Annual ADFSL Conference on Digital Forensics, Security and Law

2015 Proceedings

May 20th, 4:20 PM

Continuous Monitoring System Based on Systems' Environment

Eli Weintraub
Tel Aviv Afeka College of Engineering, Israel, Head of Information systems specialization

Yuval Cohen
Tel Aviv Afeka College of Engineering, Israel, Head of Information systems specialization

Scholarly Commons Citation

Weintraub, Eli and Cohen, Yuval, "Continuous Monitoring System Based on Systems' Environment" (2015). *Annual ADFSL Conference on Digital Forensics, Security and Law*. 10.
https://commons.erau.edu/adfsl/2015/wednesday/10

CONTINUOUS MONITORING SYSTEM BASED ON SYSTEMS' ENVIRONMENT

Eli Weintraub
Tel Aviv Afeka College of Engineering
Israel
Head of Information systems specialization

Yuval Cohen
Tel Aviv Afeka College of Engineering
Israel
Head of Production Management specialization

ABSTRACT

We present a new framework (and its mechanisms) of a Continuous Monitoring System (CMS) having new improved capabilities, and discuss its requirements and implications. The CMS is based on the real-time actual configuration of the system and the environment rather than a theoretic or assumed configuration. Moreover, the CMS predicts organizational damages taking into account chains of impacts among systems' components generated by messaging among software components. In addition, the CMS takes into account all organizational effects of an attack. Its risk measurement takes into account the consequences of a threat, as defines in risk analysis standards. Loss prediction is based on a neural network algorithm with learning and improving capabilities, rather than a fixed algorithm which typically lacks the necessary environmental dynamic updates. Framework presentation includes systems design, neural network architecture design, and an example of the detailed network architecture.

Keywords: Continuous Monitoring, Computer security, Attack graph, Software vulnerability, Risk management, Impact propagation, Cyber attack, Configuration management

1. INTRODUCTION

Personal and organizational computing systems are sometimes subject to cyber-attacks which may cause damage to organizational data, software and computers (Mell et al. 2007). This paper focuses on threats generated by hostile attackers. Vulnerabilities are weaknesses or exposures stemming from bugs that are potential causes to security failures: loss of confidentiality, integrity or availability. An attack is performed by exploiting software vulnerabilities in the target computing system. Exploits are planned to attack certain components having specific vulnerabilities. Langer (2011) states that Stuxnet warm included a process of checking hardware models and configuration details, and also downloads program code from the controller to check if it was the "right" program before launching an attack. This leads to planning defense systems that are sensitive to changes in their environment. Users' computers might be damaged by exploited vulnerabilities. Organizations make decisions on actions they have to take, in order to limit their risks according to the amount of potential damage and vulnerability characteristics (Tom, 2008).

Several software products are usually used for defending computers from cyber attacks. Antivirus software, antispyware and firewalls are examples to some of these tools. Several tools are based on periodic assessment of the target computer by comparing computers' software to the known published vulnerabilities. Antivirus engines store features of known malware and hash signatures, using classification algorithms to identify hostile software. Signature scanning technique is the most widely used technology in anti-virus programs (Symantec, 1997). Those tools are naturally effective only against known threats and not against new unpublished threats. Heuristic Antivirus scanners detect viruses by analyzing the program's structure or its behavior instead of looking for signatures**.** Heuristic scanners are able to identify new unpublished malware. Intrusion Detection System (IDS) monitor the events occurring in a computer or network, searching for violations or threats to computer security policies and security practices. Static and dynamic code analysis techniques are aimed to identify malicious activities by analyzing attempts to execute code or identifying unusual behavior (Scarfone and Mell, 2007). Contrary to the popular techniques such as antivirus, antispyware and firewall, our model analyzes vulnerabilities at the time before fixes are publicly distributed. Moreover, our model uses a prediction algorithm which uses historical data of exploits, together with computer's configuration, to predict losses of the new vulnerabilities.

Information Security Continuous Monitoring system (ISCM) is defined by NIST as: Maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions (Dempsey et al., 2011). We use the acronym CMS since we do not limit our model to software. CMS's monitor computer systems in a near real time process aimed at detecting vulnerabilities and cyberspace security risks, and alarming the appropriate functions within the organization. Contemporary systems use vulnerabilities databases (which are continually updated as new vulnerabilities are detected) and a scoring algorithm which predicts potential business losses.

Computers are at risk to known threats until the moment a patch is programmed for the vulnerable specific software, an activity that may last weeks or months. Even after a patch is prepared by the software vendor a computer might still be at risk until the moment the new patch is loaded to the vulnerable system. Loading patches to computer systems is usually performed as a periodic process, not continuously. The reason for this is avoiding too many interrupts required for uploading and activating the patch on the production environment. In today's environment of zero-day exploits, conventional systems updating for security vulnerabilities has become a cumbersome process. There is an urgent need for a solution that can rapidly assess system vulnerabilities and immediately fix them (Nũez, 2008). Although zero-day vulnerabilities are kept secret by hackers for exploits programming, after a 90-days period vendors like Google use to automatically disclose the vulnerability to the public even if no fix was written. Our system deals with risks at the time the vulnerability is published but not yet fixed in the operational organizational environment.

Operating techniques for monitoring, detecting and alerting of security threats on a regular basis are defined as Security Continuous Monitoring (SCM) systems. After identifying these risks, tools evaluate the potential impacts on the organization, sometimes suggesting risk mitigation activities to the organization to support organizational risk management decisions (Dempsey, 2011). SCM's are aimed at closing the gap between the zero-day of identifying the vulnerability, until the moment the computer is loaded by the corresponding patch fixing the vulnerability. The time gap may be considerably long.

In this paper we describe a mechanism of a new SCM system framework that will produce better detection and prevention than existing SCM systems. Our framework is based on four main elements: (1) Knowledge concerning the specific computers' configuration of the target system and interrelationships among systems' components. (2) A prediction algorithm which runs continuously and predicts the potential losses. (3) Risk assessment is based on vulnerability consequences. (4) A learning algorithm which continuously improves the predicted losses.

The rest of the paper is organized as follows: In section 2 we describe current known solutions. In section 3 we present the proposed framework including systems' architecture. In section 4 we describe the scoring algorithm which predicts vulnerability losses. We present a neural network model for loss prediction and learning.

In section 5 we conclude and describe future research directions.

2. EXISTING SOLUTIONS

SCM systems are using external vulnerabilities databases for evaluation of the target computers' risk. There are several owners of vulnerability databases (Dempsey et al., 2011): The Sans Internet Storm Center services and The National Vulnerability Database (NVD). Vulnerability Identification Systems (VIS) aimed to identify vulnerabilities according to three categories: code, design, or architecture. Examples for VIS are: the Common Vulnerabilities and Exposures (CVE), and The Common Weakness Enumeration (CWE). In this work we shall use NVD vulnerabilities database as an example.

Risk evaluation uses scoring systems which enable parameters estimation for assessing the impacts of vulnerabilities on the organization. The Common Vulnerability Scoring System (CVSS) is a framework that enables user organizations receive IT vulnerabilities characteristics (Mell et al., 2007).
CVSS uses three groups of parameters to score potential risks: Basic parameters, Temporal parameters and Environmental parameters. Each group is represented by a score compound parameters ordered as a vector, used to compute the score. Basic parameters represent the intrinsic specifications of the vulnerability. Temporal parameters represent the specifications of a vulnerability that might change over time due to technical changes. Environmental parameters represent the specifications of vulnerabilities derived from the local IT specific environment used by users' organization. CVSS enables omitting the environmental metrics from score calculations, those are cases that users' environment has no effect on the score. CVSS is a common framework for characterizing vulnerabilities and predicting risks, used by IT managers, risk managers, researchers and IT vendors, for several aspects of risk management.

CVSS is an open framework which enables managers to deal with organizations' risks and make decisions based on facts rather than evaluations. User organizations adopting CVSS framework may gain the following benefits:

- A standard scale for scoring vulnerabilities and risks. The scale enables organizations normalize vulnerabilities according to specific IT platforms. The computed scores enable users to get rational decisions in correlation to vulnerability risks.
- Open framework: user organization can see the characteristics of vulnerability and the logical process of scores evaluation.
- Prioritized risks: organizations using the environmental parameters may benefit by considering changes in its IT environment according to predicted risk scores.

There are few other vulnerability scoring systems besides CVSS differing by what they measure. CERT/CC puts an emphasis on Internet infrastructure risks. SANS vulnerability system considers users' IT configuration and usage of default parameter definitions. Microsoft's scoring system emphasizes attack vectors and impacts of the vulnerability.
Generally, Basic and Temporal parameters are specified and published by products' vendors who have the best knowledge of their product. Environmental parameters are specified by the users who have the best knowledge of their environments and vulnerabilities' business impacts.

This paper focuses mainly on environmental metrics.

The organizational damage caused by vulnerability is influenced by the specific IT environment which is exploited. CVSS environmental parameters specify the characteristics of a vulnerability that is associated with user's IT components compounding the environment. Environmental parameters are of three groups:

I. Collateral Damage Potential (CDP):
   A group of parameters which measure the economic potential loss caused by a vulnerability.
II. Target Distribution (TD):
   Parameters indicating the percentage of vulnerable components in user environment. A large proportion indicates more impacts on organizational potential damages.

III. Security Requirements (CR, IR, AR):
Security requirements are parameters which indicate user's sensitivity to security risks.
This group of parameters is subdivided to certain parameters indicating the Confidentiality (CR), Integrity (IR), and Availability (AR) of the vulnerable component. High security requirements might cause higher security damages, thus more economic losses.

Categorization of IT components according to security requirement measures should be performed by users encompassing all assets. Doing so raises the possibility to predict the organizational losses. Federal Information Processing Standards (FIPS) requirements demands implementation of a categorization system (Dempsey et al., 2011), but does not require using any particular scale, thus risk comparisons among users systems is difficult.

3. THE PROPOSED FRAMEWORK

Federal organizations are moving from periodic to continuous monitoring implementing SCM's which will improve national cyber security posture (Hardy, 2012). The proposed framework includes four capabilities which are not found in current models:

- Real time environmental metrics.

Metric evaluations are based on the components of the system as updated in the systems' CMDB (Keller and Subramanianm, 2009). There are several commercial products for asset inventory management such as IBM Tivoli or Microsoft System center. This capability enables basing predictions on real IT environment rather than on user's evaluations. According to Grimalia et al. (2009) it is impossible for organizations to make precise estimates of the economic losses caused by an attack without having full knowledge of users' IT environment. Kotenko and Chechulin (2012) state that network configuration should be monitored continually and available vulnerabilities must be analyzed in order to provide the necessary security level.

The proposed CMS examines a database of published asset vulnerabilities, compares in real time computers' assets for existing exposures, and calculates computers' potential losses. Loss evaluation is performed by considering vulnerabilities even before patches are prepared and loaded on the computers' system.

- Components interdependencies.

Current systems focus on the IT infrastructure but not on the interdependencies among components. Several researchers stress the need to deal with interdependencies (Albanese et al., 2013; Jakobson, 2011). Jajodia S, et al. (2011) presents a model that maps possible multi-step environmental vulnerabilities, enabling organizational damage estimations. Kotenko and Chechulin (2012) present a system based on attack modeling using attack graphs, evaluating security risk based on attack model. Wang et al. (2006) propose an automated process aimed at hardening a network against multi-step intrusions.
Our framework deals with loss prediction by looking for past attacks on systems' components by learning from their past organizational impacts. The proposed algorithm takes into account component dependencies, predicting all potential direct and indirect impacts on the organization stemming from the specific vulnerable component. Loss prediction is implemented by a neural network which represents IT components and interdependencies between components such as reading and writing from neighboring components. The process of predicting loss is based on propagation of signals among components, starting from the vulnerable component, ending at the organizational losses as stated by the user. Signals between components represent the varying kinds of dependencies.

- Risk assessment based on consequences.

Risk analysis theory defines risk as a triple that specifies the scenario of an event, the likelihood that the occurring event and event consequences appearing regularly as threat x vulnerability x consequences. According to Collier et al. (2014) CVSS fails to connect risk assessment to risk management. According to CVSS, risk damage potential values are estimated by organizations (Mell et al., 2007). According to the proposed framework, potential loss prediction is based on the actual losses of similar past attacks on the specific vulnerable component, performed through the similar attack vector. In cases when there has not been in the past a similar attack, prediction will be based on past losses stemming from past attacks on the specific component concerning all attack vectors.

- A Learning algorithm.

Hardy (2012) states that predictive analysis should be used for threat modeling. Threat projection algorithms are also presented by Holsopple and Yang (2008) to estimate plausible futures. We use predictive analysis for loss prediction, based on historical data of losses caused by past attacks on vulnerable components. The predictive analysis uses a learning algorithm since the organization learns how to deal with the vulnerable component, improves its software, thus limiting or preventing damages. According to the proposed framework loss prediction is based on environmental parameters, and actual losses of past events. Both environmental parameters and losses are related to changes: environmental characteristics are subject to changes which occur all the time in operational systems and actual losses of past events which are continuously updated according to users' findings about incidents' impacts. Losses caused by past attacks may be noticed long after the time of the attack. Such late losses should update the predicted loss calculated by the algorithm.
We describe the proposed framework architecture (Figure 1) and its main components.

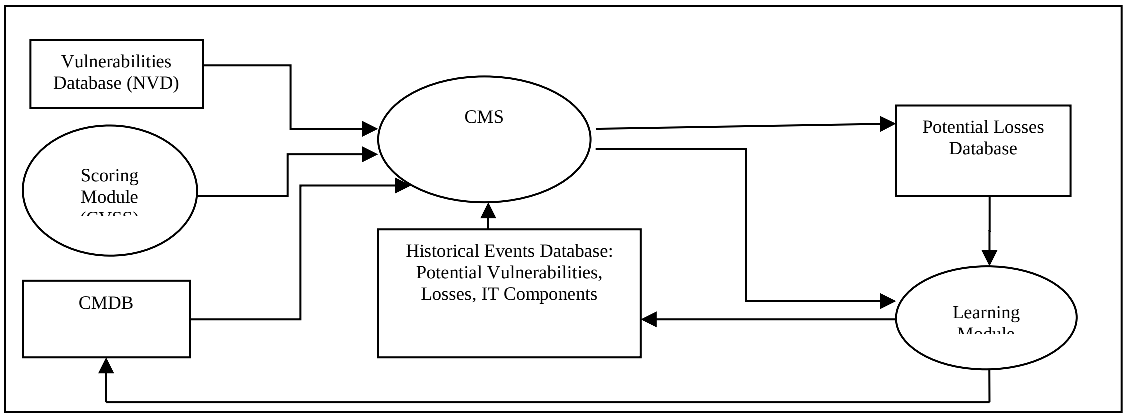

Vulnerabilities Database (NVD)
CMS
Potential Losses Database
Scoring Module (CVSS)
Historical Events Database: Potential Vulnerabilities, Losses, IT Components
CMDB
Learning Module

Figure 1 Continuous Monitoring System architecture

Vulnerabilities Database includes all known vulnerabilities and their specification as published by Database owners or government agencies. As an example for vulnerability specifications NVD defines: category, vendor name, product name, published start and end dates, vulnerability update dates, severity, access vector, access complexity and security requirements of a patch (Hardy, 2012).

Scoring module (CVSS) is an algorithm which computes potential losses according to the parameters of three groups. As stated above there are also other known algorithms, some of them for public use other commercial.

CMDB is a database which includes all hardware and software components of the target system and all components' characteristics. Components are dealt in a resolution of a hardware machine. Software is dealt in the resolution of programs or physical exe files or DLL's. Data is handled in the resolution of database or table, not data items. Input/output are dealt by screen-name or output message. The target system might be one computer or a group of organizations' computers. CMDB includes all components in the computers' environment, components which interface with the target system directly or indirectly up to external and end-users' interfaces. CMDB includes also the security requirements (CR, IR, AR) of each component. Security requirements are specified by systems' owners according to business potential losses. CMDB includes also all interfaces among components. For each interface are indicated the direction of data transfer between the components and the probability of occurrence of that connection according to systems' operational history.

Historical events database includes all cyber attacks on the system and their details. For each event indicated the vulnerability which was used to exploit and all computer components involved in the incident. Also are indicated the economic loss caused to the organization by the attack as evaluated by the organizational users or risk management.

Potential Losses Database includes the predicted losses computed by the system. Systems' owner is informed about the potential predicted loss of all components at risks and makes his decisions concerning each component. The owner might disable a component or a computer when loss potential is high. In cases a patch is not yet developed, the owner might continue using the risky component or monitoring the component closely with higher awareness to possible exploits. In cases a patch was developed but not yet loaded on operational systems the owner might decide either remediate and deploy the patch, defer deployment to appropriate times considering organizational constraints, or reject deployment in cases the potential loss is limited.

The system runs continuously based on the neural network predicting losses of new vulnerabilities. Updates to neural networks' parameters due to the learning process are performed periodically according to operational constraints.

The system start the continuous process computing loss prediction in two cases: first is whenever a new vulnerability is publishes and indicated in the NVD. Second is whenever a change is made or intended to be made in a system component or in systems' environment. In the case of testing a new component, the system computes losses as a simulation, before decision is made to move the component to the operational environment. Loss evaluation is based on NVD, CVSS, CMDB and the Historical Events Database. Whenever a component is found to be vulnerable according to NVD, the system performs a propagation process which computes all impacts on components which read or write data from the vulnerable component. Propagation algorithm runs until the final output is been transferred to the users or written to the output files. Propagation process uses CMDB to lead the process of interactions among components. The Learning algorithm writes the potential computed losses in the Historical Events Database. The Learning module forecasts the future potential losses caused by a specific vulnerability which was exploited on a component. Prediction will performed by running the neural network. Actual damage will be updated by organizations' owner on a regular basis to capture also delayed outcomes of a past vulnerability. The learning algorithm will improve economic prediction accuracy losses which will be based on the updated environment and the updated actual losses.

4. LOSS SCORING AND LEARNING

Scoring algorithm is implemented through the neural network. The architecture of the network is described in table 1 and a detailed design example of a network illustrated in table 2. Network design is based on Han and Kamber (2006). Implementation may be done using data mining software tool such as SAS business analytics software, or Weka.

The network represents all parameters impacting on the vulnerable component comprising the input layer. Parameters include vulnerability characteristics as updated in NVD. Characteristic example parameters are vulnerability category, vulnerability severity, and parameters describing components' specification such as vendor name and product ID (such as operating system version).

The input layer includes all CVSS parameter groups: Basic metrics, Temporal and Environmental metrics. As illustrated in table 2 parameters are categorized as they appear in CVSS. For example vulnerability access complexity includes three categories: high, medium and low.

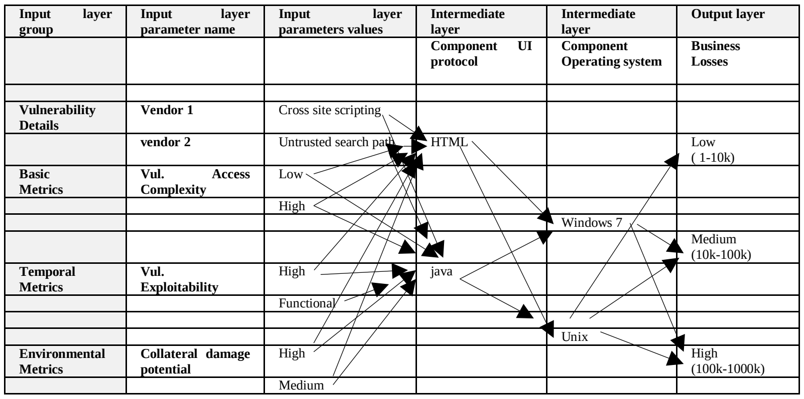
The hidden layers include a number of layers which represent messages from the exploited component to all other systems' components such as the operating systems in use, database, communication protocols used, UI programming language, and all other application components called directly or indirectly by the vulnerable component. The neural network represents the logical workflow of messaging and data transfers between the component and all other systems' components.

The output layer of the neural net represents losses occurring due to cyber attacks on components. Losses are categorized to low, medium, high and fatal. Losses represent actual business damages by past attacks on a component, as reported by the organization. Losses are reported on a regular basis until all late-effects are known, sometime in the future. This requires the nomination of a security person that would be responsible for regular reports of the vulnerabilities and damages.
Neural net input signals are represented by zeros and ones according to the existence of the specific parameter. Messages between neural network nodes are binary. Arcs between nodes represent kinds of dependencies between components. Output layer categories are also binary.
At the end of the process the system presents the predicted business loss category for attacks on one component. Each activation process of the network uses all computed weights between network nodes. The network may be programmed to predict attacks using a specific vulnerability or otherwise attacks using all vulnerabilities on that component.
After prediction of the business losses, the organization decides ways of mediating the vulnerability, whether to accept the risk, try to attenuate the risk, wait for a patch or live without the risky component.

The learning process is activated on a periodic basis generating updated weights to network arcs and components. The learning process is activated by three event types: (1) accepting indicators to a new vulnerability (2) Loss updates concerning past organizational losses (3) Changes performed to the computing configuration or environment. The training and learning process runs on the historical database of attacks by several forward and backward propagation processes until networks' termination conditions exist.

The proposed approach differs from existing scoring models such as CVSS by dynamic generation of the calculations involved in the scoring process. CVSS uses fixed coefficients which were calculated at a specific point in the past. Our framework predicts losses on a continuous basis, and updates network coefficients through learning on a periodic (or nearly continuous) basis subject to operational constraints.

Table 1 Neural Network architecture



Input layer group	Input layer parameter name	Input layer parameters values	Intermediate layer	Intermediate layer	Output layer
			Component UI protocol	Component Operating system	Business Losses
Vulnerability Details	Vendor 1	Cross site scripting			
	vendor 2	Untrusted search path	HTML		Low (1-10k)
Basic Metrics	Vul. Access Complexity	Low			
		High			
				Windows 7	
					Medium (10k-100k)
Temporal Metrics	Vul. Exploitability	High	java		
		Functional			
				Unix	
Environmental Metrics	Collateral damage potential	High			High (100k-1000k)
		Medium			

Table 2 Example of a Neural Network layer design

		Input layer		Intermediate layers				Output layer
Input layer group	Input layer parameter name	Input layer param values	Component UI protocol	Component Operating system	Database	Application components	Application components	Business Losses
Vulnerability Details	Vulnerability Category	Cross site scripting						

		Buffer overflow	HTML					
		SQL Injection						
	Vul. Vendor	Red Hat		Windows 7		Com. x		
		Algosec						
		Quantum						
	Vul. Product ID	1234						
		3456		Unix				
	Vul. Severity	High			Oracle			
		Medium					Com. 1	
		Low						
	Vul. start duration	Less 1 week	JAVA	Windows 8				
		Less 1 month						
		Less 1 year						
		More 1 year						
Basic Metrics	Vul. Access Complexity	Medium				Comp y		
		High						
		Low						
	Vul. Access Vector	Local						Medium (10k-100k)
		Adjacent						
		Network	Javascript					
	Vul. Access	Multiple		Windows XP			Com. 2	
	Authentication	Single			DB2			
		None						
	Vul. Confidentiality Impact	None						
		Partial						
		Complete						
	Vul. Integrity Impact	None		Windows NT				
		Partial						
		Complete						
	Vul. Availability	None	AJAX					
		Partial						
		Complete						
				OS/360				
Temporal Mertics	Vul. Exploitability	Unproven			SQL			
		Proof of concept						
		Functional						
		High						
		Not Defined						
	Vul. Remediation Level	Official Fix						High (100k-1000k)
		Temporary		Os X				
		Workaround						
		Unavailable						
		Not Defined						
	Report Confidence	Unconfirmed	.NET					
		Uncorroborated						
		Confirmed						
		Not Defined						
Environmental Metrics	Collateral damage potential							
			C				Com. 3	
		Input layer		Intermediate layers				Output layer
Input layer group	Input layer parameter name	Input layer param values	Component UI protocol	Component Operating system	Database	Application components	Application components	Business Losses
	Target component Distribution	Low						
		Medium			SQL			
		High						
	Target CR		C++					Fatal (1000k -)
	Target IR						Com. 4	
				android		Comp w		
	Target AR							

5. CONCLUSIONS

In this work we described a new framework of Security Continuous Monitoring (SCM) system and its mechanisms, including neural network architecture aimed at increasing security of information systems by improving and accelerating loss prediction. The system introduces four new capabilities: (1) Continuous real-time loss prediction software agent using real time environmental parameters for an improved loss prediction algorithm. (2) Components' interdependencies are used by a propagation algorithm for loss prediction, (3) Risk prediction is based on actual losses reported by the organization and (4) a learning algorithm which is based on a process of updating the facts concerning vulnerabilities' actual losses and real-time IT configuration.
The framework enables getting improved recommendations to computer owners concerning new relevant vulnerabilities. The framework also enables improved security management of the operating systems. For example, in cases where a vulnerability to the new asset is publicly known but still un-patched, loading a new version of a software component will be prevented by performing a preliminary simulation test which analyzes vulnerabilities of the new component, incorporated in the operational environment.

Several future research directions exist: performing a proof of concept of the framework for evaluation of the model, investigating defense methods against attack vectors involving several different vulnerabilities, searching new hidden vulnerabilities in a production environment. Further research could extend the resolution of the entities used in our model so that entities will include data items with the appropriate specifications such as security requirements, and interdependencies between components indicating data transfer between data items.

REFERENCES

Albanese M., Jajodia S., Jhawar R., and Piuri V., (2013). Reliable Mission Deployment in Vulnerable Distributed Systems, proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Budapest, Hungary, June 24-27, 2013.

Collier Z. A., DiMase D., Walters S., Tehranipoor M., Lambert J. H., Linkov I., (2014). Cybersecurity Standards: Managing Risk and Creating Resilienc, Computer, Vol. 47 Issue No. 09 September 2014, IEEE.

Dempsey K., Chawia N. S., Johnson A., Johnston R., Jones A., C., Orebaugh A., Scholl M., and Stine K., (2011). Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations, NIST.

Grimalia M. R., Fortson L. W., and Sutton J. L., (2009). Design considerations for a cyber Incident Mission Impact Assessment (CIMIA) Process, Proceedings of the 2009 International Conference on Security and Management (SAM09), Las Vegas.

Han J., and Kamber M., (2006) Data Mining: Concepts and Techniques, 2nd ed. San Francisco, CA, Morgan Kaufmann Publishers.

Hardy M. G., (2012). Beyond Continuous Monitoring: Threat Modeling for Real-time Response, SANS Institute.

Holsopple J., and Yang S. J., (2008). FuSIA: Future Situation and Impact Awareness, in Proceedings of the 11th International Conference on Information Fusion, Cologne, Germany, July 1-3 2008, ISIS.IEEE.

Jajodia S., Noel S., Kalapa P., Albanese M., and Williams J., (2011). Cauldron: Mission-Centric Cyber Situational Awareness with Defense in Depth, in Proceedings of the Military Communications Conference, (pp. 1339-1344), USA.

Jakobson G., (2011). Mission Cyber Security Situation Assessment Using Impact Dependency Graphs, The 14th International Conference on Information Fusion, Chicago, USA, July 5-8, 2011.

Keller A. and Subramanianm S., (2009). Best practices for deploying a CMDB in large-scale environments, Proceedings of the IFIP/IEEE international conference on Symposium on Integrated Network Management, pages 732-745, NJ, IEEE Press Piscataway.

Kotenko I. and Chechulin A., (2014). Fast Network Attack Modeling and Security Evaluation based on Attack Graphs, Journal of Cyber Security and Mobility Vol. 3 No. 1 pp 27-46.

Langer L., (2011). Stuxsnet: Dissecting a Cyber Warfare Weapon, Security and Privacy IEEE, Volume: 9 Issue: 3, pages 49-51, NJ, USA.

Mell P., Scarfone K., and Romanosky S., (2007). CVSS - A Complete Guide to the Common Vulnerability Scoring System, Version 2.0, Retrieved on October 13, 2014 from http://www.first.org/cvss/cvss-guide.

Nũez Y. F., (2008). Maximizing an organization's information security posture by distributedly assessing and remeding system vulnerabilities, 2008 IEEE, International Conference on Networking, Sensing and Control, China, April 6-8, 2008.

Scarfone K., and Mell P., (2007). Guide to Intrusion Detection and Prevention Systems (IDPS), NIST, 2007.

Symantec, (1997). Understanding Heuristics: Symantec's Bloodhound Technology, White paper XXXIV.

Tom S., Christiansen D., Berrett D., (2008). Recommended Practice for Patch Management of Control Systems, DHS National Cyber Security Division Control Systems Security Program.

Wang L., Noel S., and Jajodia S., (2006). Minimum-cost network hardening using attack graphs, Computer Communications 29, Issue 18, pp. 3812–3824.