CNT4406/5412 Network Security

Firewalls and Intrusion Detection Systems

Zhi Wang

Florida State University

Fall 2014

Outline

Firewall

- filtering firewall
- proxy firewall

Outline

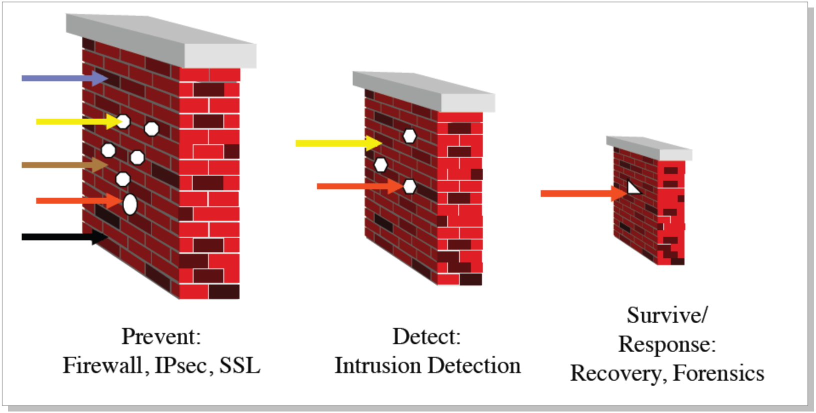
Firewall

- filtering firewall
- proxy firewall

Intrusion Detection System (IDS)

- rule-based IDS
- anomaly detection IDS
- host-based v.s. network-based IDS

Internet Security Mechanisms



Goal: prevent if possible; detect quickly otherwise; and always have a backup plan to confine the damage

Basic Terms

- Vulnerabilities
- Intrusions (attacks) and intrusion detection systems (IDS)
- Alert or alarm: warnings generated by IDS

Example Attacks

- Disclosure, modification, and destruction of data
- Compromise a host and use it as a stepping stone to attack others
- Monitor and capture user passwords, then impersonate the user

Introduction

Firewall:any barrier intended to thwart the spread of a destructive agent

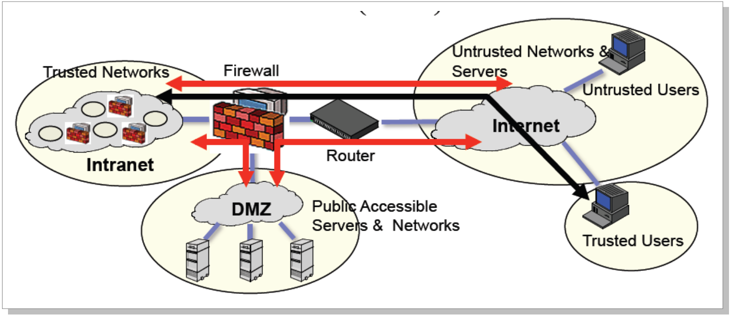
- Provides secure connection between networks
- Enforce security policies for network communications

Introduction...

- Many organizations have distinct needs
  - ➠ public data (e.g., website) accessible to anyone
  - ➠ internal data only accessible to employees

Introduction...

- Many organizations have distinct needs
  - ➠ public data (e.g., website) accessible to anyone
  - ➠ internal data only accessible to employees
- Solution: inner and out (DMZ) networks



Firewall Capabilities

- Control access
  - ➠ restrict incoming and outgoing traffic according to security policies

Firewall Capabilities

- Control access
  ➠ restrict incoming and outgoing traffic according to security policies
- Log traffics (for later analysis)

Firewall Capabilities

- Control access
  ➠ restrict incoming and outgoing traffic according to security policies
- Log traffics (for later analysis)
- Network address translation (NAT)

Firewall Capabilities

- Control access
  ➠ restrict incoming and outgoing traffic according to security policies
- Log traffics (for later analysis)
- Network address translation (NAT)
- Encryption/decryption

Firewall Limitations

- Network perimeter no longer clear or exists!
  - ➠ unprotected ingress points: notebook and other mobile devices

Firewall Limitations

- Network perimeter no longer clear or exists!
  ➠ unprotected ingress points: notebook and other mobile devices
- "Insider" attacks can bypass the firewall

Firewall Limitations

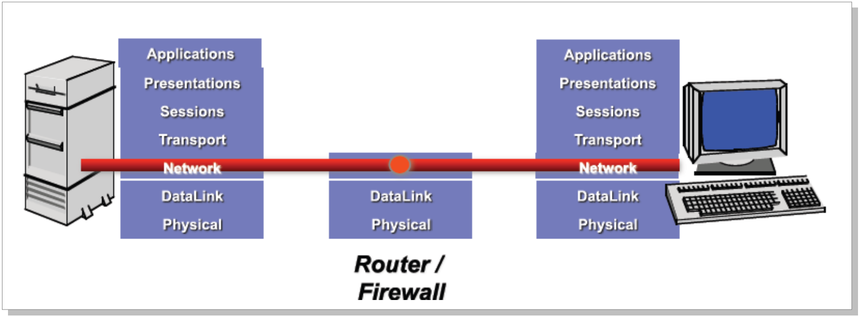
- Network perimeter no longer clear or exists!
  ➠ unprotected ingress points: notebook and other mobile devices
- "Insider" attacks can bypass the firewall
- Converting high-level security policy to firewall rules is non-trivial

Filtering

- Compare traffic to patterns (traffic selector), then apply the action of the first matched rule
- Two styles: packet filtering (stateless) and session filtering (stateful)

Packet Filtering

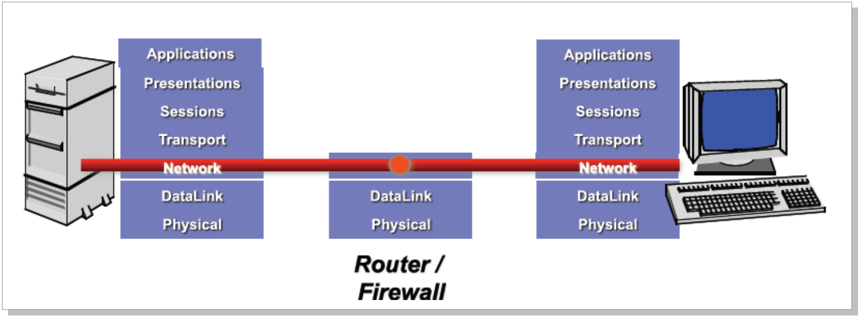
Patterns select packets by matching header field values of a single packet



Packet Filtering

Patterns select packets by matching header field values of a single packet

- e.g., source IP address and source port number
- destination IP address and destination port number
- transport protocol id



Packet Filtering...

- Decisions are made on a per-packet basis
  ➠ no state information about previous packets is maintained

Packet Filtering...

- Decisions are made on a per-packet basis
  - ➠ no state information about previous packets is maintained
  - ➠ e.g., how to handle fragmented packets?

Packet Filtering...

- Decisions are made on a per-packet basis
  ➠ no state information about previous packets is maintained
  ➠ e.g., how to handle fragmented packets?
  ➠ tiny-fragment attack: fragment the packet so most of the TCP header in a second fragment

Packet Filtering...

- Decisions are made on a per-packet basis
  ➠ no state information about previous packets is maintained
  ➠ e.g., how to handle fragmented packets?
  ➠ tiny-fragment attack: fragment the packet so most of the TCP header in a second fragment
- Easy to implement but having limited capabilities

Session Filtering

- Decisions are made in the context of connections (flows)

Session Filtering

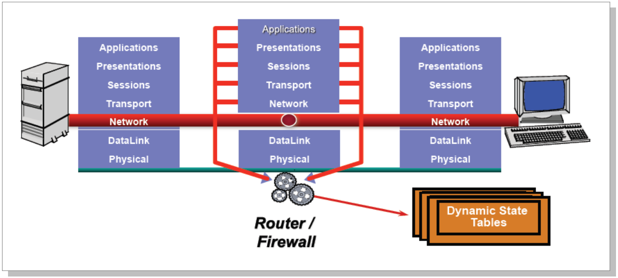
- Decisions are made in the context of connections (flows)
  - ➠ if packet starts a new connection: check rules for new connections

Session Filtering

- Decisions are made in the context of connections (flows)
  ➠ if packet starts a new connection: check rules for new connections
  ➠ if packet is part of an existing connection: check rules for the existing connection, and then update the state of the connection

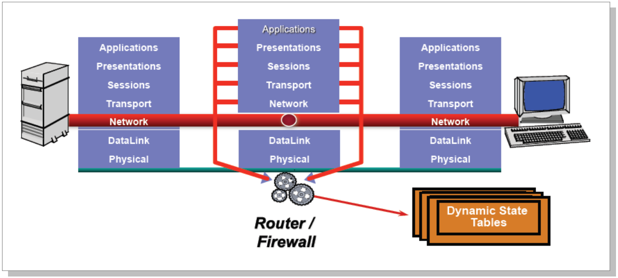
Session Filtering...

- More powerful than packet filtering
  - ➠ can recognize more sophisticated threats
  - ➠ can implement more complex policies (examples?)

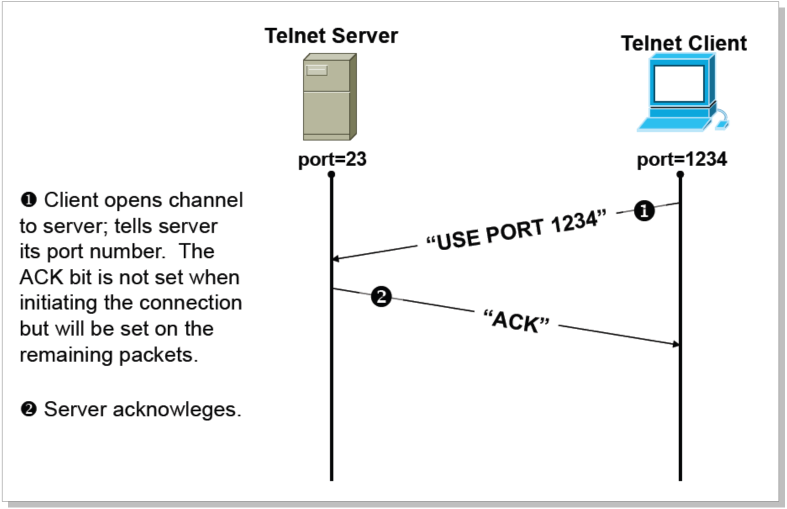


Session Filtering...

- More powerful than packet filtering
  - ➠ can recognize more sophisticated threats
  - ➠ can implement more complex policies (examples?)
- More complicated to implement



Application: Telnet



PF Rules for Telnet

Format:

access-list <rule number> <permit|deny> <protocol> <SOURCE ip address|any|IP address and mask> [<gt|eq port number>] <DEST ip address|any|IP address and mask> [<gt|eq port number>]

PF Rules for Telnet

Format:

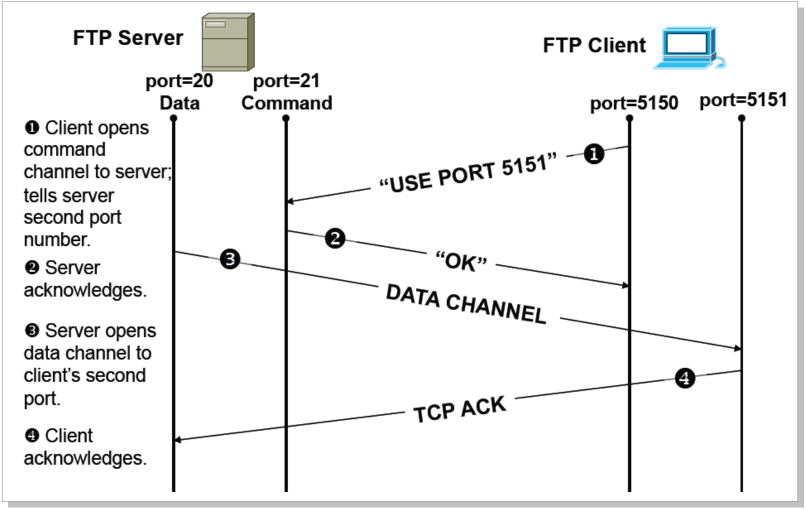
access-list <rule number> <permit|deny> <protocol> <SOURCE ip address|any|IP address and mask> [<gt|eq port number>] <DEST ip address|any|IP address and mask> [<gt|eq port number>]

The following rules allow user to telnet from 172.168.10.11 to any destination, but not vice-versa

```
access-list 100 permit tcp host 172.168.10.11 gt 1023 any eq 23
 ! Allows packets out to remote Telnet servers
access-list 101 permit tcp any eq 23 host 172.168.10.11 established
 ! Allows returning packets to come back in.  It verifies that the ACK bit is set

interface Ethernet 0
 access-list 100 out   ! Apply the first rule to outbound traffic
 access-list 101 in    ! Apply the second rule to inbound traffic
```

Application: FTP



PF Rules for FTP

The following rules allow user to FTP (not passive FTP) from any IP to the FTP server (172.168.10.12) (problems?)

```
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 21
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 20
  ! Allows packets from any client to the FTP control and data ports
access-list 101 permit tcp host 172.168.10.12 eq 21 any gt 1023
access-list 101 permit tcp host 172.168.10.12 eq 20 any gt 1023
  ! Allows FTP server to send packets back to any IP address with TCP ports > 
1023

interface Ethernet 0
 access-list 100 in     ! Apply the first rule to inbound traffic
 access-list 101 out    ! Apply the second rule to outbound traffic
```

UFW - Uncomplicated Firewall (Session Filtering)

- UFW is a managing interface for a netfilter firewall (in Ubuntu)

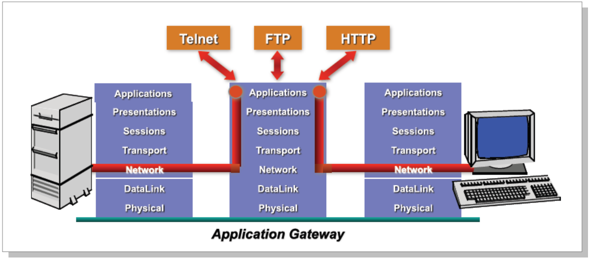
UFW - Uncomplicated Firewall (Session Filtering)

- UFW is a managing interface for a netfilter firewall (in Ubuntu)
- UFW has simple syntax
  - ufw enable|disable|reload
  - ufw default allow|deny|reject
  - ufw allow|deny|reject PORT[/protocol]
  - ufw allow|deny|rejec [proto PROTO] [from ADDR [port PORT]] [to ADDR [port PORT]]

  - e.g., ufw enable; ufw default deny; ufw allow ssh

Proxy Firewalls

- Serve as relays for connections
- Two styles: application level and circuit level



Application Proxy

- Understand specific application protocols, e.g., HTTP, SMTP, Telnet
  ➠ proxy "impersonates" both ends of the connections, like MITM

Application Proxy

- Understand specific application protocols, e.g., HTTP, SMTP, Telnet
  ➠ proxy "impersonates" both ends of the connections, like MITM
- Proxy can arbitrarily process/inspect application payload
  ➠ e.g., check mail for virus before forwarding

Application Proxy

- Understand specific application protocols, e.g., HTTP, SMTP, Telnet
  ➠ proxy "impersonates" both ends of the connections, like MITM
- Proxy can arbitrarily process/inspect application payload
  ➠ e.g., check mail for virus before forwarding
- Must write new proxies to support new application protocols

Application Proxy

- Understand specific application protocols, e.g., HTTP, SMTP, Telnet
  ➠ proxy “impersonates” both ends of the connections, like MITM
- Proxy can arbitrarily process/inspect application payload
  ➠ e.g., check mail for virus before forwarding
- Must write new proxies to support new application protocols
- May require client to be configured to use the proxy

Application Proxy

- Understand specific application protocols, e.g., HTTP, SMTP, Telnet
  ➠ proxy "impersonates" both ends of the connections, like MITM
- Proxy can arbitrarily process/inspect application payload
  ➠ e.g., check mail for virus before forwarding
- Must write new proxies to support new application protocols
- May require client to be configured to use the proxy
- Computationally expensive

Circuit-level Proxy

- Sets up two connections, one to inside user, one to outside server
  - ➠ proxy at the TCP level, rather than application level
  - ➠ client must be aware they are using a circuit-level proxy

Circuit-level Proxy

- Sets up two connections, one to inside user, one to outside server
  - ➠ proxy at the TCP level, rather than application level
  - ➠ client must be aware they are using a circuit-level proxy
- Users must authenticate to proxy before being connected to outside

Circuit-level Proxy

- Sets up two connections, one to inside user, one to outside server
  - ➠ proxy at the TCP level, rather than application level
  - ➠ client must be aware they are using a circuit-level proxy
- Users must authenticate to proxy before being connected to outside
- Example protocol: SOCKS

Attack Stages

- **Intelligence gathering:** probe the system to determine vulnerabilities
- **Planning:** decide what resources to attack and how
- **Attack execution:** launch the attack
- **Hiding:** cover traces of the attack
- **Maintaining access:** install "backdoors" for feature access

IDS

- Detect if attacks are being attempted or if system has been compromised

IDS

- Detect if attacks are being attempted or if system has been compromised
- IDS should be:

IDS

- Detect if attacks are being attempted or if system has been compromised
- IDS should be:
  ➠ accurate, fast, flexible, easy to understand and manager

Measuring Accuracy

- Events are actions occurring in the system (e.g., file access, login, etc)
- An intrusion is an event that is a part of an attack
- An alarm is generated if an event is diagnosed as being an intrusion

	intrusion	non-intrusion
alarm	**true positive**	**false positive**
no alarm	**false negative**	**true negative**

Measuring Accuracy...

- **True positive rate:** fraction of intrusions correctly detected
- **False negative rate:** fraction of intrusion incorrectly detected
  ➠ FNR = 1 - TPR

Measuring Accuracy...

- **True positive rate:** fraction of intrusions correctly detected
- **False negative rate:** fraction of intrusion incorrectly detected
  ➠ FNR = 1 - TPR
- **True negative rate:** fraction of non-intrusion correctly diagnosed
- **False positive rate:** fraction of non-intrusion incorrectly diagnosed
  ➠ FPR = 1 - TNR

Measuring Accuracy...

- It is trivial to have 100% TPR or 0% FPR
  ➠ how?

Measuring Accuracy...

- It is trivial to have 100% TPR or 0% FPR
  ➠ how?
- Need both...challenging

Example

70,000 events, 300 intrusions, 2,800 alarms of which 298 are correct diagnose, 2,502 are not:

➠ TPR $= \frac{298}{300} = 99.3\%$

➠ FNR = 1 - TPR $= 0.7\%$

Example

70, 000 events, 300 intrusions, 2, 800 alarms of which 298 are correct diagnose, 2, 502 are not:

- ➠ TPR $= \frac{298}{300} = 99.3\%$
- ➠ FNR = 1 - TPR $= 0.7\%$
- ➠ TNR $= \frac{70{,}000-300-2052}{70{,}000-300} = 96.4\%$
- ➠ FPR $= 3.6\%$

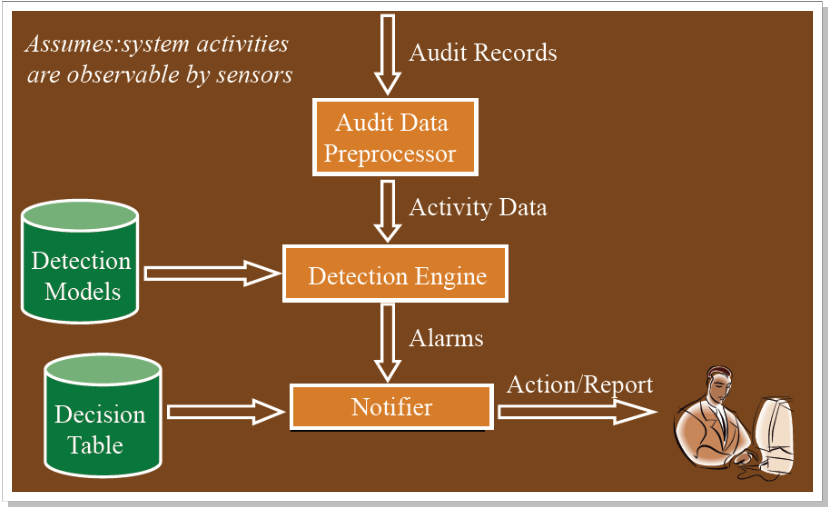
Base-Rate Fallacy

- IDS often suffers from base-rate fallacy
  ➠ intrusions are rare events; non-intrusions are common
  ➠ correctly detected intrusions are swapped by incorrectly detected non-intrusions!

Base-Rate Fallacy

- IDS often suffers from base-rate fallacy
  ➠ intrusions are rare events; non-intrusions are common
  ➠ correctly detected intrusions are swapped by incorrectly detected non-intrusions!
- Previous example: only 298 out of $2,800$ alarms (10.6%) are correct
  ➠ in reality, often less than 1% alarms are real intrusions

Components of IDS



Signature-Based IDS

- Detect attack using signatures
  ➠ characteristics of real attacks, e.g., illegal syscall sequences

Signature-Based IDS

- Detect attack using signatures
  ➠ characteristics of real attacks, e.g., illegal syscall sequences
- Only detect already-known attacks

Signature-Based IDS

- Detect attack using signatures
  ➠ characteristics of real attacks, e.g., illegal syscall sequences
- Only detect already-known attacks
- FPR is low, but FNR is high

Anomaly Detection

- Define a model of "normal" behavior, try to detect deviation from it
- Potentially detect new (not previously-encountered) attacks
- FNR is low, FPR is high

Anomaly Detection

- Define a model of "normal" behavior, try to detect deviation from it
- Potentially detect new (not previously-encountered) attacks
- FNR is low, FPR is high

- Which is better?

Anomaly Detection

- Define a model of “normal” behavior, try to detect deviation from it
- Potentially detect new (not previously-encountered) attacks
- FNR is low, FPR is high

- Which is better? ➠ maybe a combination

Signature v.s Anomaly Detection

- Password file modified
- Four failed login attempts
- Failed connection attempts on 50 sequentially-numbered ports
- User who usually logins around 10am from dorm logins at 4:30 from an IP address in Transylvania
- UDP packet to port 1434 (Slammer Worm)

Example Signatures

- A sequence of connection attempts to a large number of ports
- A network packet that has lots of NOOP instructions in it
- Program input containing a very long string
- A large number of TCP SYN packets sent, with no ACKs

Signature Generation

- Goal: fast, automatic extraction of signatures for new attacks

Signature Generation

- Goal: fast, automatic extraction of signatures for new attacks
- Attack signatures are usually attack-specific
  ➠ how to automatically generate valid variants of the signature

Signature Generation

- Goal: fast, automatic extraction of signatures for new attacks
- Attack signatures are usually attack-specific
  ➠ how to automatically generate valid variants of the signature
- Program obfuscation
  ➠ find signature that are difficult to conceal/obfuscate

Anomaly Detection

- Define a profile of "normal" behavior (the training phase)
  ➠ work best for small, deterministic systems
- IDS compares operational system to this profile and flags deviations

Example Metrics

- Frequency of an event ➠ alert if too high
- Time between events ➠ alert if too small
- Resource utilization ➠ alert if too high
- Statistical measures (mean, standard deviation etc)

Example Metrics...

- Markov process: expected likelihood of transition from one system state to another, or from one output to another
- Short sequences of events
  ➠ e.g., system call sequences

Building Profiles

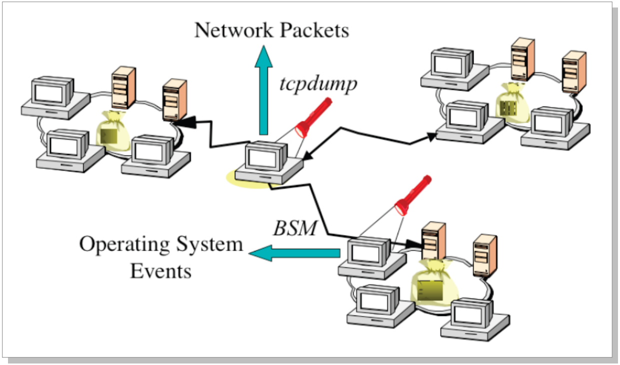
- Profiles are updated regularly to keep up with the current status, older data should be "aged" out
  ➠ e.g., $m_i = \alpha \times most\_recently\_measured\_value + (1 - \alpha) \times m_{i-1}$

Building Profiles

- Profiles are updated regularly to keep up with the current status, older data should be "aged" out
  ➠ e.g., $m_i = \alpha \times most\_recently\_measured\_value + (1-\alpha) \times m_{i-1}$
- Risk: attacker trains IDS to accept his activity as normal
  ➠ training data should be free of attacks, or attacks must be clearly marked as is

Where is IDS Deployed?

- **Host-based** IDS monitors activities on a single host
- **Network-based** IDS monitors traffic (e.g., packet headers)



Host-Based IDS

- Use OS monitoring mechanisms to find compromised applications
  ➠ e.g., file accesses and system calls
- Advantage: better visibility into behavior of individual apps
- Example: virus/rootkit detection

Host-Based IDS: Problems

- Need an IDS for every machine
- May be tampered by the attacker on the same machine
- Only local view of the attack

Example: Tripwire

- Records hashes of critical files and binaries
- Periodically check the file by re-computing and comparing the hash

Example: Tripwire

- Records hashes of critical files and binaries
- Periodically check the file by re-computing and comparing the hash

- Ways to bypass?

Network-Based IDS

- Passively inspect network traffic and monitor traffic pattern
  ➠ protocol violations, unusual connection patterns...
- Advantage: single NIDS can detect many hosts and look for widespread patterns of activity

NIDS: Problems

- may be defeated by encryption
- not all attacks arrive from the network
- must process huge amount of network traffic
  ➠ overload NIDS with huge data streams, then attack

NIDS Example: Snort

- Popular open-source NIDS
- Large ruleset for vulnerabilities (more than 4000)

> "Date: 2005-04-05
> Synopsis: the Sourcefire Vulnerability Research Team (VRT) has learned of serious vulnerabilities affecting various implementations of Telnet […] Programming errors in the telnet client code from various vendors may present an attacker with the opportunity to overflow a fixed length buffer […] Rules to detect attacks against this vulnerability are included in this rule pack"

Snore Ruleset Categories

- Backdoors
- Chat
- DDoS
- Finger
- FTP
- ICMP
- IMAP

Multimedia
MySQL
NETBIOS
NNTP
Oracle
P2P

POP
RPC
Scan
Shellcode
SMTP
SNMP
SQL

Telnet
TFTP
Virus
Web…
X11

Snort Rule Syntax

Each snort rule has a rule header and rule options

- **Rule header:** action, protocol, source (IP/port), direction, dest (IP/port)
- **Rule option:** alert message, info on which parts of packet to be inspected

Snort Rule Examples

- alert icmp $EXTERNAL_NET any <> $HOME_NET any
  (msg:"DDOS Stacheldraht agent->handler (skillz)";
  content:"skillz";
  itype:0;
  icmp_id:6666; reference:url,staff.washington.edu/dittrich/misc/stacheldraht.analysis;
  classtype:attempted-dos;
  sid:1855; rev:2;)
- alert any any -> 192.168.1.0/24 any
  (flags:A; ack:0; msg: “NMAP TCP ping”;)
  # nmap send TCP ACK pkt with ack field set to 0
- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
  (msg:"WEB-IIS cmd.exe access"; flow:to_server,established;
  content:"cmd.exe";
  nocase;
  classtype:web-application-attack;
  sid:1002; rev:5;)

Detect Attack Signatures

- Scanning signatures in individual packets (stateless) is not enough
  ➠ attack can fragment the packet

Detect Attack Signatures

- Scanning signatures in individual packets (stateless) is not enough
  ➠ attack can fragment the packet
- Recording just previous packet is not enough
  ➠ attacker can send packets out-of-order

Detect Attack Signatures

- Scanning signatures in individual packets (stateless) is not enough
  ➠ attack can fragment the packet
- Recording just previous packet is not enough
  ➠ attacker can send packets out-of-order
- Attacker can use TCP tricks so certain packets are seen by NIDS but dropped by the receiving application

Final Exam

- **Schedule**: Monday (Dec 10), 10:00-12:00p.m./Classroom
- **Scope**: cumulative, everything covered in class
  ➠ except for virtualization
- **Style**: close-booked, one-page cheat sheet (U.S. letter paper, single-sided); pen and eraser only, no calculator or smartphone etc
- **Format**: similar to mid-term, concepts + Q&A, but longer