Security

Amir H. Payberah
amir@sics.se

Amirkabir University of Technology
(Tehran Polytechnic)

The Security Problem

Security

- Protection is strictly an internal problem.
  - How do we provide controlled access to programs and data stored in a computer system?

Security

- Protection is strictly an internal problem.
  - How do we provide controlled access to programs and data stored in a computer system?
- Security requires both protection system and the consideration of the external environment within which the system operates.

The Security Problem

- A system is secure if its resources are used and accessed as intended under all circumstances.

The Security Problem

- A system is secure if its resources are used and accessed as intended under all circumstances.
- Threat is potential security violation.

The Security Problem

- A system is secure if its resources are used and accessed as intended under all circumstances.
- Threat is potential security violation.
- Attack is attempt to breach security: accidental or malicious.

Security Violation Categories

- Breach of confidentiality: unauthorized reading of data

Security Violation Categories

- Breach of confidentiality: unauthorized reading of data
- Breach of integrity: unauthorized modification of data

Security Violation Categories

- Breach of confidentiality: unauthorized reading of data
- Breach of integrity: unauthorized modification of data
- Breach of availability: unauthorized destruction of data

Security Violation Categories

- Breach of confidentiality: unauthorized reading of data
- Breach of integrity: unauthorized modification of data
- Breach of availability: unauthorized destruction of data
- Theft of service: unauthorized use of resources

Security Violation Categories

- Breach of confidentiality: unauthorized reading of data
- Breach of integrity: unauthorized modification of data
- Breach of availability: unauthorized destruction of data
- Theft of service: unauthorized use of resources
- Denial of Service (DoS): prevention of legitimate use

Security Violation Methods

- Masquerading (breach authentication): pretending to be an authorized user to escalate privileges

Security Violation Methods

- Masquerading (breach authentication): pretending to be an authorized user to escalate privileges

- Replay attack: repeat a valid data transmission

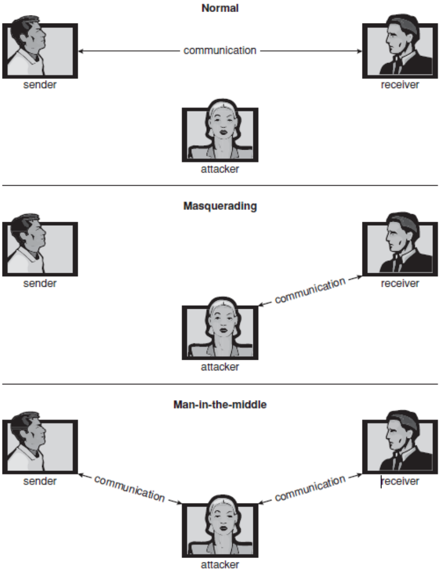
Security Violation Methods

- Masquerading (breach authentication): pretending to be an authorized user to escalate privileges
- Replay attack: repeat a valid data transmission
- Man-in-the-middle attack: intruder sits in data flow, masquerading as sender to receiver and vice versa

Security Violation Methods

- Masquerading (breach authentication): pretending to be an authorized user to escalate privileges
- Replay attack: repeat a valid data transmission
- Man-in-the-middle attack: intruder sits in data flow, masquerading as sender to receiver and vice versa
- Session hijacking: intercept an already-established session to bypass authentication

Standard Security Attacks



Security Measure Levels

- Security must occur at four levels to be effective:
  - Physical: data centers, servers, connected terminals
  - Human: only appropriate users have access to the system
  - OS: protection mechanisms, debugging
  - Network: intercepted communications, interruption, DOS

Program Threats

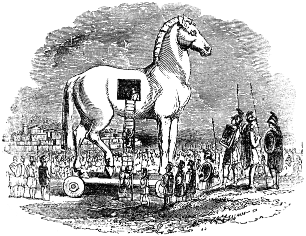
Program Threats

- Many variations, many names
  - Trojan horse
  - Trap door
  - Logic bomb
  - Stack and buffer flow
  - Viruses

Trojan Horse

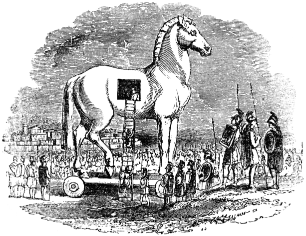
Trojan Horse (1/2)

- Code segment that misuses its environment.



Trojan Horse (1/2)

- Code segment that misuses its environment.
- Exploits mechanisms for allowing programs written by users to be executed by other users.
- Example:
  - A text-editor program has a code to search the file to be edited for certain keywords.
  - If any are found, the entire file may be copied to a special area accessible to the creator of the text editor.



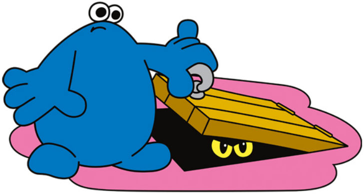
Trojan Horse (2/2)

- Variation of Trojan horse:
- Emulating a login program
- Spyware: accompanies a program that the user has installed.
  - Download ads to display on the user's system
  - Create pop-up browser windows when certain sites are visited
  - Capture information from the user's system and return it to a central site: covert channel

Trap Door

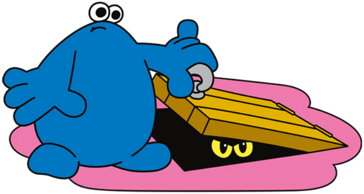
Trap Door

- A hole in the software that only designer of a program is capable of using.



Trap Door

- A hole in the software that only designer of a program is capable of using.
- Could be included in a compiler.
- Difficult to detect: we have to analyze all the source code for all components of a system.



Logic Bomb

Logic Bomb

- Program that initiates a security incident under certain circumstances.

- Hard to detect: because under normal operations, there would be no security hole.

Stack and Buffer Flow

Stack and Buffer Overflow (1/6)

- The most common way for an attacker to gain unauthorized access to the target system.

Stack and Buffer Overflow (1/6)

- The most common way for an attacker to gain unauthorized access to the target system.
- The attack exploits a bug in a program.
  - E.g., The programmer neglected to code bounds checking on an input field.
  - The attacker sends more data than the program was expecting.
  - The attacker can write a program to do the next page steps.

Stack and Buffer Overflow (2/6)

1. Overflow an input field, for example, a web-page form expects a user name, until it writes into the stack.

Stack and Buffer Overflow (2/6)

1. Overflow an input field, for example, a web-page form expects a user name, until it writes into the stack.

2. Overwrite the current return address on the stack with the address of the exploit code loaded in step 3.

Stack and Buffer Overflow (2/6)

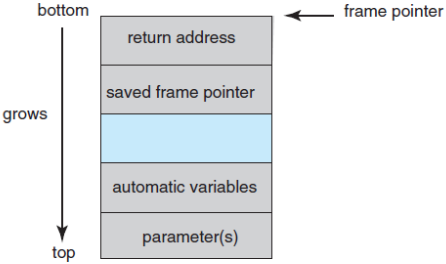
1. Overflow an input field, for example, a web-page form expects a user name, until it writes into the stack.
2. Overwrite the current return address on the stack with the address of the exploit code loaded in step 3.
3. Write a simple set of code for the next space in the stack that includes the commands that the attacker wishes to execute, for instance, spawn a shell.

Stack and Buffer Overflow (3/6)

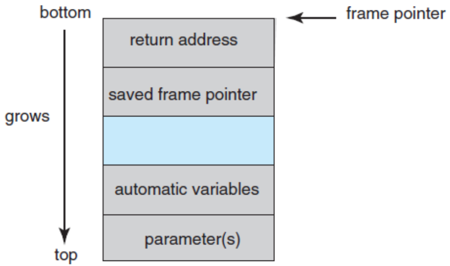
```
#include <stdio.h>
#define BUFFER_SIZE 256

int main(int argc, char *argv[])
{
   char buffer[BUFFER_SIZE];

   if (argc < 2)
      return -1;
   else {
      strcpy(buffer,argv[1]);
      return 0;
   }
}
```

- Lack of bounds checking
- If the command line input is longer than `BUFFER_SIZE`?

Stack and Buffer Overflow (4/6)

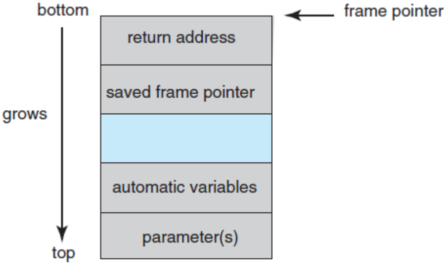


Stack and Buffer Overflow (4/6)



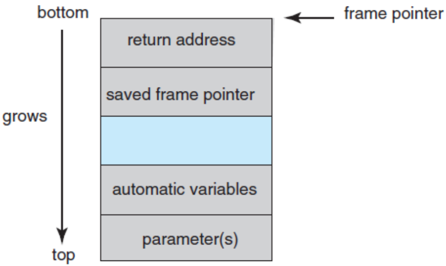
- Automatic variables: variables defined locally to the function.

Stack and Buffer Overflow (4/6)



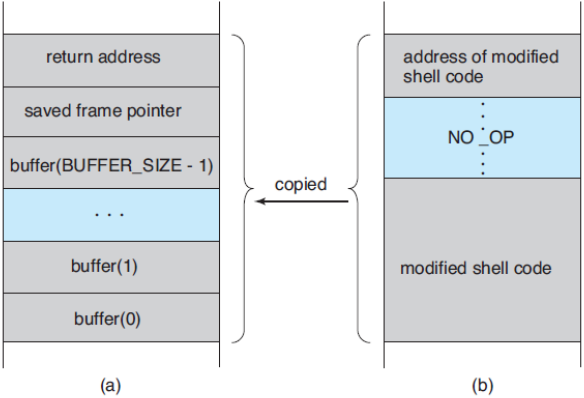
- Automatic variables: variables defined locally to the function.
- Frame pointer: the address of the beginning of the stack frame.
  - Can vary during the function call.

Stack and Buffer Overflow (4/6)



- Automatic variables: variables defined locally to the function.
- Frame pointer: the address of the beginning of the stack frame.
  - Can vary during the function call.
- Return address: where to return control once the function exits.

Stack and Buffer Overflow (5/6)



Stack and Buffer Overflow (6/6)

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    execvp(''\bin\sh'',''\bin \sh'', NULL);
    return 0;
}
```

- The cracker could replace the return address with the address of the code segment containing the attacking program.

Viruses

Viruses

- Code fragment embedded in legitimate program.
- Self-replicating: designed to infect other computers.
- Very specific to CPU architecture, OS, applications.
- Usually borne via email or as a macro.

System and Network Threats

System and Network Threats

- System and network threats involve the abuse of services and network connections.

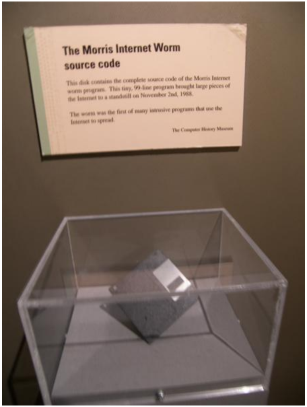
- They create a situation in which the OS resources and user files are misused.

System and Network Threats

- System and network threats involve the abuse of services and network connections.
- They create a situation in which the OS resources and user files are misused.
- They include:
  - Worms
  - Port scanning
  - Denial of Service (DoS)

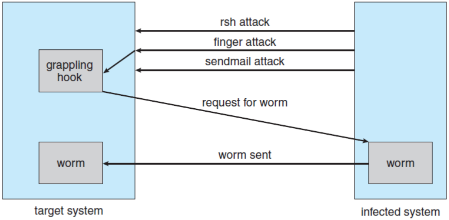
Worms (1/2)

- An standalone program that replicates itself in order to spread to other computers.
- Often, it uses a computer network to spread itself.
- Unlike a computer virus, it does not need to attach itself to an existing program.
- The Morris worm (the Internet worm) is the first computer worms distributed via the Internet: 1988



Worms (2/2)

- Morris worm: exploited UNIX networking features in `rsh` and bugs in `finger` and `sendmail` programs.
- It made up of two programs: a grappling hook program and the main program.
- The grappling hook program uploaded main worm program.



Port Scanning

- A means to detect a system's vulnerabilities to attack.
- Automated attempt to connect to a range of ports on IP addresses.
- Detection of answering service protocols, OS and version running on system.

Port Scanning

- A means to detect a system's vulnerabilities to attack.
- Automated attempt to connect to a range of ports on IP addresses.
- Detection of answering service protocols, OS and version running on system.
- `nmap` scans all ports in a given IP range for a response
- `nessus` has a database of protocols and bugs (and exploits) to apply against a system.

Denial of Service (DoS)

- Overload the targeted computer preventing it from doing any useful work.
- Distributed denial-of-service (DDoS) come from multiple sites at once.
- Consider traffic to a web site.
- CS students writing bad `fork()` code.

Cryptography

Cryptography

- Cryptography is used to constrain the potential senders and/or receivers of a message.

Cryptography

- Cryptography is used to constrain the potential senders and/or receivers of a message.

- Based on secrets, called keys.

Cryptography

- Cryptography is used to constrain the potential senders and/or receivers of a message.

- Based on secrets, called keys.

- Cryptography enables a recipient of a message to verify that the message was created by some computer possessing a certain key.

Cryptography

- Cryptography is used to constrain the potential senders and/or receivers of a message.

- Based on secrets, called keys.

- Cryptography enables a recipient of a message to verify that the message was created by some computer possessing a certain key.

- Similarly, a sender can encode its message so that only a computer with a certain key can decode the message.

Encryption (1/2)

- Encryption algorithm consists of:
- Set K of keys
- Set M of messages
- Set C of ciphertexts (encrypted messages)

Encryption (1/2)

- Encryption algorithm consists of:
- Set K of keys
- Set M of messages
- Set C of ciphertexts (encrypted messages)
- A function $E : K \to (M \to C)$
  - For each $k \in K$, E_k is a function for generating ciphertexts from messages.

Encryption (1/2)

- Encryption algorithm consists of:
- Set K of keys
- Set M of messages
- Set C of ciphertexts (encrypted messages)
- A function $E : K \rightarrow (M \rightarrow C)$
  - For each $k \in K$, E_k is a function for generating ciphertexts from messages.
- A function $D : K \rightarrow (C \rightarrow M)$
  - For each $k \in K$, D_k is a function for generating messages from ciphertexts.

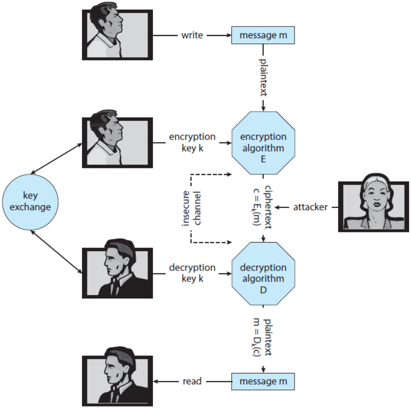
Encryption (2/2)

- An encryption algorithm must provide this essential property:
  - Given a ciphertext $c \in C$, a computer can compute m such that $E_k(m) = c$ only if it possesses k.

- Thus, a computer holding k can decrypt ciphertexts to the plaintexts.

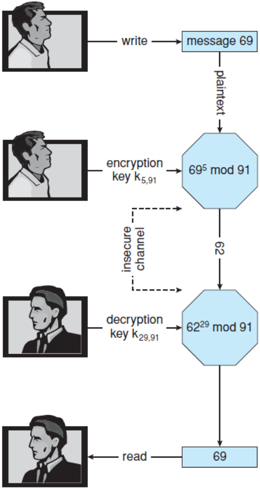
Symmetric Encryption

- Same key used to encrypt and decrypt.
  - k must be kept secret.
- E.g., DES, Triple-DES, AES, RC4



Asymmetric Encryption

- Each user having two keys:
  - Public key: published key used to encrypt data.
  - Private key: key known only to individual user used to decrypt data.
- Most common is RSA block cipher.



Authentication (1/2)

- Constraining set of potential senders of a message.
- This sort of authentication is similar to but distinct from user authentication.

Authentication (1/2)

- Constraining set of potential senders of a message.
- This sort of authentication is similar to but distinct from user authentication.
- Algorithm components:
  - A set K of keys
  - A set M of messages
  - A set A of authenticators

Authentication (1/2)

- Constraining set of potential senders of a message.
- This sort of authentication is similar to but distinct from user authentication.
- Algorithm components:
  - A set K of keys
  - A set M of messages
  - A set A of authenticators
  - A function $S : K \rightarrow (M \rightarrow A)$: for each $k \in K$, S_k is a function for generating authenticators from messages.
  - A function $V : K \rightarrow (M \times A \rightarrow \{\texttt{true}, \texttt{false}\})$: for each $k \in K$, V_k is a function for verifying authenticators on messages.

Authentication (2/2)

- For a message m, a computer can generate an authenticator $a \in A$ such that $V_k(m, a) = \texttt{true}$ only if it possesses k.

Authentication (2/2)

- For a message m, a computer can generate an authenticator $a \in A$ such that $V_k(m, a) = \texttt{true}$ only if it possesses k.
- Thus, computer holding k can generate authenticators on messages so that any other computer possessing k can verify them.

Authentication (2/2)

- For a message m, a computer can generate an authenticator $a \in A$ such that $V_k(m, a) = \texttt{true}$ only if it possesses k.

- Thus, computer holding k can generate authenticators on messages so that any other computer possessing k can verify them.

- Practically, if $V_k(m, a) = \texttt{true}$ then we know m has not been modified and that send of message has k.

Authentication - Hash Function (1/2)

- Creates small, fixed-size block of data message digest (hash value) from m.

Authentication - Hash Function (1/2)

- Creates small, fixed-size block of data message digest (hash value) from m.
- Hash Function H must be collision resistant on m.

Authentication - Hash Function (1/2)

- Creates small, fixed-size block of data message digest (hash value) from m.
- Hash Function H must be collision resistant on m.
- Must be infeasible to find an $m' \neq m$ such that $H(m) = H(m')$.
  - If $H(m) = H(m')$, then $m = m'$.

Authentication - Hash Function (1/2)

- Creates small, fixed-size block of data message digest (hash value) from m.
- Hash Function H must be collision resistant on m.
- Must be infeasible to find an $m' \neq m$ such that $H(m) = H(m')$.
  - If $H(m) = H(m')$, then $m = m'$.
- The message has not been modified.

Authentication - Hash Function (1/2)

- Creates small, fixed-size block of data message digest (hash value) from m.
- Hash Function H must be collision resistant on m.
- Must be infeasible to find an $m' \neq m$ such that $H(m) = H(m')$.
  - If $H(m) = H(m')$, then $m = m'$.
- The message has not been modified.
- E.g., MD5 (128-bit hash), and SHA-1 (160-bit hash)

Authentication - Hash Function (2/2)

- Not useful as authenticators.
  - For example $H(m)$ can be sent with a message.
  - But if H is known someone could modify m to m' and recompute $H(m')$ and modification not detected
  - So must authenticate $H(m)$: MAC and digital signature

Authentication - MAC

- Symmetric encryption used in message-authentication code (MAC) authentication algorithm.

- Cryptographic checksum generated from message using secret key.

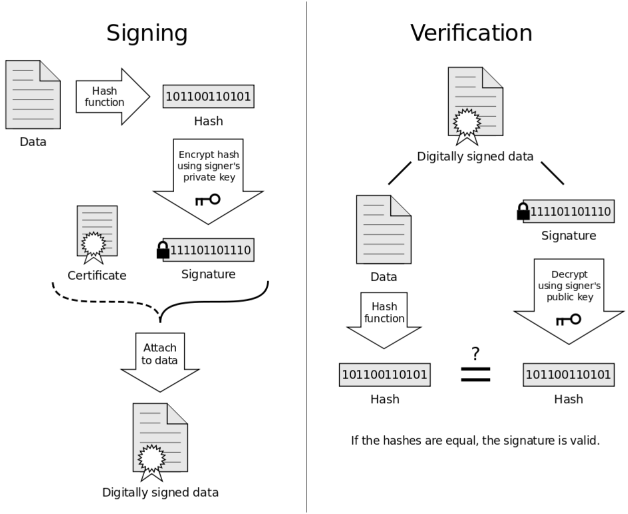
Authentication - MAC

- Symmetric encryption used in message-authentication code (MAC) authentication algorithm.
- Cryptographic checksum generated from message using secret key.
- Note that k is needed to compute both S_k and V_k, so anyone able to compute one can compute the other end digital signature.

Authentication - Digital Signature (1/2)

- Based on asymmetric keys and digital signature algorithm.
- Authenticators produced are digital signatures.
- Anyone can verify authenticity of a message.
- In a digital-signature algorithm, computationally infeasible to derive k_s from k_v.
- k_v is the public key and k_s is the private key.

Authentication - Digital Signature (2/2)



Authentication - Key Distribution

- Symmetric key
  - Sometimes done out-of-band, e.g., via a paper document or a conversation
  - If a user wants to communicate with N other users privately: she needs N keys.

Authentication - Key Distribution

- Symmetric key
  - Sometimes done out-of-band, e.g., via a paper document or a conversation
  - If a user wants to communicate with N other users privately: she needs N keys.
- Asymmetric keys
  - The keys can be exchanged in public.
  - A user needs only one private key, no matter how many other people she wants to communicate with.

Authentication - Key Distribution

- Symmetric key
  - Sometimes done out-of-band, e.g., via a paper document or a conversation
  - If a user wants to communicate with N other users privately: she needs N keys.
- Asymmetric keys
  - The keys can be exchanged in public.
  - A user needs only one private key, no matter how many other people she wants to communicate with.
  - Man-in-a-middle attack.
  - What we need is a proof of who owns a public key? digital certificate

Authentication - Digital Certificates

- Digital certificate is an electronic document used to prove ownership of a public key.

Authentication - Digital Certificates

- Digital certificate is an electronic document used to prove ownership of a public key.

- It includes information about:
  - The key
  - Its owner's identity
  - The digital signature of an entity that has verified the certificate's contents are correct.

Authentication - Digital Certificates

- Digital certificate is an electronic document used to prove ownership of a public key.
- It includes information about:
  - The key
  - Its owner's identity
  - The digital signature of an entity that has verified the certificate's contents are correct.
- Certificate authority are trusted party - their public keys included with web browser distributions.

User Authentication

User Authentication

- Our earlier discussion of authentication involves messages and sessions. But what about users?

User Authentication

- Our earlier discussion of authentication involves messages and sessions. But what about users?

- If a system cannot authenticate a user, then authenticating that a message came from that user is pointless.

User Authentication

- Our earlier discussion of authentication involves messages and sessions. But what about users?

- If a system cannot authenticate a user, then authenticating that a message came from that user is pointless.

- How do we determine whether a users identity is authentic?

User Authentication

- Our earlier discussion of authentication involves messages and sessions. But what about users?

- If a system cannot authenticate a user, then authenticating that a message came from that user is pointless.

- How do we determine whether a users identity is authentic?
  - The user's possession of something (a key or card)

User Authentication

- Our earlier discussion of authentication involves messages and sessions. But what about users?
- If a system cannot authenticate a user, then authenticating that a message came from that user is pointless.
- How do we determine whether a users identity is authentic?
  - The user's possession of something (a key or card)
  - The user's knowledge of something (a user identifier and password)

User Authentication

- Our earlier discussion of authentication involves messages and sessions. But what about users?
- If a system cannot authenticate a user, then authenticating that a message came from that user is pointless.
- How do we determine whether a users identity is authentic?
  - The user's possession of something (a key or card)
  - The user's knowledge of something (a user identifier and password)
  - An attribute of the user (fingerprint, retina pattern, or signature)

Password

- User identity most often established through passwords.

Password

- User identity most often established through passwords.
- Encrypt the passwords
  - But keep them secret anyway (i.e. Unix uses superuser-only readable file `/etc/shadow`)
  - Use algorithm easy to compute but difficult to invert.
  - Only encrypted password stored, never decrypted.

Password

- User identity most often established through passwords.
- Encrypt the passwords
  - But keep them secret anyway (i.e. Unix uses superuser-only readable file `/etc/shadow`)
  - Use algorithm easy to compute but difficult to invert.
  - Only encrypted password stored, never decrypted.
- One-time passwords
  - Use a function based on a seed to compute a password, both user and computer.

Implementing Security Defenses

Implementing Security Defenses (1/2)

- Defense in depth is most common security theory: multiple layers of security.

Implementing Security Defenses (1/2)

- Defense in depth is most common security theory: multiple layers of security.

- Security policy describes what is being secured.

Implementing Security Defenses (1/2)

- Defense in depth is most common security theory: multiple layers of security.
- Security policy describes what is being secured.
- Vulnerability assessment compares real state of a system/network to the security policy.

Implementing Security Defenses (1/2)

- Defense in depth is most common security theory: multiple layers of security.
- Security policy describes what is being secured.
- Vulnerability assessment compares real state of a system/network to the security policy.
- Intrusion detection systems to detect attempted or successful intrusions.
  - Signature-based detection spots known bad patterns.
  - Anomaly detection spots differences from normal behavior.

Implementing Security Defenses (2/2)

- Virus protection
  - Searching all programs or programs at execution for known virus patterns.

Implementing Security Defenses (2/2)

- Virus protection
  - Searching all programs or programs at execution for known virus patterns.

- Auditing, accounting, and logging of all or specific system or network activities.

Summary

Summary

- Security and Protection

Summary

- Security and Protection
- Security violation categories: breach of confidentiality / integrity / availability, theft of service, and DoS

Summary

- Security and Protection
- Security violation categories: breach of confidentiality / integrity / availability, theft of service, and DoS
- Security violation methods: masquerading, replay attack, man-in-the-middle, and session hijacking

Summary

- Security and Protection
- Security violation categories: breach of confidentiality / integrity / availability, theft of service, and DoS
- Security violation methods: masquerading, replay attack, man-in-the-middle, and session hijacking
- Program threats: Trojan horse, trap door, logic bomb, stack and buffer flow, viruses

Summary

- Security and Protection
- Security violation categories: breach of confidentiality / integrity / availability, theft of service, and DoS
- Security violation methods: masquerading, replay attack, man-in-the-middle, and session hijacking
- Program threats: Trojan horse, trap door, logic bomb, stack and buffer flow, viruses
- System and network threats: worms, port scanning, DoS

Summary

- Security and Protection
- Security violation categories: breach of confidentiality / integrity / availability, theft of service, and DoS
- Security violation methods: masquerading, replay attack, man-in-the-middle, and session hijacking
- Program threats: Trojan horse, trap door, logic bomb, stack and buffer flow, viruses
- System and network threats: worms, port scanning, DoS
- Cryptography: encryption (symmetric/asymmetric), authentication (MAC/digital signature), key distribution and digital certificate

Summary

- Security and Protection
- Security violation categories: breach of confidentiality / integrity / availability, theft of service, and DoS
- Security violation methods: masquerading, replay attack, man-in-the-middle, and session hijacking
- Program threats: Trojan horse, trap door, logic bomb, stack and buffer flow, viruses
- System and network threats: worms, port scanning, DoS
- Cryptography: encryption (symmetric/asymmetric), authentication (MAC/digital signature), key distribution and digital certificate
- User authentication: password

Questions?