COP 4610

Operating System Principles

Security

1

The Security Problem

- System **secure** if resources used and accessed as intended under all circumstances
  - Unachievable!
- Intruders (malicious users) attempt to breach security
- **Threat** is potential security violation
- **Attack** is an attempt to breach security
- Attack can be **accidental** or **malicious**
- Easier to protect against accidental than malicious misuse

2

Security Violation Categories

- CIA triad:
  - 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

3

Security Violation Methods

- **Masquerading**
  - Pretending to be an authorized user to escalate privileges
- **Replay attack**
  - As is or with message modification
- **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

4

Security Measure Levels

- Impossible to have absolute security, but make cost to perpetrator sufficiently high to deter most intruders
- Security must occur at four levels to be effective:
  - **Physical**
  - **Human**
  - **Operating System**
  - **Network**

5

Program Threats

- Many variations, many names
- **Trojan Horse**
  - Code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
- **Trap Door**
  - Specific user identifier or password that circumvents normal security procedures
  - Typically meant for benign purposes

6

Program Threats (Cont.)

- **Logic Bomb**
  - Program that initiates a security incident under certain circumstances
- **Stack** and **Buffer Overflow**
  - Exploits a bug in a program (overflow either the stack or memory buffers)
  - Failure to check bounds on inputs, arguments
  - Write past arguments on the stack into the return address on stack
  - When routine returns from call, returns to hacked address
    - Pointed to code loaded onto stack that executes malicious code
  - Unauthorized user or privilege escalation

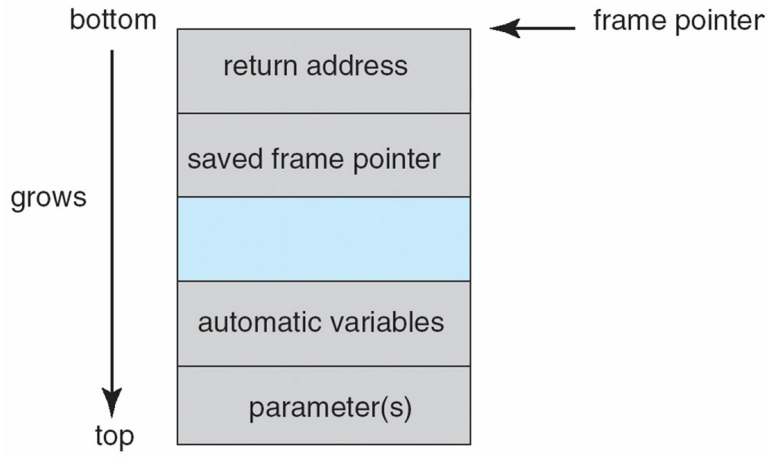
7

C Program with Buffer-overflow Condition

```
#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;
  }
}
```


8

Layout of Typical Stack Frame



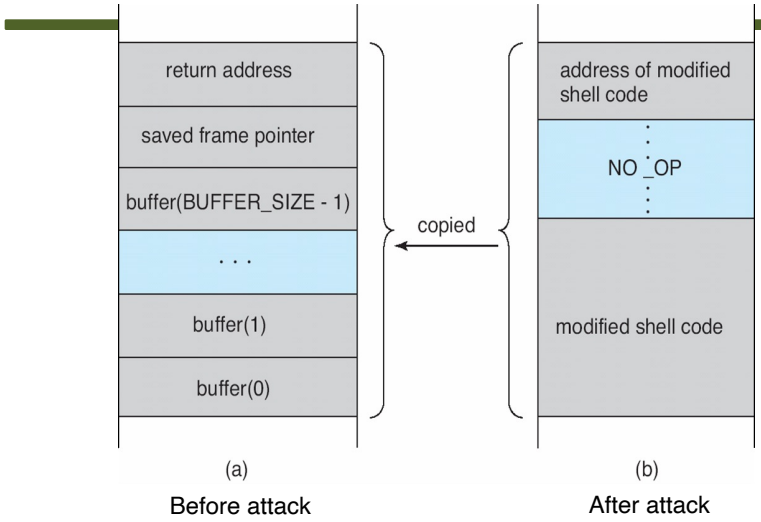
9

Modified Shell Code

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  execvp(''\bin\sh'',''\bin
  \sh'', NULL);
  return 0;
}
```


10

Hypothetical Stack Frame



11

Program Threats (Cont.)

- **Viruses**
  - Code fragment embedded in legitimate program
  - Self-replicating, designed to infect other computers
  - Very specific to CPU architecture, operating system, applications
  - Usually borne via email or as a macro
    - Visual Basic Macro to reformat hard drive

```
Sub AutoOpen()
Dim oFS
  Set oFS = CreateObject(''Scripting.FileSystemObject'')
  vs = Shell(''c:command.com /k format c:'',vbHide)
End Sub
```

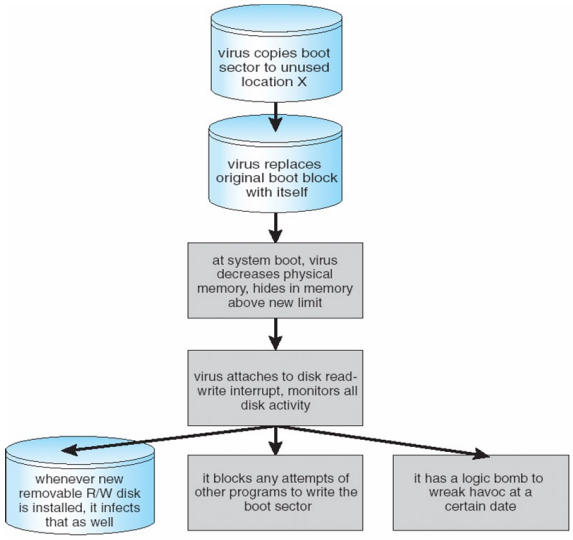

12

Program Threats (Cont.)

- **Virus dropper** inserts virus onto the system
- Many thousands of virus types
  - File / parasitic
  - Boot / memory
  - Macro
  - Source code
  - Polymorphic to avoid having a **virus signature**
  - Encrypted
  - Stealth
  - Tunneling
  - Multipartite
  - Armored

13

A Boot-Sector Computer Virus



14

The Threat Continues

- Attacks still common, still occurring
- Attacks moved over time from science experiments to tools of organized crime
  - Targeting specific companies
  - Creating **botnets** to use as tool for spam and DDOS delivery
  - **Keystroke logger** to grab passwords, credit card numbers
- Why is Windows the target for most attacks?
  - Most common
  - Everyone is an administrator
  - Monoculture considered harmful

System and Network Threats (Cont.)

- **Worms** – use **spawn** mechanism; standalone program
- Internet worm
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs
  - Exploited trust-relationship mechanism used by *rsh* to access friendly systems without use of password
  - **Grappling hook** program uploaded main worm program
    - 99 lines of C code
  - Hooked system then uploaded main code, tried to attack connected systems
  - Also tried to break into other user accounts on local system via password guessing
  - If target system already infected, abort, except for every 7th time

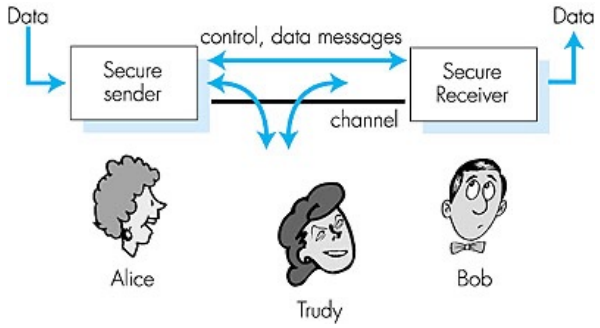
System and Network Threats (Cont.)

- **Port scanning**
  - Automated attempt to connect to a range of ports on one or a range of IP addresses
  - Detection of answering service protocol
  - Detection of 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
  - Frequently launched from **zombie systems**
    - To decrease traceability

System and Network Threats (Cont.)

- **Denial of Service**
  - Overload the targeted computer preventing it from doing any useful work
  - **Distributed denial-of-service (DDOS)** come from multiple sites at once
  - Consider the start of the TCP/IP-connection handshake (SYN)
    - How many started-connections can the OS handle?
  - Consider traffic to a web site
    - How can you tell the difference between being a target and being really popular?
  - Accidental – CS students writing bad `fork()` code
  - Purposeful – extortion, punishment

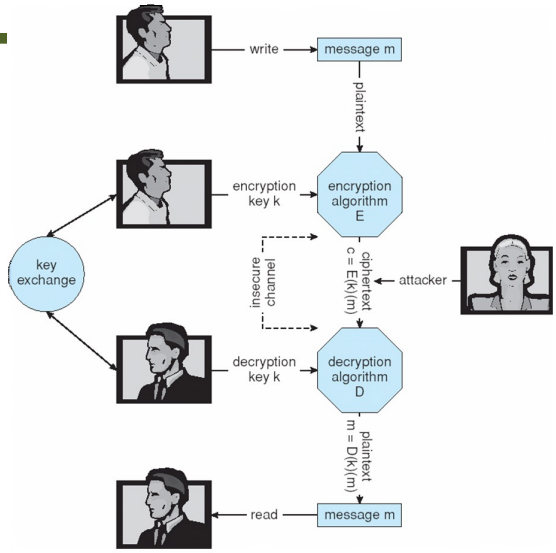
Friends and Enemies



- Bob, Alice want to communicate "securely"
- Trudy, the "intruder" may intercept, delete, add messages

19

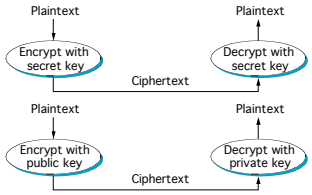
Friends and Enemies

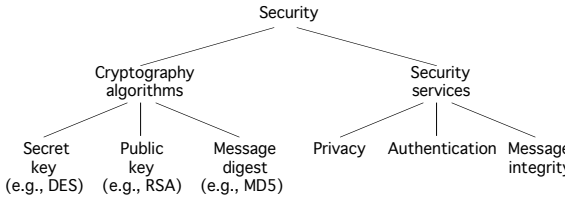


20

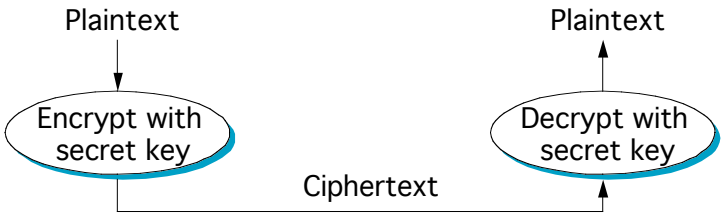
Overview

- Cryptography functions
  - Secret key (e.g., DES)
  - Public key (e.g., RSA)
  - Message digest (e.g., MD5)
- Security services
  - Privacy: preventing unauthorized release of information
  - Authentication: verifying identity of the remote participant
  - Integrity: making sure message has not been altered





Secret Key (DES)



Symmetric Key

Substitution cipher: substituting one thing for another

– monoalphabetic cipher: substitute one letter for another

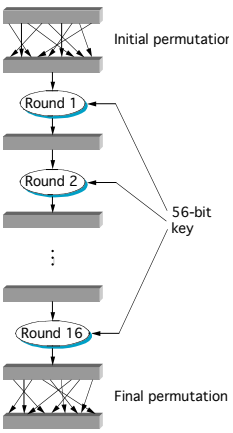
```
plaintext:  abcdefghijklmnopqrstuvwxyz
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

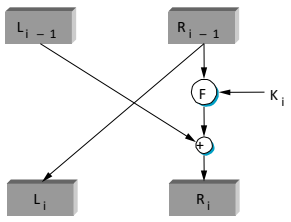
```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```


Data Encryption Standard

- 64-bit key (56-bits + 8-bit parity)
- 16 rounds

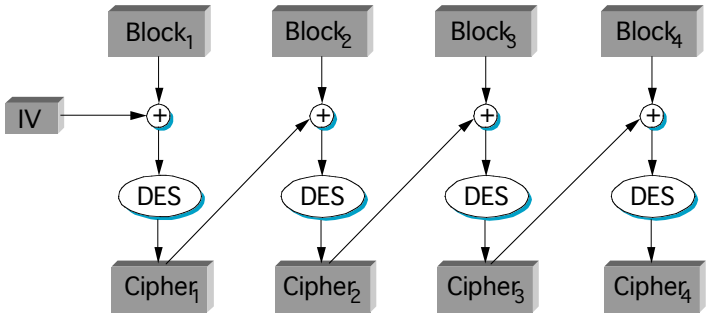


- Each Round

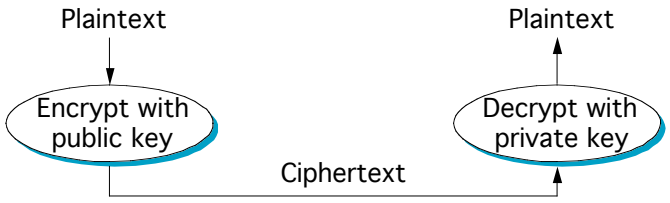


Data Encryption Standard

- Repeat for larger messages



Public Key (RSA)



- Encryption & Decryption

$c = m^e \bmod n$

$m = c^d \bmod n$

RSA (cont)

- Choose two large prime numbers *p* and *q* (each 256 bits)
- Multiply *p* and *q* together to get *n*
- Choose the encryption key *e*, such that *e* and (*p* - 1) x (*q* - 1) are relatively prime.
- Two numbers are relatively prime if they have no common factor greater than one
- Compute decryption key *d* such that

$$d*e = 1 \ mod \ ((p - 1) \times (q - 1))$$

- Construct public key as (*e*, *n*)
- Construct private key as (*d*, *n*)
- Discard (do not disclose) original primes *p* and *q*

RSA Example

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).
d=29 (so *ed-1* exactly divisible by z).

	letter	m	m^e	$c = m^e \bmod n$
encrypt:	l	12	1524832	17

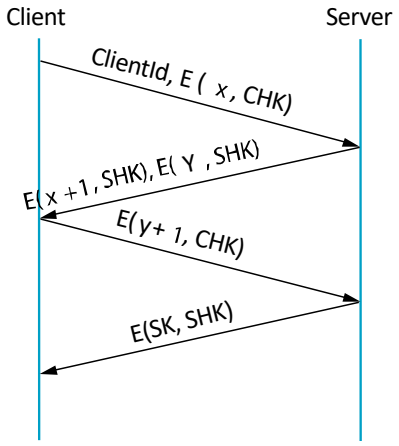
	c	c^d	$m = c^d \bmod n$	letter
decrypt:	17	481968572106750915091411825223072000	12	l

Message Digest

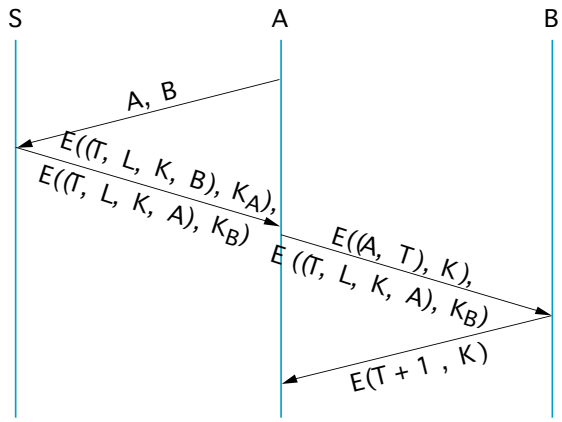
- Cryptographic checksum
  - just as a regular checksum protects the receiver from accidental changes to the message, a cryptographic checksum protects the receiver from malicious changes to the message.
- One-way function
  - given a cryptographic checksum for a message, it is virtually impossible to figure out what message produced that checksum; it is not computationally feasible to find two messages that hash to the same cryptographic checksum.
- Relevance
  - if you are given a checksum for a message and you are able to compute exactly the same checksum for that message, then it is highly likely this message produced the checksum you were given.

Authentication Protocols

- Three-way handshake

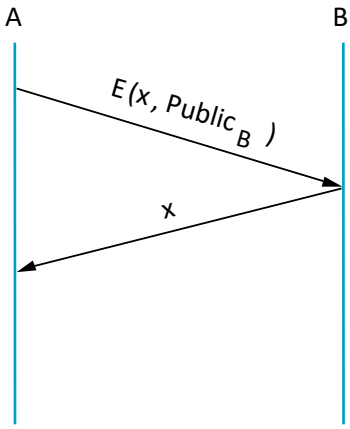


Trusted third party (Kerberos)



31

Public key authentication



32

Message Integrity Protocols

- Digital signature using RSA
  - special case of a message integrity where the code can only have been generated by one participant
  - compute signature with private key and verify with public key
- Keyed MD5
  - sender: *m* + MD5(*m* + *k*) + E(E(*k*,*rcv_public*), *snd_private*)
  - receiver
    - recovers random key using the sender's public key
    - applies MD5 to the concatenation of this random key message
- MD5 with RSA signature
  - sender: *m* + E(MD5(*m*), *private*)
  - receiver
    - decrypts signature with sender's public key
    - compares result with MD5 checksum sent with message

33

Public Key Distribution

- Certificate
  - special type of digitally signed document:
    "I certify that the public key in this document belongs to the entity named in this document, signed X."
  - the name of the entity being certified
  - the public key of the entity
  - the name of the certification authority
  - a digital signature
- Certification Authority (CA)
  - administrative entity that issues certificates
  - useful only to someone that already holds the CA's public key

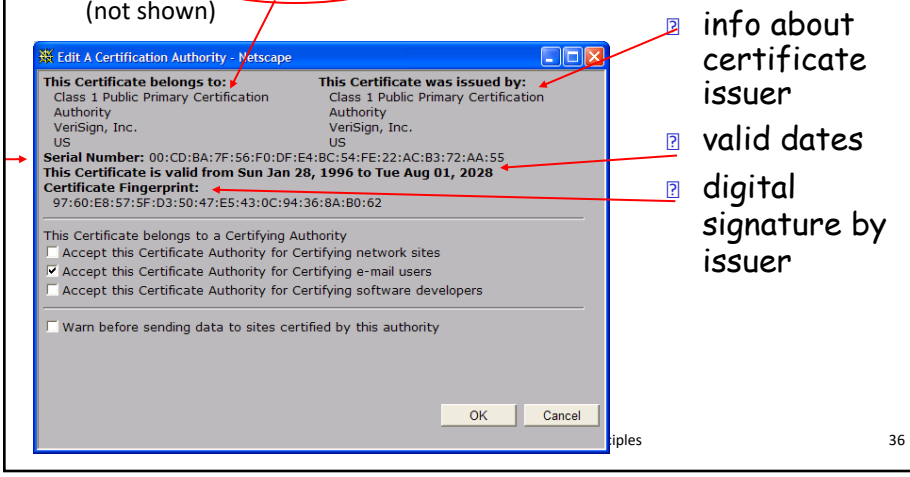
34

Key Distribution (cont)

- Chain of Trust
  - if *X* certifies that a certain public key belongs to *Y*, and *Y* certifies that another public key belongs to *Z*, then there exists a chain of certificates from *X* to *Z*
  - someone that wants to verify *Z*'s public key has to know *X*'s public key and follow the chain
- Certificate Revocation List

Certificate

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- info about certificate issuer
- valid dates
- digital signature by issuer

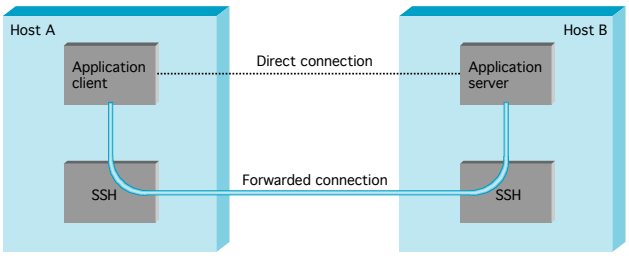
User Authentication

- Crucial to identify user correctly, as protection systems depend on user ID
- User identity most often established through **passwords**, can be considered a special case of either keys or capabilities
- Passwords must be kept secret
  - Frequent change of passwords
  - History to avoid repeats
  - Use of "non-guessable" passwords (link)
  - Log all invalid access attempts (but not the passwords themselves)

37

Secure Shell (SSH)

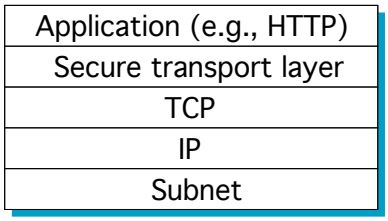
- Remote login service (replaces telnet and rlogin).
- Provides authentication, integrity, and confidentiality.
- SSH version 2: SSH-TRANS, SSH-AUTH, SSH-CONN.



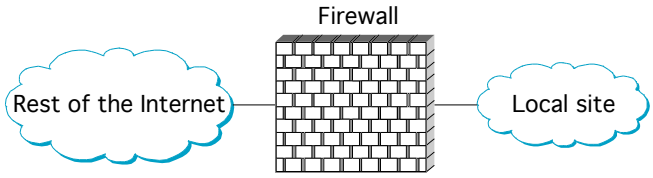
38

Transport Layer Security (TLS)

- Secure Socket Layer (SSL).
- Secure HTTP (HTTPS).
- Handshake protocol and record protocol.



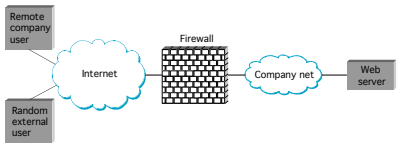
Firewalls



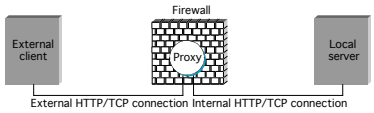
- Filter-Based Solution
  - example
    (192.12.13.14, 1234, 128.7.6.5, 80)
    (*,*, 128.7.6.5, 80)
  - default: forward or not forward?
  - how dynamic?

Proxy-Based Firewalls

- Problem: complex policy
- Example: web server



- Solution: proxy



- Design: transparent vs. classical
- Limitations: attacks from within