The Search for Efficiency in Automated Intrusion Response for Distributed Applications

Yu-Sung Wu, Gaspar Modelo-Howard, Bingrui Foo, **Saurabh Bagchi**, Eugene Spafford

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
- Intrusions/security breaches to these systems occur
- Ways to make a system survivable
  - At design/implementation phase
    - Eliminate vulnerabilities
    - Policy/Access Control/Cryptography/Formal Verification
  - In production phase
    - Use IDS (system logs checking/network packet sniffing/virus, worms scanning, detecting files modifications…) to identify misuses/anomalies
    - Perform incident/intrusion response (IRS) to detected misuses/anomalies
      - Containment and Recovery

Focus of this work

Why Intrusion Response System (IRS)?

- Intrusions/security breaches to computing systems occur
- A survivable system needs to provide functionality through intrusions
- Human intervention after IDS alert can be costly and slow
- IRS needed to take reports from IDS and carry out actions to counter the intrusion
- Existing examples of IRS
  - Anti-virus software which disables access to worm executables or files infected with virus
  - Iptables which terminates a session on matching a malware signature
  - Web browser blocks access to known malware / phishing websites

Intrusion Response System State-of-the-Art

- Summary on existing IRS
  - Most of them are stand-alone and target one machine box only
  - They are tied through static mapping to specific IDS alerts
- IRS for Distributed Systems
  - An environment of multiple interconnected boxes with heterogeneous and cooperating services
  - Few general-purpose IRS solutions exist for distributed systems
  - The most common way is to use the stand-alone solutions separately and independently on the boxes
    - E.g., Have McAfee anti-virus software installed on the workstation boxes, and CISCO IPS on the edge routers

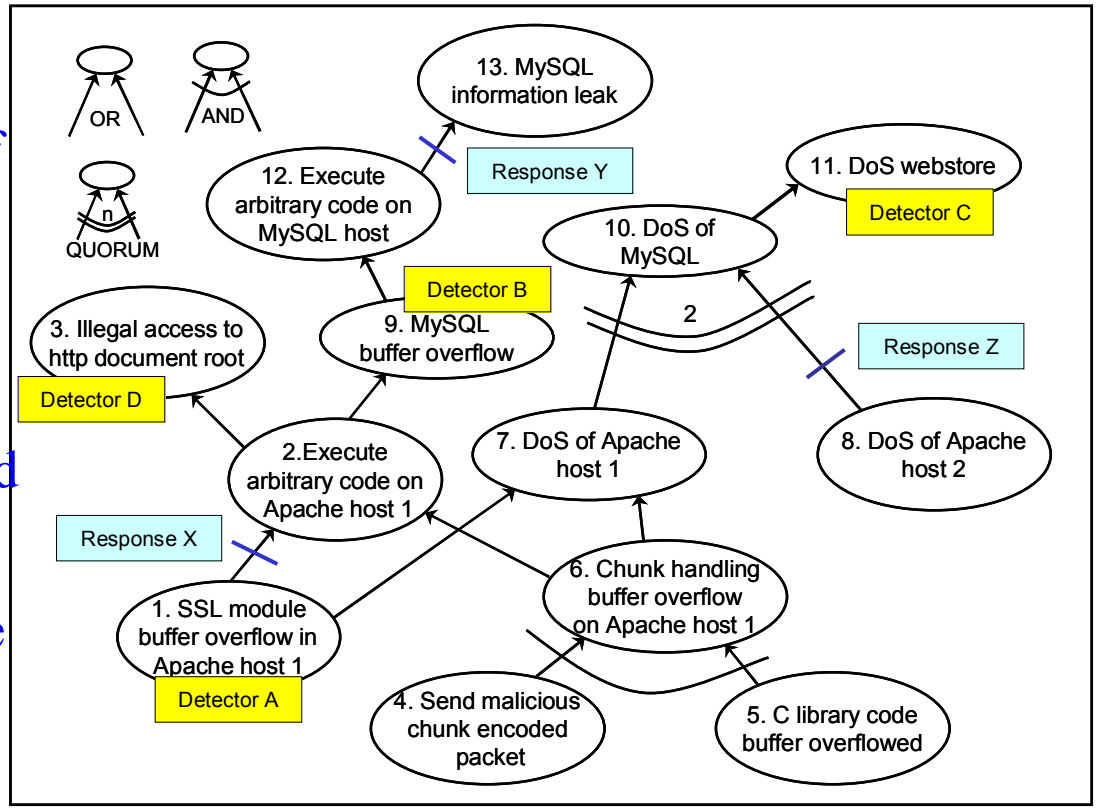
IRS for Distributed Systems

- Drawback of the "common way"
  - Each IRS/IDS pair does not leverage the detection reports from the other IDSs
    - Existing research on alert correlation has shown clear advantages of doing so
  - The IRS/IDS pair does not consider the effects from the response actions carried out by the other IRS/IDS pairs
    - This can lead to redundant response actions and denial-of-service of the system at worst
  - At best, locally optimal decisions from each IRS/IDS pair
    - There is no guarantee of system-wide global optimality

ADEPTS/SWIFT IRS: Basics

- Specifically designed for distributed systems
- Use I-GRAPH (a variant of attack graph) as the core binding between detectors (stand-alone IDSs) and response actions (stand-alone IRSs)

- Detector is associated with some I-GRAPH nodes to tell if the node goal is achieved
- Inference done based on alerts and graph structure
- Response actions are associated with edges
- A response is meant to stop the progression of the intrusion from the source node to the destination node



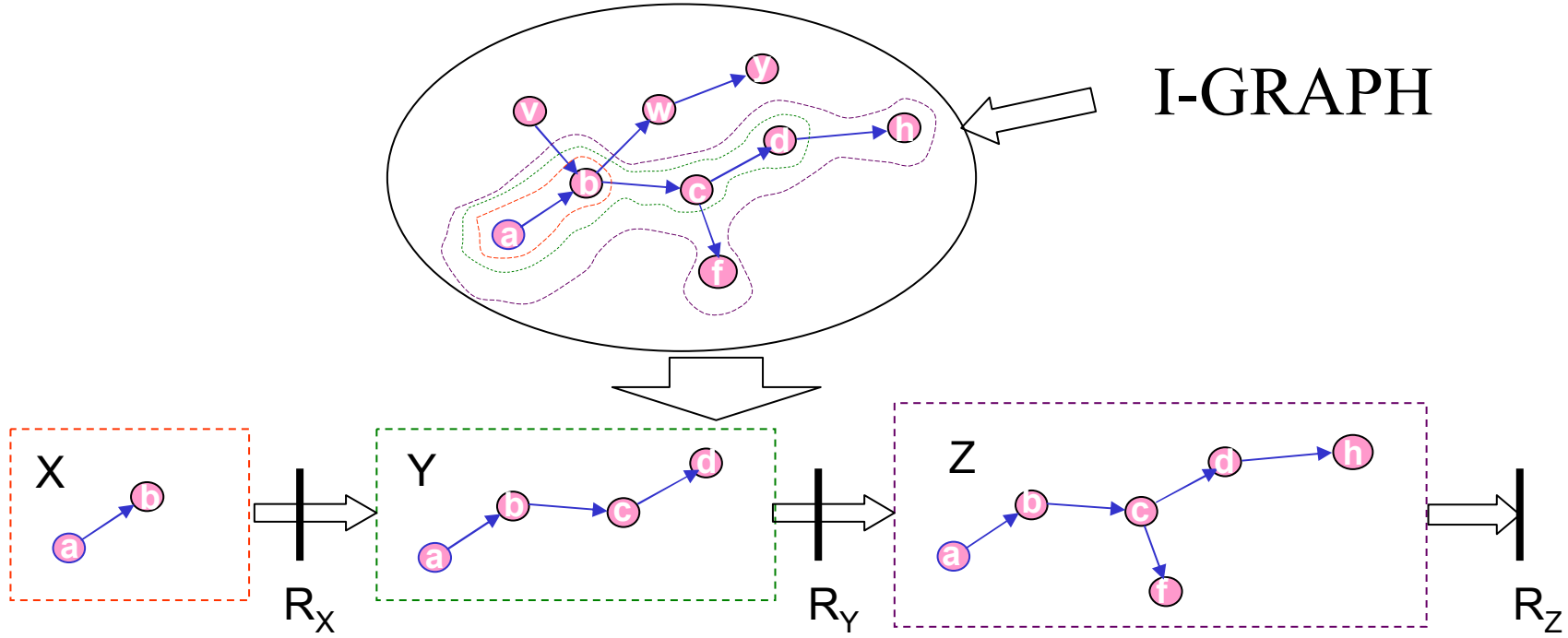
PURDUE UNIVERSITY

Attack Model

- Multi-stage attack
  - One stage achieves privilege on a service
  - Elevated privilege is used to compromise a connected service
  - Ultimately some end goal is sought to be achieved, such as, gaining read access to the credit card database
- External network-based attack
  - Starts from the exterior of the system
  - First compromises the external-facing services, such as, the web server

Attack Snapshots

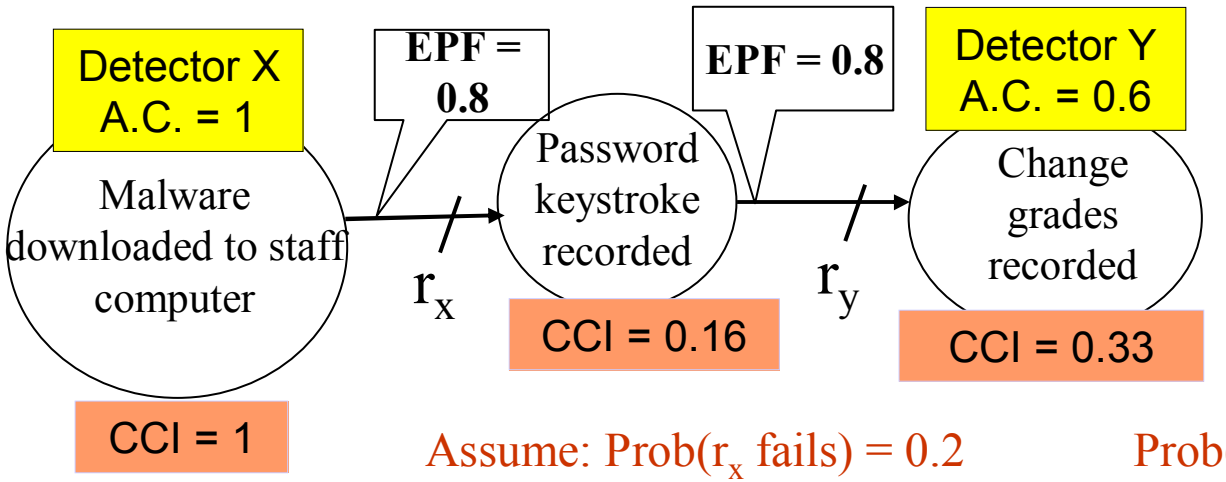
- The dynamics between intrusion and response



- 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 set of response actions R_k for deployment

Diagnosis of Achieved I-GRAPH Nodes

- Goal is to perform inference and determine (probabilistically) which nodes have been achieved



Assume: Prob(r_x fails) = 0.2 Prob(r_y fails) = 0.4

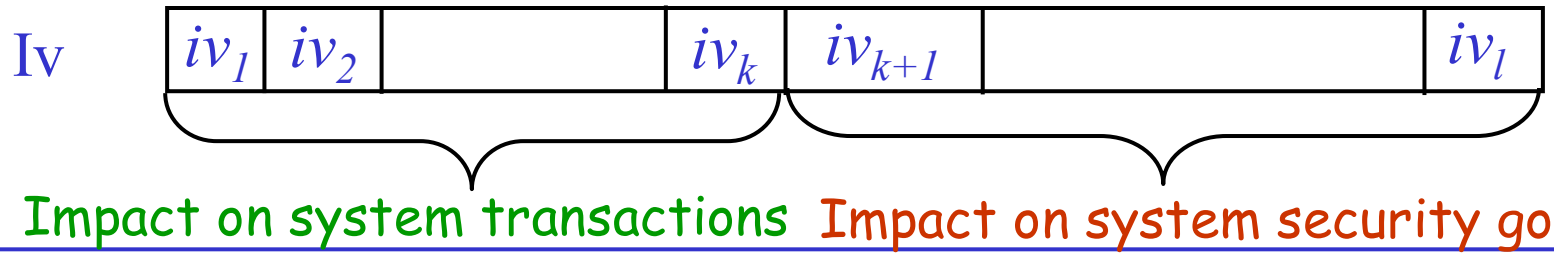
For an edge e connecting node a to b in I-GRAPH with response r :

$$cci(b) = \begin{cases} cci(a) \cdot \Pr(r \text{ fails}) \cdot e.EPF & \text{, if } b \text{ has no detector} \\ cci(a) \cdot \Pr(r \text{ fails}) \cdot e.EPF \cdot AC(x)/2 & \text{, if } b \text{ has detector } x \end{cases}$$

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

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 from an attack to the system can be quantified through a vector Iv
  - Each element in the Iv corresponds to the impact on each transaction/security goal ▪ [0, 1]



Optimality of Response Actions

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

$$Cost(RC_i) = |Iv(RC_i)| \left| \sum_{k=1}^{m} Iv(n_k) \Pr(n_k) - \sum_{k=1}^{n} Iv(r_k) \right|$$

$Iv(n_k)$: Impact from reaching an I-GRAPH node

This is conditioned by the probability of reaching that node

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

- The response combination RC_i is said to be optimal for the given attack if it achieves minimum Cost(RC_i)

Compare ADEPTS with SWIFT

- ADEPTS [Elsevier Journal of Computer Networks 07, DSN 05]
  - Achieve local optimality w.r.t. each flagged I-GRAPH node

For each node n_i and for each child node $c_{i,k}$ of n_i, pick the response $r_{i,k,m}$ such that

$$\left| Iv(c_{i,k}) \Pr(c_{i,k}) - Iv(r_{i,k,m}) \right| \text{ is minimized.}$$

- SWIFT
  - Achieve global optimality w.r.t. whole I-GRAPH (domain graph)

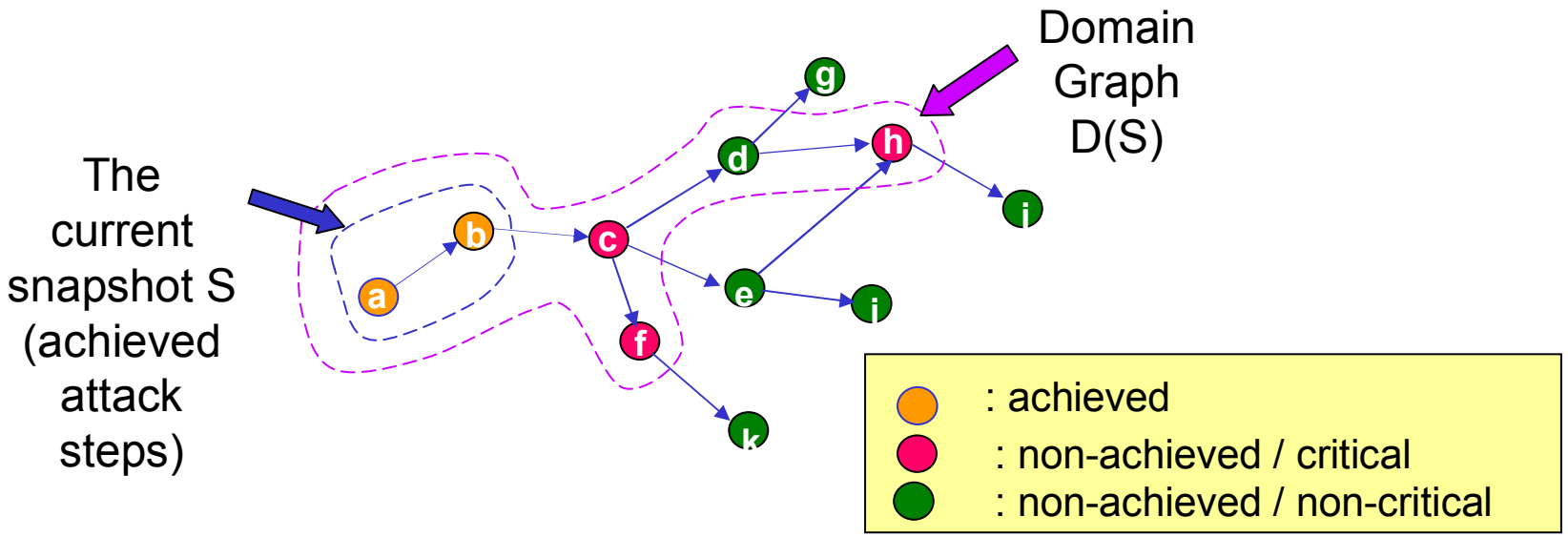
$$Cost(RC_i) = |Iv(RC_i)| \left| \sum_{k=1}^{m} Iv(n_k) \Pr(n_k) - \sum_{k=1}^{n} Iv(r_k) \right| \text{ is minimized}$$

Types of Response Actions

- Given a snapshot *s* and I-GRAPH *G*
  - In terms of containment, consider all response actions applicable to the graph (*G*-*s*)
  - In terms of recovery, consider all response actions applicable to the graph *s*
- *s* is typically not huge, as its size is linear in the number of detectable steps in a multi-stage attack
- But (*G*-*s*) can be huge
  - A function of applicable attack steps (vulnerabilities) in all services in the application system
  - However, for nodes in (*G*-*s*) which are far away from *s*, the likelihood of them being reached is lower than for closer ones

Domain Graph

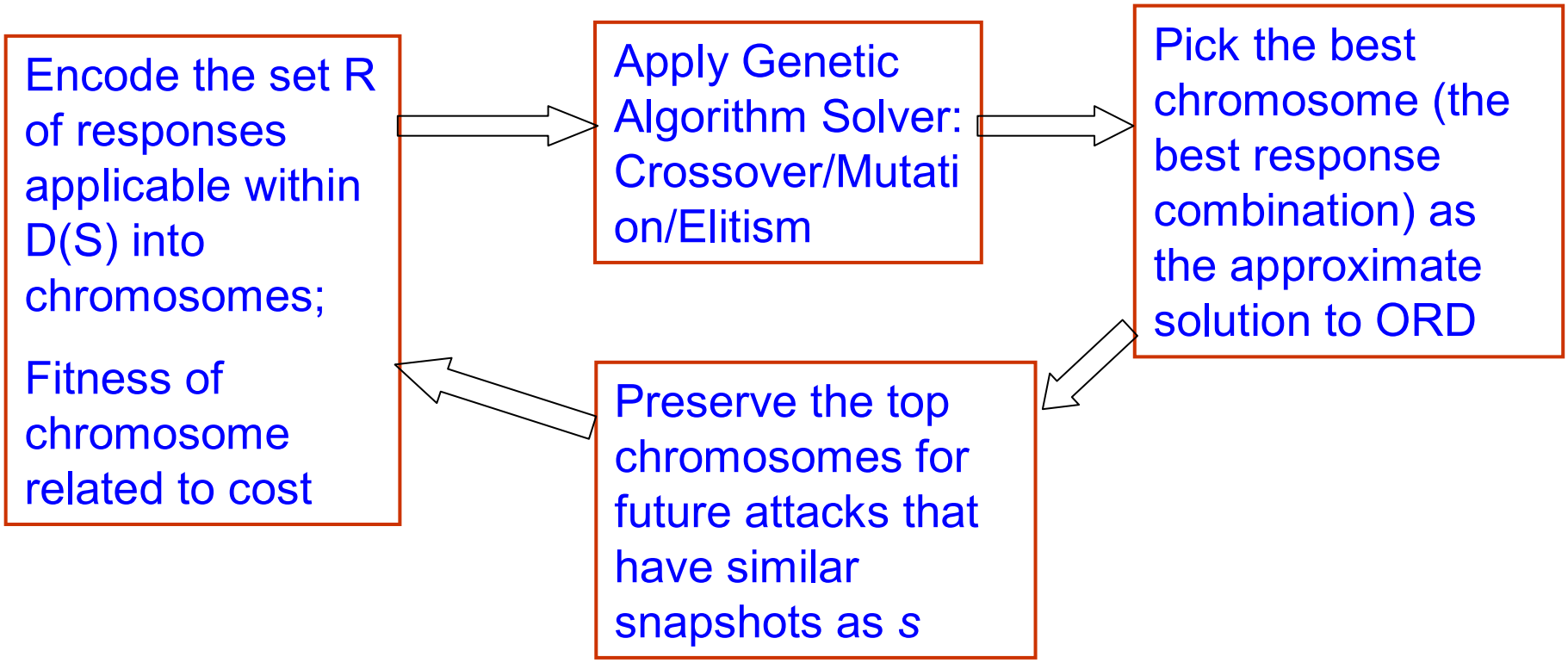
- Our solution is to limit the response search space for a snapshot *s* to a subset of (*G-s*), namely the **Domain Graph** *D(s)*
- *D(s)* includes critical nodes from I-GRAPH
  - A node *n* is critical if CCI_n*IV_n is greater than a given threshold



PURDUE UNIVERSITY

Approximate O.R.D. with Genetic Algorithm

- Optimal Response Determination is proved to be NP-hard by mapping the Set Covering Problem to it

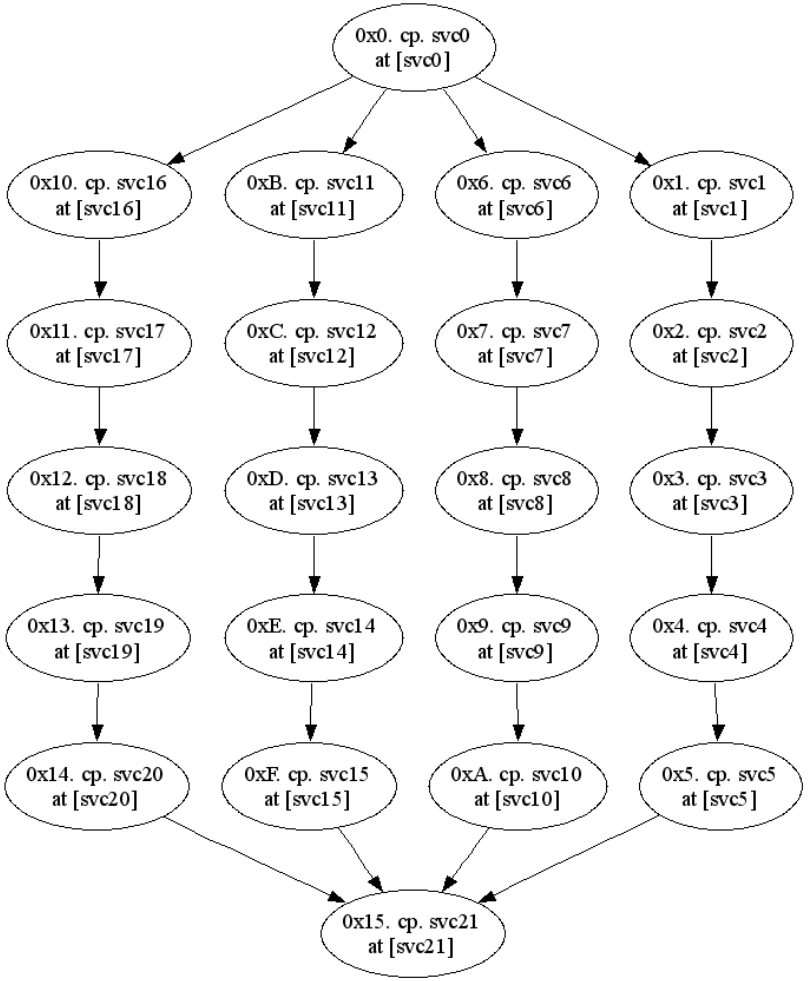


Why Choose GA?

- Approximate solution technique is called for since exact solution is NP-hard
- Trade-off between the optimality of the solutions and the computational expense is adjustable by controlling the size of the domain graph and the number of evolutions in the GA based solver
- Good solutions (response combinations) from previous instances are preserved into the chromosome population for future instances of similar attacks
  - This speeds up the search for good response combinations for attacks which share some stages with previously observed ones

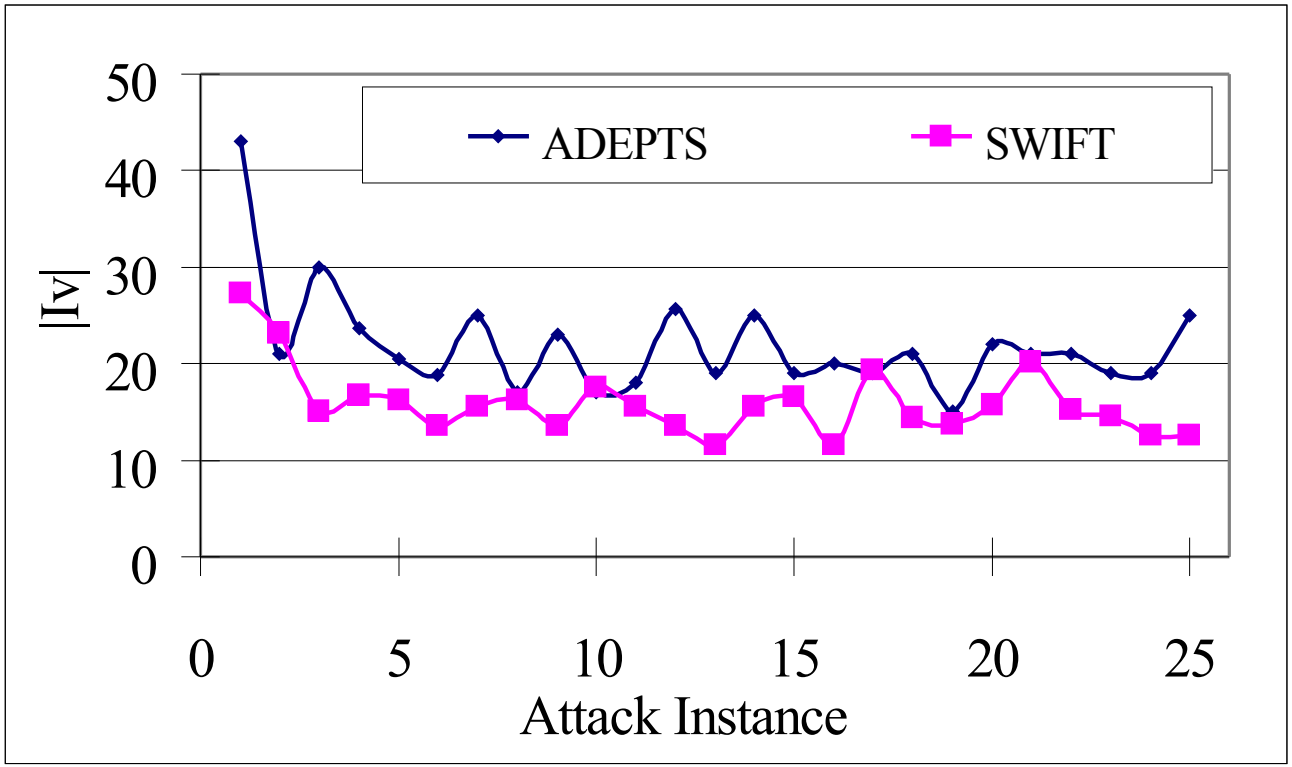
Experiment 1: Survivability for Micro-Benchmark

- A micro-benchmark Attack Scenario
  - a regular structure with each node representing a unique service being affected
  - 22 single-node responses
  - 10 dual-node responses for each pair of nodes {(1,6), (11,16), (2,7), (12, 17),…}
  - A dual-node response has a lower IV than the combined IVs from the two counterpart single-node responses



PURDUE UNIVERSITY

Experiment 1: Survivability for Micro-Benchmark



- SWIFT gives a 27% improvement
- Decreasing trend with history
- Even for early attack instances SWIFT performs better

Experiment 2: Learning from History to Reduce Search Space Size

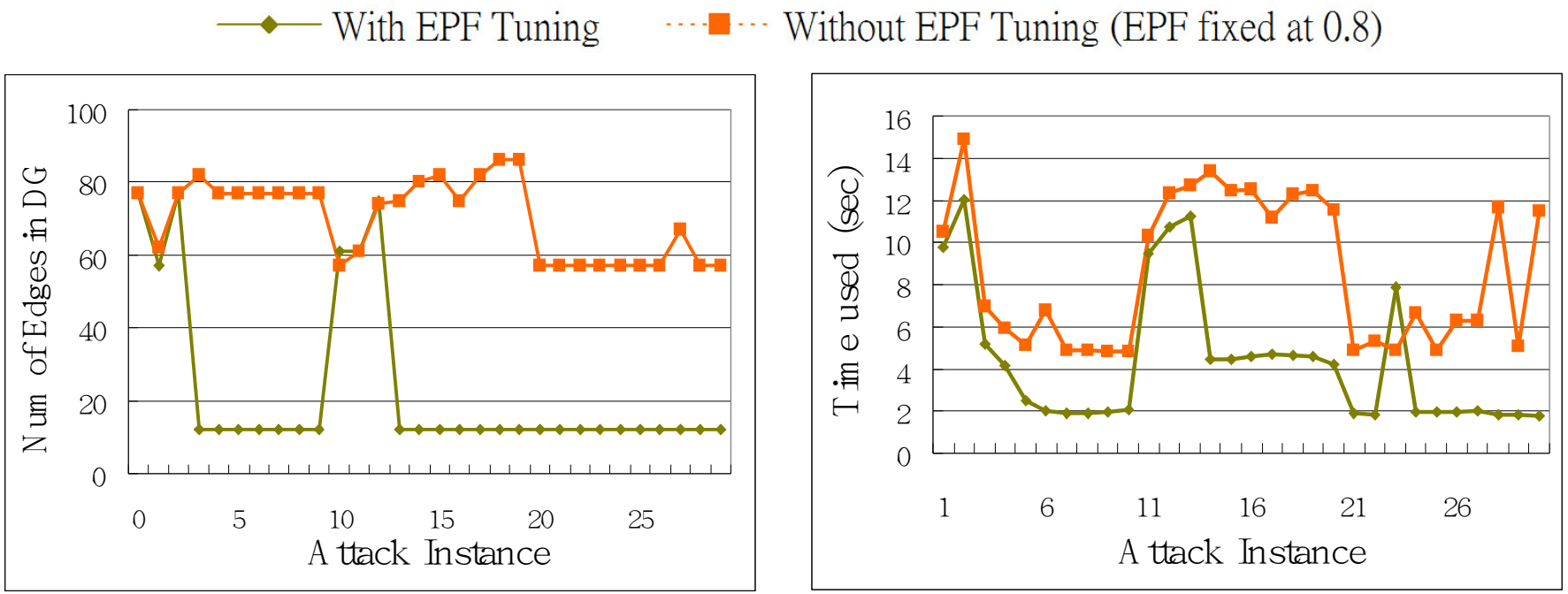
- Attack Scenarios

Attack Scenario EPFAS.1
(response available on svc3)

Attack Scenario EPFAS.2
(no response available)

- Attack Instances 0-9: EPFAS.1
- Attack Instances 10-19: EPFAS.2
- Attack Instances 20-29: EPFAS.1

Experiment 2: Learning from History to Reduce Search Space Size

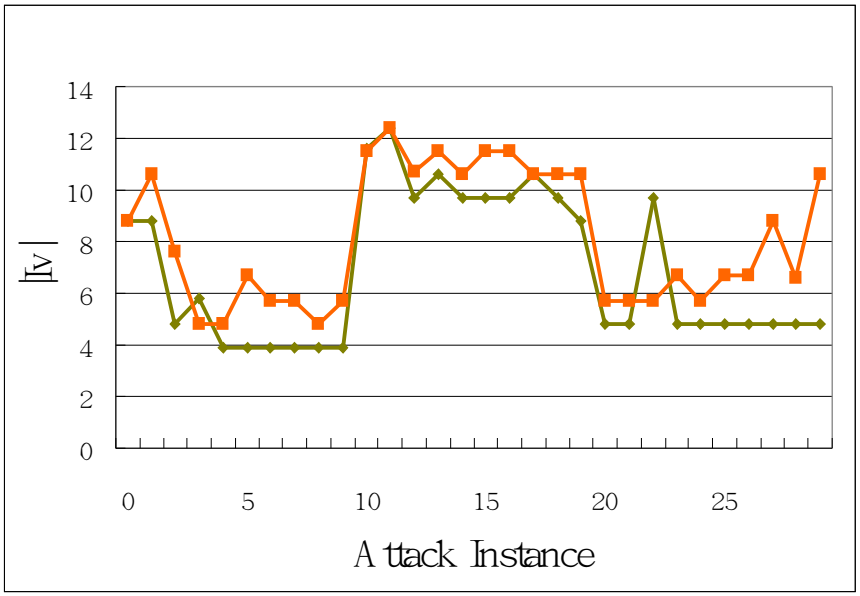


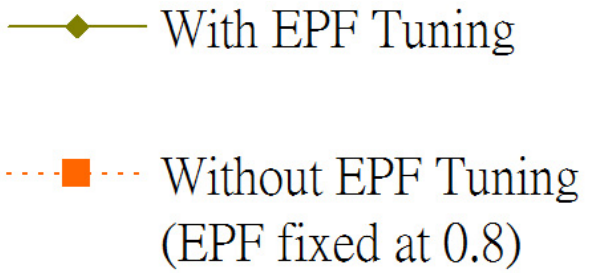
(a) # of edges in the Domain Graph generated from the 3rd snapshot

(b) Time used in response decision

- Reduced size of domain graph ▪ Faster response determination
- SWIFT adapts to reduce size of search space even with a new attack
- SWIFT prunes search space from memory of identical attack

Experiment 2: Learning from History to Reduce Search Space Size



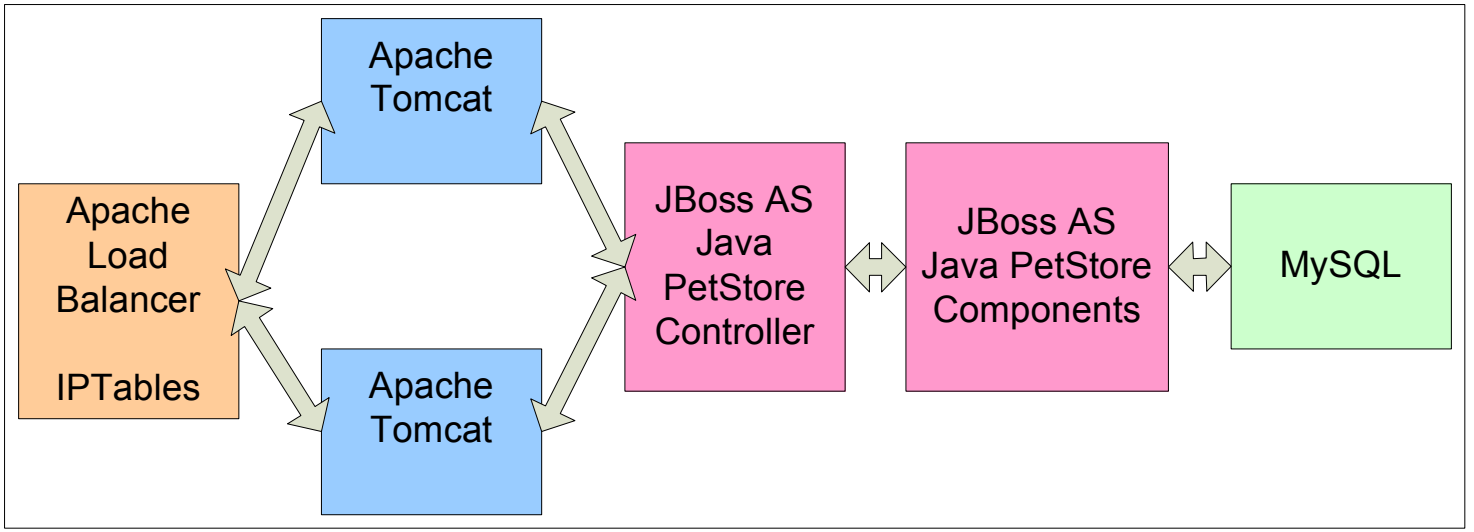


(c) |Iv| vs. Attack Instance

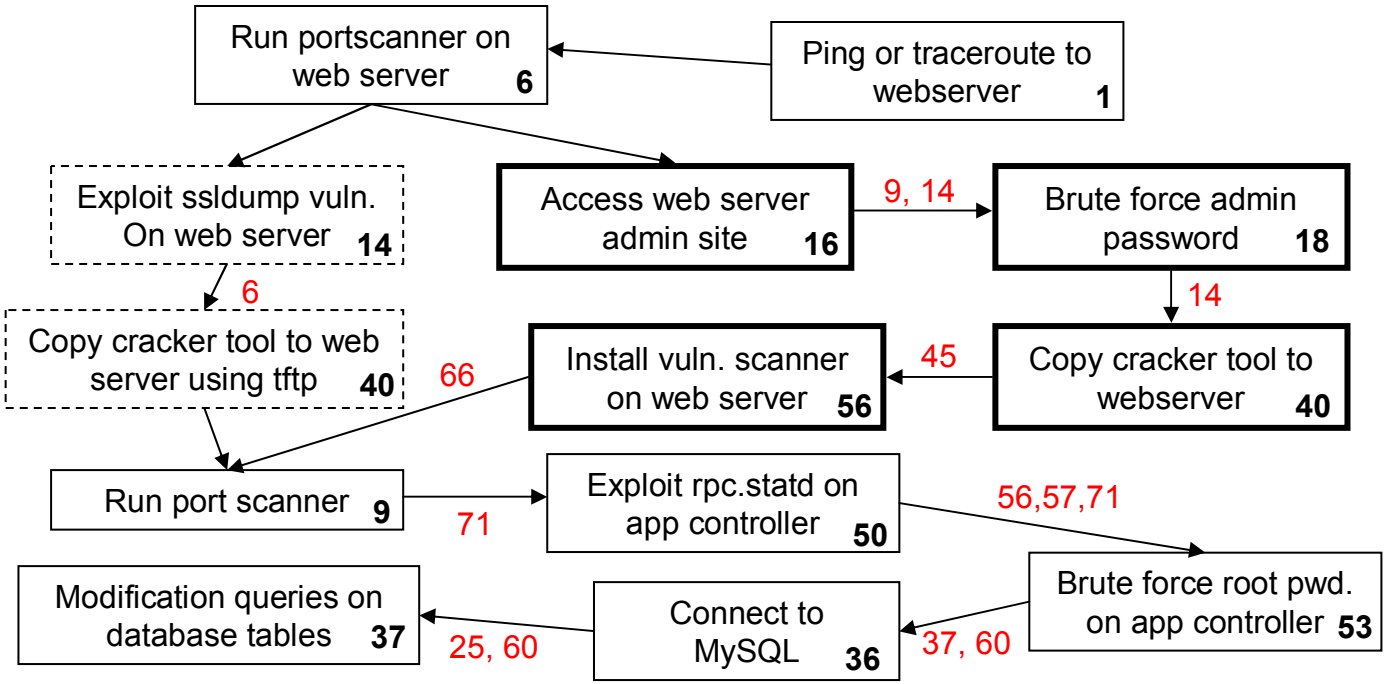
- Attack instances 10-19 have high *Iv* values since no response is available
- SWIFT uses memory of EPFAS.1 (starting attack instance 20) to pick previously used optimized responses

Experiment 3: Survivability in an e-Commerce system

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



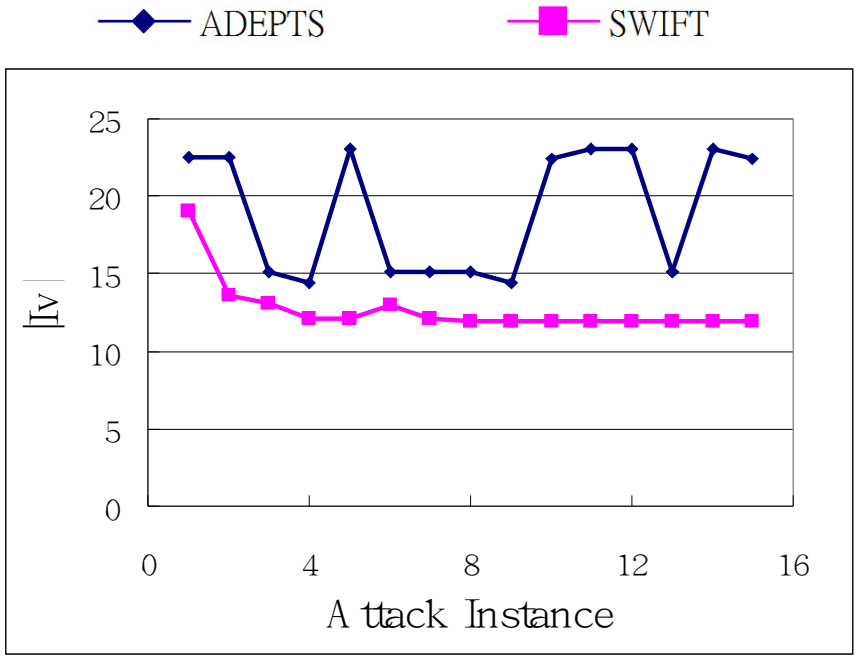
Experiment 3: Attack Scenarios



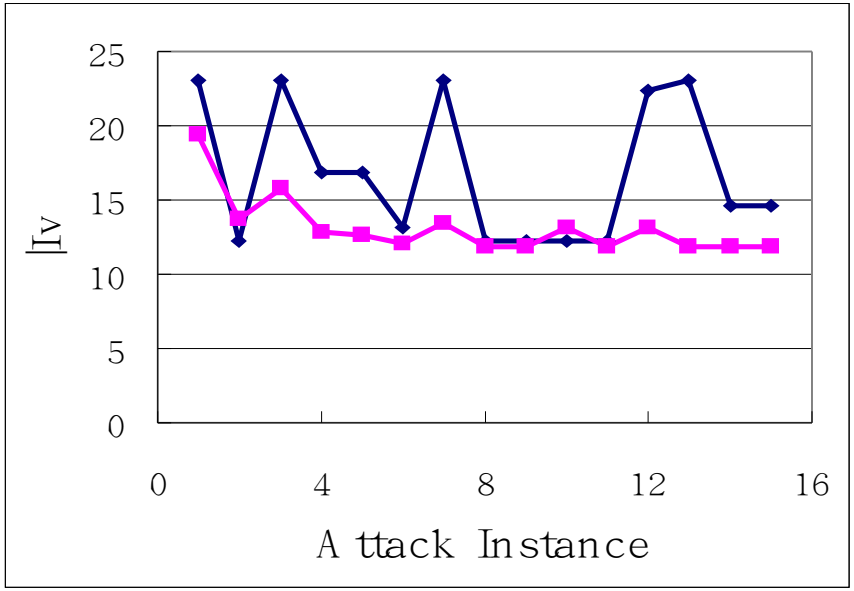
Dashed line: AS3, Thin solid line: AS3 and AS4, Thick line: AS4

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.

Experiment 3: Survivability for an example attack in a e-Commerce system



AS3

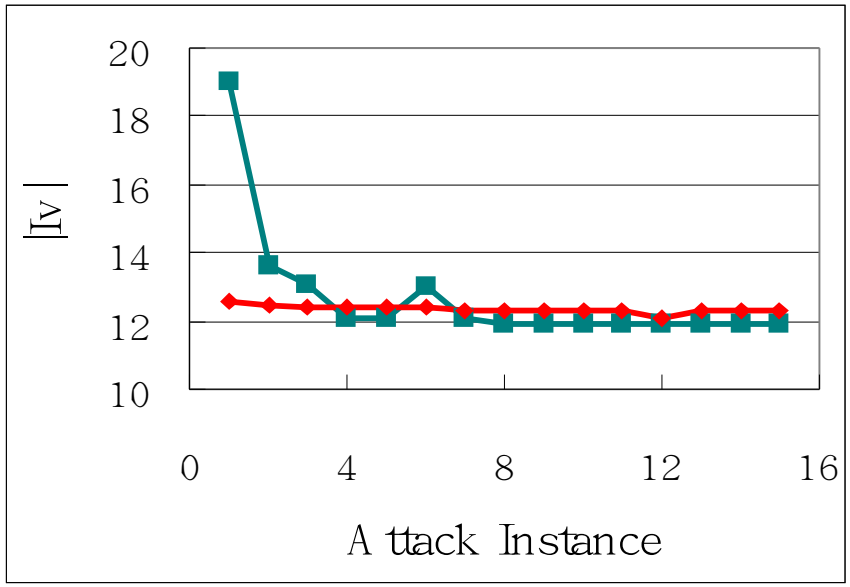


AS4

- SWIFT has consistently lower Iv than baseline
- For AS3, baseline performance is wildly fluctuating since it deploys responses close to nodes that are achieved
  - Such responses can fail or succeed
- For AS4, the performance of SWIFT and baseline are closer
  - There are more local responses available

Experiment 3: Responding to Attack Variants

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



- 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

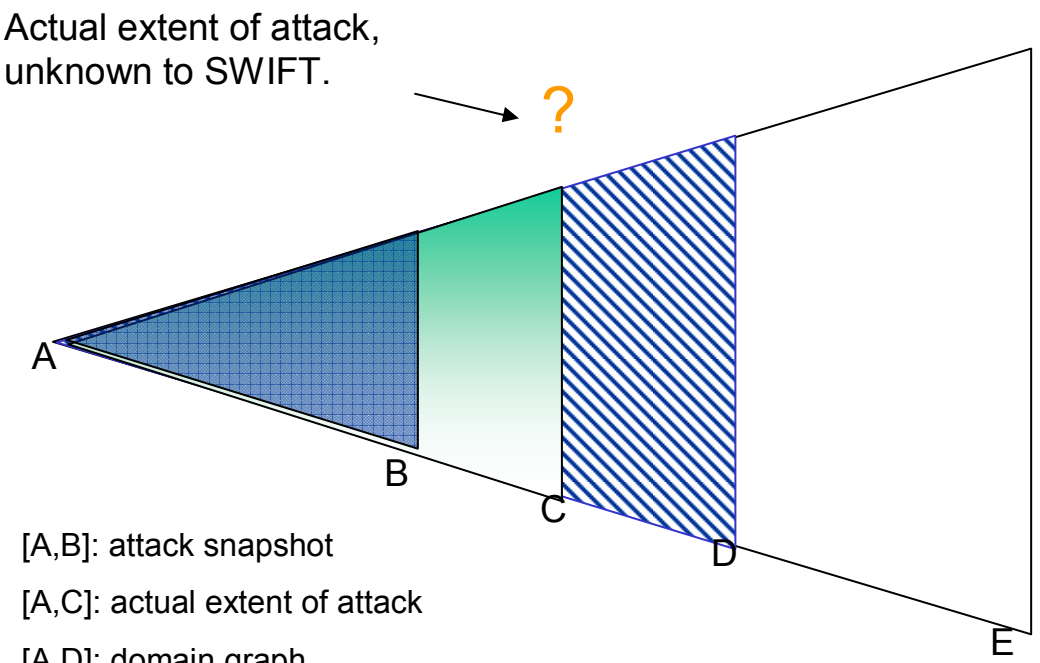
Conclusion

- Defined a framework to reason about optimality of intrusion responses in distributed systems
- The framework is implemented using genetic algorithm-based solver since exact solution is NP-hard
- Solution is made to learn from similar attacks
- Experiments with example multi-stage attacks indicate that globally optimizing response choices is beneficial

Further Work

- In our experiments, it takes only about 4-5 attack instances for SWIFT to adequately adapt to an attack
  - This number can be reduced if history information about attacks can be shared across systems
- The construction of I-GRAPH from vulnerabilities / system-configuration is a challenge
  - Most systems do not have formal specifications
- Real world / third-party dataset for multi-stage attack is scarce but we need to collect them to validate our approach

Domain Graph



[A,B]: attack snapshot

[A,C]: actual extent of attack

[A,D]: domain graph

[A,E]: I-Graph

Survivable Systems and Intrusion Response

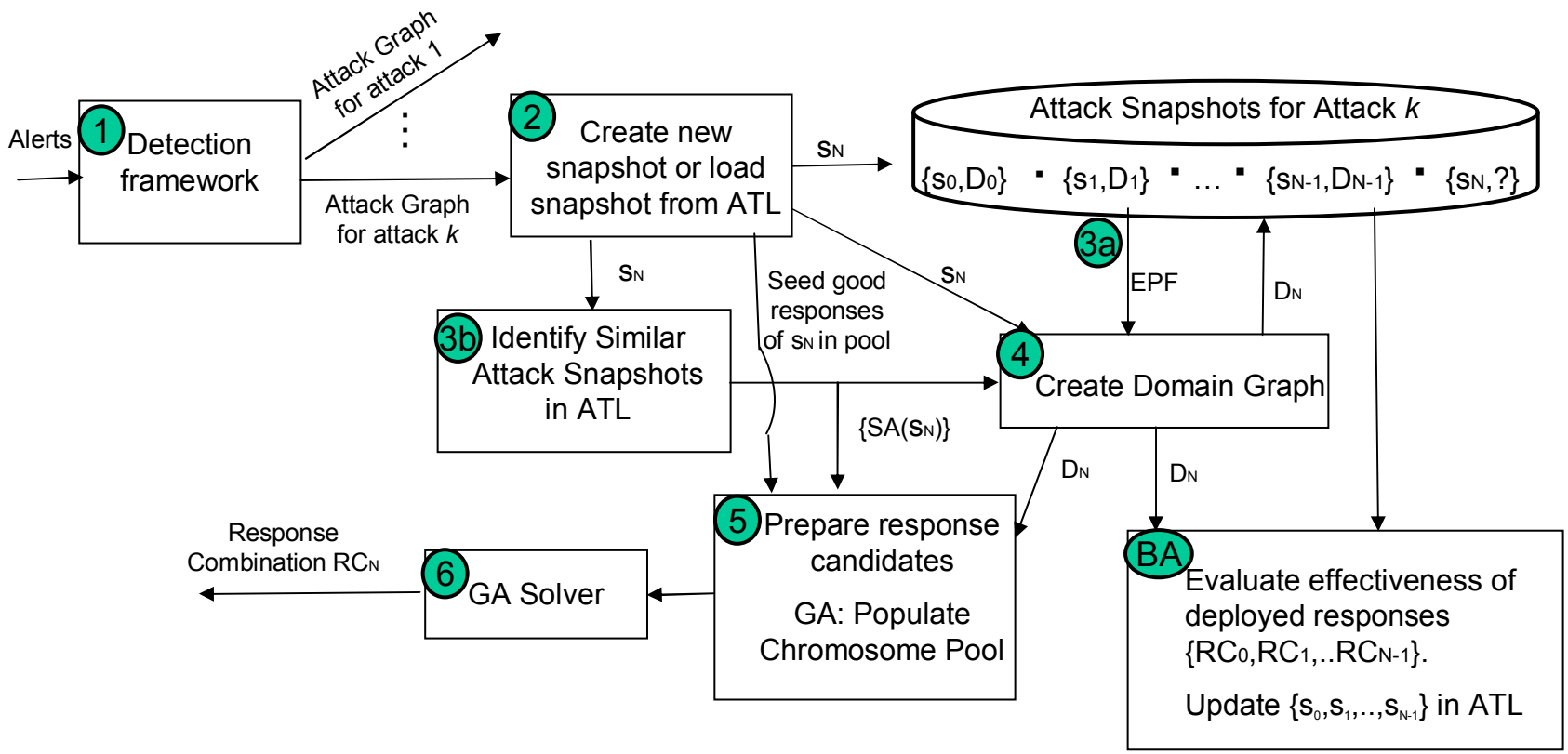
- Modern life heavily depends on computer systems
- Intrusions/security breaches to these systems occur
- Ways to make a system survivable
  - At design/implementation phase
    - Eliminate vulnerabilities
    - Policy/Access Control/Cryptography/Formal Verification
  - In production phase
    - Use IDS (system logs checking/network packet sniffing/virus, worms scanning, detecting files modifications…) to identify misuses/anomalies
    - Perform incident/intrusion response (IRS) to detected misuses/anomalies
      - Containment and Recovery

Focus of this work

I-GRAPH Structure

- The nodes are of four types
  - N_A: Specific attack manifestation – carries specific attack signature
  - N_B: Generic attack manifestation – not tied to specific attack signature, but coupled with a detector
  - N_C: High level parametrized manifestation – no corresponding detector alert
  - N_D: Intermediate nodes used for providing AND/OR/Quorum logic in the I-GRAPH
- Edges are directed
  - Edge from n_1 to n_2 indicates n_1 must causally precede n_2

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)

Experiment 1: Survivability for Micro-Benchmark

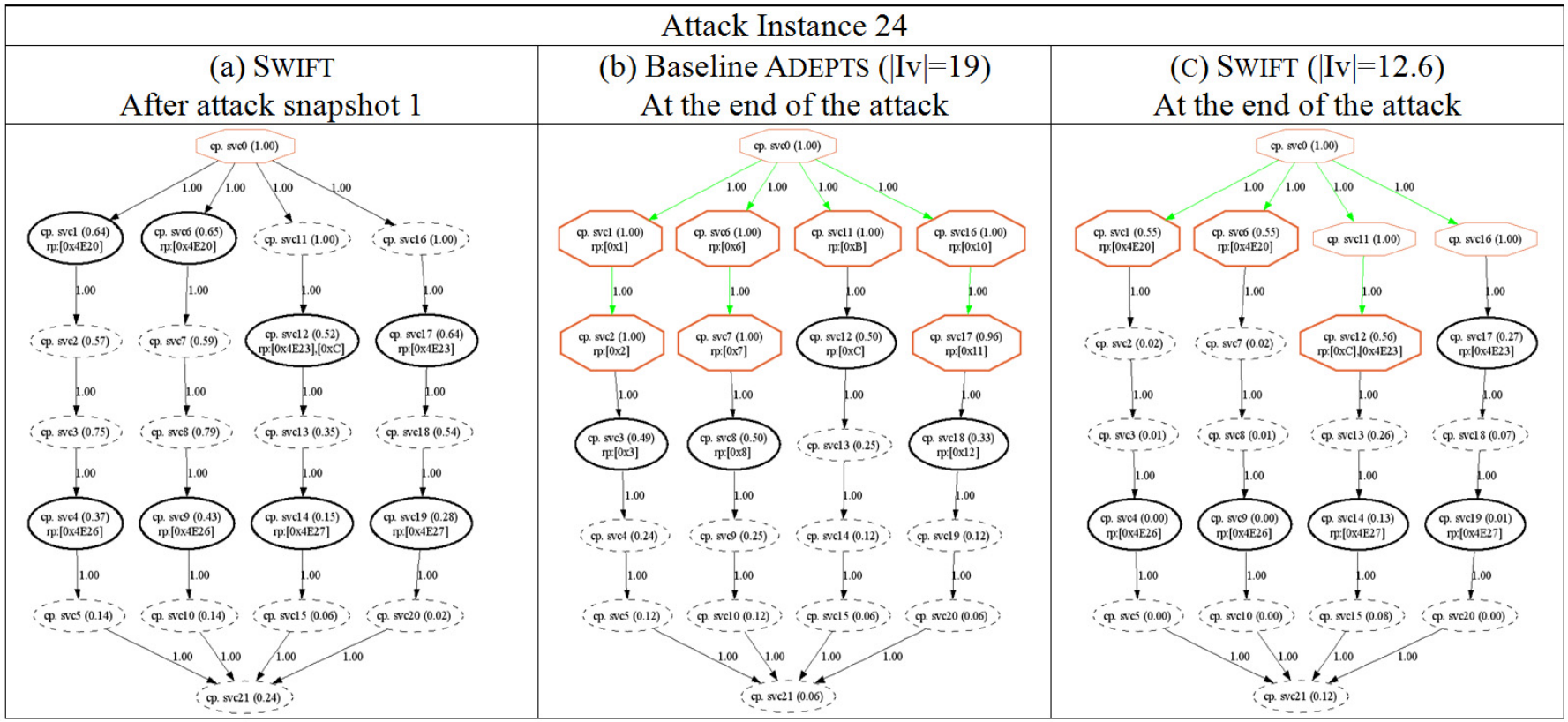


Table 1. Detailed attack snapshots from attack instance 24