Host Level IDS

CSC 790

WAKE FOREST
UNIVERSITY
Department of Computer Science

Spring 2014

IDS Taxonomy

- IDS can be categorized based on...
  - What they look for (measure normal or look for badness)
  - Where they are located (network, host, etc...)
  - What they use to detect (simple statistics, string matching, machine learning, etc...)
  - When detection occurs (real-time or off-line)
  - Architecture (centralized or distributed)

IDS Categories Based on Events (*as a reminder*)

- IDS can be categorized based on the use of event streams
  - **Anomaly detection** or **misuse detection**
- **Anomaly detection** attempts to find *abnormal behavior*
  - Must first define *normal behavior* (based on history)
  - System attempts to identify patterns of activity that deviate
  - *Recognize normal events, not an attack*
- **Misuse detection** is the complement of anomaly detection
  - Have *known* attack descriptions (**signatures**)
  - Events stream are constantly matched against the signatures
  - *Don't recognize normal, know attack events*

IDS Categories Based on Scope

- Can further categorize IDS based on *scope*: **network** or **host**
- **Host** implemented on a single machine
  - Only responsible for the host on which it resides
  - Maintains/observes audit files, system calls, etc...
  - For example Tripwire and OSSEC
- **Network** implemented in a centralized or distributed fashion
  - Only responsible for the network
  - Measures traffic and/or scans packet data
  - For example Network Flight Recorder (NFR)
- *Neither category is comprehensive... only applicable to certain types of attacks*

Pros and Cons

	Signature-based (knowledge-based)	Anomaly-based (behavior-based)	Stateful protocol analysis (specification-based)
Pros	Simple for known attacks Detail contextual analysis	Effective for new attacks Less dependent on OS Can detect privilege abuse	Know and trace protocol states Detects unexpected command seq
Cons	Ineffective for unknown attacks Little understanding of states and protocols Signatures/patterns updating Time consuming	Normal can change Unavailable during learning Difficult to trigger alerts	Resource consuming Can't detect masquerade attacks

IDS Design Principles

Anomaly	Self-learning	Non-time series	Rule modeling	Observe the traffic and formulate rules that describe the normal operation of the system
			Descriptive stats	Mono-modal stats from system parameters for a profile, use a distance vector for observed and profile
		Time series	ANN	Artificial neural network to learn normal
	Programmed	Descriptive stats	Simple stats	Program certain stats to baseline
			Simple rule-based	More complex (chained) programmed stats
			Threshold	Stupid simple stats
		Default deny	State series model	Program required states for normal use
Signature	Programmed	State model	State transition	Encode the intrusion as a number of different states
			Petri-net	States form a petri-net (tree)
		Expert system		Expert system is employed to reason about the security state of the system, given rules that describe intrusive behavior
		String matching		Substring matching of the characters in text that is transmitted between systems
		Simple rule-based		Similar to expert system, faster but not as advanced
Signature inspired	self-learning		Feature selection	Similar to the more powerful expert system, but not as advanced

Intrusion Categories

- **Well known intrusions**
  - Intrusions that are *well known* (static)
  - Intrusions are simple to execute and have little variability
- **Generalizable intrusions**
  - Intrusions are similar to the well known intrusions, but have a larger or smaller degree of variability
  - Intrusions often exploit more general flaws
- **Unknown intrusions**
  - Intrusions have the weakest coupling to a specific flaw, or one that is very general in nature
  - Intrusion detection system does not really know what to expect

IDS Characteristics

- **Time of detection**
  - Detect intrusions in (near) real-time, or process audit data with some delay
- **Granularity of data-processing**
  - Process data continuously, or process data in batches at regular intervals
- **Source of audit data**
  - Network data or host data
- **Response to detected intrusions**
  - Passive, respond by notifying the proper authority, and they do not themselves try to mitigate
  - Active, can be categorized as exercising control over the protected system, or exercising control over the attacker

- **Locus of data-processing**
  - Audit data can either be processed in a central location, irrespective of whether the data originates from onepossibly the samesite or is collected and collated from many different sources in a distributed fashion
- **Locus of data-collection**
  - Audit data for the processor can be collected from many different sources (distributed), or from a single source (centralized)
- **Security**
  - Ability to withstand hostile attack against the intrusion detection system itself
- **Degree of inter-operability**
  - Degree that the system can operate in conjunction with other IDS and accept audit data from different sources

Network and Host IDS

- Network IDS (NIDS) uses information from the network
  - Sniffing packets and/or considering traffic patterns
- Host IDS (HIDS) uses information from the host
  - Process information, resource utilization
  - Log files, file contents, file attributes
  - Received or sent network traffic
- Therefore a difference is the visibility of the IDS

Advantages of HIDS (*supposedly*)

- Potentially lower false positive rate
  - If the HIDS observes any traffic, it was legitimately accepted by the host (not simply scanning traffic or exploits blindly sent)
- Leverages existing hardware (whatever that means...)
- Not resource intensive (really? now this sounds like an advertisement)
- No interruption/complication to network infrastructure (OK, legit)

Disadvantages of HIDS

- Potential *blind spots*
  - In many cases, if an event is not logged or stored in the file system, it's invisible
  - Cannot parse unknown traffic (e.g. new protocols)
- Requires installation of HIDS application on every host
- HIDS may introduce vulnerabilities

Open Source SECurity

- Open Source SECurity (OSSEC) is an example HIDS
  - First release in 2005
  - Initially used for integrity check (log analysis) *nix systems
  - Can scale to 100+ hosts (initially used for servers)
  - In 2009, purchased on Trend Micro, but still support open source
  - Currently operates on Windows, Solaris, Linus, BSD. etc...
- Performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response
  - Since log based, also referred to as Log-Based IDS (LIDS)

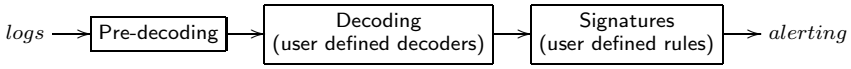
OSSEC Models

- Two operation models
  - Local, just protect one system
  - Client-server, for centralized analysis
- Client-server model operation
  - Clients receive configuration from server
  - Server receives (compress and encrypted) logs from the clients

OSSEC Log Monitoring

- Monitors (apply ruleset to) certain logs by default
  - Syslog
  - Apache HTTP logs
  - Mail logs
  - Delicious pecan log rolls
- Can be configured to monitor any log it has access to (can operate at different privileges)

OSSEC Log Flow



- Once log information arrives, three processes are applied
  - Pre-decoding - extract known fields (time, hostnames, etc...)
  - Decoding - extract information based on user-defined expressions
  - Signatures - Apply user-defined rules to extracted information

Log Pre-Decoding

- Extract generic information from logs
  - Hostname, program/process name, and time from syslog
  - Logs must be well formatted (*no laughing please*)

```
Mar 11 09:10:49 fender sshd[3234]:  Accepted password for whitenpm from 10.104.251.77 port 39210 ssh2
Mar 11 09:10:49 fender sshd[3234]:  pam_unix(sshd:session):  session opened for user whitenpm by (uid=0)
Mar 11 09:15:01 fender CRON[3478]:  pam_unix(cron:session):  session opened for user root by (uid=0)
Mar 11 09:15:01 fender CRON[3478]:  pam_unix(cron:session):  session closed for user root
Mar 11 09:15:18 fender sshd[3234]:  pam_unix(sshd:session):  session closed for user astley
Mar 11 09:15:18 fender sshd[3234]:  pam_unix(sshd:session):  Steven you should try http:\\goo.gle\QMET
```

- Decoded form for the first SSHD message above,

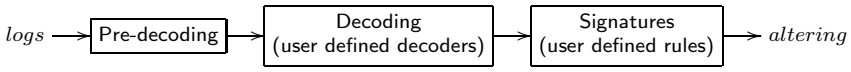
```
time/date -> Mar 11 09:10:49
hostname -> fender
program_name -> pam_unix
log -> Accepted password for whitenpm from 10.104.251.77 port 39210 ssh2
```


Log Decoding

- Process key information from the logs
  - OSSEC includes several prepackaged decoders
  - Operator can define custom decoders
- Create a decoder that extracts user name and source IP from SSHD
  - Assume `sshd` was pre-decoded and stored as `program_name`

```
<decoder name="sshd-success">
   <program_name>sshd</program_name>
   <regex>^Accepted \S+ for (\S+) from (\S+) port </regex>
   <order>user, srcip</order>
</decoder>
```

Log Rules



- After decoding, rules are applied
  - OSSEC includes +600 rules
  - Internally stored as a tree
  - User-based rule definitions done in XML
  - Matches done on decoded log information
- Two types of rules
  - Atomic rules are based on a single event
  - Composite rules are based on patterns across multiple logs

Default (Included) Rules (partial list)

Rule Name	Description
apache_rules.xml	Apache HTTP server rules
arpwatch_rules.xml	Arpwatch rules
attack_rules.xml	Common attack rules
cisco-ios_rules.xml	Cisco IOS firmware rules
courier_rules.xml	Courier mail server rules
Firewall_rules.xml	Common firewall rules
ftpd_rules.xml	Rules for the ftpd daemon
hordeimp_rules.xml	Horde Internet Messaging Program rules
ids_rules.xml	Common IDS rules
imapd_rules.xml	Rules for the imapd daemon
local_rules.xml	OSSEC HIDS local, user-defined rules
mailscanner_rules.xml	Common mail scanner rules
msauth_rules.xml	Microsoft Authentication rules
vms-exchange_rules.xml	Microsoft Exchange server rules
netscreenfw_rules.xml	Juniper Netscreen firewall rules
ms_ftpd_rules.xml	Microsoft FTP server rules
mysql_rules.xml	MySQL database rules
named_rules.xml	Rules for the named daemon
ossec_rules.xml	Common OSSEC HIDS rules
pam_rules.xml	Pluggable Authentication Module (PAM) rules
pix_rules.xml	Cisco PIX firewall rules
policy_rules.xml	Policy specific event rules
postfix_rules.xml	Postfix mail transfer agent rules
postgresql_rules.xml	PostgreSQL database rules

Simple Rules 411

- Basic rules meed the following
  - Rule id (any positive integer)
  - Rule level, from 0 (lowest, actually ignored) to 15 (highest)
  - Pattern, which can be written in regex form
- Consider a simple rule to log every message decoded as `sshd`

```
<rule id = "515" level = "5">
<decoded_as>sshd</decoded_as>
<description>Logging every decoded sshd message</description>
</rule>
```

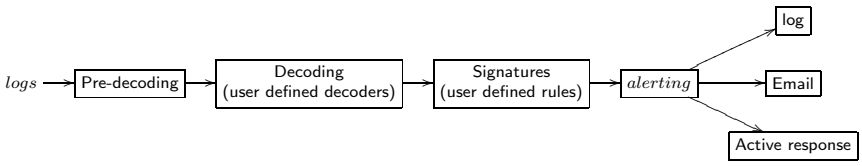

Dependent Rules

```
<rule id = "515" level = "5">
<decoded_as>sshd</decoded_as>
<description>Logging every decoded sshd message</description>
</rule>

<rule id="525" level="7">
<if_sid>515</if_sid>
<match>^Failed password</match>
<description>Failed password attempt</description>
</rule>

<rule id="535" level="13">
<if_sid>525</if_sid>
<hostname>^fender</hostname>
<srcip>!192.168.2.0/24</srcip>
<description>Higher severity!  Login failure on fender, beardie is about...</description>
</rule>
```

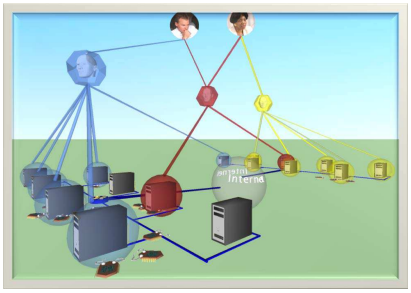
OSSEC Rule Matches



- If rule triggered then OSSEC can
  - Log (default behavior)
  - Send email and/or SMS alerts
  - Execute a response script

```
<rule id="100002" level="2">
   <if_sid>601</if_sid>
   <decoded_as>ar_log</decoded_as>
   <options>alert_by_email</options>
   <group>active_response_notification</group>
   <action>firewall-drop.sh</action>
   <status>add</status>
   <description>Active response firewall-drop.sh was run, host blocked</description>
</rule>
```


Swarm Intelligence (*PNNL Project*)



- Defense using swarm intelligence and simple software agents
  - Swarm of digital ants, each finds evidence per machine
  - Group of findings will indicate the actual problem
  - Movement based on pheromone, swarm an infected machine
- Better (faster and more robust) than having an IDS per machine?

Swarm Design

- Actually a hierarchy of agents, lower two levels...
  - **Sentinel** - resident per machine receives information per agent
  - **Sensors** - wander the network, there are several types of Sensors each looking for a certain type of evidence
- General operation is as follows
  - When a Sensor arrives to a computer, it performs a simple test
  - Test results given to Sentinel, determine if system is *healthy*
  - If results are helpful, then reward Sensor which attracts others

What type of IDS is this? (Note, "failed" is not an answer)

What are the advantages?