CERIAS Tech Report 2005-01

ADEPTS: ADAPTIVE INTRUSION CONTAINMENT IN DISTRIBUTED SERVICE ENVIRONMENTS

by Bingrui Foo, Yu-Sung Wu, Saurabh Bagchi, Gene Spafford, and Blake Matheny

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

ADEPTS: Adaptive Intrusion Containment in Distributed Service Environments

Abstract

Distributed systems with multiple interacting services, such as distributed e-commerce systems, are suitable targets for malicious attacks because of the potential financial impact. Intrusion detection in such systems has been an active area of research, while the problem of containment has received relatively less attention. Containment seeks to localize the effect of the intrusion to some parts of the system while allowing the other parts to continue to provide service. In this paper, we present the design and implementation of an Adaptive Intrusion Tolerant System, *ADEPTS*, for automatically containing intrusions in a distributed system. ADEPTS uses a directed acyclic graph of intrusion goals, called *I-DAG*, and a graph of service interactions, called *SNet*, as the underlying representations in the system. The containment action in ADEPTS initially has the goal of preventing the spread of the intrusion by modifying its path of escalation in the I-DAG. Failing that, it adopts a more drastic response of modifying the interactions of the services in the SNet. There is also a feedback mechanism for the effectiveness 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 whose value depends on the operational services in the face of an intrusion.

Keywords: automated intrusion response, intrusion containment, e-commerce system, survivability, distributed services.

1 Introduction

Distributed systems comprising multiple services, interacting among themselves to provide end-user functions are becomingly 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 definitions of e-commerce are varied but typically capture the features of near 24X7 accessible service over the internet, and multiple distributed services interacting through standardized interfaces. The huge financial stakes involved in e-commerce make their distributed system infrastructure prime candidates for computer security attacks. In order to meet the challenges of always-on, on-demand service availability, an e-commerce system needs to be resilient to security attacks. This has led to interest in securing such 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 systems), or against patterns of expected system behavior (anomaly based systems). However, resilience to intrusions 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. In this paper, we focus on one of the goals of the response, namely, *containment*, which implies restricting the effect of the intrusion to a subset of the services in the system. Thus, a system's functionality is no longer considered as simply up or down. Instead, containment may allow a possibly restricted set of 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 include mandating interactions pass through additional checks, intermediaries, or be executed over slower channels. Third, the system will have to consider the possibility of certain responses failing and the intrusion spreading as a consequence. The system should have the ability to adapt the response or deploy a different response and maintain a history to guide future response choices.

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 aims to improve a metric called *survivability* of the system. Survivability is a quantified measure of the ability of a system to deliver its essential services when it is subject to failures or attack. At the high level, this is achieved by identifying the compromised, or potentially compromised, set of services and then enforcing a containment boundary around them such that the services outside the boundary can be relied upon to provide dependable functionality.

To achieve the high level goal, ADEPTS uses an **I**ntrusion-**D**irected **A**cyclic **G**raph (I-DAG) as the underlying representation for intrusion events. In an I-DAG, each node represents an intermediate or ultimate goal of an attack. ADEPTS uses a graph representation called Service Net, or *SNet*, for representing the interactions between the services. In the SNet, the nodes represent services, the edges represent the interaction between services, and therefore, a path for the intrusion to spread from one service to its neighbor. There are three broad phases in the execution of ADEPTS. In the first phase, the likelihood of a node in the I-DAG having been achieved, is determined, based on alerts from the intrusion detection framework in the system and the position of the node in the I-DAG. ADEPTS is agnostic about the detector used, though a confidence value associated with the generated alert can make the containment more accurate. In the second phase, an appropriate response on an I-DAG edge is deployed. The goal of the response is to prevent a higher level node in the I-DAG from being achieved. The response is chosen based on three factors – its effectiveness in containing previous attacks of the same type, its disruptivity to normal users of the system, and the likelihood of the I-DAG node having been achieved. The three factors are carefully chosen to take into account the trade-offs in deciding on a response action. For example, a drastic response such as shutting down a directory lookup service may be very effective, but is also likely to be very disruptive to normal users. If a response fails, progressively escalating responses on higher level I-DAG edges may be taken. The third phase of ADEPTS is only invoked if the intrusion spreads to such an extent that the survivability of the system drops below a threshold. In this phase, a more systemic response than the directed I-

DAG response is deployed, and service interactions are modified in an attempt to contain the spread of the intrusion. For example, the write access to a database by a suspect application server may be withdrawn.

The design of ADEPTS is realized in an implementation which provides intrusion containment service to a large distributed e-commerce system. The measurements bring out the latency and adaptability of ADEPTS and its effectiveness in improving the survivability of the system under different kinds of simulated attacks.

The 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, uncertainty in detection, and escalation due to failed response actions. These are of critical importance in an intrusion tolerance system applied to a real-world system.

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 for specific classes of attacks.

The rest of the paper is organized as follows. Section 2 refers to related research. Section 3 presents the data structures in ADEPTS. Section 4 describes the different phases of ADEPTS. Section 5 provides the specific configuration of the system being evaluated in this paper and experiments and 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. There have been hundreds of intrusion detection systems (IDS) available as research prototypes to high-priced enterprise systems and anywhere in between. However, 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 large 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. The delay has important consequences on the chance of success of the attack which was explored by Cohen using simulated attack scenarios [8]. The results show 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. This insight has led to research in survivable systems engineering pioneered by CERT at CMU [9]. Survivability is the metric using which the systems are benchmarked and is defined as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents ([10],[11]). The researchers identify 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 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.

The majority of automated IRSs provide a static mapping between type of attack and the corresponding response action ([37], see [21] for tabulation of current IRSs). Snort provides a rudimentary feature of dropping suspect packets before it reaches the receiver (Snort-inline) and a feature of terminating connections outside the data processing path (Snort Flexible Response). In [16], the authors propose a domain specific language called BMSL in which patterns of anomalous or expected system calls and network packets can be specified. If an anomalous pattern matches, this is taken as an indication of intrusion and one of a set of fixed responses taken. The static IRSs can only handle well understood attacks and cannot deal with failures of the deployed response actions.

There is a comparatively tiny volume of work on adaptive IRSs. In [17], the authors propose a network model that allows an IRS to evaluate the effect of a response on the network services. There are some studies which present taxonomy of offensive and defensive responses to aid in selection of coherent responses in an automated response system ([13],[18],[19]). However, they do not show how automated responses are chosen and deployed. Cooperating Security Managers (CSM) [12] is a distributed and host-based intrusion detection and response system. Depending on the suspicion level CSM assigns to the user and the classification of the attack according to the taxonomy in [13], it employs one of eight different response sets. CSM uses the suspicion level of the user as the only determining factor in the choice of response. A second system called EMERALD ([14],[15]) is a distributed intrusion detection and response system, which consists of hierarchical collections of monitors. A monitor provides localized real-time analysis of infrastructure elements and network services and may interact with its environment passively (reading activity logs or network packets) or actively (via probing). Each monitor includes an instance of the EMERALD resolver, a countermeasure decision engine capable of fusing the alerts from its associated analysis engines and invoking response handlers to counter malicious activity. EMERALD uses two factors in determining the response – the amount of evidence furnished for the intrusion and the severity of the response. In CSM, EMERALD, and BMSL, detection is the main focus of the work and response considered as a side-issue. None of the systems uses record of the past performance of the intrusion detection system, keeps track of the success or failure of the deployed response for use in tempering future responses, or provides a framework for easily incorporating these factors in the automated response determination.

The work that is most closely related to ours is the Adaptive, Agent-based Intrusion Response System (AAIRS) ([20][21]). In AAIRS, multiple IDSs monitor a computer system and generate intrusion alarms. It generates an attack confidence metric based on historical accuracy of the IDSs and passes it to an Analysis Agent. Their decision algorithm for determining if an alarm corresponds to a new attack or an existing attack is adopted by us in ADEPTS. The Analysis agent analyzes an incident and generates an abstract course of action to resolve the incident, using the Response Taxonomy agent from [18] to classify the attack and to determine a response goal. The Analysis agent passes the selected course of action to the Tactics agent, which decomposes the abstract course of action into very specific actions and then invokes the appropriate components of the Response Toolkit. 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. It leaves many unanswered questions, including, what is the algorithm to determine a sequence of response actions to an incident, how does the system measure the success of previous responses, or how are multiple concurrent attacks handled.

There is some previous work on protecting distributed systems against flooding based distributed denial of service (DDoS) attacks in an automated manner ([22][23],[24],[25]). The method used is a two step one – in the first the router traffic is analyzed to detect the physical interfaces through which the DDoS traffic enters the network, followed by installing packet filters or rate limiting rules at the appropriate routers. This body of work targets only prevention and does not handle cases when the prevention fails. The solutions are specific to DDoS attacks, and both the steps are often human labor intensive.

The notion of survivability has been used by several researchers ([26],[27],[28]). The measure has loosely been interpreted as the ability of a system to provide *essential services* in the face of an ongoing intrusion or failure. However, most often the measure has been proposed as one which takes values only in a few discrete steps. It is often unclear how the essential services are defined and further discussion about what services or computer systems are needed to support the essential services is missing. Fault trees have been used extensively in root cause analysis in fault tolerant systems (see [29] for pointers). They have also been used to a limited extent in secure system design – in modeling how a security violation occurs ([30],[32]), or in evolving requirements for a secure application [31]. 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 [33],[34]. The notion of privilege graphs introduced in [34] has some similarity to our I-DAG. 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 therefore difficult to predict.

3 ADEPTS Information Store

ADEPTS uses three primary data structures, which are described here – I-DAG, SNet, and Response Repository. We provide the precise quantitative definition of survivability in Section 3.4. A collection of all the notations used in the paper can be found in the Appendix in Table 3.

3.1 Intrusion DAG (I-DAG)

In the I-DAG representation, each intrusion is considered to have an ultimate goal. The ultimate goal is dissected into intermediate goals and the path to reach the ultimate goal through achieving the intermediate goals is represented. Each goal corresponds to a single node in the I-DAG, the ultimate goal being at the root of the tree. These goal nodes are organized in causal order. Generally speaking, a parent node can be achieved if *any* or *all* its children nodes are achieved. Each node stores two sets of services – a *Cause Service Set* (CSS) and an *Effect Service Set* (ESS). The former set includes all services that may be compromised *in order to achieve the goal* and the latter set includes all the services that are taken to be compromised *once the goal is achieved*.

For a node *X*, its child nodes connected via AND edges represents the condition that *all* the goals corresponding to the child nodes have to be achieved before the goal corresponding to *X* can be achieved. For a node *Y*, its child nodes connected by OR edges represents the condition that the goal corresponding to *any* one of its children has to be achieved before the goal corresponding to *Y* can be achieved. For simplicity in processing, ADEPTS does not allow a mix of AND and OR arcs at a node. An intermediate node is used for aggregating intermediate Boolean expressions.

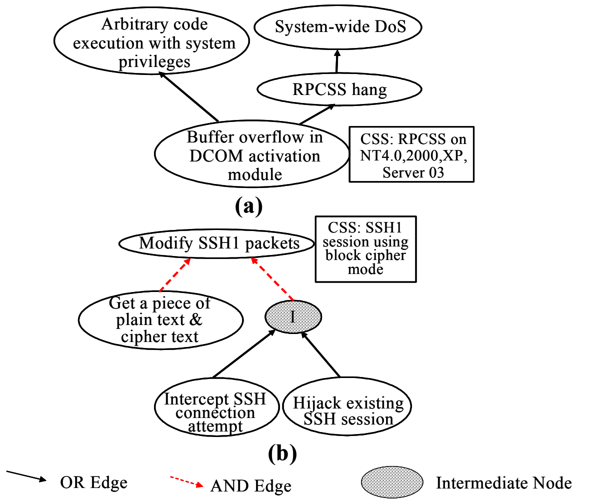

Arbitrary code execution with system privileges
System-wide DoS
RPCSS hang
Buffer overflow in DCOM activation module
CSS: RPCSS on NT4.0,2000,XP, Server 03
(a)
Modify SSH1 packets
CSS: SSH1 session using block cipher mode
Get a piece of plain text & cipher text
I
Intercept SSH connection attempt
Hijack existing SSH session
(b)
OR Edge
AND Edge
Intermediate Node

Figure 1: Sample fragments of I-DAG created from CERT Vulnerability Database

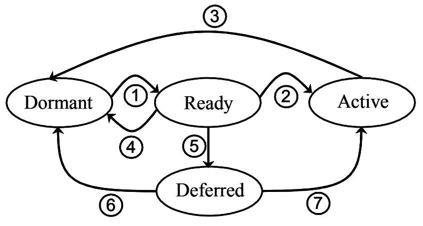
I-DAG creation is a manual process but is suggested rather directly by the organization of several of the most common vulnerability databases, such as CERT's Vulnerability Notes database [38] and Mitre's Common Vulnerabilities and Exposures (CVE) database [39]. We will consider the former database and illustrate the creation of two fragments of the I-DAG used in ADEPTS. A vulnerability in the DCOM interface within the Remote Procedure Call Service (RPCSS) of most Windows OS versions (VU ID#254236 and 483492), which has one of the highest values of CERT's vulnerability metric is chosen. The vulnerability may be exploited to execute arbitrary code with local system privileges or launch a denial of service attack by causing the RPCSS to hang. Thus, a single vulnerability can lead to multiple higher level goals and is part of the reasoning that drove our decision to use DAGs, rather than the more commonly used trees. As a second example, we consider an attack enabled by a vulnerability in the SSH1 protocol that allows packets encrypted with a block cipher to be modified without notice (VU ID#13877). This attack is specified with a mix of conjoint and disjoint pre-conditions which immediately translate to the AND and OR edges of the I-DAG. The I-DAG fragments corresponding to the two attacks are shown in Figure 1(a) and (b).

3.2 Service Net (SNet)

ADEPTS uses a directed graph, called *SNet*, to model services and their interactions in executing the operations supported by the system. In SNet, the nodes represent services in the system. The directed edges represent links with which one service can affect another. For example, there may be a node to represent the Apache web server and another to represent the MySQL database. Apache may write some data to the MySQL database through a MySQL library call. This would be represented by an edge from the web server to the database. Each edge corresponds to a specific mode of interaction between the services, namely, read, write, execute, and request-response. The SNet and the I-DAG are both structures for representing the system ADEPTS is meant to protect. The connection is provided by the fact that the SNet nodes are a superset of the services in the ESSs of the I-DAG nodes. Each SNet edge has an associated numerical value, called the *Propagation Time*, indicating the time required for the service at the head to influence the service at the tail of the edge. For edges which represent request-response interactions, the value can be given by the time required for the transaction to be completed, while for other kinds of edges, the value comprises three factors – the time to instantiate the message, the latency on the communication medium, and the time to process the message. This value needs to be updated through runtime statistics.

3.3 Response Repository

A global response repository is used to store all the responses available in ADEPTS. The responses are associated with the edges in the I-DAG and when a response is active, it implies that, with a certain coverage, the attacker is unable to reach the destination goal node from the source node. As a side-effect, a response may impact a service interaction in the SNet. A response has four phases in its lifetime – dormant, ready, deferred, and active. The state transition diagram of a response is shown in Figure 2.



1: Intrusion detected, response chosen; **2**: Response deployed; **3**: Response deactivated by system admin; **4**: Response not deployed due to subset relation with an active response; **5**: Response deferred due to opposition relation with an active response; **6**: Response re-evaluated and not selected; **7**: Response re-evaluated and selected and deployed

Figure 2: State transition diagram for a response in ADEPTS

A response is *dormant* when it resides passively in the repository, it becomes *ready* when an intrusion occurs and the response is chosen for deployment, and it becomes *active* once the response is deployed. It can be *deferred* if it is inconsistent with a currently active response. The response may be deactivated by the system administrator after a length of time or after a Time to Live (TTL) for the response expires, taking it back to the dormant state.

Each response is associated with a *Disruptivity Index* (DI) and an *Effectiveness Index* (EI). Each index is a real number in the range (0,1). DI captures the notion of how disruptive the response is to the normal users of the system. The higher the DI value, the more disruptive the response is anticipated to be. The second parameter, EI maintains the history of how effective the previous deployments of the response have been, in containing past instances of the intrusion. The EI is incremented and decremented according to the algorithm outlined in Section 4.1.4. Each response optionally has a subset, superset, or opposition relation with every other response in the system. A response R1 is a *subset* (*superset*) of a response R2 if R1's (R2's) effect is subsumed in R2's (R1's) effect. A response R1 is in *opposition* to a response R2 if the effect of R2 is diminished or nullified by the effect of R1. Opposition is assumed to be a commutative relation. At the time of deploying R1 when R2 is in the active state, R1 is moved to the dormant state if it is a subset of R2. If R1 is in opposition to R2, it is moved to the deferred state and re-evaluated once R2 becomes dormant.

3.4 Survivability

The survivability metric is used to quantify the effectiveness of ADEPTS. The general notion of survivability used is similar to previously proposed definitions ([9][10][11]), namely, the ability of a network computing system to provide essential services in the presence of attacks and failures, and recover full services in a timely manner after the causes have disappeared. In our proposed method, survivability is a real number in the range (0,1). In ADEPTS, at the highest level of abstraction, a set of *High-Level System Goals* (HLSG) is identified. Each HLSG is divided into *High-Level System Transactions* (HLST), each with a weight variable indicating the importance of the transaction to the owner of the system. The weight is called the *Survivability Contribution* (θ) ($\sum_i \theta_i = 1$). Each HLST is achieved by a conjunction and disjunction of several *Service Interaction Chains* (SIC). Each SIC is given by a causal ordering of two or more services in the system. Each SIC is also assigned a survivability contribution θ as described below. The system survivability is reduced by this measure if the SIC is non-functional, i.e., if any of the services in the chain is considered to be compromised. Note that chains of services rather than individual services contribute to the survivability of the system.

Consider there are *N* HLSTs in the system – T_1, T_2, …, T_N, with corresponding survivability contributions θ_1, θ_2, …, θ_N ($\sum_{i=1}^{N} \theta_i = 1$). HLST T_i is achieved by a conjunction of multiple composite SICs C_{ij}, where $j=1,\ldots,p_i$, i.e., each composite SIC must be functional for the HLST to be achieved. A composite SIC C_{ij} is achieved by a disjunction of multiple SICs C_{ijk}, $k=1,\ldots,p_{ij}$, i.e., any SIC in the composite SIC can be functional. Next, we compute the θ due to a SIC C_{ijk}. The intuition is that each composite SIC C_{ij} gets the entire θ of the HLST it helps achieve (θ_i). When apportioning this survivability contribution to the different SICs, an option would be to make it zero as long as any other SIC in the composite SIC is functional, and equal to θ_i otherwise. However, this survivability measure is not smooth and a single failure in a service can cause it to drop abruptly. Therefore, each SIC is assigned a θ independent of the status of the other SICs. Let us define the length of a SIC to be the number of service interactions in the chain, i.e., the number of services involved – 1. It is assumed that it is *easier* for longer SICs to be disrupted since more number of services are involved. Therefore, the survivability contribution is biased in favor of shorter SICs. Suppose, the lengths of the SICs C_{ij1}, C_{ij1}, …, $C_{ijp_{ij}}$ are respectively L_{ij1}, L_{ij2}, …,

$L_{ijp_{ij}}$. Let L = L_{ij1}+ L_{ij1}+… $L_{ijp_{ij}}$ The survivability contribution is allocated in the proportion (L- L_{ij1}): (L- L_{ij2}):… (L- $L_{ijp_{ij}}$).

The survivability of the system at any point in time is given by the sum of the survivability contributions of the different HLSTs. For an unimpaired system, this is 1. At runtime, in the event of an intrusion, the survivability contribution of an HLST is reduced by the survivability contribution of each SIC that becomes non-functional. The reduction is bounded by keeping any HLST survivability contribution to be non-negative. The computation is shown Table 1 through a synthetic example derived from the TPC-W benchmark [40]. Three HLSTs are considered, which are implemented by different combinations of 9 services, S1 through S9. The three HLSTs considered are the buy confirm web interaction (T1), search request web interaction (T2), and order inquiry web interaction (T3). At runtime, an intrusion causes services S2, S3, S4, and S6 to be compromised. The impaired system HLST θ is arrived at by reducing the unimpaired system HLST θ by the reductions in the SIC θ's.

HLST	**SIC**	**Unimpaired System**		**Impaired System**	
		HLST θ	SIC θ's	HLST θ	Dec. SIC θ's
T1	AND[(S1,S3,S2), OR[(S5,S2,S6),(S1,S7,S8)]]	0.5	0.5 0.25 0.25	0.0	0.5 0.25 0.0
T2	AND[(S6,S2,S5)]	0.3	0.3	0.0	0.3
T3	AND[OR[(S4,S5,S6,S7), (S1,S7,S8)]]	0.2	0.08 0.12	0.12	0.08 0.0
Aggregate System Survivability		**1.0**		**0.12**	

Table 1: Sample calculation of the system survivability without and with intrusions

4 ADEPTS Phases

ADEPTS operates in several phases which are pictorially represented in Figure 3.

The first step is an alert generated by the detection framework in the system. The alert is associated with an I-DAG node and optionally has a confidence value associated with it, denoting the level of confidence that the alert is correct. An example of such a detection framework is provided in [6]. ADEPTS is neutral to the choice of the detection framework and the confidence value, while useful in making the containment action more accurate, is not essential.

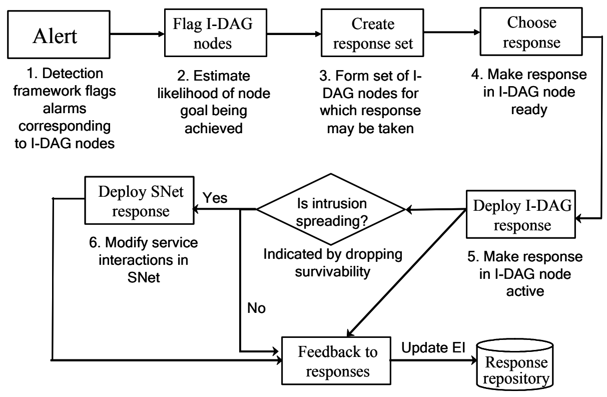

Alert
Flag I-DAG nodes
Create response set
Choose response
1. Detection framework flags alarms corresponding to I-DAG nodes
2. Estimate likelihood of node goal being achieved
3. Form set of I-DAG nodes for which response may be taken
4. Make response in I-DAG node ready
Deploy SNet response
Yes
Is intrusion spreading?
Deploy I-DAG response
6. Modify service interactions in SNet
Indicated by dropping survivability
5. Make response in I-DAG node active
No
Feedback to responses
Update EI
Response repository

Figure 3: Schematic showing different phases of ADEPTS

4.1 Algorithm for Determining I-DAG Response

4.1.1 *CCI Computation Algorithm*

When a detector in the system flags an intrusion alarm and there is a node in the I-DAG corresponding to the event, the confidence of the alarm is assigned to the node. If the detector does not provide an inbuilt confidence value with the alert, then the accuracy of the detector is used as the confidence. The *Compromised Confidence Index* (CCI) of a node in the I-DAG is defined as the probability that the node has been compromised. It is computed based on the confidence of the alert corresponding to the node and the CCI of its immediate lower goal nodes. Mathematically, the CCI of a node is given by

$$\text{CCI} = f(f'(\text{CCI}_i), \text{Confidence of alert})$$

where the CCI_is correspond to the indices of the children. The function f' is given by *max* for OR-edges connecting the children to the parent, and by *min* for AND-edges. The intuition is that for an OR-edge, the parent node can be achieved if any of its children are achieved and therefore the likelihood is the maximum of that 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. The function *f* for the current design of ADEPTS is the statistical mean. For nodes with no children, their CCI is the confidence of the detector. For nodes with no corresponding alerts, their CCI is a propagation of their children's CCIs.

When new alerts are passed to this algorithm, the I-DAG is traversed in a bottom-up manner, starting only from the nodes corresponding to the new alerts, and the CCIs of the nodes are calculated till the root nodes are reached. This traversal easily lends itself to parallelization where multiple threads can be used to traverse non-overlapping sections of the I-DAG. The current ADEPTS design uses thread pools with an empirically determined optimum number of threads for a given size of the I-DAG and given number of concurrent alerts.

4.1.2 Response Set Computation Algorithm

The Threshold of a node, τ_N, is defined as a user-defined value such that if the CCI of the node is greater than τ_N, the system concludes the node goal has been achieved. After the bottom-up CCI computation, the second step is to traverse the I-DAG top-down starting only from the root nodes that were reached during the first step. This also lends itself to parallelization of the traversal and a thread pool is used in ADEPTS. During the top-down traversal, each node is labeled as one of: (i) *Strong Candidate* (SC). A node with CCI greater than τ_N; (ii) *Weak Candidate* (WC). A node with CCI less than or equal to τ_N, but a higher level SC node can be reached through it; (iii) *Non-Candidate* (NC). A node that is neither an SC nor a WC node.

Next, the nodes are placed in a Response Set, indicating to the next phase where responses should be deployed. There are two ways in which a node can be placed in the response set: (i) It is an SC node and all its parents are NC nodes, i.e., it is the highest SC node; or (ii) It is an SC or a WC node and there is at least one immediate parent which is an NC node. The motivation behind the first class is that it is the highest level SC node and action should be taken to prevent higher level NC nodes being reached. The motivation behind the second class is that the node may have been compromised with differing degree of certainty but there is at least one higher level uncompromised node that may be reached through it.

4.1.3 Response Determination Algorithm

For each node placed in the Response Set, the responses on all its outgoing edges in the I-DAG are considered. For each such response, an index called the *Response Index* (RI) is computed. RI is a real number $\in (0, 1)$, with a higher value indicating it is more apt to take the response. RI is calculated as the weighted sum of three measures

– the CCI of the node, the Disruptivity Index (DI), and the Effectiveness Index (EI). The three factors bring out the important trade-offs guiding the choice of a response between effectiveness, surety it is needed, and inconvenience to legitimate users. DI and EI were introduced in Section 3.3. For each node in the Response Set, the response with the highest RI is deployed unless it has an *opposition relation* with an active response. In such a case, the response with the next highest RI is chosen. For a response to be deployed, its RI must be above a response threshold, τ_R. This is to ensure that only responses with a minimal suitability are deployed. When there is no response above τ_R for an edge, ADEPTS evaluates edges higher up in the I-DAG. As nodes are separated further apart from the node at which the alert was flagged, the CCI is expected to decrease but the response could still be chosen if the EI and the DI are suitably high. If multiple concurrent alerts are input to ADEPTS, multiple responses may be chosen and deployed.

4.1.4 Response Feedback Algorithm

The feedback to the response system is provided by dynamically varying the Effectiveness Index (EI) of the response. After a response has been deployed by ADEPTS, 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 an indication that the response action possibly failed and its EI is decreased.

The amount by which the EI is decreased depends on whether the response is on an AND edge or on an OR edge to the higher level node. 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 (currently 0.2) for responses on all the edges. If the response is on an OR edge, then the edge may not have been used to reach the higher level node. The EI is decreased in the proportion of the CCI values of the nodes, the total decrease being the same as in the AND case, i.e., 0.2. This process is illustrated by an example I-DAG in Figure 4 with the CCI values of the nodes shown

Consider there are active responses on the edges EB and DB and node B is achieved. The EI of each response is updated as $EI = EI(1-0.2)$. If there is an active response on the edge GC and the node A is achieved, the EI is updated as $EI = EI(1-0.2(\frac{0.9}{0.9+0.7})(\frac{0.6}{0.6+0.8}))$.

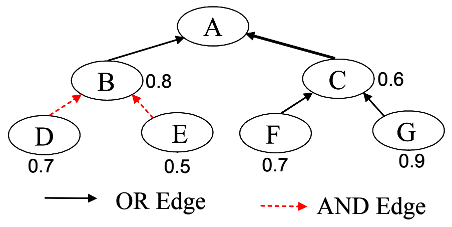


Figure 4: Sample I-DAG for illustration of response feedback algorithm

When a response's Time To Live (TTL) expires or when an administrator manually deactivates a response before expiry, the EI of the response action is increased by a fixed percentage (currently 0.2) under the intuition that the response was successful since further alerts were not observed.

4.2 Algorithm for Determining SNet Response

The I-DAG response may fail due to the incompleteness of the attack paths in it, or the fast spread of the intrusion. The SNet response which modifies the service interactions is the second line of defense. The SNet is expected to be relatively static once the system is deployed and more complete than the I-DAG. The SNet response is guided by two thresholds – a desirable survivability (S_{des}) and a minimum tolerable survivability (S_{min}). As the intrusion spreads from the compromised services (the epicenter), the survivability of the system drops. When the survivability drops below S_{des}, the SNet response is triggered. This is led by the belief that at this point I-DAG responses are incapable of preventing the spread.

Essentially, the SNet response enforces a pessimistic containment boundary (CB) around services that are already compromised or will potentially be compromised within a given radius of the epicenter services. The radius is given in terms of time and is computed as the maximum of the boundary that will keep the system survivability above S_{min} and the latency of ADEPTS in enforcing the CB. This balances the two conflicting factors in a conservative or pessimistic manner. On the one hand, a small CB keeps system survivability high, but may be useless since compromised services may be left outside the boundary, thereby motivating a larger CB. The services that lie on the CB are stopped with the goal of preventing the spread of the intrusion outside the boundary. From a graph theoretic point of view, this can be looked upon as removing the nodes from the graph thereby disconnecting the graph into two components – one that contains compromised or suspect services and one that contains services whose delivered functionality can be trusted.

5 Experiments and Results

5.1 Testbed

ADEPTS is deployed on a realistic distributed e-commerce system comprising multiple services to enable web-enabled transactions. The primary services and the applications that implement them are web client (MS Internet Explorer), web server (Apache), application server (Zope), directory server (OpenLDAP), database server (MySQL), portal server (Zope), and mail server (sendmail). The configuration of the servers and the SNet corresponding to the services are shown in the Appendix (Figure 10 and Figure 11). Different vulnerabilities in the services are identified and several different kinds of attack scenarios drawn up leading to an I-DAG with 34 nodes, which is shown in the Appendix in Figure 12. The I-DAG is automatically generated using Graphviz [7] from a specification of the vulnerabilities. The highest level goal achieved through the attack scenarios is the shutdown of the web store. Some high level goals are executing arbitrary code on the web server, leaking of confidential information, and network DoS. A response repository is used for responses on the I-DAG edge with a total of 49 responses. A response may also affect service interactions, such as the response of disabling the mod_SSL functionality in the Apache core cuts the interaction between the Apache core and the mod_SSL module. Some of the responses have high DI, such as the above mentioned one, and some have low DI, such as disabling SSLv2 handshaking in mod_SSL module.

5.2 Experiment 1: Scalability test

This experiment is meant to bring out the scalability of ADEPTS with respect to the size of the system it is deployed on and the number of concurrent alerts it can handle. The size of the system is represented by the size of the I-DAG, since a larger sized system is expected to have a larger number of ways of attacking it. The output metric is the time for deployment of the I-DAG responses, measured from the time that the alerts enter the system.

The I-DAG sizes of 100, 400, and 900 nodes are used. Alerts are generated randomly for a fraction of the nodes in the I-DAG. The ADEPTS latency and the number of responses deployed are shown in Figure 5 and Figure 6.

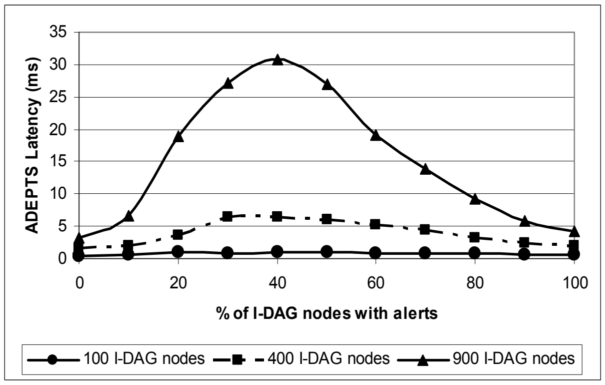


Figure 5: Variation of ADEPTS latency with number of concurrent alerts

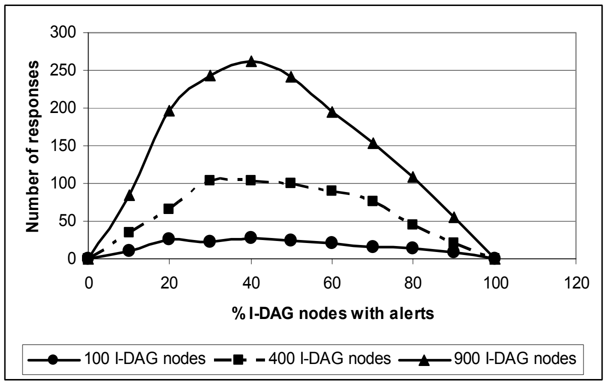


Figure 6: Variation of number of responses with number of concurrent alerts

It is seen that the ADEPTS latency rises from a very small value to a maximum for 40% of the nodes being flagged and then decreases. This is attributable to the fact that a major component of the latency is the number of responses deployed and the number of responses peaks at 40%. With a small number of alerts, the number of responses is understandably small and it increases as the number of alerts increases. However, beyond a certain number of alerts, responses on the higher level edges of the I-DAG are favored over multiple lower level responses. The time to deploy a higher level response is not substantially higher and therefore the time comes down. Another observation is that with even 360 concurrent alerts (1000 node I-DAG, 40% flagged), about 262 responses are deployed and the latency is 250 ms, which can be expected to be tolerable for several classes of attacks.

5.3 Experiment 2: Escalation test

The goal of this experiment is to illustrate the effectiveness of the algorithm for the choice of an I-DAG response. Since different kinds of attacks are difficult to implement and inject in the real system, alerts corresponding to different types of attacks are *simulated* for the evaluation. Thus, the algorithms under evaluation are real implementations, while the alarms are simulated.

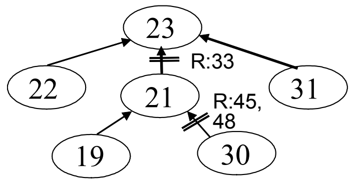


Figure 7: Fragment of I-DAG for Experiment 2

Repeated attacks are simulated on a fragment of the I-DAG nodes and three responses are evaluated (Figure 7). The response threshold (τ_R) and the node threshold (τ_N) are set to 0.5. The DI of the responses 45, 48, and 33 are set at 0.5, 0.6, and 0.3, respectively. The EI of each response is initially set to 0.5. The RI computation formula is RI = 0.1 CCI + 0.8 EI + 0.1 (1 - DI).

Three repeated instances of an attack are simulated, with each instance having two steps. In the first step, nodes 19 and 30 are flagged and in the second, node 23 is flagged. The steps are all shown in Table 2. Some of the entries are highlighted and marked for discussion.

t	Alerts (Node,Confidence)	Response Set	Response ID 45			Response ID 48			Response ID 33			Response Deployed
			CCI	EI	RI	CCI	EI	RI	CCI	EI	RI	
0	No alert	ϕ	0	0.5	NC	0	0.5	NC	0	0.5	NC	ϕ
1	(19,0.2), (30,0.8)	**30**$^{(a)}$	0.8	0.5	0.53	0.8	0.5	0.52	0.4	0.5	NC	**45**$^{(b)}$
2	(23,0.8)	**23**$^{(c)}$	0.8	**0.42**$^{(d)}$	NC	0.8	0.5	NC	0.4	0.5	NC	ϕ
3	(19,0.2), (30,0.8)	30	0.8	0.42	0.466	0.8	0.5	0.52	0.4	0.5	NC	**48**$^{(e)}$
4	(23,0.8)	23	0.8	0.42	NC	0.8	0.42	NC	0.4	0.5	NC	ϕ
5	(19,0.2), (30,0.8)	30	0.8	0.42	0.466	0.8	0.42	0.456	0.4	0.5	0.51	**33**$^{(f)}$
6	(23,0.8)	23	0.8	0.42	NC	0.8	0.42	NC	0.4	0.4	NC	ϕ

Table 2: Demonstration of response adaptation (NC: Not computed, ϕ: Null set)

In (a), 30 is included in the Response Set since it is a SC node and its parent node is a NC node. In (b), response 45 is deployed in preference to 48 since 48 has a higher DI and the other factors are equal. In (c), 23 gets into the Response Set and 30 gets evicted since 21 becomes a WC and therefore 31 no longer has a NC node as parent. In (d), the EI of response 45 is decreased since 23 is flagged, indicating the likelihood that the response failed and consequently the intrusion escalated. The decrease is calculated as 0.5(1-0.2.(0.8/0.8+0.2)) = 0.42. In (e), response 48 is chosen over 45 since the latter's EI has been reduced due to the previous inferred failure. Item (f) shows an instance of escalation where ADEPTS chooses a more wide-ranging response at a higher level edge in the I-DAG since the immediate responses have failed previously.

5.4 Experiment 3: Effect of Intrusion Propagation on Survivability

This experiment brings out the effect of an expanding containment boundary (CB) on the survivability of the system. For the system, 12 different HLSTs are used, such as browsing the web store. The simplification used is that each HLST is implemented by a single SIC. The most important HLST is storing customers' personal data (θ=0.33) and least important HLST is the ability of backup servers to contact the DHCP server (θ=0.033). The results shown in Figure 8 are with the SSL module in the Apache web server (service ID 1) as the epicenter. The events correspond to additional nodes in the SNet being compromised. The most immediate result is that the expanding CB decreases the survivability. Thus, the pessimistic CB algorithm can use this result to decide on the CB radius given the minimum tolerable survivability. It is also noted that for some cases, expanding the radius does not degrade the survivability. This is possible if the expansion does not cut any new service interaction, or the service interaction cut does not contribute to the survivability.

5.5 Experiment 4: Survivability Improvement in ADEPTS

This experiment demonstrates the effectiveness of ADEPTS in improving the survivability of the system. ADEPTS is compared to the baseline system with no containment. Two different settings of Adepts are used, one for the best case where the I-DAG response action is always successful, and another where all the response actions on the I-DAG are unsuccessful and the most drastic action on the SNet edges to enforce the pessimistic boundary is successful. It can be argued that if all interactions between two services are disabled in a timely manner, the containment is guaranteed to succeed and hence, the latter case is the worst case for ADEPTS. The survivability plots are shown in Figure 9. The desirable survivability (S_{des}) and the minimum tolerable survivability (S_{min}) are set at 60% and 30%, respectively. The different events correspond to I-DAG nodes being achieved starting from node 27 and in the order 14, 8, 15, 6, 19, 21, and 23. The fragment of the I-DAG showing these nodes is presented in the Appendix (Figure 12).

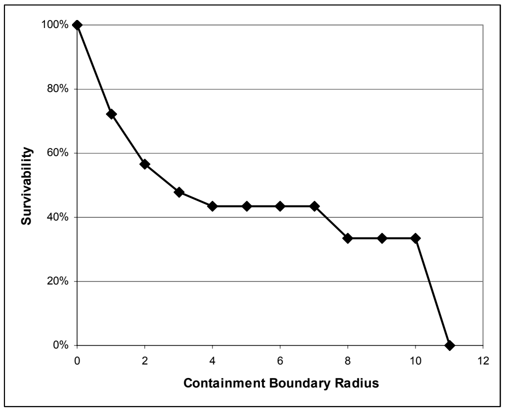


Figure 8: Survivability variation with CB

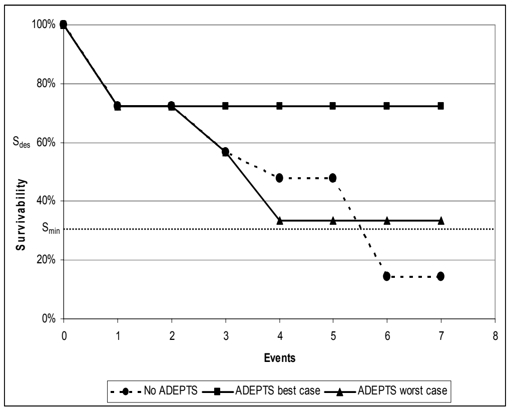


Figure 9: Survivability with and without ADEPTS

In the baseline system, the survivability drops continuously finally stabilizing at a low value of 14%, below S_{min}. In the ADEPTS best case, the I-DAG response on the edge (14, 8) is successful and the survivability stays above S_{des}. In the worst-case, the pessimistic CB is enforced once the survivability drops below S_{des} and the containment algorithms manages to keep the survivability above S_{min}. Notice that in the ADEPTS worst-case, the survivability is temporarily worse than in the baseline case. This occurs when the pessimistic CB algorithm draws a large CB.

6 Conclusions

In this paper we have presented the design and implementation of a system called ADEPTS for automated containment of intrusions in a system of distributed interacting services. It uses directed containment using attack paths represented in an I-DAG, and if that fails, attempts containment through modifications of service interactions represented in the SNet. The survivability metric is proposed for benchmarking an intrusion containment system, such as ADEPTS. ADEPTS is evaluated using a real implementation on a realistic distributed e-commerce testbed, with simulated alerts. The experiments show that it is scalable with system size and number of concurrent alerts, is adaptive to failed responses and escalation of intrusions, and can improve the survivability compared to a baseline system with no containment mechanisms.

In the future we are considering, more fine-grained responses on the SNet, such as modifying specific modes of interaction. We wish to investigate what effect it has on the propagation speed of attacks. We are looking at passive service discovery to discover the SNet in a semi-automated manner. We believe that some minimal inputs from the system administrator would be necessary. We are also investigating making more dynamic updates to the interaction times between the services.

References

[1] S. Garfinkel and G. Spafford, "Web Security & Commerce," O'Reilly, 1997.
[2] A. Ghosh, "E-Commerce Security," John Wiley and Sons, Inc., Third Avenue, New York, 1998.
[3] V. Hassler, "Security Fundamentals for E-commerce," Artech House, 2001.
[4] L. D. Stein, "Web Security: A step-by-step reference guide," Addison Wesley, Reading, Massachusetts, 1999.
[5] Forrester Research Inc., "US E-Commerce Overview: 2003 to 2008," Techstrategy Brief Series, July 2003.
[6] Yu-Sung Wu, Bingrui Foo, Yongguo Mei, and Saurabh Bagchi, "Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS," In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC '03), December 8 - 12, 2003.
[7] AT&T Research, "Graphviz - open source graph drawing software," Available at: http://www.research.att.com/sw/tools/graphviz/
[8] F. B. Cohen, "Simulating Cyber Attacks, Defenses, and Consequences," Available at http://all.net/journal/ntb/simulate/simulate.html, May 13, 1999.
[9] Carnegie Mellon, Software Engineering Institute, "Survivable Network Technology," Available at: http://www.sei.cmu.edu/organization/programs/nss/surv-net-tech.html
[10] R. H. Anderson, A. C. Hearn, and R. O. 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 Coordination Center, Software Engineering Institute, Carnegie Mellon University, 1997.
[11] R. Ellison, R. Linger, T. Longstaff, N. Mead, "Case Study in Survivable Network System Analysis," CMU/SEI-98-TR-014, ADA355070, Software Engineering Institute, Carnegie Mellon University, 1998.
[12] G. B. White, E. A. Fisch, and U. W. Pooch, "Cooperating Security Managers: A Peer-based Intrusion Detection System," IEEE Network, vol. 10, no. 1, January/February, 1996, pp. 20-23.
[13] E. A. Fisch, "Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior," Ph.D. Dissertation, Texas A&M University, College Station, TX, 1996.
[14] P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," in Proc. 20th National Information Systems Security Conf., Baltimore, MD, October 7-10, 1997, pp. 353-365.
[15] P. G. Neumann and P. A. Porras, "Experience with EMERALD to Date," in Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, April 11-12, 1999.
[16] T. Bowen, D. Chee, M. Segal, R. Sekar, T. Shanbhag, P. Uppuluri, "Building Survivable Systems: An Integrated Approach based on Intrusion Detection and Damage Containment," DARPA Information Survivability Conference and Exposition (DISCEX '00), pp. 84-99, vol. 2, Jan 2000.
[17] Thomas Toth and Christopher Kruegel, "Evaluating the Impact of Automated Intrusion Response Mechanisms," 18th Annual Computer Security Applications Conference (ACSAC '02), December 9 - 13, 2002.
[18] Curtis A. Carver, Jr., and Udo W. Pooch, "An Intrusion Response Taxonomy and its Role in Automatic Intrusion Response," Proceedings of the 2000 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 6-7 June, 2000.
[19] U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions," Proc. 1997 IEEE Symposium on Security and Privacy, Oakland, CA, May 4-7, 1997, pp. 154 - 163.
[20] Daniel Ragsdale, Curtis Carver, Jeffery Humphries, and Udo Pooch, "Adaptation Techniques for Intrusion Detection and Intrusion Response Systems," In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashville, Tennessee, October 8-11, 2000, pp. 2344-2349.
[21] Curtis A. Carver, John M.D. Hill, and Udo W. Pooch, "Limiting Uncertainty in Intrusion Response," Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June, 2001.
[22] Recourse Technologies, "ManHunt product description," Available at: http://www.recourse.com/products/manhunt/features.html.
[23] Dan Schnackenberg, Harley Holliday, Randall Smith, Kelly Djahandari, Dan Sterne, "Cooperative Intrusion Traceback and Response Architecture (CITRA)," DARPA Information Survivability Conference and Exposition (DISCEX II'01), June 12 - 14, 2001.
[24] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday, and T. Reid, "Autonomic Response to Distributed Denial of Service Attacks," In Proceedings of the 4th International Symposium on Rapid Advances in Intrusion Detection, RAID 2001, Davis, CA, USA, October 2001.
[25] 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, February 5, 2001. Available at: http://www.research.att.com/~smb/papers/DDOS-lacc.pdf
[26] K. Kyamakya, K. Jobman, M. Meincke, "Security and Survivability of Distributed Systems: An Overview," At the 21st Century Military Communications (MILCOM '00), 2000.

[27] S. Jha, J. Wing, R. Linger, T. Longstaff, "Survivability Analysis of Network Specifications," In Workshop on Dependability Despite Malicious Faults, International Conference on Dependable Systems and Networks (DSN), June 2000.

[28] Matti A. Hiltunen, Richard D. Schlichting, Carlos A. Ugarte, and Gary T. Wong, "Survivability through Customization and Adaptability: The Cactus Approach," DARPA Information Survivability Conference and Exposition (DISCEX 2000), pp. 294--307, January 2000.

[29] Daniel P. Siewiorek and Robert S. Swarz, "Reliable Computer Systems – Design and Evaluation," Chapter 5: Evaluation Criteria, pg. 350, A. K. Peters Ltd., Third Edition.

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

[31] Guy Helmer, Johnny Wong, Mark Slagell, Vasant Honavar, Les Miller, and Robyn Lutz, "A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System," In Proceedings of the First Symposium on Requirements Engineering for Information Security, 2001.

[32] B. Schneier, "Attack Trees: Modeling Security Threats," Dr. Dobbs Journal, December 1999, Available at: http://www.counterpane.com/attacktrees-ddj-ft.html.

[33] 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'96).

[34] R. Ortalo, Y. Deswarte, M. Kaaniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security," IEEE Transactions on Software Engineering, Volume: 25 , Issue: 5 , pp. 633-650, Sept.-Oct. 1999.

[35] "Snort Flexible Response Add-On," Available at: http://cerberus.sourcefire.com/~jeff/archives/snort/sp_respond2/

[36] "Snort In-line," Available at: http://sourceforge.net/projects/snort-inline/

[37] "The Linux Intrusion Detection System (LIDS) Project," Available at: http://www.lids.org

[38] US-CERT, "US-CERT Vulnerability Notes Database," At: http://www.kb.cert.org/vuls

[39] Mitre Corporation, "Common Vulnerabilities and Exposures," At: http://www.cve.mitre.org/

[40] Transaction Processing Performance Council (TPC), "TPC-W: A transactional web e-commerce benchmark," At: http://www.tpc.org/tpcw/

7 Appendix

7.1 Notation List

This paper introduced several notations and Table 3 aggregates the different notations for reference.

Notation	Full form	Notation	Full form	Notation	Full form
I-DAG	Intrusion Directed Acyclic Graph	SNet	Service Net	CCI	Compromised Confidence Index
CSS	Cause Service Set	ESS	Effect Service Set	RI	Response Index
DI	Disruptivity Index (for a response)	EI	Effectiveness Index (for a response)	τ_N	Threshold CCI for an I-DAG node
HLSG	High-Level System Goal	HLST	High-Level System Transaction	τ_R	Threshold RI for an I-DAG response
θ	Survivability Contribution	SIC	Service Interaction Chain	S_{min}, S_{des}	Minimum tolerable and desirable survivability

Table 3: List of notations used in the paper

7.2 Service Topology

ADEPTS is deployed on a testbed with a distributed set of services constituting an e-commerce system. The service topology is shown in Figure 10.

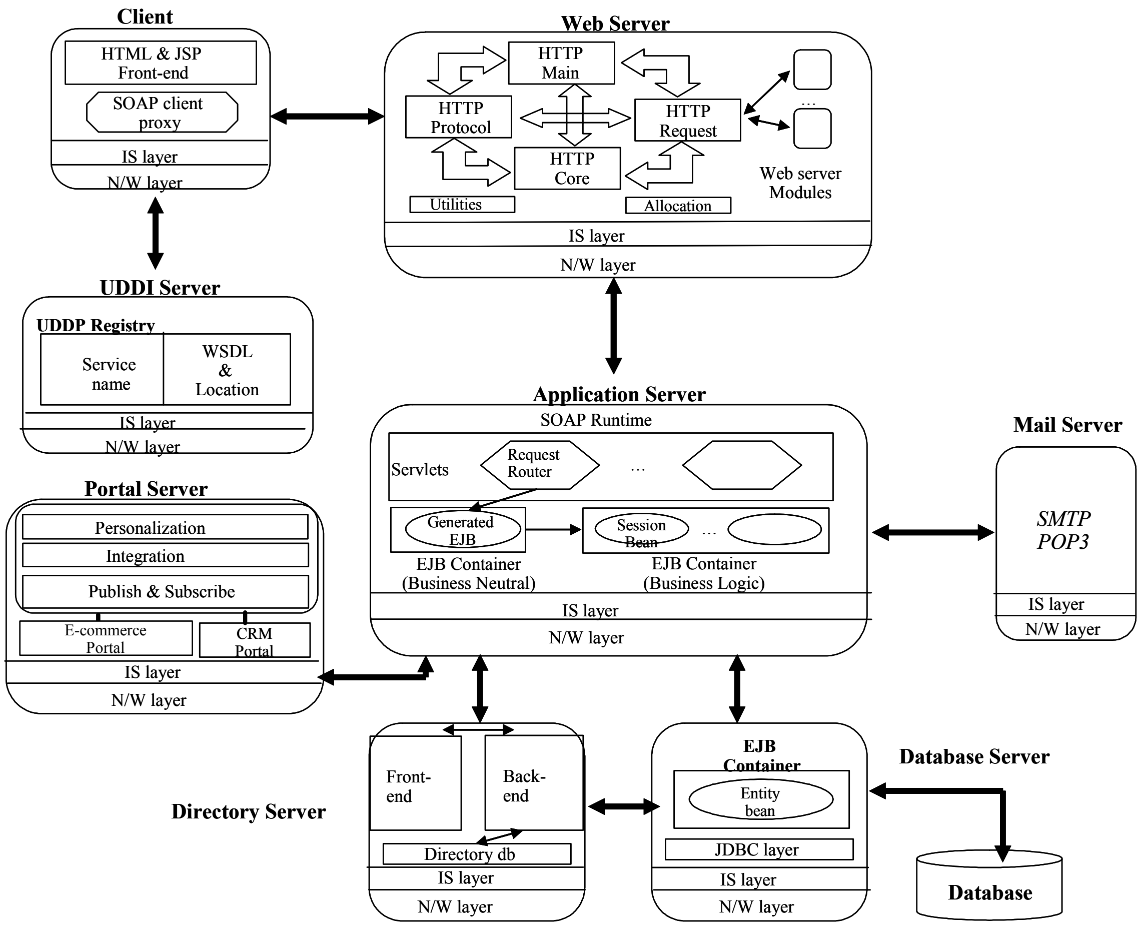


Figure 10: Distributed E-commerce System on which ADEPTS is Deployed

The SNet representing these services is shown in Figure 11.

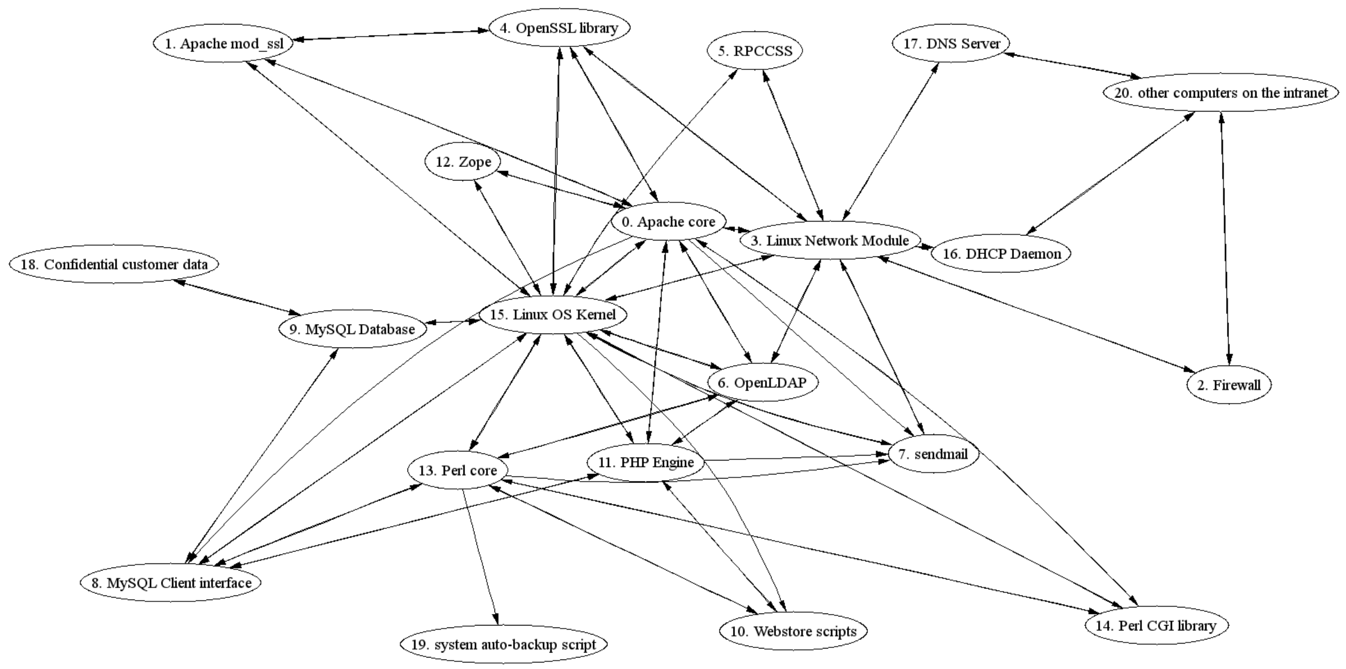


Figure 11: SNet for the Services in the E-Commerce Testbed

7.3 Attack Scenarios

The attack scenarios simulated are represented through the I-DAG. The entire I-DAG comprises 33 nodes and is too large to be shown here. A fragment of the I-DAG showing the path used in experiment 4 is given in Figure 12. A sample response is also shown on an I-DAG edge.

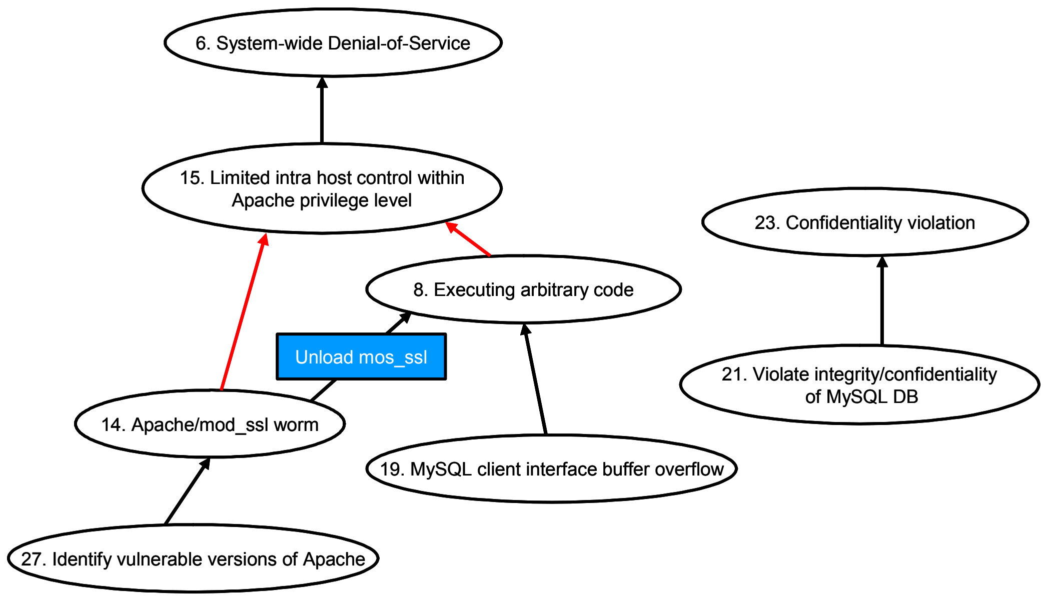

6. System-wide Denial-of-Service
15. Limited intra host control within Apache privilege level
23. Confidentiality violation
8. Executing arbitrary code
Unload mos_ssl
21. Violate integrity/confidentiality of MySQL DB
14. Apache/mod_ssl worm
19. MySQL client interface buffer overflow
27. Identify vulnerable versions of Apache

Figure 12: Fragment of I-DAG used for the Experiments