Activity and Event Network Graph and Application to Cyber-Physical Security

Paulo Gustavo Quinan, Issa Traoré, and Isaac Woungang

Abstract The Activity and Event Network (AEN) is a new large graph model that enables describing and analyzing continuously in real-time key security relevant information about the operations of networked systems and data centers. The model allows identifying long-term and stealthy attack patterns, which may be difficult to capture using traditional approaches. The current chapter focuses on defining the model elements and the underlying graph construction algorithms, and presents a case study based on a cyberphysical security dataset.

Keywords Activity and event network · AEN graph · Graph model · Long-term attack · Attack patterns · Stealthy attack · Graph construction algorithm · Cyber-physical security · Probability model · Framework · Architecture

1 Introduction

Recently, it was discovered that a state-sponsored hacker group has been infiltrating the European Union's (EU) diplomatic communications network for years, downloading thousands of sensitive cables. The attack ran undetected for a three-year period and targeted more than 100 organizations and institutions, such as the United Nations and ministries of foreign affairs and finance. The attack is a type of emerging threat consisting of targeted and long-term campaigns delivered by skilled hackers who have clearly defined objectives and relentlessly work towards achieving their aims. These breaches can go undetected for a long period of time because of the hackers' ability to adapt to and escape defensive methods.

P. G. Quinan (✉) · I. Traoré
University of Victoria, Victoria, Canada
e-mail: quinan@uvic.ca

I. Traoré
e-mail: itraore@ece.uvic.ca

I. Woungang
Ryerson University, Toronto, Canada
e-mail: iwoungan@ryerson.ca

I. Traore et al. (eds.), *Artificial Intelligence for Cyber-Physical Systems Hardening*,
Engineering Cyber-Physical Systems and Critical Infrastructures 2,
https://doi.org/10.1007/978-3-031-16237-4_10

Noticeably, there has been an evolution from volume-based attacks towards stealth like low and slow style attacks. Although volumetric attacks often occur within a set time frame, low and slow attacks rely on an ongoing stream of malicious requests and have no distinct beginning or end. This makes their detection by current Intrusion Detection Systems (IDSs) and Security Information and Event Management (SIEM) tools challenging.

The Activity and Event Network (AEN) graph model is a new security knowledge graph whose goal is to spearhead the development of a new generation of security data analytics techniques that can gain better situational awareness of the threat environment and allow detecting, responding and investigating sophisticated and stealthy attacks using data from both the traditional security ecosystem and beyond the organization perimeter. It leverages the large dynamic uncertain multigraph theory to coherently express and analyse security data across various heterogeneous data sources and meaningfully link seemingly innocuous and unrelated events to expose hidden and long-term attack patterns. The purpose of the current chapter is to define the AEN graph model elements and corresponding graph construction algorithms. Furthermore, a cyberphysical security dataset is used to illustrate some of the threat detection capability of the AEN model. One of the prime targets of long term attacks are cyber-physical systems, where quite often security is treated as an afterthought in system design and configuration.

The remainder of the chapter is structured as follows. Section 2 defines the theoretical foundation of the AEN graph model. Section 3 presents the data sources used to construct the graph. Section 4 defines the AEN graph model elements by presenting the node and edge types involved. Section 5 gives an outline of the AEN underlying probability model. Section 6 gives an overview of the AEN framework architecture and present a case study based on BoT-IoT cyberphysical security dataset. Section 7 makes some concluding remarks.

2 AEN Graph Theoretic Model

To start describing the AEN model, it is important to consider the most basic constituent elements of a network of interconnected devices like hosts, how they communicate with one another and that those elements have their own characteristics. The model, therefore, needs to incorporate those pieces of information. However, the data used to extract them might be incomplete, noisy or simply incorrect. That means the model must also be able to describe said uncertainty.

Also of importance is the network's dynamism: At any moment devices or hosts can be added or removed from the network; they can start or stop exchanging data; IP addresses can be recycled; devices can be infected and later be fixed, patched or updated etc. It follows that to be able to track the past existence of elements and their relationships through time, the model needs to maintain information on the time spans in which each element has existed. The dynamic graph model defined by Casteigts et al. [1] is used to formalize this control, with two changes: the first is

that the model is expanded to combine probabilistic and temporal existence, while the second is the exclusion of latencies since they are negligible in the scope of this work. With that, important chronological relationships can be identified.

To summarize, the graph model has the following characteristics:

- Nodes have their own attributes. Thus nodes are labelled;
- Nodes can have multiple relationships at the same time. Thus nodes can have multiple edges between them;
- Relationships have a source and a destination. Thus edges are directed;
- Relationships have their own properties. Thus edges are labelled;
- Both relationships and nodes can be uncertain. Thus nodes and edges are weighed by probabilities of correctness, that is, that the information they represent is correct;
- Nodes and edges change through time and have an existence time span;
- Changes occur continuously;
- Processing times and latencies are negligible.

Those characteristics define the graph as a *Dynamic Uncertain Directed Multigraph*. Formally, it is defined as the 11-tuple

$$G = (N, E, s, t, \Sigma_N, \Sigma_E, \ell_N, \ell_E, \mathcal{T}, \pi_N, \pi_E) \tag{1}$$

where

- N is the set of all nodes of the graph as mentioned above;
- E is the set of edges representing relationships between nodes;
- $s : E \to N$, which assigns edges to their source nodes;
- $t : E \to N$, which assigns edges to their target nodes;
- Σ_N is a finite alphabet of available node labels;
- Σ_E is a finite alphabet of available edge labels;
- $\ell_N : N \to \Sigma_N$ is a map describing the labels of nodes;
- $\ell_E : E \to \Sigma_E$ is a map describing the labels of edges;
- $\mathcal{T} \subseteq \mathbb{R}^+$, that is, the time domain of the graph is in the positive real numbers;
- $\pi_N : N \times \mathcal{T} \to (0, 1]$, which assigns correctness probability values to nodes over time;
- $\pi_E : E \times \mathcal{T} \to (0, 1]$, which assigns conditional correctness probability values to edges over time given their endpoints.

3 AEN Data Sources

To construct the graph, heterogeneous data sources are used to extract data features capable of identifying nodes, relationships and their attributes. These involve both data sources available within the security perimeter as well as external data sources available through third-party services. Examples of internal data sources include

network traffic logs, flow data, system logs, firewall logs, IDS alerts, anti-virus (AV) logs and email security logs, while examples of external data sources include Domain Name Server (DNS) queries and WHOIS. Of special note are the data sources which report security events or suspicious activity, like IDSes or SIEMs. Generically they are called *detectors* and their reports or logs are generically called *alerts*.

The features extracted from the data sources can be divided into two categories related to how they are collected:

1. **First-order** features, which are collected directly from the sources;
2. **Second-order** features which are derived from the available data either by actively using different tools and services or by mining them into aggregates.

First-order features can be further divided into two groups according to where they are sourced from:

- **Network**, which are the more traditional data used by IDS derived by analysis of the network traffic. Its features are: IP Address, Transport Protocol, Transport Port, and Content-Type.
- **Application**, which are the data collected from log analysis of known applications in the network and from service calls either to or from said applications in case those applications are programmed to provide such functionality. This would include data sources like hypervisor logs, syslogs and IDS alerts. Its features are: Application, User, Authentication State, Attack Type, Alert Confidence, and Device fingerprint.

Second-order features can also be further divided into two groups:

- **Third-Party**, which are collected by actively contacting external services or performing scans in order to collect more data about a certain entity. Its features are: *Domain name*, *IP Address*, *Name server*, *Autonomous System Number (ASN)*, *Location*, *Name*, *Email*, and *Operating System (OS)*.
- **Aggregates**, which are mined from the aforementioned features and from the model itself and may consist of temporal or spatial aggregates.

4 Graph Model Elements

In this section, we present the AEN graph model elements, specifically the node and edge types involved, and provide some illustrative examples. By analyzing the first and second-order features described in the preceding section, it is possible to identify some distinguishable characteristics in them that give clues into how the graph can be constructed. The first one is how each feature can be better used to model the network. Some of them, like IP addresses and domain names, can form relationships among themselves. These features are the nodes of the graph and their relationships

are the edges. On the other hand, features like protocol and port are better employed as descriptors of said relationships and, therefore, are used to define attributes of either nodes or edges.

More generally, the nodes of the model are defined by any feature for which useful relationships could be formed, and the edges of the graph are defined by those relationships and their direction. The remaining features are used to define attributes of either nodes or edges.

4.1 AEN Nodes

Nodes can be classified into two distinct groups:

- **Active nodes**: Also called actors, active nodes are nodes that can be a source, a target, or a stepping stone (i.e., intermediary) in an attack. Examples of actors are hosts (either labelled by IP addresses or domain names), users (usually identified by authenticators like user name and email address) and devices. Active nodes can be further divided into:
  - *Internal actors*, which belong to or are under the control of the organization;
  - *External actors*, which are located outside the perimeter of the organization.
- **Passive nodes**: Nodes that carry some information or granular attributes for actors like DNS derived domain names and location nodes.

There are different types of nodes with each type having different features, as follows:

- **Account**: Represents an account in a system or application. It is derived from application and system logs and has the following properties:
  - Identifier
  - Application
- **Alert Group**: Aggregation of related detector alerts, called raw alerts internally, that might be generated in a short time frame thus associated to a single event or attack. It has the following properties:
  - Protocol
  - Source IP
  - Source port
  - Destination IP
  - Destination port
  - Service
  - Classification
  - Start time
  - Stop time

  - Severity
  - Confidence
  - Alert count

- **Domain**: Represents the domain name of an IP address. It is derived from a reverse DNS lookup of an IP address. When an IP address is identified, a DNS Reverse DNS lookup cycle is formed until the complete Domain/IP relationship is found. This is useful to identify attacks using Fast-Flux DNS and Algorithmically Generated Domains (AGDs). It has the following property:

  - Name

- **Host**: Represents a network host that communicates with other hosts in the network. It is an active node type and it is the most important node type of the model, from which almost all other elements originate. It is derived from different data sources like network data (packets or flow data), logs and alerts. It may be labelled with host-specific aggregates and historical information like if it was ever part of an attack, etc. It has the following property:

  - Identifier: Piece of information that can be used to consistently and uniquely identify a host though time, like IP address, MAC address, fingerprint or just a generic identifier value. Note that the IP address is not the ideal identifier for this case but given its ubiquity, it is used when other identifiers are not available.

- **IP Address**: Represents an individual IP address. It can be a first-order feature, derived from network data, logs or alerts, or a second-order feature when it is derived from DNS lookups or Network Address Translation (NAT) table conversions. It has the following property:

  - IP address

- **IP Range**: Represents a range of IP addresses. It is derived from WHOIS queries and has the following property:

  - Classless Inter-Domain Routing (CIDR) block

- **Location**: Represents a geographical location. It is derived from WHOIS queries or from IP geolocation services. It is useful to correlate hosts from similar locations and attribute attacks. It has the following property:

  - Tag: a combination of city, state/province, region and country according to what is available on the result of the query

- **Organization**: Represents an organization which controls a range or IP addresses or owns domain names. It is derived from WHOIS queries and has the following property:

  - Name

- **Person**: Represents a person who is listed as being an administrator or owner of a range of IP addresses, organization or domain names. It is derived from WHOIS queries and has the following properties:
  - Name
  - Email

4.2 AEN Edges

Each node type has different types of edges originating from and terminating in them. The different types of edges and their features (when applicable) are as follows:

- **Authentication Attempt** $Account \rightarrow Host$: Represents an authentication attempt of an account into a host. It has the following properties:
  - Timestamp
  - Source IP
  - Successful: Whether the authentication was successful or not
- **Triggered By** $AlertGroup \rightarrow Host$: Describes the source of an alert group
- **Used** $Host \rightarrow IPAddress$: Expresses that a host used an IP address to send data
- **IP Located At** $IPAddress \rightarrow Location$: Links an IP address to its location as defined by the WHOIS or geolocation query performed when the IP address was first added to the graph or when the relationship was considered stale
- **Part Of** $IPAddress \rightarrow IPRange$: Links an IP address with its IP range as defined by the WHOIS query performed when the IP address was first added to the graph or when the relationship was considered stale
- **Resolved To** $IPAddress \rightarrow Domain$: Links an IP with its domain name as returned by the reverse DNS query performed when the IP address was first added to the graph or when the relationship was considered stale
- **Controls** $Organization \rightarrow IPRange$: Expresses that an organization controls an IP Range as described by the WHOIS queries performed when the IP Range was first added to the graph or when the relationship was considered stale
- **Organization Located At** $Organization \rightarrow Location$: Links an IP address to its location as defined by the WHOIS query performed when the first IP belonging to the organization was first added to the graph or when the relationship was considered stale
- **Owns** $Person \rightarrow Domain$: Represents an ownership relationship between a person and a domain as described by the WHOIS query performed when an IP that resolved to this Domain was first added to the graph or when the relationship was considered stale
- **Session** $Host \rightarrow Host$: Represents a communication session between two hosts on the same protocol, ports and within a certain activity time window. Its direction

is based on who initiated the connection. It is primarily an aggregation of network data, but data from other sources like logs and alerts that can be used to identify such communication is also used in order to fill up possible gaps in the network data available. It has the following properties:

- Start time
- Stop time
- Protocol
- Source port
- Destination port
- Source size: Sum of the length of all packets from source to destination
- Destination size: Sum of the length of all packets from destination to source
- TCP state: Only applicable for TCP packets, it describes the state of the TCP connection as of the last packet belonging to this session, that is, if the connection has been established (handshake completed), finished or has only ever started but not been established
- Packet count: The number of packets exchanged between the hosts as part of the session
- Fragmented packet count: The number of fragmented packets exchanged between the hosts as part of the session
- Alert count: The number of alerts generated as part of the session

5 AEN Probability Model

5.1 Probability Model Definition

The basic probability assignment involved in the AEN model consists of the probability of correctness, π, for graph elements (nodes and edges) and feature confidence. At inception, each graph element is assigned an correctness probability which stems from the originating data.

Most graph elements are derived from data that in some way can be categorized as deterministic; like when packets related to a TCP handshake between two hosts are received, the model can be certain that those two hosts are communicating and in what direction, or when an application reports that an authentication attempt was made for a certain account there is no doubt regarding the application or the account being used. In these cases, π is trivial and set to 1, or more technically to $1 - \epsilon$, where ϵ represents the inherent probability the data injected into the system has been faked, tampered or corrupted.

On the other hand, a few data sources are inherently imperfect, which introduces a layer of uncertainty to the nodes, relationships and attributes they generate and thus must be taken into consideration. Examples of these data sources include IDSes or other detectors, user/device fingerprinting schemes, geolocation services etc. More

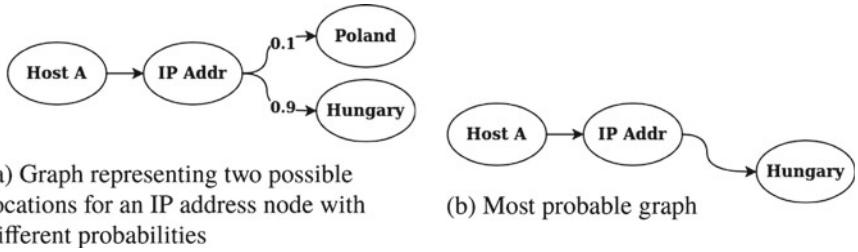

Fig. 1 Example of probability modelling in the AEN graph and the resulting most probable sub-graph derivation

than that, in some cases the data have a higher probability of being faked or spoofed, like a stray IP packet not part of a established communication between two hosts which is more likely to have a spoofed source address. In these cases, the correctness probability for the graph elements will correspond to the expected accuracy of the data.

As an example, consider the hypothetical scenario where the geolocation service assigns for a given IP address 90% probability of it being from Hungary and 10% probability of it being from Poland at a given point in time $t \in \mathcal{T}$. Based on that, one *IP Address* node and two *Location* nodes will be added to the graph. The two countries exist beyond any doubt so their π are set to 1. Likewise, the IP address, at least as a member of the set of all possible IP addresses, also exists, and is therefore also assigned $\pi = 1$. From there, one edge between the *IP Address* and each of the two *Location* nodes constrained to t are also added with their probability of correctness being set to the respective probability returned by the geolocation service. In such a manner, both possible, but conflicting, relationships can be modelled and, at a later point, used as part of different inference processes.

Figure 1 depicts a sample graph based on the above scenario with (a) showing the full graph representing the 2 conflicting relationships and (b) showing the resulting "most probable graph" derivation. Future data might result in updated values and thus different derivation results.

In the above scenario, all probabilities are equivalent to probabilities of existence, however, that is not always the case. Consider now an alert added to the graph. Short of an attacker being able to inject a false alert into the system, the model can be sure that the alert exists and was generated by the detector. Therefore, its probability of existence is equal to $1 - \epsilon$. That, however, doesn't properly represent the uncertainty regarding the alert. Instead, what is important in this case is describing the probability of the alert being correct, that is, not a false alarm.

Generally, a detector can be considered as a binary classifier that classifies events as either malicious (or at least suspicious) or benign and generates alerts when an event is classified as malicious. As a classifier, the detector has an accuracy which underpins the probability of correctness of the alerts it generates. Moreover, some detectors will also provide a confidence score for the alert which can be used to further refine the alert's probability.

Therefore, to calculate π for an alert, we first need to define its expected accuracy. There are different levels of granularity that can be used when performing the accuracy calculation. The coarsest grain (and simplest) calculation would be to calculate the precision (Positive Predicted Value—PPV) of the detector as a whole by evaluating it against different datasets and use that as the surrogate. The precision is used in this case because it describes the ratio of true positives among all predicted positive elements, in other words, it is congruent to the probability of a predicted positive observation (an alert in our case) being a true positive.

This is simple but the values obtained may be too generic and thus not as useful. A finer grained approach would be to group alerts based on a common factor and calculate the accuracy for each group separately. The grouping can be done in several different ways like by severity, by family of attacks, by individual attack type inside a family or even by individual rules or anomaly metrics. These would provide more specific probability values but on the other hand would require much more data in order to be calculated meaningfully. If there are no examples or not enough examples of a given alert in the datasets, then the respective values cannot be calculated using this approach or at least not calculated meaningfully. In these cases, the general detector accuracy would have to be used.

In cases where a confidence score is available, we compute the alert probability by combining the detector accuracy and the confidence score into a probability value through score calibration. The score calibration process is described in more detail in a separate report [10].

Finally, recall that alerts are not added directly to the graph, instead, similar alerts of the same type, origin and target that occur in a short time frame are grouped into alert groups. In most cases, all alerts of the same type will have the same π, hence, the probability of any alert can be used as the probability of the group. The exception to that is cases where different alerts have different confidence scores. In those cases, to obtain π of an alert group A from it's constituent alerts A_i we pick the probability of the alert with the highest confidence. Formally, $\pi(A)$ is defined as:

$$\pi(A) = \max_{i=1}^{n} \pi(A_i) \tag{2}$$

The probability of correctness serves as a starting point to inferences, derivation or other types of analyses in the graph as seen in the following sections.

5.2 *Probability Model Usage and Application*

It is expected that the basic probability assignment (i.e. alert probability) will be fed to the model by leveraging the threat detection systems (IDSes, AVs, anti-phishing systems) already available in the organization. However, this is not required. Regardless of whether there are some pre-existing IDSes in the organization, the AEN will provide independently its own threat detection schemes which encompass attack fin-

gerprinting, graph clustering and unsupervised statistical threat detection and that will be fed back to the model. As a result, the AEN threat detection scheme will also provide some of the classification schemes mentioned in the above probability model. As new threats are discovered by the AEN threat detection schemes, alerts will be generated and incorporated in the graph model along with the alerts generated by other pre-existing schemes if applicable. Although the AEN threat detection schemes could leverage the alerts generated by pre-existing schemes, it is our goal, currently, to keep them separate to ensure the independence of the AEN schemes. In future work, we will explore how pre-existing alerts information can be leveraged by the AEN threat detection schemes for detection purpose.

Currently, we use the AEN probability model mainly for threat assessment and visualization, and providing underlying context and explanatory information. This is based on the concept of *threat horizon* and *reverse threat horizon*.

The horizon of node u can be understood to be the set of all possible nodes which u could have affected or exchanged data with. In the case of an attack, all the nodes for which a direct journey exists are nodes that the attacker could have compromised directly, even if only temporarily. On the other hand, nodes for which only indirect journey exists can only have been compromised if somewhere along the journey the attacker was able to permanently compromise the system either through some kind of outbound connection to its servers or through the installation of an automated malware.

That demonstrates the importance and capacity of the horizon in regards to the forensic analysis of the network. It provides a complete view of the network from the point of view of the originating node and defines what could possibly be within its reach should it be a threat to the network. In other words, the horizon can be viewed as the subset of nodes which can be threatened by another node.

Under this optics, the horizon of a node u is here defined as its ***Threat Horizon***, denoted as $\overrightarrow{TH_u}$, and defined as the subset of N in which all elements are reachable from u. Formally:

$$\overrightarrow{TH_u} \subseteq N, \quad \overrightarrow{TH_u} = \{w \in N : u \rightsquigarrow w\} \tag{3}$$

Inversely, the ***Reverse Threat Horizon*** of a node v, denoted as $\overleftarrow{TH_v}$, identifies, from the point of view of a target node, which network nodes could have attacked and compromised it. It is defined as the subset of N in which all elements can reach v. Formally:

$$\overleftarrow{TH_v} \subseteq N, \quad \overleftarrow{TH_v} = \{w \in N : w \rightsquigarrow v\} \tag{4}$$

The Threat Horizon can be used as a starting point of any node specific analysis. The first step is to select the best source node for the Threat Horizon, which is characterized by its power to link distinct attacks together. Therefore, the selection, when available, of owners, common domains or users might result in a combined Threat Horizon of multiple attacks.

The same applies for identifying the target node of the Reverse Threat Horizon. Generically, those nodes are called the **focal points** of analysis.

By using the basic probability model defined above for the AEN model we can derive probability measures for the threat horizon and reverse threat horizon, and use these values to guide threat assessment, visualization, and forensic analysis. This is an ongoing work that will be presented in more details in future papers.

6 Graph Construction and Framework Implementation

6.1 Framework Architecture

The system architecture of the AEN framework is depicted in Fig. 2.

The central component of the system is the AEN Engine, which is responsible for maintaining the AEN graph. It provides key functionalities like processing and aggregating incoming data through data receivers, attack fingerprint matching, proactive third-party data collection to supplement other incoming data, anomaly detection and the probabilistic model underlying the AEN graph.

The engine stores the graph in a custom-made, in-memory, graph database with capabilities to add, update, remove and search for graph elements. Persistence is obtained via frequent storage of snapshots which can be reloaded from the disk in case of a failure or to review a previous graph state. These snapshots can also be shared with other tools and systems that provide other auxiliary operations or functionalities like visualization of the graph or scalability via read replicas.

Both the engine and the graph database are implemented in Java and are executed in a single process which allows for direct memory access of the graph elements, thus avoiding any extra serialization overhead.

The engine provides two operation modes. The first one is the online mode, in which real-time data is continuously collected from external machines and devices by a client application which then pre-processes and sends the live data to the engine.

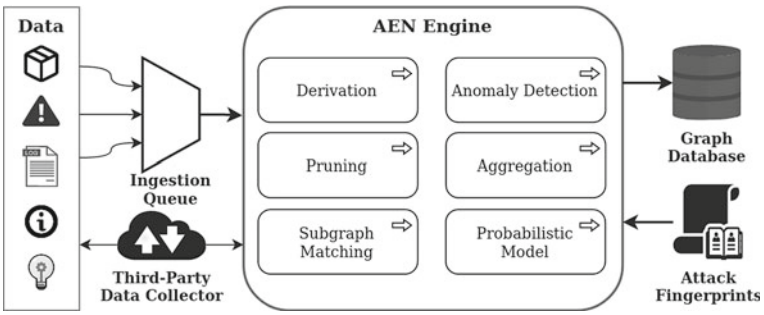

Fig. 2 AEN system architecture

Table 1 Bot-IoT files used in experiment

Type	Name
Data exfiltration	IoT_Dataset_data_theft__00002_20180618111101.pcap
Data exfiltration	IoT_Dataset_data_theft__00013_20180618112736.pcap
OS scan	IoT_Dataset_OSScan__00001_20180521140502.pcap
OS scan	IoT_Dataset_OSScan__00003_20180521150020.pcap
Service scan	IoT_Dataset_ServiceScan__00007_20180515133133.pcap
Service scan	IoT_Dataset_ServiceScan__00007_20180521224912.pcap
UDP DDos	IoT_Dataset_UDP_DDoS__00019_20180604180729.pcap

The second one is the offline mode, in which previously collected data is added to the graph directly. In both cases, AEN Engine uses data ingestion queue which sorts the data chronologically and controls the flow of data.

The current implementation supports network traffic data, either raw or flow data, some syslogs, IDS alert logs from Snort, Zeek and custom alerts derived from Kitsune's anomaly detection output and pre-collected IP addresses information derived from DNS and WHOIS queries, which the engine can also actively query for if the data is not available.

6.2 Case Study Based on a Cyperphysical Security Dataset

To better demonstrate the functionalities and capabilities of the AEN Graph, we performed an experiment using the BoT-IoT dataset [2–7], provided by the Cyber Range Lab of the University of New South Wales (UNSW) Canberra, which consists of legitimate and simulated Internet of Things (IoT) network traffic, along with different botnet attacks.

The graph was generated using a subset of the dataset with 1.4 GB of pcap data comprising part of the available UDP-based DDoS, OS scans, service scans and data exfiltration attacks. Table 1 lists the selected files. This data resulted in a graph with 236 nodes, including 56 hosts, and 337, 349 edges, including 337, 073 sessions.

Figure 3 shows a zoomed out visualization of most of the resulting graph. Blue nodes are hosts while the edges between them are sessions. To help visualization, multiple edges between the same nodes are combined into a single, thicker, edge. Note how the graphical representation of the data clearly shows a cluster of activity around a few hosts with high inter-connectivity and outside of that a host with high centrality that initiated connections to several different hosts. It's those kind of patterns that can be identified by matching algorithms to expose otherwise hard to identify relationships between hosts and anomalous or known malicious behaviours.

By zooming in and adding labels to the graph elements, it is possible to see in more detail how the different node types are interconnected through the different edges. Figure 4 shows that visualization.

Fig. 3 Zoomed out AEN graph sample derived from the BoT-IoT dataset

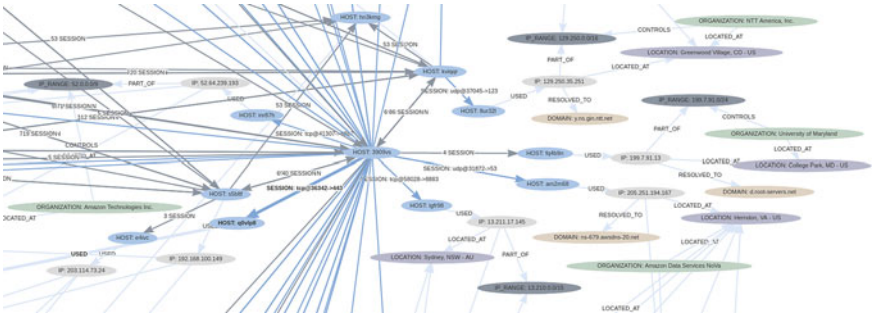

Fig. 4 Zoomed in AEN graph sample derived from the BoT-IoT dataset

To label the graph, we derived host labels from the dataset's flow-level labels such that any host is labelled as malicious if it originated any flow labelled as being part of an attack. Out the 56 hosts in the graph, 8 are labelled as malicious, a prevalence of 14.3%.

As a base of comparison, we first ran the data used to build the graph through Snort IDS using the rules available on its website for registered users. To classify hosts, we followed a similar rule based on the Snort alerts, with hosts that were the source of any alert being classified as malicious by Snort.

The results show a high number of errors, with it only being able to correctly classify 4 out of the 8 malicious hosts and 34 out of the 48 benign hosts. A high number of type II errors is expected given that Snort is a signature based IDS but on the other hand, the high number of false positives (type I errors) was not expected for the same reason. Table 2 presents the resulting confusion matrix.

Table 2 Confusion matrix of the Snort IDS

		Predicted	
		Malicious	Benign
Actual	Malicious	4 (50%)	4 (50%)
	Benign	14 (29%)	34 (71%)

Table 3 Confusion matrix of the AEN's anomaly detection algorithms

		Predicted	
		Malicious	Benign
Actual	Malicious	8 (100%)	0 (0%)
	Benign	4 (8%)	44 (92%)

More specifically, the performance of the Snort IDS for the dataset is thus: Sensitivity of 50%, specificity of 71%, precision of 22% and negative prediction rate of 89%. That gives a F1-score of 0.3 and a Matthews Correlation Coefficient (MCC) of 0.15.

Afterwards, we executed the anomaly detection algorithms of the AEN [9] against the generated graph and followed the simple classification rule in which each host identified as anomalous was classified as malicious.

The anomaly detection algorithms were able to identify all 8 hosts as anomalous but also identified further 4 benign hosts as anomalous. Table 3 presents the resulting confusion matrix.

Compared to the results obtained by Snort, our algorithms exhibited a much higher performance with a sensitivity of 100%, specificity of 92%, precision of 67% and negative prediction rate of 100%. Furthermore, the F1-score is 0.8 and the MCC is 0.78.

Finally, we also performed matches against our attack fingerprint database as described in [8]. As in the other cases, host classification adhered to the simple rule where a host was classified as malicious if it was reported as matching by the detector. In this case, that meant the host was identified as being the source of an attack pattern that matched any of the stored fingerprints.

Like Snort, the fingerprint were able to correctly classify the same 4 out of the 8 malicious hosts. However, it did not misclassify any of the 48 benign hosts, which is expected given that (a) it is by nature signature based and (b) only a few fingerprints are available. Table 4 presents the resulting confusion matrix.

Compared to the results obtained by Snort, the fingerprint matching also showed a much higher performance with a sensitivity of 50%, specificity of 100%, precision of 100% and negative prediction rate of 92%. Furthermore, the F1-score is 0.67 and the MCC is 0.68.

Table 4 Confusion matrix of the fingerprint matching

		Predicted	
		Malicious	Benign
Actual	Malicious	4 (50%)	4 (50%)
	Benign	0 (0%)	48 (100%)

When compared with the anomaly detection, it showed a better false positive rate compared to a worst false negative error which is natural given their different characteristics and shows they can be complimentary.

7 Conclusion

The AEN graph model is a new paradigm that allows the capture and analysis of the activities and events involved in the operation of networked systems and data centers. In the current chapter, we have defined the graph model elements and provided an overview of the underlying probability model. While the focus of the current chapter is on graph construction only, future papers will present in more detail our approaches for threat detection using the AEN graph model.

References

1. Casteigts A, Flocchini P, Quattrociocchi W, Santoro N (2012) Time-varying graphs and dynamic networks. Int J Parallel Emergent Distrib Syst 27(5):387–408
2. Koroniotis N (2020) Designing an effective network forensic framework for the investigation of botnets in the Internet of Things. Ph.D. thesis, UNSW Canberra
3. Koroniotis N, Moustafa N (2020) Enhancing network forensics with particle swarm and deep learning: the particle deep framework. In: International conference on artificial intelligence and applications
4. Koroniotis N, Moustafa N, Schiliro F, Gauravaram P, Janicke H (2020) A holistic review of cybersecurity and reliability perspectives in smart airports. IEEE Access 8:209802–209834
5. Koroniotis N, Moustafa N, Sitnikova E (2020) A new network forensic framework based on deep learning for internet of things networks: a particle deep framework. Futur Gener Comput Syst 110:91–106
6. Koroniotis N, Moustafa N, Sitnikova E, Slay J (2018) Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In: Hu J, Khalil I, Tari Z, Wen S (eds) Mobile networks and management. Springer International Publishing, Cham, pp 30–44
7. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B (2018) Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset
8. Nie C, Quinan PG, Traoré I, Woungang I (2022) Intrusion detection using a graphical fingerprint model. In: 2022 22nd IEEE international symposium on cluster, cloud and internet computing (CCGrid), pp 806–813

9. Quinan PG, Traore I, Gondhi UR, Woungang I (2022) Unsupervised anomaly detection using a new knowledge graph model for network activity and events. In: Renault E, Boumerdassi S, Mühlethaler P (eds) Machine learning for networking. Springer International Publishing, Cham, pp 117–130
10. Yousef WA, Traore I, Briguglio W (2022) Classifier calibration: with application to threat scores in cybersecurity. IEEE Trans Dependable Secure Comput, pp 1–1