EVE and ADAM: Situation Awareness Tools for NATO CCDCOE Cyber Exercises

Francisco Jesús Rubio Melón
ISDEFE, Madrid
SPAIN
jrubio@isdefe.es

Teemu Uolevi Väisänen
VTT Technical Research Centre of Finland Ltd, Oulu
FINLAND
Teemu.Vaisanen@vtt.fi

Mauno Pihelgas
NATO CCDCOE, Tallinn
ESTONIA
mauno.pihelgas@ccdcoe.org

ABSTRACT

We present a new situation awareness visualisation tool, the Events Visualisation Environment (EVE), and its internal events aggregator module, the Advanced Data Aggregation Module (ADAM), which have been successfully used during the most recent cyber exercises (i.e., Locked Shields and Crossed Swords) organised by the NATO Cooperative Cyber Defence Centre of Excellence (NATO CCDCOE).

The functional requirements for EVE and ADAM were based on the unique cyber exercise needs for analysis and game development, and were finalised after we had completed a state-of-the-art review to look for suitable tools that could meet our requirements.

The main purpose of EVE is to visualise security alerts on any given network map. ADAM, the supporting events aggregation module, processes, combines and filters incoming notifications from various types of sensors, and makes them ready to be visualised by EVE. EVE offers an intuitive and real-time visualisation that is easily understandable at first glance by both technical and non-technical staff. It also allows for recording and playback, and considers attack types, game phases, attack sources, and targets.

The information required by EVE is obtained from different sensors operating on the network. EVE allows for a very simplified communication channel with them, based on JSON formatted messages sent over an HTTP POST request. The sensors used during the cyber exercises to test the tools are also described here.

The tools have provided an enhanced situation awareness experience over previous cyber exercises organised by NATO CCDCOE, and can be used in other exercises or, more generally, in real-life, production-ready environments. EVE (with ADAM included) is published as an open source tool, which is freely available on the GitHub page of the NATO CCDCOE.

Keywords: aggregation; alerts; filters; network maps; situation awareness; visualisation

1 INTRODUCTION

Graphical visualisation is a key component of any situation awareness and security management tool. The field of information visualisation forms a whole research area in computer science [1]. Visualisation allows operators to deal with large amounts of raw data generated from different security monitoring tools, showing them as meaningful graphs and pictures [2]. However, our experience with existing tools, whether commercial or open source, has convinced us that there is ample room for improvement, and that new approaches can be explored to overcome some of the challenges involved in the field of situation awareness.

Providing real-time situation awareness is indeed challenging, and visualisation of data is undeniably one of the best techniques to be used [3]. Although most modern tools rely on GeoIP databases for attack visualisation, the technique is not compatible with making use of network diagrams drawn with the help of tools such as Microsoft Visio that depict real or fictitious network maps. However, these network maps are commonly used both in cyber security exercises and in real-world production environments.

The EVE and ADAM toolset was developed primarily for the purposes of the NATO CCDCOE cyber exercises (i.e., Locked Shields [4] and Crossed Swords [5]) where they have been successfully used to visualise detected attacks carried out by the Red team in the fictional, yet realistic Blue team networks. The exercises involve many participants, each representing one of five different roles: Blue team (i.e., defence), Red team (i.e., attacks), Yellow team (i.e., situational awareness), Green team (i.e., exercise infrastructure), and White team (i.e., exercise management, media game, etc.).

This paper builds on the visualisation aspects of our previous work on a framework called Frankenstack [6] which was published at MILCOM 2017. Here, we describe an approach that provides a clear and simple visualisation of security alerts depicted on a network map. The underlying idea is to abstract away all security sensors operating on a given monitored environment, combine and integrate their outputs, and visualise the most relevant information regarding security alerts on the network map. The sensors themselves, while crucial to detecting security threats, will not be visualised. Instead, the visualisation will concentrate on the results obtained by combining the information they provide. In addition, we want to show them in a meaningful and intuitive way and in real time.

The system we decided to build should overcome some of the perceived limitations of previous tools (see ‘Related Work’ section below) and therefore we set up to achieve the following goals:

- **Intuitive**. Draw simple graphs on a network map, easily understood by both technical and decision-making staff.
- **Focused**. Visualisation tools tend to present far too much information, usually overloading users with sophisticated graphs and comprehensive details, and distracting them from the relevant information.
- **Cohesive**. The tool should be able to combine information from multiple, unspecified sensors. Some existing situation awareness tools may suffer from presenting information that is not aggregated, normalised or segmented enough prior to the visualisation phase.
- **Contextualised**. We would like to show information in its own context. For example, it has been customary to draw a network map from a PCAP file and see only devices that were sending traffic in when the file was captured. We instead needed a tool that shows any other known devices, either detected or not by sensors or attack activity.
- **Multipurpose**. Many visualisation tools are meant for presenting only one type of event, such as traffic in networks, content of packets, access to honeypots, or resource use of computers. We need a tool that is able to deal with any type of events.
- **Real-time**. The alerts should be visualised as close as possible to real time.
- **Ease of communication between tool and sensors**. Pre-existing situation awareness tools typically tend to make use of proprietary message formats, and combining that information may become unfeasible. Our solution should allow the integration of the new tool with as many sensors as desired, in a simple and easy to implement way for system administrators or security experts.
- **Web interface**. To improve the ease and simplicity of the tool from the user’s perspective.

In the following chapter of this paper we present a brief literature review concerning situation awareness tools and techniques. Next, in chapter 3 we describe in detail the tools we have developed – EVE, and its internal events aggregator module, ADAM – to show further details of our situation awareness model. This is followed by chapter 4 about the sensors we have deployed and configured to test the tool, which are essential to the working of the system, since they provide the data on which EVE feeds. We then proceed to show the results, extract conclusions and identify lessons learned. Potential improvements and future work are discussed in chapters 5 and 6 respectively.

2 RELATED WORK

A lot of research has been done on existing tools for visualising cyber security events [1, 7, 8, 9]. Many of the visualisation tools are meant to handle and present only certain type of data [10]. Very few tools have considered how to present cyber information to people with less technical experience and knowledge [2]. Some tools support interoperability with other applications and utilities [10].

Various types of visualisation techniques have been used. Graphs based on lines, bars or map-like visualisation [11], bubbles [2], IP matrices [12], integrating logical, geometrical, and temporal visualisation in 3D [13], 3D fish tanks [14], hierarchical network maps [15], or circles [16] exist. In addition, there are tools such as Alerta [17] for consolidating and de-duplicating alerts from multiple sources and visualising them as textual lines. In a recent paper [18], Arendt et al. presented CyberPetri, which is a circle-packing visualisation tool. It is a sub-component of Ocelot that was previously presented in [19] as a user-centred decision support visualisation tool. Perhaps one of the closest visualisation tool to EVE is GARNET [20] which provides a graphical network and attack graph, but it does not enable use of own network maps.

Regarding monitoring deceptive tools such as honeypots, which were an integral part of our test scenario, map-like visualisations are useful for visualising geo-referenced data (for example, a map showing the countries most infected by a given piece of malware. However, they do not solve the problem of visualising access to decoys in private organisational networks. For example, in honeymaps [21], infected devices and honeypots are first mapped to countries using IP geolocation and then shown on a World map. In a Kippo-Graph [22], information such as used passwords, commands and IPs can be visualised. Unfortunately, it does not provide the option to visualise anything other than access to a Kippo-based honeypot. T-Pot [23] is a multi-honeypot platform which is using Kibana [24] for visualizing events.

Kibana provides various views to present vast amounts of data and it has been commonly used to visualise data from Intrusion Detection System (IDS) tools such as Suricata or Snort or from honeypots [25]. However, Kibana lacks the functionality to visualise alerts on a custom network map. Similarly, Graylog [26] can be integrated with various IDS tools and honeypots to visualise suspicious activity.

Standardisation of descriptions, representations, and exchange of events has been done. The main purpose has been to create unified event languages for interoperability, and to improve the audit process and users' ability to interpret and analyse event logs and audit data. One example has been MITRE's Common Event Expression (CEE) [27], which is a formal specification of event syntax, transport and field definitions.

In summary, all the visualisation tools presented in this section can, in principle, be used to analyse various types of attacks and log types; however, none of them were able to fulfil all our requirements described in the introduction.

3 EVE AND ADAM

EVE is the visualisation tool we developed to meet the requirements stated above. It is freely available on the GitHub page of the NATO CCDCOE [28]. To describe it, it is useful to first define the main elements and concepts handled by the application:

- **Network map**. A structured network consisting of a variety of different subnets, servers, workstations, routers, firewalls, etc. In the largest cyber exercise where we tested EVE, Locked Shields 2017, the network consisted of over 2,000 virtualized servers and workstations, with over 100 different subnets.
- **Sensors**. These are the monitoring systems that scan either the full network or any of its elements in a permanent search for security violation attacks. Our testing environment (the cyber exercises developed and run by NATO CCDCOE) consisted of:
  - Intrusion Detection/Protection Systems (IDS/IPS)
  - System logs monitoring tools. In particular, OS command execution loggers (such as Snoopy Logger) [29]
  - Deception techniques and tools, such as decoys, honeypots, and honeytokens

Detection tools [6] lie outside the network map and are not considered to be a part of it. System logs are part of every machine in the network map. Some decoys such as honeypots are usually represented in the network map, but for example baits stored in real devices are not. The training audience of the cyber exercise (either red or blue teams) generally have no knowledge about the sensors deployed for the exercise. However, if training monitoring teams is one of the goals of the exercise, information about sensors becomes valuable.

- **Events**. From EVE's perspective, an event is any security violation attack discovered by a sensor. For example, the detection by an IDS system of the string " ' or 1 = 1 # " within an HTTP request (a trivial indicator of an SQLi attack) constitutes an event. Analogously, the detection by Snoopy, or inside the system logs, of a suspicious "nc -l" command, a likely indicator of a reverse shell, will generate the corresponding event. Equally, we also consider insider threats such as access to honeypots, decoy files or other decoy resources (such as URL honeytokens), even if the access to honeypots takes place without executing any intelligent human-like behaviour. However, as it can be seen from Table 2, it is possible also to visualize very specific events that certainly have required intelligent human behaviour and attack chains describing the access to various machines in the environment. EVE's events are characterised by the origin of the attack (any element or subset of the whole network), the destination of the attack (again, any element or subnet), the time of detection and the payload of the attack (for example, "SQL injection").
- **Equivalent Events**. In general, each sensor will specialise in detecting a subset of all possible security violation events. However, it is important to note that at any given time more than one sensor can simultaneously detect the same security violation attack, possibly making use of different clues. Those events, while different, are considered to be "equivalent" by ADAM (EVE's aggregator module). It is also possible to find that a given sensor may report events that can be all considered to be equivalent (for example, an attack that proceeds through repeating variations of a given pattern). Strictly speaking, according to ADAM's implementation, two events are equivalent if they share the same source of attack, the same destination of attack and take place at approximately the same time (not longer than five seconds apart)
- **Alert**. All equivalent events are combined into a unique single "alert". They constitute the main visualisation goal of EVE. In fact, EVE was designed to visualise alerts, not events. As events arrive, ADAM will try to find correlations among events and pass the processed information to EVE.

3.1 Functionality

The general goals stated in the introduction were further elaborated and we built EVE and ADAM with the following capabilities:

Core Functionalities:

1. The application´s main goal is to visualise:

  a. The network map;

  b. Security alerts (drawn on top of the network map).

2. The network map is assumed to be static, and completely known before the exercise takes place. It is not automatically built out of discovered elements. See below for further discussion on this topic and alternative approaches.

3. The network map, however, is not fixed once and forever, and EVE considers the fact that different exercises will require completely different underlying network maps.

4. Dynamic input data on security events are provided to the application by a set of sensors monitoring the network.

5. The communication between the sensors and the web application is homogeneous across the different sensors and is based on simple and ad-hoc protocol. Sensors' administrators can configure them to send their output data to EVE in an easy and convenient way.

6. The application internally implements the algorithms that aggregate incoming data, the security events, into alerts.

Visualisation:

1. The visualisation takes place on a single web page.

2. The alerts visualisation page offers an intuitive and clear picture, ready to be fully understood at a first glance without requiring highly technical skills.

3. The network map is visualised as an image with standard icons for its elements.

4. The alerts are visualised on top of the underlying network map using easy to understand clues such as red arrows and coloured circles to show source and destination of the current attack.

5. Alerts are shown as soon as any sensor reports the first event, without any noticeable delay.

6. One alert at a time: every time a new alert is detected the previous one will cease to be shown.

7. The visualisation is kept up to date automatically, without any interaction from the user – no need for page reloading, or clicking on any 'update' link.

8. EVE's web page is interactive: clicking on the alerts representation triggers a popup with additional, detailed information meant for technical staff.

9. EVE keeps a record of all alerts detected during the execution of the cyber exercise.

10. At any moment, EVE offers the option to access and view those recorded attacks in any desired timespan (the so called "replay" mode).

3.2 EVE Architecture

EVE follows a modular structure that favours continuous improvements and ease of development. It consists of the following key modules:

- **Input Module**. This is the module responsible for receiving all incoming data sent to EVE from the sensors. It is implemented as a RESTful-like service that can easily be invoked from the sensors using standard HTTP POST requests.
- **ADAM** (Advanced Data Analysis Module). This module is responsible for combining events to generate the alerts that will be visualised by EVE. ADAM defines precisely which events must be considered as "equivalent" (and therefore, be tied to a single, unique, alert). It plays a similar role to event correlators tools, such as SEC [30]. However, it is fully customised and optimised to meet EVE's needs.
- **Network Definition Module**. This module defines and stores the underlying network structure: the elements (subnets, hosts, etc.) contained in it, their relationship and relevant information associated to each of them, such as element type, IP's, domain names, etc. Currently, the network map is static, known before EVE starts, and does not change with time. But the network map and its associated information can be arbitrarily changed to suit any new cyber exercise or real-world scenario.
- **Graphical Interface Module**. This handles the representation of the network map, the alerts and the user interaction.
- **Internal Control Module**. This module listens for incoming events, sends them to ADAM for analysis and combination into alerts, stores them into a relational database and finally pushes the alerts to the Graphical Interface Module to be rendered on top of the network map.

3.3 Communication with EVE

All events detected by the external sensors, and possibly from additional event correlators or event aggregators, are sent to EVE within an HTTP POST request to an EVE's web service. The parameter of the request contains a JSON formatted "message". A typical message looks like the one shown in Listing 1:

```
{
    "source"  : {
        "IPV4"        :   "192.168.8.17",
        "IPV6"        :   ""
    },

    "target"  : {
        "type"        :   "host",
        "IPV4"        :   "",
        "IPV6"        :   "fe80:1181:150::33/64",
        "name"        :   "hostX"
    },

    "payload" : {
        "name"        :   "SQLi",
        "sensor"      :   "IDS",
        "evidence"    :   "SQL command 'or 1=1' found in URL",
        "url"         :   "http://www.example.com/query"
   } ,

    "related"  : {
        "type"        :   "honeypot",
        "IPV4"        :   "192.168.8.171",
        "IPV6"        :   "",
        "description" :   "Discussion about SQL vulnerabilities in 192.168.8.17 was stored in
                          bait files of user megan.fox in 192.168.8.171 (laptop2.int.ex)"
    }
}
```

Listing 1: Example message sent to EVE

The names of the fields within this JSON message were chosen to be as descriptive as possible. However, it will be useful to define them in detail since it will allow us to gain further insight into EVE's underlying model.

- **Source**. This represents the origin of the security breach event as detected by the sensor. Note that either IPV4 or IPV6 IP or both addresses may not be known.
- **Target**. This is the destination of the attack. The “type” field can be “host” or “network”. For example, an nmap network scan is of type=network. The sensor that detected this event must include at least one of the following fields: IPV4, IPV6 or name (= target hostname).
- **Payload**. The objective of the security violation:
  - *name*: is a free text such as “SQLi” or “memory corruption”
  - *sensor*: can be any of: syslog, snoopy, honeypot, ids, pcap, netflow, influx, other
  - *evidence*: a format-free and short text which serves as “proof” or indicator of the attack detected by the sensor.
  - *url*: if filled, it points to a web page containing additional information regarding the event.
- **Related**. This carries information regarding an additional host involved in the attack. For example, if an attack is proxied through an intermediate server into another network segment, or fake user credentials were collected from a honeypot and used by the attacker elsewhere in the network.

Although this format could be improved, it is easy to implement by sensor administrators where EVE is deployed and, according to our experience, it worked smoothly and did not introduce any noticeable delay.

3.4 EVE Implementation

EVE has been implemented as a Java web Application intended to be deployed on a Tomcat 8 server. It stores all incoming events as well as the alerts on a MySQL database using JPA persistence API. The visualisation is carried out using JavaScript and HTML5 canvas API.

EVE makes use of HTML5 WebSockets to push the alerts to the web page where they are automatically drawn on top of the network map. WebSockets allow us to keep the visualisation on the web page up to date without the need of any manual or periodic reloading.

4 SENSORS

During execution at NATO CCDCOE’s cyber exercises, several sensors and event correlators were used to provide information to EVE. This section describes them briefly.

4.1 Detection tools

Various IDS systems were used during execution, three of which are Suricata [31], Bro [32], and Snoopy [29]. Suricata is an open source network intrusion detection (NIDS) and prevention (NIPS) tool, performing protocol analysis, content searching and matching. It inspects the network traffic using rules and signature language, and has Lua scripting support for detection of threats. Bro was configured to analyse Address Resolution Protocol (ARP) and IPv6 Neighbor Discovery Protocol (NDP) to detect new and perhaps undocumented machines on the monitored networks. Snoopy is an open source library for logging all executed commands and arguments on a system. All were easily connected to EVE.

When it comes to open-source NIDS and NIPS tools, Suricata and Snort [33] are currently the two main competitors. We chose to set up Suricata in our environment because of better performance and scalability [34]. Moreover, we had on-site expertise of one of the core Suricata Developers.

4.2 Simple Event Correlator (SEC)

SEC [30] is an event correlation tool for streamed event processing. It processes a stream of events, to detect and act on certain event groups that occur within predefined time windows, in a way analogous to the ADAM module. SEC is platform-independent (written in Perl), and runs as a single process. SEC was used as a main aggregator that connected the IDS tools, logs and other inputs to EVE. We have open-sourced our SEC ruleset in GitHub under the name of frankenSEC [35].

4.3 Defensive cyber deception techniques

Web honeytokens (decoy URLs) were one of the used deceptive techniques. Red team access to these decoys was monitored by tailing web server logs and comparing the result to lists of honeytokens using our Python script [36]. Unique web pages were served by a custom WebDAV server, which is and which usage in earlier cyber exercises is briefly described in [3]. During the exercises, the server was modified to reply not only to normal HEAD, GET and POST request methods, but also to PUT, TRACE, PROPFIND, TRACK, INDEX, DEBUG, and OPTIONS. When access to honeytokens was detected, additional information was obtained from a lookup dictionary (see Table 1 for an example), and a security event was raised and sent directly to EVE. As with any other IDS tools, the events could have been written to Syslog and forwarded via aggregators to EVE. However, we preferred to follow a simpler path and notify EVE directly.

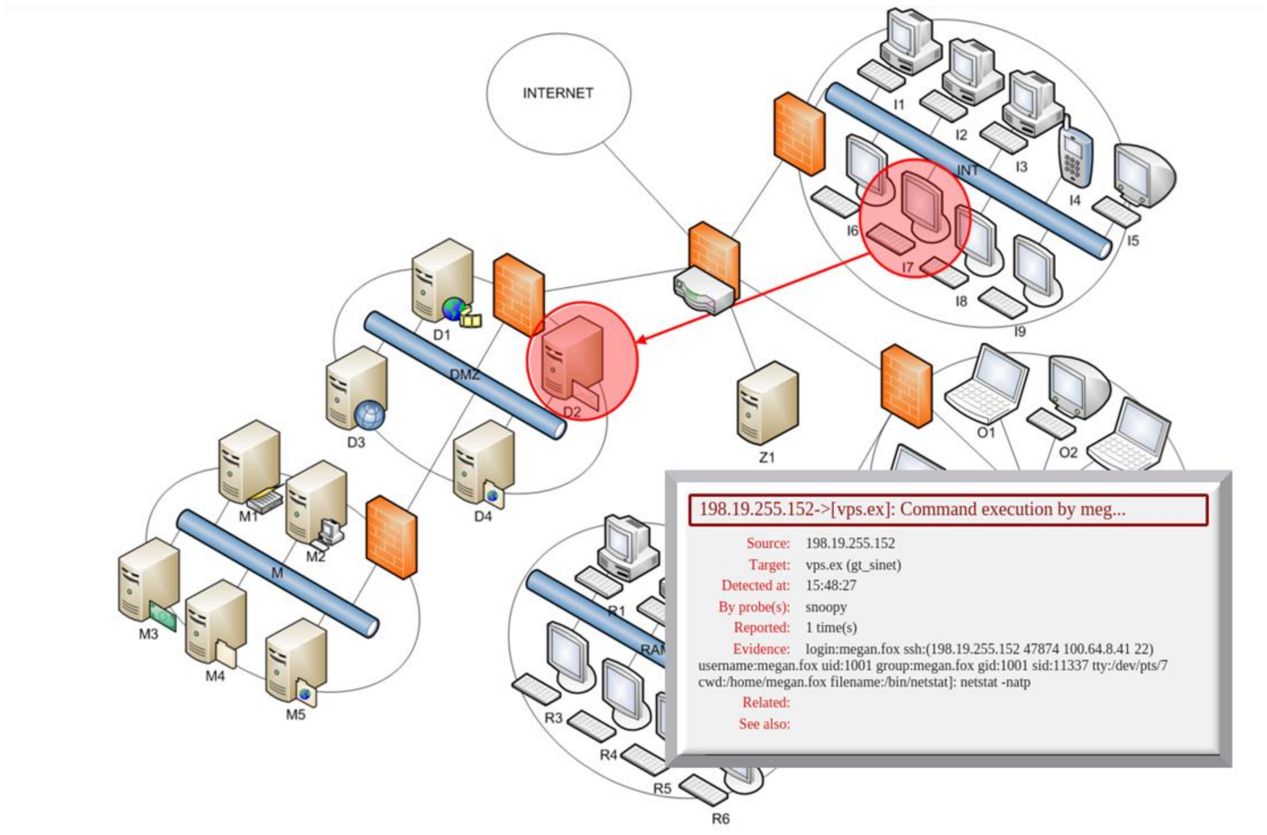
Generation and distribution of baits and decoys were mostly automated using Python and Bash, or SSH and SaltStack, but some were set up manually (for example, in browsers). Some of the baits were stored inside encrypted or obfuscated files, as presented also in our recent paper [37]. Various types of baits were stored in both real devices and honeypots, but for keeping the exercise more exciting, we shall not reveal more locations, except for the command histories mentioned in Table 1. Location “everywhere” in Table 1 means that the same bait was stored in several multiple places in the machine. User “megan.fox” is a real user account in real devices, and users starting with “cowrie” are only inside Cowrie honeypot. For example, “cowrie-all” means that the bait was setup to all user accounts. The last line in the example dictionary means that a bait would have been stored into browser histories of all users in DMZ network.

Table 1: Examples of decoy URL dictionary

URL decoy token	**User(s) having the bait to URL decoy token**	**Location of the bait inside the device**	**IP of machine or network segment**	**Machine name or network segment**
auth/crmsnia/f/login/sqleditor.jsp	host-root	bash_history	123.3.5.123	D1.DMZ
admin/brllia/sql/f/tools/dump.asp	cowrie-celine.dion	files	222.2.4.222	I5.INT
setup/apache/configs/admin.php	megan.fox	everywhere	222.2.4.5	I1.INT
configuration-panel/ins/chck.html	cowrie-all	files	14.3.3.4	M2.M
IIS/web/5rvlia/2/admin/login.php	all	browser_history	123.3.5.0	DMZ

5 RESULTS AND DISCUSSION

Figure 1 presents an example of EVE’s visual representation. On a given network map, two red circles are drawn around the source (I7) and target (D2) of the attack, and a red arrow connects them. Superimposed on the image we can find an informative popup that shows details about the current alert. As soon as a new alert comes in, the circles and arrows are redrawn, and the popup is automatically refreshed. Other parts of the web interface, not shown here, allow analysts to view past alerts by clicking on a list containing name and time. Additionally, users can choose to “replay” past alerts condensed to a given time span, e.g., a five-minute review of a full day of attacks. Due to the confidential nature of NATO CCDCOE cyber exercises, we cannot provide EVE’s screenshots of real network maps used in the exercises.



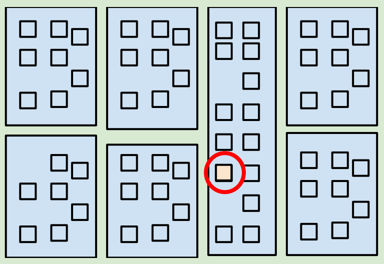
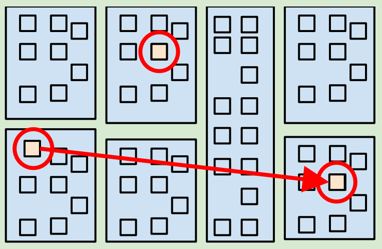
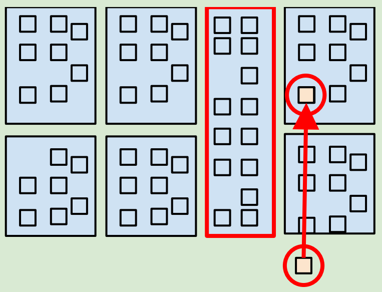
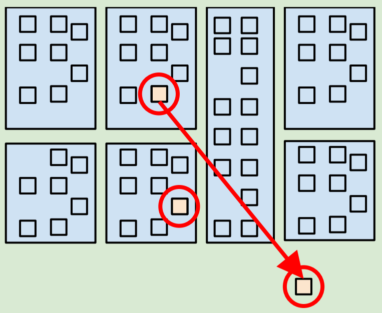
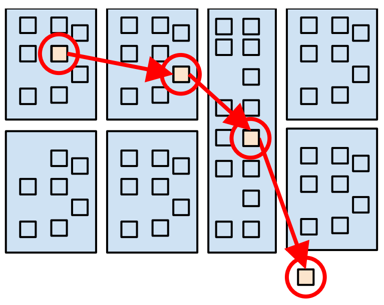
198.19.255.152->[vps.ex]: Command execution by meg...

Source:	198.19.255.152
Target:	vps.ex (gt_sinet)
Detected at:	15:48:27
By probe(s):	snoopy
Reported:	1 time(s)
Evidence:	login:megan.fox ssh:(198.19.255.152 47874 100.64.8.41 22) username:megan.fox uid:1001 group:megan.fox gid:1001 sid:11337 tty:/dev/pts/7 cwd:/home/megan.fox filename:/bin/netstat]: netstat -natp
Related:	
See also:	

Figure 1: Simplified EVE UI representation

Table 2 shows schematic representations of how EVE would present information under some common use cases. It is worth noting that the Table 2 does not include all possible use cases, but the most common ones.

Table 2: Examples of EVE use cases

Use case	**EVE outcome**	**Schematic visualisation**
Malicious internal activity within a given server is detected by one or more sensors	The server is highlighted.	
A user tries to log into a server T (target) from another server O (origin) using bait decoy credentials stored in D (decoy such as honeypot).	All three parties (T, O and D) are highlighted. An arrow is drawn from O to T.	
Attack from external server O, to workstation D. At the same time, a network has been compromised (or internal IP scan was detected)	Highlight D, O and network and draw arrow from O to D.	
Leakage of information located in server T (detected by data loss prevention tools), using credentials stolen from D. Data is transferred from D to O.	Highlight all three servers. Draw an arrow from O to D.	
Information exfiltrated via tunnels between various machines in different network segments (it is likely that adversary has access to all machines used in the exfiltration).	All devices participating in the exfiltration are highlighted and arrows are drawn between them. Note: This functionality is not yet available, but it is a work in progress. Workaround solution is to draw multiple alerts in succession.	

6 POSSIBLE IMPROVEMENTS

Several improvements can be incorporated to make EVE a more useful alerts visualisation tool, and we are currently planning to tackle them in future upgrades. We will outline the most relevant here.

6.1 Improving the ADAM module

The ADAM module is responsible for generating the alerts out of the events reported by the sensors. In its present state, ADAM defines two events as equivalent (and therefore, aggregated into the same alert) if they are reported within a timespan of no longer than 5 seconds from each other and they share the same source, target and payload.

However, ADAM could be made *more intelligent* by considering some of the following aspects:

- A dynamic interval between equivalent events. For example, the attacker may choose to run an attack sending payload with an interval longer than 5 seconds. Given that source, destination and payload type of the attack are generally known, ADAM's algorithms could be improved by tuning the number of seconds between two equivalent events as a function of the payload type.
- Currently, if two events have a different target they will not be considered as equivalent. However, some attacks may be directed to two or more different targets while being completely equivalent events. For example, a network scan that is originating from a single source, but directed at a segment with multiple target hosts.
- ADAM could generate more meaningful alerts if they could be labelled according to a given taxonomy. For example, making use of categories like "web attack - PHP injection", "web attack - XSS", etc. Essentially, it all amounts to sharing the taxonomy with the sensors that supply the information and making sure that those sensors include the category and possible subcategories within each event notified to EVE.
- ADAM could possibly produce graded alerts (for example, on a scale of 1 to 5) according to variables such as destination target, risk incurred and payload. In addition to this, when using cyber deception for defense, techniques presented in [37] could be used to give more details about criticality of breaches.

6.2 Improving the visualisation module

The visualisation aspects could be improved in a number of ways:

- Adding the option of choosing the alert that the user wants to have displayed according to their grade or category.
- In some scenarios it may be interesting to represent attack phases. EVE should then be able to highlight related hosts simultaneously and draw several arrows for some given alerts. This would allow visualising more detailed breach event descriptions, such as the ones presented in [37].
- Making use of visualisation aids like shadows, gradients and fading. For example, keeping past alerts using lighter colours while maintaining the current one depicted in a bright colour.

6.3 Other potential improvements

- In addition to being a mere display tool, EVE could send a notification (e.g., email, SMS, IRC or Slack message) to predefined recipients every time an alert of a given type or level is raised. Those notifications can be further improved by narrowing the recipients of the notification according to additional criteria. For example, notification of alerts on a given target only to those responsible for the management of that target. This functionality can be easily achieved by coding the corresponding module or by using an existing tool such as Alerta CLI [38].

- EVE could be coupled to some reporting tool or library to generate all sort of reports regarding incident typology, timing, payload type, targets, etc.

7 FUTURE WORK

NATO CCDCOE's follow-up Hackathon event was held in December 2017 [39]. It was primarily used to explore the possibility of improving EVE by offering additional functionality aimed at red team participants. Although EVE was meant as a tool for cyber security alerts visualisations, it will be interesting to add an alternative view using the same incoming data. This would allow the Red team to see the attacks as they are detected by the security sensors near real time. The view will start out as a blank web page, with no information relative to the underlying network map (which will always remain hidden). As red team attacks proceed, events come into the system, and alerts are processed, a network map will unfold automatically, revealing more as the exercise progresses. The purpose of this view is twofold: as an educational tool, with which Red team members can learn how monitoring sensors detect their activity; and as a tool that documents their activity by depicting the network using information gathered from their own attacks. Additionally, the infrastructure team could compare this dynamically generated network map with the real map. In a real-world scenario, it is generally expected that some differences may arise between both maps, as network maps can be inaccurate and usually do not show long-forgotten servers and workstations.

The tool is currently meant to be used in static environments with known network maps. However, considering that network maps might not be accurate, EVE's functionality would be greatly improved if it were able to modify the map, so that new devices, connections, network segments, or services are added as soon as they are detected by sensors. This functionality is intended for Green team members only, as it reveals the network map.

EVE, and its associated module ADAM, will be used again during the next editions of NATO CCDCOE cyber exercises, which will provide further tests and opportunity for improvements and bug fixing.

ACKNOWLEDGEMENTS

The authors of this paper thank Hillar Aarelaid for his initial ideas and insights for the need for such a visualisation tool. We also thank Markus Kont for infrastructure build-up and Eric Leblond for Suricata development and optimizations. Furthermore, we also would like to acknowledge the collaboration of Jalmar Jerlei, which allowed us to explore the possibility of dynamic network maps.

AUTHORS' CONTRIBUTIONS

F. Jesús Rubio contributed with the final functional definition and full implementation of EVE [28]. Teemu Väisänen contributed with literature review and information about deception techniques; baits, decoys and related sensors. Mauno Pihelgas contributed to the literature review, provided additional requirements for EVE and was responsible for the implementation of the event correlation ruleset frankenSEC [35].

REFERENCES

[1] V. T. Guimarães, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco and L. Z. Granville, "A Survey on Information Visualization for Network and Service Management," *IEEE Communications Surveys & Tutorials,* vol. 18, no. 1, pp. 285-323, 2016.

[2] S. McKenna, D. Staheli, C. Fulcher and M. Meyer, "BubbleNet: A Cyber Security Dashboard for Visualizing Patterns," *Computer Graphics Forum,* vol. 35, no. 3, pp. 281-290, 2016.

[3] T. Väisänen, L. Trinberg and N. Pissanidis, I accidentally malware - what should I do… is this dangerous? Overcoming inevitable risks of electronic communication, Tallinn, Estonia: NATO CCD COE, 2016.

[4] NATO CCDCOE, "Locked Shields 2017," 2017. [Online]. Available: https://ccdcoe.org/locked-shields-2017.html. [Accessed 20 March 2018].

[5] NATO CCDCOE, "Crossed Swords Exercise," 2018. [Online]. Available: https://ccdcoe.org/crossed-swords-exercise.html. [Accessed 20 March 2018].

[6] M. Kont, M. Pihelgas, K. Maennel, B. Blumbergs and Lepik, "Frankenstack: Toward Real-time Red Team Feedback," in *MILCOM 2017 - IEEE Military Communications Conference*, Baltimore, 2017.

[7] A. W. Luse, Exploring utilization of visualization for computer and network security, Iowa State University, 2009.

[8] G. Markowsky and L. Markowsky, "Visualizing Cybersecurity Events." In Proceedings of the International Conference on Security and Management (SAM)," in *The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp),* 2013.

[9] J. Skurek, "A Survey of Tools for Monitoring and Visualization of Network Traffic," 2015. [Online]. Available: https://is.muni.cz/th/410312/fi_b/skurek_thesis_final.pdf.

[10] G. A. Fink, C. L. North, A. Endert and S. Rose, "Visualizing cyber security: Usable workspaces," in *VizSec 2009 - 6th International Workshop onVisualization for Cyber Security*, 2009.

[11] J. J. Fowler, T. Johnson, P. Simonetto, M. Schneider, C. Acedo, S. Kobourov and L. Lazos, "IMap: Visualizing network activity over Internet maps," in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, 2014.

[12] H. Koike, K. Ohno and K. Koizumi, "Visualizing cyber attacks using IP matrix," in *IEEE Workshop on Visualization for Computer Security (VizSEC 05),* 2005.

[13] Y. Hideshima and H. Koike, "STARMINE: a visualization system for cyber attacks," in *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation (APVis '06),* Darlinghurst, Australia, 2006.

[14] O.-M. Latvala, T. Keränen, S. Noponen, N. Lehto, M. Sailio, M. Valta and P. Olli, "Visualizing network events in a muggle friendly way," in *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA),* London, 2017.

[15] F. Mansmann and S. Vinnik, "Interactive Exploration of Data Traffic with Hierarchical Network Maps," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, no. 6, pp. 1440-1449, 2006.

[16] S. Foresti, J. Agutter, Y. Livnat, S. Moon and R. Erbacher, "Visual correlation of network alerts," *IEEE Computer Graphics and Applications,* vol. 26, no. 2, pp. 48-59, 2006.

[17] N. Satterly, "Alerta," [Online]. Available: http://alerta.io/. [Accessed 20 March 2018].

[18] D. Arendt, D. Best, R. Burtner and C. L. Paul, "Cyberpetri at CDX 2016: Real-time network situation," in *IEEE Symposium on Visualization for Cyber Security (VizSec 2016),* 2016.

[19] D. L. Arendt, R. Burtner, D. M. Best, N. D. Bos, J. R. Gersh, C. D. Piatko and C. L. Paul, "Ocelot: user-centered design of a decision support," in *IEEE Symposium on Visualization for Cyber Security (VizSec 2015),* 2015.

[20] L. Williams, R. Lippmann and K. Ingols, "GARNET: A graphical attack graph and reachability network evaluation tool," in *Visualization for Computer Security,* 2008.

[21] The Honeynet Project, "HoneyMap - Visualizing Worldwide Attacks in Real-Time," [Online]. Available: http://www.honeynet.org/node/960. [Accessed 20 March 2018].

[22] BruteForce Lab, "Kippo-Graph," [Online]. Available: http://bruteforcelab.com/kippo-graph. [Accessed 20 March 2018].

[23] Deutsche Telekom AG Honeypot Project, "T-Pot - Multi-Honeypot Platform rEvolution," [Online]. Available: https://dtag-dev-sec.github.io/mediator/feature/2017/11/07/t-pot-17.10.html. [Accessed 20 March 2018].

[24] Elastic, "Kibana," [Online]. Available: https://www.elastic.co/products/kibana. [Accessed 20 March 2018].

[25] Vanimpe, "Using ELK as a dashboard for honeypots," [Online]. Available: https://www.vanimpe.eu/2014/12/13/using-elk-dashboard-honeypots/ . [Accessed 20 March 2018].

[26] Graylog Inc., "Graylog," [Online]. Available: https://www.graylog.org/. [Accessed 10 April 2017].

[27] The MITRE Corporation, "Common Event Expression," [Online]. Available: http://cee.mitre.org/. [Accessed 20 March 2018].

[28] NATO CCDCOE, "Events Visualization Environment," 2017. [Online]. Available: https://github.com/ccdcoe/EVE. [Accessed 20 March 2018].

[29] B. Skufca, "Snoopy Logger," [Online]. Available: https://github.com/a2o/snoopy. [Accessed 20 March 2018].

[30] R. Vaarandi, "SEC - simple event correlator," [Online]. Available: https://simple-evcorr.github.io/. [Accessed 20 March 2018].

[31] Open Information Security Foundation, "Suricata," [Online]. Available: https://suricata-ids.org/. [Accessed 20 March 2018].

[32] The Bro Project, "The Bro Network Security Monitor," [Online]. Available: https://www.bro.org/. [Accessed 20 March 2018].

[33] Cisco, "Snort," [Online]. Available: https://www.snort.org/. [Accessed 20 March 2018].

[34] M. Pihelgas, A Comparative Analysis of Open-Source Intrusion Detection Systems, Tallinn: Tallinn University of Technology & University of Tartu, 2012.

[35] NATO CCDCOE, "frankenSEC," 2017. [Online]. Available: https://github.com/ccdcoe/frankenSEC. [Accessed 28 November 2017].

[36] T. U. Väisänen, "honeystuff," 2016. [Online]. Available: https://github.com/uolevi/honeystuff/blob/master/logparser.py. [Accessed 20 March 2018].

[37] T. Väisänen, "Categorization of cyber security deception events for measuring the severity level of advanced targeted breaches," in *Proceedings of ECSA '17,* Canterbury, United Kingdom, 2017.

[38] N. Satterly, "Alerta CLI," [Online]. Available: http://alerta.readthedocs.io/en/latest/cli.html. [Accessed 20 March 2018].

[39] NATO CCDCOE, "Frankencoding," 2016. [Online]. Available: https://github.com/ccdcoe/Frankencoding. [Accessed 28 November 2017].