Collaborative Intrusion Detection System : A Framework for Accurate and Efficient IDS

Yu-Sung Wu, Bingrui Foo, Yongguo Mei, Saurabh Bagchi
Dependable Computing Systems Lab
School of Electrical and Computer Engineering
Purdue University

http://shay.ecn.purdue.edu/~dcsl

Outline

- Intrusion Detection
- Our Approach
- System Design
- Results
- Conclusions

Intrusion Detection

- Detect deviation of allowable system behaviors or subversion of security policy
- IDSs are based on two alternative choices
  - Anomaly based: Specify the normal behavior (ex: system load goes unexpected high for a long period)
  - Misuse based: Specify the patterns of attacks (ex: detect a string like ‘rm –rf /’)
- Metrics for evaluating IDSs
  - False positives, or False alarms (often seen in anomaly based IDS)
  - False negatives, or Missed alarms (often seen in misuse based IDS)

Challends of current IDS

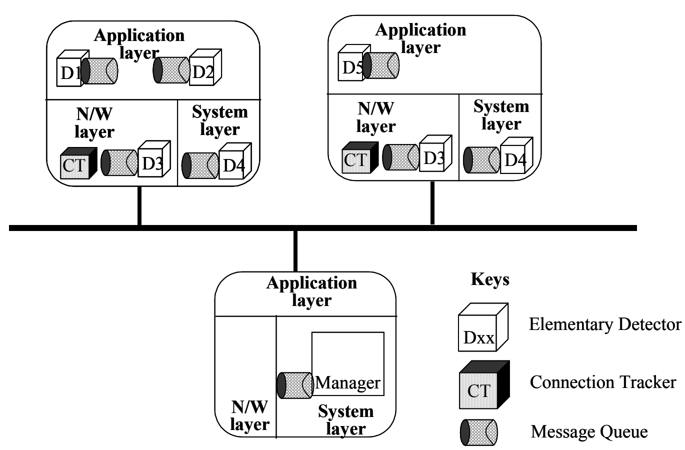
- Traditional IDS only probes at a point of a system
  - Limited view of the whole system
  - The coverage and accuracy of your detection depends solely on the ingenious pattern description or signature definition corresponding to that specific point
  - Loose rules => Better coverage but more False alarms (ex: “/usr/bin/gcc”)
  - Strict rules => Better accuracy but more missing alarms (ex: “/usr/bin/gcc wormX.c”)
- Our approach: **C**ollaborative **I**ntrusion **D**etection **S**ystems (CIDS)
  - Multiple detectors specialized for different parts of system
  - Manager infrastructure for combining alarms from multiple detectors

CIDS Approach - Motivation

- Single IDS (detector) can have false positives (false alarms) or false negatives (missed alarms)
  - It only tells you YES or NO.
  - Usually can't tell you how much the alarm can be trusted.
- Single IDS (detector) is specialized for certain kinds of attacks
  - Limited view of the whole attack => less accuracy
  - An single attack could have multiple symptoms (cascaded attack)
- Combining information from multiple detectors might help detection accuracy
- Future automatic responses mechanism will heavily rely on the quality of the alarms from IDS
- Timing and correlation information might be useful for estimating speed of propagation of attack

CIDS Approach

- Overview



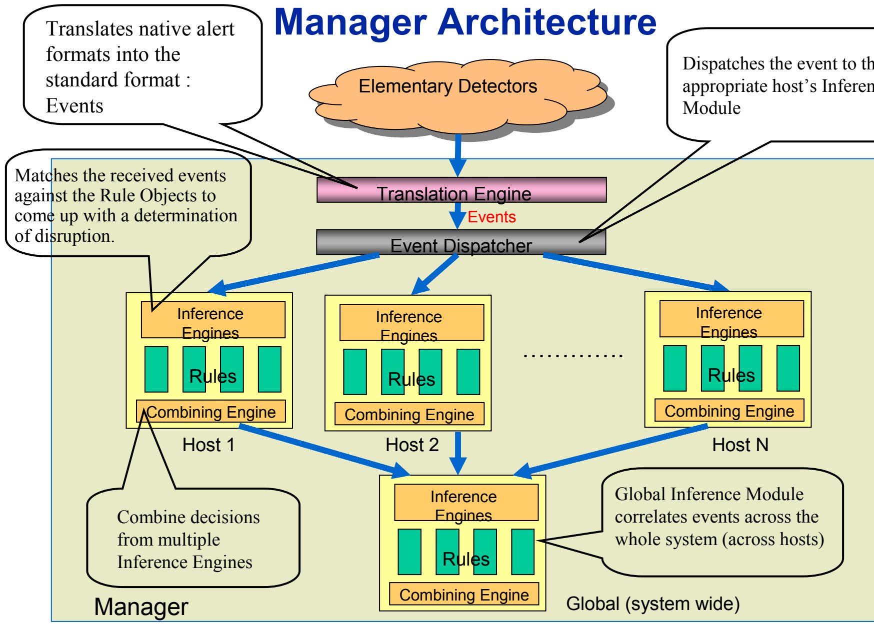
CIDS Components

- Elementary Detectors (EDs): Specialized detectors distributed through the system
  - The EDs may be off-the-shelf and minimal change is required for integration into CIDS (e.g. Snort, Libsafe)
  - Different hosts may have different configurations of EDs
- Message Queue (MQ): Communication layer for multiple CIDS components
  - Secure through a shared secret key and hash digest
- Connection Tracker (CT): Kernel level entity to track which process has active connection on which port (bridge between NIDS and HIDS)
- Manager: Workhorse of CIDS responsible for collating alerts from EDs and generating a combined alert which is expected to be more accurate

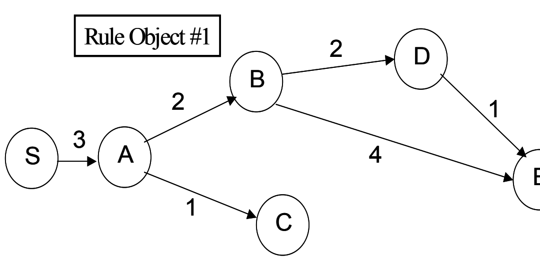
Manager Architecture

- Manager communicates with other entities through MQ and has shared secret key with each ED
- Manager components are
  - Translation engine: Translates native alert formats into CIDS format
  - Event dispatcher: Dispatches the event to the appropriate host's Inference Engine instance
  - Inference Module: An Inference Module contains multiple Inference Engines and a Combining Engine. We have an Inference Module for each host and we also have aglobal Inference Module.
  - Inference Engine: Matches the received events against the Rule Objects to come up with a determination of disruption.
    - A separate instance of the Local Inference Engine for each host
    - A Global Inference Engine for correlating the results from the local engines
    - Rule Objects store the rules, one for each class of disruption
  - Combining Engine: If multiple types of inference engine, this combines the detection decisions from inference engines.

Manager Architecture



Graph-Based Inference Engine



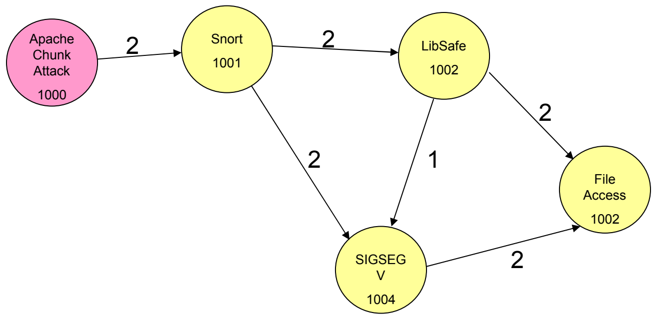
- Rules are represented as graphs
- Nodes are events and Edges represent sequencing of events
- Edge weights represent assurance values indicating likelihood of sequence

- Assurance Value (AV) for an attack given by sum of edge weights
- An event is matched with a rule object if it is *fusionable*, i.e., belongs to the same disruption instance
- Discounted Assurance Value (DAV) for partial matches

$$DAV = AV \times (Partial\ path\ length/Complete\ path\ length)$$

$$Alert\ Probability = DAV/(maximum\ AV)$$

Graph-Based Inference Engine (cont'd)



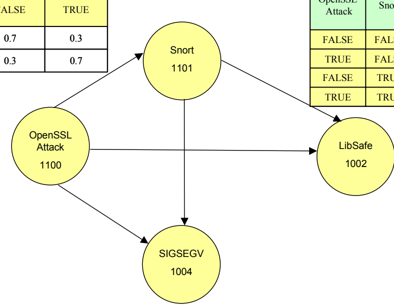
- AC,Snort => 2/7=0.286
- AC,Libsafe =>[(2+2)*2/3]/7=0.38
- AC,Snort,Libsafe=>(2+2)/7=0.57

Bayesian Network Based Inference Engine

- In a Bayesian Network, the nodes represent random variables modeling the events and edges the direct influence of one variable on another
- Three step process for creating rule object
  - Nodes to represent events
  - Edges to represent conditional probability relations among the events
  - Creation of table with conditional probability values

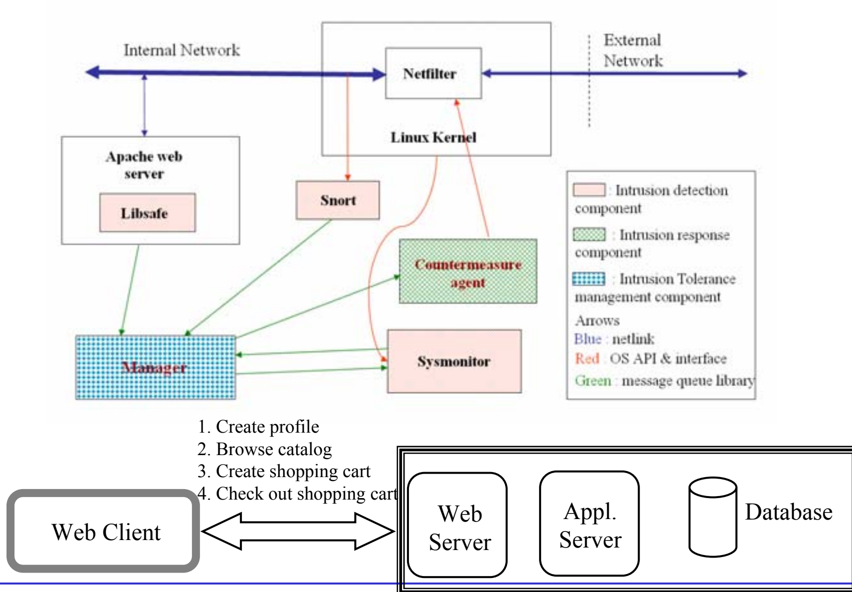
OpenSSL Attack	Snort	
	FALSE	TRUE
FALSE	0.7	0.3
TRUE	0.3	0.7

OpenSSL Attack	Snort	LibSafe	
		FALSE	TRUE
FALSE	FALSE	0.9	0.1
TRUE	FALSE	0.3	0.7
FALSE	TRUE	0.8	0.2
TRUE	TRUE	0.1	0.9



- Bayesian Network toolbox used for solving
- Input is fusionable event stream
- Output is conditional probability of root (the start node – OpenSSL Attack here)

CIDS System: Current Implementation



CIDS Elementary Detectors

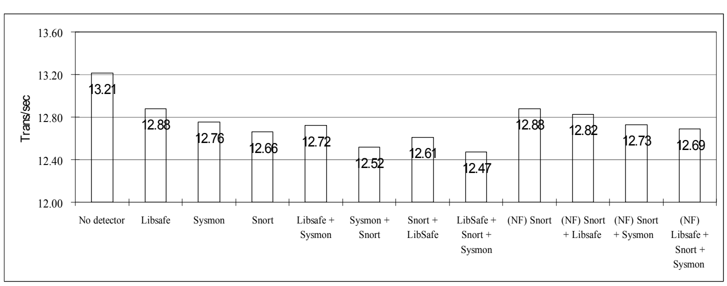
- Application level: *Libsafe*. Middleware to intercept “unsafe” C function calls and prevent stack overflow attacks.
- Network level: *Snort*. Sniffs on incoming network packets and matches against rulebase to perform misuse based detection.
- Kernel level: *Sysmon*. Home-grown new detector.
  - Intercepts system calls for file accesses and executions.
  - Takes a set of rules for disallowed accesses or executions
    - Can be specified using wildcards or directory tree
  - Intercepts signals of interest that can flag illegal operations.
    - SIG_SEGV to indicate segmentation violation that may be caused by buffer overflow

Simulated Attacks

- Three classes of attacks, multiple types within each class, and multiple variants within each type
  - Buffer overflow: Can be used to overwrite parts of stack and write and execute malicious code
    - Apache chunk attack
    - Open SSL attack
  - Flooding: Overwhelm the network with redundant or malicious packets causing a denial of service
    - Ping flood
    - Smurf
  - Script based: Exploit poorly written scripts which do not do input validation to execute arbitrary commands
    - Used unchecked Perl *open()* and *system()* calls

Results: Performance – Without Attacks

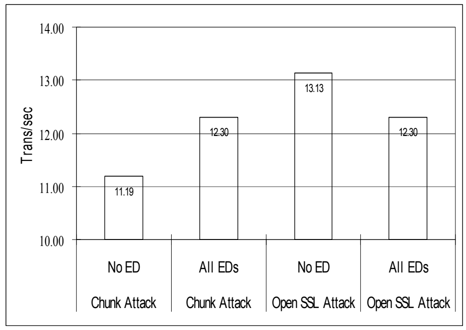
- Measured without and with attacks
- 30 web clients running concurrently
- (Transactions/second) of workload transaction measured
- When multiple EDs present, manager with both Inference Engines is deployed



No Intrusion

- Degradation overall: 3.95% with Snort rules modified, 5.60% without

Results: Performance – With Attacks



- OpenSSL Attack performance degradation is 6.33%
- Chunk Attack performance improves!!!
  - Having Libsafe prevents core dumping
- Highest performance degradation due to Matlab Bayesian Network toolbox

Results: Accuracy of Detection

	Snort	Libsafe	Sysmon (Signal)	Sysmon (File)	CIDS (Alert Prob. > 0.5 ?)
No attacks	Yes (1807,1933)	No	No	No	No attack
Open SSL	Yes (1881,1887)	No	Yes	*R1*	Yes
Open SSL variant	No	No	Yes	*R1*	Yes
Apache Chunk	Yes (1807, 1808, 1809)	Yes	Yes	*R1*	Yes
Smurf 1000	Yes (499)	No	No	No	Yes
Smurf 500	No	No	No	No	No
Ping Flooding	Yes (523, 1322)	No	No	No	Yes
Script	No	No	No	Yes	Yes

- Yes: Detected. Figures in parentheses are the rule numbers within Snort. Sysmon(File) is the file access detection part, Sysmon(Signal) is the illegal signal detection part; R1: The attack was not successful in creating a file.

Conclusion

- CIDS can accommodate best-of-breed detection techniques (existing off-the-shelf detectors can be easily integrated) and provides management and correlation facility
- Two algorithms for correlating alerts
  - Graph-based
  - Bayesian network based
- Both false alarms and missing alarms are reduced
- Output of the two correlation algorithms are probability values telling you how possible that attack has happened.
- Performance degradation after using CIDS is around 3.95% under normal operations (without attacks) and 6.33% when being operated under OpenSSL attacks