ADEPTS : Adaptive Intrusion Response using Attack Graphs in an e-Commerce Environment

Bingrui Foo, Yu-Sung Wu, Yu-Chun Mao, Saurabh Bagchi, Eugene Spafford

Purdue University

Dependable Computing Systems Lab

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

Outline

- Intrusion Response for Distributed System
- ADEPTS
- System Design
- Experiments & Results
- Conclusions

Intrusion Response for Distributed System

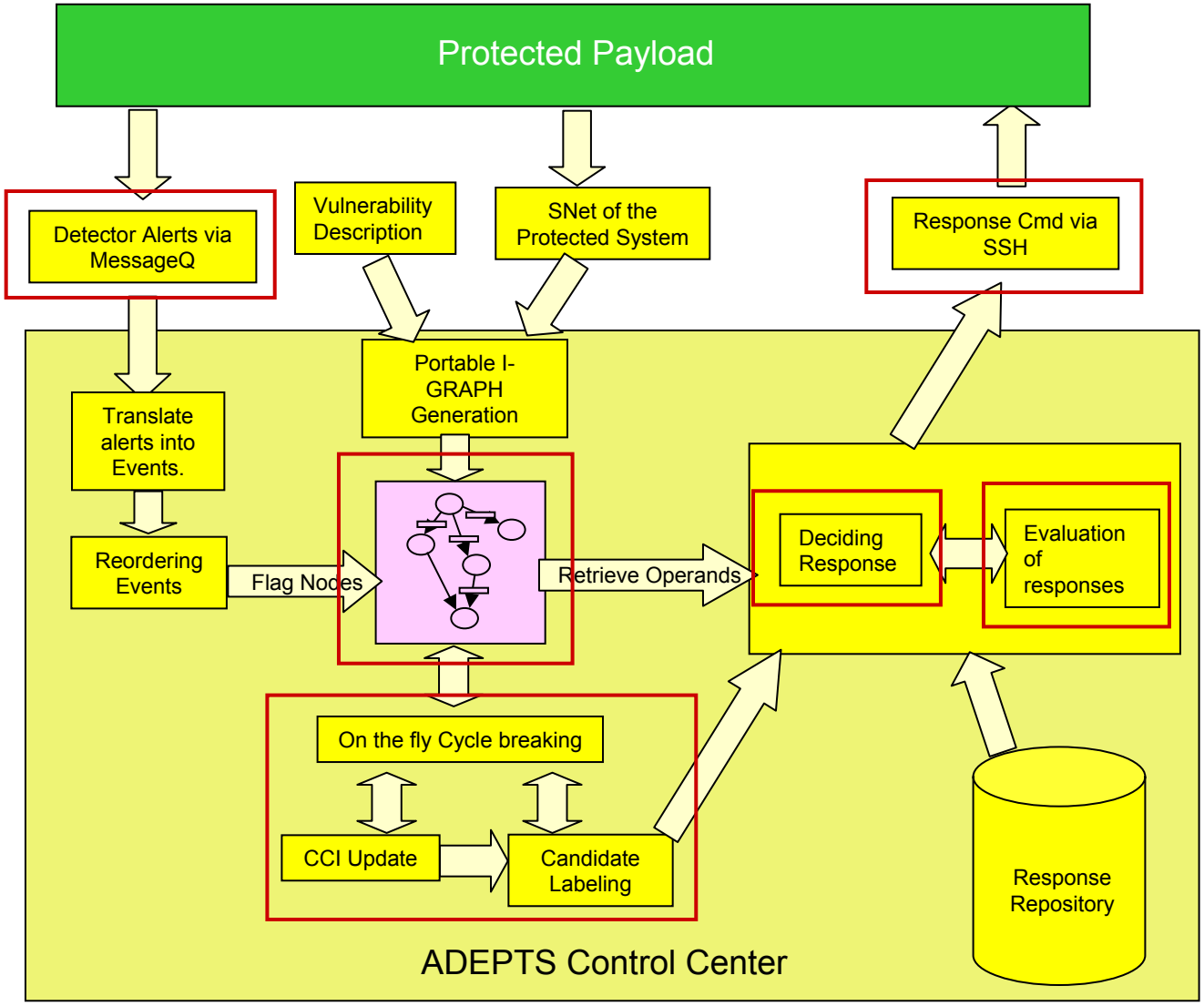
- eCommerce System
  - Interconnected entities and services (customers, bank, warehouse, database, web applications, and etc.)
  - A favorable target of cyber attacks (intrusions)
  - Denial-of-service, Vandalizing, Stealing information, Illegal transactions
- Intrusion Detection for distributed system (eCommerce System)
  - Comparably abundant existing work
  - Snort, Tripwire, Buffer overflow detector, application-level detectors
- Intrusion Response for distributed system (eCommerce System)
  - Typically requires the administrator to check the detection log files, identify the compromised region, and enforce the containment
    - Not automatic. Long reaction time.
  - Local Response : some IDS (e.g. Snort, Libsafe) provides basic response capability. Anti-Virus software disabling access to infected files
    - Function Locally. Fight the fire at the fire station.

ADEPTS: High Level Approach

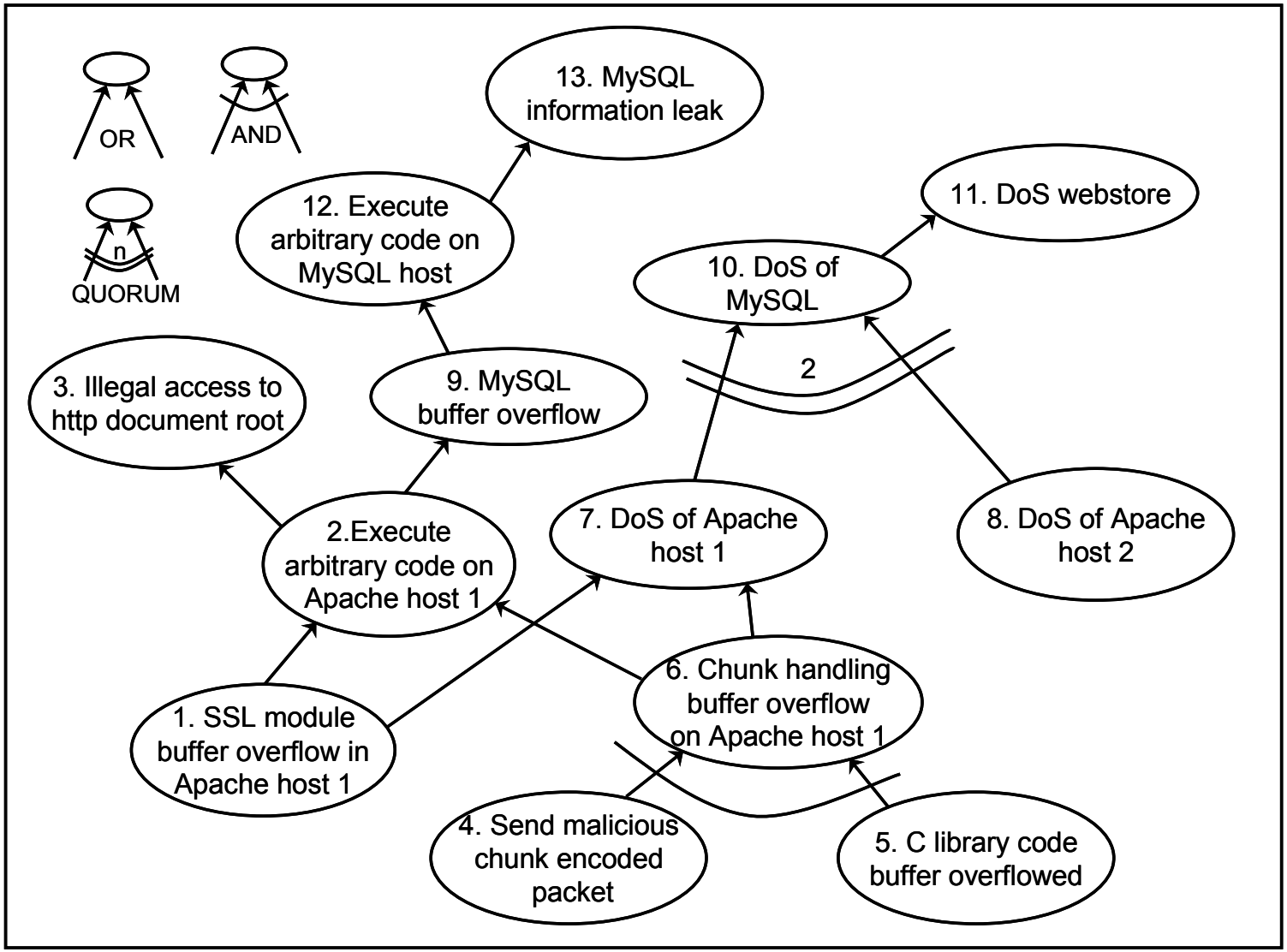
- A general framework for automatic & systematic intrusion response
- Incorporate detectors and response agents, which are deployed across the protected payload
- Use Attack Graph (I-GRAPH) to model the causal relations among intrusion goals
- Upon an intrusion, ADEPTS firstly estimates the compromised region (achieved intrusion goals)
- Systematically deploys a set of preferred responses to contain the affected region (prevent more intrusion goals from being achieved)
- Feedback mechanism for evaluating effectiveness of deployed responses and adjusting response preference

Process Flow & Architecture View of ADEPTS

1. Detection framework flags alerts
2. I-GRAPH parameters updated
3. Determine locations to take responses
4. Available responses determined based on attack parameters and I-GRAPH
5. Best responses chosen and deployed
6. Evaluation of deployed responses



I-GRAPH

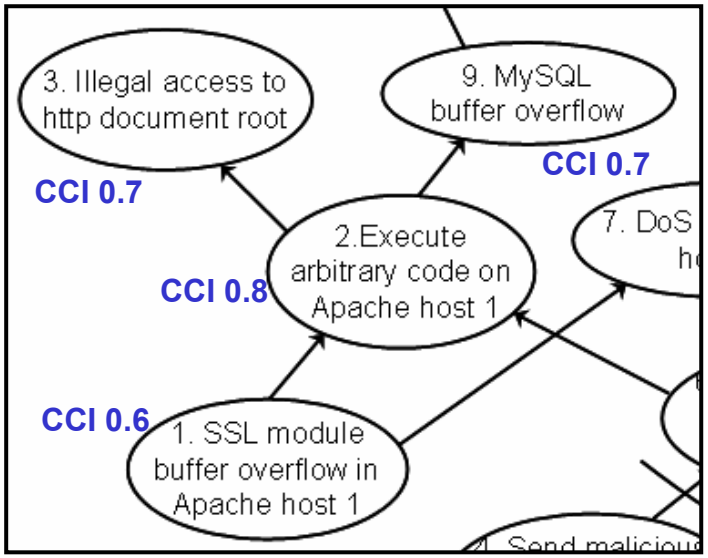

OR
AND
n
QUORUM
13. MySQL information leak
12. Execute arbitrary code on MySQL host
11. DoS webstore
10. DoS of MySQL
2
3. Illegal access to http document root
9. MySQL buffer overflow
2.Execute arbitrary code on Apache host 1
7. DoS of Apache host 1
8. DoS of Apache host 2
6. Chunk handling buffer overflow on Apache host 1
1. SSL module buffer overflow in Apache host 1
4. Send malicious chunk encoded packet
5. C library code buffer overflowed

Determining how likely a node is compromised

- The *Compromised Confidence Index* (CCI) of a node in the I-GRAPH is the measure of the likelihood that an attacker has reached that node

$$CCI = \begin{cases} \text{alert confidence} & \text{, nodes with no children} \\ f(CCI_i) & \text{, nodes with no detectors} \\ \text{ave}(f(CCI_i), \text{alert confidence}) & \text{, otherwise} \end{cases}$$

$$f = \begin{cases} \max(CCI_i) & \text{, OR edges} \\ \min(CCI_i) & \text{, AND edges} \\ \begin{cases} \text{Mean}(CCI_i \mid CCI_i > \tau_N) & \text{, quorum met} \\ 0 & \text{, quorum not met} \end{cases} \end{cases}$$



where CCI_i corresponds to the CCI of the i^{th} child and τN is a per node threshold

Picking Responses

- After determining a set of likely-compromised nodes, the response decision module will search the response repository for the responses whose opcodes and operands are applicable to the intrusions on these compromised nodes
- Each response command has an associated RI (response index) value, with a larger RI value indicating a more preferred response

RI=EI-DI

EI: Effectiveness Index DI: Disruptivety Index

iptables -A INPUT -s ip_address -j DROP

Snort 1807 WEB-MISC Chunked-Encoding transfer attempt from source IP a.b.c.d

iptables -A INPUT -s a.b.c.d -j DROP

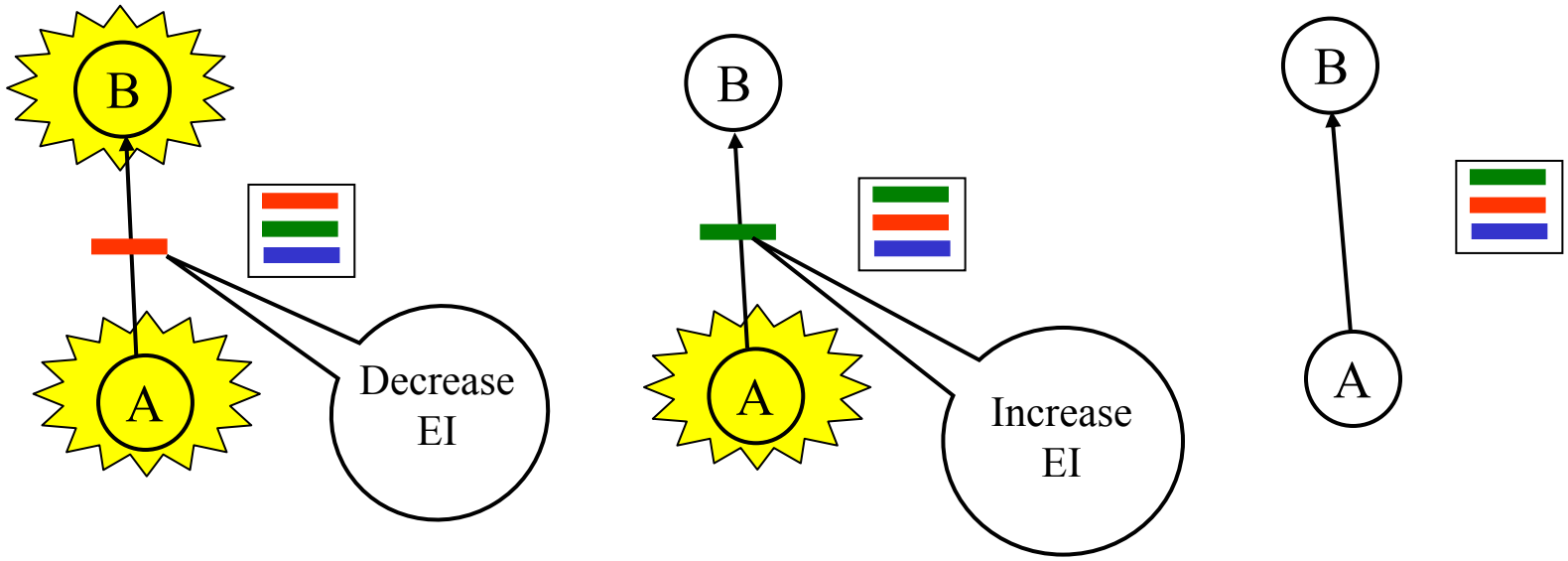
The response command with the highest RI value

Response Repository

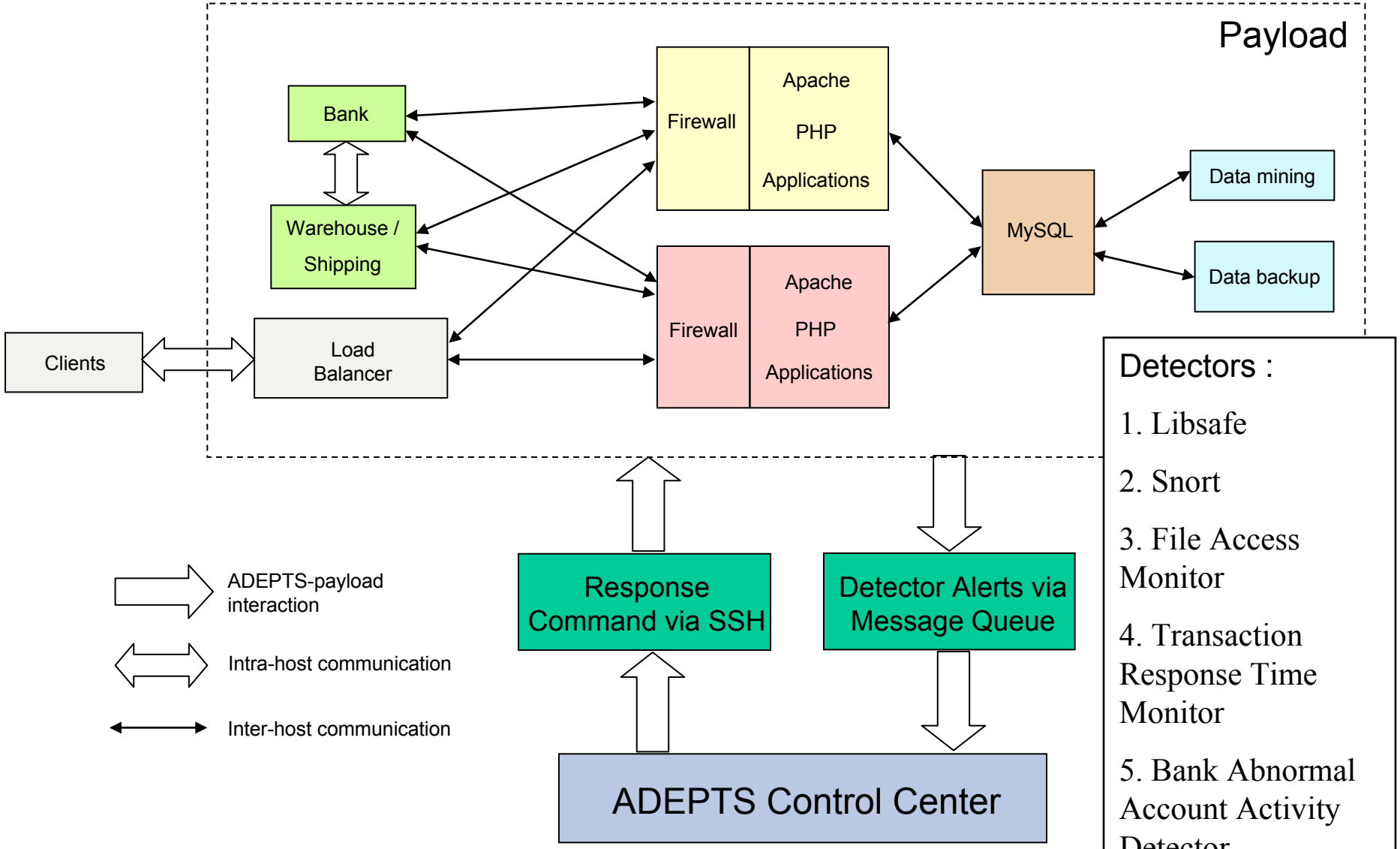
Command type	Commands		Explanation
	Opcode	Operands (opr1 And opr2 And …)	
General	KILL_PROCESS	PROCESS_ID	Kill process
	SHUT_DOWN	SERVICE_NAME / HOST	Shut down/restart a service/host
	RESTART/REBOOT	SERVICE_NAME / HOST	
	DISABLE	USER_ACCOUNT	Freeze a user account
File	DENY_FILE_ACESS	FILE_NAME	Disable read, write, and execute access to a file, valid for the super user.
	DISABLE_READ	FILE_NAME	Disable read/write access to a file, valid for the super user.
	DISABLE _WRITE	FILE_NAME	
Network	BLOCK_INPUT	REMOTE_IP	Blocking incoming packets associated with the command operands.
		REMOTE_IP LOCAL_PORT	
		REMOTE_PORT PROTOCOL	
		LOCAL_PORT	
	BLOCK_OUPUT	REMOTE_IP	Blocking outgoing packets associated with the command operands.
		REMOTE_IP LOCAL_PORT	
		REMOTE_PORT PROTOCOL	
		LOCAL_PORT	
	BLOCK_FORWARD	SOURCE_IP DESTINATION_IP	Blocking forwarding packets associated with command operands.
DoS	LIMIT_RATE	SYN	Limiting rates of a type of packets
		ICMP_ECHO	
		ICMP_HOST_UNREACHABLE	
		SYN_ACK	
		UDP_PACKET	

Feedback Mechanism

- After responses are deployed, we can judge whether the deployed responses are effective or not by checking if intrusions are still propagating (higher level nodes in the I-GRAPH keep getting flagged). ADEPTS will then adjust the EI values of the responses so that effective responses will be more preferable in a future run.



Testbed



Detectors :

1. Libsafe
2. Snort
3. File Access Monitor
4. Transaction Response Time Monitor
5. Bank Abnormal Account Activity Detector

Experiments & Results

- Attack Scenarios

Steps	Scenario 0	Scenario 1	Scenario 8
0	Exploit Apache mod_ssl buffer overflow.	Use php_mime_split (CVE-2002-0081) buffer overflow to insert malicious code into Apache.	ModSSL Buffer overflow in Apache.
1	Insert malicious code.	'ls' to list webstore document root and identify the script code informing the warehouse to do shipments.	A shell is created with Apache privilege.
2	Ip/port scanning to find vulnerable SQL server.	Send shipping request to warehouse and craft the request form so that a warehouse side buffer overrrun bug fills the form with a victim's credit card number.	Issue crontab command to exploit a vunerability in cron daemon for creating a root privilage shell.
3	Buffer overflow MYSQL to create a shell (/bin/sh).	Unauthorized orders are made.	Root privilege shell created out of the vulnerable cron daemon.
4	Use malicious shell to steal information stored in MySQL.		Corrupt the data stored in web server document root.

Experiments and Results

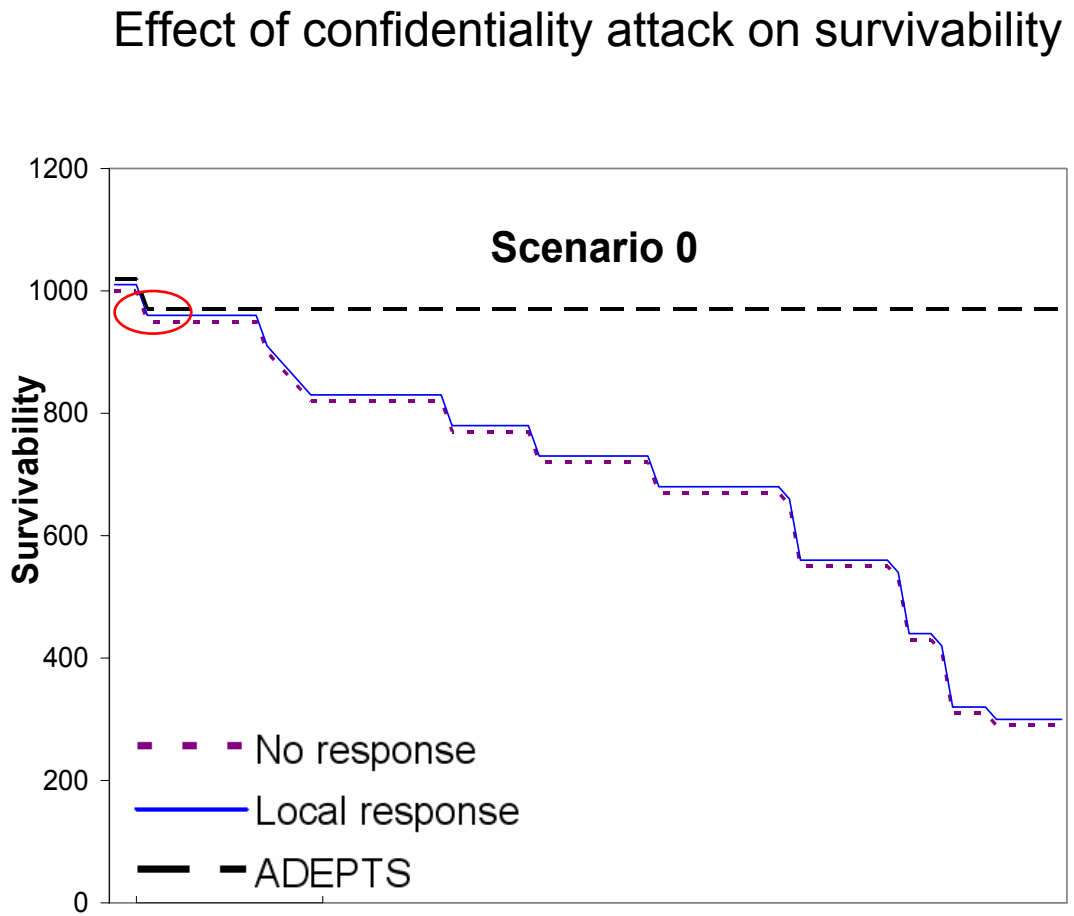
- Survivability Metric
  - We define a set of transactions and a set of security goals. We use the survivability metric in the experiment to demonstrate the benefit of adopting ADEPTS in terms of maintaining the survivability of the underlying e-Commerce system.

$$\text{Survivability} = 1000 - \sum \text{unavailable transactions} - \sum \text{failed security goals}$$

Name	Weight
Browse webstore	10
Add merchandise to shopping cart	10
Place order	10
Charge credit card	5
Admin work	10

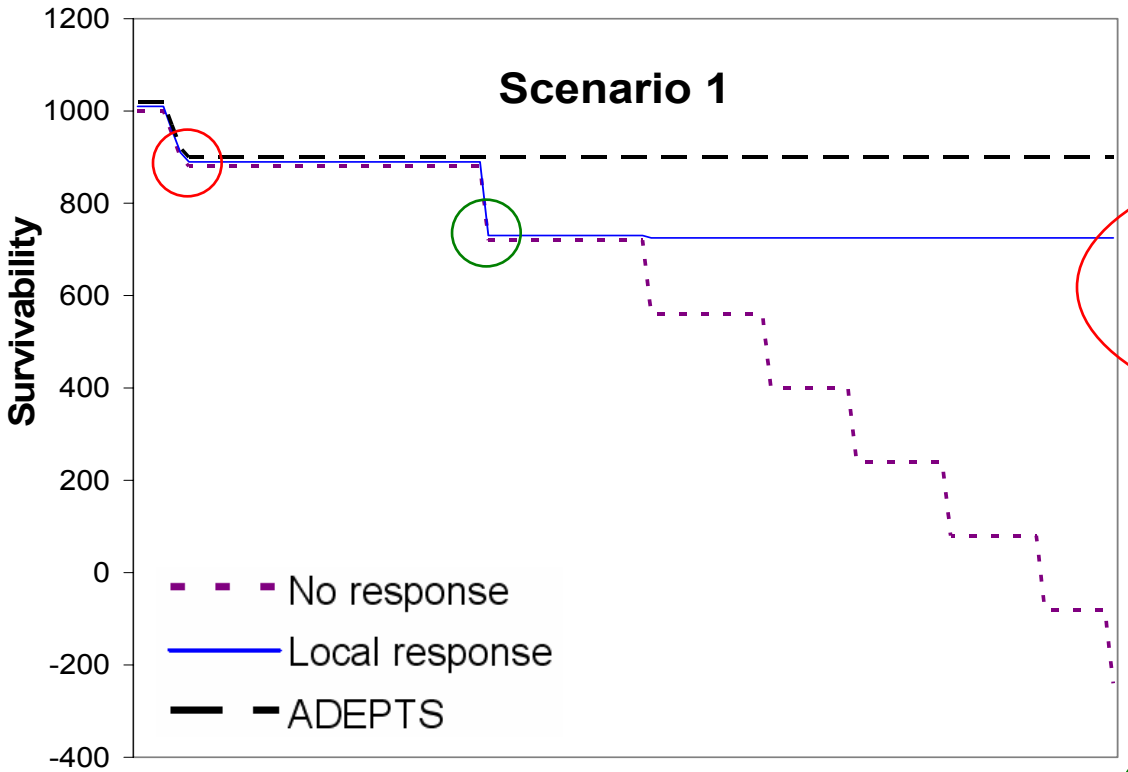
Illegal read of file	20
Illegal write to file	30
Illegal process being run	50
Corruption of MySQL database	70
Confidentiality leak of customer information stored in MySQL database	100
Unauthorized orders created or shipped	80
Unauthorized credit card charges	80
Cracked administrator password	90

Experiment #1 Survivability Improvement



Steps	Scenario 0
0	Exploit Apache mod_ssl buffer overflow.
1	Insert malicious code.
2	Ip/port scanning to find vulnerable SQL server.
3	Buffer overflow MYSQL to create a shell (/bin/sh).
4	Use malicious shell to steal information stored in MySQL.

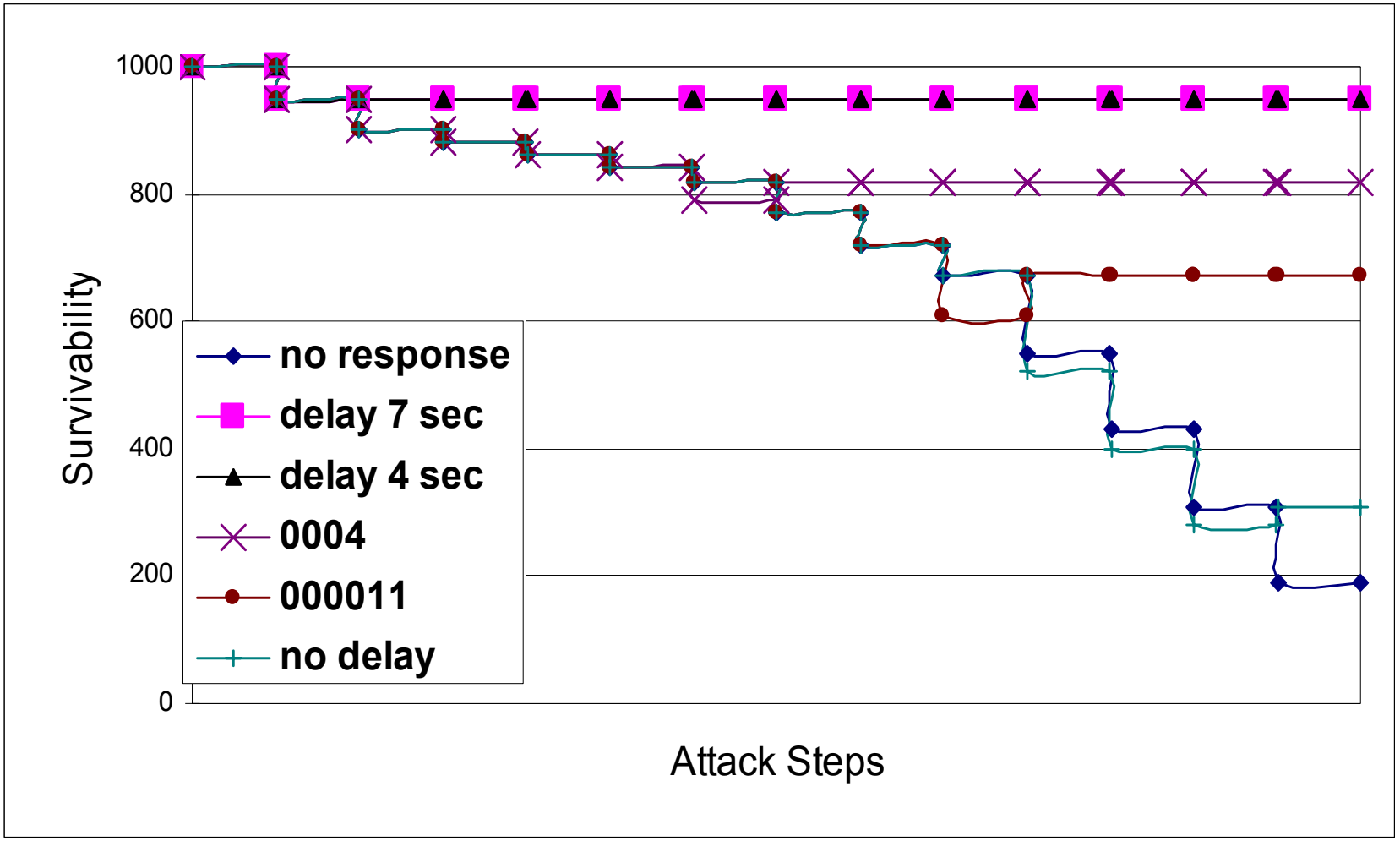
Effect of illegal transactions on survivability



Scenario 1
Use php_mime_split (CVE-2002-0081) buffer overflow to insert malicious code into Apache.
'ls' to list webstore document root and identify the script code informing the warehouse to do shipments.
Send shipping request to warehouse and craft the request form so that a warehouse side buffer overrrun bug fills the form with a victim's credit card number.
Unauthorized orders are made.

Experiment #2 Impact from speed of propagation

Impact from speed of propagation on survivability with Scenario 0.



Experiment #3 Feedback Mechanism

- This experiment is composed of multiple runs of scenario 8 (we show runs 1,3, and 6)

Run 1	
Step	Attack impacts
0	Apache MOD_SSL Buffer Overflow
1	Apache Privilege Shell Created
	X R1 X R2
2	Executing crontab command
EI [R1] 1.1 → 1.072497 EI [R2] 1.1 → 1.058745	X R3 X R4
3	Put malicious data into Apache user's crontab
EI [R2] 1.058745 → 1.024404 EI [R4] 1.1 → 1.064321 EI [R3] 1.1 → 1.076214 EI [R1] 1.072497 → 1.049305	X R5
4	Root privilege Shell created out of cron daemon
5	Via the root shell, the attacker tampers with files under http document directory.
EI [R2] 1.024404 → 0.99328 EI [R4] 1.064321 → 1.031984 EI [R5] 1.1 → 1.077720 EI [R3] 1.076214 → 1.054415 EI [R1] 1.049305 → 1.028052	**O R6** X R7 X R8
Attack Stopped EI [R8] 1.1 → 1.21 EI [R7] 1.1 → 1.21 EI [R6] from 1.1 → 1.21	

Run 3	
Step	Attack impacts
0	Apache MOD_SSL Buffer Overflow
1	Apache Privilege Shell Created
	X R1 X R7
2	Executing crontab command
EI [R7] 1.21 → 1.164620 EI [R1] 0.96081 → 0.936787	**O R6** X R4
3	Put malicious data into Apache user's crontab
EI [R7] 1.16462 → 1.126844 EI [R4] 1.1 → 1.064321 EI [R1] 0.936787 → 0.916530 EI [R6] 1.331 → 1.302219	X R3
Attack Stopped EI [R3] up from 1.01072 → 1.111792	

Run 6	
0	Apache MOD_SSL Buffer overflow
1	Apache privilege shell created
	O R6 O R7
Attack Stopped EI [R7] 1.1 → 1.21 EI [R6] 1.401466 → 1.541612	

R1	Block port 443 from attacker's src IP
R2	Kill the Apache privilege shell
R3	Block attacker's source IP
R4	Kill crontab process
R5	Restart Aapache
R6	Reboot Apache's host machine
R7	Deny access to crontab command and kill process if still running
R8	Set Apache's HTTP document directory to READONLY

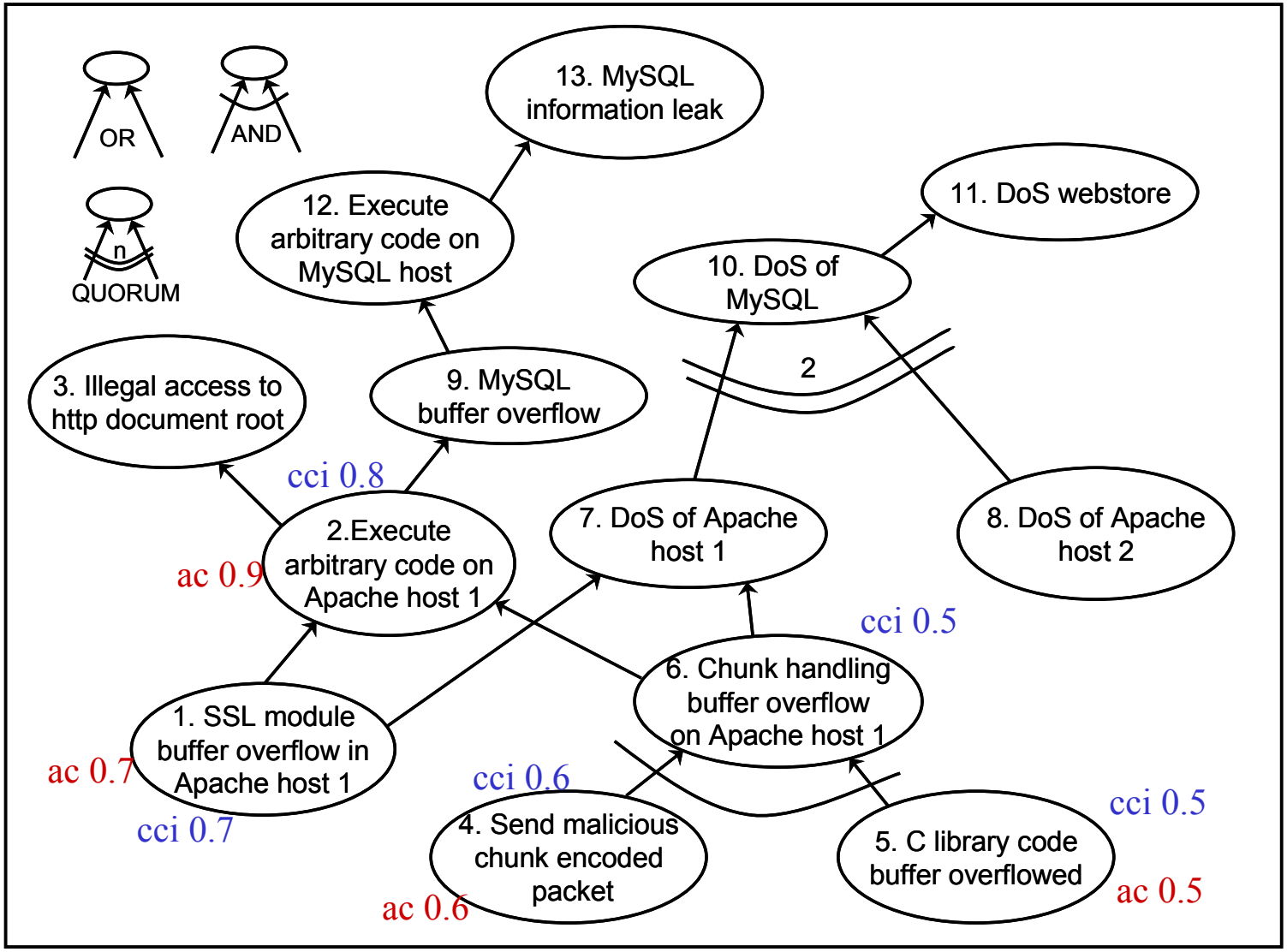
Conclusion

- ADEPTS provides a framework for automatic & systematic intrusion response
- Incorporate existing detection tools (Snort, Libsafe, …) and response tools (iptables, LIDS, …)
- An attack is viewed at a higher level
  - Mapping of attack sub-goals in the I-GRAPH
  - Enable the determination of the containment boundary for taking responses
  - Enable the feedback mechanism to decide whether deployed responses are effective
- Future Work
  - The various aspects in ADEPTS still need to be looked in more details for understanding their true performance
  - Provide recovery to the compromised parts

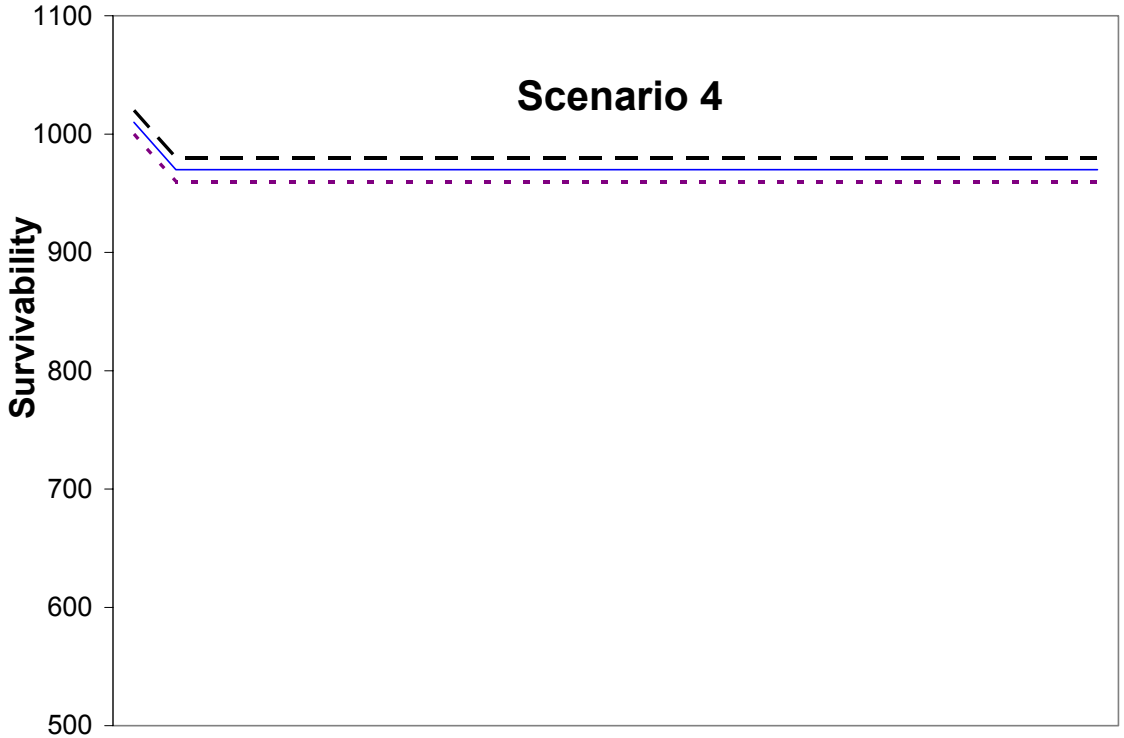
Experiments & Results

1. Survivability improvement
   - ADEPTS v.s. Local Response v.s. No Response
   - How well is ADEPTS in terms of maintaining the survivability
2. Impact from speed of propagation
   - ADEPTS's processing takes time
   - How well is ADEPTS in holding the survivability as speed of propagation increases?
3. Feedback Mechanism
   - Modification of EI values
   - Making effective responses more preferable

I-GRAPH


OR
AND
n
QUORUM
13. MySQL information leak
12. Execute arbitrary code on MySQL host
11. DoS webstore
10. DoS of MySQL
2
3. Illegal access to http document root
9. MySQL buffer overflow
cci 0.8
2.Execute arbitrary code on Apache host 1
ac 0.9
7. DoS of Apache host 1
8. DoS of Apache host 2
cci 0.5
6. Chunk handling buffer overflow on Apache host 1
1. SSL module buffer overflow in Apache host 1
ac 0.7
cci 0.7
cci 0.6
4. Send malicious chunk encoded packet
ac 0.6
5. C library code buffer overflowed
cci 0.5
ac 0.5

Effect of denial of service attack on survivability



Scenario #4
DDoS attack via issueing huge amount of leagal transactions (i.e. product search)

- http://www.linuxsecurity.com/content/view/117640/49/

- In the above example, all possible response messages are sent. Namely, both sender and receiver of an attack packet get a TCP reset and a sender gets 3 ICMPs (port, host, network unreachable). In fact, ICMPs can be used to stop UDP transmissions and TCP RSTs are used to tear down TCP sessions. That is how it looks in packet capture (tcpdump and browser run on host "anton", snort runs on host "fw"):
- In our tests, snort (v 1.8.4 and beta v. 1.9.1) does not always kill the HTTP connection using the RST and/or ICMPs. In most of the cases connection is reset and sometimes it remains running and the file (dummy " cmd.exe" placed on Apache web server) is successfully downloaded. The possible explanation is that RST arrives too late for the connection to be reset since the response from server comes earlier with the right sequence number. The delayed RST is then discarded. Thus RST/ICMP is not a reliable security mechanism (exactly as claimed in the snort documentation).