SAGE: Intrusion Alert-driven Attack Graph Extractor

Azqa Nadeem*
Delft University of Technology

Sicco Verwer†
Delft University of Technology

Shanchieh Jay Yang‡
Rochester Institute of Technology

ABSTRACT

Attack graphs (AG) are used to assess pathways availed by cyber adversaries to penetrate a network. State-of-the-art approaches for AG generation focus mostly on deriving dependencies between system vulnerabilities based on network scans and expert knowledge. In real-world operations however, it is costly and ineffective to rely on constant vulnerability scanning and expert-crafted AGs. We propose to automatically learn AGs based on actions observed through intrusion alerts, without prior expert knowledge. Specifically, we develop an unsupervised sequence learning system, SAGE, that leverages the temporal and probabilistic dependence between alerts in a suffix-based probabilistic deterministic finite automaton (S-PDFA) – a model that accentuates infrequent severe alerts and summarizes paths leading to them. AGs are then derived from the S-PDFA on a per-objective, per-victim basis. Tested with intrusion alerts collected through Collegiate Penetration Testing Competition, SAGE compresses over 330k alerts into 93 AGs. These AGs reflect the strategies used by the participating teams. The AGs are succinct, interpretable, and capture behavioral dynamics, *e.g.,* that attackers will often follow shorter paths to re-exploit objectives.

Index Terms: Security and privacy—Intrusion/anomaly detection and malware mitigation—Intrusion detection systems; Human-centered computing—Visualization—Visualization application domains—Visual analytics; Computing methodologies—Machine learning—Learning paradigms—Unsupervised learning;

1 INTRODUCTION

Security Operation Centers (SOCs) typically receive thousands of intrusion alerts on a daily basis[1]. While alert correlation techniques help to reduce alerts from intrusion detection systems (IDS) [1, 26, 27], they do not show the attack progression and attacker strategies, *i.e.,* they show *what* the attackers did, but do not provide insight into *how* the infrastructure was exploited.

Attack graphs (AG) are models of attacker strategies that have been widely used for visual analytics [2, 5] and network hardening [15, 16]. Existing approaches to generate AGs are expensive due to their expert-driven nature — they utilize a significant amount of prior knowledge [1, 21, 22] and published vulnerability reports [9, 12, 23, 25]. In real-world operations however, it is costly and ineffective to rely on constant vulnerability scanning and expert-crafted AGs. Meanwhile, SOCs often possess large volumes of intrusion alerts from prior security incidents. We show, for the first time, that these alerts can be used as a basis to generate AGs.

In this paper, we propose SAGE — IntruSion alert-driven Attack Graph Extractor. SAGE leverages sequence learning to mine patterns from intrusion alerts, models them using an automaton, and represents them in the form of an attack graph. The two core phases of SAGE are shown in Figure 1. A tool such as SAGE can have important implications for the training and evaluation of defensive controls. SAGE augments existing IDSs by compressing several thousands of alerts into a handful of AGs. SOC analysts can triage alerts by visualizing and filtering AGs of interest.

Class imbalance presents a major challenge for machine learning (ML) based attacker strategy identification — severe alerts are scarce, and low-severity alerts (produced by network scans) are prevalent, which are not very valuable for analysts [29]. While most ML solutions discard infrequent patterns, we propose a suffix-based probabilistic deterministic finite automaton (S-PDFA) — a model that accentuates infrequent severe alerts, without discarding low-severity alerts. The S-PDFA summarizes attack paths leading to severe attack stages. It can differentiate between alerts that have identical signatures but different contexts, *e.g.,* scanning at the start, and scanning midway through an attack are treated differently because the former indicates reconnaissance and the latter indicates attack progression. AGs are then extracted from the S-PDFA on a per-objective, per-victim basis. A vertex in an AG represents a group of alerts generated by an attacker action, and an edge captures the temporal relationship between actions (as determined by the S-PDFA), showing attack progression. These graphs not only enable forensic analysis of prior security incidents, they also unlock a new means to derive intelligence regarding attacker strategies without having to investigate thousands of intrusion alerts.

We demonstrate the effectiveness of SAGE on distributed multi-stage attack scenarios, *i.e.,* where multi-member teams progress through various attack stages in order to compromise numerous targets. Penetration testing competitions provide an ideal setting to study such attacks. To this end, we use open-source intrusion alerts collected through Collegiate Penetration Testing Competition (CPTC)[2]. 93 AGs are generated from ∼330k alerts. The AGs reflect the actual pathways taken by the penetration testers, even with an imperfect IDS. They are succinct, and effective in highlighting strategic differences between participating teams. They reveal behavioral dynamics, *e.g.,* that attackers often follow shorter paths to re-exploit an objective after they have discovered a longer one. We also show how to rank attackers based on the uniqueness and severity of their actions. Thus, our contributions are:

1. We develop SAGE, a tool that automatically generates succinct high-severity attack graphs from intrusion alerts, without prior knowledge about vulnerabilities or network topology.
2. We apply SAGE on alerts from a penetration testing competition. The AGs are effective in attacker strategy comparison.

2 RELATED WORK

Attack graphs (AG) are a frequent area of research in the VizSec community. Kaynar *et al.* [16] proposed a taxonomy of the existing expert-driven AG generation approaches in the network security domain. These approaches generally utilize a knowledge base, making them unsuitable for zero-day vulnerabilities. Specifically, MulVAL [23] is an attack graph generator that has been widely used as a foundation for other works [12, 25], which takes the network topology and vulnerability information as input. Other techniques focus on path reachability [5, 32] and complexity reduction of expert-driven AGs [11, 14], instead of exploring additional data sources

*e-mail: azqa.nadeem@tudelft.nl
†e-mail: s.e.verwer@tudelft.nl
‡e-mail: Jay.Yang@rit.edu

[1]https://blog.paloaltonetworks.com/2020/09/secops-analyst-burnout/

[2]https://www.globalcptc.org/

2639-4332/21/$31.00 ©2021 IEEE
DOI 10.1109/VizSec53666.2021.00009

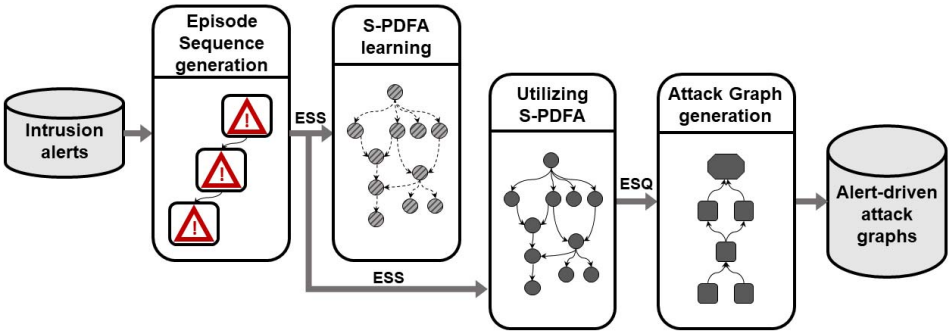

Figure 1: SAGE workflow: Intrusion alerts go in, attack graphs come out. An S-PDFA is learned in the first phase, and the model is utilized in the second phase to extract *alert-driven attack graphs*.

for AG construction. In addition, *Process mining* (PM) has been used to visualize alert datasets [4, 6] without actually extracting AGs. *Hidden markov models* (HMM) have been used to build alert forecasting systems [10], and *markov chains* have been used to build alert correlation systems [8]. Specifically, Moskal *et al.* [19] have used markov chains to model attacker strategies from intrusion alerts in the form of sequences. They use Jenson-Shannon divergence to measure similarity between such sequences. These approaches do not construct AGs and have several shortcomings if used to this aim: PM uses alert signatures as identifiers, which makes it impossible to differentiate alerts with identical signatures but different contexts, *i.e.,* those that lead to different paths. Markov chains have a similar weakness. HMMs do model context, but they are considerably difficult to interpret due to their non-deterministic nature. In this paper, we borrow initial ideas from [19] and leverage the temporal and probabilistic dependence between alerts to generate alert-driven AGs. The probabilistic deterministic finite automaton (S-PDFA) that SAGE uses has more expressive power than markov chains, *i.e.,* it does model context, and is easier to interpret. We show how to construct objective-oriented AGs that visualize large volumes of alerts without high cognitive load. To the best of our knowledge, SAGE is the first successful approach for this challenging problem.

3 Alert-driven Attack Graphs

SAGE (IntruSion alert-driven Attack Graph Extractor)[3] takes raw intrusion alerts as input, and transforms them into aggregated sequences that are used to learn a model summarizing attack paths in the data. Attack graphs (AG) are extracted from this model on a per-objective, per-victim basis (see Figure 1). The AGs are succinct and interpretable as they compress large volumes of alerts in order to show how an attack transpired. They also provide an effective means for attacker strategy comparison. SAGE is agnostic to host and network properties. It is released as a docker container for cross-platform support.

The first step towards building AGs is to arrange intrusion alerts in sequences that characterize an attacker strategy. An IDS alert contains the attacker and victim IP addresses, the targeted service *tServ* derived from destination port[4], and the attack stage *mcat* derived from the existing Action-Intent framework by Moskal *et al.* [18] (see appendix), based on MITRE ATT&CK [28]. Raw intrusion alerts are often noisy and contain duplicates. Thus, cleaning and aggregating them is necessary. We aggregate alerts into groups, such that they likely belong to the same attacker action. In literature, such an aggregation is called a hyper-alert or an attack episode. Grouping alerts that appear in bursts is a common way to construct episodes. We use the method in [19] to aggregate alerts into *attack episodes*, and assume that the episodes closely characterize attacker actions. For an attack stage *mcat*, an episode is defined as $\langle st, et, mcat, mServ \rangle$, where *st* and *et* are start/end times, and *mServ* is the most frequently targeted service during the episode. We create time-sorted *episode sequences* (ES) for each (attacker,victim) combination, and partition the ES whenever a low-severity episode follows a high-severity one, signaling the start of a new attack attempt (see $mcat \rightarrow$ severity mapping in appendix). These are called the *episode sub-sequences* (ESS).

We propose a *suffix-based probabilistic deterministic finite automaton* (S-PDFA), which is a suffix-variant of the probabilistic deterministic finite automaton [30]. Instead of predicting the future, the S-PDFA can be used to predict the past. Since the high-severity *mcat*'s are at the end of episode sub-sequences, we specifically learn a suffix model to determine which episodes eventually lead to high-severity attack stages. We provide all the ESS's from CPTC-2018 to the Flexfringe automaton learning framework [31]. Flexfringe uses univariate *symbol* sequences comprised of $\langle mcat, mServ \rangle$ to learn the S-PDFA (see appendix). The model summarizes attack paths in the dataset and clusters them based on behavioral similarity. It also brings infrequent high-severity actions into the spotlight, without discarding low-severity ones. This is tricky because while most clustering approaches discard infrequent patterns, the S-PDFA salvages them by setting appropriate parameters in Flexfringe (see Section 4), which also results in an interpretable model.

The states in an S-PDFA can be considered as milestones achieved by attackers, providing contextual meaning to the episodes' attack stages. Prior work by Lin *et al.* [17] has utilized this context to cluster similar car-following behaviors. We follow the same idea and convert episode sequences into state sequences (ESQ): we replay each ESS through the S-PDFA and augment it with state identifiers (*sID*), resulting in its corresponding ESQ. Finally, the ESQs are transformed into a graph (AG) via Graphviz (see Figure 2). These graphs are generated on a per-objective (obj), per-victim (*vic*) basis. An obj is defined as $\langle mcat, mServ, sID \rangle$, *i.e.,* one of the high-severity attack stages from [18] (since they specify end-goals), the targeted service, and the state identifier. For an AG with the root vertex $\langle vic, obj \rangle$, only the ESQs concerning the victim *vic*, and containing an episode with obj are included. If an obj is achieved multiple times in an ESQ, each attempt is shown as an individual path in the graph. The S-PDFA may assign different *sID*'s to the same $\langle mcat, mServ \rangle$, corresponding to the different contextual means of obtaining the obj, each of which appears as a sub-objective in the graph. Also, all attackers that obtain obj are shown in one graph to aid strategy comparison. Thus, an AG is a compressed representation of intrusion alerts related to $\langle vic, obj \rangle$. It shows how an attack transpired, including similarities between attacker strategies.

[3] https://github.com/tudelft-cda-lab/SAGE

[4] Derived from open-source Port→Service mapping from IANA.

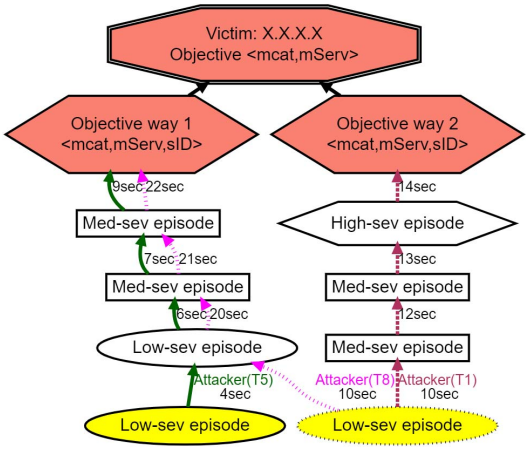

Figure 2: Notional alert-driven AG showing paths towards an objective. *Vertex labels* are $\langle mcat, mServ, sID \rangle$, *i.e.,* attack stage, targeted service and state identifier. Low-severity actions are *ovals*, medium-severity are *boxes*, high-severity are *hexagons*. The first action in a path is *yellow*, while the objective-variants are *red*. Actions that occur too infrequently for Flexfringe are *dotted*. Edge label shows time since first alert. Edge style shows team: T1 (*Dashed*), T5 (*Solid*), T8 (*Dotted*).

Table 1: Workload reduction in the CPTC-2018 dataset.

	Alerts (raw)	Alerts (filtered)	Episodes	ES/ ESQ	ESS	AGs
T1	81373	26651	655	103	108	53
T2	42474	4922	609	86	92	7
T5	52550	11918	622	69	74	51
T7	47101	8517	576	63	73	23
T8	55170	9037	439	67	79	33
T9	51602	10081	1042	69	110	30

4 Experimental Setup

Dataset. We generate attack graphs for the open-source Collegiate Penetration Testing Competition dataset, *i.e.,* CPTC-2018 [20]. It contains Suricata alerts generated by different student teams tasked with compromising a fictitious network, *i.e.,* an automotive company. Each team has access to fixed-IP machines. Beyond the attackers' IP information, no ground truth is available regarding the attacker strategies and attack progression. Six teams (*i.e.,* T1, T2, T5, T7, T8, T9) produce 330,270 alerts. The competition lasted 9 hours.

Parameter selection. SAGE has five parameters: we set $t = 1.0$ sec to discard repeated alerts [19], and window length $w = 150$ sec to aggregate alerts into episodes. We set three Flexfringe parameters: *symbol_count*, *state_count* and *sink_count*, all set to 5. These parameters are selected based on the properties of the dataset, primarily dependent on the frequency of severe alerts. The experiments are executed on Intel Xeon W-2123 quad-core processor, 32 GB RAM.

S-PDFA model quality. Quantifying S-PDFA model quality is a difficult problem [7, 24]. A common option is to measure its prediction power using *Perplexity* [3, 30]. Compared with suffix trees and markov chains, our S-PDFA achieves the best perplexity, showing its ability to capture patterns in the sequences (see appendix).

5 Results and Discussion

SAGE compresses 330,270 alerts into 93 attack graphs (AG), and discovers 70 objectives that are obtained by targeting 19 victim hosts. This leads to a considerable workload reduction. Table 1 shows this reduction on a per-team basis. Note that multiple teams can share one AG if they all obtain that objective. Furthermore, the AGs are succinct and effective in highlighting differences between attacker strategies. We evaluate the complexity of the AGs using the model simplicity metric given in [6], *i.e.,* $Simplicity(AG) = \frac{|V|}{|E|}$, where $|V|$ and $|E|$ are the number of vertices and edges, respectively. The AGs have an average simplicity of 0.81, with 21.7 vertices on average, where AGs with more than 30 vertices are considered as complex [6]. Each AG represents about 500 alerts, on average.

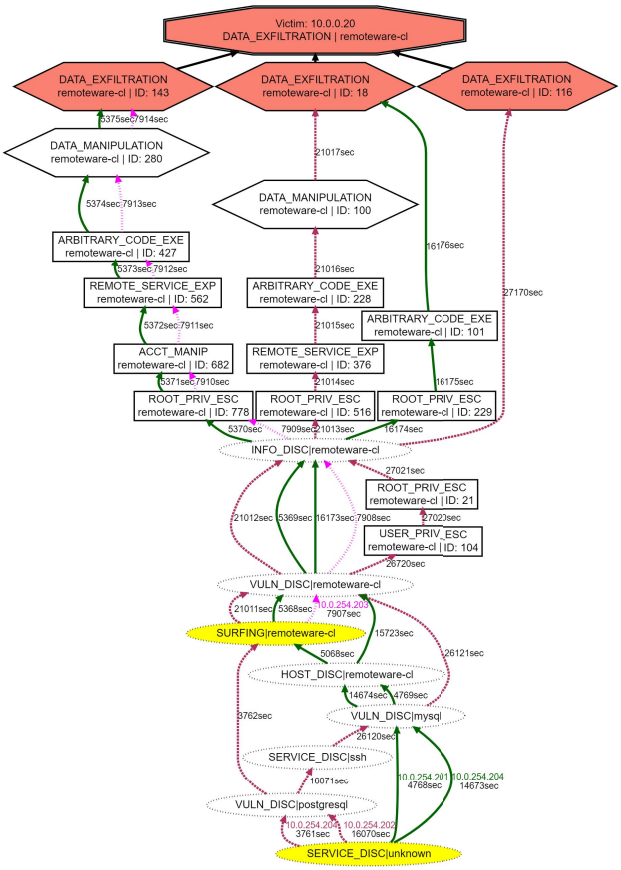

Figure 3: Data exfiltration/remoteware by T1, T5, and T8. T1 and T5 make two attempts, while T8 makes a single attempt. The S-PDFA discovers three contextual ways of exfiltrating data from this victim.

1. AGs show attack pathways: The AGs provide insight into attacker strategies. Figure 3 shows that three teams (T1, T5, T8) use remoteware-cl to exfiltrate data from 10.0.0.20 (the absence of other teams means they were unable to obtain this objective). The teams self-reported that they had found a chatting application on this host that contained credentials, which were exfiltrated using a combination of privilege escalation and arbitrary code execution. T1 finds two distinct paths to complete this objective, first after 5.8 hours and then again after 7.5 hours since the start of the competition. T5 also finds two paths, but considerably earlier in the competition. The S-PDFA identifies three contextually distinct exfiltration states based on the differences in the paths that lead up to the objective. For example, $\langle$`data_exfiltration, remoteware-cl, 116`$\rangle$ can be reached with much fewer steps compared to the others, and it also happens much later in the competition, implicitly capturing attackers' increasing experience. Moreover, in cases where multiple attack attempts are made, the subsequent attempt is shorter than the first in 84.5% of the cases, providing evidence for SAGE's ability to capture behavioral dynamics.

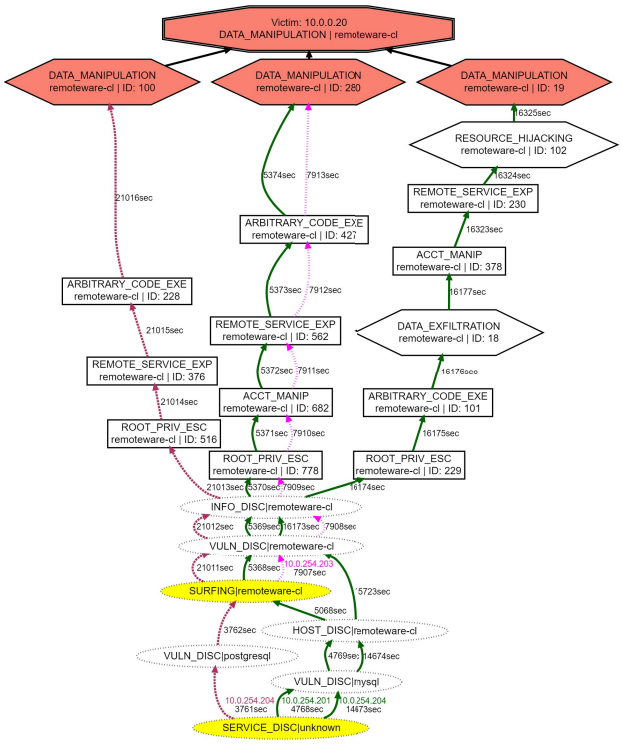

Figure 4: Data manipulation/remoteware is a sub-graph of Figure 3.

2. AGs show strategic differences: Interestingly, an AG of data manipulation (Figure 4) over the same victim and service from Figure 3 are partial sub-graphs of each other, due to overlap in paths that attain both objectives. There are three variants of data manipulation, of which two are also present in the exfiltration AG, *i.e.,* ⟨`data_manipulation, remoteware-cl, 100`⟩ and ⟨`data_manipulation, remoteware-cl, 280`⟩. T5 finds an additional path to reach ⟨`data_manipulation, remoteware-cl,` 19⟩ after it has reached the objective ⟨`data_exfiltration, remoteware-cl, 18`⟩ from the previous AG. This actionable intelligence can be used to disrupt the cyber kill-chain [13]. Additionally, the AG shows differences in attacker strategies, *e.g.,* T5 and T8 perform account manipulation while T1 does not; and resource hijacking is a step in one of T5's paths but not in the other. It also shows that T1 has found the shortest path to perform data manipulation on the victim using remoteware-cl.

3. AGs allow attacker performance evaluation: Each vertex in the attack graphs signifies a new milestone achieved by the teams. We argue that the fraction of unique milestones, *i.e.,* $\langle mcat, mServ, sID \rangle$, discovered by a team provides a metric for its performance. A medium-severity attack stage forms a stepping-stone towards a high-severity attack stage. Hence, high-severity vertices are twice as important as medium-severity vertices, *i.e.,* $\frac{(2*sev)+(1*med)}{3}$, where *sev* and *med* are the number of high- and medium-severity milestones discovered by a team, respectively. Table 2 shows the evaluation of all six teams based on the 93 AGs, ranked according to their score. It shows the number of unique high- and medium-severity vertices discovered by the teams during the competition. T5 scores the highest points, while T2 scores the lowest points. T1 comes in second, solely because they discover the highest number of medium-severity vertices compared to any other team. Overall, this metric provides a simple way to rank attackers based on the uniqueness and severity of the alerts they raise.

Table 2: Attacker evaluation based on the fraction of unique vertices discovered during CPTC-2018 (*rank = score*).

Team	Severe vertices (out of 70)	Medium vertices (out of 148)	Weighted average percentage
T5	28 (40%)	40 (27%)	35.67
T1	18 (26%)	62 (42%)	31.33
T9	23 (33%)	36 (24%)	30.0
T7	22 (31%)	26 (18%)	26.67
T8	15 (21%)	32 (22%)	21.33
T2	3 (4%)	8 (5%)	4.33

5.1 Discussion: Visual Analytics enabled by SAGE

Utilizing observables such as intrusion alerts to obtain intelligence regarding attacker strategies will noticeably benefit SOC analysts. Visualizing such strategies in a way that communicates the correct message to SOC analysts is another important challenge. SAGE is one of the first attempts toward addressing these challenges: SAGE extracts targeted (objective-oriented) attack graphs (AG) from intrusion alerts without prior knowledge. In doing so, SAGE opens up numerous research opportunities for the VizSec community.

- Although SAGE significantly reduces the volume of alerts to analyze, it still ends up with several AGs. The prioritization of AGs in general, and attack paths in particular, remains an open problem. A query mechanism to filter and replay specific parts of an attack, *e.g.,* vaguely similar to PERCIVAL [2], will be highly useful.
- SAGE can be used to improve IDS rules. It utilizes most of the alerts, which are aggregated for visual analytics. Executing test attacks for which no corresponding paths can be found in the resulting AGs hint towards missing or faulty IDS rules.
- Comparative visual analytics for AGs is another open challenge. For a given victim that is attacked by different attackers at different times, highlighting the attack progression similarity for visual comparison is an interesting direction.
- In forensic analysis, alert-driven AGs can be used to point towards specific victim machines for which additional evidence is required. This evidence can be used to corroborate the success of critical milestones for an investigated attack. For example, an AG containing data exfiltration step highlights the need for investigating other data sources to check whether data were indeed exfiltrated.

6 Conclusions and Future Work

In this paper, we propose SAGE, a novel unsupervised sequence learning tool that generates succinct attack graphs (AG) directly from raw intrusion alerts, without a priori knowledge. SAGE is capable of representing several thousands of alerts in just a handful of AGs, which is beneficial for alert triaging and visual analytics. SAGE models the temporal and probabilistic dependence between alerts in a suffix-based probabilistic deterministic finite automaton (S-PDFA). The S-PDFA brings infrequent severe alerts into the spotlight and summarizes paths leading to them. AGs are then extracted from the S-PDFA on a per-objective, per-victim basis. SAGE generates 93 AGs from ∼330k alerts collected through Collegiate Penetration Testing Competition with six attacker teams. The AGs provide a clear picture of the attack progression, and show strategic differences between attackers, *e.g.,* they show that attackers often follow shorter paths to re-exploit an objective. They are also used to rank interesting attackers based on the severity of the alerts they raise.

Future work will focus on applying SAGE to additional datasets, adding interactive capabilities to alert-driven AGs, evaluating AGs with security analysts, and leveraging readily-available domain knowledge to map AG vertices to high-level attacker actions.

REFERENCES

[1] F. M. Alserhani. Alert correlation and aggregation techniques for reduction of security alerts and detection of multistage attack. *International Journal of Advanced Studies in Computers, Science and Engineering*, 5(2):1, 2016.

[2] M. Angelini, N. Prigent, and G. Santucci. Percival: proactive and reactive attack and response assessment for cyber incidents using visual analytics. In *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8. IEEE, 2015.

[3] B. Balle, R. Eyraud, F. M. Luque, A. Quattoni, and S. Verwer. Results of the sequence prediction challenge (spice): a competition on learning the next symbol in a sequence. In *International Conference on Grammatical Inference*, pp. 132–136, 2017.

[4] Y. Chen, Z. Liu, Y. Liu, and C. Dong. Distributed attack modeling approach based on process mining and graph segmentation. *Entropy*, 22(9):1026, 2020.

[5] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer. Visualizing attack graphs, reachability, and trust relationships with navigator. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security (VizSec)*, pp. 22–33, 2010.

[6] S. C. De Alvarenga, S. Barbon Jr, R. S. Miani, M. Cukier, and B. B. Zarpelão. Process mining and hierarchical clustering to help intrusion alert visualization. *Computers & Security*, 73:474–491, 2018.

[7] C. De la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

[8] O. B. Fredj. A realistic graph-based alert correlation system. *Security and Communication Networks*, 8(15):2477–2493, 2015.

[9] N. Gao, Y. He, and B. Ling. Exploring attack graphs for security risk assessment: a probabilistic approach. *Wuhan University Journal of Natural Sciences*, 23(2):171–177, 2018.

[10] I. Ghafir, K. G. Kyriakopoulos, S. Lambotharan, F. J. Aparicio-Navarro, B. AsSadhan, H. BinSalleeh, and D. M. Diab. Hidden markov models and alert correlations for the prediction of advanced persistent threats. *IEEE Access*, 7:99508–99520, 2019.

[11] J. Homer, A. Varikuti, X. Ou, and M. A. McQueen. Improving attack graph visualization through data reduction and attack grouping. In *International Workshop on Visualization for Computer Security (VizSec)*, pp. 68–79. Springer, 2008.

[12] H. Hu, J. Liu, Y. Zhang, Y. Liu, X. Xu, and J. Huang. Attack scenario reconstruction approach using attack graph and alert data mining. *Journal of Information Security and Applications*, 54:102522, 2020.

[13] E. M. Hutchins, M. J. Cloppert, R. M. Amin, et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[14] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *2009 Annual Computer Security Applications Conference*, pp. 117–126. IEEE, 2009.

[15] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pp. 49–63. IEEE, 2002.

[16] K. Kaynar. A taxonomy for attack graph generation and usage in network security. *Journal of Information Security and Applications*, 29:27–56, 2016.

[17] Q. Lin, Y. Zhang, S. Verwer, and J. Wang. Moha: A multi-mode hybrid automaton model for learning car-following behaviors. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):790–796, 2018.

[18] S. Moskal and S. J. Yang. Framework to describe intentions of a cyber attack action. *arXiv preprint arXiv:2002.07838*, 2020.

[19] S. Moskal, S. J. Yang, and M. E. Kuhl. Extracting and evaluating similar and unique cyber attack strategies from intrusion alerts. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 49–54. IEEE, 2018.

[20] N. Munaiah, A. Rahman, J. Pelletier, L. Williams, and A. Meneely. Characterizing attacker behavior in a cybersecurity penetration testing competition. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–6. IEEE, 2019.

[21] 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*, pp. 245–254, 2002.

[22] P. Ning, D. Xu, C. G. Healey, and R. S. Amant. Building attack scenarios through integration of complementary alert correlation method. In *NDSS*, vol. 4, pp. 97–111, 2004.

[23] X. Ou, S. Govindavajhala, and A. W. Appel. Mulval: A logic-based network security analyzer. In *USENIX security symposium*, vol. 8, pp. 113–128. Baltimore, MD, 2005.

[24] R. Parekh and V. Honavar. Learning dfa from simple examples. *Machine Learning*, 44(1-2):9–35, 2001.

[25] S. Roschke, F. Cheng, and C. Meinel. A new alert correlation algorithm based on attack graph. In *Computational intelligence in security for information systems*, pp. 58–67. Springer, 2011.

[26] R. Sadoddin and A. Ghorbani. Alert correlation survey: framework and techniques. In *Proceedings of the 2006 international conference on privacy, security and trust: bridge the gap between PST technologies and business services*, pp. 1–10, 2006.

[27] S. Salah, G. Maciá-Fernández, and J. E. DíAz-Verdejo. A model-based survey of alert correlation techniques. *Computer Networks*, 57(5):1289–1317, 2013.

[28] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas. Mitre att&ck: Design and philosophy. *Technical report*, 2018.

[29] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer. Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on dependable and secure computing*, 1(3):146–169, 2004.

[30] S. Verwer, R. Eyraud, and C. De La Higuera. Pautomac: a probabilistic automata and hidden markov models learning competition. *Machine learning*, 96(1-2):129–154, 2014.

[31] S. Verwer and C. A. Hammerschmidt. Flexfringe: a passive automaton learning package. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 638–642. IEEE, 2017.

[32] L. Williams, R. Lippmann, and K. Ingols. Garnet: A graphical attack graph and reachability network evaluation tool. In *International Workshop on Visualization for Computer Security (VizSec)*, pp. 44–59. Springer, 2008.

A APPENDIX

A.1 Attack stages

Table 3 provides the Action-Intent mapping that is used by SAGE to map default alert signatures to attack stages.

Table 3: Attack stages and their severity from Moksal *et al.*

Acronym	Attack stage	Severity
SURFING	Surfing	Low
HOST_DISC	Host Discovery	Low
SERVICE_DISC	Service Discovery	Low
VULN_DISC	Vulnerability Discovery	Low
INFO_DISC	Information Discovery	Low
USER_PRIV_ESC	User Privilege Escalation	Med
ROOT_PRIV_ESC	Root Privilege escalation	Med
BRUTE_FORCE_CREDS	Brute force Credentials	Med
ACCT_MANIP	Account Manipulation	Med
PUBLIC_APP_EXP	Public Application Exploitation	Med
REMOTE_SERVICE_EXP	Remote Service Exploitation	Med
COMMAND_AND_CONTROL	Command and Control	Med
LATERAL_MOVEMENT	Lateral movement	Med
ARBITRARY_CODE_EXE	Arbitrary code execution	Med
PRIV_ESC	Privilege escalation	Med
NETWORK_DOS	Network Denial of Service	High
RESOURCE_HIJACKING	Resource hijacking	High
DATA_MANIPULATION	Data manipulation	High
DATA_EXFILTRATION	Data exfiltration	High
DATA_DELIVERY	Data delivery	High
DATA_DESTRUCTION	Data destruction	High

A.2 S-PDFA for CPTC-2018

Figure 5 shows the S-PDFA learned for the entire CPTC-2018.

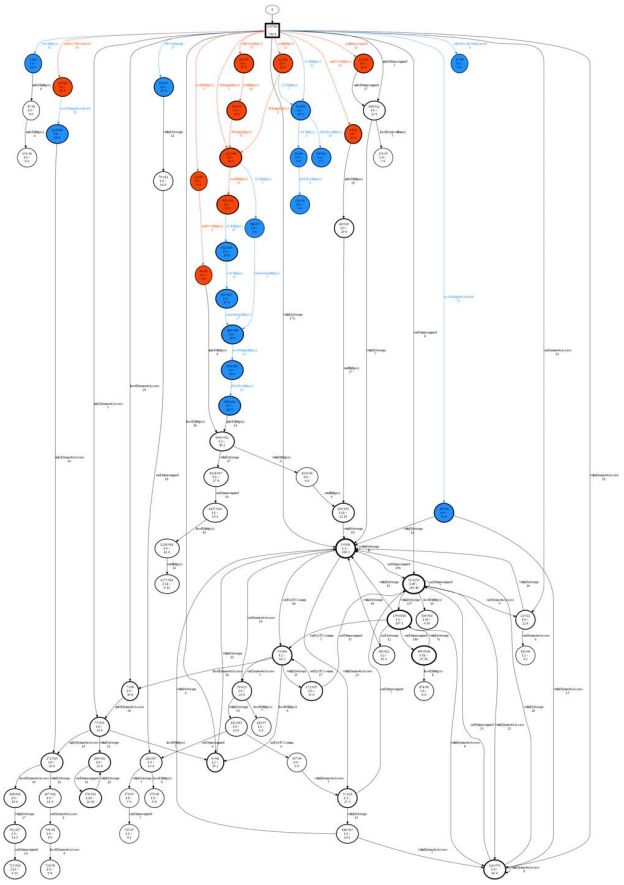

Figure 5: The S-PDFA for 6 teams in CPTC-2018. State color denotes severity: red = high, blue = medium, white = low.

A.3 S-PDFA model quality

Model quality is often quantified using a model selection criterion, measuring a trade-off between model size and fit. Perplexity is defined as $2^{-\frac{1}{N}\sum_{i=1}^{N} log_2 P(x_i)}$ where N is the number of traces and $P(x_i)$ returns the probability of the x_i trace. *The lower the value, the better the model fits with the data*. We compute perplexity for both the training data and an unseen test set using an 80-20 split. The former shows how well the model fits with the training data, and the latter shows how well the model captures patterns in the overall data. We compare the perplexity values against two suffix models: (a) suffix tree and (b) markov chains. Table 4 shows the perplexity for each variant on both training and test data. It shows that a suffix tree provides the best fit with the training data, as expected. The S-PDFA is about twice as "perplexed". On the test data, the S-PDFA gives the best perplexity value, demonstrating that the model accurately captures many of the patterns present in the data that are missed by the other models.

Table 4: Perplexity of suffix models (bold = best value).

	Suffix tree	**Markov chains**	**S-PDFA**
Training data	***1265.4***	13659.6	2397.8
Holdout test set	13020.7	11617.8	***9884.6***