Collaborative Intrusion Detection System & Application to VoIP

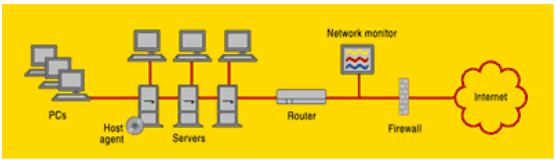
Saurabh Bagchi
Dependable Computing Systems Lab
School of Electrical and Computer Engineering
Purdue University

Joint work with: Vinita Apte, Bingrui Foo, Yu-Sung Wu, Eugene H. Spafford

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

Intrusion Detection: What?

- Intrusion Detection System (IDS) detects deviation of allowable system behaviors or subversion of security policy
- IDS market is divided into two groups
  - Host-based
  - Network-based



Intrusion Detection: A little bit of past and present

- Past:
  - IDSs came about in 1970s and looked very different
  - Mainframe computers were pricey beasts and people had to pay good money for cycles on them
  - IDS were for checking if anyone was stealing cycles
- Present:
  - The US Treasury Department estimates cybercrime proceeds in 2004 were $105 billion, greater than those of illegal drug sales.
  - 2005 FBI/CSI Computer Crime and Security Survey found nearly nine out of 10 U.S. businesses suffered from a computer virus, spyware or other online attack in 2004 or 2005
  - The process of downloading and installing the latest Microsoft patches which may be as small as 70 megabytes (MB) or as large as 260 MB, takes longer then the time it takes for an unpatched computer to be compromised
  - IDS and Firewalls are the first line of defense

Intrusion Detection: Types

- IDSs are based on two alternative choices
  - Anomaly based: Specify the normal behavior of users or machines
    - Example: System CPU utilization between midnight and 5 am is no more than 50%
  - Misuse based: Specify the signatures of attacks
    - Example: Detect a string like 'rm –rf /'
- Metrics for evaluating IDSs
  - False positives, or False alarms
  - False negatives, or Missed alarms

False alarms are often seen in anomaly based IDS
Missed alarms are often seen in misuse based IDS

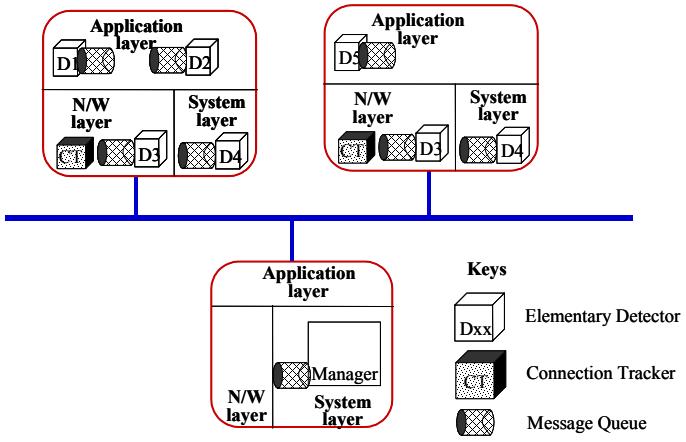
Challenges of current IDS

- 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

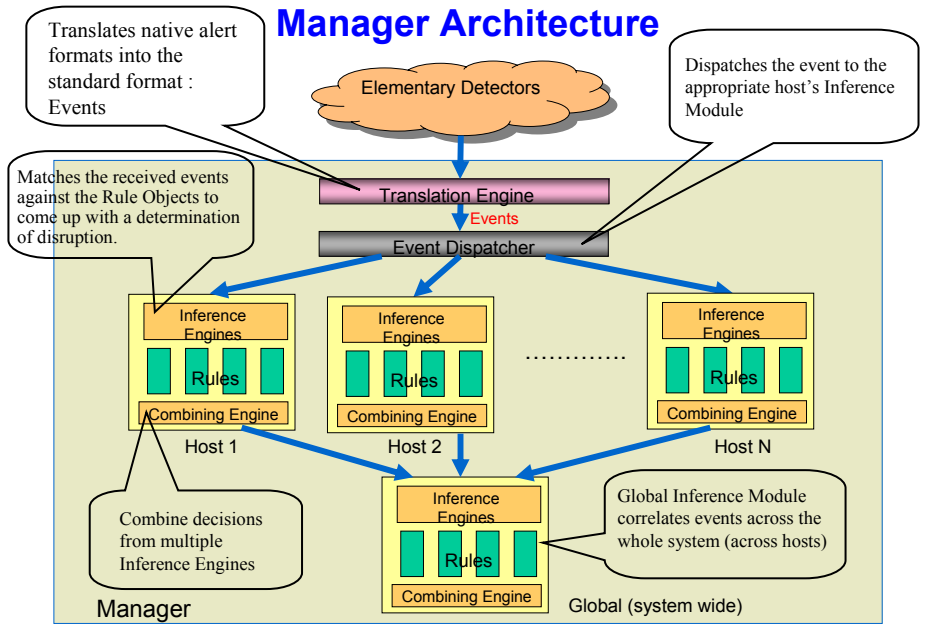
CIDS Architecture



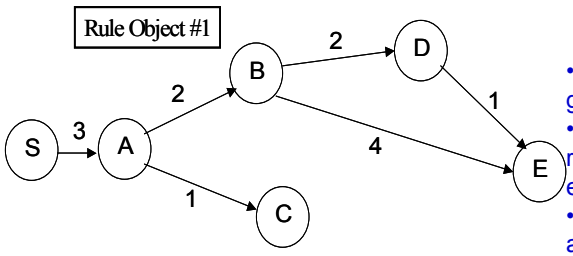
CIDS Components

- 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: Responsible for collating alerts from EDs and generating a combined alert expected to be more accurate

Manager Architecture



Graph-Based Inference Engine



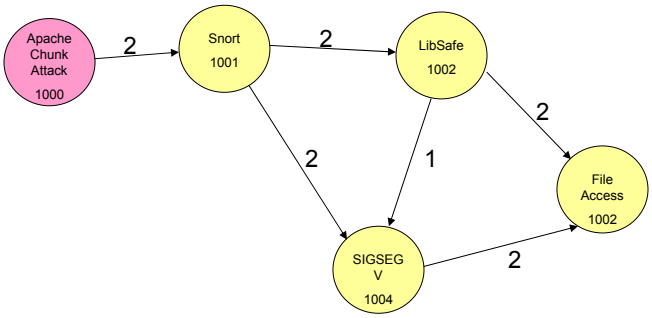
- Rules are represented as graphs
- Nodes are events and Edges represent sequencing of events
- Edge weights represent assurance values indicating likelihood of sequence

- 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 × (Partial path length/Complete path length)

Alert Probability = DAV/(maximum AV)

Graph-Based Inference Engine (cont'd)



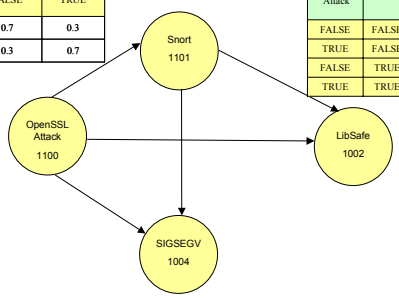
- AC,Snort => 2/7=0.286
- AC,Libsafe =>[(2+2)*2/3]/7=0.38
- AC,Snort,Libsafe=>(2+2)/7=0.57

PURDUE UNIVERSITY

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

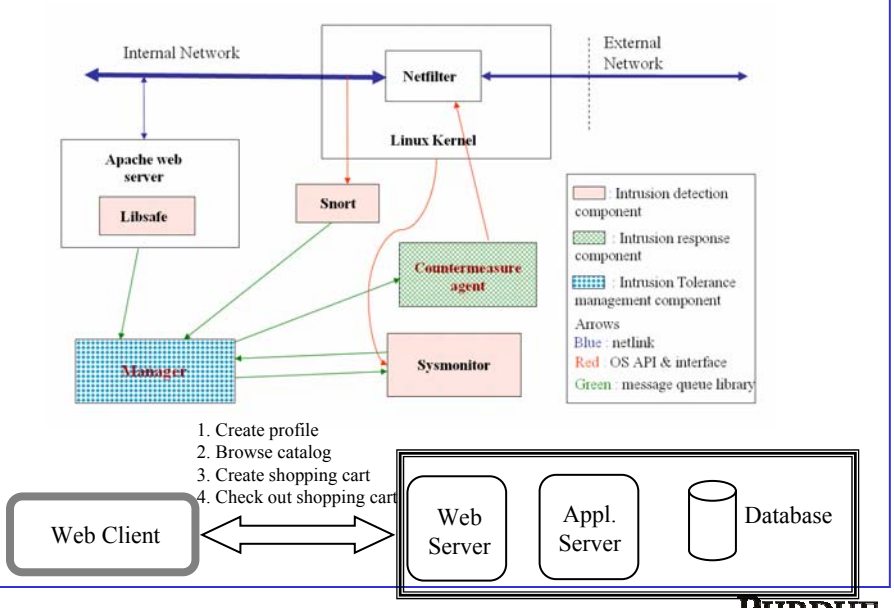


OpenSSL Attack	Snort	LibSafe	
		FALSE	TRUE
FALSE	FALSE	0.9	0.1
TRUE	FALSE	0.3	0.7
FALSE	TRUE	0.8	0.2
TRUE	TRUE	0.1	0.9

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

PURDUE UNIVERSITY

CIDS System: Current Implementation



CIDS Elementary Detectors

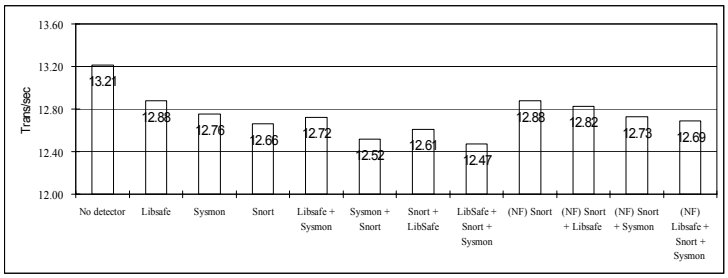
- Application level: *Libsafe*. Middleware to intercept “unsafe” C function calls and prevent stack overflow attacks.
- Network level: *Snort*. Sniffs on incoming network packets and matches against rulebase to perform misuse based detection.
- Kernel level: *Sysmon*. Home-grown new detector.
  - Intercepts 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

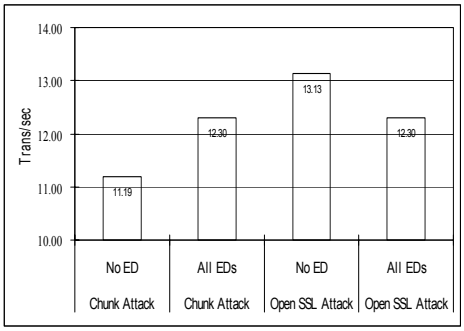
- Measured without and with attacks
- 30 web clients running concurrently
- (Transactions/second) of workload transaction measured
- When multiple EDs present, manager with both Inference Engines is deployed



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.

CIDS Summary

- 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

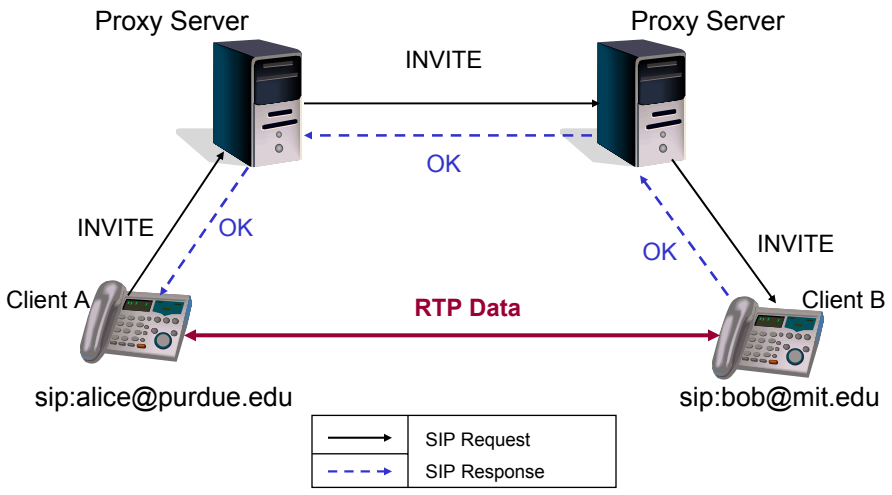
Application of CIDS to a Voice over IP (VoIP) Application

VoIP Basics

- First launched in Feb 1995 – Vocaltec Inc.
- Allows transport of voice traffic over IP networks
  - Cost savings due to convergence of networks
  - Do away with internal EPABX systems
- Protocols
  - RTP: Data transport
  - SIP, H.323: Signaling
- Some VoIP predictions:
  - 12 million VoIP users by end of 2007 [Forrester Research]
  - 100 million VoIP users by the end of 2011 [ON World Research]

"By 1985, machines [computers] will be capable of doing any work Man can do." – Herbert A. Simon, of Carnegie Mellon University, one of the founders of the field of artificial intelligence (1965)

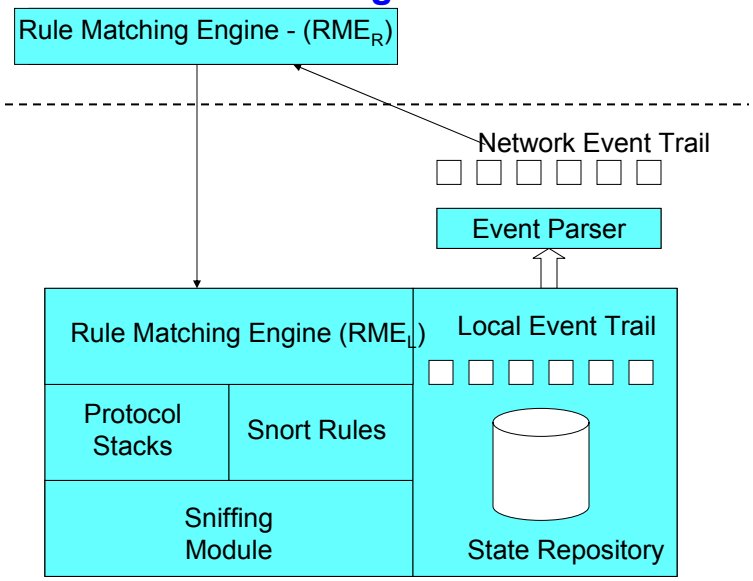
VoIP Call Mechanism



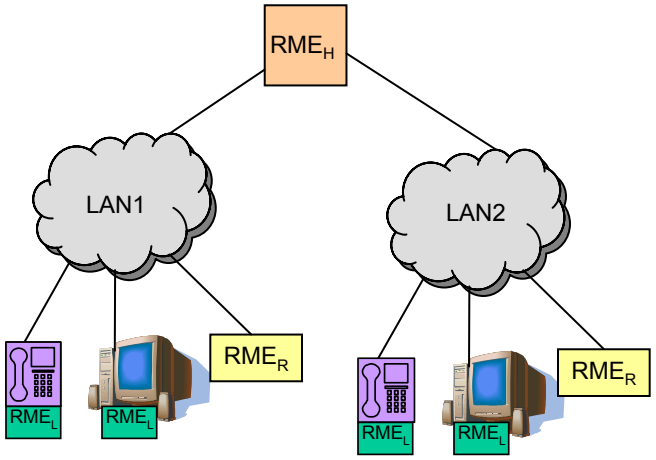
Distributed IDS for VoIP Application

- Distributed Intrusion Detection System for VoIP networks - **SPACEDIVE**
- Why build an IDS just for VoIP?
  - Soft real-time requirements
  - Attack can take place across a session
  - Attack across protocols
- Why correlation based detection?
  - Multiple components in a VoIP system
  - One attack may span many components
  - One detector cannot have a complete view of the different stages of attack
- Approach:
  - Local detector using fast network pattern matching
  - Remote detector to correlate alerts
  - Stateful and cross-protocol detection

SPACEDIVE Design – Local Level



SPACEDIVE Design – Remote Level



Low Level Rule Language

- Snort rules
  - Limited capability for remembering state
  - Events span multiple packets
  - Events span multiple protocols
- New constructs
  - var – set the value of a state variable
  - event – trigger a local event
  - net_event – trigger a network-level event
  - seqwin (protocol specific - RTP) – specify maximum tolerance for out-of-order packets
  - Connectors: AND/OR/NOT/BEFORE/AFTER

Sample Rules

```
1. alert udp Client_IP any -> Server_IP 5060
   (content:"INVITE"; var invite;)

2. alert udp Server_IP any -> Client_IP any
   (content:"sip:OK"; var ok;)

3. event(ok AFTER invite;)  # trigger local event

4. alert rtp my_IP 6000 (seqwin:50; var:dos);

5. event(dos;)
6. net_event(dos;)          # network event –
                      # notification
                      # sent to RME_R
```


High Level Rule Language

- Specified at the RME_Rs
- General format

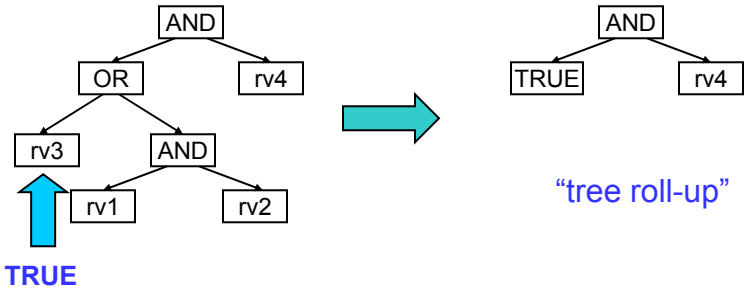
$$R = ((where_i : what_i)conn_i)response \quad (i = 1, \ldots, N)$$

- Sample Rule

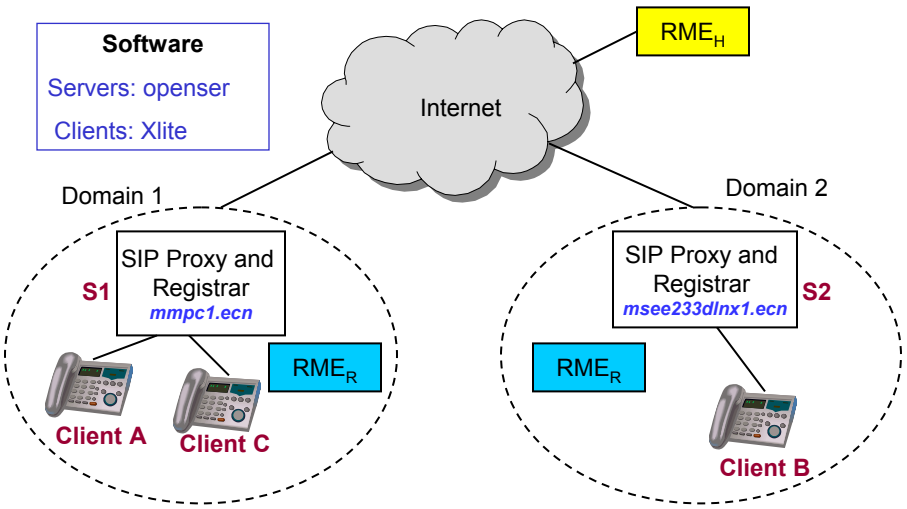
```
(C1:RTP_DATA) AFTER (C2:SESSION_TERM) alert
```

Efficient Matching

- *event ((rv3 OR (rv1 AND rv2)) AND rv4)*



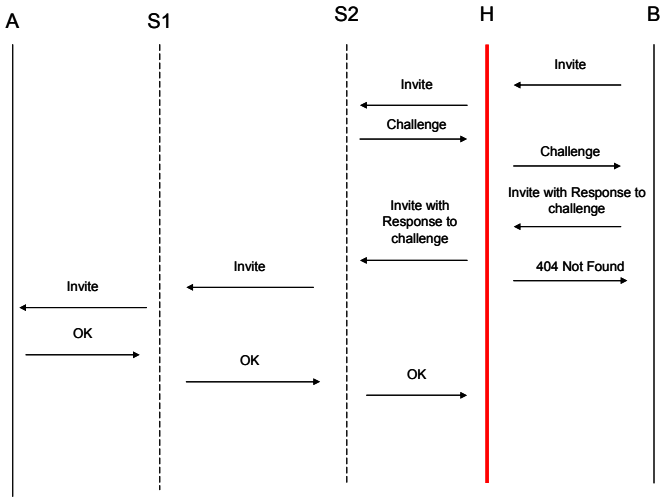
Experimental Testbed



Attack Scenarios

- Call Hijacking
- **Man in the Middle Attack**
- BYE Attack
- Compromised SIP Proxy
- Billing Fraud
- **Denial of Service (DoS) Attack**

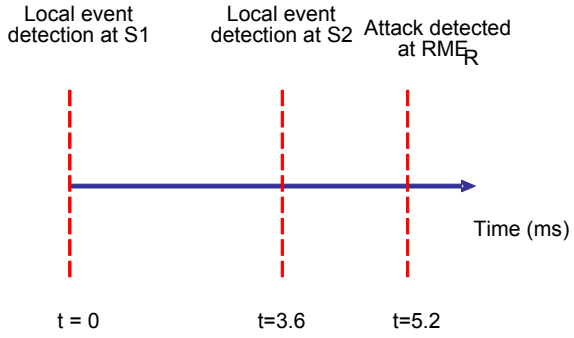
Man in the Middle Attack



Rules for Detection

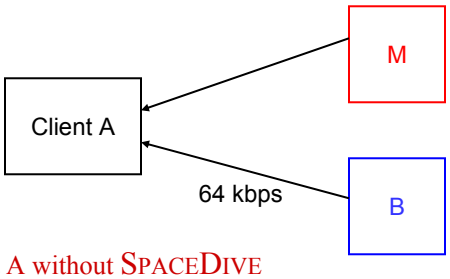
Component	Rule
S1	`alert udp A_IP any -> S1_IP any (var:rv1; content:OK;)` `net_event (rv1;)`
S2	`alert udp S1_IP any -> S2_IP any (var:rv2; content:OK;)` `net_event (rv2;)`
B	`alert udp S2_IP any -> B_IP any (var:rv3; content:OK;)` `net_event (rv3;)`
RME_H	`(S1:SIP_OK) AND (S2:SIP_OK) AND (NOT(B:SIP_OK)) alert`

Timeline for correlated detection



- Detection latency depends upon timeout interval
- Timeout interval is a configurable parameter dependent on
  - Network delay
  - Processing capacity

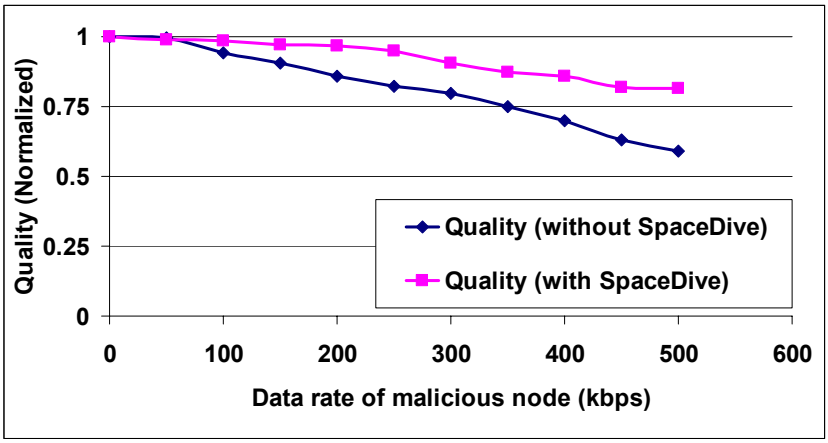
Resilience to DoS



- Case 1: Client A without SPACEDIVE
- Case 2: Client A with SPACEDIVE

```
drop rtp A_IP 6000 (seqwin:50; var:dos);
```


Resilience to DoS: Results



SpaceDive Summary

- We designed and built an IDS for VoIP systems
- We showed that it could correlate across protocols and components
- It was scalable and could perform fast matching of network packets

I. Vinita Apte, Yu-Sung Wu, Saurabh Bagchi, Sachin Garg, and Navjot Singh, "SPACEDIVE: A Distributed Intrusion Detection System for Voice-over-IP Environments," Appeared at the IEEE International Conference on Dependable Systems and Networks (DSN), June 25-28, 2006, Philadelphia, USA.

II. Saurabh Bagchi, Yu-Sung Wu, Sachin Garg, Navjot Singh, and Tim Tsai, "SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments," In Proceedings of the IEEE Dependable Systems and Networks Conference (DSN), pp. 401-410, June 28-July 1, 2004, Florence, Italy.

III. Yu-Sung Wu, Bingrui Foo, Yongguo Mei, and Saurabh Bagchi, "Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS," In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), pp. 234-244, December 2003.

What's Next?

- What do we do once we know with reasonable certainty that an intrusion is in progress
  - Send alert to the poor system admin at midnight asking her to hurry into the lab and do something
  - Place the order for a new cluster of machines and tell my lab mates that their data and code is gone – heck! I detected it
  - Provide automatic containment of the current attack and response to make the system less vulnerable to future attacks
- We did it for client-server VoIP applications. What about peer-to-peer VoIP applications? Think Skype.
  - Looser trust relations
  - Arbitrary joins and leaves of entities

Come Join Us at DCSL

- We are looking for 2 PhD students who can make it happen
- Desirables
  - Good programming skills in C and C++
  - Good network programming skills
  - Not afraid to do experiments with real systems, real code, and real attacks
- If interested, send mail to: dcsl@ecn.purdue.edu explaining why you think there is a fit and attaching a plain text resume

Backup Slides

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 a global 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.

Performance of Rule Matching

- Rule matching overhead
- Defined 4 categories of rules:
  - Type 0: Snort rule matched in Snort
  - Type 1: Snort rule matched in SPACEDIVE
  - Type 2: Use **`var`** construct to set the value of a variable.
  - Type 3: Create local event in the event trail

Performance of Rule-Matching: Results