Event Prioritization and Correlation based on Pattern Mining Techniques

Mona Lange and Ralf Möller
Institute of Information Systems
Universität zu Lübeck
Lübeck, D-23562 Germany
Email: {moeller, lange} @ifis.uni-luebeck.de

Gregor Lang
Friedrich-Alexander Universität
Erlangen-Nürnberg
Erlangen, D-91058 Germany
Email: gregor.lang@fau.de

Felix Kuhr
Technische Universität
Hamburg-Harburg
Hamburg, D-21071 Germany
Email: felix.kuhr@tu-harburg.de

***Abstract*—With the growing deployment of host and network intrusion detection systems in increasingly large and complex communication networks, managing low-level events from these systems becomes critically important. A network has multiple tasks, which consist of multiple network services aiding the execution of a task. An emerging track of security research has focused on event prioritization and correlation to rank the criticality of events and reduce the number of low-level events. To prioritize and correlate events, the ongoing tasks in an enterprise network are identified, as the goal of network operators is to protect ongoing tasks when a security breach occurs. The prioritization of an event depends on the criticality of an ongoing task that is potentially threatened by the event. Additionally, in order to support network operators, we correlate all events that target the same task. A particular task may depend on multiple network services and involve multiple network devices.So, if one network service becomes unavailable, other network services will be affected over time since they are dependent on one another. Unfortunately, dependency details are often not documented and are difficult to discover by relying on human expert knowledge. In order to solve this problem, a network dependency analysis based on network traffic is conducted. We rely on pattern mining techniques to discover tasks in a monitored enterprise network. A formal description of the identified tasks is provided and events are prioritized and correlated based on this model. The pattern mining based network dependency analysis algorithm is evaluated based on a real-world network and three networks that where created with a network simulator.**

I. Introduction

Malicious actors exploiting cyberspace have been identified by the United States intelligence community as the top national security threat [1]. Similarly, Dell's annual threat report states a 100% increase in SCADA attacks [2]. This report is based on an analysis of data gathered by Dell's global response intelligence defense network that consists of millions of security sensors in more than 200 countries. Due to the rising of cybercrime, there has been a great demand for information security systems, e.g., intrusion detection systems.

In the following we understand an Intrusion Detection System (IDS) as a system, which has been developed to monitor an enterprise network for malicious events. An IDS can either be a software or hardware-based system and resulting information is referred to as events or alerts. Generally, these systems are positioned to detect cyber attacks originating from outside a company's own network. IDSs are placed in positions where high volumes of network traffic occur as they can only report on malicious activity that they are able to observe. To monitor network traffic these systems generally are based on a low-level network traffic model. Thus, low-level events events are created due to the low-level of data that IDSs analyze. Knowledge of both networking technology and infiltration techniques is required to correctly interpret high amounts of high paced low-level events. Furthermore, IBM's 2015 cyber security intelligence index reveals that approximately half of all cyber attacks originate from within a company's own network [3]. Cyber attacks means in effect that at least one network service was exploited.

We depend on network services in many aspects of our daily lives (e.g., email, smart homes, medical services, electricity supply, water supply). Network services operate on distributed sets of clients and servers and rely on supporting network services, such as Kerberos, Domain Name System (DNS), and Active Directory. Hence, network services need to interact with each other in order to function correctly. Since, engineers use the divide-and-conquer approach to implement a new task, they are able to reuse network services and do not need to re-implement complex customized ones. Unfortunately, implementation and dependency details are often not documented, and are difficult to discover by referring to human expert knowledge, for obvious reasons.

Currently, conventional network security approaches focus on perimeter protection instead of identifying the most business critical assets and protect those. Stuxnet [4] or Flame [**?**] have taught us that in order to protected critical infrastructures against these advanced persistent threats, perimeter protection simply is not enough. To underpin this statement we refer to the director Sean McGurk of the National Cybersecurity and Communications Integration Center (NCCIC) at the Department of Homeland Security [5]:

"In our experience in conducting hundreds of vulnerability assessments in the private sector, in no case have we ever found the operations network, the SCADA system or energy management system separated from the enterprise network. On average, we see 11 direct connections between those networks. In some extreme cases, we have identified up to 250 connections between the actual producing network and the enterprise network."

This work is funded by the Seventh Framework Programme (FP7) of the European Commission as part of the project PANOPTESEC (http://www.panoptesec.eu, GA 610416)

IDS events can only be prioritized if the implications of a network service or network device failing are understood. For example an IDS event that is linked to a critical network service needs to be prioritized over an event that is linked to a network service of little importance. To monitor the status of a BP based on all incoming IDS events, all events that target involved network services need to be correlated. In the content of this work we conduct a network dependency analysis based on network traffic to identify activity patterns that we denote as tasks. We formally describe our notion of a task, before prioritizing and correlating events based on task data structures. An evaluation of the identified tasks is based on a real-world network and three networks that where created with the network simulator ns-3 [6].

II. Related Work

Most approaches correlate intrusion events and extract attack scenarios to predict the next attacker action [7], [8], [9], [10], [11]. However, different intrusion event approaches have different focuses. Several researches have concentrated on how to reduce the amount of events as well as decreasing the false positive rate [12], [13], [14]. Others cluster similar events [10], [11] to discover high-level attack scenarios or represent and reason with operator preferences regarding the events they analyze [15]. Others focus on solving the problem of aggregating events into multi-step attacks as a data mining problem [8], [9]. Regardless of their slightly different research focuses, event correlation approaches can be separated into three different categories: Similarity-based, statistics-based and knowledge-based approaches. Similarity-based event correlation approaches rely on features to compare the similarity of two events or a single event with a cluster of events. Statistics-based security focused approaches [10], [12], [14], [11] focus on clustering similar events. Knowledge-based event correlation methods rely on formal knowledge representations to provide predicate-based event correlation. A common drawback of these approaches is that they rely on a lot of human input[10], [12], [16]. As the complexity of computer networks is growing faster than the ability to understand them, the quality of the human input is questionable. Statistics-based event correlation methods depend on underlying attribute distributions (such as Gaussian distribution) of deviations from what is expected. Valdes et al. [7] use statistical methods on multiple event metrics to establish a lower level of correlation before discovering attack step correlations. Farhadi et al. [8] studied how to correlate event patterns using Hidden Markov Models and focused on machine learning based attack plan recognition to correlate and predict events. Unfortunately, most of the presented approaches implicitly rely on a threat agent model [7], [8], [9], [10], [11]. However, threat agents range from individual hackers, to organized hacker groups, organized crime, industrial espionage, disgruntled employees, terrorist groups to nation state attacks. Due to the lack of information on these different threat agents, predicting their next possible action and validating the proposed adversary model is very difficult. Rather than focusing on how events affect ongoing tasks in the monitored network, the above mentioned approaches focus on aggregating similar events or extracting attack scenarios to predict the next attacker action. Evaluating the quality of a threat model is difficult as little information is available about different threat agents. However, network operators know the purpose of their network and monitored networks provide a multitude of information available for data mining problems. Among the available information is network traffic, routing tables, Network Address Translation (NAT) rules and events. This information can be used to exploit pattern mining approaches to discover tasks in the monitored network. The focus of the approach introduced in the context of this work is to monitor how events affect ongoing tasks in the monitored network. The reasoning behind this approach is that events affecting business-critical tasks have more impact in the eyes of a network operator than others. Also, network operators can judge whether they agree with the identified tasks. Due to little information being available about threat agents, we argue that it is very difficult to conduct a similar evaluation with attacker model based approaches. In contradiction to knowledge-based approaches we focus on minimizing the required human input. The approach introduced in the following is able to (i) automatically discover activities in the network and (ii) correlate incoming events based on targeted tasks.

III. Network Model

Modeling an IT network requires a basic understanding of the Open Systems Interconnection (OSI) model. For understanding network connectivity, the following layers of the OSI model are of particular interest: data link layer, network layer, transport layer and application layer. We define a network device as a physical device on the network. The data link layer physically links network devices using MAC addresses to identify devices. However, the data link layer only provides point-to-point connectivity. For enabling network connectivity beyond a point-to-point communication, a network layer protocol such as the Internet Protocol (IP) is required. IP addresses are used to identify source and destination of an end-to-end connection. In other words, MAC addresses allow a point-to-point connection, while IP addresses provide an end-to-end connection. Therefore, switches and routers are used to forward packets, i.e. they act as intermediate hosts.

Definition 1 (Device): Let MAC and IP be non empty sets of MAC and IP addresses, respectively ($MAC \cap IP = \emptyset$), then $D \subseteq \mathcal{P}(MAC) \setminus \{\emptyset\} \times \mathcal{P}(IP)$ is the set of (network) devices.

Definition 1 allows a device to be assigned multiple MAC addresses and IP addresses. Being able to assign multiple MAC addresses to a network device is needed as routers and switches supply multiple point-to-point endpoints. However, switches do not necessarily need to have IP addresses as they work on the data link layer. From this it follows that they are not visible on the network layer.

Definition 2 (Network): A (communication) network N is tuple (D, E) consisting of a finite set of devices D and a finite set of edges E, along with $E \subseteq \{\{d_i, d_j\} | d_i, d_j \in D, d_i \neq d_j\}$.
The end-to-end connection between network devices is the basis of this network model and can automatically be derived based on routing tables or router configuration. Due to the derivation of the network formally introduced in Definition 2, data link layer devices such as switches are opaque. In Example 1 we introduce a running example based on a power grid's communication network and illustrate the corresponding network in Figure 2.

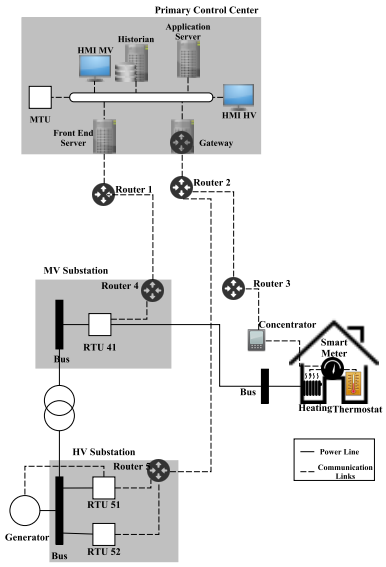

Fig. 1. Running example of a communication network in a power grid.

Example 1: The schematic diagram in Figure 1 illustrates an excerpt of a power grid network. A primary control center hosts a Human Machine Interface for Medium Voltage Substation (HMI MV) and an HMI for High Voltage Substation (HMI HV). A so called Historian stores historical event and measurement data, and an Application Server analyzes the status of the monitored power grid. Remote Terminal Units (RTUs) collect data from sensors and control actuators situated at remote sites and send data to a Master Terminal Unit (MTU) through the network. The remote sites consist of High Voltage (HV) and Medium Voltage (MV) Substations. Buses are connection points and within the running example they interconnect a generator that is located within the HV Substation via a tranformator and a MV Substation to a "smart home". The "smart home" incorporates a smart meter, intelligent heating and a thermostat. Figure 2 shows the corresponding network of the schematic diagram introduced in Figure 1. The network is represented as an undirected graph as introduced in Definition 2. Network devices are represented as nodes and communication links correspond to edges.

Generally, devices that are end points in your network can host network services. An application can rely on multiple

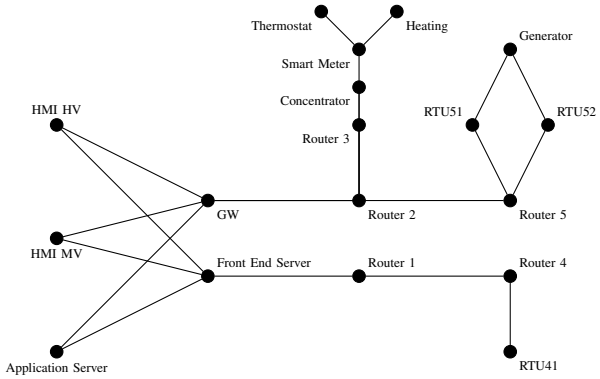

Fig. 2. A small excerpt of the network described in Example 1.

network services to achieve a certain goal. However, a single network service can unambiguously be assigned to an application given that it is not running on a dynamically assigned port. In this paper, we do not make a formal distinction between network services and applications, and we use the term network services. These network services are located on the application layer and are able to communicate end-to-end with each other through a transport layer protocol.

Definition 3 (Network Service): To derive all network services hosted by a device d_j, we define the relationship $HOSTS(d_j)$, which returns all network services hosted by d_j. In order to derive the device a service s is hosted by, we write $HOSTS^{-1}(s)$, and associate service s_i with device d_j by writing s_i^j. Given a service $s_i^j \in S$, the corresponding device d_j can be derived by $d_j = HOSTS^{-1}(s_i^j)$.
Below we show how to identify network dependencies based on the presented network model.

IV. Network Dependency Analysis

Tasks consist of multiple network services that depend on each other to fulfill a common goal. Network services can be directly or indirectly dependent on each other. In order to automatically derive tasks, we need to identify direct and indirect dependencies between network services. Thus, we design a network dependency analysis approach to identify direct and potential indirect dependencies between network services in data networks. The aim of network dependency analysis is to find typical sequences of operation in data networks. A network captures devices that are communication endpoints and additional intermediate devices, over which endpoints communicate. Network devices that are endpoints can host network services. Based on network traffic analysis we will detect network services and determine how they communicate in an end-to-end manner with each other. We conduct a network dependency analysis based on packet headers (e.g. IP, UDP and TCP) and timing data in network traffic.

A. Direct Dependency

Data is exchanged in the form of packets between the network service s_i^j and s_k^l. We term these end-to-end interactions

between network services as direct dependencies. The direct dependency between network services s_i^j and s_k^l is denoted as

$$SDEP = \{(s_i^j, s_k^l) \mid s_i^j \text{ sends a packet to } s_k^l \text{ in the period under consideration}\} \quad (1)$$

Based on Definition 3 a direct dependency between network services s_i^j and s_k^l leads to a direct dependency between the respective hosts d_j and d_l. This can be written as

$$DDEP = \{\Big(HOSTS^{-1}(s_i^j), HOSTS^{-1}(s_k^l)\Big) \mid (s_i^j, s_k^l) \in SDEP\} \quad (2)$$

Definition 4 (direct dependency): Let a direct dependency between network services be denoted as SDEP and defined in Equation 1. We write $\delta(s_i^j, s_k^l)$ to denote $(s_i^j, s_k^l) \in SDEP$. Based on Definition 3, a direct dependency between network services results in a direct dependency between devices. We write $\delta(HOSTS^{-1}(s_i^j), HOSTS^{-1}(s_k^l)) = \delta(d_j, d_l)$ to state the derivation of $DDEP$ based on $SDEP$. Over time, network services that are directly dependent exchange packets. A direct dependency $\delta(s_i^j, s_k^l)$ means that a network device d_j hosts a service s_i^j that sends data to another service s_k^l, which is located on network device d_l. This means s_k^l depends on service s_i^j and a delay, disruption, degradation or failure in service s_i^j will lead to delay, disruption, degradation or failure of service s_k^l. Packets include timestamps that denote when the package was exchanged between the two hosts d_j and d_l via the network service s_i^j and s_k^l. A continuous timeline is monitored for finite period of time, which is sliced into k timesteps of a predefined size. Predefined time steps can for example be seconds, milliseconds or nanoseconds. Based on this model we can see when data was exchanged and are able to count the number of packets exchanged within a time step. The packages exchanged between directly dependent network services $\delta(s_i^j, s_k^l)$ over time constitute a vector from $\mathbb{N}^k$. We use $\delta(s_i^j, s_k^l)$ in a predicative way as well as for denoting time series, hence we write $\delta(s_i^j, s_k^l) \in \mathbb{N}^k$.

B. Indirect Dependency

Beyond direct dependencies there exist more complex dependencies in an IT network. Estimating complex dependencies with $SDEP^+$ would derive complex dependencies based on device connectivity. This would overestimate the number of indirect dependencies and forgo the derivation of a deeper semantic understanding of complex dependencies in a network. To derive a deeper semantic understanding of complex dependencies in a network, we analyze communication patterns and derive indirect dependencies based on "similar" communication patterns. Given a direct dependency $\delta(s_i^j, s_k^l)$, all network services hosted by $HOSTS^{-1}(s_k^l) = d_l$ are candidates for an indirect dependency. An indirect dependency implies that the data received by s_k^l is processed on device d_l. Then, data is sent to another network service s_m^l. Due to the processing of data on device d_l, the request might be sent to s_m^l τ_{delay} time steps later. Hence, the communication patterns of both direct dependencies would be similar, although shifted by τ_{delay} time steps.

Example 2: Consider the schematic diagram of a network in an electrical power grid presented in Figure 2 with an operator using HMI MV. The operator intends to send a request to a substation's Remote Terminal Unit (RTU). Hence, an HMI MV sends this request to the Front End Server (FES). The FES transfers the request to a router, which transmits the request to the corresponding router of the substation. The substation's router then sends the request to the RTU. Based on the assumption that communication is based on TCP/UDP, the involved network devices send and receive requests via hosted network services. Problems at any of these involved network services may lead to a failed request for the operator, namely the request not being sent. Operators need to understand the respective indirect dependencies in order to locate the reason why the task "send request to RTU" failed. Also, indirect dependencies help an operator understand the impact of a failed task. If the task "send request to RTU" involves a highly critical device, then a failure of the task could be critical, too.

In order to understand the general case, we consider two direct dependencies $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$ that are hosted by network devices d_j, d_l and d_o. In some sense, the network services are adjoined.

$$ISDEP = SDEP \bowtie_{HOSTS^{-1}(2)=HOSTS^{-1}(1)} SDEP \quad (3)$$

If a network service s_i^j sends a request to s_k^l, which processes the data on device d_l, and an application on d_l sends a request via network service s_m^l to s_n^o, then we refer to the two direct dependencies $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$ as indirect dependency $(s_i^j, s_k^l, s_m^l, s_n^o)$. For brevity, we denote this indirect dependency as $\iota(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))$. Note that network service s_k^l and s_m^l are both hosted by device d_l, i.e. $HOSTS^{-1}(s_k^l) = HOSTS^{-1}(s_m^l)$. Based on Definition 3 an indirect dependency $ISDEP$ between network services leads to an indirect dependency $IDDEP$ between the involved devices. The relation $IDDEP$ is defined as

$$\text{IDDEP} = map(\lambda\,((s', s'', s''')) \,.\, \big(HOSTS^{-1}(s'), HOSTS^{-1}(s''), HOSTS^{-1}(s''')\big), \Pi_{1,2,4}\, ISDEP) \quad (4)$$

The assumption of indirect dependencies is that data being exchanged within a direct dependency $\delta(s_i^j, s_k^l)$ is also used by a direct dependency $\delta(s_m^l, s_n^o)$. Hence, the two direct dependencies are dependent on each other. Consequently, a failure, degradation or delay of $\delta(s_i^j, s_k^l)$ can lead to failure, disruption or degradation of the other direct dependency $\delta(s_m^l, s_n^o)$ and vice versa. Additionally, all direct dependencies are also described by communication patterns. Example 2 describes a scenario containing such indirect dependencies.

Normalized cross-correlation [17] according to Equation 5 is used to detect how similar both communication patterns $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$, which are both vectors of length k, are. Equation 6 is used to detect the temporal shift between the $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$. The normalized cross correlation

$\rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))$ is defined as

$$\rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))[\tau] = \frac{1}{k}\sum_{n=0}^{k} \frac{\left(\delta(s_i^j, s_k^l)[n] - \overline{\delta(s_i^j, s_k^l)}\right) \cdot \left(\delta(s_m^l, s_n^o)[n+\tau] - \overline{\delta(s_m^l, s_n^o)}\right)}{\sigma_{\delta(s_i^j, s_k^l)} \sigma_{\delta(s_m^l, s_n^o)}}, \quad (5)$$

where $\sigma_{\delta(s_i^j, s_k^l)}, \sigma_{\delta(s_m^l, s_n^o)}$ are the standard deviations and $\overline{\delta(s_i^j, s_k^l)}, \overline{\delta(s_m^l, s_n^o)}$ are the means of $\delta(s_i^j, s_k^l)$ and $\delta(s_m^l, s_n^o)$, respectively.

The point in time τ_{delay}, where both signals are best aligned can be found with

$$\tau_{delay} = \text{argmax}_t \rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))[\tau]. \quad (6)$$

If $\rho(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))[\tau_{delay}] > \theta$, we consider both communications to be correlated and therefore indirectly dependent and shifted by τ_{delay}. The delay τ_{delay} is added to every indirect dependency in $ISDEP$.

C. Clustering Dependencies

Aside from intermediate devices (e.g. routers and switches), network devices can be categorized into client and server network devices. In the following we will refer to client network devices as clients and server network devices as servers. Clients and servers are able to send requests. Servers additionally provide network services that answer these requests. Generally, the number of clients by far surpasses the number of servers. Requests are often sent through a dynamically assigned port. These ports are in the range from $2^{15} + 2^{14}$ to 2^{16} and are available for private, customized or temporary purposes.

Definition 5 (Cluster Network Service): Let S be a set of services that are hosted by device d_j. All network services communicating through a dynamically assigned port, are grouped by $s_*^j \in S$, which is called cluster network service.

D. Tasks

Indirect dependencies are elementary building blocks for deriving tasks in communication networks. We consider indirect dependencies as the smallest possible task, because according to our communication approach a similar communication pattern is detected. Hence, we conclude all network services within the involved direct dependencies are dependent on each other. An indirect dependency $\iota(\delta(s_i^j, s_k^l), \delta(s_m^l, s_n^o))$ contains the notion that a failure or delay of $\delta(s_i^j, s_k^l)$ leads to a failure or delay of $\delta(s_m^l, s_n^o)$. Tasks broaden this notion and cluster all network services that might lead to or be affected by a delay or failure. Hence, the set of tasks TS is defined as

$$TS = SCC\left((S, map(asSet, ISDEP))\right), \quad (7)$$

where SCC denotes the strongly connected components (SCC) of the hypergraph given as parameter ($asSet$ maps a tuple into a set of components). Figure 3 illustrates a possible set of tasks TS. To find the devices TD associated to a task, we use $dev : \mathbb{P}(S) \to \mathbb{P}(D)$ as

$$dev(t) = map\left(HOSTS^{-1}, t\right) \quad (8)$$

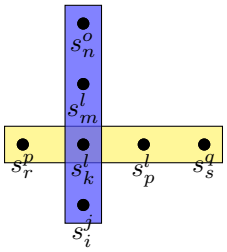

Fig. 3. An example of a set of tasks TS.

V. Event Processing

After having conducted a network dependency analysis, the focus of this section is to use previously identified network dependencies for event processing. Incoming event streams are heterogeneous data streams. Hence, events have to be normalized before prioritizing and correlating them.

A. Event Normalization

Based on syslog-ng [18], we integrate events generated by heterogeneous IDS sensors into a standardized data formats called Intrusion Detection Message Exchange Format (IDMEF) [19]. IDMEF was the first attempt to address the problem of formatting and standardizing events [20]. The IDMEF data model is a standard representation of events with a predefined set of attributes.

B. Event Prioritizing and Correlation

Considering a set of events E, every event can be linked to a set of network services according to Equation 9,

$$ES : E \to \mathbb{P}(S), \quad (9)$$

with a mapping function ES, an event set E and a network service set S. Based on the identified set of network services, a set of involved devices can be derived with the function dev.

Events that affect the same devices are correlated into a single IDMEF event. Whether two events $e_x, e_y \in E$ are correlated is derived by,

$$\begin{aligned} \forall e_x, e_y \in E : ES(e_x) \cap ES(e_y) \neq \emptyset \\ \implies \{e_x, e_y\} \in IDMEFEV \end{aligned} \quad (10)$$

Correlated events $CORREV$ are defined as the SCCs of the graph (S, $IDMEFEV$). In order to support an operator in understanding how a correlated event affects the monitored network, we map events to a set of devices.

$$\begin{aligned} &affectedDevices : E \to \mathbb{P}(D) \\ &affectedDevices(e) = \bigcup_{\substack{t \in TS, \\ \tilde{e}_{xy} \in CORREV}} \{dev(t) \mid ES(\tilde{e}_{xy}) \cap t \neq \emptyset\} \end{aligned} \quad (11)$$

An operator is then able to see all events and all devices, which might potentially by affect them. All network devices are assigned criticality values according to the following equation.

$$CRIT : D \to \{\text{low, medium, high}\} \quad (12)$$

Based on the criticality map for devices, we are able to prioritize correlated events by defining the following order relation on $CORREV$:

$$\leq = \{(e_x, e_y) \in CORREV \times CORREV \mid reduce(\max, map(CRIT, affectedDevices(e_x)), low) \leq reduce(\max, map(CRIT, affectedDevices(e_y)), low)\} \quad (13)$$

A correlated event is ranked as the highest criticality category of all affected network devices. So, a correlated event is assigned a criticality value and it can be ranked into one of the three criticality categories $\{$low, medium, high$\}$. This allows a network operator to prioritize events that might potentially affect more critical network devices.

VI. Experimental Evaluation

The purpose of the experimental evaluation is to prove the following Hypothesis 1.

Hypothesis 1: Normalized cross correlation is able to uncover true positive indirect dependencies between network services.

According to the definitions presents below we implemented an algorithm, which is implemented in C++ and we use OpenCV [21] to apply normalized cross correlation to communication patterns. We analyze the implemented algorithm based on a data set from a real-world enterprise network. In order to get a ground truth from the identified indirect dependencies are shown to a network operator. The network operator classify the indirect dependencies as true positive or false positive. Given a listing of the identified indirect dependencies, all of them where classified as true positives. However, giving an exhaustive list of all indirect dependencies in the monitored network was outside the field of knowledge of the network operator.

This is why we explored the possibility of creating a communication networks synthetically in order to have a known ground truth. The underlying structure of our synthetically created communication networks is based on three topologies introduced in the "Guide to Industrial Control System Security" from [22], which is modeled with the network simulator Ns-3 [6] and we add 10% of indirect dependencies randomly. Implementing the test network gives us the ground truth to evaluate the results of the implemented algorithm.

TABLE I
Results of the analyzed data sets.

data set	network services	number of direct dependencies	number of true positive indirect dependencies	number of false positive indirect dependencies
enterprise network	802	829	103	?
ns-3 network 1	48	144	16	1
ns-3 network 2	54	162	19	2
ns-3 network 3	47	141	15	1

The results of the analyzed data sets is shown in Table I and show few false positives.

VII. Conclusion

In this paper we proposed an event prioritization and correlation approach. In order to correlate all events potentially affecting the same task, we proposed a approach for identifying a network's underlying tasks based on network traffic. Underlying tasks are identified by comparing communication patterns between network service via normalized cross-correlation.

In this work we formally identify tasks as consisting of multiple network services that depend on each other to fulfill a common goal. Future work includes merging indirect dependencies that have multiple common network services into a single dependency. Also, we plan to investigate communicable formats for these low-level tasks.

Acknowledgment

This work has been partially supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 610416 (PANOPTESEC). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

References

[1] J. R. Clapper, "Statement for the record, worldwide threat assessment of the us intelligence community," http://www.dni.gov/index.php/newsroom/testimonies/209-congressional-testimonies-2015/1174-statement-for-the-record-worldwide-threat-assessment-of-the-u-s-ic-before-the-sasc, 2014.
[2] Dell Inc., "Dell security annual threat report," Tech. Rep., 2015.
[3] I. Corporation, "2015 cyber security intelligence index," july 2015.
[4] R. Langner and P. Pederson, "Bound to fail: Why cyber security risk cannot simply be "managed" away," *Cyber Security Series*, 2013.
[5] Subcommittee on National Security, Homeland Defense, and Foreign Operations. (2011) Cybersecurity: Assessing the immediate threat to the united states. [Online]. Available: http://www.youtube.com/watch?v=x1URPa1jG60
[6] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN, Technical Report, 2012.
[7] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *Recent advances in intrusion detection*. Springer, 2001, pp. 54–68.
[8] H. Farhadi, M. AmirHaeri, and M. Khansari, "Alert correlation and prediction using data mining and HMM," *The ISC International Journal of Information Security*, vol. 3, no. 2, 2015.
[9] M. GhasemiGol and A. Ghaemi-Bafghi, "E-correlator: an entropy-based alert correlation system," *Security and Communication Networks*, vol. 8, no. 5, pp. 822–836, 2015.
[10] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 202–215.
[11] X. Peng, Y. Zhang, S. Xiao, Z. Wu, J. Cui, L. Chen, and D. Xiao, "An alert correlation method based on improved cluster algorithm," in *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, vol. 1. IEEE, 2008, pp. 342–347.
[12] B. Morin, L. Mé, H. Debar, and M. Ducassé, "M2D2: A formal data model for IDS alert correlation," in *Recent Advances in Intrusion Detection*. Springer, 2002, pp. 115–137.
[13] T. Pietraszek, "Using adaptive alert classification to reduce false positives in intrusion detection," in *Recent Advances in Intrusion Detection*. Springer, 2004, pp. 102–124.
[14] R. Smith, N. Japkowicz, M. Dondo, and P. Mason, "Using unsupervised learning for network alert correlation," in *Advances in Artificial Intelligence*. Springer, 2008, pp. 308–319.

[15] K. Tabia, S. Benferhat, P. Leray, and L. Mé, "Alert correlation in intrusion detection: Combining AI-based approaches for exploiting security operators' knowledge and preferences," in *Security and Artificial Intelligence (SecArt)*, 2011, p. NC.
[16] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 245–254.
[17] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 95–102.
[18] Balabit, "Syslog-ng," https://www.balabit.com/network-security/syslog-ng, 2015.
[19] H. Debar, D. A. Curry, and B. S. Feinstein, "The intrusion detection message exchange format (IDMEF)," in *IETF*, 2007.
[20] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *Recent Advances in Intrusion Detection*. Springer, 2001, pp. 85–103.
[21] G. Bradski, "Opencv," *Dr. Dobb's Journal of Software Tools*, 2000.
[22] K. Stouffer, K. Falco, and K. Scarfone, "National institute of standards and technologies - guide to industrial control systems (ics) security," National Institute of Standards and Technology, Tech. Rep., 2011.