AlertVision: Visualizing Security Alerts

Jina Hong[1], JinKi Lee[2], HyunKyu Lee[2], YoonHa Chang[2], KwangHo Choi[2], and Sang Kil Cha[1(✉)]

[1] KAIST, Daejeon, Korea
{jina3453,sangkilc}@kaist.ac.kr
[2] AhnLab, Seongnam, Korea
{jinki.lee,hyunkyu.lee,yoonha.chang,kwangho.choi}@ahnlab.com

Abstract. Security is not just a technical problem, but it is a business problem. Companies are facing highly-sophisticated and targeted cyber attacks everyday, and losing a huge amount of money as well as private data. Threat intelligence helps in predicting and reacting to such problems, but extracting well-organized threat intelligence from enormous amount of information is significantly challenging. In this paper, we propose a novel technique for visualizing security alerts, and implement it in a system that we call AlertVision, which provides an analyst with a visual summary about the correlation between security alerts. The visualization helps in understanding various threats in wild in an intuitive manner, and eventually benefits the analyst to build TI. We applied our technique on real-world data obtained from the network of 85 organizations, which include 5,801,619 security events in total, and summarized lessons learned.

Keywords: Threat intelligence · Alert visualization · Alert correlation

1 Introduction

Security is a growing concern for enterprises and organizations with ever-evolving attack techniques. Today's security threats involve complex attack scenarios, and are designed to cause persistent damage against specific targets. They are often called an Advanced Persistent Threat, or APT in short [37]. APT actors typically leverage 'advanced' techniques such as code obfuscation and metamorphism [27] in order to thwart the detection.

Traditional defense approaches, e.g., Intrusion Detection System (IDS) [2], are *not* sufficient to handle APTs, because their focus is only on *attack instances*. That is, conventional defenses are mainly about understanding the behavior of malware instances, analyzing what kind of vulnerabilities are exploited, or figuring out what kind of techniques are used to bypass defenses. However, such information can vary depending on the victim as well as the attack campaign. Furthermore, responding to each and every threat by analyzing them is not feasible anyways in practice as they appear on a daily basis.

B. B. Kang and J. Jang (Eds.): WISA 2018, LNCS 11402, pp. 173–184, 2019.
https://doi.org/10.1007/978-3-030-17982-3_14

To deal with APTs, enterprises now try to utilize Threat Intelligence (TI), which is well-refined knowledge about threats with *outward* focus. That is, TI includes information beyond attack instances such as the behavioral patterns of the threat actors, their intent, and their characteristics. It is widely known that TI can help prevent security threats in a *proactive* manner [1].

Although TI-based defense is a promising direction, extracting TI from massive information obtained in wild is challenging because there are too many attack instances to consider. Companies employ Security Information and Event Management (SIEM) systems to detect threats and collect the corresponding events, which typically produce thousands of events per hour. It is not clear how to interpret and correlate those events to understand the attackers behind the scene. Furthermore, there can be false alerts from SIEM systems, which can easily confuse the TI generation process.

The current best practice in building TI is to correlate alerts generated from various IDS/IPS systems and identifies high-level patterns of current attacks. This process is often called *alert correlation* [23], and it can be used to identify unknown threats in the future. Most research in this field currently focuses on improving their accuracy [30,32,36], but an automated way of visualizing the correlation between security alerts is largely unexplored to date.

In this paper we present a simple and effective approach to visualize security alerts obtained from SIEM systems. We argue that such visual aids help analysts understand the characteristics of the attacks and the attackers behind, which often do not change regardless of the attack campaign: attackers tend to behave similarly even though the actual attack methodology may vary. To this end, we implement AlertVision, a visualization system for SIEM alerts, and evaluate it on real-world SIEM logs, which constitute 5,801,619 alerts in total.

To visualize security alerts, AlertVision first groups them based on their property, and produces a set of alert sequences. Each grouped sequence represents a feature of attack incidents, e.g., an attack source IP or a target service. Our system then computes similarity between the sequences, and visualizes their relationships in a graph. The key intuition here is that two or more features that are seemingly irrelevant can be similar to each other, and visualizing their relationship can often help understand the meaning of the incidents. To figure out the similarity between two distinct event sequences, it leverages a sequence alignment algorithm used in bioinformatics [34].

The primary challenge of AlertVision is to draw a graph where the coordinates of the nodes are not known, but only the distances, i.e., the similarity, between them are known. We leverage a force-directed graph drawing algorithm [7], which can draw a graph in a space based only on their relative distances. The resulting graph provides a useful insight to analysts because it can reveal that two *seemingly different* alert sequences are indeed similar to each other in the graph. Unlike traditional cluster analysis such as hierarchical clustering, the graph instantly presents visual evidence to analysts.

Our main contributions are as follows.

1. We propose a technique for visualizing relationship between attackers, which can help in understanding the meaning of security incidents.
2. We evaluated our technique on a large dataset obtained from real SIEM devices in wild.
3. We empirically show that security analysts can benefit from our visualization framework in terms of detecting previously unknown attacks.

2 Background

This section introduces the concept of local sequence alignment algorithm and force-directed graph layout algorithm, which serve as the basis of our alert visualization approach.

2.1 Local Sequence Alignment Algorithm

Sequence alignment is a way of arranging sequences. There are mainly two categories: local and global sequence alignment. Local sequence alignment algorithm finds similar subsequences between two sequences. Global alignment algorithm aims to obtain an end-to-end alignment between two sequences, whereas local alignment algorithm focuses on subsequences. Since we are dealing with SIEM event sequences that are different in their size and their look, we use local sequence alignment algorithm to obtain the similarity between the subsequences.

The most popular local sequence alignment algorithm is Smith-Waterman [34], which is a variation of Needleman-Wunsch algorithm [25]. Smith-Waterman algorithm is widely adopted in various areas in security such as malware analysis [15] and intrusion detection [3]. The algorithm takes in two sequences $s_1 = a_1, a_2, \ldots, a_m$ and $s_2 = b_1, b_2, \ldots, b_n$ of length m and n, respectively, and computes a scoring matrix H as follows. First, it constructs a $(m+1)$-by-$(n+1)$ scoring matrix H, where $H_{k0} = H_{0l} = 0$ for $0 \leq k \leq m$ and $0 \leq l \leq n$. It then fills in the scoring matrix with the following equation where $s(a, b)$ is a similarity score of the two elements a and b, and W_k is the penalty of having a gap of length k:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \geq 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \geq 1}\{H_{i,j-l} - W_l\}, \\ 0. \end{cases} \quad (1 \leq i \leq m,\ 1 \leq j \leq n)$$

Finally, it traces back from a cell in H of the highest score to the one with a score 0, which constitutes the most similar subsequence of s_1 and s_2.

The time complexity of the classic Smith-Waterman algorithm is $O(m^2 n)$, but Gotoh *et al.* [8] proposed an algorithm of $O(m + n)$ time complexity, and Myers *et al.* [24] showed an algorithm of $O(n)$ space complexity. There are also several linear-time and linear-space sub-optimal algorithms [11], which

make local sequence alignment even more practical. Furthermore, there are several recent attempts to leverage GPU to accelerate the Smith-Waterman algorithm [26,28].

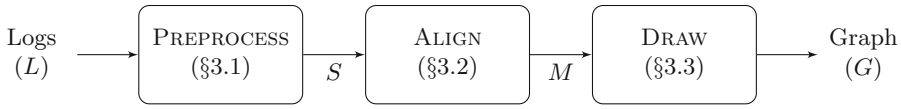

Fig. 1. Overview of AlertVision.

2.2 Force-Directed Graph

Force-directed graph drawing [7] is an algorithm used for graph layout and visualization. It takes advantage of the idea of Coulomb's law and Hooke's law to determine the position of nodes. In particular, there are attractive forces between nodes that are far apart, and are repulsive forces between nodes that are close to each other. The algorithm moves nodes based on these forces until it reaches an equilibrium state. We leverage this idea to visually represent security alerts. Although alert logs typically do not have the notion of coordinates, we can assign specific positions for each alert based on their relative similarities with force-directed graph drawing. As a result, we can apply a simple and cheap clustering algorithm such as k-means clustering to perform a cluster analysis on alert logs instead of using an expensive one such as hierarchical clustering [33].

3 AlertVision Design

At a high level, AlertVision takes in security logs generated from SIEM systems and returns a graph that visually correlating security alerts in the logs. Figure 1 shows the overall architecture of AlertVision. AlertVision consists of three major modules: Preprocess, Align, and Draw. First, Preprocess parses alert logs L and produces sequences of alerts S. Next, Align finds similar subsequences from S using a local sequence alignment algorithm, and produces a matrix M that stores similarity between every pair of S. Finally, Draw returns a graph where a sequence in S represents a node based on their similarity M.

3.1 Preprocess

AlertVision first preprocesses alert logs L to generate a set of alert sequences S by grouping alerts based on a specific attack feature. An attack feature includes a source IP address initiated the attack and a corresponding attack signature. By grouping alerts based on a feature, we can potentially realize relationship between feature values. For example, we may be able to realize the similarity between specific attacks if we visualize alert sequences grouped by their attack signatures.

In our current implementation, we focus on logs obtained from Network Intrusion Detection Systems (NIDS). In particular, we focus on source IP addresses of alert logs. By definition, every entry in NIDS logs contains its source IP address, i.e., an IP address that initiated the attack. By collecting a sequence of alerts for each IP address, we know what kind of attacks are introduced from an IP address in which order. Furthermore, assuming that attack payloads sent from the same IP address are from the same attacker, we can group the logs, and can potentially figure out similarities between attackers. From our experiments we found that an attacker tends to use the same set of IP addresses during an attack campaign even though actual payloads they use may differ. One notable example is APT, which typically includes multiple stages of independent attacks.

3.2 Align

Align takes in a set of grouped sequences S, and produces a similarity matrix M, which contains similarity scores for every pair of grouped sequences in S. The similarity scores are used to visualize the relationship between the sequences in the next step. To compute M, we focus on local similarity between two sequences. Specifically, we first use Smith-Waterman algorithm to compute a local alignment with the gap penalty $W_k = 1$, and the similarity score 2 and -2 for matching and mismatching elements, respectively. In our implementation, we say two alerts match if they have the same IDS signature.

Since Smith-Waterman returns the most similar subsequence of given two sequences, we use the subsequence as the measure of similarity. In particular, we compute the sum of similarity score (in the scoring matrix H) for every element in the subsequence, and normalize the sum by dividing it by the minimum length of the two sequences, because the sum may differ significantly based on the length of the given sequences. Note that any resulting subsequence can only be as long as the minimum length of the given sequences. Thus, the normalized similarity should be always less than two, and greater than zero. To make the score be in the range from zero to one, we further divide the score by two, which is the maximum similarity score we gave.

For instance, given two sequences $s_1 = a_1, a_2, \ldots, a_m$ and $s_2 = b_1, b_2, \ldots, b_n$ where $m < n$, let us assume that we have obtained the most similar subsequence $s_3 = c_1, c_2, \ldots, c_l$, and the sum of the similarity score for s_3 was x. We then normalize the sum with: $\frac{x}{2m}$. Each element in the resulting matrix M represents a normalized similarity score.

3.3 Draw

The final step of AlertVision is to draw a graph based on the similarity matrix M we computed in the Align phase. Each nodes in the resulting graph represents a sequence of alerts generated in the Preprocess step. The key challenge here is to decide where to place each node in a graph because there is no such notion as position for each of the sequences. To draw a graph based only on the relative distances between nodes, we leverage force-directed graph drawing [7] discussed in

Sect. 2.2. To represent the relationship between nodes, we draw edges only when two nodes are similar to each other based on our similarity measure. Specifically, we draw an edge between two nodes when their similarity score is higher than 0.9, i.e., 90%. The algorithm starts by placing every node in random positions in a two-dimensional coordinate plane, and terminates when all the nodes are in an equilibrium state.

4 Evaluation

We now evaluate AlertVision on real-world alert logs obtained from real SIEM devices. Specifically, we answer the following questions to evaluate our system.

1. Can we observe some meaningful correlation between alert sequences that are close to each other in a graph generated from AlertVision? (Sect. 4.2)
2. How do sequence clusters change over time? Can we see similar clusters over time? (Sect. 4.3)
3. Is there a specific attack incident that we can identify from the generated graphs? (Sect. 4.4)

4.1 Experimental Setup

We collected 6-months (from January to June in 2017) logs from real SIEM devices installed in 85 enterprises, which constitute 5,801,619 alerts for NIDS in total. There were 96,260 unique source IP addresses used in the alerts excluding private IP addresses; since one private IP address does not stand for one independent attacker, we disregarded private IP addresses. We ran Preprocess to make a mapping from a source IP to an alert message, which resulted in 96,260 mappings in total. We then removed mappings which have a sequence of only a single alert. Note that such a short sequence cannot affect the result of Smith-Waterman algorithm and removing them can help reduce overhead of Align. As a result, we obtained 29,268 unique attack sequences in total. In the rest of this section, we discuss our research questions based on the results of Preprocess.

4.2 Alert Sequence Correlation

We ran Align and Draw on the sequences obtained in Sect. 4.1. Figure 2 presents six graphs we obtained by running AlertVision on monthly logs from Jan. 2017 to Jun. 2017. Each node (dot) in the graphs represents a sequence, i.e., a group of alerts. The graphs clearly show which sequences are similar to each other: we can easily recognize clusters of nodes from the graphs. We found that each cluster in the graphs contain similar attack sequences. For example, SQL injection attacks formed a large cluster in each of the graphs, and several web-based attacks such as XSS and XPATH injection formed multiple clusters that were close to each other. To further analyze the correlation between the alerts, we grouped the nodes based on their attack characteristics.

Particularly, there were 504 unique attack signatures in our dataset, and we manually categorized them into six categories based on their attack characteristics: (1) SQL injection, (2) vulnerability scanning, (3) XSS, (4) SSH password guessing, (5) web-based attacks, and (6) known CVE exploitation. We separated the XSS group with the web-based attack group because we found relatively

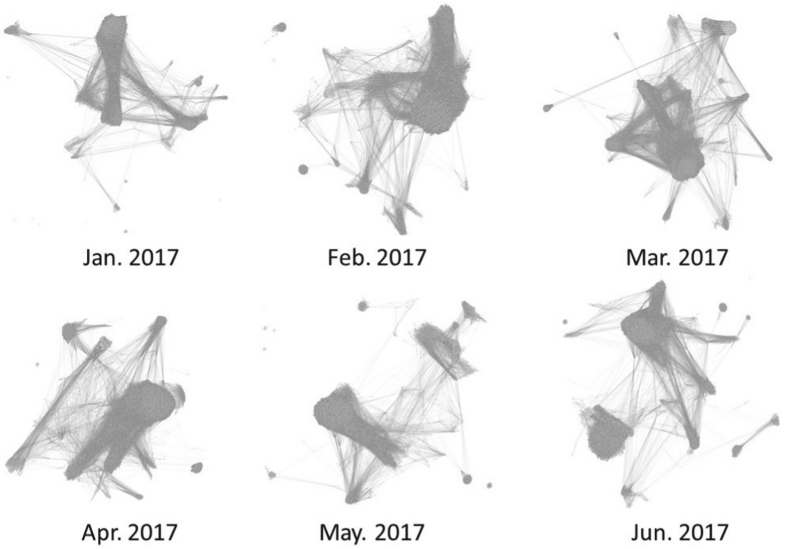

Fig. 2. Visualization of 6-month alert logs we collected from real-world SIEM devices.

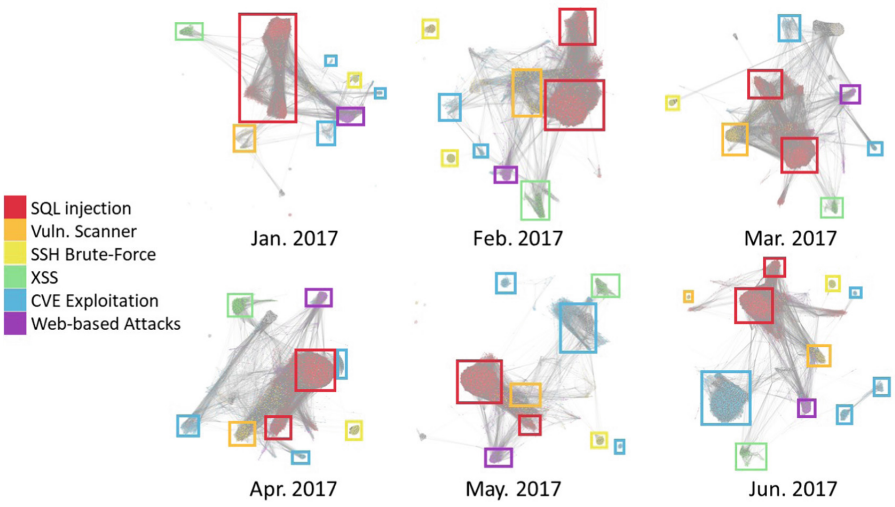

Fig. 3. Visualization of 6-month alert logs with categorization. (Color figure online)

many attack instances for XSS compared to other web-based attacks. The CVE exploitation group includes any attacks that are associated with known CVE. For example, we observed many exploits on Apache Struts in early 2017, which is associated with CVE-2017-5638.

Table 1. Attack reuse rate for each attack type based on the nodes in Jan. 2017.

Attack type	Feb.	Mar.	Apr.	May	Jun.
SQL injection	16.5%	11.6%	7.3%	5.6%	4.3%
Vulnerability scanning	63.8%	62.7%	70.4%	51.9%	54.4%
XSS	30.5%	27.2%	37.3%	27.8%	17.0%
SSH Brute-Forcing	19.8%	8.4%	5.5%	3.2%	1.9%
Web-based attacks	30.2%	15.1%	9.3%	11.2%	12.0%
CVE exploitation	18.8%	18.8%	18.8%	12.5%	6.3%

Figure 3 shows nodes in each of the groups in different colors. It is obvious from the graphs that our automated graph visualization algorithm was able to cluster attack sequences into meaningful clusters.

4.3 Attacks over Time

Do clustered sequences in our graphs change over time? We found that the same IP addresses tend to perform distinct attacks over time. For example, 83% of nodes that performed XSS in Jan. 2017 used different attack vectors other than XSS in Jun. 2017. Table 1 summarizes the attack reuse rate, which is the rate between the number of nodes that reuse the same attack type and the total number of nodes, for each attack type we consider. We computed the reuse rate based on the nodes in the graph of Jan. 2017. For example, only 4.3% of the nodes used for SQL injection in Jan. 2017 were used for SQL injection again in Jun. 2017. Notably, over 50% of the vulnerability scanners were using the same IP addresses over time.

We note that we can easily identify such a change by analyzing graphs with AlertVision, because we can easily highlight specific nodes when drawing graphs. Furthermore, the current implementation of AlertVision provides a graphical user interface that allows analysts to click nodes in the graph to see detailed information about them.

4.4 TI Case Study

Does the information that we obtained from AlertVision match with existing threat intelligence? To answer this question, we checked if any of the attackers' IP addresses in our dataset are listed in the IBM X-Force TI service [12]. We found that several known IP addresses for attackers in the TI were indeed in

the same group in our graphs. Figures 4 and 5 show that known command-and-control (C&C) servers and botnet addresses from the TI were in the same group in the graphs, respectively. Both figures illustrate a case for Jan. 2017, but the same trend appears in other graphs.

This result signifies the value of AlertVision as a tool that helps analysts understand the meaning of the attacks. For example, the IBM TI shows some of the nodes in our graphs are identified as a bot, but other nodes in the graph that are close the identified bots may be other bots controlled by the same botnet master as their behaviors are the same as the identified bots.

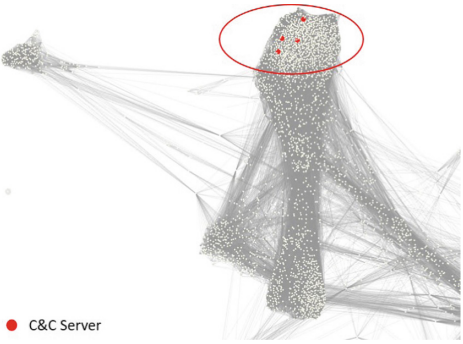

Fig. 4. Clustered C&C servers.

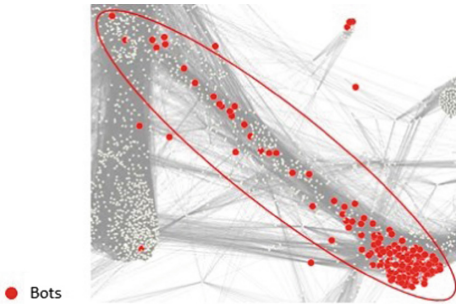

Fig. 5. Clustered botnet bots.

5 Related Work

Leveraging data mining and big data analytics for security has a long history. For instance, behavior-based anomaly detection [6,18] is a powerful defense mechanism that is still being used today. However, such techniques only focus on detecting attack instances, but not on identifying and analyzing the actors of the attacks.

Many researchers have recently turned their attention to refining security data obtained from various sources to build TI and to understand the meaning of threat instances due to recent advances in security threats. There are currently several attempts to classify threats [5,13,16,21,35,39] by leveraging ontology formally defined for describing security threats [5]. Although effective, those approaches are largely manual. Several attempts to defining data structures for TI have been made too. STIX [1] provides a unified way for expressing TI. Qamar *et al.* [29] recently extends STIX to represent semantics and contextual information of TI. Kapetanakis *et al.* [14] leverage traces on the victim machines left by attackers, e.g., modified/deleted files or registry entries, in order to generate attacker profiles. However, collecting such information is not feasible in practice as it requires installing host-based logging application for every machine, which may raise privacy concerns. On the other hand, our approach

only uses the existing SIEM events in order to generate profiles. Furthermore, our system visualizes the relevance between security alerts, which can provide valuable insight for the TI analysts. Note that AlertVision presents a *unique design point* in mining useful knowledge from security alerts with visualization. Therefore, our technique is complementary to the existing works.

There have been a wide range of research on correlating similar SIEM events, which is often called, alert correlation [20,31,40]. Alert correlation techniques are used to detect botnets [9,17] as well as to discover attack patterns from alert logs [4,23,30,32,38]. Ours is in the same line of research, but our focus is not on correlating attack instances themselves, but on visually representing the similarity between attack sources.

Visualizing security alerts has been studied by several researchers, but they mostly focus on how to graphically representing the raw data itself, but not on visualizing the meaning of them. Some of them can only be applied to specific attack types such as Worm [10] and DoS attacks [22]. Livnat *et al.* [19] propose a general method for representing alerts based on their detection time and their location in a network topology, but it does not capture the correlation between those alerts.

6 Conclusion

In this paper, we presented a novel visualization technique for providing practical insights for security analysts. We applied our technique on a large-scale dataset obtained from real enterprise networks, and showed its effectiveness in terms of understanding attacks and extracting TI from alert logs. The proposed technique is indeed used internally now in AhnLab, Korea.

Acknowledgements. We thank anonymous reviewers for their helpful feedback. This research was supported by AhnLab.

References

1. Barnum, S.: Standardizing cyber threat intelligence information with the structured threat information expression (STIX^{TM}). Technical report, MITRE (2012)
2. Cha, S.K., Moraru, I., Jang, J., Truelove, J., Brumley, D., Andersen, D.G.: SplitScreen: enabling efficient, distributed malware detection, pp. 377–390 (2010)
3. Coull, S., Branch, J., Szymanski, B., Breimer, E.: Intrusion detection: a bioinformatics approach. In: Proceedings of the Annual Computer Security Applications Conference, pp. 24–33 (2003)
4. Cuppens, F., Ortalo, R.: LAMBDA: a language to model a database for detection of attacks. In: Proceedings of the International Workshop on the Recent Advances in Intrusion Detection, pp. 197–216 (2000)
5. Fenz, S., Ekelhart, A.: Formalizing information security knowledge. In: Proceedings of the International Symposium on Information, Computer, and Communications Security, pp. 183–194 (2009)

6. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for Unix processes. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 120–128 (1996)
7. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Softw.: Pract. Exp. **21**(11), 1129–1164 (1991)
8. Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. **162**(3), 705–708 (1982)
9. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the USENIX Security Symposium, vol. 5, pp. 139–154 (2008)
10. Heoh, S.T., Ma, K.L., Wu, S.F., Zhao, X.: Case study: interactive visualization for internet security. In: Proceedings of the IEEE Conference on Visualization, pp. 505–508 (2002)
11. Huang, X., Miller, W.: A time-efficient, linear-space local similarity algorithm. Adv. Appl. Math. **12**(3), 337–357 (1991)
12. IBM: IBM X-Force threat intelligence. https://www.ibm.com/security/xforce
13. Jouini, M., Rabai, L.B.A., Aissa, A.B.: Classification of security threats in information systems. Procedia Comput. Sci. **32**, 489–496 (2014)
14. Kapetanakis, S., Filippoupolitis, A., Loukas, G., Murayziq, T.S.A.: Profiling cyber attackers using case-based reasoning. In: Proceedings of the UK Workshop on Case-Based Reasoning (2014)
15. Kirat, D., Vigna, G.: MalGene: automatic extraction of malware analysis evasion signature. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 769–780 (2015)
16. Kotenko, I., Polubelova, O., Saenko, I., Doynikova, E.: The ontology of metrics for security evaluation and decision support in SIEM systems. In: Proceedings of the International Conference on Availability, Reliability and Security, pp. 638–645 (2013)
17. Lee, K., Kim, J., Kwon, K.H., Han, Y., Kim, S.: DDoS attack detection method using cluster analysis. Expert Syst. Appl. **34**(3), 1659–1665 (2008)
18. Lee, W., Stolfo, S.J.: Data mining approaches for intrusion detection. In: Proceedings of the USENIX Security Symposium, pp. 79–93 (1998)
19. Livnat, Y., Agutter, J., Moon, S., Erbacher, R.F., Foresti, S.: A visualization paradigm for network intrusion detection. In: Proceedings of the Annual IEEE SMC Information Assurance Workshop, pp. 92–99 (2005)
20. Luh, R., Marschalek, S., Kaiser, M., Janicke, H., Schrittwieser, S.: Semantics-aware detection of targeted attacks: a survey. J. Comput. Virol. Hacking Tech. **13**(1), 47–85 (2017)
21. Luh, R., Schrittwieser, S., Marschalek, S.: TAON: an ontology-based approach to mitigating targeted attacks. In: Proceedings of the International Conference on Information Integration and Web-based Applications and Services, pp. 303–312 (2016)
22. McPherson, J., Ma, K.L., Krystosk, P., Bartoletti, T., Christensen, M.: PortVis: a tool for port-based detection of security events. In: Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security, pp. 73–81 (2004)
23. Mirheidari, S.A., Arshad, S., Jalili, R.: Alert correlation algorithms: a survey and taxonomy. In: Wang, G., Ray, I., Feng, D., Rajarajan, M. (eds.) CSS 2013. LNCS, vol. 8300, pp. 183–197. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03584-0_14

24. Myers, E.W., Miller, W.: Optimal alignments in linear space. Bioinformatics **4**(1), 11–17 (1988)
25. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**(3), 443–453 (1970)
26. Okada, D., Ino, F., Hagihara, K.: Accelerating the Smith-Waterman algorithm with interpair pruning and band optimization for the all-pairs comparison of base sequences. BMC Bioinform. **16**(1), 321 (2015)
27. O'Kane, P., Sezer, S., McLaughlin, K.: Obfuscation: the hidden malware. IEEE Secur. Priv. **9**(5), 41–47 (2011)
28. de Oliveira Sandes, E.F., de Melo, A.C.M.A.: Retrieving Smith-Waterman alignments with optimizations for megabase biological sequences using GPU. IEEE Trans. Parallel Distrib. Syst. **24**(5), 1009–1021 (2013)
29. Qamar, S., Anwar, Z., Rahman, M.A., Al-Shaer, E., Chu, B.T.: Data-driven analytics for cyber-threat intelligence and information sharing. Comput. Secur. **67**, 35–58 (2017)
30. Ramaki, A.A., Amini, M., Atani, R.E.: RTECA: real time episode correlation algorithm for multi-step attack scenarios detection. Comput. Secur. **49**, 206–219 (2015)
31. Salah, S., Maciá-Fernández, G., DíAz-Verdejo, J.E.: A model-based survey of alert correlation techniques. Comput. Netw. **57**(5), 1289–1317 (2013)
32. Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., Rajarajan, M.: Intrusion alert prioritisation and attack detection using post-correlation analysis. Comput. Secur. **50**, 1–15 (2015)
33. Sibson, R.: SLINK: an optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973)
34. Smith, T., Waterman, M.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)
35. Spring, J., Kern, S., Summers, A.: Global adversarial capability modeling. In: Proceedings of the IEEE eCrime Researchers Summit on Anti-phishing Working Group, pp. 1–21 (2015)
36. Strasburg, C., Basu, S., Wong, J.S.: S-MAIDS: a semantic model for automated tuning, correlation, and response selection in intrusion detection systems. In: Proceedings of the IEEE International Conference on Computer Software and Applications Conference, pp. 319–328 (2013)
37. Tankard, C.: Advanced persistent threats and how to monitor and deter them. Netw. Secur. **2011**(8), 16–19 (2011)
38. Treinen, J.J., Thurimella, R.: A framework for the application of association rule mining in large intrusion detection infrastructures. In: Proceedings of the International Workshop on the Recent Advances in Intrusion Detection, pp. 1–18 (2006)
39. de Vergara, J.E.L., Vázquez, E., Martin, A., Dubus, S., Lepareux, M.N.: Use of ontologies for the definition of alerts and policies in a network security platform. J. Netw. **4**(8), 720–733 (2009)
40. Zhou, C.V., Leckie, C., Karunasekera, S.: A survey of coordinated attacks and collaborative intrusion detection. Comput. Secur. **29**(1), 124–140 (2010)