Model-Based Autonomic Security Management of Networked Distributed Systems

Qian Chen

Model-based autonomic security management of networked distributed systems

By

Qian Chen

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2014

Model-based autonomic security management of networked distributed systems

By

Qian Chen

Approved:

Sherif Abdelwahed
(Major Professor)

Bryan A. Jones
(Committee Member)

Thomas H. Morris
(Committee Member)

Pan Li
(Committee Member)

James E. Fowler
(Graduate Coordinator)

Jason Keith
Interim Dean
Bagley College of Engineering

Name: Qian Chen

Date of Degree: December 13, 2014

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. Sherif Abdelwahed

Title of Study: Model-based autonomic security management of networked distributed systems

Pages of Study: 176

Candidate for Degree of Doctor of Philosophy

This research focuses on the development and validation of an autonomic security management (ASM) framework to proactively protect distributed systems (DSs) from a wide range of cyber assaults with little or no human intervention. Multi-dimensional cyber attack taxonomy was developed to characterize cyber attack methods and tactics against both a Web application (Web-app) and an industrial control system (ICS) by accounting for their impacts on a set of system, network, and security features. Based on this taxonomy, a normal region of system performance is constructed, refined, and used to predict and identify abnormal system behavior with the help of forecasting modules and intrusion detection systems (IDS). Protection mechanisms are evaluated and implemented by a multi-criteria analysis controller (MAC) for their efficiency in eliminating and/or mitigating attacks, maintaining normal services, and minimizing operational costs and impacts. Causes and impacts of unknown attacks are first investigated by an ASM framework learning module. Attack signatures are then captured to update IDS detection algorithms and MAC protection mechanisms in near real-time.

The ASM approach was validated within Web-app and ICS testbeds demonstrating the effectiveness of the self-protection capability. Experiments were conducted using real-world cyber attack tools and profiles. Experimental results show that DS security behavior is predicted, detected, and eliminated thus validating our original hypothesis concerning the self-protection core capability. One important benefit from the self-protection feature is the cost-effective elimination of malicious requests before they impede, intrude or compromise victim systems. The ASM framework can also be used as a decision support system. This feature is important especially when unknown attack signatures are ambiguous or when responses selected automatically are not efficient or are too risky to mitigate attacks. In this scenario, man-in-the-loop decisions are necessary to provide manual countermeasures and recovery operations.

The ASM framework is resilient because its main modules are installed on a master controller virtual machine (MC-VM). This MC-VM is simple to use and configure for various platforms. The MC-VM is protected; thus, even if the internal network is compromised, the MC-VM can still maintain "normal" self-protection services thereby defending the host system from cyber attack on-the-fly.

Key words: Autonomic Computing, Self-Protection, Cyber Security, Cyber Attack Taxonomy, Intrusion Estimation, Intrusion Detection/Protection System, Network Forensics, Intrusion Mitigation and Response System, Security Model-based Validation.

DEDICATION

I dedicate my dissertation to God and my loving family. I would like to express a special gratitude to my dearest parents, Jimmy and Lily Chen as well as my adorable dog, Momo.

ACKNOWLEDGEMENTS

I would like to thank Professor Sherif Abdelwahed, who gave me an opportunity to perform research in his group. His knowledge and dedication to teaching and research has been an inspiration. He also constantly encouraged me to apply for faculty jobs. Without his encouragement, guidance, and support, it would not have been possible for me to work in academia.

I would also like to thank Professor Thomas H. Morris, who extended his time, laboratory facilities, tools, and proofreaded my research paper and gave me excellent advice. In addition, I would like to thank the help from his graduate students. I wish to thank Professor Bryan A. Jones for providing great insight into enriching my research through his valuable comments. I am very thankful for Professor Pan Li and Professor Ioana Banicescu who provided me encouragement and support when I applied for academic positions.

Next, I would like to thank Dr. Frederick Sheldon, who is a wise teacher and a helpful friend. He spent hours and hours carefully proofreading my papers, the dissertation proposal, as well as this dissertation. He also gave me numerously valuable suggestions and expanded my understanding of cyber security.

I am grateful to my faithful families at The Church in Starkville. I would like to thank brother and sister Chang, brother and sister Feng, brother and sister Han, brother and sister

Oswalt, sister Chinling Wang, and sister Sunshine for their encouragement, support, and endless prayers for my graduation.

I would like to thank my research group members Rajat Mehrotra, Rui Jia, Ranjit Amgai, and Jian Shi for their helps and interesting discussions. I would also like to thank all of my friends in the United States, China, and Japan.

Finally, I would like to thank faculties, staffs, and students of the Department of Electrical Computer Engineering and the Department of Computer Science and Engineering department. I am really proud to be a part of the Mississippi State Bulldog family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AIC	Akaike's Information Criterion
ARIMA	Autoregressive Integrated Moving Average
ARPANET	Advanced Research Projects Agency Network
APA	Advanced Persistent Attack
ARP	Address Resolution Protocol
ASM	Autonomic Security Management
CIA	Confidentiality, Integrity, and Availability
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CST	Cyber Security Taxonomy
DS	Distributed System
DNS	Domain Name System
DoS	Denial of Service

DMZ	Demilitarized Zone
FCS	Function Code Scanning
GAO	(U.S.) Government Accountability Office
GP	Gas Pipeline
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
ICMP	Internet Control Message Protocol
ICS	Industrial Control System
ICT/IT	Information and Communications Technology
IDS	Intrusion Detection System
I/O	Input/Output
IPS	Intrusion Prevention System
IPsec	Internet Protocol Security
IPv4	The Internet protocol version 4
IRS	Intrusion Response System
MAC	Multi-Criteria Analysis Controller

MC-VM	Master Controller Virtual Machine
MITM	Man-in-the-Middle
MPCI	Malicious Parameter Command Injection
MTU	Master Terminal Unit
NIST	National Institute of Standards and Technology
NFA	Network Forensics Analysis
OWASP	The Open Web Application Security Project
PROMETHEE	Preference Ranking Organization METHod for Enrichment Evaluation
PID	Proportional-Integral-Derivative
PIDS	Performance-based Intrusion Detection System
PII	Personally Identifiable Information
PLC	Programmable Logic Controller
RSE	Residual Standard Error
RSS	Residual Sum of Squares
RTU	Remote Terminal Unit
PSI	Pound Per Square Inch
TCP/IP	Transmission Control Protocol/Internet Protocol

TCP SYN	TCP SYNchronize
TLS	Transport Layer Security
SCADA	Supervisory Control and Data Acquisition
SIDS	Signature-based Intrusion Detection System
SSH	Secure Shell
SSL	Secure Sockets Layer
SQL	Structured Query Language
UDP	User Datagram Protocol
VM	Virtual Machine
XSS	Cross-Site Scripting

CHAPTER 1

INTRODUCTION

Distributed systems (DSs) are defined as a collection of independent, possibly heterogeneous, computers communicating and coordinating with each other to work on a single computational problem, but appearing and functioning as a centralized system. The deployment of DSs inherently improves system performance, reliability, availability, and scalability. The failure of a single node will not disable the entire system, and system expansion and modification are easy to realize without changing the inherent structure of the system.

Due to the evolution of powerful microprocessors and the development of high-speed computer networks, DSs are easily constructed and commonly used in large scale high performance computing environments such as e-commerce, financial services, health care, government, and entertainment. The widespread employment of commercial-off-the-shelf (COTS) applications, operating systems, and network protocols makes DSs a logical choice for large scale computational problems. However, even though performance and reliability are positively impacted, the security of such systems is reduced due to increased system size and complexity resulting in an increased attack surface. This is true because sophisticated attackers can compromise DSs by exploiting vulnerabilities residing in software and communication channels (i.e., complexity is the enemy of security). Failure to protect DSs from these cyber assaults can lead to financial loss, reputational damage, and widespread service disruption. Organized cyber criminals have demonstrated they can even obtain confidential military, government, or intelligence information, which endangers national critical infrastructure, security, and safety.

In the early days of computing systems when Internet technology has not been pervasively used, the security of computing systems was enhanced by protecting computers and their peripheral devices from unauthorized access. In today's cyberspace, although DSs are utilizing a variety of security guidelines and policies, the confidentiality, integrity, and availability (CIA) of these systems are still largely at risk from sophisticated continuous and persistent cyber attacks. According to the U.S. Government Accountability Office (GAO) [14], cyber incidents in fiscal year 2012 were nearly three times more than in the same period four years earlier. Harry Wilson [127] reported that "every minute of every day, a bank is under cyber attack" to illustrate the significance of enhancing cyber security of contemporary DSs.

Autonomic computing technology, inspired by biological autonomic nervous systems, aims to develop computational systems that are capable of configuring, optimizing, healing, and protecting themselves under different working conditions. As the complexity of contemporary computing systems increases and the limits of human man-in-the-loop capacities are obtained, self-protection functionality from autonomic computing technology is an essential option to supplement the system administrators' responsibility to anticipate and defend the system as a whole from malicious activity [73]. To enable self-protection functions, vulnerability assessments are taken to identify system weaknesses and apply a set of baseline security controls. Early warning and prevention mechanisms are necessary to anticipate system security states and proactively eliminate cyber attacks before they can compromise internal networks. Intrusion detection systems, adaptive learning processes, and response efficacy assessments can be integrated along with the employment of closed-loop feedback controls. Such techniques are fundamental to effectively apply autonomic computing technology to DS for proactively anticipating and defending against cyber assaults with little or no human intervention.

Consequently in this work, an autonomic security management (ASM) framework is developed. The application of this framework can anticipate and prevent upcoming attacks carried out by either external or internal malicious actors. Sophisticated attacks that evade the first line of defenses are identified by intrusion detection systems in real-time. ASM uses adaptive learning modules to analyze signatures of novel attacks without degrading or disrupting normal system operation. After gathering attack signatures, recommended responses are evaluated by the controller, which considers both the short and long-term effects of unmitigated attack impacts, reduced operational cost, and recovered system performance to normal levels. The optimal responses are then selected and executed. Therefore, a system adopting the ASM framework has the ability to protect itself from both known and unknown attacks with little or no human intervention in the presence of sudden changes (compared to the nominal profile) that occur within the external computing environment.

1.1 Motivation

Cyber assaults come in a variety of forms for achieving a series of objectives performed by malicious actors. One of the most common and pernicious threats of real-world cyberspace is the data breach [17], which intentionally (i.e., exfiltration) or unintentionally (i.e., extrusion) releases confidential information to untrusted environments. The TJX data breach incident that happened in 2007 is one example of a highly publicized data breach. The attackers, using a man-in-the-middle attack, connected their laptops to TJX's in-store wireless networks to intercept and decode messages between hand-held payment scanners. As a result, the perpetrators compromised the company's database server and stole 94 million customers' personal data and credit card information. Lack of security patching, anti-virus software updating, and authentication cost the company hundreds of millions of dollars in financial damages and greatly damaged their reputation [6].

Supervisory control and data acquisition (SCADA) systems are a type of industrial control system (ICS), which often distributes across different control networks and adopts many aspects of Information and Communications Technology (ICT) to monitor and control physical (commonly referred to as cyber-physical) processes. Stuxnet is a highly publicized computer worm, which subverted Iranian enrichment facilities to temporally derail Iran's nuclear program in 2010 by damaging roughly 1000 centrifuges. Stuxnet includes a root kit to compromise programmable logic controllers of SCADA systems. Social engineering was used to bridge the Iranian "air-gaped" enrichment facilities by creating a situation that fooled victims into breaching normal security procedures (i.e., carrying a flash drive infected by the worm into the secure facilities). Using a man-in-the-middle attack, the worm repeatably sent fake commands to increase the operating speed of the Iranian IR-1 centrifuge from 1,064 hertz to 1,410 hertz for 15 minutes before returning to its normal running frequency [28, 108]. Eventually, there equipment failed to operate normally due to physical damage from the higher frequency pulses.

Nowadays, cyber security gets the highest concern among local, state, and federal government as well as industry. Security solutions such as anti-virus software, firewalls, and intrusion detection/prevention systems (IDS/IPS) are being rapidly developed and deployed. The application of these security countermeasures somewhat reduces security risks of DSs, however, attackers are still successfully compromising these DSs. A recent example is the Target Corporation data breach which occurred in the 2013 holiday shopping season. Attackers stole more than 40 million customers' credit and debit card information [4], an insidious reminder that certainly validates current social and political concerns.

Adversaries will likely never stop attacking DSs with even more highly sophisticated and organized cyber attack mechanisms. Due to the practical limitations and unique drawbacks of commercial security products, organizations employing these security solutions are far from attack-proof. For instance, signature-based and heuristic-based anti-virus software is helpless in combating new malicious codes that exploit unknown vulnerabilities [90]. Traditional firewalls provide little protection from authorized (and unauthorized) internal adversaries. Most IDSs provide passive responses to attacks, and are plagued by the problem of high false alarm rates [106], which ultimately lead to improper responses and high performance overheads. An intrusion prevention system (IPS) combining a firewall and an IDS reacts to cyber attacks actively. However, the low protection speed of an IPS, especially if the IPS has numerous pre-defined rules, leaves more time for adversaries to pilfer DS data (and control).

Moreover, because the utilization of a onefold security control is insufficient to strengthen an organization's cyber security posture, autonomic computing technology using multiple controls in both series and in parallel (also refereed to the defense in depth strategy) has been shown in this research to improve the security of DSs. This innovative technology anticipates upcoming attacks and takes steps to eliminate or mitigate attack impacts with little or no human intervention. This thesis develops a complete security design called the ASM framework, and shows the integration of this framework within a generic DS is straight forward and beneficial to defenders.

1.2 Thesis Statement

This research proposes the employment of autonomic computing technology to DSs (both enterprise systems and ICSs) from a wide range of cyber assaults. To this end, this work claims the following:

- It is practical for DSs to acquire the self-protection feature using autonomic computing technology. As a result, DSs can proactively anticipate attacks, prevent or detect as well as actively and autonomously respond to known and unknown attacks in real-time with minimal overheads.

- Self-protecting systems employ past attack experience [104] to anticipate upcoming attacks and block malicious connections. They can also adopt attack signatures and system security trends to detect known and novel anomalies in real-time, followed by a mitigating course of action that most appropriately responds in a way that does not disrupt the performance of normal system services. Autonomic computing technology can therefore enable DSs to adaptively learn novel attack patterns so that the underlying detection and response evaluation algorithms can dynamically change in ways that reduce the risk from zero-day attacks.

1.3 Research Assumptions

This study is conducted based on the following assumptions:

- Cyber intrusions will lead to anomalies that are detectable and analyzable.

- Zero-days attacks are novel attacks that have never been seen before in our virtual environment and they are detectable.

- The ASM framework contains a master controller virtual machine (MC-VM), on which the configured and installed autonomic components (e.g. forecasters, IDSs, network forensics analysis tools, and intrusion response systems) closely interact to strengthen the security of the DS environment. The MC-VM operates in a stealth mode, i.e., the MC-VM is hidden from both internal and external users and devices. Hence, we assume that the MC-VM is attack-proof.

- Devices, components, and modules of the ASM framework operate continuously until they are terminated by authorized system administrators. Neither physical nor cyber attack will disrupt the normal performance of the ASM approach. For example, they could live within the protected environment of a hypervisor.

- The communication link between virtual SCADA servers and field devices is the Ethernet.

1.4 Research Hypothesis

Given the assumptions listed above, this research postulates that: the statements below can be utilized in a cost effective way to make various types of DSs more resilient to cyber attacks. In doing so, the ASM framework must show that the following listed functional/non-functional requirements can be realized/utilized together to that effect.

1. The instantaneous security state of DSs (computing systems and ICS systems) captured by system models accurately and precisely reflects resource, network, and security features, and that the "normal" security state can be accurately and precisely represented by observations from those features.

2. The predefined "normal" security state(i.e., profile) of the DS is utilized as the standard from which to determine the correct system security state. Anomalies are identified by comparing the "normal" operational region with predicted and real-time monitored measurements.

3. The forecaster module of the ASM framework anticipates upcoming intrusions and produces early warnings so that prevention mechanisms can be deployed early enough to reduce cyber risks (loss exposure).

4. Intrusion detection systems detect both known and unknown attacks, as well as classify known attacks into categories using data mining techniques and pattern matching algorithms for the purpose of improving overall ASM performance (e.g., reduced false alarms, overhead, and time to recovery).

5. Novel attack signatures and their impacts on system, network, and security features can be investigated by the adaptive learning module in real-time. Therefore, intrusion detection and response evaluation algorithms, as well as the set of recommended responses are updated dynamically in near real-time. Consequently, a self-protecting system is capable of protecting against unknown attacks, but not without postulate #2 holding.

6. The most appropriate responses are chosen and initiated by the controller from a set of candidate responses to eliminate or mitigate attack impacts, recover system performance back to normal, and maintain normal system operations with a low overhead.

7. The basis for choosing one candidate response from another is recovery time, system resource recovery, and operational costs. In some cases, multiple sequential responses can be deployed on the monitored vault.

8. The ASM framework is extensible and can be easily instantiated into different computing environments

1.5 Research Objective

This section summarizes the claims and rationale mentioned from above. Due to limits of human capacities, defects within security mechanisms, and security concerns created by the pervasive use of computers in large scale systems, cyber security maintenance and enhancement are becoming crucial challenges. There is great urgency for developing innovative solutions to effectively improve the security of cyberspace and avoid the types of losses described above. For this purpose, we lay emphasis on the self-protection characteristic of our ASM approach. The objective of this research is to apply autonomic computing technology to self-protect DSs from known and unknown cyber attacks while maintaining the highest possible level of underlying service performance with little or no human intervention.

A risk assessment is necessarily first performed offline to characterize the system boundary and select relevant features needed to identify the“normal” operating region. If the normal operating region changes over time (e.g., a new node is added to the “system” being protected) then this change must be retrained offline. However, the ASM framework will adapt autonomously to new (unknown) attack scenarios by virtue of the learning and IDS modules combined functionality. This research is then validated through experiments by carrying out real-world cyber intrusions to attack a three-tier web application and a SCADA system from various known and unknown attacks. The experimental results have been analyzed and some conclusions have been drawn as described in the next section. The ASM specific goals are summarized as follows:

- Classify cyber attack patterns and map the motivations and goals of malicious actors to each attack class (to validate hypothesis #1 and #2).
- Design a self-protection framework to 1) autonomously anticipate and detect system anomalies, 2) protect the DS from malicious attacks, 3) analyze unknown attack signatures and adverse impacts on the system performance, 4) efficiently maintain system performance and underlying services even if hosts are compromised, and 5) mitigate and eliminate disruptions and recover the system behavior back to its normal state (to validate hypothesis #3, #4, #5, and #6).
- Install, configure testbeds, and validating the ASM framework for simulating a three-tier web application and a SCADA system controlling and monitoring a gas pipeline and a water storage tank (to validate hypothesis #8).

- Apply the framework to representative testbeds, generate realistic attack models, and conduct experiments to validate self-protection mechanisms and functionality (to validate hypothesis #1-#7).

1.6 Research Contributions

The research contribution of this thesis follows from a demonstrating self-protection features of the ASM utilizing autonomic computing technology to proactively protect and avoid the impacts from cyber assaults and maintain the highest possible level of underlying service performance and integrity with little or no human intervention. Specifically this research thesis makes the following contributions:

- The ASM approach developed in this research can switch between fully-autonomous and semi-autonomous modes: The ASM approach can be tailored to various computing environment contexts; the autonomic property provides precise control to enable the highest possible level of service performance and integrity. An ASM system can also be used as a semi-autonomous system. In which case, a system administrator can select and initiate appropriate responses to mitigate ambiguous cyber assault signatures in concert with any multi-criteria analysis controller (MAC) determinations.

- The ASM framework is simple to configure and deploy: The main modules of the ASM approach are installed on a MC-VM. As a result, the autonomic computing system can be easily realized by adding the MC-VM to most of the popular networked platforms.

- The adoption of the ASM framework supports reliable, sustainable, and resilient self-protection performance. Most of the components and modules are installed on the MC-VM, which operates in a stealth mode, i.e., the MC-VM is hidden from internal and external users and devices to protect it from compromise. Hence, a partially compromised computing environment does not impact the functioning of the MC-VM. On the contrary, the MC-VM coordinates the ASM components toward achieving the highest levels of performance possible; resilient (operated and maintained) to survive a cyber incident while sustaining critical functions. In other words, the victim system performance shall be maintained toward sustaining critical functions and returning service delivery to normal operations even in the presence of compromising failures irrespective of whether instigated by external or internal attackers.

In this research we demonstrated through rigorous experimentation and analysis that the ASM framework can be utilized cost effectively to make DSs resilient. Furthermore, under the stated assumptions, that DSs are amenable to applying autonomic computing technology to proactively anticipate and prevent the effects of cyber attacks. Even if these attackers evade the first line of defense, their impacts on system performance and functional disruption can be detected and mitigated by ASM deployed intrusion detection systems (IDSs) and intrusion response systems (IRSs). Furthermore, live network forensics analysis has been integrated into the ASM framework so that the proposed self-protecting system can defend against novel zero-day attacks without disrupting normal services. Unique and efficient system models for a web application and industrial control systems are developed. An enterprise system, a gas pipeline, and a water storage tank testbed have been developed and demonstrated to validate this thesis.

1.7 Organization

Subsequent to this introductory chapter the remainder of this document is organized as follows:

- **Chapter 2** briefly reviews the motivations for this research and presents a detailed introduction of the multi-tier client-server architecture and SCADA systems. Related scholarly works about autonomic computing technology and techniques, theories, and security solutions for establishing a self-protecting system are surveyed. Before concluding this chapter, we compare our research with the literature to differentiate the contributions of this research.

- **Chapter 3** introduces some well publicized cyber attacks that illustrate the threat to computing and supervisory control and data acquisition (SCADA) systems. We classify commonly seen cyber attacks and analyze their impacts on system network, and security resources. We also incorporate typical features such as attack targets, goals, scopes, information impact, and defense strategy into our classification scheme.

- **Chapter 4** presents research efforts of this thesis: the development of the ASM approach and its extension for the protection of computing and SCADA systems from various cyber attacks. This work deploys a complete security architecture that provides risk assessment, attack estimation (anticipation) and detection, live network forensics analysis, and intrusion response for improving the cyber security and resiliency of DSs.

- **Chapter 5** introduces two different testbeds. The first replicates a three-tier web application system. The second provides a virtual SCADA system that controls and monitors a gas pipeline and a water storage tank. The self-protection function of the ASM approach was validated through a set of experiments using real-world cyber attack tools and profiles.

- **Chapter 6** summarizes the research objective and contributions. In addition, future research and outlook for autonomic computing and self-protection systems are discussed.

CHAPTER 2

BACKGROUND

Along with the obvious benefits from the broadly global deployment of the ubiquitous Internet comes the pervasive and decisively "cyber" security concerns of momentous proportion. These concerns have manifested from cyber attacks, which are able to infect and consequently affect millions if not billions of hosts and networks. Cyber attacks have resulted in multi-billions of dollars in losses from undermining information, privacy, and cyber-physical security.

What many have described as catastrophic can be mostly boiled down to violations of Confidentiality, Integrity, and Availability (CIA). Indeed, within the cyber-physical world, practitioners rank *availability* first because of its crucial role in potentially disrupting control systems for industrial automation and energy distribution. In the banking and commerce sectors, however, *confidentiality* and *integrity* reign supreme. There is no doubt that the effects from disruptive technologies that leverage cyber-infrastructure will continue to influence every aspect of our technologically and cyber enabled lives for the foreseeable future.

Not surprisingly, the need to defend computing systems against adversaries in computer networks can be traced back to 1986, when a German hacker, in connecting to the Advanced Research Projects Agency Network (ARPANET), was able to exfiltrate classified military information [111, 3]. Today, according to "The 2013 Cost of Cyber Crime Study" conducted by the Ponemon Institute [46, 15], the average financial loss due to cyber crime was $11.56 million/year for each of the U.S. organizations that were surveyed. The cost of cyber crime in 2013, considered to be an important benchmark, was 76 percent more than in that same period four years earlier. Resolving a cyber attack on average takes 32 days, which represent an increase by almost 130 percent during the last four years. The study also revealed that, on average, 122 successful attacks have compromised U.S. organizations per week, which was 16 percent more than the number reported in 2012. No doubt, both cyber attack frequency and the resulting disruption and damage are on the rise!

Autonomic computing, which was first introduced as a vision in 2001 by IBM's senior vice president of research, Paul Horn [73]. Autonomic computing aims to enable self-managed computing systems toward relaxing the need for human intervention in directly governing system resources. The underlying concept was first inspired by the autonomic nervous system that regulates heart rate, body temperature, and digestion from the unconscious efforts of the brain [73]. In an analogous way, the autonomic capacity self-manages system resources and elements in four aspects: self-configuration, self-optimization, self-healing, and self-protection. The first three autonomic computing aspects focus on 1) automated configuration of components following high-level policies; 2) improvement of system performance and efficiency; and 3) detection, diagnosis, and repair of system failures.

The self-protection aspect can be used to secure and safeguard computing system communications, properly grant authorization privileges, and evaluate performance of security mechanisms toward achieving greater resiliency. By continuously monitoring measures of performance and security, autonomic systems can ascertain differences associated with normal versus abnormal system behavior. With the usage of mathematical techniques for intrusion detection and response evaluation algorithms, the autonomic approach can efficiently and automatically respond to a wide range of cyber attacks. Since human capacities cannot possibly protect large scale complex DSs from increasingly sophisticated cyber adversaries combined with high volumes of network traffic, the application of autonomic computing can supplement or replace human intervention needed for the management of cyber security posture and situational awareness.

In this thesis, a multi-tier enterprise application architecture testbed and a virtual supervisory control and data acquisition (SCADA) testbed are used to develop exploits of a web application and a process control system. Both the multi-tier web application and the SCADA system adopt the client-server model. The functionality of autonomic computing is encapsulated into the internal network of the web application and the control and field device networks of the SCADA system. In this way, the testbeds can accurately simulate the context either self-protection feature of an autonomic enterprise system or an industrial control system (ICS) to validate both the self-protection feature as well as the extensibility of the ASM framework and portability.

The autonomic security management (ASM) framework is tailored to provide interfaces that support the self-protection property. The master controller virtual machine (MC-VM) incorporates the online-monitoring capability to enable continuous collections of system “security relevant” behavior. The data pre-processing module ingests the real-time observations. Subsequently, the processed datasets are sent to the forecaster, and are used to predict the future security states of the system. Output from the forecaster is a projection of future system behavior. If the system suffers from activities reminiscent of a pending/upcoming attack, early warnings will be sent to the multi-criteria analysis controller (MAC) to dispatch protection mechanisms designed to eliminate and/or mitigate such attacks, so-called “active response mechanisms.”

As shown in Figure 2.5, if attackers evade the first line of defense (intrusion forecasting and protection), real-time anomalies can be detected by the second line of defense via an intrusion detection system (IDS) line of defense. Appropriate responses are then invoked by the MAC for the purpose of regulating system security performance back to the “normal” region. Protecting the system from novel attacks is a challenge for most distributed systems. This is true because the attack patterns and their adverse impacts are unknown. However, by applying autonomic computing system principles, the system can learn signatures of unknown attacks with the help of an adaptive learning module. Consequently, the

ASM approach can protect against known and unknown cyber intrusions as was demonstrated by experiments conducted within our testbeds for distributed systems (DSs).

As described above, the ASM approach has been applied to self-protect a three-tier web application and a SCADA system from a variety of cyber attacks. Experimental results show that the computing environment security relevant behavior can be predicted, and that anomalies can be prevented and/or detected. One important benefit from the self-protection feature is the cost-effective elimination of malicious requests before these requests can intrude on victim systems. These results support the thesis that by employing autonomic computing technology we can secure systems and networks cost efficiently.

2.1 Introduction to Web Services

DSs are more extensible and resilient to failures. Consequently, they are often more economical than centralized computing systems. As a result, DSs are more commonly used in e-commerce, search engines, information service, financial service, and social networking applications. The basic architecture employs the client-server model. Adopting this model, clients send requests to servers and wait for their reply to obtain specific services (e.g., database services, web services, mail services, and application services) [118]. Most Internet application protocols such as FTP, SSH, Telnet, HTTP, and TLS/SSL rely on the TCP/IP model, which is a reliable connection-oriented protocol that naturally supports the client-server model [118]. TCP connections must be established before sending client requests to servers. After servers respond to clients, the TCP connections are normally closed.

Numerous web services rely on the HTTP protocol as the underlying message transport. Generally, these web services adopt a three-tier client-server architecture that contains a web tier, an application tier, and a data tier. As shown in Figure 2.1, the web tier, also known as the presentation tier, gives the requester access to a server-based application visible from the graphical user interface. The application (logic) tier manages business logic, processes commands, and transforms data. The data tier provides information data usually stored in databases or file systems.

Web servers are the most easily targeted tier since they directly connect to external actors and allow public accesses. To enhance their performance, bandwidth, availability, and maintainability, web servers can be placed in a demilitarized zone (DMZ) as shown in Figure 2.1 [133]. Inbound firewalls are configured in a default deny-all mode, namely, deny all incoming messages. Thus, incoming traffic to each tier is restricted by access groups, service ports, protocols, and source Internet Protocol (IP) addresses.

As shown in Figure 2.1, the firewall located in front of the web tier only allows traffic whose destination port is 80 (HTTP) and/or 443 (HTTPS) to access the web server. To mitigate external intrusions, two firewalls are placed between the application tier and data tier. The firewall in front of the application tier filters all requests to the application server from the web server group except the messages whose destination port is 8000 (application service). The firewall in front of the database server filters all traffic but permits the application server group access to the service running on port 3306 (MySQL). Administrative access from corporate networks on port 22 (Linux SSH) and/or port 3389 (Windows Remote Desktop) is permitted for all three groups (web, application, and data tiers) [1, 120].

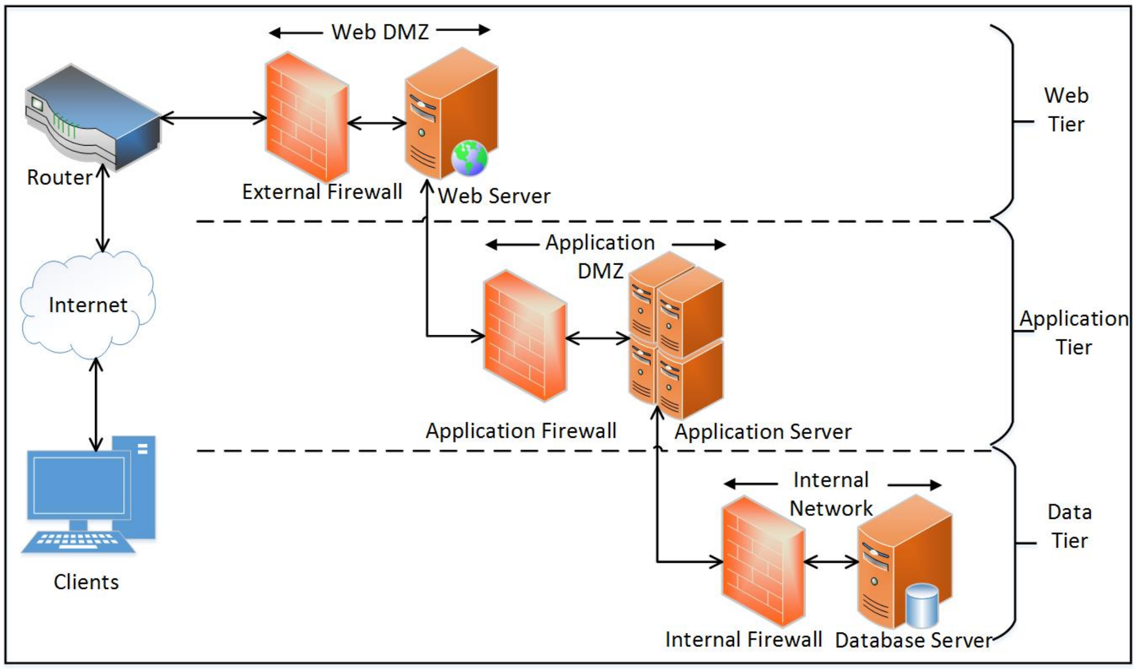


Figure 2.1

Three-tier Architecture and Security Group Firewall Placement for a DS [1]

2.2 Introduction to SCADA Systems

Supervisory Control and Data Acquisition (SCADA) systems are a type of industrial control system (ICS) that adopts many aspects of Information and Communications Technology (ICT/IT) to monitor and control physical (commonly referred to as *cyber-physical*) processes. A SCADA system is usually composed of a set of devices including sensors, actuators, programmable logic controllers (PLCs), remote terminal units (RTUs), human machine interfaces (HMIs), and master terminal units (MTUs) (as shown in Figure 2.2). Field devices such as PLCs and RTUs collect and convert sensor sourced analog measurements to digital data. The digital data are then sent back to MTUs, located in the control network, via communication links (e.g., Internet, radio, microwave, and satellite). In near real-time this data is processed by MTUs and displayed on HMIs to enable operators to make intervening control decisions. HMIs can also be used to store historical data as well as to send automatic and/or operator generated commands to field devices for process control within the cyber-physical infrastructure.

Contemporary SCADA systems are often distributed across different control networks. Such Industrial Automation systems are widely used nowadays to monitor and control industrial processes (e.g., manufacturing), infrastructure processes (e.g., gas pipeline), and facility processes (e.g., heating, ventilation, and air conditioning systems), to name only a few. Even though reliability and recovery of physical processes are positively impacted, the cyber security of SCADA systems and controlled processes is reduced in part because of the common use of open and standardized protocols (e.g., Modbus, ICCP, and DNP3) and Internet-based cyber communications. Consequently, the security issues from ICT/IT systems are mostly inherited by SCADA systems. Organized attackers can compromise SCADA systems by exploiting common vulnerabilities residing in operating sys-

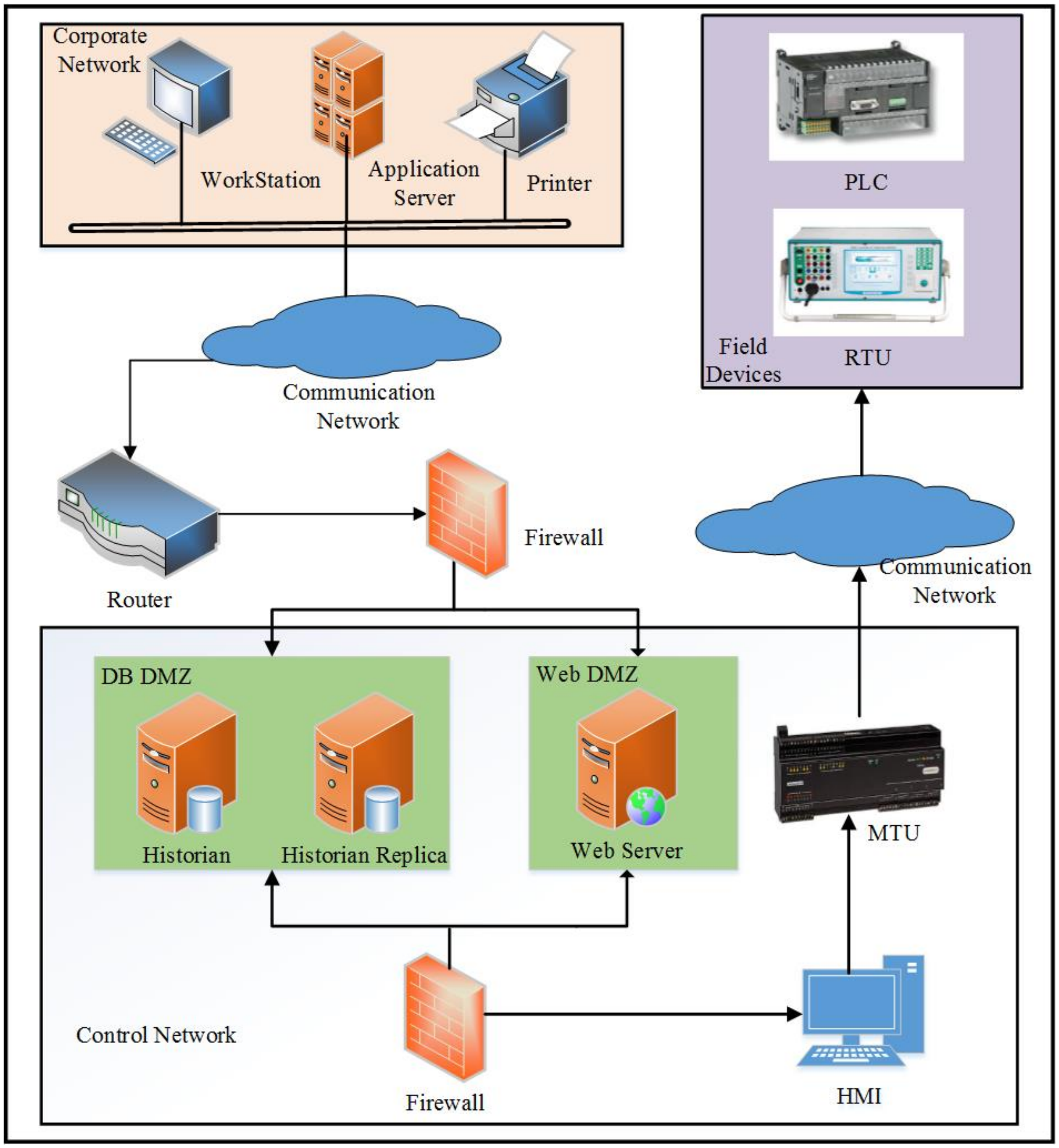


Figure 2.2

SCADA Architecture and Security Group Firewall [112]

tems, transmission protocols, communication channels, and field devices. Compared with ICT/IT systems, failure to protect SCADA systems is more dangerous. This is true because some of these cyber attacks have the potential for causing severe financial loss, property damage, and widespread service disruption.

To enhance the security of ICS, we follow the NIST (National Institute of Standards and Technology) ICS security guideline [112] to place a pair of firewalls between the corporate network and the control network, as shown in Figure 2.2. Servers within the control network such as historian servers and web servers are situated in DMZs so that malicious packets sent from the corporate network proceeding to the control network are blocked by the first firewall. Besides eliminating illegitimate traffic from compromised servers, the second firewall positioned in the control network can also reduce the influence of control network traffic to the shared web server. This two-firewall architecture separates responsibilities of control group devices and IT group servers. As a result, this architecture helps the SCADA system enhance its security.

Even if organizations design their internal networks with the adoption of DMZs following recommended guidelines for securing web servers [120] and ICS [112], perpetrators still may have the opportunity to compromise internal control networks. For instance, the external firewall in Figure 2.1 denies all traffics except HTTP and HTTPS messages. Therefore, malicious users could exploit HTTP and HTTPS protocol vulnerabilities to compromise web servers by using flooding-based HTTP and XML denial of service (DoS) attacks. Using this scenario, adversaries aim at slowing down web server performance thereby blocking legitimate client requests and undermining information availability, known as also called denial of information attacks. The application tier can be compromised using IP spoofing attacks, in which circumstance the attackers evade the firewall in front of the application server to remotely access the server. The database servers can be maliciously accessed without necessarily possessing a valid username or password by launching SQL injection attacks. These attacks typically result in leakages of client personally identifiable information (PII), financial information, employment records, and even trade secrets.

Similar to the web services, following ICS security guideline is not enough to eliminate or prevent sophisticated cyber attacks from continuously compromising the control network or the field device network. Adversaries can interrupt the physical process by flooding the communication network so that neither field devices or the HMI can communicate with each other. U.S. Department of Energy developed the roadmap to achieve energy delivery systems cybersecurity [10]. The roadmap includes five strategies such as building a culture of security, accessing and monitoring risk, developing and implementing new protective measures to reduce risk, managing incidents, and sustaining security improvement. By 2020, resilient energy delivery systems are expected to survive cyber incidents while maintaining critical functions.

Our testing and evaluation of the ASM framework is described in Chapter 5. We used realistic attack scenarios to validate the self-protection function of in a laboratory scale web application and SCADA testbeds. From the experimental results, we can see that the ASM approach continuously monitors security states of the web application and ICS systems, provides the defense-in-depth strategy to mitigate cyber incidents, quickly returns the system performance back to normal, and learns from such incidents in order to avoid compromise by similar attacks in the future.

2.3 Autonomic Computing and Self-Protection

Kephart [74] summarizes that an autonomic computing system must possess eight key properties: 1) self awareness (i.e., needs to know itself), 2) self configure and reconfigure under various circumstances, 3) continuously optimizes itself, 4) failure recovery, 5) must self-protect from cyber attacks, 6) know its environment and act accordingly, 7) function within an open environment, and 8) optimize and anticipate needed resources. To establish an autonomic computing system, one must employ control theory that provides finite methodologies for diagnosing and repairing system problems. This can be achieved with the utilization of feedback loops, autonomic managers of self-managing systems monitor managed elements (e.g., system resources, network resources, web servers and database systems) and their external environment. After analyzing this information to determine the desired (baseline) values of the observables (reference inputs), autonomic managers construct and execute plans to regulate systems based on current system states [73, 55]. Figure 2.3 illustrates the autonomic computing model, which is the combination of autonomic elements and feedback control theory.

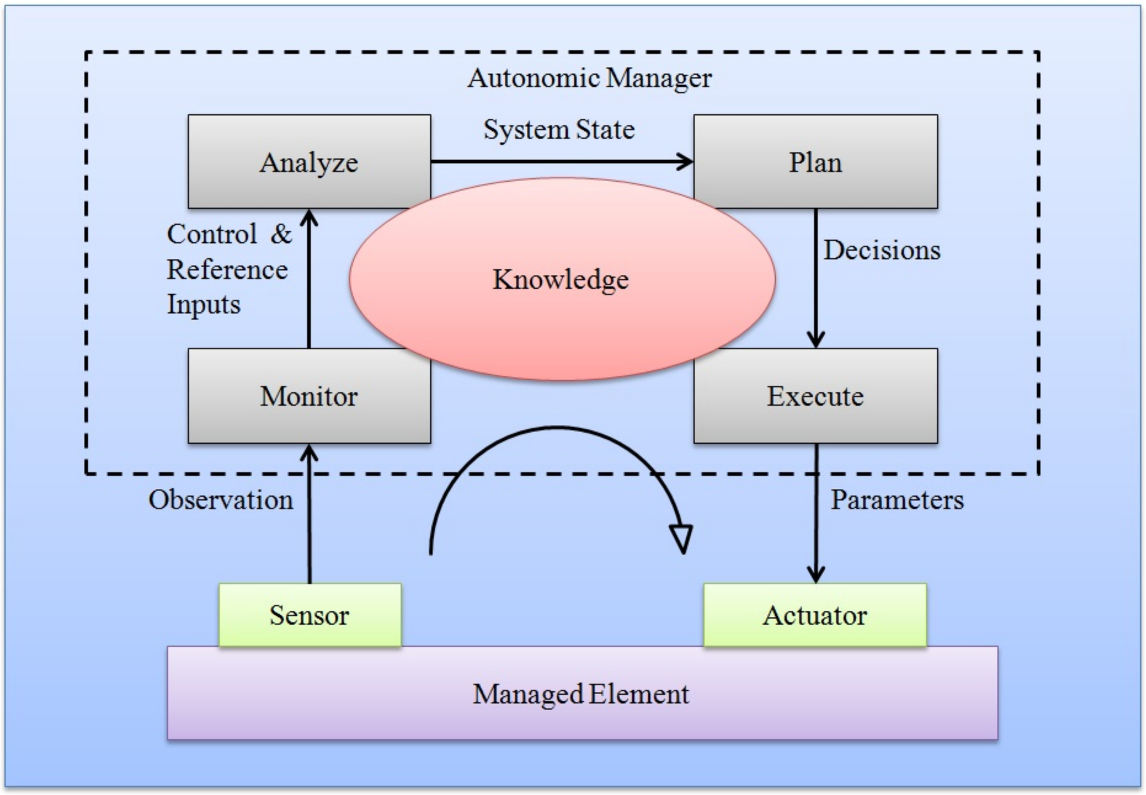


Figure 2.3

Autonomic Computing Model [73]

In large scale distributed environments, autonomic computing is a promising technique to enable systems to self-manage without requiring human commands. Recently, this new technology has seen rapid development within diverse areas. Fault management in telecommunication domain has applied autonomic computing technology to identify faults relying on the analysis of event messages [110]. Self-managing features have also been added to database systems, providing such systems with the potential to optimize resources, provide configuration wizards, and dynamically allocate memory resources. Self-managing databases can also detect, isolate, correct, and recover themselves when faults or attacks are injected into computing systems [82, 102, 58, 54]. Autonomic computing participates in achieving wisdom webs as well. The intelligent web regulates their functions and application services, and specifies their roles, settings, and relationships using web services. Wisdom webs are also able to automatically discover, extract, and obtain knowledge from web-related data sources [77, 135, 80]. Autonomic computing has also been injected into web service systems to schedule tasks, allocate bandwidth, balance loads, control network traffic, and control CPU power autonomously [26, 24, 25]. In addition, autonomic industrial control systems have been partly realized to self-manage critical infrastructures. As a result, process control systems can protect themselves from cyber attacks and faults. Large scale decentralized systems containing the self-optimization characteristic regulate policies to manage critical infrastructures. The self-configuration feature allows systems to adjust their configuration toward maintaining maximal resilience for critical infrastructures in real-time working environments [27, 64, 56, 47].

Autonomic computing technology not only successfully configures, optimizes, and heals managed elements as described in recent research, but also protects enterprise and industrial control systems from cyber attacks. In the following subsections, essential components, theories, and techniques employed to realize self-protecting systems are presented. First, the control theory and its application to self-protecting systems are discussed. Second, the functionality of self-protecting systems is introduced in detail along with a self-protecting model. Third, state-of-the-art self-protecting systems are surveyed.

2.3.1 The Employment of Control Theory to Self-Protection

Self-protecting systems proactively anticipate cyber assaults relying on early reports generated by sensors. These systems also take steps to eliminate or mitigate cyber attacks with little or no human intervention. They adopt the feedback control theory to monitor managed elements, analyze system performance, and select and execute appropriate plans continuously to enhance security, reduce the operator load from system management, and consequently reduce human errors and cut operational costs.

The block diagram of a self-protecting system adopting the feedback control loop shown in Figure 2.4 is taken from a paper by Diao et al. [55]. Similar to Diao's research, this thesis presents a self-managing system using control theory to regulate user connections in the context of a constantly changing operating environment. The same type of feedback control is used to describe how control theory is used in a self-protecting system.

The essential elements in Figure 2.4 for implementing a self-protecting system are described as follows:

- Target System: This refers to the victim system that is attacked by adversaries. For example, the victim can be a multi-tiered web application environment, which contains web servers, application servers, and database systems; or possibly only a single server (e.g. a historian server or a HMI) or a field device (e.g. a PLC or a RTU) placed into ICS networks.

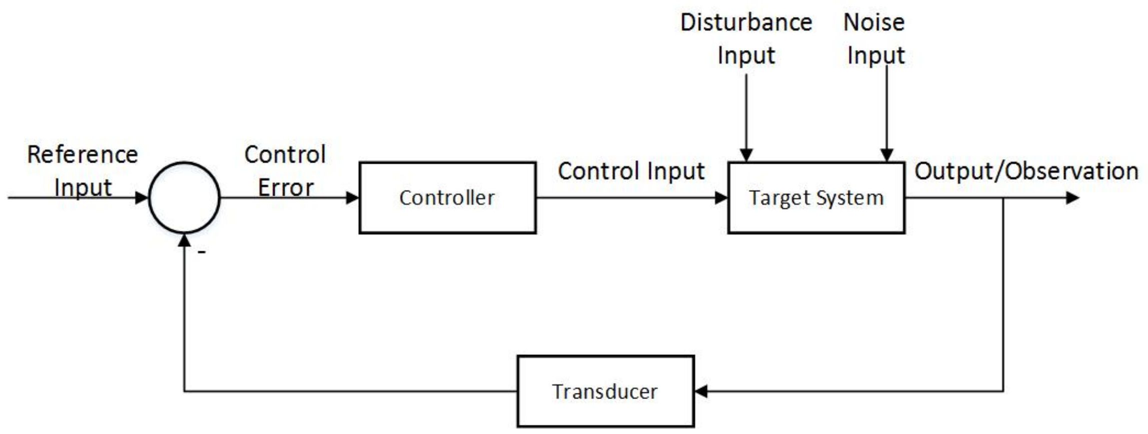


Figure 2.4

Feedback Block Diagram [55]

- Reference Input: Reference inputs are the desired values from target system outputs. In reference to security aspects, the desired system outputs are: 1) providing "normal" services to legitimate clients and 2) eliminating and protecting systems from all cyber assaults.

- Control Input: Control inputs are effective countermeasures or responses that are able to eliminate or mitigate cyber attacks thereby improving the security of DSs within their specific operating environments.

- Control Error: Control errors results from large differences between the *Reference Input* baseline and current observations. The differences could be Euclidean distances between measured system states and the "normal region" of a secured system. The differences also provide the basis for determining the likelihood that the system performance is within the "normal" operating range.

- Controller: The controller evaluates recommended responses or policies, and initiates the most appropriate responses needed to recover the compromised system based on control environment inputs.

- Disturbance Input: Any attacks undermining confidentiality, integrity, and availability of DSs are represented by disturbance input. In the case of web applications, disturbance inputs include malicious requests, unauthorized accesses, spoofing, DoS attacks, and eavesdropping (i.e., MITM).

- Noise Input: This refers to missing data or noisy data collected by sensors.

- Transducer: Transduces system security states in real-time to enable the security states of the *Target System*'s security state to be compared with the *Reference Inputs* (i.e., "normal region") to ascertain the Control Errors.

- Output/Observation: The output of the controller is the observation of the *Target System* security states, which is composed of real-time values from the system, network, and security features. These selected features are addressed in Section 3.2 and include *CPU Time*, *Memory Utilization*, *Packet Received/Sent Rate*, *TCP Connection Performance*, and *Permission* (or *Access Right*).

2.3.2 Self-Protection Model and Functionality

To better understand how a self-protecting system is architected, Figure 2.5 gives a model for adopting control theory to assess vulnerabilities, and to baseline the normal

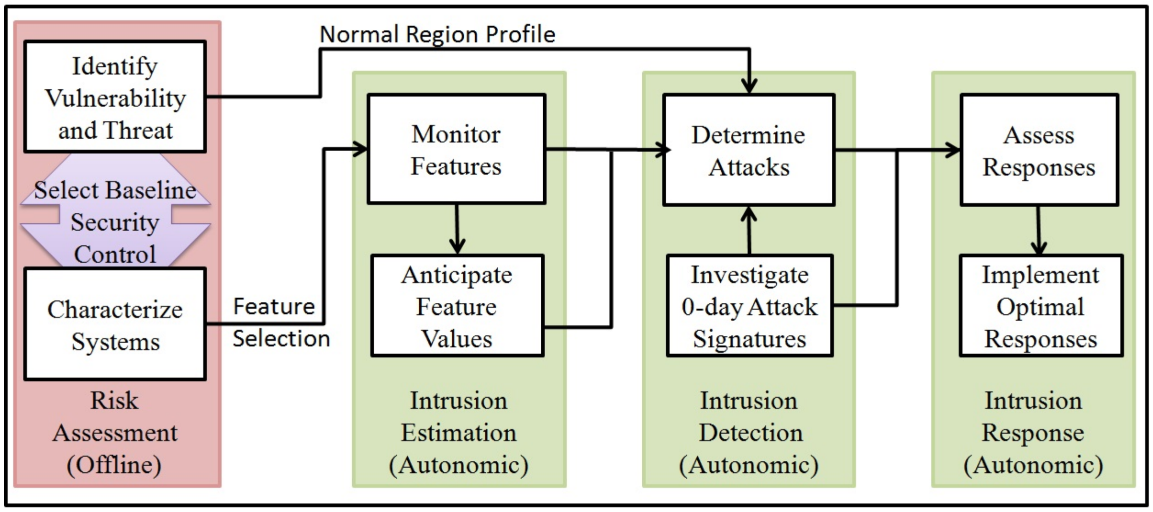

Normal Region Profile
Identify Vulnerability and Threat
Select Baseline Security Control
Characterize Systems
Risk Assessment (Offline)
Feature Selection
Monitor Features
Anticipate Feature Values
Intrusion Estimation (Autonomic)
Determine Attacks
Investigate 0-day Attack Signatures
Intrusion Detection (Autonomic)
Assess Responses
Implement Optimal Responses
Intrusion Response (Autonomic)

Figure 2.5

A Self-Protection System Model

region of the DS behavior for a specific operational environment. The self-protecting system, as the first line of defense, anticipates upcoming anomalies, sends early warnings to the controller, and protects the system by signaling the controller to initiate an active response to selected threats. Sophisticated attacks that evade the first line of defense due to incorrect estimation of future system security states may be thwarted by an IDS. Live network forensics analysis is used to learn the causes and impacts of unknown attacks. Novel signatures are utilized to update detection algorithms and the active response library. As a result, similar attacks can be removed in the future by the first line of defense. The self-protecting system is constructed from four components: risk assessment, early warning and prevention, intrusion detection, and intrusion response. The functionality of each component is discussed as follows:

- Risk Assessment: The first step to establishing a self-protecting system is to assess its security risks. Risk assessment helps an organization to determine the impact and likelihood that a given threat will and can successfully exploit a particular vulnerability. This module is divided into two main components for characterizing systems and identifying system vulnerabilities and threats. DS performance is first analyzed and characterized, followed by the establishment of the "normal region" which represents secure system performance using normal datasets of relevant features. These features are the same as the "dimensions" discussed in Section 3.1 that impact system, network, and security resources. Outputs from this module indicate the degree to which the DS are vulnerable. System administrators select and initiate baseline security controls based on the risk assessment result which in-turn prioritizes the active response to abnormal unwanted or unauthorized activities. Examples of baseline security controls include firewalls, anti-virus software, access control lists, security policies, and honey pots. Attributes (described from Subsection 3.2.1 to Subsection 3.2.3) selected by the feature selection module are the same parameters used to determine the "normal region." Real-time observations are collected by the monitor module, which is an operating system snap-in such as Windows Performance Monitor and Linux System Activity Monitor.

  Most research that focuses on risk assessment, intrusion detection, and forensics analysis include feature selection functions. Selection of appropriate features is crucial for assessing attack impacts and detecting (true positive) anomalies correctly. Proper feature selection can also greatly reduce the complexity of the data processing and analysis phase. A risk assessment module involving expert knowledge is difficult to realize "autonomously" [47]. Consequently, the establishment of the "normal region" (i.e., an operational profile of steady state activities) and the analysis of particular attack impacts and response effects are usually studied offline.

- Early Warning and Prevention: System sensors continuously monitor selected features. Estimation of the system security state uses historical observation to project future feature values using statistical methods combined with system models. Upcoming attacks are therefore anticipated by comparing the "normal region" with the estimated outputs of the system model using time series forecasting methods (e.g., time series forecasting methods such as Kalman Filter or the autoregressive integrated moving average (ARIMA) method). Early warnings, therefore, are sent to the controller, and appropriate control mechanisms are executed to neutralize attacks (on-the-fly).

- Intrusion Detection: The IDS is a data mining tool that enables real-time event analysis. This module provides accountability for intrusive activities thereby detecting anomalous system behavior. To this end, three types of IDSs (host-based, network-based, and hybrid IDSs) have been developed using signature and anomaly techniques. The anomaly detection technique comparing real-time system performance with the normal system model detects known and unknown attacks. The signature detection technique identifies known attacks relying on matching observations to known misuse patterns. To identify zero-day attacks (i.e., attacks that exploit previously unknown vulnerabilities using perhaps unknown tactics and methods) the signature database must be upgraded frequently. Properly utilizing detection techniques identify both known and unknown attacks with low false alarm rates. In Figure 2.5, the intrusion detection module has the ability to analyze evolving zero-day attacks. This functionality is realized using live forensics analysis tools to learn unknown attack signatures. These signatures are adopted by online detection algorithms and defense mechanisms. As a result, the previously unknown attacks can be autonomously incorporated into the corpus of known attacks used subsequently by the first line of defense.

- Intrusion Response: The utilization of intrusion response systems (IRSs) enables the self-protecting system to mitigate attack impacts and regulate system behavior back to normal. IRSs can be divided to static and dynamic types. A dynamic IRS is commonly used to enable ASM. This is because it allows the execution of lower ranking responses for the protection of the compromised system if the so-called "optimal" responses are not sufficient to mitigate attack impacts. If lower ranking responses are implemented and successfully regulate system performance, the dynamic IRS will escalate rankings of these responses by modifying the initial values of the assessed criteria. Examples of relevant criteria include response effects (e.g., how fast can responses regulate system behavior back to normal), operational costs, and other influences on normal operations.

Feedback control theory is the primary means used to realize self-protecting systems. Therefore, after the self-protecting system realizes all of the steps mentioned above, sensors will still continuously monitor selected features to identify whether system performance does actually return to normal running behavior. When abnormal behavior reoccurs, the system will repeat the self-protection processes mentioned here until the system reaches the baseline normal operational range of states.

2.3.3 State-of-the-art Self-Protecting Systems (SPSs)

Self-protecting systems know their environment and proactively enhance network and information security. As described in Figure 2.5 before a self-protecting system can be useful, the system boundary (e.g., routers, firewalls, web servers, application servers, databases) must first be determined so that those security controls can be baselined during the risk assessment phase. Abnormal system behavior is then a quantifiable deviation from the baseline. Anticipated abnormal system behavior is the early stages of a departure from the baseline. SPSs compare each new request to the anticipated behavior, followed by the execution of an appropriate control. One example of self-protection in a DS is introduced by Claudel et al. [45]. Data from the sensors and actuators of the proposed system are utilized to identify abnormal (malicious) requests and isolate compromised nodes automatically. Dean et al. [51] designed and implemented an unsupervised behavior learning system for virtualized cloud computing infrastructures. Their system uses the selforganizing map learning technique captures system performance and anticipates unknown anomalies.

Qu et al. [97] developed an autonomic control and management framework along with self-configuration and self-protection functions. Their system detects most Denial of Service (DoS) attacks and scanning attacks by employing autonomic components such as online monitoring, feature selection, anomaly analysis, and protective action. This method, however, cannot detect exploits or remote to local attacks. Wailly et al. [123] built a virtual environment self-protecting architecture to secure infrastructure-level cloud resources. The architecture has flexible security policies, enables cross-layer defenses, and implements multiple control loops. This is an open architecture for detecting and reacting to cyber attacks. Hariri et al. [68] developed an online monitoring and self-protection mechanism applying Hotelling's T^2 methodology to calculate the normal region of network flows. The system is protected automatically by discarding malicious flows that are not inside the regions. Gelenbe et al. [63] provided an autonomic approach to defending against DoS attacks. They traced attacking flows from the compromised node and dropped malicious packets upstream.

The self-protection feature also has been used to protect database systems. IBM's Self-ProtectIng DatabasE Research (SPIDER) project monitors various log files, and detects malicious queries aimed at compromising database systems. The damaged parts of the system are isolated while other functions are kept running normally [66].

The self-protecting SCADA system is still a new research area. Up to now the only team we know of is at the University of Arizona [47]. They have applied autonomic computing technology to protect SCADA systems against DoS attacks, buffer overflow attacks, and reset_output attacks [1]. Their self-protecting system combines monitors, feature selection modules, and an anomaly-based IDS. When anomalous behavior is detected, the action module is invoked to maintain the normal operational performance using a pre-selected policy. In their design, policies that have low impacts on property damage are executed automatically to shorten the protection time, while policies with high impacts (e.g., safety) must be executed by trained human operators.

None of the "state-of-the-art" self-protecting approaches discussed above has been fully realized as a complete security design based on the autonomic computing concept developed by Kephat et al. [73]. Using their definition, a fully autonomic computing system must assess risks, anticipate upcoming attacks, prevent the system from being compromised by anticipating attacks, detect both known and unknown attacks, learn the signatures of unknown attacks, and actively respond to attacks. Our ASM approach address all of these properties. The system contains monitors, an IDS, and an IRS. Additionally, our self-protecting system can also be switched between fully-autonomous and semi-autonomous modes for the case when responses are likely to prevent high impact calamities (e.g., financial loss, property damage, and safety). High stakes calamities will enable man-in-the-loop intervention. The ASM framework also includes forecasting modules, a network forensic analysis module (NFAM), and a MAC. The forecasting modules proactively anticipate up-

[1]Reset_Output Attack: Attackers maliciously write 0 to all output registers.

coming attacks using historical data and a physical model of the network infrastructure. In this way, known attacks can be rapidly eliminated before they compromise the control network. The NFAM is used to examine causes and impacts of unknown attacks to enable novel attack signatures to update detection algorithms and the active response library. Accordingly, similar attacks can be estimated and protected in the future. The MAC, a key component, is used to evaluate and select the most appropriate responses to protect against cyber attacks.

Compared with the Arizona team [47], our self-protecting SCADA system is a complete security design that proactively anticipates upcoming attacks, detects real-time cyber assaults, investigates unknown attack signatures, and updates existing algorithms so that normal infrastructure operations are maintained and SCADA system security is enhanced with little or no human intervention.

We will now review the literature concerning techniques, theories, and tools relevant to IDSs, network forensics analysis, and IRSs. All are essential to realizing a self-protecting system within our ASM framework.

2.3.4 Intrusion Detection Systems

An intrusion detection system (IDS) is a data mining tool used to identify cyber attacks. Besides quickly identifying attacks, it has many other benefits such as enabling the collection of intrusion information, recording malicious events, generating reports, and alerting system administrators by raising an alarm.

Recent literature shows that there are several challenges to overcome for today's IDSs to achieve their goals. The first challenge is the slow speed at which IDSs detect anomalies, especially for signature-based IDSs that match observations to misuse patterns. The second challenge is that most prevalent IDSs cannot detect or adapt automatically to novel zero-day attacks. Most existing IDSs deployed in DSs use a signature-based technique. Known attacks can be easily classified based on attack patterns but not zero-days. On the other hand, by adopting the "anomaly detection technique", IDSs can detect both known and unknown attacks. However, without proper classification, deploying accurate counter measures is impossible. The third challenge of current IDS research is the high rate of false alarms. An IDS is an important component in a comprehensive security design, and many security solutions cannot be deployed without an accurate attack detection capability. Moreover, inaccurate detection will adversely affect response selection by the IRS. The obvious result will provide attackers with greater opportunity to compromise the whole system. The following paragraphs overview recent research addressing these three challenges.

The bottleneck of detection speed often plagues signature-based IDSs. This type of IDS must match traffic patterns to a list of rules. The detection speed is influenced by the length of the shortest pattern (LSP) of rules and the memory access latency [81]. To accelerate the detection speed, recent research has deployed a variety of fast pattern matching algorithms such as the Modified Wu-Manber (MWM) algorithm [42], the Set-wise Boyer-Moore (SWBM) algorithm [59], and the FNP algorithm introduced in [81]. Tan et al. [117] extracted partial sample strings from patterns of attacks to enhance the implementation speed.

Using modern graphics processors is another efficient method to address the first challenge. Gnort is an example that yields a detection speedup from two to ten times [122]. Researchers demonstrated that a vulnerability specification and its optimized architecture called VESPA [123] can reduce the time to detection. VESPA uses design principles like fast primitive matching, explicit state management, and parsing of relevant message parts.

The second challenge of today's IDSs is the inability to detect zero-day attacks or classify known attacks. Bolzoni et al. [35] demonstrated an IDS called Panacea, which integrates anomaly-based with signature-based IDSs using machine learning methodologies to address these challenges. The adoption of Support Vector Machine and RIPPER Rule Learner techniques allows Panacea to classify known and unknown attacks. Boggs et al. [34] examined user-defined content from HTTP requests to detect zero-day attacks across multiple web servers. The abnormal requests were identified by local detection sensors. This approach is very efficient at detecting previously unseen attacks with a false positive rate of only 0.03%. However, this method only examines HTTP attacks and cannot detect attacks that exploit vulnerabilities of other network protocols (e.g., UDP, TCP, ICMP and DNS).

Amann et al. [29] presented an input framework that integrates external information into the IDS decision process. External information is the third-party intelligence that includes known botnet servers, malware registries and blacklists. Frequently updating external intelligence does not cause a delay in detecting anomalies; however, this framework may bring additional security concerns from attackers who game the system. Artificial immune systems inspired by their biological counterpart have been applied to scan network traffic and identify zero-day attacks. Danforth [48] developed a Web Classifying Immune System (WCIS) employing an artificial immune system, which achieved a high rate of accuracy for identifying and classifying SQL injection attacks, XSS attacks, and buffer overflow attacks.

The third challenge of IDSs is how to improve the accuracy of the detection rate (i.e., reduce false positive/negative). In a large-scale distributed system, the choice and placement of detectors influence the accuracy of the overall detection function. Modelo-Howard et al. [87] used a Bayesian network model to evaluate the effects of detector configuration and placement against the accuracy of the detection rate and the likelihood of an attack goal being achieved. Selecting the correct features to monitor is another key to improve detection rates. Chu et al. [44] built a user behavior model adopting statistical methods based on device access logs to catch anomalies and protect the networking infrastructure. The user behavior model contains several aspects that can best distinguish normal activities from attacks. These aspects contain failed login attempts, login access patterns, and user behavior. Researchers have shown that the IDS with log analysis approach effectively identified potential intrusions and misuses with an acceptable overall alarm rate. Continuous tuning of the intrusion detection model can be used to enhance the accurate detection rate, and Yu et al. [132] has presented a case in point. The paper introduces the ADAT IDS that uses a buffer for holding suspicious predictions. Only the predictions whose confidence values are higher than pre-defined thresholds are sent to the system operators for identifying false predictions. The feedback is then used to tune the detection model so that misclassified attacks are dropped. resulting in higher detection accuracy.

2.3.4.1 IDSs in Industrial Control Systems Environment

The IDS is also a popular research topic for enhancing ICS security. Recent research focuses on the development of anomaly-based and signature-based IDSs to identify SCADA-specific attacks. The anomaly detection technique comparing real-time system performance with the normal system behavior is considered better than signature-based IDSs for detecting known and unknown attacks in industrial environments [62]. Yang et al. [50] presented an anomaly-based IDS using an auto-associative kernel regression (AAKR) model and a statistical probability ratio test to detect four types of denial of service (DoS) attacks as well as insider attacks. They first collected 62 variables, and after analysis, only five variables were adopted as anomaly indicators since these variables dominate the detection accuracy. The researchers concluded, based on their experimental trials, that their approach can generally be used to detect common SCADA attacks with a high accuracy rate.

Düssel et al. [57] extracted attack features using a sliding window approach over incoming byte sequences. Their results have established a payload-based anomaly detection IDS to detect both known and zero-day SCADA-specific attacks. The researchers validated their approach using two datasets. One contained 1,000,000 HTTP packets of 42 attack instances exploiting 11 vulnerabilities, and the second dataset was composed of more than 822,000 TCP packets covering 19 attack instances exploiting eight different vulnerabilities. The detection rate of this approach was 88%-92% for detecting unknown attacks. This sliding window IDS is considered a useful method to detect SCADA attacks. Linda et al. [79] constructed an anomaly-based IDS using a neural network based modeling algorithm. This IDS generated no false positives on detecting previously unseen malicious real-time traffic.

Snort [20] is a widely used signature-based IDS for detecting and classifying SCADA-specific attacks using pre-defined rules. Yang et al. [130, 131] proposed a set of snort rules to detect attacks targeting the IEC/104 protocol. Their rules can detect both known malicious activities and the source so that, similar attacks will be prevented. Oman and Phillips [93] generated snort rules based on attacker IP address, pre-defined legal commands, and malicious telnet ports to detect unauthorized accesses to SCADA devices. Even though their signature-based IDS can identify known attacks and provide attack signatures to intrusion response systems, the shortcoming of their approach is obviously, their inability to detect zero-day attacks.

From the literature review, we can summarize that an IDS that quickly detects and classifies known and unknown attacks with a high detection accuracy is one of the most essential components of a self-protecting system. The output of this module provides the security state of the system. If the system is compromised, network forensics analysis must run to learn unknown attack signatures, followed by internalizing the necessary responses needed to recover system performance and defend against such attacks.

2.3.5 Network Forensics Analysis (NFA)

Network forensics aim at finding out causes and impacts of cyber attacks by capturing, recording, and analyzing of network traffic and audit files [96]. NFA helps to characterizing zero-day attacks, and has the ability to monitor user activities, business transactions, and system performance. As a result, attack attribution, what attack methods and tactics were used and attack duration, can be analyzed and derived using NFA.

Table 2.1 lists all of the self-protecting systems and IDSs described in Subsections 2.3.3 and 2.3.4. Recent articles claim that self-protecting systems are able to mitigate attack impacts by "static" IRS. Protection mechanisms selected by this type of IRS remain the same during the entire attack period [106]. None of these self-protecting systems, however, contains NAF. Notwithstanding, a static IRS is limited in its ability to fully and efficiently eliminate these attacks. To establish a fully autonomic computing system, as described in Subsection 2.3.2, the NFA must be activated once unknown attacks have been identified by the IDS. Some recent research performs the NFA as a post-process analysis, but live

NFA is required to support the 24/7 system availability. In this way using live NFA, the investigation of unknown attack signatures will not disrupt normal client requests. Additionally, live NFA is executed immediately after unknown attacks are detected, which reduce latency and quickly closes the window of vulnerability.

Philli [96] illustrated a generic process model for network forensics in real-time and post attack scenarios. The first phase is preparation, which makes sure security solutions such as monitors, firewalls, and IDSs have already been deployed. The second phase is detection, which identifies cyber attacks and captures network traffic, followed by the third incident response phase. The fourth phase is collection and preservation. Traffic data are collected and pre-processed for storage on backup devices for presentation. Together these phases identify attack indicators, classify attack patterns, determine attack paths, and provide documentation offline.

The challenges of next generation state-of-the-art NFAs are described by Philli [124]. The analytical complexity of real-world large scale DSs grows exponentially, therefore, making full NFA computationally infeasible. In some NFA scenarios accurate vulnerability states are not available by virtue of the fact that the NFA process is fundamentally post-compromise. Also, analyzing extremely large amount of auditing data in very short time space is computationally infeasible. Additionally, in NFA it is often very difficult to distinguish primary attack signatures from trivial signatures. Let's examine today's live NFA tools.

Investigating high-volume auditing data in real-time depends primarily on the abundance of computational resources. Normally, the size of one-days worth of auditing files for large scale systems is larger than 100 GB. Data concentrators are used, for example, by data historians within the context of the distribution systems (i.e., the smart grid). Cloud storage platforms are, used to store and process such data. Chen [41] used the cloud storage and computing platform to analyze offline phishing attacks. Phishing filter functions can effectively scale to detect phishing attacks. Research described in [119, 32] adopts the Hadoop map reduce model to analyze a high volume of log files and extracts attack signatures for intrusion detection and prevention.

Contemporary NFA approaches manually investigate attack patterns after DSs are compromised. Real-time and dynamic forensics systems, which investigate attack causes and impacts automatically, have been deployed to address the scalability issue of such manual processes. The utilization of the cloud-based approaches makes NFA more practical to deploy. Nonetheless, the current question about how to realize a dynamic and automatic NFA system is the subject of much research. For example, Wang et al. [124] developed immune agents, which automatically generate crucial evidence of unknown attacks for rapid active responses. An automatic method was proposed by Wang et al. [125] to make regular expression signatures of the HTTP attacks offline. This approach first extracts application session payloads to identify common substrings and their positional constraints. Using these extracts, attack signatures are transformed into regular expressions.

Li et al. [78] designed a network-based signature generator, namely LESG. The generator automatically analyzes zero-day buffer overflow vulnerabilities and generates length-based signatures via a three-step algorithm. The generator selects field candidate signatures, optimizes the signature lengths, and derives the optimal signature that produces minimal false alarms. The LESG has lower computational overhead than research by Brumley et al. [38], which generates vulnerability signatures using regular expressions. False alarm rates of an IDS applying the LESG are significantly reduced and the attack detection speed is improved. The LESG is also tolerant to noisy traffic.

The utilization of NFA allows the self-protecting system to investigate and learn the causes and impacts of an unknown attack. In the end, the DS can be protected from both known and unknown attacks autonomously (i.e., without human intervention).

2.3.6 Intrusion Response Systems

An intrusion response system (IRS) is a critical part of the self-protecting system for ensuring appropriate responses are dispatched to react to protect the DS and recover system performance back to normal. The tight interaction between the NFA module and the IRS module makes future attacks, which have similar signatures, less likely to succeed [33]. The development of an autonomous IRS focuses on two key elements: configuring suitable responses and evaluating recommended responses dynamically.

Responses are protection mechanisms that have the ability to eliminate or mitigate cyber assaults. Efficient responses include “dropping malicious requests,” “isolating compromised nodes,” and “shutting down hosts.” IRSs that integrate recommended responses provide a mapping of alerts from the detector of an IDS to a response. A static IRS constantly assigns responses to attacks whether or not they are optimal. Snort_inline [20] is an example of a static IRS. New and dynamic responses involve attack classification, attack characteristic analysis, and optimal responses selection. One instance of a dynamic IRS is ADEPTS [60, 128], which evaluates responses with respect to a local optimal criterion and limits the size of response plans. The following paragraphs survey the protection mechanisms that are able to eliminate cyber attacks, followed by the study of dynamic IRSs and their evaluation methods.

Protection mechanisms, which are configured and implemented to defend the system against specific cyber attacks, vary according to the types of cyber attack they target. In other words, responses that can mitigate DoS attacks are not efficient in removing SQL injection attacks. Five categories of web attacks are illustrated in Section 3.3: scanning attacks, spoofing attacks, injection attacks, broken authentication and session management, and DoS attacks. In the subsequent paragraphs, responses that protect web applications from these attacks are reviewed.

Scanning attacks are conducted by adversaries to discover vulnerable services. Port scanning, IP addresses scanning, and version scanning attacks are discussed in Section 3.3. Techniques such as TCP scanning, UDP scanning, and ICMP scanning gather the IP addresses of victim hosts looking for open ports. To protect DS from scanning attacks, packet filters are placed in the victim system to discard scanning packets. Transparent proxies, which hide actual open ports from external users, are also somewhat effective mechanisms to defend the system from scanning attacks.

A successful spoofing attack will utilize legitimate identity credentials to enable or bypass authentication mechanisms and unauthorized remote system access. Securing Internet Protocol communications by authenticating and encrypting IP packets is a promising defense measure (e.g. using the Internet Protocol Security (IPsec)). Another example is Secure Socket Layer and Transport Layer Security (SSL/TLS), which uses public key infrastructure to secure transport layer transactions between clients and servers [105, 94, 72]. Mechanisms like secure remote password protocol and dynamic security skins are generally used by transaction based web sites to defend against authentication attacks [53, 116]. In addition, both access control lists, which block communications between malicious clients and servers, and packet filters, which reject spoofing attack packets are useful spoofing countermeasures.

Injection attacks insert illicit data into web applications and force victim systems to execute undesirable codes or commands and spread malware. One of the best methods to protect from injection attacks are “secure coding best practices.” The Open Web Application Security Project (OWASP) lists twelve secure coding practices such as input validation, output encoding, and file management checklists [9]. Limiting web application coding privileges, reducing debugging information, and testing web applications regularly can also mitigate injection attacks [8].

Brute force attacks and man-in-the-middle attacks are broken authentication and session management attacks, which exploit flawed credential management functions. Once attackers successfully break authentication mechanisms or hijack active sessions, it may be possible for them to gain access to all accounts and do anything legitimate users could do. To protect against broken authentication and session management attacks, a simple but efficient method is to block IP addresses. Another simple and effective practice is to create and follow security guidelines for choosing strong passwords. No default usernames and/or passwords are allowed. Disabling remote logins via the secure shell and creating a large number of honeypot accounts in financial systems is a common practice used to mitigate brute force attacks [69, 95, 100].

Flooding-based DoS attacks as discussed above (Subsection 3.3.5), aim at impeding legitimate clients from accessing host/server resources. There are numerous traditional mechanisms the host can employ to defend against DoS attacks. Mudhakar et al. [92] used a twofold mechanism to filter application layer DoS attacks. This approach uses hidden ports to filter illegitimate packets. Then web server resources are allocated to admit clients based on priority levels assigned by the congestion control element. This protection mechanism works well for external DoS attacks, but it is ineffective in eliminating internal DoS attacks (i.e., compromised internal nodes initiate the flood). Performance overheads from applying this approach are very high resulting from the necessary adjustments needed to employ hidden ports and to monitor/manipulate client priority levels.

Stavrou et al. [109] introduced WebSOS, an adaptation of the Secure Overlay Services (SOS) architecture that only allows legitimate users to access web servers when DoS attacks have compromised the server. Patil and Kulkarni [85] proposed a lightweight mechanism that uses trust to differentiate between legitimate users and attackers who launch HTTP flooding attacks. However, it is unclear whether this mechanism will reduce high utilization of web server resources caused by application layer DoS attacks.

The first key element of an IRS is selecting and configuring protection mechanisms. Any of the above mentioned mechanisms can eliminate or at least mitigate the variety of potential attack tactics. The second element is a way to evaluate these candidate response mechanisms for choosing the most appropriate mitigation/recovery response. The most effective response, however, may not protect a system from other attack types. To accurately evaluate these responses, Luo et al. [83] designed a game-based multi-stage intrusion defense system, which allows administrators to evaluate which type of responses are optimal with respect to different measures (i.e., selectivity, rapidity, and etc). The optimal responses are the ones with the highest utility values based on administrator estimations of expectations and variance. Another useful evaluation method is to employ cost-sensitive modeling to select optimal responses. This method accounts for the balance between an estimated cost of potential damage versus the cost of implementing a particular set of responses [71, 75].

Dewri et al. [52] addressed a multi-objective optimization problem for how to secure systems with limited budgets. The researchers created a system attack tree model to evaluate measures based on total security control costs and residual damages. This approach provides concrete options to the system administrators about the quality (i.e., cost) of the different trade-off solutions. However, system administrators are not allowed to modify their decisions during run-time. Stakhanova et al. [107, 114, 113] evaluated finite responses for three criteria: response system impacts, response goodness, and response operational costs. Their response evaluation relies on expert knowledge, but the evaluation cannot distinguish which action is the best if all responses have similar descriptions.

Designing dynamic IRSs must account for installing and configuring suitable protection mechanisms as well as selecting corresponding evaluation methods. The dynamic response system is an essential module in a self-protecting system that actively reacts to (thwarts) attacks after they are detected by the IDS. This approach reduces the time gap between attack detection and response, which reduces the (time) window of opportunity for the attacker.

2.4 Validating the Self-Protection Functionality

Research concerning the state-of-the-art for self-protecting systems focuses on the most significant modules and techniques namely IDSs, NFA, and IRSs. Table 2.1 classifies these related works into six categories including protection goals, self-protection level, decision making, protection tempo, normal operation, and complexity. These five categories represent the functionality of a self-protecting system that autonomously and proactively identifies and responds to known and unknown attacks.

The protection goal category research is divided into techniques that only detect and/or respond to known attacks and techniques that can defend against both known and unknown attacks. These self-protecting systems should integrate risk assessment, intrusion estimation, intrusion detection, network forensic analysis, and intrusion response modules. There is no published research at this time that fully integrates all five modules. The self-protection level category contains subclasses including "monitor and detect," "forensic analysis," and "response." The risk assessment level category is out of scope for this research because expert knowledge is required to characterize systems and identify vulnerabilities as well as selecting which features to monitor.

The decision making category is important for a fully autonomic or semi-autonomic system. Fully autonomic systems protect themselves from cyber attacks without human intervention, while semi-autonomic computing systems require expert information to help in analyzing attack signatures and in selecting the appropriate and most effective responses. Both types, fully and semi-autonomic depend on IDS and NFA to play an important role detecting and learning the cause and impact of each attacks. In what follows, we use the terms "automatic" and "manual" to designate the respective decision making category of autonomic and semi-autonomic.

The protection tempo category is a measure depending on which approach is invoked to proactively defend against attacks, concurrently (as it happens), or as postpartum (after the fact). Ideally self-protecting systems are evaluated based on their ability to maintain normal operations. A further criterion would be the complexity of implementing the framework in real-life system.

In comparing this thesis with the recent literature shown in Table 2.1, we find the proposed approach is both easy to employ for monitoring and detecting, as well as respond to both known and unknown attacks. Further, the ASM approach not only proactively estimates the upcoming known attacks, but also detects and mitigates unknown attacks in real-time. Unknown attack signatures are stored so that offline analysis of attack patterns is practical. Our approach also makes it possible to switch the proposed self-protecting system from a fully autonomic mode to a semi-autonomic mode. Even if the controller makes a wrong decision to deploy an ineffective active response, the system performance can still be recovered back to normal by system administrator intervention.

2.5 Summary

To summarize, this chapter reviewed research motivations, introduced the multi-tier client-server architecture, and presented the primary security challenges of the DS that adopts this client-server architecture. Related scholarly work concerned with autonomic computing technology, self-protecting systems, intrusion detection techniques, NFA, and

Table 2.1

Classification of Related Works

Solution Source	Protection Goals		Self-Protection Levels			Decision Making		Protection Tempo			Normal Operations		Complexity		
	Known Attacks	Unknown Attacks	Monitor and Detection	Forensics Analysis	Response	Fully-Autonomous/Automatic	Semi-Autonomous/Manual	Proactive	Online	Offline	Disruption	Maintenance	Easy	Medium	Hard
Thesis: The Autonomic Security Management Approach	×	×	×	×	×	×	×	×	×	×		×	×		
JADE [45], Qu et al. [97], VESPA [123]	×	×	×		×	×			×		×			×	
UBL [51]	×	×	×			×			×			×		×	
QoP [68], Gelenbe et al. [63]	×	×	×		×	×			×			×	×		
SPIDER [66]		×	×		×	×			×			×		×	
Cox et al. [47]	×	×	×		×	×	×		×	×		×		×	
FNP [81]	×		×			×			×	×	–	–		×	
MWM [42], SWBM [59], Tan et al. [117]	×		×			×			×		–	–		×	
Gnort [122]	×		×			×			×			×		×	
Panacea [35], Boggs et al. [34], Input Framework [29], WCIS [48]	×	×	×			×			×		–	–		×	
Chu et al. [44]		×	×	×		×			×		–	–		×	
ADAT [132]	×		×				×		×		–	–		×	
Yang et al. [50], Düssel et al. [57], Linda et al. [79]	×	×	×			×			×		×		×		
Snort [20], Yang et al. [130, 131], Oman and Phillips [93]	×		×			×			×		×		×		
Philli [96]	×	×	×	×			×		×	×		×			×
Wang et al. [125]	×	×		×		×				×		×		×	
LESG [78], Brumley et al. [38]		×	×	×		×			×			×	×		
Immune Agent [124]	×	×	×	×		×			×			×		×	
Chen et al. [41]	×		×	×	×				×		×		×		
Bhatt and Yano [32], Piromsopa et al. [119]	×	×	×	×		×			×			×		×	
ADEPTS [60, 128]	×	×	×		×	×	×	×	×			×			×
Luo et al. [83]	×				×		×	×	×			×		×	
Ikuomola [71]	×		×		×	×				×		×		×	
Strasburg [114, 113]	×		×			×	×		×			×		×	
Stakhanova et al. [75, 107]	×		×		×	×				×		×		×	
Dewri et al. [52]	×	×			×	×	×		×			×			×

intrusion response systems was surveyed. Furthermore, we compared our research with the open literature to illustrate the benefits of adopting our approach.

CHAPTER 3

SECURITY CHALLENGES IN DISTRIBUTED COMPUTING AND INDUSTRIAL CONTROL SYSTEMS

As discussed in Chapter 2, even if organizations design their internal networks with the adoption of DMZs following recommended guidelines for securing web servers and Industrial Control Systems (ICS) [120, 112], perpetrators may still have the opportunity to compromise these systems. For example, as shown in Figure 2.1, the external firewall denies all traffic except HTTP and HTTPS messages; therefore, malicious users could exploit HTTP and HTTPS protocol vulnerabilities to compromise web servers by using flooding-based HTTP and XML denial of service (DoS) attacks. Using this scenario, adversaries aim at slowing down web server performance thereby blocking legitimate client requests and undermining information availability; the so-called denial of information attacks. The application tier can also be compromised using IP spoofing attacks, in which circumstance the attackers can evade the firewall to remotely access application servers. The goals of these attacks include exfiltration of confidential information and/or undermining data and code integrity. The SQL injection attacks allow adversaries to access database servers without necessarily possessing a valid username or password. These attacks typically result in data leakage of, for example, client personally identifiable information (PII), financial information, employment records, and even trade secrets.

A wide range of taxonomies aim at classifying cyber attacks. By sorting attacks according to the attack classification inherent characteristics of these taxonomies provides system administrators with just the essential and necessary information needed to detect and mitigate attacks. Recent research classifies cyber attacks by studying attack vectors, targets, purposes, scopes, information impact, and defense strategy [43, 76, 40, 70, 121, 39, 129, 67].

In the next section, cyber attack scenarios against the web application and SCADA systems are introduced along with their inherent characteristics and constraints including actors, motivations, goals, attack mechanisms, and adverse effects.

3.1 Related Work

Attack vectors are paths by which attackers gain access to computing systems and networks for delivering a malicious payload. Simmons et al. [43] considered vulnerabilities exploited by successful attacks, and classified cyber attacks into nine categories: misconfiguration, kernel flaws, buffer overflows, insufficient input validation, symbolic links, file descriptor, a race condition, incorrect file/directory permission, and social engineering.

Similar to Simmons's classification, Heerden et al.'s classification dimensions depend on attack mechanisms [121]. Assaults are categorized into the *access class*, the *data* *manipulate class*, and the *information gathering class*. The access class contains brute force, buffer overflow, spear phishing, and physical attacks; the data manipulate class is divided into network-based (e.g., denial of service attacks), virus-based (e.g., trojan, virus, and worm), and web application-based attacks (e.g., SQL injection and cross-site scripting attacks). Information gathering involves scanning and physical attacks. Although researchers emphasize that their taxonomy observes mutually exclusive requirements, certain attacks can be classified into multiple categories. For example, "physical attacks" belong to both the *access class* and the *information gathering* class.

3.1.1 Attack Targets

Hansam and Hunt classified attacks depending on their targets [67]. These targets can be operating systems, network, local, user, or web applications [43]. The targets can also be classified according to hardware, software, and network categories [121]. Web services like governmental websites, financial institution websites, online discussion forums, news and media websites, as well as military/defense networks websites [84] are the target sectors classified by Kjaerland [76].

3.1.2 Attack Purposes/Goals

This class indicates the objectives of the attack actors for launching (stepping-stone) assaults. Simmonds et al. [103] classified attack goals based on their potential to damage information confidentiality, integrity, availability, and authentication. Heerden [121] refined the classification of attacks by a further refinement based on attack goals, namely, goals to change, destroy, disrupt, and exfiltrate data. The attack purposes presented by Uma and Padmavathi include reconnaissance, access, and denial of service type attacks [84]. Reconnaissance attacks belong to the "phase class" referred to in the attack stages discussed in [121].

3.1.3 Attack Scopes

The size of attacks and the targeted organizations are two categories for the attack scope category. The attack scope for targeted organizations can be categorized into the corporate network, government network, and private network [121]. By calculating the area of nodes that have been compromised [65], attack scopes can also be divided into malicious large scale and non-malicious small scale [84] or local, global, and hybrid [61].

3.1.4 Information Impact

As stated above successful cyber attacks have the potential to affect sensitive information. For example, attacks may cause file distortion and destruction, service disruption, and information exfiltration, discovery and disclosure [43, 76]. Information impacts can also be defined as financial and be distinguished by differing degrees of property damage such as minor, major, or catastrophic [121, 86, 37].

3.1.5 Defense Strategy

Defense mechanisms include differing attack mitigation strategies. Attack recovery remediation strategies may be used as well to classify cyber attacks. Simmons et al. [43] further classified mitigation techniques into three subclasses: elimination of infected hosts from networks, using a whitelist/blacklist strategy to block adversaries, and the adoption of reference advisement (e.g., United States Computer Emergency Readiness Team (US-CERT) vulnerability notices [11]). The application of software patches is also a remediation technique presented by Simmons et al. Additionally, effective remediation techniques also include filtering malicious packets, employing IDSs to monitor and detect malicious behavior, and the use of secure communication channels (e.g., TLS and SSL crypto protocols) [2].

3.2 A New Cyber Attack Taxonomy

In this section, we develop a new cyber attack taxonomy to classify attacks based on their impact to on system resources, network resources, and security-specific resources. Each categories contains several dimensions. These dimensions are “selected features” monitored by sensors, and are used to determine whether the system security state is “normal” or “abnormal.” Details of the selected features are discussed in Chapter 4.

3.2.1 Impact on System Resources

This dimension discusses the effects to operating system resources caused by cyber attacks and is classified to three subclasses:

- CPU Time: Provides a measure of time that the processor works using threads to serve requests. A cyber attack may result in an abnormal increase or decrease of CPU Time on compromised servers, which in-turn affects user wait time.
- Memory Utilization: Indicates the amount of physical memory resources used or held by Kernel level services or user processes. Some attacks lead to memory leaks that aim at exhausting available memory resources of the compromised host, while some attacks maliciously terminate processes to disrupt or destruct underlying services of web applications.
- I/O Read/Write Rate: This subclass shows process read and write rates caused by I/O. Malicious requests disrupt file systems by illegitimately writing or hogging the physical disk which in-turn disrupt the rate of legitimate I/O.

3.2.2 Impact on Network Resources

This classifier filters based on the attack's impact on network resources. General network attacks exhaust the bandwidth of servers to disrupt legitimate communications.

Therefore, cyber attacks may evoke extremely high throughput causing excessive TCP connection rates.

- Packets Received/Sent Rate: Cyber attacks send flooding requests to the web server, and sharply increase the number of packets that are received or sent per second to/from the compromised network interfaces which in-turn affect the rate of received/sent packets.
- Total Byte Received/Sent Rate: Similar to the packet rate, cyber attacks easily influence the total bytes received and sent on network interfaces.
- TCP Connection Performance: As mentioned above, most Internet application protocols are based on the TCP/IP model [118]. Consequently, the TCP connection performance of each server reflects attack intensity. TCP connection performance contains connection failures, which show the number of half-open TCP connections and include segment received/sent rates, which in-turn determine TCP segment received and sent rates.

3.2.3 Impact on Security Resources

This classifier reflects the security of web applications, which can be used to identify normal from abnormal services. The impact on security resources contains the following dimensions:

- Permission: This dimension measures the failures of file open or data access functions. Attackers may gain unauthorized user privileges or administrator privileges to access confidential data. Adversaries who obtain such high privileges can also undermine the integrity of the database servers by modifying, deleting or adding malicious data. Actors that bridge the maximum number of access attempts are normally deemed to be attackers.

- Login Rate: This parameter counts how many times users try to login to the servers (including parameters such as the number of logins to the server per user per second and the total number of login attempts). Login Rate is a key parameter to identify brute force attacks and attacks that try to block legitimate client accounts.

- Packet Header: The Internet protocol version 4 (IPv4) messages are composed of packet headers and data. A TCP/IP message header contains source and destination IP addresses. It also includes source and destination port numbers.

- Payload: Requests sent from users to the web servers contain malicious commands that are threats to the web application security. These attacks can be easily distinguished from normal network traffic by examining of packet payloads.

Section 3.1 and 3.2 has presented eight classifiers to characterize the nature of attacks, which are the classification by attack target, attack purpose or goal, attack scope, information impact, defense strategy, or by their impacts on system resources, network resources, and security resources. The last three classifiers, classification by impacts on system resources, network resources, and security resources, are unique classifiers determined by analyzing attack signatures. An intrusion detection system is a component of a self-protecting mechanism, which detects and classifies attacks based on these three classifiers.

3.3 Analysis of Cyber Attacks in Distributed Computing and Industrial Control Systems

In this section, web application and SCADA security risks are analyzed and classified based on their effects to system resources, network resources, and security resources.

3.3.1 Scanning Attacks

Adversaries scan devices in the computing network to gather meaningful information before launching sophisticated attacks to undermine security. Commonly used scanning techniques to gather computer network information including IP address scanning, port scanning, and version scanning. For gathering ICS information, adversaries always carry out function code scanning (FCS) attacks to learn vendor names, product codes, and revision numbers of SCADA devices [88].

1. IP Address Scanning: This attack may allow adversaries to discover how many servers are placed in the network and their respective IP addresses. When the adversaries obtain the IP addresses of each device, they can further discover type of the services, applications and operating systems running on these devices.

2. Port Scanning: After attackers obtain the IP addresses of devices in the web service system, they may scan the open server ports to determine whether vulnerabilities of available applications can be exploited.

3. Version Scanning: Attackers can even collect operating system, services and applications types and versions by launching version scanners. This information may be utilized in the future to launch successful DoS attack and authentication attacks.

4. Function Code Scanning: In the SCADA environment, attackers launch a FCS attack by sending a query with public function codes, user defined function codes, reserved function codes, and error function codes [88] to a remote terminal unit (RTU). If the FCS attack is successful in gathering information from the ICSs, the attackers can carry out disruptive attacks in the future exploiting function code vulnerabilities.

The effects of all four types of scanning attacks to system, network, and security are now discussed.

- Impact on System Resources: Scanning attacks do not directly impact this criterion. However, they do just increase the percentage of wasted CPU cycles to the victim server or host.

- Impact on Network Resources: The dimension of *TCP Connection Performance*, especially *connection failures* increase sharply caused by scanning attacks.

- Impact on Security Resources: Scanning attacks either increase the *Login Rate* or the rate at which actions illegitimately obtain *Permission*. These attacks undermine the security of the computing environment. For most server ports, the payloads of the scanning packets sent from adversaries to the victim hosts are empty. This specific signature can be used to identify scanning attacks from normal client requests.

- Attack Goal: Scanning attacks aim at gathering and preparing information to advance further successful attacks.

3.3.2 Spoofing Attacks

Spoofing attacks happen when adversaries pretend to be legitimate users. Masquerading and impersonation are two types of spoofing attacks. Masquerading is a passive attack. Attackers first exfiltrate legitimate account credentials and then log in to the computing system as legitimate users. Impersonation is also known as replay attacks. This spoofing attack is more active than masquerading. Attackers capture authentication traffic and replay that traffic to gain access to the computing system. Once adversaries control the compromised computing system, they can exfiltrate confidential data, exhaust system resources, or spread malware to compromise other networks. Internet Protocol (IP) address spoofing and Address Resolution Protocol (ARP) spoofing attacks are examples of spoofing attacks that evade the authentication process/step of web applications.

5. IP Spoofing: Adversaries disguise themselves by sending malicious packets from false source IP addresses to victim hosts. IP spoofing attacks allow attackers to evade IP address based authentication.

6. ARP Spoofing: Adversaries link their media access control addresses with legitimate IP addresses; therefore, messages sent between victim servers and legitimate clients can be intercepted by the attackers.

The analysis of spoofing attacks and their effects to system, network, and security are as follows:

- Impact on System Resources: Null.
- Impact on Network Resources: *TCP Connection Performance* may be influenced on the server side since attackers have intercepted legitimate message transmission.
- Impact on Security Resources: This attack does not impact *Login Rate* or *Permission*. However, analyzing packet headers can differentiate masqueraded requests from legitimate packets.
- Attack Goal: Financial/economic and disruptive

3.3.3 Injection Attacks

In the web application context, adversaries can exploit the vulnerabilities of Structured Query Language (SQL), JavaScript, and computer programs to trick the interpreter by inserting untrusted data. As a consequence, adversaries may obtain access to the database, attack web users, and propagate computer worms. On the other hand, in the ICS network, adversaries may inject malicious parameters, malicious commands, or malicious responses to reduce the security of ICSs. In the following section, SQL injection, Cross-site Scripting attacks, and SCADA-specific malicious parameter command injection (MPCI) attacks are studied and analyzed:

7. SQL Injection: SQL Injection, which is the leading cause of data breaches, are the most crucial existing web application security flaw now and for a long time in the past. Attackers exploit the vulnerabilities of the SQL language and insert commands to bypass authentication. In 2008, five attackers successfully broke into servers of a dozen companies using SQL injection methods, and stole a total of 160 million credit card numbers [115].

8. Cross-Site Scripting (XSS): Attackers undermine code integrity in the application tier and deliver malicious scripts to the victims' web browsers. Once victims have executed malicious scripts, user sessions can easily be hijacked by attackers. XSS attacks can also inject malware and redirect users to phishing websites to undermine the security of web servers [19].

9. Malicious Parameter Command Injection: Attackers inject false commands to overwrite SCADA system devices such as remote terminal registers, so that normal infrastructure operations or device communications can be interrupted. Malicious device set points such as proportional-integral-derivative (PID) control parameters,

physical system control modes, and alarm conditions are usually injected by adversaries for successfully launching MPCI attacks to take control over process control systems.

The analysis and classification of injection attacks are as follows:

- Impact on System Resources: Injection attacks do not necessarily impact this classifier, or slightly increase *CPU Time* or *I/O Read and Write Rate* of victim servers.
- Impact on Network Resources: Injection attacks do not disrupt network resources. They may slightly increase *Packet Receive Rate* and *Total Bytes Received Rate*.
- Impact on Security Resources: Injection attacks improperly grant *Permission* to malicious users. *Login Rate* may be increased due to numerous attempts by adversaries to gain unauthorized access to computing systems.
- Attack Goal: Attackers aim to access data without authentication by tricking the interpreter to execute malicious commands contained in the injected data. Financial gains are the main reason for attackers to carry out injection attacks. Although 19% of data breaches described by the 2013 Verizon's investigation report [12] were shown to be espionage based.

3.3.4 Broken Authentication and Session Management

Attackers exploit vulnerabilities in authentication mechanisms to assume legitimate users' identities. A successful attack may compromise all accounts. As a result, adversaries can do whatever legitimate users could do [19]. Brute force, session spotting, replay attacks, and session expiration mechanisms allow attackers to obtain administrator privileges. In the following section, brute force and man-in-the-middle attacks are introduced and concluded by a discussion of the effects and goals of authentication attacks.

10. Brute Force: Systems can be compromised by brute force attacks if they do not establish strong password policies. These attacks take advantage of weak passwords and small encryption keys. Such attacks send guessed values of account usernames, passwords, and credit card numbers to the server. Attackers may repeat sending their guesses until they successfully access system resources from compromised accounts.

11. Man-in-the-Middle (MITM) Attacks: A MITM attack sniffs traffic from legitimate clients to web servers to obtain legitimate clients' session IDs, credentials, usernames, and passwords. Session spotting is the mechanism by which attackers sniff session IDs and credentials, and use this information to access legitimate user accounts.

Attacks that exploit authentication vulnerabilities slightly impact system and network resources of the victim server before adversaries successfully enter the internal computing network. The successful attempt allows malicious users to pilfer whatever they like when they obtain system administrator's privileges. The impact analysis in the following list focuses on bypassing the authentication rather than a full control of the compromised system.

- Impact on System Resources: This attack does not affect this classifier, or just slightly increases *CPU Time*.
- Impact on Network Resources: This attack does not impact network resources, or just slightly increases *Packet Received/Sent Rate* and *TCP Connection Performance*.
- Impact on Security Resources: *Login Rate* may be increased due to numerous login attempts from adversaries before the success. *Permission* is granted improperly so that malicious users are allowed to access privilege accounts.
- Attack Goal: Financial/economic or espionage.

3.3.5 Denial of Service (DoS)

DoS attacks are intended to make resources unavailable using various methods and tactics to exhaust system and network resources. One example is flooding-based DoS attacks that send a tremendous number of packets to swap web servers. In this way, bogus requests block legitimate requests by overwhelming the CPU. Attackers can also exploit programming vulnerabilities to initiate the DoS attacks. For instance, infinite loops in programs could decrease server performance. Additionally, attackers block legitimate users from accessing their accounts by launching brute force attacks. In this variant case, the protection mechanisms automatically lock the victim accounts if the failed attempts are greater than predefined thresholds (e.g., 3-5 times). The DoS attack is also a highly publicized threat that attempt to disturb normal operations or functioning of the cyber-physical systems. Several types of DoS attacks including TCP SYN flood, UDP flood, ping of death, and HTTP flood are enumerated here:

12. TCP SYN Flood: Malicious users flood the victims with a large number of half-opened TCP connections exhausting *Memory* resources of the server.
13. UDP Flood: Attackers send a large numbers of UDP messages that aim at exhausting the *CPU Time* of the victim to impede processing legitimate client requests.
14. Ping of Death: Attackers send oversized IP packets (larger than 65,536 bytes) to overwhelm input buffers to crash the victim recipient machine.
15. HTTP Flood: Attackers send HTTP GET or POST requests to a target web server. HTTP flood attacks can exhaust the *CPU Time* of the server since the server is kept busy with processing numerous requests.

DoS attacks impact system and network utilization significantly, and the detailed analysis is as follows:

- Impact on System Resources: Flooding-based DoS attacks consume significant amounts of web servers *CPU Time*. A large number of requests addressed to the database servers can cause memory exhaustion, especially when extensive amount of data are pulled from the database. DoS attacks may also sharply increase *I/O Read/Write Rate* by inserting malicious data into the file system.

- Impact on Network Resources: *Packet Received Rate* and *Packet Sent Rate*, *Total Bytes Received Rate* and *Total Bytes Sent Rate* and *TCP Connection Performance* may be influenced notably due to the tremendous number of requests sent to the victim host.

- Impact on Security Resources: *Login Rate* is increased if attackers aim at flooding legitimate user accounts. However, most DoS attacks cannot typically evade authentication mechanisms which would illicitly grant inappropriate privileges to the adversaries.

- Attack Goal: DoS attacks undermine information availability and disrupt normal delivery of services from the servers.

3.4 Summary

This chapter classifies well-known web application and ICS attacks based on their impacts on system, network, and security resources. Through the analysis, 15 cyber attack patterns and their influences are investigated. Information about attack patterns gathered here are beneficial for crafting self-protecting mechanisms that can estimate, detect, and respond to these known attacks. The same parameters (i.e., dimensions) can be used to investigate unknown attacks using a network forensics analysis module, which has been discussed in Section 2.3.5.

CHAPTER 4

AUTONOMIC SECURITY MANAGEMENT FRAMEWORK FOR WEB APPLICATIONS AND INDUSTRIAL CONTROL SYSTEMS

This chapter presents an autonomic security management (ASM) approach, which is a complete security design integrating intrusion estimation, intrusion prevention, intrusion detection, and intrusion responses, as shown in Figure 4.1. The application of this approach to a distributed system (DS) enhances the security of the system by proactively preventing upcoming attacks, detecting real-time anomalies, actively responding to mitigate attack impacts, and maintaining normal system performance. To better implement the functionality of the ASM framework and to tightly couple the framework modules, a master controller virtual machine (MC-VM) is employed. The essential autonomic components and modules are installed on the MC-VM. Therefore, simply adding the MC-VM to a DS can enable integration of the ASM framework toward self-protection realization.

In the following sections, we present how to develop and apply the ASM framework to strengthen the security postures for a web application and an industrial control system (ICS).

4.1 Autonomic Security Management Approach for a Web Application

Figure 4.2 shows three devices, a router, a main host, and a MC-VM. In this architecture, the router forwards client requests to the internal network. These incoming client requests may include malicious messages that disrupt normal services provided by the multi-tier web application running on the main host. To protect the main host from potential attacks, the router redirects recognizable malicious packets to the MC-VM, which filters known malicious messages. The steps to realize a self-protection function are shown in Figure 4.1. In both Figure 4.1 and Figure 4.2, we see that the forecasting process/module estimates the future environment (ω) and security (λ) parameters of the system using historical observations. Estimated future system security states (identified by the intrusion detection system (IDS)) are then provided to the multi-criteria analysis controller (MAC).

The MAC assesses recommended responses using fuzzy-logic and the Preference Ranking Organization METHod for Enrichment Evaluation (PROMETHEE II) decision making techniques. The appropriate response can then be proactively initiated to defend the network against upcoming attacks. If the forecaster provides a bad estimation then the protection mechanisms cannot efficiently impede upcoming assaults. As a backup, the real-time observations are also delivered to the performance-based IDS (PIDS) to identify known attacks. The proper response can then be deployed from the PIDS to strengthen the security of the DS. On this basis, the application of the MC-VM can easily self-protect the DS from known attacks.

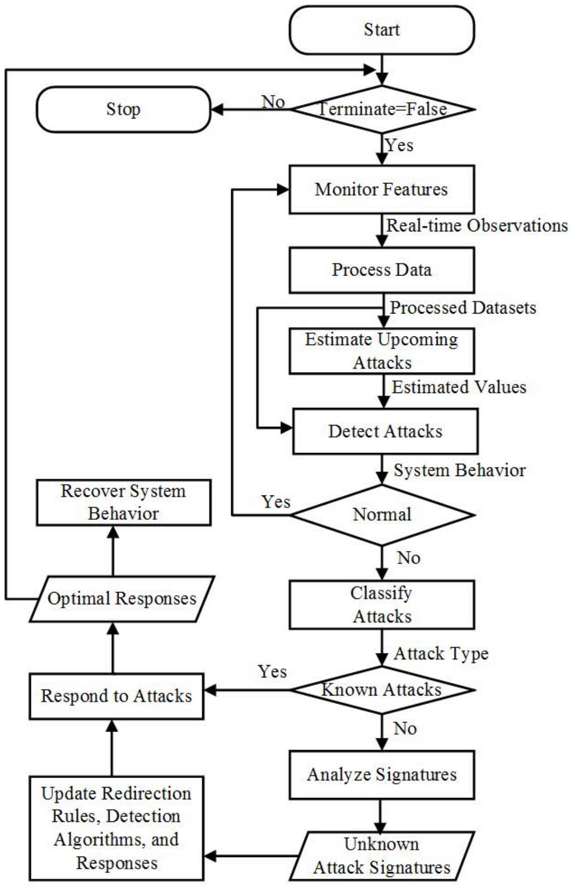


Figure 4.1

Outline of the ASM Approach

To defend the system against unknown attacks, modules running on the MC-VM need to cooperate with the main host and the router. Since unknown attack signatures have never been seen before, the redirection module of the router cannot redirect such attacks to the MC-VM. Thus, unknown attacks get sent directly to the main host. A learning module is used on the main host to investigate causes and impacts of the unknown attacks as well as extract unique attack patterns to be used by the forecasting module, which resides on the main host. The redirection rules of the router are then updated using the new captured attack signatures. Consequently, on the MC-VM, detection algorithms of the PIDS and the assessment of recommended responses by the MAC are dynamically updated with novel attack signatures. Using this scheme, similar attacks will hence be redirected to the MC-VM as known attacks, and thereby eliminating the risk of future penetrations from previously novel attacks.

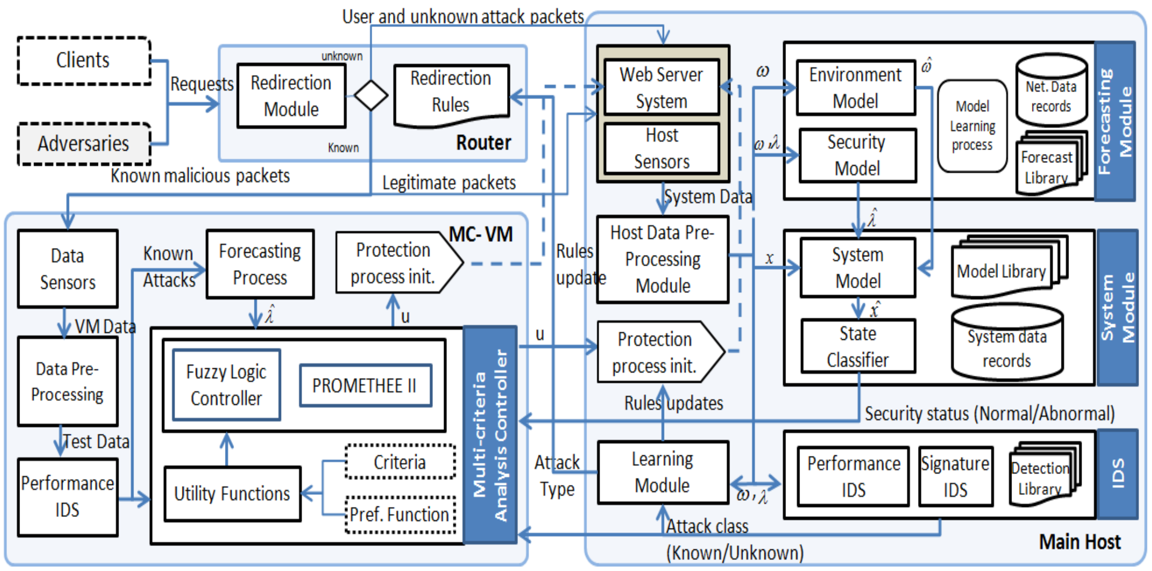


Figure 4.2

Main Components of the Autonomic Security Management Framework

In the following sections, autonomic components and modules running on the router, the MC-VM, and the main host are introduced along with the theories, techniques, and security solutions that have been applied to construct comprehensive self-protecting systems.

4.1.1 Data Sensors

Data sensors are installed on the MC-VM and the main host for collecting real-time observations of system performance, network utilization, and system security states. These observations are used to estimate upcoming attacks by forecaster modules and identify attacks by intrusion detection systems. To this end, the relevant features of these three aspects are first selected offline involving expert knowledge, experimental results, literature reviews, and security guidelines. The real-time observations of these features are gathered and delivered to the data processing module. These features are briefly summarized and listed in Table 4.1.

4.1.2 The Data Processing Module

Real-time observations of certain features monitored by sensors are incomplete under some conditions (e.g., heavy traffic or DoS attack). In this case, the data processing module processes measurements collected by the monitor module. One simple method is to delete the incomplete dataset. The formatted and pre-processed datasets are then forwarded to the intrusion estimation module to anticipate future attacks. Meanwhile, IDSs use these datasets for identifying anomalies on-the-fly.

4.1.3 Forecasting Module

The forecasting module uses recorded historical data (past attack experiences) to predict trends from the system environment and security parameters. To this end, an autoregressive integrated moving average (ARIMA) model is applied to predict environmental parameters (e.g. *packet_recv*) and security parameters (e.g. *memory*, *CPU_idle*, and *Num_of_attacks*). The ARIMA model is simple, and efficient to apply for short term forecasting [134].

The ARIMA method predicts a value in a response time series as a linear combination of previous observations and error terms [13]. The method is represented as a function ARIMA (p,d,q) that describes each observation as a function of previous p observations, derivatives of historical data d, and the correlation with the previous q errors.

Note that we use normal letters to indicate historical observations and hatted letters for estimations. For example, x is the measured performance state of the system, and $\hat{x}$ is the estimated value of the performance state that typically depends on measured values of x at earlier time steps. The ARIMA model has the general form,

$$(1-\sum_{i=1}^{p}\rho_i L^i)(1-L)^d y_t = \mu(1-\sum_{i=1}^{p}\rho_i) + (1-\sum_{j=1}^{q}\nu_j L^j)\epsilon_t \tag{4.1}$$

where L is the lag operator and defined as $L^1 y_t = y_{t-1}$; ρ_i and υ_j are weighting coefficients; μ is the intercept term, which is close to the mean value of the time series; ϵ_t is the forecast expected error of $(y_t - \hat{y}_t)$; and $\hat{y}_t$ denotes a forecast of observation y_t.

The amount of historical data increases as the real-time measurements are periodically saved in a historical database. The *auto.arima* method from the forecast package of R [101] is used to select the optimal ARIMA model variant, thereby adapting to the increasing amount of historical data (i.e., non-static p, d, q observations). By default, *auto.arima* limits p and q to the range [0,5], and the best fitted model is the one with the lowest Akaike's Information Criterion (AIC) value [36]:

$$AIC = n\ln(\frac{RSS}{n}) + 2k \tag{4.2}$$

where k is the number of estimated parameters including the variance, and n is the number of observations. In our experiment $k = p + q$, and *RSS* (Residual Sum of Squares), is the estimated residual of the fitted model [101]. The coefficients ρ_i, υ_j and the intercept term μ are calculated from the fitted model ARIMA (p,d,q) and prior data. Hence, the value of y_t can be estimated. The descriptions of the environment and security parameters used to estimate the host security state are summarized in the "Predict" group of Table 4.1. The outputs from the forecasting modules and relevant analysis are described in Chapter 5 through realistic attack simulations that attempt to compromise the main host web application.

4.1.4 System Module

The system module uses the estimated trending of both the environment and security parameters in conjunction with the current state of the host system to predict its future state. In our system, both security parameters *memory* and *CPU_idle* are estimated by the forecasting module on the main host. The forecasting process on the MC-VM uses the same technique to estimate *Num_of_attacks*. The future state of the host $\hat{x}$ is represented as,

$$\hat{x}(k) = f(x(k-1), \hat{\omega}(k), \hat{\lambda}(k), u(k)) \tag{4.3}$$

where $\hat{\omega}(k)$ and $\hat{\lambda}(k)$ denote estimations of the environment and security parameters at time step k. $u(k)$ is the control mechanisms (responses) to eliminate the upcoming attacks. Next, the estimated system state variables $\hat{x}(k)$ is sent to the state classifier which determines whether the system is "Normal" or "Abnormal" based on the distances between the estimated values and the system's normal region as shown in Figure 4.3.

As shown in Figure 4.3, $x \in X$ is the system state variables, $\omega \in \Omega$ is the environment parameters, and $\lambda \in \Lambda$ is the security parameters. In Figure 4.3 and 4.4, system security regions ($Z = X \times \Omega \times \Lambda$) include the normal region (i.e., Z_N), known attack regions (i.e., $Z_A^1 ... Z_A^a$), and an unknown attack region (i.e., Z_U). Predicted parameter values that are outside the *normal region* Z_N indicate likely cyber attacks targeting the host.

Figure 4.4 shows the five-state control process of the ASM approach. $z \in Z_N$ denotes the current DS security state is normal. $z \in Z_A^n$, where $n \in [1...a]$ (e.g., $z' \in Z_A^2$ as shown in Figure 4.3) denotes that the current security state deviates from the normal region (i.e., it is very likely the DS is being attacked using a known signature). When the system security state does not fall in any given regions, the system is likely being compromised by an unknown attack. For example, in Figure 4.3, the system security state $z'' = [x'' \; \omega'' \; \lambda'']$. $Z_U = [Z_N \bigcup_{n \in [1,a]} Z_A^n]^c$, where A^c means the complement of a set A. In this case, Z_U is the

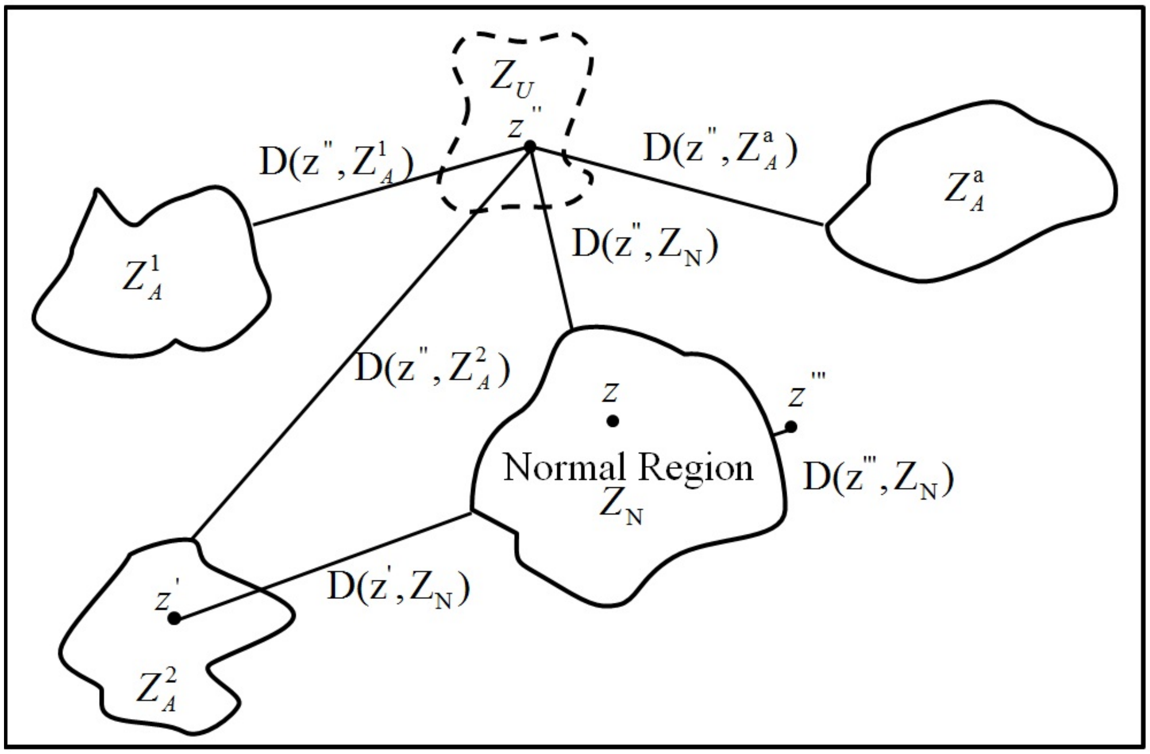
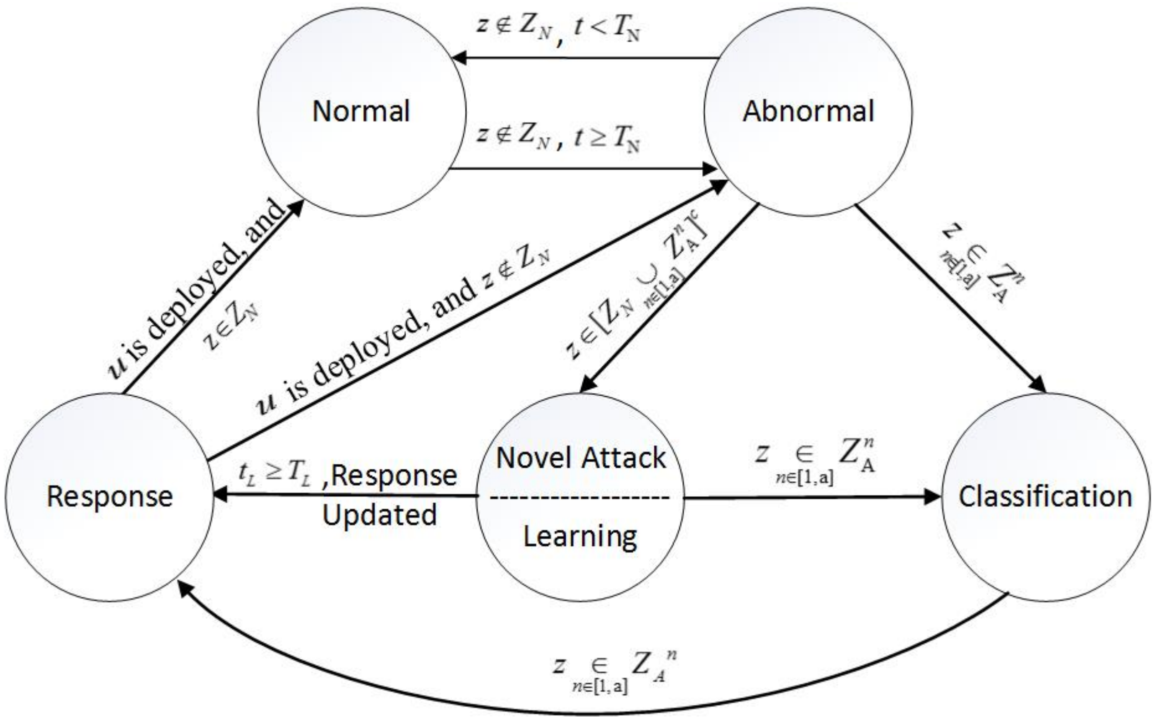


Figure 4.3

Security States

complement of the set Z_N (the normal region) and the set Z_A^n (known attack regions), and it is defined as an unknown attack region that has never been seen before. For any region Z_i and a state z, we define the distance $D(z, Z_i)$ between z and Z_i as $\min_{z' \in Z_i} \|z - z'\|$, where $\|.\|$ is an appropriate norm. If the minimum distance is less than the normal distance threshold ϵ, the current system security state is considered normal. For instance, in Figure 4.3, the system security state z''' is normal because $D(z''', Z_N) = \min_{z_n \in Z_N} \|z''' - z_n\| < \epsilon$.

We define a time threshold T_N, which can be used to distinguish observed transient abnormal system behavior from actual cyber attacks. Noisy data collected by the sensors



Normal
Abnormal
Response
Novel Attack
Learning
Classification
$z \notin Z_N,\ t < T_N$
$z \notin Z_N,\ t \geq T_N$
u is deployed, and $z \in Z_N$
u is deployed, and $z \notin Z_N$
$z \in [Z_N \cup_{n\in[1,a]} Z_A^n]^c$
$z \in_{n\in[1,a]} Z_A^n$
$t_L \geq T_L$,Response Updated
$z \in_{n\in[1,a]} Z_A^n$
$z \in_{n\in[1,a]} Z_A^n$

Figure 4.4

Outline of the Control Process

may look similar to abnormal system security states. In this scenario, when the abnormal behavior status lasts less than T_N seconds, the controller will not implement any protection mechanism. If the abnormal system security state is caused by cyber attacks, z is identified as a known attack or a novel attack by IDS. If the upcoming traffic is a known attack, the most appropriate protection responses, u, derived by the multi-criteria analysis controller (described in Subsection 4.1.7) are initiated. Conversely, if a novel zero-day attack or evolving attack is present, the learning module will capture its signature. This learning process may last longer than T_L seconds (the threshold for learning novel attack signatures). In this case, the most appropriate responses that have been updated and are available by time T_L will be deployed to defend the system against unknown anomalies.

4.1.5 Intrusion Detection System

The intrusion detection system (IDS) is a data mining tool that allows real-time event analysis. The goal of this tool is to provide accountability for intrusive activities. To this end, this module uses parameters of system performance and attack signatures to identify cyber attacks that have not been prevented by the first line of defense (i.e., the forecaster). The performance-based IDS (PIDS) on the MC-VM employs anomaly and misuse detection methods to detect and classify all types of known attacks. Off-line training models of the PIDS are developed using data-mining techniques and normal data combined with various attack data. Because only ten attributes (i.e., those shown in the PIDS group of Table 4.1) are selected for inclusion in historical training datasets, the detection time is significantly reduced compared with signature-based IDSs (SIDSs) that fully adopt all 41 features of the KDD99 dataset [5], in which case, the SIDSs require a large number of rules. The resulting high-speed performance IDS ensures much faster real-time detection from incoming flows with a low overhead of detection latency. The IDS is also able to quickly assign known attacks to their categories, which lightens and improves the selection of optimal protection methods for the controller. However, numerous a priori normal and intrusion datasets must be provided to improve the accuracy of detection. In addition, cyber attacks that affect the system or network performance insignificantly are difficult to detect (i.e., train for). To improve the detection accuracy, we also designed a SIDS, and run it on the main host to compensate for these drawbacks.

The SIDS detects anomalies based on attack patterns such as the payload and control information of communication packets, the number of failed login attempts, and the time spent on a specific page by malicious users (more details of relevant parameters can be found in the SIDS group of Table 4.1). This attack takes a longer time to detect attacks, especially when a large set of detection rules are necessary. Thus, the SIDS is only used when the PIDS on the main host (in the IDS module of Figure 4.2) mis-classifies attacks or fails to detect attacks due to an unnoticeable influence on the main host. In Chapter 5, for an example, we use the ModSecurity module to secure the Apache server from web application attacks that do not significantly influence system or network performance. The ModSecurity module performs as a SIDS along with an active response mechanism to detect and eliminate cyber assaults without disrupting normal services. The PIDS, installed on the main host, adopts the anomaly detection approach, which can detect anomalies with fast detection speed as discussed above.

The Naive Bayesian classifier is employed to establish the off-line training model, among other data mining techniques, to train the PIDS. Using the Naive Bayesian classifier is desirable because it does not require a large amount of training sets and yet provides highly accurate estimates. In our experiment, historical data are composed of 2000 samples for each category. Since we developed six realistic attack vectors (UDP Flooding Attacks, TCP SYN Attacks, ICMP Flooding Attacks, Ping of Death Attacks, SQL Injection Attacks, and Exhaustion Attacks) to validate the self-protection feature of the ASM approach, the total amount of historical datasets are around 14000. 2000 observations of system normal performance are among these training data as well. Moreover, the parameters for each feature are learned separately based on the independence assumption. The learning process of the Naive Bayesian classifier is simple, particularly when the number of features is large.

The Naive Bayesian classifier is employed to establish the off-line training model, among other data mining techniques, to train the PIDS. Using the Naive Bayesian classifier is desirable because large training sets are unnecessary and yet provide highly accurate estimates.

The Naive Bayesian classifier computes the conditional probability of attack categories relying on the relationship between both independent and dependent security relevant features [91]. The most likely attack category C_{nb} given features $x_1, ...x_K$ is described as follows.

$$C_{nb} = \arg\max_{c_i \in C} P(c_i) \prod_{j=1}^{K} P(x_j|c_i) \tag{4.4}$$

where $P(x_j|c_i)$ is estimated using m-estimates:

$$P(x_j|c_i) = \frac{n_a + mp}{n + m} \tag{4.5}$$

where c_i is one of the following categories C={Normal, UDP Flooding Attacks, TCP SYN Attacks, ICMP Flooding Attacks, Ping of Death Attacks, SQL Injection Attacks, Exhaustion Attacks}. n is the number of training examples for which $c = c_i$. n_a represents the number of examples for which $c = c_i$ and $x = x_j$. p is the a priori estimate for $P(x_j|c_i)$, and m is the equivalent sample size. The number of features (or attributes), K, is 10. The details of PIDS attributes are listed in Table 4.1.

4.1.6 Learning Module

Unknown attacks can be identified by an IDS adopting the anomaly detection technique. We first established the normal region of the victims by experimental and expert experience, and then real-time observations that deviate from the normal region are identified as anomalies. If these anomalies do not belong to any categories contained in the training model, they are considered unknown attacks (Illustrated in Figure 4.3). Unknown attacks may include (slowly) evolving attacks or zero-day attacks, in which case, adversaries exploit unknown vulnerabilities to undermine system security. In both scenarios, signatures of unknown attacks have not been revealed. As a result, no effective response can be performed by the MAC to recover system performance.

The learning module, installed on the main host, monitors and analyzes network traffic. This module derives and obtains the causes and impacts of novel attacks, as well as their unique signatures. The learning module only captures and saves malicious packets to storage rather than logging all events [96]. This avoids exhausting system resources.

The DS must be available 24/7 to offer continuous non-stop data access to their users. Thus, when the learning module analyzes attack signatures, normal services cannot be interrupted. Live forensics analysis is our efficient approach to capture attack signatures while maintaining normal operations using forensics tools (e.g., Wireshark [21]) and statistical theories (e.g., Naive Bayesian Network).

Through the learning module analysis, new rules about novel attack signatures are automatically added to rule sets of the SIDS. Meanwhile, new redirection rules for the router are added that designate the destination IP addresses and port numbers sourcing malicious activities. Such malicious messages will be redirected to the MC-VM. Known attacks are identified and eliminated by the modules running on the MC-VM. Novel attack signatures are also helpful in updating candidate responses and evaluating appropriate responses for the MAC. For instance, snort_inline drops malicious packets based on string matching; therefore, new dropping-rules for malicious activities are appended to existing snort_inline rule sets. To defend against similar future attacks, the protection process initialization module configures the candidate responses, and the MAC then selects the optimal response (e.g. snort_inline) to counter a given attack.

4.1.7 Multi-criteria Analysis Controller (MAC)

The MAC evaluates candidate responses as an intrusion response system (IRS) and initiates the most appropriate responses necessary to recover the system performance under two conditions 1) if estimations of the DS performance are abnormal, or 2) real-time observations are identified as attacks. Fuzzy-logic and the Preference Ranking Organization METHod for Enrichment Evaluations II (PROMETHEE II) are two efficient methods to assess candidate responses. The response evaluation depends on multiple criteria such as quality of service performance, security assessments, and implementation costs.

The fuzzy-logic control is generally simple but not precise. For instance, the efficiency of two protection methods within the same fuzzy class cannot be distinguished. Accordingly, we employed the PROMETHEE II optimization method because it is more efficient at identifying decisions which have long term impacts on the system behavior. Although the PROMETHEE II method involves more expert preferences than the fuzzy-logic method, it provides a more comprehensive and rational framework to structure decision problems. Consequently, even if two responses have similar (but not exact) properties, the evaluation results of these two responses are not the same. A comparison of the performance of the MAC applying these two approaches is provided in Subsecection 5.4, which favors the PROMETHEE II method over the fuzzy-logic method.

Candidate responses are rated using the same weighted criteria for both approaches. Relative weights reflect the importance of criteria using fuzzy numbers. Very important criteria are set to 1, and the unimportant criteria are 0. In our work, the selected criteria are considered equally important so that they are all set to 1 by default. Fuzzy numbers from 0 to 1 are used to normalize criteria, where 0 represents effective responses, and 1 represents that candidate responses fail to defend against cyber attacks.

In the following paragraphs, we first discuss the pros and cons of candidate responses, and then introduce criteria that are used to evaluate these responses.

4.1.7.1 Protection Initiation Unit

This unit is responsible for initializing and configuring recommended responses. The following responses are utilized in our approach to defend against known and unknown cyber attacks.

- Snort_inline (u_1): It is an intrusion protection system [20] combining an intrusion detection system and a firewall to actively respond to attacks. Snort_inline drops malicious requests if control information and payloads of incoming packets match the pre-defined rules. This protection mechanism does not block legitimate requests. The pre-defined rules can be updated in real-time to quickly identify previously unseen attacks. However, a large number of rules induce computational overheads by slowing down the execution speed of the recovery action.
- Packet Filtering (u_2): This response eliminates all packets, both legitimate and illegitimate, sent to the compromised ports on the main host. It ultimately protects the host from flooding-based DoS attacks and rapidly mitigates network congestion. However, elimination of legitimate packets impedes appropriate transactions between legitimate users and web servers. In addition, this action only works for Open Systems Interconnection (OSI) upper layers (higher than layer three) network protocol transmissions where packets involve destination ports. Notably, this response does not operate for Internet Control Message Protocol (ICMP) packets since they have no port abstractions.

- Authentication Process (u_3): The process of authentication provides facilities to generate cryptographic keys safely. The use of public and private keys improves the authentication of data so that only legitimate users are able to access system resources. However, this action cannot recover system or network resources that have already been exhausted.
- Active Replica (u_4): Replica servers can replace compromised servers to maintain constant functionality of the system. This response allows quick resumption of service but with high implementation costs. One big drawback of this protection mechanism is that it cannot prevent future attacks from compromising the replica server.
- Network Disconnection (u_5): Disablement of the network interface can be used to block network attacks once they are detected. This is a quick action to eliminate all kinds of exploits being transmitted by network packets except worms and viruses that have already been planted into the main host. However, as the server is not connected to the network, the system obviously cannot provide normal services (i.e., requests cannot be received or served).
- Process Termination (u_6): Excessive consumption of system resources is an obvious signature of exhaustion attacks. Once exhaustion attacks are identified and classified by IDSs, terminating abnormal services or processes can be used to mitigate such attacks. The terminated processes are logged in case system administrators would want to view or analyze malicious activities offline. This method, however, may terminate normal services, which results in service disruption. To correctly terminate malicious processes and recover the host from attacks, a list of processes that are

not allowed to terminate must be provided by subject matter experts (SMEs), who carefully study system maintenance and security guidelines. In certain situations, human intervention is required for authoring process termination.

- ModSecurity (u_7): This module runs on Apache servers [30] and blocks malicious web requests when they are detected. In this module, various user-defined I/O filters can be configured using regular expression operators (regex) to filter incoming and outgoing host packets that match the pre-defined rules. This capability is embedded in the Apache server and efficiently blocks SQL Injection attacks, but the overhead here is linearly proportional to the number of rules. Unfortunately, ModSecurity only detects and responds to HTTP attacks compromising Apache servers.
- Host Shutdown (u_8): Shutting down the host can be initiated as a last resort to protect hardware and data from cyber attacks. The implementation of this response is fast, and protects the DS from a large variety attacks. Obviously, the latency of restarting the host negatively affects the performance/availability of web services.

4.1.7.2 Protection Evaluation Unit

Candidate responses are evaluated based on eight aspects with respect to performance, cost, speed, and protection effectiveness. The eight criteria are discussed as follows:

- Recovery Time (r_1): This aspect reflects how fast candidate responses could recover the system behavior of the compromised main host back to normal.

- CPU Utilization Recovery (r_2): This aspect reflects how much CPU utilization are consumed when the mitigation responses are applied.
- Packet Rate Recovery (r_3): This aspect reflects how efficient mitigation responses are to mitigating traffic congestion.
- Legitimate Data Transmission Recovery (r_4): The web application provides services to clients; therefore, the legitimate data transmission criterion reflects how well mitigation responses can maintain normal system operations.
- Connection Rate Recovery (r_5): This aspect is used to evaluate how well mitigation responses reduce an abnormally high connection rate.
- Failure Login Rate Recovery (r_6): This aspect reflects whether candidate responses could enhance data integrity and access authentication.
- Memory Utilization Recovery (r_7): This aspect is used to assess how well the mitigation responses can reduce abnormally high memory utilization.
- Cost (r_8): This criterion reflects the financial cost of implementing candidate mitigation responses.

Criterion values of candidate responses for different attacks are varied. For instance, in our experiments, snort_inline is the best response to protect UDP flooding attacks, but Packet Filtering is more efficient than snort_inline to defend against TCP SYN attacks. Therefore, experts must take into account all criteria associated with different types of attacks when assigning initial values to each response.

4.1.7.3 Fuzzy-logic Method

Fuzzy-logic is one of the most efficient approaches to determining the optimal responses among candidate responses toward protecting the web application from cyber attacks. The first step of the fuzzy-logic method is to evaluate the efficiency of responses and the importance of the relative weights of criteria. Efficiency and relevant weight scores are assigned by decision makers following fuzzy IF-THEN rules, which are normalized to the range [0, 1] represented by fuzzy numbers. For instance, if a response ultimately reduces utilization of the CPU (due to some disruptive attacks), then its value is 0; otherwise, the response is assigned 1. The most important criteria weights are assigned 1, and least important ones are 0. The cost $Cost(u)$ of using a response $u \in U$ is defined as follows, where U denotes the set of protection actions:

$$Cost(u) = \sum_{r \in R} w_r \ v(r, u) \tag{4.6}$$

where R is the set of criteria mentioned in Subsection 4.1.7.2; $v : R \times U \to \mathbb{R}$ is a map that assigns a value to each criterion; $r \in R$ reflects its relative effectiveness under the response $u \in U$; the relative weight of the criterion r is represented by w_r.

The fuzzy-logic controller shown in Figure 4.2 sorts responses by their costs as defined above. The best reaction, u^*, is the one corresponding to the smallest cost.

$$u^* = \arg\min\{\sum_{r \in R} w_r v(r, u) | u \in U\} \tag{4.7}$$

In the Performance Evaluation subsection (Subsection 5.4 below), we give an example to demonstrate how a fuzzy-logic controller determines the optimal response among eight candidate protection methods for recovering the system from SQL Injection attacks, along with the ranking of these responses in Table 5.7.

4.1.7.4 PROMETHEE II Method

The PROMETHEE II approach is more complex but more precise than the fuzzy-logic method. It can be used when the fuzzy-logic approach fails to make an efficient decision. The procedures for this method are outlined as follows:

- Compute and compare preference degree: Initially we determine the preference degree of each response with respect to others for each criterion. The difference between evaluations of two protection actions u and u' based on pair-wise comparison is represented by $d_r(u, u')$; and $g_r(u) \in [0, 1]$ denotes the evaluation value of action u for criteria r.

$$(\forall u, u' \in U)(\forall r \in R) \quad d_r(u, u') = g_r(u) - g_r(u') \tag{4.8}$$

- Select a preference function: There are six types of preference function, i.e., Usual, Quasi, V-Shape, Level, V-Shape with Indifference and Gaussian Criterion [98]. Preference functions are used to “translate the difference between the evaluations obtained by two responses into a preference degree ranging from zero to one” [31].

The preference functions for each criterion are decided by system administrators. The preference degree between two actions, u, u', is:

$$P_r(u, u') = F_r\left(d_r(u, u')\right) \tag{4.9}$$

where F_r is the preference function for criterion r, and P_r is the preference degree for criterion r.

- Calculate global preference index: Weights of criteria determine the relative importance of the criteria. Normally, weights are decided by system administrators in the same way as the thresholds of the preference functions. The sum of weights is equal to 1. $\pi(u, u')$ represents the preference of protection mechanism u over u', and is defined as the weighted sum of preference degrees for each criteria. In our experiment, all eight weights are equally important, so we assign $1/8$ to each weight.

$$\pi(u, u') = \sum_{r \in R} w_r P_r(u, u') \tag{4.10}$$

- Calculate positive and negative outranking flows: For each response $u \in U$, the positive (negative) outranking flow represents the extent to which a response is higher (lower) than other responses. The following formula shows the positive, ϕ^+ and negative, ϕ^-, outranking flows. The net outranking flow ϕ is basically the difference.

$$\phi^+(u) = \frac{1}{n-1} \sum_{u_j \in U} \pi(u, u_j) \tag{4.11}$$

$$\phi^-(u) = \frac{1}{n-1} \sum_{u_j \in U} \pi(u, u_j) \tag{4.12}$$

$$\phi(u) = \phi^+(u) - \phi^-(u) \tag{4.13}$$

where ϕ denotes the ranking of the responses. $u_j \in U$, and $j \in [1, n]$. n is the number of candidate responses. The better responses have higher ϕ values. The cost of responses u is inversely proportional to the net outranking flow, which is expected to be minimized. The cost thus can be represented by $-\phi$.

To realize a self-protecting system that proactively protects from upcoming attacks, actively responds to known and unknown anomalies in real-time, and efficiently maintains normal services, an autonomic security management approach has been deployed. This approach adds a MC-VM to a computing environment, and the close interaction between the router, the main host, and the MC-VM easily strengthens the security of DSs with little or no human intervention. In the above sections, the essential autonomic components and their mechanisms have been discussed. In the following sections, we will discuss how to apply the ASM approach to protect ICSs from various cyber attacks.

4.2 Autonomic Security Management Framework for Industrial Control Systems

In this section, we will introduce how to realize a self-protecting SCADA systems with few modifications of the ASM framework that has been developed in Section 4.1. The self-protecting SCADA system is constructed from four components: risk assessment (done offline), early warning and prevention (intrusion estimation), intrusion detection, forensic analysis, and intrusion response as shown in Figure 4.5. As a result, the self-protecting SCADA system properly anticipates upcoming anomalies, sends early warnings to the multi-criteria analysis controller (MAC), and protects the system by signaling the MAC to initiate an active response to selected threats.

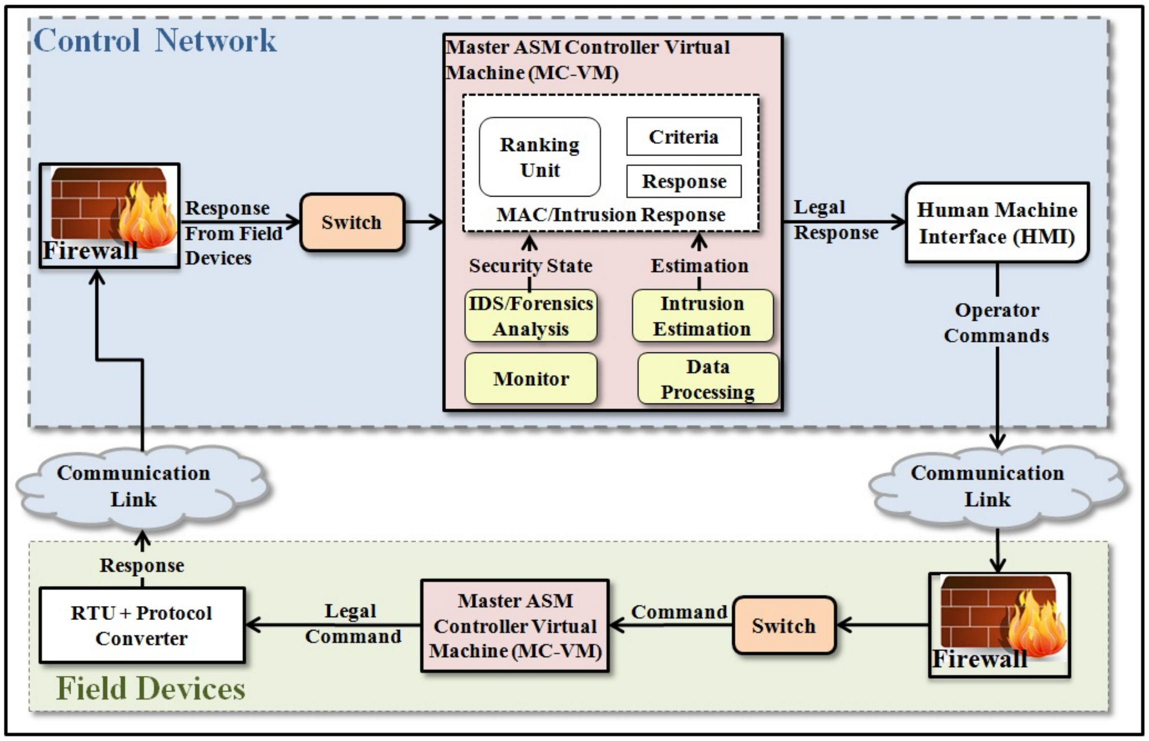


Figure 4.5

The Autonomic SCADA System Testbed

Similar to the self-protecting web application discussed in Section 4.1, sophisticated attacks that evade the first line of defense of the self-protecting SCADA system due to incorrect estimation of future system security states may be thwarted by a second layer IDS. Live forensics analysis learns causes and impacts of unknown attacks. Novel signatures are utilized to update detection algorithms and the active response (countermeasures) library.

As a result, similar attacks that occur in the future can be preempted by the first line of defense.

As we discussed above, adding MC-VMs to the SCADA networks can easily realize a self-protecting SCADA system. Therefore, we placed one MC-VM in the control network and another in the field device network to protect the SCADA system against cyber assaults (see Figure 4.5). Since the SCADA system is serial based, operational commands are sent to the control network via the default Modbus serial protocol (e.g. the Modbus RTU protocol). To support long distance transmissions and enable communications between computers and field devices, protocol converter software in the RTU changes Modbus RTU protocol responses to Modbus TCP/IP messages and delivers the Modbus TCP/IP messages to the control network. The MC-VM located in the control network filters illegitimate responses, and only legitimate responses sent from the RTU can be forwarded to the HMI. The commands sent from the HMI to the RTU are monitored by the MC-VM located in the field device network. Cyber attacks are therefore anticipated, detected, analyzed, and prevented; making the security of the SCADA system stronger.

In the following subsections, we focus on intrusion estimation, intrusion detection, and the MAC modules since techniques or selected features employed in these modules (SCADA-specific) are slightly different from a self-protecting web application. Techniques and theories for developing other autonomic components are the same. Note that, the normal operational region (i.e., profile) of the SCADA system is constructed offline by applying expert knowledge, experimental results, literature reviews, and security guidelines.

The normal operational regions vary according to the complexity of each unique SCADA environment and are captured by different physical system models.

4.2.1 SCADA-Specific Intrusion Estimation

Real-time observations of SCADA systems and physical systems (e.g. a gas pipeline or a water storage tank) are monitored and processed by the sensors and the data processing module, and then these observations are sent to the intrusion estimation module to anticipate the upcoming system security states. Besides the environmental parameters (e.g. *packet_recv*) and security parameters (e.g. *memory*, *CPU_idle*, and *Num_of_attacks*) introduced in Section 4.1.3, for process control systems, control variables must be predicted by the intrusion estimation modules as well. Different physical infrastructures have various control variables. In this case study, we only discuss the gas pipeline and water storage tank systems; therefore, control variables such as the gas pressure and the water level are predicted.

To predict the future security stateof the ICS, besides using the ARIMA method combined with system models and historical datasets, we also construct a linear model that captures the behavior of the physical system (gas pipeline and water storage tank systems). This linear model is derived from empirical I/O data while the gas pipeline/water storage tank system which is controlled automatically. The linear model represents dynamics that are linear in time k

$$\omega(k) = Ak + B \tag{4.14}$$

where $\omega \in \mathbb{R}$ is the gas pressure/water level. $k \in \mathbb{Z}$ is the positive integer number of samples. For the gas pipeline, the period of gas pressure that goes up and down is 17 samples long (as shown in Figure 4.6). During this period, every gas pressure sample is 50 ms. The total period is 850 ms. When $1 \leq k \leq 13$, the gas pressure climbs. The coefficients of the linear model are: $A = 0.02316$ and $B = 4.77521$, and the residual standard error (RSE) is 0.0008275. When $14 \leq k \leq 17$, the gas pressure goes down. $A = -0.07672$, $B = 5.18498$, and RSE is 0.001948. The linear model fits the observed data perfectly since RSE values are close to 0.

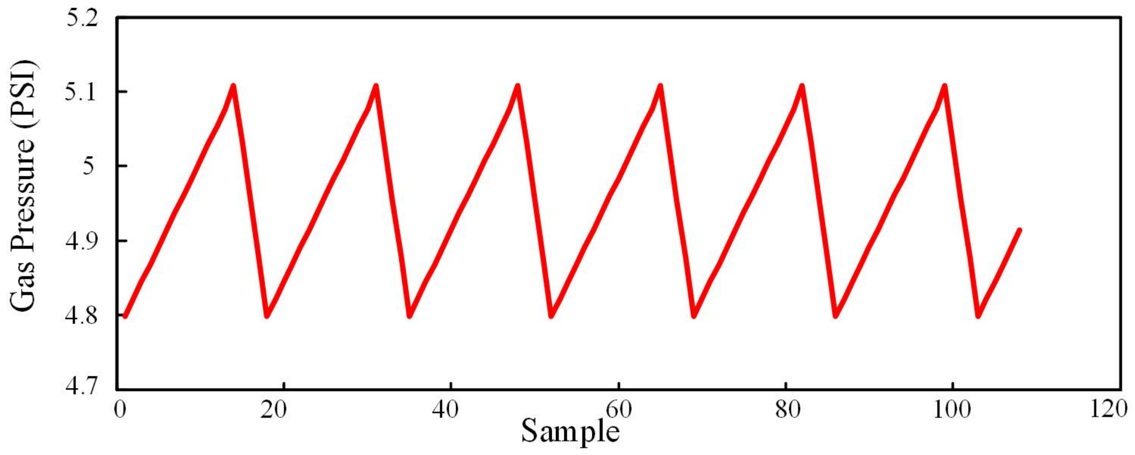


Figure 4.6

Normal Trend of Gas Pressure

For the water storage tank system, the water level increases and decreases periodically when it is controlled automatically (as shown in Figure 4.7). From observations of the laboratory-scale water storage tank, we find out that the period of the water level is 80 samples long. When $1 \leq k \leq 35$, $A = 0.256$ and $B = 51.181$. When $36 \leq k \leq 39$, $A = -1.976$ and $B = 62.090$. When $40 \leq k \leq 45$, $A = 0.03249$ and $B = 56.71783$. When $46 \leq k \leq 80$, $A = -0.202$ and $B = 56.686$. Therefore, the future values of the control variables, namely, gas pressure and water level, can be predicted using this linear model, and $\hat{\omega}(k+1) = A(k+1) + B$.

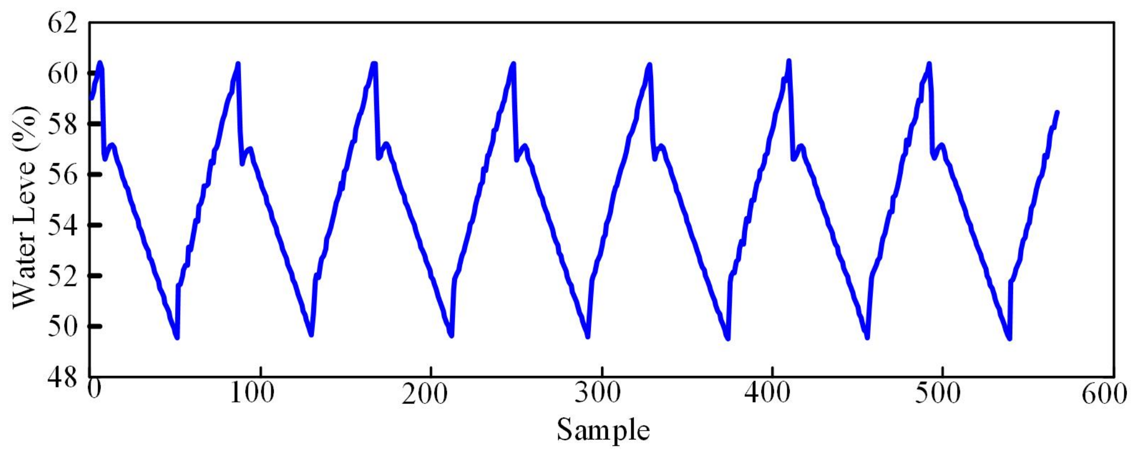


Figure 4.7

Normal Trend of Water Level

Upcoming attacks are then be anticipated by comparing the "normal region" with the predicted outputs of the physical system model. Early warnings are sent to the MAC, and appropriate control mechanisms are executed to neutralize attacks (on-the-fly).

4.2.2 SCADA-Specific Intrusion Detection

As described in Section 4.1.5, the selected features which represent the variances of system and network resources are the same as the features selected to establish the "normal region." A Weka Rotation Forest classifier [22] has been used to develop an off-line training model for the performance-based IDS, which provides a 99% true positive detection rate. The selected features for constructing the PIDS are shown in the PIDS group in Table 4.1. The PIDS can also quickly assign known attacks to specific categories, which lightens and improves the selection of optimal protection methods for the MAC. However, cyber attacks that insignificantly affect the system or network performance are difficult to detect (i.e., train for) by the PIDS. Therefore, similar to the IDSs developed for detecting web application cyber attacks, we have designed a rule-based signature IDS based on snort [20] for detecting such attacks. For SCADA-specific IDS, besides features listed in the SIDS group (see Table 4.1), we also summarize specific signatures of the gas pipeline and water storage tank systems (listed in *"Gas Pipeline"* and *"Water Storage Tank"* groups). These features can be used to identify known attacks such as function code scanning, malicious parameter command injection, and denial of service attacks.

4.2.3 SCADA-Specific Multi-criteria Analysis Controller (MAC)

The MAC evaluates candidate responses as an IRS and initiates the most appropriate responses necessary to recover the SCADA system and physical process performance under two conditions: 1) if estimations of the SCADA system performance are abnormal, or 2) real-time observations are identified as attacks. The fuzzy-logic and PROMETHEE II are still works for evaluating the optimal responses for SCADA systems to defend against cyber attacks. Criteria are slightly different from the criteria selected in Section 4.1.7.2. The selection of criteria are still determined by experienced system administrators. For evaluating candidate responses (i.e. *Dropping Malicious Commands, Packet Filtering, Network Disconnection, Serial Port Disconnection, Replacement of Compromised Devices, One time Authentication, Termination of Physical Processes, and Isolation of Compromised Devices*), we take into account four criteria for evaluating candidate responses as follows:

- Enhancement of Security (C_1): Confidentiality, integrity and availability are three fundamental facets of the security. The response that efficiently enhances the security of SCADA systems is assigned a fuzzy number 0, otherwise, assigned a value greater than zero and less than or equal to 1. In other words, normalized response values that assess malicious activities are assigned a number in the range of [0,1].

- Operational Costs (C_2): The implementation of responses may exhaust computer and human resources. For example, the response "dropping the malicious commands" consumes computer CPU and memory resources to analyze protocol data units of communication messages. It also consumes storage resources for recording all known attack signatures. As a result, the implementation of a response increases operational cost. Values for this criterion are assigned a number in the range of [0,1]. The lowest cost responses are assigned 0; otherwise, they are assigned a value greater than zero as appropriate. The normalization process is initially accomplished offline.

- Maintenance of Normal Operations (C_3): The execution of responses are not permitted to disrupt normal performance of critical functions. The responses that have no impact on normal operations are assigned 0. The value of Criterion 3 is initialized to 1 if the implementation of selected responses is disruptive to normal operations; otherwise, values are assigned in the range of [0,1].

- Impacts on Property, Finance, and Human Lives (C_4): SCADA systems that control critical infrastructures will typically impact safety, economics, and property if and when they fail. The responses that greatly affect potential impacts cannot be executed autonomously. Candidate responses such as " termination of physical processes" and "isolation of the MTU or RTU" may have very costly impacts. Therefore, for obvious reasons such decisions must be left to human intervention. On the other hand, low impact responses can be performed autonomously. Examples of low impact responses include "dropping malicious commands," "replacing the compromised devices," and "adding one time authentication mechanisms."

Eight recommended responses (*Dropping Malicious Commands, Packet Filtering, Network Disconnection, Serial Port Disconnection, Replacement of Compromised Devices, One time Authentication, Termination of Physical Processes, and Isolation of Compromised Devices*) have been installed and configured offline. The MAC in Figure 4.5 evaluates these potential responses and selects optimal response(s) as countermeasures to specific types of cyber attacks. Since all criteria mentioned above are equally important, weights of each criterion (i.e., criterion 1-3) for a response u_i is $1/3$ (as the sum of the weights is equal to 1). The Criterion 4 decides whether the implementation of the response requires human intervention or not. If the value of Criterion 4 is higher than 0.5, the implementation of such responses must be approved by system administrators because these are deemed to have high impact responses. We use "Semi-Auto" to represent the requirement for human intervention and "Auto" to represent autonomous implementation (e.g. see Table 5.8).

4.3 Summary

In this chapter, we developed an autonomic security management approach to realize the self-protection function so that a web application and an ICS can proactively protect themselves from upcoming attacks, actively respond to known and unknown anomalies in real-time, and efficiently maintain normal services/functions. This approach simply adds a MC-VM to a computing and ICS environment to strengthen the security of DSs with little or no human intervention. In the next chapter, the self-protection function of the ASM approach will be validated through experiments from realistic cyber attack scenarios.

Table 4.1

Selected Features

Purpose	Parameters	Description	Purpose	Parameters	Description
PIDS	CPU_sys	CPU system utilization (%)	SIDS	File_status	Unauthorized file access state
	CPU_user	CPU User utilization (%)		TTL	Time to live
	CPU_idle	CPU Idle utilization (%)		drop_rate	Server packet drop rate
	memory	Available Memory (KByte)		payload	Data transmitted to the servers
	byte_recv	Data received by the server (Byte)		protocol	Packets transmission protocol
	byte_sent	Data Sent by the server (Byte)		dst_port	Destination port number(servers)
	packet_recv	Packet number received by the server		src_dst	Average No. of connections with same source and dest. IP and port No. (%)
	packet_sent	Packet number sent by the server		f_login	The number of failed logins
	io_read	I/O read requests per sec		locked_account	No. of accounts locked for exceeding max. No. of incorrect login
	io_write	I/O write requests per sec		retransmission_rate	Packet retransmission rate
Predict	CPU_idle	CPU idle utilization (%)		log_file_info	User login, file access, process exec.
	memory	Available Memory (KByte)		HTTP_rate	Request rate of HTTP packets
	packet_recv	Packet number received by the server		time_page	Average access time per web-page
	No._of_attacks	Number of eliminated malicious packets for specific attacks		hit_rate	No. of page hits per clients per second
Gas Pipeline	Gas Pressure Set Point	The desired or command value of the gas pressure	Water Storage Tank	Water Level Set Point	The desired or command value of the water level
	Gas Pressure Alarm Conditions	Alarm level set points including "H" and "L", which represents the high level and low level alarms. In normal case, the gas pressure should be in the rage of "L" and "H"		Water Level Alarm Conditions	Alarm level set points including "HH","H","L", and "LL", which represents the high level and low level alarms. In normal case, the water level should be in the range of "L" and "H"
	Gas Pressure	Monitored real-time values of the gas pressure		Water Level	Monitored real-time values of the water level
Rule Update	PID Gain	Tuning parameters for a PID controller (e.g. proportional/integral/derivative gain)	Gas Pipeline and Water Storage Tank	System Mode	The system mode includes "Auto", "Manuel", and "Off"
	payload	Data transmitted to the servers	Rule Update	dst_port	Destination port number (servers)
	protocol	Packets transmission protocol		log_file_info	User login, file access, process exec.

CHAPTER 5

EXPERIMENT

In this section, we demonstrate the developed autonomic security management (ASM) framework by simulating nine realistic cyber attacks, which impact availability of web services, threaten data confidentiality, damage file system integrity, gather information of industrial control systems (ICSs), and disrupt ICS normal operational commands. Experimental results show that the application of the ASM approach efficiently realizes a self-protecting system that protects against exploitation of vulnerabilities, enhances host/server security, and maintains the underlying services without (necessarily) requiring human intervention. Only in exceptional cases are manual interventions necessary.

5.1 Web Application Protection Testbed

The experimental testbed configuration settings are shown in Table 5.1. The testbed includes two web servers installed on the main host; they are the IBM Websphere Application Server [126] and the Apache HTTP Server Version 2.2 [30].

5.1.1 Baseline Measures

Normally, clients send 300 to 2000 random requests every 30 seconds to either the IBM Websphere Application Server or the Apache HTTP Server by Httperf, a tool for measuring web server performance [16]. The IBM Websphere Application Server receives legitimate client requests from Daytrader [49] and replies to clients with data extracted from the DB2 Express database. The Apache HTTP Server presents a login web page to the clients, and relevant account information is retrieved for clients from the MySQL database if the client can be properly authenticated. Both the DB2 Express database and the MySQL database are installed on the main host.

Table 5.1

Testbed Configuration Setting

Name	OS	Cores	RAM	Description
Main Host	Win XP	1	1 GB	a VM running on the VMware ESX hypervisor
Replica	Win XP	1	1 GB	a VM running on the VMware ESX hypervisor
MC-VM	CentOS 5	2	4 GB	a VM running on a CentOS host machine
Routers	CentOS 5	1	1 GB	VMs running on a Win 7 host machine
Attackers	CentOS 5 Win 2003	1	1 GB	Attack VMs running on a Win 7 host machine
Clients	CentOS 5	1	1 GB	VMs running on a Win 7 host machine

5.1.2 Test Cases

Experiments using UDP Flooding attacks, TCP SYN attacks, and Memory and CPU Exhaustion attacks targeted the IBM Websphere Application Server were performed. Database Insertion Web attacks and SQL Injection attacks attempt to compromise the Apache HTTP Server and the MySQL database server. HTTP Flooding attacks target either port 80 of the main host, on which port the Apache HTTP Server is listening, or port 8080, on which port the IBM Websphere Application Server is listening.

Two nodes in our testbed with two ranges of IPs were used to simulate real-world attackers and clients, namely, sending malicious messages and client requests from various source IP networks. The routers receive incoming requests, and examine these requests for possible redirection. The sample time needed to monitor the system and network utilization on the main host and replica host was 1 second, and the sample time needed for the MC-VM to process the monitored data was 2 seconds. The threshold of available memory resources was set to 313 MB, and the threshold of the percentage of CPU idle time was 20% made available to the main host. The normal packet received rate was between the range 0-368 packets/sec. Early warning signals were generated when estimations of CPU idle time and available memory resources were lower than their thresholds, or if the estimation of the packet received rate deviated outside its normal region. The prediction traces (CUP idle percentage, memory utilization, packet received rate, and number of Dropped malicious packets)of known attacks were omitted in the figures when estimations equaled zero.

5.2 Hypothesis 1: Self-Protecting Computing System

The complete design of the ASM framework allows for a close interaction between the various autonomic security modules of the ASM framework. This section describes four realistic unknown cyber attack experiments which target to undermine the security of a web application. These experiments demonstrate that by using the ASM approach, the self-protecting system has been realized to proactively estimate upcoming attacks, recover system performance, and maintain resilient "normal" web services without human intervention. As a result, the window of vulnerability was essentially closed by reducing the protection latency.

5.2.1 Unknown UDP Flooding Attacks

UDP flooding attacks are DoS attacks that use the connectionless user datagram protocol. Adversaries send a large amount of UDP messages to the victim system. As a result, legitimate users are unable to access the web service. In this experiment, the host forecaster estimated that the performance of the main host would be abnormal after 75 seconds. As shown in Figure 5.3, the main host was compromised as the intensity of attacks increased *byte_recv* and *packet_recv* significantly. Meanwhile, we observed that the *CPU_idle* and *memory* usage were decreased. At around 150 seconds, the host figure shows that the observations of all parameters return to normal. Changes in the feature values illustrate that the implementation of the most appropriate response, **snort_inline** (u_1), successfully protected the host system against unknown UDP attacks (as shown in Table 5.2). At this moment, system and network utilization of the MC-VM increased because malicious messages were redirected and analyzed by autonomic modules on the MC-VM (sample 75 in Figure 5.2). When new rules of unknown UDP flooding attacks were inserted into the rulesets of snort_inline causing it to drop malicious requests. This action frees the web server to only deal with legitimate requests.

Table 5.2

Evaluation of Protection Methods for Unknown UDP Flooding Attacks

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
r_1	0.3	0.3	0.2	0.2	0.2	0.3	0.2	0
r_2	0.2	0.2	1	1	0	0	0.6	1
r_3	0	0	1	1	0	0	1	1
r_4	0	1	0.8	0	1	1	1	1
r_5	0	0	1	1	0.5	0.5	1	1
r_6	1	1	0	0	0	0	1	1
r_7	0	0	1	1	0	0	0.5	1
r_8	0.2	0.2	0.2	0.8	0.5	0.8	0.4	0
Total Value (Ranking)	0.325 (1)	0.1775 (3)	-0.1774 (5)	-0.307 (8)	0.269 (2)	0.164 (4)	-0.249 (7)	-0.202 (6)

Figure 5.1 shows estimations generated by forecasters. In this figure, the estimated number of UDP flooding attacks reveals that adversaries kept carrying out UDP flooding attacks up until sample 12. Since the characteristics of these attacks were captured by the

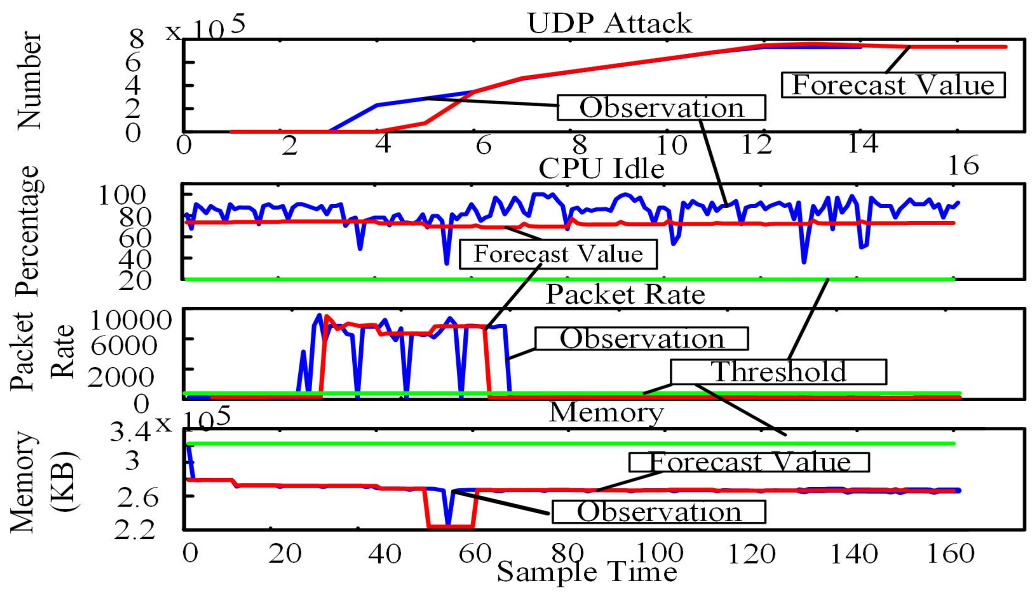


Figure 5.1

Estimated vs Measured Parameter Values for UDP Flooding Attacks

live NFA tool and snort_inline was updated in real-time, snort_inline kept dropping similar UDP flooding packets until sample 12. At this point, the forecaster running on the MC-VM, estimates that the host would no longer suffer from those attacks. Figure 5.1 shows that the initial estimations are close to what were observed. The slight fluctuation between estimations and observations is not significantly harmful because the estimated trends show to be consistent with real-time observations. Thus, forecasters adopting the ARIMA model are shown to be useful for enabling a more trustworthy and resilient web service.

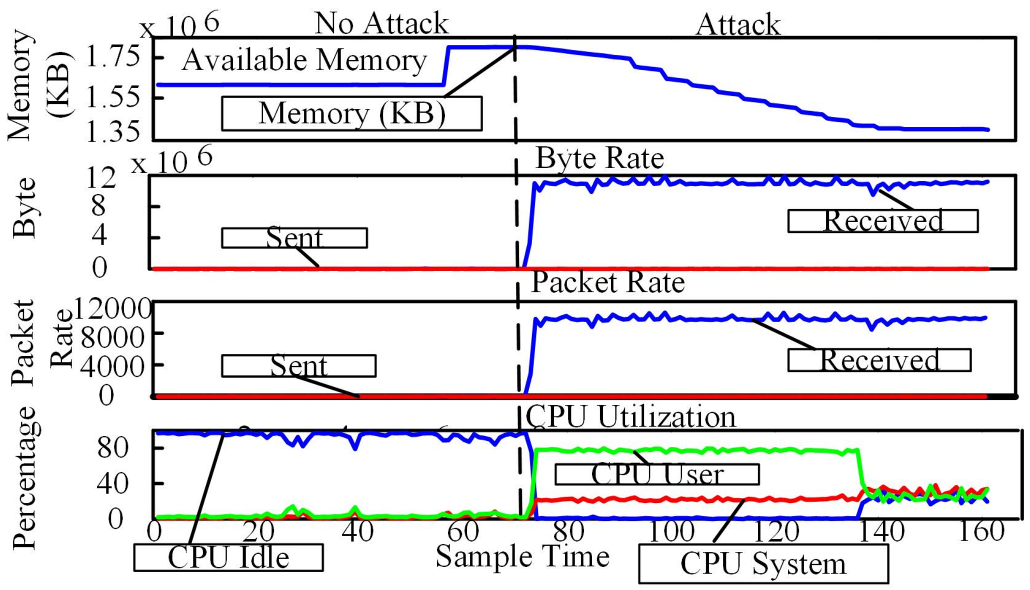


Figure 5.2

MC-VM Parameters for UDP Flooding Attacks

5.2.2 Unknown TCP SYN Attacks

TCP SYN attacks are one type of DoS attacks. Attackers create a tremendous number of half-opened TCP connections that lead to server memory resource exhaustion. Figure 5.4 shows the estimate for four features and their respective observations throughout the attack experiment. Estimations of the main host features are that *CPU_idle* and *memory* are inside the normal region; however, from samples 40 to 70, the estimation of *packet_recv* was much higher than its threshold; consequently, the intrusion detection system can determine that the web server was compromised by a DoS attack. Since the TCP SYN attacks were unknown attacks, the learning module installed on the main host was invoked to capture novel attack signatures. Captured novel attack signatures were used to update

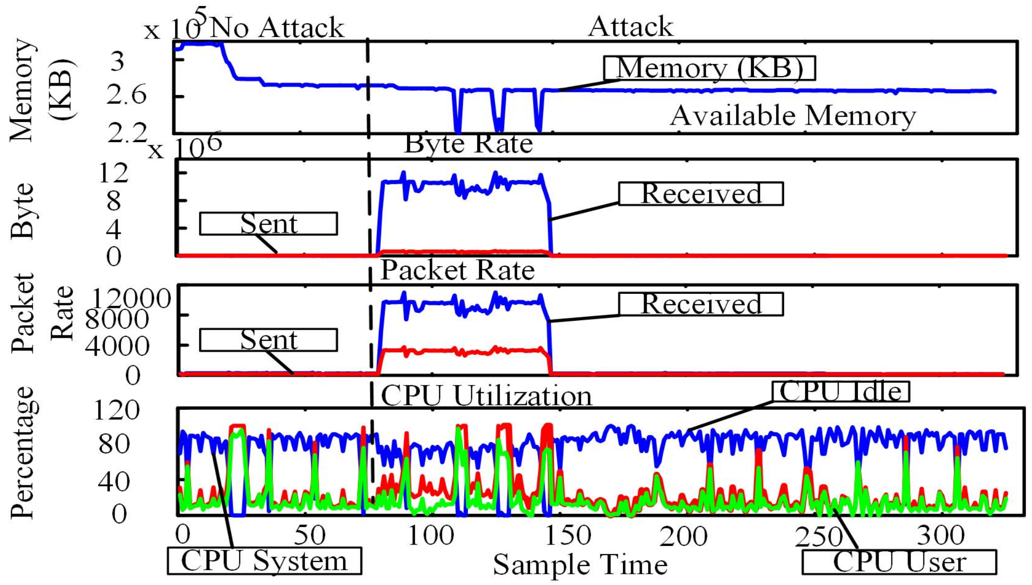


Figure 5.3

Host Parameters for UDP Flooding Attacks

redirection rules of the router. As a result, the TCP SYN attacks were redirected to the MC-VM, from which malicious packets were eliminated.

The optimal response needed to protect the web application from TCP SYN attacks is **Packet Filtering** (u_2) (as shown in Table 5.3). This response was assessed and selected by the multi-criteria analysis controller (MAC) using the fuzzy-logic and PROMETHEE II methods. Once Packet Filtering had been implemented after 95 seconds (shown in Figure 5.6), the performance of the main host is recovered back to normal. Figure 5.5 shows the system and network utilization of the MC-VM. After 50 samples, both *byte_recv* and *packet_recv* increased sharply while the values of the *memory* feature dropped sharply. *CPU_sys* increased from 45% to 60% because a large number of malicious packets were

Table 5.3

Evaluation of Protection Methods for Unknown TCP_SYN Flooding Attacks

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
r_1	0.3	0.3	0.2	0.2	0.2	0.3	0.2	0
r_2	0.2	0.2	1	1	0	0	0.6	1
r_3	0	0	1	1	0	0	1	1
r_4	0	1	0.8	0	1	1	1	1
r_5	0	0	1	1	0.5	0.5	1	1
r_6	1	1	0	0	0	0	1	1
r_7	0	0	1	1	0	0	0.5	1
r_8	0.2	0.2	0.2	0.8	0.5	0.8	0.4	0
Total Value (Ranking)	0.117 (4)	0.416 (1)	-0.206 (6)	-0.331 (8)	0.253 (2)	0.148 (3)	-0.287 (7)	-0.111 (5)

redirected to the MC-VM. Consequently, the CPU utilization subfigure, *CPU_idle*, was less than 20%. From 95 samples onwards, the adversaries stopped bombarding DSs with TCP SYN attacks as indicated by the performance of the MC-VM returning to normal (see Figure 5.5).

Figure 5.4 shows estimations generated by the ARIMA forecaster. The estimated numbers of TCP SYN attacks increased linearly from samples 7 to 15. Therefore, ARIMA forecaster correctly predicted that an increased number of TCP SYN attacks would be directed to the web server. The most appropriate response, Packet Filtering, continuously filtered out malicious packets to successfully defend the web server from upcoming TCP SYN attacks until the point where no those attacks ceased at the main host (i.e., after sample 15).

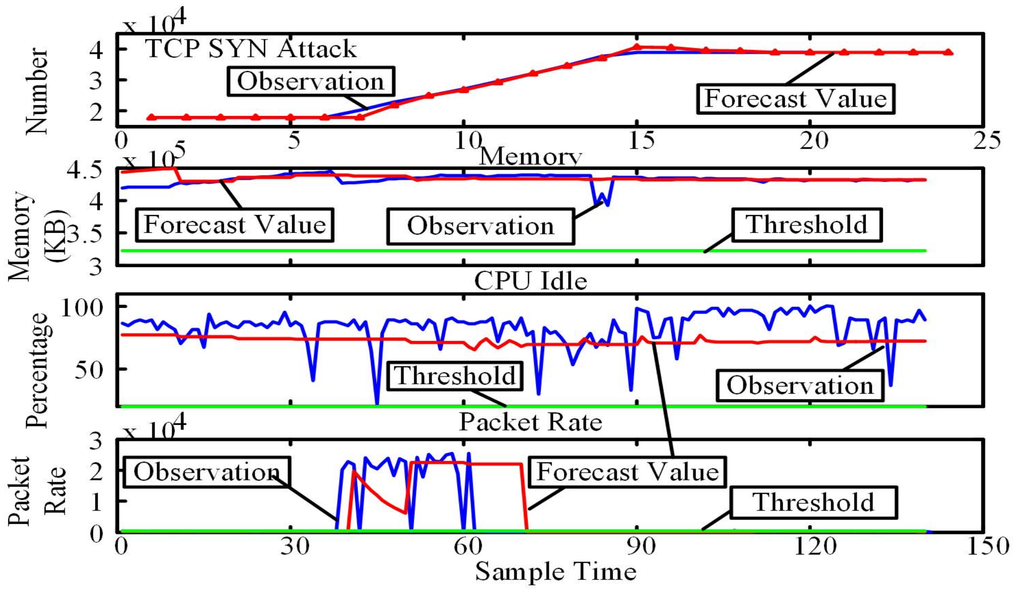


Figure 5.4

Estimated vs Measured Parameter Values for TCP SYN Attacks

5.2.3 Database Insertion Web Attacks

This experiment simulates a malicious user bypassing the authentication process to access database resources. Adversaries carry out database insertion attacks to create new legitimate accounts so that they can escalate their privileges for the purpose of obtaining confidential information. Abnormal "malicious" insertion commands may lead to database

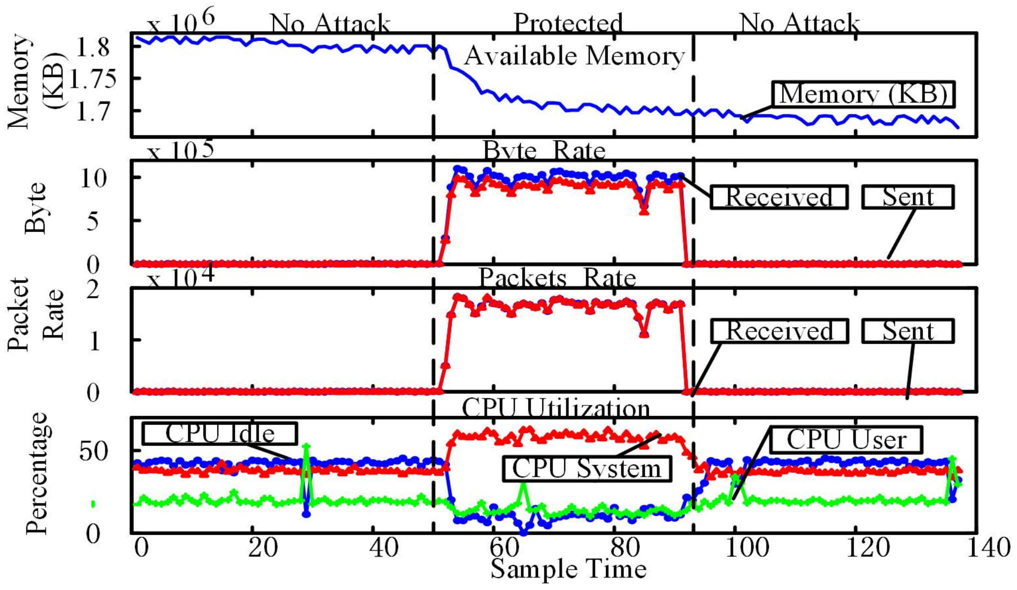


Figure 5.5

MC-VM Parameters for TCP SYN Attacks

corruption. In contrast with DoS attacks, database insertion attacks compromise system confidentiality and integrity as well as undermine the authorization policies for files stored in the database. Notably, this attack, as shown in Figure 5.8, does not significantly influence system performance as monitored by the MC-VM. For this reason, it is hard to detect that the main host was or is being compromised. The signature-based IDS is able to identify database insertion attacks even when the performance-based IDS fail to detect such attacks. **ModSecurity** (u_8) was the most appropriate response chosen by both the fuzzy-Logic and PROMETHEE II method (as shown in Table 5.4). Thus, the response initiated by the MAC is to filter malicious requests. The learning module investigated attack causes and impacts. Therefore, similar future attacks can be anticipated by the forecast-

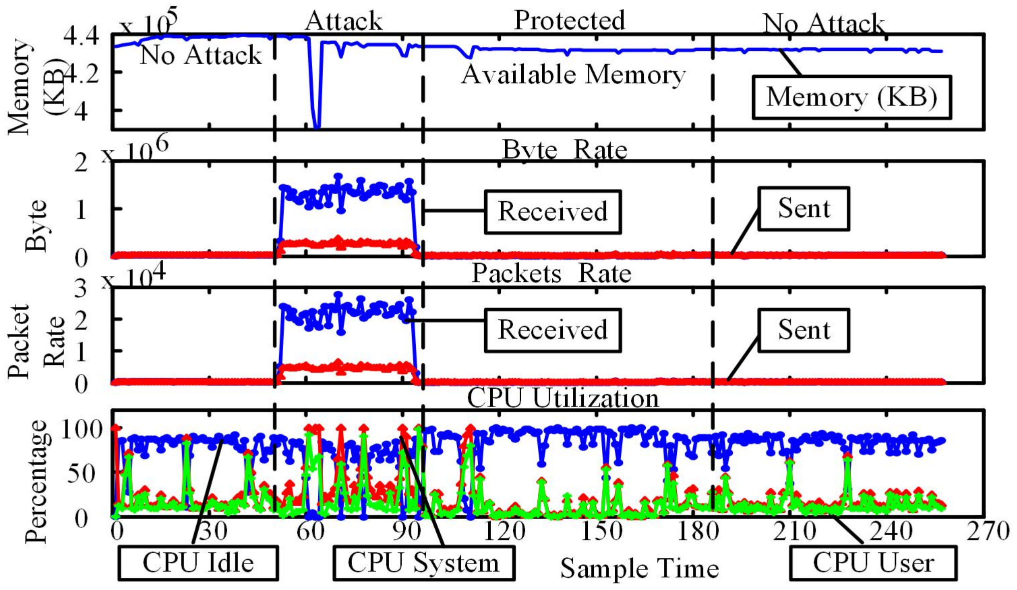


Figure 5.6

Host Parameters for TCP SYN Attacks

ing process. Figure 5.7 shows that the estimated number of database insertion attacks was similar to the measured numbers. ModSecurity kept working until the estimated number of attacks became stable, which shows that the attackers stopped compromising the system by database insertion attacks(i.e., at sample 50 in Figure 5.7).

5.2.4 HTTP Flooding Attacks

HTTP flooding attacks are DoS attacks that compromise the main host by posting numerous HTTP packets to port 80. We assumed that in each experiment only one server received the client requests. During this experiment, the IBM Websphere Application Server was listening to the port 8080 for receiving and serving to legitimate client requests. Mean-

Table 5.4

Evaluation of Protection Methods for Database Injection Attacks

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
r_1	0.3	0.3	0.2	0.2	0.2	0.3	0.2	0
r_2	0.2	0	1	1	0	0	0.6	0
r_3	0.2	0	1	1	0	0	1	1
r_4	1	1	0.8	0	1	1	1	0
r_5	0.2	0	1	1	0.5	0.5	1	0.5
r_6	1	0	0	1	0	0	1	0
r_7	0	0	1	1	0	0	0.5	0
r_8	0.2	0.2	0.2	0.8	0.5	0.8	0.4	0
Total Value (Ranking)	0.053 (5)	0.239 (2)	-0.231 (6)	-0.346 (8)	0.170 (3)	0.076 (4)	-0.306 (7)	0.346 (1)

while the port 80 of the main host was maliciously opened by adversaries; thus, a large amount of "fake" HTTP packets were sent to this port by attackers. Figure 5.10 shows that the estimation of the host *packet_recv* feature was higher than its threshold, while the estimation of *memory* was lower than its threshold. The predicted values exceed the "normal" thresholds, and in this case, the learning module runs immediately to capture the signatures of this potentially novel attack.

Before sending HTTP packets to the web server, TCP connections must be established successfully; therefore, any TCP packets sent to the destination port 80 may be suspect.

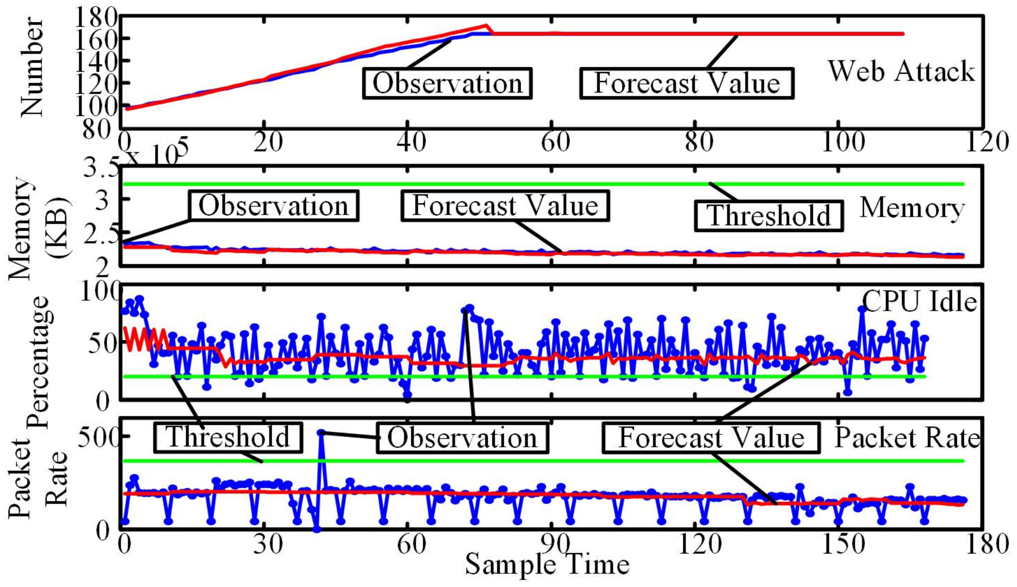


Figure 5.7

Estimated vs Measured Parameter Values for Database Insertion Attacks

These packets are therefore redirected to the MC-VM first, on which **Packet Filtering** (u_2) is the most appropriate response executed to mitigate the potential flooding attacks (as shown in Table 5.3). Figure 5.12 shows the packet filtering method efficiently deals with such an attack after 240 seconds. Figure 5.11 illustrates that suspicious flows were redirected to the MC-VM, and that malicious messages were dropped beginning at sample 120. The estimated number of HTTP flooding attacks was equal to the observed number, as shown in Figure 5.10.

Since the technique we employed to implement flow redirection can only redirect UDP, TCP, and ICMP packets to the MC-VM, the HTTP flooding attack was identified as a type of TCP flooding attack. The Packet Filtering response can block the establishment of

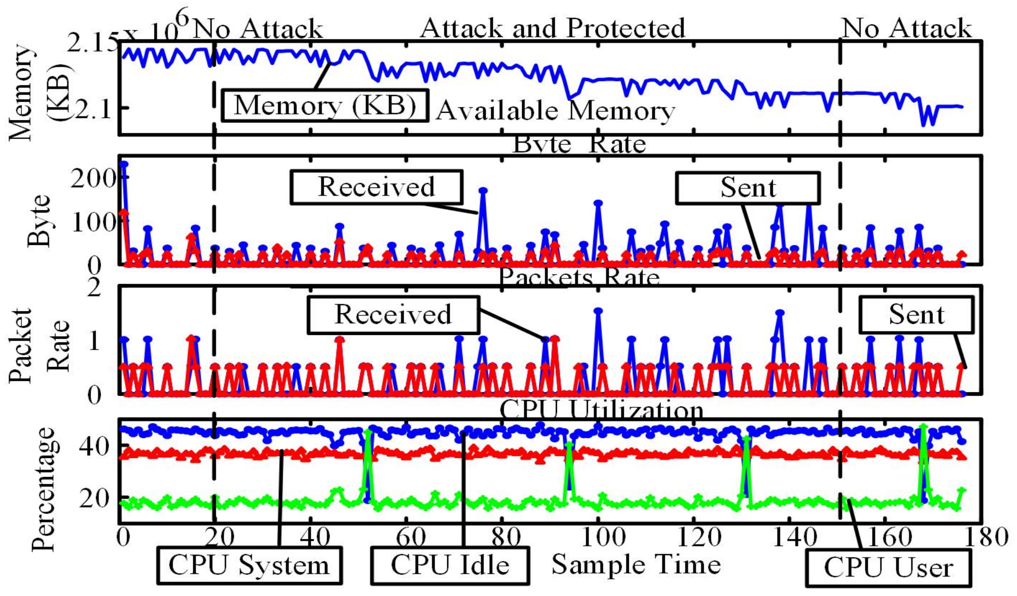


Figure 5.8

MC-VM Parameters for Database Insertion Attacks

TCP connections to protect the principal host from HTTP flooding attacks as is shown in Figure 5.12 beginning at sample 240.

5.2.5 Summary

The ASM framework is composed of forecasters, IDS, NFA, and IRS modules that can proactively anticipate upcoming attacks, detect real-time cyber assaults, investigate unknown attack signatures, and update detection algorithms, redirection rules, and responses so that similar attacks can be anticipated and prevented from occurring again. The four experiments: UDP flooding attacks, TCP SYN attacks, database insertion web attacks, and HTTP flooding attacks validate that the employment of the ASM approach successfully re-

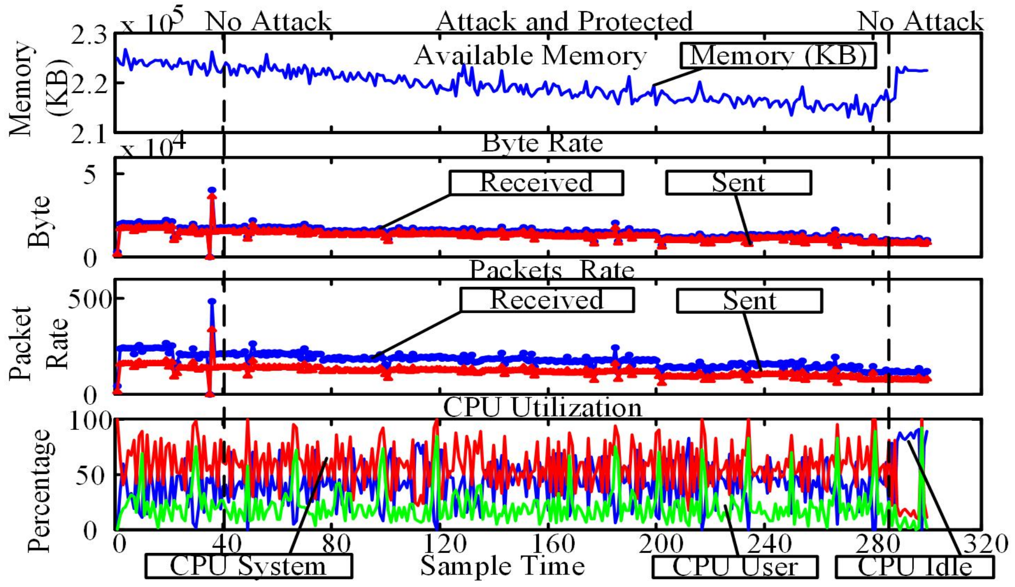


Figure 5.9

Host Parameters for Database Insertion Attacks

alizes a self-protecting system rapidly responding to and correcting the situation within at most 75 seconds. Moreover the ASM framework, in all cases was able to maintain normal services without human intervention.

5.3 Hypothesis 2: Decision Support Computing System

The DS employing the ASM approach provides both fully-autonomous and semi-autonomous functionality. When the causes and impacts of unknown attacks are ambiguous or when the most appropriate responses selected by the MAC are not sufficient to mitigate attacks, human intervention is necessary to maintain optimal system performance and protect the DS from cyber attacks. In the following experiments, CPU exhaustion and

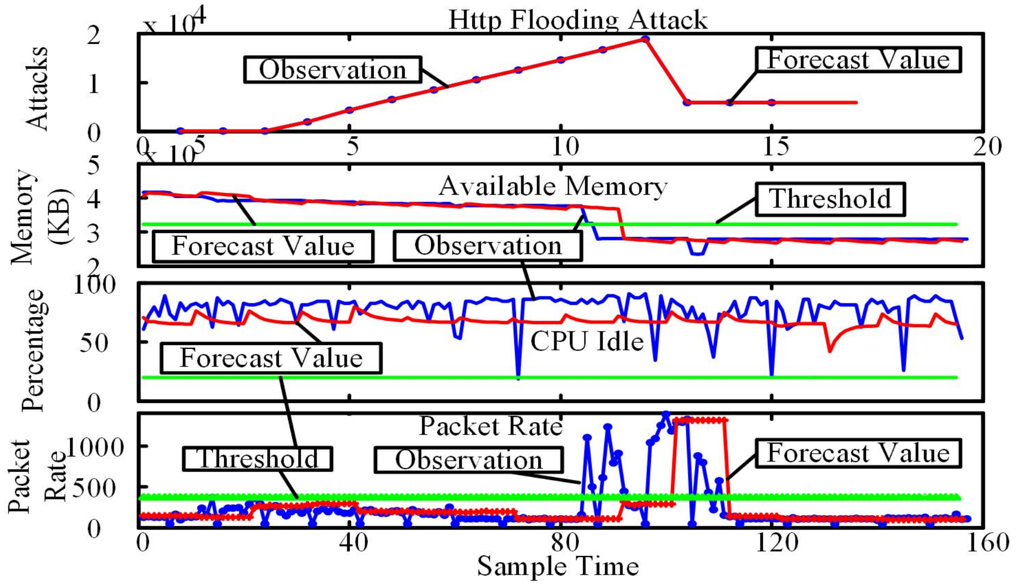


Figure 5.10

Estimated vs Measured Parameter Values for HTTP Flooding Attacks

memory exhaustion attacks were launched. Since the investigation of these attacks hardly provided any useful information, a trained system administrator was needed to choose and deploy the best responses to counter unknown attacks.

5.3.1 CPU Exhaustion Attacks

Adversaries that bypass authentication processes can send malicious commands to the web server. The processing of such requests consumes precious CPU resources from the main host. Figure 5.13 shows how, due to the impact of CPU exhaustion attacks, the estimation of the *CPU_idle* and *memory* utilization of the main host dropped below their thresholds after 130 seconds. Observations of these two features, as shown in Figure 5.15,

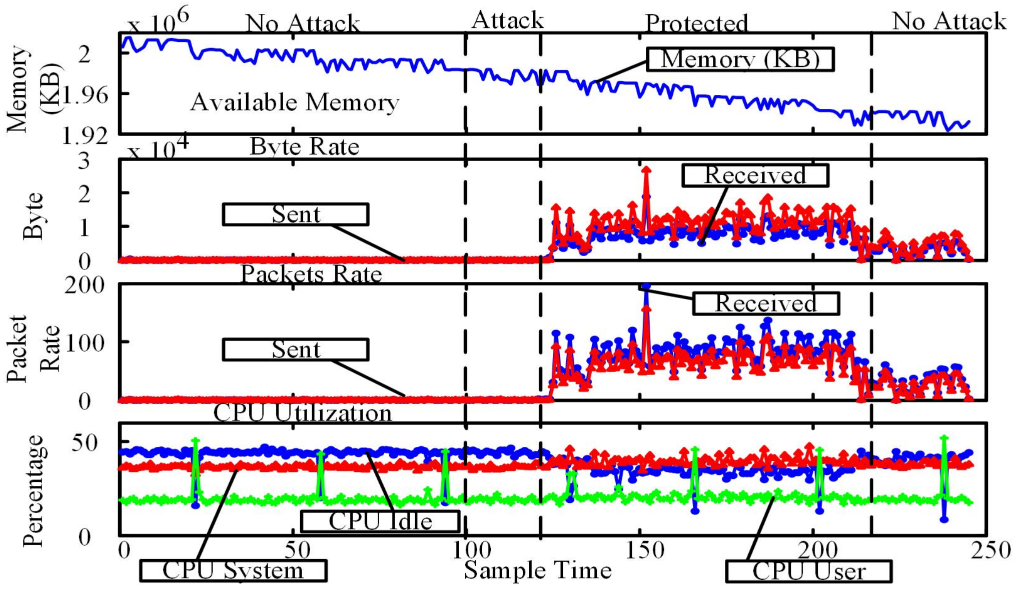


Figure 5.11

MC-VM Parameters for HTTP Flooding Attacks

reveal that they were exhausted by 120 seconds. Because of forecasting delays, in the order of 10 seconds, adversaries were able to bypass the prevention system. Subsequently, attacks were detected by the IDS installed on the main host, and the learning module captured these unknown attack signatures. However, due to the learning module's failing to investigate causes and impacts of the attacks, only limited information about this attack are derived.

Yet, IDSs were still able to detect the anomaly because the real-time observations of *CPU_idle* and *memory* deviated from the normal region. Both the fuzzy-logic and PROMETHEE II controllers determined that **Process Termination** (u_7) was the most appropriate response to release the CPU and memory resources (see Table 5.5). However, as

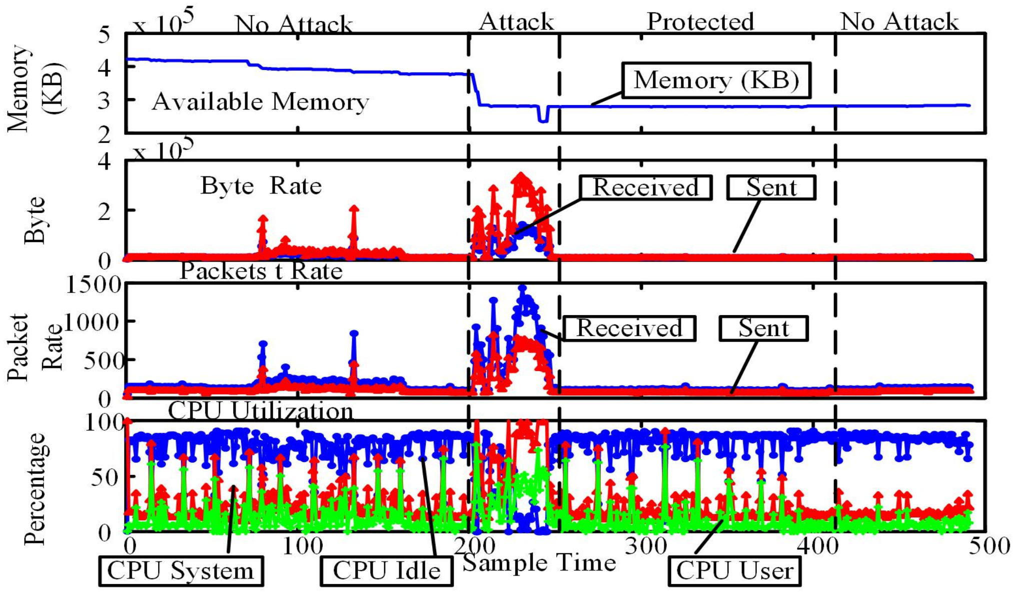


Figure 5.12

Host Parameters for HTTP Flooding Attacks

is described in Subsection 4.1.7.1, before implementing this response, human intervention is necessary to ensure that normal services are not mistakenly terminated. We discovered, unfortunately that, since the learning module was ineffective in learning the signatures of the unknown attacks that no reliable advice could be offered to system administrators. In this scenario, to safely maintain the proper performance of the main host, the system administrator decided to deploy the **Active Replica** response so that the replica server can concurrently receive and process legitimate requests. Meanwhile, the learning module continuously tries to investigate unknown attack signatures by monitoring and analyzing the performance of the main host.

Table 5.5

Evaluation of Protection Methods for CPU and Memory Exhaustion Attacks

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
r_1	0.3	0.3	0.2	0.2	0.2	0.3	0.2	0
r_2	1	1	1	1	0	0	0	1
r_3	1	1	1	1	0.5	0.5	1	1
r_4	1	1	0.8	0	1	1	0	1
r_5	1	1	1	1	0.5	0.5	1	1
r_6	1	1	0	1	0.5	0.5	1	1
r_7	1	1	1	1	0	0.5	0	1
r_8	0.2	0.2	0.2	0.8	0.5	0.8	0.4	0
Total Value (Ranking)	-0.227 (7)	-0.227 (8)	-0.023 (4)	-0.146 (6)	0.259 (2)	0.106 (3)	0.311 (1)	-0.053 (5)

The values of *byte_recv* and *packet_recv* apparent from the host parameters shown in Figure 5.15, were very low after sample 115. Since unknown attacks consumed CPU and memory resources of the main host, values of *CPU_idle* and *memory* deviated from the normal region until adversaries stopped launching attacks after sample 270. This observation stems from the fact that the authenticated client requests were sent to the replica host by the router and suspicious requests were redirected to the main hosts.

Figure 5.16 shows the performance of the replica server. From 130 seconds onwards, legitimate client requests were handled by the replica host. As a result, two features,

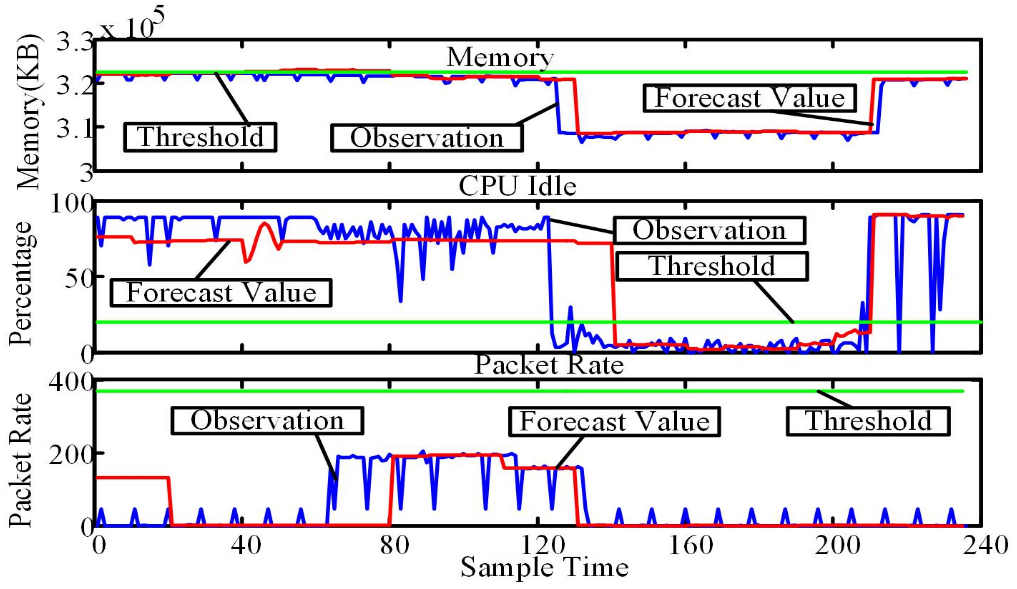


Figure 5.13

Estimated vs Measured Parameter Values for CPU Exhaustion Attacks

byte_recv and *packet_recv* of the replica host were increased, but the remaining system parameters performed normally. Figure 5.14 shows that modules on the MC-VM also worked normally.

5.3.2 Memory Exhaustion Attacks (MEA)

Signatures from worms and viruses vary, but most of them affect the performance of the main host as well as the normal services provided by the web application. In this experiment, memory exhaustion attacks were launched to simulate memory starvation of the main host caused by worms or viruses. The memory exhaustion attack is first implanted into the main host by adversaries. The web application runs next, which consumes a large

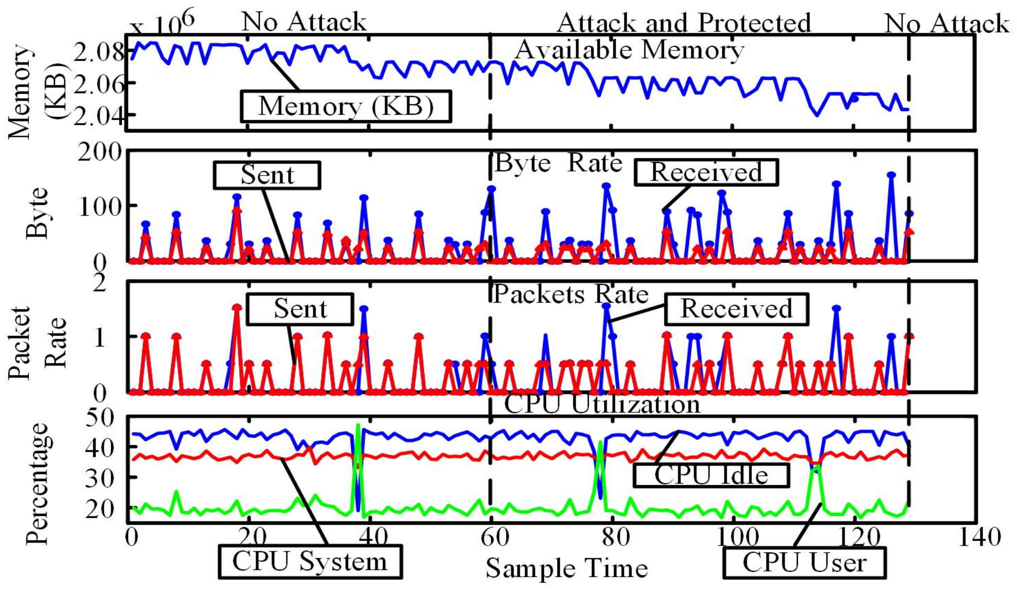


Figure 5.14

MC-VM Parameters for CPU Exhaustion Attacks

amount of memory resources starting at sample 20 as shown in Figure 5.19. Prior to sample 20, the *memory* feature value was less than 100 MB. However, by sample 20, the estimated values of selected features as shown in Figure 5.17 indicate a problem. This figure also shows that the memory utilization was correctly estimated (by the Forecasters) 10 seconds after the memory exhaustion occurred. Moreover, the forecasters correctly predicted values of the two features, *CPU_idle* and *packet_recv*. Given these estimations, the IDS was able to detect abnormal system performance. From Figure 5.19 we see that, after 60 seconds, observations of the *memory* feature returned to normal in concert with the timing of the response **Process Termination**, which was deployed by the system administrator. Therefore, the MC-VM worked properly as shown in these figures.

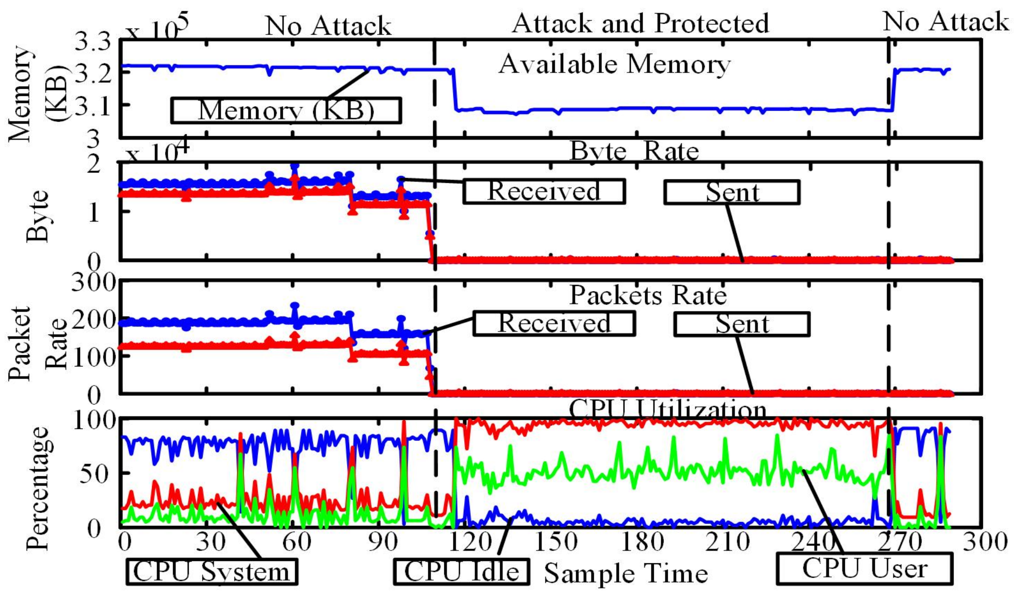


Figure 5.15

Host Parameters for CPU Exhaustion Attacks

In conclusion, the ASM framework works well applying autonomous decisions made by MAC using either one of the fuzzy-logic or PROMETHEE II methods. Furthermore, the framework supports man-in-the-loop decisions, which are necessary to recover the main host from some unknown attacks. This decision support capability can be used to avoid the potential of significant property damage or service disruption which, in exceptional cases, cannot be avoided by the autonomous controller. This is especially important for the case when little information about unknown attacks is available.

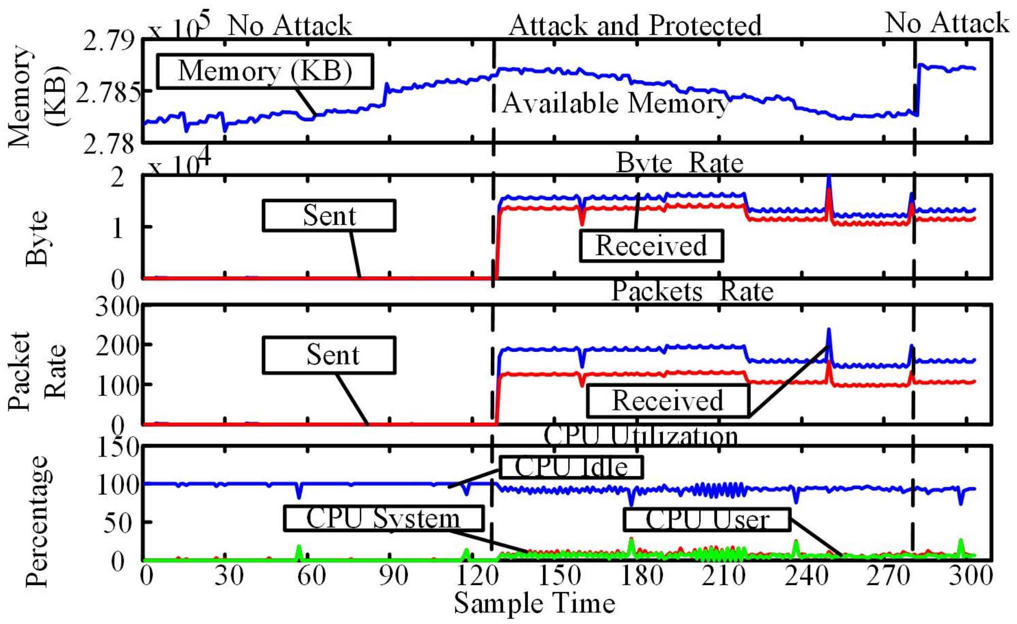


Figure 5.16

Replica Host Parameters for CPU Exhaustion Attacks

5.4 Hypothesis 3: Performance Evaluation of Decision Making Methods

In this section, SQL injection attacks were launched to undermine the confidentiality of the database server. To better understand the fuzzy-logic and PROMETHEE II decision making methods for recommended responses, Table 5.6 provides the initial values of each criterion for the eight recommended responses. Table 5.7 shows the rankings of recommended responses developed by the fuzzy-logic and the PROMETHEE II methods using the same initial values shown in Table 5.6.

The cost of the **Packet Filtering** response is equal to the cost of the **ModSecurity** response as assessed by the fuzzy-logic method. In this scenario, the MAC randomly selected

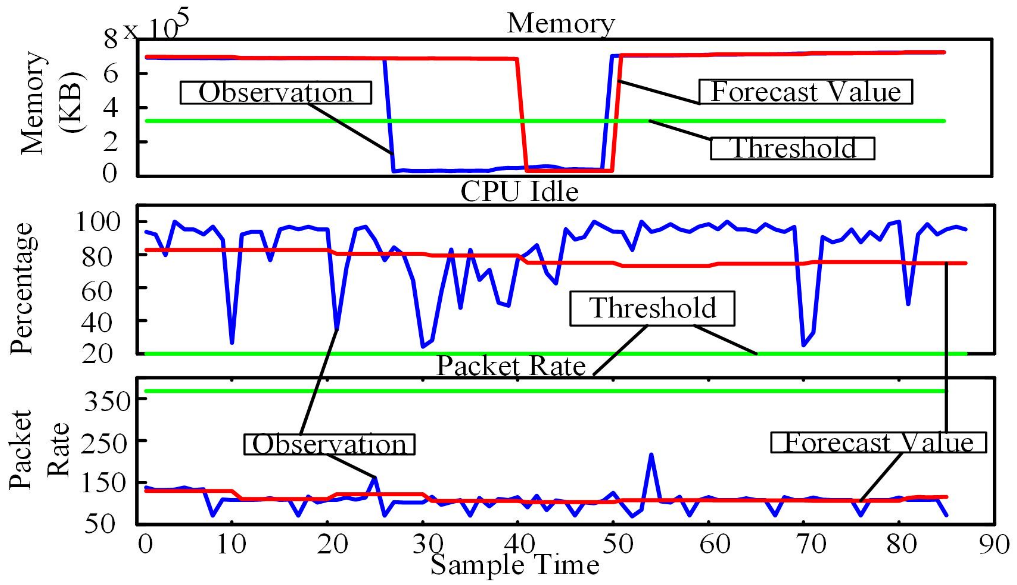


Figure 5.17

Estimated vs Measured Parameter Values for MEA

Packet Filtering to protect the main host from SQL injection attacks. The PROMETHEE II method selected the ModSecurity response which was deployed to counter the SQL injection attack because it had the largest outranking flow value, 0.3461 (see Section 4.1.7 for a description of how this value is determined).

In this experiment, adversaries injected malicious SQL commands and forced the web server to send back confidential client information. As there were 100746 records of client information stored in the database server, the total bytes sent by the web server were extremely large. In addition, the buffer size of the adversaries' machines was only 16 KB, thereby forcing client information sent by the web server to be divided into small pieces. As a result, a large amount of outgoing HTTP packets of the web server are monitored. We

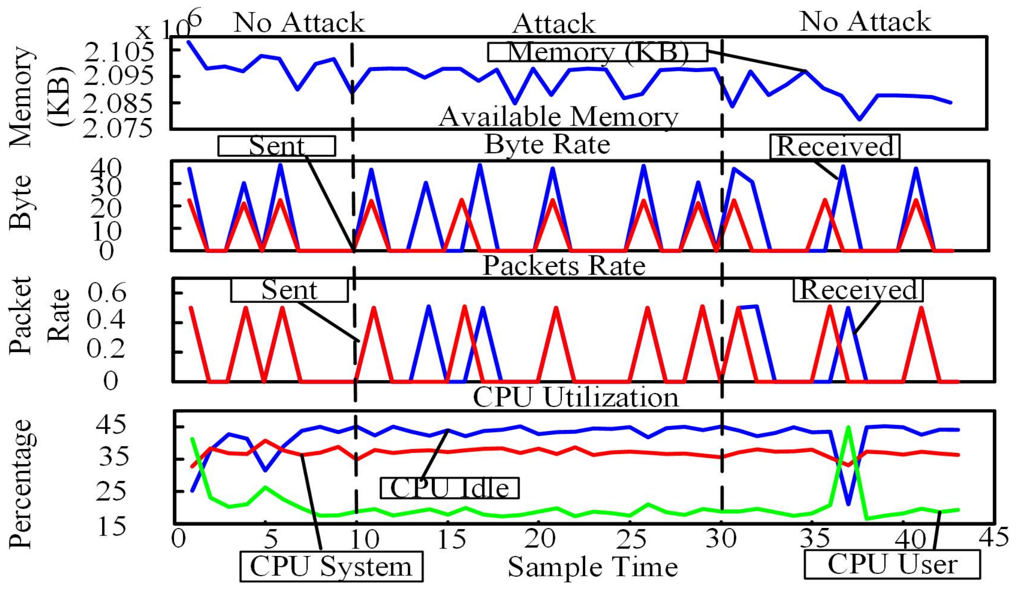


Figure 5.18

MC-VM Parameters for MEA

found experimentally, that the MAC using the PROMETHEE II method, gives a better and faster operational result than the fuzzy-logic method, namely, maintaining normal services in the face of SQL injection attacks.

- Fuzzy-logic Controller: Figure 5.21 describes the performance of the main host for this case. SQL injection attacks compromised the server at 75 seconds. Observations of the host feature, *packet_recv*, significantly differ from observations during a DoS attack on the main host. However, values of *packet_sent* and *byte_sent* for the main host were significantly increased and *memory* was decreased.

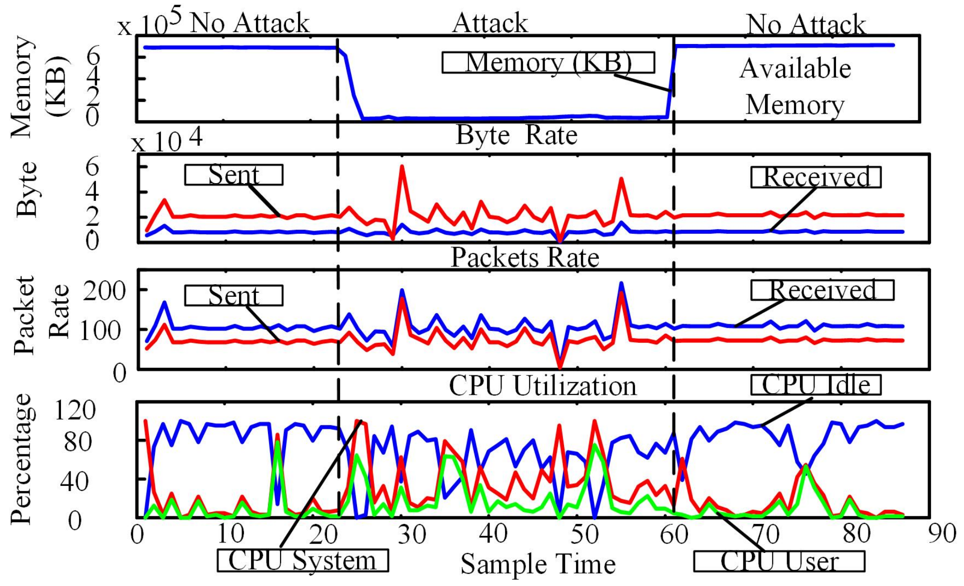


Figure 5.19

Host Parameters for MEA

The host-forecasting module predicted that the host would be exploited because the predicted *packet_recv* was higher than its threshold, as shown in Figure 5.22. After 75 seconds (see Figure 5.20), suspicious flows were analyzed and eliminated by Packet Filtering, which was the optimal response selected by the fuzzy-logic controller. In this case, SQL injection attacks were seen as TCP attacks, and the forecasting module estimated that TCP attacks would compromise the host system, as shown in Figure 5.22. From 150 seconds onwards (see Figure 5.21), the main host was protected from SQL injection attacks. Unfortunately, the web server in the main host did not receive or send any packets. Consequently, the Packet Filtering method

Table 5.6

Evaluation of Protection Methods for SQL Injection Attacks

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
r_1	0.3	0.3	0.2	0.2	0.2	0.3	0.2	0
r_2	0.2	0	1	1	0	0	0.6	0
r_3	0.2	0	1	1	0	0	1	1
r_4	1	1	0.8	0	1	1	1	0
r_5	0.2	0	1	1	0.5	0.5	1	0.5
r_6	1	0	0	1	0	0	1	0
r_7	0	0	1	1	0	0	0.5	0
r_8	0.2	0.2	0.2	0.8	0.5	0.8	0.4	0

eliminated SQL injection attacks, however this response disrupts normal service delivery.

- PROMETHEE II Controller: In the same scenario as described for the fuzzy-logic controller, this controller selected ModSecurity to defend against SQL Injection attacks. The ModSecurity module installed on the Apache server is able to block malicious SQL injection commands once they have been detected. Figure 5.25 demonstrates the estimated number of attacks increased linearly, which shows that the main host was continuously being compromised by SQL injection attacks. Once identified, the attack packets were dropped, and therefore neither the host nor the MC-VM was impacted by SQL injection attacks.

Table 5.7

The Ranking of Recommended Responses for SQL Injection Attacks

	PROMETHEE II			Fuzzy Control
Protection Method	ϕ^+	ϕ^-	ϕ (Rank)	Cost (Rank)
Snort_inline	0.2441	0.1945	0.0526 (5)	3.1 (5)
Packet Filtering	0.3195	0.1080	0.2391 (2)	1.5 (1)
Authentication Process	0.1200	0.3511	-0.2311 (6)	5.2 (6)
Active Replica	0.1140	0.4602	-0.3462 (8)	6 (8)
Network Discon.	0.2648	0.0951	0.1697 (3)	2.2 (3)
Host Shutdown	0.2342	0.1578	0.0764 (4)	2.6 (4)
Process Termination	0.0724	0.3789	-0.3065 (7)	5.7 (7)
ModSecurity	0.4255	0.0794	0.3461 (1)	1.5 (1)

Compared with the response, Packet Filtering, ModSecurity provides a faster system recovery by dropping malicious requests before these requests can posted to the database. In addition, ModSecurity does not disrupt legitimate client requests or normal services when it is run to eliminate this type of attack. Therefore, from this experiment, we conclude that the PROMETHEE II controller provides more efficient evaluations of responses than the fuzzy-logic method.

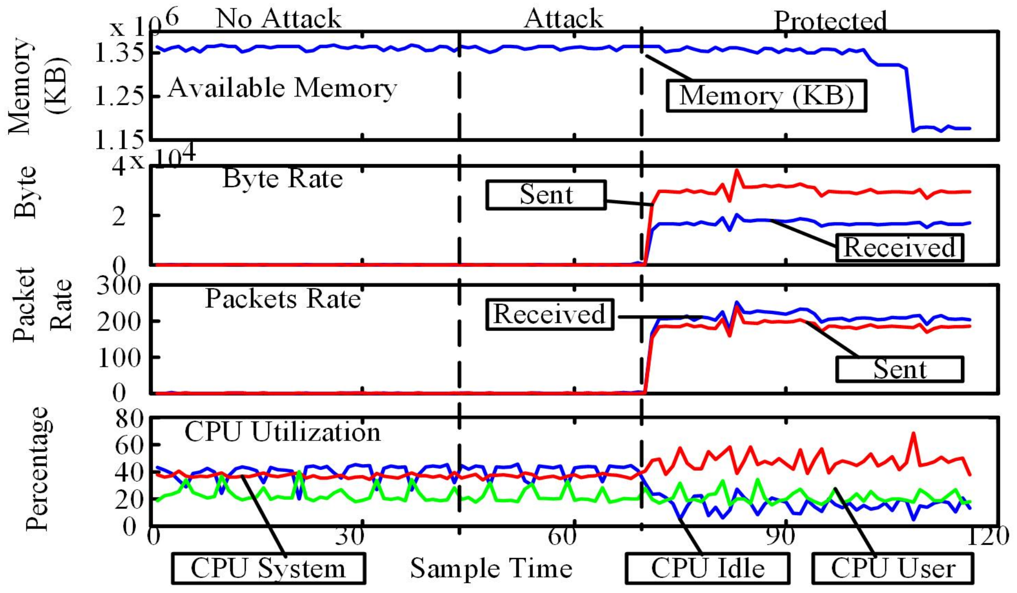


Figure 5.20

MC-VM Parameters for SQL Injection Attacks Protected by Packet Filtering

5.5 SCADA System Virtual TestBed

We also simulate three realistic SCADA-specific cyber attacks, scanning attacks, malicious parameter command injection (MPCI) attacks, and alerted alarm condition attacks. Scanning attacks gather information about the control system such as the network architecture and device characteristics. MPCI attacks inject false commands to overwrite remote terminal registers, which may interrupt normal infrastructure operations or device communications [88]. Alerted alarm condition attacks maliciously change the alarm condition values of the physical processes. In the following sections, we present a case study of a serial based SCADA system that monitors and controls a gas pipeline and a water storage tank system. Autonomic modules installed on the MC-VMs closely interact with each

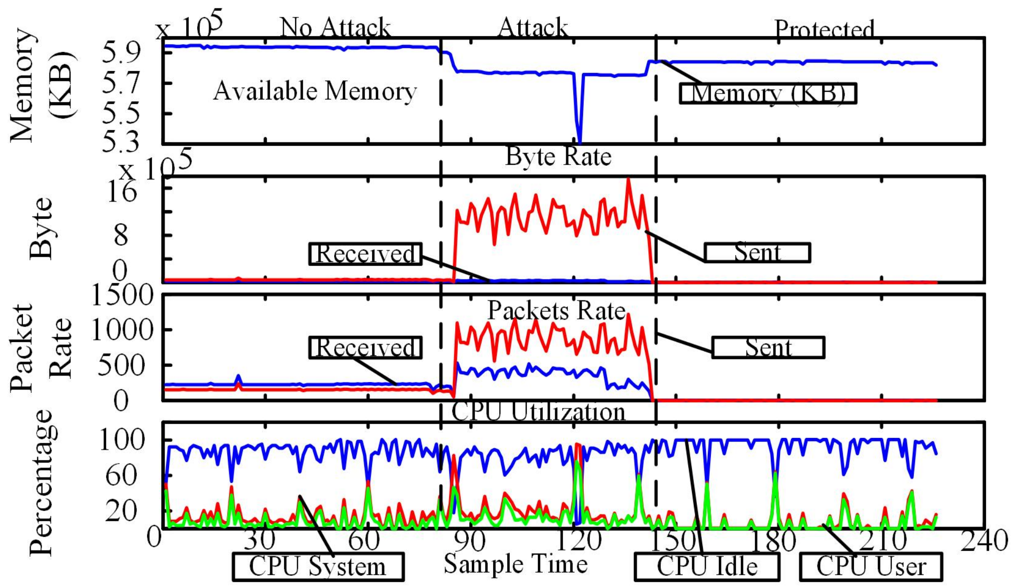


Figure 5.21

Host Parameters for SQL Injection Attacks Protected by Packet Filtering

other to protect the SCADA system from cyber assaults. Experimental results demonstrate that the application of the ASM approach efficiently realizes a self-protecting SCADA system.

5.5.1 Virtual Testbed

We use a virtual SCADA testbed developed by Reaves et al. [99] to validate the self-protection functionality of the ASM framework. The architecture of this virtual SCADA testbed [99] presents a laboratory scale ICS containing a process simulator, virtual device, configuration files, and data loggers [99]. Physical processes including the gas pipeline and water storage tank system are modeled by the process simulator. The remote terminal units

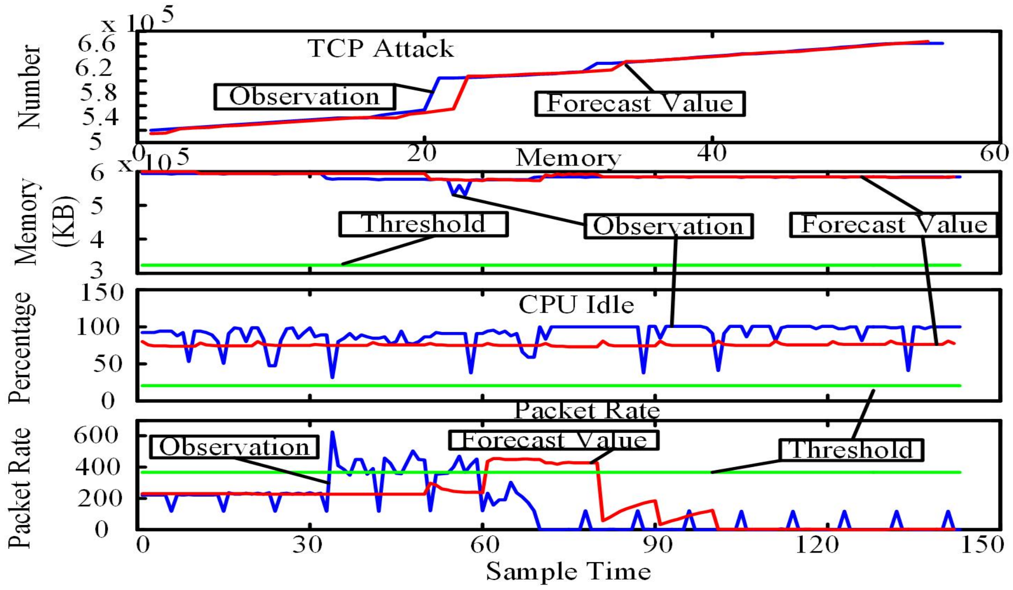


Figure 5.22

Estimated vs Measured Parameter Values Protected by Packet Filtering

(RTUs) and master terminal units (MTUs) are modeled by the virtual device. Communication protocols and transmission speeds are set/modified for process simulators and virtual devices by the configuration files. The configuration files are also used to establish/adjust connections between other testbed components.

To validate the ASM Framework, we used both cyber physical process models, namely the gas pipeline and water storage tank systems [89]. These two systems can be controlled either manually or automatically. In the manual mode system administrators turn on/off the water pump or air pump to add water into the water tank or to push air into the gas pipeline. The water level and the gas pressure are regulated by opening and closing release valves. When the water level or gas pressure exceed their set points, operators turn on release

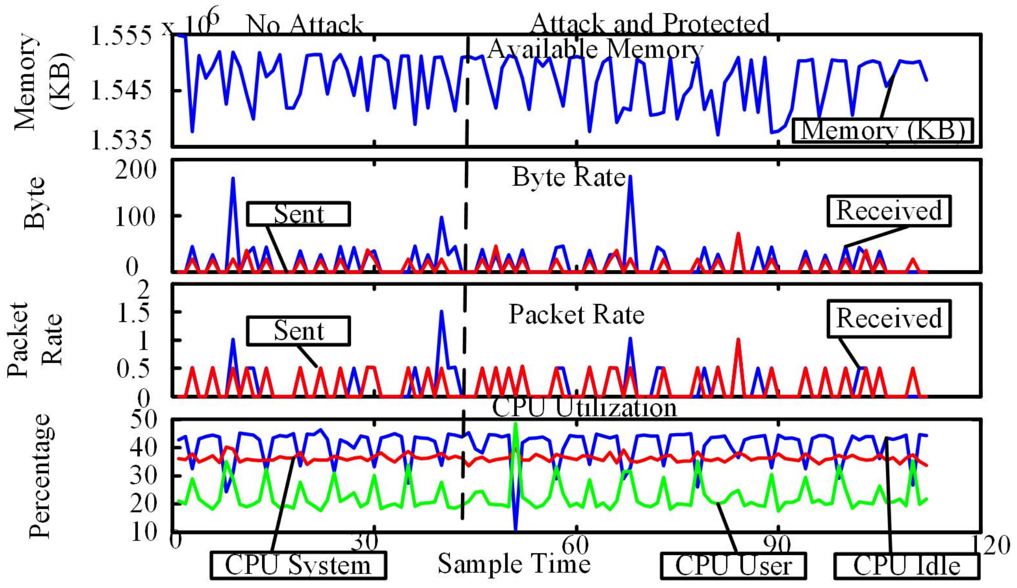


Figure 5.23

MC-VM Parameters for SQL Injection Attacks Protected by ModSecurity

valves to either drain off water or release gas. Conversely, when either the water level or gas pressure are below their set points then the release valves must be closed (manually. when using manual mode, and automatically otherwise).

When the gas pipeline system is controlled in the automatic mode, a proportional-integral-derivative (PID) control scheme containing a pressure set point and five PID set point registers (i.e., PID gain, PID reset, PID deadband, PID rate, and PID cycletime) is used by the RTU to control the gas pressure. Output registers of the RTU provide the gas pressure measured in pounds per square inch (PSI), the air pump state, and the relief valve state. Figure 5.26 shows the HMI used to monitor and control the gas pipeline system.

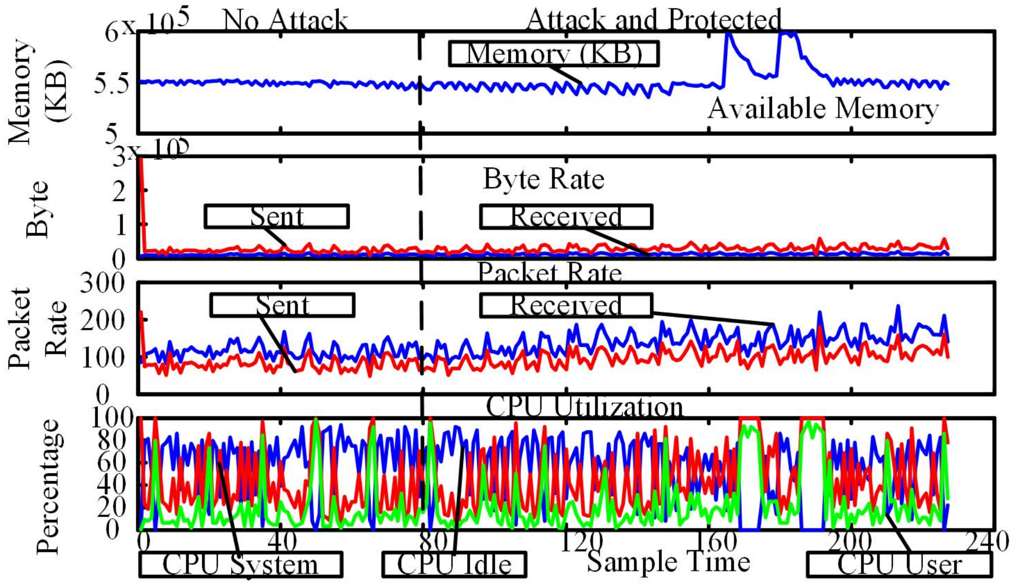


Figure 5.24

Host Parameters for SQL Injection Attacks Protected by ModSecurity

The water storage tank system's capacity is 2-liters. The water level is constantly monitored by an analog meter. The water level readout is presented as percentage of the tank capacity. In the automatic mode, the water level is maintained between high (H) and low (L) set points by turning on or off the water pump automatically. The water storage tank also activates an alarm if the water level is above or below the high high (HH) level or the low low (LL) level alarm set points (i.e., thresholds). Figure 5.27 shows the HMI used to monitor and control the water storage tank system.

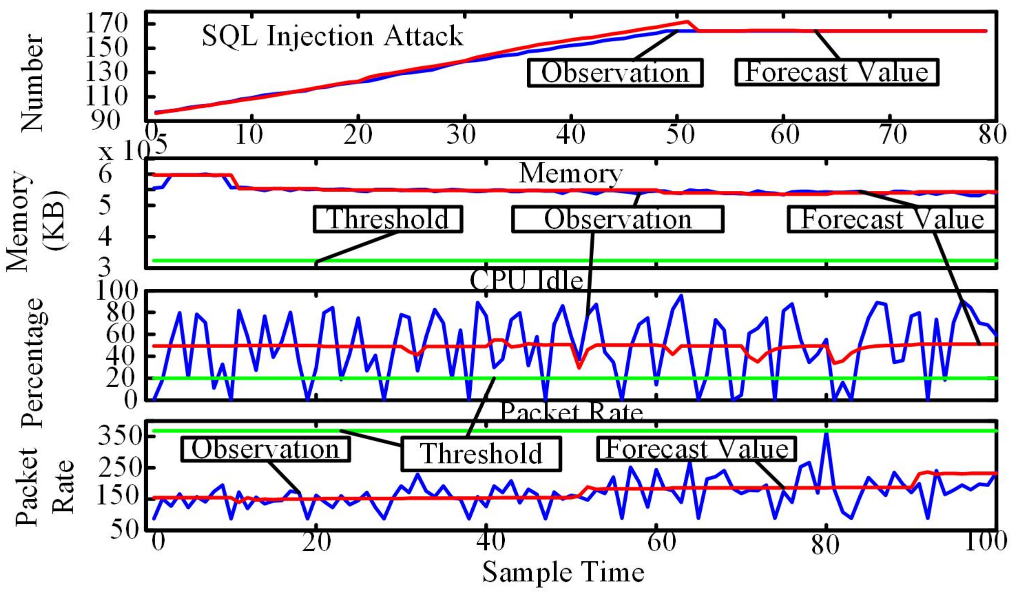


Figure 5.25

Estimated vs Measured Parameter Values Potected by ModSecurity

5.6 Hypothesis 4: Self-protecting SCADA System

In this section, we carry out function code scanning (FCS) attacks to validate the self-protection function provided by the ASM approach for SCADA systems. FCS attacks aim to collect information of SCADA systems but do not significantly impact human lives. Therefore, the fully-autonomic functions of the ASM approach can be applied to anticipate, detect, and protect ICSs without human intervention.

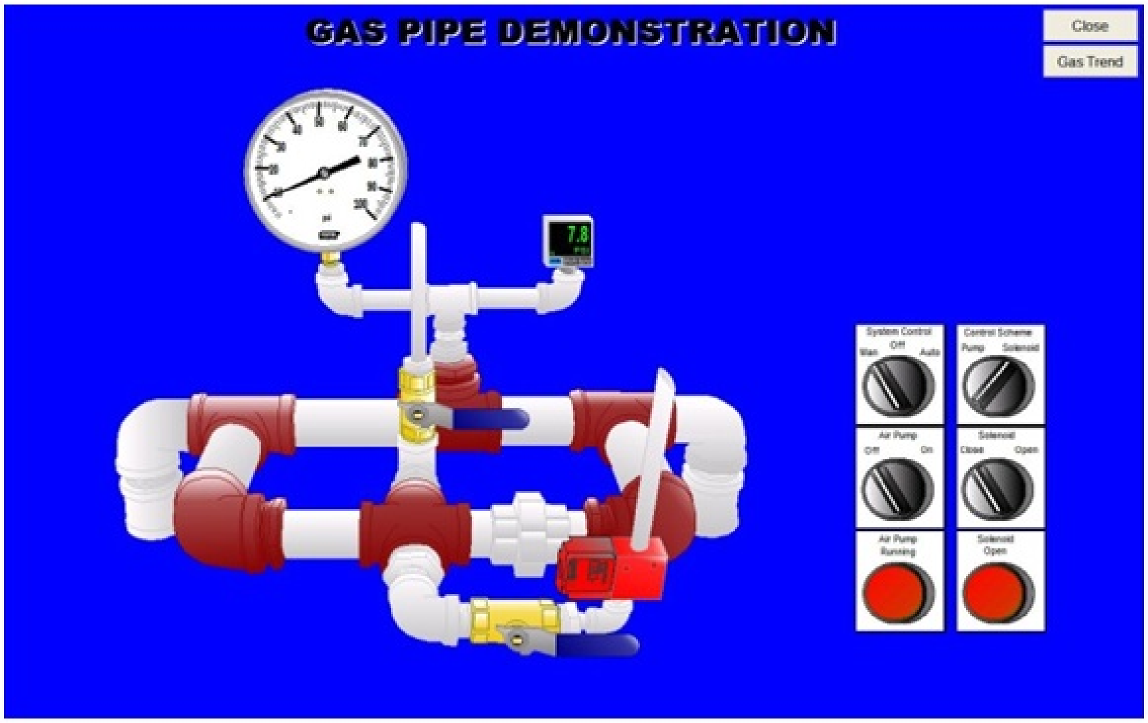


Figure 5.26

Gas Pipeline Human Machine Interface (HMI)

5.6.1 Function Code Scanning Attacks

The function code field of a Modbus TCP Frame is a single byte as shown in Figure 5.28. Attackers launch a FCS attack by sending a query with public function codes, user defined function codes, reserved function codes, and error function codes [88] to the RTU. If the attack is successful launched, the attackers can carry out disruptive attacks in the future exploiting function code vulnerabilities. The predicted gas pressure values under FCS attacks are represented in red lines shown in Figure 5.29, and the actual observations are shown as blue lines.

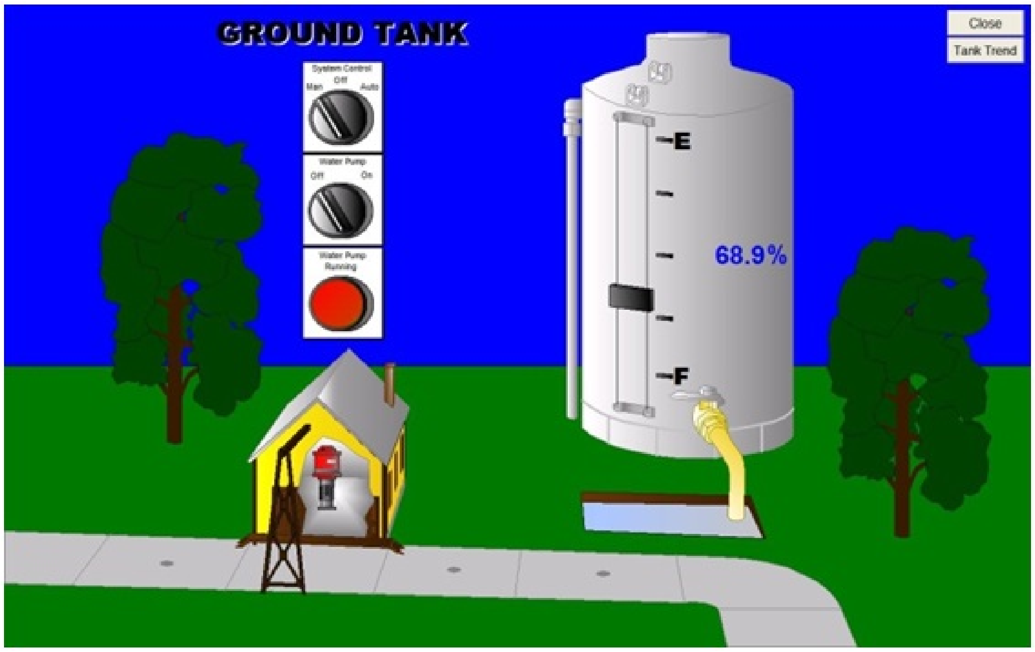


Figure 5.27

Water Tank Human Machine Interface (HMI)

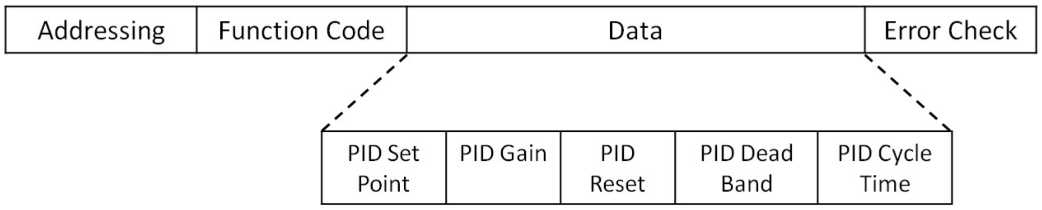


Figure 5.28

Modbus/TCP Frame

Table 5.8 is the ranking of eight recommended responses evaluated by the PROMETHEE II method for protection from the FSC attack. The initial fuzzy-values of four criteria for recommended responses were provided based on experimental results and ex-

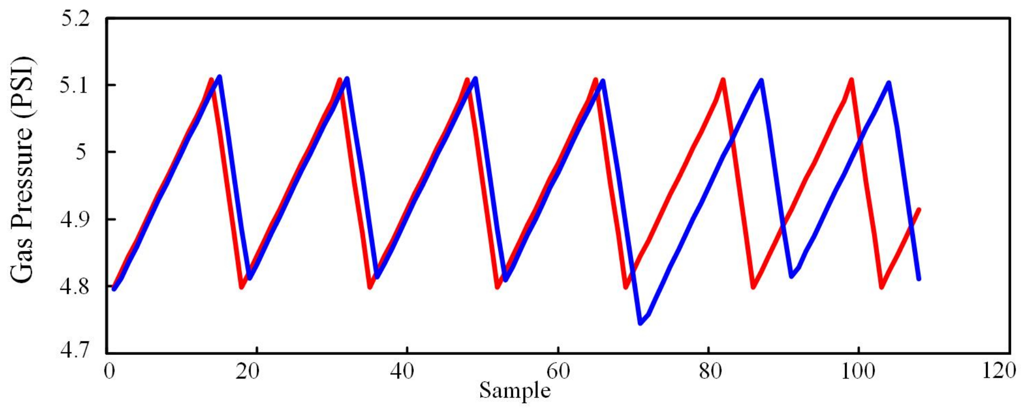


Figure 5.29

Observation and Prediction of Gas Pressure under FCS Attacks

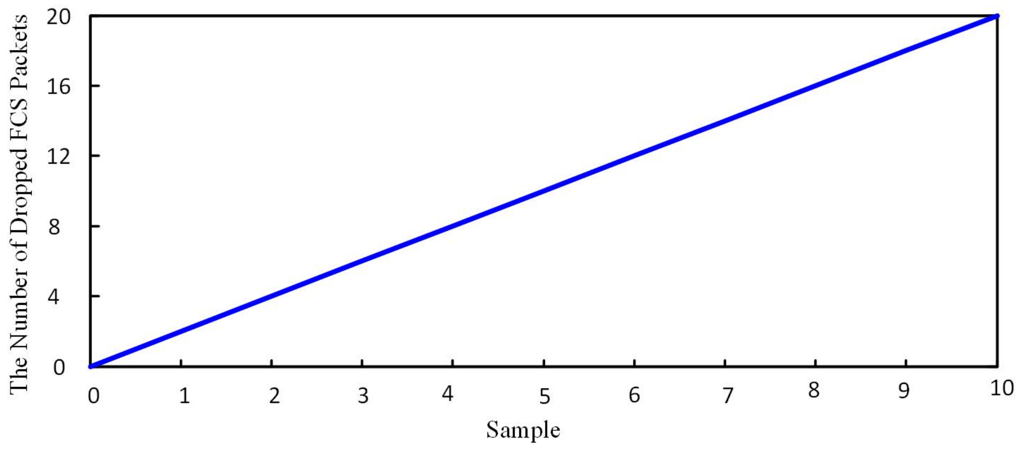


Figure 5.30

The Number of Dropped FCS Attack Packets

pert knowledge. The optimal response evaluated by the MAC using the PROMETHEE II method to defend against FCS attacks is “Dropping Malicious Commands.” This response has a low impact on disruption of normal operations and is executed automatically by the

MAC (Response total values rounded to the nearest thousandth are listed in the rightmost column of Table 5.8). The number of dropped FCS packets are shown in Figure 5.30. Since all scanning packets have been dropped, attackers cannot obtain meaningful function codes via the attack, and the window of vulnerability is closed. Note that we do not demonstrate the gas pressure trend under the FSC attack because this attack does not influence the normal performance of the gas pipeline system.

Table 5.8

Assessment of Recommended Responses Example for FCS Attacks

Ranking	Response	C1	C2	C3	C4	Total Value (Auto/ Semi-Auto)
1	Dropping Malicious Commands	0	0.3	0	0.2	0.165 (Auto)
2	Packet Filtering	0	0.5	0.4	0.2	0.077 (Auto)
3	Replacement of Compromised Devices	0	1	0	0.6	0.054 (Semi-Auto)
4	One time Authentication	0.5	0.5	0.2	0.2	-0.002 (Auto)
5	Serial Port Disconnection	0	0.8	0.8	1	-0.034 (Semi-Auto)
6	Network Disconnection	0	0.8	1	1	-0.058 (Semi-Auto)
7	Termination of Physical Processes	0	1	1	1	-0.089 (Semi-Auto)
8	Isolation of Compromised Devices	1	0.8	0.4	0.6	-0.113 (Semi-Auto)

5.7 Hypothesis 5: Decision Support SCADA Systems

ICS must provide 24/7 services; therefore, the protection of ICS security cannot disrupt normal operational commands. For example, the implementation of responses such as *Network Disconnection* and *Termination of Physical Processes* can eliminate cyber attacks and protect physical systems; however, these responses also disrupt normal functions provided by the process control system. In addition, preventing the implementation of responses which may result in high impact calamities (e.g., financial loss, property damage, and safety), require that man-in-the-loop intervention must be enabled to choose and deploy the best responses countermeasure. In the following sections, we demonstrate the semi-autonomous functions of an ICS applying the ASM approach.

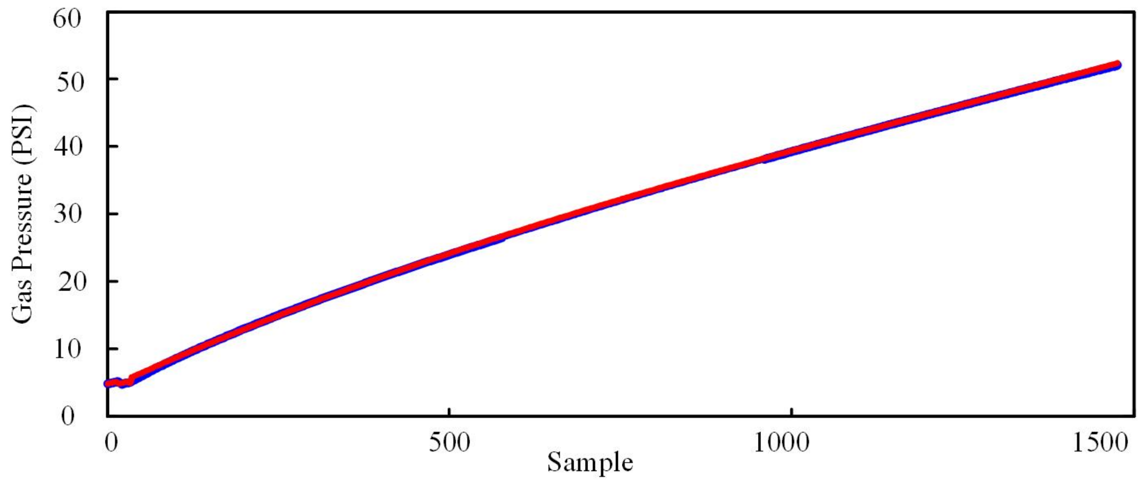


Figure 5.31

Observation and Prediction of Gas Pressure under MPCI Attacks

5.7.1 Malicious Parameter Command Injection Attacks

We carried out a MPCI attack that modifies the PID gain and the gas pressure set point value in the data field of the Modbus/TCP frame. Normally, the gas pipeline infrastructure is controlled in an “Auto” mode, in which case gas pressure is maintained between 4.75 PSI and 5.11 PSI. The MPCI attack overwrites remote terminal registers. Meanwhile the gas pressure set point is changed from 5.00 to 2.93, and the PID gain is changed from 115 to 1.05610820582e-38. As a result, the “Auto” control mode can no longer maintain the proper gas pressure, causing gas pressure to increase sharply. The predicted gas pressure values under MPCI attacks are represented in red lines shown in Figure 5.31, and the actual observations are shown as blue lines.

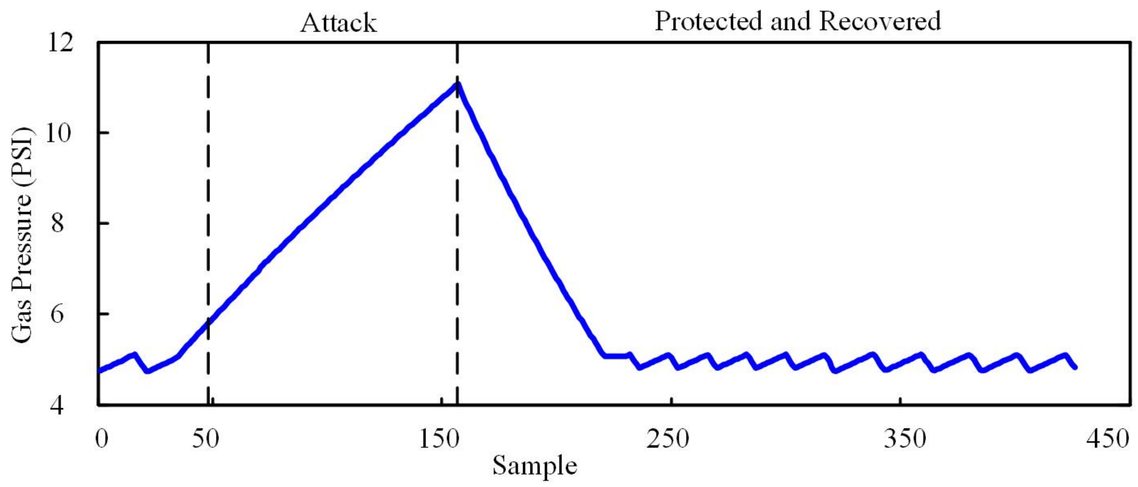


Figure 5.32

Protecting and Maintaining Gas Pressures by the ASM Approach

Table 5.9 lists the ranking of recommended responses (i.e., countermeasures) to MPCI attacks. Since this attack has never seen before, the live forensic analysis tool captures and investigates the suspicious packets after the attack has been detected by the IDS. Attack signatures such as PID gain and gas pressure set point values are analyzed, and these signatures are used to update detection algorithms as well as the rank of the eight pre-defined responses. "Packet Filtering" is executed by the MAC to protect the system against future MPCI attacks from sample 155 onwards shown in Figure 5.32. Meanwhile, the experienced system administrator regulates the set point and PID gain back to normal values at sample 220. Therefore, after sample 230, the system is set back to "Auto", and the gas pressure returns back to the normal region even in the face of continuous and similar MPCI attacks.

5.7.2 Alerted Alarm Condition Attacks

A malicious command modifying the alarm condition is carried out to disrupt normal operations of a water storage tank when the water storage tank was set to the "Auto" control mode. In a general case, the pump turns on when the water level reaches the low alarm condition (represented by L). When the water level increases to the high alarm condition (denoted by H), the pump turns off automatically. The attack first evades the authentication process, and then sends an illicit command to change L setpoint from 50.00% to 40.00%, and the alter H setpoint from 60.00% to 70.00%. HH (the high high alarm) setpoint is

Table 5.9

Assessment of Recommended Responses Example for MPCI Attacks

Ranking	Response	C1	C2	C3	C4	Total Value (Auto/Semi-Auto)
1	Packet Filtering	0	0.5	0.4	0.2	0.093 (Auto)
2	Replacement of Compromised Devices	0	1	0	0.6	0.069 (Semi-Auto)
3	Dropping Malicious Commands	0.5	0.3	0	0.2	0.040 (Auto)
4	One time Authentication	0.5	0.5	0.2	0.2	-0.016 (Auto)
5	Serial Port Disconnection	0	0.8	0.8	1	-0.018 (Semi-Auto)
6	Network Disconnection	0	0.8	1	1	-0.042 (Semi-Auto)
7	Isolation of Compromised Devices	0.3	0.8	0.4	0.6	-0.053 (Semi-Auto)
8	Termination of Physical Processes	0	1	1	1	-0.074 (Semi-Auto)

modified to 80.00% from 70.00% and LL (the low low alarm) is changed to 10.00% from 20.00%.

In the normal case, the water level of the water storage tank is controlled between 50.00% and 60.00% as shown from sample 1 to sample 67 in Figure 5.33. After sample 67, the water level is increased to 82.82% by the malicious command. The highest value of the water level shown in the figure is higher than the HH setpoint. Afterwards, the pump is turned off, and the water level drops to 40.11%. The control mode of the water storage tank is still “Auto.” Thus, at sample 130, the pump is turned on again, and the water level

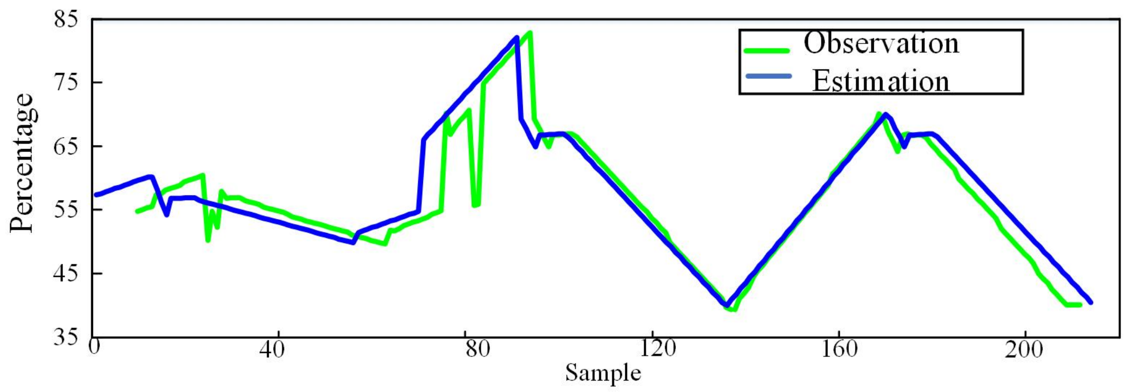


Figure 5.33

Observations vs Estimations of the Water Level without the ASM Approach

increases to 69.07%. After the water level increases to the modified H value, the pump is turned off, and the water level then eventually drops down to 40.0%.

Table 5.10 shows the rankings of recommended responses for the protection of the SCADA system from the AAC attack. The initial fuzzy-values of four criteria for recommended responses were provided based on experimental results and expertise knowledge. The optimal response determined by the MAC is “Replacement of Compromised Devices.” As this response may have a high impact on disruption of normal operations, the implementation of the response must be authorized manually. Figure 5.34 shows that, at sample 94, the attack modified the alarm thresholds, and that the water level is increased to an abnormal 65.99%. At sample 104 when “Replacement of Compromised Devices” is implemented, a replica RTU replies to the MTU who in turn sends commands to control the water level of this critical infrastructure. As a result, the water level is regulated rapidly

Table 5.10

Assessment of Recommended Responses Example for AAC Attacks

Ranking	Response	C1	C2	C3	C4	Total Value (Auto or Semi-Auto)
1	Replacement of Compromised Devices	0	0.5	0	0.5	0.141 (Auto or Semi-Auto)
2	Packet Filtering	0	0.5	0.4	0.2	0.085 (Auto)
3	Dropping Malicious Commands	0.5	0.3	0.1	0.2	0.049 (Auto)
4	One time Authentication	0.8	0.3	0.2	0.2	-0.010 (Auto)
5	Serial Port Disconnection	0	0.8	0.8	1	-0.026 (Semi-Auto)
6	Termination of Physical Processes	0	0.8	1	1	-0.052 (Semi-Auto)
6	Network Disconnection	0	0.8	1	1	-0.052 (Semi-Auto)
8	Isolation of Compromised Devices	0.5	0.8	0.8	0.6	-0.137 (Semi-Auto)

back to normal demonstrating the proficient and efficient application of the ASM framework.

The other five responses, *Network Disconnection, Serial Port Disconnection, One time Authentication, Termination of Physical Processes, and Isolation of Compromised Devices*, are less efficient than the three selected responses. This is true because the execution of *Network Disconnection* and *Serial Port Disconnection* may block legitimate requests, which disrupts normal operations. *Termination of Physical Processes* temporarily terminates the gas pipeline infrastructure, which may result in severe financial loss. *One time Authentication* and *Isolation of Compromised Devices* can reduce the risks and enhance the SCADA system security. However, the implementation of these two responses is more

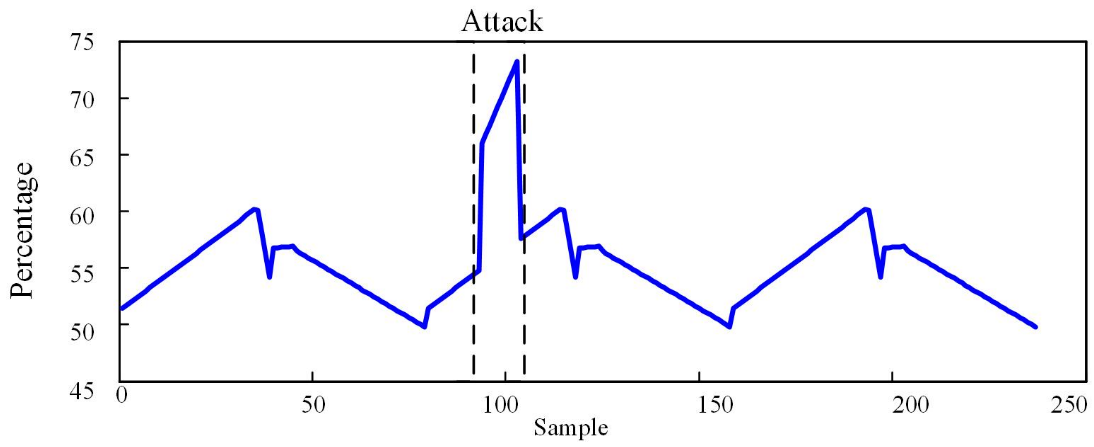


Figure 5.34

Protecting and Maintaining Water Level by the ASM Approach

expensive than *Dropping Malicious Commands* and *Packet Filtering*, which only drop malicious packets or filter all messages sent from illegitimate hosts.

5.8 Summary

In this chapter we attacked a three-tier web application and a SCADA system using multiple real-world cyber attack scenarios to validate the effectiveness of the ASM framework. The ARIMA model is used to anticipate the upcoming attacks. Both the performance-based and anomaly-based IDSs identify and classify known and unknown attacks. The network forensics analysis techniques and tools investigate and capture unknown attack signatures. The MAC uses the fuzzy-logic or PROMETHEE II method to evaluate eight candidate responses, and the most appropriate responses are deployed to counter a particular attacks. We also include one experiment for evaluating the performance of two different optimal responses selected by the fuzzy-logic and PROMETHEE II methods. The experiment demonstrates that given the same input values, the response selected by PROMETHEE II is more cost-effective.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

Cyber security enhancement and maintenance are becoming crucial challenges due to the complexity and the limits of human capacities,vulnerabilities at all layers of the stack, defects within security mechanisms, and security concerns created by the pervasive use of computers in critical large scale distributed system (DS) deployments. In the recent years, we have seen a significant increase of cyber incidents in both enterprise and industrial control systems (ICSs). For example, the 2013 Target data breach and the 2010 Stuxnet worm. There is great urgency for developing innovative solutions to effectively improve the security of cyberspace, avoid losses, as well as normal levels of computing services and function, at low cost and with little or no human intervention.

6.1 Research Objective

The objective of this research is to apply and validate a comprehensive autonomic computing framework and methodology to self-protect DSs from known and unknown cyber attacks while maintaining the highest possible level of underlying service performance.

6.2 Findings

This research has developed an autonomic security management (ASM) framework, and validated that the ASM framework has the ability to self-protect in dynamic environments. An offline risk assessment is necessary to characterize the system boundary and select relevant features for identifying the "normal" operational region of the DS cases study. Classifying attack patterns according to malicious actors' motivations and goals is necessary to ensure efficient and effective operation, within a given context of the ASM framework.

The forecaster module of the ASM framework correctly anticipates upcoming known cyber attacks and produces early warnings. Prevention mechanisms are deployed early enough to significantly reduce cyber risks. We ascertained specific data mining techniques and pattern matching algorithms to produce highly accurate detection/prevention rates for both known and unknown attacks using an intrusion detection system (IDS) approach. Novel attack signatures and their impacts on system, network, and security features were investigated by the adaptive learning module in real-time. In addition, intrusion detection and response evaluation algorithms, as well as the set of recommended responses were updated dynamically in near real-time.

Moreover, the ASM framework also contains a MAC (multi-criteria analysis controller), which selects and implements the most appropriate responses from a set of candidate responses to eliminate or mitigate attack impacts, recover system performance back to normal, and maintain normal system operations with a low overhead. To this effect, we studies two case studies, namely, a web application system and an ICS and the outcomes from this research include the architecture, design, and implementation of the framework, which provides close interaction between forecasting modules, IDSs, adaptive learning modules, and the MAC. Through the use of experimentation including experimental attack scenarios, the core self-protection functionality has been validated. This was shown empirically for both a real-life web application and a realistic ICS. Normal operational service and functioning of these systems were maintained without violating CIA requirements.

6.3 Contributions

The research contribution of this thesis follows from 1) demonstrating self-protection features of the ASM framework, 2) utilizing autonomic computing technology to proactively protect and avoid the impacts from cyber assaults, and 3) maintaining the highest possible level of underlying service performance and integrity with little or no human intervention. Specifically, this research thesis makes the following contributions by virtue of the following functionality and outcomes:

- The ASM approach framework can switch between fully-autonomous and semi-autonomous modes; can be tailored to various computing environment contexts; provide the autonomic property for precise control and to enable the system to maintain a high level of service availability and integrity. The ASM framework can also be run in a semi-autonomous mode. In which case, a system administrator can select and initiate appropriate responses to ascertain and control situational awareness of ambiguous cyber assault signatures in concert with any multi-criteria analysis controller (MAC) determinations. This is especially, useful in scenarios of high risk consequence.

- The ASM framework is simple to configure and deploy: The main modules of the ASM approach are installed on a MC-VM. As a result, the autonomic computing system can be easily realized by adding the MC-VM to most popular networked platforms.

- The ASM framework supports reliable, sustainable, and resilient self-protection performance. Most of modules are installed on the MC-VM, which operates in a stealth mode, hidden from internal and external users and devices to protect itself from compromise. Hence, a partially compromised computing environment does not necessarily impact the functioning of the MC-VM. On the contrary, the MC-VM maintains the victim system behavior toward achieving their normal service delivery operations even in the presence of compromising failures irrespective of whether instigated by external or internal attackers. Knowledge of its existence does not pose a threat for all known scenarios.

6.4 Conclusions

Autonomic computing is growing rapidly for managing database systems, web servers, and cloud-based computing systems. This technology adopts the feedback control theory to monitor managed elements, analyze system performance, and select and execute appropriate plans continuously to enhance the security. The pervasive utilization of autonomic computing technology can relieve the operator from a heavy system management, work load, reduce human errors, and cut operational costs. The ASM framework developed here along with the application of different, somewhat theoretical, autonomic modules has been proven usable, effective, and extensible. This implementation realizes fully-autonomic functions including attack estimation, detection, investigation, and protection, which are the first steps in developing a general framework to protect DSs from known and unknown cyber attacks.

6.5 Future Research and Outlook for Autonomic Computing

This dissertation has presented a comprehensive step by step approach for developing an autonomic security management framework for web application and ICSs hosted in distributed environments. The development of the self-protecting DSs uses very broad design concepts. Therefore, the ASM framework is extensible and can be easily instantiated into diverse computing environments. We presented the application of the developed ASM system framework to a specific live web service and a realistic ICS deployment. However, the research contributions of this dissertation can be further extended to the following topics.

- **Internet of Things-based (IoT-based) Ecosystems:** IoT is a global network infrastructure, which has the ability to connect, communicate with, and remotely manage huge numbers of sensors and automated devices via the Internet [18, 23]. IoT technology has been applied to various fields such as retail, logistics, pharmaceutical, food, health, smart homes, and transportation (e.g., ports, rail and roadways). However, only when strong security and privacy are in place will IoT be most feasible and practical [7]. One of the future trends of IoT is towards autonomic resources since it becomes impossible to manage the increasing complexity by system administrators. The ASM approach presented here easily realizes self-protection functionality by adopting system models and integrating intrusion estimation, detection, and protection techniques.

- **Autonomic Risk Assessment:** A self-protecting system relies on the precise selection of the most relevant "security" features so that standard system security states can be efficiently derived, and the resulting deployed countermeasures can be effective. The self-protecting system uses the standard operational region to detect and classify anomalies. In complex systems, an autonomic risk assessment process must characterize the system boundary, select relevant features to be monitored, and construct the standard operation region automatically. In addition, the autonomic risk assessment must model effects of recommended responses on the system security. As far as we know, the autonomic risk assessment function does not already exist. However, An autonomic risk assessment process is a promising research topic because it is an essential module for the successful implementation of a self-protecting system.

Network Forensic Analysis: In real-world, zero-day attacks can last from 19 days to 30 months. Most realistic zero-day attacks are difficult to detect. Therefore, network forensic analysis is necessary to investigate causes and impacts of zero-day attacks. An autonomic network forensic analysis tool is a promising research topic in the near future. The autonomic network forensic analysis tool is supposed to be composed of deep-analysis systems and flow-analysis systems. Such tools have the ability to analyze log files, traffic flows, user activities, asset data, and vulnerabilities.

REFERENCES

[1] "Amazon Web Services: Overview of Security Processes," http://awsmedia.s3.amazonaws.com/pdf/AWS_Security_Whitepaper.pdf.

[2] "Network Attack and Defence," http://www.cl.cam.ac.uk/~rja14/Papers/SEv2-c21.pdf.

[3] "Security of the Internet," http://www.cert.org/encyc_article/tocencyc.html.

[4] "Target Warns Customers of Scams after Massive Data Breach," http://www.foxnews.com/us/2013/12/24/target-warns-phishing-emails-after-massive-data-breach/.

[5] "KDD Cup 1999 Data,", 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[6] "TJMaxx Data Breach," 2007, http://www.consumeraffairs.com/tj-maxx-data-breach.

[7] "Internet of Things in 2020 Roadmap for the Future," Sep. 5th 2008, http://www.smart-systems-integration.org/public/documents/publications/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v3.pdf.

[8] "SQL Injection Protection: A Guide on How to Prevent and Stop Attacks,", 2009, http://searchsecurity.techtarget.com/tutorial/SQL-injection-protection-A-guide-on-how-to-prevent-and-sto-attacks\#SQL4.

[9] "OWASP Secure Coding Practices Quick Reference Guide," 2010, https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf.

[10] "Roadmap to Achieve Energy Delivery Systems Cybersecurity," 2011, http://energy.gov/sites/prod/files/Energy%20Delivery%20Systems%20Cybersecurity%20Roadmap_finalweb.pdf.

[11] "Making Security Measurable-Vulnerability Management," 2012, http://measurablesecurity.mitre.org/directory/areas/vulnerabilitymanagement.html.

[12] "2013 Data Breach Investigations Reort," 2013, http://middlegateinc.com/whitepapers/2013_Data_Breach_Investigation_Report.pdf.

[13] "The ARIMA Procedure," 2013, http://www.okstate.edu/sas/v8/saspdf/ets/chap7.pdf.

[14] "CYBERSECURITY A Better Defined and Implemented National Strategy is Needed to Address Persistent Challenges," 2013, http://www.gao.gov/assets/660/652817.pdf.

[15] "HP Reveals Cost of Cybercrime Escalates 78 Percent, Time to Resolve Attacks More Than Doubles," 2013, http://www8.hp.com/us/en/hp-news/press-release.html?id=1501128\#.U254PIFdWSo.

[16] "Httperf," 2013, http://www.hpl.hp.com/research/linux/httperf/httperf-man-0.9.txt.

[17] "Innovations," 2013, https://www.thalesgroup.com/sites/default/files/asset/document/thales_innovations1_eng_3.pdf.

[18] "IoT Privacy Data Protection Information Security," 2013, http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=1753.

[19] "OWASP Top 10-2013 The Ten Most Critical Web Application Security Risks," 2013, https://www.owasp.org/index.php/Top_10_2013-Top_10.

[20] "Snort_Inline," 2013, http://snort-inline.sourceforge.net/oldhome.html.

[21] "Wireshark," 2013, http://www.wireshark.org/about.html.

[22] "RotationForest," 2014, http://weka.sourceforge.net/doc.packages/rotationForest/weka/classifiers/meta/RotationForest.html.

[23] "Security in the Internet of Things," 2014, http://www.windriver.com/whitepapers/security-in-the-internet-of-things/wr_security-in-the-internet-of-things.pdf.

[24] S. Abdelwahed, J. Bai, R. Su, and N. Kandasamy, "On the Application of Predictive Control Techniques for Adaptive Performance Management of Computing Systems," *Network and Service Management, IEEE Transactions on*, vol. 6, no. 4, december 2009, pp. 212 –225.

[25] S. Abdelwahed, N. Kandasamy, and S. Neema, "A Control-Based Framework for Self-Managing Distributed Computing Systems," *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, New York, NY, USA, 2004, WOSS '04, pp. 3–7, ACM.

[26] S. Abdelwahed, N. Kandasamy, and S. Neema, "Online Control for Self-Management in Computing Systems," *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, 2004, pp. 368–375.

[27] B. Al Baalbaki, Y. Al-Nashif, S. Hariri, and D. Kelly, "Autonomic Critical Infrastructure Protection (ACIP) system," *Computer Systems and Applications (AICCSA), 2013 ACS International Conference on*, 2013, pp. 1–4.

[28] D. H. J. M. Aleksandr Matrosov, Eugene Rodionov, "Stuxnet Under the Microscope," *ESET*, January 2011, `http://www.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf`.

[29] B. Amann, R. Sommer, A. Sharma, and S. Hall, "A Lone Wolf No More: Supporting Network Intrusion Detection with Real-Time Intelligence," *Research in Attacks, Intrusions, and Defenses*, vol. 7462 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2012, pp. 314–333.

[30] "Apache HTTP Server Version 2.2 Documentation-Apache HTTP Server," 2013, `http://httpd.apache.org/docs/2.2/`.

[31] M. Behzadian, R. Kazemzadeh, A. Albadvi, and M. Aghdasi, "PROMETHEE: A Comprehensive Literature Review on Methodologies and Applications," *European Journal of Operational Research*, vol. 200, no. 1, 2010, pp. 198 – 215.

[32] P. Bhatt and E. T. Yano, "Analyzing Targeted Attacks using Hadoop applied to Forensic Investigation," *The Eighth International Conference on Forensic Computer Science*, Brasilia, Brazil, 2013.

[33] G. M. H. Y.-S. W. S. B. E. H. S. Bingrui Foo, Matthew W. Glause, "Intrusion Response Systems: A Survey," *Network Security: Know It All: Know It All*, J. Joshi, ed., Morgan Kaufmann Publishers Inc., 2008, chapter 10, pp. 309–342.

[34] N. Boggs, S. Hiremagalore, A. Stavrou, and S. Stolfo, "Cross-Domain Collaborative Anomaly Detection: So Far Yet So Close," *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, eds., vol. 6961 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 142–160.

[35] D. Bolzoni, S. Etalle, and P. Hartel, "Panacea: Automating Attack Classification for Anomaly-Based Network Intrusion Detection Systems," *Recent Advances in Intrusion Detection*, E. Kirda, S. Jha, and D. Balzarotti, eds., vol. 5758 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2009, pp. 1–20.

[36] H. Bozdogan, "Model Selection and Akaike's Information Criterion (AIC): The General Theory and its Analytical Extensions," *Psychometrika*, vol. 52, no. 3, 1987, pp. 345–370.

[37] M. J. Brian Cashell, William D. Jackson and B. Webel, "The Economic Impact of Cyber-Attacks," 2004, `http://www.fas.org/sgp/crs/misc/RL32331.pdf`.

[38] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha, "Towards Automatic Generation of Vulnerability-Based Signatures," *Security and Privacy, 2006 IEEE Symposium on*, May 2006, pp. 15 pp.–16.

[39] D. F. C. Meyers, S. Powers, "Taxonomies of Cyber Adversaries and Attacks," April, 2009.

[40] I. M. Chapman, S. P. Leblanc, and A. Partington, "Taxonomy of Cyber Attacks and Simulation of Their Effects," *Proceedings of the 2011 Military Modeling & Simulation Symposium*, San Diego, CA, USA, 2011, MMS '11, pp. 73–80, Society for Computer Simulation International.

[41] Z. Chen, F. Han, J. Cao, X. Jiang, and S. Chen, "Cloud Computing-Based Forensic Analysis for Collaborative Network Security Management System," *Tsinghua Science and Technology*, vol. 18, no. 1, Feb 2013, pp. 40–50.

[42] Y.-H. Choi, M.-Y. Jung, and S.-W. Seo, "L+1-MWM: A Fast Pattern Matching Algorithm for High-Speed Packet Filtering," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008, pp. 2288–2296.

[43] S. S. D. D. Q. W. Chris Simmons, Charles Ellis, "AVOIDIT: A Cyber Attack Taxonomy. Technical Report: CS-09-003," 2009, `http://issrl.cs.memphis.edu/files/papers/CyberAttackTaxonomy_IEEE_Mag.pdf`.

[44] J. Chu, Z. Ge, R. Huber, P. Ji, J. Yates, and Y.-C. Yu, "ALERT-ID: Analyze Logs of the Network Element in Real Time for Intrusion Detection," *Research in Attacks, Intrusions, and Defenses*, D. Balzarotti, S. Stolfo, and M. Cova, eds., vol. 7462 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2012, pp. 294–313.

[45] B. Claudel, N. De Palma, R. Lachaize, and D. Hagimont, "Self-Protection for Distributed Component-Based Applications," Berlin, Heidelberg, 2006, SSS'06, pp. 184–198.

[46] A. Corrin, "Frequency, Cost of Cyberattacks on the Rise," 2013, `http://fcw.com/Articles/2013/10/08/cyberattacks-frequency-costs-rise.aspx?Page=1`.

[47] D. P. Cox, *The Application of Autonomic Computing for the Protection of Industrial Control Systems*, doctoral dissertation, Tucson, AZ, USA, 2011, AAI3473606.

[48] M. Danforth, "WCIS: A Prototype for Detecting Zero-Day Attacks in Web Server Requests," *USENIX LISA'11: 25th Large Installation System Administration Conference,Boston,MA*, 2011.

[49] "Apache Geronimo v2.0 Document: DayTrader," 2013, `https://cwiki.apache.org/GMOxDOC20/daytrader.html`.

[50] A. U. Dayu, Yang and H. J. Wesley, "Anomaly-Based Intrusion Detection for SCADA Systems," *55th Intl. Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT 05)*, Nov. 12-16, 2006., pp. 12–16.

[51] D. J. Dean, H. Nguyen, and X. Gu, "UBL: Unsupervised Behavior Learning for Predicting Performance Anomalies in Virtualized Cloud Systems," San Jose, California, USA, 2012, ICAC '12, pp. 191–200.

[52] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal Security Hardening Using Multi-Objective Optimization on Attack Tree Models of Networks," Alexandria, Virginia, USA, 2007, CCS '07.

[53] R. Dhamija and J. D. Tygar, "The Battle Against Phishing: Dynamic Security Skins," Pittsburgh, Pennsylvania, 2005, SOUPS '05, pp. 77–88.

[54] Y. Diao, F. Eskesen, S. Froehlich, J. Hellerstein, L. Spainhower, and M. Surendra, "Generic Online Optimization of Multiple Configuration Parameters with Application to a Database Server," *Self-Managing Distributed Systems*, vol. 2867 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2003, pp. 3–15.

[55] Y. Diao, J. Hellerstein, S. Parekh, R. Griffith, G. Kaiser, and D. Phung, "A Control Theory Foundation for Self-Managing Computing Systems," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 12, 2005, pp. 2213–2222.

[56] I. Dusparic and V. Cahill, "Autonomic Management of Large-Scale Critical Infrastructure," (HotAC III), June 2008.

[57] P. Düssel, C. Gehl, P. Laskov, J.-U. Bußer, C. Störmann, and J. Kästner, "Cyber-critical Infrastructure Protection Using Real-time Payload-based Anomaly Detection," *Proceedings of the 4th International Conference on Critical Information Infrastructures Security*, Berlin, Heidelberg, 2010, CRITIS'09, pp. 85–97, Springer-Verlag.

[58] S. Elnaffar, W. Powley, D. Benoit, and P. Martin, "Today's DBMSs: How Autonomic Are They," *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, 2003, pp. 651–655.

[59] M. Fisk and G. Varghese, *Applying Fast String Matching to Intrusion Detection*, Tech. Rep., Los Alamos National Laboratory, University of California San Diego, 2004.

[60] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. Spafford, "ADEPTS: Adaptive Intrusion Response using Attack Graphs in an E-Commerce Environment," *DSN 2005.*, 2005, pp. 508–517.

[61] C. Fung, "Collaborative Intrusion Detection Networks and Insider Attacks," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 2, no. 1, 2011, pp. 63–74.

[62] I. Garitano, R. Uribeetxeberria, and U. Zurutuza, "A Review of SCADA Anomaly Detection Systems," *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*, vol. 87 of *Advances in Intelligent and Soft Computing*, Springer Berlin Heidelberg, 2011, pp. 357–366.

[63] E. Gelenbe and G. Loukas, "A Self-Aware Approach to Denial of Service Defence," *Comput. Netw.*, vol. 51, no. 5, Apr. 2007, pp. 1299–1314.

[64] W. M. M. E. W. F. Glenn A. Fink, Jereme N. Haack, "A Cooperative Cyber Defense for Securing Critical Infrastructures," `http://csweb.cs.wfu.edu/~fulp/CSC790/nspw08.pdf`.

[65] P. Golle, D. Greene, and J. Staddon, "Detecting and Correcting Malicious Data in VANETs," *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, New York, NY, USA, 2004, VANET '04, pp. 29–37, ACM.

[66] H. Hacıgümüş, "SPIDER: An Autonomic Computing Approach to Database Security Management," *Secure Data Management*, vol. 4165 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2006, pp. 175–183.

[67] S. Hansman and R. Hunt, "A Taxonomy of Network and Computer Attacks," *Computers & Security*, vol. 24, no. 1, 2005, pp. 31 – 43.

[68] S. Hariri, G. Qu, R. Modukuri, H. Chen, and M. Yousif, "Quality-of-Protection (QoP)-An Online Monitoring and Self-Protection Mechanism," *IEEE J.Sel. A. Commun.*, vol. 23, no. 10, Sept. 2006, pp. 1983–1993.

[69] C. Herley and D. A. F. Florêncio, "Protecting Financial Institutions from Brute-Force Attacks," *SEC*. 2008, vol. 278, pp. 681–685, Springer.

[70] V. Igure and R. Williams, "Taxonomies of Attacks and Vulnerabilities in Computer Systems," *Communications Surveys Tutorials, IEEE*, vol. 10, no. 1, 2008, pp. 6–19.

[71] A. Ikuomola, A. Sodiya, and J. Nehinbe, "A Framework for Collaborative, Adaptive and Cost Sensitive Intrusion Response System," *Computer Science and Electronic Engineering Conference (CEEC), 2010 2nd*, 2010, pp. 1–4.

[72] S. Jiang, S. Smith, and K. Minami, "Securing Web Servers Against Insider Attack," *ACSAC 2001.*, 2001, pp. 265–276.

[73] J. Kephart and D. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, jan 2003, pp. 41 – 50.

[74] J. O. Kephart, "Autonomic Computing: The First Decade," *Proceedings of the 8th ACM International Conference on Autonomic Computing*, New York, NY, USA, 2011, ICAC '11, pp. 1–2, ACM.

[75] N. Kheir, N. Cuppens-Boulahia, F. Cuppens, and H. Debar, "A Service Dependency Model for Cost-Sensitive Intrusion Response," Athens, Greece, 2010, ESORICS'10.

[76] M. Kjaerland, "A Taxonomy and Comparison of Computer Security Incidents from the Commercial and Government Sectors," *Computers & Security*, vol. 25, no. 7, 2006, pp. 522 – 538.

[77] Y. Li and N. Zhong, "Web Mining Model and its Applications for Information Gathering," *Knowledge-Based Systems*, vol. 17, 2004, pp. 207 – 217, Special Issue: Web Intelligence.

[78] Z. Li, L. Wang, Y. Chen, and Z. J. Fu, *Network-Based and Attack Resilient Length Signature Generation for Zero-Day Polymorphic Worms*, Tech. Rep., 2007.

[79] O. Linda, T. Vollmer, and M. Manic, "Neural Network based Intrusion Detection System for critical infrastructures," *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, June 2009, pp. 1827–1834.

[80] J. Liu, "Web Intelligence (WI): What Makes Wisdom Web," *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. 2003, pp. 1596–1601, Elsevier.

[81] R.-T. Liu, N.-F. Huang, C.-N. Kao, C.-H. Chen, and C.-C. Chou, "A Fast Pattern-Match Engine for Network Processor-Based Network Intrusion Detection System," *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, April 2004, vol. 1, pp. 97–101 Vol.1.

[82] G. M. Lohman and S. S. Lightstone, "SMART: Making DB2 (More) Autonomic," *Proceedings of the 28th International Conference on Very Large Data Bases*. 2002, VLDB '02, pp. 877–879, VLDB Endowment.

[83] Y. Luo, F. Szidarovszky, Y. B. Al-Nashif, and S. Hariri, "Game Theory Based Network Security," *J. Information Security*, vol. 1, no. 1, 2010, pp. 41–44.

[84] G. P. M. Uma, "A Survey on Various Cyber Attacks and Their Classification," *International Journal of Network Security*, vol. 15, no. 6, 2013, pp. 391–397.

[85] U. L. K. Manisha M. Patil, "Mitigating App-DDoS Attacks on Web Servers," *International Journal of Computer Science and Telecommunications*, vol. Volume 2, Issue 5, August 2011, pp. 13–18.

[86] P. Meunier, "Classes of Vulnerabilities and Attacks," *Wiley Handbook of Science and Technology for Homeland Security*, vol. 2, 2008, p. 1–18.

[87] G. Modelo-Howard, S. Bagchi, and G. Lebanon, "Determining Placement of Intrusion Detectors for a Distributed Application through Bayesian Network Modeling," *Recent Advances in Intrusion Detection*, R. Lippmann, E. Kirda, and A. Trachtenberg, eds., vol. 5230 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2008, pp. 271–290.

[88] T. Morris and W. Gao, "Classifications of Industrial Control System Cyber Attacks," *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security*, Leicester, UK, September 16-17 2013, p. scadaattack.

[89] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi, "A control system testbed to validate critical infrastructure protection concepts," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 2, 2011, pp. 88 – 103.

[90] R. Moskovitch, N. Nissim, and Y. Elovici, "Privacy, Security, and Trust in KDD," Springer-Verlag, Berlin, Heidelberg, 2009, chapter Malicious Code Detection Using Active Learning, pp. 74–91.

[91] W. Y. M. S. Muda, Z. and N. Udzir, "A K-Means and Naive Bayes Learning Approach for Better Intrusion Detection.," *Inform. Technol. J.*, 2010, pp. 648–655.

[92] S. Mudhakar, I. Arun, Y. Jian, and L. Ling, "Mitigating Application-Level Denial of Service Attacks on Web Servers: A Client-Transparent Approach," *ACM Trans. Web*, vol. 2, no. 3, July 2008, pp. 15:1–15:49.

[93] P. Oman and M. Phillips, “Intrusion Detection and Event Monitoring in SCADA Networks,” *Critical Infrastructure Protection*, E. Goetz and S. Shenoi, eds., vol. 253 of *IFIP International Federation for Information Processing*, Springer US, 2008, pp. 161–173.

[94] R. Oppliger, R. Hauser, and D. Basin, “SSL/TLS Session-Aware User Authentication: A Lightweight Alternative to Client-Side Certificates,” *IEEE COMPUTER*, vol. 41, no. 3, 2008, pp. 59–65.

[95] J. Owens and J. Matthews, “A Study of Passwords and Methods Used in Brute-Force SSH Attacks,”.

[96] E. S. Pilli, R. Joshi, and R. Niyogi, “Network Forensic Frameworks: Survey and Research Challenges,” *Digital Investigation*, vol. 7, no. 1, 2010, pp. 14 – 27.

[97] G. Qu, O. A. Rawashdeh, and D. Che, “Self-Protection against Attacks in an Autonomic Computing Environment,” *I. J. Comput. Appl.*, vol. 17, no. 4, 2010, pp. 250–256.

[98] S. Raju and N. Kumar, *Multicriterion Analysis In Engineering And Management*, PHI Learning, New Delhi, India, 2010.

[99] B. Reaves and T. Morris, “An open virtual testbed for industrial control system security research,” *International Journal of Information Security*, vol. 11, no. 4, 2012, pp. 215–229.

[100] J. Richter, “Distributed Protection against Distributed Brute Force Attacks,” `http://www.feec.vutbr.cz/EEICT/2011/sbornik/03-Doktorske%20projekty/08-Informacni%20systemy/08-xricht14.pdf`.

[101] Y. K. Rob J. Hyndman, “Automatic Time Series Forecasting : The Forecast Package for R,” *Journal Of Statistical Software*, vol. 27, no. 3, 2008, pp. 1–22.

[102] P. J. H. V. M. J. R. A. S. M. S. D. C. Z. Sam Lightstone, Guy M. Lohman, “Making DB2 Products Self-Managing: Strategies and Experiences,” 2006, `http://ece.ut.ac.ir/dbrg/seminars/AdvancedDB/2007/Farzinvash%20leila%20-%20Dehghani%20nazanin/Report1/Refrences/9.pdf`.

[103] A. Simmonds, P. Sandilands, and L. Ekert, “An Ontology for Network Security Attacks,” *Applied Computing*, vol. 3285 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2004, pp. 317–323.

[104] D. I. S. C. H. G. D. S. P. T. C. V. Slawomir Gorniak, Rodica Tirtea, “Enabling and Managing End-to-End Resilience,” Jan.24 2011.

[105] "Protecting Your Website With Always On SSL,", May 1 2012, http://otalliance.org/resources/AOSSL/OTA_Always-On-SSL-White-Paper.pdf.

[106] N. Stakhanova, S. Basu, and J. Wong, "A Taxonomy of Intrusion Response Systems," *Int. J. Inf. Comput. Secur.*, vol. 1, no. 1/2, Jan. 2007, pp. 169–184.

[107] N. Stakhanova, C. Strasburg, S. Basu, and J. S. Wong, "Towards Cost-Sensitive Assessment of Intrusion Response Selection," *Journal of Computer Security*, vol. 20, no. 2-3, 2012, pp. 169–198.

[108] H. Stark, "Mossad's Miracle Weapon: Stuxnet Virus Opens New Era of Cyber War," 2011, http://www.spiegel.de/international/world/mossad-s-miracle-weapon-stuxnet-virus-opens-new-era-of-cyber-war-a-778912-3.html.

[109] A. Stavrou, D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: An Overlay-based System For Protecting Web Servers From Denial of Service Attacks," *Elsevier Journal of Computer Networks, special issue on Web and Network Security*, vol. 48, 2005, p. 2005.

[110] R. Sterritt, "Towards Autonomic Computing: Effective Event Management," *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE*, 2002, pp. 40–47.

[111] C. Stoll, "The Cuckoo's Egg," http://www.jimloy.com/books/cuckoo.htm.

[112] K. A. Stouffer, J. A. Falco, and K. A. Scarfone, *SP 800-82. Guide to Industrial Control Systems (ICS) Security*, Tech. Rep., Gaithersburg, MD, United States, 2013.

[113] C. Strasburg, N. Stakhanova, S. Basu, and J. Wong, "A Framework for Cost Sensitive Assessment of Intrusion Response Selection," *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, 2009, vol. 1, pp. 355–360.

[114] C. Strasburg, N. Stakhanova, S. Basu, and J. S. Wong, "Intrusion Response Cost Assessment Methodology," Sydney, Australia, 2009, ASIACCS '09, pp. 388–391.

[115] B. Sullivan, "160 Million Credit Cards Later, 'Cutting Edge' Hacking Ring Cracked," 2008.

[116] V. Surwase and S. Durugkar, "Dynamic Security Skins-Mutual Authentication," *BIOINFO Security Informatics*, vol. 1, 2011.

[117] L. Y. L. P. TAN Jianlong, LIU Xia, "Speeding up Pattern Matching by Optimal Partial String Extraction," .

[118] A. S. Tanenbaum and M. v. Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[119] J. Therdphapiyanak and K. Piromsopa, "Applying Hadoop for Log Analysis Toward Distributed IDS," *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, New York, NY, USA, 2013, ICUIMC '13, pp. 3:1–3:6, ACM.

[120] M. Tracy, W. Jansen, K. A. Scarfone, and T. Winograd, *SP 800-44 Version 2. Guidelines on Securing Public Web Servers*, Tech. Rep., Gaithersburg, MD, United States, 2007.

[121] I. B. Van Heerden, RP and I. Burke, "Classifying Network Attack Scenarios Using An Ontology," University of Washington, Seattle, 22-23 March 2012, 7th International Conference on Information Warfare and Security, pp. 311–324.

[122] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. Markatos, and S. Ioannidis, "Gnort: High Performance Network Intrusion Detection Using Graphics Processors," *Recent Advances in Intrusion Detection*, R. Lippmann, E. Kirda, and A. Trachtenberg, eds., vol. 5230 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2008, pp. 116–134.

[123] A. Wailly, M. Lacoste, and H. Debar, "VESPA: Multi-Layered Self-Protection for Cloud Resources," San Jose, California, USA, 2012, ICAC '12.

[124] D. Wang, T. Li, S. Liu, J. Zhang, and C. Liu, "Dynamical Network Forensics Based on Immune Agent," *Natural Computation, 2007. ICNC 2007. Third International Conference on*, Aug 2007, vol. 3, pp. 651–656.

[125] Y. Wang, Y. Xiang, W. Zhou, and S. Yu, "Generating Regular Expression Signatures for Network Traffic Classification in Trusted Network Management," *J. Netw. Comput. Appl.*, vol. 35, no. 3, May 2012, pp. 992–1000.

[126] "IBM-Open Source Application Server-WebSphere Application Server Community Edition,", 2013, `http://www-01.ibm.com/software/webservers/appserv/community/`.

[127] H. Wilson, "Every Minute of Every Day, a Bank is under Cyber Attack," Oct. 2013, `http://www.telegraph.co.uk/finance/newsbysector/banksandfinance/10359563/Every-minute-of-every-day-a-bank-is-under-cyber-attack.html`.

[128] Y.-S. Wu, B. Foo, Y.-C. Mao, S. Bagchi, and E. H. Spafford, "Automated Adaptive Intrusion Containment in Systems of Interacting Services," *Comput. Netw.*, vol. 51, no. 5, Apr. 2007, pp. 1334–1360.

[129] Y. X. Xiaowei Li, “A Survey on Server-Side Approaches to Securing Web Applications,” *ACM Computing Surveys*, vol. 46(4), 2014.

[130] M. K. L. T. S. S. Yang, Y., B. Pranggono, and H. Wang, “Intrusion Detection System for IEC 60870-5-104 Based SCADA Networks,” *Power and Energy Society General Meeting (PES), 2013 IEEE*, July 2013, pp. 1–5.

[131] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, and H. Wang, “Rule-based intrusion detection system for SCADA networks,” *Renewable Power Generation Conference (RPG 2013), 2nd IET*, Sept 2013, pp. 1–4.

[132] Z. Yu, J. J. P. Tsai, and T. Weigert, “An Adaptive Automatically Tuning Intrusion Detection System,” *ACM Trans. Auton. Adapt. Syst.*, vol. 3, no. 3, Aug. 2008, pp. 10:1–10:25.

[133] M. Zaikin, *IBM WebSphere Application Server Network Deployment V7.0 Core Administration Guide,Chapter 1. Architecture*, July, 2011, `http://java.boot.by/ibm-377/ch01.html`.

[134] Y. Zhai, “Time Series Forecasting Competition Among Three Sophisticated Paradigms,” 2009.

[135] N. Zhong, J. Liu, and Y. Yao, “In Search of the Wisdom Web,” *Computer*, vol. 35, no. 11, 2002, pp. 27–31.