Achieving High Survivability in Distributed Systems through Automated Response

Yu-Sung Wu
Dependable Computing Systems Lab (DCSL) &
The Center for Education and Research in Information Assurance and Security (CERIAS)
School of Electrical and Computer Engineering
Purdue University

Survivable Systems and Intrusion Response

- Modern life heavily depends on computer systems
  - An inter-connected world
  - Physical boundaries disappearing
- Intrusions/security attacks to these systems occur
  - Malware / botnet / sophisticated attacks against organizations
    - GhostNet – a suspected cyber espionage network of over 1,295 infected computers in 103 countries (30% of which are high-value targets) in 2009

Survivable Systems and Intrusion Response

- Ways to make a system survivable
  - At design/implementation phase
    - Eliminate vulnerabilities
    - Policy / Access Control /
      - Challenge : “User Fr
        Control in Windows Vista o Linux ?)
  - In production phase
    - Use IDS to identify misuses/anomalies
      - system logs checking / system call hooking / network packet sniffing / virus scanning / VMM-based root kit detection..
    - Perform incident/intrusion response
      - Containment and Recovery
      - Stay transparent under normal operations
      - Intervene only when attacks are detected

Intrusion Response System

(focus of this work)

Existing Automated Response System

- Traditional Anti-Virus (AV) Product
  - Scan / Quarantine virus-infected files
- Host-based Intrusion Prevention System (HIPS)
  - An integration of (host-based) firewall, system-level action control, vulnerability detection and sandboxing on top of a traditional AV product.
  - Monitor malicious activities
    - virus, probing from network, attempt to modify critical entries in system registry, visiting phishing websites…
  - Response actions
    - Block access to known phishing websites
    - Quarantine infected files
    - Lock-up internet connection
    - Request user permission to continue on with suspicious activities
  - Norton 360, McAfee Total Protection, TrendMicro Internet Security Pro…

Existing Automated Response System

- Network-based IPS (NIPS)
  - A purpose-built hardware/software to inspect network traffic
    - Content-based detection
      - worm infections / hacks…
    - Rate-based detection
      - for denial of service attack
    - Protocol-analysis
      - existence of large amount of data in the User-Agent field of an HTTP request,…
  - Constantly engaged proactive response actions
    - Rate-limiting, traffic sanitization, IP address / port-number black/whilte-listing
  - Reactive response actions
    - Drop connection, terminate session, update firewall rules
  - Cisco IPS 4200 Series, 3Com Unified Security Platforms, Juniper SSG, …

Existing Automated Response System: Shortcomings

- Stand-alone systems / Minimal collaboration among IDS/ IPS boxes.
  - Attacks against distributed systems cause correlated damages to multiple system components.
  - Correlation of alerts improves both the detection accuracy and the understanding of an attack in distributed systems

Existing Automated Response System: Shortcomings

- Static mapping between detector and response action
  - Example: If "/bin/sh" is detected in network traffic (potential attempt to create a shell), then "black-list the source IP".
    - What if the response is not effective? What if it's a false alarm? What if the created shell only has limited privilege and is not really harmful?
- Pure NIPS or pure HIPS strategy is often not desirable
  - NIPS alone at the perimeter of a system
    - Limited view of attack manifestations
    - False alarm can cause degradation of system performance
    - Some organizations are interested in letting attack keeps propagating into the system till a point when significant damage is imminent
  - HIPS alone inside the system
    - Rely on host data for detection
    - More intrusive to applications
    - Last line of defense

PURDUE UNIVERSITY

Thesis Statement

- BASELINE Model of Automated Response in Distributed Systems
  - A collection of (detectors, response actions) pairs :
    - $\{(D_1,R_1), (D_2,R_2),\ldots, (D_k,R_k), \ldots, (D_N,R_N)\}$
  - For each pair, a mapping $f_k : D_k \rightarrow R_k$
  - f_k is designed based on expert knowledge
- Proposed Model of Automated Response in Distributed Systems
  - The set of all the detectors D and the set of all the response actions R

$$D = \bigcup_{k=1}^{N} D_k \quad R = \bigcup_{k=1}^{N} R_k$$

  - History of past attacks H
  - A mapping $f : (D,H) \rightarrow R$
  - f is designed to maximize expected system survivability based on the information accumulated in H and detectors D
  - f is designed to tolerate new types of attacks

PURDUE UNIVERSITY

Thesis Statement

- Evidence is proposed to show the validity of the following hypotheses:
  - The proposed model describes a set of responses, from which the expected system survivability is the upper bound of the expected system survivability from any set of responses generated from the BASELINE model.
  - In a practical system, it is possible to identify cases when the proposed model yields a higher system survivability than the BASELINE model.
  - It is possible that the use of history information in the proposed model can further improve system survivability.

Contribution (till Prelim)

- A Unified Framework for Automated Response in Distributed systems
  - Our system provides an integration over "detectors" found in existing IDS systems and "response actions" found in existing IPS systems.
    - Enable the collaboration of IDS / IPS technologies originally scattered across a system
- Dynamic Automated Response
  - The binding between detectors and response actions are determined dynamically based on
    - severity of the attack
    - the effectiveness of response
    - the cost of response
- => ADEPTS

Contribution (post Prelim)

- Adaptive Automated Response
  - Estimate the actual escalation of attack steps
    - avoid unnecessary responses
  - Estimate the effectiveness of response actions
    - avoid ineffective responses
- Response for Attack Variants
  - Use history of past similar attacks to improve response for new attack variants
- Optimality of Response Actions
  - To quantify how good a set of response actions from an IRS is
  - How to generate a set of (close to) optimal response actions in the runtime
- Response for Zero-day Attacks
  - Online attack graph generation based on system configuration and alerts
  - Conceptualization of attack graphs
- => SWIFT & ORIGIN (Zero-day Attacks)

Attack Model

- Multi-step (multi-stage) attack
  - Attack originates outside the network
  - Each step achieves certain privilege on a service
  - Elevated privilege is used to compromise a connected service
  - Ultimately some end goal is sought to be achieved
    - gaining read access to the credit card database
    - launching a DDoS to a targeted victim

Multi-Stage Attack Example

1. Malware downloaded as an e-mail attachment to CFO's office PC.

R1

1. Malware downloaded as an e-mail attachement.
2. CFO opens the e-mail, and the malware gets the CFO's security password through keystroke recording.

R2

1. Malware downloaded as an e-mail attachement.
2. CFO opens the e-mail and the malware gets the CFO's security password through keystroke recording.
3. The hacker uses the CFO's password to transfer money from the corporate bank account into his bank account.

R3

1. …
2. …
3. …
4. Corporate account is emptied.

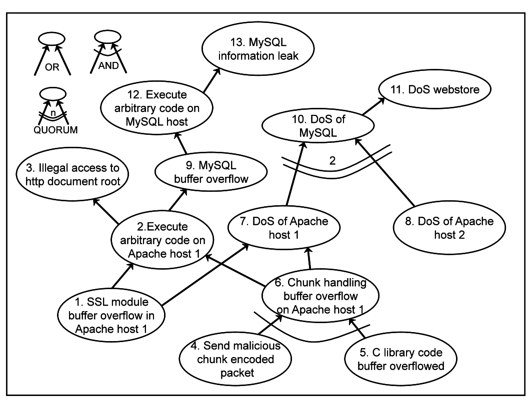
R_1: Remove the malware with anti-malware tools.

R_2: Change the CFO's security password.

R_3: Freeze the corporate account.

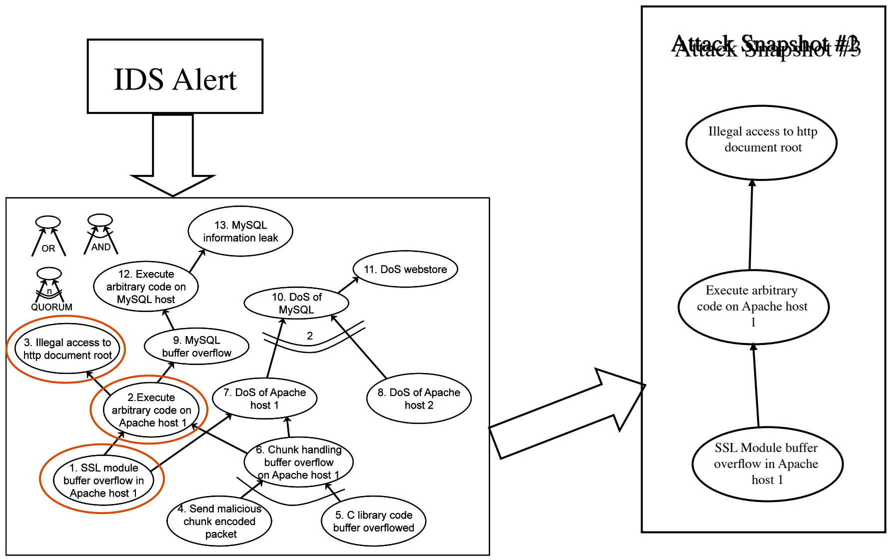
I-Graph

- An attack graph that models all potential (worst-case scenario) attack steps and their causal relations for a target system
  - Can be built with techniques such as Sheyner [S&P'02], Ou [CCS'06], …



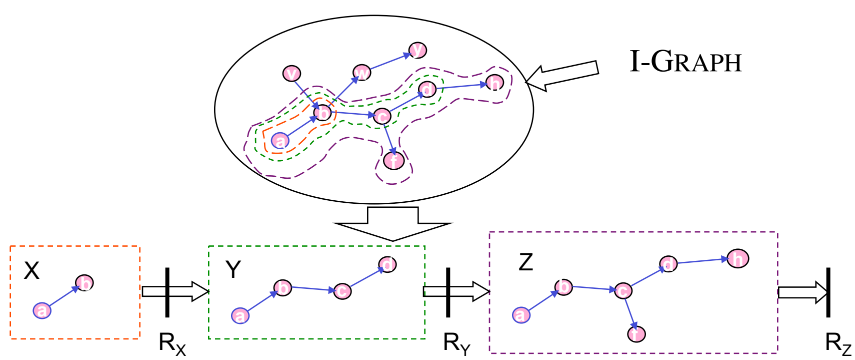
PURDUE UNIVERSITY

Attack Snapshot



Dynamics between attack and responses

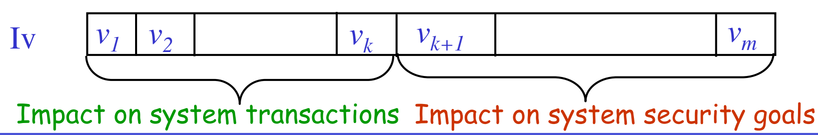
- Successive attack snapshots created for incoming IDS alerts



- Assuming an attack includes three "snapshots" X, Y, and Z
- Each snapshot includes I-GRAPH nodes which have been achieved as part of the attack thus far
- Following each snapshot k, SWIFT determines a response combination R_k (a set of response actions) to deter the escalation

Impact Vector

- A system has transaction goals and security goals that it needs to meet through the time of operation
  - Example: provide authentication service & preserve privacy of sensitive data
- Attacks are meant to impact some of these goals
- Deployed responses also impact some of these goals
  - For example, by temporarily disabling some functionality for legitimate users as well
- Assume the impact can be quantified through a vector Iv
  - Each element in the Iv corresponds to the impact on each transaction/ security goal ∈ [0, ∞]



PURDUE UNIVERSITY

Optimality of Response Actions

- We formally define the cost for a response combination (a set of response actions) RC_i as:

$$Cost(RC_i) = \sum_{n_k \in \text{I-GRAPH}} Iv(n_k) \times Pr(n_k) + \sum_{r_k \in RC_i} Iv(r_k)$$

$Iv(n_k)$: Impact from reaching an attack step node n_k

$\Pr(n_k)$: Probability of reaching node n_k

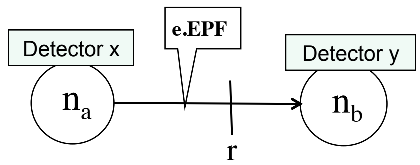
$Iv(r_k)$: Impact from deploying the response r_k

- The response combination RC_i is said to be optimal for a given attack if it achieves the minimal Cost(RC_i)
  - In ADEPTS, optimality achieved "per node and per out-going edge"

PURDUE UNIVERSITY

Determine $Pr(n_k)$: Compromised Confidence Index

- Goal is to determine the probability of each attack step being achieved



$Pr(n_k) \propto cci(n_k)$

EI(r)=1-Prob(r fails)

For an edge e connecting node n_a to n_b in I-GRAPH with response r :

$cci(n_b) = $ $cci(n_a)$ Pr(r fails) e.EPF , if n_b has no detect or

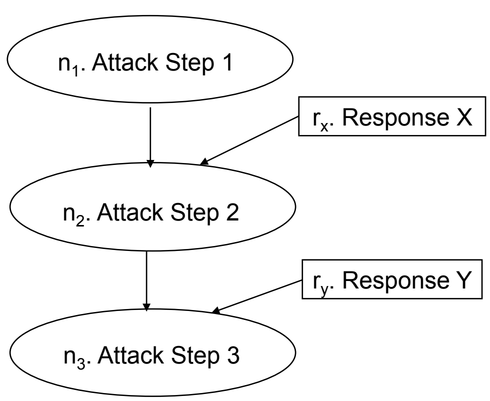
$\alpha[cci(n_a)$ Pr(r fails) e.EPF(y) $]/2$, if n_b has detector y

e.EPF : The edge propagation factor of edge *e*. This models an adversary's likelihood of taking this edge

PURDUE UNIVERSITY

Determine $Pr(n_k)$: Bayesian Inferencing

n_1=T	0.2



r_x=T	0.4

n_1	T		F	
r_x	T	F	T	F
n_2=T	0.3	0.8	0	0

r_y=T	0.5

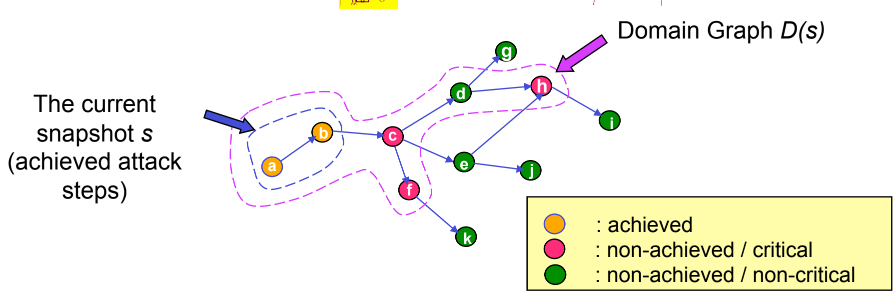
n_2	T		F	
r_y	T	F	T	F
n_3=T	0.2	0.9	0	0

PURDUE UNIVERSITY

Domain Graph

- Limit the response search space for a snapshot *s* to a subset of I-GRAPH, namely the **Domain Graph** *D(s)*
- *D(s)* includes critical nodes from I-GRAPH
  - A node *n* is critical if |Prob(*n*)*Iv(*n*)| is greater than a given threshold
  - Also include nodes on the path leading to critical nodes

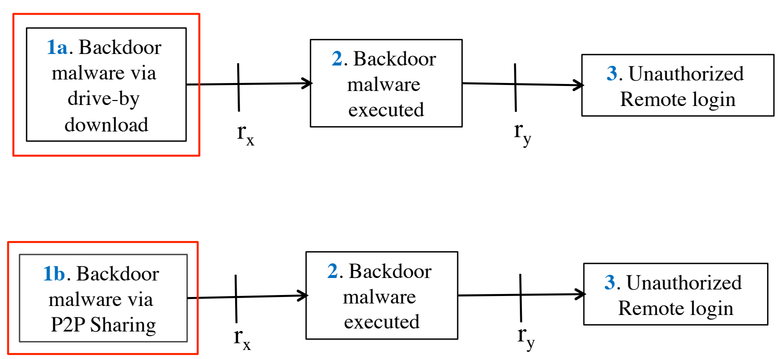
⇨ *Cost*(*RC*) is minimized.



PURDUE UNIVERSITY

Utilize History from Similar Attack

- Variations in attacks are common



r_x : Disallow execution of the downloaded file
r_y : Block connections from external network

PURDUE UNIVERSITY

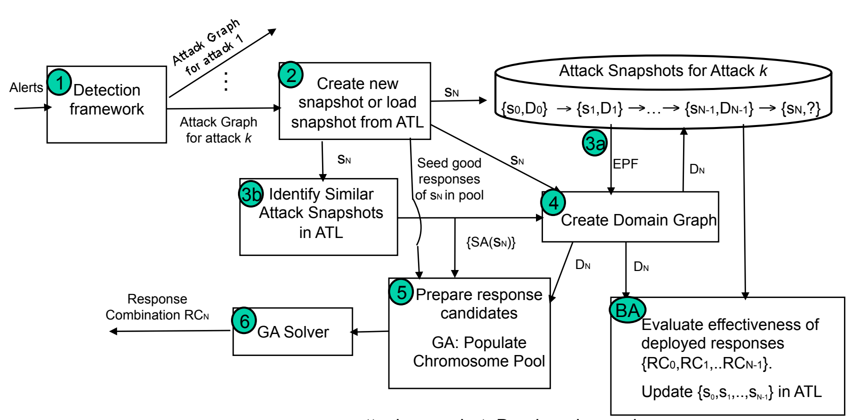
Utilize History from Similar Attack

- Similarity of Attack Snapshots

$$Sim(s_x, s_y) = \frac{\#\text{ of nodes and edges in } s_x \cap s_y}{\#\text{ of nodes and edges in } s_x \cup s_y}, \text{ assume } s_x \cup s_y \text{ is non-empty}$$

- History information from a similar attack snapshot
  - EI values of responses
  - EPF values of edges
  - Effective Response Combinations

Summary of the process in SWIFT

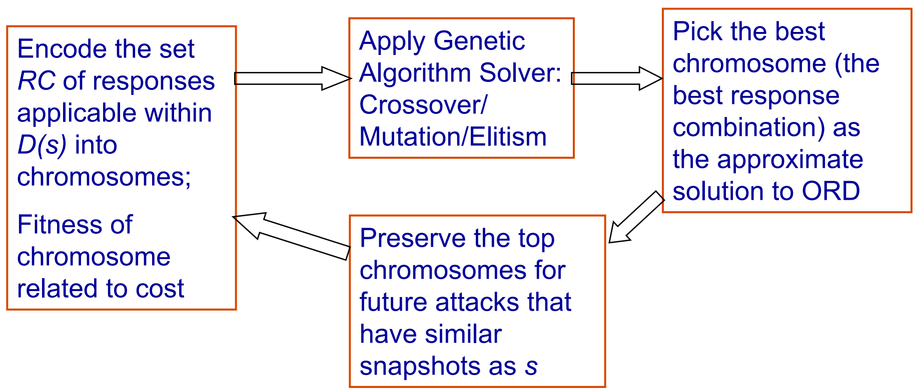


s_N: attack snapshot, D_N: domain graph

Edges represent flow of information, encircled numbers in a box represent the temporal ordering in the execution flow (3 happens before 4, while 3a and 3b are concurrent, BA implies step occurs between attacks)

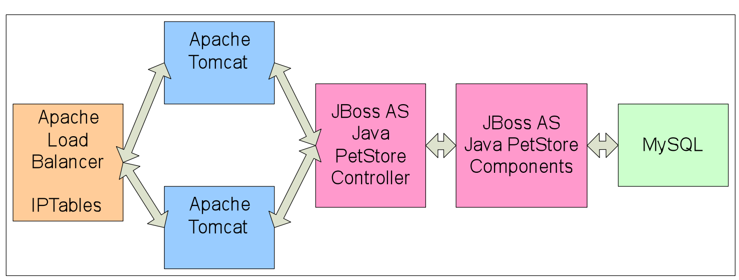
Approximate O.R.D. with Genetic Algorithm

- We proved Optimal Response Determination (O.R.D.) to be NP-hard by mapping the Set Covering Problem to it



Experimental Testbed

- A three-tier e-commerce system as the reference basis for constructing attack scenarios



Experimental Setup

- Detectors and Response Actions:
  - SNORT
  - Iptables
  - LIDS (program / file MAC. similar to SELinux)
  - Kill process (the kill command on UNIX-like systems)
  - Bank Credit Card Account Activity Monitor
  - File Access Monitor (log file access that falls outside a pre-defined white-list)
- BASELINE (LOCAL RESPONSE)
  - Snort is configured to block source IP address, which emanates malicious traffic via Snort rule action and Iptables
  - Bank CC Account Monitor freezes account when suspicious transaction is detected
  - Mimic what we see as the current mainstream IDS / IPS / IRS deployment paradigm

Two Sample Attack Scenarios

Steps	Scenario 0	Scenario 1
0	Exploit Apache mod_ssl buffer overflow.	Use php_mime_split (CVE-2002-0081) buffer overflow to insert malicious code into Apache.
1	Insert malicious code.	'ls' to list webstore document root and identify the script code informing the warehouse to do shipments.
2	Ip/port scanning to find vulnerable MySQL server.	Send shipping request to warehouse and craft the request form so that a warehouse side buffer overrun bug fills the form with a victim's credit card number.
3	Buffer overflow MySQL to create a shell (/bin/sh).	Unauthorized orders are made.
4	Use malicious shell to steal information stored in MySQL.	

PURDUE UNIVERSITY

- **Survivability Metric**

  Survivability = 1000 − Σ unavailable transactions − Σ failed security goals − cost of deployed resposnes

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

Transactions

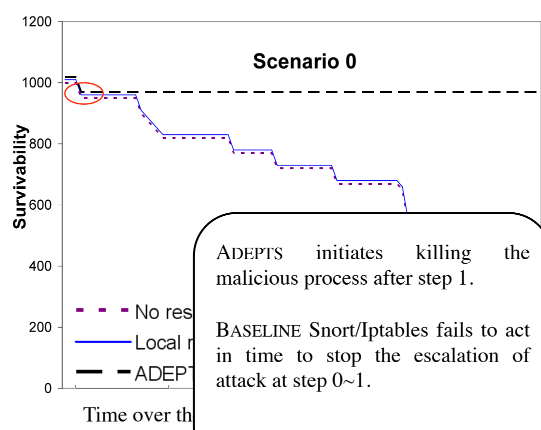
Illegal read of file	20
Illegal write to file	30
Illegal process being run	50
Corruption of MySQL database	70
Confidentiality leak of customer information stored in MySQL database	100
Unauthorized orders created or shipped	80
Unauthorized credit card charges	80
Cracked administrator password	90

Security Goals

PURDUE UNIVERSITY

Survivability Improvement over Local Responses

Effect of confidentiality attack on survivability



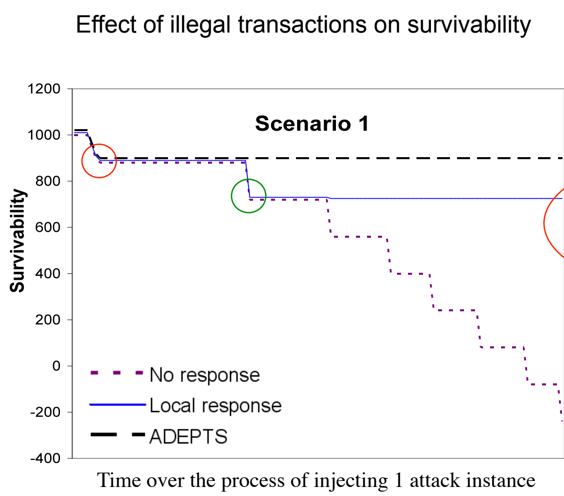
ADEPTS initiates killing the malicious process after step 1.

BASELINE Snort/Iptables fails to act in time to stop the escalation of attack at step 0~1.

Steps	Scenario 0
0	Exploit mod_ssl buffer overflow in Apache.
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.

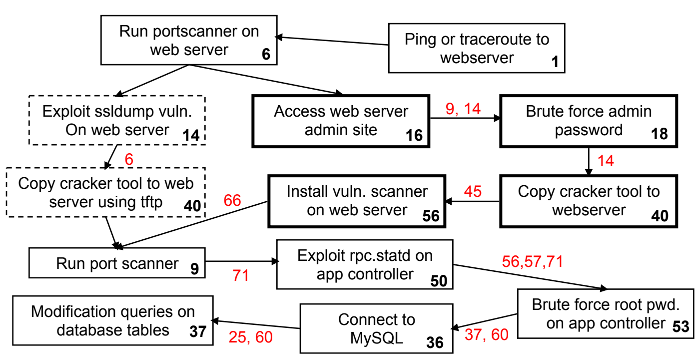
PURDUE UNIVERSITY

Survivability Improvement over Local Responses



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.

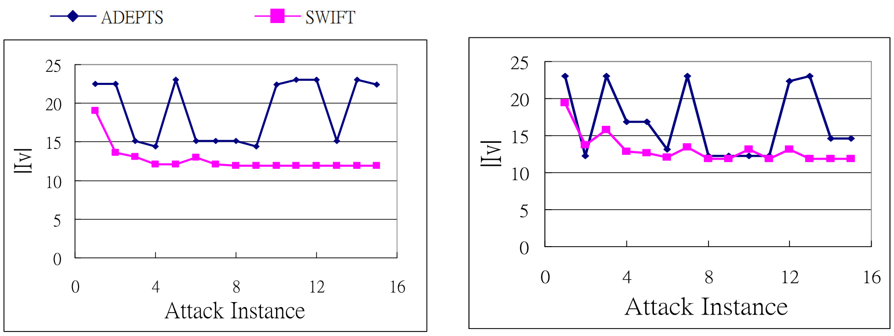
ADEPTS v.s. SWIFT on E-Commerce Attack Scenario



Dashed line: AS3, Thin solid line: AS3 and AS4, Thick line: AS4

Attack scenarios 3 and 4, used for experimental evaluation.
Dashed box: AS 3, Thick box : AS 4; Thin box: Common to AS 3 and AS 4.
Effectiveness of R_{60} set erroneously low and others set erroneously high.

ADEPTS v.s. SWIFT on E-Commerce Attack Scenario



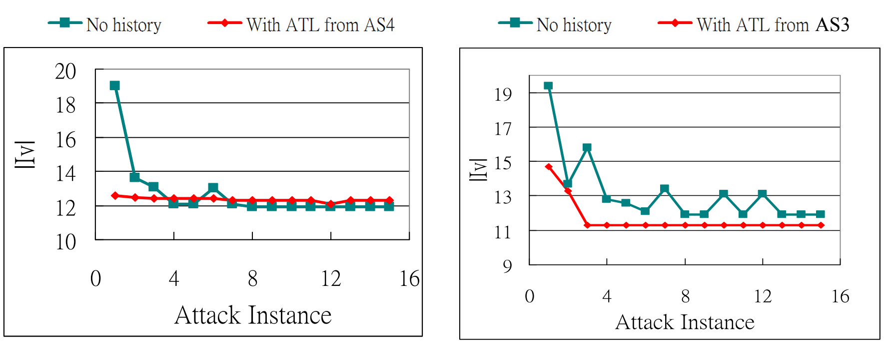
AS3 AS4

- SWIFT has consistently lower |Iv| than ADEPTS
- For AS3, ADEPTS' performance is wildly fluctuating since it deploys responses close to nodes that are achieved
  - Such responses can fail more often due to insufficient time for full deployment
- For AS4, the performance of SWIFT and baseline are closer
  - There are more local responses available

PURDUE UNIVERSITY

Response for Attack Variants

(a) Execute AS4 15 times, then execute AS3; (b) Execute AS3 15 times, then execute AS4



(a) (b)

- Difference lies in resilience to first attack instance
- Lower |Iv| implies SWIFT would be able to respond better to damaging attacks, if an attack with shared stages has been observed before

PURDUE UNIVERSITY

ORIGIN : Response for Zero-day Attacks

- Challenge
  - Zero-day attacks exploit unknown vulnerabilities
    - Assume "generic" detectors can pick up some of the attack stages
      - Buffer overflow detectors / Array bounds check (Java, C#, …)
      - Application level detector (e.g. excessive # of failed logins)
      - Deletion / modification of key system files / registry
- Contributions
  - Online modeling of Zero-day attacks from detectable attack stages
    - Can't assume an I-GRAPH encompassing all possible zero-day attacks
  - Conceptualization: abstract the knowledge in ATL to deal with Zero-day attacks
    - Many zero-day attacks bear similar concepts from past attack: Example: implanting malware => stealing credentials => unauthorized activity

Online modeling of Zero-day Attack

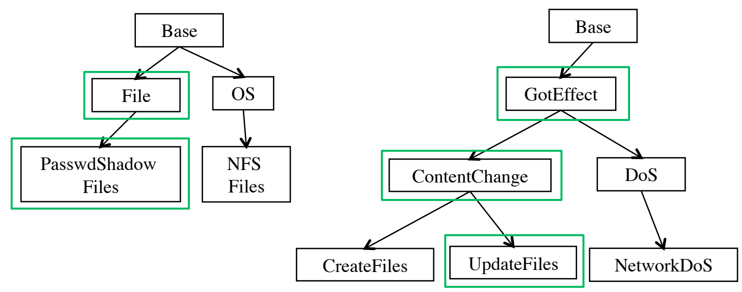
- Define an attack stage as a pair of (detector alert D, component C)
  - Literally, receiving alert D from a detector associated with component C in the protected system.
- An object-oriented description of the configuration of the protected system
  - Components in the system
  - Detectors associated with components
  - Connection flows between associated components/detectors
    - Information flow
    - Privilege propagation flow
- Generate attack graph for an ongoing attack in the runtime

Conceptualization of Attack Graph

- Conceptualize the component and the detector alert for each Attack Stage.

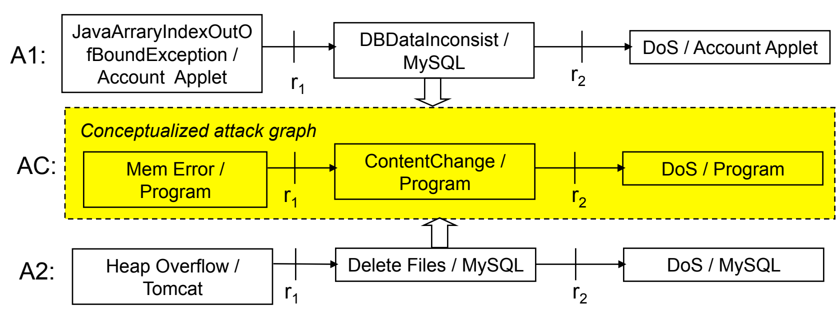
C_Lv: 2
D_Lv: 3

C : PasswdShadow Files
D : ContentChange



Conceptualization of Attack Graph

- Conceptualized attack may match with an attack in the ATL



r_1: Disable connection from tomcat/applet to MySQL

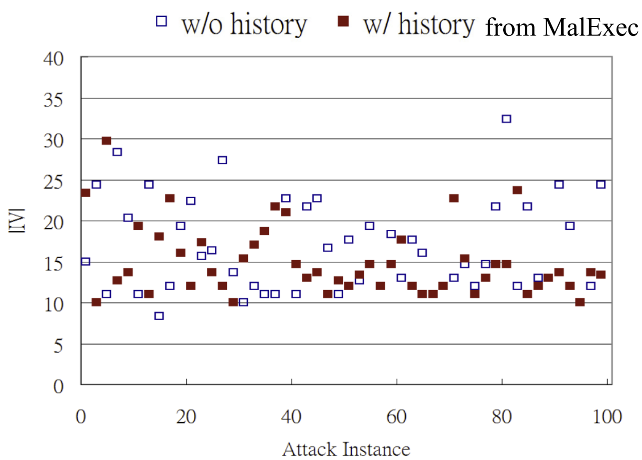
r_2: Rollback to last data files checkpoint

ORIGIN / Response for Zero-day Attacks

- Experiment Overview
  - Use three attack scenarios which bear similarities after being conceptualized
    - MIT LLDoS (used in many attack graph publications)
    - MalExec (Ou CCS'06)
    - ModSSL (synthetically created with EPF / EI parameters contradicting with the other two scenarios)
  - Compare |Iv| with/without conceptualization
    - SWIFT and ADEPTS perform almost like BASELINE
      - The topologies of zero-day attacks are assumed non-existent in the I-GRAPH.
      - ORIGIN does not use pre-built I-GRAPH
      - Response EI tuning in SWIFT and ADEPTS is the only advantage from our IRS over a BASELINE

PURDUE UNIVERSITY

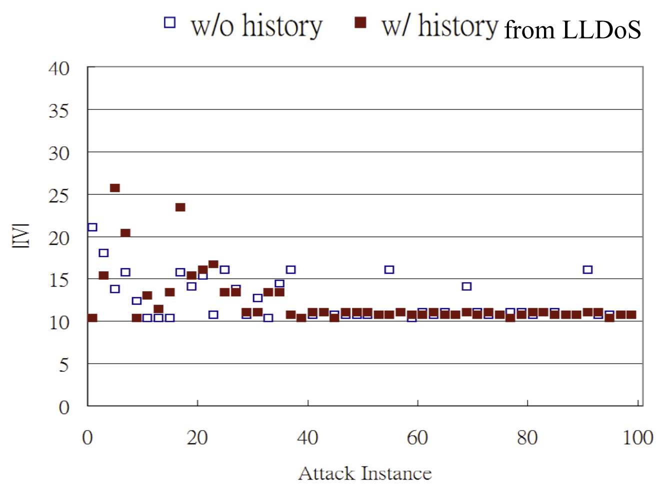
Response for Zero-day Attacks



Running LLDoS with **No** conceptualization

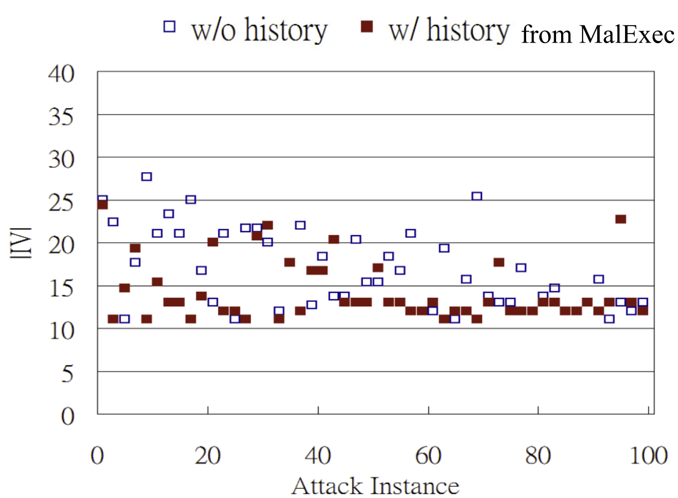
PURDUE UNIVERSITY

Response for Zero-day Attacks



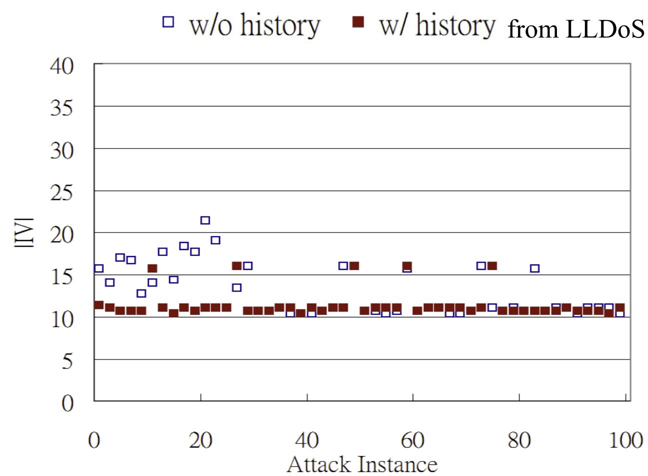
Running MalExec with **No** conceptualization

Response for Zero-day Attacks



Running LLDoS with conceptualization

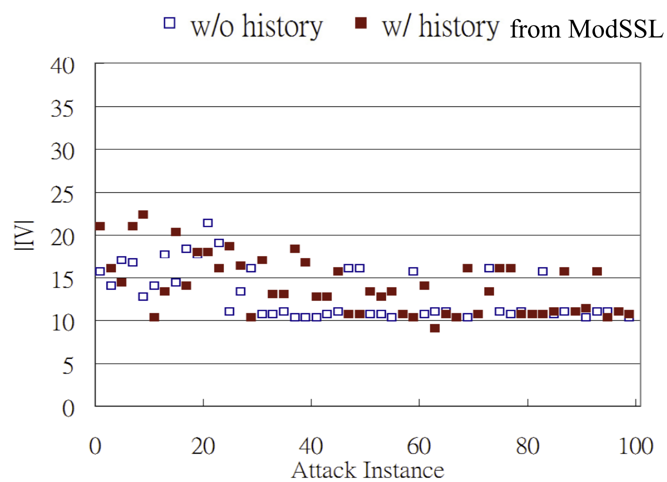
Response for Zero-day Attacks



Running MalExec with conceptualization

PURDUE UNIVERSITY

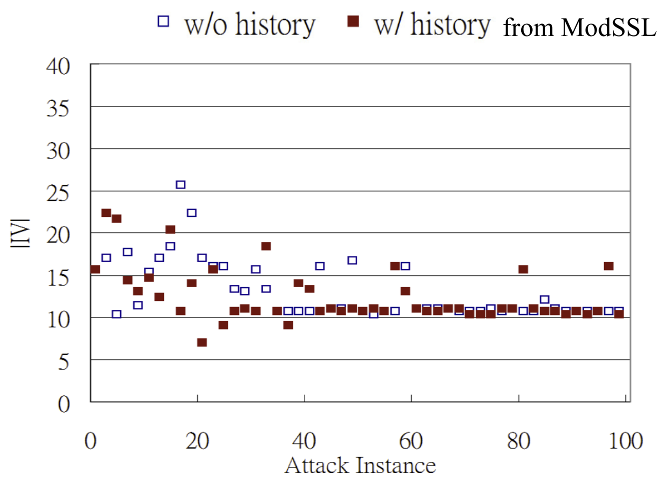
Response for Zero-day Attacks



Running MalExec with conceptualization

PURDUE UNIVERSITY

Response for Zero-day Attacks



Running MalExec with **No** conceptualization

Conclusion

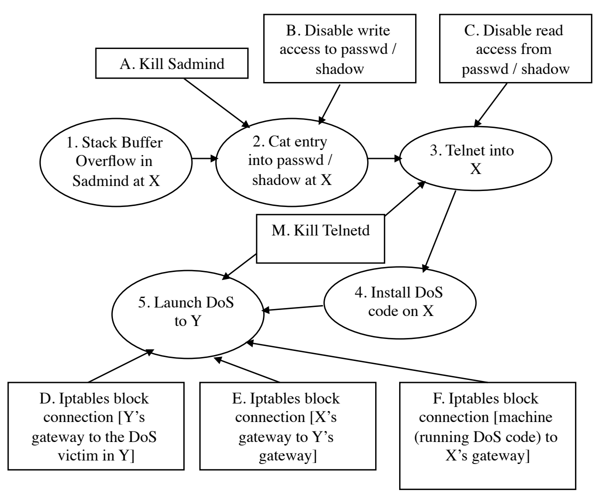
- Propose a unified framework of dynamic and adaptive automated response system for distributed systems
  - Improved survivability over existing baseline solution
- Define a framework to reason about and approach the optimality of responses
  - Further improved survivability by finding and deploying globally optimized response
- Use conceptualization to utilize history from past attacks to achieve effective responses to Zero-day Attacks

Further Work

- Share history information about attacks across systems
  - Similar to sharing virus / malware signatures nowadays
  - Aim to shorten / eliminate the adaption phase
- Conceptualization can hurt
  - This occurs when using poisonous history from a conceptualized past attack whose characteristic is actually very different from the current one being handled

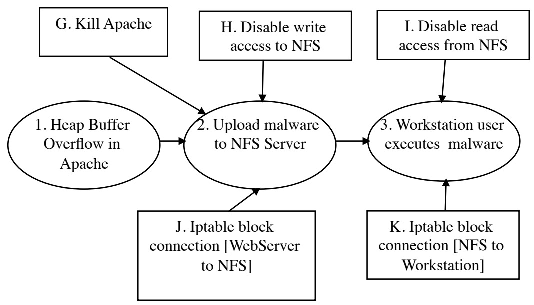
Response for Zero-day Attacks

- AS : MIT LLDoS

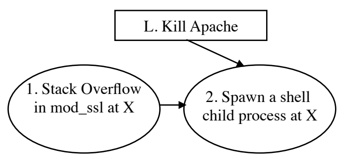


Response for Zero-day Attacks

- AS : MalExec



- AS : ModSSL



Proof of Thesis Statement #1

1. WLOG, assume an attack, which includes detector alerts $D_1, D_2, \ldots, D_N$.
2. In the BASELINE model, assume mappings $\{\overline{f_1}, \overline{f_2}, \ldots, \overline{f_N}\}$ gives the highest expected system survivability, where $\overline{f_k} : D_k \rightarrow \overline{R_k}$
3. In the proposed model, we can have a mapping $\overline{f}$ constructed as follow

$$\overline{f} : (\{D_1, D_2, \ldots, D_N\}, \varnothing) \rightarrow \{\overline{R_1}, \overline{R_2}, \ldots, \overline{R_N}\}$$

4. We now have a mapping $\overline{f}$ in the proposed model which describes the set of responses, which yields the same highest expected system survivability as from the BASELINE model. This corresponds to the upper bound of the expected system survivability from any sets of responses from the BASELINE model.