2018 3(1) 17-24 Submission: 2018/02/10, Peer review: 2018/02/20, Accepted: 2018/02/25, Published: 2018/03/30

International journal of crisis & safety

Publication state: Japan
ISSN: 2189-7603

Publisher: J-INSTITUTE
Website: http://www.j-institute.jp

Corresponding author
E-mail: boseok4u@knu.ac.kr

Peer reviewer
E-mail: editor@j-institute.jp

http://dx.doi.org/10.22471/crisis.2018.3.1.17

Enterprise SECURITY Management for IoT Services Based on Event Correlation in Republic of KOREA

Park Bo-seok

Kyungpook National University, Daegu, Republic of Korea

Abstract

With radical development of information and communication Technology, Internet of Things(IoT) era has come. All the things around us are connected through internet so that it enables objects to exchange data with connected devices and is expected to offer new advanced services that goes beyond the value where each existing objects could have offered respectively. Concerns regarding security threat are being raised in adopting IoT as the number of internet-connected appliances are rapidly increasing. So, we need to consider how to protect and control countless objects. However, traditional security systems including intrusion detection systems(IDSs), firewalls(FWs), anti-viruses(A/Vs), etc., focus on low level attack or anomalies, and raise alerts independently. And IoT services have different types of security frameworks. As a result, it is difficult for human security manager or attack response systems to understand the alerts and take appropriate actions.

In this paper describes the analysis of security methods in the area of IoT and describes a mechanism that analyzes logs generated by IoT devices attacks. Data collected from the lightweight application is sent to the network component for further analysis. The collaborative component is used for collecting data in the distributed network and indicates the possible attacks. Also this paper suggests enterprise security management including IoT services which are based on distributed environments, and presents a practical technique to address this issue, and introduces Event Correlation Model(ECM) which is a simple free text causal language. We show how the concept of class in object-oriented methodology is used to provide scalability to our approach. Graph and coding theories are used for correlation.

[Keywords] ***Safety of IoT Services, Network Security Management, Enterprise Security Management, Log Analysis, Event Correlation***

1. Introduction

The Internet of Things vision is to successfully emerge, for connecting everyday existing objects and embedding intelligence into our environment. The automatic exchange of information between two systems or two devices without any manual input is the main objective of the IoT and it gives a new dimension to the world of information and communication. IoT have security threats, while exchanging information. Different types of security frameworks are used in IoT.

It's been just over a year since the world witnessed some of the world's top online Web sites being taken down for much of the day by Mirai, a zombie malware strain that enslaved IoT devices such as wireless routers, security cameras and digital video recorders for use in large-scale online attacks. A botnet, which is adding new bots every day, has already infected one million businesses during the past years in Republic of Korea and could easily eclipse the size and devastation caused by Mirai, Reaper and IoTroop[1]. The amount

of logs generated by IoT gateway is too enormous to extract anomalies. So we need to correlate logs.

The definition of correlation is "A causal, complementary, parallel, or reciprocal relationship, especially a structural, functional, or qualitative correspondence between two comparable entities"[2]. The Security event correlation can be defined loosely as the process of making sense of a very large numbers of security events, where making sense entails throwing some of them away, observing cause-and-effect relation between others, inferring an alarm from the culprit event in a misbehaving enterprise[3].

There are several approaches to the event correlation task: rule-based reasoning(RBR), model-based reasoning(MBR), state transition graphs(STG), and so on[3].

A common approach to the event correlation task is to represent knowledge and expertise in a rule-based reasoning system[4]. An RBR system consists of three basic parts including a working memory, a rule base, a reasoning algorithm[3]. The working memory consists of facts. The rule base represents knowledge about what other facts to infer, or what actions to take, given the particular facts in working memory. The reasoning algorithm is the mechanism that actually makes the inference. Using an RBR system to develop an event correlator that covers the entire domain of the enterprise is not a good idea. The enterprise is large, dynamic, and generally hard to understand.

An MBR system represents each component in the enterprise as a model[4]. A model is either a representation of a physical entity(e.g., a firewall, router, anti-virus system, intrusion detection system), or a logical entity(e.g., network session, suspicious process). And a model represents a physical entity is in direct communication with the entity(e.g. via SNMP, Syslog). A description of a model includes three categories of information: attributes, relations to other models, and behaviors. Event correlation is a result of collaboration among models(i.e. a result of the collective behaviors of all models).

The key concepts in the STG approach are a token, a state, an arc, a movement of a token from one state to another state via an arc, and an action that is triggered when a token enters a state[5][6]. The measure of correlation is dependent on the knowledge which represents the attack scenarios.

2. Event Model on IoT Services

2.1. Event normalization

There are multiple pieces of information that can be correlated such as the accepted packet on the perimeter router, the accepted packet on the firewall, the IDS alert that detected a web exploit headed for the webserver, all coming from the same source IP address. Along with the results of the integrity check, the collection of these alarms and status make it easier to confirm and reinforce the determination that the web server was indeed compromised and further action is necessary. In analysis it is ideal to access all the logs from the entire enterprise from a single console, and have them stored in one common database.

A relational database is the most logical central storage facility because it supports querying and reporting. Without a correlation engine, an analyst would first need to get the logs from all these devices, normalize them, and insert them into the database in a common format. In order to have real correlation an absolute prerequisite is normalization[7].

A typical enterprise environment consists of many different types of network devices ranging from border routers, IoT devices, to firewalls, to authentication servers, along with an even wider range of application servers like webservers, email servers, and always-critical database servers. These devices generate logs that are critical to an analyst who is responsible for the security of the site[8]. It is seldom if ever the case that two manufactures will use the same logging mechanism or format their logs identically. For example, a Cisco PIX will not report an accepted packet in the same way as a Check Point firewall or even the same as a Cisco

Router. The fact that the formats are all different makes it virtually impossible to store the log data in a common location such as a database without normalizing the events first.

The <Figure 1> shows example logs from different IoT devices are all reporting on the exact same packet traveling across the network. These logs represent a remote printer buffer overflow that connects to Apache servers over port 80 used for administrating console.

Figure 1. Original IoT device logs.

Check Point:
"14" "30 Jan 2018" "12:10:29" "eth-s1p4c0" "ip.of.firewall" "log" "accept" "www-http" "155.23*.***.***" "10.10.10.10" "tcp" "4" "1355" ""
"" "" "" "" "" "" "" "" "" "firewall" " len 68"

Cisco Router:
Jan 30 12:10:27: %SEC-6-IPACCESSLOGP: list 102 permitted tcp 155.23*.***.***(1355) -> 10.10.10.10(80), 1 packet

Cisco PIX:
Jan 30 2017 12:10:28: %PIX-6-302001: Built inbound TCP connection 125891 for faddr 155.23*.***.***/1355 gaddr 10.10.10.10/80 laddr 10.0.111.22/80

Snort:
[**] [1:971:1] WEB-IIS ISAPI .printer access [Classification: Attempted Information Leak] [Priority: 3]
01/30-12:10:29.100000 155.23*.***.***:1355 -> 10.10.10.10:80
TCP TTL:63 TOS:0x0 ID:5752 IpLen:20 DgmLen:1234 DF
AP Seq: 0xB13810DC Ack: 0xC5D2E066 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 493412860 0
[Xref => http://cve.mitre.org/cgi- bin/cvename. cgi?name=CAN-2018-0241]
[Xref => http://www.whitehats.com/info/IDS533

All these formats are different and would be practically useless to store in a database without normalizing them first. Looking at the Check Point record it contains the following fields: event id, date, time, firewall interface, IP address of the firewall interface, logging facility, action, service, source IP, target IP, protocol, source port, some Check Point specific fields and then the size of the datagram. This is the most obscure format and it is especially hard to read with all the empty fields that are represented by double quotes.

So that these events possibly be productively stored in a database, It must first be decided which fields are interesting and develop a schema to accommodate the different fields that are populated by these devices. Choosing the fields must be content driven not based on semantic differences between what Check Point may call target address and what Cisco calls destination address. To accomplish this normalization, a parser must be coded to pull out those values from the event and populate the corresponding fields in the database. The events shown on <Figure 1> can be normalized as follow fields. [Date]:[Time]:[Event_Name]:[Src_IP]:[Dst_IP]:[Target_Port]:[Device_Type]:[ETC]

This would be ideal for an analyst investigating an incident. With the data organized like this one could pull all records containing a value that is of interest or sort by any field that may be relevant. The problem is that entering this data into a spreadsheet manually is relatively easy in low volume but to get a program to do it is much more difficult. For instance the Check Point firewall reports target port as www-http not 80 like most devices. Therefore there must be a lookup mechanism to ensure that www-http gets translated into port 80 otherwise this value would be useless during correlation.

2.2. Event consequence

Let's define the set of all events generated by node i as Σ_i. The total set of all the Σ_i in a system with n nodes is $\Sigma_T = \Sigma_1 \cup \Sigma_2 \cup \ldots \Sigma_n$. It is clear that the system's alphabet will increase linearly as the number of nodes in the system increases.

Security systems can be characterized as a collection of objects organized in a way to provide specific services to the end user[9]. An object can be defined as a distinct entity in the system. It can be a hardware entity, a software process, a communication link, etc. A node consists of one or more objects. A system consists of a finite set of node classes(e.g., IDS, FW, etc.). For each of these classes, there exist one or more instances of such a class.

3. Event Correlation

3.1. Event correlation model

We define a pattern as a concatenation of events, e.g. p1 = $e_1e_2e_3$ without regard to the arrival sequence. Therefore, $e_1e_2e_3$, $e_3e_2e_1$, $e_2e_1e_3$, etc., are the same pattern. Furthermore, the multiplicity of an event within a pattern is equivalent to a single occurrence. Σ^k denote the set of words with length k. Thus, Σ^k = {w| w is a word over Σ and (|w| = k)}. For example, if Σ = {e_1, e_2, e_3}, Σ^2 = {e_1e_2, e_1e_3, e_2e_3}. Let denote that $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots = \cup \Sigma^k$. Note that Σ^+ is the set of words that might be constructed from one or more events of Σ, and is the largest set of possible words we might observe from an entity being modeled. Let L(C) be the language of a class, e.g., an object. A language is a set of words that can be constructed from the alphabet. Therefore, L(C) is a subset of Σ^+.

An event pattern may consist of tens or hundreds of events. These events are the contribution of many objects throughout the system. The number of events generated by each object and the number of objects involved in a pattern are dependent on the type of the pattern and the size and complexity of the system. Given this framework, the correlation process is divided into two phases: local correlation, which is an object level correlation, and global correlation, which combines the outcomes of the local correlation process in a way to determine a higher-level picture of the system behavior.

This model is appropriate for the problem at hand. What we are saying is that objects generate events in two situations: (1)changes in the state of the object itself; and (2)the object's reaction to external changes. In both cases, an object's behavior is based on its own limited view of the system. The global correlation process collects and assembles these object's views to determine the system view.

3.2. Correlation matrix

As events arrive at the Enterprise Security Manager(ESM), or are read from the event log offline, we use the topology information part of the event to identify the source of the event. The topology information provides us with the node instance and the object name of the source of the event as well as the node's hierarchy in the system. The event will be stored in the object's queue in a FIFO fashion. If this is the first event generated by the object, a new queue is created and the event is placed first in the queue. The event is assigned a number based on the object's alphabet and is labeled as an uncorrelated event. This means that the event has not been identified as part of a pattern yet. The queue length is a system dependent parameter. We have used a length of twenty based on our system's empirical evidence. Using this technique, old events will be pushed out of the queue as more events arrive at the queue[10].

With the arrival of every new event in an object's queue, uncorrelated events are used to form a vector called a "running vector". A similarity measure is applied to the running vector and every vector(pattern) in the object's class template. The vector that is most similar to the running vector is declared as its match. All events that contributed to the pattern are labeled as "correlated" and can be removed from the queue. A local correlation result, i.e., a composite event, will be generated as a result of the match. If however, no match is found, no action is taken.

Defining bipartite graphs in local correlations start with the definition of event-pattern pairs. This knowledge is stored in text files, which are then read into the system during initialization[11].

In local correlation where an object class within a node class is the focus, each event-pattern pair consists of the three fields:

•LCEC: Local Composite Event Code which is the result of a local correlation;

•PEC: Primitive Event(s) Code is a set of primitive events represented numerically. This is a subset of the object's alphabet;

•LCED: Local Composite Event Description. It is a text description of the local correlation result. It is used by the human operator to describe the correlation result if needed.

To illustrate the above, we use <Figure 2> as an example. First, assign a unique LCEC value for each pattern as follows: $p_1 = 1$, $p_2 = 2$, and $p_3 = 3$. Second, assign a unique PEC value for each event as follows: $e_1 = 1$, $e_2 = 2$, $e_3 = 3$, $e_4 = 4$, $e_5 = 5$, $e_6 = 6$, $e_7 = 7$, and $e_8 = 8$. Third, assign a unique LCED field to each pattern. This can be a simple description of the pattern, which can be used by the operator. Finally, each field is separated by a delimiter(e.g., vertical bar |).

1 |3 5 6 7| Firewall blocked a session

2 |1 2 4 5| Firewall passed a session

3 |3 7 8| Firewall detected an intrusion

Figure 2. Correlation matrix.

	P_1	P_2	P_3
e_1	0	1	0
e_2	0	1	0
e_3	1	0	1
e_4	0	1	0
e_5	1	1	0
e_6	1	0	0
e_7	1	0	1
e_8	0	0	1

3.3. Global and temporal correlation

We define a correlation zone(CZ), as a logical entity that represents a collection of objects. A CZ is modeled as a bipartite graph where one set of nodes represents the correlation patterns and the other represents the composite events. A CZ subscribes to the composite events, which are received from objects within the system. The bipartite graph must consist of one or more pattern nodes and the composite events must be from two or more objects within the system. The objects can be located within the same node or different nodes or a combination of the two. A CZ is represented by a queue, which receives composite events from the objects it subscribes to. Like the local correlation decoder, the CZ uses the same approach to identify arriving patterns. The CZ is identical to the concept of an object and its approach to event correlation. The only difference is that a CZ does not generate events and therefore it does not have its own alphabet. Instead, the user defines its alphabet and its language according to known causal relationships. The global correlation process is recursive. This means that a CZ class can be made of smaller set of CZs. This concept fits well given the hierarchical nature of security systems for example network based intrusion detection system, internal/outside firewalls.

While the events in local correlation are the object's alphabet, the global correlation's alphabet is made up of the composite events subscribed to by the CZ based on casual relationships. Therefore, a CZ can be represented by any arbitrary bipartite graph that consists of patterns and composite events. Our objective in this step of the correlation process, however, is to address the scalability issue while finding a solution that works. This can be achieved by using the class concept we explained in the local correlation step.

Defining bipartite graphs in global correlations is similar to that of local correlation. It starts with the definition of event-pattern pairs. This knowledge is stored in text files, which are then read into the system during initialization.

The temporal relationships between events are as important as causal relationships. We plan to utilize the temporal relationships in two different areas, (1)reducing false positives and therefore improve the accuracy of the correlation results; and (2)solving the problem of variable length codes.

Local and global entities are represented by queues where events arrive for correlation. Spurious and delayed events among others

are the cause of false positives. The events stay in the queue until they are flushed out by the FIFO effect. This FIFO effect is helpful but not enough to solve the false positive problem completely. Next, we introduce ways to incorporate the temporal relationships between events to improve the false positive problem.

There are three levels at which a temporal relationship can operate: (1)event level; (2)pattern level; and (3)queue level. We start with the event level temporal relationships. Here, events are considered as independent entities. Only the aging concept will be considered at this level. Three different types of aging techniques can be employed: (1)Threshold: an event will be active until a predefined value is reached when it is considered obsolete and should be removed from the queue. This behavior can be modeled as a step function; (2)Mathematical functions: an event is assigned a weight based on its age in the queue. Upon arrival in the queue, an event has a weight of 1. This weight decreases as time goes by until the weight becomes zero and the event is removed from the queue. The weight value follows the behavior of some function. Finding the best model for event aging is not simple and is system dependent. Simple functions such as linear or truncated exponential can be easily implemented and might be a good starting point.

If we treat all the different events within a queue equally, then the same techniques described for an event level can be applied to a queue. As an example, consider the threshold level technique. Here, we assign a threshold value for a queue where any event older than that value will be considered obsolete and must be removed. The queue level temporal relationships can be seen as a simplified version of event level temporal relationships.

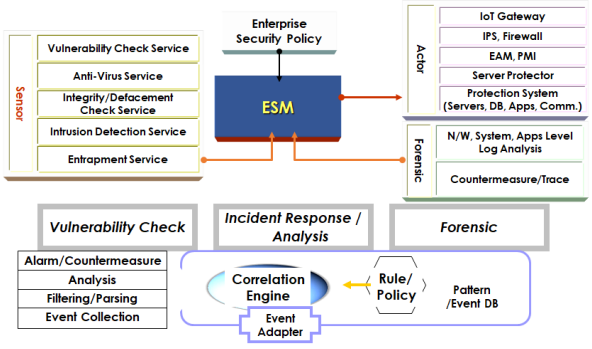
The last technique uses the pattern level temporal relationships. Here, every set of events that make up a pattern is treated differently. This is true of the same event belonging to different patterns.

There is a need for a temporal language to allow for the specification of a variety of temporal relationships between a set of events. In detecting communities[5], an overview of existing temporal logic programming languages was given. While in machine learning[3], provided composite event specification language that can be used to express complex temporal relationships between events. For now, we will define the following single operator: All events comprising a pattern must arrive within a specified period of time. This temporal constrained operator is useful in eliminating old and unrelated events.

4. Implementations and Evaluations

4.1. Implementations

Figure 3. Architecture of enterprise security management system.

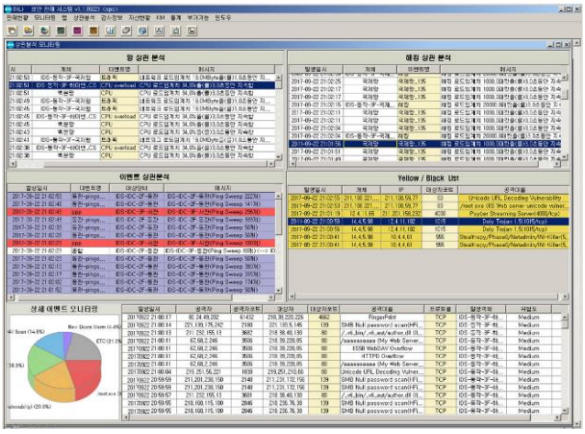


We have implemented an off-line intrusion alert correlator using the method discussed in Section 3. <Figure 3> shows the architecture. It consists of an event manager, an event log, event correlator, and a security manager. All these interact with a DBMS, which provides persistent storage for the intermediate data as well as the correlated alerts. The program was written in Java, with JDBC to access the database. The knowledge base contains the necessary information about hyper-alert types as well as implication relationships between predicates. In our current implementation, the hyper-alert types and the relationship between patterns are specified in an ECL. When the alert correlator is initialized, it reads the ECL file, and then converts and stores the information in the knowledge base. Our current implementation assumes the alerts reported by IDSs are stored in the database. Using the information

in the knowledge base, the alert event manager generates hyper-alerts as well as auxiliary data from the original alerts. The correlation engine then performs the actual correlation task using the hyper-alerts and the auxiliary data.

<Figure 4> shows the correlated logs. Pie graphs depicted on left side are clustered for each networks(domains) and log table depicted on right side are listed by original logs.

Figure 4. Examples of correlated logs.



4.2. Evaluations

To consolidate precision and recall into one measure, we can use the harmonic mean of precision and recall[12]. Purity can be easily manipulated to generate high values. Consider when nodes represent singleton attack(of size 1) or when we have very large pure combination attacks(ground truth = majority label). In both cases, purity does not make sense because it generates high values.

A more precise measure to solve problems associated with purity is the normalized mutual information(NMI) measure, which originates in information theory[13][14]. Mutual information(MI) describes the amount of information that two random variables share. In other words, by knowing one of the variables, MI measures the amount of uncertainty reduced regarding the other variable. Finally we can evaluate the ratio of false positive and false negative NMI measures as follows,

$$NMI = \frac{\sum_{h\in H}\sum_{l\in L} n_{h,l}\log\frac{n\cdot n_{h,l}}{n_h n_l}}{\sqrt{(\sum_{h\in H} n_h \log\frac{n_h}{n})(\sum_{l\in L} n_l \log\frac{n_l}{n})}}.$$

where L and H are labels and found detections; n_h and n_l are the number of data points in global correlation h and with label l. l is the number of nodes in community h and with label l; and n is the number of nodes.

An NMI value close to one indicates high similarity between communities found and labels. A value close to zero indicates a long distance between them. As a results we observed that NMI integrated into one.

5. Conclusions

As large organizations move to manage enterprise security as a critical business process, correlation of events from disparate security devices is absolutely essential for success. Without the efficiency and effectiveness that correlation introduces into the security process workflow, organizations will never catch up to the level of threat that they current face. A foundation technology for correlation is data normalization. A key attribute that should be focused on when evaluating enterprise management solutions is the scope and deployment of the normalization mechanisms. Effective correlation will only be available if the normalization process can capture and organize 100% of the relevant security information for every device and source in the network.

So, we have introduced a framework for event correlation in security systems. We showed how the concept of class in object-oriented methodology is used to provide scalability to our approach. Graph and coding theories are used for correlation. Finally, temporal reasoning was investigated for this framework.

We introduced model, ECM with a simple free text causal language. We introduced a new decoder for pattern identification. Entities and their queues are created only if they generate events and are deleted once their queues are empty. This helps reduce memory requirements and decrease system complexity. Temporal relationships have been shown to be effective alongside the causal relationships between events.

6. References

6.1. Journal articles

[3] Buczak AL & Guven E. Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153-1176 (2015).

[5] Kim P & Kim SW. Technique for Detecting Interaction-based Communities in Dynamic Networks. *Journal of KIISE*, 22(8), 357-362 (2016).

[6] Ryu SH & Kim SW. Development of an Integrated IoT System for Searching Dependable Device based on User Property. *Journal of Korea Multimedia Society*, 20(5), 791-799 (2017).

[8] Koh KS & Lee SH & Ahn SW. Study on the Direction of Security Control of IoT Environment. *Journal of Korea Convergence Security*, 15(5), 53-59 (2015).

[10] Nitha KP. Study on the Direction of Security Control of IoT Environment. *International Journal on Recent and Innovation Trends in Computing and Communication*, 3(4), 1970-1974 (2015).

[12] Kim P & Kim SW. Detecting Community Structure in Complex Networks Using an Interaction Optimization Process. *International Journal of Physica A*, 46(5), 525-542 (2017).

[13] Park BS & Lee TH & Kwak J. Blockchain Based IoT Device Authentication Scheme. *Journal of the Korea Institute of Information Security & Cryptology*, 27(2), 343-351 (2017).

6.2. Thesis degree

[4] Jang HJ. Real-time Intruder Tracing in the Large Distributed Network. Kyungpook National University, Doctoral Thesis (2002).

6.3. Books

[7] John Allison. The 6 Categories of Critical Log. SANS Technology Institute (2013).

[14] Jain AK & Dubes RC. Algorithms for Clustering Data. Prentice Hall (1988).

6.4. Conference proceedings

[2] Dain O & Cunningham R. Fusing a Heterogeneous Alert Stream into Scenarios. ACM Workshop on Data Mining for Security Applications (2001).

[11] Chand N. Comparative Analysis of SVM and its Stacking with other Classification Algorithm for Intrusion Detection. Advances in Computing, Communication & Automation (2016).

6.5. Additional references

[1] https://research.checkpoint.com (2018).

[9] https://www.zingbox.com (2018).

Author

Park Bo-seok / Kyungpook National University Adjunct Professor

B.A. Kyungpook National University

M.A. Kyungpook National University

Ph.D. Kyungpook National University

Research field

- Master Slave Security Management based on Mobile Code, Kyungpook National University, Master's Thesis (1999).
- Enterprise Security Management for IoT Services, Symposium on Ubiquitous and Web Information Technology, (2017).

Major career

- 2001~2004. Netsecue Tech, Chief Manager of R&D Center.
- 2004~present. Kyungpook National University, Adjunct Professor.