STARLORD: Linked Security Data Exploration in a 3D Graph

Laetitia Leichtnam, Eric Totel, Nicolas Prigent, Ludovic Mé

STARLORD : Linked Security Data Exploration in a 3D Graph

Laetitia Leichtnam[1], Éric Totel[1], Nicolas Prigent[2], and Ludovic Mé[1]

[1]CentraleSupelec (Cidre Team, UMR IRISA, joint team with Inria, CNRS and University of Rennes 1)
[2]LSTI

ABSTRACT

In this paper, we present a novel modelization and visualization approach for heterogeneous sources of data. We represent our data by using a model inspired by STIX. Then, we use clustering algorithms to select interesting information to explore in a visualization panel. The visualization is based on a 3D graph representation that highlights the link between malicious event and allows to focus on relevant security artifacts. We illustrate our approach with two case studies using datasets containing network capture of the wannacry attack.

Index Terms: C.2.3 [Computer-Communication Networks]: Network Operations—network management, network monitoring;

1 INTRODUCTION

Cyber analysts investigate incidents by correlating heterogeneous pieces of information coming from multiple sources. In this paper, we propose a visual tool that helps cyber analysts exhibit relations between the security-related events and more specifically events from the logs produced by the various sources in the monitored system.

To that end, we build a graph-based modelization of security data to emphasize the relations between them. Then, we partition our graph nodes into clusters. This allows to select suspicious clusters and to present a relevant visualization of the data items. The visualization is interactive in order to help navigation in the data graphs.

This paper is structured as follows. First, we present the data sources in Section 2 and the data model in Section 3. Section 4 and 5 describe how nodes are automatically merged and clustered. Section 5 presents the prototype and the design choices we made. Section 7 shows the interest of our approach on real world examples. Our approach is discussed and future works are presented in section 8. Related works are identified in section 9.

2 DATA SOURCES

Cyber analysts have to discover malicious events, establish their criticality and rebuild the global attack scenarios based on large amounts of raw information having little *a priori* relations between them. While very heterogeneous, the data sources at hand can be split into three categories: static internal sources, dynamic internal sources and external sources.

Static Internal Sources constitute the static description of the supervised infrastructure. They are used to grasp the attacks criticality and to have an overview of the internal system. We can for example cite the network cartography, operating systems and software versions, criticality of the infrastructure, internal incident reports.

Dynamic Internal Sources represent the system state in real-time. They constitute the traces of the activity in the monitored system. Examples are Intrusion Detection System (IDS) logs, firewall logs, system logs, application logs and network captures.

External Data Sources are used to correlate the internal events with known attacks. We can cite for example, Indicator of Compromise (IOC) databases such as OTX [1] or MISP [3], Common Vulnerability Exploit(CVE) Databases [12], external incidents reports, malware database, Intrusion Detection System (IDS) rules like Emerging Threat Rules [4] and hardware and software editor security reports. These pieces of data can be matched with the items present in dynamic internal sources.

The diversity of cybersecurity sources is an asset but also a challenge. Indeed, multiple sources make it easier to correlate the information but the heterogeneity of the data and the diversity of their respective formats require to first build a model representing all the inputs. In our experiments, and without loss of generality, we used two log families as an input : logs coming from the Bro IDS [15] and IOCs from the OTX database.

3 A MODEL INSPIRED BY STIX

Several formats have been proposed to represent cyber security data such as OpenIOC [11], Structured Threat Information eXpression [14] or Cyber Observable eXpression (CybOX) [13].

Structured Threat Information Expression (STIXv2) is a language used to exchange Cyber Threat Intelligence (CTI). It is graph-based in the sense that it describes the information as a connected graph. STIX domain objects define the graph nodes and STIX relationships define the edges. A relation is a link between STIX Objects that describes the way in which the objects are related. Most relations are represented using STIX Relationship Objects (SROs), while other embedded relationships are represented as ID references. This graph-based language allows for flexible, modular, structured, and consistent representations of CTI.

We chose the STIXv2 language because of its comprehensiveness compared to other models. It can represent a whole campaign of attack but also the targets, the threat actors, the tools used, the vulnerabilities exploited and the responses to the incident where other languages describe only parts of them. Another advantage is that it is based on a graph schema which allows us to apply to the graph mining techniques that will be detailed later.

While STIXv2 is mainly used to exchange attack components information, we use it to describe what happened in our system. We focused on three types of STIX objects: `Observed Data` that describes what was seen in the information system, `Indicator` that describes an artifact that was part of an attack and `Campaign` that describes a set of related malicious activities or attacks. We also used the following CyBOX objects included in STIXv2 to represent our items: `Domain Name`, `File`, `IP Address`, `DNS Query`, `Network Connection`, `Socket Address`, `Hostname` and `Port`.

In our experiments, we used two logs families as an input : logs coming from the Bro IDS [15] and IOCs from the OTX database. Each Bro log is parsed to get valuable items that can be related to an IOC. Each log correspond to an `Observed Data`. Figures 1, 2, 3 and 4 describe the objects and relationships extracted from the various log files. Each figure describes the object and relationships

for one event from the log files. The object will become the nodes of our graph and the relationships the vertices.

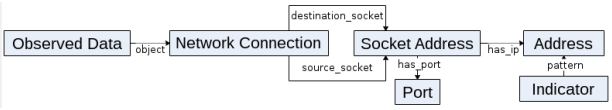


Figure 1: Objects and Relationships extracted from Bro logs conn.log

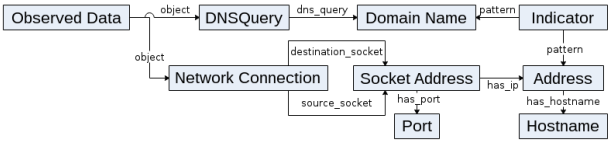


Figure 2: Objects and Relationships extracted from Bro logs dns.log

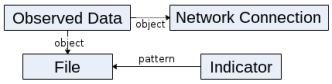


Figure 3: Objects and Relationships extracted from Bro logs files.log

While many IOC are available, we decide to focus on `Domain`, `Hostname`, `IPv4`, `IPv6` and `FileHash` as security elements because they can all be found in our Bro logs and can easily be matched with the CybOX objects from our model. For each IOC, we seek in the object build from our logs if we have a match and we create an `Indicator` object and a relationship between the `Indicator` created and the object. The global description of our model is shown in figure 5.

4 Merging Node

Given that some observed data share some common aspects (same source address, same destination port, etc.), some objects corresponding to the same observed data are present multiple times in our logs. To create relations between observed data, we delete all but one instance each object and translate all the relations targeting or coming from all the identical instances of this object to the one we keep.

As an illustration, we can consider what happens when merging a connection log and an notice log. The notice log shares the attributes connection uid, source address, destination address, source port and destination port with the connection log. The automatic merging operation therefore deletes the objects `Network Connection`, `Socket` , `Address`, `Port` created from the alert log and link the object Indicator to the object `Network Connection` of the connection log. We can view the result in figure 6.

In a traditional data model, security items would have been represented as object attributes which force to search in all objects to retrieve one artifact. The advantage of this representation is to focus on one particular object and directly see all the objects that are related to it.

5 Clustering

In our proposal, since we have no *a priori* information about the observed data, all relations are considered equals and no weight is applied to the links. The Louvain algorithm [2] is a method to extract clusters from large networks. It maximizes the modularity function in a partition. In other words, it ensures that the number of links (since all links have the same weight) are more important in the partition than between the partitions.

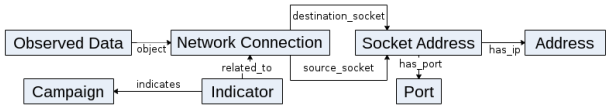


Figure 4: Objects and Relationships extracted from Bro logs notice.log

We apply this algorithm to our graph. Once the clusters are built, we declare IOC and the NIDS alerts as *values of interest*. The clusters containing such value of interest are selected to appear in the representation since they are actually exploitable. At this point all clusters are disjoint. We search all direct links between the selected clusters in the initial graph to help the analyst to see if there is a relevant relation between two points of interest located in two different clusters and create these links so as to group the clusters that can be grouped.

Depending on the attack, the importance of each relation could vary. We let the analyst investigate on the importance of each link according to its semantic and make an interpretation of it.

6 Prototype

In this section, we present our prototype. We first describe the application architecture. Then, we present the design choices we made. Finally, we present the various interactions the analyst can have with the tool.

6.1 Application Architecture

Our prototype is made of four elements. The *data collection and modelization module* collects data from multiple sources and parse them into parameters. Objects and Relationships derived from these parameters are then created, merged and integrated in a *Graph DB database* that allow simple and fast retrieval of complex hierarchical structures that are difficult to model in relational databases. Then, the *clustering module* extracts the nodes which are points of interest. Finally, the *visualization interface* displays the elected clusters and allows interactions with the user to ease exploration.

6.2 Design

In our early experiments, we first tried to generate 2D graphs representations. However, we had to deal with the fact that numerous lines were crossing one another and that this representation was very difficult to understand. As a consequence, we decided to create 3D graphs representations using Unity [17] even if we were aware of the the well-known occultation problem they can cause.

In our representation, nodes of the graph are represented as colored sphere. Each color represents a class object : red for `Campaign`, yellow for `Indicator`, blue for `IP Address`, violet for `Socket Address`, rosa for `Network Connection`, purple for `Port`, turquoise blue for Domain, ligth turquoise blue for `DNS Query`, white for `Observed Data` and black for `File`. The palette used is designed for being used by color-blind persons [10].

We used a force-directed graph drawing algorithm to generate our representations [8]. This algorithm assigns forces among the set of edges and the set of nodes, according to their relative positions and then uses these forces to simulate the motion of the nodes and edges. Once the forces on the nodes and edges of a graph have been defined, the behaviour of the entire graph is simulated as if it was a physical system: the forces are applied to the nodes, pulling them closer together or pushing them further apart. This is repeated iteratively until the system comes to a mechanical equilibrium state, i.e. when their relative positions do not change anymore from one iteration to the next. As a result, all the edges are of more or less equal length and there are as few crossing edges as possible [9].

Figures 7 and 8 show examples of generated representations. First of all, our proposal allows the operator to easily compare the relative

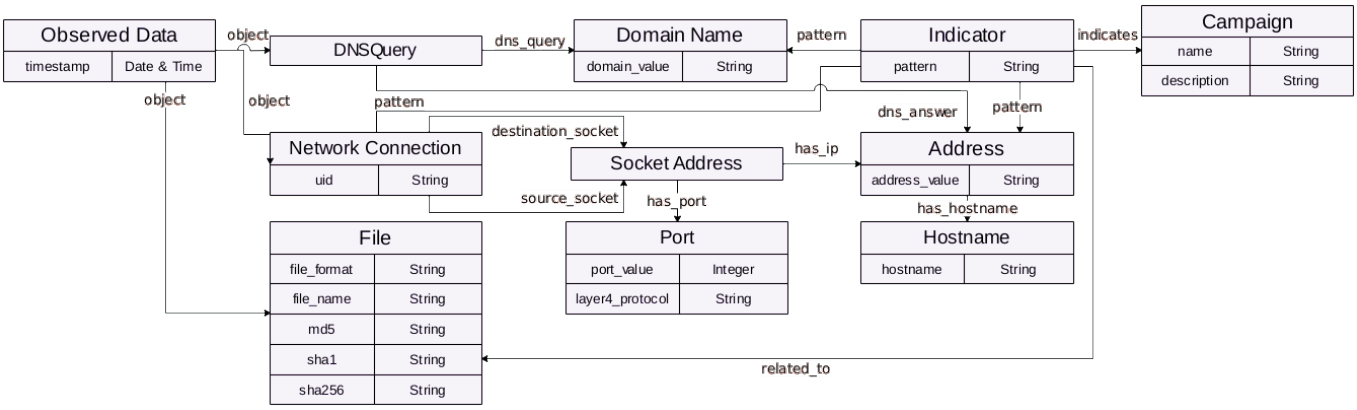


Figure 5: Modelization of the security elements

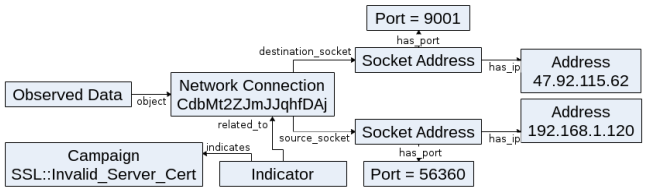


Figure 6: Objects and Relationships fusioned between conn.log et notice.log

density of different parts of the graph as well as to see if a point of interest is involved in a lot of events. Some specific patterns can also be easily identified. For instance, the ”furball” on Figure 7 allows the analyst to see easily that the value of interest in the center was involved in a lot of events (in this case, this is port `TCP 445`). Finally, our proposal also allows to follow a potential propagation by finding a path between objects. We have to underline here that links in the graph have very various semantics and therefore that the solution leaves the analyst interpret them freely (while still being able to veryfing in the log files for the specific meaning of each of them). While this could be hazardous, our early experiments showed that it allowed a more ”open-minded” exploration by analysts.

We also took advantage of the features of Unity to allow the analyst to navigate the data. He or she can zoom and move around the data. The analyst can also click on any node to obtain more information on the related point of interest. Finaly, the analyst can also create new nodes and links in order to enrich the data and answer to “What-if” question.

7 Case Study

In this case study, we use two datasets made of logs from Bro IDS that were generated in the Stratosphere Labs [5] as part of the Malware Capture Facility Project in CVUT University, Prague, Czech Republic. The objective of this project is to store long-lived real botnet traffic and to generate labeled netflows files. Table 1 summarizes the objects and relations extracted from the datasets.

7.1 Wannacry dataset 1

The CTU-Malware-Capture-Botnets-252-1 dataset describes the wannacry attack. The capture lasted 501 seconds. Bro output files were generated from this capture. The infected host is 192.168.1.120. The ransomware connected to the killswitch domain during the capture. It did not encrypt files but tried to infect other computers. 5174 connections were logged in Bro files. There were 8 DNS requests

Table 1: Objects and Relationships extracted from the datasets

Object & Relationship	Wannacry 1	Wannacry 2
Address	4969	14373
AttackCampaign	1	6
DNSQuery	8	16
Domain	3	3
File	0	7
Indicator	1	3
NetworkConnection	5174	15298
ObservedData	5174	15298
Port	4944	10540
SocketAddress	9935	24999
Total Objects	30209	80543
Total Relationships	35418	95948

including 4 requests to the killswitch domain.

In the visualization of the selected clusters in figure 7, we can see on the left the representation of the address scan on port 445/tcp. This port is represented at the middle. On the right of the graph, the IOC representing the killswitch is linked with other objects from its clusters.

Our visualization was able to select 616 interesting nodes out of 30209 and 642 interesting edges out of 35418. The generated representation shows the scan and allows the operator to focus on object highly related to the IOC detected in the data. We investigated why two separated graphs were generated. Looking at the raw logs, we discovered that the DNS request was made by an IPv6 address whereas the scan came from an IPv4 address and was targeting IPv4 addresses.

7.2 Wannacry dataset 2

CTU-Malware-Capture-Botnets-253-1 also describes a wannacry attack. This time due to a problem in the DNS server provider, the domain did not resolved to an IP and therefore the infection went on. The capture lasted 895 seconds. The infected host was 192.168.1.120. This capture used a mitm proxy on all TLS ports and a redirection to a honeypot on port 445/tcp.

In this capture, thanks to our proposal, we detect the killswitch `www.ifferfsodp9ifjaposdfjhgosurijfaewrwergwea.com`. We also discover two malicious addresses, 193.23.244.244 and 81.30.158.223 that are associated to the wannacry campaign. The representation also shows a scan on port 445/tcp. We focus in Figure 8 on clusters with the killswitch and IOCs. In the visualization we can see that the point of interest are concentrated on three clusters. The first one concentrates two alerts and one IOC referring to an

Figure 7: Representation of selected clusters in dataset 1

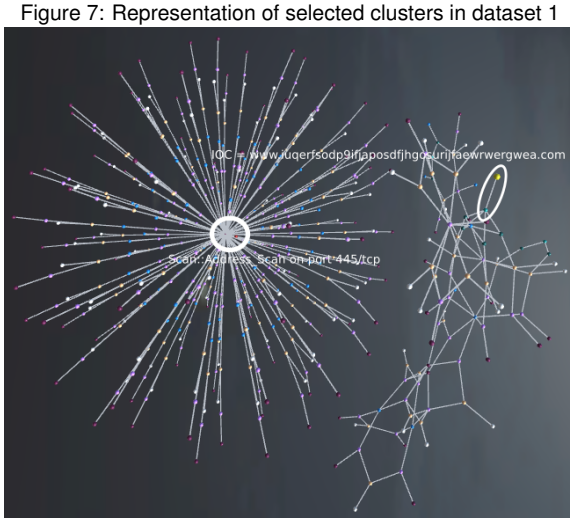
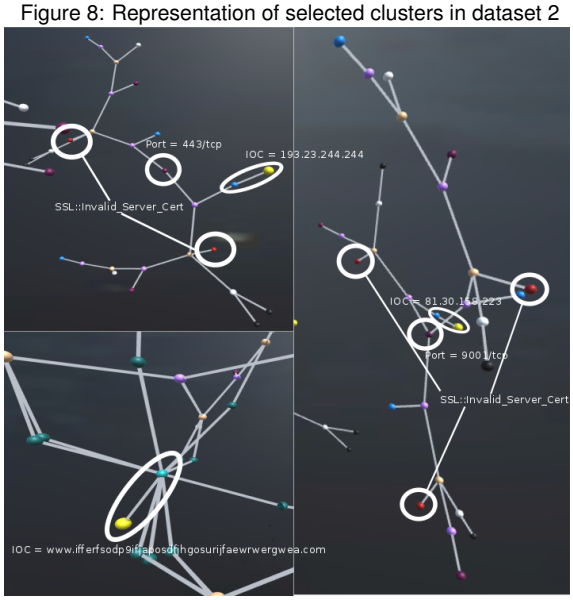


Figure 8: Representation of selected clusters in dataset 2



IP Address, the second concentrates 3 alerts and one IOC, also referring to an IP Address. The last one, but also the more dense contains only one IOC, the killswitch. This domain is directly linked with 9 `DNSResquest`. By following paths sourcing from the malicious DNS request, two more DNS request coming from the same socket can be identified.

8 Discussion and Future Work

Our proposal allows the analyst to explore the data and to interpret results. He or she can see the object shared between two logs and try to rebuild an attack chain. The objects selected are relevant and can direct the operator for his or her first analysis. Some pattern can be recognized such as the scan or a conversation between two IP addresses on the same port.

Some limitations coming from the clusters selection remain. Indeed, the clusters created by the Louvain algorithm are disjointed: an object can not belong to two clusters and links between clusters are suppressed. As we select only interesting clusters (i.e. clusters for which we have at least one IOC), we can lose some interesting object present in a cluster that was not selected. To avoid this, we can introduce the notion of "bridge node" that is to say nodes from which we can go to another cluster even if it is not selected. The second point of evolution is the scalability. We made visualization for bro logs and IOC in our case study. The next step is to integrate other types of data and try our model on bigger datasets.

9 Related works

Other work have addressed exploration of IDS alerts. Among them, [16] proposes to build an automated multi-stage log-based intrusion analysis through the use of community discovery algorithms. The tool, named HERCULE, builds multi-dimensional weighted graphs by correlating log entries. Our work split the log entries within multiple objects and put all the semantic on the object and not on the relation. By contrast, our proposal also includes external sources such as IOC and proposes a visual representation for exploration.

Force-directed graphs have also been used previously for security purposes. In [6], the authors propose to visualize network traffic using force-directed graph drawing techniques. Data points are connected by times, displaying sequences of events with similar characteristics. The simulation is based on one type of log: firewall logs. Our proposal, by contrasts, establishes relation between heterogeneous sources of data.

Finally, an other proposal was made to correlate heterogeneous data sources. In [7] for instance, the authors designed a security-oriented log visualization tool, CORGI, that allows security experts to visually explore and link numerous types of log files through multiple representations.

10 Conclusion

In this paper, we presented a new tool that allows operators to explore security data. It is based on a graph representation of attributes coming from heterogeneous sources and notably Threat Intelligence. The visualization highlights the relations between security elements and malicious event or IOC. It gives a starting point for the analysts to explore the data and rebuild an attack scenario by following the links between the nodes. We demonstrated the interest of this visualization using two datasets containing network captures of the wannacry attack.

References

[1] Alienvault. AlienVault Open Threat Exchange, 2017.
[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks.
[3] CIRCL. Malware Information Sharing Platform, 2017.
[4] Energing. Emerging Threat Rule, 2017.
[5] S. Garcia. Malware Capture Facility Project, 2017.
[6] L. Girardin and D. Brodbeck. A visual approach for monitoring logs. In *LISA*, vol. 98, pp. 299–308, 1998.
[7] C. Humphries, N. Prigent, C. Bidan, and F. Majorczyk. Corgi: Combination, organization and reconstruction through graphical interactions. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, VizSec '14, pp. 57–64. ACM, New York, NY, USA, 2014.
[8] S. G. Kobourov. Force-directed drawing algorithms. 2004.
[9] S. G. Kobourov. Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011*, 2012.
[10] M. Krzywinski. 15-color palettes for color blindess, 2017.
[11] Mandiant. The OpenIOC Framework, 2017.
[12] MITRE. Common Vulnerabilities and Exposures, 2017.
[13] Oasis. Cyber Observables Expression, 2017.
[14] Oasis. Structured Threat Information eXchange, 2017.
[15] V. Paxson. The Bro Network Security Monitor, 2017.
[16] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu. Hercule: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ACSAC '16, pp. 583–595. ACM, New York, NY, USA, 2016.
[17] Unity Technologies. The OpenIOC Framework, 2017.