Volume 5, Issue 10, October-2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Seclius with Markov Chain Monte Carlo Information Flow-Based Security Metric for Intrusion Detection System

R . Dhanapakkiyam
M.Phil Research Scholar
PG & Research Department of Computer Science
Government Arts College, Coimbatore,
Tamilnadu, India

Dr. K. Saraswathi
Assistant Professor
PG & Research Department of Computer Science
Government Arts College, Coimbatore,
Tamilnadu, India

Abstract—Energy delivery system infrastructure in IT systems is critical to monitor because of the Intrusion Detection Systems (IDSes) which often produce thousands of alerts daily that must be dealt with by administrators manually. To overcome the limitations, information flow-based system security metric called Seclius with Markov Chain Monte Carlo (MCMC) method is introduced. The proposed work gives security measure for a given user as the probability value determined from the Markov Chain Monte Carlo (MCMC). The Consequence tree (CT's) root value is measured by evaluating real time Intrusion Detection System (IDS) alerts to detect system and network assets' security which are affected by attackers. Because of this reason our proposed work is different from existing methods. Two components are used in this online evaluation that are: 1) a dependency graph, and 2) a consequence tree. These two components are designed to identify the information in each IDS alert to assess the security state of the different assets in the organization. The performance and accuracy of Seclius-MCMC are evaluated and Seclius in a testbed environment are implemented to reproduce a process control environment, and a realistic multistep attack scenario.

Keywords— Intrusion Detection Systems (IDS), system security metric, information flow-based analysis, Markov Chain Monte Carlo (MCMC).

I. INTRODUCTION

Security maintenance is very difficult to systems and networks against attackers. The increased number of security incidents [1] shows that current approaches of building systems do not sufficiently address the increasing variety and sophistication of threats and block attacks before systems are compromised. Therefore, organizations must detect malicious activity that are occurring, so efficient intrusion detection systems (IDSes) [2] are used to monitor the systems and identify misbehavior. However, IDSes alone are not enough to make operators to understand the security state of their organization, because monitoring sensors usually report all potentially malicious traffic with respect to the actual network configuration, vulnerabilities, and mission impact. With large volumes of network traffic, even small error rates of IDSes can identify false alarms. Sometimes eventhough true intrusions are detected, the actual mission threat is often unclear, and operators are unsure regarding the actions they should take. In fact, to respond effectively to system and in order to prioritize their response and recovery actions, security administrators need to update estimated summaries regarding the security status of their mission-critical assets precisely and continuously, based on alerts that occur. This is even stronger in the context of cyber-physical energy delivery systems infrastructures, in which a cyber-side malicious activity can have catastrophic physical consequences, e.g., the Trans-Siberian pipeline explosion due to a computer Trojan horse [3] or the Stuxnet computer worm [4] which was specifically designed to compromise the programmable logic controllers used in nuclear power plants.

To detect network attacks, special systems have been developed; these systems are called Network Intrusion Detection Systems (NIDS). In order to find known attacks or unusual behaviour, these systems traditionally inspect the contents (payload) of every packet [5]. The problem of packet inspection is impossible, to perform it at multiple Gigabits per second (Gbps) [6-7]. To achieve high-speed lines, it is therefore important to investigate alternatives to packet inspection. One option is the attention of researchers and operators is intrusion detection. With this, the communication patterns within the network are analyzed, instead of the contents of individual packets. Nowadays special measurement systems are available, for every IP addresses and port numbers, such as the time data exchange has started, the time it has stopped, the amount of transferred bytes and the number of sent packets. These systems use this information in the form of Netflow [8-9] records and analyse them. These analysis systems can then be used to detect intrusions. In opinion, flow-based detection is opposite method of packet inspection, and should not be taken as a replacement. Both approaches can be combined into a two-stage detection process. At the first stage, flow-based approaches can be used to detect certain attacks. At the second stage, packet inspection can be used to additionally protect critical servers or selected systems, for which the first stage has discovered suspicious activities. Flows are formed by specialized accounting modules that are placed in network routers. The same modules are responsible for exporting the flows to external collectors. Flow-based Intrusion Detection Systems will analyse these flows and detect attacks. Lower amount of data is considered to handle data compared to traditional NIDS, flow-based NIDS. Moreover, considering the network load measured in bytes, the overhead due to Net flow is in average 0.2%. Flow based intrusion detection is a logical choice for high-speed networks. However, there might exist situations in which the benefit of using flows is not used well. The worst case scenario would be when a flow is created for each packet passing through the monitoring point, as a consequence of a distributed DoS attack (DDoS), for example. In this case, the number of flows would increase dramatically and extra load would be put on the monitoring and analysis systems. To mitigate this problem, or, in general, to improve the performance of routers and monitoring stations, sampling techniques or flow aggregations [10] can be applied. Second, previous techniques for IDS alert correlation and system security state estimation usually focus only on the attack paths [11-12] and subsequent privilege escalations [13], without considering dependencies between system assets. However, in practice, there are often unsuccessful attacks that cause partial damage to systems, such as a Web server crash as the result of an unsuccessful buffer overflow exploitation. The proposed work gives security measure for a given user as the probability value determined from the Markov Chain Monte Carlo (MCMC). The Consequence tree (CT's) root value is measured by evaluating real time Intrusion Detection System (IDS) alerts to detect system and network assets' security which is affected by attackers. Because of this reason our proposed work is different from existing methods. Two components are used in this online evaluation that are: 1) a dependency graph, and 2) a consequence tree. These two components are designed to identify the information in each IDS alert to assess the security state of the different assets in the organization. In summary, the contributions and benefits of this paper are the followings.

- If critical assets have been compromised, our system introduces a defense-centric metric to probabilistically determine. The metric fills the gap between low-level IDS alerts and the need for comprehensive situational awareness.
- The main limitations of existing security metrics are addressed by 1) minimizing human involvement and removing assumptions about attackers' behaviors and vulnerabilities, and 2) considering probabilistic dependencies between assets to understand the impact of malicious intrusions.
- Experimentation results show how Seclius with Markov Chain Monte Carlo (MCMC) (Seclius-MCMC) can be used to rank IDS alerts or highlight critical assets based on the priority.
- The performance and accuracy of Seclius-MCMC are evaluated and Seclius in a testbed environment are implemented to reproduce a process control environment, and a realistic multistep attack scenario.

Compare results with the Seclius approaches to generating the system dependency graph, namely instruction level taint-tracking and the syscall interception, and show that the syscall interception provides a reasonably accurate dependency model while causing low computational overhead on the system

II. RELATED WORK

Much research has been conducted over the past decade on the generic topics of system situational awareness and security metrics. In this section, discuss how Seclius contributes to enhance the state-of-the-art. In particular, [13] uses an approach called Topological Vulnerability Analysis (TVA) to match network configuration with attack simulation in order to optimize IDS sensor placement and to prioritize IDS alerts. The main issue with attack-graph-based techniques is that they require important assumptions about attacker capabilities and vulnerabilities. The attack graph is covered by placing IDS sensors, using the few sensors. This reduces the sensor's cost, with the effort of deploying, configuring, and maintaining them, while maintaining complete coverage of potential attack paths. The sensor-placement problem we pose is an instance of the NP-hard minimum set cover problem. An efficient greedy algorithm which works well in practice is used to solve this problem. There have been several efforts to take into account unknown vulnerabilities during the system security analysis.

Zhang et al [14] evaluates the unknown vulnerabilities in well-known software applications through analysis of the past available historical data set about their know vulnerabilities and the corresponding statistics. The authors conclude that the vulnerabilities do not follow a particular pattern and also are not predictable accurately. In particular, one would like to be able to predict the likelihood that a piece of software that contains a yet-to-be-discovered vulnerability, which must be taken into account in security management due to the increasing trend in zero-day attacks. Empirical study on applying data-mining techniques on National Vulnerability Database (NVD) data is conducted with the objective of predicting the time to next vulnerability for a given software application.

Ou [15] introduces a logic-based hypothetical analysis of system components that would be affected if a particular currently-unknown vulnerability existed in the system. Built an end-to-end system, MulVAL, that is based on the methodology discussed in this thesis. In MulVAL, a succinct set of Datalog rules captures generic attack scenarios, including exploiting various kinds of software vulnerabilities, operating-system semantics that enables or prohibits attack steps, and other common attack techniques. The reasoning engine takes inputs from various off-the-shelf tools and formal security advisories, performs analysis on the network level to determine if vulnerabilities found on individual hosts can result in a condition violating a given high-level security policy.

Although it is impossible to predict the existence of any specific zero-day vulnerability the rationale behind NetSPA by Ingols et al [16] is that, it is possible to hypothesize zero-day vulnerability in specific software applications to ensure that the impact of an eventual zero-day can be understood and minimized. The hypothetical analysis of unknown vulnerabilities provides a great solution to find the impact of vulnerability in any specific point in the system; however, a complete system security analysis would require hypothesizing vulnerabilities in every possible system point that is not an optimally scalable solution for large-scale infrastructures. While these approaches are important in planning for future attack scenarios, we take a different perspective by relying on past consequences, actual security requirements, and low-level system characteristics, such as file and process dependencies, instead of hypothetical attack paths. As a result, our method is defense-centric rather than attack-centric and does not suffer from the issues of unknown vulnerabilities and incomplete attack coverage.

A security metric measures or assesses the extent to which a system meets its security objectives. Since meaningful quantitative security metrics are largely unavailable, the security community primarily uses qualitative metrics for security. In [17] proposes a novel quantitative metric for the security of computer networks that is based on an analysis of attack graphs. The metric measures the security strength of a network in terms of the strength of the weakest adversary who can penetrate the network. An algorithm that computes the minimal sets of required initial attributes for the weakest adversary to possess in order to successfully compromise a network is proposed; given a specific network configuration, set of known exploits, a specific goal state, and an attacker class (represented by a set of all initial attacker attributes). It also shows by example, that diverse network configurations are not always beneficial for network security in terms of penetrability

Other important defense-centric approaches include FuzMet [18]. Manually filled knowledge bases of alert applicability, system configuration, or target importance is used to associate a context with each alert and to provide situational awareness accordingly. Seclius offers a more practical solution by minimizing and simplifying the required manual inputs. It does by learning characteristics of low-level system automatically in order to evaluate precisely the extent to affect the critical components of the organization. Such a damage assessment feature has previously been explored via file-tainting analysis for malware detection, for offline forensic analysis using backtracking. At the network level, [15] a vulnerability definition language (OVAL) and a network dependency graph is created to measure the potential impact of vulnerabilities.

In [19], information flow is tracked across multiple layers, at the instruction and OS process level. Compared to our approach, this cross-layer technique is more precise, but it requires implementation in a virtual environment and can cause major performance degradation. Damage assessment plays a very important role in securing enterprise networks and systems. Good awareness about the effects and impact of cyber attack actions would enable security officers to make the right cyber defense decisions and take the right cyber defense actions. A good number of damage assessment techniques have been proposed in the literature, but they typically focus on a single abstraction level (of the software system in concern). As a result, damage assessment techniques which are existing and tools are still very limited in satisfying the needs of comprehensive damage assessment which should not result in any “blind spots”.

Attack graphs [20] provide practical attack context and relationships among vulnerabilities, researchers have been trying to evaluate network security based on attack graphs. However, previous works focus their attention on specific evaluations they concerned, and each does things in his own way. There is no any other way of telling network administrators how to measure network security in a general way. In this paper, a new metric framework, whose main goal is to guide people, is proposed to perform evaluations based on attack graphs. The main components of proposed metric framework include security index, target of evaluation, elementary attribute, composition algorithm, and arithmetic operators. Relative definitions and analysis of these five components are also given. The following examples show the applications of our metric framework, and validate it.

III. SYSTEM OVERVIEW AND PROPOSED MONTE CARLO MARKOV CHAIN FLOW-BASED SECURITY METRIC METHODOLOGY

The first manual input required by Seclius, namely the consequence tree (CT) is discussed. The goal of the CT is to find critical IT assets and organizational security requirements. The criticality level of individual assets within an organization is an environment-specific issue. In other words, the criticality levels depend on the organizational missions and/or business-level objectives. For instance, a power grid process control environment whose mission is generation of power and its secure delivery to the customers is considered. In such an environment, provision of high-availability guarantees for a programmable logic controller, which is used to control an important power generator, is more critical than for a historian, which is used to store historical sensor measurements for later analyses. Hence, Seclius requires system administrators to manually provide the list of organizational critical assets.

Functions that provide the critical assets are: a simple list (meaning that all items are equally important), a weighted list, or a more complex combination of assets. In this work logical tree structure is chosen. A good trade-off between simplicity and expressiveness is existing, and the fact that it can be represented visually makes it a particularly helpful resource for administrators. The formalism of the CT follows the traditional fault tree model; however, unlike fault trees, the leaf nodes of the CTs in Seclius address security requirements (confidentiality, integrity, and availability) of critical assets, rather than dependability criteria. The CT formalism has two major types of logical nodes, namely AND and OR gates. An organizational CT is designed, such a way that the administrator starts with the tree’s root node, which identifies the main high-level security concern/requirement, e.g., ‘‘Organization is not secure.’’ The rest of the tree recursively defines how different combinations of the more concrete and lower-level consequences can lead to the undesired status described by the tree’s root node. The recursive decomposition procedure stops once a node explicitly refers to a consequence regarding a security criterion of a system asset, e.g., ‘‘availability of the Apache server is compromised.’’ These nodes are in fact the CT’s leaf consequence nodes, each of which takes on a binary value indicating whether its corresponding consequence has happened (1) or not (0). Throughout the work, use the C, I, and A function notations to refer to the CIA criteria of the assets. The leaves’ values can be updated by IDSes, e.g., Samhain

[21]. The CT is derived as a Boolean expression, and the root node's value is consequently updated to indicate whether the organizational security is still being maintained.

A CT indeed makes subjective aspects of the system. Its leaf nodes list security criteria of the organization's critical assets. Additionally, the CT encodes how critical each asset is using the depth of its corresponding node in the tree; that is, the deeper the node is, the less critical the asset is in the fulfillment of the organizational security requirements. Furthermore, the CT formulates redundant assets using AND gates. Seclius requires administrators to explicitly mention the redundancies because it is often infeasible to discover redundancies automatically over a short learning phase. As mentioned in the previous section, the CT captures only the subjective security requirements, and does not require deep-knowledge expertise about the IT infrastructure, thanks to the dependency graph (DG). The goal of the DG is to free the administrator from providing low-level details about the organization. Those details are automatically captured by Seclius through a learning phase, during which the interactions between files and processes are tracked in order to probabilistically identify direct or indirect dependencies among all the system assets. For instance, in a database server, the administrator only needs to list the sensitive database files, and Seclius later marks the process mysql as critical because it is in charge of reading and modifying the databases. Such a design greatly reduces the resources and time spent by administrators in deploying Seclius.

Each vertex in the DG represents an object, namely a file, a process, or a socket, and the direct dependency between two objects is established by any type of information flow between them. For instance, if data flow from object o_i to o_j, then object o_j becomes dependent on o_i; the dependency is represented by a directed edge in the DG, i.e., $o_i \rightarrow o_j$. To capture that information, Seclius intercepts syscalls and logs them during the learning phase. In particular, are interested in the syscalls that cause data dependencies among the OS-level objects. A dependency relationship is stored by three elements: a source object, a sink object, and their security contexts. When the learning phase is over, syscall logs are automatically parsed and analysed line by line to generate the DG. Each dependence edge is tagged with a frequency label indicating how many times the corresponding syscalls were called during the execution.

In this work make use of the Bayesian network formalism to store probabilistic dependencies in the DG; a conditional probability table (CPT) is generated and associated with each vertex. This CPT encodes how the information flows through that vertex from its parents (sources of incoming edges) to its children. For example, if some of the parent vertices of a vertex become tainted directly or indirectly by attacker data, the CPT in the vertex saves the probability whose vertex (specifically, the OS-level object represented by the vertex) also gets tainted. Bayesian analysis gives us a very simple recipe for learning from data: given a set of unknown parameters or latent variables z that are of interest, specify a prior distribution p(z) quantifying what we know about z before observing any data. Then we quantify how the observed data x relates to z by specifying a likelihood function p(x|z). Finally, apply Bayes' rule $p(z|x) = p(z)p(x|z)/\int p(z)p(x|z)dz$ to give the posterior distribution, which quantifies what know about z after seeing the data . The two most popular approximation methods for this purpose are variational inference and Markov Chain Monte Carlo (MCMC). Variational inference casts Bayesian inference as an optimization problem where we introduce a parameterized posterior approximation $q_\theta(z|x)$ which is fit to the posterior distribution by choosing its parameters $\mathcal{L}$ to maximize alower bound L on the marginal likelihood:

$$\log p(x) \geq \log p(x) - D_{KL}(q_\theta(z|x)||p(z|x)) \qquad (1)$$

$$= \mathbb{E}_{q_\theta(z|x)}[\log p(x) - log(q_\theta(z|x)] \qquad (2)$$

A popular alternative to variational inference is Markov Chain Monte Carlo (MCMC) method. MCMC starts by taking a random draw z_0 from some initial distribution $q(z_0)$ or $q(z_0|x)$. Rather than optimizing this distribution, however, MCMC methods subsequently apply a stochastic transition operator to the random draw z_0:

$$z_t \sim q(z_t|z_{t-1}, x) \qquad (3)$$

By choosing the transition operator $q(z_t|z_{t-1}, x)$ and iteratively applying it many times, the outcome of this procedure, z_T , will be a random variable that converges in distribution to the exact posterior $p(z|x)$. The advantage of MCMC is that the samples it gives us can approximate the exact posterior arbitrarily well if are willing to apply the stochastic transition operator a sufficient number of times. The central idea of this paper is that we can interpret the stochastic Markov chain $q(z|\, x) = q(z_0|x) \prod_{t=1}^{T} q\,(z_t|z_{t-1}, x)$ as a variational approximation in an expanded space by considering $y = z_0, z_1, \ldots, z_{t-1}$ to be a set of auxiliary random variables. Integrating these auxiliary random variables into the variational lower bound (2), obtain

$$\mathcal{L}_{aux} = \mathbb{E}_{q(y,z_T|x)}[\log p(x, z_T) r(y|x, z_T)] - log(q(y, z_T|x)] \qquad (4)$$

where $r(y|x, z_T)$ is an auxiliary inference distribution which we are free to choose, and our marginal posterior approximation is given by $q(z_T\,|x) = \int q(y, z_T\,|x)dy$. The marginal approximation $q(z_T\,|x)$ is now a mixture of distributions of the form $q(z_T\,|x, y)$. Since this is a very rich class of distributions, auxiliary variables may be used to obtain a closer fit to the exact posterior [22]. The choice $r(y|x, z_T) = q(y|x, z_T\,)$ would be optimal, but again often intractable to compute; in practice, good results can be obtained by specifying a $r(y|x, z_T)$ that can approximate $q(y|x, z_T\,)$ to a reasonable degree. One way this can be achieved is by specifying $r(y|x, z_T)$ to be of some flexible parametric form, and optimizing the lower bound over the parameters of this distribution. In this paper consider the special case where the auxiliary inference distribution also has a Markov structure just the posterior approximation: $r(z_0, \ldots, z_{t-1}|x, z_T\,) = \prod_{t=1}^{T} r_t\,(z_{t-1}|x, z_t)$, in which case the variational lower bound can be rewritten as

$$\log p(x) \geq \mathbb{E}_q[\log p(x, z_T) - log(q(y, z_T|x)] + log\, r(z_0, .. z_{t-1}|x, z_T) \qquad (5)$$

where the subscript t in q_t and r_t highlights the possibility of using different transition operators qt and inverse models r_t at different points in the Markov chain. By specifying these q_t and r_t in some flexible parametric form, can then optimize the value of (4) in order to get a good approximation to the true posterior distribution. The key insight behind the recent work in stochastic gradient variational inference is that if all the individual steps of an algorithm like this are differentiable in the parameters of q and r, which denote by θ, then so is the algorithm's output L. Since L is an unbiased estimate of the variational lower bound, its derivative is then an unbiased estimate of the derivative of the lower bound, which can be used in a stochastic optimization algorithm. Obtaining gradients of the Monte Carlo estimate of Algorithm 1 requires the application of the chain rule through the random sampling of the transition operators $q_t(z_t|x, z_{t-1})$. This can in many cases be realized by drawing from these operators in two steps: In the first step we draw a set of primitive random variables u_t from a fixed distribution $p(u_t)$, and we then transform those as $z_t = g_\theta(u_t, x)$ with a transformation $g_\theta()$ chosen in such a way that z_t follows the distribution $q_t(z_t|x, z_{t-1})$. If this is the case we can apply backpropagation, differentiating through the sampling

IV. SIMULATION SETUP AND EVALUATION

In this section the accuracy and performance of the various components of Seclius are evaluated and Seclius is integrated to Markov Chain Monte Carlo (MCMC) (Seclius –MCMC) schemas. The experiments in this section were conducted on a virtual machine so that could easily reset the testing environment to a clean state. The host computer had a 2.20 GHz AMD Athlon 64 3700 + CPU, 1MB of cache, and 2.0 GB of RAM. The guest OS, using VirtualBox, was running Ubuntu 9.04 with a Linux 2.6.22 kernel.

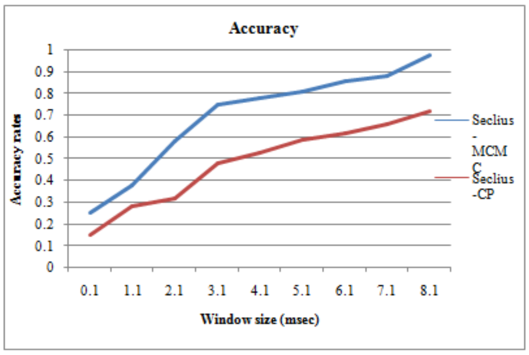

Fig. 1. Accuracy of syscall taint tracking vs. methods

Numbers of false positives and false negatives depends directly on the size of t. varied t when tracking the Apache process and compared the results to TEMU's. The false positive rate was calculated as #FalsePositives=(#FalsePositives/ #TrueNegatives). Similarly, the false negative rate was estimated using #FalseNegatives=(#FalseNegatives/#TruePositives). Fig. 1 shows the results for the various window sizes that are employed. The false positive rate grows linearly up to 1.0 as the window size exceeds 8.1 milliseconds. The false negatives, on the other hand, quickly drop to zero before the window size gets to 2 milliseconds. Based on these results, 1.5 milliseconds seems to be a reasonable window size for our environment, as both the false positive and negative rates remain fairly low, the results are shown in table 1.

TABLE I. ACCURACY OF SYSCALL TAINT TRACKING VS METHODS

	Accuracy rates	
Window size (msec)	**Seclius-MCMC**	**Seclius-CP**
0.1	0.25	0.15
1.1	0.38	0.28
2.1	0.58	0.32
3.1	0.75	0.48
4.1	0.78	0.53
5.1	0.81	0.59
6.1	0.86	0.62
7.1	0.88	0.66
8.1	0.98	0.72

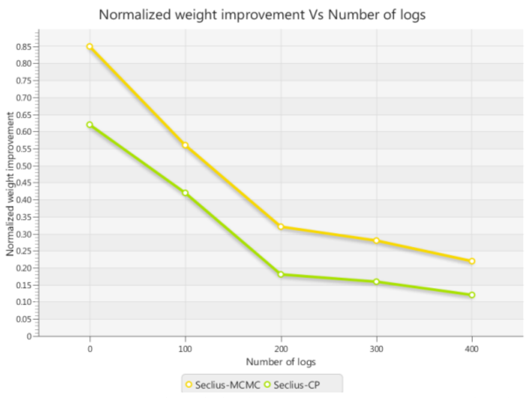

Fig. 2. DG's edge label convergence vs methods

Fig.2 shows the normalized label updates, i.e., the difference between the last and the updated values, while syslogs were being parsed between Seclius and Seclius is integrated to Markov Chain Monte Carlo (MCMC) (Seclius –MCMC) schemas . As shown in the Fig. 2, it took about 3 minutes on average for the edge labels to converge. The launched request flow caused a spike in the graph, but the normalized label updates quickly converged back to zero again.

TABLE. II . DG'S EDGE LABEL CONVERGENCE VS METHODS

Number of logs	Seclius-MCMC	Seclius-CP
	Normalized weight improvement	
0	0.85	0.62
100	0.56	0.42
200	0.32	0.18
300	0.28	0.16
400	0.22	0.12

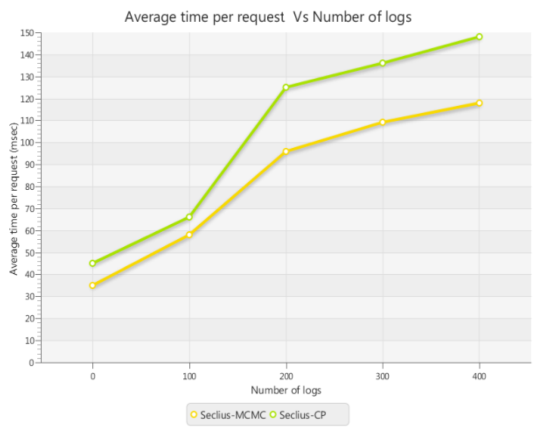

Fig. 3. Overhead syscall vs methods

Fig. 3 shows each technique caused overhead within the system in terms of end-point response time for the Apache Web server. The server was benchmarked against different numbers of concurrent clients. As shown in the Fig.3, the syscall interception caused less than 10 percent overhead on average compared between Seclius. Seclius is integrated to Markov Chain Monte Carlo (MCMC) (Seclius –MCMC) schemas, which was not instrumented, whereas the TEMU engine caused an approximately 850 percent slowdown in the Web server response time, which is clearly not acceptable for practical uses.

TABLE III. OVERHEAD SYSCALL VS METHODS

Number of logs	Seclius-MCMC	Seclius-CP
	Average time per request (msec)	
0	35	45
100	58	66
200	96	125
300	109	136
400	118	148

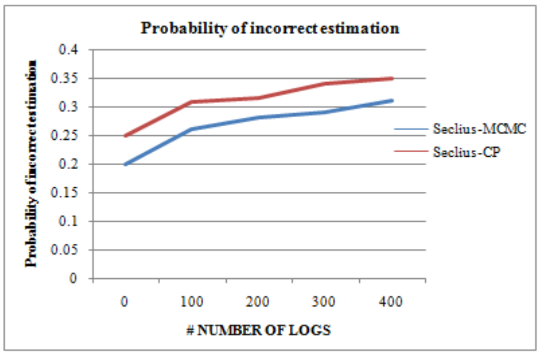

Fig.4. Probability of Incorrect Estimation vs methods

Fig.4 shows the results for different Seclius and Seclius is integrated to Markov Chain Monte Carlo (MCMC) (Seclius –MCMC) schemas. When generated tree approaches depth is 16, i.e., 216 leaf nodes, the probability of getting incorrect security estimation goes to zero for a fixed number of missing nodes. Note that when the number of missing nodes, shown on the graph, is larger than the tree order, all nodes are in fact missing; in other words, security is estimated without use of the CT, and hence, the estimated security is correct with an approximately 0.5 chance.

TABLE IV. PROBABILITY OF INCORRECT ESTIMATION VS METHODS

Number of logs	**Seclius-MCMC**	**Seclius-CP**
	Probability of incorrect estimation	
0	0.2	0.25
100	0.26	0.31
200	0.28	0.315
300	0.29	0.34
400	0.31	0.35

V. CONCLUSION AND FUTURE WORK

In this paper Seclius is integrated to Markov Chain Monte Carlo (MCMC), to measure the probability that critical assets have been directly or indirectly compromised by an online security evaluation framework that leverages dependencies between Operating System (OS)-level objects. First, the subjective security requirements are captured by consequence tree and administrator input is minimized. Second, the evaluated security value will be affected by the accuracy of the underlying intrusion detection solutions, i.e., if some malicious events are missed by the intrusion detectors. The major contribution in this work is it shows the use of the system dependency graph and the security requirements to evaluate the security of any given state; in other words, do not claim to have come up with a new intrusion detection technique. Second, Seclius is integrated to Markov Chain Monte Carlo (MCMC) (Seclius-MCMC) processes IDS alerts online to measure actual attack consequences and does not rely on assumptions about attacker behaviors or system vulnerabilities. Seclius-MCMC to evaluate system security, takes under consideration the intrusion detection inaccuracies, i.e., false positive and negative rates, if provided. Security evaluation is done by Seclius-MCMC based on the past consequences, which are easier to detect exploitations. Additionally, as Seclius-MCMC is information flow-based metric, when the system has not yet been attacked, Seclius usually evaluates the system security to be close to absolute, but not 100 percent secure.

REFERENCES

[1] Sharma, Z. Kalbarczyk, R. Iyer, and J. Barlow, "Analysis of Credential Stealing Attacks in an Open Networked Environment,"*in 4th International Conference on In Network and System Security (NSS),* pp. 144-151,2010.

[2] Axelsson S., *Intrusion Detection Systems: A Survey and Taxonomy*, U.S. Dept. Energy, Washington, DC, Tech. Rep. 99-15, 2000.

[3] Reed T. and Bush G., *At the Abyss: An Insider's History of the Cold War*. New York, NY, USA: Random House Publ. Group, 2005.

[4] Falliere N., Murchu L.O., and Chien E., *W32.Stuxnet Dossier*, Symantic Security Response, Mountain View, CA, USA, Tech. Rep. 99-15, Oct. 2010.

[5] Roesch M., Snort, intrusion detection system, Jul. 2008. [Online]. Available: http://www.snort.org

[6] Paxson V., "Bro: a system for detecting network intruders in real-time," *Computer networks* , vol. 31, no. 23–24, pp. 2435–2463, 1999.

[7] Lai H., Cai S., Huang H., Xie J., and Li H., "A parallel intrusion detection system for high-speed networks*," in proceedings of the second international conference applied cryptography and network security (ACNS'2004)*, pp. 439–451,2004.

[8] Gao M., Zhang K., and Lu J., "Efficient packet matching for gigabitnetwork intrusion detection using TCAMs," *in proceedings of 2th international conference on advanced information networking and applications* , pp. 249–254,2006.

[9] Cisco.com, Cisco IOS NetFlow Configuration Guide, Release 12.4, http://www.cisco.com, Jul. 2008.

[10] Claise B., Cisco Systems NetFlow Services Export Version 9,RFC 3954 (Informational), Jul. 2008

[11] Song S. and Chen Z., "Adaptive network flow clustering," *in IEEE international conference on networking , sensing and control (ICNSC07),* pp. 596–601,2007.

[12] Xie P., Li J.H., Ou X., Liu P., and Levy R., ''Using Bayesian Networks for Cyber Security Analysis,'' *in Proc. IEEE/IFIP International Conference. DSN*, pp. 211-220,2010.

[13] Noel S. and Jajodia S., ''Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs,'' *Journal Networking System Management* , vol. 16, no. 3, pp. 259-275, 2008.

[14] Zhang S., Caragea D., and Ou X., ''An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities,'' *in Proceedings. Database Expert System0 Applications*, pp. 217-231,2011.

[15] Ou X. and Appel A.W., *A Logic-Programming Approach To Network Security Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 2005.

[16] Ingols K., Chu M., Lippmann R., Webster S., and Boyer S., ''Modeling Modern Network Attacks and Countermeasures Using Attack Graphs,'' *in Proceedings of IEEE ACSAC,* pp. 117-126,2009.

[17] Pamula J., Jajodia S., Ammann P., and Swarup V., "A Weakest- Adversary Security Metric for Network Configuration Security Analysis," *in Proceedings 2nd ACM Workshop Qual. Protect*., pp. 31-38,2006.

[18] Alsubhi K., Al-Shaer E., and Boutaba R., "Alert Prioritization in Intrusion Detection Systems," *in Proceedings of IEEE NOMS*, pp. 33-40, 2008.

[19] Liu P., Jia X., Zhang S., Xiong X., Jhi Y.C., Bai K., and Li J., *Cross- Layer Damage Assessment for Cyber Situational Awareness*, in Cyber Situational Awareness, vol. 46. New York, NY, USA: Springer-Verlag, pp. 155-176,2010.

[20] Xie A., Wen W., Zhang L., Hu J., and Chen Z., ''Applying Attack Graphs to Network Security Metric,'' *in Proceedins of International Conference MINES*, vol. 1, pp. 427-431,2009.

[21] Wotring B., Potter B., Ranum M., and Wichmann R., *Host Integrity Monitoring Using Osiris and Samhain*. Boston, MA, USA: Syngress Publ., 2005.

[22] Salimans, Tim and Knowles, David A, "Fixed-form variational posterior approximation through stochastic linear regression, " *Bayesian Analysis*, vol.8,no.4,pp.837–882, 2013.