REALIZING CYBER RESILIENCE WITH HYBRID INTRUSION TOLERANCE ARCHITECTURES

by

Ajay Nagarajan
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

_____	Dr. Arun Sood, Dissertation Director
_____	Dr. Duminda Wijesekera, Committee Member
_____	Dr. Andrew Loerch, Committee Member
_____	Dr. Foteini Baldimtsi, Committee Member
_____	Dr. Sanjeev Setia, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date: _____	Spring Semester 2017 George Mason University Fairfax, VA

Realizing Cyber Resilience with Hybrid Intrusion Tolerance Architectures

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Ajay Nagarajan
Master of Science
George Mason University, 2010
Bachelor of Engineering
Anna University, Chennai, India, 2007

Director: Arun Sood, Professor
Department of Computer Science

Spring Semester 2017
George Mason University
Fairfax, VA

DEDICATION

For my parents Dr. Jagadha and Nagarajan who have provided me with every opportunity to pursue my dreams. For my sister Dr. Aarthi and my life coach Dr. Krishna who have inspired and supported me throughout this endeavor. For my loving wife Shruti who has been my pillar of strength. For the memory of my grandmother Bagyam.

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful to Dr. Arun Sood for guiding and mentoring me throughout this process. Dr. Sood introduced me to Intrusion Tolerance and taught me how to conduct research, present findings, write technical publications and think critically. Dr. Sood has been an inspiration and a constant source of guidance and support for me at all times.

I would also like to thank Dr. Duminda Wijesekera, Dr. Andy Loerch, Dr. Richard Carver, Dr. Foteini Baldimtsi and Dr. Hakan Aydin for all their guidance and their time serving on my dissertation committee. Their valuable mentorship and feedback helped shape this dissertation to its current state.

I should also thank my sister Dr. Aarthi and my brother-in-law Dr. Krishna who were monumental in me pursuing and completing my doctorate at George Mason University.

I would be remiss if I did not thank my wife Shruti who has constantly supported me and stood by me at all times, good and bad through the years.

Finally, I would like to acknowledge the hand of God, without which none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

REALIZING CYBER RESILIENCE WITH HYBRID INTRUSION TOLERANCE ARCHITECTURES

Ajay Nagarajan, M.S.

George Mason University, 2017

Dissertation Director: Dr. Arun Sood

The current approach to security is based on perimeter defense and relies on firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS). These systems require a priori information about attack patterns and system vulnerabilities. With ever-increasing bandwidth and thousands of unique malware signatures coming out every day, it is becoming impractical to prevent every intrusion. And so, intrusion tolerance assumes that intrusions are inevitable and focuses efforts on minimizing the impact of intrusions. The variety and complexity of cyber-attacks is increasing. Various industry data breach investigation reports suggest that customized malware are difficult to detect and data ex-filtration often occurs over a period of days, weeks and months. The attackers' strong motivation leads to organized and targeted cyber-attacks. The current intrusion detection and prevention approaches are reactive in nature and inadequate to prevent all attacks.

Given the clear need to design intrusion tolerant architectures, my work focuses on extension and application of recovery driven intrusion tolerance systems that make the attacker work harder by reducing the server's exposure time to the internet. This approach relies on using hybrid architectures that combine reactive and proactive systems to protecting the cyber infrastructure. My research framework entails a) Proposing hybrid architectures founded on SCIT, a recovery driven intrusion tolerance approach; b) determining the influencing factors of each hybrid strategy and studying the impact of their variations within the context of an integrated intrusion defense strategy; c) defining economic models to assess the efficacy of proposed hybrid architectures; d) using mathematical models to evaluate proposed hybrid architectures and assess optimal operational parameters; and e) validating research using test bed experiments and simulations outlining impact of proposed architectures on system attack surface and intruder work factor.

To system architects and executive management alike, this work can constitute as the basis for making informed decisions while piling layers of security as part of defense-in-depth strategy.

CHAPTER ONE - INTRODUCTION

This Introduction Chapter describes the motivation for my research and the contributions of this Dissertation in the areas of realizing cyber resilience using hybrid intrusion tolerance architectures and evaluating their security benefits.

1.1 Motivation

Today's approach to security is based on perimeter defense and relies heavily on firewalls, Intrusion detection systems (IDS) and Intrusion prevention systems. Despite years of research and investment in developing such reactive security methodologies, our critical systems remain vulnerable to cyber-attacks. Present reactive security strategies like Firewalls, Intrusion Detection Systems (IDSs) and Intrusion Prevention systems have issues of false alarms, missed detections, inaccurate reports, and delays between compromise and detection. The variety and complexity of cyber-attacks is increasing, along with the number of successful intrusions to mission and business systems. Recent breach reports like Office of Personnel Management [OPMB2015] reported system compromise detection in July 2015, whereas the malware had resided in the system for nearly a year before that. So we infer that not only the Intrusion Detection System / Intrusion Prevention System (IDS/IPS) failed to prevent the intrusion, but current systems were not able to detect the presence of the intruder long after the compromise.

Intrusion detection is known to be a hard problem, and current cyber defense systems reportedly detect less than half the malware. Still servers and apps account for majority of the total records compromised. Verizon DBIR 2015 [Veri2015] underscores this problem by noting that only 9% of the compromises were detected within minutes or seconds. The others took hours, days, weeks and in some cases months. Thus, current cyber defenses cannot protect systems against customized malware and other zero day attacks; once an attack is successful, it can persist for many weeks. This emphasizes the need for a recovery-based Intrusion Tolerance approach since detection triggered Intrusion Tolerance Systems (ITS) might again fall short of the needs.

Current Information Technology systems operate in a relatively static configuration and primarily focus on intrusion avoidance. For example, names, addresses, software stacks, networks, and various other configuration parameters remain static over extended periods of time. At the same time the variety of malware is increasing - Symantec reports [Syma2016] identifying more than a million new unique pieces of malware each day. Thus preventing all intrusions is very hard. We believe that intrusions are inevitable. Current experience shows that in spite of prevention devices, the criminals are able to ex-filtrate data and damage systems. Industry studies by Verizon [Veri2015] and Mandiant [Mand2013] show that criminals are often in the compromised systems for months. According to the 2013 Verizon Business Data Breach Investigation Report [Veri2013], the average time an intruder resides in a system from initial compromise to the point of discovery is more than 34 days. The current static server approach is a legacy

design striving solely for simplicity and performance despite the increasing concern of malicious exploitation of system vulnerabilities.

Moving Target Defense (MTD) is the idea of managing change across various system and network dimensions in order to increase the intruder work factor by increasing the intruder work complexity and decreasing visibility of systems to the intruders. Traditionally MTD strategies have presented two significant challenges to adoption. First, for the sake of security, MTD cannot ignore performance and end user productivity. Most customer facing systems don't have the luxury of adding security that slows down performance. Customers tend to move on if the experience is slow and tedious. Secondly, traditional MTD design generally consists of complex processes involving memory address randomization, network address shuffling, instruction set randomization and more [DEva2011]. All of these techniques are designed to prevent attacks and have the potential to be resource hogs thereby slowing down throughput in certain cases.

SCIT based Moving Target Defense acknowledges that trying to prevent each intrusion is impractical. Therefore, we shift the emphasis to minimizing losses occurring from intrusions rather than preventing intrusions. SCIT systems are designed to be complementary to reactive systems [ANag2010]. Primary goal of SCIT-MTD is to reduce the intruder's window of opportunity to execute an attack and increase the costs of their foot-printing, scanning and attacking efforts. Since by design, the SCIT-MTD attack surface of the system is constantly changing, the system vulnerabilities are difficult to exploit. The process of compromising a system involves identifying system

vulnerabilities and customizing attacks to exploit them. Ever-changing attack surface presents a stiff challenge to the intruders. SCIT – MTD can be used with diversification approaches to further increase the attacker difficulty.

1.1.1 Motivating Examples

As cyber defense efforts increase, passive efforts such as establishing anti-virus software, firewall protection, or improving password strength and encryption, and the organization's workload are constantly challenged by the need to apply patches immediately. Symantec Internet Security Threat Report 2016 suggests that security researchers are uncovering more than a million new unique pieces of malware every day, overwhelming malware analysis resources [Syma2016]. Increasingly, automated analysis technologies are used to keep up with the volume, but they still lack the precision to decipher compressed, encrypted, and obfuscated malware [RBej2005]. McAfee crash of tens of thousands of PCs globally illustrates the unpredictable system effects after compromise and their collateral damage, which creates even more uncertainty and less dependability for Enterprise Security [DKra2010].

The current reactive cyber defense approaches are expensive and inadequate. We expect that, automated recovery and Intrusion Tolerance System (ITS) will be useful in addressing the increasing malware and patch workload, but what are the cost impacts of malicious threats and false positives on dependability and security attributes?

In reports of recent breaches, it has become clear that intruders were in the system for long periods. Not only did the IDS/IPS fail to prevent the intrusion, these systems

were not able to detect the presence of the intruder. To illustrate this point, we refer to the following data breach reports:

1. Verizon 2016 Data Breach Investigation Report [Veri2016] focuses on over 100,000 incidents that occurred in 2015. The report suggests that the average time to compromise (time taken by the adversary to exploit a vulnerability and compromise a system) is in the order of minutes, however, the average time to discover (time taken by the defender to discover the compromise / breach) is still in the order of days. Therefore, the intruder is in the compromised system for a prolonged duration before the compromise is detected and system recovery initiated.
2. Following are some recent security breach events that illustrate detection delay when it comes to detecting compromise:
    - Home Depot reported in September 2014 – Time to Discover 5 months [Home 2014]
    - PF Chang's reported in July 2014 – Time to Discover 11 months [PFCh2014]
    - Sony reported in Nov 2014 – Time to Discover ~ 1 year [Sony2014]
    - Office of Personnel Management (OPM) reported in July 2015 – Time to Discover ~ 1 year [OPMB2015]
3. Network Solutions breach [NetS2009] of June - July 2009 resulted in 600,000 records compromised and the breach was detected after 2 months.

4. Wyndham Hotels breach [Wynd2010] was detected in January 2010, with an estimated start date of October 2009.

From these typical data, we conclude that any strategy that will shorten the duration of the breach would lead to better protection of services and data.

A new approach has slowly emerged during the past decade, and gained impressive momentum recently: intrusion tolerance (IT). That is, handling— react, counteract, recover, mask— a wide set of faults encompassing intentional and malicious faults collectively called as intrusions, which may lead to failure of the system security properties if nothing is done to counter their effect on the system state. In short, instead of trying to prevent every single intrusion, these are allowed, but tolerated: the system has the means to trigger mechanisms that prevent the intrusion from generating a system failure. One such intrusion tolerance approach is Self-Cleansing Intrusion Tolerance (SCIT) [YHua2006].

1.2 Contribution

Motivated by the challenges of providing practical intrusion tolerance solutions in conjunction with existing detection and prevention strategies, the contributions of this dissertation can be summarized as follows: "Apply Redundancy, Diversity and Re-configuration techniques in proposing, designing and validating hybrid intrusion tolerance architectures that extend traditional SCIT systems to enable resilience, restoration and recovery of Information Systems at a granular level".

Below is a summary of contributions made through each chapter in this dissertation. Significance of these contributions and their practical applicability in the space of information security is laid out in section 1.3 that follows.

a) Chapter 2: Survey intrusion tolerance architectures and classify them using a taxonomy based on the algorithms and technology used to provide resilience to systems, applications and services. Intrusion tolerance techniques are broadly categorized as hardware and software based intrusion tolerance. Software based intrusion tolerance is further categorized into detection-based, algorithm-based, recovery-based and hybrid.

b) Chapter 3: Propose a framework to assess the performance of security architectures in terms of reducing data ex-filtration. Propose hybrid approaches that combine recovery-driven intrusion tolerant SCIT architecture with existing IDS solutions as part of a multi layered defense strategy to protecting the cyber infrastructure. Specifically, a comparison of the following 4 hybrid architectures is performed from the perspective of minimizing data ex-filtration: (1) Network IDS only; (2) SCIT only; (3) Network IDS + Host IDS; (4) Network IDS + SCIT. The effectiveness of SCIT and IDS security architectures in terms of minimizing data ex filtration losses is analyzed using decision trees and Monte Carlo simulations. From the view point of reducing data ex-filtration we discover that Network IDS + SCIT is the preferred solution. This contribution led to publication [ANag2010].

c) Chapter 4: Establish a framework that uses Receiver Operating Characteristic (ROC) curve analysis and damage cost models to trade-off the true positive rate and false positive rate for comparing alternate security architectures. This framework provides a baseline for making informed decisions and choosing operating parameters for various architectures. In this work, the framework is employed in performing a comparison between an IDS only solutions and an IDS + SCIT hybrid solution. This analysis provides optimal value(s) of Probability of Detection by evaluating the potential damage from a missed intrusion and costs of processing false positives. This research proposes an approach which involves determining the influencing factors of each strategy and studying the impact of their variations within the context of an overall integrated intrusion defense strategy. This contribution led to publication [ANag2011].

d) Chapter 5: Leverage the concept of Attack Surface [15, 16] and its use as a security metric to compare the relative security of multiple security architectures. This research proposes the use of Attack Surface Shifting / Reduction as a metric to compare Moving Target Defenses (MTD) by assessing its impact on intruder work factors. As part of validating this hypothesis, a test bed experiment was conceptualized and built to perform attack surface assessment and compare the following architectures: 1) Static Systems; 2) Basic-SCIT (redundant, not-diverse) and 3) Diverse-SCIT (redundant, diverse). A test-case for assessing impact of dynamic attack surface on intruder work factor is also presented to back up research claims. This contribution led to publication [ANag2014].

e) Chapter 6: Propose a game theoretic attack / protect cyber economic model to facilitate designing architectures that are resilient and tilt the asymmetric cyber economic costs in favor of the defender. This research formalizes system security state transitions and intruder / defender work factors associated with all of those state transitions. A series of test-bed experiments were designed to compare SCIT and non-SCIT security architectures in the face of two sample real world exploits. This component of my research incentivizes logical and architectural solutions that create an ecosystem where the sum of all defender work factors in defending an enterprise over a period of time is much less than the sum of all intruder work factors involved in compromising the enterprise and ex-filtrating data.

f) Chapter 7: Propose µ-SCIT, a hybrid architecture that adds modularity to SCIT using Operating System level virtualization. The proposed architecture is built on top of OpenVZ container-based virtualization for Linux. The added modularity affords the ability to perform more frequent targeted granular rotations at the level of processes and applications. This in turn extends ability of SCIT to work with long running applications and handle long transactions using container check-pointing and migration.

g) Chapter 8: Propose a ‘stand-alone’ and a ‘collaborative’ architecture which make use of information provided by the enterprise Security Information and Event Management (SIEM) solution to perform adaptive intrusion tolerance in unsupervised learning environments. Resilient systems need to be adaptive, and to achieve this goal, this research shows how environmental information can be used

to adaptively change system and operational parameters. In this work, two hybrid architectures a) Stand-alone adaptive SCIT and b) Peer-to-peer collaborative SCIT are proposed that can perform adaptive intrusion tolerance on the basis of real time enterprise health. This contribution led to publication [ANag2012a].

h) Chapter 9: Explored game design for cyber-security training. One of the overlooked aspects of cyber-security is the human factor. Technologies cannot account for human errors and lack of security hygiene. This can only be addressed by security awareness and training. The objective of this research is to teach everyday users the requisite cyber security skills through gaming, beyond the current state-of-practice. Because the skill level of the trainees is also wide ranging, from causal computer users, to software engineers, to system administrators, to managers, the games must also be capable of training this wide range of computer users. Computer games can provide a media for delivering training in an engaging format at levels appropriate for the individual trainees. This work entailed the following components: (1) describe the state of practice by describing the gaming tool used in most cyber challenges at high schools and colleges in the US, i.e., the cyber security gaming tool CyberNEXS, (2) outline some of the additional topics that should be addressed in cyber security training and (3) some other approaches to game design that might prove useful for future cyber security training game development beyond CyberNEXS. This contribution led to publication [ANag2012].

1.3 Significance

As cyber defense efforts increase, passive efforts such as establishing anti-virus software, firewall protection, or improving password strength and encryption, and the organization's workload are constantly challenged by the need to apply patches immediately. Security researchers are uncovering close to 55,000 new malware samples a day, overwhelming malware analysis resources [McAf2010]. With ever growing bandwidth and more people getting access to the internet, it is safe to assume that these security concerns are here to stay.

The current reactive cyber defense approaches are expensive and inadequate. In addition to the cost of licensing and implementing these tools, there is also an ever increasing cost of administering and managing these security tools. All of the detection triggered approaches such as IDS / IPS are plagued by false positives that demand man hours for analysis. Given the number of unique malwares and amount of network traffic today's enterprises are faced with on a daily basis, it is impractical to build an ability to deal with every one of the alerts generated by perimeter devices. As an inference, it is essential to explore techniques that strive to secure an enterprise on an on-going basis irrespective of environment changes or intruder actions.

My research in the area of recovery driven intrusion tolerance is significant due to the following reasons:

a) Industry breach investigation reports consistently highlight the fact that an intruder resides on a compromised environment for extended periods of time (sometime months to years) while ex-filtrating sensitive data. The proposed

hybrid approaches and the results presented with respect to their abilities to thwart data ex-filtration can assist system architects in revisiting their existing enterprise security setups or in designing new ones ground up. The framework presented to assess data ex-filtration potential in a current setup can be used as a tool by system administrators in assessing the health of their environment.

b) The proposed framework that uses Receiver Operating Characteristic (ROC) curve analysis and damage cost models to trade-off the true positive rate and false positive rate for comparing alternate security architectures can be used as an effective tool by system and network administrators in assessing operating parameters for their various security tools. This framework provides a baseline for making informed decisions. A major shortcoming of various modern IDS / IPS solutions is the number of false positives that are generated. These false positives demand substantial man hours in terms of analysis in order to determine whether they are action worthy or not. Analysis using the proposed framework can provide optimal value(s) of Probability of Detection by evaluating the potential damage from a missed intrusion and costs of processing false positives.

c) The proposed µ-SCIT architecture extends the capabilities of SCIT which is currently being used for short to medium transactions to be able to accommodate long running transactions. µ-SCIT also provides a technology solution to perform targeted recovery at a more granular level than SCIT.

d) Models defined and presented as part of this research to assess the impact of security architectures on system attack surface and intruder / defender work factors can help

- system architects design resilient cyber ecosystems;
- system architects explore proactive recovery;
- system / network administrators with assessing efficacy of their current security architecture;
- System / network administrators in performing system and service configuration changes that have meaningful impacts on intruder and defender work factors.

1.4 Dissertation Organization

This Dissertation comprises 10 chapters, including this Chapter 1. Chapter 2 introduces intrusion tolerance and traditional fault tolerance concepts. It classifies intrusion tolerance architectures using a taxonomy that is based on techniques and algorithms used to achieve tolerance. Chapter 3 proposes hybrid architectures that consist of both IDS and Intrusion Tolerance Systems (ITS) and performs a comparison of the proposed architectures using decision trees and Monte Carlo simulation from the viewpoint of containing data ex-filtration. Chapter 4 defines a framework that uses Receiver Operating Characteristic (ROC) curve analysis and damage cost models to trade-off the true positive rate and false positive rate of detection solutions for comparing alternative security strategies. This chapter presents an approach which involves determining the influencing factors of each strategy and studying the impact of their

variations within the context of an overall integrated intrusion defense strategy. Chapter 5 investigates the impact of SCIT based moving target defense architectures (as proposed in the previous chapters) on system attack surface. Chapter 6 highlights the impact of SCIT based moving target defense architectures (as proposed in the previous chapters) on intruder and defender work factors. Chapter 7 proposes µ-SCIT, an architecture which adds modularity to SCIT using container-based virtualization. Chapter 8 introduces intrusion tolerant architectures built on top of SCIT that are self-adapting based on real time enterprise health and threat information feeds from SIEM. Chapter 9 explores the applicability of game design to cyber security training – a key missing link in addressing the human factor quotient of the cyber security paradigm. Chapter 10, in summary, provides conclusions of my dissertation research.

CHAPTER TWO – RELATED WORK

This chapter presents related work from the recent past including a survey of intrusion tolerance architectures that classifies them using a taxonomy based on the algorithms and technology used to provide resilience to systems, applications and services. The Intrusion tolerance techniques are broadly categorized as hardware and software based intrusion tolerance. Software based intrusion tolerance is further categorized into detection-based, algorithm-based, recovery-based and hybrid.

2.1 Overview

There is a significant body of research on distributed computing architectures, methodologies and algorithms, both in the fields of dependability and fault tolerance, and in security and information assurance. These are commonly used in a wide spectrum of situations: information infrastructures; commercial web-based sites; embedded systems. Their operation has always been a concern, due to the use of Commercial Off The Shelves (COTS) products, compressed design cycles, openness. While they have taken separate paths until recently, the problems to be solved are of similar nature: keeping systems working correctly, despite the occurrence of mishaps, which we could commonly call, faults (accidental or malicious); ensure that, when systems do fail (again, due to accidental or malicious faults), they do so in a non-harmful way. In classical dependability, and mainly in distributed settings, fault tolerance has been the subject of

the many solutions published over the years. Classical security-related work has on the other hand privileged, with few exceptions, intrusion prevention, or intrusion detection without systematic forms of processing the intrusion symptoms.

A new approach has slowly emerged during the past decade, and gained impressive momentum recently: intrusion tolerance (IT). That is, the process of handling (react, counteract, recover, mask) a wide set of faults encompassing intentional and malicious faults collectively called intrusions, which may lead to failure of the system security properties if nothing is done to counter their effect on the system state. In short, instead of trying to prevent every single intrusion, these are allowed, but tolerated: the system has the means to trigger mechanisms that prevent the intrusion from generating a system failure.

The term "intrusion tolerance" has been used for the first time in [JFra1985], and a sequel of that work lead to a specific system developed in the DELTA- 4 project [YDes1991]. In the following years, a number of isolated works, mainly on protocols, took place that can be put under the IT umbrella [MCas199, MRei1995, KKih2001, LAlv2000, DMal2001, Gate2000, MHil2001], but only recently did the area develop explosively, with two main projects on both sides of the Atlantic, the OASIS and the MAFTIA projects, doing structured work on concepts, mechanisms and architectures. One main reason is concerned with the fact that distributed systems present fundamental problems in the presence of malicious faults. On the other hand, classical fault tolerance follows a framework that is not completely fit to the universe of intentional and/or malicious faults. These issues will be discussed below.

The purpose of this chapter is to make an attempt to systematize these new concepts and design principles. The chapter describes the fundamental concepts behind intrusion tolerance (IT), tracing their connection with classical fault tolerance and security, and identifying the main delicate issues emerging in the evolution towards IT. We discuss the main strategies and mechanisms for architecting IT systems, and report on recent advances on distributed IT system architectures. For the sake of clarifying our position, we assume an 'architecture' to be materialized by a given composition of components. Components have given functional and non-functional properties, and an interface where these properties manifest themselves. Components are placed in a given topology of the architecture, and interact through algorithms (in a generic sense), such that global system properties emerge from these interactions.

2.2 Classical Fault Tolerance and Security

Dependability has been defined as that property of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behavior as it is perceptible by its user(s); a user is another system (human or physical) which interacts with the former [AAvi1986].

Dependability is a body of research that hosts a set of paradigms, amongst which fault tolerance, and it grew under the mental framework of accidental faults, with few exceptions [JFra1985, JDob1986], but we will show that the essential concepts can be applied to malicious faults in a coherent manner.

Malicious failures make the problem of reliability of a distributed system harder: failures can no longer be considered independent, as with accidental faults, since human

attackers are likely to produce "common-mode" symptoms; components may perform collusion through distributed protocols; failures themselves become more severe, since the occurrence of inconsistent outputs, at wrong times, with forged identity or content, can no longer be considered of "low probability"; furthermore, they may occur at specially inconvenient instants or places of the system, driven by an intelligent adversary's mind. The first question that comes to mind when addressing fault tolerance (FT) under a malicious perspective is thus: How do you model the mind of an attacker?

Traditionally, security has evolved as a combination of: preventing certain attacks from occurring; removing vulnerabilities from initially fragile software; preventing attacks from leading to intrusions. For example, in order to preserve confidentiality, it would be unthinkable to let an intruder read any confidential data at all. Likewise, integrity would assume not letting an intruder modify data at all. That is, with few exceptions, security has long been based on the prevention paradigm. However, let us tentatively imagine the tolerance paradigm in security [AAde2002]:

- assuming (and accepting) that systems remain to a certain extent vulnerable;
- assuming (and accepting) that attacks on components/sub-systems can happen and some will be successful;
- ensuring that the overall system nevertheless remains secure and operational.

2.3 Intrusion Tolerance Concepts

What is Intrusion Tolerance? As said earlier, the tolerance paradigm in security assumes that systems remain to a certain extent vulnerable; assumes that attacks on components or sub-systems can happen and some will be successful; ensures that the

overall system nevertheless remains secure and operational, with a quantifiable probability.

The following subsections outline the most commonly used intrusion tolerance concepts, techniques, algorithms and technology implementations.

2.3.1 AVI Composite Fault Model

The mechanisms of failure of a system or component, security-wise, have to do with a wealth of causes, which range from internal faults (e.g. vulnerabilities), to external, interaction faults (e.g., attacks), whose combination produces faults that can directly lead to component failure (e.g., intrusion). An intrusion has two underlying causes:

- Vulnerability - fault in a computing or communication system that can be exploited with malicious intention
- Attack - malicious intentional fault attempted at a computing or communication system, with the intent of exploiting vulnerability in that system

Which then lead to:

- Intrusion - a malicious operational fault resulting from a successful attack on vulnerability

This well-defined relationship between attack/vulnerability/intrusion is what we call the AVI composite fault model. The AVI sequence can occur recursively in a coherent chain of events generated by the intruder(s), also called an intrusion campaign. For example, a given vulnerability may have been introduced in the course of an intrusion resulting from a previous successful attack.

Vulnerabilities are the primordial faults existing inside the components, essentially requirements, specification, design or configuration faults (e.g., coding faults allowing program stack overflow, files with root setuid in UNIX, naïve passwords, unprotected TCP/IP ports). These are normally accidental, but may be due to intentional actions, as pointed out in the last paragraph.

Attacks are interaction faults that maliciously attempt to activate one or more of those vulnerabilities (e.g., port scans, email viruses, malicious Java applets or ActiveX controls). The event of a successful attack activating vulnerability is called an intrusion. This further step towards failure is normally characterized by an erroneous state in the system which may take several forms (e.g., an unauthorized privileged account with telnet access, a system file with undue access permissions to the hacker). Intrusion tolerance means that these errors can for example be unveiled by intrusion detection, and they can be recovered or masked. However, if nothing is done to process the errors resulting from the intrusion, failure of some or several security properties will probably occur.

2.3.2 Tolerance Techniques

Redundancy, Diversity and Re-configuration are commonly applied principles for fault tolerance against accidental faults. Their use in security is attracting increasing interest; however it is less general and less of an accepted principle. The following section helps better understand how the above principles can be applied to security.

a) Redundancy: Redundancy in general has been long understood to be a valid defense against physical faults. Its application to security however has only become

popular recently. Security encompasses multiple attributes (confidentiality, availability and integrity) and defending against multiple threats. Voted redundancy is often seen as the stereo-typical form of redundancy. In this case, multiple replicas vote on a decision and the decision's outcome will reside upon the majority of the votes. Voted Redundancy is applicable in scenarios where there is a low probability for majority of the replicas to be compromised. Redundancy of Resources is another form of redundancy used in security where ideally the intended service can be provided (possibly in a degraded fashion) if at least 1 out of N redundant resources remains available. Security benefits in terms of availability is quiet apparent in this case. Primary objection for the application of redundancy to security has been that if an attacker can penetrate a certain defense, the same attacker would have no problems penetrating two copies of the same. This leads us to the second principle of diversity.

b) Diversity: Redundancy alone is not enough in most of the cases. Having multiple copies of the same system/defense only creates slight separation. It is straightforward for the attacker who has penetrated one copy to penetrate the other. To avoid this, there is need for further isolation between two copies. Diversity provides this. Diversity is the property that the redundant components should be substantially different in one or more aspects, from hardware diversity and operating system diversity, to software implementation diversity. Additionally, diversity is also applied to time and space, in that diverse services should be co-located at multiple sites to protect against local disasters, and that clients may use time diversity by requesting service at different times. Diversity makes it unlikely for redundant components to be attacked / penetrated at

the same time. This also adds extra workload on the attacker with limited resources. Although increased diversity reduces the risk of correlated faults, it increases the complexity of the system.

c) Voting: Redundancy is a key component in providing any kind of tolerance. As a consequence of having redundant components in ITS system, it is also paramount that the systems' non-faulty components can agree on valid output data in the presence of the faulty components. While all the replicas of a response are considered equally reliable, the output must be based on cross-comparison of available replicas, possibly augmented by knowledge of the application. Voting is used to resolve any differences in redundant responses and to arrive at a consensus result based on the responses of perceived non-faulty components in the system. It has two complementary goals: masking of intrusions, thus tolerating them, and providing integrity of the data. The process involves comparing the redundant responses and reaching agreement on the results to find the "correct" response. Common metrics for comparison are Edit Distance and Hash Code. Edit Distance is useful for comparing data where we need to consider modification (insert/delete/replace) costs. A number of variants of the edit distance computations exist: simple edit distance, hamming distance, episode distance etc. The common UNIX utility diff uses such an approach. Hash Code is a useful metric for large data streams. When computing edit-distance is not-possible or computationally intensive, a digest of the data can be used as a metric. The hash code can be computed using some digest function such as CRC, MD5, or SHA. These metrics are used in agreement

algorithms to arrive at a plausible "correct" response. A leader/delegate usually passes on the chosen replica to the client. Common voting algorithms [PLor1989] include:

- Formalized Majority Voting: This is the most commonly used algorithm, also known as consensus or majority voting. Here, the replicas are partitioned such that the difference between no two replicas in a partition is greater than a threshold. If the partition with the highest number of replica entries forms the absolute majority, one output from that partition is chosen as the final response.
- Generalized Median Voting: In this method, a middle value is selected from the set of N replicas by systematically locating those which differ by greatest amount and eliminating them from consideration.
- Formalized Plurality Voting: This algorithm is similar to formalized majority voting algorithm but for the fact that a relative majority is considered instead of absolute majority.

Voting can be applied at various layers of the networking stack including application layer as well as the middleware layer [AFra2002]. SITAR [FWan2003] uses edit distance comparison [Rupp2002] and formalized majority voting as the primary algorithms. DIT [AVal2002] uses hash code comparison and formalized majority voting in its architecture. However, both architectures adapt to different algorithms based on the security posture at any given time. Some of the common mechanisms used to thwart attacks against this mechanism include diversity, unpredictable leader election and more redundancy.

d) Re-configuration: The occurrence of intrusions and the consistent isolation of faulty components lead to a decrease in the number of available fault-free components. Traditional intrusion detection systems are mostly reactive. The usual response after an intrusion is detected is to perform a post-mortem and take corrective and recovery actions. This is generally a manual task for the administrator and involves some downtime for the server. Survivable systems on the other hand, aim to have none or minimal downtime for the service as far as clients is concerned. They dynamically and adaptively reconfigure the system so that the service can be uninterrupted. Reconfiguration can be proactive or reactive and can help in prevention, elimination as well as tolerance. Reconfiguration can be effected in several different forms:

- Rollover: The affected component is transparently replaced by a pristine replica of it.
- Shifting: All the traffic directed to the affected server is routed to another safe server.
- Load sharing: If the unavailability or degradation in performance is caused by high load, some form of load sharing or balancing may be employed.
- Blocking: If a client is perceived to be offending or is suspicious, the system may decide not to service it.
- Fishbowling: Fishbowling is similar to blocking. Unlike blocking, however, fishbowling allows the targeted user to continue receiving service. But, it protects normal users from being effected by the attacker's intent.

- Changing the system's posture: The system's multiple layers of defense can be turned off/on based on the current operating environments and threat indication.
- Rejuvenation: The affected component is restarted to restore it to a pristine state wiping out any memory resident or volatile attacks.

e) Secret Sharing: Secret sharing (also known as Threshold Scheme) was proposed by Adi Shamir in his classical paper "How to share a secret" [ASha1979]. The general idea is to devise a method to divide data D into n pieces in such a way that it needs k shares to reconstruct original data D, anything less reveals no information at all. This elegant idea has found many applications in key management schemes as well as cryptography. In terms of its application in Intrusion Tolerance Systems (ITS), there are two primary ways of using it. First and in its very native form, data shares can be stored in distributed physical locations such that even if n - k - 1 shares were attacked and compromised, the confidentiality are still kept and original data can be reconstructed, therefore the tolerance. In fact, this form not only employ threshold scheme, redundancy technique also comes into play due to the nature of dispersion of data. Second, data itself can be encrypted with a secret key, and this key is to be divided into n shares using threshold scheme. This form doesn't exactly provide any redundancy to the original data per se, however, to gain access of the information, you do need k shares of encryption key to construct original key, which essentially provide "joint control or custody" of information.

Threshold schemes help ensure confidentiality and survivability. One of the most representative projects is OASIS [JLal2003], a survivable storage system developed at

CMU. PASIS makes use of threshold schemes to analyze trade-offs among security, availability and performance. Draper Laboratory's CONTRA [JLep2003] provides protection and tolerance by camouflaging the messages sent from the source to destination using threshold schemes. COCA [LZho2002] also relies on threshold scheme to tolerate faults.

f) Indirection: Indirection is a common technique in computer science. In intrusion tolerance, it is often layered, and occurs at several levels. Indirection allows designers to insert protection barriers and fault logic between clients and servers. Also, since the indirection is hidden outside of the black box system, clients see only what looks like a COTS server. There are at least four main types of indirection used by intrusion tolerant systems: proxies, wrappers, virtualizations, and sandboxes. They are briefly summarized below:

Proxies: A proxy server, usually transparent, is often the first line of defense of a system. The proxy server accepts all client requests, and uses its own logic to perform a variety of functions, including load-balancing, validity testing, signature based testing, and fault masking. The proxy acts as the sole client access point, hiding all behavior behind the proxy from clients. However, one caveat is that proxy efficiency is paramount to prevent performance bottlenecks. A Scalable Intrusion Tolerance Architecture for Distributed Services (SITAR) and Hierarchical Adaptive Control for QoS Intrusion Tolerance (HACQIT) [HAcq2002] use proxies to export the server interface to clients, protecting their many functions. Both are a kind of firewall and load balancing proxy. Since the proxy is the client endpoint, it is a likely target of attack.

Wrappers: Wrappers are most commonly placed directly around servers (or other wrappers), and inspect requests and responses before sharing them with other components (but not end clients). The wrapper also differs from the proxy in its intimate knowledge of the server. Though a single proxy/wrapper can be the sole line of defense, commonly the wrapper is behind other indirections, and is used to add functionality to a server without changing the COTS server itself. Wrappers are commonly employed, such as in SITAR [FWan2003] and Willow [JKni2002]. SITAR uses wrappers to allow COTS servers to speak a SITAR internal language, and Willow's uses wrappers to augment the abilities of servers. Since the wrappers are treated by the rest of the system as the COTS servers, arguably this addition does not add substantial burden to protection of the COTS servers.

Virtualizations: Generally speaking, virtualizations are naming indirections and are often used subtly, without glorification. When requesting a new virtualized service, the indirection happens as the virtual name is translated into a real name, allowing the real service to be referenced. Thereafter, direct access is allowed, reducing the performance cost of indirections (but then references are not moderated). Some examples of common virtualizations include memory subsystems, the DNS system, and RPC. By virtualizing the names, the details are delayed until needed, and changed as appropriate. In our review, RFITS [RRam2000] and ITSI [DBri2003] exhibited noteworthy virtualizations. RFITS is essentially based on virtualization (specifically, the existence of a large namespace from which mappings can be dynamically created and changed between an endpoints virtual channel and the actual channel they communicate with). By

detecting flood attacks and negotiating these changes unpredictably between endpoints, RFITS can survive much denial of service flood attacks. RFITS uses cryptography to protect these negotiations. ITSI also uses virtualization, as an alternative to a proxy, by having multiple hardware interfaces share the same MAC address, and using another technique to determine which interface is the true recipient. Since ITSI uses a hardware implementation, its virtualization is not vulnerable to many types of attacks.

Sandboxes: Sandboxes are common tools used to separate users, servers, and other untrusted components. Essentially, the idea is to run each untrusted component within a sandbox, where all interactions with other systems and subsystems are moderated (and usually significantly restricted). Faults can then be tolerated by the sandbox, by rolling back system state, instantiating a new component to respond, or rejecting the requester (or a variety of other methods). Sandboxes thus provide a window between the untrusted execution, and its results taking effect. If a fault can be detected before the results are committed, they can be safely aborted. Sandboxes are commonly used to protect against faulty mobile code, and to test and diagnose suspected attacks, because the faulty behavior can be limited to within the sandbox. Two of the projects reviewed, ITSI and HACQIT, use sandboxes. ITSI's fishbowling is one of the possible outcomes of reconfiguration, where the communication with servers is protected within its fishbowl. HACQIT uses a sandbox as an analysis workbench. Whenever a possible intrusion is detected, the logs leading up to it and the compromised server are transferred to the sandbox, and it determines an attack signature within the safety of a sandbox without further risk to critical systems.

2.3.3 Algorithms commonly used in Intrusion Tolerance Systems

Byzantine Fault Tolerance: The object of Byzantine fault tolerance is to be able to defend against Byzantine failures, in which components of a system fail in arbitrary ways. Correctly functioning components of a Byzantine fault tolerant system will be able to correctly provide the system's service assuming there are not too many Byzantine faulty components. In the case of 'f' byzantine faulty nodes, at least '2f + 1' total nodes (f+1 good nodes) are required for the system to behave in a fault tolerant manner.

Fragmentation-Redundancy-Scattering: Fragmentation-redundancy-scattering provides intrusion tolerance by assuring confidentiality, integrity and availability in the case of an intrusion. Fragmentation – split the data into fragments such that isolated fragments contain no significant information. Redundancy- add redundancy so that fragment modification or destruction would not affect legitimate access. Scattering – isolate individual fragments such that not all of them can be compromised at the same time.

Markov Decision Process (MDP): MDP provide a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker. At each time step, the process is in some state, and the decision maker may choose any action that is available in states. The process responds at the next time step by randomly moving into a new state, and giving the decision maker a corresponding reward. The probability that the process moves into its new states' is influenced by the chosen action. Specifically, it is given by a state transition function. Thus, the next states' depends on the current state sand the decision maker's

action a. In this manner, MDP's help formalize the problem of intrusion tolerance by adding model parameters like state transition probabilities and associated costs. For example, an MDP can capture the cost of recovering a system from a 'compromised' state to a 'clean' state.

Artificial immune systems (AIS) are computational systems inspired by the principles and processes of the human immune system. The algorithms typically exploit the immune system's characteristics of learning and memory to solve a problem. Artificial immune system is developed around defense similar to Intrusion Tolerance Systems. The idea of applying artificial immune to intrusion tolerance systems is to make ITS' adjust and learn according to the extent of damage caused by an attack.

2.3.4 Technologies commonly used in the implementation of Intrusion Tolerance Systems

Virtualization: The use of virtualization technology has become popular in the recent years. There are now architectures which use virtualization for realizing intrusion tolerance in network-based services. Furthermore, hypervisor allows the implementation of efficient proactive recovery strategies to cope with undetectable intrusions.

Cloud Computing: Cloud computing distributes computing tasks to virtual resource pools which are constituted of a large number of computers, and cloud computing ensures that various applications can get access to computing power, storage space and various software services when needed. Given the need for replication and diversity to implement effective intrusion tolerance, ITS' almost always are resource demanding. Given the plethora of resources available in a cloud computing environment, intrusion tolerance can be best executed.

Hardware Technologies like FPGA: With impressive advances in hardware technology over the past few years, a new interest has developed in hardware based intrusion tolerance. One such hardware technology is Field Programmable Gate Arrays (FPGA). It is an integrated circuit which is designed to be configured by the customer/end-user after manufacturing. The ability to update its functionality offers advantages for many applications. With respect to intrusion tolerance, dynamically re-configurable FPGA's have been used to provide self-healing in the event of an intrusion. Other hardware devices like security enhanced Chip Multiprocessor (CMP) have also been used to achieve hardware based intrusion tolerance.

2.4 Intrusion Tolerance Systems Taxonomy

This section will employ the following taxonomy to classify intrusion tolerance architectures:

- Hardware-based intrusion tolerance
- Software-based intrusion tolerance
- Detection-triggered
- Algorithm based
- Recovery based
- Hybrid

2.4.1 Hardware Based Intrusion Tolerance

In a modular and distributed systems context, hardware fault tolerance today should rather be seen as a means to construct fail-controlled components, in other words, components that are prevented from producing certain classes of failures. This

contributes to establish improved levels of trust-worthiness, and to use the corresponding improved trust to achieve more efficient fault-tolerant systems. Distributed algorithms that tolerate arbitrary faults are expensive in both resources and time. For efficiency reasons, the use of hardware components with enforced controlled failure modes is often advisable.

In [RSha2009], the authors propose a novel self-healing IDS using dynamically reconfigurable FPGA based hardware to provide confidentiality, data integrity, authentication and non-repudiation. To make the designed reconfigurable IDS fault tolerant, a self-healing autonomous restructuring algorithm is used. The moment an internal fault is detected, the faulty module is replaced by the spare unit both functionally and structurally. This self-healing of hardware is implemented with the help of four cores, doing the task of fault identification, spare module identification, and structural and functional information detection and finally restructuring. Target application – Shared internet resources; Technology used – FPGA hardware.

In [SWei2005], the authors propose a system design using a chip multiprocessor (CMP) to provide intrusion tolerance and self-recovery for server applications. It uses a multi-point defense and recovery system to defeat remote exploits. A checkpoint based approach is employed to recover server applications under attack. It takes a snapshot of the application's context and memory state before it handles the next request. If the request turns out to be malicious, the system discards the malicious request and rolls back the application's state to a known good one through check-pointing. Target application – Server applications; Technologies used – Multi-core processors, Virtualization.

Self-Cleansing Intrusion Tolerance/ Hardware Enforced Security (SCIT/HES) [DArs2007] proposes a scalable hardware framework that complements the software components of SCIT to enforce and guarantee the six SCIT primitives, also presented in the paper. Non-violation of these SCIT primitives results in efficient intrusion tolerance.

2.4.2 Software Based Intrusion Tolerance

Software based intrusion tolerance can be classified into the following categories.

2.4.2.1 Detection Triggered

In these ITS architectures, recovery mechanism / tolerance is triggered on detection of an intrusion / error. Most of these ITS' depend on intrusion detection systems for detection purposes. There are a number of ITS' that are detection-triggered, some of which are briefly explained below. The major drawback of such ITS architectures is their dependence on the IDS. If the IDS does not happen to detect an intrusion, the tolerance mechanism will not be triggered into place thereby failing to tolerate that particular attack.

SITAR [FWan2003] proposes an intrusion tolerant architecture for distributed services, especially COTS servers. Emphasis of SITAR is on providing continued availability. SITAR employs redundancy, diversity and adaptive reconfiguration to achieve tolerance with the use of byzantine fault tolerance algorithm and voting. SITAR is detection triggered and depends on the ability to detect compromises.

DPASA [PPal2007] provides an architecture for survivable systems with multi-layer defense for preventing intrusion and for detecting / responding to intrusions that cannot be prevented. The system uses proxies to create a high barrier to entry for an

attacker, continually monitors all parts of the system at all layers and supports dynamic reconfiguration of the system to recover from damages after the attack.

The Willow architecture [JKni2002] is a survivable architecture that combines fault avoidance, elimination and tolerance. It has a powerful reconfiguration mechanism along with a general control structure that continually monitors network state. Distributed IDS monitors all components of the distributed computing environment and triggers the tolerance mechanism once an error / intrusion is detected.

In DIT [AVal2003], the authors propose an adaptive architecture that triggers response mechanisms on detecting intrusions. This architecture consists of hardened intrusion tolerance proxies that mediate client-requests and also an alert management system based on the EMERALD [PNeu1997] intrusion detection framework.

HACQUIT [JRey2003] performs online attack identification by using similarity rules for generalization of attack signatures. HACQUIT claims to protect against previously unknown attacks that are similar to existing attack signatures. COTS supplied design diversity along with fault tolerant techniques are used in this paper to achieve intrusion prevention.

ITSI [DBri2003] presents the approach taken on the Intrusion Tolerant Server Infrastructure (ITSI). ITSI uses smart network interface controllers (NIC) in its implementation to identify and isolate intrusions; prevent them from spreading and also to provide service availability under attack and during recovery. Smart NIC's are based on distributed firewall technology developed by Secure Computing under DARPA's ADF program.

VM-FIT [HRei2007] architecture uses virtualization technology to realize intrusion-tolerant network based services. In this case, the guest OS hosts the service and distributes the requests to a cluster of replicas. The hypervisor is entirely isolated from the guest OS such that an intrusion at the guest OS level would not affect the hypervisor / other trusted components. Hypervisor has complete control over the guest OS such that it can terminate / recover a guest OS if need be.

In [ASai2009], the authors propose an architecture that is based on fault tolerance principles such as redundancy and diversity to improve system resilience to intrusions. The paper proposes a generic intrusion tolerant architecture specifically for web servers that involve a cluster of mediating proxies to handle client requests in order to provide service availability and integrity.

Authors of [ZKal2008] propose a multi-layer architecture for intrusion tolerant web services. The idea is to implement tolerance in case of malicious failures by the use of software fault tolerance techniques. The architecture uses a single service implementation (no redundant system components) and adds some functional capabilities to build an intrusion tolerant web service. In this case, intrusion detection triggers intrusion containment, system recovery and reconfiguration.

ITUA [MCuk2001] looks at developing a middleware based intrusion tolerance solutions that would help tolerate staged attacks. Reconfiguration or adaptation is a key component of how ITUA provides tolerance. It instills unpredictability at different levels of adaptation with the motive to increase the length of time an application can survive an attack.

Intrusion Tolerance model for E-commerce system [YKim2007] proposes an adaptive intrusion tolerance technique for E-commerce systems in particular. In this model, the authors propose an architecture where the application function is separate and the middleware carries out the intrusion tolerance function.

Randomized Failover Intrusion Tolerant System (RFITS) [RRam2000] is a research effort sponsored by DARPA/IPTO which developed survivability design patterns for building Denial of Service (DoS) resistant information systems. Here, the emphasis is on availability of the critical service. Randomized failover makes system posture unpredictable thereby providing enough time for attack neutralization.

CoBFIT [HRam2004] presents a component based framework for building intrusion tolerant distributed systems. In this paper, the authors describe the CoBFIT implementation of a prototype intrusion-tolerant group communication system. The design and implementation of CoBFIT framework include characteristics like portability, re-configurability, flexibility and adaptability that are necessary for system dependability.

2.4.2.2 Algorithm Driven

As opposed to detection triggered intrusion tolerance architectures, the following architectures do not depend on IDS to trigger recovery / tolerance. The following ITS architectures depend on algorithms such as Byzantine Fault Detection, Fragmentation-redundancy-scattering, Markov Decision Process algorithms and Artificial Immune System for providing intrusion tolerance. Although Byzantine fault detection has been commonly used for some time now, the implementation of other algorithms for intrusion

tolerance purposes is fairly recent. Some of the algorithm driven intrusion tolerance techniques are briefly explained below.

OASIS [JLal2003] is a DARPA funded multi-layer intrusion tolerance technique. The goal of OASIS is to "allow sustained operation of mission critical functions in the face of known and future cyber-attacks against information systems". As a part of OASIS, close to 30 projects are funded – one of which is ITUA which was discussed earlier; another project is the hybrid COCA [LZho2002] system which we will briefly describe under Hybrid intrusion tolerance architectures.

MAFTIA [DPow2001] was the first project to use fault-tolerance techniques to build intrusion tolerant applications. The biggest contribution of MAFTIA was proposing an approach to tolerate both accidental faults and malicious attacks. MAFTIA employs Byzantine agreement protocols, threshold cryptography and voting algorithms to tolerate arbitrary failures.

In [MSli2009], authors propose an intrusion tolerance framework based on intermediate signature verification protocol introduced in [MSli2008]. The framework here is specifically meant for heterogeneous wireless sensor networks. In this architecture, intermediate signature verification algorithm is used to detect compromised nodes. Once detected, the compromised node is pushed into a tolerance state after which it is either isolated or recovered.

[TZha2005] proposes a secret sharing based compiler solution to realize intrusion tolerance in secure software. The major contribution of this paper is the introduction of intrusion tolerance in secure software which is critical given their vulnerability. Here,

secret sharing provides better data confidentiality and integrity and the authors also propose mechanisms to recover from data tampering and to achieve intrusion tolerance.

The Starfish System [KKih2003] intends to provide intrusion detection and tolerance for middleware applications in an asynchronous distributed system. The system contains a central highly secure trusted core which is surrounded by "arms" that have fewer security guarantees. In case of a vulnerability / intrusion in an arm, it can be removed from the trusted core. Similarly, new arms can be added to the trusted core as well.

CC-VIT [YTan2010] uses virtualization to construct an intrusion tolerance system for the cloud computing platform. CC-VIT is a modified Byzantine fault tolerant architecture that allows the system to tolerate F faulty replicas in a total of N=2F+1 replicas. The system also ensures that only F+1 replicas are required for proper functioning during the intrusion-free stage.

In the paper [FAnj2000], the author uses Fragmentation-Redundancy-Scattering algorithm to realize intrusion tolerance in a mobile environment. The author also considers the important factor of user mobility while proposing the intrusion tolerance scheme.

Zhilei Cui et al [ZCui2009] look at applying artificial intelligence concepts to intrusion tolerance. Based on artificial immune systems, the authors propose an intrusion tolerant system that can adapt and learn depending on the extent of damage caused by an intrusion. Authors also propose the construction of a behavioral database rather than updating the huge virus database frequently.

Patrick Kreidl [OKre2010] proposes a simple Markov decision process model for intrusion tolerance under the assumptions that every attack has to bypass a number of steps before the system gets compromised and that the defensive systems in place cannot prevent all the attacks. The author also uses simulation experiments to study costs tradeoff between system performance and security.

Fault and Intrusion Tolerance in Object-Oriented Systems [BRan1991] briefly discusses the technique called Fragmented Data Processing (FDP) which is used to enhance the security of information in a distributed computing environment. This paper proposes the application of FDP on object-oriented systems to better provide intrusion tolerance to application programs. FDP here is strongly related to the traditional Fragmentation Redundancy Scattering (FRS) fault tolerance technique.

Ineffective damage containment on a compromised critical database can potentially make the database useless. To counter this problem, authors of [PLui2001] present a multi-phase damage confinement approach with the first phase or the confining phase aggressively confining the damage and the following un-confining phases subsequently relax confinement. The aggressive confining phase makes sure that the damage does not spread beyond the first phase and in the process can cause loss of service availability.

2.4.2.3 Recovery Based

Recovery based systems are proactive intrusion tolerance systems. These ITS architectures function under the assumption that every system that is exposed to the internet is compromised. Irrespective of whether or not an IDS triggers an alert, recovery-

based ITS architectures periodically restore the system to the last known good configuration to avoid sustained presence of attacker on the compromised system. Although recovery based ITS architectures do not require an IDS as part of the framework, they could complement each other to achieve defense in depth.

A Rejuvenation Methodology of cluster recovery [KAun2005] presents a cluster recovery model based on concept of Software Rejuvenation. Software rejuvenation is a technique for dealing with software faults and performance degradation by refreshing or restarting it. The proposed model provides the luxury of deciding which application components are vulnerable to longevity flaws and choosing them alone for rejuvenation. This in turn increases the availability of the service as well as reduced losses due to down time.

Self-Cleansing Intrusion Tolerance (SCIT) [YHua2006] employs a cluster of servers each providing identical services. Using round-robin cleansing, at any point in time, a server in the cluster can have one of three states: offline cleansing, offline spare and online transaction processing. A SCIT server A is exposed to the internet for a period of time known as “Exposure Time” after which another server B in the cluster takes its place. Once server A comes offline, it is cleansed and restored to its last known good configuration.

In FOREVER [PSou2008], the authors introduce a service that can be used to improve the resilience of intrusion tolerant replicated systems by tolerating an arbitrary number of faults. This is achieved by using both recovery and evolution techniques. Recovery techniques in FOREVER include time-triggered periodic recoveries and event-

triggered recoveries. Once a recovery is performed, evolution techniques are used to modify the respective vulnerabilities that may be exploited by a malicious attacker.

Hans P. Reiser and Rudiger Kapitza, the authors of [HRei2007] review the benefits of using a hypervisor-based replication infrastructure for implementing proactive recovery. They propose a proactive recovery system that uses virtualization to create a new system image before shutting down the one to be recovered. This is a stateless replication system, idea of which is to minimize system unavailability.

In SPARE [RKap2009], the authors propose an approach that uses virtualization support as typically found in the cloud environment to reduce the resource demands of performing Byzantine Fault Tolerance. They also propose the use of spare replicas that are periodically updated in a suspended state to aid in proactive recovery which helps maximize availability. There are a number of related publications by the authors under the umbrella REFIT: Resource-Efficient Fault and Intrusion Tolerance [TDis2011] [TDis2011a] [TDis2010] [RKap2010].

Worm-IT [MCor2007] proposes a new intrusion tolerant group communication system with membership service. Worm-IT is a multi-node system and can tolerate an arbitrary number of malicious nodes. Worm-IT does not require failure detection of primary-members of the group communication system.

2.4.2.4 Hybrid

Some systems combine two or more of the techniques discussed above to provide a hybrid solution for intrusion tolerance. We will briefly discuss some of these hybrid intrusion tolerance solutions.

P. Sousa et al, the authors of [PSou2010] propose an approach where reactive mechanisms would complement existing proactive recovery techniques to build an intrusion tolerant replicated system that is highly resilient to faults. The reactive mechanisms give the non-faulty replicas the capability to detect other replicas getting compromised. The proactive-reactive recovery service is designed based on a hybrid distributed system [PVer2006].

CloudFIT [HRei2011] is an effort to build an architecture for intrusion tolerant applications that can be deployed dynamically in the cloud. Author also explores the feasibility of applying existing BFT algorithms to increase security and availability in the proposed architecture. In CloudFIT, recovery is handled by a component that can trigger proactive recoveries and also handle event triggered recoveries.

EU CRUTIAL [PSou2009] presents a demonstration of a family of protection devices for critical information infrastructures. These protection devices called CRUTIAL Information Switches (CIS) are responsible for enforcing sophisticated access control policies of both incoming and outgoing traffic. CIS by themselves are intrusion-tolerant and self-healing in order to achieve high resilience. CIS are placed at network boundaries similar to firewalls; however they are responsible for enforcing access control policies on a global scale, all across the interconnected infrastructure.

Cornell Online Certification Authority (COCA) [LZho2002] is a fault-tolerant and secure online certification authority. COCA uses threshold cryptography algorithm to sign the certificates it generates for local and wide area networks. COCA also uses redundancy in the form of server replicas to assure availability. Given there are ‘3t+1’

COCA servers up, COCA may tolerate up to ‘t’ faulty servers as per Byzantine Fault Tolerance.

2.5 Open Problems

Let us analyze a few open problems that arise when intrusion tolerance is viewed from a security or fault tolerance perspective. To start with, what contributes to the risk of intrusion? Risk is a combined measure of the probability of there being intrusions, and of their severity, that is, of the impact of a failure caused by them. The former is influenced by two factors that act in combination: the level of threat to which a computing or communication system is exposed; and the degree of vulnerability it possesses. The correct measure of how potentially insecure a system can be (in other words, of how hard it will be to make it secure) depends: on the number and nature of the flaws of the system (vulnerabilities); on the potential for existing attacks on the system (threats). Informally, the probability of an intrusion is given by the probability of there being an attack activating a vulnerability that is sensitive to it. The latter, the impact of failure, is measured by the cost of an intrusion in the system operation, which can be equated in several forms (economic, political, etc.).

Should we try and bring the risk to zero? And is that feasible at all? This is classical prevention/removal: of the number, power, and severity of the vulnerabilities and the attacks the system may be subjected to. The problem is that neither can be made arbitrarily low, for several reasons: it is too costly and/or too complex (e.g., too many lines of code, hardware constraints); certain attacks come from the kind of service being deployed (e.g., public anonymous servers on the Internet); certain vulnerabilities are

attached to the design of the system proper (e.g., mechanisms leading to races in certain operating systems). And even if we could bring the risk to zero, would it be worthwhile? It should be possible to talk about acceptable risk: a measure of the probability of failure we are prepared to accept, given the value of the service or data we are trying to protect. This will educate our reasoning when we architect intrusion tolerance, for it establishes criteria for prevention/removal of faults and for the effort that should be put in tolerating the residual faults in the system. Further guidance can be taken for our system assumptions if we think that the hacker or intruder also incurs in a cost of intruding. This cost can be measured in terms of time, power, money, or combinations thereof, and clearly contributes to equating 'acceptable risk', by establishing the relation between 'cost of intruding' and 'value of assets'.

A malicious-fault modelling methodology is required that refines the kinds of faults that may occur, and one that does not make naïve assumptions about how the hacker can act. The crucial questions put in this section will be addressed in the rest of the dissertation.

CHAPTER THREE – SCIT AND IDS ARCHITECTURES FOR REDUCED DATA EX-FILTRATION

This chapter proposes a framework to assess the relative performance of different security architectures in terms of their effectiveness in reducing data ex-filtration. The chapter explores various hybrid approaches that combine recovery driven SCIT methodology with existing IDS solutions as part of a multi layered defense strategy to enforce cyber resilience.

3.1 Overview

Today's approach to security is based on perimeter defense and relies heavily on firewalls, Intrusion detection systems (IDS) and Intrusion prevention systems. Despite years of research and investment in developing such reactive security methodologies, critical systems remain vulnerable to cyber-attacks. In this approach, it is assumed that intrusions are inevitable and the effort is focused on minimizing losses. Towards this end a recovery based limited exposure time system called Self Cleansing Intrusion Tolerance (SCIT) is introduced. In this chapter, architectures that combine SCIT architecture with existing IDS approaches are investigated. The effectiveness of SCIT and IDS security architectures in terms of minimizing data ex filtration losses is analyzed using decision trees and the results of Monte Carlo simulation is presented.

The variety and complexity of cyber-attacks is increasing. Verizon 2009 Data Breaches Investigation Report [Veri2009] shows that customized malware is difficult to

detect and data ex-filtration often occurs over a period of days, weeks and months. The attackers' strong motivation leads to organized and targeted cyber-attacks. The current intrusion detection and prevention approaches are reactive in nature and inadequate to prevent all attacks. It is safe to conclude that intrusions are inevitable, and have adopted an intrusion tolerance approach. In [YHua2006, ABan2009] a Self-Cleansing Intrusion Tolerance (SCIT) approach is introduced. SCIT is a recovery driven intrusion tolerance system that makes the attacker work harder by reducing the server's exposure time to the internet.

More recently, a combination of reactive and proactive systems has been proposed [PSou2007]. Such hybrid approaches, with multiple layers of defense is seen as a desirable approach to protecting the cyber infrastructure. In this chapter, the usefulness of adding IDS systems to an intrusion tolerance approach is explored. Specifically, in this chapter a combination of IDS and SCIT architectures is studied. 4 architectures are compared: (1) Network IDS only; (2) SCIT only; (3) Network IDS + Host IDS; (4) Network IDS + SCIT. From the view point of reducing data ex-filtration it is discovered that Network IDS + SCIT is the preferred solution.

The rest of the chapter is divided into 6 sections. In the next section recent reports to motivate this study are discussed. Section 3.3 provides an introduction to SCIT and how it reduces losses. Section 3.4 presents the methodology utilized in this chapter to gauge the effectiveness of a security strategy. Section 3.5 gives an overview of various security architectures compared in this chapter along with decision trees representing

their functionality. Section 3.6 gives an account of the Monte-Carlo simulation, the parameters used and the results obtained.

3.2 Motivating Examples

In reports of recent breaches, it has become clear that intruders were in the system for long periods. Not only did the IDS/IPS fail to prevent the intrusion, these systems were not able to detect the presence of the intruder. To illustrate this point, refer to the following data breach reports:

Verizon DBIR [Veri2009] focuses on 90 studies conducted in 2008. 285 million consumer records were compromised. Some of the parameters used in this chapter are derived from this report. The average Intruder Residence Time (time between system compromise and breach containment) was more than 28 days and on average 675 records were compromised per day.

Following are some recent security breach events that illustrate detection delay when it comes to detecting compromise:

- Home Depot reported in September 2014 – Time to Discover 5 months [Home 2014]
- PF Chang's reported in July 2014 – Time to Discover 11 months [PFCh2014]
- Sony reported in Nov 2014 – Time to Discover ~ 1 year [Sony2014]
- Office of Personnel Management (OPM) reported in July 2015 – Time to Discover ~ 1 year [OPMB2015]

From these incidents, it can be concluded that any strategy that will shorten the duration of the breach would lead to better protection of data files. Consequently, in the analysis, focus is on the estimated records ex-filtrated because of malicious activity.

3.3 SCIT Framework

In [YHua2006] SCIT, an intrusion tolerant technique that provides enhanced server security was presented. SCIT research has focused on critical servers that are most prone to malicious attacks. The technique involves multiple virtual instances of a server. These are rotated and self-cleansed periodically irrespective of the presence or absence of intrusions. Self-cleansing refers to loading a clean image of the server's OS and application into the Virtual Machine. Rotation here refers to the process of bringing an exposed virtual server off-line, killing it, restarting it and in the meanwhile, bringing another virtual server online to assure availability. By doing so, in the event of an intrusion, the intruder is denied prolonged residence on the server. Once the virtual server's exposure time to the Internet is completed, the virtual server instance is automatically rotated by a controller. This virtual instance of the server is what is referred to as virtual server throughout this chapter.

Every virtual server is rotated through 6 states as shown in Figure 3.1a. Active state (or) Exposed state is the state in which the virtual server is on-line. If the exposed virtual server is busy processing an earlier query, the new incoming requests are put in a queue. The queries that are in the queue of a virtual server and are not processed during its exposed state are processed in its quiescent state / grace period. In grace period, no incoming queries are accepted. The virtual server is killed and restarted in the Kill VM / Start new VM states. A virtual server in Online-spare / Live-spare state suggests that it's ready to go on-line. In addition to the states mentioned, there is also an Archive state – in

this state a VM that is no longer exposed and is ready to be killed is archived for offline forensics / future analysis / patching.

VMware is used in this implementation, though the SCIT approach is not reliant on this virtualization approach. The SCIT Controller ensures the constant rotation of the virtual servers.

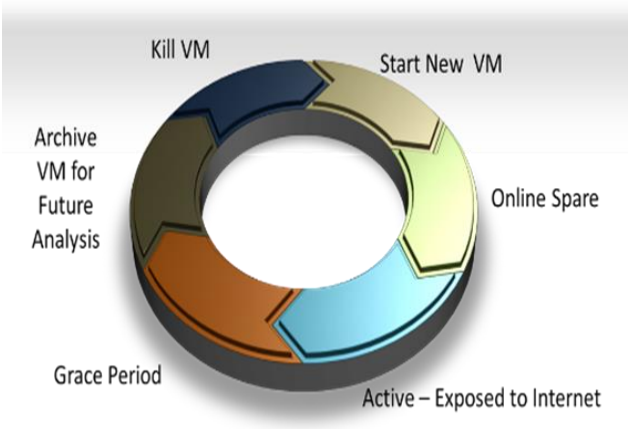


Figure 3.1a: SCIT State Diagram

Figure 3.1b: SCIT Server Rotation

This illustrative example in Figure 3.1b shows 3 different time periods. At any given time, there are five servers online and three servers being wiped clean. In each case a different set of servers is being cleaned. Eventually every server will be taken offline, cleaned and restored to its pristine state. SCIT technology can be used to build a variety of servers that meet enhanced security requirements. It is best suited to servers that are

designed to handle short transactions – the lower the exposure time the shorter the transaction.

3.4 Methodology to calculate data ex-filtration costs

3.4.1 Overview

Four SCIT / IDS architectures are considered. Two alternatives are standalone – NIDS only and SCIT only. In PCI DSS [PCID2014] and in DODi 8500.2 [DoDi2003], host IDS are suggested in addition to Network IDS, thus NIDS + HIDS systems are considered. Finally, NIDS and SCIT are treated. To evaluate the potential losses from each of these systems the approach of [JUlv2003] is followed. Decision trees are developed that represent the functionality of respective security architectures. The conditional probabilities in the decision trees help characterize their security properties. These decision trees are translated into decision guidance systems (DGS) by modeling them on Gnumeric - an open-source spreadsheet software suitable for Monte Carlo simulation. There are 4 DGS' - one each for NIDS, SCIT, NIDS + HIDS, NIDS + SCIT architectures.

The DGS built on top of the decision tree using Gnumeric takes incoming traffic (in terms of queries) as input and divides the traffic into 4 categories: Confirmed Intrusion (CI), Non-intrusions (NI), False Alarms (FA) and Missed Intrusions (MI). Gnumeric's inbuilt Monte-Carlo simulation capabilities are used to generate incoming network traffic. In the case of Intrusions and Missed Intrusions, there would be an Intruder Residence time (IRT) associated with it. Section 3.6 expands on IRT and how it is modeled in the simulation. Using this IRT and the parameters from Verizon DBIR

[Veri2009] from section 3.2, data ex filtration costs in terms of records compromised are calculated.

3.4.2 Assumptions

In the analysis it is assumed that

- In the malicious data ex-filtration process, records are stolen at a uniform rate.
- No records are stolen if the IDS correctly identifies an intrusion.
- There is a constant cost associated with:
    - Performing Intrusion Detection on a single query (incoming traffic) --- C(I)
    - SCIT processing of a query (incoming traffic) --- C(T)
    - Responding to one intrusion alarm --- C(R)

Since the objective is to characterize the effectiveness of the security architecture in terms of least data ex filtrated, constant costs are ignored. However, there is provision in the decision guidance systems to include these costs if need be.

3.5 SCIT/IDS Scenarios

Each of the four SCIT / IDS architectures are considered and are explained briefly. Decision tree representations of each of the architectures are discussed. The decision trees provide a mechanism to estimate costs associated with each of the outcomes (Confirmed Intrusion (CI), Non-intrusions (NI), False Alarms (FA) and Missed Intrusions (MI)). This helps to get a better idea of data ex-filtration costs suffered in each of the IDS and / or SCIT scenarios. It is emphasized that no loss occurs in the case of confirmed intrusion, since IDS detects those.

A number of probability values (p1…p34); (q1…q6) make up the following decision trees, however, it's interesting to note that not all of them contribute equally in determining the outcome. For example, sensitivity analysis performed on the NIDS decision tree suggests that each of the possible outcomes (CI, NI, FA and MI) are most sensitive to change in the value of p1. They are less sensitive to change in the values of q1 & q2. They are least sensitive to change in the values of p4…..p13.

In all the decision trees that follow, (p1…pn) and (q1…qn) represent conditional probabilities.

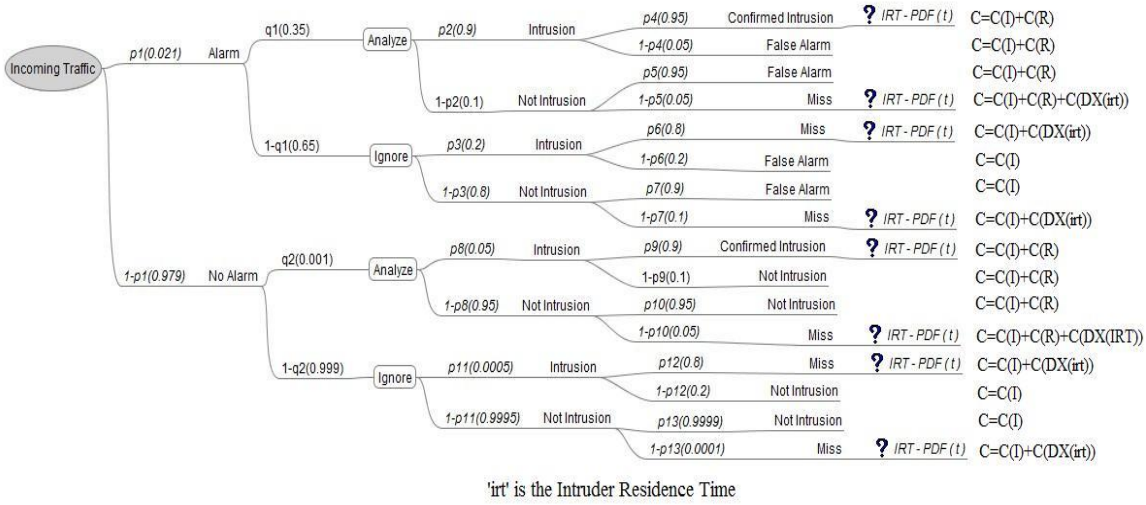


Figure 3.2: NIDS Decision Tree

3.5.1 NIDS

In this case, a stand-alone independent Network Intrusion Detection System (NIDS) security architecture is considered. The decision tree in Figure 3.2 represents NIDS functionality and its effectiveness in finding intrusions and minimizing data ex

filtration. In Figure 3.2, values within braces next to the probability variables represent respective values considered to perform Monte-Carlo simulation. For instance, p1 (0.021) indicates that a value of 0.021 has been utilized for probability variable 'p1' in the simulation. Entire incoming traffic is monitored by the NIDS. Based on what it sees, there is a probability p1 of NIDS triggering an alarm and a probability 1-p1 of NIDS determining the traffic to be safe. In case of an Alarm, a probability q1 is associated with initiating a response and a probability 1-q1 associated with ignoring the Alarm. For instance, intrusions with severity (1, 2) are responded to and alarms with low severity ratings (3 to 6) are ignored. Such decisions are often made in security operations centers because of manpower limitations and the large number of alarms generated by the IDS.

In the case of responding to an alarm and analyzing it, there is a probability p2 that the alarm ends up being categorized as an intrusion and a probability 1-p2 of it being safe. Again, no security procedure in place is ideal, there is an error rate associated with it. For example, traffic which is categorized as an intrusion, in reality could be an intrusion (confirmed intrusion) with a probability of p4 or could be a false alarm (error on NIDS's part) with a probability of 1-p4. A similar explanation follows anything that is categorized as a non-intrusion. On ignoring an Alarm, incoming traffic is let through without further analysis. This traffic in reality could be an intrusion (error on system administrator's part – ignoring the alarm) or a non-intrusion (error on NIDS' part). In this case, intrusions are characterized as Misses and non-intrusions as False Alarms.

In the case of a No-Alarm; the system administrator can still opt to analyze the traffic just to make sure the system is functioning the way it is supposed to. This could be

on the basis of his / her suspicion or could be a random check to determine if all things are well. The procedure that follows is similar to the one discussed in the case of an Alarm.

In cases of Missed Intrusion traffic, damage is done to the system. In these cases, an intruder remains in the system for IRT duration of time causing damage, where IRT is the intruder residence time. In the simulation, the IRT-Probability Density Function (Section 3.6) is used to estimate IRT. In this scenario the amount of damage that could be caused to the system is unbounded, since IRT is unbounded.

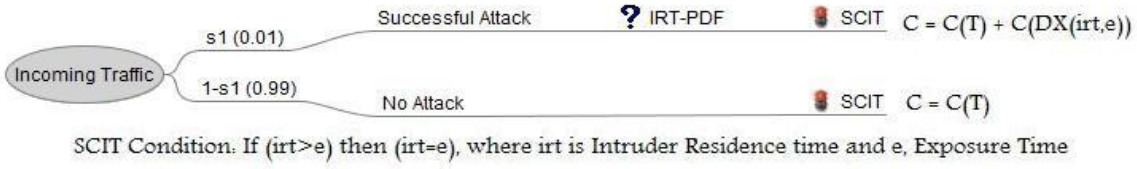


Figure 3.3: SCIT Decision Tree

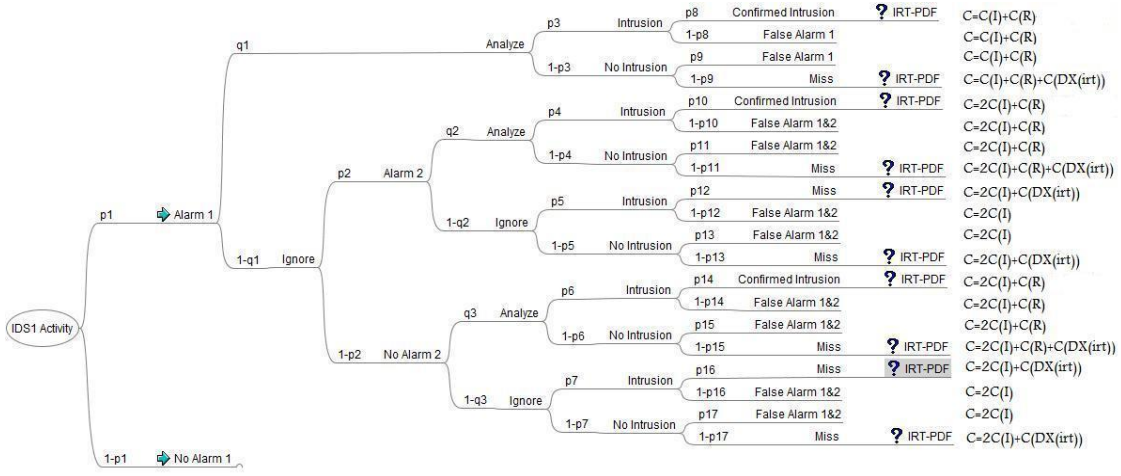


Figure 3.4: NIDS – HIDS Decision Tree

3.5.2 SCIT

The security architecture in this case consists of a standalone SCIT system. There is no intrusion detector in the system. In other words, all potential attacks are successful since there are no IDS / IPS to check for them. Figure 3.3 represents SCIT's decision tree. The incoming traffic is classified as either being a successful attack or not. This is not done by the system since SCIT treats all incoming traffic in the same manner. There is a probability ‘s1’ associated with the incoming traffic being an attack and a probability ‘1-s1’ associated with it being safe traffic. In the case of an attack, the intruder remains in the system for IRT duration of time causing damage.

In the case of incoming traffic being safe, there is no IRT associated with it. Estimation of IRT is provided in section 3.6. In the case of an attack, estimated cost is C(T)+C(DX(irt,e)), where C(T) is the cost of SCIT implementation and C(DX(irt,e)) is the cost of data ex filtrated by the intruder in IRT duration of time. Since SCIT is in place, IRT can never be greater than SCIT's exposure time 'e'. And so the maximum possible damage that can be caused to the system by the intruder is now C(DX(e)) where ‘e’ is the Exposure Time. In the case of safe (no attack) traffic, estimated cost is C(T) and no data loss occurs.

3.5.3 NIDS + HIDS

This architecture is an extension of NIDS. An additional layer of security in the form of Host IDS (HIDS) is added to the system. NIDS+HIDS systems could either have two IDS's running in parallel or have one followed by the other. NIDS+HIDS is considered to be serial, with the NIDS tuned to the network needs, and HIDS tuned to the

specific needs of the host. The first IDS (NIDS) performs its task exactly in the manner illustrated in the case of NIDS in section 3.5.1. If IDS 1 does not trigger an alarm or if IDS 1 alarm is ignored then IDS 2 (HIDS) is run to see if it triggers an alarm (Note, there is a small probability 'q4' of system administrator analyzing the traffic even though IDS 1 does not trigger an alarm. IDS 2 is not run in these cases). This adds another layer of security in the sense that IDS 2 could pick up an intrusion that IDS 1 had missed. According to [JUlv2003], unless one of the IDS' is worthless, it is better to use both in combination than to use single IDS. They suggest that since there is no incremental cost to getting IDS2 report, the expected cost from using an IDS composed of two independent detectors is the same regard-less of whether the response decision is made sequentially or in parallel. In a serial IDS-IDS setup, it is advisable to have the better performing IDS as IDS 1.

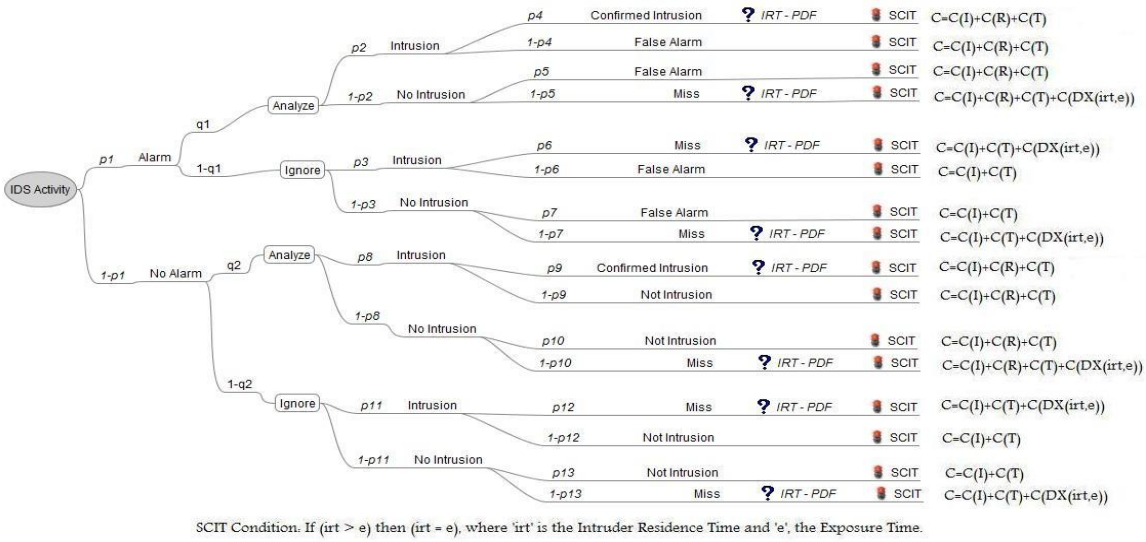


Figure 3.5: NIDS – SCIT Decision Tree

3.5.4 NIDS+SCIT

The system here is an extension of a previous case, NIDS. An additional layer of security - SCIT - is added to the NIDS. In cases where an intruder resides on the system for IRT duration of time, SCIT comes into play. As pointed out, in the case of NIDS, potential damage that can be caused to the system is unbounded. This is primarily because IRT remains unbounded in NIDS. On adding SCIT, IRT is no longer unbounded. SCIT introduces a metric called 'Exposure Time'. Since SCIT is pro-active and performs self-cleansing after time 'e', where 'e' is the Exposure Time; an upper bound is set on IRT. With SCIT the maximum damage C (DX (irt)) that can be caused to the system is C (DX (e)) since (irt <= e). NIDS+SCIT performs better than standalone SCIT since NIDS helps identify certain intrusions before they can cause damage and have to be tolerated.

3.6 Monte Carlo Simulation

Methodology as presented in section 3.4 was followed to perform the Monte-Carlo Simulation. The decision trees represented above are captured in the simulation. The values used for the probabilities have been chosen on the basis of discussions with experienced managers. Certain assumptions were made in the process of simulating the decision trees based on these discussions: A) There are nearly twice as many False Alarms as Confirmed Intrusions and B) Out of the 50,000 incoming queries – 500 are potential attacks (as shown in Figure 3.3). Once the decision trees are incorporated in the Gnumeric spreadsheet format with all probability values plugged in, the inbuilt Monte-Carlo simulation feature in Gnumeric can be used to simulate the incoming traffic. Table 3.1 summarizes the parameters used in the simulation. Primary objective of the

simulation was to compute a mean / total damage cost (in terms of records lost) in each of the SCIT / IDS cases given incoming traffic of 50,000 queries.

The Intruder residence time used in the simulation is modeled as a Pareto distribution. It is assumed that IRT can take values between 0 hours and 2 months with mean being 48 hours. As compared to the examples in Section 3.2, this is a very conservative choice. Using the 28 days average, noted in Section 3.2, would be even more advantageous to SCIT. This average is incorporated in Intruder Residence Time Probability Density Function (IRT-PDF), which gives a relation between IRT values and their respective probabilities of occurrence.

3.6.1 Probability values chosen for the simulation

The values of (q1...q2) and (p1...p13) are the same for NIDS and NIDS+SCIT. These values are presented in Figure 3.2 within parenthesis next to respective variables. In the case of SCIT, probability values are presented in Figure 3.3. In case of NIDS + HIDS, the probability values are given below – variables followed by their value:

q1 (0.35) | q2, q5 (0.1) | q3, p7 (0.01)

p8, p9 (0.95) | p18, q4, q6, p23 (0.001) | p33 (0.9999)

p1 (0.021) | p2,p6,p22,p19 (0.05) | p5,p21 (0.3)

p4,p12,p14,p20,p28,p30 (0.8) | p16,p32 (0.7)

p17,p3,p10,p11,p13,p15,p24,p25,p26,p27,p29,p31 (0.9)

3.6.2 Results of the Simulation

Data loss measured in number of records is the metric for assessing effectiveness of security architecture. The results in Table 3.2 show data ex filtration costs in records.

This table shows that the potential for damage is high for NIDS only and NIDS + HIDS alternatives. The records ex-filtrated are about the same for both scenarios. If SCIT is deployed then the ex-filtration losses are significantly reduced. The loss rate is dramatically impacted by the exposure time chosen. To illustrate this feature, the result for the case of 4 minute and 4 hour exposure times[1] is reported. The best scenario is a combination of NIDS and SCIT. For NIDS+SCIT (ET 4 minutes) the records lost are less than 0.16% of the NIDS only loss and 0.19% of NIDS+HIDS loss.

Table 3.1: Parameters used in the simulation

Simulation metrics	**Value (units)**
Number of queries used	50,000
Query Inter Arrival Time	10 ms to 18 ms
Intruder Residence Time (IRT)	0 minutes to 2 months
Mean IRT (modeled as Pareto distribution) against respective probabilities of occurrence.	48 (hrs)
Exposure time of SCIT (ET)	Case 1: 4 (hrs) Case 2: 4 (minutes)
Mean number of records stolen per day	675.4 records / breach
Mean number of records stolen per hour	28.15 records / breach

Table 3.2: Results of the Monte-Carlo simulation

Case	**Total Damage (records)**	**No. of Breaches**	**Mean Damage (records/breach)**
NIDS	245,962 (100%)	192	1,281
SCIT: ET 4h SCIT: ET 4m	55,364 (23%) 1,015 (0.4%)	508 508	109 2
NIDS+HIDS	210,578 (86%)	164	1,284
NIDS+SCIT: ET 4h NIDS+SCIT: ET 4m	20,931 (9%) 383 (0.16%)	191 191	110 2

[1] The prototypes that are have built have an Exposure Time (ET) of 1 minute, but in this analysis a higher ET is taken to show the effectiveness of SCIT architecture.

3.7 Summary

The SCIT architecture provides a robust security mechanism that guarantees certain security properties by limiting the exposure time. An important advantage of SCIT compared to IDS solutions is that SCIT does not generate false alarms, and thus can help reduce the intrusion alerts management costs. Thus SCIT also provides administrative and economic benefits which make it a reasonable choice to be included in security architecture. In particular, this is expected to be of interest in environments where technical skills are limited. Examples of such environments are found in military tactical settings, in remote and rural locations, small organizations and in newly emerging countries. The simulation studies presented suggest that a combination of an NIDS with SCIT on host servers provides a robust architectural solution in the face of new attacks.

CHAPTER FOUR – COMBINING INTRUSION DETECTION AND RECOVERY FOR ENHANCED SYSTEM DEPENDABILITY

This chapter presents a framework that uses Receiver Operating Characteristic (ROC) curve analysis and damage cost models to trade-off the true positive rate and false positive rate for comparing alternate detection based security architectures. In this work, the framework is employed in performing a comparison between IDS only solutions and an IDS + SCIT hybrid solution. This analysis provides a strategy for optimizing configuration of intrusion detection systems by evaluating the trade-off between potential damage from a missed intrusion and the costs of processing false positives.

4.1 Overview

Current cyber defenses are reactive and cannot protect against customized malware and other zero day attacks which persist for many weeks. Using Receiver Operating Characteristic curve analysis and damage cost models, the true positive rate and false positive rate are traded-off to compare alternative architectures. This analysis provides optimal value(s) of Probability of Detection by evaluating the potential damage from a missed intrusion and costs of processing false positives. In this chapter, an approach which involves determining the influencing factors of each strategy and studying the impact of their variations within the context of an integrated intrusion defense strategy is proposed. The goal is to manage the intrusion risks by proactively scheduling recovery for dependable networks.

The variety and complexity of cyber-attacks are increasing, along with the number of successful intrusions to mission and business systems. Recent breach reports like Wyndham Hotels [Wynd2010] reported system compromise detection in February 2010, whereas the malware had resided in the system since October 2009. So it is inferred that not only the Intrusion Detection System / Intrusion Prevention System (IDS/IPS) failed to prevent the intrusion, but current systems were not able to detect the presence of the intruder long after the compromise.

Motivated by the above observations, this research focus has been on a method which consists of two important approaches to enhance cyber defense. First, recognizing that intrusion detection is a hard problem, can the focus be shifted to minimizing losses resulting from intrusions? If this strategy is successful, it is anticipated that the reduced demands on the IDS will in turn lead to fewer false positives. Second, this model uses real world data from recent breach reports and their average costs to evaluate the cost reductions that can be achieved by using a combination of intrusion detection and tolerance architectures. Previously, the classical approach to assess architectures has been based on Single Loss Expectancy and Annual Loss Expectancy. More recently decision trees have been used [JGaf2001]. In the former, many assumptions are required, and in the latter a lot of data have to be collected. These approaches are good for analyzing systems for which past data can be used. But is this useful for architectural decisions for the future? The use of ROC (Receiver Operating Characteristic) curve based analysis is proposed, which is a powerful tool system administrator can use with enterprise specific data to build economic models and to compare alternate architectures. DARPA funded

Lincoln Lab IDS evaluation [RLip2000] was a pioneering paper that evaluated many IDS by generating normal traffic similar to that seen on Air force bases. They used ROC curves to present their results. McHugh [JMcH2000] published a critique of Lincoln Lab's work in 2000 which primarily considered issues associated with Lincoln's experimental dataset. McHugh pointed out the following problems in Lincoln's application of ROC analysis to IDS evaluation, which are a lack of "appropriate units of analysis, bias towards possibly unrealistic detection approaches and questionable presentation of false alarm data" [JMcH2000]. In Section 4.4, these issues are treated.

In this chapter, an IDS only solution is compared with IDS and SCIT (Self Cleansing Intrusion Tolerance) combination, SCIT being the approach to intrusion tolerance which is classified in the recovery-based category [QNgu2010]. From this assessment, optimal value(s) of Probability of Detection and other operational parameters can be selected to balance the potential damage from a missed intrusion and the cost of false positive processing. In this approach, it is stipulated that providing an upper bound on the time between the compromise and recovery has many advantages since it does not require the assumption that the system will be able to detect either the intrusion attempt or the compromise.

The rest of the chapter is organized as follows. In Section 4.2, the motivation for dependability recovery requirements is developed. Section 4.3 briefly reviews the intrusion tolerance approach. Sections 4.4, explains ROC Analysis usefulness to assess IDS architectures. . Sections 4.5, applies a cost model to evaluate how three different cases behave for a set of hypothetical ROC curves. Section 4.6 is the conclusion.

4.2 Motivation

As cyber defense efforts increase, passive efforts such as establishing anti-virus software, firewall protection, or improving password strength and encryption, and the organization's workload are constantly challenged by the need to apply patches immediately. Security researchers are uncovering close to 55,000 new malware samples a day, overwhelming malware analysis resources [McAf2010]. Increasingly, automated analysis technologies are used to keep up with the volume, but they still lack the precision to decipher compressed, encrypted, and obfuscated malware [RBej2005]. McAfee recent crash of tens of thousands of PCs globally illustrates the unpredictable system effects after compromise and their collateral damage, which creates even more uncertainty and less dependability for Enterprise Security [DKra2010].

The current reactive cyber defense approaches are expensive and inadequate. It is expected that, automated recovery and Intrusion Tolerance System (ITS) will be useful in addressing the increasing malware and patch workload, but what are the cost impacts of malicious threats and false positives on dependability and security attributes?

4.3 Intrusion Tolerance Approach

ITS architecture objective is to tolerate unwanted intrusions and restore the system to its normal state. Various ITS approaches are reviewed by Nguyen and Sood [QNgu2010]. In this paper, the recovery-based SCIT (Self-Cleansing Intrusion Tolerance) model is used [QNgu2010], which is applicable to servers that are open to the Internet, such as Web, and DNS servers [ABan2009]. Using round-robin cleansing, at any point in time, a server in a SCIT cluster can have one of the three states: offline

cleansing, offline spare and online transaction processing. The duration that a SCIT server is exposed to the Internet is called its Exposure Time. The architecture is simple, and does not rely on intrusion detection. Implementation of SCIT scheme can be based on virtualization. The interfaces between controller and the group of servers to be protected are trusted.

Another benefit of a recovery-based ITS is to shrink down breach duration, which has the effect of reducing losses and their costs. Indeed, this intrusion tolerance strategy would mitigate the effects of malicious attacks. Intrusion detection is known to be a hard problem, and current cyber defense systems reportedly detect less than half the malware. Still servers and apps account for 98% of the total record compromised. Verizon DBIR 2010 [Veri2010] underscores this problem by noting that only 11% of the compromises were detected within minutes or hours. Thus, current cyber defenses cannot protect systems against customized malware and other zero day attacks; once an attack is successful, it can persist for many weeks. This emphasizes the need for a recovery-based Intrusion Tolerance approach since detection triggered ITS might again fall short of the needs.

4.4 Receiver Operating Characteristics (ROC)

ROC analysis has been long used in signal detection theory to present the tradeoff between hit-rates and false-positive rates of classifiers. ROC analysis was initially used during World War II in the analysis of radar signals to differentiate signal from noise. It was soon introduced in Psychology to map the perceptual detection of signals [JSwe1996]. ROC curves are useful for assessing the accuracy of predictions. A ROC

curve plots the fraction of true positives (hits) versus the fraction of false positives, and hence has a direct relationship with diagnostic decision making. The ideal prediction method would yield a co-ordinate (0, 1) on the ROC curve. This represents 100 % true positives and zero percent false-positives, and is referred to as the perfect classification.

4.4.1 Using ROC to assess IDS quality

The most attractive feature of ROC analysis is the fact that the tradeoff between probability of detection and probability of false positive can be derived directly. This allows a system administrator to instantly determine how well a classifier performs and also to compare two classifiers. We care about false positives in addition to the probability of detection since there is a need to characterize human workload involved in analyzing false positives generated by traffic. According to [RLip2000], false positive rates above 100's per day could make IDS almost useless even with high probability of detection since security analysts must spend hours each day investigating false positives.

DARPA funded Lincoln Lab IDS evaluation [RLip2000] appears to be the first to perform tests to evaluate many IDS by generating normal traffic similar to that on a government site. McHugh [JMcH2000] reviews and analyzes the validity and adequacy of artificial data used to estimate real world system performance. In this chapter, a methodology to compare various IDS's, each of which is represented by a ROC curve is presented. Verizon's 2010 results representing a cross section of multiple industries are utilized. Furthermore, these data validate firsthand real world evidence over a broad five year range from 2004-2009 with the addition of US Secret Service confirmed cases.

The Lincoln Lab experiment used ROC for presenting the results of the evaluation. McHugh [JMcH2000] criticized Lincoln Lab's use of ROC curves primarily on the following grounds. It is attempted to address each of these concerns in this work:

Determining appropriate units of analysis: Unit of analysis is the quantity of input on which a decision is made. Lincoln lab used sessions as the unit of analysis, the problems of which were outlined in [JMcH2000]. McHugh also emphasized the need for using similar units of analysis across all IDS's to be evaluated. In this case, a simple system and consistently use query / packet is considered as the unit of analysis across all IDS's.

Errors per unit time: In [RLip2000], a pseudo-ROC curve with x-axis as False Positives per day instead of Percentage False Positives was used. This led to two incomparable units being used on two axes, and the results in turn became strongly influenced by factors like the data rate that should typically be irrelevant. In this chapter, the probability of detection and that of false positives for all ROC curves are consistently used. In such a case, given that the distributions of signal and noise are realistic, McHugh [JMcH2000] recognizes that the ROC presentation should give a good account of detector performance in similar environments. Given enough characterizations of the signal and noise distributions, McHugh further acknowledges that it is even possible to investigate optimal detectors.

McHugh [JMcH2000] criticizes Lincoln Lab's methods of scoring and constructing ROC curves which lead to problems like bias towards unrealistic detection approaches, but not the use of ROC curves itself. In this case, the emphasis is not on

constructing ROC curves but on comparing IDS's using the cost-model once they have their respective ROC curves. While there is a need for alternative taxonomies, the scoring method from the attacker's perspective is still utilized for real world incidents.

According to [RLip2000], there have been a number of similar efforts. In order to be able to compare multiple IDS systems, the ROC curves should be generated using similar or preferably same test data. According to Orfila et al. [AOrf2006], if two ROC curves intersect at some point, there is no way of claiming that one is better than the other since some system administrators might want high probability of detection (top right corner of ROC curve) and some might want low probability of false positive (bottom left corner of ROC curve).

Stolfo et al. [FSto2000] presents an alternative method to perform evaluation based on cost metrics. Authors help formalize the costs involved in evaluating an IDS into three types: 1) Damage cost, 2) Challenge cost or Response cost and 3) Operational cost.

In [CDru2004], Drummond et al. propose the use of cost curves for evaluating classifiers. Cost curves plot expected cost vs. Probability Cost Function (PCF). Here PCF is a function of probability of detection, probability of false positive and its corresponding costs. Although cost curves are good to compare classifiers, the representation does not provide for the system administrator to quickly see the cost trend of operating at different points (P_f, P_d) on the ROC curve. Also [CDru2004] does not suggest a way to determine the expected cost of operating at a point on ROC curve.

In [JGaf2001], Gaffney et al. argued that both ROC analysis and cost analysis methods are incomplete. They used decision analysis techniques and provide an expected cost metric that reflects IDS's ROC curve based on a decision tree approach. This cost model requires a lot of data to be collected and does not reflect the magnitude of actual costs associated with breach events. For this, a cost-model for the calculation of expected cost of operating at any point on the ROC curve is proposed.

4.5 Cost Model

In this section, it is aimed to overcome each of the shortcomings of earlier approaches by proposing a cost model that consists of two elements:

- A formula for the expected cost of operating at any point on the ROC curve
- Cost metrics derived from published breach investigation reports

4.5.1 Expected Cost Calculation

The cost of operating IDS at any point on the ROC curve (P_f, P_d) is a combination of the following:

- Operational Costs – Cost involved in operating the IDS and keeping it running.
- Damage Costs – the amount of damage caused by an intruder in case of a successful attack.
- Response Costs – the cost involved in responding to a potential intrusion on detection.

Out of the three costs mentioned above, operational costs and response costs greatly vary from organization to organization based on a number of factors like size of the organization, type of organization etc. Since these two costs are not entirely

quantifiable, for the purposes of this chapter, the objective function proposed in [JHan1966] is employed:

Expected Cost of operating at any point on the ROC curve = Cost of Misses + Cost of False Positives

Thus, for every point on the ROC curve (P_f, P_d), there is an expected cost:

Expected Cost = (C_m*p*P_m) + (C_f*(1-p)*P_f),

Where

C_m – Cost of a miss

C_f – Cost of a false positive

P_m – Probability of a miss = (1-Pd)

P_f – Probability of a false positive

p – Prior probability of Intrusion

P_d – Probability of detection

Note that this expected cost is for one incoming query. If there are 'n' incoming queries, the above expected cost must be multiplied by 'n'. The value of metrics used in the cost model is summarized in Table 4.1.

Table 4.1: Metrics values used in the Cost Model

Metrics	Value	Explanation	Ref
Median number of records lost per breach (M)	1,082	Removes outliers. Better estimate of the "typical value"	[Veri2010]
Average cost of compromised record (D)	$ 204	Direct Cost: $ 60 + Indirect Cost: $144	[SWid2010]
Cost of a Miss (C_m)	$220,000	M * D = 1082 * $ 204	[Veri2010], [SWid2010]
Cost of a False Positive (C_f)	$ 400	Assumption: Labor Cost + Overhead Cost = $ 400	
Median Compromise Duration per breach	14 days	Compromise to	[Veri2010]

		Discovery time + Discovery to Containment time	

In this chapter, the probability of detection P_d and that of a false positive P_f will constitute the operational parameters.

The median number of records lost for assessing damage is used. In many cases, the outliers in breach data can skew the data, because most of the losses come from only a few breaches. Therefore, the Mean becomes highly skewed and is not a good estimate of the typical number of records lost per breach. Median is a better estimate of the typical value [SWid2010].

4.5.2 Evaluating Classifiers using the proposed Cost Model

For the purposes of this chapter, it is not addressed how the ROC curves are constructed. Proper construction and use of ROC curves in Intrusion / Anomaly detection have been addressed in [RMax2004]. It is just shown how the cost model can be implemented once they are constructed. Figure 4.1 gives a family of hypothetical ROC curves, each representing a classifier. The cost model will be implemented on these ROC curves in three different cases to evaluate the classifiers' behaviors:

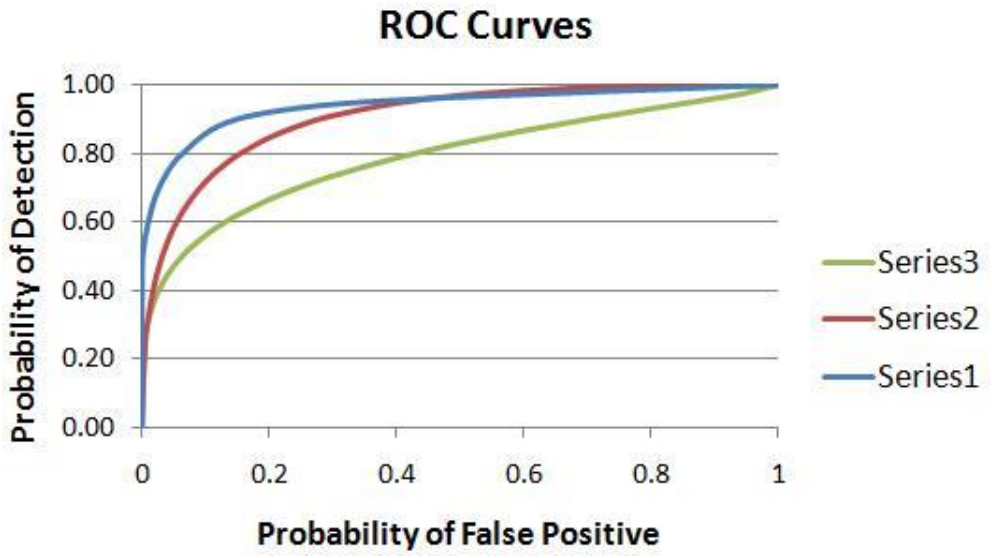


Figure 4.1: Receiver Operating Curves

Table 4.2 provides the values of the parameters used in the cost model in each of the three cases. Within each case, the value of 'p' remains the same for both IDS and SCIT+IDS. Therefore, the number of intrusions that occur in each of these architectures are the same since Number of intrusions = [Number of incoming queries * Prior probability of intrusion (p)]. The baseline IDS and SCIT+IDS scenarios are provided for Case 1. Case 2 and Case 3 help investigate the impact of 'C_m' and 'p' on system cost and security. Figures 4.2 through 4.7 illustrate this. It is noted that the y-axis scale is different in Figure 4.6.

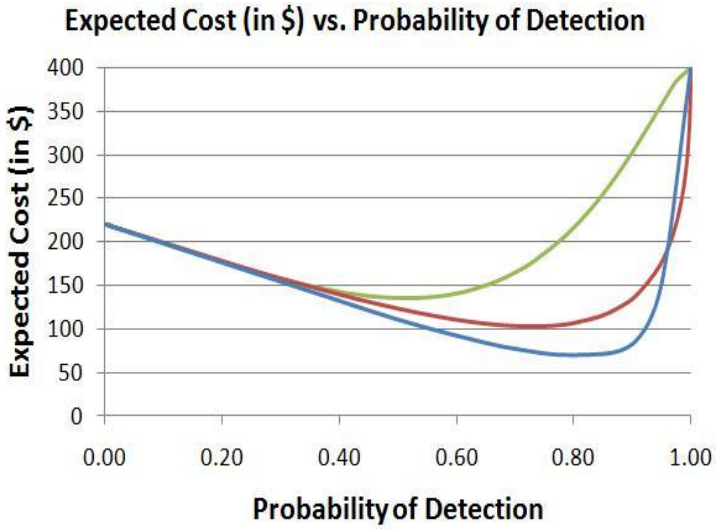
CASE 1a. IDS: (Figure 4.2)

This is a stand-alone IDS system. The cost keeps decreasing as Probability of Detection (P_d) is increasing. As P_d increases, number of misses decrease along with the significant associated costs. However, after a threshold, if the value of P_d is increased, the

expected cost stops decreasing and starts increasing rapidly. At this point, the cost of False Positives exceeds the cost of misses and so the gains from containing misses start diminishing. This point is known as the "minimal cost point on the ROC curve (MCP)". For e.g., in Case 1a, the MCP for Series 1 is 70 and it occurs at (P_f, P_d) = (0.20, 0.85). MCP for each series of every case evaluated is tabulated in Table 4.3.

CASE 1b. SCIT + IDS: (Figure 4.3)

Now SCIT is added to existing IDS and the system is evaluated using the Cost Model. It is assumed that the exposure time of SCIT is 4 hours[2]. This reduces the compromise duration of the system from 14 days to 4 hours. It is assumed that data is ex-filtrated uniformly over time. Since the cost of a miss was $220,000 earlier with compromise duration of 14 days, now it significantly reduces to $2,620 for compromise duration of 4 hours.



[2] The SCIT servers tested in our lab and independently tested at Lockheed Martin and Northrop Grumman have Exposure Times of 1 or 2 minutes. Here, larger values of Exposure Time are used to emphasize the advantage of the concept.

Figure 4.2: IDS Case 1a

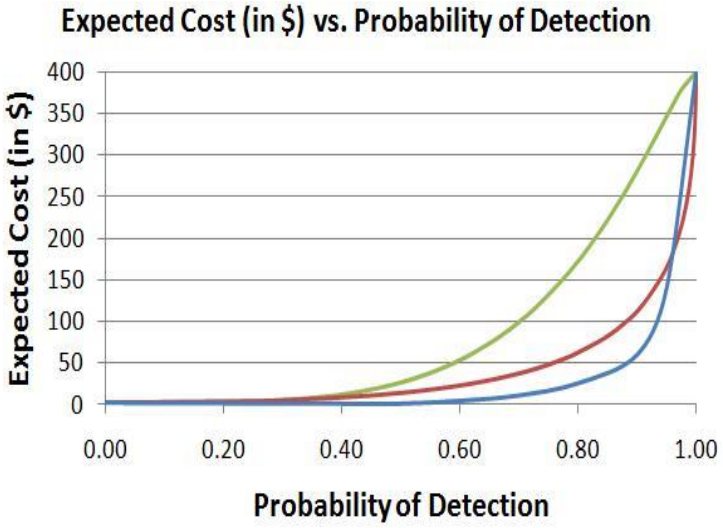


Figure 4.3: SCIT + IDS Case 1b

Table 4.2: Parameter values used in the cost model

	p	**C_m**	**C_f**	**Compromise Duration**
Case 1a: IDS	0.001	$220,000	$400	14 days
Case 1b: IDS+SCIT	0.001	$2,620	$400	4 hours
Case 2a: IDS	0.001	$60,000	$400	14 days
Case 2b: IDS+SCIT	0.001	$715	$400	4 hours
Case 3a: IDS	0.005	$220,000	$400	14 days
Case 3b: IDS+SCIT	0.005	$2620	$400	4 hours

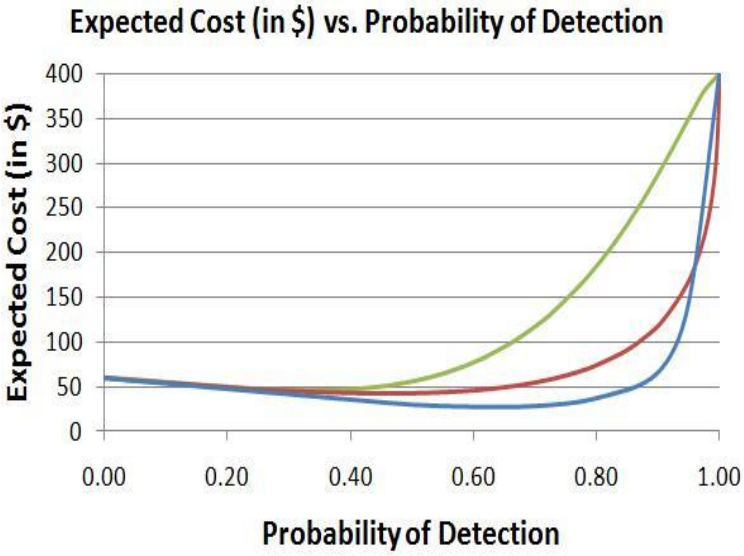


Figure 4.4: IDS Case 2a

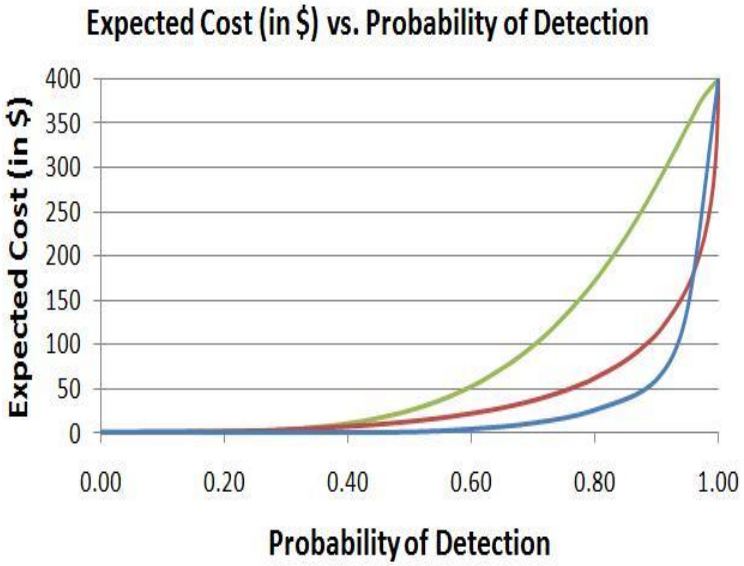


Figure 4.5: SCIT + IDS Case 2b

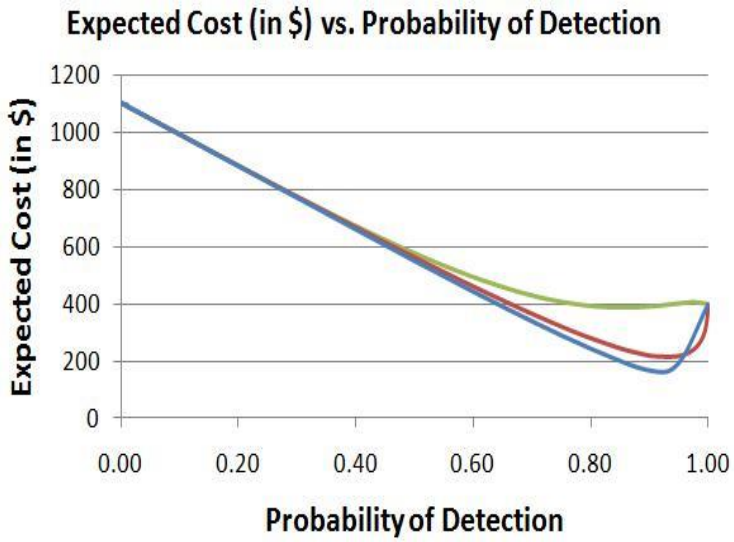


Figure 4.6: IDS Case 3a

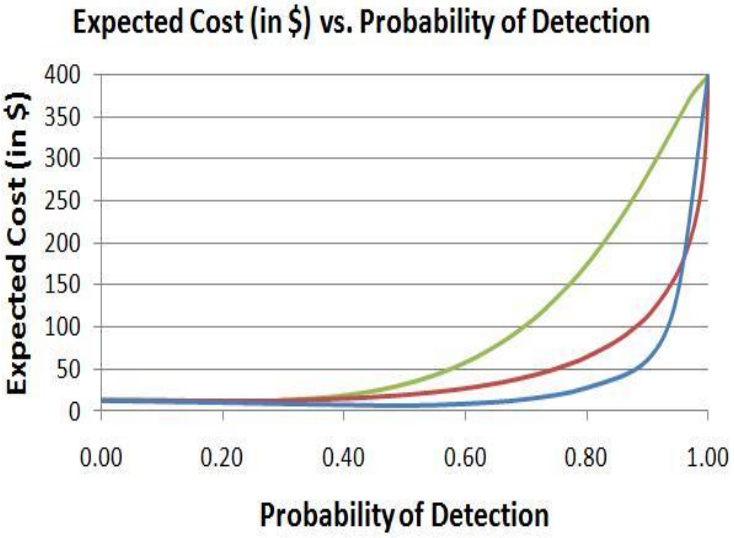


Figure 4.7: SCIT + IDS Case 3b

CASE 2. (Figures 4.4 & 4.5)

Assumption: As compared to the baseline (Case 1), IDS cost of a miss is reduced from $220,000 to $60,000.

CASE 3. (Figures 4.6 & 4.7)

Prior Probability of Intrusion is increased fivefold from p = 0.001 to p = 0.005.

4.5.3 Results: Comparison of IDS's

Figure 4.8 compares the MCP's of 3 IDS' whose performances are indicated by the ROC curves in Figure 4.1.

- Series 1 IDS clearly outperforms all the other IDS' in all three cases.
- It is most expensive to operate the IDS' in case 3 since prior probability of intrusion is high which in turn leads to more misses.

4.5.4 Results: Comparison of SCIT + IDS's

Figure 4.8 also presents the minimal cost points for IDS + SCIT. An exposure time of 4 hours is used. It is noted that as compared to the IDS only case, the costs are much lower. The minimal cost points are achieved using a much lower value of Probability of Detection which in turn leads to a lower Probability of False Positive. It can be concluded that this makes the IDS design much easier and the system easier to operate. The reliability of the IDS results also increase.

From the results, it can be seen that the benefits of adding SCIT are as follows:

- Cost of a miss is greatly reduced. As the compromise duration / exposure time of SCIT is reduced, cost of a miss further reduces.
- A larger number of misses can be tolerated now that the cost of a miss is reduced.

Table 4.3: Minimal Cost Point values

CASES	Minimal Cost Point for Figure 1 ROC Curves - Cost ($)					
	SERIES 1		SERIES 2		SERIES 3	
	IDS Only	IDS + SCIT (ET=4hrs)	IDS only	IDS + SCIT (ET=4hrs)	IDS Only	IDS + SCIT (ET=4hrs)
CASE 1	70	2	102	3	135	3
CASE 2	28	0.5	43	1	45	1
CASE 3	170	7	218	12	386	12

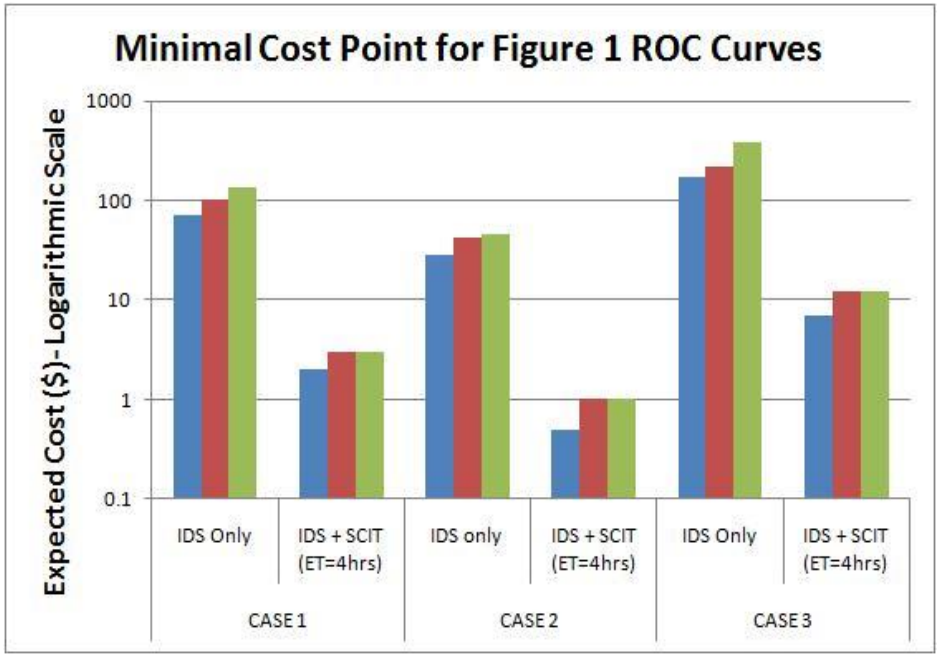


Figure 4.8: Minimal Cost Point Comparison

4.5.5 General Observations (IDS and SCIT + IDS)

As the cost of miss decreases, more misses can be tolerated and so probability of detection for achieving minimal cost point can now take lower values.

As C_m decreases, C_f has a greater influence on the expected cost and so there is an increased need to contain false positives. Note that the Probability of False Positives for achieving minimal cost point now decreases.

As prior probability of intrusion 'p' increases:

- The total number of misses' increases and so does the expected cost.
- To combat this, probability of Detection for achieving minimal cost point increases thus reducing the number of misses. (Note: Number of misses = Number of incoming queries * p * P_m).

4.6 Summary

Intrusion detection is a hard problem, making intrusions inevitable. Consequently, containing losses by an upper bound on the time between compromise and recovery shows many advantages. ROC analysis, supplemented with cost analysis using median of lost records and average cost of compromised records per breach, reveals tradeoff between high probability of detection, and low probability of false positive. This approach reduces the cost of a miss; and tolerating a larger number of misses' leads to lower false positive costs.

The SCIT architecture provides a robust security mechanism that guarantees certain security properties by limiting the exposure time. In addition, SCIT does not generate false positives and thus reduces the intrusion alerts management costs. Thus SCIT also provides administrative and economic benefits which make it a reasonable choice to be included in security architecture. In particular, this is expected to be of interest in environments where technical skills are limited. The analysis presented

suggests that a combination of IDS with SCIT on host servers provides a robust architectural solution in the face of new attacks.

CHAPTER FIVE – SCIT BASED MOVING TARGET DEFENSE REDUCES AND SHIFTS ATTACK SURFACE

This chapter leverages the concept of Attack Surface [15, 16] and its use as a security metric to compare the relative security of multiple hybrid security architectures. This work proposes the use of Attack Surface Shifting / Reduction as a metric to compare Moving Target Defenses (MTD) by assessing its impact on intruder / defender work factors.

5.1 Overview

In this chapter, Attack Surface assessment is used to evaluate SCIT. A system's attack surface is the subset of its resources that an attacker can use to attack the system. Manadhata [PMan2008, PMan2013] uses attack surface reduction / shifting as means of assessing MTD. In this chapter, the dynamically changing Attack Surface for three system architectures (1) Static Systems; (2) Basic-SCIT and (3) Diverse-SCIT are compared.

Moving Target Defense (MTD) is the idea of managing change across various system and network dimensions in order to increase the intruder work factor by increasing the intruder work complexity and decreasing visibility of systems to the intruders. Traditionally MTD strategies have presented two significant challenges to adoption. First, for the sake of security, MTD cannot ignore performance and end user productivity. Most customer facing systems don't have the luxury of adding security that

slows down performance. Customers tend to move on if the experience is slow and tedious. Secondly, traditional MTD design generally consists of complex processes involving memory address randomization, network address shuffling, instruction set randomization and more [DEva2011]. All of these techniques are designed to prevent attacks and have the potential to be resource hogs thereby slowing down throughput in certain cases.

SCIT based Moving Target Defense acknowledges that trying to prevent each intrusion is impractical. Therefore, the emphasis is to minimize losses occurring from intrusions rather than prevent intrusions. SCIT systems are designed to be complementary to reactive systems [ANag2010]. Primary goal of SCIT-MTD is to reduce the intruder's window of opportunity to execute an attack and increase the costs of their foot-printing, scanning and attacking efforts. Since by design, the SCIT-MTD attack surface of the system is constantly changing, the system vulnerabilities are difficult to exploit. The process of compromising a system involves identifying system vulnerabilities and customizing attacks to exploit them. Ever-changing attack surface presents a stiff challenge to the intruders. SCIT – MTD can be used with diversification approaches to further increase the attacker difficulty.

5.1.1 Common Security Evaluation Metrics and Attack Surface

Measurement of security has been a challenge and is of practical importance to software industry. Today two measurements are commonly used to determine the security of a system: (1) at the 'code level', the number of bugs found (or fixed from one version to the next) are counted; (2) at the 'system level', the number of times a system version is

mentioned in CERT advisories, security bulletins and vulnerability databases like MITRE CVE are counted. Manadhata [PMan2008, PMan2013] proposed Attack Surface as a security metric that focuses at the 'design level' of a system: above the level of code, but below the level of the entire system. Attack Surface is a metric to compare the relative security of two versions of the same system rather than the absolute security of a system. Given two versions, A and B, of a system, one could measure the security of A relative to B with respect to the system's attack surface. Intuitively, higher the attack surface, more the chances of the system getting compromised e.g., eliminating certain system features potentially makes it more secure.

Attack Surface assesses (a) system 'actions' externally visible to the system's users; and (b) system 'resources' accessed or modified by each action. The more actions available to a user or the more resources accessible through these actions, the more exposed the attack surface. The more exposed the attack surface, the more likely the system could be compromised.

The Formal Definition of Attack Surface is [PMan2008] "The set M of entry points and exit points, the set C of channels and the set I of un-trusted data items are the system's resources that can be used by the attacker to compromise the system. Therefore, given a system S and its environment, the system's attack surface can be represented as the triple <M, C, I>".

Attacks carried out over the years, however, show that certain system resources are more likely to be opportunities, i.e., targets or enablers, of attack than others. This leads to the idea of 'Weighted Attack Surface'. For example, services running as the

privileged user root in UNIX are more likely to be targets of attack than services running as non-root users. Since every system resource contributes unequally to the system's attack surface, author of [PMan2008] proposes the use of 'Damage Potential – Effort ratio'. The amount of damage that can be done to the system by exploiting a particular resource is the damage potential of that resource. Similarly, the amount of work that the attacker would have to put in to use that resource as an attack tool defines the effort.

5.2 Attack Surface Shifting/Reduction as a technique for Moving Target Defense

In [PMan2008] Manadhata formalized the notion of a software system's attack surface and proposed the use of system's attack surface measurement as an indicator of the system's security. Intuitively, a system's attack surface is the set of ways in which an adversary can enter the system and potentially cause damage. Hence larger the attack surface, the more insecure the system.

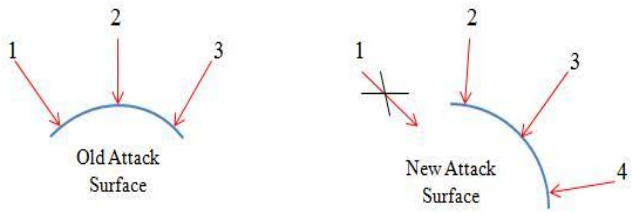


Figure 5.1: Attack Surface Shifting

Awad A. Younis et al [AYou2012] establish that there is a relationship between Attack Surface Size and Vulnerability Density. The authors also performed a case study on two different versions of Apache HTTP Server. They concluded that the version with

the bigger attack surface had more vulnerabilities as well as vulnerability density. This further supports the use of Attack Surface size as a security metric.

In [PMan2013], the author considers a scenario where system administrators are continuously trying to protect their systems from attackers. As shown in Figure 5.1, if a defender shifts a system's attack surface, then old attacks that worked in the past, e.g., attack 1, may not work anymore. Hence the attacker has to spend additional effort to make past attacks work or find new attacks, e.g., attack 4. Hence, the interaction between the defender and the intruder here can be viewed as a two player game where the action of one player has a consequence on the other. Thus, reducing or shifting a system's attack surface functions as MTD. This works in favor of the defender to increase the intruder's work factor randomly.

Attack Surface of a system can be reduced or shifted by disabling, modifying and / or enabling the system's features [PMan2013]. Disabling the existing features reduces the number of entry points, exit points, channels, and data items, and hence reduces the number of resources that are part of the attack surface. Modifying the features changes the damage potential-effort ratios of the resources that are part of the attack surface, e.g., lowering a method's privilege or increasing the method's access rights reduces the resources' contributions to the attack surface measurement. The enabled features increase the attack surface measurement by enabling new features and adding more resources to the attack surface. When existing features are disabled and new features enabled, the attack surface shifts. Table 5.1 presents four illustrative scenarios to

highlight the possible impacts of disabling, enabling or modifying features on a system's attack surface:

Table 5.1: Possible Scenarios to Reduce and Shift the Attack Surface

Scenarios	Features	Attack Surface Reduction	Attack Surface Shift
A	Disabled Existing	Yes	Yes
B	Enabled New	No	No
C	Enabled New Disabled Existing	Yes	Yes
D	Enabled New Disabled Existing	No	Yes

5.2.1 Dynamic Attack Surface

The Attack Surface of a production system increases with time. For example, the number of open sockets may increase because of programming oversight. In typical operations, the application of a security patch reduces the Attack Surface, while a patch that increases functionality increases the Attack Surface. Similarly, in cases of a Web Server serving dynamic content; the contents of dynamic web pages change in response to different client requests. Use of additional web service extensions and client side plug-ins are generally required to facilitate the use of dynamic content thereby leading to an increase in the Attack Surface. Thus, the Attack Surface is a dynamic property. The SCIT approach constantly restores software to a pristine state, and thus dynamically reduces the Attack Surface.

5.2.2 Impact of Dynamic Attack Surface on Intruder Work Factor

Information assurance mechanisms are designed to frustrate the adversary and make it difficult to launch a successful attack. There is a need to quantify the impact of a given mechanism on a particular adversary. In the age of unknown attacks, the goal of sound security architecture should be: (a) to significantly increase the intruder work factor for successful attacks. Intruder work factor is the amount of work an intruder has to put in to accomplish an attack (eg: mean time to compromise a system) and (b) To significantly increase the ratio of the attacker's work factor to generate successful attacks to the defender's work factor for responding to successful attacks [JJust2003].

In order to measure the impact of Dynamic Attack Surface on Intruder Work Factor (IWF), a test bed experiment [ANag2013] was developed. This effort was not meant to be exhaustive but representative. As part of the experiment, vulnerable versions of Apache Tomcat and Samba were exploited using pre-loaded exploits in the Metasploit Framework. Since pre-loaded exploits were used, the experimental results did not account for 'Exploit Development Time' which is often a large chunk of the time for compromising a system. Address Space Layout Randomization (ASLR) and SCIT techniques of Moving Target Defense were implemented to make the Attack Surface more dynamic. Two sample exploit conditions were used to assess the impact of Dynamic Attack Surface on IWF: (a) Remote root buffer overflow exploit of Samba and (b) WAR backdoor exploit of Apache Tomcat.

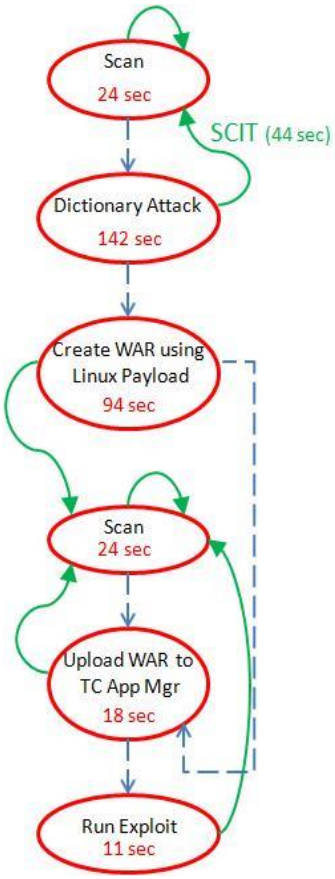


Figure 5.2: Attacker and Defender Actions – Apache Tomcat Exploit

Sample Results [ANag2013]: Figure 5.2 presents the attacker and defender actions for Case (b): the Apache Tomcat Exploit. Red circular nodes here indicate intruder activity with associated IWF in seconds, and blue dashed arrows show the transitions between intruder actions. Green solid arrows indicate impact of SCIT rotation on each intruder activity. Defender Work Factor (DWF) in this case is 44 seconds per rotation – the time taken to switch an exposed virtual instance of server with a pristine copy. This process not only moves the target but also self-cleanses the system. SCIT rotation and self-cleansing is independent of attacker activity and so it can occur at any

point of the attack life cycle. Example scenario: if SCIT rotation happens while the intruder is performing his 'Dictionary Attack' – then the attack is interrupted and the intruder would have to redo the following: (a) Scan to identify the new target and (b) Re-launch the dictionary attack on the newly identified target. In effect, a DWF of 44 seconds induces an additional IWF of 166 seconds.

Under these sample exploit conditions, the key findings of the experiment were: (a) ASLR increased the IWF and at best case, had an iterative impact on IWF. ASLR with periodic re-randomization induced a higher IWF than ASLR w/o re-randomization. (b) SCIT, at best, had a cumulative impact on IWF. These further support the notion that dynamic attack surface frustrates the adversary and increases the work required for a successful attack.

5.3 Test Bed Experiment

In this chapter, Attack Surface assessment is used to evaluate impact of the MTD solutions. The dynamically changing Attack Surface for three security configurations (1) Static systems; (2) Basic-SCIT and (3) Diverse-SCIT are compared. Static systems adopt traditional reactive systems; they are sitting ducks that are indefinitely online. System cleansing and recovery is generally manually triggered or by an IDS/IPS. Static systems with two flavors of SCIT – MTD are compared: Basic-SCIT which loads the same pristine image every-time a virtual server is self-cleansed; and Diverse-SCIT loads clean images of diverse implementations of the same service during the SCIT cycle. Microsoft Attack Surface Analyzer is used for ease of use to perform all attack surface assessment.

Configuration of System used for Test Bed Experiment:

- Gateway P-7805u
- Intel Core 2 Duo CPU P8400 @ 2.26GHz
- 4 GB RAM
- 64-bit Windows Vista Home Premium Service Pack 2

This setup is not intended to emulate real world server system configuration. This is merely a test-bed configuration used to evaluate the dynamic attack surface of varying system architectures in sample scenarios.

Assumptions made for analysis:

Since it is not plausible to determine the 'Damage Potential - Effort Ratio' of every existing system resource (there are hundreds of them); it is assumed they are all equal. This is similar to the approach taken in [PMan2008]. And so, it is arrived that

Attack Surface Size = Total Number of Attack Surface Components

5.3.1 Attack Surface Components

For the purposes of the Test Bed Experiment, the Microsoft Attack Surface Analyzer is used. In the experiment it is assumed that the following components make up the Attack Surface of a system. This is not intended to be comprehensive but summarizes the key components of any system's attack surface: (a) Running Processes – Process is an executing program; (b) Executable Memory Pages – Data Execution Prevention (DEP) is a system-level memory protection feature which enables the system to make one or more memory pages non-executable. Non-executable memory pages make it harder for the exploitation of buffer overruns. Therefore, fewer executable memory pages is better; (c) Windows – In a graphical Windows-based application, window is the area of the screen

which interacts with the user by receiving input and displaying output; (d) Kernel Objects – An object is a collection of data that the OS manages. Kernel Objects are objects that are part of the kernel-mode operating system, for example: symbolic links, registry keys; (e) Services- Windows service is a program running in the background similar to a UNIX Daemon; (f) Drivers – Software that enables the functionality of a physical or virtual device; (g) Ports – Ports are process specific communication endpoints of a system. Ports are associated with host IP addresses and the type of protocol used for communication as in TCP or UDP; (h) Named Pipes – A named pipe is a one-way or duplex pipe for communication between a pipe server and one or more pipe clients. A named pipe can facilitate inter process communication; (i) RPC Endpoints – Remote Procedure Call is an inter-process communication that allows a program to execute a procedure on a remote computer over the network. RPC Endpoints facilitate such communication; (j) Objects with weak Access Control List (ACL): These can be files, executables, registry entries etc. One example of a weak ACL is allowing non-administrators to modify files. Sum of all of these components make up the Attack Surface Size of a system.

Table 5.2: Attack Surface Size Comparison

Attack Surface Component	Pristine Apache System	Apache System after 32 days	Apache System after 4 hour Exposure	Pristine Nginx System	Nginx System after 4 hour Exposure
Running Processes	76	79	77	76	77
Executable Memory Pages	25	46	27	21	26
Windows	183	265	199	180	189
Kernel Objects	513	520	513	513	516
Services	182	189	182	181	181

Drivers	274	284	274	274	274
TCP/UDP Ports	118	138	124	101	115
Named Pipes	133	146	136	133	134
RPC Endpoints	33	35	33	33	33
Attack Surface Size	**1537**	**1699**	**1565**	**1512**	**1545**

5.3.2 Static Systems

Static systems are systems that do not incorporate proactive security strategies. In such systems, cleansing is generally triggered manually or by an alarm on discovering malicious activity. According to Verizon's Data Breach Investigation Report 2013 [Veri2013], the average time an intruder resides on the system from the point of initial compromise to the point of intrusion discovery is more than 34 days. In the case of a Static System, since there is no periodic self-cleansing or restoration, the attack surface of the system keeps on increasing with time as a result of normal system use.

To illustrate this, the attack surface size of an Apache System (a) before use (pristine) and (b) after random usage for 32 days are compared. Figure 6.4 illustrates the setup of the Apache System. Table 5.2 presents results from the attack surface analysis report. Columns 2 and 3 of Table 5.2 show the growth in Attack Surface Size of the static Apache system during usage.

Table 5.3 section (a) summarizes the security issues that were introduced during the 32 day usage. In other words, these security issues were not present in the Pristine Apache System but appeared in the Apache System after 32 days.

5.3.3 Basic-SCIT Setup

In the Basic-SCIT setup, there are multiple virtual instances of the server of which one or more are online at any given time. Once every period of time known as 'Exposure Time', the system proactively self-cleanses and rotates. Every time this happens, the Pristine Apache Image is loaded into the virtual instance that is about to go online. Figure 5.3 presents one such instance. This setup is providing a Pet Store e-commerce application service through iBatis JPetStore 4.0.5. In the experiment, the 'Exposure Time' is 4 hours. The attack surface size of the system can only increase till the point of self-cleansing. Thus there is an upper bound on the growth of the Attack Surface Size. After 4 hours, on self-cleansing, the size of the attack surface is reduced back to that of the Pristine Apache Image. This is cyclical and so the size of the attack surface is periodically reduced and kept manageable.

Table 5.2 (columns 2 and 4) compares the Attack Surface Size of a Pristine Apache System with that of the Apache System after 4 hour use. After the system has been exposed for 4 hours, it is self-cleansed and rotated thereby reducing the Attack Surface Size back to that of the Pristine Apache System. From columns 2, 3 and 4, it can be emphasized that the growth in Attack Surface Size is much less in 4 hours as opposed to 32 days. Similarly, from Table 5.3 section (a) and section (b) it is apparent that the count of security issues that arose over 32 days far outnumbered issues that arose in 4 hours.

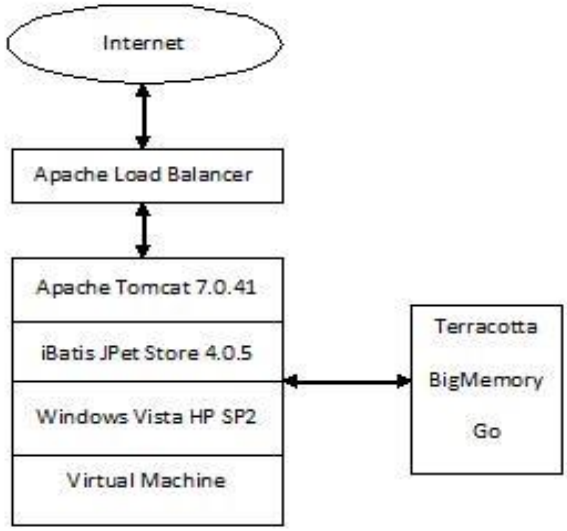


Figure 5.3: Apache System

Figure 5.4 (black series) presents the temporal attack surface of the Basic SCIT setup for the first 40 hours. The size of the system's attack surface increases for 4 hours when it is exposed and is reduced to that of the pristine Apache image on self-cleansing periodically.

Table 5.3: Security Issues that arose in (a) the Apache System during 32 day usage; (b) the Apache System during 4 hour usage and (c) the Nginx System during 4 hour usage

Security Issues on System Usage	Count
(a) Pristine Apache System VS Apache System after 32 days	
Executables with weak ACLs	17
Directories containing objects with weak ACLs	10
Registry Keys with weak ACLs	10
Processes with NX disabled	1
Services vulnerable to tampering	3
Services with Fast Restarts	1
Vulnerable Named Pipes	26
(b) Pristine Apache System VS Apache System after 4 hours	
Directories containing objects with weak ACLs	3
Processes with NX disabled	2
Services vulnerable to tampering	1

(c) Pristine Nginx System VS Nginx System after 4 hours	
Directories containing objects with weak ACLs	4
Services vulnerable to tampering	1

5.3.4 Diverse-SCIT Setup

In Diverse-SCIT, a system with two diverse implementations of the iBatis JPetStore service is used. Virtual Server 1 uses the Apache Load Balancer with Apache Tomcat 7.0.41 and Terracotta Big Memory Go Caching; whereas Virtual Server 2 uses the Nginx HTTP Server with load balancer along with MemCached v1.4.15 to provide service.

Table 5.4: Security Issues unique to each configuration

Security Issues	Count
Issues present in Apache System but not in Nginx System	
Directories containing objects with weak ACLs	4
Processes with NX disabled	1
Services vulnerable to tampering	1
Issues present in Nginx System but not in Apache System	
Directories containing objects with weak ACLs	1
Processes with NX disabled	1

Figure 5.5 presents two such virtual instances of the server, one with each configuration. In this setup, virtual servers are rotated in such a manner to alternate between the two configurations. As shown in Figure 5.4 (red series), on each self-cleansing, the size of the system's attack surface alternates between that of Pristine Apache image and Pristine Nginx image. Table 5.2 (columns 5 and 6) compares the

Attack Surface Size of the Pristine Nginx System with the Nginx System after 4 hour use. Table 5.3 section (c) lists the security issues that arose during exposure. In addition to reducing the Attack Surface Size periodically, this setup also shifts it. Table 5.4 emphasizes this shift by presenting security issues that are unique to each setup. This adds another layer of complexity to the intruder since identifying system vulnerabilities with ever changing attack surface is a challenge. An attack that used to work with the previous configuration no longer works on rotation.

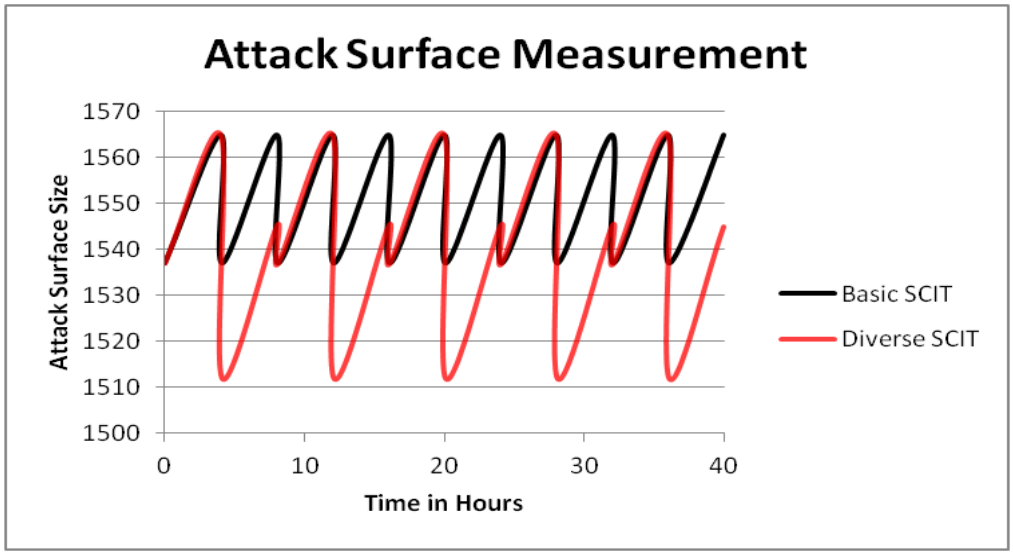


Figure 5.4: Temporal Attack Surface – Basic SCIT and Diverse SCIT

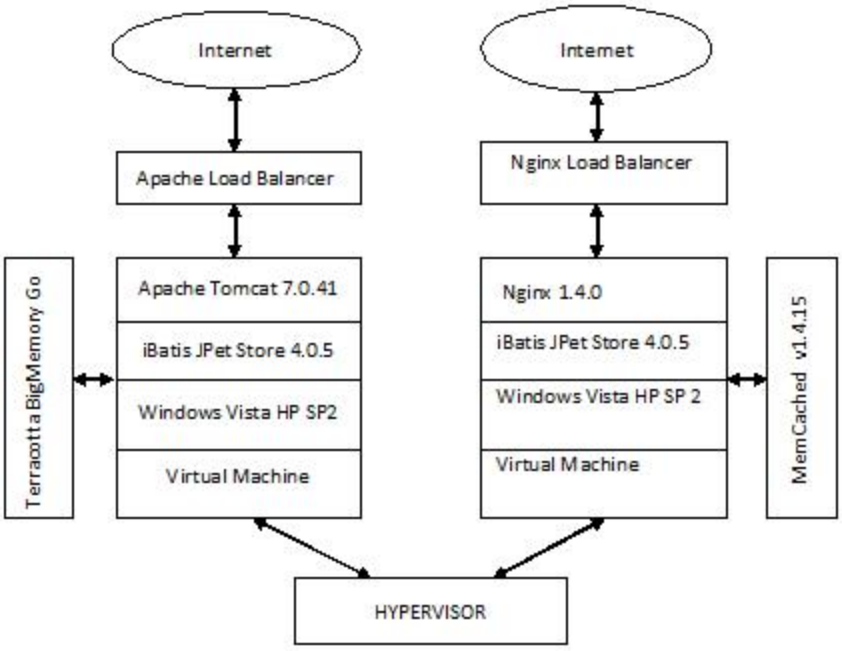


Figure 5.5: Two virtual instances of the Diverse SCIT Setup

5.4 Summary

In this chapter, Attack Surface assessment is used to evaluate the impact of the SCIT MTD solutions. The dynamically changing Attack Surface Size for (1) Static Systems; (2) Basic-SCIT and (3) Diverse-SCIT were compared using a test bed experiment. Results of the experiments that show changes in attack surface size along a timeline for the three different security configurations are presented. The results support the hypothesis that SCIT is an effective means to provide MTD by reducing / shifting attack surface periodically, thus making the hacker's task harder. With Basic-SCIT, by moving the target virtual server, the intruder is forced to restart the attack all over again. Furthermore with Diverse-SCIT, due to Attack Surface shifting, some of the attacks that worked before no longer work after self-cleansing and restoration. Traditional reactive

systems are generally static and are indefinitely online. If not for periodic management, the system attack surface size tends to keep growing with time. Gains of SCIT are further highlighted when compared to traditional static systems.

CHAPTER SIX – SCIT BASED MOVING TARGET DEFENSE: WORK FACTOR ANALYSIS

This chapter proposes a game theoretic attack / protect cyber economic model to facilitate designing architectures that are resilient and tilt the asymmetric cyber economic costs in favor of the defender. This work formalizes system security state transitions and intruder / defender work factors associated with all of those state transitions. This component of my research incentivizes logical and architectural solutions that create an ecosystem where the sum of all defender work factors in defending an enterprise over a period of time is much less than the sum of all intruder work factors involved in compromising the enterprise security and ex-filtrating data.

6. 1 Overview

This chapter explores the metrics, measures, and economics of cyber resiliency and asymmetric effects. The chapter examines an approach to shifting adversaries' current advantage in cyber conflicts in favor of defenders. If the economic drivers can be understood, which increase an adversary costs in time, it can then reduce the asymmetry impacts for cyber economic value and resiliency. If progress is to be made, cybersecurity experts will also need to view solutions in economic and policy terms, rather than just technology. Nevertheless, systems will need to integrate cyber resiliency and asymmetry with their co-dependent infrastructures.

Cybersecurity imbalance limits current solutions to an increasing rate and severity of attacks. BIGDATA volume, velocity, variety and complexity are beyond the ability of commonly used tools that capture, process and analyze these security incidents. The Internet of Things is just one example that's driving this new cyber economics. Another seeks incentives for a global security from the insider out of "like-minded, like-valued nations" for new international norms. This chapter explores the metrics, measures, and economics of cyber resiliency and asymmetry effects on the asymmetric balance and examine an approach to shift adversaries' current advantage in cyber conflicts in favor of defenders.

Today, cybersecurity needs a fourth generation cyber security approach focused on Resilience, Restoration and Recovery. Moving Target Defense (MTD) concepts that control changes across multiple system dimensions in order to increase the costs of adversary probing and attack efforts; thus, reducing the window of opportunity and increasing the uncertainty and apparent complexity for attackers. If it is assumed that perfect security is unattainable and that all systems are compromised, MTD research focuses on enabling continued safe operations in a compromised environment and to have systems that are defensible rather than perfectly secure. Self-Cleaning Intrusion Tolerance (SCIT) integrates cyber resiliency and asymmetry with their co-dependent infrastructures.

The rest of this chapter is divided into 8 sections. In the next section, related works for the motivation of this study are discussed. Section 6.3 presents market characteristics for the asymmetric cyber advantage. Section 6.4 introduces SCIT technical

and architectural methodology for gaining an asymmetric advantage. Section 6.5 presents a Cyber Economic Model. Section 6.6 applies SCIT resiliency and security for an optimal balance. Section 6.7 shows case studies of the defender and/or adversary with pre-loaded exploits, and then compares an attacker's work factor to a defender's. Section 6.8 Conclusions of SCIT architecture asymmetric cyber advantage to provide "Security-Driven Resilience".

6.2 Related Work

Cyber Economics is a developing field of study that requires the multi-disciplinary research of social and behavioral scientists, as well as, lawyers and technologists. Ponemon Institute 2015 [Pone2015] annual research provides several takeaways for better understanding the factors that can minimize the financial consequences after a data breach. Littlewood identifies the intruder work factor as a fundamental quantitative measure of security [BLit1993]. Marn-Ling Shing et al proposes the use of game theory concepts (a game matrix) for assessing the Intruder Work Factor likelihoods [SMar2011]. DARPA attempted observing Intruder Work factor in collaboration with Sandia National Labs [GSch2000]. Collectively, they are progressive efforts of the metrics for cyber costs but lack an integrated analyses for today's need of cyber resiliency and asymmetry. So research by Professors Lawrence Gordon and Vernon Loeb on the impact of investments in cyber security measures, cost of responding to security breaches, and impact of a publically acknowledged security breach on stock valuation was assimilated [LGor2002].

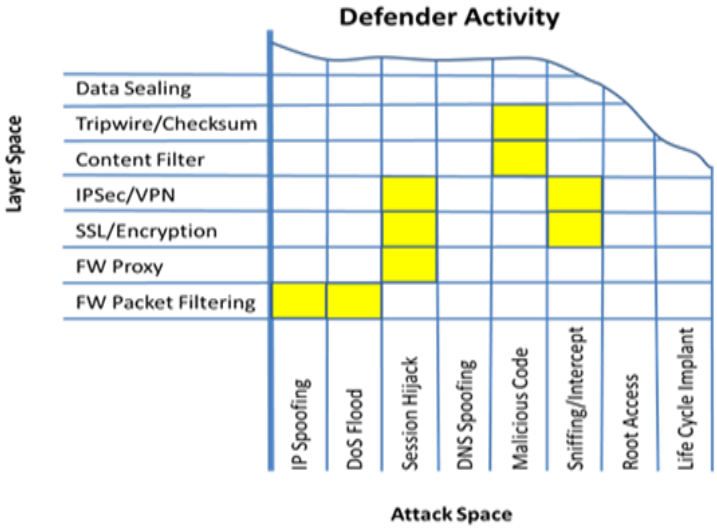
This integrated analysis begins with the intruder security work factors for fundamental quantitative measurements. Figure 6.1, shows a breakout of Adversary Time Expenditure - 95% of the attack is preparation for an attack execution; thereby, changing it would impact the asymmetric costs in today's traditional defensive secure systems [JLow2004].

Cyber resiliency and asymmetry economics needs integrated analyses inclusive of measurement that demonstrate greater value. The Ponemon Institute research reveals that the Mean Time To Identify (MTTI) and Mean Time To Contain (MTTC) the data breach are linearly related to data breach costs. However, today's reactive policies and isolated technologies have only reduced the breach discovery from 234 days to 178 days, and the time for breach containment is reduced from 83 days to 55 days [Pone2015]. If the economic drivers can be understood, which increase an adversary costs in time, the asymmetry impacts for cyber economic value and resiliency can be reduced.

Figure 6.1: Adversary time expenditure

Defenders maintain separate technologies for their activities over multiple layers that attempt to block malicious activities across the attack space. In the cases studies, this attack space is used to apply a variant of a Root Exploit, Brute Force attack, and Dictionary attack to measure some of the activities over the layer space, shown in Table 6.1 [GSchu2000]. Metrics of their resiliency in an asymmetry environment are measured. The goal is to demonstrate that cyber economics decisions should assess Intruder and Defenders Work Factors; consequently, drive the practical constraints of cost, risk, and benefits.

Table 6.1: Defender Activity



6.3 Foundations of Asymmetric Cyber Advantage

Cyber economics aids in the asymmetric balance between the attackers and defenders; whether with a choice of technologies or procedures that prevent and respond or determine an attack type and their efforts to success. Political economists Christopher Coyne and Peter Leeson [CCoy2005] characterize the current defensive strategy as "simply the sum of dispersed decisions of individual users and businesses." For instance, in the Market for Lemons: Quality Uncertainty and the Market Mechanism presents the information asymmetry between buyers and seller. This examines how the quality of goods traded in a market can degrade, leaving only "lemons" behind. The economist George Akerlof presents an adverse selection problem in how high-quality is driven from the market, which leads to a market collapse [GAke1995]. A similar problem exist in today's internet, where information asymmetry exchanges with bad actors and lower quality goods.

Similarly, Jack Hirshleifer's story on the "Island of Anarchia" represents today's internet defenses, where individual families each constructed and maintained a section of the flood wall. Thus, the island's flood defenses is dependent on the weakest link, that is, the laziest family [GHir1983]. Cyber espionage takes advantage of the asymmetric weaknesses, where an individual gains access to intellectual property and exploits this advantage at the market expense.

Cyber defenders bear these asymmetric costs in traditional defensive secure systems, when playing by the attacker's rules. SCIT reverses the advantages for the defenders to reduce the work factors by working with known host system properties –

while an intruder work factor increases in working with unknowns in the target environment. In the case of SCIT, the controllable environment is independent of the previous attacker activities.

6.4 Technical and Architectural approaches to gaining asymmetric advantage

SCIT is a fourth generation cyber security approach with a focus on Resilience, Restoration and Recovery. SCIT research focuses on the critical servers most prone to malicious attacks.

Table 6.2 shows a comparison of security solutions to highlight their different defender work factors [ANag2013]. These factors drive decisions of cyber economics in traditional perimeter defenses for more effective and efficient solutions.

Address Space Layout Randomization (ASLR) improves the effectiveness, when applied to every application to protect binaries from code-injection attacks and obfuscate a process system language to presents an ever-changing target. SCIT also presents an ever-changing target that increases the cost of their probing and attack efforts. Together, SCIT + ASLR improves the asymmetric balance effectiveness though with the added cost and reduced efficiency.

6.5 Cyber Economic Models

In the absence of other comparable economic analysis, the Gordon-Loeb Model [LGor2002] has become the "gold standard" in the area of cyber economic models. They developed an economic model for cyber security based on an analysis of organization spending as well as marginal effectiveness and return on investment of cyber

investments. Among the many findings of their research, the Gordon-Loeb Model makes two important assertions:

Table 6.2: Technical Solutions Work Factors

Security Solution	Defender Work Factors
Traditional perimeter defense reactive solutions – Firewalls, IDS, IPS, DLP	CPU Cycles - RT network traffic monitoring Man Hours - Periodic reconfiguration - Patching and updating - Responding to alarms - Dealing with false alarms - Recovery - Root cause analysis
ASLR, Instruction Set Randomization	CPU Cycles - Overhead Man Hours - Periodic reconfiguration - Patching; updating (less frequent) - Recovery - Root cause analysis
Self Cleansing Intrusion Tolerance (SCIT)	CPU Cycles - Rotation Man Hours - Adaptive exposure time - Patching and root cause analysis
SCIT + ASLR	CPU Cycles - SCIT rotation - ASLR overhead Man Hours - ASLR periodic reconfiguration - SCIT adaptive exposure time - Patching and root cause analysis

- Incremental additional investment in security provides additional benefit by reducing the potential of successful attack up to a point. Beyond this point, there is diminishing (or no) additional benefit for additional investment.
- An organization should not invest more in cybersecurity protection measures than 37% of the expected potential loss due to successful cyberattack.

Using these as baseline for cyber investment hygiene, a case can be presented for using intruder and defender work factors to quantify intruder efforts in compromising a target system and defender efforts in protecting the system. The relevance of the

following vulnerability life cycle is also taken into consideration in presenting a generic state transition diagram that represents one cycle of transition from a vulnerable state to a known good state for the target system.

A Typical Vulnerability Life Cycle is presented below:

- Discovery
- Disclosure
- Release of Patch
- Availability of Exploit

Couple of examples illustrating relevance of relationship between Vulnerability life cycle and Attacker / Defender Work Factor are presented below:

- Disclosure of a vulnerability without a patch decreases Intruder Work Factor
- Release of a Patch increases Intruder Work Factor and decreases Defender Work Factor

The state transition diagram below (Figure 6.2), in addition to representing transition between various system security states, also assign intruder and defender costs for each transitions. Intruder (or) defender actions trigger state transitions represented below and the individual costs for performing such actions.

Costs to the Intruder:

- C_{AR} – Performing target environment reconnaissance
- C_{AA} – Attacking and exploiting target
- C_{AAN} – Attacking and exploiting target w/ no protections
- C_{AAP} – Attacking and exploiting target w/ protections

- C_{AAS} – Attacking and exploiting target within a short exposure window when SCIT is in place along with other protections
- C_{AE} – Ex-filtrating data from target post compromise

$$C_{AAS} >> C_{AAP} >> C_{AAN}$$

Costs to the Defender:

- C_{DP} – Updating services patching security issues periodically
- C_{DT} – Performing testing on updates and patches before deploying them to production environment
- C_{DD} – Monitoring and detecting intrusions and other malicious activities
- C_{DI} – Performing incident response
- C_{DR} – Remediating and recovering systems reactively to last known good secure state as a response to a security incident. Cost is dependent on severity of security incident and assets compromised.
- C_{DS} – Recovering systems periodically using SCIT proactively. Fixed cost independent of malicious event severity.

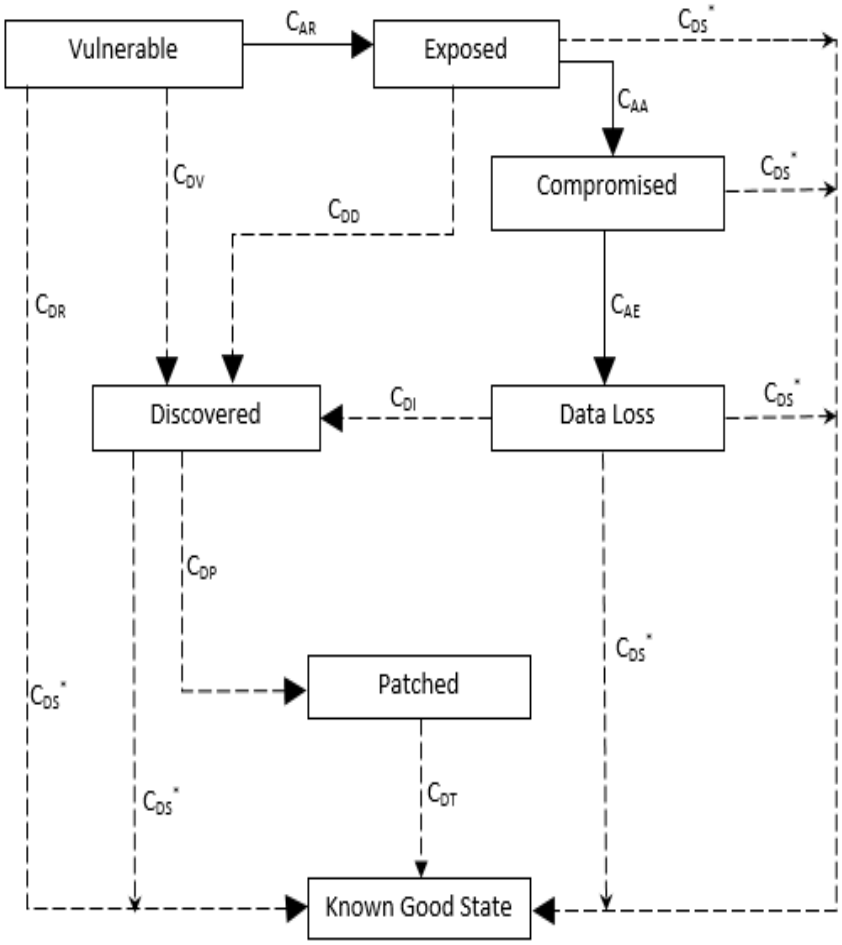


Figure 6.2 - Generic State Transition Diagram and Costs
*** State transitions costing the defender C_{DS} are only applicable in the case of architectures employing Self-Cleansing Intrusion Tolerance (SCIT)**

- C_{DV} – Cost to the defender in performing periodic vulnerability assessments, keeping up with security updates and using threat intelligence to proactively identify security issues

C_A – Sum of all attacker costs C_D – Sum of all defender costs

Primary Objective of the cyber economic model from the defender's standpoint is the following:

Sum of defender work factors across the timeline must be much less than the sum of attacker work factors.

$$C_D << C_A$$

In order to build an attack / protect economic model, a few parameters are defined in addition to the ones discussed above:

- C_{CA} – the value of the critical assets (it is assumed it is the same for both the asset owner and the intruder). The cost to protect an asset should never be greater than the value of asset itself.
- C_P – total cost of protecting the critical assets ($C_{DV} + C_{DD} + C_{DP} + C_{DT}$), per unit, together with a possible amortization of the protection technology's cost over the number of units to be protected.
- P_C – the probability of intruder exploiting the target systems and compromising the critical assets (if no protections are applied)
- P_E – the probability of intruder exploiting the target systems and compromising the critical assets (if protections are applied)
- P_S – the probability of intruder exploiting the target systems and compromising the critical assets within a short exposure window (if SCIT is in place in addition to protections applied)

$$P_S << P_E << P_C$$

An attack / protect economic model based on simple game theoretic formulation is presented below in Table 6.3. Each cell in the game table has two entries:

D – Defender's value, A – Intruder's value

Observations from Table 6.3:

Defender Perspective: It is reasonable for the defender to try and maximize the minimal advantage the defender has over the intruder, namely (D-A). For the value of (D-A) to be high, from the equations in Table 6.3, the following conditions need to be met:

- The cost of protecting the critical assets, C_P in architecture 2 (or) $C_P + C_{DS}$ in architecture 3 need to be lower than 37 % of the value of C_{CA}. The defender cost in performing SCIT (C_{DS}) is a small fixed cost that is neutral to the asset's threat environment. This fixed defender work factor however has an iterative effect on C_{AA} making the intruder's compromise the system 'n' times to achieve their goals. 'n' here is the ratio (Total Intruder Attack Duration / SCIT Exposure Window)
- Probability of intruder exploiting the target P_i needs to be low. Lower the better.
- Cost ($C_{AR} + C_{AA}$) needs to be high. Higher the better.

Table 6.3: A Game Theoretic Attack / Protect Economic Model

			DEFENDER		
			NO PROTECTION (Architecture 1)	DEFENSE IN DEPTH PROTECTION (Architecture 2)	DEFENSE IN DEPTH PROTECTION + SCIT (Architecture 3)
INTRUDER	Adversary takes no action		D: C_{CA} A: 0	D: C_{CA} - C_P A: 0	D: $C_{CA} - (C_P + C_{DS})$ A: 0
	Adversary compromises target critical assets with probability P_C and cost (C_{AAN}) if there are no protections (OR) with probability P_E and cost (C_{AAP}) if there are protections.	Failure	D: C_{CA} A: - (C_{AR} + C_{AAN}) Prob: 1 - P_C	D: C_{CA} - C_P A: - (C_{AR} + C_{AAP}) Prob: 1 - P_E	D: $C_{CA} - (C_P + C_{DS})$ A: - (C_{AR} + C_{AAS}) Prob: 1 - P_S
		Success	D: $C_{CA} - (C_{DI} + C_{DR})$ A: C_{CA} - (C_{AR} + C_{AAN}) Prob: P_C	D: $C_{CA} - (C_P + C_{DI} + C_{DR})$ A: C_{CA} - (C_{AR} + C_{AAP}) Prob: P_E	D: $C_{CA} - (C_P + C_{DS})$ A: C_{CA} - (C_{AR} + C_{AAS}) Prob: P_S
		Expected	D: C_{CA} A: P_C C_{CA} - (C_{AR} + C_{AAN})	D: C_{CA} - C_P A: P_E C_{CA} - (C_{AR} + C_{AAP})	D: C_{CA} - (C_P + C_{DS}) A: P_S C_{CA} - (C_{AR} + n*C_{AAS}) where ratio n = (Total Intruder Attack Duration / SCIT Exposure Window)

All the above conditions are best met by Architecture 3 – Defense in Depth Protection + SCIT (CDS). Therefore, Architecture 3 presents the best platform for the defender to maximize his advantage over the intruder.

Intruder Perspective: It is reasonable for the intruders to maximize their value, namely A. For the value of A to be high, from the above equations, the following conditions need to be met:

- Probability of intruder exploiting the target system (P_C or P_E or P_S) needs to be high. Higher the better.
- Cost (C_{AR} + C_{AA}) needs to be low. Lower the better.
- In the case of Architecture 3, the value of 'n' needs to be low, meaning that the total duration of attack must not be much larger than the SCIT exposure window.

Intruder loses his advantage as the SCIT exposure window gets smaller or as the total attack duration gets bigger.

6.6 Optimal Balance between Resiliency and Security

Resiliency notions have increasingly adopted and resemble security concerns, thus constituting the growth of Security-Driven Resilience. Security has pulled away from its traditional bias and focuses upon the everyday needs of people and population; and in doing so remap its scale in security. Today, the concept of resilience incorporates a vast range of contemporary risks and security challenges [UNIS2012]. Since 9/11, resiliency has increasingly become a central organizing metaphor within the expanding and multi-scale institutional framework of national security and emergency preparedness [PAde2012]. Hence, what the resilience is – becomes less important – than what it does.

Today, cyber defenders bear these asymmetric costs, and need to reduce their work factors for resiliency and security of known's like host system properties. Moving Target Defenses enable practical cyber resiliency, which reverses to a defender's advantage the everyday needs of security personnel and end users. SCIT increases the attacker costs of their probing and attack efforts; while, enabling the continued safe operation in a potentially compromised environment. Their benefit is to have defensible systems, rather than hoping for perfectly secure systems. Cyber resiliency and asymmetry needs integrated analyses of the economics drivers, which show that increase an adversary costs in time reduces the asymmetry impacts for cyber economic value and resiliency.

In order to achieve this balance between Resilience and Security in the context of SCIT, a formal model for the SCIT controller was developed which provides a framework for flexibility in the application of SCIT to facilitate meeting target environment requirements when it comes to security, resilience and performance. The SCIT controller handles the rotation of the virtual machines and is the engine behind the SCIT environment.

A formal representation was important to understand the relationships between the various temporal variables that drive the configuration and by extension, the security and performance of a SCIT environment.

Table 6.4: SCIT Temporal Configuration Variables

$\mathbf{T_E}$	Exposure time (s)
$\mathbf{T_S}$	Service time + Switch time (s)
$\mathbf{T_I}$	Query inter arrival time (ms)
$\mathbf{T_R}$	Time taken to restart server (s)
V	Number of virtual servers used
$\mathbf{T_Q}$	Time spent in quiescent state (s)
$\mathbf{Q_i(t)}$	Service requests received till time t.
$\mathbf{Q_p(t)}$	Service requests processed till time t.
N(t)	Service requests queued at time t.
$\mathbf{V_O(t)}$	Number of Virtual Servers online at time t.
$\mathbf{V_R(t)}$	Number of Virtual Servers restarting at time t.

Based on the temporal variables defined above in Table 6.4, the following relationships in Table 6.5 were developed and validated with test-bed experiments.

Table 6.5: SCIT Test Bed Configuration Variables

$N(t) = Q_i(t) - Q_p(t)$
$Q_i(t) = t / T_I$
$Q_i(t) = Q_i(t-1) + (1 / T_I)$
$T_Q = N(t) * T_S$
$Q_p(t) = Q_p(t-1) + x$ where $x = [V_O(t) - V_R(t)] / [V_O(t) * T_S]$
$T_R + T_Q < T_{Emin} * (V - 1)$
$T_{Qmax} <= [T_E * (V-1)] - T_R$
$V_O(t) \propto T_S / [T_I * T_E]$
$V_R(t) \propto 1 / T_E$

These relationships help in configuring parameters like Exposure Time, Quiescent Time and Number of Virtual Servers that need to be online at any given time to be able to meet the security and performance requirements of an environment. These relationships also provide flexibility in terms of configuring parameters to assure resilience as deemed necessary in an environment.

<u>Adaptive SCIT:</u>

- Having these metrics handy helps to take a step towards Adaptive SCIT - adding Resilience to SCIT.
- High compute environments are generally resource intensive and computing needs vary greatly with time. Above established metrics and relationships give the

ability to dynamically adapt SCIT working parameters on perceived system environment changes.

- Misconfigurations such as having too large an exposure window could impact system security. Similarly having too few virtual machines online could impact performance. There is a need to continuously monitor service demands and threat landscape in order to strike a balance between usability, performance and security at all times.

6.7 Use Cases for Defender and/or Intruder that include Work Factors

Hypothesis: "Intruder and Defender Work Factor" are quantifiable metrics. With 100K+ new unique malware samples per day [McAf2013], the goal of a security strategy should be: Increase the ratio of the attacker's work factor to the defender's work factor.

In order to validate the goal, test bed experiments were performed for which two architectures were considered: Non-SCIT and SCIT architectures.

- Test Bed Setup: Apache Tomcat 5.5.36, Gateway p7805u FX – 2.26 GHz Intel Core 2 Duo P8400; 4GB 1066 MHz DDR2 RAM, Windows Vista Premium 64 bit OS
- Pre-loaded exploits in the Metasploit Framework used – hence the results do not account for 'Exploit Development' time

Two use cases are presented, both of which show quantifiable metrics and an asymmetric cyber advantage for the Defender Work Factor against the Intruder [ANag2013].

Table 6.6 timeline of a samba remote root exploit for a buffer overflow attacks shows an asymmetric cyber advantage for an exploit in 1/2 the time of intruder work factors. An ASLR configurations slightly increases an intruder's workload, but the combination with randomization more than doubles the ratio of intruder work factor to defender work factor. Randomization increases the uncertainty and apparent complexity for attackers, shown in figure 6.3. This reduces their window of opportunity and increases their costs of probing and attack efforts. Similarly, SCIT technical and architectural approach also gains an asymmetric advantage that interrupts the Buffer Overflow. SCIT provides security and resiliency, which interrupts and forces a restart that loads a clean image of the server's operating system and application into the Virtual Machine. Thereby, provides an optimal balance between resiliency and security.

Table 6.7 WAR Backdoor use case represents the earlier Adverse Time Expenditure breakout [ANag2013]. The intruder work load factors on an Apache Tomcat is 95% of an attack's execution. Even though an intruder asymmetric cyber advantage is executing the exploit in 11 seconds, the missed opportunity is disrupting the work load factors over their 134 seconds of preparation.

Further, the cost model defined above was applied to the Apache Tomcat test bed experiment to investigate the impact of SCIT on intruder costs.

Table 6.6: Test Case Buffer Overflow Work Factors

Timeline	Intruder Work Factor	Impact of SCIT
Develop linux x86 Samba remote root exploit	Minutes to hours based on Intruder Capability	
Scan target system Samba and assess version	19 seconds	Target system location changes: re-scan
Configure brute force exploit 'trans2open' with payload 'bind_tcp' in MSF	24 seconds	Target system location changes: re-configuration
Exploit without ASLR	17 seconds	Interrupt BF attack: restart
Launch exploit with ASLR and no re-randomization	22 seconds	Interrupt BF attack and force restart
Launch exploit with ASLR and re-randomization	42 seconds	Interrupt BF attack and force restart

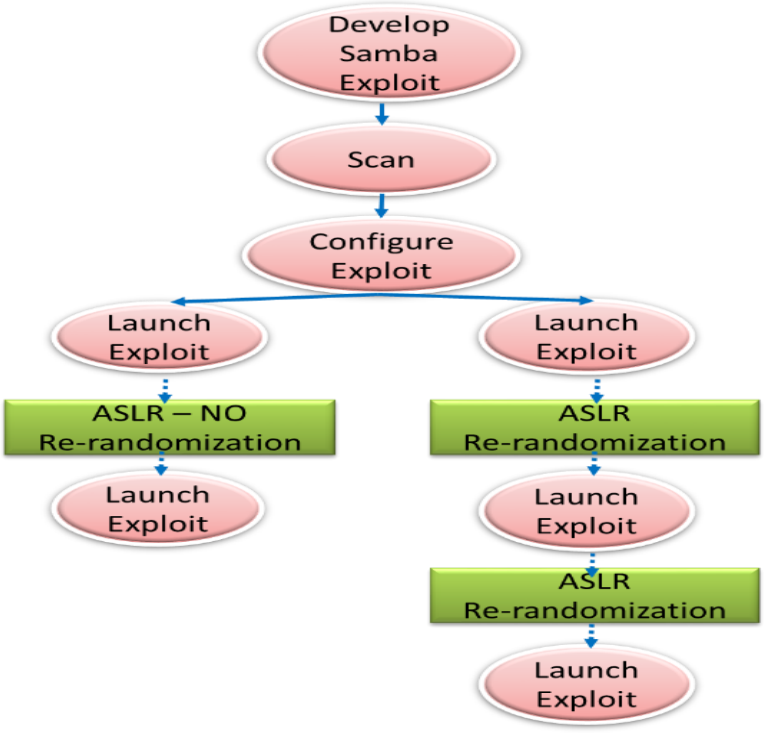


Figure 6.3: Samba test case exploit work flow

SCIT technology implements defensive side of procedures, which prevent and respond to cyber-attacks. SCIT technical and architectural approach interrupts the attacker, as shown in table 6.7 WAR Backdoor chart and Figure 6.4 flow diagram shows seven stages to force a restart and reload the server's operating system and application into the Virtual Machine. An optimal balance between security and practical cyber resiliency, where resiliency tool supports the economic work load factors for defender at an adversary expense. SCIT resilience, restoration and recover increases the uncertainty and apparent complexity for attackers, reduce their window of opportunity and increase the costs of their probing and attack efforts.

Furthermore, defensive opportunities that disrupt an intrusion are not limited to the initial beach exploit activities. For example, South Carolina Department of Revenue malicious (phishing) emails, where one user click on the embedded link unwittingly executed a malware. Over the next two months, attacker's compromised 44 systems with one malicious backdoor software to steal three database files, and another to send data out. Additionally, there were at least 33 more unique pieces of malicious software, password tools, batch scripts and administrative utilities, which executes commands to perform the attack and data theft activities [MHei2012]. Blocking these sequence, in additional to the initial, not only increases the intruder work factors but the likelihood of an intruder success.

The foundational and applied advances effect the asymmetry and resiliency in cyber economics. Collectively, they drive essential system requirements for cyber

systems, which includes traditional IT, cloud platforms, cyber-physical systems, and critical infrastructure.

Table 6.7: WAR backdoor test case work factors

Timeline	Intruder Work Factor	Impact of SCIT
Scan target network - Apache Tomcat Servers w/ versions.	C_{AR} = In the case of our test bed, 24 seconds	Target system location changes on rotation: re-scan. Thereby substantially increasing C_{AR}
Dictionary attack to get login credentials for Tomcat App Manager in MSF	C_{AA} = Depends on intruder capability and dictionary size. In this case, 142 seconds	Interrupt dictionary attack: restart BF attack. Increases C_{AA} every time an attack is interrupted.
Create WAR for tomcat_mgr_deploy using payload 'reverse_tcp'	C_{AA} = Depends on capabilities. In our simulated test bed, 94 seconds	
Upload WAR file to Tomcat App Mngr	C_{AA} = 18 seconds in our experiment	WAR file will be removed. Intruder has to repeat step every time the WAR file is automatically removed by SCIT. Substantially increases C_{AA}.
Run exploit and access JSP backdoor through browser	C_{AA} = 11 seconds in our experiment	JSP file will be removed thereby making the backdoor useless. Intruder has to repeat previous 2 steps to re-exploit the vulnerability. Substantially increases C_{AA}.

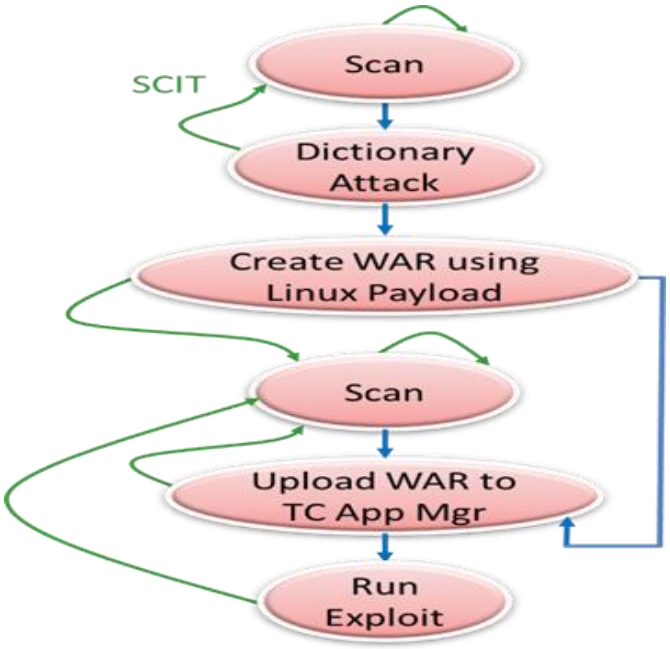


Figure 6.4: WAR backdoor test case workflow

6.8 Summary

SCIT "Security-Driven Resilience" architecture provides a robust security mechanism that guarantees certain security properties by limiting the exposure time. It is a cyber- resiliency tool guided by economic factors for defender and/or intruder. SCIT is an optimal balance between resiliency and security that provides administrative and economic benefits as a reasonable choice to be included in security architecture. Additionally, SCIT does not generate false alarms, an important advantage of SCIT compared to traditional reactive solutions like IDS, which helps reduce the intrusion alerts management costs. SCIT provides a cyber-advantage that benefits environments where technical skills are limited; for examples, environments of remote and rural locations, small organizations, tactical military settings, and emerging countries. The

simulation studies presented suggest that an implementation of SCIT on host servers provide a robust architectural solution in the face of new attacks.

CHAPTER SEVEN: μ-SCIT – ADDING MODULARITY TO SCIT

This chapter proposes μ-SCIT, a hybrid architecture that adds modularity to SCIT using Operating System level virtualization. The proposed architecture is built on top of container-based virtualization technology. The added modularity affords the ability to perform more frequent targeted granular rotations at the level of processes and applications. This in turn extends ability of SCIT to work with long running applications and handle long transactions using container check-pointing and migration.

7.1 Overview

In previous chapters, we have defined Self-Cleansing Intrusion Tolerance (SCIT), a recovery based intrusion tolerance technique that comprises of rotating Virtual Servers (VS) once every period of time known as the 'exposure window'. Rotating a virtual server here entails killing a virtual server and restarting it using a last known good configuration 'golden image'. Needless to say, there are certain challenges in performing such SCIT rotations, some of them being:

1. Resource intensive – depending on the size of the VS, killing it and restarting it once every period of time (especially if the exposure window is small) could have an impact on the performance of the system. Also, the number of virtual machines that need to be delegated to perform SCIT tasks depend on a) time it takes to do a single SCIT rotation (greater the time, more number of VS'

required) and b) target ‘exposure time’ (smaller the exposure time, more number of VS’ required).

2. Wasteful – it is well documented that most of the vulnerabilities that exist and those can be exploited are application vulnerabilities and not Operating System vulnerabilities. In order to cleanse applications that may have been compromised, it is wasteful to rotate the entire VS along with its’ operating system components.
3. Persisting long running applications – although SCIT (as is) works well for systems that perform relatively small transactions (example: E-commerce), it faces challenges in environments where there is a need to persist long running transactions (example: media, critical military applications). Persisting such long running transactions across a full VS rotation without downtime is a tough ask.

To address these challenges, this chapter proposes µ-SCIT that extends traditional SCIT by adding modularity to it. Goal here is to perform micro level rotations that are at a more granular level than full VS rotations.

7.2 Need for Modularity

7.2.1 Performance Argument: Exposure Time as a metric for proactive risk management

Security and performance must be dealt within one framework – high security with low performance is as unacceptable as low security and high performance. In this chapter, we assess the effectiveness of using exposure time as a metric to tradeoff

between security and performance. This metric exacerbates the need for adding modularity to SCIT.

Lower exposure times mean lower intruder residence times leading to less damage inflicted. Although reducing the exposure time increases security, it also increases computing overhead thus reducing service throughput. It is essential to strike a balance. We use throughput and response time to assess service performance.

We now introduce the term 'Exposure Factor':

Exposure Factor (EF) = percentage loss if a threat is successfully realized

The following observations must be kept in mind while configuring an exposure time:

- Given intrusion has not occurred, penetration risk increases with time – indicates a need for low 'exposure time';
- Given intrusion has occurred, progress of intrusion increases with time – indicates a need for low 'exposure time';
- Low exposure times lead to low exposure factor thereby resulting in minimum losses;

SCIT Controller Model:

The SCIT controller is the despatcher that keeps all the virtual servers in order by rotating them in and out to maintain exposure window and perform self-cleansing. The SCIT Controller model 'S' can be formally represented using the following:

$$S \rightarrow V_1 V_2 V_{3\ldots} V_n S$$

Where V1…Vn stand for 'n' Virtual Servers respectively

Virtual Server 1 gets exposed first. Once the exposure time of Virtual Server 1 runs out, dispatcher switches control to Virtual Server 2. Now, Virtual Server 2 becomes exposed and Virtual Server 1 is pushed into quiescent state before restarting. Once all virtual servers have been exposed once, it is Virtual Server 1's turn to be exposed again. Despatcher maintains this cycle.

Having formalized that, here are some general observations:

- Higher number of virtual servers gives the flexibility to attain lower exposure times.
- Managing large number of virtual servers take up a lot of CPU cycles thereby leading to poor query response times.
- More virtual servers lead to more frequent server rotations meaning more processing.
- It is relatively simple and inexpensive to manage a small number of virtual servers; however this results in larger exposure times and more risk of penetration.
- In this framework, number of VS on-line simultaneously has an impact on throughput and response times.
- From Simulation runs, we infer that having more number of VS on-line simultaneously results in a healthy increase in throughput. This we believe is primarily caused by the VS's in the quiescent state.
- Having more number of VS on-line leads to higher response times. This is not a desirable effect.

Taking all these observations into consideration, although lower exposure times lead to more secure servers, they also use up a lot of CPU cycles performing frequent full VS rotations thereby bringing down the service throughput. In order to address the conundrum of providing low exposure times as well as having low performance impact, this work proposes adding modularity to SCIT where critical applications are rotated on a more frequent basis thereby addressing low exposure time targets as well as not having to rotate the entire VS to achieve the same thereby addressing low performance impact requirements.

7.2.2 Security Argument

From Figure 7.1, Microsoft Security Intelligence Report Volume 8, 2009 suggests that most of the vulnerabilities that lead to intrusions are in fact application vulnerabilities and not operating system vulnerabilities. SCIT performs full virtual server rotations thereby rotating operating system components as well periodically, while in reality, these operating system components and other system software are least susceptible to compromise. To overcome such wasteful rotations, my research proposes adding modularity to the current model that treats the entire virtual server as one gigantic program.

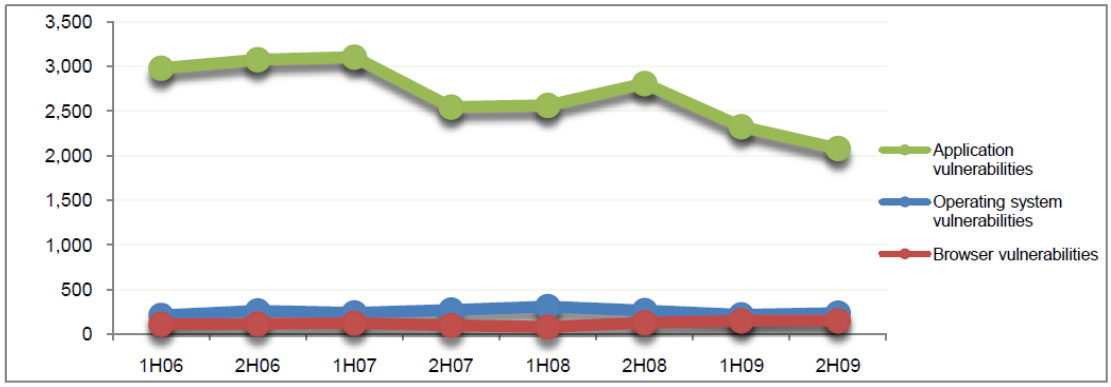


Figure 7.1: Vulnerabilities trend in Windows systems, 2006-2009 [MSSI2009]

Micro-SCIT (µ-SCIT) proposes a different design that adds granularity to the SCIT rotations by extending the base operating system functionality by means of user-space servers (or) modules (or) containers. By splitting a running virtual server into small, independent parts, the system becomes less complex and more robust, because the smaller parts are more manageable and help to isolate faults. Advantages of such a modular approach over a traditional monolithic approach are as follows:

- First, by splitting up the virtual server into multiple isolated modules or containers, we are not reducing the number of bugs but we are significantly reducing the damage that each bug can cause.
- Second, by breaking the operating system into many processes, each with its own boundaries, we greatly restrict the propagation of faults. A bug in a running application cannot inadvertently wipe out the file system by accident. This hugely helps in isolation of faults and prevention of malware propagation.

- Third, by constructing the system as a collection of modules, the functionality of each module can be clearly determined, making the entire system much easier to implement and secure.
- Fourth, this design facilitates long transactions. Persisting a long running application is now a matter of check-pointing and restoring a module specific to that application and does not involve the other components of the virtual server that are constantly being restored to last known good configuration by SCIT.

In order to add said modularity, the use of OS-level virtualization is proposed.

7.3 Operating System Level Virtualization

Operating system level virtualization is a server virtualization method where the kernel of an operating system facilitates the use of multiple isolated user-space instances, instead of just one. Such instances are called Containers or Virtual Environments (VE) or Virtual Private Servers (VPS) or Jails. A sample illustration of a container is provided in Figure 5.2. Operating systems that support container based virtualization facilitate:

- Running multiple isolated sets of processes under a single kernel instance
- Check pointing – saving complete state of container and later restarting
- Check pointing and restarting are implemented as loadable kernel modules

From kernel view, containers is a separate set of processes completely isolated from other containers and host systems. Container is an isolated entity (all inter-process communications (IPC) and parent-child relationships are within the container boundaries). Traditional h/w virtualization approaches like Xen and VMWare only support check pointing / restarting of an entire OS environment; they do not support

check pointing and restarting of a small set of processes as required by the µ-SCIT proposed above.

7.3.1 Container Check-pointing and live migration

In order to achieve the goals of µ-SCIT as outlined in Section 7.1, it was necessary to achieve container check-pointing and live migration. OpenVZ, which is a container based virtualization technology for Linux was used to implement a test bed version of µ-SCIT and to assess its effectiveness.

Basic requirements for container-based operating system virtualization to facilitate check-pointing and live migration can be outlined as follows:

- Process Id (PID) virtualization – same PID has to be assigned to a process as it had before check-pointing
- Process group isolation – make sure parent child relationships will not lead to outside a container
- N/W isolation and virtualization – all networking connections will be isolated from all other containers and host OS
- Resource virtualization – to be independent from hardware and able to restart container on a different server

In addition to the basic requirements outlined below, in order to persist a transaction / application through a SCIT rotation across servers, there is a need to maintain system consistency – meaning all relevant state information and application specific private data need to be carried over:

- Register set, address space

- Allocated resources, network connections
- Per-process private data
- Process-hierarchy, IDs

A typical container check-point / restore life cycle would go through the following:

- Freeze processes (source)
- Dump container to dump file (source)
- Stop the container (source)
- Restart the container (destination)
- Restart the processes (destination)
- Resume processes within the container (destination)

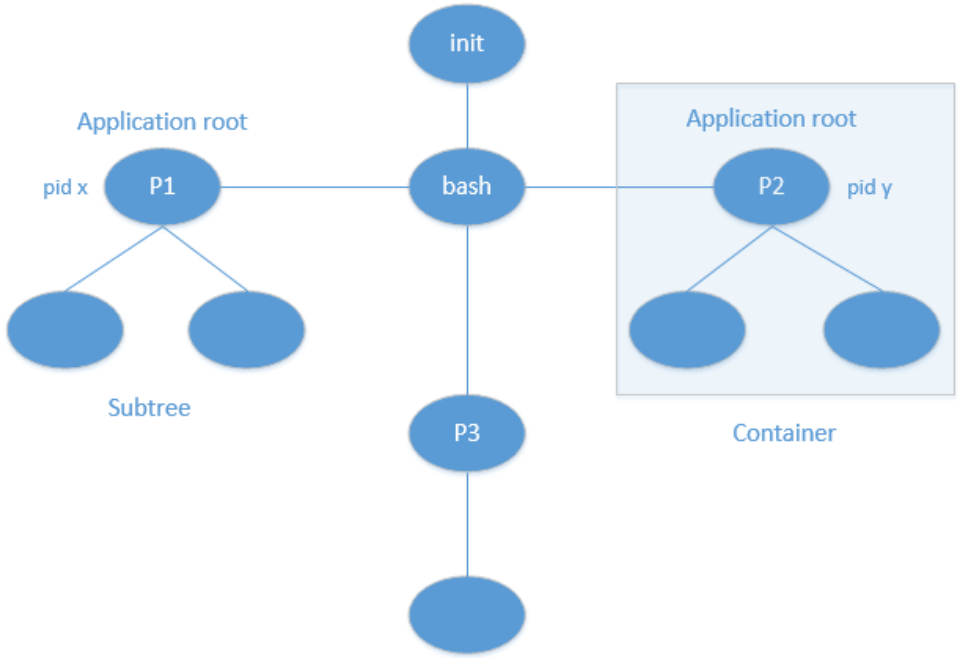


Figure 7.2: Sample illustration of container

7.3.1.1 Container migration challenges and considerations

Like any technology solution, container migration has its own sets of challenges and considerations. As noted earlier in this section, the crux of container migration is in dumping the container to a dump file and moving the dump file to target destination. Needless to say, the performance of this container migration / restoration strategy hinges on the size of the dump file. We noted in Section 7.3.1 that all container private data and relevant state information needs to be carried over to facilitate seamless resumption of long running transaction / applications without loss of availability. To achieve this goal, dump file size needs to be minimal.

As part of validating the practicality of this approach, a test-bed experiment was conceptualized and built to simulate use case scenarios. The configuration of the system used to build the test-bed is presented below. It needs to be noted that the system specs are well below current state of the art servers and this was deliberate to portray performance efficacy of this approach.

Configuration of System used for Test Bed Experiment:

- Gateway P-7805u
- Intel Core 2 Duo CPU P8400 @ 2.26GHz
- 4 GB RAM
- CentOS 6

As a result of performing multiple test bed experiments, find below a summary of dump file sizes that needed to be moved to destination in order to perform respective container migrations under various scenarios using OpenVZ.

Table 7.1: Sample container dump file sizes

Size, Mb	Scenario
0.9	1 Bash
1.2	Bash + SSH Daemon
2	Bash + Idle Apache Tomcat
2.4	POSTGRESQL Server Running with Table of 300 entries open
3	6 Bashes
3.6	MySQL Daemon
16	Acrobat Reader with an open 12 Mb file
26	Bash + Mozilla Firefox (Default Home Page) + Java VM

The test bed experiments were not meant to be complete or to represent every possible scenario. It was meant to illustrate the sample dump file sizes in scenarios while attempting to migrate some fairly common applications. As can be noted from our results, the dump file sizes are manageable and should not cause any noticeable performance impacts given today's network bandwidth and data transfer rates.

It is recognized that the scenarios highlighted above are not complete and that more resource intensive applications could lead to larger dump file sizes. In order to address those scenarios, some potential options to minimize dump file sizes are presented in the following subsection.

7.3.1.2 Minimizing Dump File Size

In order to minimize dump file sizes and facilitate seamless rotation of Virtual Private Servers, following tweaks could be effectively performed as was learned in the process of running multiple test bed experiments.

In the context of using OpenVZ in CentOS Linux for my test bed experiments, one of the options to minimize dump file size is to avoid carrying over unnecessary container private data and folders. This can be achieved by creating a file /tmp/exclude.txt in the destination VPS with entries to exclude migrating. Sample file can contain entries like:

- /tmp
- /boot
- /lib/modules
- /etc/blkid
- /etc/mtab
- /etc/lvm
- /etc/fstab
- /etc/udev

In addition to excluding migration of unnecessary folders, certain system services can also be disabled / uninstalled since they will not be required in the context of a container. Sample list provided below:

- acpid, amd (not needed)
- checkfs, checkroot (no filesystem checking is required in container)

- clock (no clock setting is required/allowed in container)
- consolefont (container does not have a console)
- hdparm (container does not have real hard drives)
- klogd (unless you use iptables to LOG some packets)
- keymaps (container does not have a real keyboard)
- kudzu (container does not have real hardware)
- lm_sensors (container does not have access to hardware sensors)
- microcodectl (container can not update CPU microcode)
- netplugd (container does not have real Ethernet device)
- irqbalance (this is handled in host node)
- auditd (not needed in container)
- lvm2-monitor (no LVM in containers)
- ntp/ntpd (clock taken from host node)

7.4 A comparison of SCIT with Container Migration

Container Migration is not meant to be a substitute for SCIT. It is meant to be complementary. Although SCIT is more resource intensive than container migration, it assures sanctity of the new virtual server going online by employing the use of an internal ‘golden image’ that is not public facing and hence not accessible from the internet. With regard to container migration, although it has its performance benefits of facilitating micro-rebooting and security benefits of performing targeted actions, there are no guarantees when it comes to the security posture of the container. Having made those

distinctions, our research proposes the use of these two solutions together complementing one another.

A brief security comparison of SCIT and container migration is provided below:

Table 7.2 Security comparison of SCIT and container migration

	SCIT	Container Migration
Restore to clean state (no malicious activity detected)	Yes	No. State information + potential malicious data carried over
Restore to clean state (malicious activity detected)	Yes	Yes, can go back to a previous clean checkpoint (if known)
Malware propagation within applications	Does not prevent	Prevents – due to use of isolated containers

7.5 µ-SCIT – Adding Modularity to SCIT using OS-level virtualization

µ-SCIT is an architecture that brings together the two technology solutions presented in the previous section – SCIT and container migration. Every virtual server in the resulting µ-SCIT architecture would resemble the illustration in Figure 7.3. It is a representation of a µ-SCIT virtual server built on top of OpenVZ virtualization for Linux. OpenVZ provides the ability to build isolated containers (or) Virtual Private Servers (VPS) that independently contain within them all the processes, child processes, network dependencies, application software and relevant container private data to function on its own. This in turn provides an ability to be able to kill, restart or migrate the VPS without having to deal with the other components and layers.

When SCIT and container migration work in tandem and as a complement to one another, the resulting µ-SCIT architecture would perform two levels of rotations as summarized below:

1. Once every ‘exposure time’, VMM kills the virtual server (say VS #1) that is exposed and starts another virtual server (say VS #2) that was in live spare. During this process, the VS is self-cleansed and is loaded from a ‘golden image’ next time it goes online. Meanwhile, VS #2 is exposed and this cycle goes on.
2. Within every virtual server (say VS #2) that is currently exposed, all critical Virtual Private Servers are rotated once every ‘container lifetime’.

Critical Virtual Private Servers are the VPS’ that have been identified beforehand during an audit or by system / network administrators to be

- Critical to services provided and/or
- Have an escalated threat posture with a potential for compromise
- Candidates for long running transactions

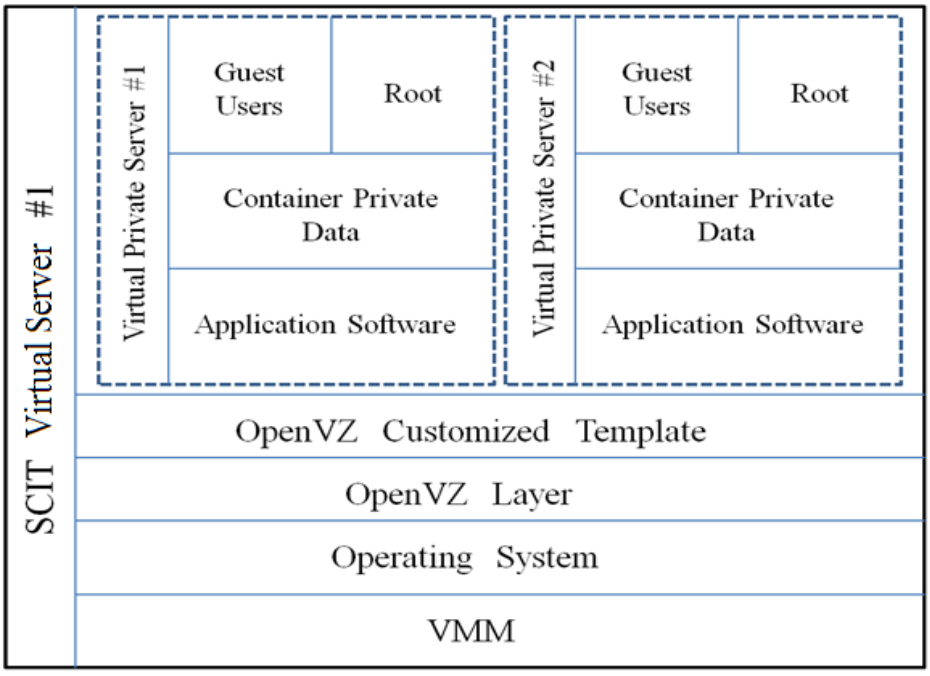


Figure 7.3: Representation of µ-SCIT Virtual Server (VS) instance

Once such a determination has been made, these VPS' go through rotations once every period of time known as 'container lifetime'. In the context of VPS, rotations imply:

- being migrated meaning check-pointed and restored (or)
- be self-cleansed meaning VPS gets replaced by a known-good version of the same

In µ-SCIT architectures, use of 'Container Lifetime' that is much smaller than 'Exposure Time' is strongly recommended thereby creating an ecosystem where critical VPS rotations take place far more frequently than SCIT rotations.

7.6 Summary

In this chapter, we proposed µ-SCIT – a hybrid approach that combines SCIT and container-based virtualization. By adding modularity to SCIT, µ-SCIT can be summarized as follows:

- Offers another layer of rotation in addition to the existing SCIT rotations
    - More granular – rotation happens at the level of VPS and not VS
    - More frequent – since the size of the VPS is relatively small compared to the VS in most cases, this affords an opportunity to perform targeted cleaning and migration on a more frequent basis
    - VPS rotations happen once every 'Container Lifetime' where Container Lifetime << Exposure Time
- Facilitates long transactions / long running applications

- Long running application needs to be identified beforehand and placed into a dedicated VPS
- VPS is migrated (check-pointed and restored) once every ‘Exposure Window’ from source server to destination server in order to persist full SCIT rotations.

CHAPTER EIGHT – RECOVERY BASED RESILIENT CYBER ECO-SYSTEM

This chapter proposes a ‘stand-alone’ and a ‘collaborative’ architecture which make use of information provided by the enterprise Security Information and Event Management (SIEM) solution to perform adaptive intrusion tolerance in unsupervised learning environments. Resilient systems need to be adaptive, and to achieve this goal, this research shows how environmental information can be used to adaptively change system and operational parameters.

8.1 Overview

Today’s approach to security is largely based on perimeter defense and reactive strategies like IDS / IPS systems, Firewalls and Anti-virus products. Past experience has repeatedly shown us that this strategy is not complete and secure. Intrusion tolerance is an approach which treats intrusions as inevitable and shifts the focus from detection and prevention to containing losses and rapid recovery. It can be suggested that a complete security strategy is one which does defense in depth and involves both traditional security strategies and intrusion tolerance. Security Information and Event Management (SIEM) is a framework which consolidates the plethora of information available from all of the network and security devices into useful information. In this chapter, a stand-alone and a collaborative architecture is proposed which make use of information provided by the SIEM framework to perform adaptive intrusion tolerance in unsupervised learning

environments. Resilient systems need to be adaptive, and to achieve this goal it is shown how environmental information can be used to adaptively change system parameters.

The variety and complexity of cyber attacks are ever increasing. Verizon Business 2012Data Breaches Investigation Report [Veri2012] shows that customized malware is difficult to detect and data ex-filtration often occurs over a period of days, weeks and months. The current intrusion detection and prevention approaches are reactive in nature and depend on a priori information which is inadequate to prevent all attacks. Events such as the VeriSign security breach [VSig2012] and the Playstation Network breach [Play2011] reinforce two notions: 1) even the most sophisticated IDS / IPS systems fail to detect / prevent every intrusion and 2) once the system is compromised, the intruder stays in the system doing damage for extended periods of time.

In addition to the shortcomings of IDS / IPS systems, the costs of operating them are high and increasing. To illustrate the issue the example of an enterprise with an average of 1,000,000 raw events occurring per day is considered. About 10,000 alerts are generated by perimeter defense systems. Out of these, 100 alerts are correlated on the basis of severity and other considerations. Assuming it takes 1.5 man hours to handle one alert, a total of 150 man hrs is required per day to handle alerts generated. The cyber security requires 365 days 24 hours per day support and in general about 30 people are required to carry out this task. How many large companies can afford such an allocation of manpower – in companies we talk to, only 2 or 3 people perform this task. What's worse, 50 % of the alerts are false positives – a tremendous waste of resources. With ever

increasing bandwidth and millions of new malware created every day, these numbers are bound to increase.

Despite years of research and investment in developing such reactive security methodologies, our critical systems remain vulnerable to cyber attacks. The reactive perimeter defense approach relies heavily on threat modeling and vulnerability elimination. It is suggested that additional attention should be given to the consequences of a successful attack. In the proposed approach, the focus is on limiting the consequences, like reducing the losses that are induced. It is believed that we must make our cyber systems more proactive and resilient. Such systems will have the property of (1) supporting continuity of operations – working even in the presence of an intruder; (2) losses, if any, must be limited; (3) systems must resume full operations, i.e. system must be restored to a known good state; and (4) the resilient system operations should be independent of the threat.

To design such a system, it is assumed that intrusions are inevitable. Therefore, the focus is shifted from modeling threats / vulnerabilities to developing methods that will minimize the consequences of an intrusion, increase the work effort of the adversary and increase the visibility of the adversary to the defenders. For this, a 'Moving Target Defense' approach to computer security is developed. The focus is on building mission resilient systems that are able to work through an attack. To ensure reliable operations, the system is restored to a pristine state once every short period of time known as the 'exposure time' – thus negating any malicious action performed by the adversary and minimizing consequences. In addition to this, redundancy is used to provide

uninterrupted service and increase overall system availability. The more frequent the computer restoration the less likely it is for the intruder to do damage. The restoration frequency can be random to confuse the adversary and increase his work effort. The shortest time between restorations is a trade-off between available system resources and the throughput of the computer. This intrusion tolerant technology is called Self Cleansing Intrusion Tolerance (SCIT) [YHua2006]. The recovery driven approach of SCIT is compared to the detection driven and other intrusion tolerance approaches in [QNgu2011].

Consistent with CrossTalk's theme for the September/October issue, in this chapter, a resilient cyber eco system is proposed in which every member is able to work together and learn from one another in near-real time to predict and prevent cyber-attacks, limit propagation of attacks across participating entities, minimize losses occurring from successful attacks and rapidly recover to a pristine state. To build such a system which is resilient to a variety of sustained attacks, a model is proposed that integrates tools and mechanisms that provide protection and detection as well as adaptive tolerance. Rest of the chapter is organized as follows: Section 8.2 provides a brief overview of how SCIT works and motivates the rest of the chapter by presenting the need for adaptive SCIT, Section 8.3 introduces Security Information and Event Management (SIEM) solutions and presents the idea on how information from SIEM solutions can be used to build adaptive intrusion tolerance systems. Two scenarios will be reviewed – stand-alone adaptive intrusion tolerance architecture and a peer-to-peer collaborative intrusion tolerance architecture.

8.2 Need for Adaptive SCIT

Resilient systems have to exhibit adaptive and recovery behavior. SCIT is recovery driven, and in this section it is shown how SCIT can be made more adaptive to the ongoing changes in the environment.

At any point of time, the resilience of a SCIT system is affected by (1) the current attacks; (2) the current workload; (3) the current data integrity level; (4) the current data availability level; and (5) the current behavior of the system [PLue2003]. The first four factors together make up the environment of the SCIT system. Two SCIT systems with different behaviors can yield different levels of resilience. This suggests that as the environment and the behavior of the system changes, the effectiveness of SCIT changes as well. To achieve the maximum amount of resilience, the SCIT system must adapt itself to its environment. Through an architecture for adaptive SCIT, we can (1) adapt SCIT to different application semantics; (2) significantly improve the cost-effectiveness of SCIT; (3) prevent dramatic performance degradation due to system environment changes; and (4) maintain trade-off between system security and system performance [PLue2003].

In the case of SCIT, the primary metric is "Exposure time". In [ANag2010], the relationship between exposure time and security of a system is illustrated in terms of data compromised. In [QNgu2009], the SCIT approach was discussed from the perspectives of effectiveness, tunable parameters, performance impact, and integration to application systems. From the derived expression for MTTSFSCIT, it was conjectured mathematically that decreasing the exposure time window will improve the resilience of a SCIT-based system. To adapt SCIT we will need to adapt the exposure time in response

to systems parameters. Increasing MTTSFSCIT would require decreasing the exposure window; hence the cycle that a SCIT server has to go through will become shorter. In this space, there is a tradeoff between system security, performance and cost. Adaptive SCIT could help balance this trade-off in real time with the use of a dynamic exposure time window given the current operating environment and system behavior.

8.3 Use of Security Information and Event Management (SIEM) Solutions

"The term Security Information Event Management (SIEM), describes the product capabilities of gathering, analyzing and presenting information from network and security devices; identity and access management applications; vulnerability management and policy compliance tools; operating system, database and application logs; and external threat data". [SIEM2012]

In addition to receiving inputs from IDS / IPS systems, a SIEM solution will be used to collect and correlate data from all the other sources mentioned in Figure 8.1 to characterize overall network behavior. This behavioral pattern is then compared with a database of normal network behavior patterns to identify irregularities. Based on the findings of this comparison and the severity of the irregularities, the SCIT controller tunes the "exposure time" of the SCIT-ized system to adapt to the current environment. Similar iterative periodic comparisons will help guide the unsupervised learning and automatic adaption of the SCIT-ized system.

8.3.1 Use of information from SIEM solutions in building adaptive Intrusion Tolerant Systems

In this section, the idea of using aggregated information from SIEM solutions to build adaptive intrusion tolerant systems is expanded on. For the purposes of this chapter, SCIT is the intrusion tolerance architecture of choice.

To address the needs outlined in section 8.2, an adaptive SCIT framework must do the following:

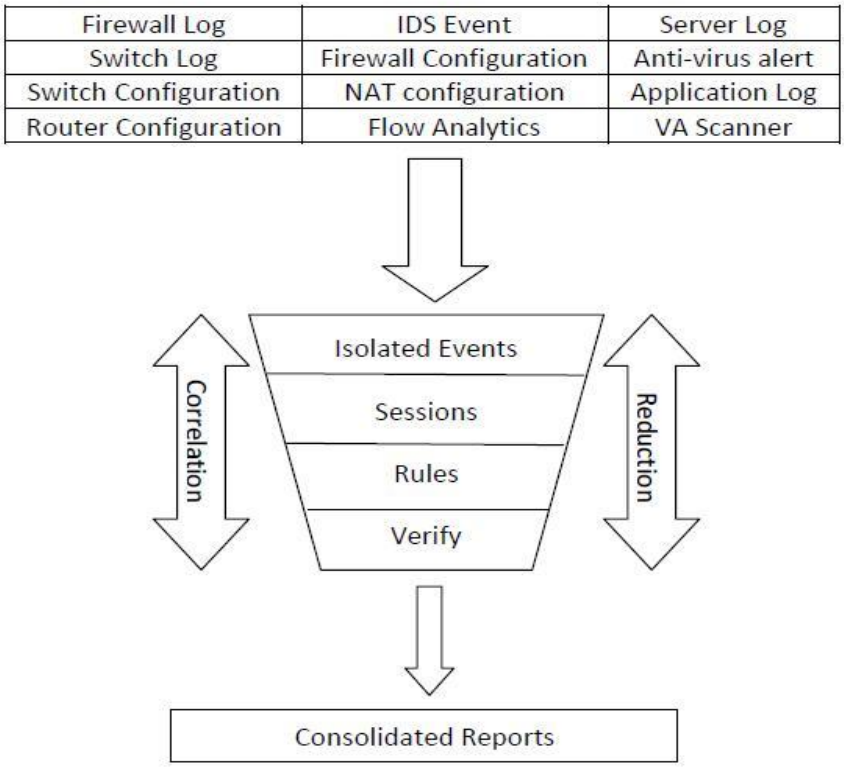


Figure 8.1: Security Information and Event Management Framework [MARS2010]

- Employ a dynamic exposure-time - the exposure window must keep changing with time as the SCIT environment and the system behavior changes.

- Constantly receive input from the SIEM framework on the current SCIT environment and state of behavior to make informed alterations to the exposure window.

Two adaptive SCIT architectures are presented with a common assumption that SCIT is deployed at Enterprise level.

8.3.1.1 Stand-alone Adaptive SCIT

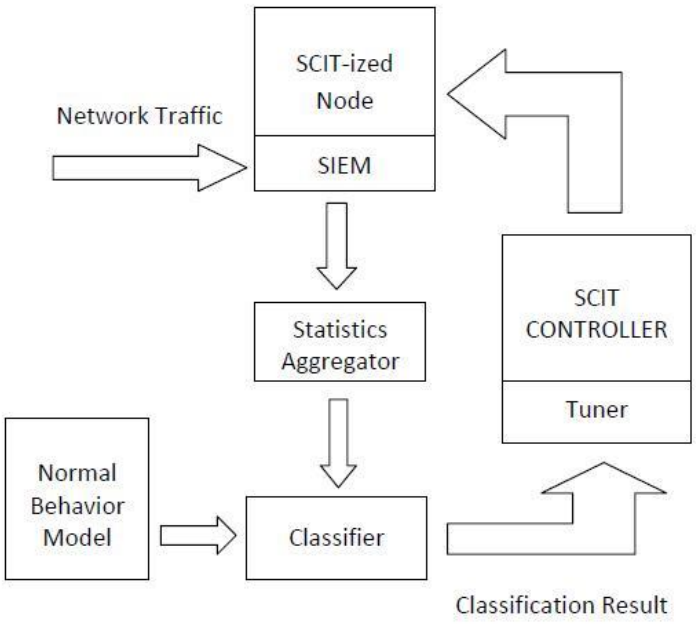


Figure 8.2: Stand-alone adaptive SCIT

In this architecture, SIEM is constantly monitoring the SCIT-ized node and periodically generates consolidated reports based on the information it has gathered and correlated from varying sources. These reports are fed into the Statistics Aggregator which converts massive information obtained from SIEM into meaningful metrics and

their respective values. Further, the classifier compares pre-defined Normal Behavior Model (in terms of metrics and values) with the current values obtained from the Statistics Aggregator. The classifier then feeds the results of the comparison to the Tuner of the SCIT Controller. Based on this, the Tuner makes an informed decision on whether or not to alter the existing "exposure time".

For example, if the results from the classifier identify malicious behavior that points to a Distributed Denial of Service (DDoS) attack, then the SCIT Controller can now reduce the "exposure time" thereby hardening the system against such an attack.

8.3.1.2 Peer-to-peer Collaborative SCIT

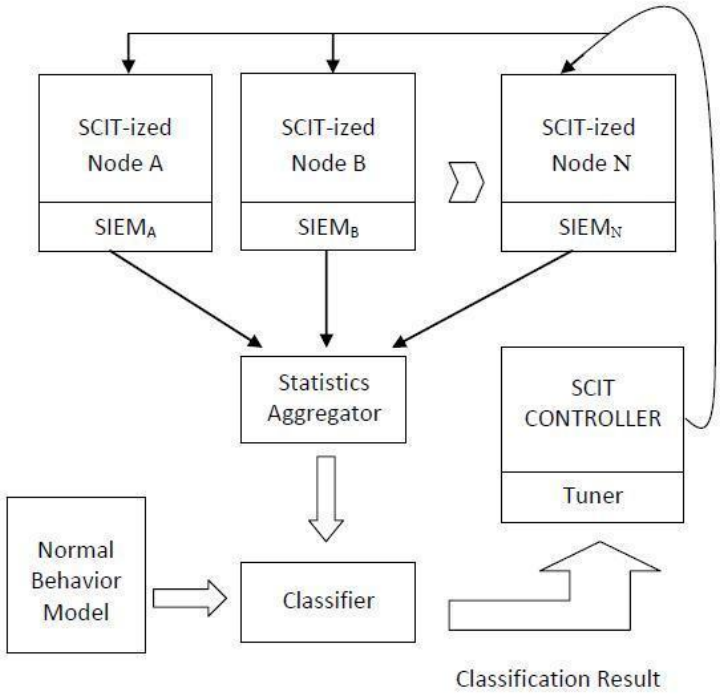


Figure 8.3: Peer-to-peer collaborative SCIT

This architecture is an extension of the stand-alone architecture. It is meant to mimic a cyber-eco-system with multiple participants in the community that offers recovery based resilience. In this case, there are 'N' SCIT-ized nodes that are online concurrently. SIEM solutions of each individual node namely SIEMA, SIEMB so on till SIEMN generate reports individually and keep forwarding them to the Statistics Aggregator periodically. The advantages of collaborative SCIT are straight forward:

- There is more information to work with – the Statistics aggregator is now fed with useful information from 'n' different SIEM solutions.
- Acts as a pre-warning system: malicious behavior in any one of the nodes in the community can now be used to warn / harden rest of the community.
- Unsupervised Learning – malicious behavior in any one node in the community can help teach an attack pattern to the rest of the community.
- Fewer chance of false positives since isolated events now carry less weightage.

8.4 Summary

Cyber-attacks are becoming more wide spread, sophisticated and consequential with time. However, detecting, handling and identifying the consequences of an intrusion are still persistent problems. This is partly due to the lack of trust between the members of the cyber eco system which impedes information sharing and collaboration. If every entity of the cyber eco-system were to collaborate with one another and took co-ordinated security decisions, it could lead to unsupervised learning systems that provide hardened proactive defense.

In this chapter, two such recovery based cyber resilient adaptive SCIT architectures were proposed. One is a stand-alone system and another is a collaborative system that encourages information sharing and promotes cyber health among communities. In addition to the periodic system self-cleansing done proactively, the system constantly part takes in unsupervised learning from other members of the eco-system to adapt to the current environment and system behavior.

CHAPTER NINE – EXPLORING GAME DESIGN FOR CYBER-SECURITY TRAINING

This chapter explores game design for cyber-security training. The objective of this research is to teach everyday users the requisite cyber security skills through gaming, beyond the current state-of-practice. Because the skill level of the trainees is also wide ranging, from causal computer users, to software engineers, to system administrators, to managers, the games must also be capable of training this wide range of computer users.

9.1 Overview

Cyber security awareness and training are vitally important and challenging. A huge number of attacks against everyday users occur routinely. Prevention techniques and responses are wide ranging but are only effectively if used effectively. The objective of this research is to teach everyday users the requisite cyber security skills through gaming, beyond the current state-of-practice. Because the skill level of the trainees is also wide ranging, from causal computer users, to software engineers, to system administrators, to managers, the games must also be capable of training this wide range of computer users. Computer games can provide a media for delivering training in an engaging format at levels appropriate for the individual trainees. In this chapter (1) the state-of-practice is described by describing the gaming tool used in most cyber challenges at high schools and colleges in the US, i.e., the cyber security gaming tool CyberNEXS, (2) some of the additional topics that should be addressed in cyber security training are

outlined and (3) some other approaches to game design that might prove useful for future cyber security training game development beyond CyberNEXS are discussed.

Cyber security training is becoming more and more vital to global security. The large number of network intrusions and malicious attacks that have taken place over the past several years only re-assures the growing need. Some of these events include: massive data breaches of consumer information at Sony and Sony PSN [Sony2011]; Stuxnet worm's stealthy attack on the Iranian nuclear program [Stux2010] and the Chinese electronic break-in at Google [GMai2011].

Intrusions are becoming more and more accepted as a norm. Ever increasing bandwidths, the phenomenon of social networking and the accessibility of mobile devices are part of the reason for this growing cyber-attack problem. Given that cyber security is a real and near threat, it demands comprehensive training in a variety of areas. Games can help here by providing an engaging interface that enhances training, draws more trainees in and simulates a variety of scenarios.

The idea of using games to support health, education, management and other sectors have already yielded positive results [MPre2001] The application of gaming concepts to training can also be equally fruitful. Furthermore, research is advancing in modeling and simulation that seems potentially applicable to cyber security and defense (cyber war) gaming [BotN2008].

9.2 Cyber Security Training

9.2.1 Awareness Topics

The goal is to train the next generation the skills necessary to attain highest achievable level of cyber security and defense against cyber-attacks. Defending against cyber-attacks in near real time is highly stressful. Typically, higher user stress levels lead to more user errors. The game design should put the player in a range of stress levels, thus enabling the user to function more effectively in real life.

Password usage and management – In today's world, passwords protect your computers, data and online accounts. Hackers are becoming increasingly sophisticated at cracking passwords using techniques like brute force attacks, dictionary based attacks and Phishing. It is therefore important to create awareness about making strong passwords the first line of defense. Techniques for creating, using and frequently changing strong passwords can be presented.

Protection from malware and spam – A recent New York Times report [SANS2002] has the Microsoft Internet Safety Enforcement team stating that the "mean time to infection of an unprotected computer on the internet is less than 5 minutes". Viruses, Worms and Trojans are the most common forms of infection and are designed to inflict loss of productivity / economic damage to the target. According to a study conducted by Ferris research, the annual worldwide economic damages from malware exceeded $130 billion in 2009. Therefore, any effective cyber security awareness session must cover the use of anti-virus / anti-malware tools along with training on scanning and updating definitions.

Patch management – Patches are additional pieces of code developed to address problems in software post-release. They enable additional functionality or fix security flaws within the software. These security flaws / vulnerabilities can be exploited if left unpatched at a later time thereby making your system open to compromise. Timely patching of security issues is critical to maintaining service / operational availability, confidentiality and integrity of the system. New patches are released on a daily basis and it often becomes difficult even for experienced system administrators to keep track of all important patches. Training on effective patch management should hence form an important part of the cyber security awareness program.

Social engineering phishing techniques – Phishing based social engineering are attacks on human judgment as opposed to software vulnerabilities and so these attacks pose a threat to unsuspecting users. As more and more users continue to access the internet daily, they become susceptible to Phishing which is a form of electronic deception. Social engineering is evolving so rapidly that security policies alone cannot protect critical infrastructures any more. Even with rigid safeguards, hackers manipulate employees using social engineering phishing techniques into compromising personal, social security and other sensitive information. Hence, it becomes important to develop a security-aware culture that keeps users / employees abreast of latest security threats. This can only be achieved through periodic cyber security training and awareness programs.

Some of the other cyber-security training and awareness topics that need to be presented are:

- Awareness of compliance policy and implications of noncompliance

- How to handle e-mails / attachments from unknown senders and SPAM. Malicious emails coming from recognizable emails are a particular challenge.
- Implementation of new technology
- Awareness on allowed and prohibited web usage – a system to monitor user activity
- Data backup and storage procedures – do's and do not's.
- Responsibility transfer between employees – how to handle?
- Incident response awareness procedures and trigger points - preliminary user steps
- Implications of shoulder surfing
- Use of personal system/ software in work environment
- Education on access control issues – separation of duties, least privilege, privilege escalation etc.
- Individual responsibility and accountability
- Physical access to spaces based on work demands
- Incentive schemes (if any)

9.2.2 Existing Training

A number of techniques exist to get cyber security awareness material disseminated through an agency. The technique chosen depends on resources available and also the type of cyber security message that is being sent out. Some of the most common techniques used are:

- Web-based awareness session – virtual classrooms
- Computer based awareness sessions – computer labs or CD-ROMs

- Teleconferencing sessions
- Instructor-led sessions
- IT security days, Cyber Security awareness week and similar events
- Posters with do's and do not's list
- Screensaver and warning banner / messages
- Periodic Newsletters
- Agency wide e-mail messages / alerts
- 'Brown bag' seminars
- Awards / Incentives program

9.2.3 Shortcomings of the current techniques: [BCon2007, LAnn2010]

- 30 minutes of information about why security is important is not going to change how users behave daily. It should be a continuous process. Most of the awareness programs now happen to be onetime a year events. Users cannot be expected to retain the information from this session and change their daily behavior. Awareness must be a continuous life cycle where users must be trained, updated and reinforced periodically. User's retention capacity must be taken into account;
- Too many topics discussed in too little time – users cannot be expected to understand / retain all of them;
- Training environments are not realistic – different stress levels have an impact on how users act;

- Most awareness programs are presented by security professionals who are bad communicators. Instructor led training headed by security professionals turn out having long information sessions that end up overwhelming people and not getting the intended point across. These sessions cannot afford to be boring; they must be involving and fun;
- If users make the same mistake a number of times even after training and reinforcement – there has to be some sort of disciplinary action. And similarly, there must be incentives for users with good security hygiene;
- One must be able to perform a measurement of user behavior (some sort of score maybe) before and after training to actually see if the training has had a positive impact. Techniques used in current security awareness programs do not facilitate this. They require an additional survey for the same;
- Except for the instructor led session, the rest of them all are passive and do not facilitate interaction with the user. Most of the time the question "Why should I be doing this?" goes un-answered;
- A successful awareness program must be able to do two things – one is to get and retain the user's attention for a span of time and two is to communicate the awareness material to the user effectively in that span of time. Current techniques are found lacking in achieving both.

9.2.4 Interactive Computer-based training

To overcome these shortcomings, the use of Interactive Computer-based training like video games for cyber awareness training is now gaining momentum. Given the current landscape, such games generally fall into two broad classes: [LAnn2010]

- First-person interaction games – Example: first person games where the user is confronted by an adversary / problem and must take a proper course of action else is penalized severely;
- Resource management simulation games – manage a virtual online environment with provided limited resources. Good choices result in a richer environment and additional resources, bad choices result in diminishing resources.

Motivation for the games is either recognition (i.e., if you do well and play fair, you will receive recognition) or certification to enhance your professional career. It is conceivable that cyber games of the future might offer financial or other incentives like prizes for first, second and third places in the competition. In regard to the later, a high quality, valued certification from a game probably does lead to career growth and the corresponding increased salary.

The primary objective of such games is cyber training. Some of the games teach advanced cyber defense concepts and penetration testing in addition. Some such existing games are:

- CyberProtect – Developed by DoD in 1999. It teaches information assurance concepts [LAnn2010];

- CyberCIEGE – Developed by Naval Postgraduate School in 2005. The game employs resource management and simulation to illustrate information assurance concepts for training and education [LAnn2010];
- Multiple micro-games by Wombat security technologies for cyber security awareness and training of US Air force personnel. Eg: Anti-Phishing Phil. Wombat is currently developing a dozen more similar micro games;
- NetWars – NetWars is an offense-oriented cyber security completion that is held completely online and made available to high school students as well. It is an online game where the primary objective is to penetrate into systems, gain access to files and provide proof for the same. It is conducted by the SANS institute and is a player in training and certification of cyber security professionals;
- CyberNEXS – is an example of a simulation game for multiple aspects of cyber security, e.g., activities that revolve around protecting systems from penetration attacks.

A brief look at CyberNEXS follows.

9.2.5 CyberNEXS gaming

CyberNEXS is considered somewhat of a de facto standard in cyber defense competitions – due to its wide spread adoption as the cyber security training and as a game for professional cyber security certification. Thousands of students have used CyberNEXS. It has a client-server architecture that provides game access to anyone with Internet access. One such training exercise is the SAIC High School 12-week Cyber Security e-Learning Pilot which makes use of the CyberNEXS training platform to

educate high school students on advanced cyber defense techniques. These students have gone on to participate in a number of cyber defense competitions over the past four years. Some such notable competitions are the Air Force Association (AFA) Cyber Patriot National High School Cyber Defense Competition, the Maryland Cyber Challenge, the State of Maine High School Competition and the San Diego Mayor's Cyber Cup.

CyberNEXS has five different models of operation, they are:

- On-site training
- Remote training
- Certification
- Competition / Gaming
- Licensing

Gaming is facilitated through the 'competition' model. Here the objective of all 5 gaming modes is to teach cyber defense and penetration testing skills to participants. There are 5 CyberNEXS gaming modes:

CyberNEXS-CND (Computer Network Defense Centralized) – CyberNEXS-CND is a realistic cyber defense exercise in which the participants (blue team) are tasked with defending their network while under attack from the red team. Blue team's primary objective is to ensure availability of their critical services and secure their host throughout the duration of the attack. Blue team also has to detect and mitigate red team's attack and communicate its findings to the administrator (white team).

CyberNEXS-CND Lite – This game mode is similar to CyberNEXS-CND. However, here the objective is only to maintain availability of critical services and secure hosts. There is no need to detect or mitigate the incoming attacks from red team.

CyberNEXS-Forensics – In this game mode, a series of cyber forensic challenges are given to the participants. The objective of the participants here is to find evidence of intrusions, discover malware, analyze payloads, analyze log and audits and trace attacks back to attackers. It is also important for the participants to effectively communicate all of their findings with the white team.

CyberNEXS-CAN (aka Computer Network Attack or Penetration testing) – The objective of this game mode is for the participants to assess a network of computers for vulnerabilities and successfully exploit the vulnerabilities to gain user or administrative control of the system. Participants can use any of the network assessment tools that are at their disposal for this. It is also important to effectively communicate their progress to a "white team", which is basically an observer team.

CyberNEXS-CTF (Capture the flag) – The Capture the Flag mode is similar to the CTF modes found in first person shooter / strategy games. There are two parts to this game. First, the participants have to assess a network of computers for vulnerabilities, exploit them and take over a series of target hosts. Secondly, once the hosts are compromised and are under control, the participants are now required to defend these hosts against other incoming attacks while maintaining availability of their critical services.

Moving forward, it is desired to build on this to develop games that are even more engaging, entertaining, and educational. Some of the things that can be done to improve on the current de facto standards are:

- Motivate participation by creating a broader certification program that could further better employment opportunities;
- Make the game even more scalable and flexible. Present optional game modes where the participant is in full control of the environment, not requiring a white team. This helps participant understand the working of the network.
- In real world, both the attacker and the defender get to make moves all the time. There is no constraint. Attacker adapts to defenders move and vice versa. Similarly, an expert system or a learning engine could potentially help in student-system game, by the system learning and adapting to the user's moves. This requires the development of a front end learning management system.

9.3 Computer Game Design

When designed well, video games can enthrall players, drawing them into a virtual world, motivating them, and challenging them. Research has also shown that games can support and enhance learning and training [BCon2007]. In this section, some important elements of game design are discussed with enhancing cyber security training in mind.

Good game designs focus on the player experience. They create goals that a player feels motivated to reach and rules that must be followed in pursuit of those goals. They are also formulated to match the knowledge and skill level of their target audience

(though it may be a wide range). Furthermore, games designed for education and training must be focused on the training goals. What do you want the player to learn? Do you want them to learn a specific procedure for patching an operating system? Do you want them to learn how to think rationally under stressful conditions? Do you want them to learn the mindset and tools of their combatant? Having a clear understanding of who the player is and what you want them to learn will help you design a game that provides both the player and instructor feedback about the player's progress.

There are several approaches to and decompositions of game design that can help jump start to the design process. Themes can provide a narrative for the game and begin to immerse the players into an alternate world. This immersion can strengthen the training results [BCon2007]. Themes can include a specific story, such as a plumber searching through a Mushroom Kingdom to save a princess (i.e. Super Mario Bros) or a less specific feel, such a dark, dangerous world or a fast paced, cartoon kingdom. When chosen well, themes make the mechanics of a game feel more natural.

In the next couple of sub-sections, other breakdowns of game design, including genres, dynamics, and core mechanics are discussed.

9.3.1 Game Genres

Game genres provide both the designers and players an instant idea about the nature of the game and the type of skills required. It should be noted that games can be a hybrid of multiple genres. Below a number of different game genres and their potential applicability to cyber awareness training are reviewed:

1) Action Games

Action Games keep the player moving and involved at all times providing an adrenaline rush. They often include a lot of hand/eye coordination and quick reflexes. First Person Shooters (FPSs), such as Quake, fall into this genre. Actions in games of this type are not complex and do not require a lot of deliberation. In cyber security, one might imagine a game to train end users to quickly recognize the subject lines of phishing emails.

Applicability Example: In cyber security, one might imagine a game to train end users to quickly recognize the subject lines of phishing emails.

2) Role Playing Games (RPGs)

RPGs generally have more developed stories and are played for longer spans of time in more expansive worlds. These games also tend to focus on character growth. As the game progresses characters obtain more experience, capabilities, and weapons. The outcome of actions in this genre can include an element of chance. Even if the player performs an action perfectly, it could still fail. Final Fantasy is an example of a game from this genre. A game for cyber security training could easily involve the player taking on the role of a system administrator to defend a group of servers that are critical to the future of the country or even a hacker that needs to break into a series of systems to obtain the information needed to save a hostage. As the player's knowledge and skills increase, they would be given more sophisticated tools and also bigger challenges to further develop their abilities.

Applicability Example: A game for cyber security training could involve the player taking on the role of a system administrator to defend a group of servers that are

critical to the future of the country or even a hacker that needs to break into a series of systems to obtain the information needed to save a hostage. As the player's knowledge and skills increase, they would be given more sophisticated tools and also bigger challenges to further develop their abilities.

3) Adventure Games

Adventure games are somewhat similar to RPGs, in that, they also focus on story, but generally adventure games also include more exploration and a number of puzzles. Myst, for example, involved exploring the world, encountering puzzles, and attempting to solve the puzzles so that additional areas could be explored. Along the way, the player pieces together the story of what has taken place in this world. This genre of games might fit quite nicely with training recovery operations after an intrusion.

Applicability Example: Possibly to training recovery operations after an intrusion.

4) Strategy Games

In strategy games, the key is balance. There are at least two opposing teams each with an equal chance of winning. There may be different units, weapons, resources, and goods available to the opponents, but they must be balanced. In strategy games, there is not a single right way to do things. Multiple strategies can be successfully enacted. There are normally also a series of different missions that lead a final completion. Command and Conquer requires players to construct bases, acquire resources, and attempt to conquer opponent bases.

Applicability Example: It is easy to see how this paradigm could be used in cyber security training. Players might use different strategies and priorities in defending

penetration attacks. If trying to train administrators through better knowledge of a hacker's mind set, players might take on the role of hacker and use different strategies to try to breach a system.

5) Sports Games

The genre of sports games might seem irrelevant to cyber security, but in fact there are possible parallels. Many sports games involve deciding on formations and calling plays. We could imagine training managers to handle security attacks in a similar fashion. What skills should his team have (or what can he afford)? What should each member of the team be doing as an attack progresses? The members of the team could be Non- Player Characters (NPCs) or real players in an asynchronous game.

Applicability Example: Students could play war games that are time constrained. This adds a dimension of stress and necessity for effective time / resource management.

6) Fighting Games

Fighting games are simple and direct, but engaging. In fighting games, the action is swift and intense and the moves are usually easy to learn. Tekken and Mortal Kombat are examples of fighting games. Opponents battling to deface and restore a website might fit in this genre.

Applicability Example: Students could play war games to compete for points for defensive blocking, and offensive cyber-attacks. A student could play cyber war against the computer or another student, or teams could play each other.

7) Casual Games

Casual games tend to be easy to learn and not difficult to master. They include video game versions of card games and board games, as well as games created just for computers, such as Tetris. Generally, a player starts a new game each play session as opposed to continuing a mission from their last session. Any number of casual games could be designed to help familiarize people with cyber security terminology and train them on more rudimentary techniques such as creating secure passwords.

Applicability Example: Any number of casual games could be designed to help familiarize people with cyber security terminology and train them on more rudimentary techniques such as creating secure passwords.

8) Sandbox Games

Finally, in sandbox or God games, there is no preset win condition. A player is provided a variety of building blocks and constructs their virtual life or virtual environment. The game system causes different events to occur that affect (positively or negatively) the player's world. For example, in The Sims, a player's kitchen might catch on fire or they might be promoted. In terms of cyber security, the player might setup a system, be it a single computer or an entire network, with various precautions and then the game could prompt changes to the system based on hardware failures, attacks, consumer complaints, etc.

Applicability Example: In terms of cyber security, the player might setup a system, be it a single computer or an entire network, with various precautions and then the game could prompt changes to the system based on hardware failures, attacks, consumer complaints, etc.

9) Simulations

Simulations normally focus on one piece of equipment or activity. The resulting experience can be true to life or exaggerated. For example, many racing game allow the players to maneuver the vehicles around the course at speeds that would not normally be possible. CyberNEXS is an example of a simulation game for cyber security (http://www.saic.com/cybernexs/), where the activity revolves around protecting systems from penetration attacks.

Applicability Examples: Both CyberNEXS and Netwars (http://www.sans.org/cyber-ranges/netwars/) are simulation games for cyber security where the activity revolves around protecting systems from penetration attacks.

9.3.2 Game Dynamics

Game dynamics are a particular pattern of play within a game and are tied to core mechanics, which will be discussed in the next section. They focus the type of actions a player can take.

1) Territorial Acquisition

Territorial acquisition revolves around a limited resource that may or may not be a land mass. The main focus of the game is to acquire as much of the limited resource(s) as possible and strategically control it. Risk and some FPSs have the territorial acquisition dynamic. In cyber security, the limited resource might be memory, network bandwidth, or entire servers.

2) Prediction

The prediction dynamic is simply prompting the player to guess what will happen and rewarding them if they guess correctly. Roulette and Rock-Paper-Scissors are examples of prediction games. A training game is unlikely to focus solely on the prediction dynamic, but it does still have a place. For example, guessing the nature the next attack to try to defend against it.

3) Spatial Reasoning

Spatial reasoning often involves puzzles (e.g. Tetris, Tic-Tac-Toe, and Connect Four). A cyber-security game might include the notion of lining up security elements to form a continuous shield from attacks and strategizing about where the next attack might come from.

4) Survival

The survival dynamic taps into the instinctual need for self-preservation. There is a constant life and death struggle that is the focus of the game. Here we could imagine a player becoming a server or router and struggling to survive against constant attacks.

5) Destruction

Every FPS includes the destruction dynamic. With this dynamic, the goal is basically to wreck everything in sight. Consider a game set in a computer, where the player uses different weapons (i.e. security techniques) to destroy various attacks he encounters.

6) Building

Because of their focus on character development, RPGs often have a building dynamic. The main objective of these games is to build a better character or in the case of

the sandbox game SimCity, a better world. How about a better network or computer system?

7) Collection

The collection dynamic can be found in card games and many platformers (e.g. collect rings, bolts, gold coins, etc.). In these games, getting the most of a resource is what determines the winner. A cyber-security parallel might involve collecting passwords or other user data.

8) Chasing and Evading

In chasing and evading games, the goal is to capture prey or escape predators. Pac-Man is a good example. In cybersecurity, a hacker might be attempting to gain control of a system or data while evading detection.

9) Trading

Trading requires cooperating with others. There are normally multiple kinds of resources that can be exchanged between players. This is common in card games. We could imagine a game where tokens corresponding to security software and techniques are traded. When someone has a full set, their system is secure and they have won the game. This would increase the trainees' awareness of cyber-security.

10) Race to the End

The race to the end dynamic has the player or players focusing on getting to a certain location first or learning a technology first. The applications to cyber security training are straightforward.

9.3.3 Game Mechanics

Game mechanics are essentially the rules of a game. They describe how the game state changes. For example, in Monopoly, if you land on an un-owned property, then you can buy it.

There are a few common classes of mechanics. The setup mechanic is at least one rule describing how the game begins. Victory conditions describe how the game is won. Not all games have victory conditions. For example, RPGs tend to have smaller goals along the way, but no explicit victory condition. Progression of play mechanics include a description of whether it is a turned based or real time game, who goes first and how, and how conflicting, simultaneous actions get resolved. Naturally, player actions are also a common class of mechanics. What actions can a player perform and how? What affect do player actions have on the game state? The final class of mechanics is a definition of game views. This is a description of exactly what information each player knows about at any given time. Some mechanics might change this view as the game progresses (e.g. lifting the fog of war).

Like game dynamics, these mechanics can help focus a game design and ensure that it is consistent and coherent.

9.3.4 Learning and Training Games

In educational games, the goal is to teach a body of knowledge. Before beginning the game design process, there should be a clear outline of exactly what the player should learn from playing the game. The game itself should motivate and reward the player to keep them playing the game and as a consequence acquiring more information or skill.

According to Annetta, there are six principles to follow when designing games for education [BCon2007]. Players should have a unique identity in the game world. This promotes them getting more emotionally involved in the game and caring about the consequences, which leads to immersion. Immersion is a heightened sense of presence that leads to the player being more engaged in the content and motivated to succeed. Interactivity further involves players in the game world by allowing them to interact with other players or NPCs. Increased Complexity keeps players challenged. Game levels can provide a platform for increasing the complexity of content and concepts, keeping players from getting bored. Informed Teaching focuses on providing feedback to the instructors. These games should track players' performances and record timings, actions, and mistakes and provide feedback to both the instructors and the players. Finally, educational games should be instructional. Players should be able to assimilate the knowledge and skills they are acquiring in the game with their existing knowledge and experiences.

9.4 Summary

Although many of the topics presented as part of the cyber security awareness program are universal, such training must always be tailored to address the needs and security policies of a particular organization. A major shortcoming of most of the current forms of cyber awareness training is that they don't require participants to think on their feet and apply security concepts in real time. And although theoretical knowledge of security concepts is important, handling a security event in a stressful environment

demands prior hands-on experience. A flexible, scalable and highly interactive video game could help simulate a similar environment for the trainees.

CHAPTER TEN - CONCLUSIONS

10.1 Summary

As part of this research, a framework to assess the performance of security architectures in terms of reducing data ex-filtration was defined. Multiple hybrid approaches were proposed that combine recovery-driven intrusion tolerant SCIT architecture with existing IDS solutions as part of a multi layered defense strategy to protecting the cyber infrastructure.

A framework was established that uses Receiver Operating Characteristic (ROC) curve analysis and damage cost models to trade-off the true positive rate and false positive rate for comparing alternate security architectures. This framework provides a baseline for making informed decisions and choosing operating parameters for various architectures.

As part of my research, I also proposed the use of Attack Surface Shifting / Reduction as a metric to compare Moving Target Defenses (MTD) by assessing its impact on intruder work factors.

A game theoretic attack / protect cyber economic model was developed to facilitate designing architectures that are resilient and tilt the asymmetric cyber economic costs in favor of the defender. This research formalizes system security state transitions and intruder / defender work factors associated with all of those state transitions.

I also proposed µ-SCIT, a hybrid architecture that adds modularity to SCIT using Operating System level virtualization. The added modularity affords the ability to perform more frequent targeted granular rotations at the level of processes and applications. This in turn extends ability of SCIT to work with long running applications and handle long transactions using container check-pointing and migration.

Finally, in order to perform adaptive intrusion tolerance that constantly learns from its ecosystem, I conceptualize and present architectures for a ‘stand-alone’ and a ‘collaborative’ architecture which make use of information provided by the enterprise Security Information and Event Management (SIEM) solution.

REFERENCES

[AAde2002] Adelsbach, A., Alessandri, D., Cachin, C, Creese, S., Deswarte, Y., Kursawe, K., Laprie, J.C., Powell, D., Randell, B., Riordan, J., Ryan, P., Simmonds, W., Stroud, R., Verissimo, P., Waidner, M., Wespi, A.: Conceptual Model and Architecture of MAFTIA. Project MAPTIA IST-1999-11583 deliverable D21. (2002)

[AAvi1986] Avizienis, A., Laprie, J.C., Randell, B.: Fundamental concepts of dependability. Technical Report 01145, LAAS-CNRS, Toulouse, Prance (2001)

[ABan2009] Anantha K. Bangalore and Arun K Sood. "Securing Web Servers Using Self Cleansing Intrusion Tolerance (SCIT)", DEPEND 2009, Athens, Greece. 2009.

[AFranz2002] A. Franz, R. Mista, D. Bakken, C. Dyreson, and M. Medidi, "Mr. fusion: A programmable data fusion middleware subsystem with a tunable statistical profiling service," in Proceedings of the InternationalConference on Dependable Systems and Networks (DSN-2002), pp. 273– 278, 2002.

[ANag2010] Ajay Nagarajan and Arun Sood, "SCIT and IDS Architectures for Reduced Data Ex-filtration" 4th Workshop on Recent Advances in Intrusion-Tolerant Systems, Chicago, USA, June 2010.

[ANag2011] Ajay Nagarajan, Quyen Nguyen, Robert Banks and Arun Sood "Combining Intrusion Detection and Recovery for Enhancing System Dependability" 5th Workshop on Recent Advances in Intrusion-Tolerant Systems, Hong Kong, China, June 2011

[ANag2012] Ajay Nagarajan, Jan Allbeck, Arun Sood and Terry Janssen "Exploring Game Design for Cyber Security Training" IEEE Cyber 2012, Bangkok, Thailand, May 2012

[ANag2012a] Ajay Nagarajan and Arun Sood "Recovery-based Resilient Cyber Ecosystem" – appeared in Crosstalk Magazine (The Journal of Defense Software Engineering) Sept/Oct issue 2012

[ANag2013] Ajay Nagarajan and Arun Sood "Measuring Work Factor in a Moving Target Host Architecture: The SCIT Case" MIT Think-shop on Multi-spectrum metrics for Cyber Defense, MIT CSAIL, Cambridge, MA, Oct' 2013

[ANag2014] Ajay Nagarajan and Arun Sood "SCIT Based Moving Target Defense Reduces and Shifts Attack Surface" 11th International Workshop on Security in Information Systems, Lisbon Apr' 2014

[AOrf2006] Orfila, Augustin. Carbo, Javier. And Ribagardo, Arturo. "Advances in Data Mining, volume 4065, chapter Effectiveness Evaluation of Data Mining based IDS, pages 377-388. Springer Berlin Heidelberg. 2006.

[ASai2009] A. Saidane et al "The Design of a Generic Intrusion-Tolerant Architecture for Web Servers" Dependable and Secure Computing, IEEE Transactions Jan-March 2009

[ASha1979] Adi Shamir "How to share a secret?", Communications of the ACM, Volume 22, Issue 11, Nov 1979.

[AVal2002] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saidi, V. Stavridou, and T. E. Uribe, "An architecture for an adaptive intrusion tolerant server," Springer-Verlag, 2002.

[AVal2003] Alfonso Valdes et al. "An Architecture for an Adaptive Intrusion-Tolerant Server". LNCS Springer/Berlin, Volume 2845/2003, pp. 569-574.

[BCon2007] Benjamin D. Cone et al "A video game for cyber security training and awareness" Computers and Security 26 (2007)

[BLit1993] Littlewood, Bev et al. "Towards Operational Measures of Computer Security" Journal of Computer Security, pp. 211-229, 1993.

[BotN2008] "A robot network seeks to enlist your computer" – NY Times,10/20/2008

[Bran1991] B. Randell and J.C. Fabre "Fault and intrusion tolerance in object-oriented systems" International workshop on Object Orientation in Operating Systems, 1991.

[CCoy2005] Coyne, Christopher J. and Leeson, Peter T., "Who's to Protect Cyberspace?" Journal of Law, Economics and Policy, 2005.

[CDru2004] Drummond, Chris. Holte, Robert C. "What ROC Curves can't do and Cost curves can". 2004.

[CRus2002] Russell, C. Security Awareness--Implementing an Effective Strategy, SANS Institute InfoSec Reading Room, 2002.

[DArs2007] David Arsenault, Arun Sood, and Yih Huang, "Secure, Resilient Computing Clusters: Self-Cleansing Intrusion Tolerance with Hardware Enforced Security (SCIT/HES)" Proceedings Second International Conference on Availability, Reliability and Security (ARES 2007), Vienna, Austria, April 2007.

[DBri2003] D. O'Brien et al "Intrusion tolerance via network layer controls" Proceedings of the DARPA Information survivability conference and exposition, 2003.

[DEva2011] David Evans, Anh Nguyen-Tuong, John Knight "Effectiveness of Moving Target Defenses" Chapter 2, Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats 2011

[DKra2010] Kravets, David. "McAfee Probing Bungle That Sparked Global PC Crash".Threat Level. http://www.wired.com/threatlevel/2010/04/mcafeebungle/. 2010.

[DMal2001] Malkhi, D., Reiter, M.K., Tulone, D., Ziskind, E.: Persistent objects in the Fleet system. In: Proceedings of the 2nd DARPA Information Survivability Conference and Exposition (DISCEX II). (2001)

[DoDi2003] DoDi 8500.2 Information Assurance Implementation http://www.dtic.mil/whs/directives/corres/pdf/850002p.pdf

[DPow2001] Powell, D. et al. "MAFTIA (Malicious- and Accidental-Fault Tolerance for Internet Applications)" Proceedings of the 2001 International Conference on Dependable Systems and Networks (DSN2001), Goteborg (Sweden), 1-4 July 2001

[FAnj2000] F. Anjum "Intrusion tolerance schemes to facilitate mobile e-commerce" IEEE international conference on Personal Wireless Communications 2000.

[FSto2000] Stolfo, S. Fan, W. Lee, W. Prodromidis, A. and Chan, P. "Cost-based modeling for Fraud and Intrusion Detection: Results from the JAM Project" Proceedings of DISCEX 2000, Los Alamitos, CA. 2000.

[FWan2003] Feiyi Wang et al "SITAR: A scalable intrusion tolerant architecture for distributed services" Proceedings of DARPA Information survivability conference and exposition, 2003

[GAke1995] Akerlof, George, “The market for “lemons”: Quality uncertainty and the market mechanism, Springer 1995.

[Gate2000] Ateniese, G., Steiner, M., Tsudik, C : New multi-party authentication services and key agreement protocols. IEEE J. of Selected Areas on Communications 18 (2000)

[GMai2011] "Google says hackers in China stole Gmail passwords"-NYTimes 06/01/2011

[GSch2000] Schudel, Gregg and Wood, Bradley J. “Adversary work factor as a metric for information assurance” Proceedings of the 2000 workshop on new security paradigms, NY, USA 2000.

[HAcq2002] Hierarchical Adaptive Control for QoS Intrusion Tolerance, a DARPA funded UC Davis project, 2002

[Home2014] Home Depot Security Breach, 2014 http://money.cnn.com/2014/09/08/technology/security/home-depot-breach/

[HRam2004] H.V. Ramasamy “CoBFIT: A component-based framework for intrusion tolerance”, Euromicro Conference, 2004

[HRei2007] Hans P. Reiser, Rudiger Kapitza “VM-FIT: Supporting Intrusion Tolerance with Virtualization Technology” Proceedings of the 1st Workshop on Recent Advances on Intrusion-tolerant Systems, 2007. Lisbon, Portugal.

[HRei2011] Hans P. Reiser “Byzantine Fault Tolerance for the Cloud” Workshop on Cryptography and Security in Clouds, Zurich, 2011.

[JDob1986] Dobson, J., Randell, B.: Building rehable secure computing systems out of unreliable insecure components. In: Proceedings of the International Symposium on Security and Privacy, IEEE (1986) 187-193

[JFrag1985] Fraga, J.S., Powell, D.: A fault- and intrusion-tolerant file system. In: Proceedings of the 3rd International Conference on Computer Security. (1985) 203-218

[JGaf2001] Gaffney, John E. Jr. Ulvila, Jacob W. (2001). “Evaluation of Intrusion Detectors: A Decision Theory Approach” Security and Privacy.

[JHan1966] J. Hancock and P. Wintz. Signal Detection Theory. McGraw-Hill. New York 1966

[JHir1983] Hirshleifer, Jack, From weakest-link to best-shot: The voluntary provision of public goods, Public Choice January 1983, Volume 41, Issue 3, pp 371-386.

[JJus2003] James E. Just et al "Learning Unknown Attacks – A start" Foundations of Intrusion Tolerant Systems, pp 374-386, 2003

[JKni2002] J. Knight, D. Heimbigner. and A. Wolf. "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications", lntrusion Tolerance System Workshop, Supplemental Volume on 2002 International Conference on Dependable .System and Network, 2002.

[JLal2003] Lala, J. H., Editor, "Organically Assured & Survivable Information Systems (OASIS): Foundations of Intrusion Tolerant Systems," IEEE Computer Society Press, http://computer.org/cspres, ISBN 0-7695-2057-X2003, 2003.

[JLep2003] J. Lepanto and W. Weinstein, "Contra: Camouflage of Network Traffic to Resist Attacks."

[JLow2004] Lowry, John et al. "Adversary Modeling to Observer Forensic Observables" Digital Forensic Research Workshop 2004.

[JMcH2000] McHugh, John (2000) "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory" TISSEC, Vol 3, Issue 4

[JRey2003] James C. Reynolds et al. "On-Line Intrusion Detection and Attack Prevention Using Diversity, Generate-and-Test, and Generalization". Proceedings of the 36th Hawaii International Conference on System Sciences, 2003.

[JSwe1996] Swets, John A. "Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers", 1996

[JUlv2003] Jacob W Ulvila, John E Gaffney Jr "Evaluation of Intrusion Detection Systems", Journal of Research of the National Institute of Standards and Technology 2003

[KAun2005] Khin Mi Mi Aung, Kiejin Park and Jong Sou Park. "A Rejuvenation Methodology of Cluster Recovery". CCGrid 2005, IEEE International Symposium Vol. 1, pp. 90 - 95, May 2005.

[KKih2001] Kihlstrom, K.P., Moser, L.E., Melliar-Smith, P.M.: The SecureRing group communication system. ACM Transactions on Information and System Security 4 (2001) 371-406

[KKih2003] K.P. Kihlstrom, P. Narasimhan “The Starfish System: providing intrusion detection and intrusion tolerance for middleware systems” Proceedings of the Eight international Workshop on Object-Oriented Real-Time Dependable Systems, 2003

[LAlv2000] Alvisi, L., Malkhi, D., Pierce, E., Reiter, M.K., Wright, R.N.: Dynamic Byzantine quorum systems. In: Proceedings of the IEEE International Conference on Dependable Systems and Networks. (2000) 283-292

[LAnn2010] Annetta, L.A. The "I's" Have It: A Framework for Serious Educational Game Design. Review of General Psychology, 14 (2).105-112.

[LGor2002] Gordon, Lawrence and Loeb, Martin, “The Economics of Information Security Investment”, University of Maryland, ACM Transactions on Information and Systems Security, November 2002.

[LZho2002] Zhou, L et al “COCA: A secure distributed online certification authority” ACM Transaction on Computer Systems 20 (2002)

[Mand2013] Mandiant “APT1: Exposing one of China’s Cyber Espionage Units” report 2013

[Mars2010] CISCO Security Monitoring, Analysis and Response System (MARS) Framework

[McAf2010] McAfee Labs. “McAfee Threats Report: Second Quarter 2010”. http://www.mcafee.com/us/local_content/reports/q22010_threats_report_en.pdf. pg 11.

[McAf2013] McAfee “Infographic: The State of Malware 2013”

[MCas1999] Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating systems Design and Implementation. (1999)

[MCor2007] Miguel Correia et al “Worm-IT – A wormhole-based intrusion-tolerant group communication system”, Journal of Systems and Software, Volume 80, Issue 2, 2007

[MCuk2001] Michael Cukier, Partha Pal et al. “Intrusion Tolerance Approaches in ITUA” – joint project of BBN technologies and Boeing.

[MHei2012] Heilman, Marshall, “South Carolina Department of Revenue” Mandiant, Public Incident Response Report, November 20, 2012.

[MHil2001] Hiltunen, M., Schlichting, R., Ugarte, C.A.: Enhancing survivability of security services using redundancy. In: Proceedings of the IEEE International Conference on Dependable Systems and Networks. (2001) 173-182

[MPre2001] Prenski M. "Digital game-based learning" New York: McGraw-Hill; 2001.

[MRei1995] Reiter, M.K.: The Rampart toolkit for building high-integrity services. In: Theory and Practice in Distributed Systems. Volume 938 of Lecture Notes in Computer Science. Springer-Verlag (1995) 99-110

[MShi2011] Marn-Ling Shing, Kuo Lane Chen, Chen-Chi Shing and Huei Lee, “A game theory approach in information security risk study” 2010 International conference on e-business, Management and Economics (IPEDR) vol.3, 2011.

[MSli2008] M. Sliti, M. Hamdi, N. Boudriga, A. Helmy, "An Elliptic Threshold Signature Framework for k-Security in Wireless Sensor Networks," The 15th IEEE International Conference on Electronics, Circuits, and Systems, 2008

[MSli2009] M. Sliti et al “Intrusion-tolerant framework for heterogeneous wireless sensor networks” IEEE/ACS International Conference on Computer Systems and Applications, 2009

[MSSI2009] Microsoft Security Intelligence Report Volume 8, 2009

[MWil2003] Wilson, M. and Hash, J. Building an Information Technology Security Awareness and Training Program, NIST, 2003.

[MYou1989] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[NetS2009] Network Solutions Security Breach, 2009
http://www.careandprotect.com/

[OASI2000] Organically assured and survivable information systems (OASIS). http://www.tolerantsystems.org

[OKre2010] O.P. Kreidl “Analysis of a Markov decision process model for intrusion tolerance” International Conference on Dependable Systems and Networks Workshops (DSN-W) 2010.

[OPMB2015] Office of Personnel Management Breach, 2015
https://www.opm.gov/cybersecurity/cybersecurity-incidents/

[PAde2012] Adey P and Anderson B (2012) Anticipating emergencies: Technologies of preparedness and the matter of security. Security Dialogue 43(2): 99–117.

[PCID2014] PCI DSS Compliance Standards (Requirement 11.4)
https://www.pcisecuritystandards.org/pdfs/pci_audit_procedures_v1-1.pdf

[PFCh2014] PF Chang's Security Breach 2014
http://www.zdnet.com/article/pf-changs-security-breach-data-stolen-from-33-locations-over-8-months/

[Play2011] "Security Experts: Playstation Network breach one of largest ever" USA Today, 04/27/2011

[PLor1989] P. R. Lorczak, A. K. Caglayan, and D. E. Eckhardt, "A Theoretical Investigation of Generalized Voters for Redundant Systems," in The Nineteenth International Symposium on Fault-Tolerant Computing, pp. 444 – 451, 1989.

[PLue2003] Luenam P. and Peng Liu "The design of an adaptive intrusion tolerant database system" Foundations of Intrusion Tolerant Systems, 2003

[PLui2001] Peng Liu and Sushil Jajodia "Multi-Phase Damage Confinement in Database Systems for Intrusion Tolerance", Proceedings of the 14th IEEE workshop on Computer Security Foundations, 2001

[PMan2008] Manadhata, P.K. "An attack surface metric" Ph.D. thesis, Carnegie Mellon University (2008)

[PMan2013] Manadhata, P.K. "Game Theoretic Approaches to Attack Surface Shifting", MTD II

[PNeu1997] Peter Neumann & Phillip Poras "Event Monitoring Enabling Responses to Anomalous Live Disturbances", 1997 National Information Systems Security Conference

[Pone2009] 2009 Annual Study: Cost of a Data Breach, Ponemon Institute LLC.

[Pone2015] Ponemon Institute, "2015 cost of data breach study: impact of business continuity management", Ponemon Instituted, June 2015.

[PPal2007] Partha Pal, Franklin Webber, and Richeard Schantz. “The DPASA Survivable JBI – A High-Water Mark in Intrusion-Tolerant Systems”, Workshop on Recent Advances in Intrusion Tolerant Systems’07, 2007.

[PSou2007] Paulo Sousa et al. “Resilient Intrusion Tolerance through Proactive and Reactive Recovery”. 13th IEEE International Symposium on Pacific Rim Dependable Computing, 2007.

[PSou2008] Paulo Sousa et al. “The FOREVER Service for Fault/Intrusion Removal”. WRAITS 2008, Glasgow, Scotland.

[PSou2009] P. Sousa et al “Intrusion-tolerant self-healing devices for critical infrastructure protection” IEEE / IFIP International Conference on Dependable Systems and Networks, 2009

[PSou2010] P. Sousa et al “Highly Available Intrusion-Tolerant Services with Proactive-Reactive Recovery” IEEE Transactions on Parallel and Distributed Systems 2010

[PVer2006] P. Verissimo, “Travelling through Wormholes: A New Look atDistributed Systems Models” Special Interest Group on Algorithms and Computation Theory News, vol. 37, no. 1, 2006

[QNgu2009] Quyen Nguyen and Arun Sood, “Quantitative Approach to Tuning of a Time-Based Intrusion-Tolerant System Architecture”, 3rd Workshop on Recent Advances in Intrusion Tolerant Systems, Portugal, June 29, 2009.

[QNgu2010] Nguyen, Quyen and Sood, Arun. “Comparative Analysis of Intrusion-Tolerant System Architectures”. IEEE Security and Privacy – Volume: PP, Issue: 99 , 2010.

[QNgu2011] Quyen L. Nguyen and Arun Sood, "Comparative Analysis of Intrusion-Tolerant System Architectures", IEEE Security and Privacy, Volume 9 Issue 4, July-Aug 2011

[RBej2005] Bejtlich, Richard. “The Tao of network security monitoring: beyond intrusion detection”, Pearson Education, Inc. 2005.

[RKap2009] Rudiger Kapitza, Tobias Distler and Hans P. Reiser “Practical Intrusion-tolerance in the Cloud”

[RKap2010] Rudiger Kapitza et al “Storyboard: Deterministic Multithreading” Proceedings of the 6th workshop on Hop Topics in System Dependability, Vancouver, Canada, 2010.

[RLip2000] R. Lippmann, et al "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation" Proceedings of DISCEX 2000, Los Alamitos, CA. 2000.

[RMax2004] R.A. Maxion and R.R. Roberts. "Proper use of ROC curves in Intrusion/ Anomaly Detection" Technical Report, University of Newcastle Nov 2004

[RRam2000] Ranga Ramanujam "Randomized Failover Intrusion Tolerant Systems (RFITS)" Architecture Technology Corporation

[RSha2009] R.Shashikumar and L.C.S. Gouda "Self-Healing Reconfigurable FPGA Based Fault Tolerant Security Model for Shared Internet Resources" IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.1, January 2009

[RUpp2002] R. Uppalli, R. Wang, and F. Wang, "Design of a ballot monitor for an intrusion tolerant system," in Supplemental Volume of the International Conference on Dependable Systems and Networks (DSN-2002), pp. B60–B61, 2002.

[RZbi2004] Rabih Zbib et al. "Intrusion Tolerance in Distributed Middleware".Information Systems Frontiers 6:1, 2004

[SANS2002] Security Awareness – Implementing an effective strategy, SANS Institute InfoSec reading room, 2002

[Siem2012] Security Information and Event Management – Wikipedia article

[Sony2011] "Sony playstation suffers massive data breach" - Reuters - 04/26/2011

[Sony2014] Sony Security Breach 2014
https://www.riskbasedsecurity.com/2014/12/a-breakdown-and-analysis-of-the-december-2014-sony-hack/

[Stux2010] "Stuxnet 'hit' Iran Nuclear Plans" - BBC - 11/22/2010

[SWei2005] Weidong Shi, Hsien-Hsin S. Lee, Guofei Gu, Laura Falk, Trevor N. Mudge, Mrinmoy Ghosh "An Intrusion-Tolerant and Self-Recoverable Network Service System Using A Security Enhanced Chip Multiprocessor" ICAC '05 Proceedings of the Second International Conference on Automatic Computing, Seattle, WA, USA

[SWid2010] Widup, Suzanne. (2010, Jul). “The Leaking Vault – Five years of data breaches” – Digital Forensics Association.

[Syma2016] Symantec Internet Security Threat Report 2016

[TDis2010] Tobias Distler, Rudiger Kapitza et al “State transfer for Hypervisor-based proactive recovery of Heterogeneous Replicated Services” Proceedings of the 5th “Sicherheit, Schutz und Zuverlassigkeit” Conference, Berlin, 2010.

[TDis2011] Tobias Distler, Rudiger Kapitza “Increasing performance in Byzantine Fault Tolerant systems with On-Demand Replica Consistency” Proceedings of the EuroSys 2011 conference, Salzburg, 2011

[TDis2011a] Tobias Distler, Rudiger Kapitza et al “SPARE: Replicas on Hold” 18th Network and Distributed System Security Symposium, San Diego, USA, 2011

[TZha2005] T. Zhang, X. Zhuang, S. Pande “Building intrusion-tolerant secure software” International Symposium on Code Generation and Optimization, 2005

[UNIS2012] UNISDR (United Nations International Strategy for Disaster Reduction) How to Make Cities More Resilient: A Handbook for Local Government. Geneva, Switzerland: UNISDR, 2012.

[Veri2009] Verizon 2009 Data Breach Investigations Report http://www.verizonbusiness.com/resources/security/reports/2009_databreach_rp.pdf

[Veri2010] Verizon Business Data Breach Investigations Report 2010

[Veri2012] Verizon Business Data Breach Investigation Report 2012

[Veri2013] Verizon Business Data Breach Investigation Report 2013

[Veri2015] Verizon Business Data Breach Investigations Report 2015

[Veri2016] Verizon Data Breach Investigations Report 2016

[VSig2012] “Key Internet Operators VeriSign hit by hackers” Reuters 02/02/2012

[Wynd2010] Hotchkiss, Kirsten “Wyndham Hotels Worldwide Breach”. http://www.wyndhamworldwide.com/customer_care/data-claim.cfm. Jun. 2010

[YAwa2012] Awad A. Younis and Yashwant K. Malaiya “Relationship between Attack Surface and Vulnerability Density: A Case Study on Apache HTTP Server”, The 2012 International Conference on Internet Computing, Las Vegas, USA, July 2012

[YDes1991] Deswarte, Y., Blain, L., Fabre, J.C.: Intrusion tolerance in distributed computing systems. In: Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy. (1991) 110-121

[YHua2006] Yih Huang, David Arsenault, and Arun Sood. “Incorruptible System Self-Cleansing for Intrusion Tolerance”. Performance, Computing, and Communications Conference, IPCCC 2006.

[YKim2007] Young-Soo Kim et al “Intrusion Tolerance model for Electronic Commerce System” Future Generation Communication and Networking (FGCN), 2007

[YTan2010] Yuesheng Tan, Dengliang Luo, Jingyu Wang “CC-VIT: Virtualization Intrusion Tolerance Based on Cloud Computing” Second International Conference on Information Engineering and Computer Science 2010

[YYor1987] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[ZCui2009] Zhilei Cui, Xi Lu, Jine Wang “Adaptive Intrusion Tolerance Strategy of the System Based on Artificial Immune” International Conference on Computational Intelligence and Software Engineering 2009

[ZKal2008] Zahra Aghajani Kalkhoran, Mohammad Abdollahi Azgomi “A Multi-Layer Architecture for Intrusion Tolerant Web Services” ICT Group, Iran University of Science and Technology, Tehran, Iran.

BIOGRAPHY

Ajay Nagarajan received his Bachelor of Engineering (B.E.) degree in Computer Science and Engineering from Anna University, India in 2007. He later received his Master of Science (M.S.) degree in Computer Science from George Mason University, USA in 2010. Since February 2015 to date, Ajay Nagarajan has been working as a practitioner in Stroz Friedberg's cyber resilience practice out of their Washington, DC office.