# Firewall&Intrusion detection System

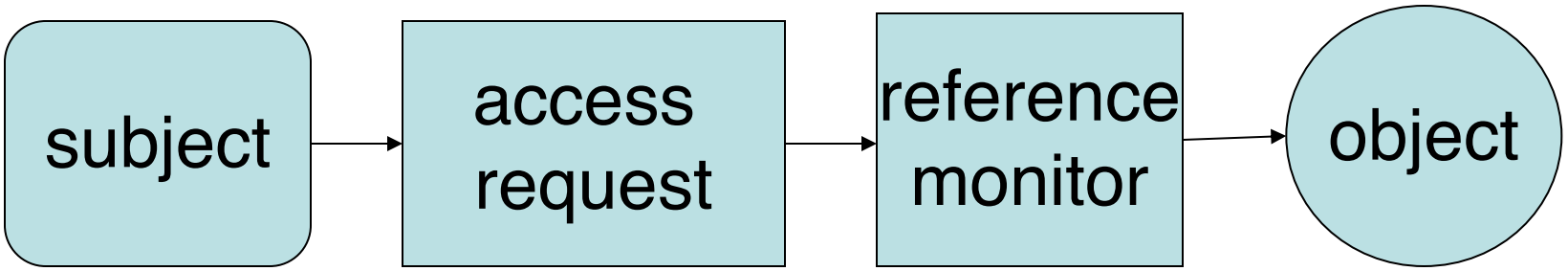
# Access Control

- Determine whether a **principal** can perform a requested **operation** on a target **object**
- **Principal:** user, process, etc.
- **Operation:** read, write, etc.
- **Object:** file, tuple, etc.
- Lampson defined the familiar **access matrix** and its two interpretations ACLs and capabilities [Lampson70]

# Why are we still talking about access control?

- An **access control policy** is a specification for an **access decision function**
- The policy aims to achieve
  - Permit the principal's intended function (availability)
  - Ensure security properties are met (integrity, confidentiality)
    - Limit to "Least Privilege," Protect system integrity, Prevent unauthorized leakage, etc.
    - Also known as 'constraints'
  - Enable administration of a changeable system (simplicity)

# A Model for Access Control




source (e.g. users, processes)

request

guard

resource (e.g. files, printers)

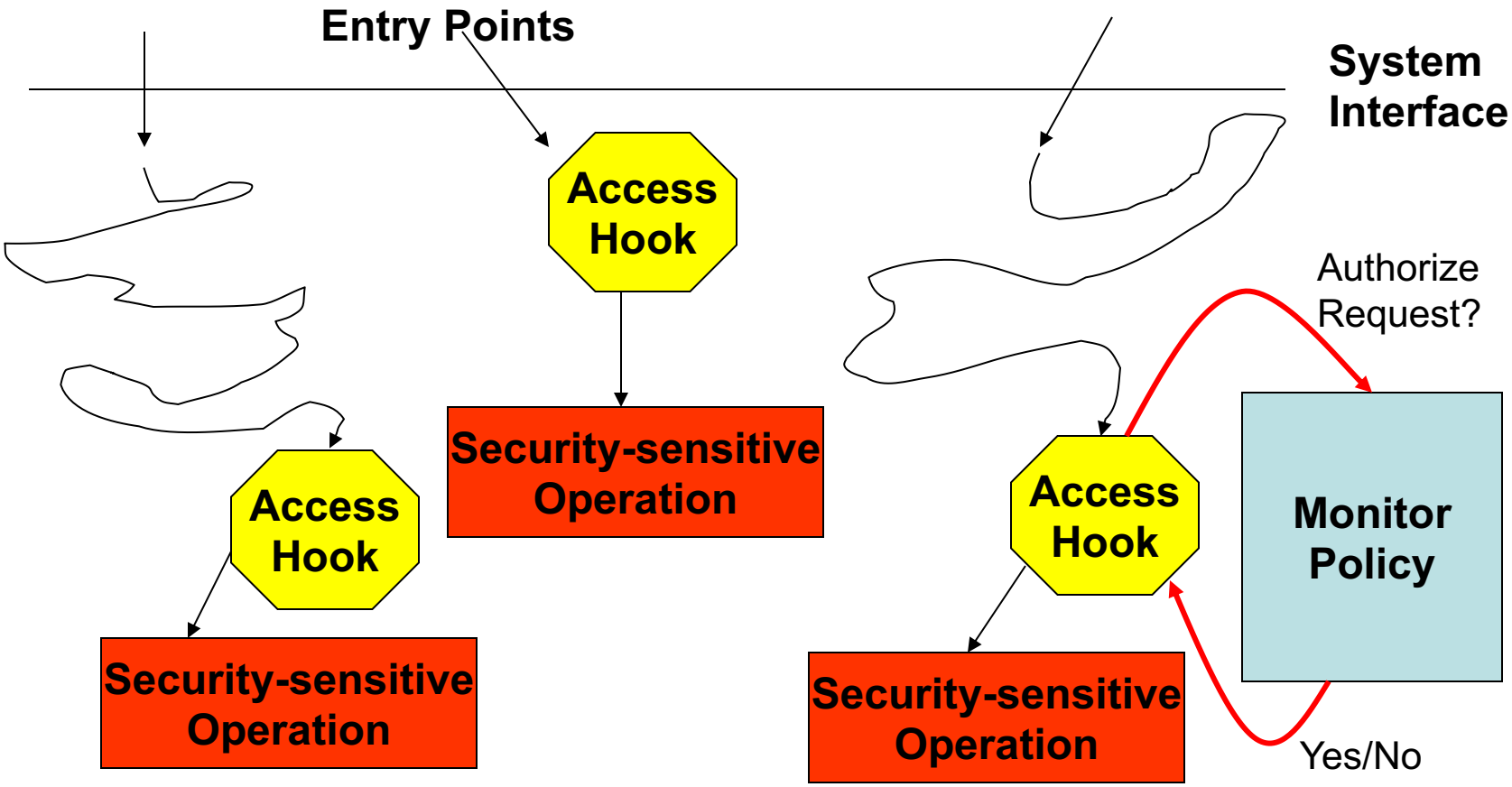
# Basic Terminology

- Subject/Principal: active entity – user or process
- Object: passive entity – file or resource
- Access operations: read, write, ...
  - Access operations vary from basic memory/file access to method calls in an object-oriented system.
  - Comparable systems may use different access operations.

# Authorization

- Access control decision is actually an *authorization* decision
- if *o* is an object, authorization answers the question "Who is trusted to access *o*?"

# Reference Monitor Model



# Reference Monitor Components

- Interface
  - Where to make access control decisions (mediation)
  - Which access control decisions to make (authorization)
  - Linux Security Modules interface
- Decision function
  - Compute decision based on request and policy
  - E.g., SELinux, LIDS, DTE, etc. modules
- Policy – our focus today
  - How to represent access control policy
  - Main mechanism issue – find mechanism to enable verification that policy achieves function and meets security guarantees

# Who Sets the Policy?

*Security policies* specify how subjects access objects. There are two mechanisms for deciding who is in charge of setting the policy:

- The owner of a resource decides who is allowed access. Such policies are called discretionary as access control is at the owner's discretion.
- A system wide policy decides who is allowed access. Such policies are called mandatory.

# Access Control Structures

- Requirements on access control structures:
  - ***The access control structure should help to express your desired access control policy.***
  - ***You should be able to check that your policy has been captured correctly.***
- Access rights can be defined individually for each combination of subject and object.
- For large numbers of subjects and objects, such structures are cumbersome to manage.
  - Intermediate levels of control are preferable.

# Access Control Matrix

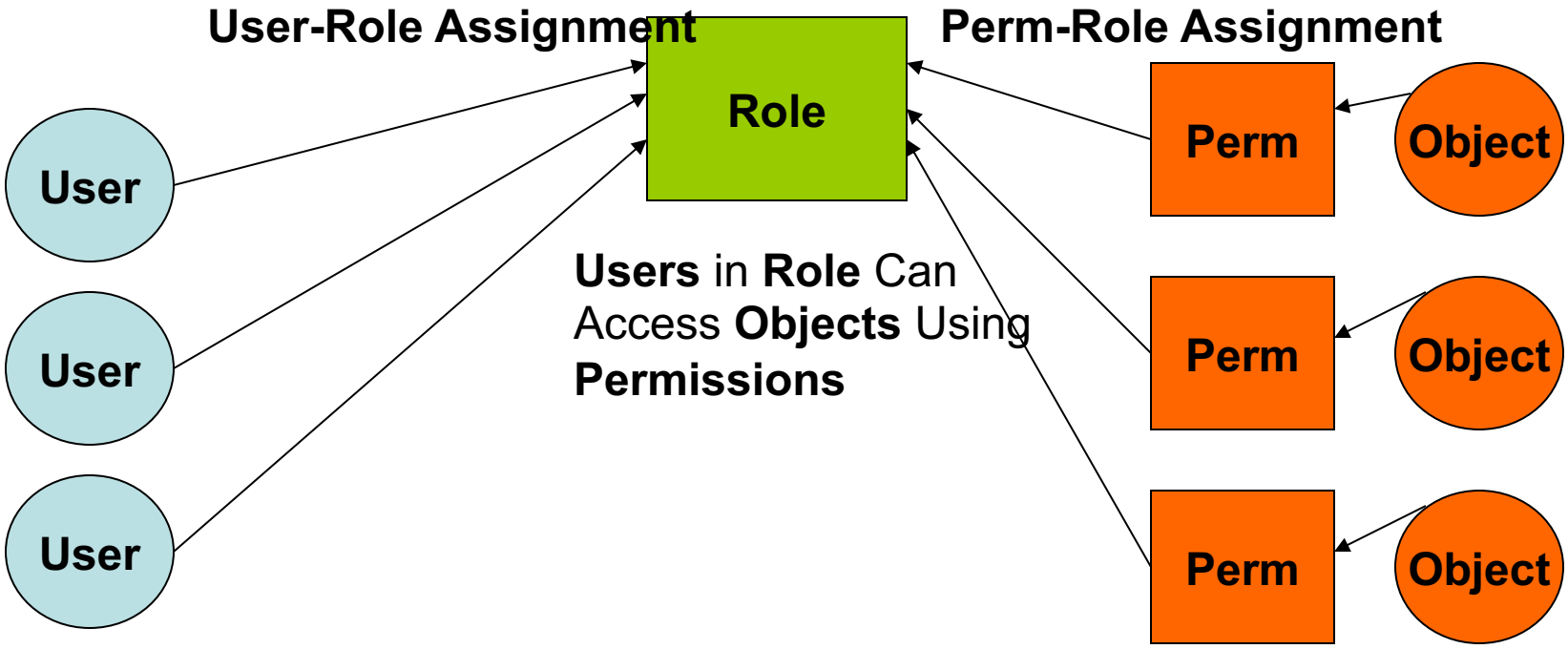
- $S$ … set of subjects
- $O$ … set of objects
- $A$ … set of access operations
- Access control matrix: $M = (M_{so})_{s \in S, o \in O}$, $M_{so} \subseteq A$; $M_{so}$ specifies the operations subject $s$ may perform on object $o$.

| | bill.doc | edit.exe | fun.com |
|---|---|---|---|
| Alice | {} | {exec} | {exec,read} |
| Bob | {read,write} | {exec} | {exec,read,write} |

# Need for Aggregate Models (RBAC)

- Practical ease of specification
  - Abstraction for users, permissions, constraints, administration
- Natural access control aggregations – based on organizational roles
  - As new employees join, their permission assignments are determined by their job
  - Permission assignment is largely static
- Central control and maintenance of access rights
- Flexible enough to enforce
  - least privilege, separation of duties, etc.

# Role-based Access Control



# Role vs. Types Data Structures

- RBAC
  - U: set of users
  - P: set of permissions
- Type Enforcement
  - E: set of subjects or objects
  - Permission Assignment
    - ST: set of subject types
    - OT: set of object types
    - O: set of operations

# Role-based Access Control Model

- Users: U
- Permissions: P
- Roles: R
- Assignments: User-role, perm-role, **role-role**
- **Sessions: S**
- Function: user(S), roles(S)
- **Constraints: C**

# RBAC Family of Models

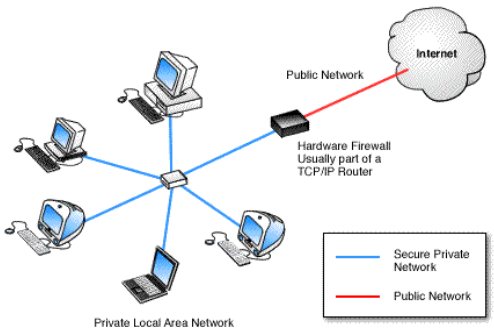
- $RBAC_0$ contains all but hierarchies and constraints
- $RBAC_1$ contains $RBAC_0$ and hierarchies
- $RBAC_2$ contains $RBAC_0$ and constraints
- $RBAC_3$ contains all
- The RBAC family idea has always been more a NIST initiative
- The RBAC families are present in the NIST RBAC standard [NIST2001] with slight modifications:
  - $RBAC_0$, $RBAC_1$ (options), $RBAC_3$ (SSD) , $RBAC_3$ (DSD)

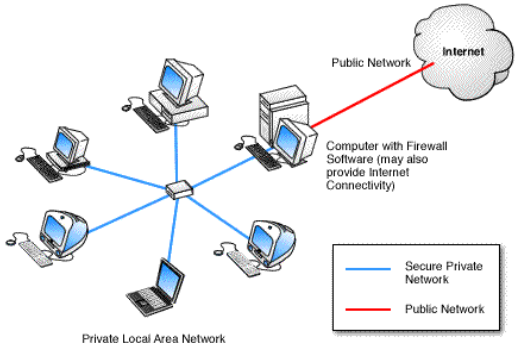
# Administration

- Discretionary Access Control
  - Users (typically object owner) can decide permission assignments
- Mandatory Access Control
  - System administrator decides on permission assignments
- Flexible Administrative Management
  - Access control models can be used to express administrative privileges

# Firewalls

- A part of computer system or network designed to stop unauthorized traffic flowing from one network to another.
- Separate trusted and untrusted components of a network.
- Main functionalities are filtering data, redirecting traffic and protecting against network attacks.



# Firewall Characteristics

- Design goals:
  - All traffic to/from inside from/to outside must pass through the firewall
  - Only authorized traffic (defined by the local security policy) will be allowed to pass
  - The firewall itself should be immune to penetration (use of trusted system with a secure operating system)

# Firewall Limitations

- cannot protect from attacks bypassing it
  - typical example: dial-in, dial-out
- cannot protect against internal threats
  - e.g. fired sysadmin ☺
- cannot effectively protect against transfer of all virus infected programs or files
  - because of heavy traffic and huge range of O/S & file types

# How do Firewalls work?

- There are different types of Firewalls.
- Some operate at the network layer.
- Others operate at the application layer.
- The higher the level the more security the firewall offers.

# Firewall Policy

- User control: Controls access to the data based on the role of the user who is attempting to access it. Applied to users inside the firewall perimeter.
- Service control: Controls access by the type of service offered by the host. Applied on the basis of network address, protocol of connection and port numbers.
- Direction control: Determines the direction in which requests may be initiated and are allowed to flow through the firewall. It tells whether the traffic is “inbound” (From the network to firewall) or vice-versa “outbound”

# Firewall actions

Accepted: Allowed to enter the connected network/host through the firewall.
Denied: Not permitted to enter the other side of firewall.
Rejected: Similar to “Denied”, but tells the source about this decision through ICMP packet.

Ingress filtering: Inspects the incoming traffic to safeguard an internal network and prevent attacks from outside.

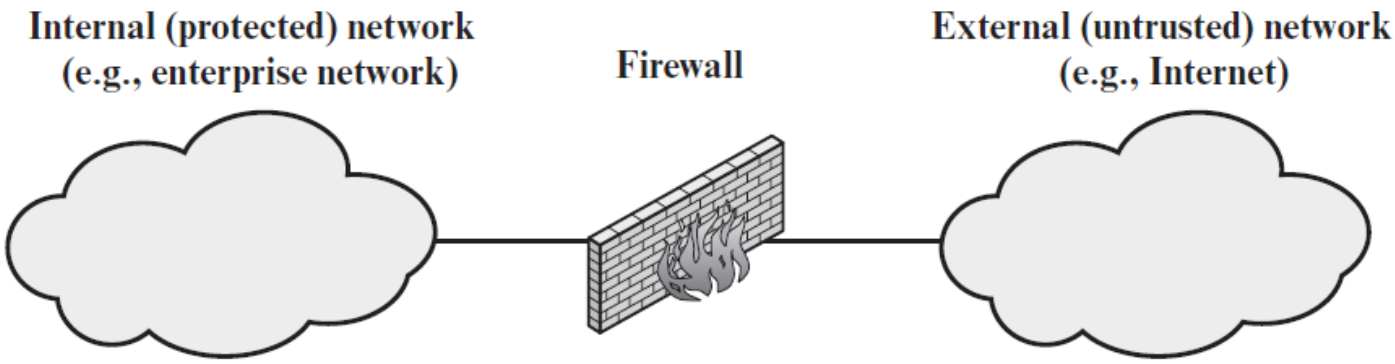
Egress filtering: Inspects the outgoing network traffic and prevent the users in the internal network to reach out to the outside network. For example like blocking social networking sites in school

# Types of Firewalls

- Packet-filtering firewall
- Application-level gateways
- Circuit-level gateways (not common, so skipped)

# Packet-filtering Firewall

- Foundation of any firewall system
- Applies a set of rules to each incoming/outgoing IP packet and then forwards (permits) or discards (denies) the packet (in both directions)
- The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header
- context is not checked



# Packet-filtering Firewall

- Filtering rules are based on
  - Source and Destination IP addresses
  - Source and Destination ports (services) and transport protocols (TCP or UDP)
- Rules are listed and a match is tried to be found starting with the first rule
  - Action is either permit or deny
  - Generally first matching rule is applied
  - If no match, then default policy is used
    - Default is either deny or permit

# Packet Filtering Examples - 1

| Rule | Direction | Source Address | Destination Address | Protocol | Destination Port | Action |
|---|---|---|---|---|---|---|
| A | In | External | Internal | TCP | 25 | Permit |
| B | Out | Internal | External | TCP | > 1023 | Permit |
| C | Out | Internal | External | TCP | 25 | Permit |
| D | In | External | Internal | TCP | > 1023 | Permit |
| E | Either | Any | Any | Any | Any | Deny |

- A: incoming SMTP traffic allowed from particular "External" IP addresses to particular "Internal" IP addresses
  - B: aimed to allow the response packets (problematic)
- C: Outgoing SMTP traffic allowed from particular "Internal" IP addresses to particular "External" IP addresses
  - D: aimed to allow the response packets (problematic)
- E: Default rule: deny (discard) the rest
  - Of course in a normal firewall there must be other "permit" rules for proper operation of other services

# Packet Filtering Examples - 2

| Rule | Direction | Source Address | Destination Address | Protocol | Destination Port | Action |
|---|---|---|---|---|---|---|
| A | In | External | Internal | TCP | 25 | Permit |
| B | Out | Internal | External | TCP | > 1023 | Permit |
| C | Out | Internal | External | TCP | 25 | Permit |
| D | In | External | Internal | TCP | > 1023 | Permit |
| E | Either | Any | Any | Any | Any | Deny |

- Rules B and D are problematic
- Rule D allows not only incoming SMTP responses, but any packet with destination port >1023
  - Malicious services use >1023 destination ports
- Solution: add source port to the rule set in order to set the application for response packets
  - For Rules B and D, source port is 25; for Rules A and C source port is >1023

# Packet Filtering Examples - 3

- Rule D is still problematic after adding source port value
  - The malicious traffic may mimic source port 25 and uses >1023 destination port
  - To resolve this issue we have to make sure that responses to SMTP requests is the ones to our requests; not a new traffic
    - Adding "TCP flag" field to the rule set helps
    - If ack flag is set, it is ack to our packet

| Rule | Direction | Source Address | Source Port | Dest Address | Protocol | Dest Port | Flag | Action |
|---|---|---|---|---|---|---|---|---|
| D | In | External | 25 | Internal | TCP | > 1023 | ACK | Permit |

- Another helper is stateful inspection (next slide)

# Stateful Inspection

- Example D shows that

  >1024 ports need to be opened

  – not only due to SMTP, all services have such a structure

    - <1024 ports are for servers, a client using a service should use a local port number between 1024 and 16383

- So the firewall should keep track of the currently opened >1024 ports
- A *stateful inspection firewall* keeps track of outbound TCP connections with local port numbers in a table and allow inbound traffic for >1024 ports if there is an entry in that table (see next slide for an example table)

# Stateful Inspection

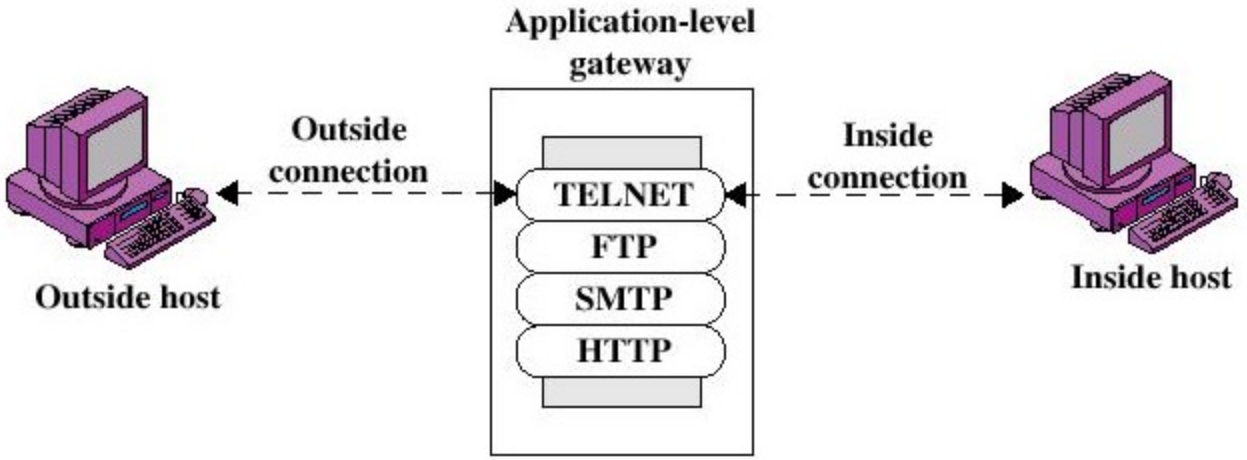
| Source Address | Source Port | Destination Address | Destination Port | Connection State |
|---|---|---|---|---|
| 192.168.1.100 | 1030 | 210.9.88.29 | 80 | Established |
| 192.168.1.102 | 1031 | 216.32.42.123 | 80 | Established |
| 192.168.1.101 | 1033 | 173.66.32.122 | 25 | Established |
| 192.168.1.106 | 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 |
| 223.212.212 | 1046 | 192.168.1.6 | 80 | Established |

# Packet-filtering Firewall

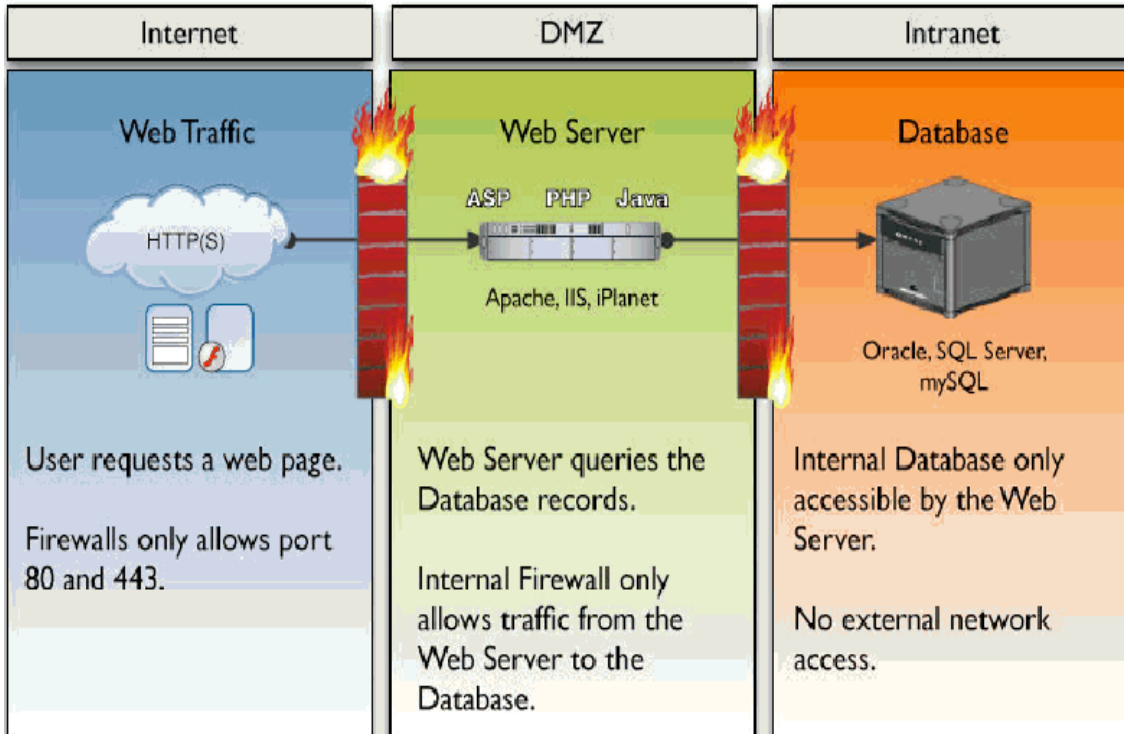
- Advantages:
  - Simplicity
  - High speed
  - Transparency to users
- Disadvantages
  - a port is either open or close; no application layer flexibility
  - IP address spoofing
    - attacker uses an internal IP address and hopes that packet penetrates into the system

# Application-level Gateway

- Application-level Gateway (proxy server)
  - Acts as a relay of application-level traffic
- Proxy obtains application specific information from the user and relays to the server
  - Optionally authenticates the users
- Only allowable applications can pass through
  - Feature-based processing is possible
- Additional processing overhead on each connection



- Firewall are common in every network deployment, so attackers use websites to get access to internal network

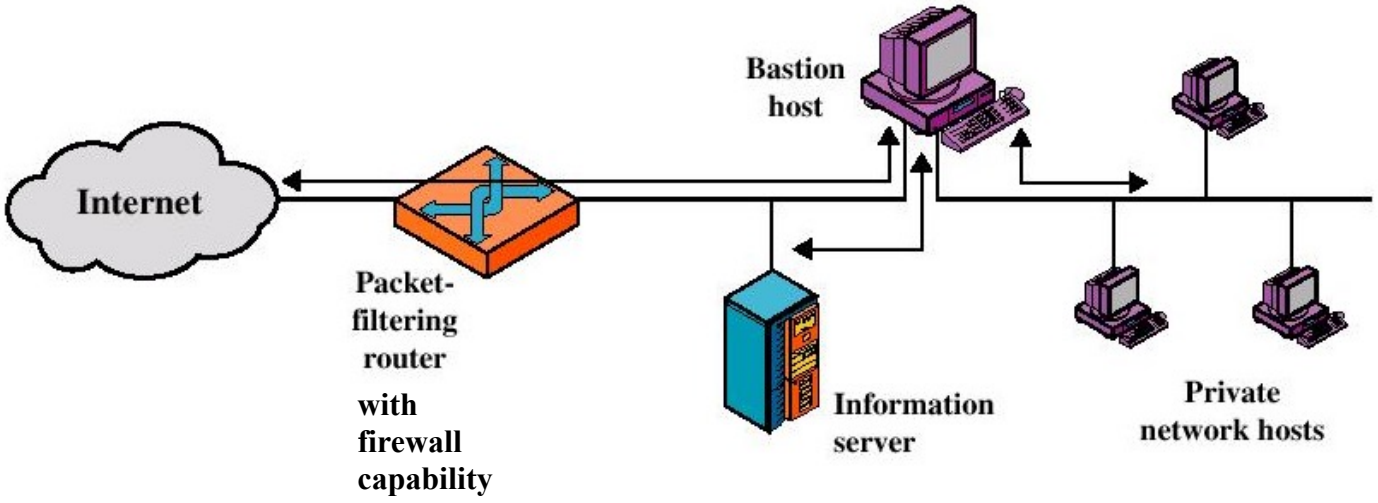


# Bastion Host

- A system identified by the firewall administrator as a critical strong point in the network security
  - Used in various firewall configuration (we'll see now)
- The bastion host serves as a platform for an application-level gateway
  - i.e. a proxy
- Potentially exposed to "hostile" elements, hence is secured to withstand this
  - Trusted system
  - Carefully configured and maintained

# Screened host firewall system (dual-homed bastion host)

- Only packets from and to the bastion host are allowed to pass through the router
- The bastion host performs authentication and proxy functions

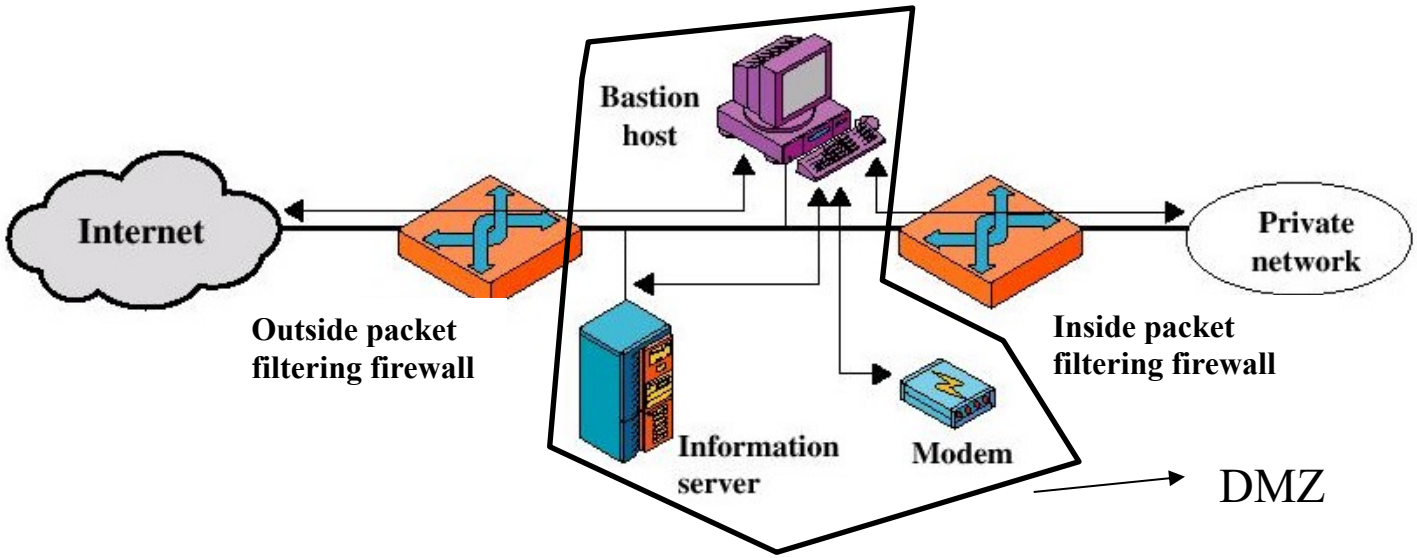


# Dual-homed Bastion Host

- Good security because of two reasons:
  - This configuration implements both packet-level and application-level filtering
  - An intruder must generally penetrate two separate systems in order to get to the internal network
- This configuration also has flexibility in providing direct Internet access to a public information server, e.g. Web server
  - by configuring the packet filtering router

# Screened-subnet Firewall System

- creates an isolated sub-network between firewalls
  - Internet and private network have access to this subnet
  - Traffic across the subnet is blocked
  - This subnet is the DMZ (demilitarized zone)
- Internal network is invisible to the Internet



# Building a Firewall using Netfilter

**Packet filter firewall implementation in Linux**

- Packet filtering can be done inside the kernel.
- Need changes in the kernel
- Linux provides two mechanisms to achieve this :

Loadable Kernel Modules: Allow privileged users to dynamically add/remove modules to the kernel, so there is no need to recompile the entire kernel.

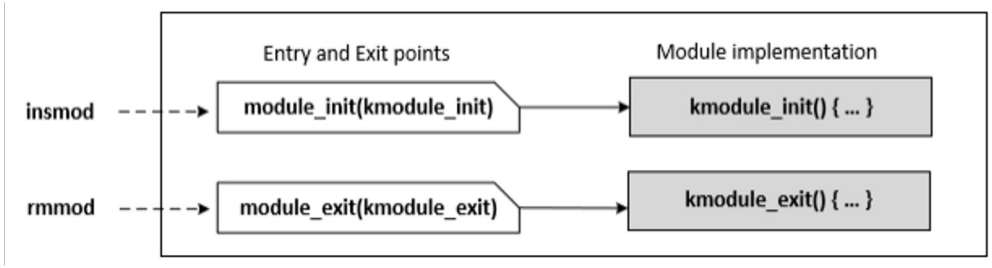
Netfilter: Provides hooks at critical points on the packet traversal path inside Linux Kernel.

# Loadable Kernel Modules

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int kmodule_init(void) {
   printk(KERN_INFO "Initializing this module\n");
   return 0;
}

static void kmodule_exit(void) {
   printk(KERN_INFO "Module cleanup\n");
}

module_init(kmodule_init);        ①
module_exit(kmodule_exit);        ②

MODULE_LICENSE("GPL");
```

Specify an initialization function that will be invoked when the kernel module is inserted.

Specify a cleanup function that will be invoked when the kernel module is removed.



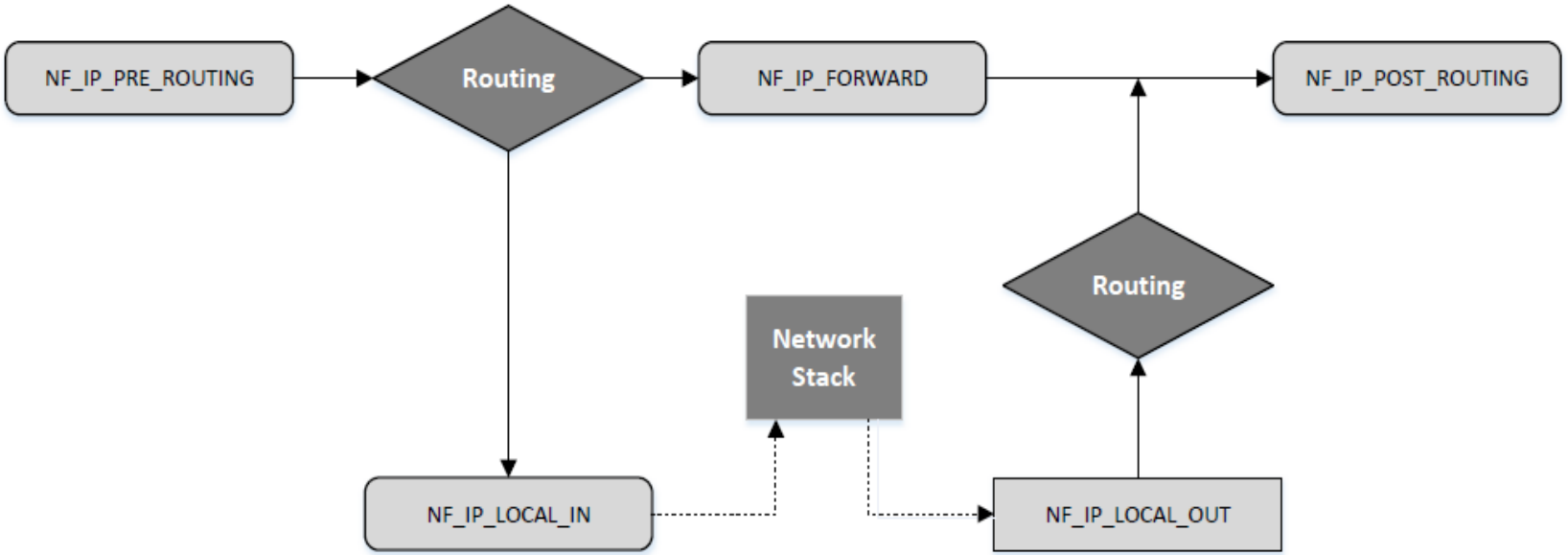
# Netfilter

- Netfilter hooks are rich packet processing and filtering framework.
- Each protocol stack defines a series of hooks along the packet's traversal path in the stack.
- Developers can use LKMs to register callback functions to these hooks.
- When a packet arrives at each of these hooks, the protocol stack calls the netfilter framework with the packet and hook number.
  - Netfilter checks if any kernel module has registered a callback function at this hook.
  - Each registered module will be called, and they are free to analyze or manipulate the packet and return the verdict on the packet.

# Netfilter: Verdict on Packets (Return Values)

`NF_ACCEPT`: Let the packet flow through the stack.
`NF_DROP`: Discard the packet.
`NF_QUEUE`: Pass the packet to the user space via nf_queue facility.
`NF_STOLEN`: Inform the netfilter to forget about this packet, The packet is further processed by the module.
`NF_REPEAT`: Request the netfilter to call this module again.

# Netfiler Hooks for IPv4



# Implementing a Simple Packet Filter Firewall

```
unsigned int telnetFilter(void *priv, struct sk_buff *skb,
                  const struct nf_hook_state *state)
{
  struct iphdr *iph;
  struct tcphdr *tcph;

  iph = ip_hdr(skb);
  tcph = (void *)iph+iph->ihl*4;

  if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23)) {
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
    ((unsigned char *)&iph->daddr)[0],
    ((unsigned char *)&iph->daddr)[1],
    ((unsigned char *)&iph->daddr)[2],
    ((unsigned char *)&iph->daddr)[3]);
    return NF_DROP;
  } else {
    return NF_ACCEPT;
  }
}
```

Decisions

The entire packet is provided here.

The filtering logic is hardcoded here. Drop the packet if the destination TCP port is 23 (telnet)

# Implementing a Simple Packet Filter Firewall

```
int setUpFilter(void) {
   printk(KERN_INFO "Registering a Telnet filter.\n");
   telnetFilterHook.hook = telnetFilter;
   telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
   telnetFilterHook.pf = PF_INET;
   telnetFilterHook.priority = NF_IP_PRI_FIRST;

   // Register the hook.
   nf_register_hook(&telnetFilterHook);
   return 0;
}

void removeFilter(void) {
   printk(KERN_INFO "Telnet filter is being removed.\n");
   nf_unregister_hook(&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);
```

Hook this callback function

Use this Netfilter hook

Register the hook

# Testing Our Firewall

```
$ sudo insmod telnetFilter.ko
$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: ...    ← Blocked!
$ dmesg
......
[1166456.149046] Registering a Telnet filter.
[1166535.962316] Dropping telnet packet to 10.0.2.5
[1166536.958065] Dropping telnet packet to 10.0.2.5

// Now, let's remove the kernel module

$ sudo rmmod telnetFilter
$ telnet 10.0.2.5
telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 12.04.2 LTS
ubuntu login:              ← Succeeded!
```

# Iptables Firewall in Linux

- Iptables is a built-in firewall based on netfilter.
- User-space program: iptables
- Usually, iptables refer to both kernel and user space programs.
- Rules are arranged in hierarchical structure as shown in the table.

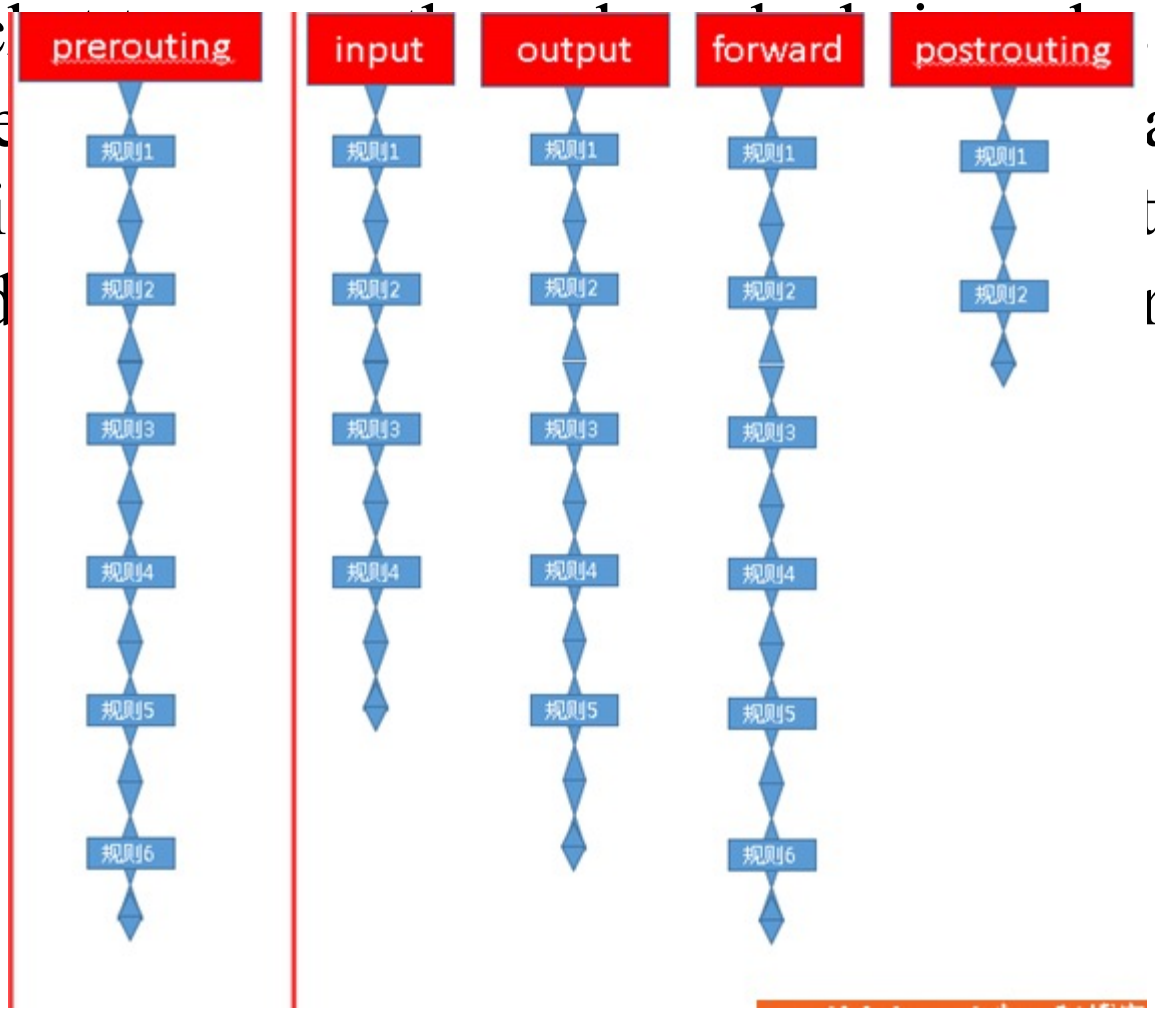
| Table | Chain | Functionality |
|---|---|---|
| filter | INPUT<br>FORWARD<br>OUTPUT | Packet filtering |
| nat | PREROUTING<br>INPUT<br>OUTPUT<br>POSTROUTING | Modifying source or destination network addresses |
| mangle | PREROUTING<br>INPUT<br>FORWARD<br>OUTPUT<br>POSTROUTING | Packet content modification |

# Iptables Firewall - Structure

- Each table contains several chains, each of which corresponds to a netfilter hook.
- Each chain indicates where its rules are enforced.
  - Example : Rules on FORWARD chain are enforced at NF_IP_FORWARD hook and rules on INPUT chain are enforced at NF_IP_LOCAL_IN hook.
- Each chain contains a set of firewall rules that will be enforced.
- User can add rules to the chains.
  - Example : To block all incoming telnet traffic, add a rule to the INPUT chain of the filter table

# Traversing Chains and Rule Matching

As a pac on the chain are atch or not. If there i tion is executed r-defined chain.



# Iptables command

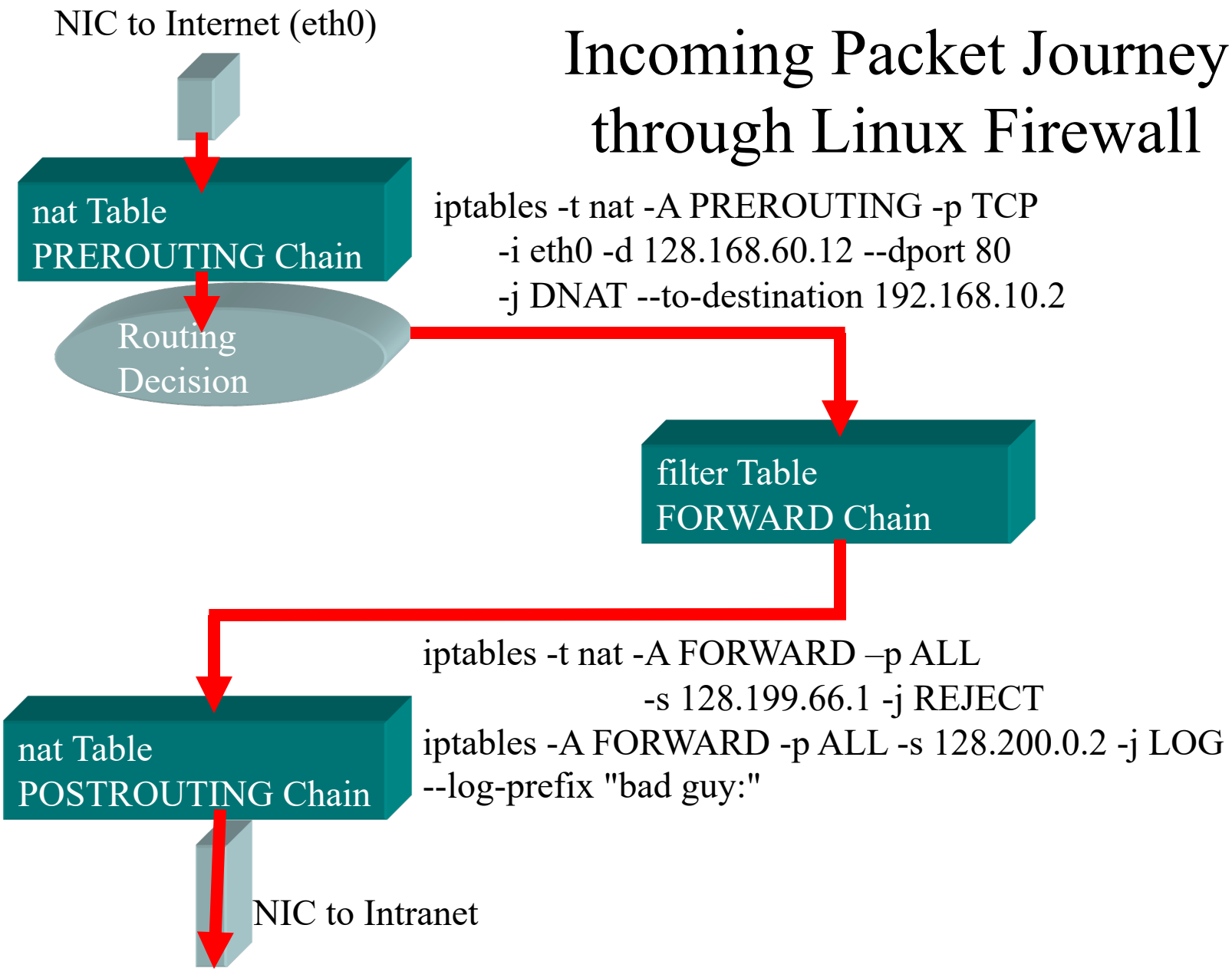
- iptables -t nat -A PREROUTING -p TCP
  -i eth0 -d 128.168.60.12 --dport 80
  -j DNAT --to-destination 192.168.10.2
- -t specify the type of tables
  -A Append to a specific chain
  -p specify the protocol
  -i specify the incoming interface
  -d specify the matched destination IP address in packet
  -j specify the “target” or operation to be performed.
  --to-destination substitute the destination IP address.

# Traversing Chains and Rule Matching

Example: Increase the TTL field of all packets by 5.
Solution: Add a rule to the mangle table and choose a chain provided by netfilter hooks. We choose PREROUTING chain so the changes can be applied to all packets, regardless they are for the current host or for others.

```
// -t mangle = Add this to 'mangle' table
// -A PREROUTING = Append this rule to PREROUTING chain

iptables -t mangle -A PREROUTING -j TTL --ttl-inc 5
```

# Incoming Packet Journey through Linux Firewall



# Overview of IDS

## What is a IDS

- Intrusions
  - authorized users of the systems who attempt to gain additional privileges for which they are not authorized,
  - authorized users who misuse the privileges given them.
- Intrusion detection
  - monitoring and collecting the events occurring in a computer system or network
  - analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network.

# **Intrusion Detection** *System* [IDS] is:

- A defense system, which detects hostile activates in a network and possibly repels it.
- Goal:
  - To thwart the attack
  - Conduct forensic investigation
  - Minimize damage
  - Learn how attack was conducted and improve system security

# 1990's – First Commercial IDSs

- 1989 – Haystack Labs releases Stalker
- 1990 – SAIC introduces CMDS
- 1990 – Todd Heberlein - NIDS and dIDS
- 1990 - Air Force's Cryptologic Support Center creates ASIM(软件+硬件）
- 1994 – Wheel Group releases NetRanger
  - First commercial NIDS

# 1998-? - Commercial Explosion

- 1998 – Snort ® released
- Prompts ISS, Cisco, Symantec, and others to create IDS products or to merge with companies that sell them

# Recent Research in IDS

- **NIDES** - distributed collection of host data, centralized analysis (extension of IDES)
- **NSM** - network traffic monitoring for anomalous packets
- **DIDS** - combines host-based (Haystack) and network monitoring (NSM)
- **CSM** - peer-to-peer distributed analysis

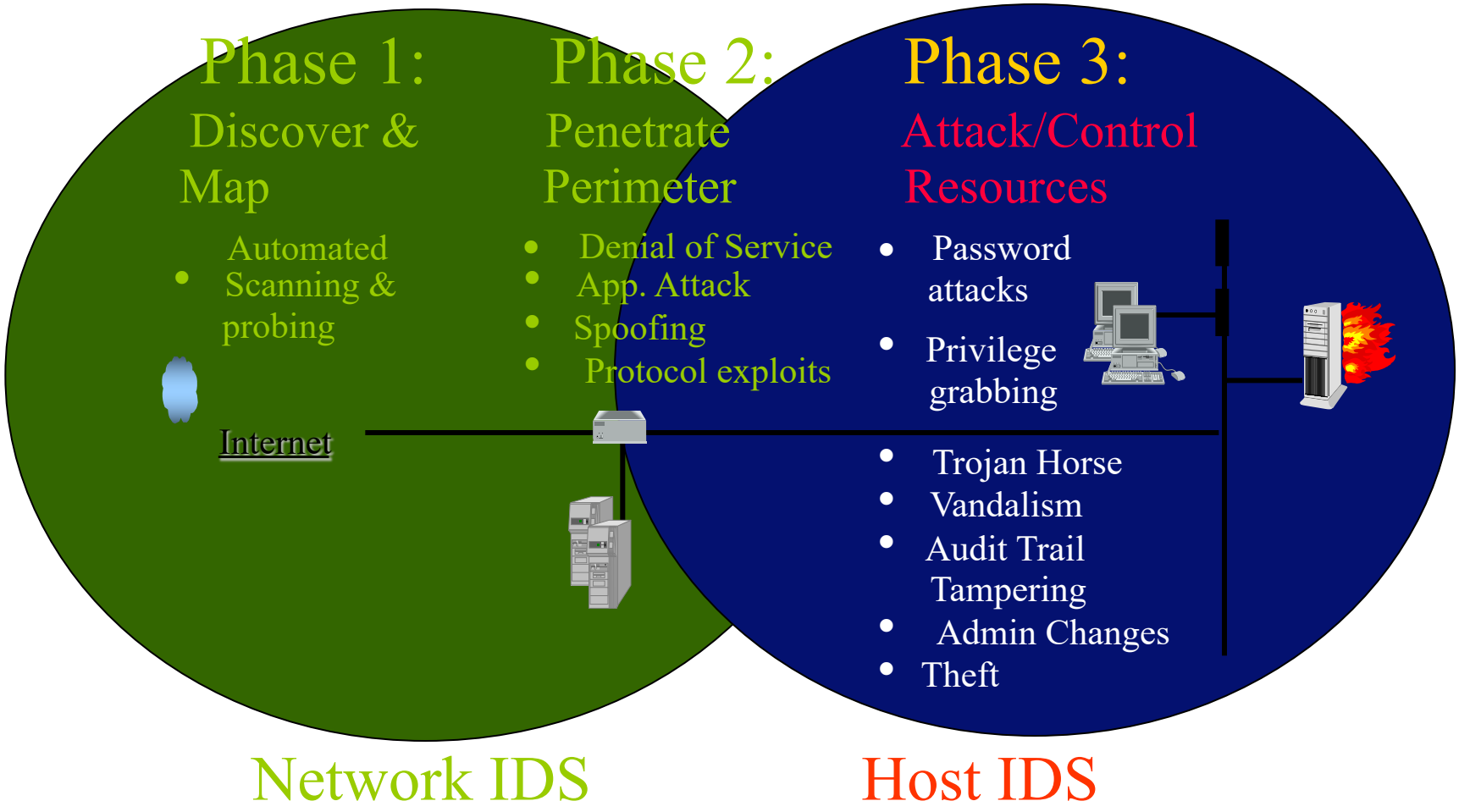
# Recent Research **(continued)**

- **Bro** - analyzes packet contents
- **GrIDS** - builds graphs of network activity and looks for anomalies
- **STAT** and **NetSTAT** - model attack with state machine. if accepted, attack occurred
- **EMERALD** - framework for building an ID system with distributed collection and analysis, modular design (extended NIDES)

# TYPES OF IDS

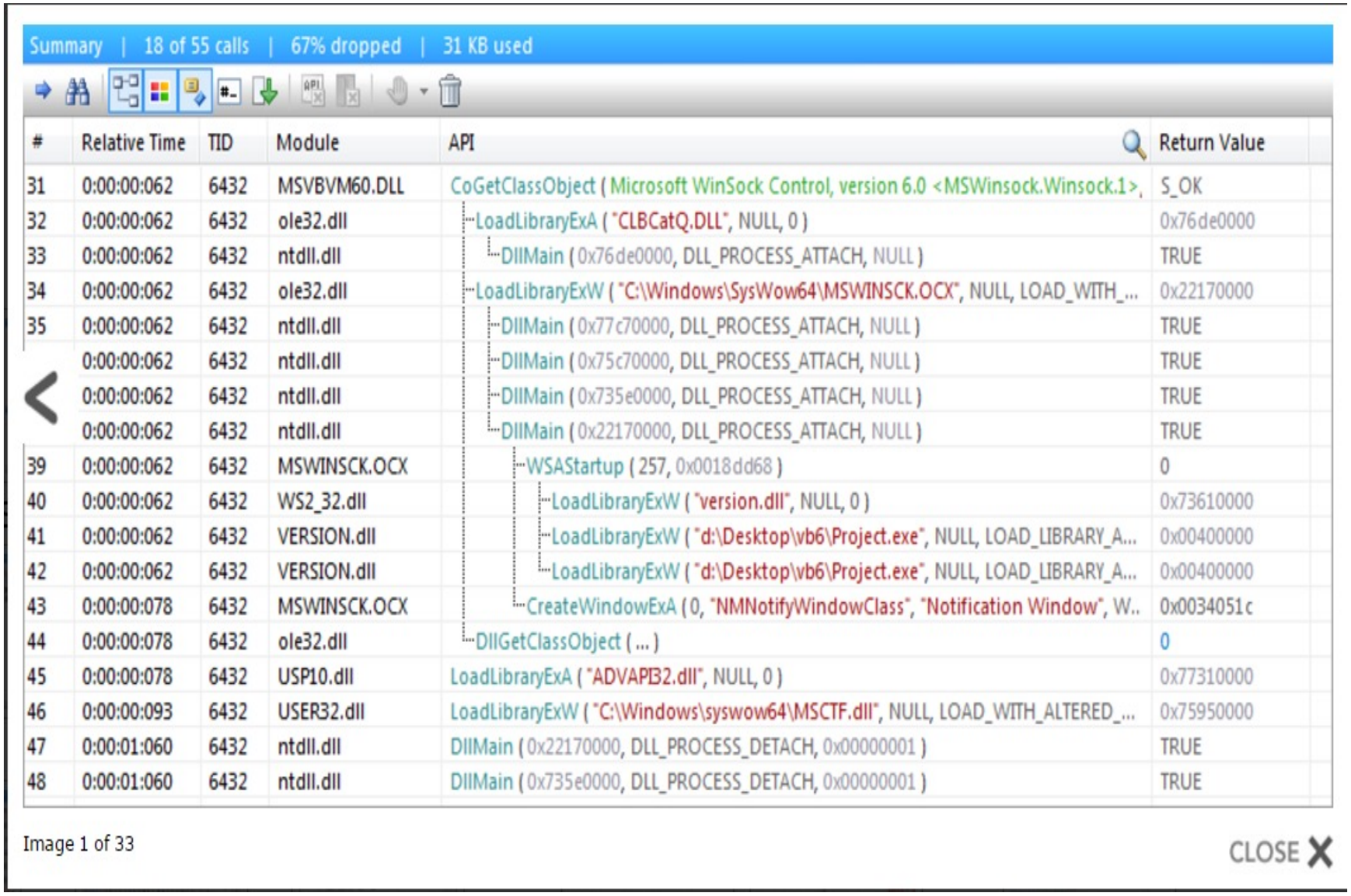
1. HOST – BASED (HIDS)
2. NETWORK – BASED (NIDS)
3. HYBRID

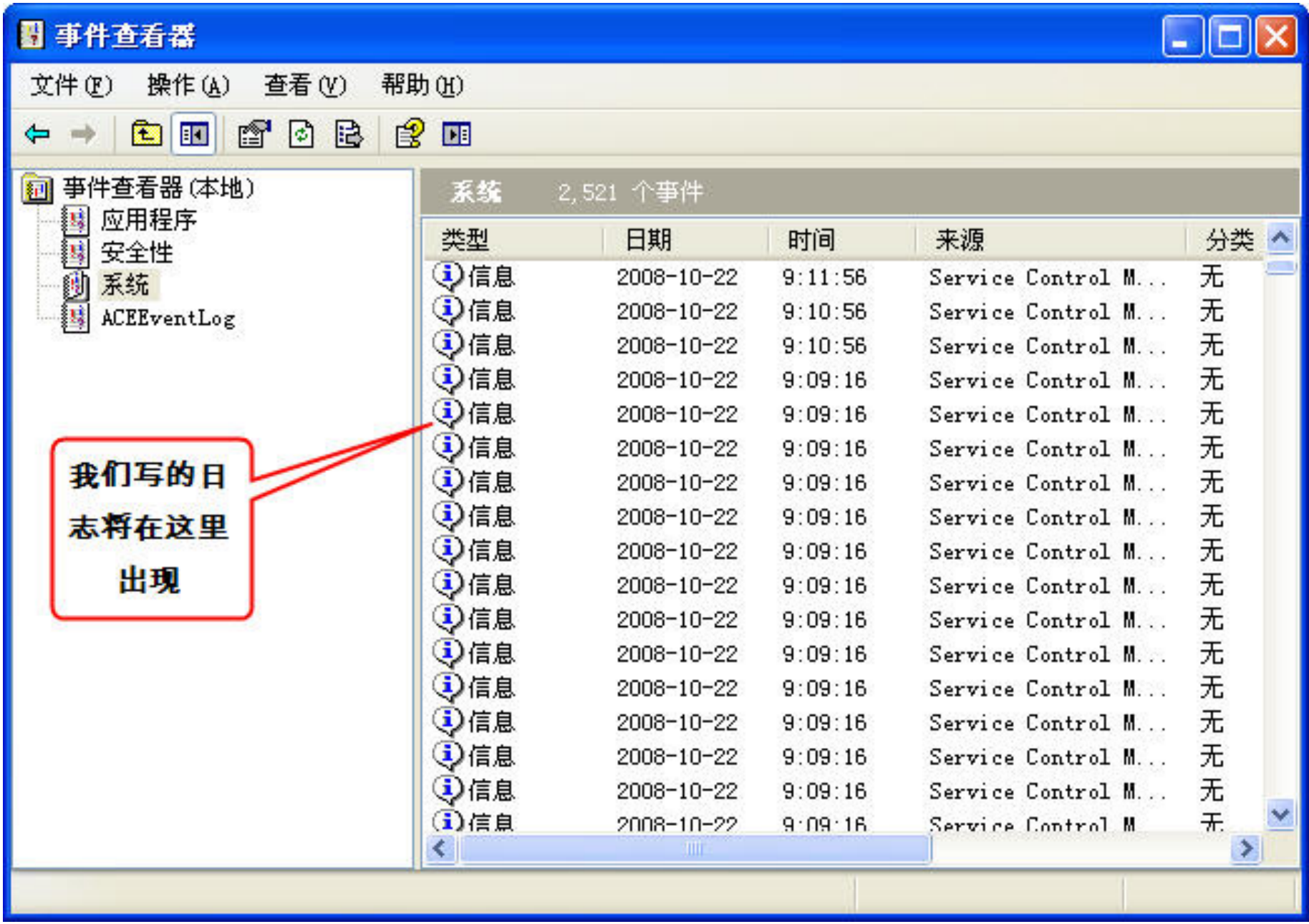
# ❖ Hacker intrusion's procedure



# HIDS

- **Host-Based IDSs**
  - Host-based IDS operate on information collected from within an individual computer system.
  - Host-based IDSs normally utilize information sources of two types, operating system audit trails, and system logs.


Summary | 18 of 55 calls | 67% dropped | 31 KB used
# Relative Time TID Module API Return Value
31 0:00:00:062 6432 MSVBVM60.DLL CoGetClassObject ( Microsoft WinSock Control, version 6.0 <MSWinsock.Winsock.1>, S_OK
32 0:00:00:062 6432 ole32.dll LoadLibraryExA ( "CLBCatQ.DLL", NULL, 0 ) 0x76de0000
33 0:00:00:062 6432 ntdll.dll DllMain ( 0x76de0000, DLL_PROCESS_ATTACH, NULL ) TRUE
34 0:00:00:062 6432 ole32.dll LoadLibraryExW ( "C:\Windows\SysWow64\MSWINSCK.OCX", NULL, LOAD_WITH_... 0x22170000
35 0:00:00:062 6432 ntdll.dll DllMain ( 0x77c70000, DLL_PROCESS_ATTACH, NULL ) TRUE
0:00:00:062 6432 ntdll.dll DllMain ( 0x75c70000, DLL_PROCESS_ATTACH, NULL ) TRUE
0:00:00:062 6432 ntdll.dll DllMain ( 0x735e0000, DLL_PROCESS_ATTACH, NULL ) TRUE
0:00:00:062 6432 ntdll.dll DllMain ( 0x22170000, DLL_PROCESS_ATTACH, NULL ) TRUE
39 0:00:00:062 6432 MSWINSCK.OCX WSAStartup ( 257, 0x0018dd68 ) 0
40 0:00:00:062 6432 WS2_32.dll LoadLibraryExW ( "version.dll", NULL, 0 ) 0x73610000
41 0:00:00:062 6432 VERSION.dll LoadLibraryExW ( "d:\Desktop\vb6\Project.exe", NULL, LOAD_LIBRARY_A... 0x00400000
42 0:00:00:062 6432 VERSION.dll LoadLibraryExW ( "d:\Desktop\vb6\Project.exe", NULL, LOAD_LIBRARY_A... 0x00400000
43 0:00:00:078 6432 MSWINSCK.OCX CreateWindowExA ( 0, "NMNotifyWindowClass", "Notification Window", W.. 0x0034051c
44 0:00:00:078 6432 ole32.dll DllGetClassObject ( ... ) 0
45 0:00:00:078 6432 USP10.dll LoadLibraryExA ( "ADVAPI32.dll", NULL, 0 ) 0x77310000
46 0:00:00:093 6432 USER32.dll LoadLibraryExW ( "C:\Windows\syswow64\MSCTF.dll", NULL, LOAD_WITH_ALTERED_... 0x75950000
47 0:00:01:060 6432 ntdll.dll DllMain ( 0x22170000, DLL_PROCESS_DETACH, 0x00000001 ) TRUE
48 0:00:01:060 6432 ntdll.dll DllMain ( 0x735e0000, DLL_PROCESS_DETACH, 0x00000001 ) TRUE
Image 1 of 33
CLOSE


事件查看器
文件(F) 操作(A) 查看(V) 帮助(H)
事件查看器(本地)
应用程序
安全性
系统
ACEEventLog
系统 2,521 个事件
类型 日期 时间 来源 分类
信息 2008-10-22 9:11:56 Service Control M... 无
信息 2008-10-22 9:10:56 Service Control M... 无
信息 2008-10-22 9:10:56 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
信息 2008-10-22 9:09:16 Service Control M... 无
我们写的日志将在这里出现

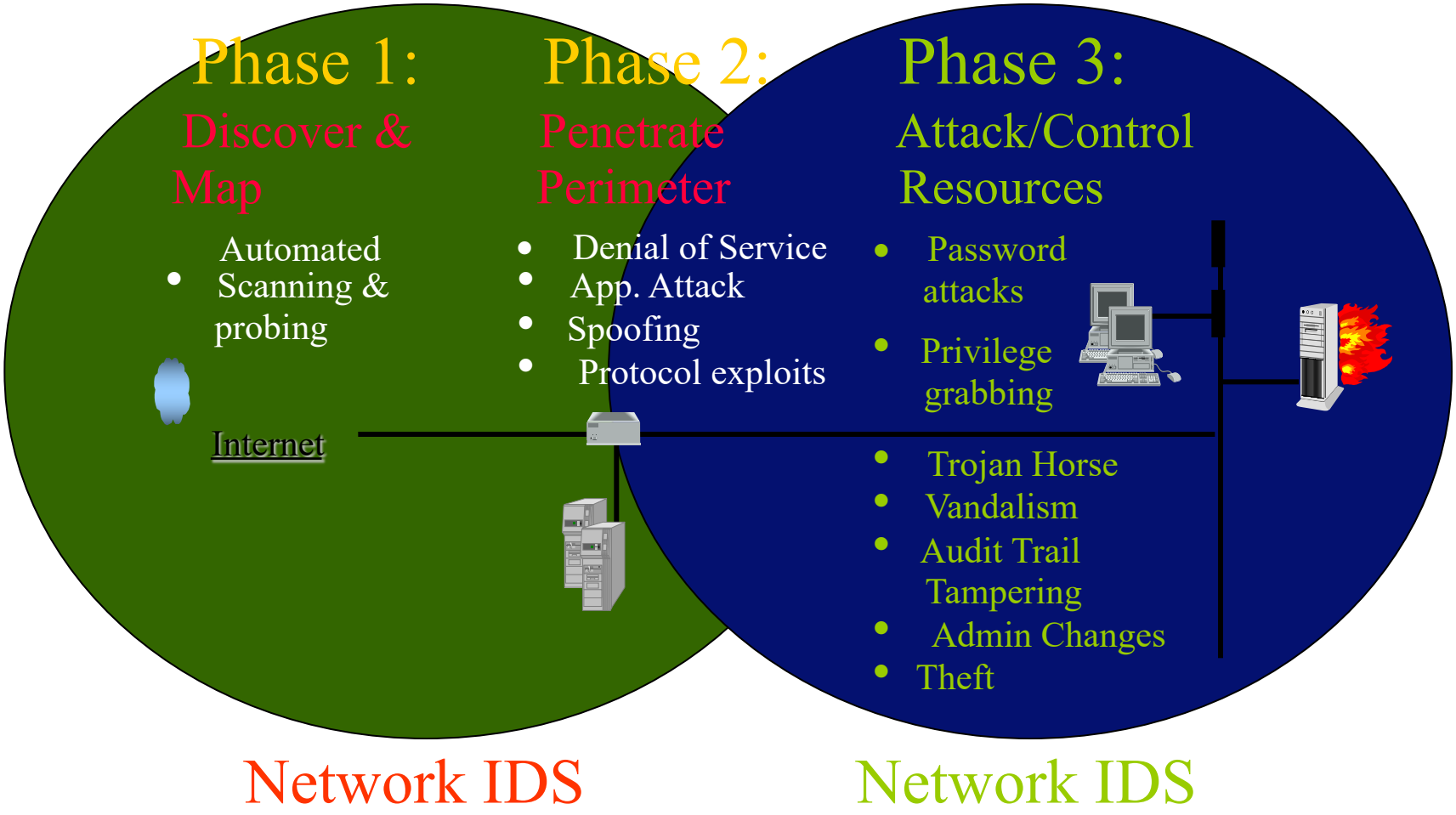
# HIDS

- ***Advantages:***
  - Host-based can detect attacks that cannot be seen by a network-based IDS.
  - host-based information sources are generated before data is encrypted and/or after the data is decrypted at the destination host
  - Host-based IDS are unaffected by switched networks.
  - When Host-based IDS operate on OS audit trails, they can help detect Trojan Horse or other attacks that involve software integrity breaches.

# HIDS

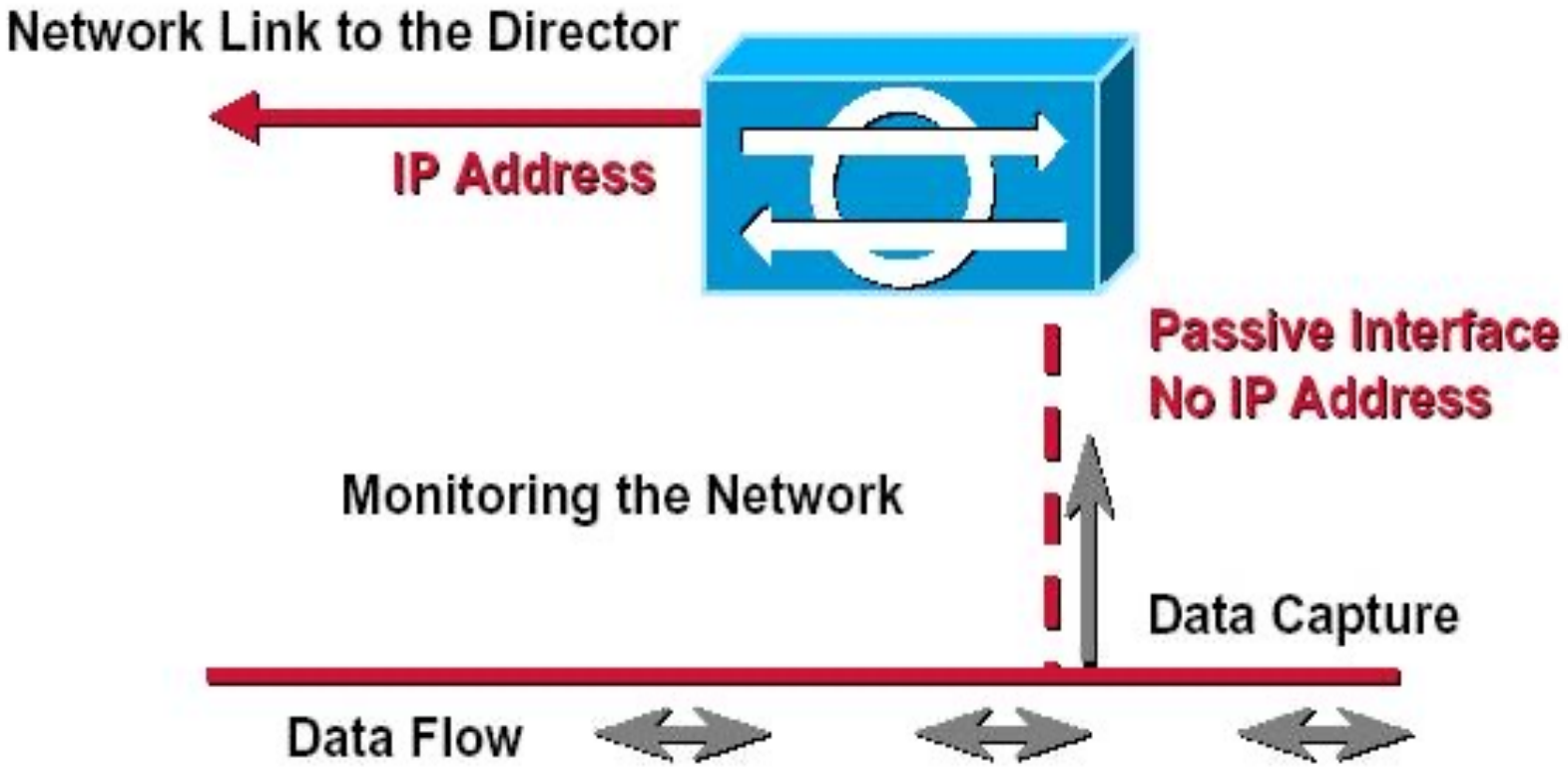
- ***Disadvantages***
  - Information must be configured and managed for every host monitored.
  - IDS may be attacked and disabled as part of the attack.
  - Host-based IDS are inflicting a performance cost on the monitored systems.
- .

# ❖ Hacker intrusion's procedure



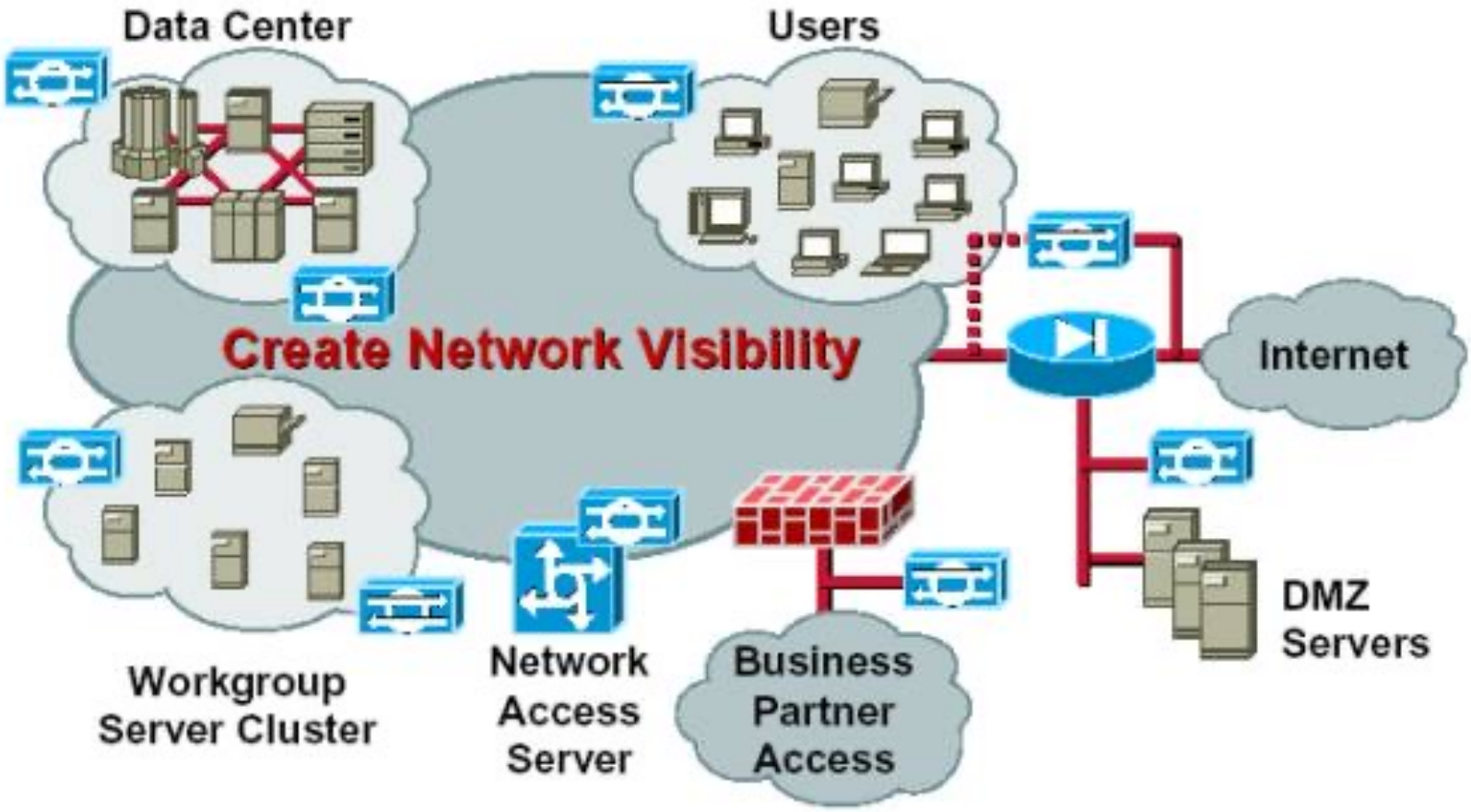
# NIDS

PASSIVE Interface to Network Traffic



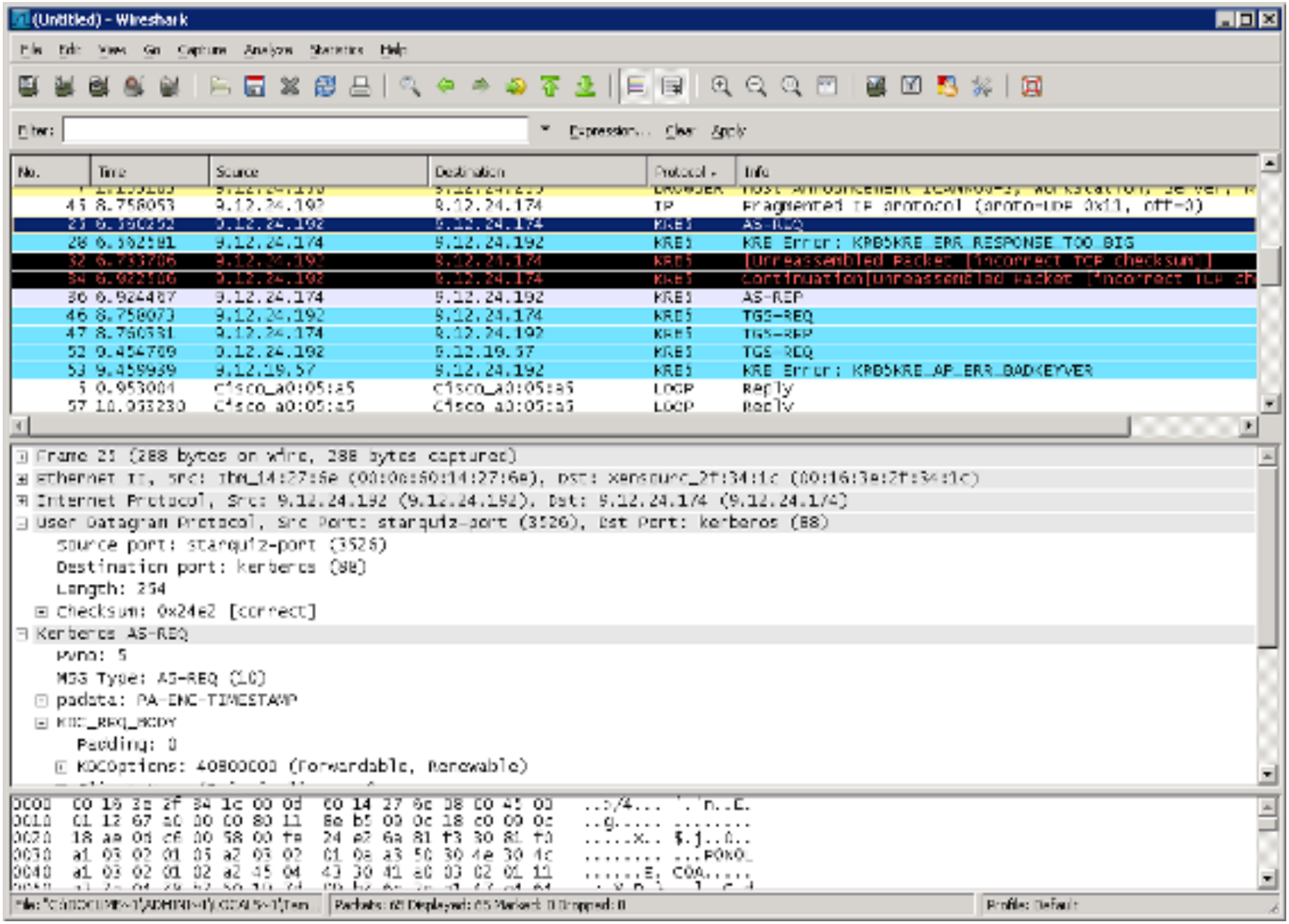
# NIDS (cont)

Sensor Placement



# NIDS

- **Network-Based IDSs**
  - Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network.
  - These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console.
  - As the sensors are limited to running the IDS, they can be more easily secured against attack.
  - Many of these sensors are designed to run in “stealth” mode, in order to make it more difficult for an attacker to determine their presence and location.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
45 8.758053 9.12.24.192 9.12.24.174 IP Fragmented IP protocol (proto=UDP 0x11, off=0)
21 6.390252 9.12.24.192 9.12.24.174 KRB5 AS-REQ
28 6.392591 9.12.24.174 9.12.24.192 KRB5 KRB Error: KRB5KRB_ERR_RESPONSE_TOO_BIG
32 6.733705 9.12.24.192 9.12.24.174 KRB5 [Unreassembled Packet [incorrect TCP checksum]]
34 6.922106 9.12.24.192 9.12.24.174 KRB5 Continuation[Unreassembled Packet [incorrect TCP ch
36 6.924487 9.12.24.174 9.12.24.192 KRB5 AS-REP
46 8.758073 9.12.24.192 9.12.24.174 KRB5 TGS-REQ
47 8.760331 9.12.24.174 9.12.24.192 KRB5 TGS-REP
52 9.454709 9.12.24.192 9.12.19.57 KRB5 TGS-REQ
53 9.459939 9.12.19.57 9.12.24.192 KRB5 KRB Error: KRB5KRB_AP_ERR_BADKEYVER
5 0.953004 Cisco_a0:05:a5 Cisco_a0:05:a5 LOOP Reply
57 10.033230 Cisco_a0:05:a5 Cisco_a0:05:a5 LOOP Reply
Frame 21 (288 bytes on wire, 288 bytes captured)
Ethernet II, Src: Ibm_14:27:6e (00:06:60:14:27:6e), Dst: Xerox_2f:34:1c (00:16:3e:2f:34:1c)
Internet Protocol, Src: 9.12.24.192 (9.12.24.192), Dst: 9.12.24.174 (9.12.24.174)
User Datagram Protocol, Src Port: starquiz-port (3526), Dst Port: kerberos (88)
Source port: starquiz-port (3526)
Destination port: kerberos (88)
Length: 264
Checksum: 0x24e2 [correct]
Kerberos AS-REQ
Pvno: 5
MSG Type: AS-REQ (10)
padata: PA-ENC-TIMESTAMP
KDC_REQ_BODY
Padding: 0
KDCOptions: 40800000 (Forwardable, Renewable)
0000 00 16 3e 2f 34 1c 00 06 60 14 27 6e 08 00 45 00
0010 01 12 67 a0 00 00 80 11 8e b5 09 0c 18 c0 09 0c
0020 18 ae 0d c6 00 58 00 fe 24 e2 6a 81 f3 30 81 f0
0030 a1 03 02 01 05 a2 03 02 01 0a a3 50 30 4e 30 4c
0040 a1 03 02 01 02 a2 45 04 43 30 41 a0 03 02 01 12
Packets: 60 Displayed: 60 Marked: 0 Dropped: 0
Profile: Default

# NIDS (cont)

Advantages

- NIDS uses a passive interface to capture network packets for analyzing.
- NIDS sensors placed around the globe can be configured to report back to a central site, enabling a small team of security experts to support a large enterprise.
- NIDS systems scale well for network protection because the number of actual workstations, servers, or user systems on the network is not critical – the amount of traffic is what matters
- Most network-based IDSs are OS-Independent
- Provide better security against DOS attacks

# NIDS (cont)

Disadvantages

- Cannot scan protocols or content if network traffic is encrypted
- Intrusion detection becomes more difficult on modern switched networks
- Current network-based monitoring approaches cannot efficiently handle high-speed networks
- Most of Network-based systems are based on predefined attack signatures--signatures that will always be a step behind the latest underground exploits

# AIDS

- **Application-Based IDSs**
  - The most common information sources used by application-based IDS are the application's transaction log files.
  - with significant domain or application-specific knowledge included in the analysis engine.
  - detect suspicious behavior due to authorized users exceeding their authorization.

# AIDS

- ***Advantages***
  - Application-based IDS can trace unauthorized activity to individual users.
  - Application-based IDS can often work in encrypted environments.

# AIDS

- ***Disadvantages:***
  - applications logs are not as well-protected as the operating system audit trails used for host-based IDS.
  - Application-based IDS often monitor events at the user level of abstraction, they usually cannot detect Trojan Horse or other such software tampering attacks.

# HYBRID

- Although the two types of Intrusion Detection Systems differ significantly from each other, but they also complement each other.
- Such a system can target activity at any or all levels
- It is easier to see patterns of attacks over time and across the network space
- No proven industry standards with regards to interoperability of intrusion detection components
- Hybrid systems are difficult to manage and deploy

# ID's Techniques

- Anomaly Detection
- Misuse Detection
- Hybrid
  - NADIR (Los Alamos)
  - Haystack (Air force, adaptive)
  - Hyperview (uses neural network)
  - Distributed IDS (Haystack + NSM)

# Anomaly Detection

- Define a profile describing "normal" behavior
  - Works best for "small", well-defined systems (single program rather than huge multi-user OS)
- Profile may be statistical
  - Build it manually (this is hard)
  - Use machine learning and data mining techniques
    - Log system activities for a while, then "train" IDS to recognize normal and abnormal patterns
  - Risk: attacker trains IDS to accept his activity as normal
    - Daily low-volume port scan may train IDS to accept port scans
- IDS flags deviations from the "normal" profile

# What's a "Profile?"

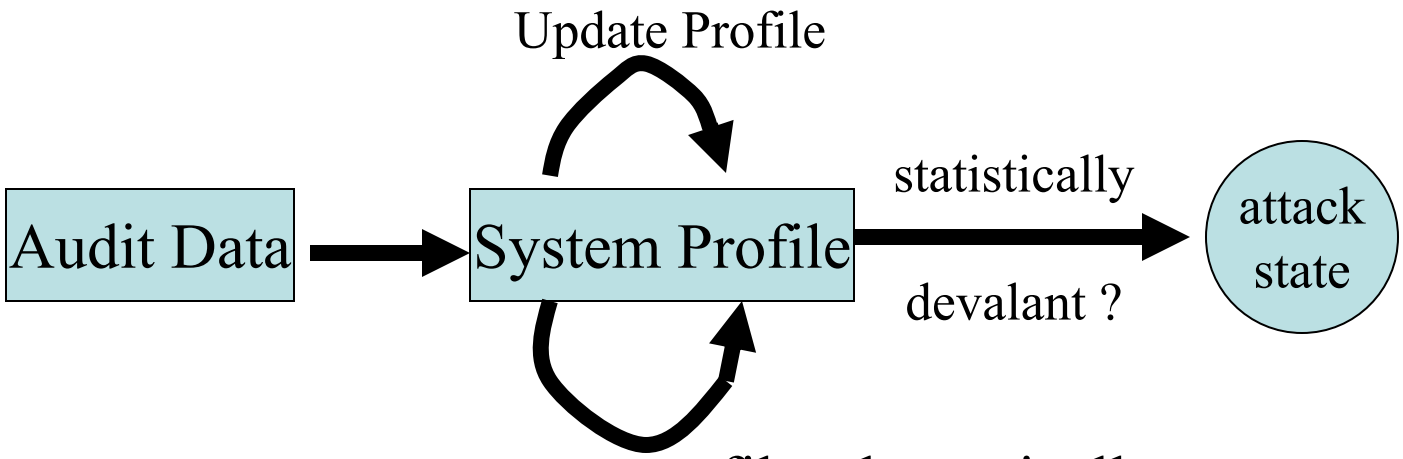
- Login and session activity
  - Login and location frequency; last login; password fails; session elapsed time; session output, CPU, I/O
- Command and program execution
  - Execution frequency; program CPU, I/O, other resources (watch for exhaustion); denied executions
- File access activity
  - Read/write/create/delete frequency; records read/written; failed reads, writes, creates, deletes; resource exhaustion

# Level of Monitoring

- Which types of events to monitor?
  - OS system calls
  - Command line
  - Network data (e.g., from routers and firewalls)
  - Processes
  - Keystrokes
  - File and device accesses

# ADSs(Contd.)

A typical anomaly detection system



Training Set

| Tid | SrcIP | Start time | Dest IP | Dest Port | Number of bytes | Attack |
|---|---|---|---|---|---|---|
| 1 | 206.135.38.95 | 11:07:20 | 160.94.179.223 | 139 | 192 | No |
| 2 | 206.163.37.95 | 11:13:56 | 160.94.179.219 | 139 | 195 | No |
| 3 | 206.163.37.95 | 11:14:29 | 160.94.179.217 | 139 | 180 | No |
| 4 | 206.163.37.95 | 11:14:30 | 160.94.179.255 | 139 | 199 | No |
| 5 | 206.163.37.95 | 11:14:32 | 160.94.179.254 | 139 | 19 | Yes |
| 6 | 206.163.37.95 | 11:14:35 | 160.94.179.253 | 139 | 177 | No |
| 7 | 206.163.37.95 | 11:14:36 | 160.94.179.252 | 139 | 172 | No |
| 8 | 206.163.37.95 | 11:14:38 | 160.94.179.251 | 139 | 285 | Yes |
| 9 | 206.163.37.95 | 11:14:41 | 160.94.179.250 | 139 | 195 | No |
| 10 | 206.163.37.95 | 11:14:44 | 160.94.179.249 | 139 | 163 | Yes |

Summarization of attacks using association rules

Rules Discovered:

{Src IP = 206.163.37.95, Dest Port = 139, Bytes ∈ [150, 200]} --> {ATTACK}

categorical temporal categorical continuous class

| Tid | SrcIP | Start time | Dest Port | Number of bytes | Attack |
|---|---|---|---|---|---|
| 1 | 206.163.37.81 | 11:17:51 | 160.94.179.208 | 150 | ? |
| 2 | 206.163.37.99 | 11:18:10 | 160.94.179.235 | 208 | ? |
| 3 | 206.163.37.55 | 11:34:35 | 160.94.179.221 | 195 | ? |
| 4 | 206.163.37.37 | 11:41:37 | 160.94.179.253 | 199 | ? |
| 5 | 206.163.37.41 | 11:55:19 | 160.94.179.244 | 181 | ? |

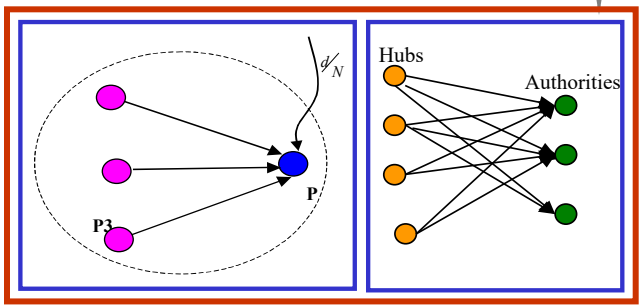
Live data

Learn Classifier

Model

Clustering & Anomaly Detection

Link Analysis



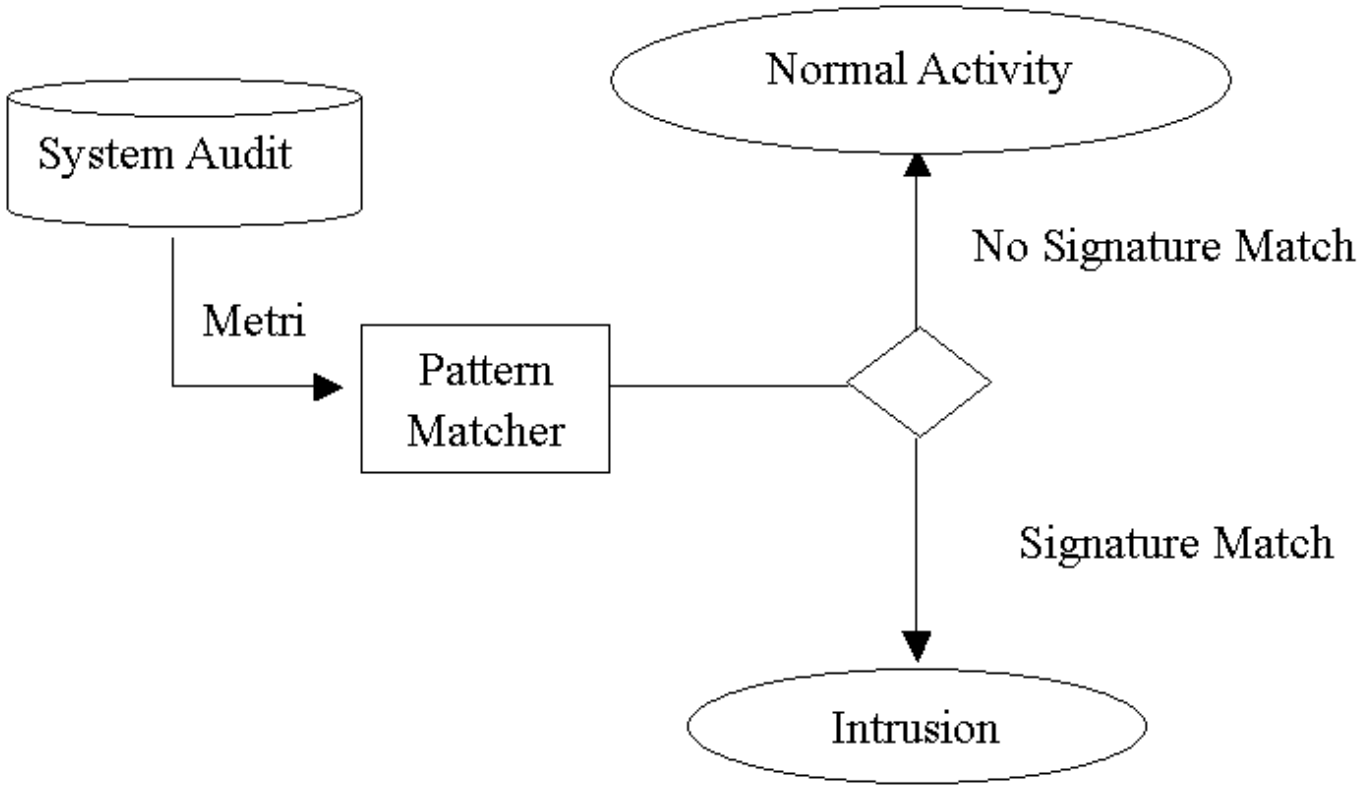

- Large data size
- High dimensionality
- Temporal nature of the data
- Skewed class distribution
- Data preprocessing
- On-line analysis

# Misuse Detection (Signature-Based)

- Set of rules defining a behavioral signature likely to be associated with attack of a certain type
  - Example: buffer overflow
    - A setuid program spawns a shell with certain arguments
    - A network packet has lots of NOPs in it
    - Very long argument to a string function
  - Example: SYN flooding (denial of service)
    - Large number of SYN packets without ACKs coming back

    …or is this simply a poor network connection?
- Attack signatures are usually very specific and may miss variants of known attacks
  - Why not make signatures more general?

A typical misuse detection system



# Pattern Matching example

```
  0  0050 dac6 f2d6 00b0 d04d cbaa 0800 4500   .P.......M....E.
 10  0157 3105 4000 8006 0000 0a0a 0231 d850   .W1.@........1.P
 20  1111 06a3 0050 df62 322e 413a 9cf1 5018   .....P.b2.A:..P.
 30  16d0 f6e5 0000 4745 5420 2f70 726f 6475   ......GET /produ
 40  6374 732f 7769 7265 6c65 7373 2f69 6d61   cts/wireless/ima
 50  6765 732f 686f 6d65 5f63 6f6c 6c61 6765   ges/home_collage
 60  322e 6a70 6720 4854 5450 2f31 2e31 0d0a   2.jpg HTTP/1.1..
 70  4163 6365 7074 3a20 2a2f 2a0d 0a52 6566   Accept: */*..Ref
 80  6572 6572 3a20 6874 7470 3a2f 2f77 7777   erer: http://www
 90  2e61 6d65 7269 7465 6368 2e63 6f6d 2f70   .ameritech.com/p
 a0  726f 6475 6374 732f 7769 7265 6c65 7373   roducts/wireless
 b0  2f73 746f 7265 2f0d 0a41 6363 6570 742d   /store/..Accept-
 c0  4c61 6e67 7561 6765 3a20 656e 2d75 730d   Language: en-us.
 d0  0a41 6363 6570 742d 456e 636f 6469 6e67   .Accept-Encoding
 e0  3a20 677a 6970 2c20 6465 666c 6174 650d   : gzip, deflate.
 f0  0a55 7365 722d 4167 656e 743a 204d 6f7a   .User-Agent: Moz
100  696c 6c61 2f34 2e30 2028 636f 6d70 6174   illa/4.0 (compat
110  6962 6c65 3b20 4d53 4945 2035 2e30 313b   ible; MSIE 5.01;
120  2057 696e 646f 7773 204e 5420 352e 3029    Windows NT 5.0)
130  0d0a 486f 7374 3a20 7777 772e 616d 6572   ..Host: www.amer
140  6974 6563 682e 636f 6d0d 0a43 6f6e 6e65   itech.com..Conne
150  6374 696f 6e3a 204b 6565 702d 416c 6976   ction: Keep-Aliv
160  650d 0a0d 0a                              e....
```

# Protocol Analysis

```
  0  0050 dac6 f2d6 00b0 d04d cbaa 0800 4500   .P.......M....E.
 10  0157 3105 4000 8006 0000 0a0a 0231 d850   .W1.@.........1.P
 20  1111 06a3 0050 df62 322e 413a 9cf1 5018   .....P.b2.A:..P.
 30  16d0 f6e5 0000 4745 5420 2f70 726f 6475   ......GET /produ
 40  6374 732f 7769 7265 6c65 7373 2f69 6d61   cts/wireless/ima
 50  6765 732f 686f 6d65 5f63 6f6c 6c61 6765   ges/home_collage
 60  322e 6a70 6720 4854 5450 2f31 2e31 0d0a   2.jpg HTTP/1.1..
 70  4163 6365 7074 3a20 2a2f 2a0d 0a52 6566   Accept: */*..Ref
 80  6572 6572 3a20 6874 7470 3a2f 2f77 7777   erer: http://www
 90  2e61 6d65 7269 7465 6368 2e63 6f6d 2f70   .ameritech.com/p
 a0  726f 6475 6374 732f 7769 7265 6c65 7373   roducts/wireless
 b0  2f73 746f 7265 2f0d 0a41 6363 6570 742d   /store/..Accept-
 c0  4c61 6e67 7561 6765 3a20 656e 2d75 730d   Language: en-us.
 d0  0a41 6363 6570 742d 456e 636f 6469 6e67   .Accept-Encoding
 e0  3a20 677a 6970 2c20 6465 666c 6174 650d   : gzip, deflate.
 f0  0a55 7365 722d 4167 656e 743a 204d 6f7a   .User-Agent: Moz
100  696c 6c61 2f34 2e30 2028 636f 6d70 6174   illa/4.0 (compat
110  6962 6c65 3b20 4d53 4945 2035 2e30 313b   ible; MSIE 5.01;
120  2057 696e 646f 7773 204e 5420 352e 3029    Windows NT 5.0)
130  0d0a 486f 7374 3a20 7777 772e 616d 6572   ..Host: www.amer
140  6974 6563 682e 636f 6d0d 0a43 6f6e 6e65   itech.com..Conne
150  6374 696f 6e3a 204b 6565 702d 416c 6976   ction: Keep-Aliv
160  650d 0a0d 0a                              e....
```

# An attack example

- Sendmail's vulnerability

  $ telnet mail.victim.com 25

  ***WIZ***

  ***shell***

  or

  ***DEBUG***

  #

  Get rootshell !

# Simple matching

- Check every packet：

  “WIZ”

  | “DEBUG”

# Check its port

- Reduce its checking range

  Port 25:{

  “WIZ”

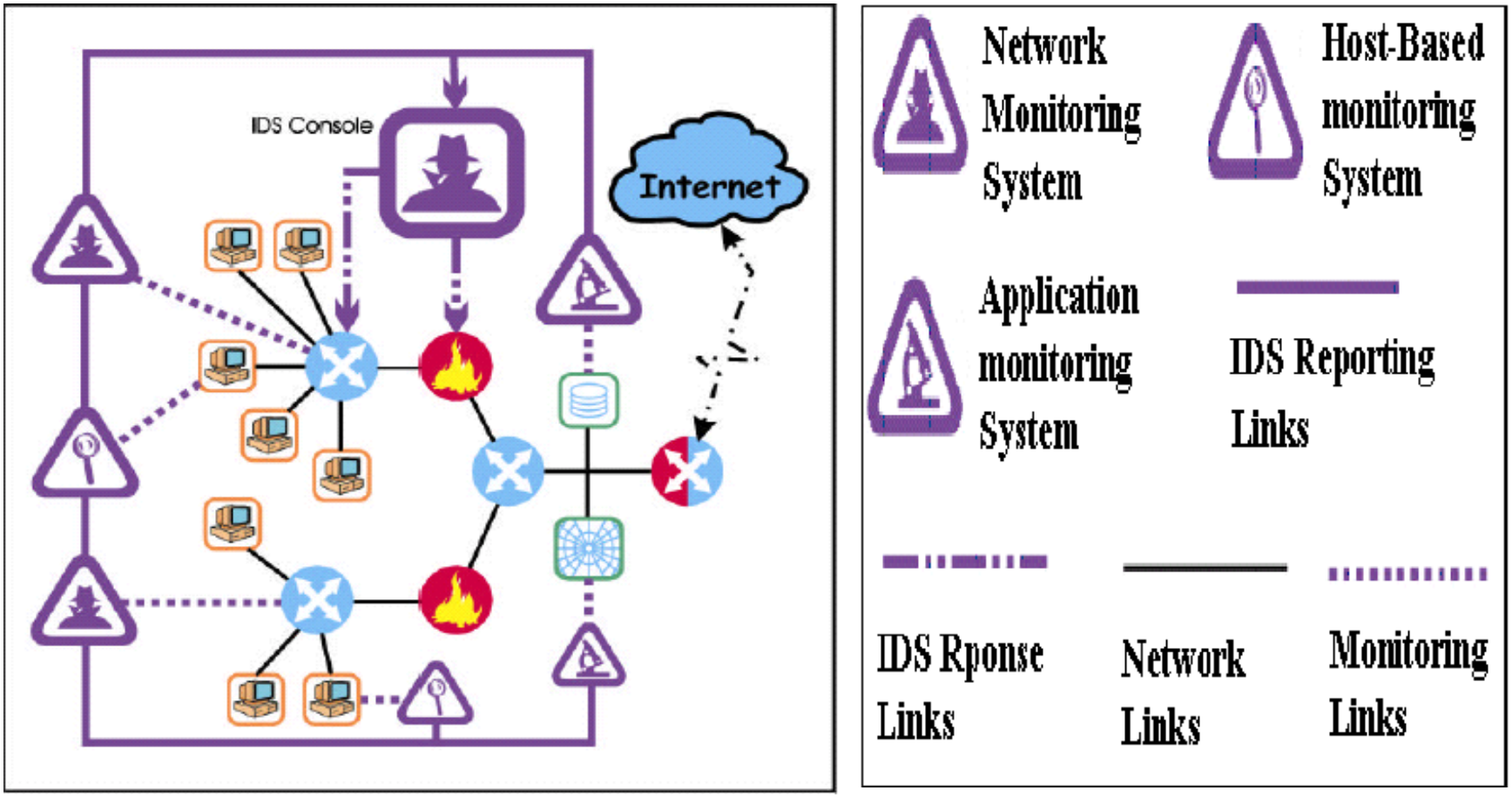
  | “DEBUG”

  }

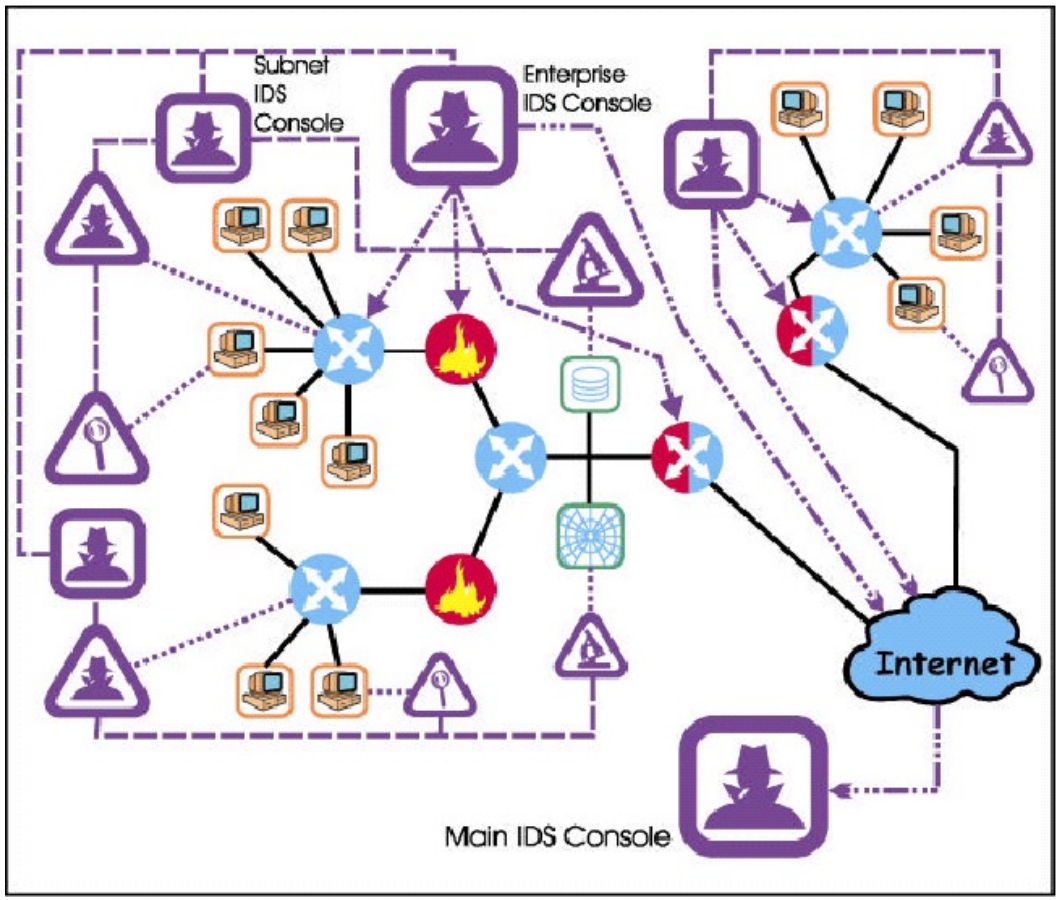
# Architecture

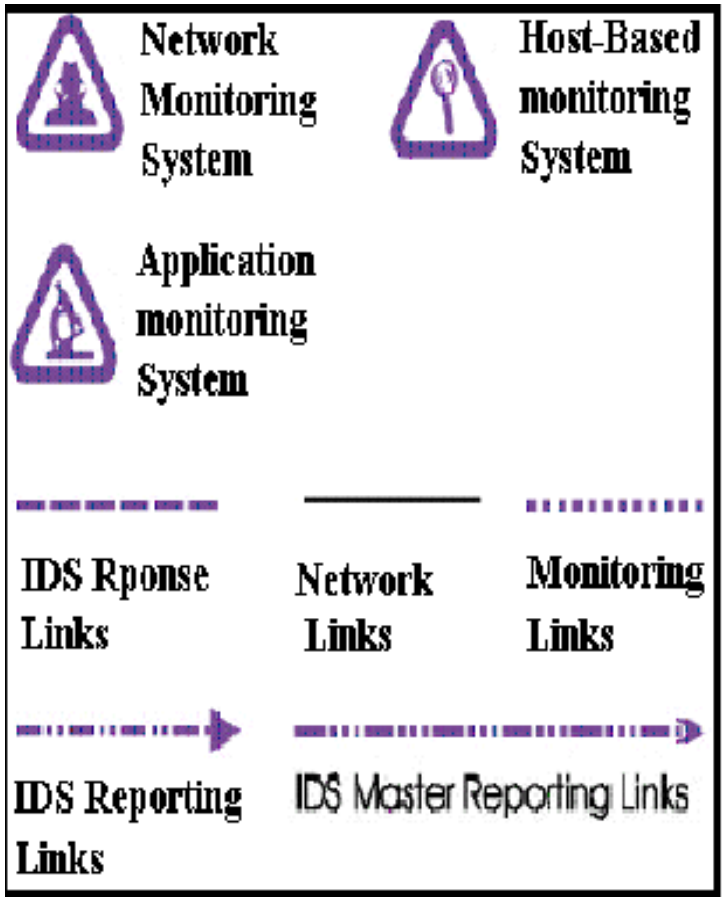
- **Components**
  - The Host: the system on which the IDS software runs.
  - The target: the system that the IDS is monitoring for problems.
- **Host-Target Co-location**
  - In early days of IDSs, most IDSs ran on the systems they protected.
- **Host-Target Separation**
  - This improved the security of the IDS as this made it much easier to hide the existence of the IDS from attackers.

# Control Strategy— Centralized

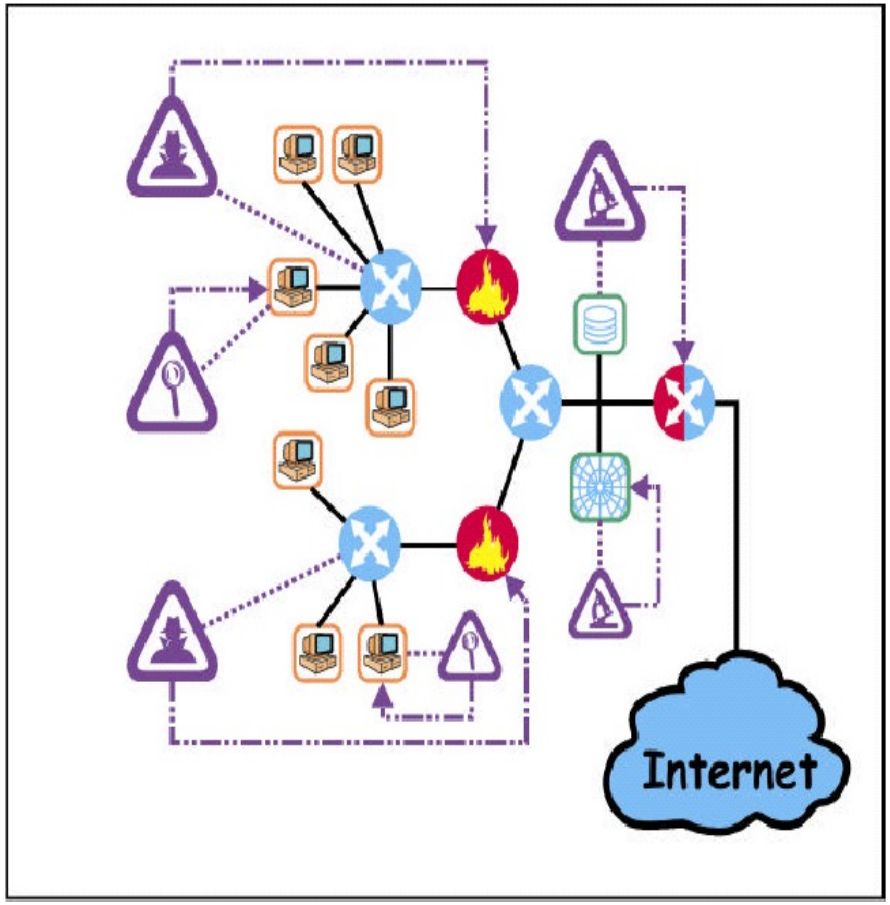


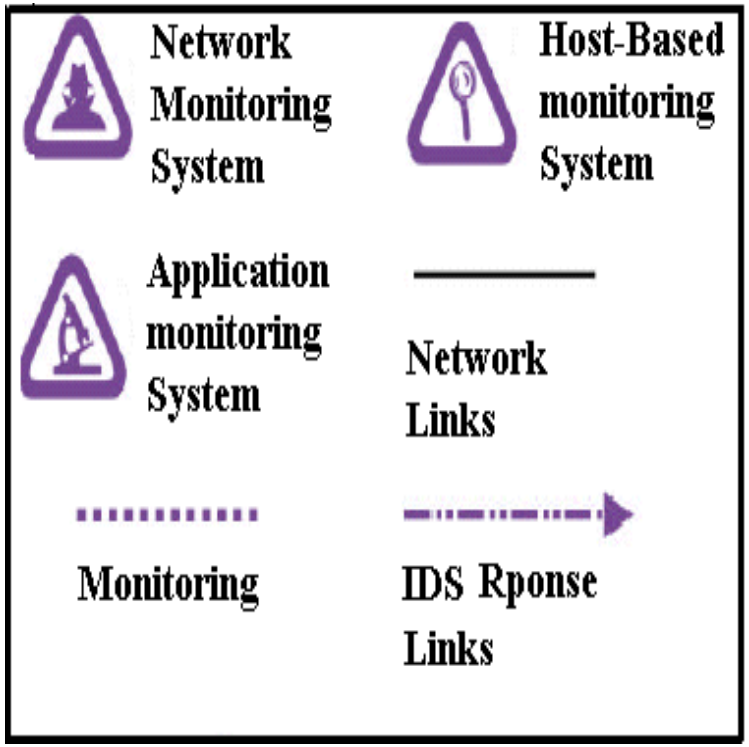
# Control Strategy — Partially Distributed



# Control Strategy — Fully Distributed

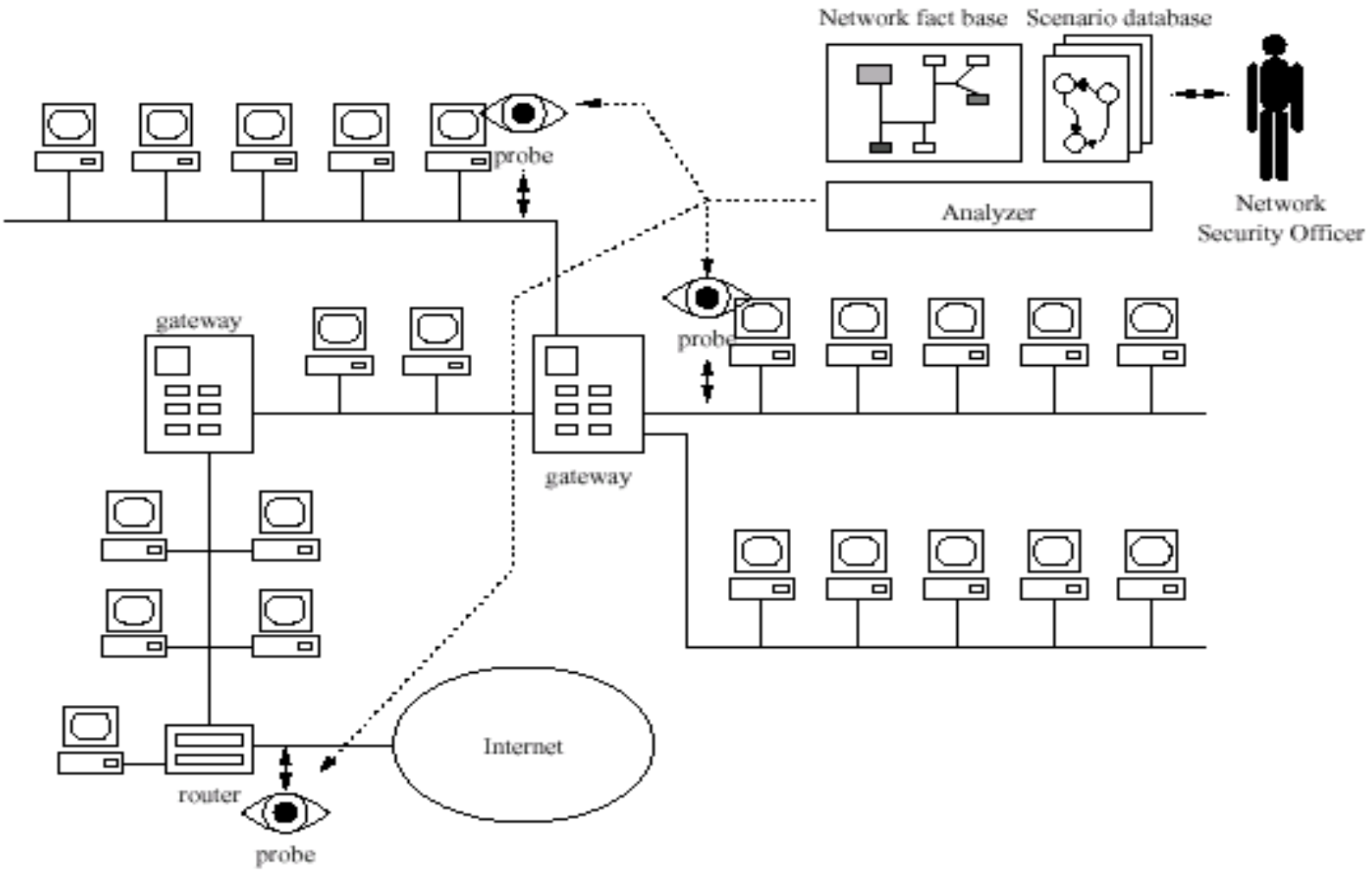


# Timing

- **Interval-Based**
  - In interval-based IDSs, the information flow from monitoring points to analysis engines is not continuous. In effect, the information is handled in a fashion similar to "store and forward" communications schemes.
  - Many early host-based IDSs used this timing scheme.
- **Real-Time**
  - Real-time IDSs operate on continuous information feeds from information sources.
  - Real-time IDS yields results quickly enough to allow the IDS to take action that affects the progress of the detected attack.

# How to deploy an IDS?



# Example:Snort

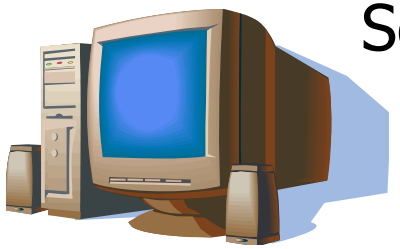
# Introducing Snort

- Snort is:
  - Small (~1.2M source distribution)
  - Portable (Linux, Solaris, *BSD, IRIX, HP-UX, WIN32)
  - Fast (High probability of detection for a given attack on "average" networks)
  - Configurable (Easy rules language, many reporting/logging options)
  - Free (GPL/Open Source Software)
  - Current version 1.8.1 as of Aug 2001

# Snort – Basic Configuration Modes

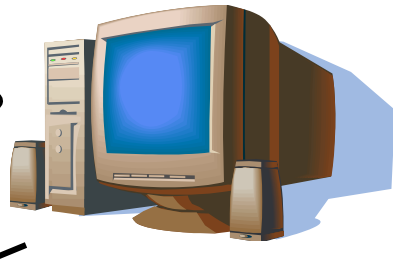
- Snort can be run in one of several configuration modes
  - Sniffer Mode – Snort reads packets off of the network and displays them on console
  - Packet Logger Mode – simply logs packets to disk
  - Network Intrusion Detection System (NIDS) mode – Snort grabs traffic from the network using libpcap, analyzes for matches to a defined rule set and generates alerts (as appropriate)
  - Inline Mode – obtains packet data from iptables (versus libpcap) and signals iptables to drop or pass packets using inline-specific rules

# A Basic Snort Architecture



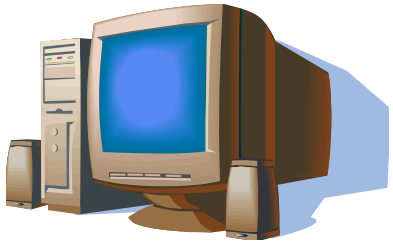
Sensor(s)

- Snort IDS
- Detect Events
- Forward Alerts



Server

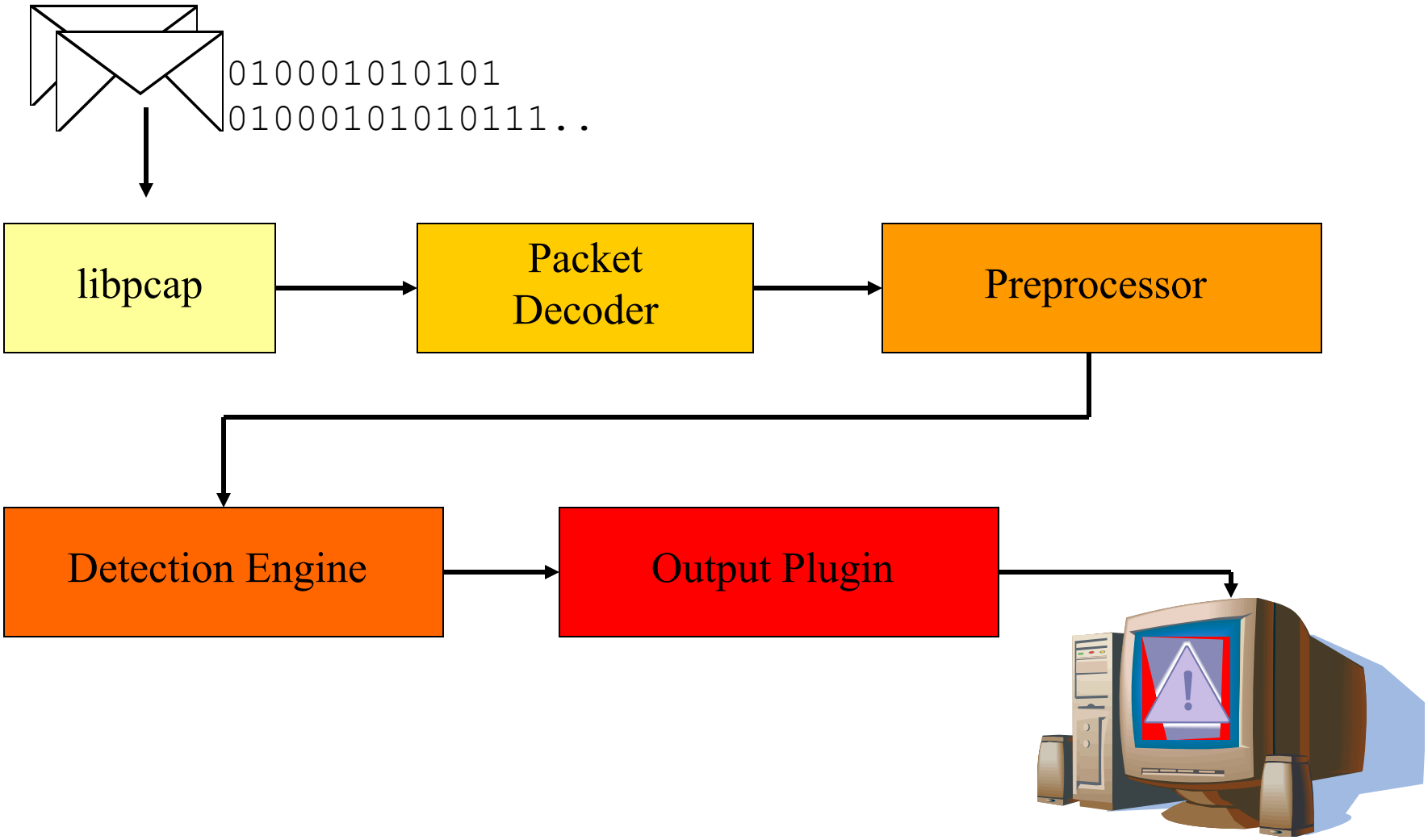
- MySQL, Apache Syslog
- Receives & Stores Alerts



Console

- Web Browser
- Displays Alerts

# Snort (Sensor) Technical Details



# Snort (Sensor) – Packet Decode

- libpcap
  - External Packet Capture Library (UNIX, Windows ports (winpcap))
  - Captures raw packets (required for Snort processing)
- Packet Decoder(s)
  - Series of Packet Decoders decode specific protocol elements of each packet (working up OSI Model)
  - As packets are decoded, decoded packet data is stored in a Snort data structure for analysis

# Snort (Sensor) – Preprocessors

- Preprocessor(s)
  - Perform a couple of functions
    - Examine suspicious packets (non-signature)
    - Manipulate packets to prepare for Detection Engine inspection (signature matching normalization)
  - Packets are passed through every Preprocessor
    - Ensures thorough packet inspection process
    - Guards against attacks designed to circumvent the IDS

- Key Preprocessors
  - Stream4
  - HTTP Inspect
  - RPC_Decode
  - Telnet_Decode
  - ARPSpoof
  - ASN1_Decode
  - Flow
  - SfPortscan
  - Performance Monitor

# Snort Tuning – Preprocessors

- Preprocessor options that are generally candidates for tuning include
  - Frag2
    - timeout = number of seconds to save inactive stream fragments in state table (default = 60, recommended = 65)
    - detect_state_problems = enables detection of overlapping fragments
    - ttl_limit number = specifies the maximum delta in TTL values that fragmented packets with the same fragment ID can have (default=7, recommended=8)

# Snort Tuning – Preprocessors (Cont)

- Stream4
  - detect_scans = detects normal TCP connect scans and stealthy scans (e.g. Half Open, and SYN-FIN scans)
  - timeout = number of seconds to keep an inactive stream in the state table (default = 30, recommended = 35)
- HTTP Inspect
  - iis_unicode_map = Unicode code point map (details code pages to use when decoding Unicode)
  - double_encode = detects double encoding attacks

# Snort Tuning – Preprocessors (Cont)

- ARPspoof
  - host IP host MAC – must specify list of hosts to be monitored via ARPspoof in snort.conf
- SfPortscan
  - sense_level = sensitivity levels for portscans (tune, as appropriate)
  - scan_type = types of port scans to detect (all, portscan, portsweep)
  - Note that the flow preprocessor is required for SfPortscan

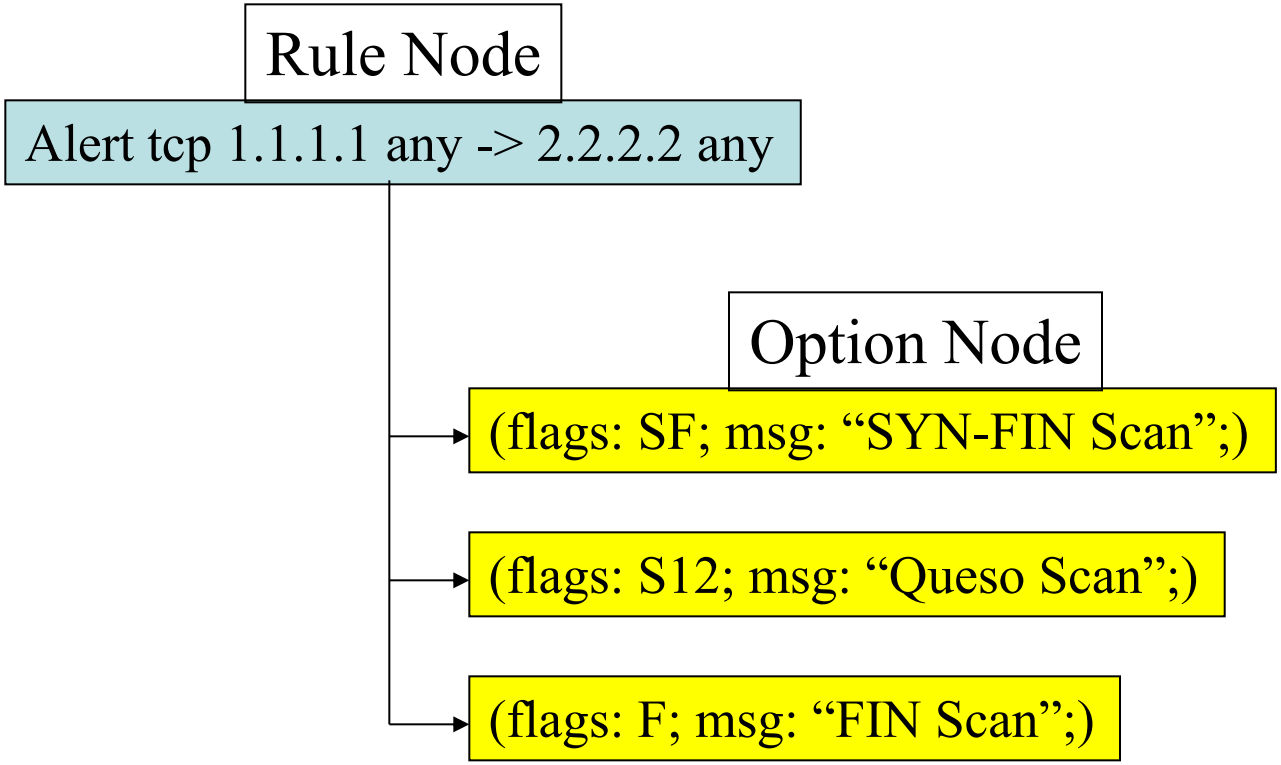
# Snort Sensor – Detection Engine

- Detection Engine
  - Performs Several Functions
    - Rule Parsing – rules are loaded into internal data structures, and guide packet inspection
    - Signature Detection – attack signatures are constructed by parsing Snort rules
  - Rules are divided into two sections
    - Rule Header – information that governs application of the signature (e.g. protocol, IP, etc.)
    - Rule Option – contains the attack signature, priority level, and attack information
  - Each packet is tested against increasingly specific signatures until there is a match (or the packet passes)

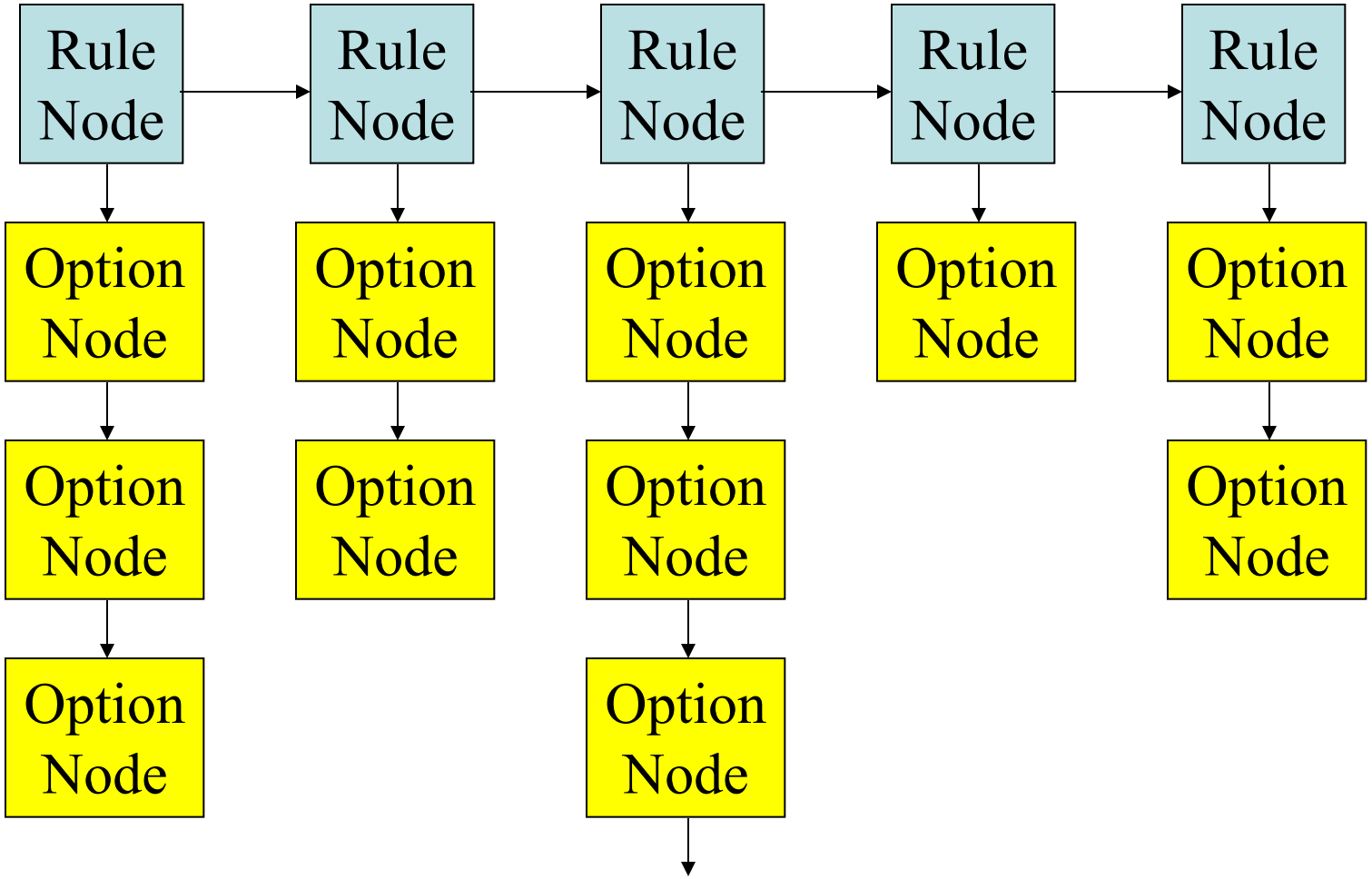
# Detection Engine: Rules

| Rule Header | Rule Options |
| --- | --- |
| Alert tcp 1.1.1.1 any -> 2.2.2.2 any | (flags: SF; msg: “SYN-FIN Scan”;) |
| Alert tcp 1.1.1.1 any -> 2.2.2.2 any | (flags: S12; msg: “Queso Scan”;) |
| Alert tcp 1.1.1.1 any -> 2.2.2.2 any | (flags: F; msg: “FIN Scan”;) |

# Detection Engine: Internal Representation



# Detection Engine: Fully Populated



# Snort Sensor – Output Plug-Ins

- Output plug-ins are invoked to generate intrusion data
  - Output plug-ins have the ability to generate alert data in various formats
  - Multiple Output plug-ins can be activated for different functions

- Key Output Plug-Ins
  - Alert_fast
  - Alert_full
  - Alert_smb
  - Alert_unixsock
  - Log_tcpdump
  - CSV
  - XML
  - Alert_syslog
  - Database
  - Unified

# Rule Format – Action

alert **tcp 10.1.1.1 any -> 10.1.1.2 80 (msg:"foo"; content:"bar";)**

- **Tells snort what the rule does**
  - **alert log pass 。 。 。**

# Rule Format – Custom Actions

- **not supported in product**

```
ruletype suspicious
{
    type log output
    log_tcpdump: suspicious.log
}

suspicious tcp 10.1.1.1 any -> 10.1.1.2 80 (msg:"foo";
   content:"bar";)
```

# Rule Format – Protocol

- **alert** tcp **10.1.1.1 any -> 10.1.1.2 80 (msg:"foo"; content:"bar";)**

- **Tells snort to look for a specific protocol**
- **Acceptable protocols:**
  - **TCP**
  - **UDP**
  - **ICMP**
  - **IP**

# Rule Format - IP Address

- **alert tcp** 10.1.1.1 **any ->** 10.1.1.2 **80 (msg:"foo"; content:"bar";)**
- **Examples**

  **10.1.1.1**

  **10.1.1.0/24**
    - **10.1.1.0 through 10.1.1.255**

  **!10.1.1.0/24**
    - **anything but 10.1.1.0 through 10.1.1.255**

  **[10.1.0.0/24,10.2.0.0./24]**
    - **10.1.0.0 through 10.1.0.255 or 10.2.0.0 through 10.2.0.255**

  **![10.1.0.0/24,10.2.0.0./24]**
    - **anything but 10.1.0.0 through 10.1.0.255 or 10.2.0.0 through 10.2.0.255**

# Rule Format - Port

- **alert tcp 10.1.1.1** any **-> 10.1.1.2** 80 **(msg:"foo"; content:"bar";)**
- **Examples:**

  **any**

  **80**

  **1:1023**
    - **1 through 1023 (inclusive)**

  **:1023**
    - **less than or equal to 1023**

  **10:**
    - **greater than or equal to 10**

  **!53**
    - **not 53**

  **!53:100**
    - **not 53 through 100 (inclusive)**

# Rule Format - Direction

- **alert tcp 10.1.1.1 any -> 10.1.1.2 80 (msg:"foo"; content:"bar";)**

**->**

- **From the first IP/Port to the second IP/Port**

**<>**

- **From either the first IP/Port to the second IP/Port**

  **OR**

- **From either the first IP/Port to the second IP/Port**

# Rule Format - variables

```
var EXTERNAL_NET any
var HTTP_PORTS 80
var SMTP_SERVERS 10.1.1.1

alert tcp $EXTERNAL_NET any ->
   $SMTP_SERVERS $HTTP_PORTS
```

# Rule Format – Body

- **alert tcp 10.1.1.1 any -> 10.1.1.2 80** (msg:"foo"; content:"bar";)

- **Meta-Data keywords**
- **Payload**
- **……..**

# Meta-Data keywords

- **Msg**
  - **msg:"my evil attack";**
- **Reference**
  - **reference:url,www.snort.org;**
- **sid**
  - **sid:100000;**
- **Rev**
  - **rev:100000;**
- **Priority**
  - **priority:3;**

# Payload

- **Content**
  - **content:"foo";**
- **Nocase**
  - **content:"foo"; nocase;**
- **Rawbytes**
  - **content:"foo"; rawbytes;**
- **Depth**
  - **content:"foo"; depth:10;**
- **Offset**
  - **content:"foo"; offset:10;**
- **Uricontent**
  - **uricontent:"foo";**

# Non-Payload options:

- **ack (TCP Acknowledge Number)**
  - **ack:0;**
- **dsize (Packet Size)**
  - **dsize:>10;**
- **id  (IP ID)**
  - **id:10;**
- **fragoffset (fragment offset)**
  - **fragoffset:0;**
- **fragbits (IP fragment bits)**
  - **fragbits:MD;**

# More non-payload options

- **ttl (IP Time To Live)**
  - **ttl:1;**
- **tos (IP type of service )**
  - **tos:30;**
- **ipopts (IP option)**
  - **ipopts:lsrr（loose source routing）;**
- **flags (TCP flags)**
  - **flags:SF;**
- **flow (TCP State)**
  - **flow:to_server,established;**

# Even more non-payload options:

- **seq (TCP Sequence Number)**
  - **seq:0;**
- **ttl (IP Time To Live)**
  - **ttl:10;**
- **window (TCP Window Size)**
  - **window:55808;**
- **itype (ICMP Type)**
  - **itype:8;**
- **icode (ICMP Code)**
  - **icode:0;**

# Even more non-payload options (again)

- **icmp_id (ICMP ID)**
  - **icmp_id:0;**
- **icmp_seq (ICMP Sequence Number)**
  - **icmp_seq:0;**
- **ip_proto (IP Protocol)**
  - **ip_proto:6;**
- **stateless (Not part of a flow)**
  - **stateless;**

# Snort Rules

- Rules which actually caught intrusions
  - **`alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 1433 (msg:"MS-SQL xp_cmdshell - program execution"; content: "x|00|p|00|_|00|c|00|m|00|d|00|s|00|h|00|e|00|l|00|l|00| "; nocase; flags:A+; classtype:attempted-user; sid:687; rev:3;)`** caught compromise of Microsoft SQL Server
  - **`alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS cmd.exe access"; flags: A+; content:"cmd.exe"; nocase; classtype:web-application-attack; sid:1002; rev:2;)`** caught Code Red infection
  - **`alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"INFO FTP \"MKD / \" possible warez site"; flags: A+; content:"MKD / "; nocase; depth: 6; classtype:misc-activity; sid:554; rev:3;)`** caught anonymous ftp server