Chapter 13
Intrusion Detect system

References

[1] Rebecca Bace, Peter Mell, Intrusion Detection Systems, NIST Special Publication on Intrusion Detection Systems;

[2] Dorothy E. Denning, An Intrusion-Detection Model, IEEE Trans. Software Engineering, Vol. SE-13, No. 2, 1987, pp222-232.

[3] Steven R. Snapp, et al. , DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype.

[4] Jelena Mirkovic, Janice Martin and Peter Reiher, A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms, Computer Science Department, University of California, Los Angeles, Technical report #020018.

[5] Giovanni Vigna and Richard A. Kemmerer, NetSTAT: A Network-based Intrusion Detection Approach, Department of Computer Science, University of California Santa Barbara.

Contents

1. Introduction
2. IDS Models
3. Major types of IDS
4. Deploying IDS
5. NetSTAT
6. BRO
7. Overview of DDOS detection
8. IP traceback
9. Strengths and Limitations of IDS

1. Introduction

- Understand what security goals intrusion detection mechanisms serve;
- How to select and configure intrusion detection systems for their specific system and network environments;
- How to manage the output of intrusion detection systems;
- How to integrate intrusion detection functions with the rest of the organizational security infrastructure.

What is a IDS

- Intrusions
  - caused by attackers accessing the systems from the Internet;
  - 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 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 Systems
  - software or hardware products that automate this monitoring and analysis process.

为什么需要IDS (1)

- **Preventing problems**
  - increasing the perceived risk of discovery
  - increasing punishment of attackers
  - affect the behavior of individual users

Why should IDS (2)

- **Detecting problems that are not prevented by other security measures**
  - The operating systems cannot be patched or updated.
  - Administrators sometimes have neither sufficient time nor resource to track and install all the necessary patches.
  - Both users and administrators make errors in configuring and using systems.
  - Discrepancies of an organization's procedural computer use policy.
  - Flaws and vulnerabilities are discovered on a daily basis.

Why should IDS (3)

- **Detecting the preambles to attacks**
  - **Adversaries attack a system typically in predictable stages.**
    - probing or examining a system.
    - searching for an optimal point of entry**.**
  - **With no IDS, the attacker is free to thoroughly examine the system with little risk of discovery or retribution.**
  - **Network with an IDS monitoring its operations presents a much more formidable challenge to that attacker.**
  - **IDS will observe the probes, will identify them as suspicious, may actively block the attacker's access, will alert security personnel.**

Why should IDS (4)

- **Documenting the existing threat**
  - IDS verify, itemize, and characterize the threat.
  - Assisting you in making sound decisions regarding your allocation of computer security resources.
  - The information from IDS regarding the source and nature of attacks, allows you to make decisions regarding security strategy.

Why should IDS (5)

- **Quality control for security design and administration**
  - Highlight flaws in the design and management of security for the system
  - Supports security management correcting those deficiencies before they cause an incident.
- **Providing useful information about actual intrusions**
  - Collect relevant, detailed, and trustworthy information about the attack
  - Supports incident handling and recovery efforts.

2. IDS Models

2.1 Intrusion Detection Expert System (IDES)

- Dorothy E. Denning, An Intrusion-Detection Model, IEEE Trans. Software Engineering, Vol. SE-13, No. 2, 1987, pp222-232.

Hypothesis

- exploitation of a system's vulnerabilities involves abnormal use, of the system;
- therefore, security violations could be detected from abnormal patterns of system usage.

Examples(1)

- *Attempted break-in*
  - Someone attempting to break into a system might generate an abnormally high rate of password failures with respect to a single account or the system as a whole.
- Masquerading or successful break-in
  - Someone logging into a system through an unauthorized account and password might have a different login time, location, or connection type from that of the account's legitimate user.
  - He might spend most of his time browsing through directories and executing system status commands

    The legitimate user might concentrate on editing or compiling and linking programs.

Examples(2)

- Penetration by legitimate user
  - A user attempting to penetrate the security mechanisms in the operating system might execute different programs or trigger more protection violations from attempts to access unauthorized files or programs. If his attempt succeeds, he will have access to commands and files not normally permitted to him.
- Leakage by legitimate user
  - A user trying to leak sensitive documents might log into the system at unusual times or route data to remote printers not normally used.

Examples(3)

- **Inference by legitimate user**
  - A user attempting to obtain unauthorized data from a database through aggregation and inference might retrieve more records than usual.
- **Trojan horse**
  - The behavior of a Trojan horse planted in or substituted for a program may differ from the legitimate program in terms of its CPU time or 1/0 activity.
- **Denial-of-Service**
  - An intruder able to monopolize a resource (e.g., network) might have abnormally high activity with respect to the resource, while activity for all other users is abnormally low.

- The model is independent of any particular
  - System
  - application environment
  - system vulnerability
  - type of intrusion

Six main components

- Subjects
  - Initiators of activity on a target system
- Objects
  - Resources managed by the system-files, commands, devices
- Audit records
  - Generated by the target system in response to actions performed or attempted by subjects on objects-user login, command execution, file access, etc.
- Profiles
  - Structures that characterize the behavior of subjects with respect to objects in terms of statistical metrics and models of observed activity. Profiles are automatically generated and initialized from templates.
- Anomaly records
  - Generated when abnormal behavior is detected.
- Activity rules
  - Actions taken when some condition is satisfied, which update profiles, detect abnormal behavior, relate anomalies to suspected intrusions, and produce reports.

I. Subjects

I. *Subjects* are the initiators of actions
II. A terminal user
III. A process acting on behalf of users or groups of users
IV. The system itself

V. Subjects may be grouped into different classes (e.g., user groups) for the purpose of controlling access to objects in the system.

II. Objects

- the receptors of actions
  - Files
  - Programs
  - Messages
  - Records
  - Terminals
  - Printers
  - user- or program-created structures

III. AUDIT RECORDS

- *Audit Records* are 6-tuples representing actions performed by subjects on objects:

1. <Subject,
2. Action,
3. Object,
4. Exception-Condition,
5. Resource-Usage,
6. Time-stamp >

IDES — Six main components

- *Action*
  - Operation performed by the subject on or with the object, e.g., login, logout, read, execute.
- *Exception-Condition*
  - Denotes which, if any, exception condition is raised on the return. This should be the actual exception condition raised by the system, not just the apparent exception condition returned to the subject.
- *Resource-Usage*
  - List of quantitative elements, where each element gives the amount used of some resource, e.g., number of lines or pages printed, number of records read or written, CPU time or 1/0 units used, session elapsed time.
- *Time-stamp*
  - Unique time/date stamp identifying when the action took place.

IV. PROFILES

- ***An activity profile*** characterizes the behavior of a given subject with respect to a given object, thereby serving as a *signature* or description of normal activity for its respective subject and object.
- Observed behavior is characterized in terms of a statistical metric and model.
- A metric is a random variable *x* representing a quantitative measure accumulated over a period.
  - a fixed interval of time
  - the time between two audit-related events
- Observations x_i *of* x obtained from the audit records are used together with a statistical model to determine whether a new observation is abnormal.

- ***Event Counter***
  - *x* is the number of audit records satisfying some property occurring during a period.
  - number of logins during an hour, number of times some command is executed during a login session, and number of password failures during a minute.
- ***Interval Timer***
  - *x* is the length of time between two related events; i.e., the difference between the time-stamps in the respective audit records.
  - the length of time between successive logins into an account.

- **Resource Measure**
  - x is the quantity of resources consumed by some action during a period.
  - the total number of pages printed by a user per day
  - total amount of CPU time consumed by some program during a single execution

Statistical Models

- Given a metric for a random variable x and n observations $x_1, \ldots, x_n$
- A statistical model of *x* is to determine whether a new observation x_{n+1} is abnormal with respect to the previous observations.

Operational Model

- operational assumption
  - abnormality can be decided by comparing a new observation of x against fixed limits.
  - applicable to metrics where experience has shown that certain values are frequently linked with intrusions.
  - an event counter for the number of password failures during a brief period, where more than 10, say, suggests an attempted break-in.

Mean and Standard Deviation Model

- This model is based on the assumption that all we know about x_1, … x_n, are mean and standard deviation as determined from its first two moments:
  - sum = x_1 + … + x_n
  - *sumsquares* = x_1^2 + ··· + x_n^2
  - *mean = sum/ n*
  - *stdev = sqrt (sumsquares / (n+1) – mean2)*
  - *mean + d * stdev*

Markov Process Model

- Applies only to event counters
- Regards each distinct type of event as a state variable
- Uses a state transition matrix to characterize the transition frequencies between states
- A new observation is defined to be abnormal if its probability as determined by the previous state and the transition matrix is too low.
- This model might be useful for looking at transitions between certain commands where command sequences were important.

Time Series Model

- This model, which uses an interval timer together with an event counter or resource measure, takes into account the order and interarrival times of the observations $x_1, \ldots, x_n$, as well as their values.
- A new observation is abnormal if its probability of occurring at that time is too low.
- A time series has the advantage of measuring trends of behavior over time and detecting gradual but significant shifts in behavior,
- The disadvantage of being more costly than mean and standard deviation.

Profile Structure

< Variable-Name,
Action-Pattern,
Exception-Pattern,
Resource-Usage-Pattern,
Period,
Variable-Type,
Threshold,
Subject-Pattern,
Object-Pattern,
Value
>

V. ANOMALY RECORDS

- **<Event, Time-stamp, Profile>**
- *Event:* indicates the event giving rise to the abnormality and is either "audit," meaning the data in an audit record was found abnormal, or "period," meaning the data accumulated over the current interval was found abnormal.
- *Time-stamp:* either the time-stamp in the audit record or interval stop time (since we assume that audit records have unique time-stamps, this provides a means of tying an anomaly back to an audit record).
- *Profile:* activity profile with respect to which the abnormality was detected (rather than including the complete profile, IDES might include a "key" field, which identifies the profile in the database, and the current state of the Value field).

VI. ACTIVITY RULES

- *Audit-record rule*
- *Periodic-activity-update* rule
- *Anomaly-record rules*
- *Periodic-anomaly-analysis* rule

Audit-record rule

Condition: new Audit.Record
Audit.Record matches Profile
Profile.Variable-Type = *t*

Body: AuditProcess(Audit-Record, Profile);

END

Periodic-activity-update rule

Condition: Clock mod p = 0

Profile.Period = *p*

Profile.Variable-Type = *t*

Body: PeriodProcesst(Clock, Profile);

END

Anomaly-record rules

Condition: new Anomaly-Record
Anomaly-Record.Profile matches profile-pattern
Anomaly-Record.Event matches event-pattern

Body: PrintAlert('Suspect intrusion of type …',
Anomaly-record);

END

Periodic-anomaly-analysis rule

Condition: Clock mod p = 0

Body: Start = Clock - *p;*

A = SELECT FROM Anomaly-Records

WHERE Anomaly-Record.Time-stamp > Start;

generate summary report of *A;*

END

2.2 Distributed Intrusion Detection System (DIDS)

Steven R. Snapp, et al. ,

DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype.

- *Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur* **DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype.**,

- The DIDS project is sponsored by the United States Air Force Cryptologic Support Center through a contract with the Lawrence Livermore National Labs.

Architecture of DIDS

- The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis.
- The components of DIDS are
  - a single *host monitor* per host
  - and a single *LAN monitor* for each broadcast LAN segment in the monitored network.
  - the *DIDS director*
- The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity
- The DIDS director is primarily responsible for its evaluation

The host monitor

- Host event generator (HEG)
  - Collects and analyzes audit records from the host's operating system.
  - The audit records are scanned for notable events.
  - These notable events are then sent to the director for further analysis. In enhancements under development,
  - Track user sessions and report anomalous behavior aggregated over time through user/group profiles and the integration of Haystack into DIDS.
- The host agent
  - handles all communications between the host monitor and the DIDS director

LAN monitor

- *LAN event generator* (LEG)
  - LAN monitor observes each and every packet on its segment of the LAN and, from these packets, it is able to construct higher-level objects such as connections (logical circuits), and service requests using the TCP/IP or UDP/IP protocols.
  - It audits host-to-host connections, services used, and volume of traffic per connection.
  - The LAN monitor reports on such network activity as *rlogin* and *telnet* connections, the use of security-related services, and changes in network traffic patterns.
- *LAN agent*

LAN monitor (cnt.)

- the LAN monitor uses several simple analysis techniques to identify significant events.
  - The events include the use of certain services as well as activity by certain classes of hosts
  - the LAN monitor uses and maintains profiles of expected network behavior.
- The LAN monitor also uses heuristics in an attempt to identify the likelihood that a particular connection represents intrusive behavior.
  - the security level for each machine on the network
  - signatures of past attacks.
  - each of the network services,
  - the level of authentication required for each of the services,

The DIDS director

- ***Communications manager***
  - It is responsible for the transfer of data between the director and each of the host and the LAN monitors.
- ***Expert system***
  - It is responsible for evaluating and reporting on the security state of the monitored system;
  - It receives the reports from the host and the LAN monitors;
  - Based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole.
- ***User interface***
  - the System Security Officer (SSO) interactive access to the entire system.
  - The SSO is able to watch activities on each host, watch network traffic, and request more specific types of information from the monitors.

Intrusion Detection Model (IDM).

I. objects
II. *event*
III. *subject*
IV. *context*
V. *threats*
VI. *security state*

Objects

- The audit records provided by the host operating system, by the LAN monitor, or by a third party auditing package.
- The objects at this level are both syntactically and semantically dependent on the source.
- All of the activity on the host or LAN is represented.

Event

- Generated by the HEG based on host audit records;
  - Certain critical audit records are always passed directly to the expert system;
  - others are processed locally by the host monitor and only summary reports are sent to the expert system;
  - In order to push as much of the processing operations down to the low-level monitors as possible, the HEG creates a more abstract object called an **event**.
- An event reported by a host monitor is called a host audit record (har). The record syntax is

  har (Monitor-ID, Host-ID, Audit-UID, Real-UID, Effective-UID, Time, Domain, Action, Transaction, Object, Parent Process, PID, Return Value, Error Code).

Event (cnt.)

- The events include the use of certain services (e.g., *rlogin* and *telnet*) as well as activity by certain classes of hosts (e.g., a PC without a host monitor).
- An event reported by a LAN monitor is called a network audit record (nar).
  - nar(Monitor-ID, Source_Host, Dest_Host, Time, Service, Domain, Status).

Subject

- This introduces a single identification for a user across many hosts on the network.
- **Network-user Identification (NID)**
  - an intruder may use several different accounts on different machines during the course of an attack.
  - a single intruder may uses multiple accounts to launch an attack, and that the behavior can be recognized as suspicious only if one knows that all of the activity emanates from a single source.
  - For example, it is not particularly noteworthy if a user inquires about who is using a particular computer
  - However, it may be indicative of an attack if a user inquires about who is using each of the computers on a LAN and then subsequently logs into one of the hosts.

- Upper layers of the model treat the network-user as a single entity, essentially ignoring the local identification on each host.

Context

- Temporal context
  - **external temporal context:** behavior which is unremarkable during standard working hours may be highly suspicious during off hours
  - **temporal proximity:** a call to the UNIX ***who*** command followed closely by a ***login*** or ***logout*** is more likely to be related to an intrusion than either of those events occurring alone. Spatial context implies
- Spatial context
  - the relative importance of the source of events
  - events related to a particular user, or events from a particular host, may be more likely to represent an intrusion than similar events from a different source;
  - a user moving from a low-security machine to a high-security machine may be of greater concern than a user moving in the opposite direction.
  - multiple events are more noteworthy when they have a common element than when they do not.

Threats

- The threats are partitioned by the nature of the **abuse** (what is the intruder doing) and the **nature of the target (** and what is he doing it to)
- Abuses
  - *Attacks:* the state of the machine is changed.
  - *Misuses:* out-of-policy behavior but the state of the machine not affected
  - *suspicious acts*

Security state

- The model produces a numeric value between one and 100 which represents the overall *security state* of the network.
- The higher the number the less secure the network.

3. Major types of IDS

Rebecca Bace, Peter Mell,

Intrusion Detection Systems,

NIST Special Publication on Intrusion Detection Systems;

- **Process model for Intrusion Detection**
  - *Information Sources*
    - network, host, and application monitoring
  - *Analysis*
    - misuse detection and anomaly detection
  - *Response*
    - active measures: automated intervention
    - passive measures: reporting IDS findings to humans, who are then expected to take action based on those reports.

3.1 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.

3.2 Goals

- **Accountability**
  - To link a given activity or event back to the party responsible for initiating it.
  - bring criminal charges against an attacker.
  - Accountability is difficult in in any system that employs weak identification and authentication mechanisms.
- **Response**
  - to recognize a given activity or event as an attack and then taking action to block or otherwise affect its ultimate goal.

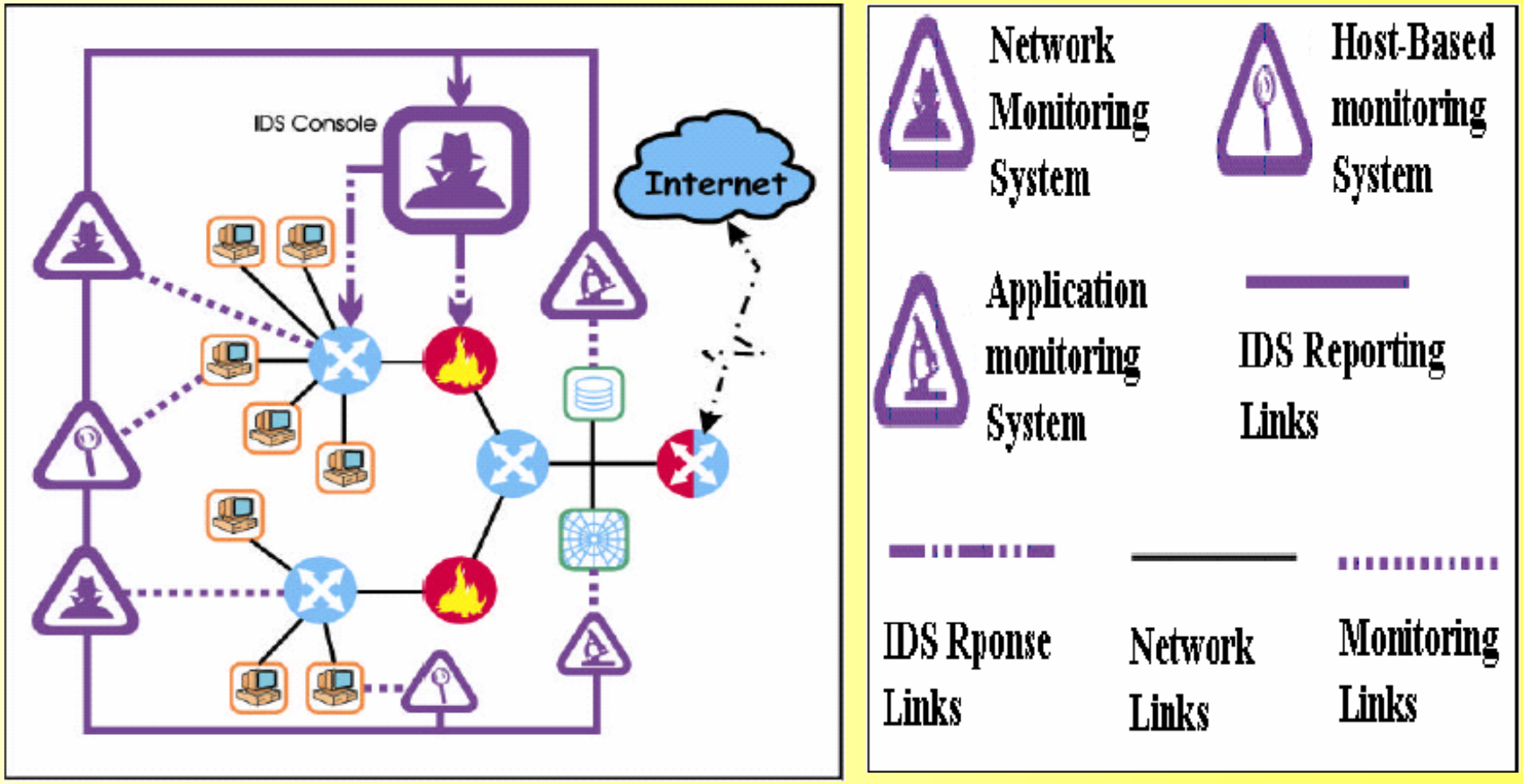
3.3 Control Strategy

How the elements of an IDS is controlled

How the input and output of the IDS is managed.

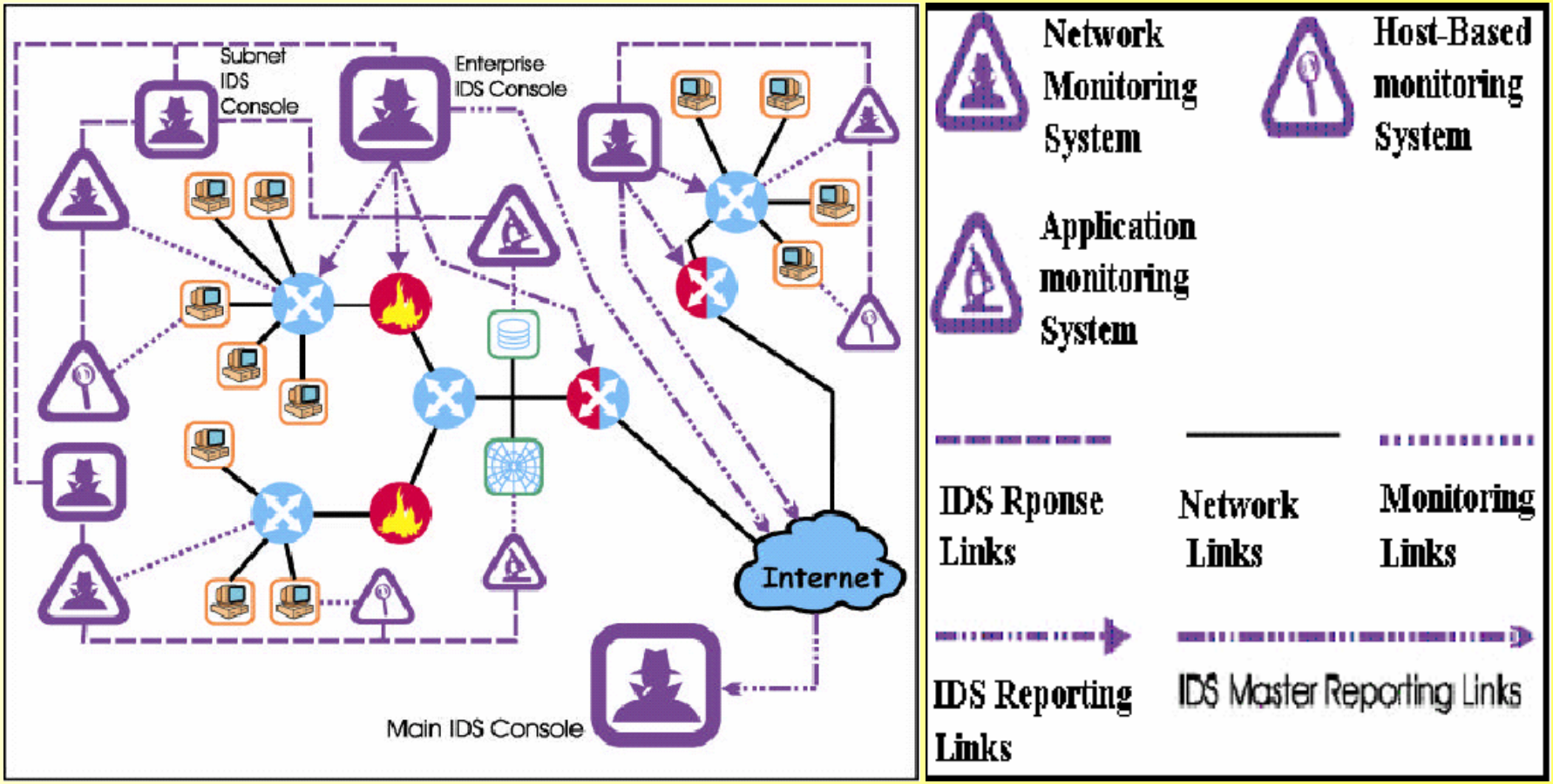
- **Centralized**
  - **All monitoring, detection and reporting is controlled directly from a central location**
- **Partially Distributed**
  - **Monitoring and detection is controlled from a local control node, with hierarchical reporting to one or more central location.**
- **Fully Distributed**
  - **Monitoring and detection is done using an agent-based approach, where response decisions are made at the point of analysis.**

3.3 Control Strategy— Centralized



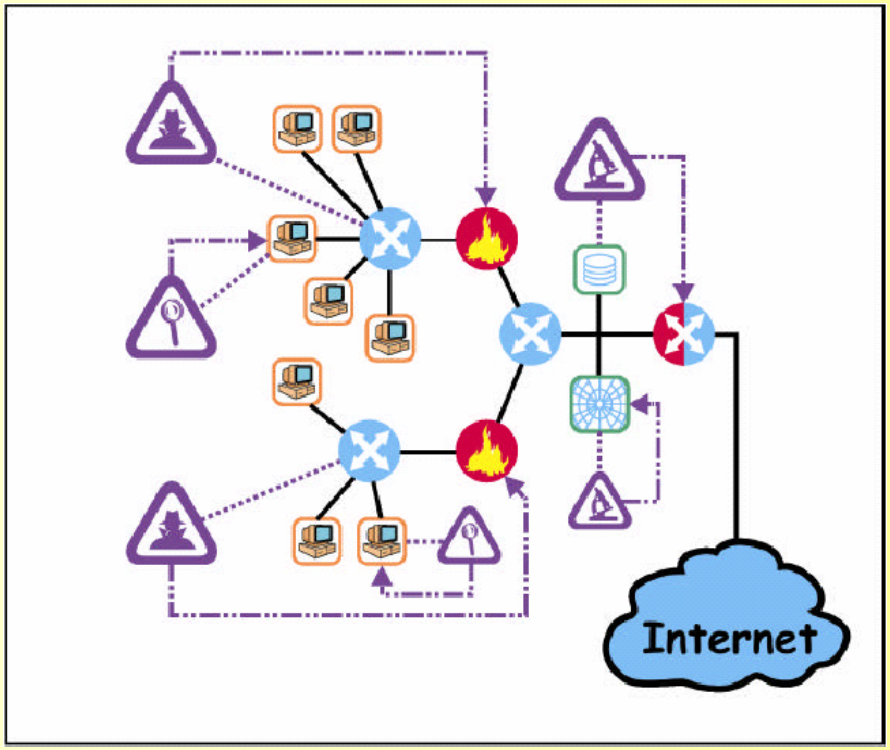
Under centralized control strategies, all monitoring, detection and reporting is controlled directly from a central location

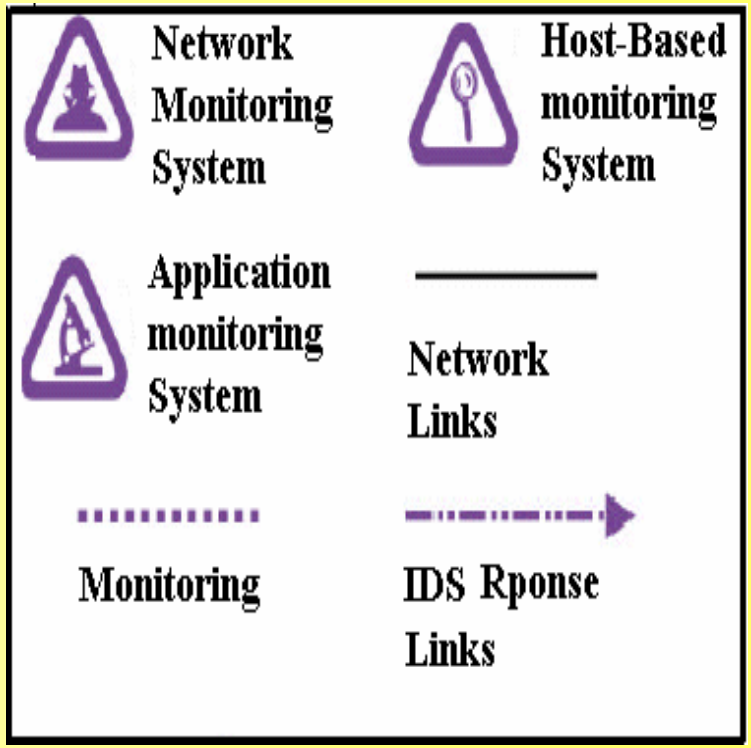
3.3 Control Strategy — Partially Distributed



Monitoring and detection is controlled from a local control node, with hierarchical reporting to one or more central location(s).

3.3 Control Strategy — Fully Distributed





Monitoring and detection is done using an agent-based approach, where response decisions are made at the point of analysis.

3.4 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.
  - Interval based IDSs are precluded from performing active responses.
- **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.

3.5 Information Sources — 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.

3.5 Information Sources — NIDS

- ***Advantages of Network-Based IDSs***
  - A few well-placed network-based IDSs can monitor a large network.
  - The deployment of network-based IDSs has little impact upon an existing network.
- ***Disadvantages of Network-Based IDS***
  - Network-based IDSs may have difficulty processing all packets in a large or busy network
  - Most switches do not provide universal monitoring ports
  - Network-based IDSs cannot analyze encrypted information.
  - Most network-based IDSs cannot tell whether or not an attack was successful;

3.5 Information Sources — 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.

3.5 Information Sources — 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.

3.5 Information Sources — 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.

3.5 Information Sources — 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.

3.5 Information Sources — AIDS

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

3.5 Information Sources — 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.
  - it is advisable to use an Application-based IDS in combination with Host-based and/or Network-based IDSs.

3.6 IDS Analysis

signature-based detection

- Misuse detectors
  - analyze system activity
  - looking for events or sets of events that match a predefined pattern of events that describe a known attack.
- Advantages
  - Misuse detectors are very effective at detecting attacks without generating an overwhelming number of false alarms.
- Disadvantages
  - they must be constantly updated with signatures of new attacks.

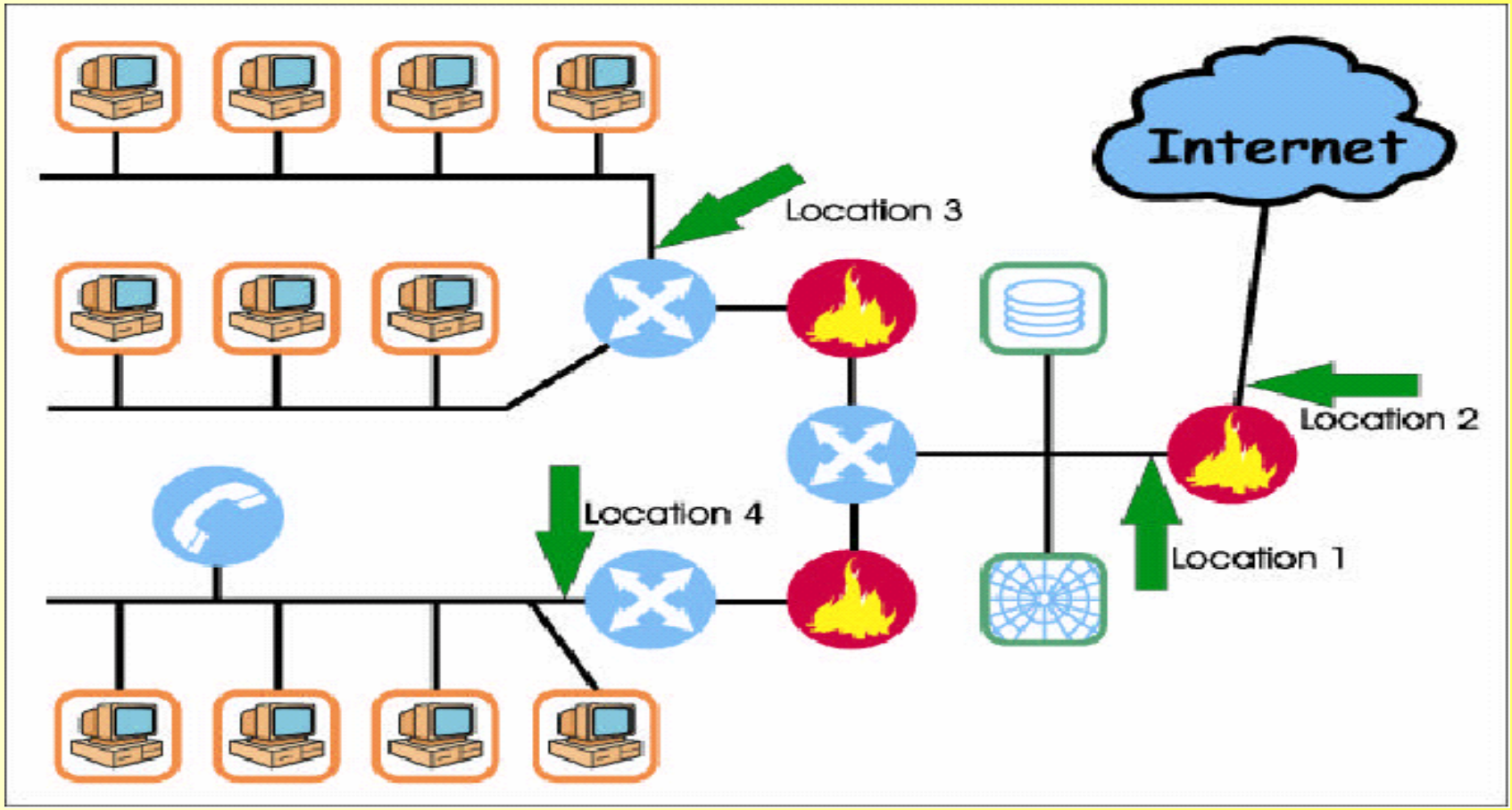
3.6 IDS Analysis

- **Anomaly Detection**
  - Anomaly detectors identify abnormal unusual behavior (anomalies) on a host or network.
  - Anomaly detectors construct profiles representing normal behavior
  - These profiles are constructed from historical data collected over a period of normal operation.

4. Deploying IDS

- Background
  - the deficiencies of today's intrusion detection products
  - the limited security skill level of many system administrators
- Requirement
  - careful planning
  - preparation
  - Prototyping
  - Testing
  - specialized training
  - compatible with the organization's network infrastructure, policies, and resource level.

Locations of Network-based IDS sensors



- **Location 1: Behind each external firewall**
  - Highlights problems with the network firewall policy or performance.
  - Even if the incoming attack is not recognized, the IDS can sometimes recognize the outgoing traffic that results from the compromised server.
- **Location 2: Outside an external firewall**
  - Documents number of attacks originating on the Internet that target the network.
  - Documents types of attacks originating on the Internet that target the network

- **Location 3: On major network backbones**
  - Monitors a large amount of a network's traffic, thus increasing the possibility of spotting attacks.
  - Detects unauthorized activity by authorized users within the organization's security perimeter.
- **Location 4: On critical subnets**
  - Detects attacks targeting critical systems and resources.
  - Allows focusing of limited resources to the network assets considered of greatest value.

- **Deploying Host-Based IDSs**
  - Installing host-based IDSs on every host in the enterprise can be extremely time-consuming, as each IDS has to be installed and configured for each specific host.
  - organizations first install host-based IDSs on critical servers.
  - Once the operation of host-based IDS is routine, more security-conscious organizations may consider installing host-based IDS on the majority of their hosts.
  - host-based systems that have centralized management and reporting functions.

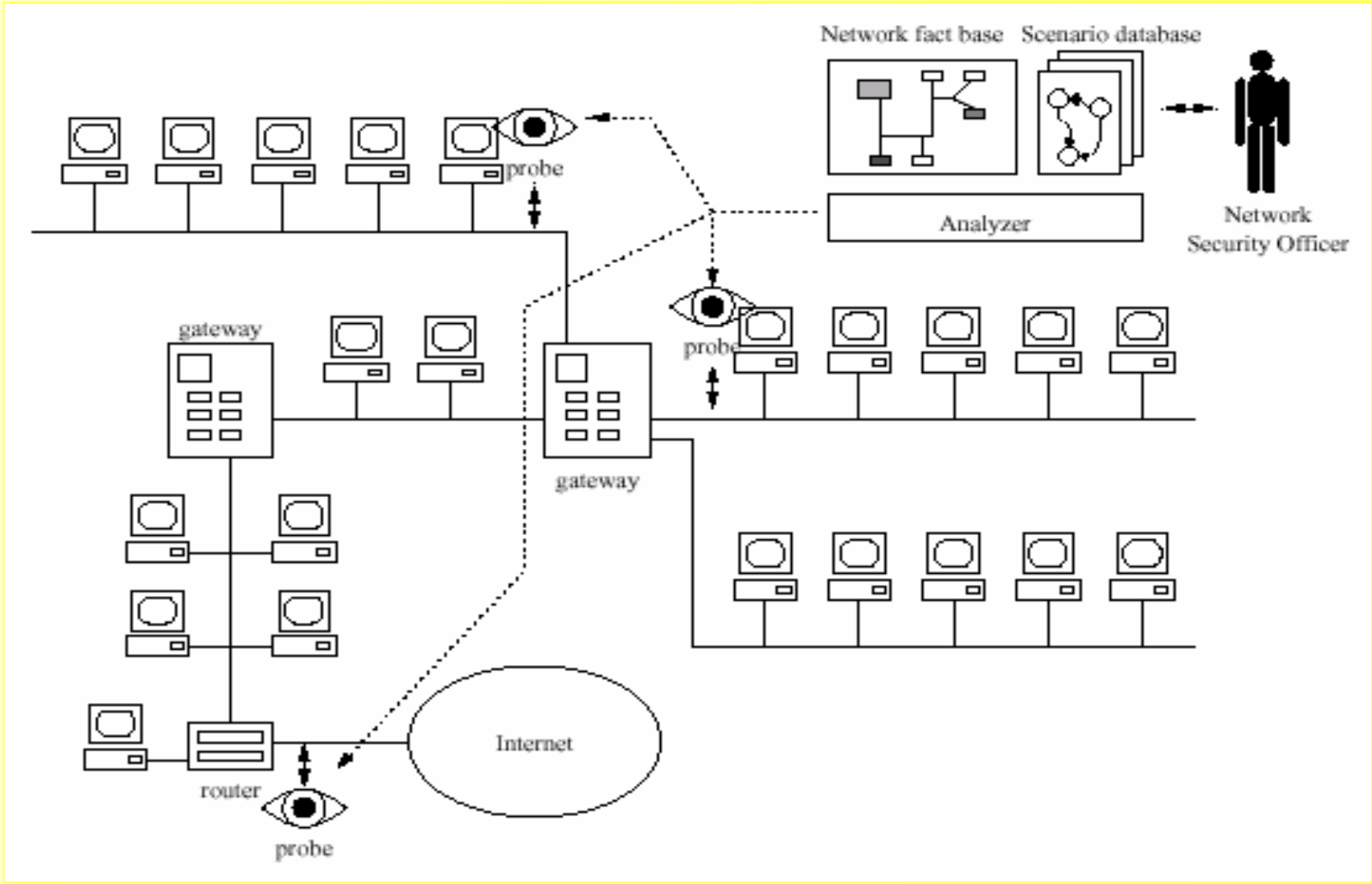
5. NetSTAT

NetSTAT: A Network-based Intrusion Detection Approach

Giovanni Vigna and Richard A. Kemmerer

University of California Santa Barbara

5.1 NetSTAT Architecture



NetSTAT Architecture

- Network Fact Base
- State Transition Scenario Database
- Probes
- Analyzer

5.2 Network Fact Base

- The network fact base component stores and manages the security relevant information about a network.
- The fact base is a stand-alone application that is used by the Network Security Officer to construct, insert, and browse the data about the network being protected.
- It contains information about the network topology and the network services provided.

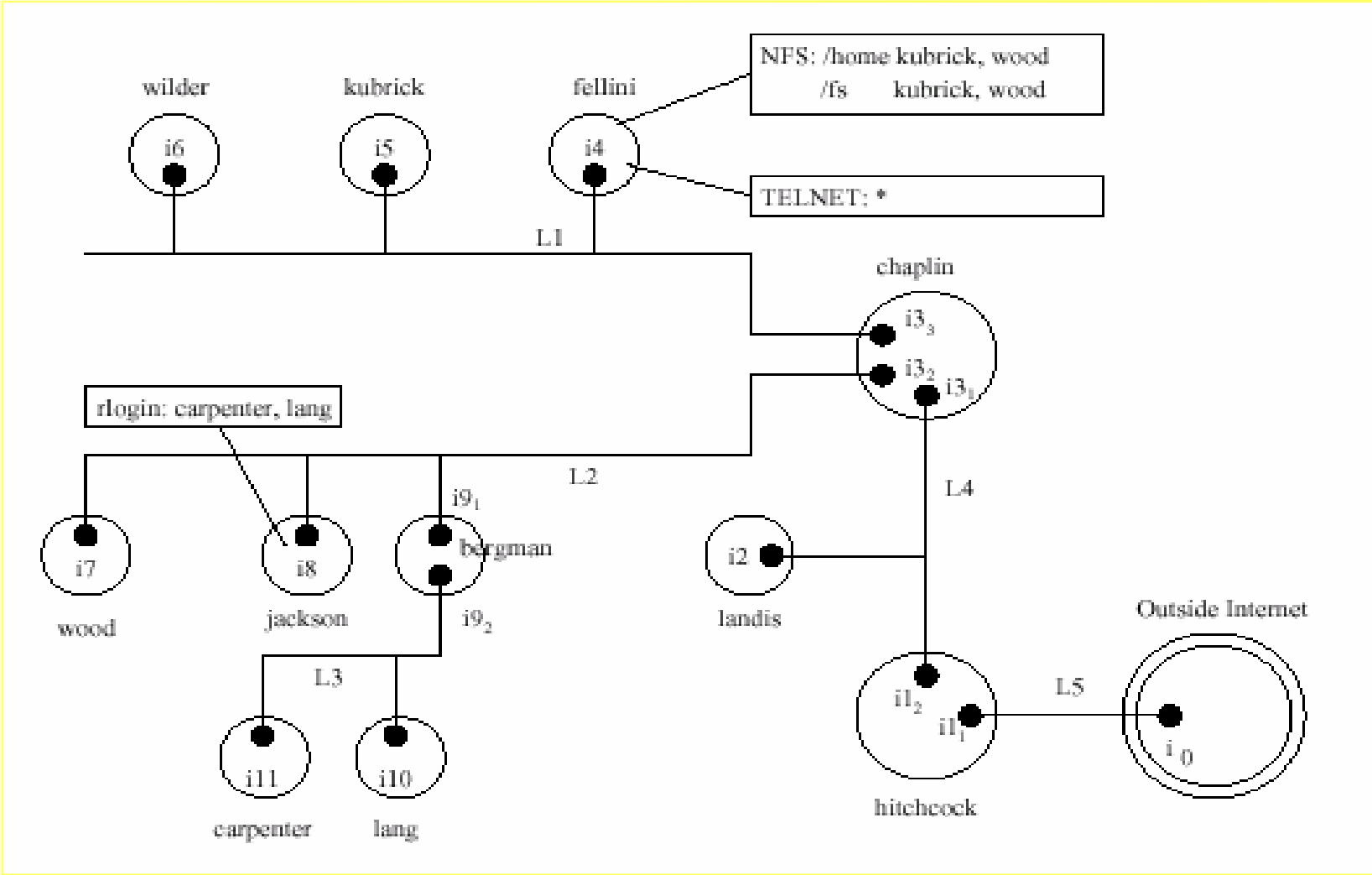
Network topology

- The network model underlying the NetSTAT tool uses *interfaces*, *hosts*,and *links* as primitive elements.
- A network is represented as a hypergraph on the set of interfaces.
- In this model, interfaces are nodes while hosts and links are edges; that is, hosts and links are modeled as sets of interfaces.

Network services

- A description of the services provided by the hosts of a network.
  - Network File System (NFS),
  - The Network Information System (NIS),
  - TELNET, FTP, \r" services, etc.
- A characterization of each service in terms of
  - The network/transport protocol(s) used
  - The access model (e.g., request/reply)
  - The type of authentication (e.g., address-based, password-based, token-based, or certicate-based)
  - and the level of traffic protection (e.g., encrypted or not)
- The network fact base contains information about how services are deployed (i.e., how services are instantiated and accessed over the network)

Hypergraph of a network



States

- the currently active connections
- the state of interaction
- the values of the network tables
  - routing tables
  - DNS mappings
  - ARP caches

Assertions — Static assertions

- Static assertions are assertions about a network that can be verified by examining the network fact base;
  - the network topology
  - the current service configuration.

- Service s in server.services |

  s.name == "www" and

  s.application.name == "CERN httpd“
- Interface i in gateway.interfaces |

  i.link.type == "Ethernet"

Assertions — Dynamic assertions

- Dynamic assertions can be verified only by examining the current state of the network.
  - ConnectionEstablished (addr1, port1, addr2, port2)

Transitions and Signature Actions

- The link-level message

  Message m {i_x, i_y} | m.length > 512

- IP datagram event

  [IPDatagram d] {i_x, i_y} | d.options.sourceRoute == true;

- TCPSegment t in [VirtualCircuit c] {i_x, i_y} |

  c.dstIP == a_y

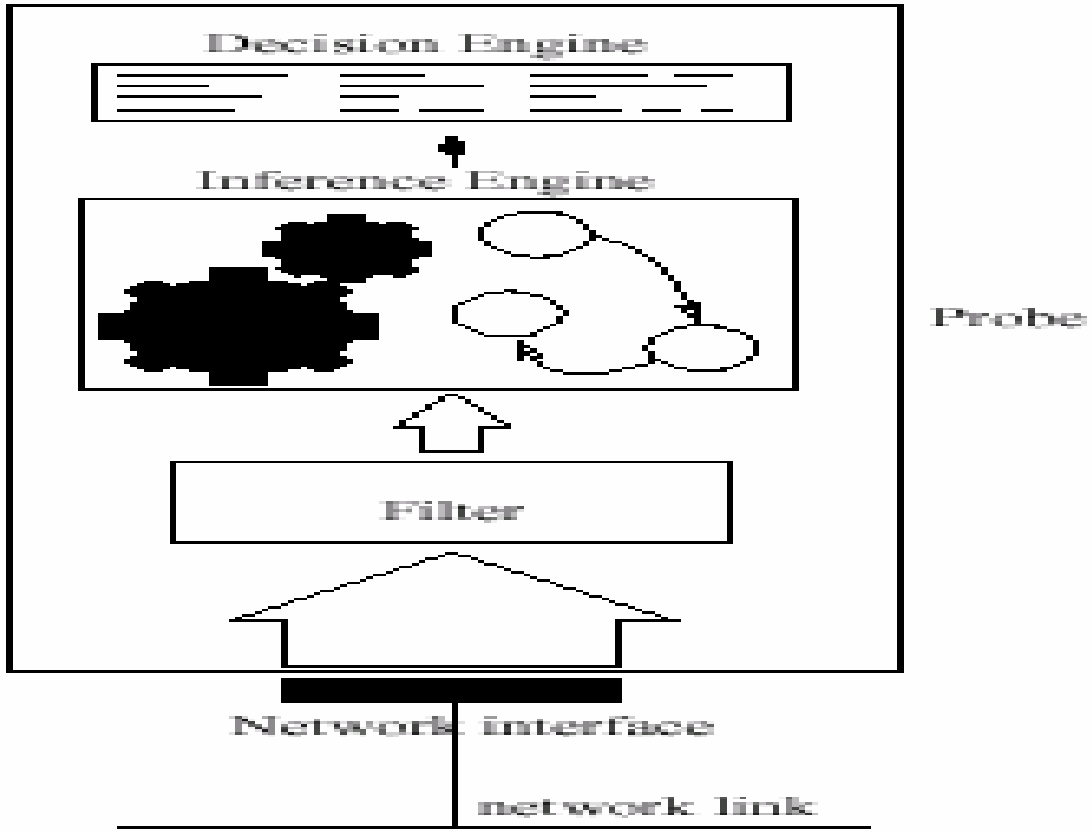
  c.dstPort == 80

  t.syn == false

  t.ack == true

- [IPDatagram d [UDPDatagram u [RPC r]]] {i_x, i_y} |
  d.dst == a_y and
  u.dst == 2049 and
  r.type == CALL and
  r.proc == MKDIR;

5.3 Probes


Decision Engine
Inference Engine
Probe
Filter
Network interface
network link

The filter

- The filter module is responsible for filtering the network message stream.
- Its main task to select following messages from the huge number of messages transmitted over a network link
  - contribute to signature actions
  - dynamic assertions used in a state transition scenario
- The filter module is configured remotely by the analyzer.
- Its configuration can also be updated at run-time
- The performance of the filter is of paramount importance

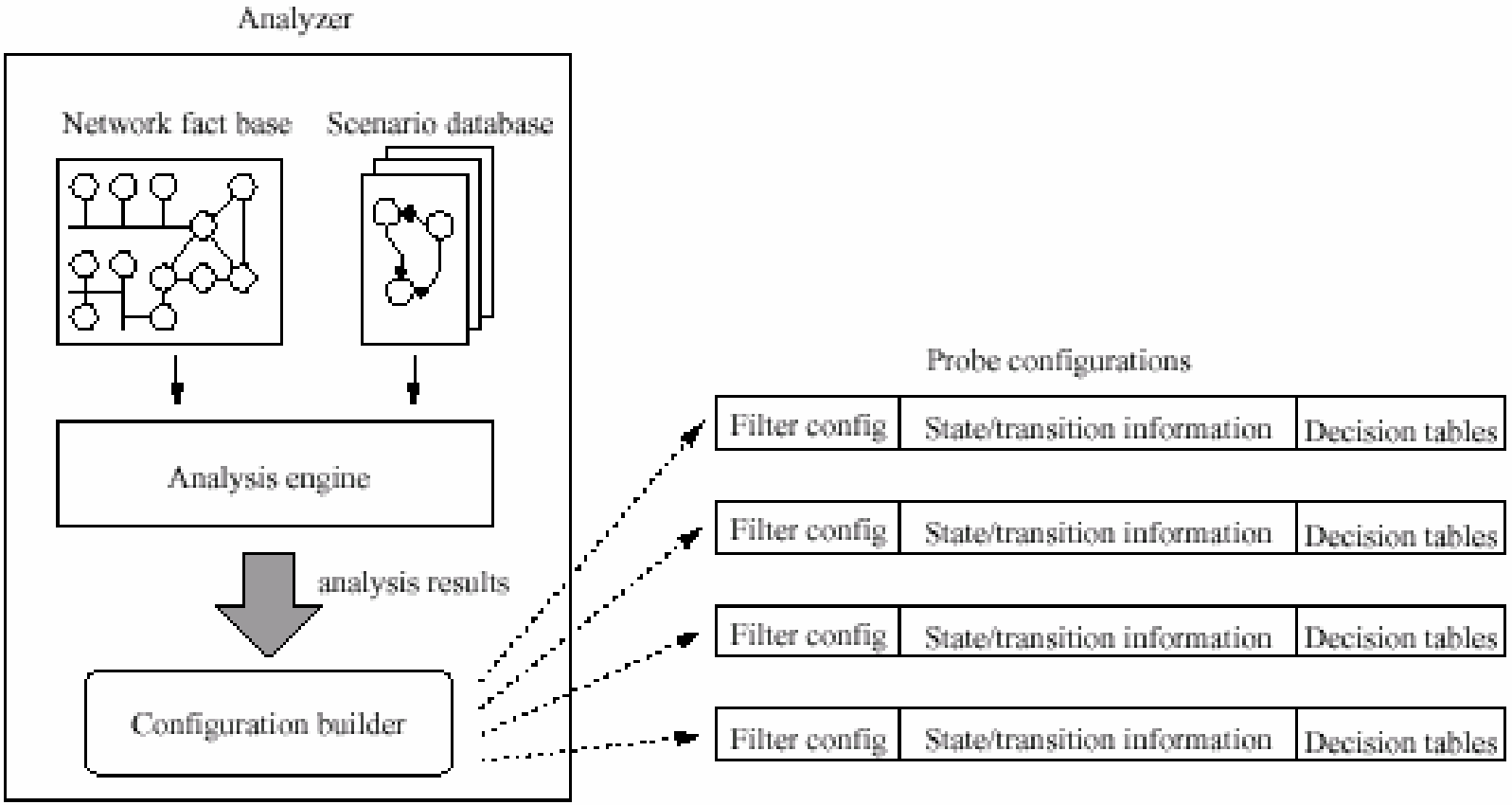
The inference engine

- The inference engine is the actual intrusion detecting component.
- This module is initialized by the analyzer with a set of state transition information representing attack scenarios.
- These attack scenarios are codified in a structure called the inference engine table.
- The event stream provided by the filter is interpreted looking for further evidence of an occurring attack.

Decision engine

- Evolution of the inference engine state is monitored by the decision engine.
- Decision engine is responsible for taking actions based on the outcomes of the inference engine analysis.
  - Informing the Network Security Officer of successful or failed intrusion attempts;
  - alerting the Network Security Officer during the first phases of particularly critical scenarios;
  - suggesting possible actions that can preempt a state transition leading to a compromised state;
  - playing an active role in protecting the network
    - injecting datagrams that reset network connections

5.4 Analyzer


Analyzer
Network fact base
Scenario database
Analysis engine
analysis results
Configuration builder
Probe configurations
Filter config
State/transition information
Decision tables
Filter config
State/transition information
Decision tables
Filter config
State/transition information
Decision tables
Filter config
State/transition information
Decision tables

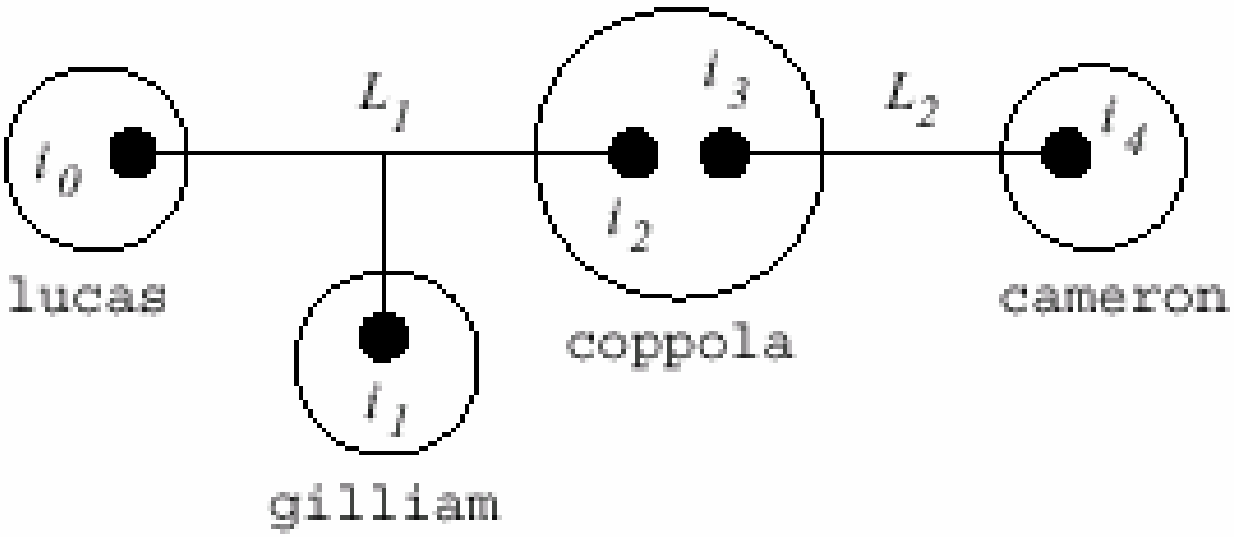
The analyzer

- The analyzer is a stand-alone application
  - used by the Network Security Officer to analyze and instrument a network for the detection of a number of selected attacks.
- The analyzer component acts as a probe manager
  - Customizes a number of general-purpose probes using an automated process based on a formal description of the network to be protected and of the attacks to be detected.
  - This customization takes the form of a set of probe configurations.
  - Each probe configuration species the positioning of a probe, the set of events to be monitored, and a description of the intrusions that the probe should detect.
  - These intrusion scenarios are customized for the particular subnetwork

The Architecture at Work

- The Network Security Officer builds a database of attack scenarios.
- The Network Security Officer builds a network fact base describing the network to be protected.
- The Network Security Officer invokes the analyzer.

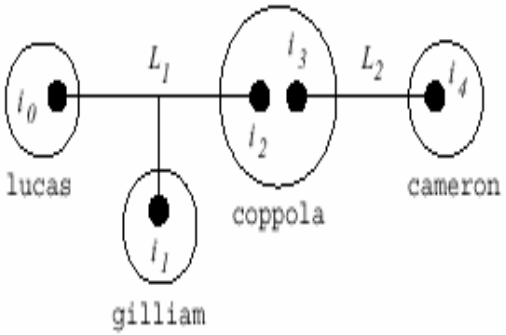
5.5 Example: IP Spoofing


L_1
i_3
L_2
i_0
i_2
i_4
lucas
coppola
cameron
i_1
gilliam

```
Message m in [IPDatagram d]{i_s, i_v} |
   d.dst == a_v and
   d.src == a_z and
   isFirst(m, d);
```

Compromised

```
Host spoofer in Network.hosts;
Interface i_s in spoofer.interfaces;
Host victim in Network.hosts |
   victim != spoofer;
Interface i_v in victim.interfaces;
IPAddress a_v in i_v.ipAddresses;
IPAddress a_z in Network.ipAddresses |
   not spoofer.ipAddresses.contains(a_z);
```



Scenarios					
spoofer	i_s	a_z	victim	i_v	a_v
lucas	i_0	a_{1_4}	gilliam	i_1	a_1
lucas	i_0	a_{1_4}	coppola	i_2	a_2
lucas	i_0	a_{1_4}	coppola	i_3	a_3
lucas	i_0	a_{1_4}	cameron	i_4	a_4
gilliam	i_1	$a_{0,2\_4}$	lucas	i_0	a_0
gilliam	i_1	$a_{0,2\_4}$	coppola	i_2	a_2
gilliam	i_1	$a_{0,2\_4}$	coppola	i_3	a_3
gilliam	i_1	$a_{0,2\_4}$	cameron	i_4	a_4
coppola	i_2	$a_{0\_1,4}$	lucas	i_0	a_0
coppola	i_2	$a_{0\_1,4}$	gilliam	i_1	a_1
coppola	i_2	$a_{0\_1,4}$	cameron	i_4	a_4
coppola	i_3	$a_{0\_1,4}$	lucas	i_0	a_0
coppola	i_3	$a_{0\_1,4}$	gilliam	i_1	a_1
coppola	i_3	$a_{0\_1,4}$	cameron	i_4	a_4
cameron	i_4	a_{0_3}	lucas	i_0	a_0
cameron	i_4	a_{0_3}	gilliam	i_1	a_1
cameron	i_4	a_{0_3}	coppola	i_2	a_2
cameron	i_4	a_{0_3}	coppola	i_3	a_3

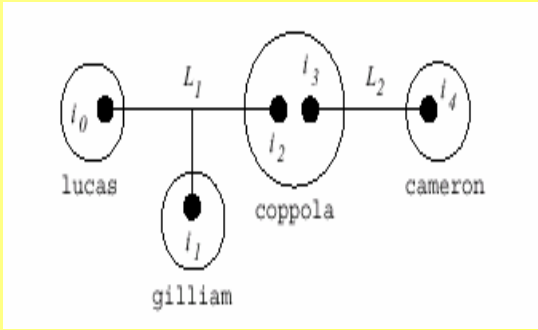
Signature

Scenarios		message
spoofer	victim	
lucas	gilliam	$\langle i_0, i_1, \langle a_{1\text{-}4}, a_1 \rangle \rangle$
lucas	coppla	$\langle i_0, i_2, \langle a_{1\text{-}4}, a_{2\text{-}3} \rangle \rangle$
lucas	cameron	$\langle i_0, i_2, \langle a_{1\text{-}4}, a_4 \rangle \rangle$, $\langle i_3, i_4, \langle a_{1\text{-}3}, a_4 \rangle \rangle$
gilliam	lucas	$\langle i_1, i_0, \langle a_{0,2\text{-}4}, a_0 \rangle \rangle$
gilliam	coppla	$\langle i_1, i_2, \langle a_{0,2\text{-}4}, a_{2\text{-}3} \rangle \rangle$
gilliam	cameron	$\langle i_1, i_2, \langle a_{0,2\text{-}4}, a_4 \rangle \rangle$, $\langle i_3, i_4, \langle a_{0,\,2\text{-}4}, a_4 \rangle \rangle$
coppla	lucas	$\langle i_2, i_0, \langle a_{0\text{-}1,4}, a_0 \rangle \rangle$
coppla	gilliam	$\langle i_2, i_1, \langle a_{0\text{-}1,4}, a_1 \rangle \rangle$
coppla	cameron	$\langle i_3, i_4, \langle a_{0\text{-}1,4}, a_4 \rangle \rangle$
cameron	lucas	$\langle i_4, i_3, \langle a_{0\text{-}3}, a_0 \rangle \rangle$, $\langle i_2, i_0, \langle a_{0\text{-}3}, a_0 \rangle \rangle$
cameron	gilliam	$\langle i_4, i_3, \langle a_{0\text{-}3}, a_1 \rangle \rangle$, $\langle i_2, i_1, \langle a_{0\text{-}3}, a_1 \rangle$
cameron	coppla	$\langle i_4, i_3, \langle a_{0\text{-}3}, a_{2\text{-}3} \rangle \rangle$

- The detection of the first message can be a tricky task.
- For example
  - consider message (i_3, i_4,<a_0, a_4>) ;
  - the message is sent from coppola to cameron and encapsulates an IP datagram from lucas to cameron.
  - This message could represent either the second step in the delivery of a normal IP datagram
  - Or the result of coppola's attempt to spoof lucas' address with respect to cameron.

To place a probe on link L1 and a probe on link L2.

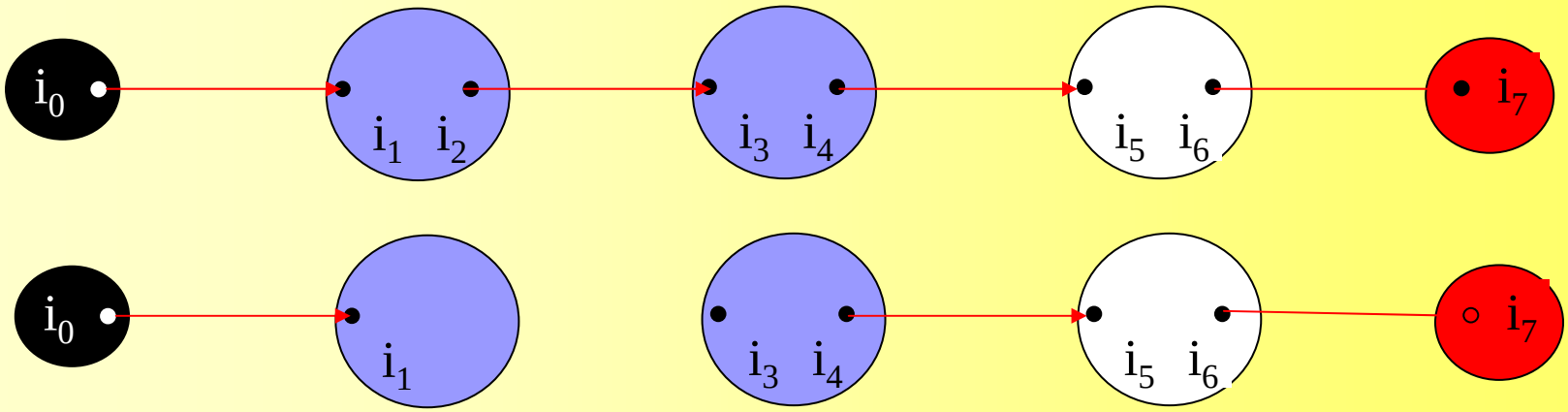
When a message directed to coppola and intended for cameron appears on link L1, the corresponding probe sends a message to the probe on link L2 so that the subsequent message appearing on L2 can be ignored.



	Probe on link L_1			
	Source interface	Destination interface	Source IP address	Destination IP address
	i_0	i_1	a_{1-4}	a_1
(1)	i_0	i_2	a_0	a_4
	i_0	i_2	a_{1-4}	a_2
	i_0	i_2	a_{1-4}	a_3
	i_0	i_2	a_{1-4}	a_4
	i_1	i_0	$a_{0,2-4}$	a_0
(2)	i_1	i_2	a_1	a_4
	i_1	i_2	$a_{0,2-4}$	a_2
	i_1	i_2	$a_{0,2-4}$	a_3
	i_1	i_2	$a_{0,2-4}$	a_4
	i_2	i_0	a_{0-1}	a_0
(3)	i_2	i_0	a_4	a_0
	i_2	i_1	a_{0-1}	a_1
(4)	i_2	i_1	a_4	a_1

	Probe on link L_2			
	Source interface	Destination interface	Source IP address	Destination IP address
	i_4	i_3	a_{0-3}	a_0
(5)	i_4	i_3	a_4	a_0
(6)	i_4	i_3	a_4	a_1
	i_4	i_3	a_{0-3}	a_1
	i_4	i_3	a_{0-3}	a_2
	i_4	i_3	a_{0-3}	a_3
(7)	i_3	i_4	a_0	a_4
(8)	i_3	i_4	a_1	a_4

This solution creates an enormous traffic overload because every message that is forwarded by coppola requires synchronizations between the two probes.



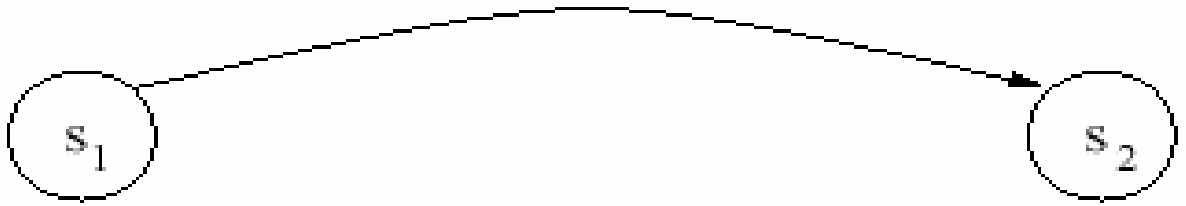
Suppose the victim is a_7. a_0 is sending a package with IP source address a_6. There is a message from interface i_2 to i_3

It should have a path from i_6 to i_2 , if it was a normal packet, even if there was not the link i_2 to i_3.

There is no path between the interface i_2 and the interface i_6, which is interface associated with the source address, if we remove the message's source interface from its link.

An “improved” description

```
Message m in [IPDatagram d]{i_s, i_v} |
    d.src == a_z and d.dst == a_v and
    not (Network.detachFromLink(m.src)).
        existsPath(m.src, a_z.interface);
```



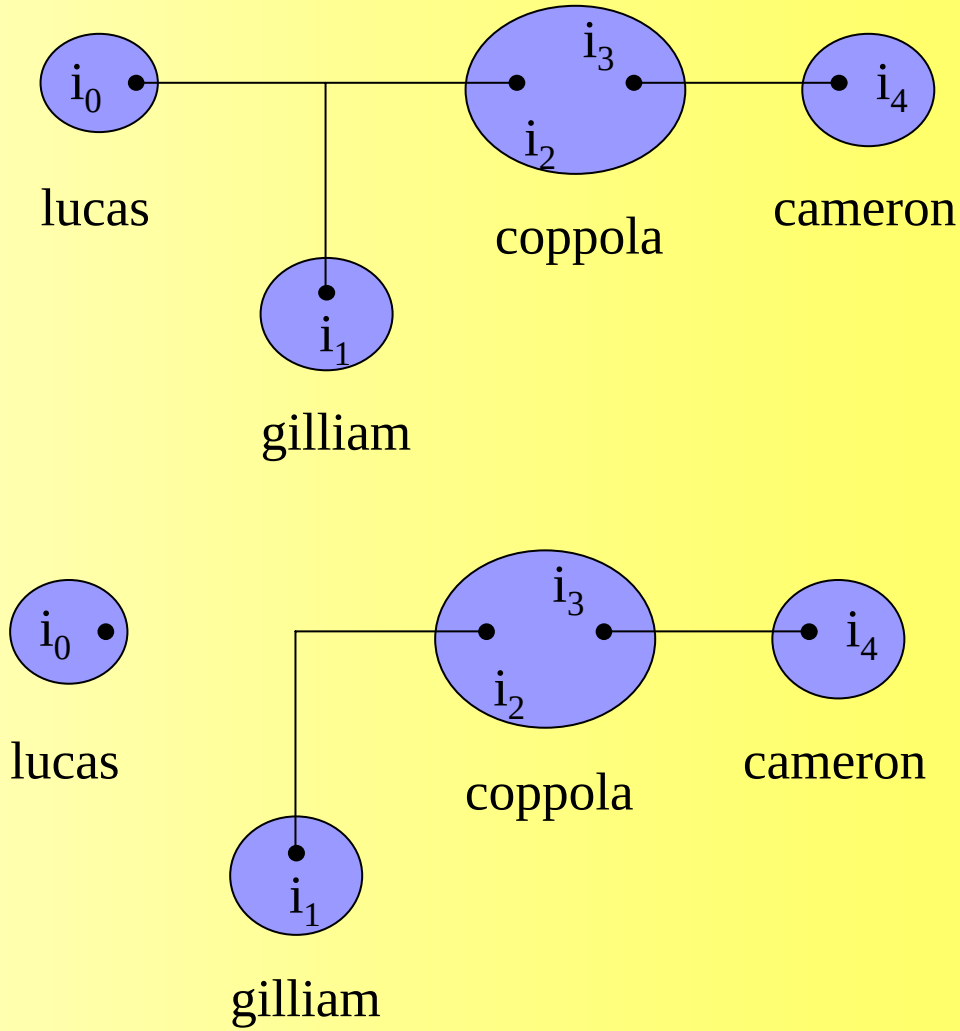

```
Host spoofer in Network.hosts;
Interface i_s in spoofer.interfaces;
Host victim in Network.hosts |
   victim != spoofer;
Interface i_v in victim.interfaces;
IPAddress a_v in i_v.ipAddresses;
IPAddress a_z in Network.ipAddresses |
   not spoofer.ipAddresses.contains(a_z);
```

Compromised

Transition

not (Network.detachFromLink(m.src)). existsPath(m.src, a_z.interface);

- $<i_0, i_2, <a_3, a_1> >$



Signature

Scenarios		*Messages*
spoofer	victim	
lucas	gilliam	$\langle i_0, i_1, \langle a_{1\_4}, a_1\rangle\rangle$
lucas	coppola	$\langle i_0, i_2, \langle a_{1\_4}, a_{2\_3}\rangle\rangle$
lucas	cameron	$\langle i_0, i_2, \langle a_{1\_3}, a_4\rangle\rangle, \langle i_3, i_4, \langle a_{1\_3}, a_4\rangle\rangle$
		$\langle i_0, i_2, \langle a_4, a_4\rangle\rangle, \langle i_3, i_4, \langle a_4, a_4\rangle\rangle$
gilliam	lucas	$\langle i_1, i_0, \langle a_{0\_2\_4}, a_0\rangle\rangle$
gilliam	coppola	$\langle i_1, i_2, \langle a_{0\_2\_4}, a_{2\_3}\rangle\rangle$
gilliam	cameron	$\langle i_1, i_2, \langle a_{0\_2\_3}, a_4\rangle\rangle, \langle i_3, i_4, \langle a_{0\_2\_3}, a_4\rangle\rangle$
		$\langle i_1, i_2, \langle a_4, a_4\rangle\rangle, \langle i_3, i_4, \langle a_4, a_4\rangle\rangle$
coppola	lucas	$\langle i_2, i_0, \langle a_{0\_1}, a_0\rangle\rangle$
		$\langle i_2, i_0, \langle a_4, a_0\rangle\rangle$
coppola	gilliam	$\langle i_2, i_1, \langle a_{0\_1}, a_1\rangle\rangle$
		$\langle i_2, i_1, \langle a_4, a_1\rangle\rangle$
coppola	cameron	$\langle i_3, i_4, \langle a_{0\_1}, a_4\rangle\rangle$
		$\langle i_3, i_4, \langle a_4, a_4\rangle\rangle$
cameron	lucas	$\langle i_4, i_3, \langle a_{2\_3}, a_0\rangle\rangle, \langle i_2, i_0, \langle a_{2\_3}, a_0\rangle\rangle$
		$\langle i_4, i_3, \langle a_{0\_1}, a_0\rangle\rangle, \langle i_2, i_0, \langle a_{0\_1}, a_0\rangle\rangle$
cameron	gilliam	$\langle i_4, i_3, \langle a_{2\_3}, a_1\rangle\rangle, \langle i_2, i_1, \langle a_{2\_3}, a_1\rangle\rangle$
		$\langle i_4, i_3, \langle a_{0\_1}, a_1\rangle\rangle, \langle i_2, i_1, \langle a_{0\_1}, a_1\rangle\rangle$
cameron	coppola	$\langle i_4, i_3, \langle a_{0\_3}, a_{2\_3}\rangle\rangle$

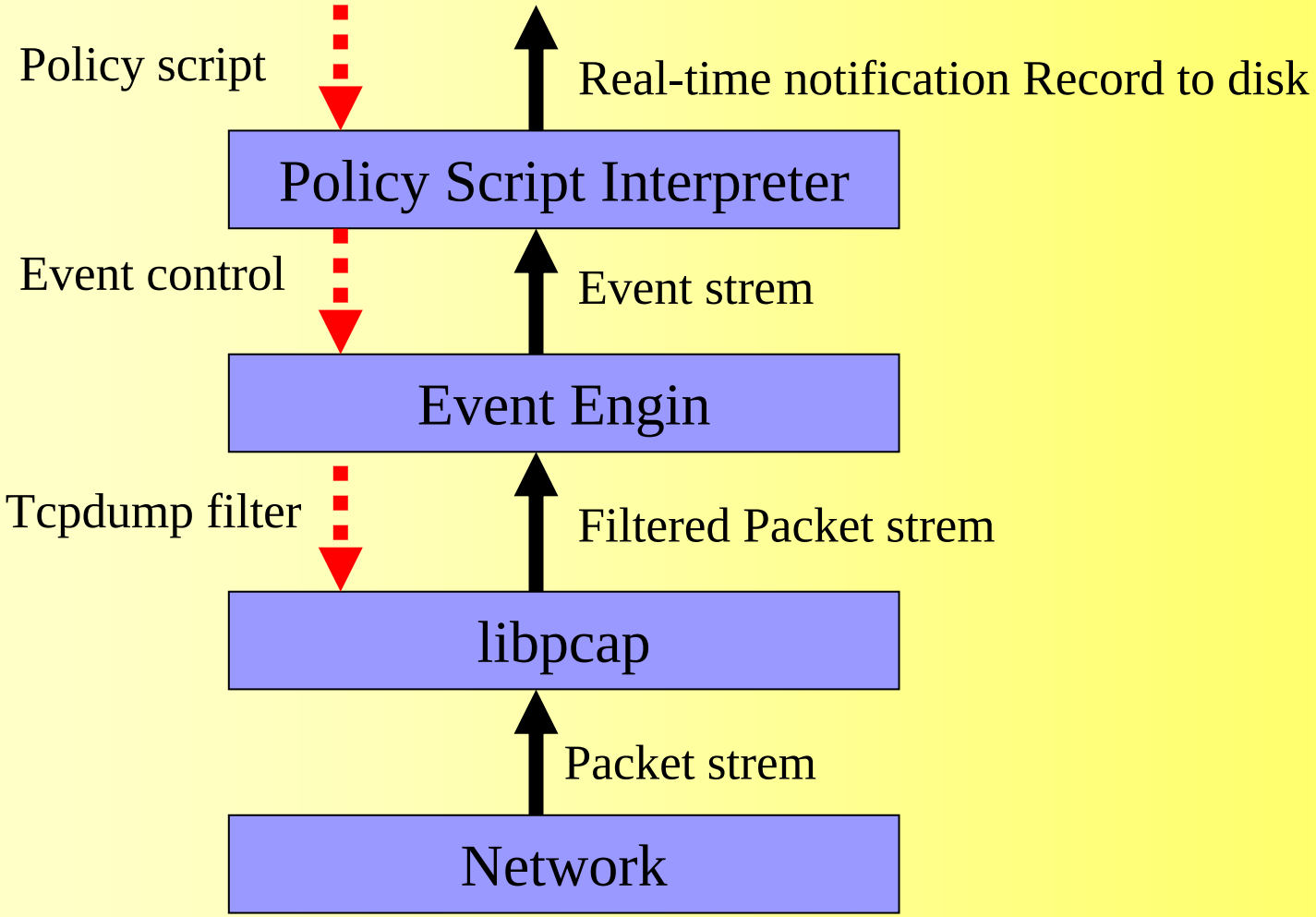
Probe configuration

Probe on link L_1			
Source interface	Destination interface	Source IP address	Destination IP address
i_0	i_1	a_{1_4}	a_1
i_0	i_2	a_{1_4}	a_2
i_0	i_2	a_{1_4}	a_3
i_0	i_2	a_{1_4}	a_4
i_1	i_0	$a_{0\_2\_4}$	a_0
i_1	i_2	$a_{0\_2\_4}$	a_2
i_1	i_2	$a_{0\_2\_4}$	a_3
i_1	i_2	$a_{0\_2\_4}$	a_4
i_2	i_0	a_{0_1}	a_0
i_2	i_1	a_{0_1}	a_1

Probe on link L_2			
Source interface	Destination interface	Source IP address	Destination IP address
i_3	i_4	a_4	a_4
i_4	i_3	a_{0_3}	a_0
i_4	i_3	a_{0_3}	a_1
i_4	i_3	a_{0_3}	a_2
i_4	i_3	a_{0_3}	a_3

6. BRO

6.1 Structure of the system



libpcap

- It isolates Bro from details of the network link technology (Ethernet, FDDI, SLIP, etc.);
- It greatly aids in porting Bro to different Unix variants (also makes it easier to upgrade to faster hardware);
- It means that Bro can also operate on tcpdump save files, making off-line development and analysis easy.

- Winnowing down the packet stream as soon as possible greatly abets monitoring at high speeds without losing packets.

libpcap

port finger or port ftp or tcp port 113 or port telnet or port login or port 111

- the fillter accepts any TCP packets with a source or destination port of 79 (Finger), 21 (FTP), 113 (Ident), 23 (Telnet), 513 (Rlogin), and any TCP or UDP packets with a source or destination port of 111 (Portmapper).

tcp[13] & 7 != 0

- capture any TCP packets with the SYN, FIN, or RST control bits set.
- delimit the beginning (SYN) and end (FIN or RST) of each TCP connection.
- extract connection start time, duration, participating hosts, ports , and the number of bytes sent in each direction.

Event engine

- This layer first performs several integrity checks to assure that the packet headers are well-formed, including verifying the IP header checksum.
- If the checks succeed, then it looks up or creates the connection state.
- It then dispatches the packet to a handler for the corresponding connection to process the data payload of the packet.

- Different changes in the connection's state generate different events.
  - When the initial SYN packet requesting a connection is seen, the event engine schedules a timer for T seconds in the future.
  - If the timer expires and the connection has not changed state, then the engine generates a connection_attempt event.
  - If the other connection endpoint replies with a correct SYN acknowledgement packet before that time, then the engine immediately generates a connection_established event, and cancels the connection attempt timer.
  - If the endpoint replies with a RST packet, then the connection attempt has been rejected, and the engine generates **connection_rejected** event.
  - If a connection terminates via a normal FIN exchange, then the engine generates **connection_finished.**

Policy script interpreter

- A key facet of Bro's design is the clear distinction between the generation of events versus what to do in response to the events, that is the separation between mechanism and policy.
- The ‘policy script interpreter’ executes **scripts** written in the specialized Bro language.
- For each event passed to the interpreter, it retrieves the (semi-)compiled code for the corresponding handler
- This code in turn can execute arbitrary Bro scripting commands, including generating new events, logging real-time notifications, recording data to disk.

7. Distributed denial-of-service (DDOS)

- Distributed denial-of-service attacks (DDoS) pose an immense threat to the Internet
- Consequently many defense mechanisms have been proposed to combat them.
- Attackers constantly modify their tools to bypass these security systems
- Researchers in turn modify their approaches to handle new attacks.

- A denial-of-service attack character
  - prevent legitimate users of a service from using that service.
- Distributed denial-of-service attack
  - deploy multiple machines to attain this goal.
- The service is denied by sending a stream of packets to a victim that
  - consumes some key resource, thus rendering it unavailable to legitimate clients
  - provides the attacker with unlimited access to the victim machine so he can inflict arbitrary damage.

- What makes DDoS attacks possible?
- How do these attacks occur?
- Why do they occur?

7.2 What makes DDoS attacks possible?

1. Internet security is highly interdependent
   - DDoS attacks are commonly launched from systems that are subverted through security-related compromises.
   - Regardless of how well secured the victim system may be, its susceptibility to DDoS attacks depends on the state of security in the rest of the global Internet.
2. Internet resources are limited.
3. Power of many is greater than power of few.
4. Intelligence and resources are not collocated
   - a desire for large throughput led to the design of high bandwidth pathways in the intermediate network.
   - malicious clients can misuse the abundant resources of unwitting network for delivery of numerous messages to a victim.

7.3 Taxonomy of DDoS Attacks

7.3.1 Classification by Degree of Automation

- ***Manual Attacks***
  - The attacker scanned remote machines for vulnerabilities
  - broke into them and installed the attack code
  - commanded the onset of the attack

- ***Semi-Automatic Attacks***
  - the DDoS network consists of handler (master) and agent (slave, daemon) machines;
  - The attacker deploys automated scripts for scanning and compromise of those machines and installation of the attack code.
  - He then uses handler machines to specify the attack type and the victim's address and to command the onset of the attack to agents, who send packets to the victim.
  - **communication mechanism** deployed between agent and handler machines
    - *direct communication*
    - *indirect communication*

- ***Attacks with direct communication***
  - The agent and handler machines need to know each other's identity in order to communicate.
  - This is achieved by hard-coding the IP address of the handler machines in the attack code.
  - Each agent then reports its readiness to the handlers, who store its IP address in a file for later communication.
  - The obvious drawback of this approach is that discovery of one compromised machine can expose the whole DDoS network.
  - Also, since agents and handlers listen to network connections, they are identifiable by network scanners.

Attacks with indirect communication

- Attacks with indirect communication deploy a level of indirection to increase the survivability of a DDoS network.
- Recent attacks provide the example of using IRC channels for agent/handler communication.
- Thus, discovery of a single agent may lead no further than the identification of one or more IRC servers and channel names used by the DDoS network.

- Automatic Attacks
  - The time of the onset of the attack, attack type, duration and victim's address is preprogrammed in the attack code.
  - It is obvious that such deployment mechanisms offer minimal exposure to the attacker
  - The hard-coded attack specification suggests a single-purpose use of the DDoS network.
  - The propagation mechanisms usually leave the backdoor to the compromised machine open, enabling easy future access and modification of the attack code.

7.3.2 Classification by Exploited vulnerability

- *protocol attacks*
- *brute-force attacks*.

7.3.2.1 *protocol attacks*

Protocol attacks exploit a specific feature or implementation bug of some protocol installed at the victim in order to consume excess amounts of its resources.

- TCP SYN attack
  - **The exploited feature is the allocation of substantial space in a connection queue immediately upon receipt of a TCP SYN request.**
  - **The attacker initiates multiple connections that are never completed, thus filling up the connection queue indefinitely.**
- CGI request attack
  - Consumes the CPU time of the victim by issuing multiple CGI requests.
- authentication server attack
  - The signature verification process consumes significantly more resources than bogus signature generation.
  - He sends numerous bogus authentication requests to the server, tying up its resources.

7.3.2.2 brute-force attacks

- **initiating a vast amount of seemingly legitimate transactions.**
- **deliver higher traffic volume than the victim network can handle.**

- ***Filterable Attacks***
  - Filterable attacks use bogus packets or packets for non-critical services of the victim's operation
  - thus can be filtered by a firewall.
  - UDP flood attack or an ICMP request flood
- ***Non-filterable Attacks***
  - use packets that request legitimate services from the victim.
  - filtering all packets that match the attack signature would lead to an immediate denial of the specified service to both attackers and the legitimate clients.
  - HTTP request flood or a DNS request flood

- The difference is that
  - A victim can mitigate the effect of protocol attacks by modifying the deployed protocols at its site
  - It is helpless against brute-force attacks due to their misuse of legitimate services (non-filterable attacks) or due to its own limited resources

7.3.3. Classification by Attack Rate Dynamics

- **Continuous Rate Attacks**
  - After the onset is commanded, agent machines generate the attack packets with full force.
  - This sudden packet flood disrupts the victim's services quickly, and thus leads to attack detection.
- **Variable Rate Attacks**
  - Variable rate attacks are more cautious in their engagement, and they vary the attack rate to avoid detection and response.
    - **Increasing Rate Attacks**
    - **Fluctuating Rate Attacks**
    - **pulsing attacks.**

7.3.4. Classification by Impact

- Disruptive Attacks
  - The goal of disruptive attacks is to completely deny the victim's service to its clients. All currently known attacks belong to this category.
- Degrading Attacks
  - The goal of degrading attacks would be to consume some portion of a victim's resources.
  - Since these attacks do not lead to total service disruption, they could remain undetected for a significant time period.

7.4 Taxonomy of DDoS Defense Mechanisms

7.4.1 Classification by Activity Level

- Preventive Mechanisms
  - **Attack Prevention Mechanisms**
    - System security
    - Protocol security
  - **DoS Prevention**
    - Resource Accounting
    - Resource Multiplication
- Reactive Mechanisms
  - **attack detection strategy**
  - **response strategy**
  - **cooperation degree**

7.4.2 Classification by Deployment Location

7.4.1 Classification by Activity Level

7. 4.1.1 Preventive Mechanisms

- The goal of preventive mechanisms is either
  - to eliminate the possibility of DDoS attacks altogether
  - to enable potential victims to endure the attack without denying services to legitimate clients.
- *attack prevention*
- *denial-of-service prevention*

7.4.1.2 Reactive mechanism

- **attack detection strategy**
  - **Pattern Attack Detection**
  - **Anomaly Attack Detection**
  - **Hybrid Attack Detection**
  - **Third-Party Attack Detection**
- **attack response strategy**
  - **Agent Identification Mechanisms**
  - **Rate-Limiting Mechanisms**
  - **Filtering Mechanisms**
  - **Reconfiguration Mechanisms**

(1) Attack prevention Mechanisms

- Attack prevention mechanisms modify the system configuration to eliminate the possibility of a DDoS attack.
  - *system security*
  - *protocol security*

Aattack prevention — *system security mechanism*

- Increase the overall security of the system;
  - monitored access, applications that download, install security patches, firewall systems, virus scanners, access lists for critical resources.
- against illegitimate accesses to the machine;
- Removing application bugs and updating protocol installations

Attack prevention — protocol security mechanism

- Many protocols contain operations that are cheap for the client but expensive for the server.
- Such protocols can be misused to exhaust the resources of a server
  - **TCP SYN attack, the authentication server attack, the fragmented packet attack**
- Examples of protocol security mechanisms include guidelines for a safe protocol design in which resources are committed to the client only after
  - **sufficient authentication is done**
  - **the client has paid a sufficient price**

  deployment of powerful proxy server that completes TCP connections.

(2) Denial-of-Service Prevention Mechanisms

- Denial-of-service prevention mechanisms enable the victim to endure attack attempts without denying service to legitimate clients.
- This is done either by
  - enforcing policies for resource consumption
  - ensuring that abundant resources exist so that legitimate clients will not be affected by the attack.

Attack detection strategy

- Pattern Attack Detection
- Anomaly Attack Detection
- Hybrid Attack Detection
- Third-Party Attack Detection

Response strategy (1)

- **Agent Identification Mechanisms**
  - Agent identification mechanisms provide thevictim with information about the identity of the machines that are performing the attack.
  - This information can alleviate the impact of the attack.
  - Agent identification examples include
    - traceback techniques

      [1] D. Dean, M. Franklin and A. Stubblefield, An algebraic approach to IP Traceback.

      [2] S. M. Bellovin, ICMP traceback messages.

    - approaches that eliminate spoofing

      [1] P. Ferguson and D. Senie, Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing.

Response strategy (2)

- **Rate-Limiting Mechanisms**
  - Rate-limiting mechanisms impose a rate limit on a stream that has been characterized as malicious by the detection mechanism.

S.Floyd, et al, Pushback Messages for Controlling aggregates in the Network, Internet draft.

  - This is usually deployed when the detection mechanism has a high level of false positives or cannot precisely characterize the attack stream.
  - The disadvantage is that they allow some attack traffic through through, so extremely high scale attacks might still be effective even if all traffic streams are rate-limited.

Response strategy (3)

- **Filtering Mechanisms**
  - Filtering mechanisms use the attack characterization to filter out the attack stream completely.
  - Examples include dynamically deployed firewalls

    T. Darmohray and R. Oliver, Hot spares for DDoS attacks, http://www.usenix.org/publications/login/2000-7/apropos.html.
  - Filtering mechanisms run the risk of accidentally denying service to legitimate traffic.
  - Worse, clever attackers might leverage them as denial-of-service tools.

Response strategy (4)

- **Reconfiguration Mechanisms**
  - Reconfiguration mechanisms change the topology of the victim or the intermediate network to either add more resources to the victim or to isolate the attack machines.

cooperation degree strategy

- ***Autonomous Mechanisms***
  - Autonomous mechanisms perform independent attack detection and response. They are usually deployed at a single point in the Internet and act locally.

- ***Cooperative Mechanisms***
  - Achieve significantly better performance through cooperation with other entities.
  - Mechanisms deploying pushback provide examples of cooperative mechanisms.
    - **They detect the occurrence of a DDoS attack by observing congestion in a router's buffer, characterize the traffic that creates the congestion, and act locally to impose a rate limit on that traffic.**
    - **the rate limit requests can be propagated to upstream routers who otherwise may be unaware of the attack**

cooperation degree strategy

- ***Interdependent Mechanisms***
  - Interdependent mechanisms cannot operate autonomously; they rely on other entities either for attack detection or for efficient response.
  - Traceback mechanisms provide examples of interdependent mechanisms. A traceback mechanism deployed on a single router would provide almost no benefit.

7. 4.2. Classification by Deployment Location

- Victim-Network Mechanisms
  - DDoS defense mechanisms deployed at the victim network protect this network from DDoS attacks and respond to detected attacks by alleviating the impact on the victim.

7. 4.2. Classification by Deployment Location(2)

- Intermediate-Network Mechanisms
  - DDoS defense mechanisms deployed at the intermediate network provide infrastructural service to a large number of Internet hosts.
  - Victims of DDoS attacks can contact the infrastructure and request the service, possibly providing adequate compensation.
  - Pushback and traceback techniques are examples of intermediate-network mechanisms.

7. 4.2. Classification by Deployment Location(3)

- Source-Network Mechanisms
  - The goal of DDoS defense mechanisms deployed at the source network is to prevent customers using this network from generating DDoS attacks.
  - Such mechanisms are necessary and desirable, but motivation for their deployment is low since it is unclear who would pay the expenses associated with this service.

8. IP Traceback

8.1 IP Traceback — The problem

- It has been long understood that the IP protocol permits anonymous attacks.
- In his 1985 paper on TCP/IP weaknesses, Morris writes:

  The weakness in this scheme [the Internet Protocol] is that the source host itself fills in the IP source host id, and there is no provision in ... TCP/IP to discover the true origin of a packet.

  — R. T. Morris, A Weakness in the 4.2BSD Unix TCP/IP Software, Technical Report Computer Science #117, AT&T Bell Labs, Feb. 1985.

8.2 IP Traceback — Ingress filtering

- Configure routers to block packets that arrive with illegitimate source addresses.
  - Ingress filtering is most feasible in customer networks or at the border of Internet Service Providers (ISP) where address ownership is relatively unambiguous and traffic load is low.
  - As traffic is aggregated from multiple ISPs into transit networks, there is no longer enough information to unambiguously determine if a packet arriving on a particular interface has a “legal” source address.
  - Moreover, on many deployed router architectures the overhead of ingress filter becomes prohibitive on high-speed links.

8.3 IP Traceback — Link testing

- Most existing trace back techniques
  - start from the router closest to the victim
  - interactively test its upstream links
    - until they determine which one is used to carry the attacker's traffic.
  - This technique assumes that an attack remains active until the completion of a trace
  - This is inappropriate for
    - Attacks that are detected after the fact
    - Attacks that occur intermittently,
    - Attacks that modulate their behavior in response to a traceback

8.4 IP Traceback — Logging

- An approach is to log packets at key routers and then use data mining techniques to determine the path that the packets traversed.
  - This scheme has the useful property that it can trace an attack long after the attack has completed.
  - However, it also has obvious drawbacks, including potentially enormous resource requirements

8.5 IP Trace back — ICMP Trace back

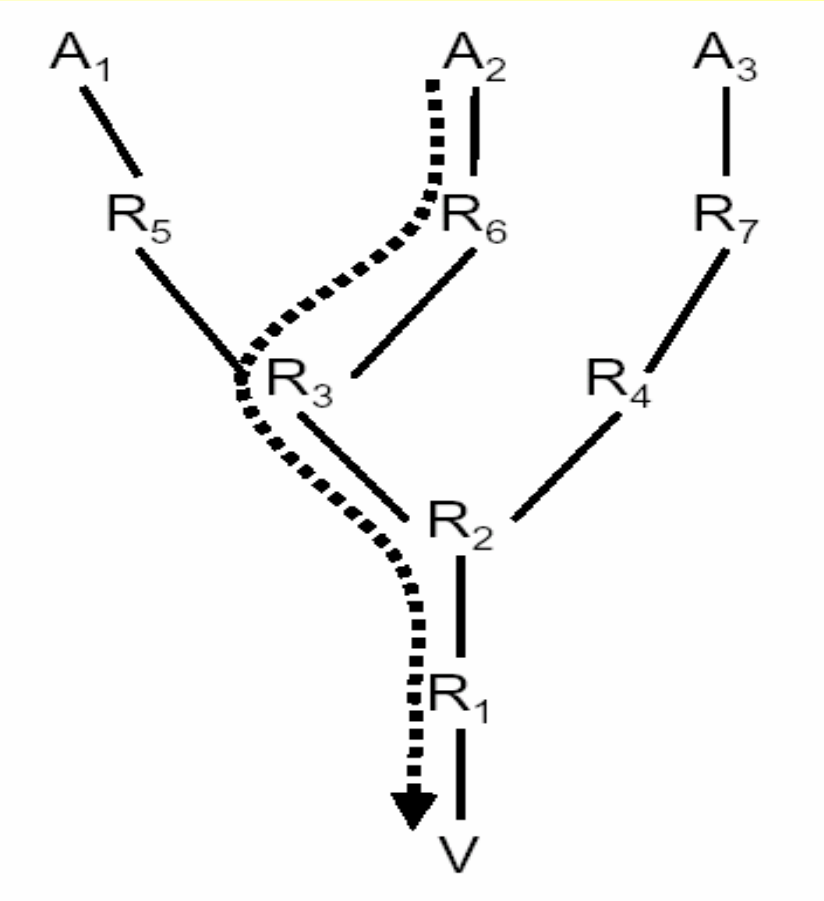
- Use of explicit router-generated ICMP traceback messages.
- The principle idea in this scheme is for every router to
  - With low probability (e.g., 1/20,000)
  - Sample one of the packets it is forwarding
  - Copy the contents into a special ICMP traceback message including information about the adjacent routers along the path to the destination
  - During a flooding-style attack, the victim host can then use these messages to reconstruct a path back to the attacker.
  - S. M. Bellovin，ICMP Traceback Messages，Internet Draft:draft-bellovin-itrace-00.txt, Mar，2000.

8.5 IP Trace back — marking

- Tracing flooding attacks by “marking” packets, either probabilistically or deterministically, with the addresses of the routers they traverse.
  - H. Burch and B. Cheswick, Tracing Anonymous Packets to Their Approximate Source, Dec. 1999.
  - The victim uses the information in the marked packets to trace an attack back to its source.
  - It does not require interactive cooperation with ISPs and therefore avoids the high management overhead of input debugging.

8.5 IP Trace back — marking

- It does not require significant additional network traffic
- It potentially be used to track multiple attacks.
- Like logging, packet marking can be used to trace attacks “post-mortem” – long after the attack has stopped.
- Marking algorithms can be implemented without incurring any significant overhead on network routers.



- **Network as seen from the victim of an attack, V .**
- **Routers are represented by R_i, and potential attackers by A_i.**
- **The dotted line represents a particular *attack path* between an attacker and the victim.**

- The *exact traceback* problem is to determine the attack path and the associated attack origin for each attacker.
- The exact attack origin may never be revealed.
- We define the *approximate traceback* problem as finding a candidate attack path for each attacker that contains the true attack path as a suffix.
  - We call this the *valid suffix* of the candidate path.
  - For example, (R5, R6, R3, R2, R1) is a valid approximate solution to above figure because it contains the true attack path (R6, R3, R2, R1) as a suffix.
- We say a solution to this problem is *robust* if an attacker cannot prevent the victim from discovering candidate paths containing the valid suffix.

8.5.1 Two components

- All marking algorithms have two components:
  - A *marking procedure* executed by routers in the network
  - A *path reconstruction procedure* implemented by the victim.

- A router "marks" one or more packets by augmenting them with additional information about the path they are traveling.
- The victim attempts to reconstruct the attack path using only the information in these marked packets.
- The *convergence time* of an algorithm is the number of packets that the victim must observe to reconstruct the attack path.

8.5.2 Basic assumptions

- an attacker may generate any packet,
- multiple attackers may conspire,
- attackers may be aware they are being traced,
- packets may be lost or reordered,

- attackers send numerous packets,
- the route between attacker and victim is fairly stable,
- routers are both CPU and memory limited,
- routers are not widely compromised.

8.5.3 Node append algorithm

- *Marking procedure at router* R*:*

  for each packet w, append R to w

- *Path reconstruction procedure at victim* v*:*

  for any packet w from attacker extract path (Ri…Rj) from the suffix of w

- The node append algorithm is both robust and extremely quick to converge.
- High router overhead incurred by appending data to packets in flight.
- the length of the path is not known *a priori*, it is impossible to ensure that there is sufficient unused space in the packet for the complete list.

8.5.4 Node sampling

- To reduce both the router overhead and the per-packet space requirement,
- we can sample the path one node at a time.
- A single static “node” field is reserved in the packet header
- Upon receiving a packet, each router chooses to write its address in the node field with some probability p.
- After enough packets have been sent, the victim will have received at least one sample for every router in the attack path.

- As stated in section 3, we assume that the attacker sends enough packets and the route is stable enough that this sampling can converge.

8.5.4 Node sampling

- Although it might seem impossible to reconstruct an ordered path given only an unordered collection of node samples, it turns out that with a sufficient number of trials, the order can be deduced from the relative number of samples per node.

- The probability that a packet will be marked by a router and then left unmolested by all downstream routers is a strictly decreasing function of the distance to the victim.
  - If we constrain p to be identical at each router, then the probability of receiving a marked packet from a router d hops away is $p(1- p)^{d-1}$,

8.5.4 Node sampling

- Routers contributing relatively few samples seem far away from the victim
- Random variability can easily lead to misordering unless a very large number of samples are observed.
- For instance, if d = 15 and p = 0.51, the receiver must
- receive more than 42,000 packets on average before it receives a *single* sample from the furthest router.
- To guarantee that the order is correct with 95% certainty requires more than seven times that number.

- Therefore, this technique is not robust against multiple attackers.

  If there are multiple attackers then multiple routers may exist at the same distance – and hence be sampled with the sample.

9. Strengths and Limitations of IDS

Intrusion detection systems perform the following functions well

- Monitoring and analysis of system events
- Testing the security states of system
- Baselining the security state of a system, then tracking any changes to that baseline
- Recognizing patterns of system events
- Managing operating system audit and logging mechanisms and the data they generate
- Alerting appropriate staff by appropriate means
- Measuring enforcement of security policies
- Allowing non-security experts to perform important security monitoring functions.

9. Strengths and Limitations of IDS

Intrusion detection systems cannot perform the following functions

- Compensating for weak or missing security mechanisms in the protection infrastructure. Such as
  - firewalls, identification and authentication, link encryption, access control mechanisms, and virus detection and eradication.
- Instantaneously detecting, reporting, and responding to an attack, when there is a heavy network or processing load.
- Detecting newly published attacks or variants of existing attacks.
- Effectively responding to attacks launched by sophisticated attackers
- Automatically investigating attacks without human intervention.
- Resisting attacks that are intended to defeat or circumvent them
- Compensating for problems with the fidelity of information sources