2009

A framework for cost-sensitive automated selection of intrusion response

Christopher Roy Strasburg
Iowa State University

A framework for cost-sensitive automated selection of intrusion response

by

Christopher Roy Strasburg

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Information Assurance; Computer Science

Program of Study Committee:
Johnny S. Wong, Co-major Professor
Samik Basu, Co-major Professor
Tom Daniels

Iowa State University

Ames, Iowa

2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank the committee members for their time, instruction, inspiration, and support. In particular, my advisers Dr. Johnny S. Wong and Dr. Samik Basu for their discussion, feedback and input through many meetings and e-mail exchanges and for guiding me to the right ideas. I am thankful for the opportunity to work with them. I also owe a debt of gratitude to Ames Laboratory, especially Diane DenAdel and my coworkers. It is only by their schedule flexibility, encouragement, and support that this work was possible. Last but not least, I would like to draw attention to the collaborative discussion between members of Dr. Wong's group, both inside and outside of group meetings, especially Dr. Natalia Stakhanova, Xia Wang, Fred Stanley, Ryan Babbitt, and Tanmoy Sakar. Without their ideas and contributions, this would be a very short document.

CHAPTER 1. INTRODUCTION

Information security is a challenging problem, and the extension of this challenge into computer security is as old as the computer itself. From physical access controls to file system permissions to network based security, mechanisms to secure digital data have evolved along with the progression of computer technology. The implication is that data which is worth processing, is worth stealing, manipulating, or destroying.

One of the mechanisms developed in response to the growth of information flow is the use of intrusion detection systems (IDS). Intrusion detection is the analysis of system activity or data to detect malicious behavior. A typical IDS will send an alert or log an event when anomalous activity is detected, providing a mechanism for an administrator to respond to the alert. These systems are deployed in various forms and inside environments, broadly including network based intrusion detection systems (NIDSs) and host-based intrusion detection systems (HIDSs).

NIDSs look at network traffic data and other centralized information sources to determine if an interaction between systems indicates that an attack is occurring. Snort is an example of a network intrusion detection system which has gained wide popularity in the past decade. Snort monitors network traffic in real time, analyzing and comparing packet data against a set of pre-defined detection rules. Packets which match enabled detection rules trigger a pre-defined action, typically an e-mail or log alert. NIDSs have a broad range of detection capabilities, including denial of service, SQL injection, port scanning, and other network based malicious activities.

Conversely, HIDSs focus on a host, monitoring file system, memory, user, and other host-based activities for indicators that the host is under attack or compromised. Open Source

Security (OSSEC) is a popular host based intrusion detection system. OSSEC can be installed on a variety of platforms and performs system log analysis, file system monitoring, and system configuration auditing. These data sources are compared against pre-defined rules and policies to detect malicious behavior and security policy violations.

In this context, we consider intrusion response as an action taken to prevent, contain, or stop an on-going attack. For both NIDSs and HIDSs, when an alert is received a system administrator must determine and implement an appropriate response. A typical example of this process is the generation of a Snort alert indicating a port scan from a certain IP address. Upon reviewing this alert, the system administrator may choose to take action, such as blocking that IP address via the network firewall. The delay from the time the activity is detected to the time the response is deployed can be significant.

Automated intrusion response is a mechanism to select and deploy a response from an automated source, i.e. without human intervention. The primary advantage of automated response is a reduction in the delay of response from the time of detection. Research along these lines includes investigation into real-time and preemptive response. A second advantage is the potential for more consistent and accurate responses across systems and organizations. A system administrator responding to an on-going attack may not always consider all the costs of taking an action, or may fail to consider an available response which would be less costly in the interest of meeting the threat.

In spite of these advantages, adoption of automated response has seen slow due to discomfort by both system administrators and system owners. Reasons for this include the complexity of implementing automated response models, perceived loss of control over systems, and the imprecision of available response selection methods. An informal survey of nine system administrators yielded a number of responders who indicated discomfort with employing automated response. However, as one responder to the survey noted, the proposed approach takes a balanced view of the situation, while their tendency is to over-react to address the incident. This indicates a need for standardized metrics and increased automation, even for experienced professionals.

The motivation for this work is based on the increasing need for real-time response, inadequacy of statically defined cyber protection, varying intuition displayed by system administrators in response selection, and the need for common metrics to advance the field of intrusion response research.

1.1 Intuitions for response selection

The traditional approach to intrusion response selection is based on an intuitive assessment of factors such as likelihood and severity of intrusion, extent of the potential intrusion damage, effectiveness of suitable response actions, expected duration of the response, etc. As this process is often conducted by the system administrator, it incorporates expert knowledge, and thus is generally accurate. However, being dependent on human expertise, this process relies on imprecise judgments creating inconsistency in intrusion handling among systems. In addition, response selection often introduces a significant delay in reacting to the failure, and thus is not appropriate in critical environments and unsuitable for automated response systems.

Intuitively, all these factors define the evaluation of intrusion response. However, they are rarely applied in evaluation of intrusion response component in the published literature mainly due to the lack of common interpretation and the absence of concrete metrics with clear conceptual meaning for measuring these factors. For example, while most of the existing models supporting automatic response selection introduce response cost as one of the factors in the selection of the suitable response strategy, they generally do not agree on what constitutes the response cost and how it can be measured. Some suggest that response cost includes the labor cost of personnel involved in response deployment and criticality of the attack [10], others see response cost as a measure of response effectiveness to a detected attack and its disruptiveness to legitimate users [6].

1.2 Increasing need for real time response

The proliferation of complex and fast-spreading intrusions against computer systems brought new requirements to intrusion detection and response, demanding not only advances in intru-

sion detection mechanisms but also the development of increasingly sophisticated and automated intrusion response systems.

The growing use of automation by attackers to spread attacks and damage systems has put an emphasis on the delay between the detection and mitigation of an intrusion. Small increments of time on a system can mean significant additional damage done by the attacker. For instance, the CodeRed worm was known for its rapid rate of infection [31]. One researcher, presenting a biological model for Internet worm propagation, concluded: "Because high-speed worms are no longer a theoretical threat, we need to automate worm defenses; there is no conceivable way for system administrators to respond to threats of this speed." [12]

1.3 Inadequacy of cyber protections

The majority of existing automatic intrusion response systems rely on the mapping of attacks to pre-defined responses [20, 21]. These approaches allow the system administrator to deal with intrusions faster, more efficiently, and effectively. However, they lack flexibility mainly because few of these systems take into account intrusion cost factors. A similar trend can be seen in the history of firewalls and intrusion detection systems.

Adaptive IDS trends. The early widespread adoption of intrusion detection was in the form of rule-based systems which triggered alerts when exact matches were identified. While these systems were easy to understand, control, and extend, attackers began to exploit the rigidness of a rule-based detection framework.

For example, a Snort rule is listed in figure 1.1 where the detection parameters consist of the following elements:

Source Network / Port: `$EXTERNAL_NET any`

Destination Network / Port: `$HOME_NET 910`

Flow: `flow:to_server,established`

Packet content: `"|10 23|Tg"; depth:4; isdataat:726,relative; content:!"|10 23|Tg"; within:712; distance:14`

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 910 (msg:"EXPLOIT DATAC RealWin SCADA
System FC_INFOTAG/SET_CONTROL buffer overflow attempt"; flow:to_server,established;
content:"|10 23|Tg"; depth:4; isdataat:726,relative; content:!"|10 23|Tg";
within:712; distance:14; metadata:policy balanced-ips drop, policy security-ips
drop; reference:bugtraq,31418; reference:cve,2008-4322; classtype:attempted-user;
sid:14769; rev:1;)
```

Figure 1.1 An example of a Snort rule.

This rule will check network traffic from any external host and port to any internal host on port 910 to see if the byte sequence ‘‘`|10 23|`’’ followed by characters `Tg` is seen. This sequence must occur within the first 4 bytes of the payload, and there must be at least 726 bytes of data following the payload. In addition, something not equal to the byte sequence ‘‘`|10 23|Tg`’’ appears no further than 712 bytes of the initial byte sequence, and there should be no more than 14 bytes after the last match.

An attacker, to evade detection, needs to vary the parameters of the attack outside of those specified in the rule. For instance, changing the location of the sequence ‘‘`|10 23|`’’ in the packet to after byte 4, or filling the payload with more than 726 bytes before listing something that is not ‘‘`|10 23|`’’. While in this case a new rule can be added to cover the change when it is detected, this approach requires constant updates to detect new attacks. Alternatively, the rule can be broadened and generalized, making the IDS less efficient, and increasing the number of false positives.

In response, the intrusion detection research community began to investigate anomaly based intrusion detection. These systems attempt to dynamically adapt to the data source they are monitoring. Instead of applying a fixed set of rules, anomaly based IDS learns characteristics of normal behavior and generates alerts when abnormal behavior is exhibited. For instance, if a network host suddenly begins exchanging data with a host that has never been contacted before at odd times of the day, an alert may be sent to system administrators to investigate the event. Anomaly based IDS is still a very active area of research.

Dynamic firewall development. Firewalls control network traffic flow by inspecting packet headers. They determine an action to take based on one or more header fields (most commonly some combination of source IP address, destination IP address, source port and destination port). In the context of the five layers of the Internet, firewalls traditionally operate at the TCP/IP layer, using information in the TCP/IP headers to make decisions about whether to permit packets through, drop packets, send reject messages, or perform other actions. Stateful firewalls were developed to address the need to track connection state data to determine how to handle a packet. For instance, packets from the Internet to a client which would normally be dropped are permitted if they are part of an established TCP session.

While these types of firewall rules are sufficient to prevent traffic which meets a set of statically defined parameters, the firewall must allow all traffic to those services which need to be accessed from a remote location.

Thus, since a firewall traditionally only examines packet header data when determining the policy to apply to a specific packet, data driven attacks against a remote public service cannot generally be differentiated from legitimate user traffic (i.e. SQL-injection, repeated authentication attempts). In some cases statically defined application-layer responses or event analysis can mitigate the problem, for instance monitoring the logs and blocking connections when more than X failed logins occur from a given source. Again, an attacker can bypass such mechanisms by varying the parameters of the attack. A system which blocks an IP address if five failed logons are seen within a 30 second interval, for instance, can be attacked at a rate of four logons every 30 seconds without being detected or blocked.

To address these limitations, technologies have been developed to allow users to self-open firewall holes by logging in to a web interface, incorporate additional decision factors such as time of day, or use application-data layer filtering which begins to overlap with intrusion detection and response at the system level.

As the limitations of statically defined security controls become apparent, development shifts to incorporate dynamic forms of both network port blocking (protocol specific state tracking) and intrusion detection (anomaly based IDS). This is mirrored in the recent directions

taken in the field of intrusion response research.

Development of cost-sensitive intrusion response. Similar to the examples above, the limitations of statically defined response approaches prompted a trend toward cost-sensitive modeling of response selection [10, 2, 6, 30, 22, 29]. The primary aim of applying such models is to balance intrusion damage and response cost to ensure an adequate response without sacrificing the normal functionality of the system under attack. However, defining accurate measurements of these cost factors and ensuring their consistent evaluation in a broad range of computing environments are common challenges in using a cost-sensitive modeling approach.

1.4 Performance metrics

A recent draft document published by NIST [26] called out the need for progress defining useful performance metrics for cyber security. Established fields of study in classical science relied on relative comparative assessments early on and developed quantitative systems of measurement as they matured. Longer and shorter became meters, warmer and colder became degrees.

The need for such metrics is not recent, and significant efforts have been made toward quantifying aspects of intrusion damage and response cost. However, the body of existing work in this area does not address the timing sensitivity of automated intrusion response, ensure that an approach can be reasonably implemented by a system administrator, or attempt to apply cost-sensitive intrusion response to a broad range of environments.

One of the principle applications of good metrics for intrusion damage and response cost is the ability to automatically choose the least costly response in time to minimize the damage caused by an attack. While much of the work that has been pursued to define these metrics focuses strongly on accurate quantification, this also leads to models in which the actual computation of the response cost is time consuming to perform.

In addition to the complexity of the computation, another common problem among current approaches to intrusion detection and response cost assessment is the amount of data-gathering

required to use the model. Several promising approaches require building intricate dependency graphs or detailed system models which require prohibitive amounts of time and expertise in various aspects of the systems.

In a few cases (i.e. ADEPTS), the model can be implemented and demonstrated to work in specific domains, but an adaptive, general-case extension of the work is not immediately obvious.

1.5 Proposed Solution

Automated response systems are divided into three basic categories:

Static Responses are fixed to specific intrusions and are deployed regardless of the environment or past history of the response.

Dynamic Responses are deployed based on parameters which change over time, for instance, the past success of the response or the predicted goals of the attacker.

Cost Sensitive Responses are evaluated in the context of the system environment, organizational goals, detected attacks to estimate the negative and positive impact. The response is selected with the lowest overall cost.

This thesis presents a host-based framework for cost-sensitive intrusion response selection. A set of measurements to characterize the potential costs associated with the intrusion handling process is introduced, along with a method for evaluating each intrusion response with respect to the risk of potential intrusion damage, effectiveness of response action and response cost for a system. The overall goal is to address current roadblocks to automated response implementation, and define a metric-based framework which can be directly and universally applied to arbitrary systems with a minimal administrative burden.

The framework is composed of a set of formulas to compute the values and costs associated with the responses and intrusions, coupled with a methodology for enumerating the services provided by a system, assigning value to those services, enumerating the responses and intrusions under consideration, and combining these into an overall data store to support the

computations. The primary tradeoff made in this approach is the precision of quantification for implementability and computational efficiency.

This tradeoff is based on the observation that often a system administrator is not the primary decision maker in an organization, and may not be qualified to precisely assess the value of a system or services. However, system administrators are typically very familiar with the primary services they support, and have a better intuition about service dependency and the effect that specific actions may have on organizational resources. This familiarity is leveraged by abstracting the services provided, and directing the implementers of the IRS to focus on the broad impact of intrusions and responses to the confidentiality, integrity, and availability of these services.

A degree of abstraction is employed when defining a system, and the resources available on that system. This ensures that the framework is applicable across a wide variety of environments. Even systems in one organization may have dramatically different security policies. For instance, a public web server for informational purposes may only value availability and integrity, but not confidentiality, whereas a database system with social security numbers stored in it may have the emphasis on confidentiality and integrity, but can tolerate reasonable losses in availability. By keeping these definitions abstract, the framework can be adapted to applications, hosts, or networks of hosts.

Together with this abstraction, the use of system value as a common basis for computing response cost allows comparisons to be made between these costs even on significantly different systems. The pervasive assumption is that the value of a system is equal to the sum of the values of the services provided by the system. This assumption is trivially valid by ensuring that all resources which comprise the value of the system are included.

In addition to costs associated with the direct effects of intrusions and responses, administrative costs are accounted for which are easily overlooked when automating response selection. For example, operational costs for both intrusions (e.g. the reporting cost incurred by a successful intrusion) and responses (e.g. the cost of a person needing to undo the response action, or the resources consumed to deploy the response) are incorporated into this model.

The result is a framework specifically developed to make automated response accessible to system administrators regardless of security background. The only assumption is a familiarity with the systems they administer, and access to a consistent valuation method for each system. While a degree of input data is required, the thorough system analysis and detailed enumeration of non-essential resources required in other frameworks is avoided while maintaining sufficient quantification precision to ensure that the response with the highest expected value is selected consistently.

1.6 Contributions

The main contributions of this thesis can be summarized as follows:

1. **Separate the security policy from the cost model.** The proposed evaluation metrics are defined in terms of system resources and overall system value, which brings a common ground to the selection process. By abstracting and separating the security policy and relative values from the overall cost model, this framework can be directly applied to different environment settings. This approach provides a generalized assessment mechanism that allows the analysis of response measures for any attack in the context of the security policies of the given system.
2. **Definitions of intrusion and response cost measures.** The presented cost measures allow a full assessment of attack handling processes, considering not only direct damage caused by the intrusion but also indirect costs that often remain hidden during cost computation.
3. **Development of a system aware adaptable model.** The proposed model incorporates adaptive components based on observed behaviors of deployed responses. Those which are unsuccessful in the past are less likely to be selected in the future, while those which are successful are more likely to be selected.
4. **An implementable framework.** This work provides a step-by-step assessment process that allows system administrators with broad range of skill level to employ the approach in their daily practice. This is demonstrated by:

(a) The implementation of a full simulation as a Java Applet.

(b) A host-based Linux response system implemented and demonstrated in C.

1.7 Roadmap

The remainder of this work is organized as follows: In Chapter 2 we present a review of related work in this research area. Then we present our framework and detail the selection process in Chapter 3. Chapter 4 describes of the implementations using this framework, and presents performance results which demonstrate the correctness, intuitivity, and usefulness of this approach. Finally, we conclude with a discussion and opportunities for future development in Chapter 5.

CHAPTER 2. RELATED WORK

A number of techniques aimed at enhancing intrusion response automation have been proposed and deployed, and a comprehensive review of this research work is given in taxonomies by Natalia Stakhanova [23], and Bingrui Foo [5].

There are several categories into which intrusion response methods can be divided, but two of the more useful classes are:

Degree of Automation Notification only, automatic selection and manual deployment, or fully automated deployment.

Activity of Triggered Response Passive, including notification e-mails and alerts, or Active, including blocking network traffic, shutting down services, and other changes to system behavior.

Automated response can then be further subdivided along dimensions of:

Ability to adjust Static responses do not change over time, while adaptive responses adjust behavior based on observable changes in system state.

Time of response Proactive responses attempt to prevent the intrusion from succeeding, requiring short delays from time of detection. Delayed responses typically include actions which limit or mitigate intrusion damage, and are less sensitive to response delay.

Cooperation ability Autonomous response selection occurs independently of other systems, agents, or observed events not pertaining to the intrusion. Cooperative response selection is negotiated between two or more response selection agents.

Response selection method Static mapping is a method of response selection where the response action for a given alert is fixed, regardless of other factors. Dynamic mapping

provides flexibility by allowing the response selection engine to choose different responses depending on the system state. Cost sensitive response selection is a recent trend which attempts to minimize the cost of the response while still adequately addressing the intrusion.

The general body of research related to Intrusion Response will be presented here with a focus on automated dynamic responses. First the significant works related to automated response in general are presented, followed by a more detailed treatment of cost-sensitive automated response. Finally, a summary of contributions specifically to the area of response selection cost metrics are presented.

2.1 Automated Dynamic Response

Automated intrusion response has been an active area of both research and development efforts for several decades. A representative set of the major efforts toward automated response are presented here, excluding the body of work for cost-sensitive response which will be presented in section 2.2.

Cooperating Security Managers (CSM) [28] investigated the use of cooperating detection and response systems in a distributed and scalable architecture. They proposed an architecture by which distributed intrusion detection and response agents could operate on a host-basis, and share information with each other. One novelty to this work is the fact that no central coordinating authority is required. Agents are installed and run on individual host, and consist of multiple modules including a user tracking module, a coordination module, and an intrusion handling module.

The intrusion handling module is the component which is responsible for deploying responses. A response hierarchy is constructed as an ordering over the severity of the detected intrusion. The more severe the intrusion is deemed to be by the system, the more drastic the deployed response. While effectively a fixed mapping of responses to intrusion severities, this dynamic response system could be considered as an early form of cost-sensitive response selection in light of the consideration of user interference when determining the responses.

Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD, [13]) was developed with the goal of providing highly accurate detection, and an easily extendable architecture. The implementation consists of three major components: the Profiler, the Signature, and the Resolver. These components represent anomaly based detection, signature based detection, and automated response capabilities respectively. The Resolver component determines whether to respond to an attack based on the severity of the intrusion and the confidence that the alert is a true positive.

Another intrusion response research project, the Adaptive Agent-based Intrusion Response Systems (AAIRS) [15], seeks to address intrusion response in terms of formulating a high-level response plan, and then automatically deriving a set of actions from that plan. This work is from the same group that developed CSM, and the response selection is also based on intrusion severity with the additional component of past success rate.

The Alphatec Light Autonomic Defense System (αLADS) [1] is an automated response system which uses partially observable Markov decision processors to implement a stochastic feedback control system. While some of the details of this system are hidden due to the proprietary nature of the project, it is focused on survivable systems, allowing significant resource losses before a system is considered non-functional.

The test prototype presented in [1] was equipped with four possible responses: Observe, Notify, Kill, and Halt. These four responses were chosen partly to limit system complexity for the sake of analysis, and other responses can be added. Though the stochastic nature of the controller complicates analysis, it is significant that this system was able to halt not only anomalous behavior which it had been trained to detected, but also novel anomalies (of a similar nature). The αLADS system also proves to be very tolerant of false positives, not responding to anomalies caused through normal operation of the system.

While the αLADS system impressively accommodates false positives, and employs responses which kill processes to stop intrusions, selecting response alternatives based on a real-time cost assessment is not a part of this work. Essentially a dynamic selection of response is performed based on the detected anomalous behavior.

The Generic Authorization and Access Control API (GAA-API) [18, 19] is a modular framework for adapting existing applications to use a finely grained authorization and access control mechanism. This framework uses a simple policy language (Extended Access Control Lists, EACL) to define conditions under which a user is permitted to authenticate to an application. A three-tier authorization model is used for a request:

1. Pre-conditions are checked prior to allowing access to the object.

2. Mid-conditions are periodically (on an application-specified basis) tested during access to detect suspicious activity.

3. Post-conditions are activated upon successful completion of the authenticated action.

GAA-API has been successfully integrated into a variety of applications, including Secure Shell (SSH), the Free-S/WAN IPSec implementation for Linux, and the Apache web server.

A significant contribution of this work is the inclusion of adaptive policies for authorization and access control. As opposed to standard authentication systems which have a static policy defined in advance, GAA-API determines the policy dynamically, based on system state during the processing of the authorization request and subsequent actions. While other dynamic policy systems of this type exist, they typically either require policy switching (using different disjoint sets of policy requirements depending on some trigger, such a time of day) or changing the algorithms used to compute the policy. In GAA-API, by contrast, monitors system state and uses the current state to trigger policy changes.

Snort inline [11, 14] is a plugin for the Snort IDS which allows actions using iptables (the primary Linux firewall) and bpf (the primary BSD firewall). Whereas a typical Snort rule will generate an alert in response to a signature trigger, Snort inline will allow an administrator to define actions such as dropping the packet, dropping and logging, or rejecting the connection (either through a TCP reset, or an ICMP-HOST-UNREACHABLE packet).

This implementation is popular due to the wide deployment of Snort as an IDS, however it has the same problems that Snort has, namely a very significant degree of tuning required

to avoid prohibitive numbers of false positives, and the reactivity implied by a signature-based detection system.

Recent open-source work on integrating Snort with iptables has also developed in a more dynamic sense. Two tools developed as part of this effort are psad [17] and fwsnort [16].

Psad is based upon the Bastille-NIDS project, and adapts Snort rules to match against log messages produced from the iptables firewall on Linux. Because it utilizes iptables logs, no additional packet monitoring libraries are required, and high-overhead deep packet inspection is not needed. Actions can then be taken in iptables to block traffic, reset the session. Psad effectively implements a large subset of the Snort_inline functionality with little overhead.

Fwsnort is a related tool which parses Snort signature files, and implements corresponding iptables rules which can either log the packet (i.e. simulate Snort behavior), or drop, reject, or take some other response measure (i.e. simulate Snort_inline behavior).

Together these two tools allow a system to leverage most of the Snort framework's detection and response capabilities using only iptables and the netfilter string match library. Similarly to Snort_inline, however, these tools suffer from the static signature basis and the assumption of pre-existing rules to detect malicious behavior.

In the past decade a pantheon of automated intrusion response tools have been developed both as research projects and as commercial ventures. Many popular antivirus, host-based intrusion detection, network-based intrusion detection, log analysis, and security information management products include some form of active response capability. However, very few of these include a cost-sensitive dynamic response capability as presented in this thesis.

2.2 Cost Sensitive Response

Compared to automated response in general, the area of response cost assessment has received considerably less attention. A number of significant contributions to this area do stand out, however, particularly in the past decade.

Work by Wenke Lee et. al. [10] directly addresses the cost of deploying responses. This work introduces a cost-benefit measure which incorporates multiple dimensions of cost in the face of

an intrusion: response cost, damage cost, and operational cost. The inclusion of operational cost in particular is a significant one in that often indirect costs associated with an intrusion are overlooked in favor of the direct damage costs. However, indirect considerations such as loss of reputation, reporting requirements, and mandatory customer notifications are often drivers for preventing intrusions which otherwise cause little or no direct damage.

Their work employs an attack taxonomy which groups attacks by several categories including general technical method, whether the attack execution is local or remote, a single event or multiple events, or whether attempts are made to conceal attack activity. Points are assigned based on the criticality of resources and the degree of damage (*lethality*) caused by an attack. The damage cost of an attack is then defined to be $criticality \times base_D$, where $base_D$ is the lethality. A progress metric indicating how close the attack is to being successful is also used to scale the cost. We extend this idea to apply these cost assessment features to both intrusions and responses, combined with a framework to guide system administrators in performing consistent evaluations of the cost factors.

Another approach, Adaptive Intrusion Response using Attack Graphs (ADEPTS), proposed by Foo et al. [6], employs attack graphs to identify the actions required to achieve possible attack targets in a distributed system, and consequently, to show the objectives of suitable responses. Attacker goals are expressed as end states in the attack graph with intermediate steps leading to the fulfillment of those goals. Responses are selected to frustrate attack goals based on *effectiveness* to that particular attack in the past, *disruptiveness* to legitimate users and the *confidence level* which indicates the probability that the attack is actually taking place.

Models proposed by Toth and Kruegel [24], Balepin et al. [2] and Jahnke [9] not only consider costs and benefits of the responses, but also introduce a link between the cost of responses and the system resources in the network.

The approach proposed by Toth and Kruegel is a network-based response mechanism that employs system resources as the building block of their cost model. Including system users, network topology, and security control information, their work constructs a network depen-

dency tree which is used to evaluate the impact that response and intrusion events will have on the overall system.

The proposed cost function (*impact evaluation function*) incorporates the direct effect of an event on nodes in the resource tree as well as indirect effects propagated through the dependency relationships between nodes. To measure the degree of damage, a capability index metric is introduced. For nodes which have no dependencies, this index is a binary value indicating functionality (1.0), or non-functionality (0.0). In the case where dependencies exist, a recursive algorithm is employed to compute the capability after an event's effects are taken into account. This algorithm considers the and/or relationships in the dependency graph, and yields an index in the range $[0, 1]$.

When selecting a response, the effects of each response are applied to the model, and the resulting capability difference is used. The response which yields the greatest increase in capability is the one which is chosen. In spite of requiring complex evaluations of the graph structure, the performance results are very fast using optimized data structures for locally optimal responses. As noted in their paper, finding a globally optimized response is much more difficult and time consuming. The primary weaknesses of this approach are the complexity involved in establishing the dependency graph, and the course granularity of the resource model (i.e., only availability is taken into account, ignoring potential effects on integrity and confidentiality).

An approach to host-based intrusion detection and response has been proposed by Balepin et al. [2]. The local resource hierarchy is modeled as a directed graph where the nodes represent specific system resources and the edges are the dependencies between them. Each node is associated with a list of responses that can be applied to restore working state of resource in case of an attack. A particular response for a node is selected based on (a) *the cost of the response*, the sum of the resources that will be affected by the response action, (b) *the benefit of the response*, the sum of the nodes, previously affected by intrusion that will be restored to working state, and (c) *the cost of the resource*, the quantification of the importance of the resource. This work constructs a resource-based gain matrix to determine the relative value

of a response. The gain matrix incorporates information about the current system state, the resources affected by an attack, and the resources protected by a response. These are then combined using a probabilistic approach to determine the response which is most likely to yield a high payoff.

While accounting for many of the important factors, and using probability to represent the uncertainty in any intrusion scenario, the complexity of enumerating the possible system states and assigning probabilities to them make the implementation of this method challenging. However, one insight which is incorporated in this thesis is the use of resource ordering in order to help administrators consistently value resources. That idea is extended in this thesis to also aid in assessing response and intrusion impacts on resources.

A similar approach proposed by Jahnke [9] also attempts to quantify response measures based on modeling system resources as a dependency graph, applying this methodology to networks of systems. Although this method requires careful graph construction and validation, it allows automatic assessment of the *response success* computed through the change of the availability of resource nodes in the graph and *required effort or cost* defined as the amount of instances to be modified for deploying the selected response.

One common problem with these models is the lack of consistency. Each model approaches response cost evaluation and selection from a different perspective. Foo et al. [6] measures the response effectiveness against a detected attack based on the past experience. Lee et al. [10] relates the response cost to the required labor efforts. The works by [24], [2] and [9] consider response cost in association with the system resources, but offering varying evaluation methods. [2] measures the response cost as the sum of manually assigned costs of affected resources. [24] calculates response cost as a function of system capability reduction, while [9] essentially extends the idea in [24] by adding a fine-grained quantification of system resource unavailability.

In spite of inconsistencies, however, the emerging theme in these projects effectively establishes the idea of employing system resources in evaluation of intrusion response, and based on this promising trend we build our approach to response selection with the resources of the

system. In this context, our model can be viewed as a generalization of the existing approaches.

2.3 Response Cost Metrics

In the development of cost-sensitive intrusion response systems, some method of measuring or assessing the cost of a response (or the closely related metric of value) must be developed. A summary of recent efforts to define a cost metric is given in this section.

Early work in automated response introduced simple measures by which the desirability of different levels of response could be expressed. CSM [28] considered the severity of the detected intrusion, based on the intuition that more severe intrusions warranted sharper responses. Emerald [13] added the use of intrusion confidence, using anomaly and signature based detectors to determine the likelihood of an attack issuing an alert. The AAIRS project [15], also from the Texas A&M group, contributed the idea of adapting future response deployment to its history of success in preventing an intrusion.

αLADS [1] was a very early attempt to account for the effect a response would have on the system, choosing responses based on the likelihood of resulting in a better system state. Toth and Kruegel [24] expand on this notion and develop a measure of response negative impact, recommending that the response with the minimum negative impact be selected. Wenke Lee [10] introduced the idea of considering responses with respect to the specific intrusion context, using an intrusion taxonomy to address unknown intrusions.

Ivan Balepin, during the development of the Automated Response Broker project [2], contributed the use of system state estimation in evaluating not only the anticipated impact of an attack, but also the expected benefit of a response. This work also uses the notion of establishing an ordering over resources to allow system administrators to assign reasonable values to resource impacts. The ADEPTs work [6] is the first to explicitly use the equation $\texttt{RI} = \texttt{EI} - \texttt{DI}$ (Response Impact = Effectiveness Index - Damage Index) and demonstrated its successful application to the web-based e-commerce domain.

A high-level conceptual view of risk assessment for intrusion response has been introduced by [8]. The proposed framework essentially presents the perceived risk of system exposure to

intrusions as variable that can be guided by tolerable level of risk threshold. In this context, selecting an effective response can be viewed as choosing the countermeasure that can restore the level of exposure to the tolerable risk level. While the proposed formal model lays down several theoretical properties of the response selection, it primarily addresses the change in permission as a response measure, thus limiting its applicability to large pool of intrusion response systems. On the other hand, while providing high-level concepts of benefit and cost, the model fails to address what constitutes these metrics, limiting the implementability of the approach.

Recent work by Marko Jahnke [9] leverages sophisticated dependency maps to enable the assessment of cost in terms of availability, and calls out the need for additional assessment frameworks for intrusion and confidentiality. Finally, the work by Shiau-Huey Wang et al. [25] on assessing impact in Mobile Ad Hoc networks demonstrates the use of domain specific impact calculations and leads to the intuition about combining such metrics using a common framework.

Our approach. This work is focused on the assessment of response cost to support the selection of autonomous, proactive, and fully automated responses. The framework in general, however, can also be applied to passive or delayed responses, and may be extensible to include cooperative response selection. In addition, this framework incorporates practical metrics which can be employed by security and system administrators, and consists of well defined quantifications for cost metric components.

CHAPTER 3. SYSTEM METHODOLOGY

As seen in chapter 2 there are a number of factors which contribute to the intuition behind response selection. As the work in this area has matured, some preliminary ideas for a response cost metric have emerged:

1. The cost of the intrusion, which can be further broken down into:
   (a) The damage this intrusion causes to the system.
   (b) The operational costs associated with handling the intrusion, such as mandatory reporting or loss of reputation.
2. The cost of the response, which can also be subdivided:
   (a) The damage this response causes to the system.
   (b) The operational costs associated with deploying this response, such as system administrator time and additional computing resources required.
3. The likelihood that a response will be able to prevent a given intrusion.
4. The potential intrusion damage the response will prevent.
5. The likelihood that this intrusion is actually occurring, composed of:
   (a) The confidence that the alert is a true positive, e.g. that it is not a false alarm.
   (b) The specificity of the alert in indicating this exact intrusion, as opposed to one of a set of intrusions out of which this is one alternative.

While these quantities are easily enumerable, the form of their combination into a workable metric is not obvious. In addition, the subjective nature of some of these factors requires a careful process to achieve consistent quantification.

To address this issue, the next several sections will develop the quantification formula in a top-down approach, first establishing high level equations, and then decomposing those equations into lower level computations. We end this chapter with a discussion of how to effectively assign values to the atomic quantities which drive this model, i.e. the impact of a response on a particular resource, or the confidentiality value of a system.

We assume for this discussion that the organization has established a security policy which applies to each system, and which allows the value of a system to the organization to be expressed in terms of confidentiality, integrity, and availability (C, I, A). In this context, we consider the confidentiality, integrity, or availability of a system or a service to be the degree to which that system or service is able to preserve those qualities.

3.1 Computing the expected response value.

First an intuitive concept of the value of a response is formed. We denote the expected value of a response, `EV`, expected benefit of a response `RB`, and response cost `RC` and begin with equation 3.1

$$\texttt{EV} = \texttt{RB} - \texttt{RC} \tag{3.1}$$

This equation establishes that the cost value should be some measure of the response benefit, discounted by some measure of the response cost. For two responses with an equal cost then, we want to choose that which provides the largest benefit. For two responses with an equal benefit, conversely, choose the one with the lowest cost. Therefore, higher values of EV(r) indicate responses which are more desirable to deploy.

If one assumes that the values of `RB` and `RC` are in the range $[0, 1]$ and have the same scale, this formula has the additional property that positive values indicate responses which have more benefit than they cost, and negative values indicate responses which will do more harm than good.

3.2 Computing the response cost.

The cost of the response, as indicated above, includes evaluating the damage the response will cause to a system, and determining the operational costs of deploying the response. Assuming again that these costs are on the same scale and have normalized values in the range $[0, 1]$, the total cost is simply the sum of the damage to the system and the operational cost associated with the response. However, the cost of the response is dependent on the system on which it is applied. In addition, to retain the property that the range of this value is $[0, 1]$, we need to normalize the value by 2 as both terms have ranges $[0, 1]$. Given a response r, a system s, an impact function $\texttt{SI}(r, s)$, and an operational cost value $\texttt{OC}(r, s)$, we define the response cost function $\texttt{RC}(r, s)$ in equation 3.2.

$$\texttt{RC}(r, s) = \frac{\texttt{SI}(r, s) + \texttt{OC}(r, s)}{2} \tag{3.2}$$

To decompose this formula further, a formal definition of a system is established, and then the functions $\texttt{SI}(r, s)$ and $\texttt{OC}(r, s)$ are discussed.

3.2.1 Formal definition of system.

To provide a practical level of abstraction, we consider a system to be a composition of resources, which is the set of services provided by the system which contribute to the system's value. We overload the notation of a system to also denote the set of resources in a system s. Specifically, for a set of resources in a system, $\{R \in s\}$, and an assigned value of the system to an organization v_s, the sum of the individual values of those resources is equal to the system value: $v_s = \sum_{R_i \in s} v_{R_i}$. Note that this value may be expressed either in a static quantity, (e.g. dollars), or in quantity per unit time (e.g. dollars per minute).

Returning to the system policy in terms of confidentiality, integrity, and availability (C, I, A), this policy can be used as a set of weights with range $[0, 1]$ which, when applied to the system value, decompose it into these three dimensions. Formally, we define a system security policy as $SP_s = (w_C, w_I, w_A)$ such that $0 \leq w_{\{C,I,A\}} \leq 1$.

Finally, we assign each resource $R_i \in s$ an impact along each of these dimensions: $0 \leq \texttt{Impact}(R_i, X) \leq 1, X \in \{C, I, A\}$ where a value of 0 indicates that a compromise of the resource will have no affect on that security dimension, and a 1 indicates that a compromise of the resource would completely impair it.

3.2.2 Decomposing $\texttt{SI}(r, s)$

The system impact function is defined in terms of the system resources and the system security policy as discussed above. Intuitively we need to consider the impact of a response on the set of system resources, with resources having a greater impact on the value of the system receiving more consideration.

Remembering that each value $w_X, X \in \{C, I, A\}$ is between 0 and 1 and for each resource, $\texttt{Impact}(R_{i,X})$ is also between 0 and 1, we can define the product of these weights $(\texttt{Impact}(R_{i,X}) \times w_X)$ as the overall impact of resource R_i's security aspect X on the value of the system. A value of 1 for this product indicates a complete loss in system value if that aspect of the resource is lost, and a value of 0 indicates no value impact at all.

This definition is formalized in Equation 3.3, where $\texttt{Impact}(R_i, X)$ is the impact of resource R_i on security goal $X \in \{C, I, A\}$ as defined above and $\texttt{Damage}_{r,R_i}$ is an administrator defined value indicating the degree of negative impact a response r has on resource R.

$$\texttt{SI}(r, s) = \sum_{R \in s} \sum_{X \in \{C,I,A\}} \texttt{Damage}_{r,R_X} \times \texttt{Impact}(R, X) \times w_X \tag{3.3}$$

In order to scale this value to the range $[0, 1]$, a factor must be used to normalize $\texttt{SI}(r, s)$. Noting that the quantity is unit-less, choosing the maximum possible impact of a response (i.e. if all system resources were completely damaged), will yield a value with the desired range $[0, 1]$.

3.2.3 Decomposing $\texttt{OC}(r, s)$

Compared to $\texttt{SI}(r, s)$, the operational cost function $\texttt{OC}(r, s)$ is fairly simple. Examples of the types of costs which should be included in this value are:

1. Man-hours of labor required to deploy or manage the response.
2. Direct costs associated with a response, for instance, paying a fee to a third-party.
3. Additional resources used to support a response, such as disk-space or network bandwidth for additional logging.

Because the operational cost is derived from factors which include organization and system dependent values, this cost is directly assigned by the organization. A method for the consistent determination of these cost factors will be discussed further in section 3.5

The formal definition, equation 3.4, consists of the organization-assigned system value, v_s, and the organization assigned associated cost. It is assumed that, if the operational cost alone of a response is greater than the value of the system, the response will not be considered for deployment. Therefore we assert that $0 \leq \texttt{OC}(r, s) \leq 1$.

$$\texttt{OC}(r, s) = \frac{\texttt{Associated Cost}}{v_s} \tag{3.4}$$

3.3 Computing the intrusion cost.

The method to calculate the intrusion cost is identical to that of the response cost. The only difference is that, where the operational cost of a response is not assumed to exceed the system value, and thus should fall between 0 and 1 after normalization, no such assumption is made for the operational cost of an intrusion. Thus it is possible for the operational cost of intrusion q, $\texttt{OC}(q, s)$, to be greater than 1.

The formal definition of $\texttt{IC}(q, s)$ is defined in equation 3.5, and the equation components $\texttt{SI}(q, s)$ and $\texttt{OC}(q, s)$ are completely analagous to equations 3.3 and 3.4, respectively. The only significant difference is that, given the unbounded operational cost component, the range for this function is $0 \leq \texttt{IC}(q, s)$. However, the interpretation still stands that, e.g. $\texttt{IC}(q, s) = 2$ indicates an intrusion cost of twice the approximated system value.

$$\texttt{IC}(q, s) = \texttt{SI}(q, s) + \texttt{OC}(q, s) \tag{3.5}$$

While we have defined a metric for $\texttt{IC}(q, s)$ giving the expected cost of intrusion q on system s, the overall expected value of a response depends separately on the components of IC, and not on the quantity as a whole. Thus the notation $\texttt{IC}(q, s)$ will not appear in the equations below, however $\texttt{SI}(q, s)$ and $\texttt{OC}(q, s)$ appear separately.

3.4 Computing the response benefit.

The benefit of a response is the expected reduction in cost resulting from the response's ability to prevent some or all of the damage resulting from an intrusion. This benefit computation should take into account both the resources protected by the response should the intrusion continue to progress, and the cost reduction if the intrusion is prevented entirely.

For instance, a response which moves sensitive files to a more secure system may not stop the intrusion, but may prevent the cost associated with the disclosure of those files. Thus the system impact will be reduced, but the operational costs associated with the intrusion will not. However, an alternative response which effectively blocks the attack will both prevent the system damage and the operational cost.

Therefore it is desirable that the benefit computation express how much protection the response will provide to system resources and the probability estimate that the response will prevent the intrusion. To accommodate this, response benefit is defined in equation 3.6 as a function of the intrusion cost components, modified by the expected reduction in both system impact and operational cost of that intrusion if this response is deployed. In this equation we denote the set of possible intrusions occurring as Q.

$$\texttt{RB}(r, s) = \sum_{q \in Q} \texttt{SIReduction}(r, \texttt{SI}(q, s)) + \texttt{OCReduction}(r, \texttt{OC}(q, s)) \tag{3.6}$$

As stated above the two reduction components are computed separately. The first determines the expected reduction in system damage from the response, and is defined in equation 3.7. The second estimates the probability that the intrusion will be prevented entirely, giving the expected reduction in operational cost. This is presented in equation 3.8.

$$\texttt{SIReduction}(r, q, s) = \texttt{Prob}(q) \times \texttt{Cov}(r, q, s) \tag{3.7}$$

$$\texttt{OCReduction}(r, q, s) = \texttt{Prob}(q) \times \texttt{SF}(r, q, s) \times \texttt{OC}(q, s) \tag{3.8}$$

Equations 3.7 and 3.8 share the probability that this intrusion is actually occurring $\texttt{Prob}(q)$ $\texttt{Prob}(q)$ is a function of the indicators of q and is assigned by a system administrator. A methodological approach to assigning this value is mentioned in section 3.5.

The reduction in system impact is a function of the resource aspects protected by the response. For instance, if an intrusion will damage the availability of a web server process, and a response will protect the availability of that process, the response reduces the system damage of the intrusion.

However, the probability that the attack will be entirely thwarted does not necessarily depend on how much damage is prevented. Often there is a key component to an intrusion which must exist for that intrusion to succeed. Thus, the more interference a response causes to an intrusion, the more likely it is to prevent the intrusion from succeeding. To address this, only the resource security facet overlap ($\texttt{SF}(r, q, s)$), and not coverage, is considered in the estimate of this value.

Providing an accurate estimate of $\texttt{SF}(r, q, s)$, while dependent on the interference of the response, also includes a probability component drawn from past history. Regardless of the estimated chance of success, the most accurate information regarding the probability of future success is how well the response has performed previously. Thus we need to not only consider the estimated probability of success based on a resource analysis, but also account, in proportion to the statistical validity of the quantity, for the degree of success in past deployments.

We define the notation $\texttt{Impact}(q, R_{i,X})$ to indicate that intrusion q damages security aspect X of resource R_i, and similarly the degree of protection provided to a security facet of a resource by a response is denoted by $\texttt{Prot}(r, R_{i,X})$. The value $r_{\texttt{successful}}$ is the number of times this response has been successful in the past, and r_n is the total number of times this response has been deployed in the past. When $r_n = 0$, we consider $1/r_n = 1$. One problem in incorporating

this factor is the potential for response effectiveness to change over time, but for the model to keep the response "stuck" with a poor history. Possible techniques to address this problem include the use of a sliding window or periodic resets of r_n and $r_{\texttt{successful}}$.

In the $\texttt{SF}(r, q, s)$ equation, Equation 3.9, we look at the expected protection offered by this response against this intrusion for all resource security facets as an apriori estimate of the expected response success. As the response is tested through deployment, the historical success rate is accumulated, and that value is included with a weight proportional to the amount of historical data available.

$$\texttt{SF}(r, q, s) = \left(\frac{|R_{i,X \in \{C,I,A\}} \texttt{ protected by } r|}{|R_{i,X \in \{C,I,A\}} \texttt{ impacted by } q|} \right) \times (1/r_n) + \frac{r_{\texttt{Success}}}{r_n} \times (1 - 1/r_n) \quad (3.9)$$

Coverage is computed similarly to success factor, but with two important differences:

1. The success factor only accounts for the resources which are protected by a response, but does not consider the actual cost reduction of the response considered in the coverage computation.

2. The success factor includes a historical component. As the response is deployed, the success rate of the response is tallied and this information is used to adjust the success factor as appropriate. The coverage does not include such a component.

Coverage expresses the degree of expected protection a response will afford against a given intrusion. This is subtly different from the expectation of success; in this case we are only interested in the degree to which the resource security facets will be protected by a response, not the likelihood that the intrusion will be entirely prevented. A formalized notion of coverage is given in Equation 3.10.

$$\texttt{Cov}(r, q, s) = \sum_{R_i \in s_q} \sum_{X \in \{C,I,A\}} \texttt{MAX}(0, (\texttt{Prot}_{r,R_{i,X}} - \texttt{Impact}_{r,R_{i,X}})) \times \texttt{Impact}(R_i, X)) \times w_X \quad (3.10)$$

Coverage is based on the sum of the impact of an intrusion against each resource facet weighted by the difference between the protection offered that resource facet by the response

and the negative impact of the response on that same resource facet. Each element of the sum is weighted by the overall importance of that security facet to the system as a whole.

The coverage computation introduces the somewhat counterintuitive concept of a resource both protecting and damaging the same facet of a resource. One example of this might be the case when a network service is being attacked by a distributed denial of service (DDOS). In this case, the response *block all network traffic* may protect the availability of the resource by stopping the DDOS, but at the same time it has a severe negative impact on the availability because no network clients can access the resource any more. Therefore, the net contribution to $\texttt{Cov}(r, q, s)$ of that response on that resource should be close to 0.

3.5 Methodology for gathering data:

The equations developed above compute the expected value of a response, derived from the following basic quantities:

1. The overall value of the system to an organization.
2. The security policy of the system, in terms of confidentiality, integrity, and availability.
3. The enumerated resources in the system, and their potential to impact confidentiality, integrity, and availability on that system.
4. The set of defined intrusion alerts, representing known threats to the system. Each element of this set has an associated operational cost, and a vector denoting the impact of each intrusion on each resource security aspect.
5. The set of defined responses, representing actions the response system can take. Each element of this set has an associated operational cost, a vector denoting the impact of each response on each resource security aspect, and a vector denoting the degree of protection conferred upon each resource security aspect.
6. The probability that a specific intrusion is actually occurring, given its associated intrusion alert.

Some of these quantities are quite subjective in nature, and in order to achieve a consistent evaluation of intrusion cost, response cost, and response value across systems, a concrete

methodology needs to be established for determining these subjective values.

The goal is not to ensure that precisely the same value will be produced by two different administrators working under the same conditions, but that both administrators, having applied this methodology, will be confident that the resulting values are reasonable based on their independent assessments of the responses, intrusions, and systems.

We will illustrate the methodology mentioned here through the use of an ongoing example: a public web hosting server.

3.5.1 Determining the system value.

The system value is a quantification of its importance to the system owner. Only the system owner can define this value, and it will necessarily vary between environments. As mentioned above, this value has a unit of either absolute economic value (e.g. dollars) or economic value per unit time (e.g. dollars per minute). There is a significant body of work related to the problem of assessing a system's value, some of which overlaps significantly with the fields of risk assessment and cost-benefit analysis of security controls [27].

To demonstrate this methodology, we will maintain a running example of a system valued at $\$20,000$.

3.5.2 Assigning the security policy.

The security policy consists of a set of values, one each for confidentiality, integrity, and availability. These values are weights in the range $[0, 1]$, and indicate how much of the system value is attributed to that security aspect. Setting these values is a matter of determining how much of the system value would be lost with the loss of one of these security goals.

For the example web server, users are advised that minimal effort is made to assure confidentiality, as pages posted are assumed to be available for the public. Thus the confidentiality of the system does not add much to its value. On the other hand, the integrity of the data posted by users, and the guarantee of their website availability are the mainstays of the web hosting business. Therefore, the policy is defined as in Table 3.1

Public web hosting server	
Confidentiality	0.2
Integrity	1
Availability	1

Table 3.1 A security policy for an example public web hosting server.

Resource	C	I	A
HTTP Service	1	1	1
SFTP Service	0	1	1

Table 3.2 A set of resources for an example public web hosting server.

3.5.3 Enumerating system resources.

Once the system value is assigned, the next step is to enumerate the resources which need to be protected. Many approaches attempt to capture a comprehensive or structured set of resources in a system, however in practice this is daunting for system administrators. In addition the complexity of the resulting model may lead to a reduced confidence in the correctness of the model, and the more detailed the enumeration is, the more impact small changes in system state will have on the evaluation of a response or intrusion impact.

The focus for this methodology is on high-level applications or resources. For instance, a system administrator might assign to a public web server a set of resources consisting of {`http service`, `sftp service`}. While the system itself is a complex set of interdependent resources and services, this abstraction allows the system administrator to assign values based on their experience and intuition.

The set of resources enumerated for the example system, along with the resource weights, is given in Table 3.2

3.5.4 Assigning resource impact weights.

Each resource is assigned a tuple of weights, one for each security goal in the system, indicating the potential impact the resource can have on that system security goal. Each weight is assigned by the system administrator for each category with the goal of allowing the response selection computation to distinguish between which resources are critical to a particular security goal and which are less important.

Therefore, while the weight can be any real number, we restrict the system administrator to establishing a total ordering on the resource impacts for each security category, and selecting the maximum and minimum impact in this order. In this way we achieve a maximum separation between adjacent resources in the order, allowing the greatest distinction between resource impacts on each security category.

For the public web hosting service, the `http service`, if compromised, will impact confidentiality, integrity, and availability completely. On the other hand, the `sftp service`, used only for posting new content, can only impact the integrity and availability of the server, but not the confidentiality. The result of following this process is displayed in Table 3.2.

3.5.5 Assigning response impact values.

Assigning response impact values also consists of assigning weights to security categories, but in contrast to the resources, response impacts weights are assigned according to the degree of adverse impact a response will have on a resource's security category.

With the goal of maximum distinction, we again use the partial ordering approach to let the system administrator first order responses and then set the maximum and minimum impacts in the ordering for the security category on that particular resource.

Following the example, we will consider the following four responses:

Block host in .htaccess file – blocks a host from access to a specific area of the web site. The web server still processes the URL request.

Block host with iptables firewall rule – blocks a host from any network access. The kernel drops the packet before it gets to user space.

Shutdown the HTTP service – Makes the HTTP service completely unavailable.

Shutdown the SFTP service – Makes the SFTP service completely unavailable.

We will step through this process for one resource facet, the availability of the HTTP service. The other security facets are handled similarly, and the result is listed (along with the response protection assessment) in Table 3.3.

To evaluate the impact of responses on a specific resource the establishment and use of a total order is used again. Once ordered (possibly with some responses having equal effect) from least to greatest impact, the damage weight for responses with rank i is assigned according to the formula $\frac{i}{n_{equiv}-1}$, where n_{equiv} is the number of distinct equivalent sets in the ordering.

In this example then:

1. The response *Shutdown the SFTP service* will have no impact on HTTP availability, so it is assigned a value of 0. The response *Shutdown the HTTP service* on the other hand, will have full impact so it is given a ranking of 1.

2. The response *Block host in .htaccess file*, while having some impact on the availability of the web service, has less impact than blocking all IP traffic, so it is ranked at $0 < k_1 < k_2 < 1$, while *Block host with iptables firewall rule* is given a ranking of k_2.

3. Since the number of equivalent response sets in this ordering (not including those with no impact at all) is 3, the weight assigned to *Block host in .htaccess file* is $1/3 = 0.33$, and that assigned to *Block host with iptables firewall rule* is $2/3 = 0.67$.

3.5.6 Assigning response operational cost.

Response operational cost is computed as the sum of non-system impact costs of deploying a response divided by the value of the system. Determining the non-system impact costs of a response can be daunting, however, a few guidelines are suggested to make the process more consistent.

Block host in .htaccess file						
	Damaged			Protected		
	C	I	A	C	I	A
HTTP Service	0	0	0.33	0.33	0	0
SFTP Service	0	0	0	0	0	0
Block host with iptables firewall rule						
	Damaged			Protected		
	C	I	A	C	I	A
HTTP Service	0	0	0.66	0.66	1	1
SFTP Service	0	0	0.5	0	0	1
Shutdown the HTTP service						
	Damaged			Protected		
	C	I	A	C	I	A
HTTP Service	0	0	1	1	1	0
SFTP Service	0	0	0	0	0	0
Shutdown the SFTP service						
	Damaged			Protected		
	C	I	A	C	I	A
HTTP Service	0	0	0	0	1	0
SFTP Service	0	0	1	1	1	0

Table 3.3 A set of responses for an example public web hosting server.

In general, three broad categories cover the majority of response operational cost factors: the labor associated with the response (i.e., man-hours required to either deploy or recover from the effect of it), the direct costs involved in the response (i.e., paying a third party fee), and the additional resources required to implement the response (i.e., additional disk space or bandwidth consumed on other systems).

These values are estimated using organization specific assessment methods, and the resulting cost is divided by the total value of the system. This yields the (unit-less) relative operational cost of the response.

In the web hosting service scenario, assume that a system administrator is hired to keep the system running normally, and that the minimum one-hour charge for her services is $200. When either the HTTP or SFTP services are stopped, this individual will need to log in, review the circumstances of shutdown, and restart the services. Thus the operational cost of these two responses will be $\frac{\$200}{\$20000} = 0.01$. The other two responses do not involve any system administrator overhead, and thus have an operational cost of 0.

3.5.7 Assigning response protection values.

To evaluate the degree of protection a response confers upon a resource, an administrator must consider the degree to which potential damage to a specific resource aspect may be prevented by the response. For instance, given a database service, if the database is shutdown, there is a degree of protection given to the confidentiality of the database. If the database state is restored from a backup, there is protection of both availability and integrity.

It is assumed in this work that response actions are purposefully implemented and made available for deployment to counter a specific set of threats to security resource aspects. For instance, typically firewalls are put in place to permit the response of blocking an IP address, preserving at least the confidentiality of the network ports protected by it.

This task can be viewed as a process of translating those purposes into specific protections for specific resources. While the range of this value is a real number in $[0, 1]$, again it is preferable to maximize the distinction between the effectiveness of different responses. Therefore, a

similar process to that for assigning response system impact based on a total order.

Given the response set for this example, consider the confidentiality aspect of the HTTP service. The following process is followed by the system owner:

1. The response *Shutdown the SFTP service* will offer no protection for HTTP confidentiality, so it is assigned a value of 0. The response *Shutdown the HTTP service* on the other hand, will maximally protect HTTP confidentiality, so that response is given a value of 1.

2. The response *Block host in .htaccess file*, will protect the confidentiality of the part of the site it is deployed at and so is given a ranking of k_1, where $0 < k_1 < k_2 < 1$. *Block host with iptables firewall rule* is given a ranking of k_2 because it will offer more protection, blocking access to the entire site.

3. Since the number of equivalent response sets in this partial order (not including those offering no protection at all) is 3, the weight assigned to *Block host in .htaccess file* is $1/3 = 0.33$, and that assigned to *Block host with iptables firewall rule* is $2/3 = 0.67$.

3.5.8 Assigning intrusion impact values.

Intrusion impact is a similar concept to response impact, with a slight distinction during the assignment: impact on a resource security goal is restricted to the set $\{0, 1\}$ rather than the range $[0, 1]$.

There is no indication apriori of the degree to which an intrusion with the potential to damage a resource actually will damage that resource. Therefore, if an intrusion has the capability to damage a resource security aspect, it is given a weight of 1 for that aspect. Otherwise, it is given a weight of 0. Making this determination is left up to the system administrator and the security officer in charge of defining the security alerts.

The example system has the following set of possible intrusion alerts:

Buffer overflow in URL request – A URL request matching the buffer overflow signature is detected.

Buffer overflow in URL request			
	Damaged		
	C	I	A
HTTP Service	1	1	1
SFTP Service	0	0	0

Shell command execution attempt			
	Damaged		
	C	I	A
HTTP Service	1	1	1
SFTP Service	0	0	0

Attempt to read /etc/passwd file			
	Damaged		
	C	I	A
HTTP Service	0	0	0
SFTP Service	1	1	1

Table 3.4 A set of intrusions for an example public web hosting server.

Shell command execution attempt – A valid shell command string is identified in an HTTP request.

Attempt to read /etc/passwd file – An attempt to traverse the filesystem and read entries in /etc/passwd is detected.

Based on the system administrator's intuition, this set of intrusion is assigned impacts according to Table 3.4

3.5.9 Assigning intrusion operational cost.

Intrusion operational cost is computed in the same manner as the response operational cost, however as mentioned above, intrusion operational cost has a range > 0. In the case of intrusions we also have three broad categories into which most operational costs fall: the labor associated with manually mitigating or addressing an intrusion, the loss of reputation to the organization due to the intrusion occurring, and direct costs for contracted services such as forensic analysis.

Attack	Time to Remediate	Rate	Operational Cost
Buffer overflow in URL request	1 hour	$200	$\frac{\$200+\$1000}{\$20000} = 0.06$
Shell command execution attempt	3 hours	$200	$\frac{\$600+\$1000}{\$20000} = 0.08$
Attempt to read /etc/passwd file	5 hours	$200	$\frac{\$1000+\$1000}{\$20000} = 0.10$

Table 3.5 Intrusion Operational Costs for an example public web hosting server.

In the web server hosting example, mandatory reporting requirements cost $1000 for any intrusion, regardless of severity. In addition, system recovery takes time according to Table 3.5.

3.5.10 Determining intrusion probability.

While in general the probability of an intrusion actually occurring is difficult to determine, the value needed here is actually the probability that the intrusion is occurring, given that we have received an alarm indicating that it is occurring $p(Q|\texttt{Alarm})$. We can determine this directly by keeping a count of the number of times a specific alert is received, and having an administrator provide input on the number of times the intrusion actually occurred from the set of alarms.

This gives an estimate of the confidence with which this alert source indicates a true intrusion. In addition, it may be necessary to ascertain the probability that a specific intrusion out of a set of possible intrusions indicated by an alarm is occurring. This can be restated as the probability that, given an alarm indicating a true intrusion, the specific intrusion occurring is the one under consideration. We can denote this as the probability $p(q \in Q|\texttt{Alarm})$, and again this can be estimated by tracking the number of times the alarm is sent and an intrusion actually occurs, and the number of times intrusion q that occurred.

A body of knowledge around accurately estimating this type of probability exists, and further details are referred to appropriate papers. A starting point is found in [4], and a recent IDS specific approach is [7]. For this discussion the assumption is made that the intrusion probability is an assigned value provided as an input along with the alert.

CHAPTER 4. IMPLEMENTATION AND PERFORMANCE RESULTS

To gain further understanding about how this framework integrates with varying system policies, two implementations were developed. A Java simulation is implemented to allow interactive testing, and an implementation in C is written to test the performance of the framework on a Linux host. Details of these implementations are given below, and the results of the experiments using them are presented in section 4.2

4.1 Implementation

4.1.1 Java applet/servlet simulation

The majority of the Java implementation consists of an applet containing the computations related to the framework. The user interface is a simple GUI which allows the user to input the following:

1. System policy in terms of Confidentiality, Integrity, and Availability
2. List of system resources and the impact of each resource on system Confidentiality, Integrity and Availability.
3. List of responses available to the system, along with the operational cost of the response, the impact of the response on each resource security characteristic, and the degree of protection offered by the response to each resource security characteristic.
4. List of intrusions in the attack profile to be considered, along with the operational cost of the intrusion and the impact of the intrusion on each resource security characteristic.

The simulation can then be run by clicking a button. The user is directed to give their own ranking of the available responses in order of most preferred to least preferred, a self-evaluation

of their experience level, and their primary work area (academic, industry, or both). The user is then given a comparison of the ranking they selected with that selected by the simulation, and feedback is requested from the user about the differences. When submitted, the applet communicates with a Java servlet to transmit the results, which are then written to a file on the server.

This implementation allows a system administrator to select input values based on their intuition about the system security goals, construct a test intrusion scenario, and then determine if the proposed framework selects responses which match the actions the system administrator would take.

In addition to running the simulation and viewing the results, there are "save" and "load" options which write and read XML formatted files so that system definitions can be re-used later, or shared with others. This is particularly important if the results of the trial are to be independently verified, or re-used for the evaluation of other proposed models.

Finally, an option to show the calculations will export the values computed at different stages of the computation to a comma separated value file, suitable for loading into an Excel spreadsheet. This allows a level of transparency, enabling users to determine why (according to the model) one response is preferred to another.

A diagram of the implementation design is given in Figure 4.1

4.1.2 Host based implementation in C

The implementation for host-based cost-sensitive response on the Linux platform is written in C for performance evaluation. The goal is to provide timing data, and verify the feasibility of effectively responding to real-world attacks. There are several pieces of code in this implementation:

selectResponse.c This code implements the framework calculations. Compiled as an object file, the defined methods take structure arguments which represent the system, resources, responses, intrusions, and the attack profile under consideration. The ultimate result is the selection and execution of a response on the system.

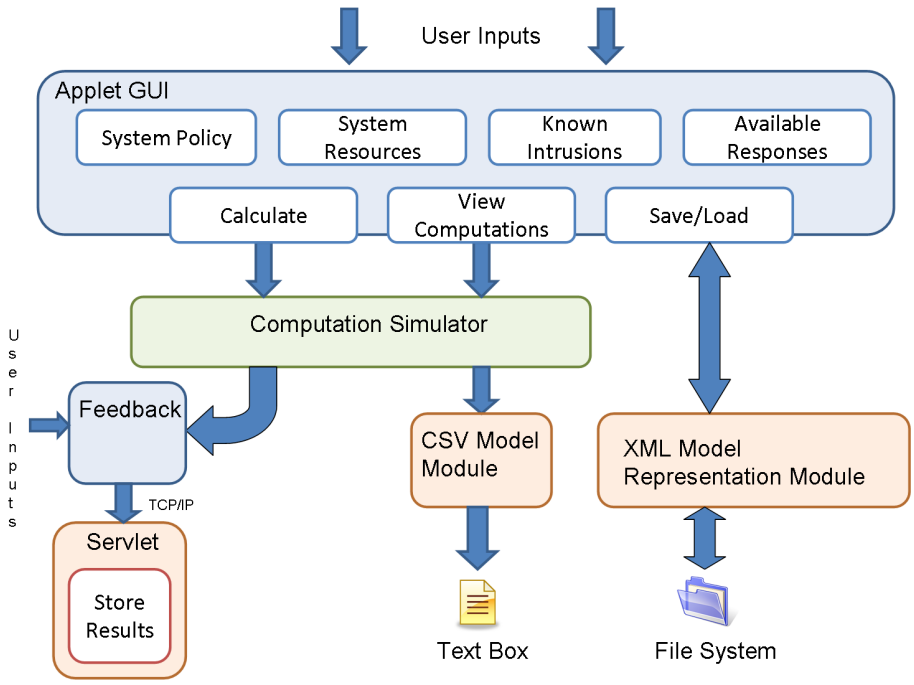

Figure 4.1 IRS simulation in Java.

simulatedIDS.c A socket-client code which takes alerts from a Snort data source and converts them to the defined message protocol to input into the response selection engine. UDP sockets are used to transmit messages.

selectResponseRun.c The data loading and socket interface for the response selection engine. This code opens a listening UDP socket, and waits for alert messages indicating a potential intrusion. When all alerts for a potential intrusion are received, an attack profile structure is created and passed to the response selection code.

selectResponseGetValues.c An alternate interface to the response selection engine. Instead of actually deploying a response, this code allows the generation of output in a human readable form. Similar to the "show calculations" option in the Java implementation, this permits users to see the costs and values as computed during the intrusion response selection process.

Execution occurs in two phases: initialization and monitoring. During the initialization phase, system security policy, set of available resources, available responses, and the definition of known intrusions are loaded into the framework and the appropriate data structures are

constructed. This data is stored in four text files which are given as arguments on the command line.

Once the model is initialized, a UDP socket is opened to receive connections from data input sources and the system waits until an intrusion notification is sent. An intrusion notification consists of four parts:

1. The intrusion name. This name must exist in the set of known intrusions, or the alert is ignored.
2. A unique identification value (IID) for this set of intrusions. This value indicates the alert messages which are part of the same potential intrusion.
3. The number of intrusions which are included in this set of possible intrusions.
4. The probability that this specific intrusion is actually occurring.

When all alerts with the same IID have been received and added to the attack profile data structure, the attack profile is passed to the response selection engine, which evaluates the responses in the context of the profile and chooses one to deploy. This implementation is designed as in Figure 4.2.

Model changes in the implementation. There are a few minor changes made to the framework in the implementation for aesthetic reasons:

1. Both implementations normalize the response system impact and intrusion damage using the max possible damage. The maximum possible impact may vary greatly from system to system depending on the number of resources and the security policy assigned to the system. Using this value to normalize SI gives a relative value which can then be compared across systems and organizations to ensure that SI is always in the range $(0, 1)$.

2. Both implementations divide the response cost and intrusion cost by 2. Since both OC and SI have range $(0, 1)$ to start with, the sum has a range $(0, 2)$. Dividing by 2 normalizes the cost range of each to $(0, 1)$.

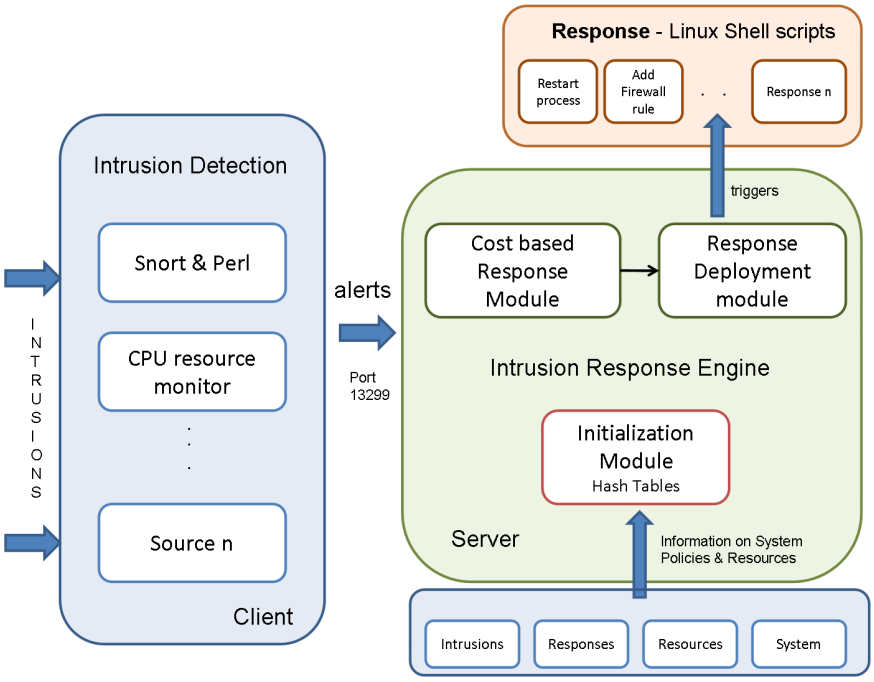

Figure 4.2 Host-based IRS implementation for Linux systems in C.

4.2 Results

Evaluating the effectiveness of a proposed system of measurement in a domain such as this is complicated by a lack of existing standards with which to compare. There are no established units or scale for many of the quantities considered, and this leads to a lack of provable characteristics. In light of these difficulties, we take an empirical approach and establish through experimentation that this framework is correct, intuitive, and useful.

4.2.1 Correctness

To establish the correctness of this framework two questions are considered: the lack of obvious flaws in the model, and the improvement in the cost benefit of automated response compared to static response selection.

Definition of a rational response. In order to determine if this approach is fundamentally flawed, the concept of a *rational response value* was established. It is a set of characteristics which should always be true of any value assigned to a response. A suite of scenarios

No impact	→	no cost
Only valueless impact	→	no cost
No OC, subset impact	→	$0 \leq \texttt{cost}_i \leq 0.5$
No OC, full impact	→	cost = 0.5
$OC = 1$, no impact	→	cost = 0.5
$OC = 1$, full impact	→	cost = 1
Response to no impact (any OC)	→	no benefit
Resource impact and protection	→	Include impact cost Impact coverage loss No Success Factor loss

Table 4.1 Rational Selection Axioms

System Resources			
Name	C	I	A
AllOnes	1	1	1
OnlyConf	1	0	0
NoValue	0	0	0

Table 4.2 The system definition used to demonstrate rationality.

is developed in order to test whether a specific response selection engine can be classified as rational. The set of axioms defined is given in Table 4.1

For simplicity, the term 'resource' above refers to a specific security goal of a resource (Confidentiality, Integrity, or Availability).

To demonstrate rationality as defined above, a set of inputs is defined to test each of these conditions. The defined system, intrusions, and responses are given in Tables 4.2, 4.3, and 4.4 respectively.

The results of these tests are given for response cost in Table 4.5, response benefit in Table 4.6, and overall response value in Table 4.7.

These results only demonstrate that the response system does not make irrational cost assessments. However, it does not determine that the cost assessments are accurate.

Comparison with a static response system. To determine the cost benefit provided by this approach, the response selection framework is adapted as a plugin tool for an intrusion

Name	Operational Cost	Resources Damaged			
		Name	C	I	A
No Impact	0	AllOnes	0	0	0
		OnlyConf	0	0	0
		NoValue	0	0	0
OnlyOC	1	AllOnes	0	0	0
		OnlyConf	0	0	0
		NoValue	0	0	0
PartialSI	0	AllOnes	0	1	0
		OnlyConf	1	0	0
		NoValue	0	0	0
AllSI	0	AllOnes	1	1	1
		OnlyConf	1	1	1
		NoValue	1	1	1
OnlyNovalue	0	AllOnes	0	0	0
		OnlyConf	0	0	0
		NoValue	1	1	1
FullImpact	1	AllOnes	1	1	1
		OnlyConf	1	1	1
		NoValue	1	1	1

Table 4.3 The intrusion set used to demonstrate rationality.

Name	Operational Cost	Resources						
		Name	C	I	A	C	I	A
No Impact	0	Damaged				Protected		
		AllOnes	0	0	0	0	0	0
		OnlyConf	0	0	0	0	0	0
		NoValue	0	0	0	0	0	0
Only OC	1	Damaged				Protected		
		AllOnes	0	0	0	0	0	0
		OnlyConf	0	0	0	0	0	0
		NoValue	0	0	0	0	0	0
PartialSI	0	Damaged				Protected		
		AllOnes	0	1	0	0	0	0
		OnlyConf	1	0	0	0	0	0
		NoValue	0	0	0	0	0	0
AllOnesSI	0	Damaged				Protected		
		AllOnes	1	1	1	0	0	0
		OnlyConf	0	0	0	0	0	0
		NoValue	0	0	0	0	0	0
OnlyNoValue	0	Damaged				Protected		
		AllOnes	0	0	0	0	0	0
		OnlyConf	0	0	0	0	0	0
		NoValue	1	1	1	0	0	0
Full Impact	1	Damaged				Protected		
		AllOnes	1	1	1	0	0	0
		OnlyConf	1	1	1	0	0	0
		NoValue	1	1	1	0	0	0
Protect NoValue	0	Damaged				Protected		
		AllOnes	0	0	0	0	0	0
		OnlyConf	0	0	0	0	0	0
		NoValue	0	0	0	1	1	1
Protect Partial	0	Damaged				Protected		
		AllOnes	0	0	0	0	1	0
		OnlyConf	0	0	0	1	0	0
		NoValue	0	0	0	0	0	0
Protect Full	0	Damaged				Protected		
		AllOnes	0	0	0	1	1	1
		OnlyConf	0	0	0	1	1	1
		NoValue	0	0	0	1	1	1
Protect Disjoint	0	Damaged				Protected		
		AllOnes	0	0	0	1	0	1
		OnlyConf	0	0	0	0	1	1
		NoValue	0	0	0	1	1	1
Everything	1.0	Damaged				Protected		
		AllOnes	1	1	1	1	1	1
		OnlyConf	1	1	1	1	1	1
		NoValue	1	1	1	1	1	1

Table 4.4 The response set used to demonstrate rationality.

Response Name	NoImpact	OnlyNovalue	OnlyOC	PartialSI	AllSI	FullImpact
NoImpact	0.000	0.000	0.000	0.000	0.000	0.000
OnlyNovalue	0.000	0.000	0.000	0.000	0.000	0.000
OnlyOC	0.500	0.500	0.500	0.500	0.500	0.500
PartialSI	0.250	0.250	0.250	0.250	0.250	0.250
AllOnesSI	0.375	0.375	0.375	0.375	0.375	0.375
FullImpact	1.000	1.000	1.000	1.000	1.000	1.000
PNovalue	0.000	0.000	0.000	0.000	0.000	0.000
PDisjoint	0.000	0.000	0.000	0.000	0.000	0.000
PPartial	0.000	0.000	0.000	0.000	0.000	0.000
PFull	0.000	0.000	0.000	0.000	0.000	0.000
Everything	1.000	1.000	1.000	1.000	1.000	1.000

Table 4.5 Results of the rational response cost test.

Response Name	NoImpact	OnlyNovalue	OnlyOC	PartialSI	AllSI	FullImpact
NoImpact	0.000	0.000	0.000	0.000	0.000	0.000
OnlyNovalue	0.000	0.000	0.000	0.000	0.000	0.000
OnlyOC	0.000	0.000	0.000	0.000	0.000	0.000
PartialSI	0.000	0.000	0.000	0.000	0.000	0.000
AllOnesSI	0.000	0.000	0.000	0.000	0.000	0.000
FullImpact	0.000	0.000	0.000	0.000	0.000	0.000
PNovalue	0.000	0.000	0.000	0.000	0.000	0.167
PDisjoint	0.000	0.000	0.000	0.000	0.250	0.750
PPartial	0.000	0.000	0.000	0.250	0.167	0.500
PFull	0.000	0.000	0.000	0.250	0.500	1.000
Everything	0.000	0.000	0.000	0.000	0.000	0.500

Table 4.6 Results of rational response benefit test.

Response Name	NoImpact	OnlyNovalue	OnlyOC	PartialSI	AllSI	FullImpact
NoImpact	0.000	0.000	0.000	0.000	0.000	0.000
OnlyNovalue	0.000	0.000	0.000	0.000	0.000	0.000
OnlyOC	-0.500	-0.500	-0.500	-0.500	-0.500	-0.500
PartialSI	-0.250	-0.250	-0.250	-0.250	-0.250	-0.250
AllOnesSI	-0.375	-0.375	-0.375	-0.375	-0.375	-0.375
FullImpact	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
PNovalue	0.000	0.000	0.000	0.000	0.000	0.167
PDisjoint	0.000	0.000	0.000	0.000	0.250	0.750
PPartial	0.000	0.000	0.000	0.250	0.167	0.500
PFull	0.000	0.000	0.000	0.250	0.500	1.000
Everything	-1.000	-1.000	-1.000	-1.000	-1.000	-0.500

Table 4.7 Results of rational response value test.

detection system (Snort) and a series of experiments are performed.

Results. We evaluate our model using the 1998 DARPA/Lincoln Lab offline evaluation data, in particular week 5 for days Monday, Thursday, and Friday, and week 6 on Thursday. The tcpdump data is replayed and Snort, an open source signature-based IDS, is used to generate alerts. Although Snort has been shown to perform poorly on DARPA data sets [3], it is freely available to general public and is currently one of the most widely used IDSs. In addition, since the focus of this work is not on the performance of intrusion detection approaches, it is appropriate to employ Snort in our experiments.

In these experiments the goal is to show the cost saved by dynamically selecting the lowest cost response compared with the cost of always deploying a statically assigned response. Static response assignment can be implemented in several ways. In these experiments, it is assumed to be implemented as a reflex response, where alerts seen from an IP address trigger an immediate static response. Based on the number of available scripts on the Internet implementing automated blocks of IP addresses along with experience, blocking the source IP address is chosen as the static response to deploy.

The cost is computed according to this framework, with the parameters defined by a system administrator for each system type. While this is, in a sense, self-selecting, the goal is

not to demonstrate intuitiveness, but to show that, under the assumption that an administrator assigns cost values according to their intuition, the proposed framework is unilaterally more cost-effective. The *cumulative response cost* metric as the primary criteria, showing the cumulative value of all responses deployed on the system over time.

The characteristics of these systems are given in Table 4.8, and the distribution of alerts generated by Snort are given in Table 4.9. The results of the experiments in comparison to a traditional system equipped with a static intrusion response mechanism are given in Figures 4.3, 4.4, 4.5.

In these figures, the difference between the cost-sensitive cumulative value and the static response cumulative value is shown by the separation between the lines. As time goes on, the value of the static response tends to remain around zero for the high integrity and availability systems, while the cost-sensitive response attains significantly more value throughout the experiment. In the high confidentiality system, however, the static response is much more competitive, with the cost-sensitive approach only achieving appreciable advantage toward the end of the trial.

This is explained by the choice of static response, *block attacker IP*. This response protects confidentiality at the expense of availability. Thus, it is generally desireable for network based attacks detected by snort in the high-confidentiality system, but the other two systems retain more value by choosing other, less costly responses such as making the file system read-only. This indicates that, even in systems where the security priority is heavily in favor of a specific static response, a cost-sensitive approach will still outperform the static approach over time.

In addition to cumulative cost, the number of times each response is deployed is given in Tables 4.10, 4.11, 4.12. From these tables it is apparent that in some cases, e.g. for the high-confidentiality system, the number of responses on which the two approaches disagree can be very minor. However, when they do disagree, the value difference for the system can be significant. While these simulated trials were run over a period of roughly four days, many systems remain deployed for years. The implementation of a cost-sensitive response capability will significantly reduce the overall impact of detected intrusions over the life of such systems.

System legend: *public web server providing remote access for affiliates and public information* System security priorities: low confidentiality 0.1, moderate integrity 0.5, high availability 0.9.	
System legend: *central file repository for distributed collaboration* System security priorities: moderate confidentiality 0.5, high integrity 0.9, low availability 0.1.	
System legend: *Medical information system for a hospital* System security priorities: high confidentiality 0.9, high integrity 0.9, low availability 0.1.	
System Resources: Web server (providing web access to files for users) FTP server (providing FTP access to get and post files) RPC server (providing remote procedures used by some of the web services and possibly clients) SNMP server (used for system monitoring and management by the administrator)	*System responses*: Block Attacker's address in .htaccess Block Attacker's IP Address in Firewall Restart Service (Web, FTP, RPC, or SNMP) Reboot System Stop Service Disable User Logins Disable specific user account

Table 4.8 The characteristics of the hypothetical systems used in the experiments.

ATTACK-RESPONSES	28	SNMP	2943
BACKDOOR-MISC-Solaris-2.5-attempt	1	TFTP-Get	400
FINGER-/-execution-attempt	18	WEB-CGI	222
FTP-.rhosts	1	WEB-FRONTPAGE	13
RPC-portmap-listing-TCP-111	2	WEB-IIS	18
RSERVICES-rsh-root	1	WEB-MISC	20
SCAN-myscan	30	X11-xopen	1
SHELLCODE-sparc-NOOP	9		

Table 4.9 Snort alert classes generated by DARPA data

Dynamic Response		Static Response	
No Response	2741	No Response	3653
Block SNMP Request	927	Block SNMP Request	-
Block Attacker .htaccess	35	Block Attacker .htaccess	-
Block Attacker IP	4	Block Attacker IP	54

Table 4.10 The number of times each response was deployed on the DARPA data for the high availability system.

Dynamic Response		Static Response	
No Response	3856	No Response	3821
Block Attacker IP	20	Block Attacker IP	55

Table 4.11 The number of times each response was deployed on the DARPA data for the high confidentiality system.

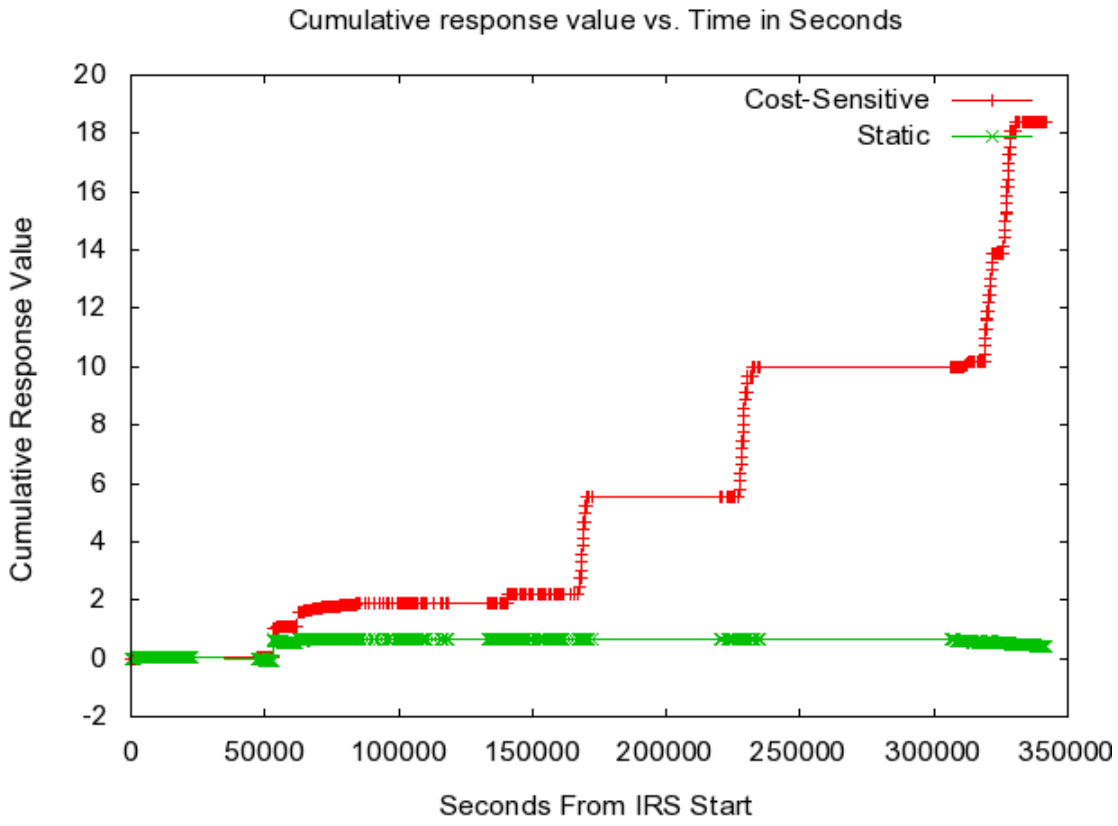

Figure 4.3 Evaluation of the response on a system with high availability requirements (public web server).

4.2.2 Scalability

An important feature of an IRS is its ability to respond in a timely fashion. One concern for this framework is the impact which a large number of resources, responses, or potentially occurring intrusions could have on performance. While the number of potential intrusions known is also a factor, implementation using hash-tables makes the performance impact of this quantity a constant factor.

Performance. A set of experiments were conducted to measure the performance of the intrusion response selection algorithm on artificial data sets. The experiments were run on Intel(R) Core(TM)2 Duo CPU with a clock speed of 1.33GHz and 1 GB of RAM. The experiments are performed on three primary variables: number of system resources, number of responses available in the system and number of suspected intrusions. For each trial, the other two parameters

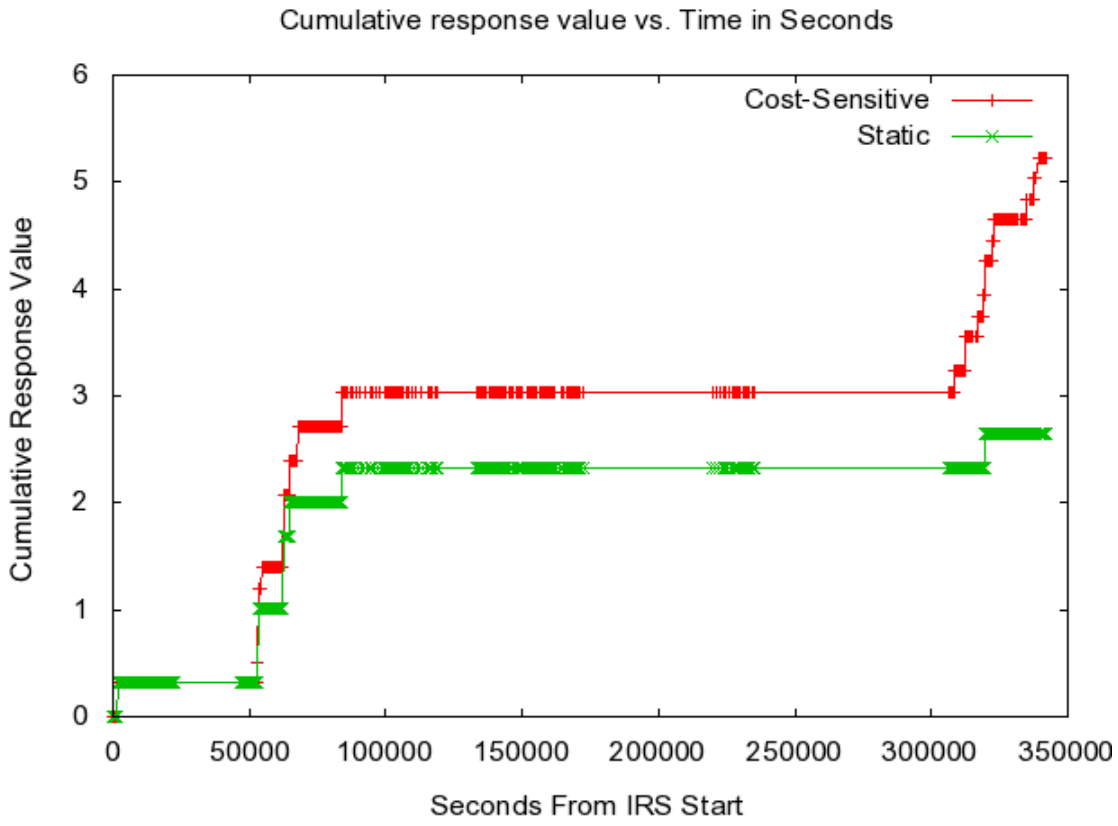

Figure 4.4 Evaluation of the response on a system with high confidentiality requirements (medical information system).

remained fixed at 100. The results of these experiments, given in Figure 4.6, show that even for a system with a significantly large number of resources to consider, the computation time for the best response strategy is only $0.015s$. As Figure 4.6 shows, the highest impact on performance is the number of suspected intrusions simultaneously considered during the response selection process. As such it took $0.221s$ to assess the available responses for 10000 suspected intrusions. While we consider very large sets of intrusions, in reality there are generally only a few possible intrusions to consider.

These results show that this approach has reasonable process time requirements, suitable for the efficient analysis of response selection in an automatic setting, as well as in support of manual response assessment during an administrator's daily routine.

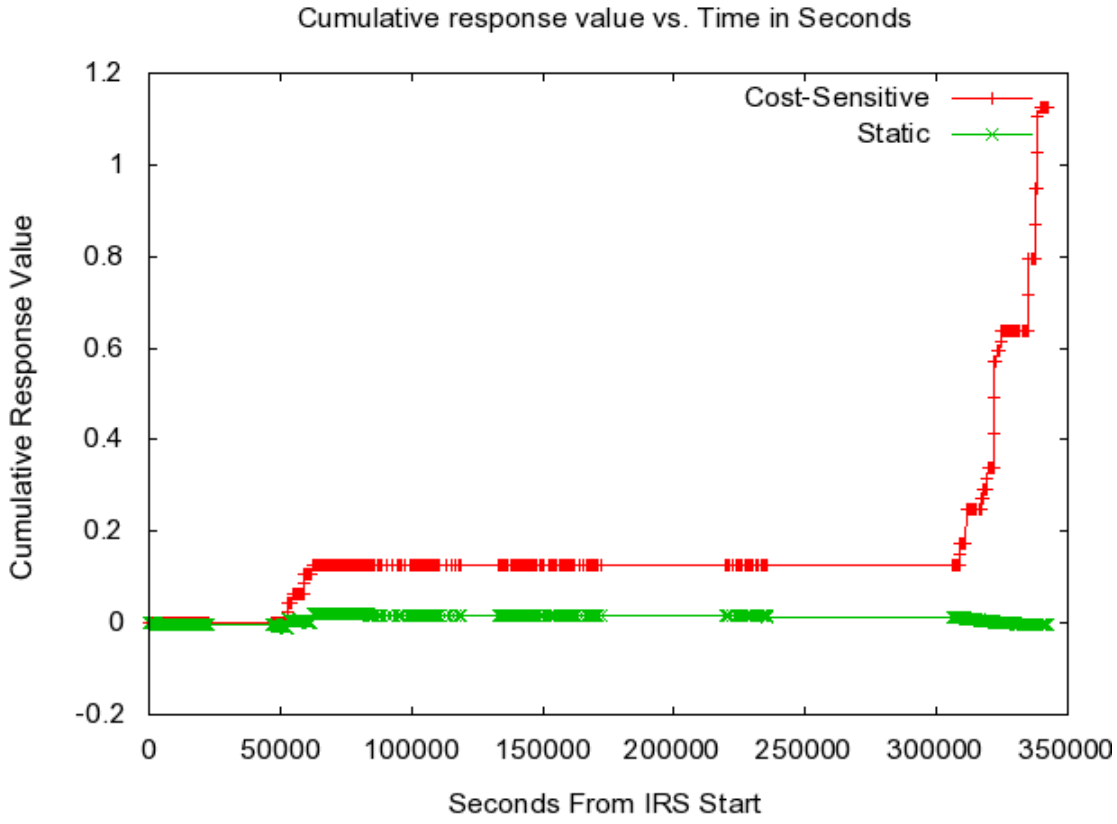

Figure 4.5 Evaluation of the response on a system with high integrity requirements (central file repository).

4.2.3 Intuitiveness

To determine the comparative effectiveness of this approach, it is compared to a system deploying a static response, and to a system implementing the ADEPTS IRS [6]. Given the tradeoffs made for the sake of implementability, performance and simplicity, it is expected that cost effectiveness would not necessarily exceed that of some of the formal approaches to cost evaluation.

In addition to comparing with static response mappings, a comparison of this method with the ADEPTS framework is performed. The ADEPTS paper presents test scenarios which are adapted for use with this framework. A comparison of responses selected is then presented in Table 4.16.

The scenario is in the context of an e-commerce website deployment. The example consists

Dynamic Response		Static Response	
No Response	-	No Response	3821
Re-Mount File System readonly	3818	Re-Mount File System readonly	-
Shutdown Web Service	58	Shutdown Web Service	-
Block Attacker IP	-	Block Attacker IP	55

Table 4.12 The number of times each response was deployed on the DARPA data for the high integrity system.

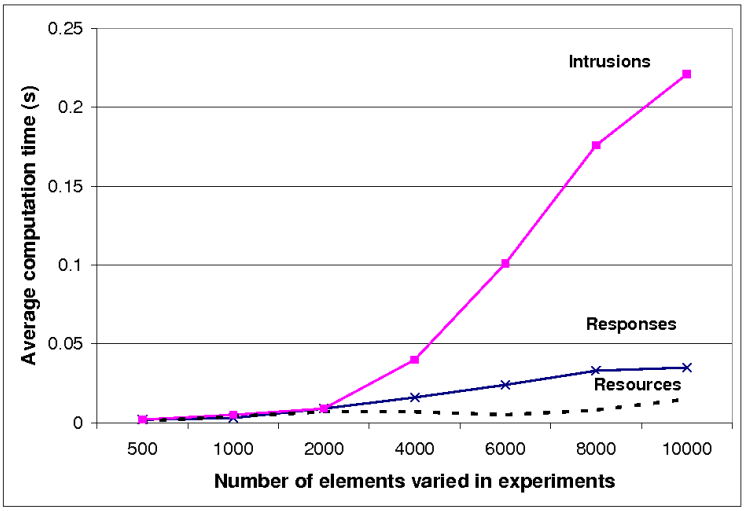

Figure 4.6 Processing time evaluation

of the following elements:

1. A set of values associated with users being able to perform certain functions (i.e. browse the web store, place an order).

2. A set of values associated with specific security goals (i.e. corruption of MySQL database, illegal read of file).

3. A set of attack scenarios, in which an attacker performs malicious actions with different goals.

To adapt this scenario for use in this work, the following steps are taken:

1. The valued user actions have sets of resources associated with them which enable those actions (i.e. availability of HTTP service, MySQL service).

2. The security goals are assigned dependencies on those resources (i.e. integrity of MySQL datastore).

3. The sum of all the values in the ADEPTS examples of both the security goals and the user actions is taken to be a representation of the overall value of the system.

4. The value contribution of each resource is then computed using the values assigned to each of the actions or security goals depending on that resource. These values are normalized as a ratio to the overall system value.

5. The malicious actions are associated with the resources and security goals that would be compromised (i.e., a buffer overflow in MySQL would results in a confidentiality, availability, and integrity loss for the MySQL service).

6. The system security policy (C, I, A) is computed from the values assigned to various security goals and the security policy area those security goals is implied (i.e. MySQL integrity contributes to system integrity).

7. Since the attack actions are known apriori to be true positives, the intrusion probability is set to 1.

8. A set of possible responses is enumerated, using the responses mentioned in the ADEPTS paper.

This adaptation effectively determines all of the information required for this model, and is outlined in Tables 4.13, 4.14, and 4.15. This natural and intuitive adaptation indicates that our model is flexible enough to accomodate the specific implementation of ADEPTS, and is a result of the adaptability of this framework to a variety of environments. Note that while the converted values may not be universally intuitive to all system administrators, they do

System Value: 565	System Policy		
	C = 0.372	I = 0.549	A = 0.080
Resources			
Resource Name	C	I	A
Apache Service	0.250	1.000	0.600
MySQL Service	0.750	1.000	0.600
Link to Warehouse/Bank	0.000	0.000	0.200
SSH Service	1.000	1.000	0.400
External network connection	0.500	0.667	1.000
Filesystem	0.750	0.333	0.800

Table 4.13 Framework input system adapted from ADEPTS example.

represent the intuition of the ADEPTS authors. Thus the comparison between the responses chosen by the adepts authors as having the most value, and the values assigned by our method is valid.

In the presentation of the ADEPTS framework, six iterations of the attack sequence are required before Adepts will select a response which stops the attack within the first two steps. However, the framework presented in this thesis not only selects the response which succeeds in stopping the attack, of the two successful responses chosen by Adepts (reboot the server, and kill crontab and block access to it), our method selects the much less costly one (kill crontab and block access to it).

This comparison shows that, while ADEPTS has advanced capabilities in adapting to select the correct response, the methodology developed in this thesis selects the most effective response without requiring adaptation, resulting in a much lower cost per incidence early in the number of intrusion iterations.

The intuitiveness of our approach is demonstrated by adapting the established characteristics of the ADEPTS example and showing that our framework accurately selects the most effective and beneficial response based on those inputs. It is also significant because, while ADEPTS is targeted at web-commerce applications, the framework presented here is abstract enough to apply across environments. In addition, in practice it is rare for system administra-

Responses								
Name	OC	Resources Damaged				Protected		
		Name	C	I	A	C	I	A
Block web ports	0.025	Apache Service	0.000	0.000	0.333	1.000	1.000	0.000
		MySQL Service	0.000	0.000	0.000	0.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.000	0.000	0.000	0.000
		External network connection	0.000	0.000	0.333	0.000	0.000	0.000
		Filesystem	0.000	0.000	0.000	0.000	0.000	0.000
Kill Apache privilege shell	0.000	Apache Service	0.000	0.000	0.000	1.000	1.000	1.000
		MySQL Service	0.000	0.000	0.000	0.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.000	0.000	0.000	0.000
		External network connection	0.000	0.000	0.000	0.000	0.000	0.000
		Filesystem	0.000	0.000	0.000	0.000	0.000	0.000
Block source IP	0.050	Apache Service	0.000	0.000	0.667	1.000	1.000	0.000
		MySQL Service	0.000	0.000	0.500	0.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.500	0.000	0.000	0.000
		External network connection	0.000	0.000	0.667	0.000	0.000	0.000
		Filesystem	0.000	0.000	0.000	0.000	0.000	0.000
Kill crontab process	0.000	Apache Service	0.000	0.000	0.000	1.000	1.000	1.000
		MySQL Service	0.000	0.000	0.000	1.000	1.000	1.000
		Link to WH/Bank	0.000	0.000	0.000	1.000	1.000	1.000
		SSH Service	0.000	0.000	0.000	1.000	1.000	1.000
		External network connection	0.000	0.000	0.000	1.000	1.000	1.000
		Filesystem	0.000	0.000	0.000	1.000	1.000	1.000
Restart Apache Service	0.000	Apache Service	0.000	0.500	1.000	0.000	1.000	1.000
		MySQL Service	0.000	0.000	0.000	0.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.000	0.000	0.000	0.000
		External network connection	0.000	0.000	0.000	0.000	0.000	0.000
		Filesystem	0.000	0.000	0.000	0.000	0.000	0.000
Reboot Apache Server	0.100	Apache Service	0.000	0.500	1.000	0.000	1.000	1.000
		MySQL Service	0.000	0.500	1.000	0.000	1.000	1.000
		Link to WH/Bank	0.000	0.000	1.000	0.000	1.000	1.000
		SSH Service	0.000	0.000	1.000	0.000	1.000	1.000
		External network connection	0.000	0.000	1.000	0.000	1.000	1.000
		Filesystem	0.000	0.000	1.000	0.000	1.000	1.000
Deny access to crontab, kill crontab process	0.100	Apache Service	0.000	0.000	0.000	1.000	1.000	1.000
		MySQL Service	0.000	0.000	0.000	1.000	1.000	1.000
		Link to WH/Bank	0.000	0.000	0.000	1.000	1.000	1.000
		SSH Service	0.000	0.000	0.000	1.000	1.000	1.000
		External network connection	0.000	0.000	0.000	1.000	1.000	1.000
		Filesystem	0.000	0.000	0.000	1.000	1.000	1.000
Set Apache HTTP directory to read-only	0.100	Apache Service	0.000	1.000	0.000	0.000	1.000	1.000
		MySQL Service	0.000	0.000	0.000	0.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.000	0.000	0.000	0.000
		External network connection	0.000	0.000	0.000	0.000	0.000	0.000
		Filesystem	0.000	0.000	0.500	0.000	1.000	0.500

Table 4.14 Framework input responses adapted from ADEPTS example.

Intrusions					
Name	OC	Resources Damaged			
		Name	C	I	A
Apache buffer overflow	0.500	Apache Service	1.000	1.000	1.000
		MySQL Service	1.000	0.000	0.000
		Link to WH/Bank	1.000	0.000	0.000
		SSH Service	0.000	0.000	0.000
		External network connection	1.000	0.000	1.000
		Filesystem	1.000	1.000	1.000
MySQL port scan	0.500	Apache Service	0.000	0.000	0.000
		MySQL Service	0.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.000
		External network connection	0.000	0.000	0.000
		Filesystem	0.000	0.000	0.000
MySQL buffer overflow	0.500	Apache Service	1.000	0.000	0.000
		MySQL Service	1.000	1.000	1.000
		Link to WH/Bank	1.000	0.000	0.000
		SSH Service	0.000	0.000	0.000
		External network connection	0.000	0.000	0.000
		Filesystem	1.000	1.000	0.000
Apache Privilege Shell Created	0.500	Apache Service	1.000	1.000	1.000
		MySQL Service	1.000	0.000	0.000
		Link to WH/Bank	1.000	0.000	0.000
		SSH Service	0.000	0.000	0.000
		External network connection	1.000	0.000	1.000
		Filesystem	1.000	1.000	1.000
Execute crontab command	0.500	Apache Service	1.000	1.000	1.000
		MySQL Service	1.000	1.000	1.000
		Link to WH/Bank	1.000	1.000	1.000
		SSH Service	1.000	1.000	1.000
		External network connection	1.000	1.000	1.000
		Filesystem	1.000	1.000	1.000
Tamper with Apache files	0.500	Apache Service	1.000	1.000	1.000
		MySQL Service	1.000	1.000	0.000
		Link to WH/Bank	1.000	0.000	0.000
		SSH Service	0.000	0.000	0.000
		External network connection	1.000	0.000	1.000
		Filesystem	1.000	1.000	1.000
Enumerate webstore docroot using 'ls'	0.500	Apache Service	1.000	0.000	0.000
		MySQL Service	1.000	0.000	0.000
		Link to WH/Bank	0.000	0.000	0.000
		SSH Service	0.000	0.000	0.000
		External network connection	1.000	0.000	0.000
		Filesystem	1.000	0.000	0.000
Root privilege shell created	0.500	Apache Service	1.000	1.000	1.000
		MySQL Service	1.000	1.000	1.000
		Link to WH/Bank	1.000	1.000	1.000
		SSH Service	1.000	1.000	1.000
		External network connection	1.000	1.000	1.000
		Filesystem	1.000	1.000	1.000

Table 4.15 Framework input intrusions adapted from ADEPTS example.

Run	Rank	Response	Rank	Response
1	1	Kill crontab process	1	Block web ports
	2	Deny access to / kill crontab	2	Kill Apache privilege shell
	3	Block attacker's source IP	3	Block attacker's source IP
	4	Kill Apache privilege shell	4	Kill crontab process
	5	Block web ports	5	Restart Apache
	6	Restart Apache	6	*Reboot Apache's host machine*
	7	—	7	Deny access to / kill crontab
	8	—	8	Set Apache docroot to read-only
3	1	Kill crontab process	1	Block web ports
	2	Deny access to / kill crontab	2	Deny access to crontab command
	3	Block attacker's source IP	3	*Reboot Apache's host machine*
	4	Kill Apache privilege shell	4	Kill crontab process
	5	Block web ports	5	Block attacker's source IP
6	1	Kill crontab process	1	*Reboot Apache's host machine*
	2	Deny access to / kill crontab	2	*Deny access to / kill crontab*

Table 4.16 A comparison of the responses deployed by ADEPTS and those deployed by our framework.

tors to tolerate several iterations of excessively similar intrusions. Typically after the intrusion is detected and responded to, additional preventative measures are implemented to enhance resistance to that specific intrusion (and often that general intrusion type). This leads to the conclusion that optimal cost savings in intrusion response is more significant as the learning time to choose the most optimal response is reduced.

4.2.4 Comparison to expert intuition

To determine how well this framework models system administrator intuition, selected administrators were asked to use the Java implementation and respond with how well the results matched the responses they would have chosen.

While the number of responses was not sufficient to provide representative quantitative results, the responses show a wide variance both from the responses chosen by our framework and from each other. This indicates that even among experts with significant system administration experience, the intuition is not universal when choosing the response to deploy, reinforcing

the need for a concrete response cost metric from a practical as well as theoretical standpoint. As one responder says: "I would have gone big, blocking the subnet, initially which may be an over-reaction. The equation results characterize a smooth process for system administrators to follow during an attack," indicating that having a metric such as that presented in this work would result in a lower tendency for human error in the face of an intrusion.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis we have presented a host-based framework for the cost-sensitive assessment and selection of an intrusion response. This framework incorporates a set of evaluation metrics for the practical assessment of costs and benefits associated with intrusions and responses, and introduces a balanced strategy for response selection according to the security policy of the specified system.

We have shown, through the implementation and analysis of this framework, that while the field of automated intrusion response is still in its infancy and significant research effort is required to address all of the weaknesses still present in such systems, sufficient understanding exists in this field to construct a system which, with minimal interaction by system administrators, can significantly reduce the cost of intrusion response. This thesis constructs an automated response framework which selects responses based on a cost assessment model characterized by:

1. **Separation of the security policy from the cost model.**
2. **Definitions of intrusion and response cost measures.**
3. **Development of a system aware adaptable model.**
4. **An implementable framework.**

By leveraging human expertise in conjunction with structured specification guidance during the system resource, response, and intrusion specification phase, a robust automated response system can be used to reduce human error and respond to intrusions accurately in a fraction of the time required for manual response.

In addition, the performance analysis indicates that automated responses systems using this cost metric can be deployed which responds quickly enough to thwart active attacks in real time using optimal responses.

5.2 Future Work

As noted above, automated intrusion response is still a relatively young field in the computer security research arena. During the course of this research work, the following areas emerged as particularly promising avenues for future exploration:

Automating the impact evaluation of intrusions and responses on system resources. One of the primary challenges in the broad deployment of automated response systems remains the degree of effort and overhead required to implement the system. While system administrator input is a core component to intrusion response cost assessment, automating the enumeration of system resources would be a dramatic reduction in the manual labor required. In addition, enabling automatic assessment of response and intrusion impact would allow the extension of this model to include previously unseen intrusions, and dynamic composition of more sophisticated responses. One concrete direction might be to automate the inclusion of Snort rules, and combining these with a dynamic assessment of impact based on signature descriptions or common vulnerability assessments (i.e. CVSS, CVE).

Expanding the response to include sets of responses which minimize cost and maximize expected benefit. Response composition (combining multiple atomic responses to deploy a more sophisticated set of response actions) is a direction which would allow not only a response to more sophisticated attacks, but would also open avenues of combining attack containment and intrusion recovery. This is a key requirement in the evolution toward self-healing systems.

Extending this framework to a distributed environment. While the work presented in this thesis emphasizes host-based application, the general principles are extensible to

distributed systems as well. This would allow a resource hierarchy to be used, and network-optimal responses to be selected.

Developing metrics for the integrity and confidentiality of a system. While a number of established metrics for availability exist (mean-time between failure, responsiveness, etc.), concretely measuring integrity or confidentiality have received less treatment. Investigating these areas would allow the incorporation of more sophisticated cost measurements, and are a key component of automatically determining the impact of intrusions and responses.

Defining in concrete terms the measure of success and failure for response actions. Permitting the response system to dynamically adjust selected responses based on past behavior requires enabling the system to correctly incorporating success and failure to specific response actions. Determining generic and scalable mechanisms for ascertaining response success, failure, and correctly ascribing these to response actions is another area where progress is needed. One approach worth pursuing is a federated model of information sharing to allow organizations to contributed past experiences with responses in specific intrusion contexts.

Explore the use of hierarchical categories for intrusions / responses. The use of a hierarchical assessment of intrusions and responses is also a noteworthy research direction. Automating the classification of intrusions or responses, and incorporating this hierarchy into the cost assessment framework would enable the cost measurement of previously unseen intrusions, and also support reducing the overhead involved in deploying this framework.

Implementation expansion. In addition to the research directions mentioned above, planned implementation expansion includes the deployment of a network-based cost-sensitive response framework at the Department of Energy's Ames Laboratory at Iowa State University, and discussions with the broader DOE community about deploying the framework in support of a federated intrusion response architecture. Both of these efforts will require considerable

additional testing and enhancements to the current design and implementation of our intrusion detection and response system.

BIBLIOGRAPHY

[1] D. Armstrong, G. Frazier, S. Carter, and T. Frazier. A Controller-Based autonomic defense system. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03)*. IEEE Computer Society, 2003.

[2] I. Balepin, S. Maltsev, J. Rowe, and K. Levitt. Using specification-based intrusion detection for automated response. In *Proceedings of RAID*, pages 136–154. Springer, 2003.

[3] T. Brugger and J. Chow. An assessment of the darpa ids evaluation dataset using snort. Technical Report CSE-2007-1, University of California, Davis, January 2007.

[4] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, 28(1):20–28, 1979.

[5] B. Foo, M. W. Glause, G. M. Howard, Y.-S. Wu, S. Bagchi, and E. H. Spafford. *Intrusion Response Systems: A Survey*, chapter 13, pages 377–412. Morgan Kaufmann Publishers, 2008.

[6] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. H. Spafford. ADEPTS: Adaptive intrusion response using attack graphs in an e-commerce environment. In *Proceedings of DSN*, pages 508–517, 2005.

[7] S. Fu-Xiong. ERRORS ESTIMATING OF INCOMPLETION AND UPDATING STRATEGY IN IDS. In *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*. IEEE, 2006.

[8] A. Gehani and G. Kedem. Rheostat: Real-time risk management. In *Proceedings of RAID*, pages 296–314, 2004.

[9] M. Jahnke, C. Thul, and P. Martini. Graph based metrics for intrusion response measures in computer networks. In *Proceedings of the IEEE LCN*, pages 1035–1042, 2007.

[10] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok. Toward cost-sensitive modeling for intrusion detection and response. *J. Comput. Secur.*, 10(1-2):5–22, 2002.

[11] W. Metcalf and V. Julien. Snort_inline. World Wide Web electronic publication, 2009.

[12] S. Misslinger. *Internet Worm Propagation*. Techn. Univ. of Munich, 2005.

[13] P. G. Neumann and P. A. Porras. Experience with EMERALD to date. In *Proceedings of the USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 73–80, 1999.

[14] P. Palazzoli and M. Valenza. Snort_inline as a solution. *hakin9*, 06/2006, Dec. 2006.

[15] D. Ragsdale, C. Carver, J. Humphries, and U. Pooch. Adaptation techniques for intrusion detection and intrusion response system. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 2344–2349, 2000.

[16] M. Rash. fwsnort: Application layer ids/ips with iptables. World Wide Web electronic publication, 2009.

[17] M. Rash. psad: Intrusion detection and log analysis with iptables. World Wide Web electronic publication, 2009.

[18] T. Ryutov and C. Neuman. Gaa-api home page. World Wide Web electronic publication, 2009.

[19] T. Ryutov, C. Neuman, and D. Kim. Dynamic authorization and intrusion response in distributed systems. In *Proceedings of the Conference on Policies for Distributed Systems and Netowrks (POLICY 2002)*. IEEE, 2002.

[20] D. Schnackenberg, H. Holliday, R. Smith, et al. Cooperative intrusion traceback and response architecture. In *Proceedings of the IEEE DARPA DISCEX I*, 2001.

[21] A. Somayaji and S. Forrest. Automated response using system-call delay. In *Proceedings of the USENIX Security*, 2000.

[22] N. Stakhanova, S. Basu, and J. Wong. A cost-sensitive model for preemptive intrusion response systems. In *Proceedings of the IEEE AINA*, pages 428–435, 2007.

[23] N. Stakhanova, S. Basu, and J. Wong. A taxonomy of intrusion response systems. In *International Journal of Information and Computer Security*, volume 1, pages 169–184, 2007.

[24] T. Toth and C. Kruegel. Evaluating the impact of automated intrusion response mechanisms. In *Proceedings of the ACSAC*, page 301, Washington, DC, USA, 2002. IEEE Computer Society.

[25] S.-H. Wang, C. H. Tseng, K. N. Levitt, and M. Bishop. Cost-sensitive intrusion responses for mobile ad hoc networks. In *Proceedings of RAID*, pages 127–145, 2007.

[26] J. Wayne. Draft directions in security metrics research. Technical Report NISTIR-7564, NIST National Institute of Standards and Technology, April 2009.

[27] H. Wei, D. Frinke, O. Carter, and C. Ritter. Cost-benefit analysis for network intrusion detection systems. In *Proceedings of the Computer Security Institute (CSI) 28th Annual Computer Security Conference*, 2001.

[28] G. White, E. Fisch, and U. Pooch. Cooperating security managers: A peer-based intrusion detection system. In *IEEE Network*, volume 10, pages 20–23, 1996.

[29] Y. Wu and S. Liu. A cost-sensitive method for distributed intrusion response. In *Proceedings of CSCWD*, pages 760–764, 2008.

[30] Y.-S. Wu, B. Foo, Y.-C. Mao, S. Bagchi, and E. Spafford. Automated adaptive intrusion containment in systems of interacting services. In *To appear in Journal of Computer Networks*, 2007.

[31] C. C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147, New York, NY, USA, 2002. ACM.