Self-Protect Computing Systems Towards Model-Based Validated Autonomic Approach

Karthi M,Sakthipriya K

Abstract— This project an autonomic model-based cyber security management approach for the Internet of Things (IoT) ecosystems. The approach aims at realize a self-protecting system, which has the ability to together estimate, detect, and respond to cyber attacks at an early stage. Proposed model-based techniques including: 1) data analysis to recognize and categorize attacks; 2) real-time evaluation and baseline security controls to predict and eliminate potential cyber attacks; and 3) a multi criteria optimization method to select the most advantageous active response for deploying countermeasures while maintaining system functions. The model framework has been developed for configuring various platforms with a master controller virtual machine. Investigational results demonstrated the efficiency of this proposed approach in protecting a Web-based application against known and unknown attacks with little or no human interference. Here we reduce the recovery time from anonymous attacks by improving attack detection accuracy and learning1 procedures to further improve the false alarm rates.

Index Terms— Autonomic computing, cyber security, self-protection, Web services.

1 Introduction

CYBER attackers typically exploit vulnerabilities in computer hardware, software, and communications protocols to target the Internet of Things (IoT) ecosystems within enterprise, industrial, and government systems. As a result, the confidentiality, integrity, and availability of these systems are undermined. Real-world cyber attack, as we known, can have catastrophic impacts (e.g., financial loss, property damages,etc.). In 2010, the Stuxnet computer worm attacked Iran's nuclear facilities resulting in physical damage to the centrifuges which severely impacted uranium enrichment efforts [1].During the 2013 holiday shopping season, cyber criminals attacked database servers of Target Corp. and compromised 70 million credit and debit cards [2].More recently, in 2014, cyber attacks also paralyzed the federal court system impeding public users from filing or reviewing records [3].

To safeguard and secure cyberspace, methodologies such as anti viruses, firewalls, intrusion detection systems (IDSs),and intrusion response systems (IRSs) are provided to detect, prevent, and react to cyber attacks. However, 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 antivirus software is helpless in combating new malicious codes that exploit unknown vulnerabilities [4]. Traditional firewalls provide little protection from authorized (and unauthorized) internal adversaries.

Autonomic computing technology, introduced in 2001 [6], allows computing systems to proactively self-configure, self optimize, self-heal, and self-protect under various operating conditions. In this paradigm, security mechanisms such as system monitoring, detection, and response mechanisms are tightly coupled to realize the self-protection function (SPF). Recent research demonstrates that SPF enables computing systems to anticipate upcoming cyber assaults relying on early reports generated by sensors [6].

To validate the ACSM approach, we built a multitier web application test bed. We also designed a master controller virtual machine (MC-VM)2 and inserted autonomic security agents into the victim host. We launched four well-known and destructive web application cyber attacks. The experimental results show that our approach autonomously protects the victim host from known and unknown cyber attacks efficiently and rapidly thereby reduces the need for human intervention and costs. All modules, tools, software, and techniques used in ACSM approach were tightly integrated using the Python programming language [10] along with a graphic user interface.

Contributions of this paper include the following.
1) The ACSM approach developed in this research can switch between fully autonomous and semiautonomous modes. The ACSM approach can be tailored to various computing environment contexts; the autonomic property provides precision control to enable the highest possible level of service performance and integrity. An ACSM system can also be used as a semiautonomous system. In this case, a system administrator can select and initiate appropriate responses to mitigate ambiguous cyber assault signatures in concert with any MAC determinations.
2) The ACSM framework is simple to configure and deploy. The main modules of the ACSM approach are installed on an MC-VM. As a result, the autonomic computing system can be easily realized by adding the MC-VM to most of the popular IoT-based platforms.
3) The adoption of the ACSM framework supports reliable, sustainable, and resilient self-protection performance.

Most significant autonomic 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 function of the MC-VM. On the contrary, the MC-VM regulates the victim system behavior to maintain its normal service delivery operations even in the presence of compromising failures irrespective of whether instigated by external or internal attackers.

2 Control-Based Security Management

In this section, we introduced and implemented the ACSM framework aimed at protecting networked computing systems

from cyber adversaries. The framework, outlined in Fig. 1, contains data sensors, a data processing module, forecasting modules, a system module, IDSs, a learning module, and a redirection module, as well as a MAC installed on a router, the MC-VM, and the main host. In this architecture, the router forward client requests to the internal network. These incoming client requests may include malicious messages that disrupt normal services provided by the multitier 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. 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 IDS) are then provided to the MAC. The MAC assesses recommended responses using fuzzy logic and the Preference Ranking Organization METHOD for Enrichment Evaluation (PROMETHEE II) decision-making techniques.

2.1 *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 (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 [11].The ARIMA method predicts a value in a response time series as a linear combination of previous observations and error terms [12]. 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 hated 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

$$\left(1-\sum_{i=1}^{p} pi\right) L\, i(1-L)^d y_t$$

$$=\mu\left(1-\sum_{i=1}^{p} p_i\right)+\left(1-\sum_{j=1}^{q} v_{jL^j}\right)\epsilon_t$$

where L is the lag operator and defined as $L1y_t = y_{t-1}$; ρ_i and υ_j are weighting coefficients; μ is the intercept term, which is close to the mean value of the time series; $\_t$ 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 [13] is used to select the optimal ARIMA model variant, thereby adapting to the increasing amount of historical data (i.e., no 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 [14]

$$\text{ACI}=n\ln\left(\frac{RSS}{n}\right)+2k$$

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 residual sum of squares (RSS), is the estimated residual of the fitted model [13]. 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 I. The outputs from the forecasting modules and relevant analysis are described in Section III through realistic attack simulations that attempt to compromise the main host web application.

2.2 System Model

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

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

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 $\hat{x}(k)$ 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. The normal region can be determined based on predefined thresholds of the environment, system, and security parameters, which are determined by system administrators or security experts during the risk assessment process. Predicted parameter values that are outside the normal region indicate likely cyber attacks targeting the host.

2.3 *Intrusion Detection System (IDS)*

The 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 (the forecaster). The PIDS on the MC-VM employs anomaly and misuse detection methods to detect and classify all types of known attacks. Offline 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 I) are selected for inclusion in historical training datasets, the detection time is significantly reduced compared with signature-based IDSs (SIDSs) which fully adopt all 41 features of the KDD99 dataset [15], in that 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 over-

head of detection latency.

2.4 *Multicriteria Analysis Controller*

The MAC evaluates candidate responses as an intrusion respThe 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 web server system performance are abnormal or 2) real-time observations are identified as attacks. Fuzzy-logic and Preference Ranking Organization METHOD for Enrichment Evaluations II (PROMETHEE II) are the two efficientonse system (IRS) and initiates the most appropriate responses necessary to recover the system performance under two conditions: 1) if estimations of the web server system performance are abnormal or 2) real-time observations are identified as attacks. Fuzzy-logic and Preference Ranking Organization METHOD for Enrichment Evaluations II (PROMETHEE II) are the 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 in 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 Section III-F.

Fuzzy numbers from 0 to 1 are used to normalize criteria, where 0 represents effective responses and 1 represents candidate responses that 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

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: It is an intrusion protection system [19] 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 predefined rules. However, a large number of rules induce computational overheads by slowing down the execution speed of the recovery action.

• Packet filtering: This response eliminates all packets, both legitimate and illegitimate, sending to the compromised ports on the main host. It ultimately protects the host from flooding-based Denial of Service (DoS) attacks and rapidly mitigates network congestion. However, elimination of legitimate packets impedes appropriate transactions between legitimate users and web servers.

• Authentication process: 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: 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: 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: 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.

3 Experiments

In this section, we demonstrate the developed ACSM framework by simulating five realistic cyber attacks, which disrupt the availability of web services, threaten data confidentiality, and damage file system integrity. Experimental results show that the application of the ACSM approach efficiently realizes a self-protecting system that closes the window on vulnerabilities, enhances host/server security, and maintains the underlying services without (necessarily) requiring human intervention. Manual interventions are required only in exceptional cases.

3.1 *Web Application Protection Test bed*

The experiments of test bed configuration settings are shown in Table II. The test bed includes two web servers installed on the main host; they are the IBM Web sphere Application Server [13] and the Apache HTTP Server Version 2.2 [20].

1) Baseline Measures: Normally, clients send 300–2000 random requests every 30 s to either the IBM Web sphere Application Server or the Apache HTTP Server by Httperf, a tool for measuring web server performance [14]. The IBM Web sphere Application Server receives legitimate

client requests from Daytrader [20] 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.

2) Performance-Based Intrusion Detection System:

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 ACSM approach, the total amount of historical datasets are around 14000. Two thousand 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.

3.2 *UNKNOWN TCP SYN ATTACK*

TCP SYN attacks are one type of DoS attacks. Attackers create tremendous number of half-opened TCP connections that led to exhaustion of server memory resources. Estimations of four features and their 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 DoS attacks. 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 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. This response was assessed and selected by the MAC adopting fuzzy-logic and PROMETHEE II methods. Once Packet Filtering had been implemented after 95 s, the performance of the main host is recovered back to normal. System and network utilization of the MC-VM. After 50 samples, both *byte_recv* and *packet_recv* increased sharply, and values of the *memory* feature dropped sharply. *CPU_sys* increased from 45% to 60%, since a large number of malicious packets were redirected to the MC-VM. Consequently, the CPU utilization subfigure, *CPU_idle*, was less than 20%. From 95 samples onward, the adversaries stopped bombarding DSs with TCP SYN attacks as shown, because the performance of theMC-VM returned to normal.

3.2 *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. How, due to the impact of CPU exhaustion attacks, the estimations of *CPU_idle* and *memory* of the main host dropped below their thresholds after 130 s. Observations of these two features, as shown in Fig. 6, were exhausted at 120 s. Because of forecasting delays in the order of 10 s, adversaries were able to bypass the prevention system. Subsequently attacks were detected by the IDS installed on the main host. 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

4 REVIEW OF RELETED RESEARCH

4.1 *SELF-PROTECTION SYSTEM*

One special property of IoT devices is that these smart devices are able to execute different actions according to their surroundings and their tasks [16]. Therefore, self-protecting systems, which know their environment and proactively protect themselves without human intervention, can be adopted to enhance information security of IoT ecosystems. Before a self-protecting system can be realized, the system boundary (e.g., routers, firewalls, web servers, application servers, and databases) must first be determined so that baseline security controls can be tailored using risk assessments. On this basis, abnormal system behavior can then be anticipated. Consequently, we can compare each new packet to the anticipated behavior, followed by the execution of appropriate controls. One example of the realization of self-protection in a distributed system is introduced by Claudel *et al*. [17]. The sensors and actuators of the proposed system are utilized to identify abnormal (malicious) requests and isolate compromised nodes automatically. Dean *et al*. [18] designed and implemented an unsupervised behavior learning system for virtualized cloud computing infrastructures. The self-organizing map learning technique captures system performance and anticipates unknown anomalies.

5 IOT IMPLEMENTATION

IoT is a global network infrastructure, which has the ability to connect, communicate with, and remotely manage a large number of sensors and automated devices via the Internet [16], [17].Only when strong security and privacy are in place will IoT be most feasible and practical [16]. One of the future trends of IoT is toward autonomic resources since it becomes impossible to manage the increasing complexity by system administrators. The ACSM approach presented here easily realizes SPF by adapting system models and integrating intrusion estimation, detection, and protection techniques with little or no human intervention. Even though we focus on the analysis of a web application, the ACSM approach prom-

ises to strengthen information security, personal privacy, and data protection of other potential IoT-based ecosystems.

One example of such an ecosystem where the ACSM framework can be adopted is to protect industrial control systems (ICSs). The MC-VM developed in this research contains essential autonomic components so that system administrators only need to add the MC-VM to their ICS network, collect normal ICS operational performance data, and construct a "normal region" using the datasets monitored by existing or installed sensors. The forecaster module will autonomously run to determine the control system security. The IDSs detect unknown attacks that bypass the protection mechanisms implemented by the forecaster. An IoT-based protection mechanism must be first determined and configured by the system administrator so that a set of recommended protection mechanisms can be evaluated by the MAC. Optimal protection mechanisms can subsequently be deployed to protect IoT devices from known and unknown attacks. As a result, the application of ACSM to IoT ecosystems will proactively protect data monitored by sensors, improve data privacy during the communication between IoT appliances, and strengthen the IoT information security posture in the general case. It may be necessary to tailor "lightweight" ACSM modules to fully adapt to all pervasive IoT environments.

6 Conclusion and Future Work

This paper introduces a model-based cyber security management approach that autonomously protects IoT-based systems from a wide range of cyber attacks with minimal impacts on their performance and function. Besides estimating, detecting, and protecting against known attacks, the proposed approach can also learn signatures of unknown attacks in real time. Unknown attacks are identified for future protection by updating detection algorithms with the underlying signatures. In our framework design, the main protection modules are installed on the MC-VM, which ensures that the self-protection mechanism continues to work even if the underlying system is compromised. A multitier web application case study is presented. Our self-protecting system successfully secured the web application from various known and unknown attacks. In the future we plan to extend our self-protection framework with dynamic reconfigured intrusion responses and digital forensic analysis in a more light-weight manner to defend IoT ecosystems against a variety of real-life intrusions.

Acknowledgment

The authors would like to thank Dr. F. Sheldon for his assistance and comments that greatly improved this paper's manuscript. The statements made herein are solely the responsibility of the authors

References

[1] C. A. Theohary, P. K. Kerr, and J. Rollins, *The Stuxnet Computer Worm:*Harbinger of an Emerging Warfare Capability. Washington, DC, USA:Congressional Res. Service, 2010.

[2] J. McElhatton. (2014, Jan.). Cyber Attack Hits Federal Judiciary *Websites, Filling Systems* [Online]. Available: http://www.washington times.com/news/2014/jan/24/federal-judiciary-websites-filing-ystemsattacked/

[3] R. Moskovitch, N. Nissim, and Y. Elovici, "Privacy, security, and trust in KDD," in Malicious Code Detection Using Active Learning. Berlin,Germany: Springer-Verlag, 2009, pp. 74–91.

[4]] N. Stakhanova, S. Basu, and J.Wong, "A taxonomy of intrusion response systems," *Int. J. Inf. Comput. Secur.*, vol. 1, pp. 169–184, Jan. 2007.

[5] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41–50, Jan. 2003.

[6] S. Hariri, G. Qu, R. Modukuri, H. Chen, and M. Yousif, "Qualityof-protection (QoP)—An online monitoring and self-protection mechanism,"*IEEE J. Sel. Areas Commun.*, vol. 23, no. 10, pp. S

[7] A. Wailly, M. Lacoste, and H. Debar, "Vespa: Multi-layered selfprotection for cloud resources," in Proc. Int. Conf. Auton. Comput. *(ICAC'12)*, San Jose, CA, USA, 2012, pp. 155–160.

[8] E. Gelenbe and G. Loukas, "A self-aware approach to denial of service defence," *Comput. Netw.*, vol. 51, pp. 1299–1314, Apr. 2007. (2014). *Python Programming Language* [Online]. Available: http://www.python.org/.

[9] SAS Inst. Inc. (2013). *The Arima Procedure* [Online]. Available:http://www.okstate.edu/sas/v8/saspdf/ets/chap7.pdf

[10] Y. K. Rob and J. Hyndman, "Automatic time series forecasting: The forecast package for R," *J. Stat. Softw.*, vol. 27, no. 3, pp. 1–22, 2008.

[11] H. Bozdogan, "Model selection and akaike's information criterion (AIC):The general theory and its analytical extensions," *Psychometrika*, vol. 52,no. 3, pp. 345–370, 1987.

[12] Third International Knowledge Discovery and Data Mining Tools Competition. (1999). *KDD Cup 1999 Data* [Online]. Avaialble: http://kdd.ics.uci.edu/ databases/kddcup99/kddcup99.html

[13] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir, "A k-means and Naive Bayes learning approach for better intrusion detection," *Inform.Technol. J.*, vol. 10, pp. 648–655, 2010.

[14] E. S. Pilli, R. Joshi, and R. Niyogi, "Network forensic frameworks:Survey and research challenges," *Digit. Invest.*, vol. 7, no. 1, pp. 14–27,2010.

[15] TheWireshark Team. (2013). *Wireshark* [Online]. Available: http://www.wireshark.org/about.html

[16] W. Metcalf and V. Julien. (2013). *Snort_Inline* [Online]. Available: http://snort-inline.sourceforge.net/oldhome.html

[17] Apache Softw. Foundation. (2013). *Apache Http Server Version 2.2 Documentation-Apache Http Server* [Online]. Available: http://httpd.apache.org/docs/2.2/

[18]] S. Raju and N. Kumar, Multicriterion Analysis in Engineering and *Management*. New Delhi, India: PHI Learning, 2010.

[19] M. Behzadian, R. Kazemzadeh, A. Albadvi, and M. Aghdasi,"Promethee: A comprehensive literature review on methodologies and applications," *Eur. J. Oper. Res.*, vol. 200, no. 1, pp. 198–215, 2010.

[20] IBM. (2013). IBM-Open Source Application Server-Websphere Application Server Community Edition [online]. Available: http://www-01.ibm.com/software/webservers/appserv/community/