Chapter 22: Intrusion Detection

- Principles
- Basics
- Models of Intrusion Detection
- Architecture of an IDS
- Organization
- Incident Response

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 allowed to do
- Systems under attack do not meet at least one of these

Example

- Goal: insert a back door into a system
  - Intruder will modify system configuration file or program
  - Requires privilege; attacker enters system as an unprivileged user and must acquire privilege
    - Nonprivileged user may not normally acquire privilege (violates #1)
    - Attacker may break in using sequence of commands that violate security policy (violates #2)
    - Attacker may cause program to act in ways that violate program's specification

Basic Intrusion Detection

- *Attack tool* is automated script designed to violate a security policy
- Example: *rootkit*
  - Includes password sniffer
  - Designed to hide itself using Trojaned versions of various programs (*ps*, *ls*, *find*, *netstat*, etc.)
  - Adds back doors (*login*, *telnetd*, etc.)
  - Has tools to clean up log entries (*zapper, etc.)*

Detection

- *Rootkit* configuration files cause *ls*, *du*, etc. to hide information
  - *ls* lists all files in a directory
    - Except those hidden by configuration file
  - *dirdump* (local program to list directory entries) lists them too
    - Run both and compare counts
    - If they differ, *ls* is doctored
- Other approaches possible

Key Point

- *Rootkit* does *not* alter kernel or file structures to conceal files, processes, and network connections
  - It alters the programs or system calls that *interpret* those structures
  - Find some entry point for interpretation that *rootkit* did not alter
  - The inconsistency is an anomaly (violates #1)

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

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 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

Models of Intrusion Detection

- Anomaly detection
  - What is usual, is known
  - What is unusual, is bad
- Misuse detection
  - What is bad, is known
  - What is not bad, is good
- Specification-based detection
  - What is good, is known
  - What is not good, is bad

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: lock user out after *k* failed sequential login attempts. Range is (0, *k*–1).
    - *k* or more failed logins deemed anomalous

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
    - Dvorak vs. non-Dvorak within the US

Statistical Moments

- Analyzer computes 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

Example: IDES

- Developed at SRI International to test Denning's model
  - Represent users, login session, other entities as ordered sequence of statistics $<q_{0,j}, \ldots, q_{n,j}>$
  - $q_{i,j}$ (statistic i for day j) is count or time interval
  - Weighting favors recent behavior over past behavior
    - $A_{k,j}$ sum of counts making up metric of kth statistic on jth day
    - $q_{k,l+1} = A_{k,l+1} - A_{k,l} + 2^{-rt} q_{k,l}$ where t is number of log entries/total time since start, r factor determined through experience

Potential Problems

- Assumes behavior of processes and users can be modeled statistically
  - Ideal: matches a known distribution such as Gaussian or normal
  - Otherwise, must use techniques like clustering to determine moments, characteristics that show anomalies, etc.
- Real-time computation a problem too

Markov Model

- Past state affects current transition
- Anomalies based upon *sequences* of events, and not on occurrence of single event
- 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 of system

Example: TIM

- Time-based Inductive Learning
- Sequence of events is *abcdedeabcabc*
- TIM derives following rules:

  R_1: $ab \rightarrow c$ (1.0) R_2: $c \rightarrow d$ (0.5) R_3: $c \rightarrow e$ (0.5)

  R_4: $d \rightarrow e$ (1.0) R_5: $e \rightarrow a$ (0.5) R_6: $e \rightarrow d$ (0.5)
- 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); adjust R_2, R_3

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
- Cannot detect attacks unknown to developers of rule sets
  - No rules to cover them

Example: NFR

- Built to make adding new rules easily
- Architecture:
  - Packet sucker: read packets from network
  - Decision engine: uses filters to extract information
  - Backend: write data generated by filters to disk
    - Query backend allows administrators to extract raw, postprocessed data from this file
    - Query backend is separate from NFR process

N-Code Language

- Filters written in this language
- Example: ignore all traffic not intended for 2 web servers:

```
# list of my web servers
my_web_servers = [ 10.237.100.189 10.237.55.93 ] ;
# we assume all HTTP traffic is on port 80
filter watch tcp ( client, dport:80 )
{
     if (ip.dest != my_web_servers)
             return;
# now process the packet; we just write out packet info
     record system.time, ip.src, ip.dest to www._list;
}
www_list = recorder("log")
```

Specification Modeling

- Determines whether execution of sequence of instructions violates specification
- Only need to check programs that alter protection state of system

System Traces

- Notion of *subtrace* (subsequence of a trace) allows you to handle threads of a process, process of a system
- Notion of *merge of traces U, V* when trace *U* and trace *V* merged into single trace
- *Filter p* maps trace *T* to subtrace *T′* such that, for all events $t_i \in T'$, $p(t_i)$ is true

Example: Apply to *rdist*

- Ko, Levitt, Ruschitzka defined PE-grammar to describe accepted behavior of program
- *rdist* creates temp file, copies contents into it, changes protection mask, owner of it, copies it into place
  - Attack: during copy, delete temp file and place symbolic link with same name as temp file
  - *rdist* changes mode, ownership to that of program

Relevant Parts of Spec

- Specification of *chmod*, *chown* says that they can only alter attributes of files that *rdist* creates
- *Chown*, *chmod* of symlink violates this rule as M.newownerid ≠ U (owner of file symlink points to is not owner of file *rdist* is distributing)

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 problems
- Specification-based vs. misuse: spec assumes if specifications followed, policy not violated; misuse assumes if policy as embodied in rulesets followed, policy not violated

IDS Architecture

- Basically, a sophisticated audit system
  - *Agent* like logger; it gathers data for analysis
  - *Director* like analyzer; it 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

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 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
  - May be virtual logs if agent is part of the kernel
    - Very non-portable
- Agent generates its information
  - Scans information needed by IDS, turns it into equivalent of log record
  - Typically, check 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
  - TTL tricks, fragmentation may obscure this
- End-to-end encryption defeats content monitoring
  - Not traffic analysis, though

Network Issues

- Network architecture dictates agent placement
  - Ethernet or broadcast medium: one agent per subnet
  - Point-to-point medium: one agent per connection, or agent at distribution/routing point
- Focus is usually on intruders entering network
  - If few entry points, place network agents behind them
  - Does not help if inside attacks to be monitored

Aggregation of Information

- Agents produce information at multiple layers of abstraction
  - Application-monitoring agents provide one view (usually one line) of an event
  - System-monitoring agents provide a different view (usually many lines) of an event
  - Network-monitoring agents provide yet another view (involving many network packets) of an event

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
  - Does not 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

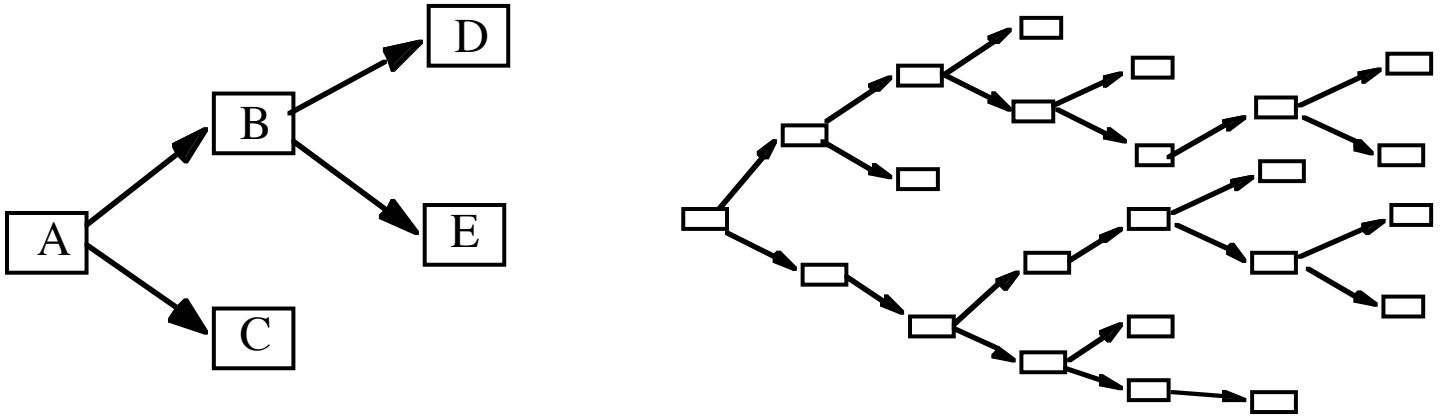
Adaptive Directors

- Modify profiles, rule sets to adapt their analysis to changes in system
  - Usually use machine learning or planning to determine how to do this
- Example: use neural nets to analyze logs
  - Network adapted to users' behavior over time
  - Used learning techniques to improve classification of events as anomalous
    - Reduced number of false alarms

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

GrIDS GUI



- GrIDS interface showing the progress of a worm as it spreads through network
- Left is early in spread
- Right is later on

Other Examples

- Courtney detected SATAN attacks
  - Added notification to system log
  - Could be configured to send email or paging message to system administrator
- IDIP protocol coordinates IDSes to respond to attack
  - If an IDS detects attack over a network, notifies other IDSes on co-operative firewalls; they can then reject messages from the source

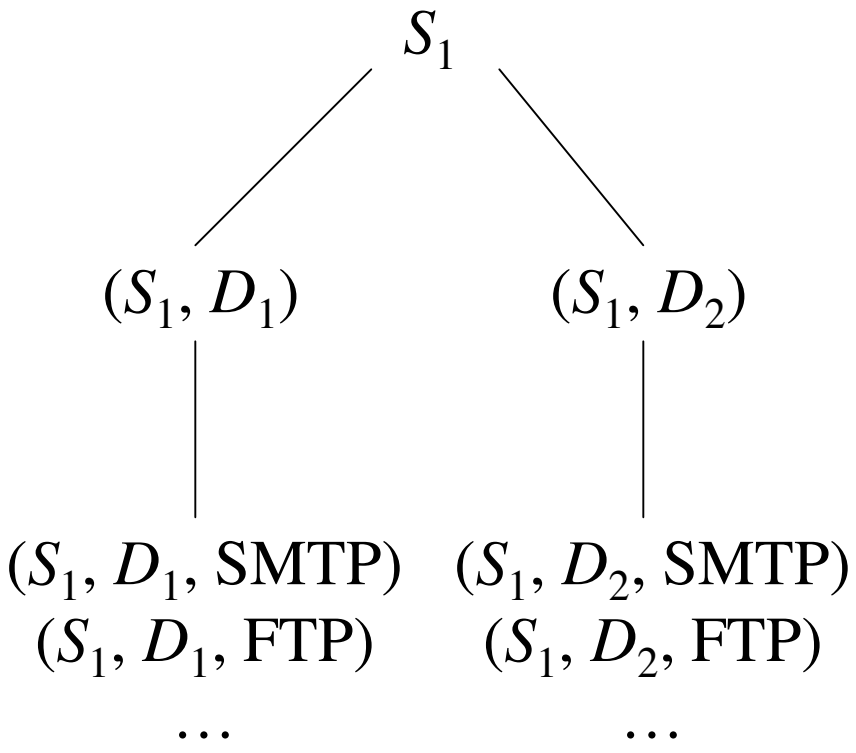
Organization of an IDS

- Monitoring network traffic for intrusions
  - NSM system
- Combining host and network monitoring
  - DIDS
- Making the agents autonomous
  - AAFID system

Monitoring Networks: NSM

- Develops profile of expected usage of network, compares current usage
- Has 3-D matrix for data
  - Axes are source, destination, service
  - Each connection has unique *connection ID*
  - Contents are number of packets sent over that connection for a period of time, and sum of data
  - NSM generates expected connection data
  - Expected data masks data in matrix, and anything left over is reported as an anomaly

Problem



- Too much data!
  - Solution: arrange data hierarchically into groups
    - Construct by folding axes of matrix
  - Analyst could expand any group flagged as anomalous

Signatures

- Analyst can write rule to look for specific occurrences in matrix
  - Repeated telnet connections lasting only as long as set-up indicates failed login attempt
- Analyst can write rules to match against network traffic
  - Used to look for excessive logins, attempt to communicate with non-existent host, single host communicating with 15 or more hosts

Other

- Graphical interface independent of the NSM matrix analyzer
- Detected many attacks
  - But false positives too
- Still in use in some places
  - Signatures have changed, of course
- Also demonstrated intrusion detection on network is feasible
  - Did no content analysis, so would work even with encrypted connections

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
- DIDS uses agents on hosts being monitored, and a network monitor
  - DIDS director uses expert system to analyze data

Attackers Moving in Network

- Intruder breaks into system A as *alice*
- Intruder goes from A to system B, and breaks into B's account *bob*
- Host-based mechanisms cannot correlate these
- DIDS director could see *bob* logged in over *alice*'s connection; expert system infers they are the same user
  - Assigns *network identification number* NID to this user

Handling Distributed Data

- Agent analyzes logs to extract entries of interest
  - Agent uses signatures to look for attacks
    - Summaries sent to director
  - Other events forwarded directly to director
- DIDS model has agents report:
  - Events (information in log entries)
  - Action, domain

Actions and Domains

- Subjects perform actions
  - session_start, session_end, read, write, execute, terminate, create, delete, move, change_rights, change_user_id
- Domains characterize objects
  - tagged, authentication, audit, network, system, sys_info, user_info, utility, owned, not_owned
  - Objects put into highest domain to which it belongs
    - Tagged, authenticated file is in domain tagged
    - Unowned network object is in domain network

More on Agent Actions

- Entities can be subjects in one view, objects in another
  - Process: subject when changes protection mode of object, object when process is terminated
- Table determines which events sent to DIDS director
  - Based on actions, domains associated with event
  - All NIDS events sent over so director can track view of system
    - Action is *session_start* or *execute*; domain is *network*

Layers of Expert System Model

1. Log records
2. Events (relevant information from log entries)
3. Subject capturing all events associated with a user; NID assigned to this subject
4. Contextual information such as time, proximity to other events
   - Sequence of commands to show who is using the system
   - Series of failed logins follow

Top Layers

5. Network threats (combination of events in context)
   - Abuse (change to protection state)
   - Misuse (violates policy, does not change state)
   - Suspicious act (does not violate policy, but of interest)
6. Score (represents security state of network)
   - Derived from previous layer and from scores associated with rules
     - Analyst can adjust these scores as needed
   - A convenience for user

Autonomous Agents: AAFID

- Distribute director among agents
- *Autonomous agent* is process that can act independently of the system of which it is part
- Autonomous agent performs one particular monitoring function
  - Has its own internal model
  - Communicates with other agents
  - Agents jointly decide if these constitute a reportable intrusion

Advantages

- No single point of failure
  - *All* agents can act as director
  - In effect, director distributed over all agents
- Compromise of one agent does not affect others
- Agent monitors one resource
  - Small and simple
- Agents can migrate if needed
- Approach appears to be scalable to large networks

Disadvantages

- Communications overhead higher, more scattered than for single director
  - Securing these can be very hard and expensive
- As agent monitors one resource, need many agents to monitor multiple resources
- Distributed computation involved in detecting intrusions
  - This computation also must be secured

Example: AAFID

- Host has set of agents and transceiver
  - Transceiver controls agent execution, collates information, forwards it to monitor (on local or remote system)
- Filters provide access to monitored resources
  - Use this approach to avoid duplication of work and system dependence
  - Agents subscribe to filters by specifying records needed
  - Multiple agents may subscribe to single filter

Transceivers and Monitors

- Transceivers collect data from agents
  - Forward it to other agents or monitors
  - Can terminate, start agents on local system
    - Example: System begins to accept TCP connections, so transceiver turns on agent to monitor SMTP
- Monitors accept data from transceivers
  - Can communicate with transceivers, other monitors
    - Send commands to transceiver
  - Perform high level correlation for multiple hosts
  - If multiple monitors interact with transceiver, AAFID must ensure transceiver receives consistent commands

Other

- User interface interacts with monitors
  - Could be graphical or textual
- Prototype implemented in PERL for Linux and Solaris
  - Proof of concept
  - Performance loss acceptable

Incident Prevention

- Identify attack *before* it completes
- 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
  - MLS systems provide natural jails

Intrusion Handling

- Restoring system to satisfy site security policy
- Six phases
  - *Preparation* for attack (before attack detected)
  - *Identification* of attack
    - *Containment* of attack (confinement)
    - *Eradication* of attack (stop attack)
  - *Recovery* from attack (restore system to secure state)
    - *Follow-up* to attack (analysis and other actions)
- Discussed in what follows

Containment Phase

- Goal: limit access of attacker to system resources
- Two methods
  - Passive monitoring
  - Constraining access

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
  - Stoll created document that attacker downloaded
  - Download took several hours, during which the phone call was traced to Germany

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

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

Firewalls

- Mediate access to organization's network
  - Also mediate access out to the Internet
- Example: Java applets filtered at firewall
  - Use proxy server to rewrite them
    - Change "<applet>" to something else
  - Discard incoming web files with hex sequence CA FE BA BE
    - All Java class files begin with this
  - Block all files with name ending in ".class" or ".zip"
    - Lots of false positives

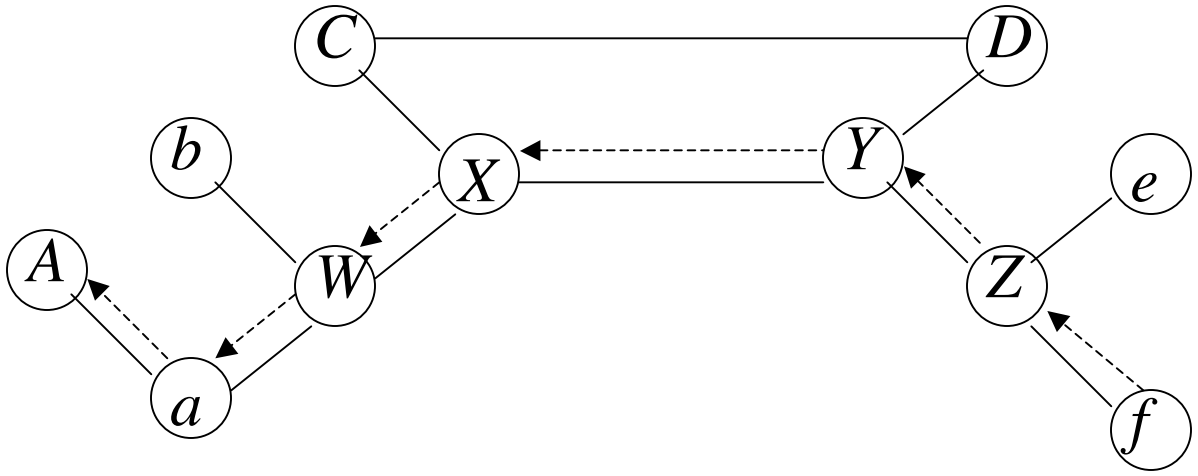
Intrusion Detection and Isolation Protocol

- Coordinates reponse to attacks
- *Boundary controller* is system that can block connection from entering perimeter
  - Typically firewalls or routers
- *Neighbor* is system directly connected
- *IDIP domain* is set of systems that can send messages to one another without messages passing through boundary controller

Protocol

- IDIP protocol engine monitors connection passing through members of IDIP domains
  - If intrusion observed, engine reports it to neighbors
  - Neighbors propagate information about attack
  - Trace connection, datagrams to boundary controllers
  - Boundary controllers coordinate responses
    - Usually, block attack, notify other controllers to block relevant communications

Example



- *C*, *D*, *W*, *X*, *Y*, *Z* boundary controllers
- *f* launches flooding attack on *A*
- Note after X xuppresses traffic intended for *A*, *W* begins accepting it and *A*, *b*, *a*, and *W* can freely communicate again

Follow-Up Phase

- Take action external to system against attacker
  - Thumbprinting: traceback at the connection level
  - IP header marking: traceback at the packet level
  - Counterattacking

Counterattacking

- Use legal procedures
  - Collect chain of evidence so legal authorities can establish attack was real
  - Check with lawyers for this
    - Rules of evidence very specific and detailed
    - If you don't follow them, expect case to be dropped
- Technical attack
  - Goal is to damage attacker seriously enough to stop current attack and deter future attacks

Consequences

1. May harm innocent party
   - Attacker may have broken into source of attack or may be impersonating innocent party
2. May have side effects
   - If counterattack is flooding, may block legitimate use of network
3. Antithetical to shared use of network
   - Counterattack absorbs network resources and makes threats more immediate
4. May be legally actionable

Example: Counterworm

- Counterworm given signature of real worm
  - Counterworm spreads rapidly, deleting all occurrences of original worm
- Some issues
  - How can counterworm be set up to delete *only* targeted worm?
  - What if infected system is gathering worms for research?
  - How do originators of counterworm know it will not cause problems for any system?
    - And are they legally liable if it does?

Key Points

- Intrusion detection is a form of auditing
- Anomaly detection looks for unexpected events
- Misuse detection looks for what is known to be bad
- Specification-based detection looks for what is known not to be good
- Intrusion response requires careful thought and planning