An Optimization Approach to Graph Partitioning for Detecting Persistent Attacks in Enterprise Networks

Hazem M. Soliman
RANK Software Inc., Toronto, Canada
hazem.soliman@ranksoftwareinc.com

***Abstract*—Advanced Persistent Threats (APTs) refer to sophisticated, prolonged and multi-step attacks, planned and executed by skilled adversaries targeting government and enterprise networks. Attack graphs' topologies can be leveraged to detect, explain and visualize the progress of such attacks. However, due to the abundance of false-positives, such graphs are usually overwhelmingly large and difficult for an analyst to understand. Graph partitioning refers to the problem of reducing the graph of alerts to a set of smaller incidents that are easier for an analyst to process and better represent the actual attack plan. Existing approaches are oblivious to the security-context of the problem at hand and result in graphs which, while smaller, make little sense from a security perspective. In this paper, we propose an optimization approach allowing us to generate security-aware partitions, utilizing aspects such as the kill chain progression, number of assets involved, as well as the size of the graph. Using real-world datasets, the results show that our approach produces graphs that are better at capturing the underlying attack compared to state-of-the-art approaches and are easier for the analyst to understand.**

I. Introduction

Advanced Persistent Threats (APTs) form a crucial part of the attack campaigns targeting enterprise networks [1]. A combination of stealth actions, sophisticated exploits and long-term operation mark them as one of the most dangerous cybersecurity attacks [2]. As a planned multi-step attack, APTs do not target a single vulnerability. Instead, an APT typically starts by gaining access to the network, exploiting several vulnerabilities, and remaining silent for extended periods of time [3]. With data exfiltration as the common end-goal of those attacks [4], the economic damage is particularly severe due to the risk of losing intellectual properties or sensitive data [5].

A leading effort to study APT attacks is the ATT&CK matrix developed by the MITRE corporation [6]. The study observed that attacks generally follow similar patterns, and provided a taxonomy describing both offensive and defensive operations. The matrix describes the tactics, commonly known as the attack phases, the techniques for carrying out a tactic, and the observed instances of such techniques in known attacks. One of the main motivations behind this paper is building a framework to extract tactic-aware incident graphs from intrusion alerts for the purpose of detecting APTs.

Due to the sophisticated multi-step nature of APTs, individual intrusion alerts are not necessarily an indication of an actual attack [7]. Instead of individual alerts, more advanced detection approaches try to reconstruct the attack plan [8]. Attack graphs are a natural solution here [9]. Besides their visually attractive properties, the topology of a directed graph is a representation of the attack plan progression. In such graphs, each vertex represents an individual alert, which might come from a variety of security solutions, and each edge represents the progression between two alerts [10].

Despite its attractive properties, an attack graph containing all alerts is usually overwhelmingly large due to the high false-positive rate of the individual alert detectors. Hence, various approaches have been proposed to partition the graph into smaller sub-graphs, reducing the burden on the security analyst. In [11] a community-detection approach is used where each resulting community is assumed to correspond to a single attack step. However, community detection algorithms focus only on the graph's topological properties [12], and fail to incorporate security context to determine whether a community represents an actual attack plan or not. In [13] alert prerequisites and consequences are defined for each alert to limit the number of edges. Besides the heavy manual work involved, this approach can not handle cases when multiple attacks are carried-out simultaneously. Finally, in [10] a separate graph is defined for each single asset. This approach results in manageable graphs, but it ignores lateral movement, a common tactic in APT attacks.

Our main goal in this paper is to provide a rigorous formulation of the attack partitioning problem enabling the extraction of security-aware incident graphs. Recognizing that community detection algorithms approximate the graph partitioning problem [14], we leverage the power of convex optimization to incorporate security context into the problem. In particular, our objectives when partitioning a graph are:

- To guarantee a proper incident, we would like to maximize the number of attack phases covered in each partition.
- To avoid impractical incidents, we would like to minimize the number of assets involved in a partition.
- To provide the analyst with an easy-to-visualize graph, we would like to have an upper limit on the number of alerts included in each partition.
- To balance how many incidents involve a single alert, our formulation allows having a reasonable and flexible limit on the number of partitions an alert might appear in. For example, limiting an alert to a single incident is overly optimistic while sharing it across many incidents is overly pessimistic. This is analogous to the concept of

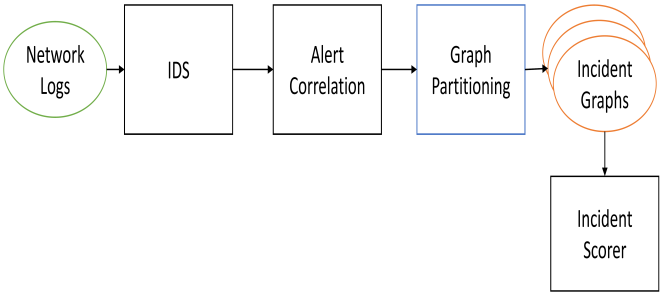

Fig. 1. Security System Architecture

overlapping communities [15].

- Perhaps most importantly, to allow more flexibility in addressing security analysts feedback, we would like to have a more tractable formulation of the problem, unlike black-box community detection algorithms.

II. System Description

A. Architecture

The architecture of our incident detection system is shown in Fig 1. The input is the set of network and/or host event logs. These logs are then analyzed by an intrusion detection system (IDS) to raise alerts for anomalous or suspicious behavior. Examples of IDSs are *Suricata*[1] and *Snort*[2]. The set of alerts is fed to the alert correlation module which builds an alert graph, whose edges indicate alert correlation and/or causality. Since the alert graph can be thousands of nodes big, the next component is the graph partitioning module, which is the focus of this paper. This module partitions the alert graph from the previous step into a set of security incidents represented as incident graphs. The final step is to assign a score for each incident. These scores are used to present detected incidents to a security analyst in a sorted manner according to their severity.

B. Mathematical Model

Consider a security system where network and host-based logs are processed against a set of signatures and behavioral anomaly detectors in order to raise a set of security alerts. We denote the set of alerts as $\mathcal{V}$ which is also the set of vertices in the alert graph.

Each alert has a set of attributes, for example, the affected network asset, source and destination IPs and source and destination port numbers for network-based alerts. Host-based alerts might have additional attributes such as process ID, parent process and globally unique identifier. A crucial attribute that we assume is present in all nodes is a set of one or more MITRE tactics that the alert is an indication of.

We define the set of edges as a set of ordered pairs $\mathcal{E} = \{e_{i,j} = (v_i, v_j) : \forall v_i, v_j \in \mathcal{V} \; and \; c(e_{i,j}) > 0\}$, where $c : \mathcal{E} \longrightarrow \mathbb{R}$ is a mapping assigning weights to edges. The directed alert graph is then defined as the pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We can group the weights together in a weight matrix $\mathbf{W} = (w_{i,j})_{|\mathcal{V}|\times|\mathcal{V}|}$ where $w_{i,j} > 0$ indicates the presence of an edge directed from v_i to v_j with weight $w_{i,j} = c(e_{i,j})$, while $w_{i,j} = 0$ indicates the absence of such an edge.

We note that the form of the correlation function $c(.)$ is an area of research by itself. From our practical experience and discussions with security analysts, an edge $e_{1,2} = (v_1, v_2)$ is typically understood as an indication of a causal relationship between v_1 and v_2, that is, v_1 is a direct cause of v_2. Mathematical models of causality are still an active area of research [16]. Alert graph building and modeling causal relationships between alerts is not the focus of this work. Instead, we assume a graph has already been constructed and the objective is to partition said graph into separate meaningful incidents.

III. Graph Partitioning

The graph partitioning problem is concerned with splitting the vertex set $\mathcal{V}$ of the graph $\mathcal{G}$ into a collection of non-empty subsets. The standard objective in such problems is to minimize the total weight of the edges connecting any two subsets [17].

A. Problem formulation

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, define the decision matrix $\mathbf{X} = (x_{i,k})_{|\mathcal{V}|\times K}$ where $x_{i,k} \in \mathbb{Z}_2$ and K is the number of partitions. The value $x_{i,k} = 1$ indicates vertex i being part of partition k, and 0 otherwise. We also refer to the column vectors of the matrix $\mathbf{X}$ as an incident or an incident vector $\mathbf{x}_k$. Although the number of partitions K is assumed as input to our optimization problem, we note that the formulation can be extended to have K as an optimization variable as well [17].

The objective of the graph partitioning problem is to partition the set of vertices in a way that minimizes the total weight of edges between any two subsets. Following [17], this can be written as

$$\min_{\mathbf{X}} \quad \frac{1}{2} trace(\mathbf{X}^T \mathbf{W} \mathbf{X}) \tag{1}$$

A set of constraints can be added to the above problem.

- Binary Constraint: a decision variable $x_{i,k}$ indicates presence or absence of the alert v_i in the partition k.

$$\mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}|\times K} \tag{2}$$

- Exclusivity or Semi-Exclusivity: this constraint controls how many partitions a single vertex can belong to. In our case this translates to how many incidents the alert is part of.

$$\mathcal{R} = \left\{ x_{i,k} \in \mathbb{Z}_2 : \sum_{k=1}^{K} x_{i,k} \leq MaxMemb \right\} \tag{3}$$

where $MaxMemb$ is an upper limit on the alert incident membership. In the case $MaxMemb = 1$, the alert can only be part of a single incident, and $MaxMemb > 1$ is a case of overlapping graph partitions [15].

[1] https://suricata-ids.org/
[2] https://www.snort.org/

- Cardinality: this constraint effectively controls the size, i.e., number of alerts, in each partition. Since the resulting incidents are presented to a security analyst for further investigation, the size of each incident must be kept within a reasonable limit for human inspection.

$$\mathcal{P} = \left\{ x_{i,k} \in \mathbb{Z}_2 : \sum_{i=1}^{|\mathcal{V}|} x_{i,k} \leq MaxCard \right\} \qquad (4)$$

where $MaxCard$ is the maximum size allowed for a partition.

The overall optimization problem can be written as

$$\begin{aligned} \underset{\mathbf{X}}{\text{minimize}} \quad & \frac{1}{2} trace(\mathbf{X}^T \mathbf{W} \mathbf{X}) \\ \text{subject to} \quad & \mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}| \times K}, \\ & x_{i,k} \in \mathcal{R} \cap \mathcal{P} \;\; \forall i = 1, ..., |\mathcal{V}|, k = 1, ..., K \end{aligned} \qquad (5)$$

This problem belongs to the class of mixed-integer quadratic programming (MIQP), and while the above formulation is not the standard one, interested readers can refer to [17] on how to convert it into a standard MIQP.

B. Linear Programming Formulation

Problem (5) can also be converted into an equivalent mixed integer linear program (MILP) [18]. Let $\mathbf{L} = diag(\sum_j w_{1j}, ..., \sum_j w_{|\mathcal{V}|j}) - \mathbf{W}$ denote the Laplacian matrix of the graph $\mathcal{G}$. Define the auxiliary variables $\mathbf{S} = (s_{i,k})_{|\mathcal{V}| \times K} = (\mathbf{s}_1, ..., \mathbf{s}_K)$ and $\mathbf{T} = (t_{i,k})_{|\mathcal{V}| \times K} = (\mathbf{t}_1, ..., \mathbf{t}_K)$. Problem (5) can now be written as [17]

$$\begin{aligned} \underset{\mathbf{X},\mathbf{S},\mathbf{T}}{\text{minimize}} \quad & \mathbf{e}^T \mathbf{S} \mathbf{e} \\ \text{subject to} \quad & \mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}| \times K}, \\ & x_{i,k} \in \mathcal{R} \cap \mathcal{P}, \\ & \mathbf{L}\mathbf{x}_k - \mathbf{t}_k - \mathbf{s}_k + C\mathbf{e} = 0, \\ & \mathbf{t}_k \leq 2C(\mathbf{e} - \mathbf{x}_k), \\ & t_{i,k}, s_{i,k} \geq 0 \;\; \forall i = 1, ..., |\mathcal{V}|, k = 1, ..., K \end{aligned} \qquad (6)$$

where $C = 2 \max_{i=1,...,|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} w_{i,j}$ and $\mathbf{e}$ is an all-ones vector.

C. Kill Chain Objective

From our internal study of the attack graphs and discussions with security analysts, a major indication of the maliciousness of an incident graph is how many of the MITRE attack tactics are covered. For example, an incident graph whose alerts are all of type *Initial Access* is much less worthy of further investigation compared to one including *Initial Access*, *Persistence*, *Lateral Movement* and *Exfiltration*.

Let $\mathcal{M}$ be the set of attack tactics and define the binary matrix $\mathbf{P} \in \mathbb{Z}_2^{|\mathcal{V}| \times |\mathcal{M}|}$ where $p_{i,m} = 1$ indicates alert v_i being an indication of attack tactic $m \in \mathcal{M}$. Then for an incident vector $\mathbf{x}_k$, define the vector

$$\mathbf{m}_k = \mathbf{x}_k^T \cdot \mathbf{P} \in \mathbb{Z}^{|\mathcal{M}|} \qquad (7)$$

A vector $\mathbf{m}_k$ has $m_{i,k} > 0$ if the tactic i is present in any of the alerts in incident k, and 0 otherwise. The 0-norm [19] of the said vector, $||\mathbf{m}_k||_0$, is the number of its non-zero elements, and in our case this is the number of tactics present in the incident $\mathbf{x}_k$. Our goal is to maximize this number.

However, 0-norms are typically minimized, not maximized, in convex optimization problems [19]. Instead, we can define the complementary tactic matrix $\hat{\mathbf{P}} \in \mathbb{Z}_2^{|\mathcal{V}| \times |\mathcal{M}|}$ where $\hat{p}_{i,m} = 1$ now indicates the absence of tactic m from alert i. We now propose another approach to handle tactic coverage in an incident. Again define the vector

$$\hat{\mathbf{m}}_k = \mathbf{x}_k^T \cdot \hat{\mathbf{P}} \in \mathbb{Z}^{|\mathcal{M}|} \qquad (8)$$

A vector $\hat{\mathbf{m}}_k$ has $\hat{m}_{i,k} > 0$ if the tactic i is missing from any of the alerts in incident k, and 0 otherwise. The infinity norm of the said vector, $||\hat{\mathbf{m}}_k||_\infty$, is the cardinality of the largest set of alerts within the incident $\mathbf{x}_k$ that any single tactic is missing from. For example, an incident of 3 alerts where a tactic is missing from all 3 of them is less desirable compared to one where the same tactic is missing from only 2 alerts. By minimizing this norm, we are trying to approximately achieve more coverage for as many tactics as possible within an incident, and minimizing the infinity norm fits conveniently within the linear programming framework as will be shown later.

D. Asset Count Objective

Incident graphs that involve a larger number of assets are less desired by security analysts. The reason being that APTs try to remain hidden and hence optimize the number of machines involved to minimize the risk of detection. The goal here is to minimize the number of assets involved in an incident $\mathbf{x}_k$. Similar to III-C, define the asset matrix $\mathbf{A} \in \mathbb{Z}_2^{|\mathcal{V}| \times |\mathcal{A}|}$ where $\mathcal{A}$ is the set of assets in the network, and $a_{i,l} = 1$ indicates alert v_i being triggered for asset $l \in \mathcal{A}$. Then for an incident vector $\mathbf{x}_k$, define the vector

$$\mathbf{l}_k = \mathbf{x}_k^T \cdot \mathbf{A} \in \mathbb{Z}^{|\mathcal{A}|} \qquad (9)$$

A vector $\mathbf{l}_k$ has $l_{i,k} > 0$ if the asset i is present in any of the alerts in incident k, and 0 otherwise. The 0-norm of the said vector, $||\mathbf{l}_k||_0$, is the number of assets involved in the incident $\mathbf{x}_k$. Our goal is to minimize this number. A standard practice in compressive-sensing is to use the 1-norm $||\mathbf{l}_k||_1$ convex relaxation of the 0-norm instead [19].

E. Main Optimization Problem

Combining the results of III-B, III-C and III-D, our main optimization problem can be written as

$$\begin{aligned} \underset{\mathbf{X},\mathbf{S},\mathbf{T}}{\text{minimize}} \quad & \gamma_0 \mathbf{e}^T \mathbf{S} \mathbf{e} + \gamma_1 \sum_{k=1}^{K} ||\hat{\mathbf{m}}_k||_\infty + \gamma_2 \sum_{k=1}^{K} ||\mathbf{l}_k||_1 \\ \text{subject to} \quad & \mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}| \times K}, \\ & x_{i,k} \in \mathcal{R} \cap \mathcal{P}, \\ & \mathbf{L}\mathbf{x}_k - \mathbf{t}_k - \mathbf{s}_k + C\mathbf{e} = 0, \\ & \mathbf{t}_k \leq 2C(\mathbf{e} - \mathbf{x}_k), \\ & t_{i,k}, s_{i,k} \geq 0 \;\; \forall i = 1, ..., |\mathcal{V}|, k = 1, ..., K \end{aligned} \qquad (10)$$

where $\gamma_0, \gamma_1, \gamma_2$ are tunable weights for the different parts of the objective function, i.e., graph cut weight, tactic coverage and number of assets respectively.

Using standard convex optimization tricks [20], Problem (10) can be re-written as

$$\begin{array}{ll}\underset{\mathbf{X},\mathbf{S},\mathbf{T},\alpha,\boldsymbol{\beta}}{\text{minimize}} & \gamma_0 \mathbf{e}^T \mathbf{S}\mathbf{e} + \gamma_1 \sum_{k=1}^{K} \alpha_k + \gamma_2 \sum_{k=1}^{K} \mathbf{1}^T \boldsymbol{\beta}_k \\ \text{subject to} & \mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}|\times K}, \\ & x_{i,k} \in \mathcal{R} \cap \mathcal{P}, \\ & \mathbf{L}\mathbf{x}_k - \mathbf{t}_k - \mathbf{s}_k + C\mathbf{e} = 0, \\ & \mathbf{t}_k \leq 2C(\mathbf{e} - \mathbf{x}_k), \\ & t_{i,k}, s_{i,k} \geq 0, \\ & -\alpha_k \mathbf{e} \leq \hat{\mathbf{m}}_k \leq \alpha_k \mathbf{e}, \\ & -\boldsymbol{\beta}_k \leq \mathbf{l}_k \leq \boldsymbol{\beta}_k \quad \forall i = 1, \ldots, |\mathcal{V}|, k = 1, \ldots, K \end{array} \tag{11}$$

The above problem is an example of a mixed integer linear program (MILP), meaning it is also NP-hard [17]. However, since removing the constraint $\mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}|\times K}$ results in a standard linear program, this problem belongs to the class of disciplined mixed integer linear programs [21]. While still NP-hard, this smaller class of problems is more attractive from the standpoint of practical solvers.

By designing the problem to be a disciplined MILP, we can use very good convex programming software like CVXPY [21], and by taking care to stay within the realm of linear programming, we can leverage a plethora of free open-source solvers such as the excellent CBC solver [3] which has given us the best results in practice.

F. Integer Relaxation

Solving the mixed-integer problem might be infeasible for large enterprise networks in practice. Removing the constraint $\mathbf{X} \in \mathbb{Z}_2^{|\mathcal{V}|\times K}$ results in a convex version which can be solved by any LP solver. We have found it best to use the following approximation scheme to round the solution to binary values

$$\tilde{x}_{i,j} = \begin{cases} 1 & j = \arg\max_{k\in\{1,\ldots,K\}} x^*_{i,k} \\ 0 & otherwise \end{cases} \tag{12}$$

where $\mathbf{X}^*$ is the solution of problem (11). We have also found that adding an objective term for the sparsity of decision variable $\mathbf{X}$ synergizes well with this approximation scheme.

IV. Results

The lack of good APT datasets, as well as the well-documented challenges of applying algorithmic and machine-learning approaches in cybersecurity [22], makes evaluating any work particularly challenging. For datasets, we use the DARPA dataset [23], the figures included in this paper are for its first scenario, and while similar observations hold for the second scenario, its figures are not included due to space limitations. We also use a private dataset from a medium enterprise network. This network has around 50 employees using primarily windows machines, as well as a few internal servers. The Suricata IDS is used to analyze the traffic between the internal network and internet gateway. As a baseline we compare against the ego-splitting framework from [15], a leading community detection algorithm. Ego-splitting utiltizes non-overlapping algorithms underneath, for which we use *Louvain* and *Modularity* from the NetworkX package [4], as these two have given the best results in practice. The evaluation metric is first and foremost a manual inspection of the resulting graphs, as well as statistics on the quality of those graphs, such as the average number of alerts.

Throughout our experiments, we used Suricata with its open rule-set as our IDS[5]. We have used $\gamma_0 = 1.0, \gamma_1 = 0.5, \gamma_2 = 0.5, MaxMemb = 2$ and $MaxCard = 15$. For the experiment with the DARPA dataset we solved the integer version of problem (11) while the relaxed version is used for the experiment with the enterprise network dataset.

In Fig. 2 we show the overall attack graph in the DARPA scenario. The goal of the attack is to exploit the Solaris sadmind vulnerability [6] on eligible hosts, install a trojan and launch a DDoS attack against a remote server using the compromised machines. The attack progression and corresponding Suricata alerts are summarized in Table I.

The incident extracted through our approach is shown in Fig. 4, while the ego-splitting community detection is shown in Fig. 3. As we can see in the two graphs, our approach has identified the correct attack path, while the ego-splitting is more focused on the interconnectivity of the nodes and this has resulted in a confusion between the correct path and the false-positive path with HTTP and email spam alerts. Our proposed approach results in a better incident representation containing only the alerts relevant to the attack.

For the private dataset, one interesting metric was how much the size of the incident graph grows as we lengthen the analysis period, which also increases the number of alerts. In Fig. 5 we show the average incident size versus the length of the analysis period. We see how when increasing the number of alerts, as needed to discover long-term APTs, the community detection results in larger communities hindering the ability of the analyst to study them, a weakness alleviated in our model.

V. Future Work

The main weakness in the optimization approach is the ability of the solvers to handle growing graph sizes. For example, we see around 1000 alerts for a three week period in a medium enterprise network. If we would like to limit the size of each incident to 10 alerts, then the decision matrix is of size 100000. These numbers quickly approach the practical limits of the solvers, between 100000 and 1 million variables, and also make solving the integer version of the problem infeasible. We would like to study a more-practical heuristic that can provide a good approximation to problem (11).

[3] https://github.com/coin-or/Cbc

[4] https://networkx.github.io/

[5] https://rules.emergingthreats.net/open/suricata/rules/

[6] https://cve.circl.lu/cve/CVE-1999-0977

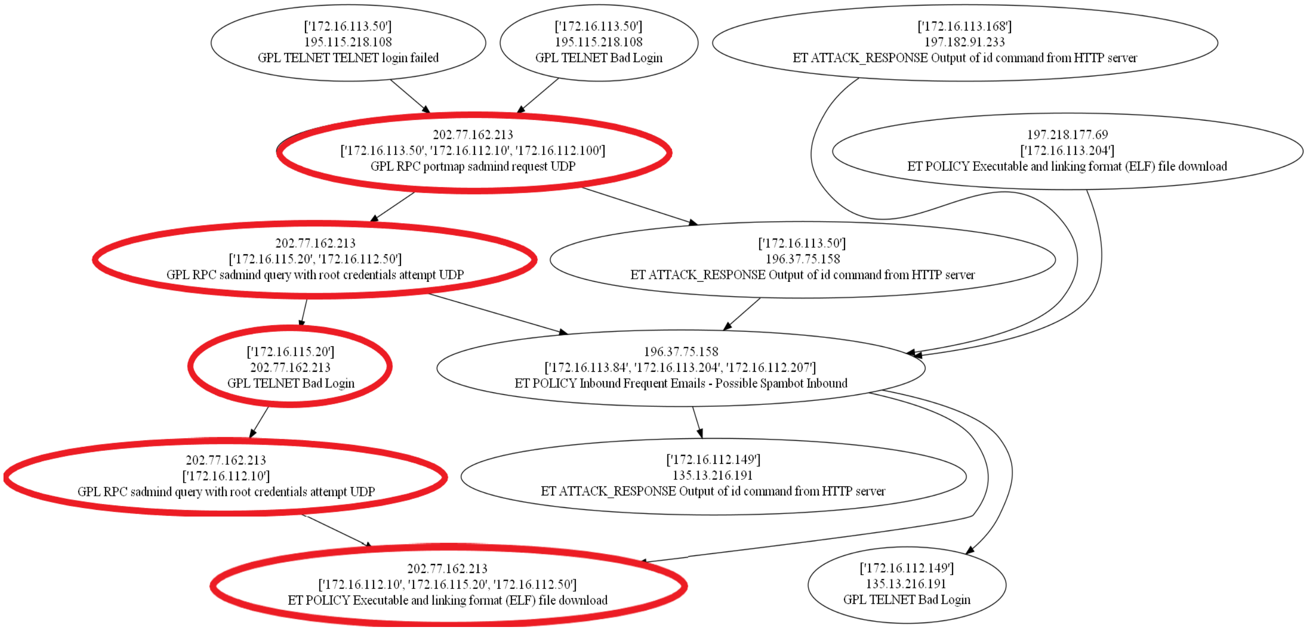

Fig. 2. DARPA overall alert graph prior to partitioning. The contents of each node are the source ip(s) (top), destination ip(s) (middle) and signature of the Suricata rule triggering the alert (bottom). Thick red nodes indicate the real steps of the attack. Some of the edges are omitted for visual clarity.

TABLE I
DARPA Attack Progression and IDS Alerts

Attack Step	**IDS Alert(s)**
An IP sweep of the enterprise hosts.	The IP scan is not reported.
Probing the IPs from step 1 to look for sadmind hosts.	Detected by rule "GPL RPC portmap sadmind request UDP" .
Exploiting the Solaris Sadmind vulnerability (CVE-1999-0977).	Detected by rule "GPL RPC sadmind query with root credentials attempt UDP".
Installation of the DDoS trojan on exploited machines.	Detected by rule "GPL TELNET Bad Login", followed by "GPL RPC sadmind query with root credentials attempt UDP" and the actual file download in "ET POLICY Executable and linking format (ELF) file download".
Launching the DDoS attack.	Detected but not connected to the other alerts due to the trojans using source IP spoofing.

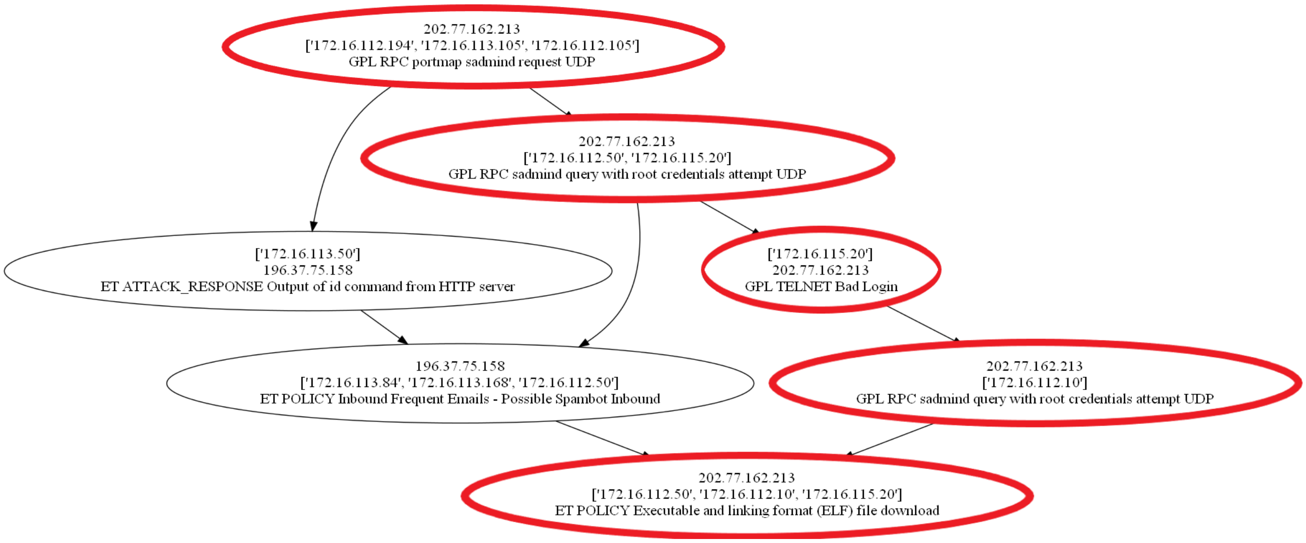

Fig. 3. DARPA incident graph through community detection.

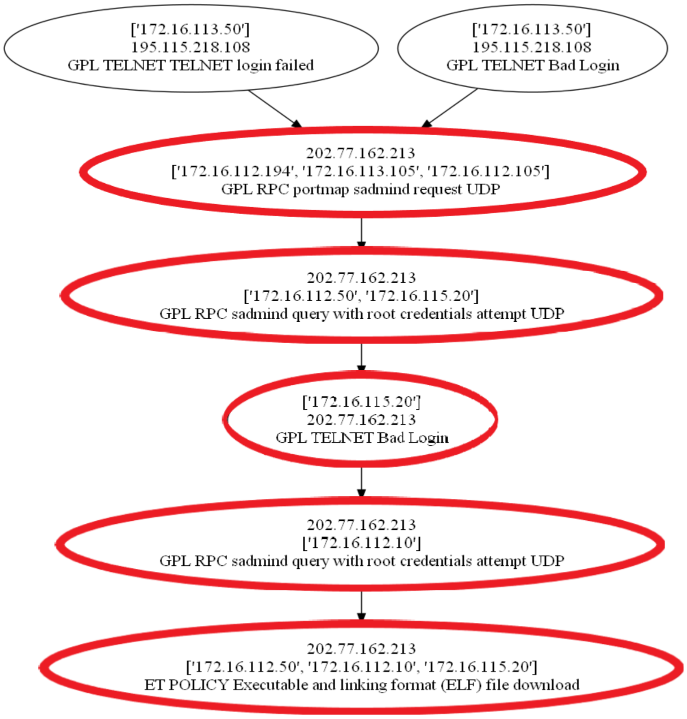

Fig. 4. DARPA incident graph through graph optimization.

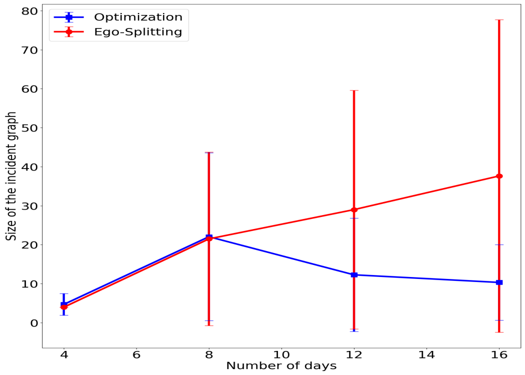

Fig. 5. Comparison of Average Incident Size.

VI. Conclusion

In this paper we proposed an optimization approach for the problem of graph partitioning as part of the detection process for APT attacks. The goal of the optimization is to minimize the graph cut between any two partitions, maximize the number of kill chain stages included in a partition, minimize the number of assets and limit the size of the partition for easy understanding by the analyst. We have also addressed how the problem can be solved in both the mixed-integer and relaxed cases. Our results showed our approach is superior to the existing community detection algorithms by generating graphs that better capture the progression of the attack. Equally important is how our approach is easily extendable to incorporate other constraints as required by the analyst, a crucial feature that is not easily applied to other community detection algorithms.

References

[1] I. Ghafir and V. Prenosil, "Advanced persistent threat attack detection: an overview," *International Journal of Advances in Computer Networks and Its Security (IJCNS)*, vol. 4, no. 4, p. 5054, 2014.

[2] D. McWhorter, "Apt1: Exposing one of china's cyber espionage units," *Mandiant. com*, vol. 18, 2013.

[3] M. Alvarez, N. Bradley, P. Cobb, S. Craig, R. Iffert, L. Kessem, J. Kravitz, D. McMillen, and S. Moore, "Ibm x-force threat intelligence index 2017 the year of the mega breach," *IBM Security,(March)*, pp. 1–30, 2017.

[4] T. R. Rakes, J. K. Deane, and L. P. Rees, "IT security planning under uncertainty for high-impact events," *Omega*, vol. 40, no. 1, pp. 79–88, 2012.

[5] H. Berghel, "Equifax and the latest round of identity theft roulette," *Computer*, vol. 50, no. 12, pp. 72–76, 2017.

[6] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," *MITRE Product MP*, pp. 18–0944, 2018.

[7] P. Cao, E. Badger, Z. Kalbarczyk, R. Iyer, and A. Slagell, "Preemptive intrusion detection: Theoretical framework and real-world measurements," in *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, 2015, pp. 1–12.

[8] X. Qin, "A probabilistic-based framework for infosec alert correlation," Ph.D. dissertation, Georgia Institute of Technology, 2005.

[9] B. Eshete, R. Gjomemo, M. N. Hossain, S. Momeni, R. Sekar, S. Stoller, V. Venkatakrishnan, and J. Wang, "Attack analysis results for adversarial engagement 1 of the darpa transparent computing program," *arXiv preprint arXiv:1610.06936*, 2016.

[10] P. Cao, "On preempting advanced persistent threats using probabilistic graphical models," *arXiv preprint arXiv:1903.08826*, 2019.

[11] S. Haas and M. Fischer, "On the alert correlation process for the detection of multi-step attacks and a graph-based realization," *ACM SIGAPP Applied Computing Review*, vol. 19, no. 1, pp. 5–19, 2019.

[12] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[13] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 245–254.

[14] C. Schulz, "Graph partitioning and graph clustering in theory and practice," *Institute for Theoretical Informatics Karlsruhe Institute of Technology (KIT).–May*, vol. 20, pp. 24–187, 2016.

[15] A. Epasto, S. Lattanzi, and R. Paes Leme, "Ego-splitting framework: From non-overlapping to overlapping clusters," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 145–154.

[16] J. Pearl et al., "Causal inference in statistics: An overview," *Statistics surveys*, vol. 3, pp. 96–146, 2009.

[17] N. Fan and P. M. Pardalos, "Linear and quadratic programming approaches for the general graph partitioning problem," *Journal of Global Optimization*, vol. 48, no. 1, pp. 57–71, 2010.

[18] W. Chaovalitwongse, P. M. Pardalos, and O. A. Prokopyev, "A new linearization technique for multi-quadratic 0–1 programming problems," *Operations Research Letters*, vol. 32, no. 6, pp. 517–522, 2004.

[19] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE signal processing magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[22] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.

[23] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.