Rethinking the Graph Visualization for Threat Reports

Mayo YAMASAKI
NTT Secure Platform Laboratories

Abstract

Understanding threat intelligence is not an easy task for analysts even if they are structured and machine-readable. Therefore, many methods to visualize the threat intelligence structure have been proposed. However, these methods utilize visualization methods developed for the general domain to support a wide variety of use cases for analyzing threat intelligence.

This paper introduces a novel visualization method, for a threat report, based on simple observations obtained by a study of threat diagram characteristics of actual threat reports. Because threat report is a reasonable bundle of intelligence and one of the most common ways to share it, by capturing these characteristics, the method visualizes graph-structured STIX 2.0 as a concise overview of the threat structure. Concretely, most of the diagrams use DAG (Directed Acyclic Graph) to represent attack flows, emphasize relationships between IoCs and other entities, and focus on differences from existing intelligence.

This paper shows the characteristics of diagrams on threat reports and observations obtained from them. Also, this paper describes how simple these observations improve the visibility of complex threat intelligence by proposing a visualization method for graph data converted from STIX 2.0, and this paper demonstrates the utility by visualizing actual threat reports gathered from the ATT&CK knowledge base.

1 Introduction

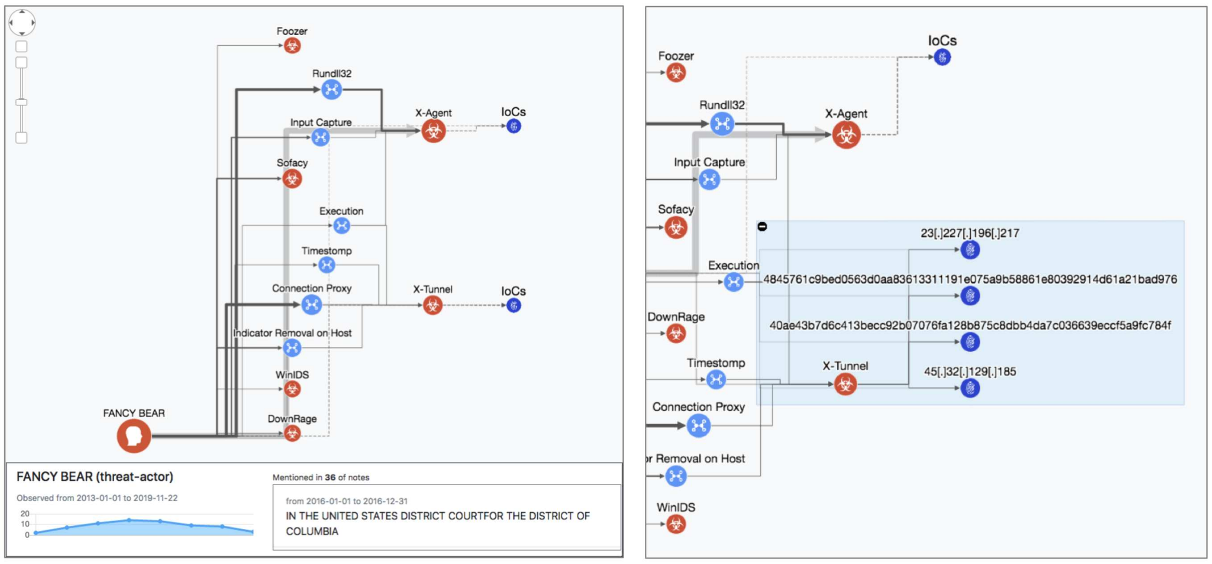
To effectively handle threat intelligence, analysis capabilities including visualization are required. However, there are limitations on the current capabilities of TIPs (Threat Intelligence Platforms), thus, the value of threat intelligence depends on analyst abilities to handle it [3,7]. Previous works use general graph layout methods for STIX (Structured Information Sharing Platform) [5,6] to support a wide variety of use cases to analyze threat intelligence [2,4]. Also, graph data of the STIX 2.0 or other knowledge models are usually the dense graph because most node combinations have some semantic relationship. For these reasons, it is difficult to understand the structures of the threat graph. This paper, in contrast, considers a method to visualize graph data for a particular use case that is representing threat graphs on a threat report that usually mentions to one or a few campaigns. The reason for choosing this particular use case is that threat report is a reasonable bundle of threat intelligence, and it's one of the most common ways to share it in the security communities.

This paper briefly summarizes a study of characteristics of threat diagrams in open-source reports and proposes a novel method to visualize graphs on a threat report, based on observations of the study. The proposed method visualizes threat graphs leveraging a hierarchical layout for the DAG (Directed Acyclic Graph) and, subdue complexities by dealing with redundant edges using both orthogonal edge routings and de-emphasizing cross-layered edges. Further, to simplify graph structure, the method conducts clustering IoC nodes based on a set of edges to preserve relationships with other entities. Finally, structure differences in the report are emphasized by visualizing the number of mentions in the set of threat reports. Figure 1 shows an example of the method.

The contributions of this paper are as follows.

1. The study of threat intelligence diagrams on open-source reports shows why and how these diagrams are created and also characteristics of diagrams that illustrate structures of threat intelligence. The results can be used to design new visualization methods.

2. This paper proposes a novel visualization method to automatically visualize graph-structured threat intelligence on a report. Hence, by providing a concise overview of the structure to analysts, this method contributes to improving analysis capabilities on any TIPs.


Foozer
Rundll32
Input Capture
X-Agent
IoCs
Sofacy
Execution
Timestomp
Connection Proxy
X-Tunnel
IoCs
Indicator Removal on Host
WinIDS
FANCY BEAR
DownRage
FANCY BEAR (threat-actor)
Observed from 2013-01-01 to 2019-11-22
20
10
0
Mentioned in 36 of notes
from 2016-01-01 to 2016-12-31
IN THE UNITED STATES DISTRICT COURTFOR THE DISTRICT OF COLUMBIA
IoCs
Foozer
Rundll32
X-Agent
Input Capture
Sofacy
23[.]227[.]196[.]217
Execution
4845761c9bed0563d0aa83613311191e075a9b58861e80392914d61a21bad976
DownRage
40ae43b7d6c413becc92b07076fa128b875c8dbb4da7c036639eccf5a9fc784f
Timestomp
X-Tunnel
45[.]32[.]129[.]185
Connection Proxy
r Removal on Host
WinIDS

(a) (b)

Figure 1: Diagrams show one threat report about the Fancy Bear selected from the ATT&CK knowledge base. (a) An overview shows a campaign and its components such as attack patterns and malware. These size of nodes and width of edges indicates each number of reports about the Fancy Bear on the ATT&CK. (b) IoC clusters contain multiple nodes which have same relationsihps.

2 Study of Diagrams on Threat Reports

The goal of this study is to extract characteristics of threat intelligence diagrams. For this study, all images are collected from 83 threat reports published on 8 different websites in three months. Theses reports have 700 images, and most of the images are screenshots to support analysis results. 616 images are used for this purpose. These images include screenshots of web pages, mobile/desktop applications, text editors, binary editors, packet logs, terminals/command prompts, Office/PDF/RTF files, and emails. Only 69 images are used for describing threat intelligence. The rest 15 images describe other information such as mathematical formulas and presentation slides. 32 of 69 images for threat intelligence show some statistics such as geographically mapped targets of attacks, line charts of scan activities, and pie charts of infected malware families. The other 37 of 69 diagrams illustrate the structure of threat intelligence such as malware behaviors and flows of attack campaigns. The goal of this survey is extracting characteristics for considering a new visualization method of graph-structured threat intelligence, therefore, only 37 threat diagrams are used for this purpose.

To obtain observations, these images are categorized by their purposes and approaches. The results of this categorization are in Figure 2. This figure shows the purposes of these diagrams are divided into three categories. First, Attack Flow is a category of purpose to explain a series of events such as infection chains and cyber kill chains. These flows describe a whole picture of one or a few campaigns. Second, C2 Infrastructure is to explain malicious server structures such as relationships between domains and IP addresses. These diagrams are used for attribution analysis. Third, Malware Behavior is for illustrating activities or structures of malware such as component relationships, file structures, and call flow graphs. These diagrams provide intelligence about malware analysis.

Because STIX 2.0 isn't designed to represent detailed malware behavior, this category is excluded from visualization targets in this paper. Also, visualization C2 Infrastructure is well achieved by general graph visualization methods because most of these diagrams are illustrated as Maltego[1]. Thus, this paper mainly focuses on Attack Flow diagrams and also partially supporting C2 Infrastructure. Through the study of threat diagrams, three key observations were obtained as follows.

O1 Most of the threat diagrams are illustrated as a DAG network. These edges only connected with nodes between adjacent layers such as flowchart diagrams for representing attack flows.

O2 Threat diagrams usually focus on relationships between IoCs and other entities rather than focus on details of each indicator. For example, it is important to illustrate

[1]https://www.paterva.com/

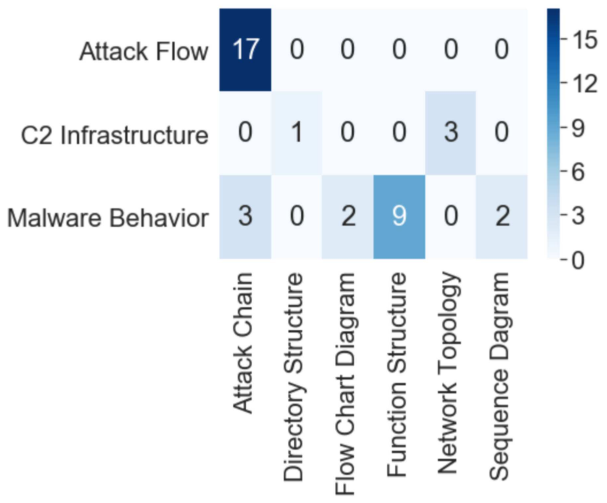


Figure 2: This heatmap shows purposes and approaches of threat diagrams in rpeorts.

node connections from different two malware to a cluster of indicators.

O3 Threat reports often emphasize differences in a threat structure such as a new exploit of vulnerability, a repeatedly observed IP address, and supporting a new attack pattern.

This paper assumes that manually created these diagrams have characteristics for enabling better visibility. Therefore, by satisfying the observations, it is possible to develop effective visualization methods.

3 Visualizing Treat Graph

Except for a particular case that is cycles of *variant-of* relationship, STIX 2.0 is practically DAG networks. It means that the observation O1 is consistent with the design of the STIX. The visualization method for DAG data has been proposed such as Sugiyama's framework [8]. The framework arranges all edges in the same direction by organizing the nodes in subsequent layers. Figure 3 (a) and (b) show differences between a general layout method for the directed graph[2] and Sugiyama's framework respectively. Although naively applying the framework improves visibility, there are two major problems. First, ordering hierarchies of nodes isn't intuitive. For example, because threat actor nodes are pointed from indicators and campaigns, Sugiyama's framework illustrates indicator and campaign nodes into high layers than actor nodes. In contrast, actual diagrams layouts reversed the order. Second, there are many crossed edges linked between non-adjacent layers due to the density of graph data. Actual diagrams have no these kinds of cross-layered edges.

To solve these problems, the proposed method implements three improvements. First, the method remaps the directions of edges. Specifically, directions of all *attributed-to* and *indicates* relationships on STIX are reversed to correspond to our intuitions. Second, orthogonal edge routing is used to layout edges as a series of right-angled lines. All edges have a primary direction along either the x-axis or y-axis, which can be used to bundle edges in a layer. Third, to de-emphasize cross-layered edges, this paper formulates finding cross-layered edges as the longest path problem. The definition of the cross-layered edge is that an edge between two nodes is a cross-layered edge if these two nodes have another longer path than the edge. The longest path problem on DAG equals the shortest path problem with negative weights that can be solved by using the Bellman-Ford algorithm [1]. This algorithm is dynamic programming and this computation cost is O(VE) time, where V and E are the numbers of nodes and edges respectively. Thus, the computation cost of finding the shortest paths for each node pair is O(SVE) time, where S is the number of source nodes of all edges. These improvements convert figure 3 (b) to (c). The diagram looks like a DAG network that has no cross-layered edges (O1).

Even if the proposed method focuses on a threat report, these reports often have tens to hundreds of entities because one campaign generates a large number of indicators. To hide redundant edges linked to these entities, the proposed method clusters these indicators based on relationships with other type entities because most of the threat reports only focus that (O2). Figure (d) shows the result.

To emphasize differences of a threat report from other reports, the proposed method mapped numbers of reports mentioned to each node or each into node size and edge width (O3). To demonstrate the utility of the method, this paper focuses on a particular threat actor APT28 also known as Fancy Bear. 36 threat reports about APT28 from the ATT&CK[3] were manually converted into STIX 2.0 format. In these conversions, actors, malware, attack patterns, indicators, tools, and vulnerabilities were converted into STIX. Figure 1 shows the resutls.

The proposed method is implemented by using Cytoscape.js[4] with the Cytoscape-Klay extension[5] which implements Sugiyama's framework.

4 Discussion

Figures 1 and 3 show that the proposed method improves visibility on a threat report. However, systematic experiments

[2]https://github.com/cytoscape/cytoscape.js-cola

[3]https://attack.mitre.org/groups/G0007/

[4]https://js.cytoscape.org

[5]https://github.com/cytoscape/cytoscape.js-klay

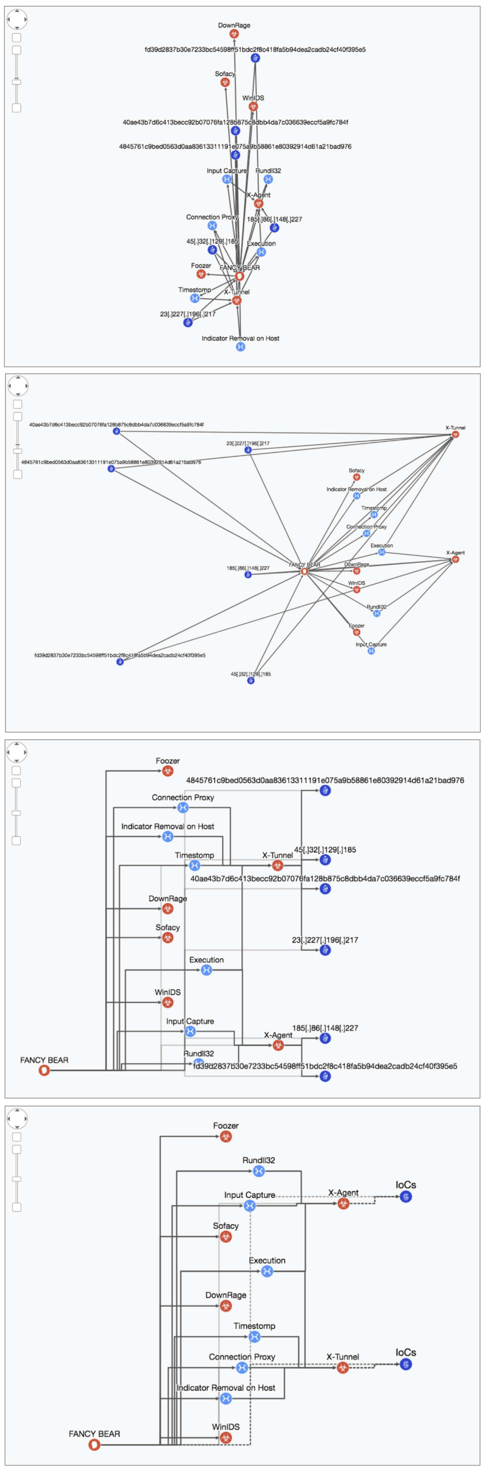


Figure 3: These diagrams show improvements of the proposed method. (a) is a general graph layout. (b) is a layered layout for DAG. (c) is applied to orthogonal edge routing and de-emphasizes cross-layered edges. (d) is applied to clustering IoCs.

to evaluate validity with more large data are necessary. Because of the cost limitation to create structured data for the evaluation, this paper only shows few examples. Figure 4 (a) shows another example based on a STIX example file[6]. In this case, there are some islands and many campaigns, and they cause the complexity of layouts. Because these visualizations are also problems in another layout method[7] as Figure 4 (b), it is necessary to handle these cases to provide better visualization. Although there are these problems, figure 4 shows the proposed method made a little improvement in visualizations.

Additionally, it is required to create rich STIX data to improve visualization. For example, kill-chain phrases are useful to reflect nodes how they are used in campaigns. In actual threat diagrams, nodes are plotted according to their context such as order by an IP address for a gate, dropped malware, and another IP address for C2. To determine theses context, kill-chain phrases for each node are required. However, most of threat intelligence feeds often lack these details.

5 Conclusion

This paper shows the characteristics of diagrams on actual threat intelligence reports and also proposed a new method to visualize graph-structured threat intelligence for a report. The method can be used to improving threat analysis capabilities on any TIPs.

References

[1] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

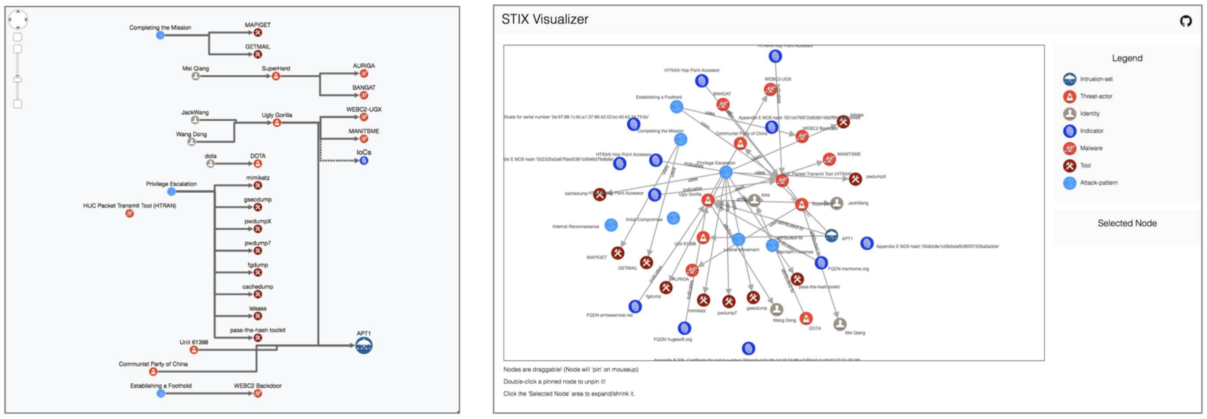
[2] Fabian Böhm, Florian Menges, and Günther Pernul. Graph-based visual analytics for cyber threat intelligence. *Cybersecurity*, 1(1):16, 2018.

[3] ENISA. Exploring the opportunities and limitations of current threat intelligence platforms, 2018. https://www.enisa.europa.eu/publications/exploring-the-opportunities-and-limitations-of-current-threat-intelligence-platforms.

[4] Cambridge Intelligence. Eclecticiq: Understanding cyber threat intel as a graph. https://cambridge-intelligence.com/eclecticiq-case-study/.

[5] MITRE. Standardizing cyber threat intelligence information with the structured threat information expression, 2012. https://www.mitre.org/publications/technical-papers/standardizing-cyber-threat-intelligence-information-with-the.

[6]https://oasis-open.github.io/cti-documentation/stix/examples
[7]https://oasis-open.github.io/cti-stix-visualization/



(a) (b)

Figure 4: These diagrams show the same graph constructed from an APT1 STIX file. (a) proposed method hierarchically illustrates entities and collapses indicators into one node. (b) STIX Visualizer naively shows node relationships.

[6] OASIS. Introduction to stix. `https://oasis-open.github.io/cti-documentation/stix/intro`.

[7] Clemens Sauerwein, Christian Sillaber, Andrea Mussmann, and Ruth Breu. Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives. *Wirtschaftsinformatik und Angewandte Informatik*, 2017.

[8] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.