conditions checked by the kernel for the presence of the Apple-only dynamic code-signing entitlement.



Passcodes and Touch ID

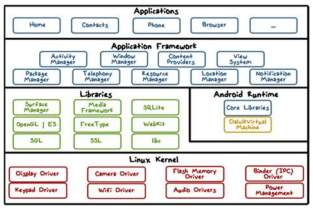
- By setting up a device passcode, the user automatically enables Data Protection.
  - Maximum failed attempts
  - Progressive passcode timeout
- Touch ID based on fingerprints provides convenience

iOS Quiz

- Which are true?
  - A. Each app runs in a sandbox and has its own home directory for its files
  - B. All iOS apps must be reviewed and approved by Apple
  - C. iOS apps can be self-signed by app developers
- ANS: A, B

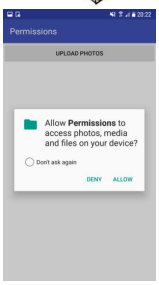
Android Security Overview

- Android is implemented in the form of a software stack architecture consisting of a Linux kernel, a runtime environment and corresponding libraries, an application framework and a set of applications.
- Each application runs its own instance of the Dalvik Virtual Machine (VM). Programs are commonly written in Java and compiled to bytecode for the Java virtual machine, which is then translated to Dalvik bytecode. The compact Dalvik Executable format is designed for systems such as smartphones that are constrained in terms of memory and processor speed.



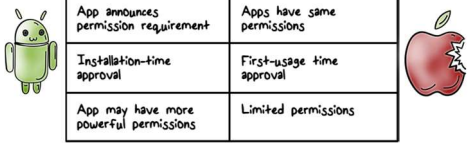
Application Sandbox

- Each app runs in its own Dalvik VM as its sandbox, which provides CPU protection and memory protection
- Each app is granted a set of permissions at install time, and cannot perform operations that require permissions it doesn't have.
  - The sandbox is based on UNIX-style user separation of processes and file permissions.
  - Permissions are implemented by mapping them to Linux groups with necessary read/write access to relevant system resources (files or sockets), enforced by the Linux kernel.



Android Sandbox vs iOS Sandbox

- The main difference between Android and iOS sandboxes is how permissions are granted.



Android	iOS
App announces permission requirement	Apps have same permissions
Installation-time approval	First-usage time approval
App may have more powerful permissions	Limited permissions

Code Signing

- All apps self-signed by developers, not by any central authority
- Code signing is used for
  - making sure updates for an app come from the same author (same origin policy)
  - establishing trust relationships between apps for code/data sharing, and running in the same process

Android Apps Quiz

- Which are true?
  - A. Android apps can be self-signed.
  - B. Android apps can have more powerful permissions than iOS apps.
- ANS: A, B

Summary

- Use WPA2 for WiFi security
- iOS has cryptographic keys and modules built into its device hardware, uses mandatory code signing and a very restricted app distribution model, and runs app in a sandbox with run-time protection mechanisms such as ASLR and DEP
- Android is based on Linux and the sandbox model is based on Unix-style user separation, and its apps are self-signed

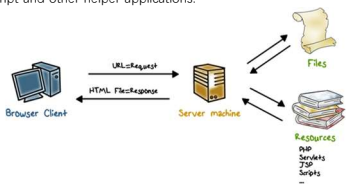
CH25 Web Security

Outline

- Overview of Web and security vulnerabilities
- Cross Site Scripting
- Cross Site Request Forgery
- SQL Injection

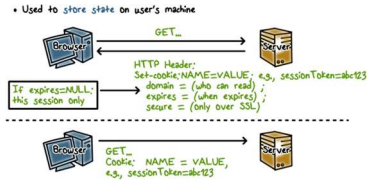
How the Web Works

- The web browser and the web server communicate using HyperText Transfer Protocol (HTTP)
- Browser requests "documents" (or scripts) through URL.
- The Server responds with "documents" in Hyper-Text Markup Language (HTML), which can include not just text but also graphics, video/audio, postscript, JavaScript, etc.
- Browsers display html documents and embedded graphics, it can run JavaScript and other helper applications.



Cookies

- HTTP is a stateless protocol, that is, each request is its own TCP connection. For example, if you log into your bank's web site, each click on a URL generates a separate TCP connection. In order to carry information across multiple http requests, such as user authentication, cookies are used.
- A cookie is created by the web server when the user logs into the site. It contains not only user identity information but also security information such as access, expiration time, and if SSL is required.
- The user's browser stores the cookie, and includes it in subsequent requests so that the server knows that these requests are related, for example, they belong to the same user login session.



Cookie Quiz

- Which are true?
  - A. Cookies may be created by ads that run on websites
  - B. Cookies are created by websites a user is visiting
  - C. Cookies are compiled pieces of code
  - D. Cookies can be used as a form of virus
  - E. Cookies can be used as a form of spyware
  - F. All of the above
- ANS: A, B, E
  - C is false, since cookies are plain text.
  - D is false, since cookies are not executable code like a virus
  - E is true, since cookies store user preferences and browsing history.

The Web and Security

- Web pages may contain dynamic contents, e.g., JavaScript
  - Sent from a website(s)
  - Run on the user's browser/machine
- Can a browser trust these contents?
  - In some cases, the browser can authenticate the website, but in many cases, authentication is not required.
  - But even if a website is authenticated, the contents that it sends may not be trustworthy because it may have security vulnerabilities that allow attackers to inject malicious content that gets passed onto the users visiting the website.
  - Or the website includes contents or links to other websites, which may also have security vulnerabilities.

The Web and Security cont'd

- Websites may run applications (e.g., PHP) to generate response/page
  - According to requests from a user/browser
  - Often communicate with back-end servers
- These web applications may have security vulnerabilities.
- Furthermore, many websites do not authenticate users, so attackers may send requests designed to exploit security vulnerabilities.

Web Browser Quiz

- Which are true?
  A. Web browser can be attacked by any web site that it visits
  B. Even if a browser is compromised, the rest of the computer is still secure
  C. Web servers can be compromised because of exploits on web applications
- ANS: A, C
  - B is false, since browser compromise may lead to malware installation, data theft, etc.)
  - C is true, e.g., website defacing, stolen credit cards

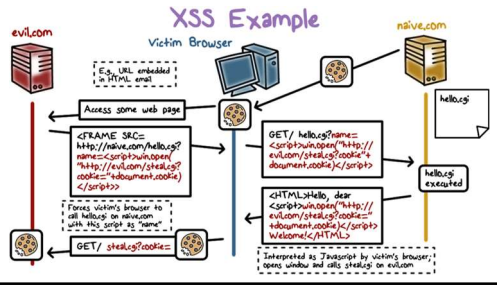
Cross-Site Scripting (XSS)

- Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites.

Cross-Site Scripting (XSS) cont'd

- Supposr a website allows users to input data and then echos the data back, that is, include the user-input data in the html page to the user's browser. Such websites include social networking sites, blogs, etc.
- Suppose the browser sends to the site <script type="text/javascript">alert("Hello World"); </script> as his "name"
- The script will be included in the html page sent to the user's browser; and when the script runs, the alert "Hello World" will be displayed
- What if the script is malicious, and the browser had sent it without the user knowing about it? Then the browser would execute the malicious script.

XSS Example



XSS Example Explanations

- The user has logged into a vulnerable, trusted site naive.com and his browser now stores a cookie from naive.com.
- The user is phished and clicks a URL to visit evil.com.
- evil.com returns a page that has a hidden frame that forces the browser to visit naive.com and invoke hello.cgi, with a malicious script as the "name" of the user.
- hello.cgi at naive.com echoes the name, which is actually the malicious script in the html page that is sent back to the user's browser.
- The user's browser displays the html page and executes the malicious script since it **came from a trusted site** naive.com, which may steal the cookie to naive.com and send it to the attacker.
- So what if evil.com gets the cookie for naive.com? Cookies can include session authenticators for naive.com, that is, the attacker can now impersonate the user.

XSS Query Quiz

- Which are true?
  - A. When a user's browser visits a compromised or malicious site, a malicious script is returned
  - B. To prevent XSS, any user input must be checked and preprocessed before it is used inside html (e.g., ensure that it is data, not a script)
- ANS: A, B

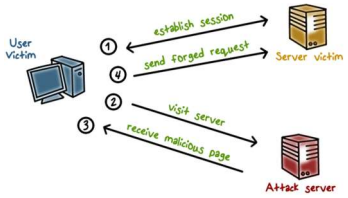
505

Cross-Site Request Forgery (XSRF)

- Browser runs a script from a "good" site and a malicious script from a "bad" site
  - This can happen when the user has logged into the good site and keeps the session alive, e.g., the user has logged into Gmail, and has not logged off. Meanwhile, the user may be browsing other sites, include a bad site that sends malicious script to the browser.
- Malicious script can make forged requests to "good" site with user's cookie.

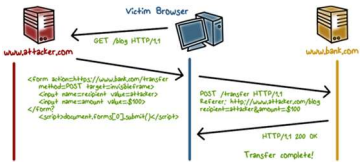
XSRF Illustration

- 1. User logs in and establishes a session with a good site, and keeps the session alive.
- 2, 3. Meanwhile, user browses a bad site, e.g., because he is phished, and the browser runs a malicious script from the bad site.
- 4. The malicious script then sends forged request to the good site.



XSRF: Example

- User logs into bank.com, forgets to sign off, so the Session cookie remains in browser state
- User then visits a malicious website, which sends a HTML page that contains a hidden iframe that includes this malicious element:
  - <form name=BillPayForm
  - action=http://bank.com/BillPay.php>
  - <input name=recipient value=badguy> …
  - <script> document.BillPayForm.submit(); </script>
- When the user's browser displays the HTML page, actions will be performed on the Bill Payment form of the bank.com page as if the users are entering these values. Because the iframe is invisible, the user knows nothing about it. The browser will send a request on behalf of the user without his knowledge.
- Since the user is still logged into bank.com, the user's cookie is also sent to the bank along with the request. So the bank website believes the request is from the user, and fulfills the payment request.



508

XSRF Mitigations

- Checking the http Referer header to see if the request comes from an authorized page.
- Use synchronizer token pattern where a token for each request is embedded by the web application in all html forms and verified on the server side.
- Log off immediately after using a web application.
- Do not allow browser to save username/password and do not allow web sites to "remember" user login
- Do not use the same browser to access sensitive web sites and to surf the web freely

509

XSS vs. XSRF

- XSS (Cross-Site Scripting) doesn't need an authenticated session and can be exploited when the vulnerable website doesn't do the basics of validating or escaping input.
- CSRF (Cross-site Request Forgery) happens in authenticated sessions when the server trusts the user/browser

Structured Query Language (SQL)

- Widely used database query language
- Retrieve a set of records, e.g.,
  SELECT * FROM Person WHERE Username='Lee'
- Add data to the table, e.g.,
  INSERT INTO Key (Username, Key) VALUES ('Lee', lfoutw2)
- Modify data, e.g.,
  UPDATE Keys SET Key=ifoutw2 WHERE PersonID=8

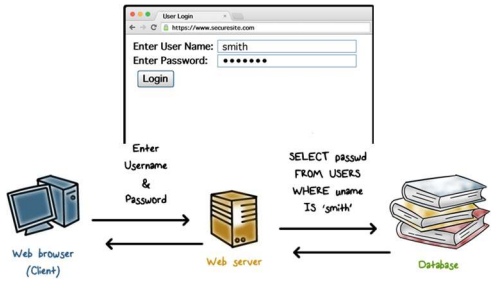
Sample PHP Code

- Sample PHP

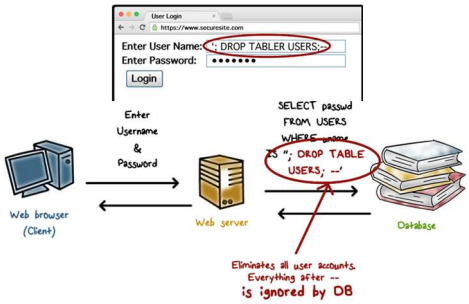
```
$selecteduser = $_GET['user'];
$sql = "SELECT Username, Key FROM Key".
       "WHERE Username='$selecteduser'";
$rs = $db->executeQuery($sql);
```

- What if 'user' is a malicious string that changes the meaning of the query?

Example Login Prompt



Example SQL Injection Attack



SQL Injection Quiz

- Which is the better way to prevent SQL injection?
  A. Use blacklisting to filter out "bad" input
  B. Use whitelisting to allow only well-defined set of safe values
- ANS: B.
  - A is infeasible, since there can be many possible ways to inject malicious strings.

Summary

- Both browser and servers are vulnerable: dynamic contents based on user input
- XSS: attacker injects a script into a website and the user's browser executes it
- XSRF: attacker tricks user's browser into issuing request, and the website executes it
- SQL injection: attacker inject malicious query actions, and a website's back-end db server executes the query