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

Bingrui Foo, Yu-Sung Wu, Yu-Chun Mao, Saurabh Bagchi, Eugene Spafford[+]
School of Electrical and Computer Engineering *[+]School of Computer Sciences*
Purdue University *Purdue University*
{foob, yswu, maoy, sbagchi, spaf}@purdue.edu

Abstract

Distributed systems with multiple interacting services, especially e-commerce systems, are suitable targets for malicious attacks because of the potential financial impact. Compared to intrusion detection, automated response has received relatively less attention. In this paper, we present the design of automated response mechanisms in an intrusion tolerant system called ADEPTS. Our focus is on enforcing containment in the system, thus localizing the intrusion and allowing the system to provide service, albeit degraded. ADEPTS uses a graph of intrusion goals, called I-GRAPH, as the underlying representation in the system. In response to alerts from an intrusion detection framework, ADEPTS executes algorithms to determine the spread of the intrusion and the appropriate responses to deploy. A feedback mechanism evaluates the success of a deployed response and uses that in guiding future choices. ADEPTS is demonstrated on a distributed e-commerce system and evaluated using a survivability metric.

Keywords: *automated intrusion response, intrusion containment, distributed e-commerce systems, survivability, attack graphs.*

1. Introduction

Distributed systems comprising multiple services interacting among themselves to provide end-user functions are becoming an increasingly important platform for business-to-business (B2B) and business-to-consumer (B2C) systems. As an example, electronic commerce, or e-commerce, has been touted as the next wave in the Internet revolution. The huge financial stakes involved in e-commerce make the distributed system infrastructure supporting e-commerce prime candidates for computer security attacks.

Such motivations have long led to interest in securing distributed systems through detection of intrusions. This is typically achieved by analyzing the signatures of incoming packets and either matching them against known attack patterns (misuse-based signatures), or against patterns of expected system behavior (anomaly-based signatures). In order to meet the challenges of always-on, on-demand service availability, an e-commerce system needs to be resilient to security attacks. Resilience must include both intrusion detection and intrusion response. Compared to the problem of detection, automated response has received far less research attention. This has typically been considered the domain of system administrators who manually "patch" the system in response to detected attacks. However, as networked e-commerce services become ubiquitous and they are often placed in environments difficult to reach for human intervention, automated tools for intrusion response gain importance.

The rudimentary response mechanisms often bundled with anti-virus or intrusion detection system (IDS) products overwhelmingly consider only immediate local responses that are directly suggested by the detected symptom. For example, a file being infected with a virus may cause the anti-virus product to quarantine the file and disable all access to the file, or a suspect packet being flagged by a network IDS may cause the specific network connection to be terminated. While these may be applicable in stand-alone systems, they do not account for interaction effects among multiple components. The few available dedicated intrusion response systems are found to be lacking in one or more dimensions that make them unsuitable for protecting dynamic and complex distributed systems. Some of the commonly observed shortcomings are the system may have a static mapping of symptoms from the detector to the response, may not take feedback into account for determining future responses, may assume perfect detectors with no missed and no false alarms, or may assume perfect success rate for a deployed response. The complex interactions and the complex software running the distributed applications, the non-determinism in the execution environment, and the reality of new forms of intrusions surfacing would make any one of the above shortcomings fatal for an intrusion response system for a distributed enterprise.

In this paper, we focus on one of the most important kinds of automated response, namely, *containment*. Containment implies restricting the effect of the intrusion to a subset of the entire set of services, which may allow users access to limited functionality of the system. For example, browsing a store catalog and checking on a previously placed order may be available, while placing new orders may not be. There are several challenges to the problem of containment. First, the systems often have close coupling between the services with frequent interactions of different kinds, such as read, write, and execute. This allows a compromised service to spread the effect to multiple services. A second challenge is that the existing interactions between e-commerce system components should not be substantially altered during normal execution in order to enforce containment during periods of intrusion. Examples of unacceptable change may include mandating interactions pass through additional checks inlined in the execution path, intermediaries, or be executed over slower channels. Third, the system will have to consider the possibility of imperfect detectors providing false alarms or missing alarms, and imperfect response actions, which do not have 100% coverage.

In this paper, we present the design and implementation of an Adaptive Intrusion Tolerant System, *ADEPTS*, for containing intrusions in a distributed system of interacting services. ADEPTS uses an **I**ntrusion-**G**raph (*I-GRAPH*) to represent paths for the intrusion to spread from one service to its neighbor. Alarms from a detection system, which may be off-the-shelf or from our previous work [1], are mapped to the I-GRAPH nodes. ADEPTS estimates the likely path of spread of the intrusion from the alarms and the structure of the I-GRAPH and then determines the appropriate response(s) to take. This decision is based on the disruptivity of the response to legitimate system activities, the previous success of the response, and the confidence that the determined intrusion is indeed taking place. The response has the goal of preventing the escalation of the intrusion and possible spread from one service to another. ADEPTS can function in multiple levels of "paranoia" depending on the policy level, from an aggressive mode with an elevated threat perception to a conservative mode.

The metric used to evaluate an intrusion tolerant system has to be carefully chosen. Low-level metrics, such as the latency of detection or false and missed alarm rates do not fully capture the effect of an intrusion on the system's functionality. We propose the use of the metric called *survivability* [2] for evaluating the effect of ADEPTS. We define it such that its value depends on the set of high-level system transactions that can be achieved and the set of high-level system goals (e.g., keep users' private information secure) that are *not* violated in the event of an intrusion. A high level transaction relies on certain chains of interactions between multiple services being functioning. Preserving a high level goal implies thwarting certain intrusion goals from being reached.

The design of ADEPTS is realized in an implementation which provides intrusion response service to a realistic distributed e-commerce system. The e-commerce system mimics an online book store system and two auxiliary systems for the warehouse and the bank. Real attack scenarios are injected into the system and ADEPTS' responses are deployed, which bring out the latency of the response action and the adaptive nature of ADEPTS. The survivability of the system is compared with no response mechanism, local responses only, and with ADEPTS.

We believe this paper breaks new ground in the following ways:

1. ADEPTS is the first system, to the best of our knowledge, that provides a structured methodology for containing intrusions in a distributed system. It is also the first system to aggregate the factors of severity of a response, its effectiveness, and the possibility of escalation to determine the appropriate set of responses.
2. ADEPTS can handle multiple concurrent alerts, imperfect detectors, and escalation due to failed response actions. It can also deal with unanticipated alerts and unknown vulnerabilities in the system components. Each of these is of critical importance in an intrusion tolerance system applied to a real-world system.
3. ADEPTS is demonstrated on a realistic distributed testbed with realistic transactions and attack scenarios.

However, the work presented here *does not* have as its goal any of the following: intrusion detection for an e-commerce system, provide a methodology for structuring or composing an e-commerce system, design novel response actions for specific services in an e-commerce system, or provide a shrink-wrapped intrusion tolerance system to make an e-commerce system resilient to specific classes of attacks.

The rest of the paper is organized as follows. Section 2 refers to related research. Section 3 presents the design of ADEPTS. Section 4 describes the implementation and the e-commerce testbed on which ADEPTS is deployed. Section 5 presents the experiments and the results. Section 6 concludes the paper with mention of some future work.

2. Related Research

The devastating impact of computer security attacks to today's electronic world has spurred enormous interest in intrusion detection research, both from academic and commercial quarters. In order to guarantee the requirement for continuous availability of the services, it is also important to consider how the system reacts once the intrusion is detected. The majority of current IDSs stops with flagging alarms and relies on manual response by the security administrator or system administrator. This results in delays between the detection of the intrusion and the response which may range from minutes to months. Cohen showed using simulated attack scenarios that given a ten hour delay from detection, 80% of the attacks succeed and given thirty hours, almost all the attacks succeed irrespective of the skill level of the defending system's administrator [3]. This insight has led to research in survivable systems engineering pioneered by CERT at CMU. Survivability is loosely defined as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents ([2],[4]). The researchers identify the four key properties of survivable systems, namely, resistance to attacks, recognition of attacks and damage, recovery of essential and full services after attack, and adaptation and evolution to reduce effectiveness of future attacks. The part of the ADEPTS system presented in this paper is motivated by the requirement to provide the second and the fourth properties.

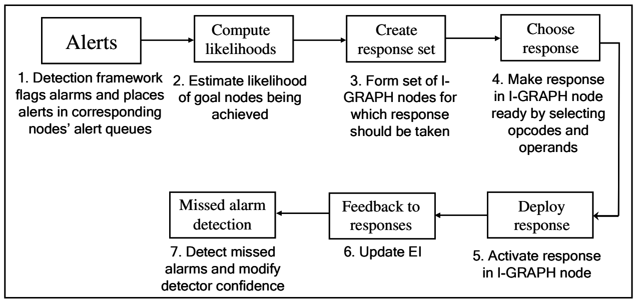
Intrusion response systems (IRS) can be considered to cover the last three properties and are therefore suitable for comparison with ADEPTS. A majority of the IRSs are *static* in nature in that they provide a set of preprogrammed responses that the administrator can choose from in initiating a response. This may reduce the time gap between detection and response, but still leaves a potentially unbounded window of vulnerability. The holy grail is an IRS that can respond to an attack automatically. A handful of systems provide adaptive responses. In [5], the authors propose a network model that allows an IRS to evaluate the effect of a response on the network services. The system chooses in a *greedy* manner the response that minimizes the penalty. There are some studies which present taxonomy of offensive and defensive responses to aid in selection of coherent responses in an automated response system ([6],[7],[8]). Cooperating Security Managers (CSM) [9] is a distributed and host-based intrusion detection and response system. CSM proactively detects intrusions and reactively responds to them using the Fisch DC&A taxonomy [6]. It uses the suspicion level of the user as the only determining factor in the choice of response. A second system called EMERALD [10] uses two factors in determining the response – the amount of evidence furnished to support the claim of intrusion and the severity of the response. None of the systems uses record of the past performance of the intrusion detection system as measured by the incidence of false positives and false negatives. None keeps track of the success or failure of the deployed response nor provide a framework for easily incorporating these factors in the automated response determination. Another adaptive IRS is the Adaptive, Agent-based Intrusion Response System (AAIRS) [11]. The work provides a good framework on which the IRS can be built. However, it does not provide any of the system-level techniques and algorithms that will be required for the AAIRS to work in practice. There is some previous work on protecting distributed systems against flooding based distributed denial of service (DDoS) attacks in an automated manner through rate limiting ([12],[13]).

Fault trees have been used extensively in root cause analysis in fault tolerant systems. They have also been used to a limited extent in secure system design ([14],[15]). We use an attack graph representation with nodes as intermediate goals since the same intermediate goals show up in several attack paths. Graph theoretic approaches to modeling the temporal nature of security attributes is found in [16],[17]. The notion of privilege graphs introduced in [17] has some similarity to our I-GRAPH. However, they represent only attacks launched by escalating the privilege level of the attacker and the arcs are marked with weights representing the difficulty of the privilege escalation. The weights are dependent on several factors, such as the expertise and resources of the attacker, and are therefore difficult to predict.

3. Design of ADEPTS

3.1. Overview

The goal of ADEPTS is to monitor and track intrusions as they occur in real-time and deploy various wide-ranging responses to contain and restrict the spread of attacks in the system. The system is subdivided into ADEPTS and the *payload* system, which includes the embedded detectors. The deployment of ADEPTS requires no modification to the payload and no access to its source code. The I-GRAPH models the paths an attacker can traverse to reach certain goals that adversely affect the payload. Our motive in designing ADEPTS is to proactively prevent an attacker from moving from one attack goal node to another, by responding appropriately at specific nodes. Here we give a high-level description of the process flow shown in Figure 1. As an alert comes into ADEPTS, it is mapped to nodes in the I-GRAPH followed by the execution of the response determination algorithm.



Alerts
Compute likelihoods
Create response set
Choose response
1. Detection framework flags alarms and places alerts in corresponding nodes' alert queues
2. Estimate likelihood of goal nodes being achieved
3. Form set of I-GRAPH nodes for which response should be taken
4. Make response in I-GRAPH node ready by selecting opcodes and operands
Missed alarm detection
Feedback to responses
Deploy response
7. Detect missed alarms and modify detector confidence
6. Update EI
5. Activate response in I-GRAPH node

Figure 1. Overall ADEPTS process flow

Throughout ADEPTS, three policy levels are used to control the behavior of the relevant algorithms − aggressive, moderate, and conservative. The three policies can be abstracted to represent a ratio of missed alarms to false alarms, with the aggressive policy having the lowest ratio and the conservative policy having the highest ratio.

3.2. I-GRAPH structure

The I-GRAPH is used as the underlying representation for knowledge about intrusions, as they spread achieving progressively wider sets of goals.

In the I-GRAPH representation, each intrusion goal is represented by one node in the graph. The final goal of the intrusion may be disrupting some high level system functionality, such as "Denial of service achieved against the online store". This final step will be achieved through multiple small to moderate sized steps. A successful execution of a step is looked upon as achieving an intermediate intrusion goal and captured as an I-GRAPH node. The intrusion goals have dependency relationships between one another. For example, in order to corrupt the data in the backend database server, one may need to exploit a vulnerability in the front-end web server. The edges are used to model this kind of dependency.

The *parents* of a node are the nodes reached by the outgoing edges of the node. They correspond to higher goals relative to the goal of the node. The *children* of a node are the nodes with outgoing edges to the node. They correspond to lower goals relative to the goal of the node.

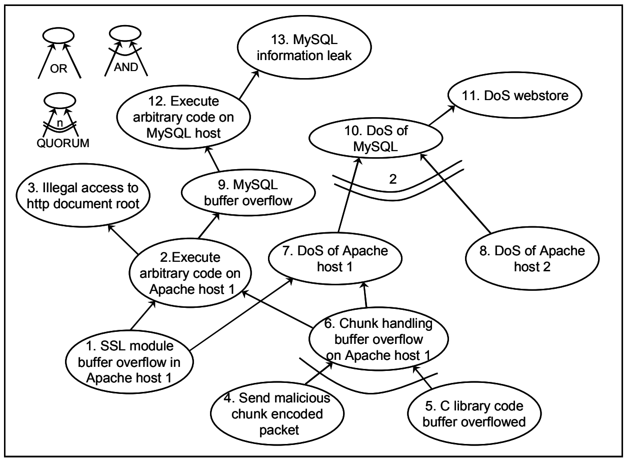

OR
AND
QUORUM
n
13. MySQL information leak
12. Execute arbitrary code on MySQL host
11. DoS webstore
10. DoS of MySQL
2
3. Illegal access to http document root
9. MySQL buffer overflow
2.Execute arbitrary code on Apache host 1
7. DoS of Apache host 1
8. DoS of Apache host 2
6. Chunk handling buffer overflow 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

Figure 2. A section of the I-GRAPH

In the I-GRAPH, edges are categorized into three types – OR, AND, and Quorum edges. For a node with incoming OR edges to be achieved, at least one of its child nodes needs to be achieved, while for AND edges, all the child nodes have to be achieved. For Quorum edges, one can assign a Minimum Required Quorum (MRQ) on it, which represents the minimum number of child nodes whose goals need to be achieved in order for the node with incoming Quorum edges to be achieved. Conforming to the traditional definition of quorums in fault tolerant systems, one may think MRQ as the minimum number of service replicas whose loss will affect the functionality of the service. An example fragment of the I-GRAPH used in our payload system, a distributed e-commerce system, is shown in Figure 2.

3.3. I-GRAPH generation

A key issue in the usability of ADEPTS is the ease with which the I-GRAPH can be generated and updated as system configuration changes or new vulnerabilities are brought to light. We employ a semi-automated method called Portable I-GRAPH Generation (PIG) for this. PIG requires two inputs – vulnerability descriptions and system services description (SNet). Of the two inputs, the SNet is target system dependent. This is a directed graph, in which each node represents an individual service in the target system and an edge from node A to node B represents an *intrusion-centric channel*. An intrusion-centric channel means if A is compromised, then the intrusion can spread to B through the channel. An intrusion-centric channel may be of five kinds – (i) DOS channel: if the source service is subjected to a successful DoS attack, then the destination service can also be subjected to DoS; (ii) Network channel: there is a network data connection between A and B; (iii) Shared file channel; (iv) Shared memory channel; (v) Super channel: which combines the functionality of all of the above. The SNet is currently manually created for the target system, though in the future, some tool which can perform service discovery and interaction discovery (each an area of current research [18]) can perform this task automatically.

The second input to PIG is the target independent vulnerability descriptions. Information on the vulnerabilities can be obtained by querying the common vulnerability databases, such as CERT, Bugtraq, and CERIAS-VDB. For use in PIG, the vulnerability is specified through four fields – (i) *Name*: which is primarily useful for human reference. (ii) *Affected service*: which gives the service(s) in the SNet affected by the vulnerability; (iii) *Manifestation*: this is a Boolean expression in disjunctive normal form composed of five elementary manifestations, namely, leaking of information, execution of arbitrary code, incorrect behavior of service, DoS, and service termination. (iv) *Dependent vulnerability and services*: which denotes the dependence on other vulnerabilities and services that have to be compromised to exploit this vulnerability. The vulnerability definitions are analogous to the virus definitions used in anti-virus products. They can be developed either by the ADEPTS developer or by a third party. The basic idea behind the I-GRAPH generation algorithm is that when a vulnerability description is read in, a corresponding node in the I-GRAPH is created, thus creating a one-to-one map. In the next step, the algorithm checks for nodes in the I-GRAPH that this newly created node can get connected to. For this step, it relies on information from both the SNet and the vulnerability descriptions to decide whether spread of the intrusion is possible from the newly created node to the other nodes and vice-versa.

It is noted that though temporary countermeasures are usually provided (before patches appear) in some vulnerability databases (e.g. CERT), they typically require disabling affected services. Without ADEPTS, disabling affected services when a vulnerability is discovered can be quite disruptive; but with ADEPTS, the advantage is that the decision is made at runtime based on system status, and can be configured to be aggressive or conservative (with respect to disabling the services). Also some detector rules are due to anomalous behavior observed in the system and not directly attributable to a vulnerability, and therefore no patches are applicable.

3.4. Determining response locations

3.4.1. CCI computation algorithm. The goal of the algorithm is to determine, based on the received alerts from the detectors, which of the I-GRAPH goal nodes are likely to have been achieved. Each detector provides confidence values for its alerts, termed alert confidence. If the detector does not provide an inbuilt confidence value with the alert, then the alert confidence value is set to one. When a detector flags an intrusion, the alerts are placed in the I-GRAPH nodes with the corresponding intrusion event. The Compromised Confidence Index (CCI) of a node in the I-GRAPH is a measure of the likelihood that the node has been achieved. It is computed based on the alert confidence corresponding to the node and the CCI of its immediate children nodes. Mathematically, the CCI of a node is given by

$$CCI = \begin{cases} alert\ confidence, & no\ children \\ f'(CCI_i), & no\ detectors \\ f(f'(CCI_i, alert\ confidence), & else \end{cases}$$

$$f' = \begin{cases} max(CCI_i), & OR\ edge \\ min(CCI_i), & AND\ edge \\ \begin{cases} mean(CCI_i \mid CCI_i > \tau), & Quorum\ met \\ 0, & Quorum\ not\ met \end{cases} & Quorum\ edge \end{cases}$$

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

The intuition is that for an OR edge, the node can be achieved if any of its children nodes is achieved and therefore the likelihood (due to its children) is the maximum of all of its children. For an AND edge, all the children nodes have to be achieved and therefore the likelihood is as much as the least likely child node. For Quorum edges, if the quorum is not met, then the higher goal is *not* achieved, but if met, the likelihood of it being achieved only depends on the children nodes that have achieved the quorum. The function *f* allows various weights to be assigned to determine the relative effect of the alert confidence and the children's CCI. The function for the current design is the statistical mean.

When new alerts arrive, the nodes corresponding to these alerts are reordered within a fixed time window and passed to this algorithm. The I-GRAPH is traversed in breadth-first-search (BFS) order starting from the lowest nodes with new alerts, and the CCIs of the nodes are computed until each reachable node has been traversed at most once. This prevents infinite cycling to occur even though there may be cycles in the I-GRAPH. The disadvantage of such a traversal is that some causal relation between nodes may be lost. However, the alerts are usually temporally ordered according to the order in which the events occurred, thus the causal order is more likely to be obeyed in the CCI computation. Since the CCI of a parent node is dependent on that of its child nodes, a BFS traversal starting from the lowest node with an alert, rather than DFS, is justified.

The alert confidence used to update the CCI is chosen based on policy. For an aggressive policy, the maximum alert confidence in the alert queue is used; for a moderate policy, the maximum of a subset of alert confidences based on the most recent alerts is chosen; for a conservative policy, the alert confidence corresponding to the most recent alert is chosen. The alert confidence provided by a detector has to be moderated by the confidence on the detector. ADEPTS has a mechanism to determine if a detector misses alarms and adjust the detector confidence accordingly. Qualitatively, if ADEPTS sees that for a given node, its children nodes as well as parent nodes are flagged but it is not, then it anticipates probabilistically that the detectors have missed flagging the alert.

3.4.2. Response set computation algorithm. The purpose of this algorithm is to determine the nodes where the current attack will most likely spread to. This will allow the response algorithm to deploy appropriate responses at those locations. The I-GRAPH is traversed in reverse order of the CCI computation algorithm, continuing until all reachable nodes are traversed at most once. During the traversal, each node is labeled as one of: (i) Strong Candidate (SC), if CCI $> \tau$; (ii) Weak Candidate (WC), if CCI $\leq \tau$ but further traversal across only AND edges can reach a SC node; (iii) Very Weak Candidate (VWC), if CCI $\leq \tau$ but further traversal across any type of edge can reach a SC node; (iv) Non-Candidate (NC), otherwise. If the CCI of a node is computed to be greater than τ, the system concludes the node has been achieved, where τ is a deployment parameter. Therefore the SC label on a node is a strong indicator that the node has been achieved, while the WC or VWC label indicates smaller likelihoods due to evidence from their parents.

Next, some nodes are placed in a *response set*, indicating to the response system where responses should be deployed. For an aggressive policy, all SC nodes, and WC and VWC nodes which have at least one immediate NC parent node are placed in the response set. For a moderate policy, all SC and WC nodes that have at least one immediate NC parent node are chosen. For a conservative policy, all SC nodes that have at least one immediate NC parent node are chosen. The aggressive, moderate, and conservative policies provide increasingly less disruption as well as less protection.

3.5. Response Repository

The deployment of the response is achieved by a *Response Repository*, a *Response Control Center*, and distributed *Response Execution Agents*. The Response Repository stores the responses available for deployment in a payload system. Each response in the repository consists of an opcode and one or more operands, with wildcards allowed for each. The opcode is the response command, and the operands are the different parameters that need to be specified in order to execute the response. For example, the opcode for the response command of dropping incoming packets from a remote IP to a local port is DROP_INPUT, and the corresponding operands are REMOTE_IP and LOCAL_PORT. The opcode and the operands together make up a complete response command. The response structure allows ADEPTS fine-grained customization of the available responses

3.6. Response Control Center

The opcode is selected based on the ability of the opcode to cut off the *attack-centric channels* as defined in Section 3.3. The Response set computation algorithm (Section 3.4.2.) sends to the Response Control Center the list of I-GRAPH nodes which are candidates for the deployment of responses. For each node, the Response Control Center selects a set of candidate response opcodes that can be used to prevent attacks from spreading via the node's outgoing intrusion-centric channels. The choice is determined by the type of the channel. For example, the file access based opcodes, such as DENY_FILE_ACCESS or DISABLE_WRITE, are selected as candidate response opcodes if an outgoing shared file channel is present.

After the opcodes have been chosen, the Response Control Center generates a list of complete response commands by collecting suitable operands. For this, it examines the alert events stored in the *alert queue* of the node and uses them to fill in the operands that are required by the selected opcodes. An opcode can be combined with multiple operands during this phase. For example, for an opcode KILL_PROCESS, the control center may extract PID#1 from alert event#1 and PID#2 from alert event #2, both in the alert queue. Then, the response command KILL_PROCESS PID#1, PID#2 is generated for subsequent evaluation.

3.6.1. Picking responses to deploy. For each selected response command, the Response Control Center computes the Response Index (RI). The RI takes into the account the estimated effectiveness of the response to the particular attack, measured by the Effectiveness Index (EI), and the perceived disruptiveness of the response to legitimate users of the system, measured by the Disruptiveness Index (DI). The EI and the DI are both specific to the response command (opcode-operand combination) and the node in the I-GRAPH to which the response is mapped. The RI is given by $RI = a.EI – b.DI$, where a and b are deployment parameters.

Note that EI of an identical response command may differ for different attacks that map to different I-GRAPH nodes. For example, blocking port 65000 or 16660 may be useful against the *stacheldraht* DDoS attack but is unlikely to be effective against the TFN DDoS attack. The two attacks can be differentiated by their packet signatures. The control center chooses the response with the highest RI among the candidate responses, with a threshold being used to suppress a response that falls below it. This ensures that ineffective or highly disruptive responses are not deployed. If no response is chosen for a particular node, then the next higher level node is searched for possible responses. A chosen response is deployed using a Response Execution Agent. When response mechanisms or Response Execution Agents on a particular compromised host have been disabled, responses will be taken at other hosts, as determined by the spread of the attack through the I-GRAPH.

In the event that the payload system is under multiple concurrent attacks, ADEPTS deploys responses for different alerts received in a short span of time, which may correspond to each individual attack instance. A heuristic to distinguish different concurrent attacks, involving clustering source IP addresses, destination IP addresses, source ports, destination ports, user accounts and initiated processes, proposed in [19], can be easily integrated into ADEPTS.

3.6.2. Contradiction, equivalence, subset, and super set relations between responses. Before initiating execution of the chosen responses, the Response Control Center identifies the relations between the active responses and the pending responses. The newly selected response is suppressed if the new command is a subset or the equivalent of an active response or the new response contradicts an active response. If there is overlap between the new response and an active response, the ideal strategy would be to deploy the non-overlapping part of the new response. Since it is difficult to extract the differences between responses in an automated manner, we enforce the design choice on the responses in ADEPTS that they be non-overlapping. This is possible to achieve because of the fine granularity of the responses.

3.6.3. Handling unknown alerts. In a real-world deployment, it is quite probable that the I-GRAPH for the payload system is incomplete. Thus, ADEPTS would be unable to map an incoming alert from a detector to a node. To handle this situation, ADEPTS has the provision of a general node per host. The alert would be mapped to the general node for the host that is the destination of the attack as it is easily deducible from the alert. In this case, the Response Control Center can simply report the instance to the administrator and take one of a set of pre-specified general responses. The general responses are the commands that would be possible to deploy with very little knowledge of the operands, such as killing a process (need process ID), shutting down a service (need service ID), or restarting a host (need host ID).

3.7. Providing feedback to responses

Feedback to the response system is crucial for ADEPTS, providing the runtime mechanism to bias response choices in favor of those that have been effective in the past. This feedback is provided by dynamically varying the EI of the response. After a response has been deployed, the feedback system checks to see if any active response action is deployed on an edge that can be used to reach a node in the currently computed response set. If such a response action exists, it is indication that the response action possibly failed and its EI is decreased.

The amount by which the EI of the response is decreased depends on whether the response is on an AND edge, OR edge, or Quorum edge to the node in the response set. If it is on an AND edge, then it is certain that the response failed and thus the node was achieved. Therefore, the EI is decreased by a fixed fraction for responses on all the edges. If the response is on an OR or Quorum edge, then the EI is decreased in the proportion of the CCI values of the nodes, the maximum decrease being the same as in the AND case. When a response's Time To Live (TTL) expires or when an administrator manually deactivates a response, the EI of the response action is increased by a fixed percentage under the intuition that the response was successful since further alerts were not observed.

Referencing Figure 2, suppose an active response is present on the edge between node 1 and 7, and node 10 is in the response set. Suppose the fixed fraction to decrease is α. Then for the active response,

$$EI_{new} = EI_{old} - \alpha \frac{CCI_7}{CCI_8 + CCI_7} \frac{CCI_1}{CCI_6 + CCI_1}.$$

4. Implementation of ADEPTS and testbed

4.2. Description of e-commerce application

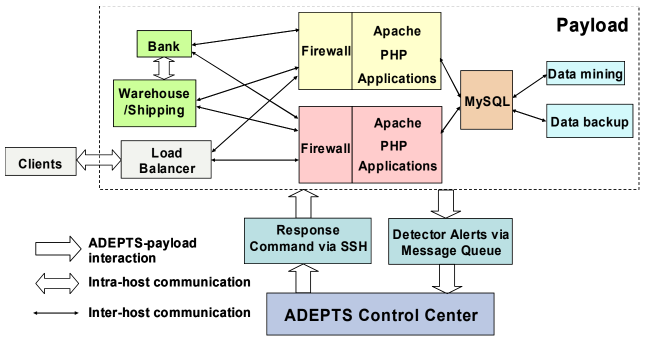


Figure 3. Layout of e-commerce testbed

Figure 3 depicts the testbed that we use for experiments. The payload system mimics an e-Commerce webstore, which has two Apache web servers running webstore applications, which are based on Cubecart (http://www.cubecart.com) and are written in the PHP scripting language. In the backend, there's a MySQL database which stores all the store's information, which includes products inventory, products description, customer accounts, and order history. There are two other organizations with which the webstore interacts – a Bank and a Warehouse. The Bank is a home-grown application which verifies credit card requests from the webstore. The Warehouse is also a home-grown application, which takes shipping requests from the webstore, checks inventory, applies charges on the customer's credit card account, and ships the product. The clients submit transactions to the webstore through a browser. Some important transactions are given in Table 1.

We set certain security goals for the system, the complement of which are specified in Table 2, along with the weights. Thus adding the word "prevent" before each gives the goal. The attached weights to the transactions and security goals are used for survivability computation in Section 5.

Table 1. List of e-commerce transactions

Name	Services involved	Weight
Browse webstore	Apache, MySQL	10
Add to shopping cart	Apache, MySQL	10
Place order	Apache, MySQL	10
Charge credit card	Warehouse, Bank	5
Admin work	Variable	10

Table 2. List of e-commerce security goals

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

4.2. Detectors

For our testbed, multiple detectors which communicate with ADEPTS through secure channels are used. We use two off-the-shelf detectors – *Snort* and *Libsafe*, and create three home-grown detectors. Snort is used for detecting intrusion patterns in network traffic while Libsafe is used to detect buffer overflows in protected C-library calls. We create a kernel-based *File Access Monitor*, which can detect file access attempts of monitored processes and compare these access attempts against preset rules to detect illegitimate activity. Also, we create a *Transaction Response Monitor*, which monitors the transaction response time of the webstore using requests from the Apache Benchmark (http://httpd.apache.org/docs-2.0/programs/ab.html). Finally, there is an *Abnormal Account Activity Detector* at the Bank, which detects abnormal account activities such as excessive number of credit card transactions on one account. The detectors used are all imperfect ones, with the possibility of missed alarms and false alarms. The detectors are not optimized for each attack scenario that the system is tested with. This is because the process is clearly labor-intensive and relies heavily on administrator expertise. For the off-the-shelf detectors, the rules are taken from the public distribution, while for the others, the rules are created by a researcher separate from the group that generates the attack scenarios.

4.3. Attack scenarios

The ADEPTS implementation is tested with different attack scenarios classified into three categories − leaking information, illegal transaction, and DoS. Each attack scenario consists of a set of attack steps, with an ultimate high-level goal. Each step of the attack scenario may be detected by none, one, or more of the detectors. A detector vector with the elements (Snort, Libsafe, File Access Monitor, Bank Monitor., Transaction Response Monitor) is assigned to each step of the attack scenario. A '1' means that step *can* be detected by the corresponding detector. We show in Table 3 one sample scenario from each category – Scenario 0 is leaking of user information in the database (Leaking information), Scenario 1 is placing unauthorized orders (Illegal transaction), and Scenario 8 is vandalize webstore (DoS).

Table 3. Attack steps for three attack scenarios

Step	Scenario 0
1	Apache ModSSL buffer overflow (10000)
2	Insert malicious code (00000)
3	IP/port scan to find vulner. SQL server (10000)
4	Buffer overflow MySQL to create a shell (00100)
5	Use shell to steal info stored in MySQL (00100)
	Scenario 1
1	Apache php_mime_split buffer overflow (10000)
2	'ls' to list webstore document root and identify code regarding warehouse shipments (00100)
3	Send shipping request to warehouse, crafting request form to cause buffer overrun to fill form with victim's credit card number (01000)
4	Make unauthorized orders (00010)
	Scenario 8
1	Buffer overflow Apache ModSSL (10000)
2	Create Apache privilege shell (00100)
3	Execute crontab command (00100)
4	Insert malicious data into Apache's crontab (00100)
5	Root privilege shell created (00000)
6	Corrupt web server document root (00100)

We also test ADEPTS with other attack scenarios involving buffer overflow attacks to steal client info, and other DoS attack scenarios entailing memory exhaustion in the Apache mime handling components or DDoS through huge number of legitimate transactions, such as product search. The entire I-GRAPH automatically generated by the PIG algorithm consists of 57 nodes and 1148 edges and is too large to be shown. A fragment of the I-GRAPH has been shown in Figure 2.

4.4. Response Repository for testbed

Four types of response commands are included in the Response Repository − *general*, *file*, *network*, and *denial-of-service types*. The *general-type* commands can be deployed to block any types of *intrusion-centric* channels in the I-GRAPH, corresponding to the super channel. The other types of commands have a one-to-one map to the kinds of intrusion channels introduced in Section 3.3. The implementation of the file-type commands is achieved by using the *Linux Intrusion Detection System* (LIDS) version 2.2.0. The implementation of the network-type commands is performed by using *iptables*. The general type commands are killing a process and restarting or shutting down a service or a host. The file-type commands are to deny any access to a file, or selectively disable read, write, or execute access. The network-type commands are to block incoming or outgoing network connections, parameterized by source or destination port, IP, or protocol. The DOS-type commands are to limit the rates of various types of packets, such as SYN, ICMP echo, ICMP host not reachable, and SYN-ACK.

5. Experiments and results

We perform three sets of experiments demonstrating the following (i) effect of attack scenarios on survivability with and without ADEPTS, (ii) the ability of ADEPTS to deploy responses as the speed of propagation of the attack varies, (iii) the adaptation in ADEPTS in choosing responses. All these experiments are conducted using the moderate policy with actual attack scenarios on the e-commerce testbed. Our experiments here show the behavior of ADEPTS under a limited number of parameter configurations. They are not meant to bring out trends in the performance of ADEPTS or provide predictability under new attack scenarios or different parameter configurations. Comparing ADEPTS to other automated IRSs mentioned in Section 2 was not possible since they are not publicly available. For experiment 1 and 2, we define the survivability based on the high level transactions and security goals in Table 1 and 2. The metric thus shows the effect of ADEPTS on the high level functioning of the e-commerce system.

Survivability = 1000 – Σ unavailable transactions – Σ failed security goals.

When a transaction becomes unavailable or the security goal is violated, the survivability drops by its corresponding weight. Transactions become unavailable due to responses, such as rebooting a host, or attacks. Security goals may be violated due to the successful execution of an attack step. If a security goal is violated multiple times during an attack, then each violation causes a decrease in the survivability.

5.1. Effect of attack scenarios on survivability

The goal of this experiment is to show the comparative performance of ADEPTS in maintaining the survivability of the e-commerce system with respect to having no responses and only local responses. Three different attack scenarios are executed and the survivability calculated at each step of the attack scenario. For local responses, the responses that came with the deployed detectors are used – Snort (IP blocking) and bank monitor (freeze credit card).

For the *leak of information* attack (Figure 4), ADEPTS far outperforms the other two. The File Access Monitor detects a malicious shell being created with Apache privileges while Snort detects an Apache SSL module buffer overflow packet. Consequently, ADEPTS deploys aggressive responses to kill the process and block all following incoming packets from the attacker. The inability of the local response implemented by Snort to drop the IP packets in time causes the attack to continue to spread. For the *illegal transaction* attack (Figure 5), the performance of the local response is noticeably worse than ADEPTS. ADEPTS deploys a successful response disallowing shell commands with Apache privileges, earlier than the local response at the bank monitor. For the *distributed denial of service* attack (Figure 5), the graph shows the inability of any of the setups to respond effectively to the attack. The responses deployed by ADEPTS to limit the overall incoming packet rate, such as, blocking packets from the IP address with the highest rate of packet transmission, allowed for a slight decrease in the effectiveness of the DoS.

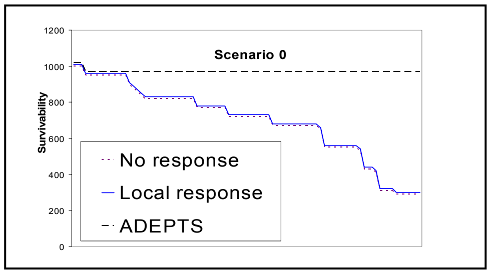


Figure 4. Survivability during scenario 0

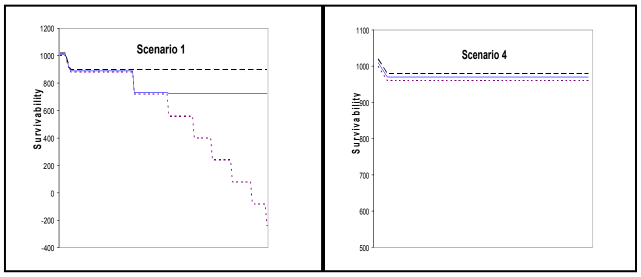


Figure 5. Survivability during scenario 1 and 4

5.2. Effect of prop. speed on survivability

This experiment inspects the relationship between the ability of ADEPTS to protect the payload system and the attack propagation speed. We vary the delay between the attack steps in scenario 0 between 0-7 s. This could simulate a variety of factors, such as the attacker's skill level, condition of the network, difficulty of an attack step, etc. Figure 6 shows the survivability with different attack propagation speeds. The legend '0004' means there is a delay of zero seconds before attack step 0 is begun, between steps 0 and 1, and 1 and 2, while there is a delay of 4 s for all subsequent steps. We see that ADEPTS performs well with delays ≥ 4 s since the attack is stopped in the very first step. If however, the first step has zero delay, then ADEPTS is only able to block the attack at a later step, leading to a decreased survivability. With no delay at all between the steps, ADEPTS is only able to block the attack at the last step, which is still better than the no response case. In all cases, ADEPTS can maintain the survivability at a constant level once the blocking is successful.

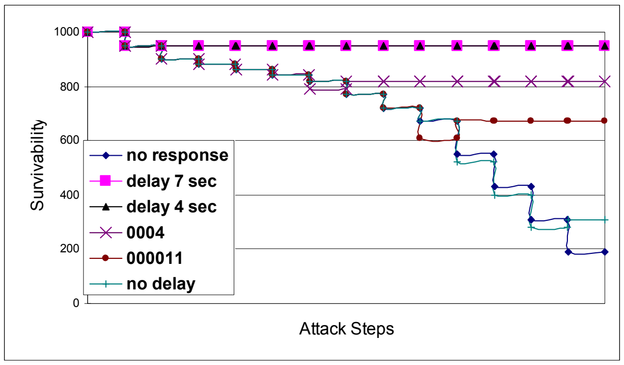


Figure 6. Effect of attack propagation speed

5.3. Adaptation of response mechanism

Table 4. Scenario 8's intrusion responses

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

Table 5. Response actions during runs

Step	Responses	EI changes
Run 1		
1		
2	X R1; X R2	
3	X R3; X R4	R1 1.1→1.072; R2 1.1→1.058
4	X R5	R2 1.058→1.024; R4 1.1→1.064 R3 1.1→1.076; R1 1.072→1.049
5		
6	**O R6** X R7 X R8	R2 1.024→0.993; R4 1.064→1.031 R5 1.1→1.077; R3 1.076→1.054 R1 1.049→1.028
Attack stopped		R8 1.1→1.21; R7 1.1→1.21 R6 1.1→1.21
Run 3		
1		
2	X R1; X R7	
3	**O R6**; X R4	R7 1.21→1.164; R1 0.96→0.936
4	X R3	R7 1.164→1.126; R4 1.1→1.064 R1 0.936→0.916; R6 1.331→1.302
Attack stopped		R3 1.010→1.111
Run 6		
1		
2	**O R6**; O R7	
Attack stopped		R7 1.1→1.21; R6 1.401→1.541

Here we demonstrate the adaptive nature of ADEPTS through which it can change the response strategy as an attack escalates. Scenario 8, having 6 steps, with delay characteristic '00015' is used for demonstrating the process. This experiment is composed of multiple runs of scenario 8 (we show runs 1, 3, and 6). We see in run 1, the initial choice of responses is poor ('X' by response implies failed) and the final step of scenario 8 is reached, where the attacker can create a root privilege shell and tamper with Apache's HTTP documents. In run 3, after a series of EI tuning from runs 1 and 2, the response of rebooting Apache is deployed in step 3. In run 6, the response of rebooting Apache is moved to step 2. The earlier the attack is blocked, the less likely the final attack goal is achieved. Thus, the choice of responses in run 6 is the best and ADEPTS is seen to improve its response choices through observing multiple attacks.

6. Conclusions

In the paper we have presented the design and implementation of an automated intrusion containment system called ADEPTS. ADEPTS uses a graph of intrusion goals called I-GRAPH. It provides a method to determine the possible path of spread of the intrusion, appropriate services where to deploy the response, and appropriately choose the response. ADEPTS is demonstrated on an e-commerce system with real attack scenarios.

We are currently investigating ways to synthesize new responses at runtime from the repository; designing protocols for better handling concurrent attacks, distinguishing between attacks, and increasing tolerance towards faulty detectors; and lastly, evaluating the convergence of its adaptation feature and systematically evaluating the performance of ADEPTS with more control parameters being varied.

References

[1] Y.-S. Wu, B. Foo, Y. Mei, and S. Bagchi, "Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS", *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC)*, Dec 8-12, 2003.

[2] R. Ellison, R. Linger, T. Longstaff, and N. Mead, "Case Study in Survivable Network System Analysis", *Technical Report CMU/SEI-98-TR-014*, SEI, CMU, 1998.

[3] F. Cohen, "Simulating Cyber Attacks, Defenses, and Consequences," Available at http://all.net/journal/ntb/simulate/simulate.html, 1999.

[4] R. Anderson, A. Hearn, and R. Hundley, "Studies of Cyberspace Security Issues and the Concept of a U.S. Minimum Essential Information Infrastructure", *Proceedings of the 1997 Information Survivability Workshop*, CERT, 1997.

[5] T. Toth and C. Kruegel, "Evaluating the Impact of Automated Intrusion Response Mechanisms", *18th Annual Computer Security Applications Conference (ACSAC)*, Dec 9-13, 2002.

[6] E. Fisch, "Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior", Ph.D. Dissertation, Texas A&M U, College Station, TX, 1996.

[7] C. Carver and U. Pooch, "An Intrusion Response Taxonomy and its Role in Automatic Intrusion Response", *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, USMA, West Point, NY, 2000.

[8] U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions", *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, Oakland, CA, May 4-7, 1997, pp. 154 - 163.

[9] G. White, E. Fisch, and U. Pooch, "Cooperating Security Managers: A Peer-based Intrusion Detection System", *IEEE Network*, vol 10, no. 1, 1996, pp. 20-23.

[10] P. Porras and P. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," *Proceedings of the 20th National Information Systems Security Conference*, Baltimore, MD, Oct 7-10, 1997, pp. 353-365.

[11] D. Ragsdale, C. Carver, J. Humphries, and U. Pooch, "Adaptation Techniques for Intrusion Detection and Intrusion Response Systems", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, Tennessee, Oct 8-11, 2000, pp. 2344-2349.

[12] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday, and T. Reid, "Autonomic Response to Distributed Denial of Service Attacks", *Proceedings of the 4th International Symposium on Rapid Advances in Intrusion Detection (RAID)*, Davis, CA, USA, Oct 2001.

[13] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network", *AT&T Center for Internet Research at ICSI (ACIRI)*, DRAFT, Available at http://www.research.att.com/~smb/papers/DDOS-lacc.pdf, Feb 5, 2001.

[14] P. Brooke and R. Paige, "Fault Trees for Security System Analysis and Design", *Journal of Computers and Security*, 22(3):256-264, Elsevier, May 2003.

[15] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, and R. Lutz, "A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System", *Proceedings of the 1st Symposium on Requirements Engineering for Information Security*, 2001.

[16] M. Dacier, Y. Deswarte, and M. Kaaniche, "Quantitative Assessment of Operational Security: Models and Tools", *LAAS Research Report 96493*, May 1996 (Extended version of "Models and Tools for Quantitative Assessment of Operational Security," *Proc IFIP/SEC* 1996).

[17] R. Ortalo, Y. Deswarte, M. Kaaniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security", *IEEE Transactions on Software Engineering*, vol 25 , issue 5 , pp. 633-650, Sep-Oct 1999.

[18] A. Brown, G. Kar, A. Keller, "An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Application Environment", *IEEE/IFIP International Symposium on Integrated Network Management*, pp. 377-390, 2001.

[19] C. Carver, J. Hill, and U. Pooch, "Limiting Uncertainty in Intrusion Response", *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, USMA, West Point, NY, Jun 5-6, 2001.