# CS-3002: Information Security

## Lecture # 11: Network Security Protocols and Defensive Mechanisms (Firewall, IDS, DNSSEC, DDoS)

Prof. Dr. Sufian Hameed
Department of Computer Science
FAST-NUCES

# This Lecture

- Firewall
- Intrusion Detection System (IDS)
- DNSSEC
- Distributed Denial Of Service (DDoS)

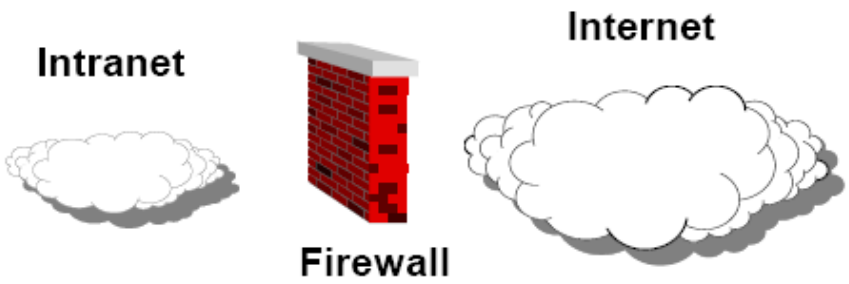
# Firewall

# Firewall

A *firewall* is any security system protecting the boundary of an Intranet against the Internet

*Tasks of a firewall:*

- *Access control* based on sender or receiver address or on addressed services (i.e. application layer protocol)
- *Behavior control*, e.g. virus checking on incoming files
- *User control*, i.e. authentication based on the source of traffic
- *Hiding* the internal network, e.g. topology, addresses, etc.
- *Logging* of passing traffic




Two fundamental concepts implemented by firewalls are

- *Packet filter*
- *Proxy server*

# Types of Firewalls

## 1. Packet Filter

- Analyzing of network traffic and filtering due to certain rules on layer 3 and 4. A filtering can use one or a combination of the following information: source address, destination address, used protocol, connection
- If the firewall is realized in combination with a router, it is also called **Screening Router**
- Cheap and simple (all types of connections can be controlled), but filtering rules are hard to define (correctly)

## 2. Proxy Server (Gateway)

- "Controlled access" to a service: the firewall intercepts a requests up to layer 7 and decides, if to forward it to the receiver
- The proxy is the only computer known to the outer world
- An access control could be done basing on user identity, used protocol, and content
- More possibilities (Logging of detailed information, authentication, ...), but for each application protocol (HTTP, SMTP, FTP, …) an own proxy is needed

# Packet Filter

Two possible principles:

- Everything that is not explicitly allowed, is denied
- Everything that is not explicitly denied, is allowed
- E.g. for your SMTP server with address 137.226.12.67 on port 25 you could define

```
From (IP * ), (port *)                  To (IP 137.226.12.67), (port 25)  DENY
From (IP 137.226.12.67), (port 25)      To (IP *), (port *)               ALLOW
```

  (I.e.: your mail server can send mails to everybody, but nobody is allowed to send mails to your mail server)
- In the order of their entry, all rules are applied till a matching one is found

*Characteristics*:

- Fast processing of packets, but only limited control on address level
- **Static packet filter** only has a fixed set of such rules
- **Dynamic packet filter** also considers a *state*:
  - Deny all packets from outer world
  - Only after a connection establishment from inside (set SYN flag), response packets coming from outside are accepted

# Proxy Server

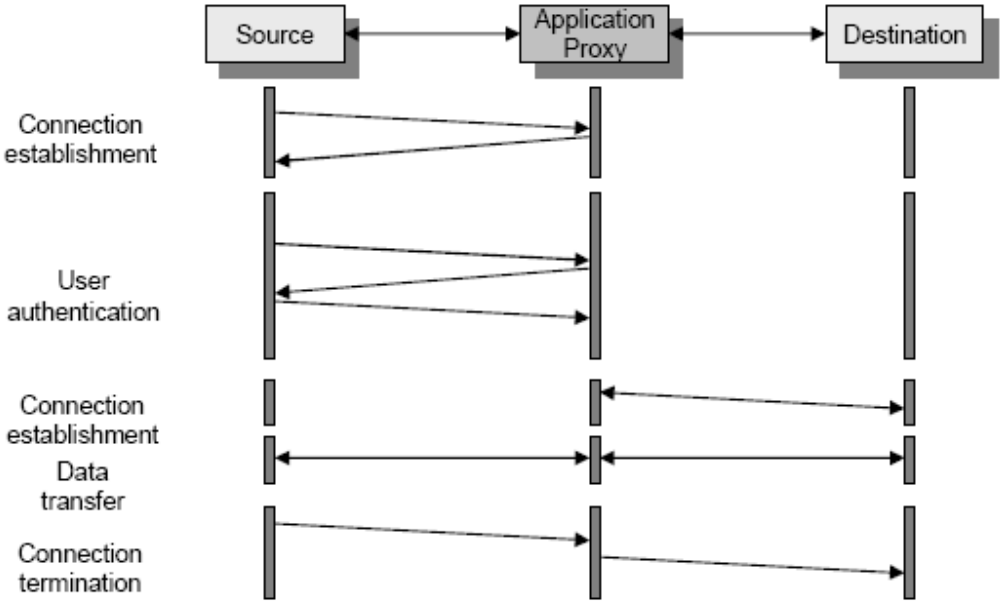
Again two possible types:

## Circuit-Level Proxy

- Works on layer 3/4 only (e.g. port numbers)
- Proxy which can be used for each type of application
- The firewall intercepts all connections, thus the network structure is hidden

## Application-Level Proxy

- Also checks information on layer 7
- An own proxy is needed for each application protocol (SMTP, FTP, HTTP, ...)
- A user maybe has to authenticate before usage
- Most possibilities, but most expensive



# Packet Filter vs Proxy Server

### Packet Filter

+ Simple
+ Low cost implementation
- Correctly specifying packet filters is a difficult and error-prone process
- Reordering packet filter rules makes specifying rules correctly even more difficult

### Proxy Server

+ User authentication is possible
+ Application protocol control (e.g. virus detection) can be integrated
+ Logging of detailed information
+ Accounting
- Proxy needed for each application protocol (expensive)
- Circuit level proxies are cheaper than application level proxies, but not able to scan application data

# Security Architectures

Question: which firewall to install? Where and how to implement it due to the security requirements?

- Personal Firewall
- Dual-Homed Host Firewall
- Screened Hosts Firewall
- Screened Subnet Firewall (Demilitarized Zone)
- Honeypot
- …

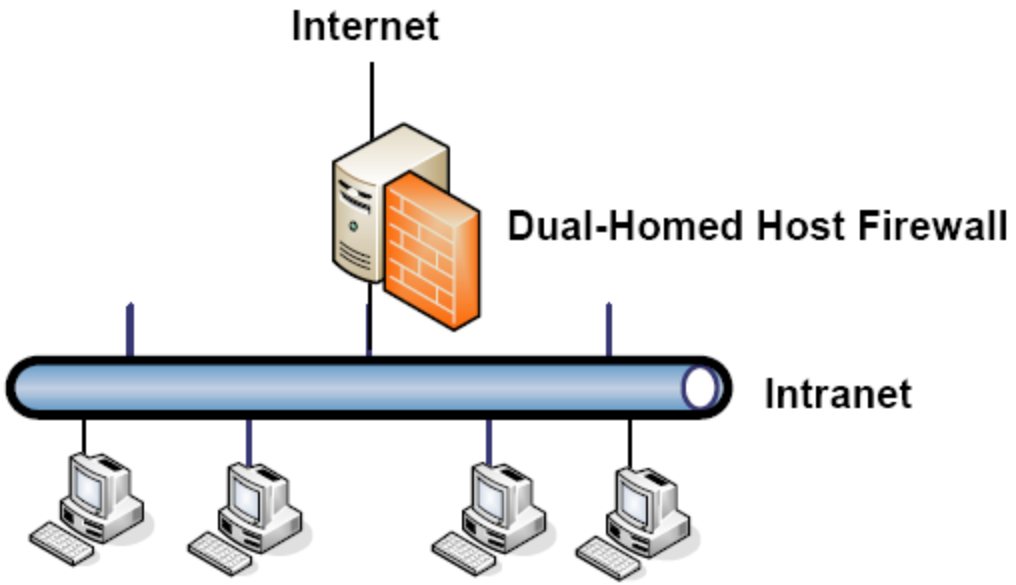
*Personal firewall*

- Not an own component, but a software installed on a host to protect exactly this host
- Part of operating systems to protect a user's machine at home
- Learning filter which can interact with the user to define filtering rules
- Normally not necessary because even at home the usual DSL router today has an intergrated firewall

# Dual-Homed Host Firewall

Simplest implementation: realize packet filter or proxy server as an own machine:

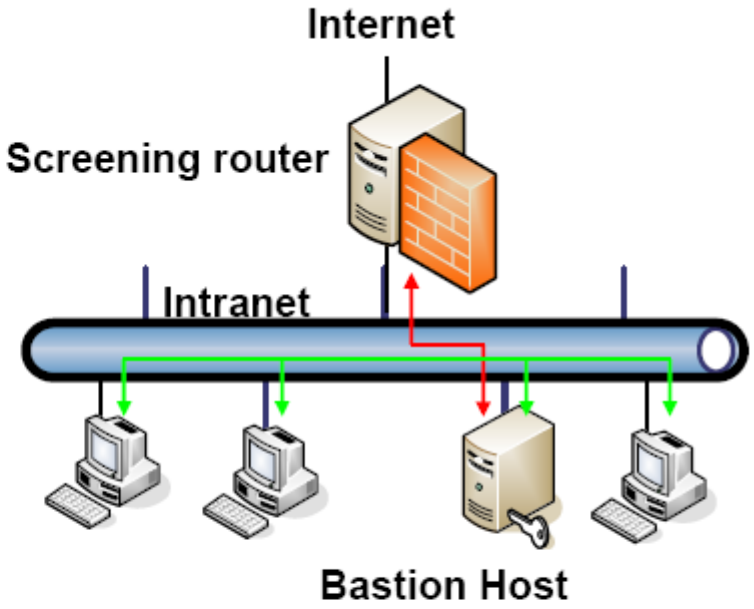
- Machine with two network interfaces
- Routes packets and processes them according to its security rules
- “All-in-one” firewall: can provide packet filter and proxy server
- Clients in the internal network can access services on the Internet either by using a proxy server in the firewall or by logging on to the firewall directly



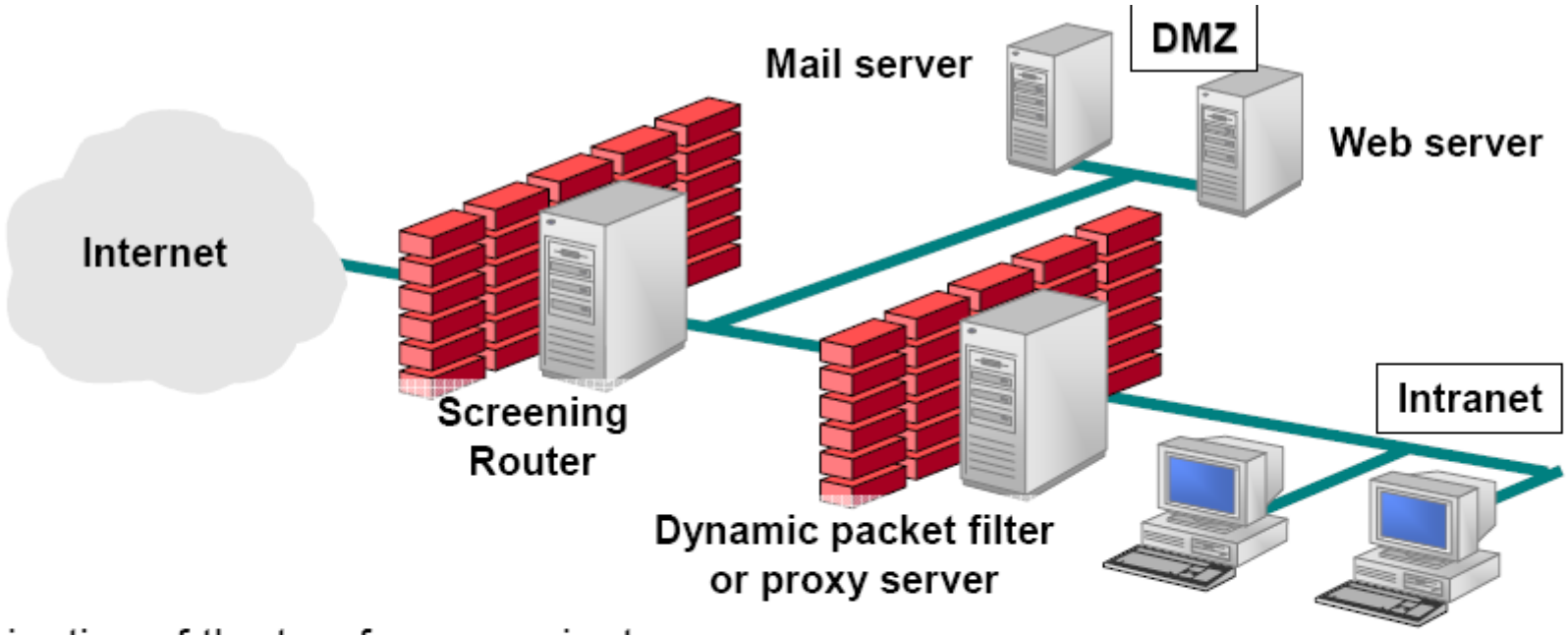
# Screened Hosts Firewall

Introduce another special machine:

- Consists of a screening router and a bastion host on the internal network
- *Bastion host*: a single machine which provides all publicly accessible servers (e.g. in principle a less protected machine because we need to allow accesses to it)
- *Screening router* performs packet filtering of incoming Internet traffic
- Screening router sends all permitted incoming traffic to the bastion host, where further access control decision can be made before packets are forwarded to other hosts
- Screening router accepts internal packets only from the bastion host



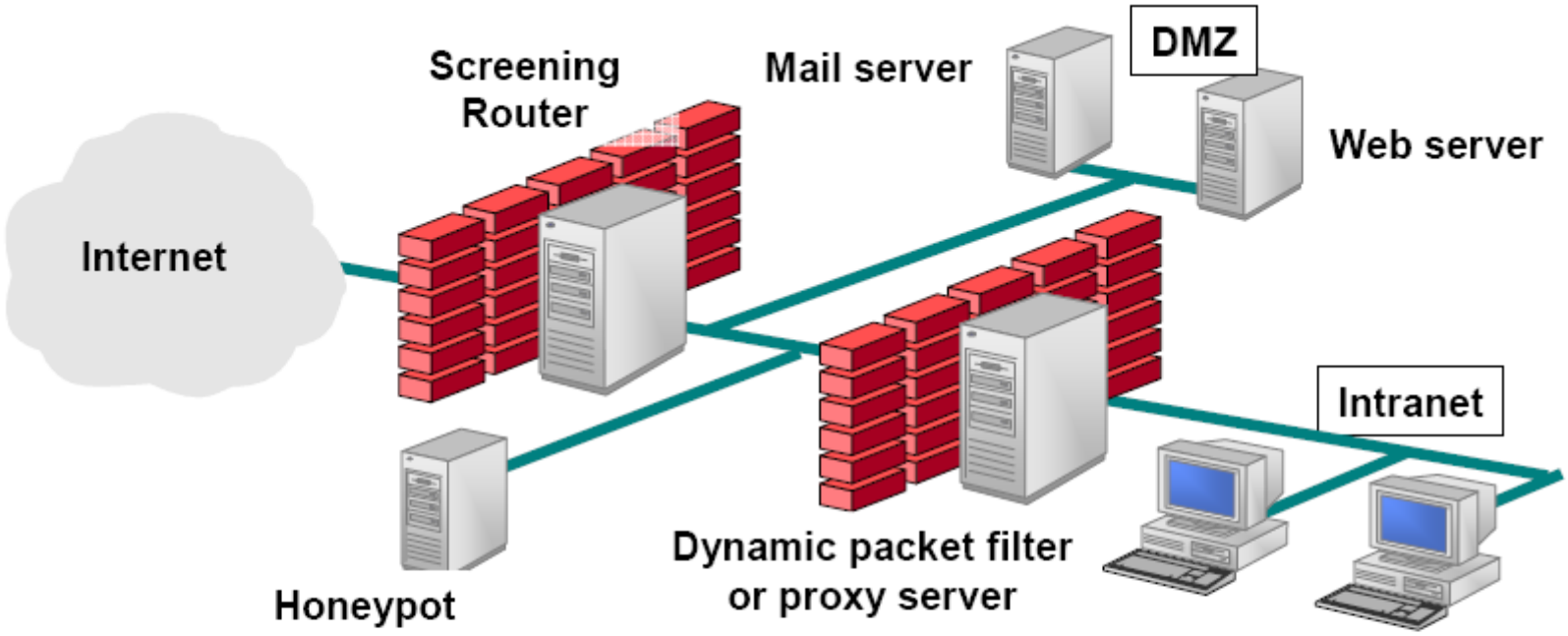
# Demilitarized Zone




Combination of the two former variants:

- All resources which have to be contacted from outside (without restrictions) are placed in an own network segment (*DMZ – Demilitarized Zone*) instead on a bastion host
- This segment is protected against the Internet only by a simple firewall (usually a screening router for packet filtering of uncritical systems, e.g. web server)
- The private network is protected by a more powerful firewall (dynamic packet filter and/or application-level proxy)

# Additional: Honeypot




- Although possible: provide a weak faked server in your DMZ to attract attackers
- The honeypot does heavy logging and provides alarm systems instead of the real application services
- Goal: get knowledge about the attackers

# Intrusion Detection

# Intrusion Detection

Firewalls…

- do not protect against internal attacks
- do not protect against errors in software
- do not protect against configuration errors
- do not protect against errors of external servers
- do not protect against connection hijacking
- can be eluded

→ **Intrusion Detection** to deal with these problems

Additionally to a firewall, let run an *Intrusion Detection System* (IDS) in your network to detect against attacks

Needed:

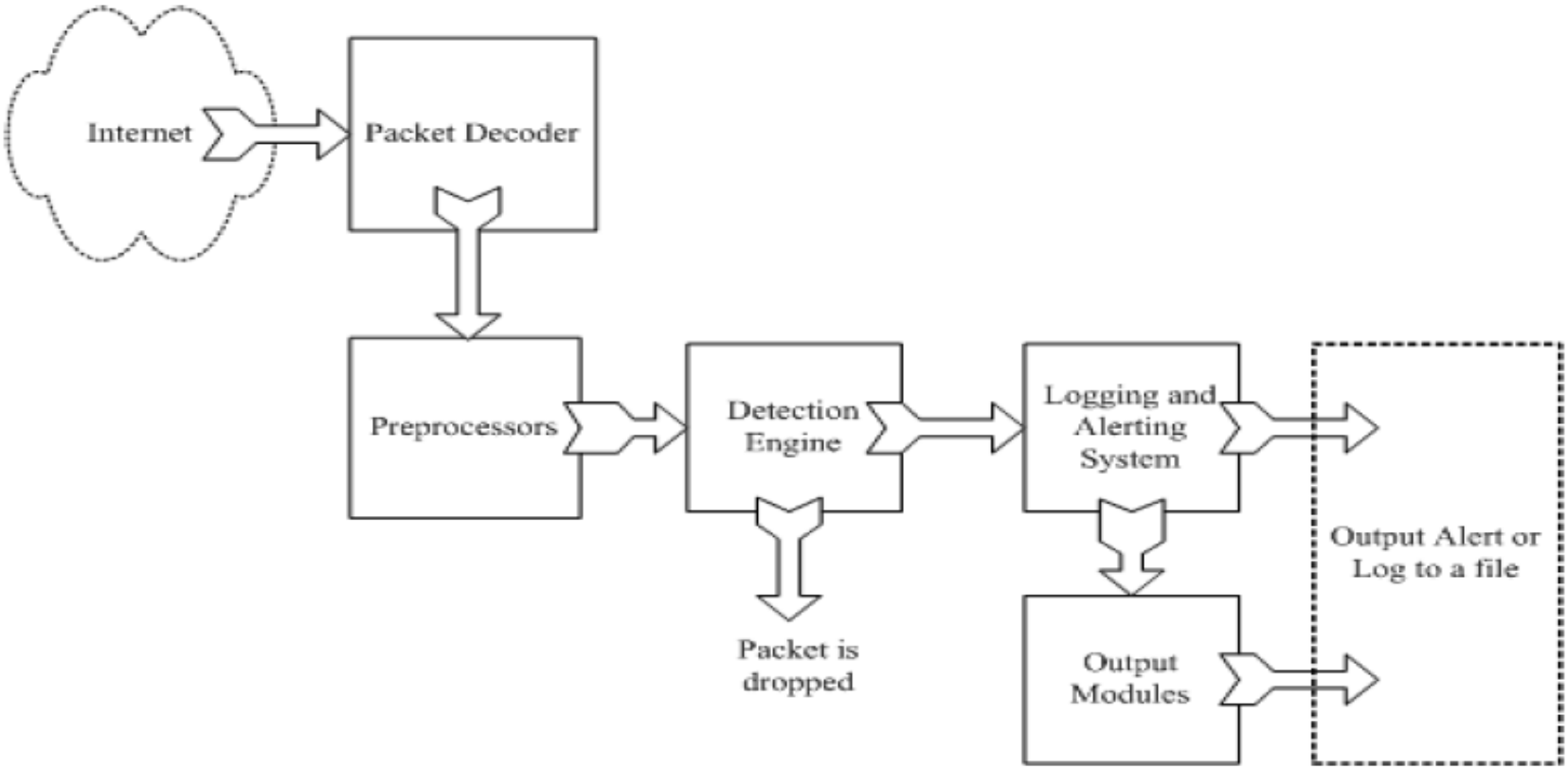
- Monitoring of the network traffic and generate events if something happens (i.e. constantly process a network audit)
- Processing of events, generating alarms
- Defining actions to be taken in presence of certain alarms

# Intrusion detection

- Many intrusion detection systems
  - Close to 100 systems with current web pages
  - Network-based, host-based, or combination
- Two basic models
  - Misuse detection model
    - Maintain data on known attacks
    - Look for activity with corresponding signatures
  - Anomaly detection model
    - Try to figure out what is “normal”
    - Report anomalous behavior
- Fundamental problem: too many false alarms

# Example: Snort

http://www.snort.org/

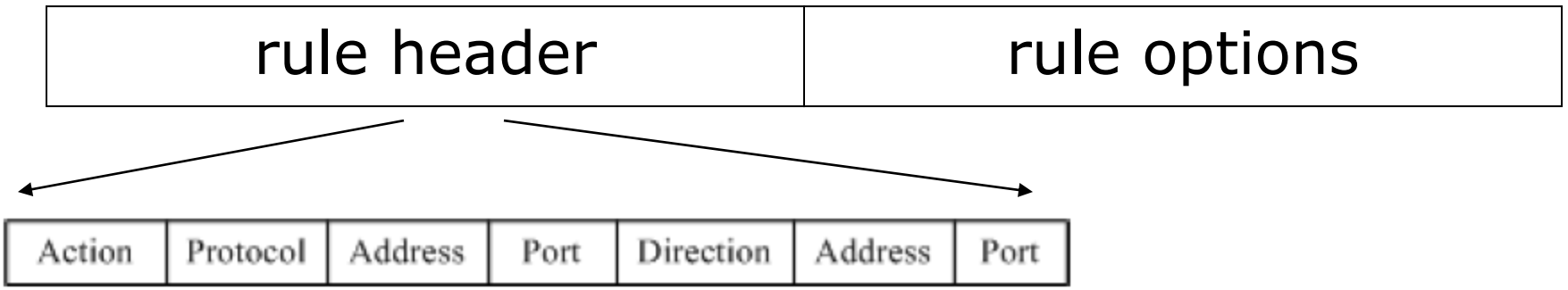


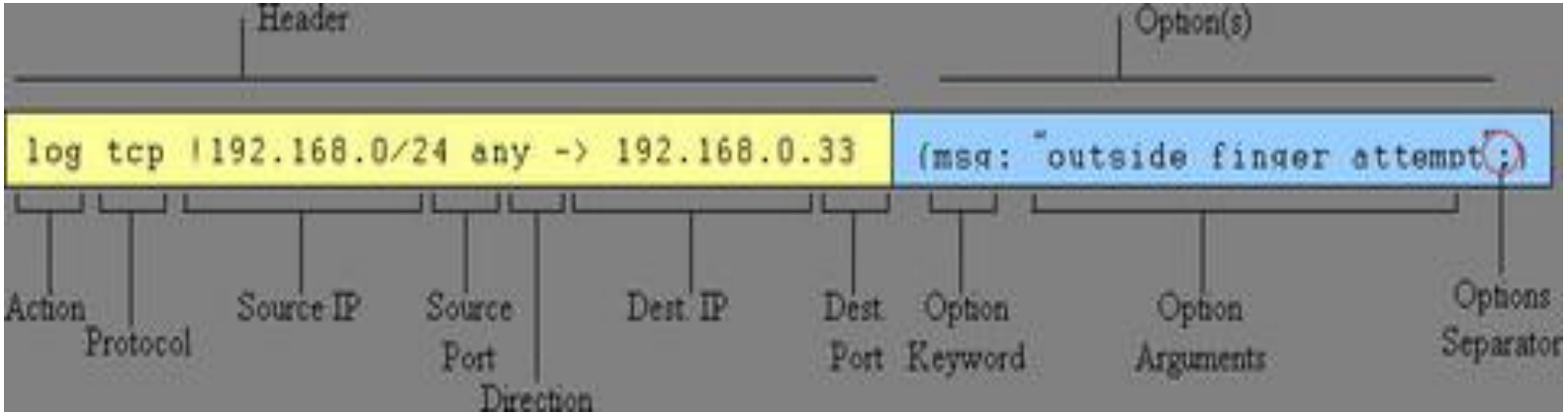

From: Rafeeq Ur Rehman, *Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID.*

# Snort components

- Packet Decoder
  - input from Ethernet, SLIP, PPP…
- Preprocessor:
  - detect anomalies in packet headers
  - packet defragmentation
  - decode HTTP URI
  - reassemble TCP streams
- Detection Engine: applies rules to packets
- Logging and Alerting System
- Output Modules: alerts, log, other output

# Snort detection rules



# Snort challenges

- Misuse detection – avoid known intrusions
  - Database size continues to grow
    - Snort version 2.3.2 had 2,600 rules
  - Snort spends 80% of time doing string match

- Anomaly detection – identify new attacks
  - Probability of detection is low

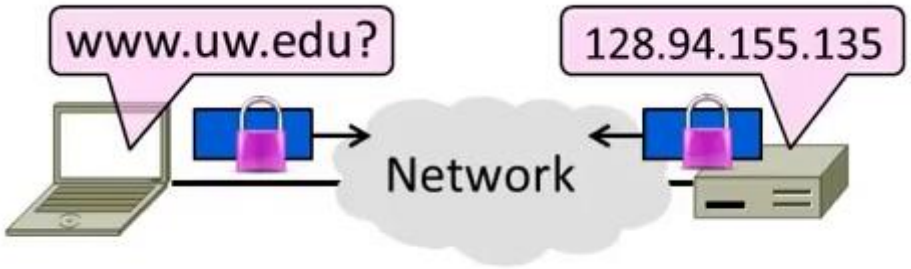
# Difficulties in anomaly detection

- Lack of training data
  - Lots of “normal” network, system call data
  - Little data containing realistic attacks, anomalies
- Data drift
  - Statistical methods detect changes in behavior
  - Attacker can attack gradually and incrementally
- Main characteristics not well understood
  - By many measures, attack may be within bounds of “normal” range of activities
- False identifications are very costly
  - Sys Admin spend many hours examining evidence

# DNSSEC

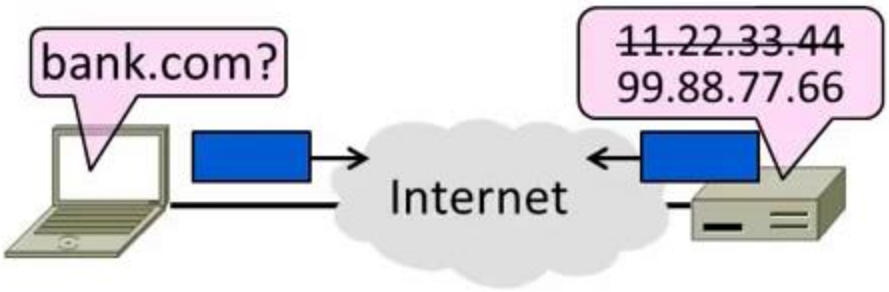
# Topic

- Securing Internet naming
  - DNS security extensions (DNSSEC)



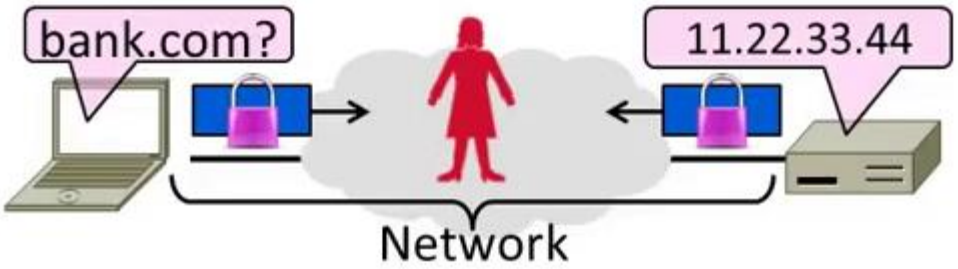
# Goal and Threat Model

- Naming is a crucial Internet service
  - Binds host name to IP address
  - Wrong binding can be disastrous …



# Goal and Threat Model (2)

- Goal is to secure the DNS so that the returned binding is correct
  - Integrity/authenticity vs confidentiality
- Attacker (Trudy) can intercept/tamper with messages on the network

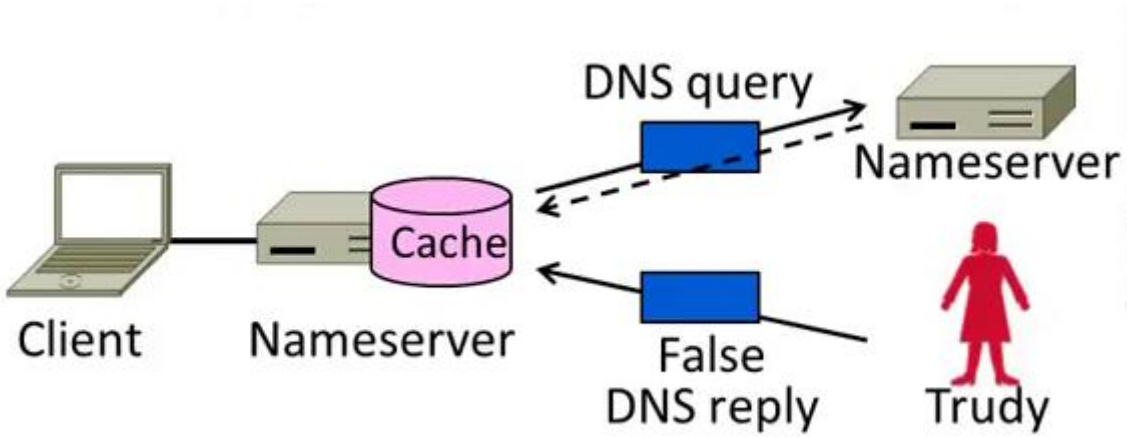


# DNS Spoofing

- Hang on – how can a network attacker corrupt the DNS?
- Trudy can trick a nameserver into caching the wrong binding
  - By using the DNS protocol itself
  - This is called DNS spoofing

# DNS Spoofing (2)

- To spoof, Trudy returns a fake DNS response that appears to be true
  - Fake response contains bad binding



# DNS Spoofing (3)

- Lots of questions!
  1. How does Trudy know when the DNS query is sent and what it is for?
  2. How can Trudy supply a fake DNS reply that appears to be real?
  3. What happens when the real DNS reply shows up?

- There are solutions to each issue …

# DNS Spoofing (4)

1. How does Trudy know when the query is sent and what it is for?

- Trudy can make the query herself!
  - Nameserver works for many clients
  - Trudy is just another client

# DNS Spoofing (5)

2. How can Trudy supply a fake DNS reply that appears to be real?

- A bit more difficult. DNS checks:
  - Reply is from authoritative nameserver (e.g., .com)
  - Reply ID that matches the request
  - Reply is for outstanding query

- • (Nothing about content though …)

# DNS Spoofing (6)

2. How can Trudy supply a fake DNS reply that appears to be real?

- Techniques:
  - Put IP of authoritative nameserver as the source IP address
  - ID is 16 bits (64K). Send many guesses! (Or if a counter, sample to predict.)
  - Send reply right after query

- Good chance of succeeding!

# DNS Spoofing (7)

3. What happens when the real DNS reply shows up?

- Likely not be a problem
  - There is no outstanding query after fake reply is accepted
  - So real reply will be discarded

# DNSSEC (DNS Security Extensions)

- Extends DNS with new record types
  - RRSIG for digital signatures of records
  - DNSKEY for public keys for validation
  - DS for public keys for delegation
  - First version in ‘97, revised by ’05
- Deployment requires software upgrade at both client and server
  - Root servers upgraded in 2010
  - Followed by uptick in deployment

# DNSSEC (2) – New Records

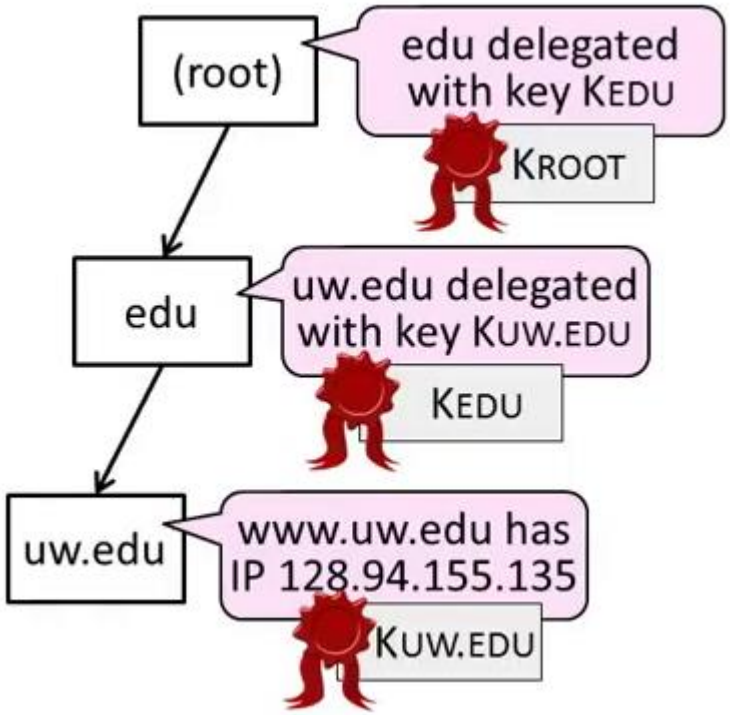
- As well as the usual A, NS records
- RRSIG
  - Digital signatures of domain records
- DNSKEY
  - Public key used for domain RRSIGs (for validation of signatures)
- DS
  - Public keys for delegated domain
- NSEC/NSEC3
  - Authenticated denial of existence (answer from an authoritative NS that really there is no domain)

# DNSSEC (3) – Validating Replies

- Clients query DNS as usual, then validate replies to check that content is authentic
- Trust anchor is root public keys
  - Part of DNS client configuration
- Trust proceeds down DNS hierarchy
  - Similar concept to SSL certificates

# DNSSEC (4) – Validating Replies

- Client queries www.uw.edu as usual
  - Replies include signatures/keys
- Client validates answer:
  1. KROOT is a trust anchor
  2. Use KROOT to check KEDU
  3. Use KEDU to check KUW.EDU
  4. Use KUW.EDU to check IP



# DNSSEC (5)

- Other features too:
  - Authoritative answers a domain record doesn't exist (NSEC/NSEC3)
  - Optional anti-spoofing to bind query and reply
  - Flags related to deployment …

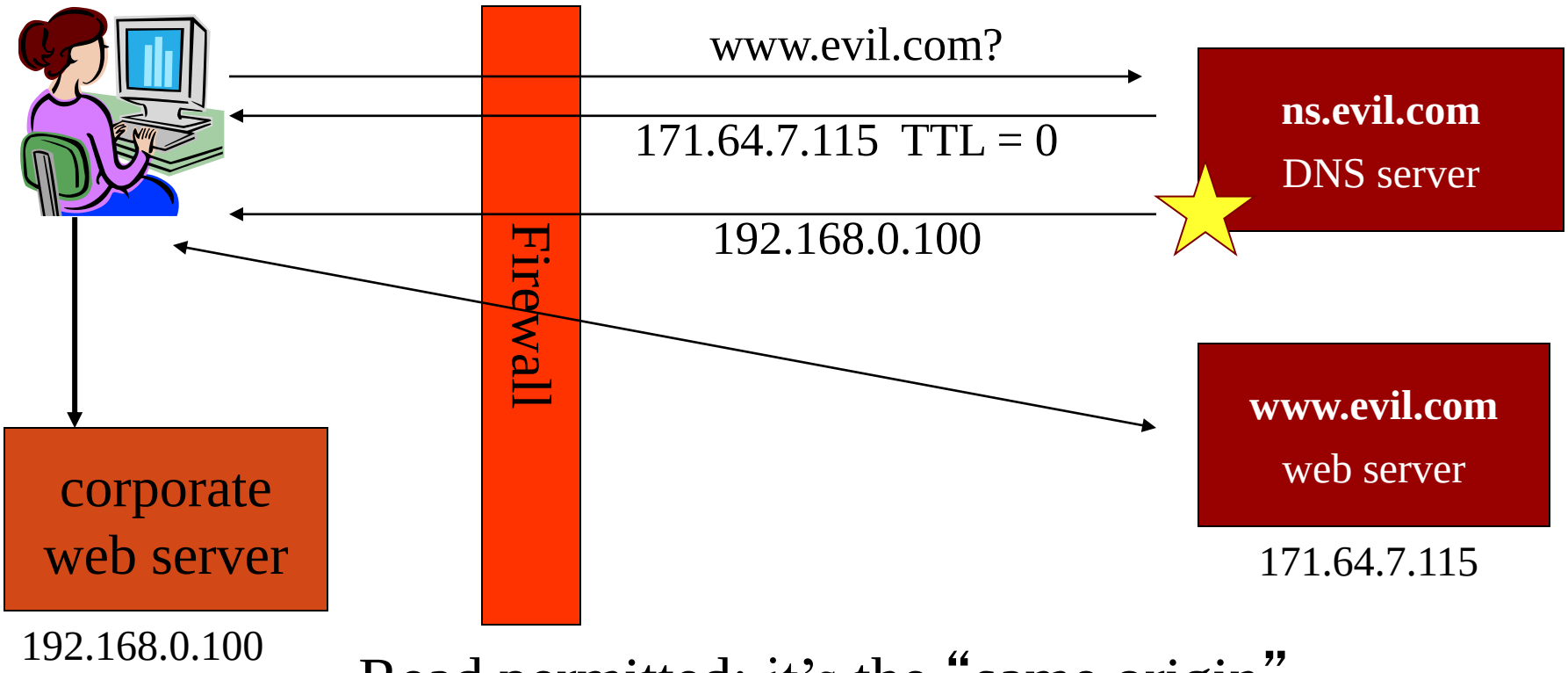
# Summary

- DNS spoofing is possible without added security measures
  - Large problem in practice!
- DNSSEC adds authentication (only) of replies to the DNS
  - Using a hierarchy of public keys

# DNS Rebinding Attack

<iframe src="**http://www.evil.com**">

DNS-SEC cannot stop this attack




Read permitted: it's the "same origin"



[DWF' 96, R' 01]

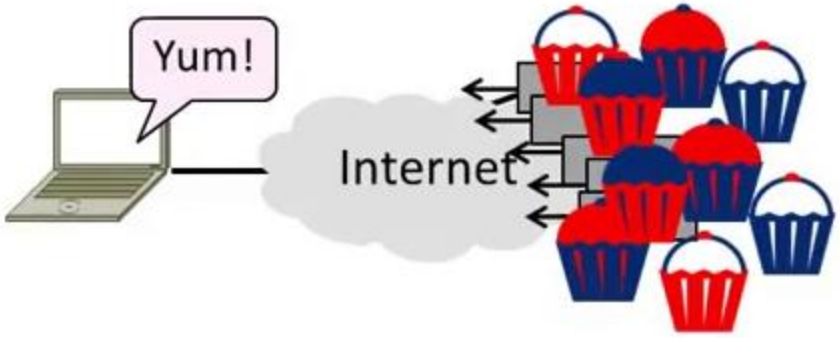
# DNS Rebinding Defenses

- Browser mitigation: DNS Pinning
  - Refuse to switch to a new IP
  - Interacts poorly with proxies, VPN, dynamic DNS, …
  - Not consistently implemented in any browser
- Server-side defenses
  - Check Host header for unrecognized domains
  - Authenticate users with something other than IP
- Firewall defenses
  - External names can't resolve to internal addresses
  - Protects browsers inside the organization

# DDoS

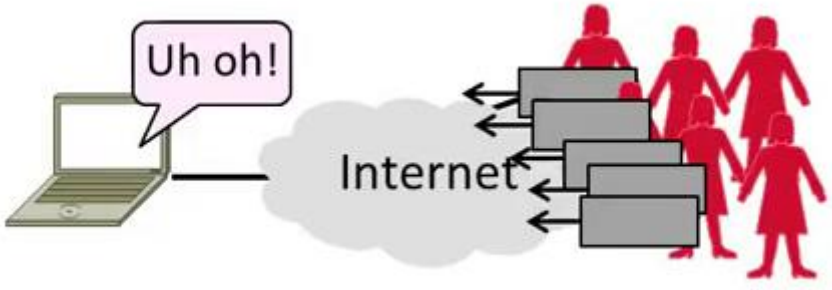
# Topic

- Distributed Denial-of-Service (DDOS)
  - An attack on network availability



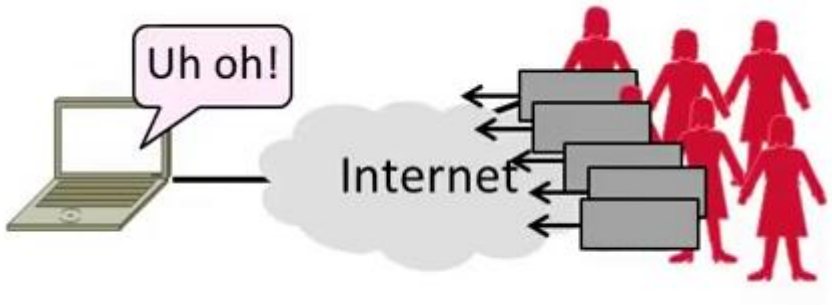
# Topic

- Distributed Denial-of-Service (DDOS)
  - An attack on network availability



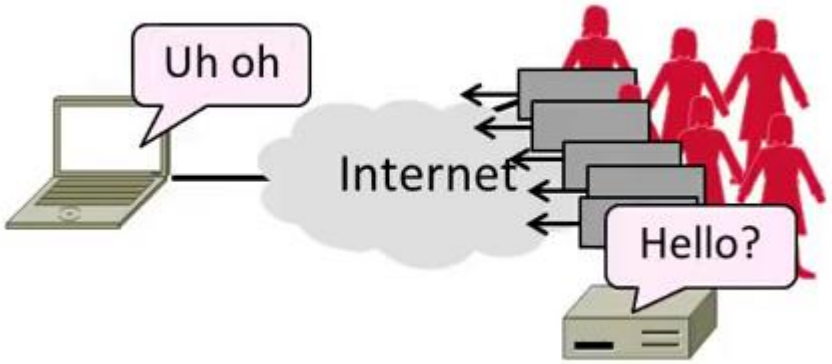
# Motivation

- The best part of IP connectivity
  - You can send to any other host
- The worst part of IP connectivity
  - Any host can send packets to you!



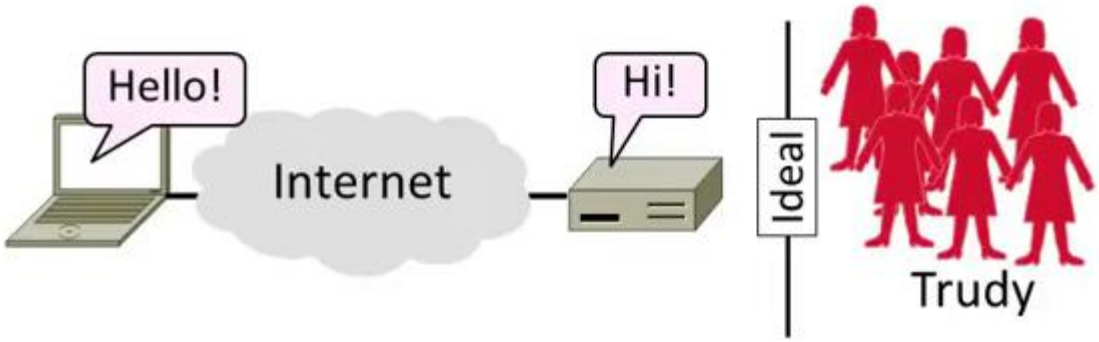
# Motivation (2)

- Flooding a host with many packets can interfere with its IP connectivity
  - Host may become unresponsive
  - This is a form of denial-of-service



# Goal and Threat Model

- Goal is for host to keep network connectivity for desired services
  - Threat is Trudy may overwhelm host with undesired traffic



# Internet Reality

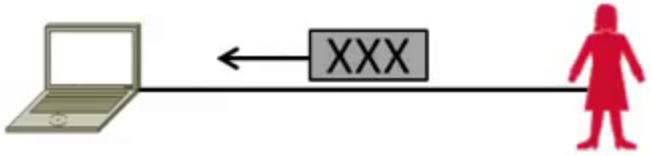
- Distributed Denial-of-Service is a huge problem today!
  - Akamai Q3-12 reports DDOS against US banks peaking at 65 Gbps of traffic flooding the bank
- There are no great solutions
  - CDNs, network traffic filtering, and best practices all help

# Denial-of-Service

- Denial-of-service means a system is made unavailable to intended users
  - Typically because its resources are consumed by attackers instead
- In the network context:
  - “System” means server
  - “Resources” mean bandwidth (network) or CPU/memory (host)

# Host Denial-of-Service

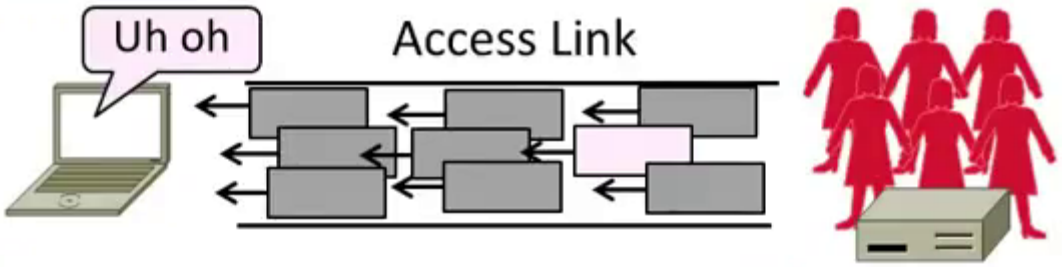
- Strange packets can sap host resources!
  - "Ping of Death" malformed packet (bug the kernel and system crash)
  - "SYN flood" sends many TCP connect requests and never follows up
  - Few bad packets can overwhelm host




- Patches exist for these vulnerabilities
  - Read about "SYN cookies" for interest

# Network Denial-of-Service

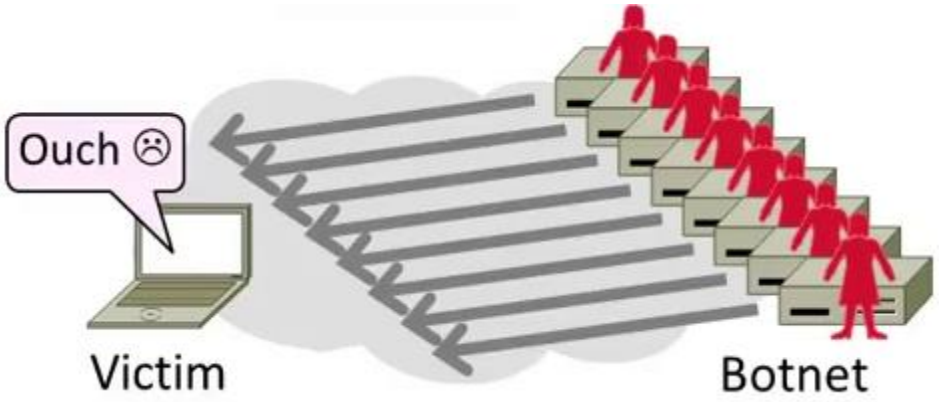
- Network DOS needs many packets
  - To saturate network links
  - Causes high congestion/loss




- Helpful to have many attackers or Distributed Denial-of-Service

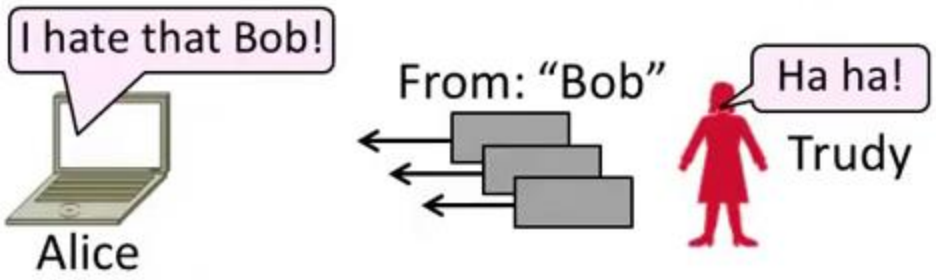
# Distributed Denial-of-Service (DDoS)

- Botnet provides many attackers in the form of compromised hosts
  - Hosts send traffic flood to victim
  - Network saturates near victim



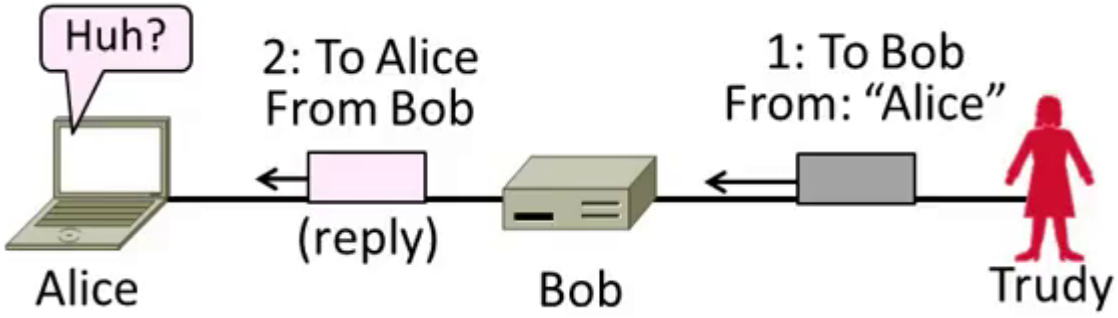
# Complication: Spoofing

- Attackers can falsify their IP address
  - Put fake source address on packets
  - Historically network doesn't check
  - Hides location of the attackers
  - Called IP address spoofing



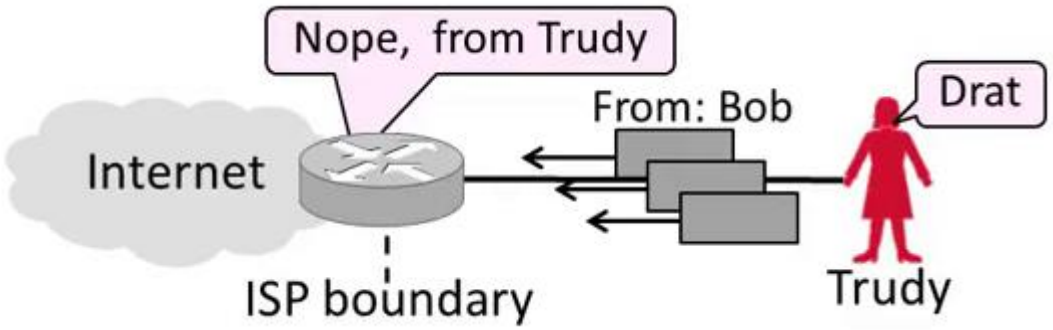
# Spoofing (2)

- Actually, it's worse than that
  - Trudy can trick Bob into really sending packets to Alice
  - To do so, Trudy spoofs Alice to Bob



# Best Practice: Ingress Filtering

- Idea: Validate the IP source address of packets at ISP boundary (Duh!)
  - Ingress filtering is a best practice, but deployment has been slow



# Flooding Defenses

1. Increase network capacity around the server; harder to cause loss
   - Use a CDN for high peak capacity
2. Filter out attack traffic within the network (at routers)
   - The earlier the filtering, the better
   - Ultimately what is needed, but ad hoc measures by ISPs today

# Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Prof. Dan Boneh (Standford)
- Prof. O. Spaniol (RWTH Aachen)
- Prof. David Wetheral (University of Washington)