UnitelmaSapienza
Università degli Studi di Roma

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Intrusion Detection Systems

Principles, Models, Architecture, Organization

Prof. F. Parisi Presicce

Definitions

- Intrusion
  - A set of actions aimed at compromising the security goals of a computing and networking resource
    - Integrity, confidentiality, availability
- Intrusion detection
  - The process of identifying and responding to intrusion activities

Goals of IDS

- Detect wide variety of intrusions
  - Previously known and unknown attacks
  - Suggests need to learn/adapt to new attacks or changes in behavior
- Detect intrusions in timely fashion
  - May need to be real-time, especially when system responds to intrusion
    - Problem: analyzing commands may impact response time of system
  - May suffice to report intrusion occurred a few minutes or hours ago

Goals of IDS

- Present analysis in simple, easy-to-understand format
  - Ideally a binary indicator
  - Usually more complex, allowing analyst to examine suspected attack
  - User interface critical, especially when monitoring many systems
- Be accurate
  - Minimize false positives, false negatives
  - Minimize time spent verifying attacks, looking for them

Why not just keep intruders out?

- **Second line of defense**.
  - Even the best intrusion prevention systems can fail.
  - Many intruders are insiders.
- **Ejection**.
  - Catch/stop intruders before they can do much damage.
- **Deterrent**.
  - Intruders may stay out if they think they will be caught.
- **Educational**.
  - Learn how intruders do what they do and use this to improve both prevention and detection techniques.

Principles of Intrusion Detection

- Characteristics of systems not under attack
  - User, process actions conform to statistically predictable pattern
  - User, process actions do not include sequences of actions that subvert the security policy
  - Process actions correspond to a set of specifications describing what the processes are (or are not) allowed to do
- Systems under attack do not meet at least one of these characteristics

D.Denning's Model

Hypothesis: exploiting vulnerabilities requires abnormal use of normal commands or instructions

- Includes deviation from usual actions
- Includes execution of actions leading to break-ins
- Includes actions inconsistent with specifications of privileged programs

Assumptions

Primary assumptions:

- System activities are **observable**
- Normal and intrusive activities have **distinct evidence**

Approaches

- **Modeling**
  - Features: evidence extracted from audit data
  - Analysis: piecing the evidences together
    - Misuse detection (rule-based approach)
    - Anomaly detection (statistical-based approach)
- **Deployment**
  - Network-based
  - Host-based
- **Development and maintenance**
  - Hand-coding of “expert” knowledge
  - Learning based on audit data

Models of Intrusion Detection

1. Anomaly detection
   - What is usual, is known
   - What is unusual, is bad
2. Misuse detection
   - What is bad, is known
   - What is not bad, is good
3. Specification-based detection
   - What is good, is known
   - What is not good, is bad

1. Anomaly Detection

Analyzes a set of characteristics of system, and compares their values with expected values; report when computed statistics do not match expected statistics

- Threshold metrics
- Statistical moments
- Markov model

Threshold Metrics

- Counts number of events that occur
- Between *m* and *n* events (inclusive) expected to occur
- If number falls outside this range, anomalous
- Example
  - Windows NT 4.0 and windows 10: lock user out after *k* failed sequential login attempts. Range is [0, *k*–1].
    - *k* or more failed logins deemed anomalous
    - k can be chosen by user, recommended 10, default is 0 !!
- Difficulties
  - Appropriate threshold may depend on non-obvious factors
    - Typing skill of users
    - If keyboards are US keyboards, and most users are French, typing errors very common

Statistical Moments

- Analyzer computes mean and standard deviation (first two moments), other measures of correlation (higher moments)
  - If measured values fall outside expected interval for particular moments, anomalous
- Potential problem
  - Profile may evolve over time; solution is to weigh data appropriately or alter rules to take changes into account

Markov Model

- Past state affects current transition
- Anomalies based upon *sequences* of events, and not on occurrence of single event
    - Over time, probability of transition developed
    - When transition with low probability occurs, event causing it considered anomalous
- Problem: need to train system to establish valid sequences
  - Use known training data that is not anomalous
  - The more training data, the better the model
  - Training data should cover *all* possible normal uses

Example: TIM

- Time-based Inductive Learning (Teng 1990)
- Learning
  - Training data is *abcdedeabcabc*
  - TIM derives following rules:
    - R_1: $ab \rightarrow c$ (1.0) R_2: $c \rightarrow d$ (0.5) R_3: $c \rightarrow a$ (0.5)
    - R_4: $d \rightarrow e$ (1.0) R_5: $e \rightarrow a$ (0.5) R_6: $e \rightarrow d$ (0.5)
- Detecting
  - Seen: *abd* triggers alert
    - *c* always follows *ab* in rule set
  - Seen: *acf* no alert as multiple events can follow c
    - May add rule R_7: $c \rightarrow f$ (0.33) and adjust R_2, R_3

Problems of Anomaly Detection

- False Positive: Anomaly activities that are not intrusive are classified as intrusive.
- False Negative: Intrusive activities that are not anomalous result in false negatives, that is events are not flagged intrusive, though they actually are.
- Computational expensive because of the overhead of keeping track of, and possibly updating several system profile metrics.

2. Misuse Modeling

- Determines whether a sequence of instructions being executed is known to violate the site security policy
  - Descriptions of known or potential exploits grouped into *rule sets*
  - IDS matches data against rule sets; on success, potential attack found
  - Commonly known as *signature systems*
- Cannot detect attacks unknown to developers of rule sets
  - No rules to cover them
  - Rule set must be continuously updated

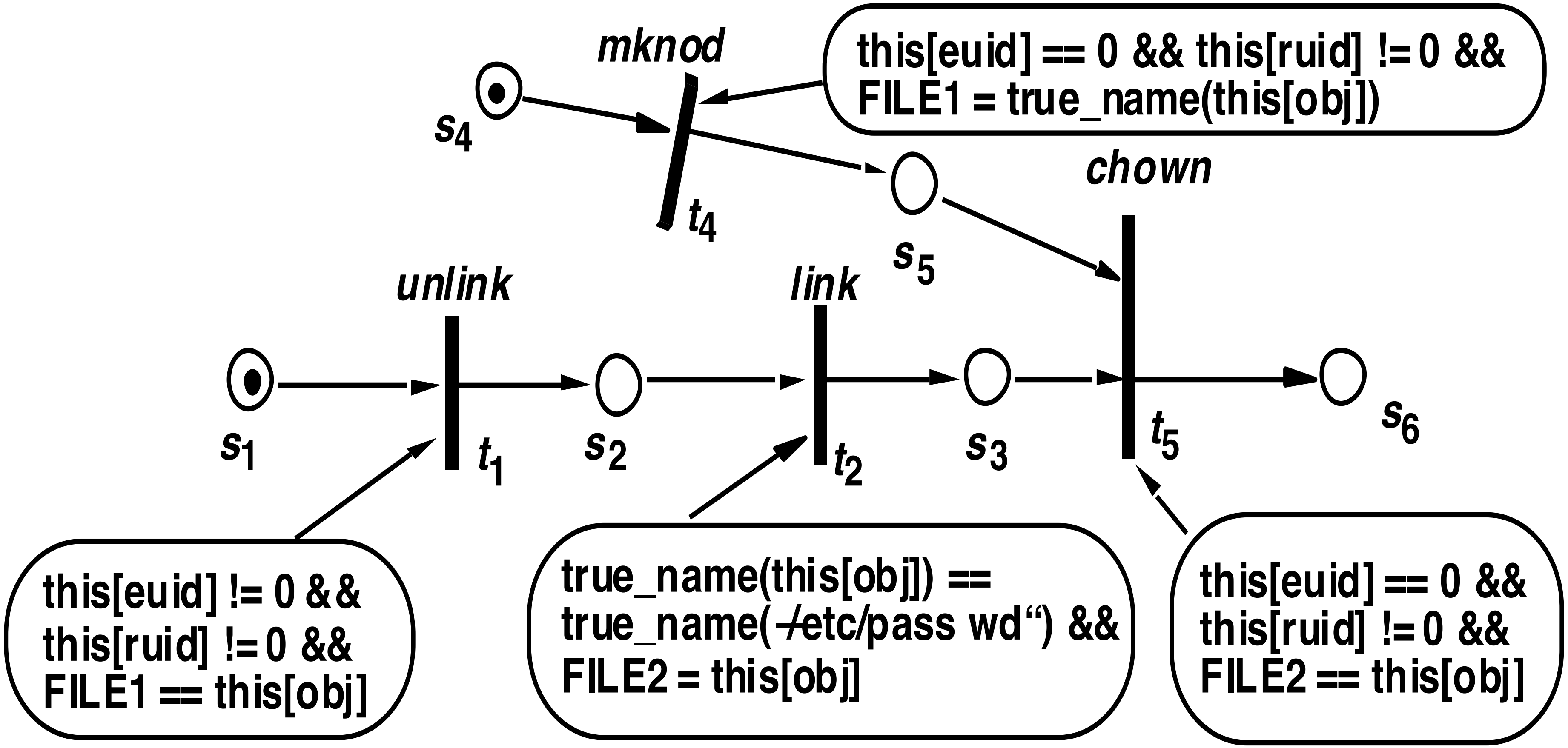
Example: IDIOT

- Event is a single action, or a series of actions, resulting in a single record and change of state
- Five categories of attacks:
  - *Existence*: attack creates file or other entity
  - *Sequence*: attack causes several events sequentially
  - *Partial order*: attack causes 2 or more sequences of events, and events form partial order under temporal relation
  - *Duration*: something exists for interval of time
  - *Interval*: events occur exactly *n* units of time apart

Example: IDIOT Representation

- Sequences of (attack) events may be interlaced with other events
- Use colored Petri nets to capture this
  - Each signature corresponds to a particular Colored Petri Automaton
  - Nodes are tokens; edges are transitions
  - Final state of the signature is compromised state
- Example: *mkdir* attack
  - Edges protected by guards (expressions)
  - Tokens move from node to node as guards satisfied

Example: IDIOT Analysis



Example: IDIOT Features

- New signatures can be added dynamically
  - Partially matched signatures need not be cleared and re-matched (info kept in state)
- Ordering the CPAs allows you to order the checking for attack signatures
  - Useful when you want a priority ordering
  - Can order initial branches of CPA to find sequences known to occur often

3. Specification Modeling

- Determines whether execution of sequence of instructions violates specification
- Only need to check programs that alter the protection state of system (potentially critical code).
  - ANY program executed by a privileged user is a potential security threat
- A formalization of what *should* happen (detects unknown attacks)
- Extra effort in analyzing program and specifying its behavior

Comparison and Contrast

- Misuse detection: if all policy rules known, easy to construct rulesets to detect violations
  - Usual case is that much of policy is unspecified, so rulesets describe attacks, and are not complete
- Anomaly detection: detects unusual events, but these are not necessarily security violations
- Specification-based vs. misuse:
  - Specification-based assumes if specifications followed, policy not violated;
  - misuse assumes if policy as embodied in rulesets followed, policy not violated
  - Spec-based=per-program, local
  - Misuse=site policy

Hybrid Methods

To resolve the disadvantages of the two conventional methods**, hybrid intrusion detection** methods combine the misuse method and the anomaly method

- detection performance of hybrid detection depends on the combination of the two detection methods

- Often hybrid detection systems independently train the two models, and then simply aggregate the results
    - consider an attack if **at least one** of the two models classifies the observation as an attack; detection rate improved but still a high false positive rate.
    - Better hybrid method regards observation as an attack only if **both models** classify it as an attack, reducing false alarms but possibly missing some

Key Performance Metrics

Algorithm/Model

- Alarm: A ; Intrusion: I
- Detection (true alarm) rate: P(A|I)
  - False negative rate P(¬A|I)
- False alarm rate: P(A|¬I)
  - True negative rate P(¬A|¬I)

Architecture

- Scalable
- Resilient to attacks

IDS Problem: *Base Rate Fallacy*

- IDS useless unless accurate
  - Significant fraction of intrusions detected
  - Significant number of alarms correspond to intrusions
- Assume 99% accuracy of intrusions detection system
  - 1% of non-intrusions generate alarm
  - 100 in 10,000 events are really intrusions
- Alarm sounds: is it a “real” intrusion?

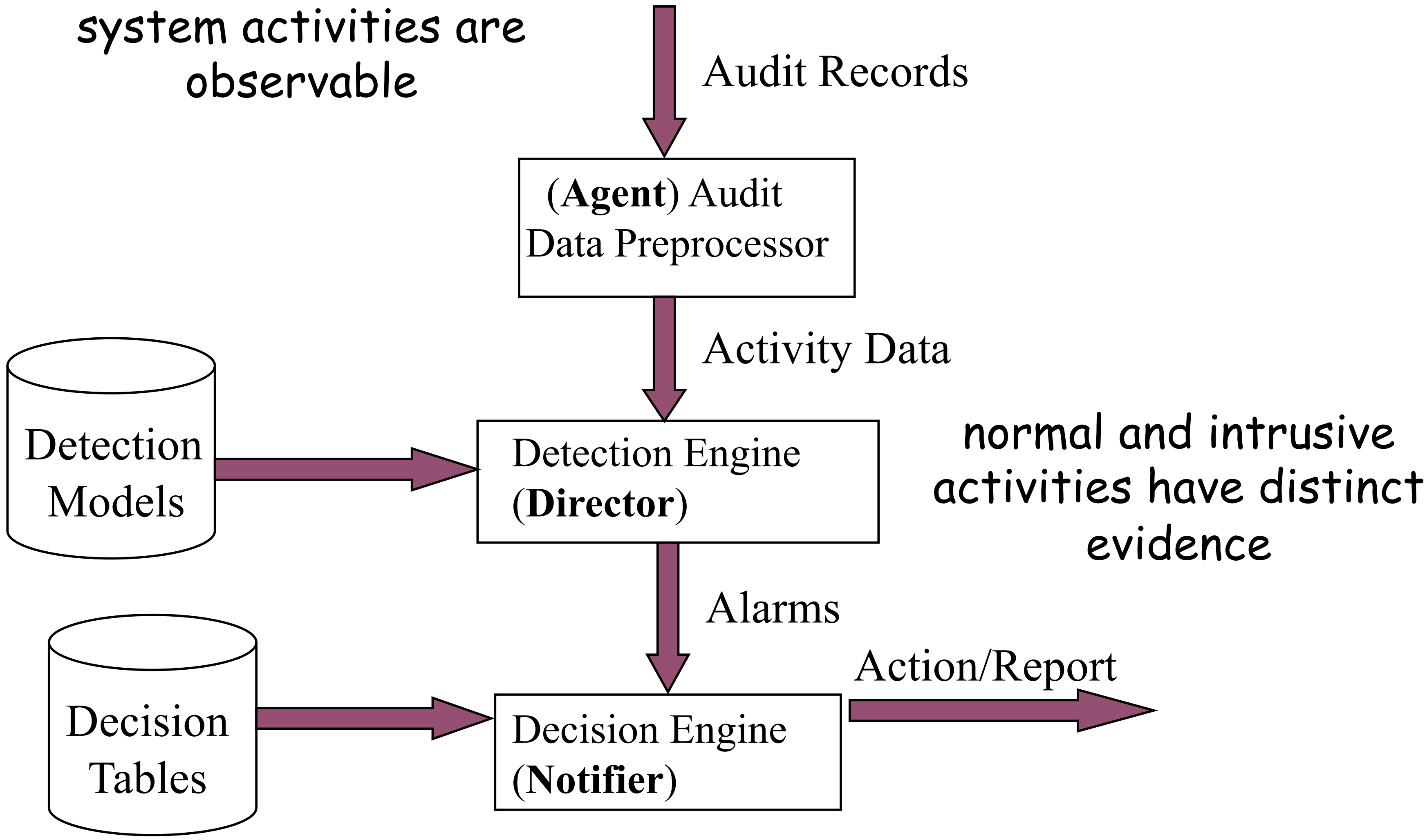
What if only 1 in 10,000 events is an intrusion?

IDS Architecture

Basically, a sophisticated audit system

- *Agent* gathers data for analysis
- *Director* analyzes data obtained from the agents according to its internal rules
- *Notifier* obtains results from director, and takes some action
  - May simply notify security officer
  - May reconfigure agents, director to alter collection, analysis methods
  - May activate response mechanism

Components of an IDS



Agents

- Obtains information and sends to director
- May put information into another form
  - Preprocessing of records to extract relevant parts
- May delete unneeded information
- Director may request agent to send other information

Example

- IDS uses failed login attempts in its analysis
- Agent scans login log every 5 minutes, sends director for each new login attempt:
  - Time of failed login
  - Account name and entered password
- Director requests all records of login (failed or not) for particular user
  - Suspecting a brute-force cracking attempt

Host-Based Agent

- Obtain information from logs
  - May use many logs as sources
  - May be security-related or not (accounting)
  - May be virtual logs if agent is part of the kernel
    - Very non-portable
- Agent may generate its information
  - Scans information needed by IDS, turns it into equivalent of log record
  - May generate own info. From state of system, typically for checking policy; may be very complex

Network-Based Agents

- Detects network-oriented attacks
  - Denial of service attack introduced by flooding a network
- Monitor traffic for a large number of hosts
- Examine the contents of the traffic itself
- Agent must have same view of traffic as destination
- End-to-end encryption defeats content monitoring
  - Not traffic analysis, though

Director

- Reduces information from agents
  - Eliminates unnecessary, redundant records
- Analyzes remaining information to determine if attack under way
  - Analysis engine can use a number of techniques, discussed before, to do this
- Usually run on separate system
  - not to impact performance of monitored systems
  - Rules, profiles not available to ordinary users

Example

- Jane logs in to perform system maintenance during the day
- She logs in at night to write reports
- One night she begins recompiling the kernel
- Agent #1 reports logins and logouts
- Agent #2 reports commands executed
  - Neither agent spots discrepancy
  - Director correlates log, spots it at once

Notifier

- Accepts information from director
- Takes appropriate action
  - Notify system security officer
  - Respond to attack
- Often GUIs
  - Well-designed ones use visualization to convey information

Types of Intrusion Detection Systems

Network-Based Intrusion Detection Systems

- Have the whole network as the monitoring scope, and monitor the traffic on the network to detect intrusions.
  - Can be run as an independent standalone machine that watches over all network traffic,
  - Or just monitor itself as the target machine to watch over its own traffic. (SYN-flood or a TCP port scan)

Types of Intrusion Detection Systems

Host-based Intrusion Detection Systems

- Misuse is not confined only to the “bad” outsiders but within organizations.
- Local inspection of systems is called HIDS to detect malicious activities on a single computer.
- Monitor operating system specific logs, including system, event, and security logs on Windows systems and syslog in Unix environments, to monitor sudden changes in these logs.
- They can be put on a remote host.

Advantage of NIDS

- Ability to detect attacks that a host-based system would miss because NIDSs monitor network traffic at a transport layer.
- Difficulty to remove evidence compared with HIDSs.
- Real-time detection and response. Real time notification allows for a quick and appropriate response.
- Ability to detect unsuccessful attacks and malicious intent.

Disadvantage of NIDS

- Blind spots. Deployed at the border of an organization network, NIDS are blink to the whole inside network.
- Encrypted data. NIDSs have no capabilities to decrypt encrypted data.

Advantage of HIDS

- Ability to verify success or failure of an attack quickly because they log continuing events that have actually occurred, have less false positive than their cousins.
- Low level monitoring. Can see low-level activities such as file accesses, changes to file permissions, attempts to install new executables or attempts to access privileged services, etc.
- Almost real-time detection and response.
- Ability to deal with encrypted and switched environment.
- Cost effectiveness. No additional hardware is needed to install HIDS.

Disadvantage of HIDS

- Myopic viewpoint. Since they are deployed at a host, they have a very limited view of the network.
- Since they are close to users, they are more susceptible to illegal tempering.

Combining Sources: DIDS

- Neither network-based nor host-based monitoring sufficient to detect some attacks
  - Attacker tries to telnet into system several times using different account names: network-based IDS detects this, but not host-based monitor
  - Attacker tries to log into system using an account without password: host-based IDS detects this, but not network-based monitor
- combined bring to the security own strengths and weaknesses that complement and augment the security of the network.
- success depends to on how well the interface receives and distributes the incidents and integrates the reporting structure between the different types of sensors in the HIDS and NIDS

Intrusion Response

If an intrusion is detected, how to protect the system.

- Goal:
  - Minimize the damage of attack
  - Thwart intrusion
  - Attempt to repair damages
- Phases
  - Incident Prevention
  - Intrusion Handling
    1. Containment Phase
    2. Eradication Phase
    3. Follow-Up phase

Incident Response Team

An *incident response team* (IRT) is a primary and centralized group of dedicated people charged with the responsibility of being the first contact team whenever an incidence occurs.

An IRT must have the following responsibilities:

- keeping up-to-date with the latest threats and incidents,
- being the main point of contact for incident reporting,
- notifying others whenever an incident occurs,
- assessing the damage and impact of every incident,
- finding out how to avoid exploitation of the same vulnerability, and
- recovering from the incident.

Incident Prevention

- Identify attack *before* it completes, ideally
- Prevent it from completing
- Jails useful for this
  - Attacker placed in a confined environment that looks like a full, unrestricted environment
  - Attacker may download files, but gets bogus ones
  - Can imitate a slow system, or an unreliable one
  - Useful to figure out what attacker wants
  - Multilevel secure systems are excellent places to implement jails.

Intrusion Handling

- Restoring system to satisfy site security policy
- Six phases
  - *Preparation* for attack (before attack detected)
  - *Identification* of attack
  - Containment of attack: limit access of attacker to resources
    - Passive monitoring
    - Constraining access
  - Eradication of attack (stop attack)
  - Recovery from attack (restore system to secure state)
  - Follow-up to attack (analysis and other actions)

Passive Monitoring

- Records attacker's actions; does *not* interfere with attack
  - Idea is to find out what the attacker is after and/or methods the attacker is using
- Problem: attacked system is vulnerable throughout
  - Attacker can also attack other systems
- Example: type of operating system can be derived from settings of TCP and IP packets of incoming connections
  - Analyst draws conclusions about source of attack

Constraining Actions

- Reduce protection domain of attacker
- Problem: if defenders do not know what attacker is after, reduced protection domain may contain what the attacker is after
  - Clifford Stoll created document that attacker downloaded from military computer system
  - Download took several hours, during which the phone call was traced to Germany before data sold to KGB

Deception

Deception Tool Kit

- Creates false network interface
- Can present any network configuration to attackers
- When probed, can return wide range of vulnerabilities
- Attacker wastes time attacking non-existent systems while analyst collects and analyzes attacks to determine goals and abilities of attacker
- Experiments show deception is effective response to keep attackers from targeting real systems

Honey Pots / Honey Nets

- Honey Pots
  - Decoy systems
  - Lure potential attackers away from critical systems
  - Encourages attacks against themselves
- Honey Net
  - Collection of honey pots
  - Connects honey pots on a subnet
  - Contains pseudo-services that emulated well-known services
  - Filled with fake information

Eradication Phase

Usual approach: deny or remove access to system, or terminate processes involved in attack

Use wrappers to implement access control

- Example: wrap system calls
  - On invocation, wrapper takes control of process
  - Wrapper can log call, deny access, do intrusion detection
  - Experiments focusing on intrusion detection used multiple wrappers to terminate suspicious processes
- Example: network connections
  - Wrapper around servers log, do access control on incoming connections and control access to Web-based databases

IDS Tools

- Snort http://www.snort.org/
  - NIDS combining the benefits of signature, protocol, and anomaly-based inspection
- Honeypot, http://www.honeyd.org
  - A honeypot is a system designed to look like something that an intruder can hack.
  - The goal is to deceive intruders and learn from them without compromising the security of the network.
- IPAudit, http://ipaudit.sourceforge.net/
  - NIDS monitors network activity by host, protocol and port

Categories of IDSs

There are several ways to distinguish/classify IDS:

- Is the system *dynamic* or *static* ?
  - i.e., does it continuously gather data, or looks at snapshots
- Is the system misuse- or specification- or anomaly-based?
  - knows what ‘unacceptable’ looks like, or what ‘acceptable’ looks like?
- Is the system integrated with defenses, primarily investigatory, or used for retaliation?
- Is the control centralized, partially distributed or fully distributed?
- Is the data gathered from the host, the network, or a combination?