Investigating the Impact of Sink State Merging on Alert-Driven Attack Graphs

The effects of allowing the sink states to merge with other sink states

Alexandru Dumitriu[1]

Supervisor(s): Sicco Verwer[1]**, Azqa Nadeem**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Alexandru Dumitriu
Final project course: CSE3000 Research Project
Thesis committee: Sicco Verwer, Azqa Nadeem, Asterios Katsifodimos

Abstract

This research paper focuses on the complex domain of alert-driven attack graphs. *SAGE* is a tool which generates such attack graphs (AGs) by using a suffix-based probabilistic deterministic finite automaton (*S-PDFA*). One of the substantial properties of this algorithm is to detect infrequent severe alerts while maintaining the context of attacks via the help of *sink states* and *state IDs*. This is a modelling assumption that we validate by answering the question driving this research: How does allowing the sink states to merge with other sink states affect the generated alert-driven attack graphs? Our research used a transparent methodology to obtain size, complexity and completeness statistics of the two algorithms. Afterwards, outstanding values in size and complexity allow us to filter insightful attack graphs, subject to head-to-head comparisons concerning their interpretability. We discovered that many remarkable changes happen in the outputted attack graphs, leading to an evident decrease in interpretability and increased loss of context. Concurrently, we do not detect substantial changes in size, complexity and completeness, leading us to believe that it is possible to unlock *SAGE*'s full potential by adding specific thresholds for merging sink states. One proposed constraint is allowing only the merges of sinks at equal distance to the victim node. This alteration leads to similar results in all metrics, including interpretability, where some AGs show improvements.

Index Terms: Alert-driven attack graphs, SAGE, Sink states, S-PDFA, Context

1 Introduction

Working in a Security Operations Centre (SOC) is a highly complex task for security analysts who manually scrutinise numerous alerts to comprehend the tactics employed by attackers [1]. The volume of alerts received daily is staggering and generated by an Intrusion Detection System (IDS), which leads to an unproductive and stressful work environment associated with the development of the 'threat alert fatigue' phenomenon [2].

Using attack graphs (AGs) is a viable option to use these alerts for visualising the strategies attackers use to infiltrate a network. Nevertheless, creating such a hypothetical view of a network requires a list of pre-existing vulnerabilities, network topology, and expert knowledge [3]. Unfortunately, most of those AGs present a substantial quantity of data and need to be better optimised to provide fast and reliable intelligence to security analysts.

On the other hand, *Nadeem et al.* worked on an AG generator *SAGE*, which uses a suffix-based probabilistic deterministic finite automaton (*S-PDFA*) to learn directly from intrusion alerts without any expert knowledge of the network topology [1]. *SAGE* manages to compress over a million intrusion alerts into less than 500 AGs, allowing forensic analysts to compare attack strategies generated through a suffix-based model, which makes it possible to detect infrequent severe alerts and models the semantics of alerts using interpretable merging criteria [4], [5]. The sink state is a critical fundamental concept that allows capturing such infrequent alerts because those are not used whilst learning the *S-PDFA* model. Since merging those could result in the algorithm inferring incorrect conclusions, *SAGE* removes them, and the states with medium or high severity are reintroduced in post-processing to showcase the full context to the cybersecurity analyst. The generated AGs aim to show what has happened on the network regarding attack episodes by providing a visual overview of what an intrusion detection system observes over time in a highly interpretable manner, allowing cybersecurity analysts to fix different security concerns.

This research area has a knowledge gap because of the need for a validation process for an unsupervised model. Therefore, this project will tackle this problem by learning a model similar to the initial *S-PDFA* with a different modelling assumption where the sinks can be merged with other sink states. We chose this specific difference in modelling assumptions because further investigating the role of sink states in *SAGE* could reveal multiple insights into how they can be used to increase the quality of AGs. The attack graphs generated by this slightly different model will be directly compared with those generated by the initial implementation regarding size, complexity, interpretability, and completeness.

To successfully evaluate *SAGE*'s quality, the aim will be to answer the following research question: *How does allowing the sink states to merge with other sink states affect the generated alert-driven attack graphs?* Successfully answering this question helps validate the performance of *SAGE*. Also, a slightly altered algorithm that uses the sink states differently to improve the current implementation according to the above mentioned metrics has been proposed.

Before conducting the experiments to try and answer this question, multiple hypothesises have been proposed:

- All of the sink states will convert to normal states.
- The interpretability of the AGs will decrease.
- The completeness will not be affected.
- The merges between sinks will lead to a loss of context.

We assess if these hypotheses were correct by following a strict methodology presenting multiple metrics for evaluation regarding size, complexity, interpretability and completeness and how these can be measured in alert-driven attack graphs.

The rest of the paper follows the subsequent structure. Section 2 delves deeper into the existing literature about alert-driven attack graphs and other related tools. Section 3 describes our comparison methodology, and we present the experimental setup in Section 4. Section 5 presents the results obtained, their discussion and a proposed improved merging criteria. The conclusions drawn from those are shared in Section 6 alongside any future work remarks. Finally, Section 7 approaches the research's reproducibility and integrity, whilst Section 8 contains the acknowledgements.

2 Related work

Due to the progress made in cybersecurity research, countless methods and models are now available for assessing computer systems' security, such as *alert-driven attack graphs*. This paper focuses on validating and enhancing an existing technology, namely, *SAGE*, which is a tool that generates such attack graphs (AGs) by using a suffix-based probabilistic deterministic finite automaton (*S-PDFA*) which is learnt with the help of *Flexfringe* [1], [6]. This section aims to explore attack graphs and the tools used to generate them in detail, analysing their advantages and limitations.

2.1 Attack Graphs

In the field of network security and risk management, attack graphs are a highly valuable tool. They depict multiple attack paths, which are a series of *"atomic attacks"*that an attacker may use to obtain administrative access to a severe element of the network [7].

Phillips and Swiler first proposed attack graphs in 1998 [8]. Since then, they have undergone substantial evolutions, with various algorithms and methods developed to optimise their creation and analysis, becoming a standard in the industry. With this visualisation technique, cybersecurity analysts can thoroughly understand the security vulnerabilities in their network and how attackers could exploit them.

However, creating and interpreting attack graphs can be complex because most AGs need to present a substantial quantity of data in a readable format, which leads to the need for optimisation in order to provide fast and reliable intelligence to security analysts. Another challenge is the scale of networks, such as the ones of enterprises, which can contain thousands of hosts, requiring efficient and real-time methods to generate valuable AGs [9].

There have been many proposed solutions to deal with these problems. Some of these solutions, like *SAGE*, which will be subsequently described, have shown potential in making it easier to manage attack graphs and improving their usefulness in assessing risks.

2.2 SAGE

SAGE is a tool that has been developed by *Nadeem et al.*, and it introduces the concept of *alert-driven attack graphs* [1]. Generating AGs directly from intrusion alerts, *SAGE* is able to detect infrequent severe alerts while maintaining the context of attacks via the help of *sink states*, which are not eliminated or merged during the *S-PDFA* learning process [1]. By taking advantage of this property, the algorithm manages to compress over 330k alerts into 93 different AGs, each containing a victim, its attackers and all possible attack paths [10]. An example of such an attack graph can be seen in Figure 1 alongside its legend in Figure 2.

2.3 FlexFringe

FlexFringe is an open-source tool for inferring deterministic finite automata (DFA) and has been used for the learning process of the *S-PDFA* model with specific parametrisation [6]. This model was specifically tuned to accentuate infrequent alerts and summarise attack paths leading to high severity nodes [1].

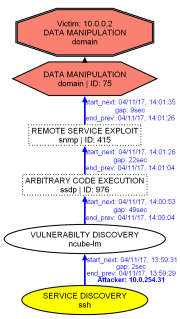

Figure 1: AG on *victim-10.0.0.2-DATA MANIPULATION*.

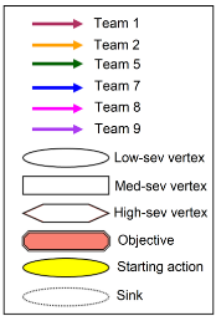

Figure 2: Legend for edges and nodes of an AG generated by *SAGE* [4].

3 Methodology

This section proposes the rigorous methodology used in this research. Subsection 3.1 presents the problem statement and is followed by the hypotheses presentation in Subsection 3.2. Afterwards, Subsection 3.3 presents the experimental workflow followed by the metrics defined in Subsection 3.4.

3.1 Problem Statement

A hard problem *SAGE* is currently facing is the side-effect of including high-severity sinks in the state sequences, which rarely leads to distinct objective types being displayed as similar in the corresponding AG [1]. Alongside the fact that the sink states are currently not used in the learning process, different modelling assumptions regarding their use should be tested to validate *SAGE's* performance and investigate possible improvements that mitigate such problems. The following methodology has been specifically designed to evaluate the current implementation of *SAGE* compared to a variation allowing merges between sink states.

3.2 Hypotheses

By comparing the AGs on the overall sets and head-to-head between individual matching attack graphs, we intend to evaluate the correctness of the following hypotheses based on the metrics which will be subsequently presented:

- Allowing the sinks to merge will lead to the disappearance of all sink nodes because they will either be merged with similar nodes or transformed into normal nodes.
- Interpretability will decrease in the implementation which allows the sink states to merge because the edge count will remain relatively consistent. However, the node count will decrease, leading to increased global density and difficulty in correctly following a path.
- There will only be no logic changes between the two implementations and, therefore, completeness. This is because the traces which generate the edges will not suffer any changes by allowing the sinks to merge.
- Due to the merges between sink states, the context will not be preserved because the cybersecurity analysts can interpret the AGs differently than they are intended when two states at different stages of the attack get merged.

3.3 Experimental Workflow

Firstly, the research was established by running SAGE with the original parameters and the *mergesinks* parameter set to one on CPTC 2017 and 2018 datasets [11]. Subsequently, the data underwent analysis using various statistical methods such as the percentage of attack graphs affected, average size delta, average simplicity delta, evolution in the presence of sinks, and the evolution in the number of objectives.

Following this, the insightful attack graphs were filtered using different techniques to narrow the data, such as detecting graphs transitioning from simple to complex or inverse. This fact was individually assessed for all graphs using a linear regression model. Finally, apart from these graphs, those showing the largest delta in size and complexity were selected for further head-to-head analysis regarding interpretability.

3.4 Metrics Definition

Our evaluation metrics consist of size, complexity, interpretability, and completeness, each playing a distinct role in the overall assessment. Size allows us to understand the breadth and depth of each graph, while complexity provides insight into the sophistication and intricacy of attack paths. Interpretability estimates the ease of understanding the graphs, a crucial aspect for cybersecurity analysts who must make efficient and correct decisions based on these graphical models. Finally, completeness ensures that all possible attack scenarios are adequately represented in the respective graphs.

Size

Size is the first metric we implemented. It is represented by the number of nodes an attack graph contains. We chose this definition of size because the number of nodes is highly representative of the size of an attack graph, unlike other values, such as the number of edges which can be much more representative of other metrics, as will be highlighted in the subsequent sections.

According to one of the hypotheses previously mentioned, which states that sink states will disappear from the attack graphs, we can expect great deltas in terms of size according to the number of sink states an attack graph contained in the baseline implementation output. Therefore, we chose this metric to allow the possibility of filtering out multiple insightful attack graphs and providing great general statistics in the two sets of attack graphs, such as the overall development of the average size.

Complexity

Complexity is a critical metric to consider in our comparison, and the chosen implementation is the one referenced in the paper describing the original implementation of *SAGE*, simplicity [1], [12]. Simplicity can be used to great extents in the context of graph representation and can be computed for each attack graph as:

$$Simplicity = \frac{Number\ of\ nodes}{Number\ of\ edges}. \quad (1)$$

This simplicity value can help filter out the insightful AGs because a high difference is due to many merges happening in the merge sinks implementation. Concurrently, this value is not always representative since very large or small attack graphs can be wrongly classified. Therefore, we have used linear regression to classify whether an attack graph is simple or complex [12].

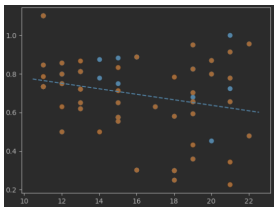

Figure 3: Simplicity over Size values plotted for baseline (Blue) and merge sinks (Orange) implementations.

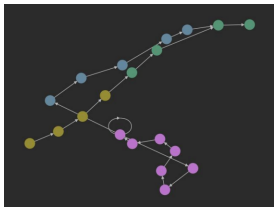

Figure 4: Communities detected in the AG in Figure 5 which are going to be used to compute modularity.

We constructed the linear regression model that can be seen in Figure 3 upon the following expert assumption: an attack graph containing less than 11 nodes can be automatically considered simple, whereas one containing at least 23 nodes can be automatically considered complex. Those values represent the 25th and 75th percentiles regarding size in the data, which is why the AGs that fall under these two categories can be automatically categorized as simple or complex. This led to the need to classify all the attack graphs with sizes between 11 and 23, and for this, a linear regression model was used where the line generated represents the boundary between simple and complex. Any attack graph which would reside below the line is complex, whereas any attack graph above the line is considered simple. This classification allowed us to observe attack graphs which were simple in the baseline implementation output but became complex after allowing the sink states to merge and the opposite. Further investigating into why this happened will lead to intriguing insights once the head-to-head comparison is complete and patterns of behaviour can be detected.

Interpretability

Interpretability can be considered the most critical metric in this comparison because a cybersecurity analyst using *SAGE* needs to understand the attack graphs generated to make the correct decisions. At the same time, interpretability is a concept hard to quantify. In the context of the AGs generated by *SAGE*, it is accomplished by comparing the two AGs regarding their readability and assessing whether there is any relative loss of context. This metric will help prove the hypothesis made regarding the fact that merging the sink states will lead to an overall loss in interpretability.

Readability

Readability measurements have been chosen because it is a very closely aligned field that is especially representative in the context of cybersecurity analysts needing to be highly efficient working with these attack graphs so that future breaches can be avoided.

According to a paper published in the field of Neuroscience on the topic of modular organization in the context of brain

networks, there exists a very well-known concept of always desiring a network to have low global density but high local density because of the benefits in robustness, readability and many other essential attributes of graphs [13].

In order to compute the global density, the following formula has been used, which was found in a paper which regards general concepts of readability in the sphere of graphs, and was afterwards adjusted to normalize the result with the help of the maximum number of edges possible [14]:

$$GlobalDensity = \sqrt{\frac{\textit{Number of edges}}{\textit{Number of nodes}^2}}. \quad (2)$$

We then computed the local density with the help of a Greedy version of the Louvain Community Detection Algorithm and obtained communities just like the one in Figure 4 [15]. After separating the attack graph in multiple communities, one can easily compute the modularity of those communities and obtain the local density with the following formula, which was first created by *M. E. J. Newman* and then simplified by *Clauset et al.* [16], [17]:

$$LocalDensity = \sum_{c=1}^{n}[\frac{L_c}{m} - \gamma(\frac{k_c}{2m})^2], \quad (3)$$

where c represents a community, n the number of communities, m the edges, k_c is the sum of degrees of the nodes, and γ is the resolution parameter.

By computing these two measurements for each pair of insightful graphs, we can asses if one proves to be more readable than another in a quantifiable manner, but these values sometimes tend to be very close, and without any significance analysis, those are not enough to draw such a complex conclusion. Therefore a Readability Protocol was created to be used as another method to assert which attack graph is more interpretable, but from a qualitative expert analysis point of view. This protocol was created based on a similar protocol developed for general graphs and was adjusted to fit the needs of a security analyst [14]. The protocol consists of the following steps:

1. Estimate the number of nodes
2. Estimate the number of attack paths
3. Locate a node based on a given label
4. Locate all medium-severity states
5. Locate all high-severity states
6. Follow an attack path from a start state to the victim state
7. Detect the three most important nodes

After completing the protocol, the results are assessed, and if all estimations and localizations are correct, it can safely be concluded that the graph is highly interpretable. In order to assess which is more interpretable, the time it took to complete the tasks for each AG is considered.

For step 7 of the protocol, a further explanation must be provided to assess if the responses are correct. We will compare the nodes provided by the expert to those obtained using the formula for degree centrality to fetch the three most important nodes [18]. We expect complete similarity between the expert's choice and the theoretical answer to consider the response correct.

Context

The initial suffix tree used by *SAGE* showcases all the data as it is but does not show any similarities between the elements and is generally too large to interpret as a whole [1]. All merges that happen afterwards cause a loss of context since the context is denoted by the state identifiers in the original *SAGE* paper [1]. This is why only probabilistically similar states are merged to reduce the loss of context and have an acceptable tradeoff between context and efficiency for the cybersecurity analyst.

The two selected AGs can be compared based on whether the one generated by allowing the merging of sinks makes any extra merge that leads to a loss of context in any of the following two manners:

1. The security analyst can correctly identify all paths in an AG but might believe that they have similar behaviours when that is not the case due to certain merges
2. The security analyst can incorrectly identify a wrong path because of the multigraph nature of the AG, which is highly accentuated by certain merges between sinks

Completeness

Completeness is represented based on the alert sequences because of the lack of a *ground truth*. The downside of using this approach is that systems like *SAGE* need to make inevitable trade-offs between efficiency and correctness that lead to the appearance of adverse effects such as false negatives [19]. This notion should be dealt with in the data-cleaning stage of any complete AG generator. However, the upside is that our changes in the process of merging sinks affect the results at a later stage, leading to the possibility of considering the alert sequences as ground truth.

Now that a solution without ground truth has been defined, we decided to apply the following two of completeness currently used in knowledge graphs to the AGs generated by *SAGE* because of their adaptability.

Population Completeness

This first concept refers to how well the distribution of the used data reflects reality. In the context of AGs, we made the parallelization to how many high-severity alerts *SAGE* represents in the AGs over their totality. The adjusted formula is:

$$PopulationComp = \frac{\textit{No of unique objectives in AGs}}{\textit{No of unique objectives}}. \quad (4)$$

Schema Completeness

This definition refers to the levels of representability of a real object's properties in the data, missing definitive attributes leading to a natural drop in completeness. In our case, not representing all attack paths due to interpretability concerns can lead to the loss of valuable information, and we chose to represent this with the following formula:

$$SchemaComp = \frac{\textit{No of paths present in AGs}}{\textit{No of total paths}}. \quad (5)$$

4 Experimental Setup

In order to set up this experiment, the first step was to obtain used in the original *SAGE* paper, CPTC-2017 and CPTC-2018 [1], [11]. Some highly important insights into the alerts present in the two datasets can be seen in Table 1. The first reason for this choice is to ensure that the obtained results are easily linkable with the initial evaluation results. In contrast, the second one relates to the possibility of encountering unexpected phenomena due to the new data selected and not due to *SAGE*'s algorithm.

CPTC-2017			CPTC-2018		
Team	Raw alerts	Filtered alerts	Team	Raw alerts	Filtered alerts
T2	2923	2904	**T1**	39710	26651
T3	3353	3293	**T2**	5012	4922
T4	7801	7232	**T5**	18447	11918
T5	1912	1890	**T7**	10001	8517
T6	8413	7485	**T8**	15560	9037
T7	4712	4220	**T9**	12841	10081
T8	7150	4944			
T9	2233	2199			
T10	5105	4949			

Table 1: Insights into data - number of raw and filtered alerts for each team contained by CPTC-2017 and CPTC-2018.

Afterwards, the SAGE tool has been run with the recommended parameters alert-filtering-window = 1.0 and alert-aggr-window = 150. Running the algorithm for the previously mentioned datasets fetched the baseline implementation's AGs and the merge sinks implementation's AGs after changing the following parameter to the *spdfaconfig.ini* file: *mergesinks = 1*. We then analysed the data with the help of the *stats_analyzer.ipynb* file that can be found on GitHub [20]. One of the main tools used in the analysis process was *Networkx*, which allowed the fast and reliable matching of *.dot* files and their further processing for head-to-head comparisons, such as community modularity and global density.

Concurrently, the alert sequences needed for the completeness analysis have been extracted and processed in the following manner:

1. For each attack path, the first edge has been mapped and stored as [*victim IP, end prev, mcat, protocol*].
2. If the attack path was represented in any episode of any sequence, it is filtered out.
3. The paths not ending in a high-severity node have been filtered out since they are outside the scope of *SAGE*.
4. The remaining alerts have been manually analysed to detect potential misses.

5 Results and Discussions

This section presents the results obtained during this research and an in-depth discussion of the underlying reasons for obtaining them. First, some consequences are explained and are followed by the comparison between AGs on the four metrics described in Subsection 3.4. Finally, a proposed merging criterion is presented and evaluated.

5.1 Discovered Consequences

Some general consequences of allowing the sinks to merge, which are not captured by the metrics, are given in Table 2. In terms of AGs generated by the two implementations, there are no differences regarding the total number, meaning that the same number of victims have been represented. Upon further investigation, the victims are the same, and all graphs generated can be matched. Furthermore, the number of starting states is the same (apart from a difference of 2 for the CPTC-2018, which is due a correct merge between 3 identical sink start states) and those correspond to each other in each pair of AGs paired. This leads to the critical observation that all attack paths represented in the baseline implementation are still present after allowing the sinks to merge with other sinks.

Concurrently, some objectives are missing after the new merges. This is due to the existence of multiple similar such states, which were also sinks and were therefore merged. All logic elements were left intact through these merges. Lastly, as hypothesised, all the sink states have disappeared, becoming normal states. This took effect in all 91 AGs where sinks were present, but, as another statistic highlights, only 41 AGs suffered changes to their node count or edge count.

	CPTC-2017		CPTC-2018	
	Base	Sinks	Base	Sinks
AGs	108	108	75	75
AGs affected	N/A	29	N/A	12
AGs with sinks	52	0	39	0
Sink States	135	0	130	0
Objectives	262	251	174	168
Start States	314	314	203	201

Table 2: Statistics on the AGs generated by the two implementations.

5.2 Size Analysis

Table 3 presents the overall amount of nodes slightly decreases for both datasets, but not in a substantial enough manner to offer any advantages to either implementation. This minimal loss is due to the extra merges between sinks, and for the same reason, the AGs containing the most sinks are losing the most states. After applying these many merges, those AGs will be further investigated in Subsection 5.4 to establish the effects on the AG's interpretability.

	CPTC-2017		CPTC-2018	
	Base	Sinks	Base	Sinks
Max nr of nodes	48	44	34	30
Avg nr of nodes	17.7	17.0	11.9	11.6
Overall Loss	N/A	4%	N/A	2%
Biggest Increase	N/A	0	N/A	0
Biggest Decrease	N/A	10	N/A	5

Table 3: Comparison on the AGs generated by the two implementations based on size.

5.3 Complexity Analysis

Table 4 shows that the overall change in the average value for simplicity for an AG is around 1%. This happens because of the MultiGraph nature of the generated AGs, which allows multiple edges between two nodes, leading to minimal differences in simplicity in most paired AGs.

Furthermore, there are multiple pairs of AGs which, according to the linear regression model, transition from simple to complex due to decreased nodes count and relatively constant edges. On the other side, no pairs of AGs transition from complex to simple. This is due AGs either maintaining the same number of nodes or having it decreased, whereas the edges are consistent since merging doesn't affect the sequences of episodes from which those are generated. Therefore, complexity-wise, the differences are not substantial on the holistic level. Still, the base implementation performs at least as well as the merge sinks in the generation of each AG.

	CPTC-2017		CPTC-2018	
	Base	Sinks	Base	Sinks
Complex	53	57	39	40
Simple	55	51	36	35
Simple->Complex	N/A	4	N/A	1
Complex->Simple	N/A	0	N/A	0
Overall Loss	N/A	1%	N/A	1%

Table 4: Complexity values for the two implementations.

5.4 Interpretability Analysis

Readability

The results for readability consist of the quantitative and qualitative head-to-head analysis completed over all the filtered AGs. The results were consistent between all 25 pairs of AGs, 14 filtered from CPTC-2017 and 11 from CPTC-2018. The pair of AGs for *victim-10.0.0.72-DATAEXFILTRATIONsmtp* is especially representative of the findings of this investigation and will be presented in-depth.

The density comparison and protocol completion results can be seen in Table 5. We can quickly notice that the global density is lower for the baseline implementation AG, whereas the local density is higher. According to the definition given in Subsection 3.4, this observation can lead to the base implementation AG being considered more readable than the merge sinks implementation. This can be further supported by the result of the protocol, which was completed successfully for both AGs, but with a substantial gap of 38 seconds in favour of the baseline implementation.

	Baseline	MergeSinks
Global Density	0.05	0.1
Local Density	0.57	0.39
Protocol Complete	Y	Y
Protocol Time	85s	123 s

Table 5: Direct comparison between two matched AGs.

This head-to-head comparison between the AGs in Figure 5 and Figure 6 plainly presents why merging the sink states with other sink states after the main merging process can be a good idea if further parameterised. By following the attack paths of the AG from the victim node, we can notice the first merge between all three instances of *ACCOUNT MANIPULATION snmp*. In this case, this merge undoubtedly leads to a decrease in readability, but if it had only happened between the respective sink states on the right side of the baseline AG, the result would have been the opposite. The same argument can be made for the following states: *BRUTE FORCE CREDENTIALS pop3* and *REMOTE SERVICE EXPLOIT snmp*.

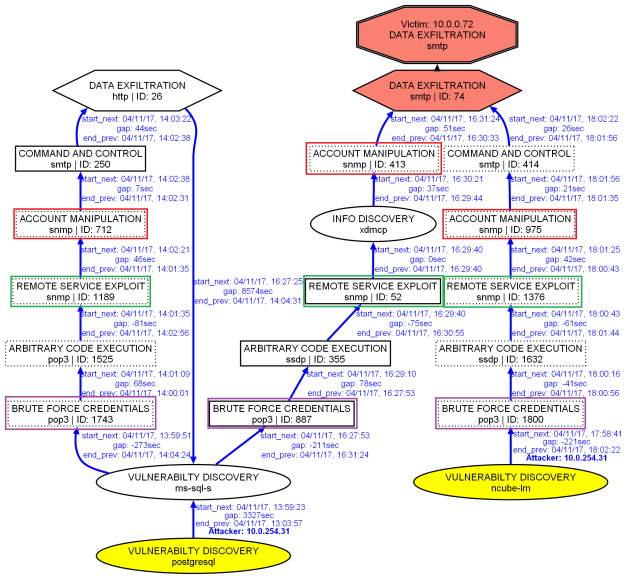

Figure 5: Readability - AG generated by baseline implementation for victim-10.0.0.72 DATA EXFILTRATION smtp where the 3 sets of sinks discussed are highlighted before being merged.

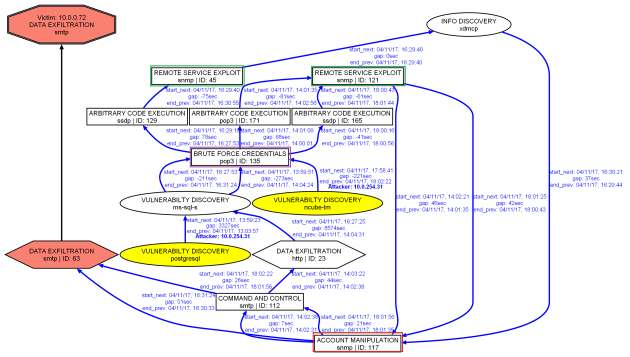

Figure 6: Readability - AG generated by merge sinks implementation for victim-10.0.0.72 DATA EXFILTRATION smtp where the sinks were merged and their new position is highlighted.

Context

In regards to the loss of context, we can see that between the AGs in Figure 7 and Figure 8, some of the context has been lost in both meanings defined in Subsection 3.4:

1. The cybersecurity analyst can analyze the attack paths and wrongly consider that an attack starting from *DATA*

DELIVERY unknown might lead to the objective in just one step, which is different from the baseline implementation since it does not allow such an interpretation.

2. The cybersecurity analyst can still analyze all attack paths correctly but might consider that two different attacks made by *Attacker: 10.0.254.31* which pass through *DATA DELIVERY unknown* showcase the same attacker behaviour, which due to the loss of ID happening in the merge is no longer true.

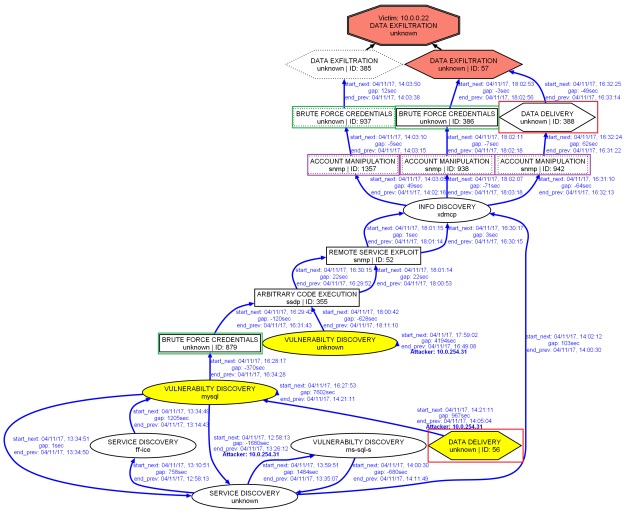

Figure 7: Context - AG generated by baseline implementation for victim-10.0.0.22-DATA EXFILTRATION unknown where the 3 sets of sinks to be merged are highlighted before being merged.

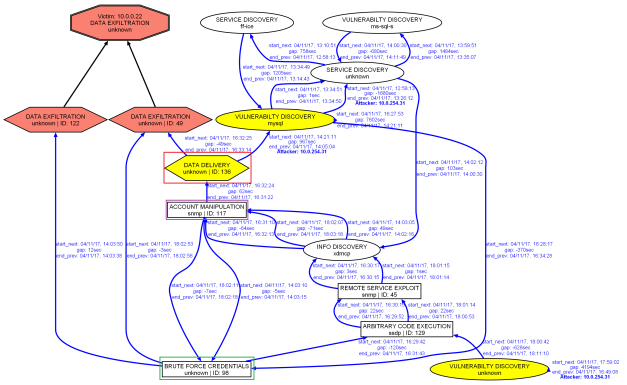

Figure 8: Context - AG generated by merge sinks implementation for victim-10.0.0.22-DATA EXFILTRATION unknown where the 3 states representing the 3 states from Figure 7 are highlighted.

By allowing the merges between sink states, the two *DATA DELIVERY unknown* states have been merged as showcased in Figure 9. The states highlighted have the appearance count set to two and four, making them highly infrequent states, sinks. By allowing this merge, FlexFringe considered the two states to be probabilistically similar, which, in this case, couldn't be further from the truth. This led to the loss of context described above, and in order to avoid such merges, some constraints need to be added. Subsection 5.6 presents a possible solution which mitigates this issue.

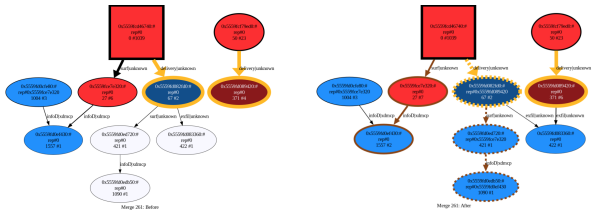

Figure 9: FlexFringe representation of the AG before and after the merge between two DATA DELIVERY unknown nodes, generated with *Jegor Zelenjak's* tool [21].

5.5 Completeness Analysis

As it was briefly presented in the complexity analysis, the number of edges between the two versions of the *SAGE* implementation does not change. We can translate this into the fact that merging the sink states with other sink states doesn't lose any extra attack paths compared to the baseline implementation. Following the manual check on the episodes, we can safely say that both implementations have the same level of completeness, as seen in Table 6. These values showcase a high number of false negatives, and the main reason for those is the condition that *SAGE* implements regarding the discarding of episodes with less than three alerts.

	CPTC-2017	CPTC-2018
Population Comp	82.44%	76.53%
Schema Comp	87.33%	88.01%

Table 6: Completeness values for the 2 datasets.

5.6 Proposed Merging Criterion

During the data analysis of this research, we have experimented with some modifications to the AGs' dot files to try and generate the *ideal* AG. During this process, we uncovered the idea of only allowing sink states to merge in post-processing if they are at an equal distance (number of nodes) from the objective. By adding this constraint, we aim to mitigate the loss in context and decrease the loss in readability whilst maintaining the same properties discovered from the size, complexity and completeness analysis. The new AGs have been generated by taking all the node and edge information from the dot files generated by the baseline, merging all the sink states which satisfy the earlier-mentioned constraint and finally reconstructing the AG.

Applying the methodology described in Section 3 to compare the new AGs with the original ones generated by the baseline implementation, we obtained highly insightful results, which can be summarised as follows:

- **Size**: A negligibly smaller number of nodes and the same number of edges. This is better than the merge sinks implementation, which loses a small number of edges due to merging sub-objectives, an undesired property due to loss of attacker behaviour.

- **Complexity**: 28 AGs have suffered changes in both datasets, which would affect this metric, and the values obtained for those AGs are all very similar to the baseline. Only 3 AGs converted from simple to complex in comparison to 5 in the merge sinks implementation.
- **Completeness**: Completeness values are unchanged since all edges from the baseline are drawn with this proposed implementation, and those represent the same attack episodes as the baseline.
- **Interpretability**: After manually analysing all 28 AGs that presented merges, we have discovered that the values obtained for density and the protocol are better than the previous merge sinks algorithm but still narrowly worse overall than the baseline. Most importantly, no loss of context has been detected since merges between nodes showcasing different attacker behaviours were not allowed because those happen at different levels.

One example where this algorithm creates an improved AG in terms of interpretability is for *Victim-10.0.0.22 DATA EXFILTRATION unknown* and can be seen in Figure 10. The global and local density values can be seen in Table 7 and showcase a clear improvement over the merge sinks implementation. Compared to the baseline, the global density is slightly higher due to the size reduction. Nonetheless, an essential aspect is that the local density is higher than in the baseline implementation, which is a highly desirable property for any graph.

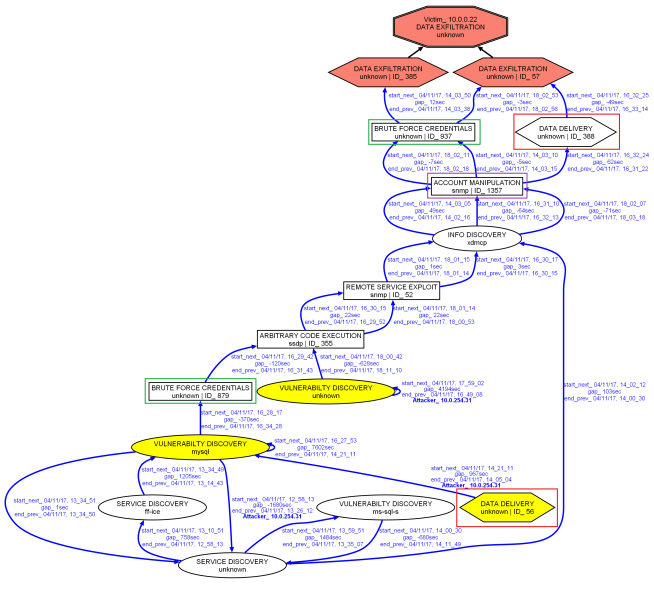

Figure 10: Generated AG for victim-10.0.0.22-DATA EXFILTRATION unknown with the constraint of allowing the merging of sinks only at equal distance from the victim node, the same sets of sinks are highlighted as in Figure 7 and Figure 8.

Compared to the AG in Figure 7, it partially allowed the merging of *BRUTE FORCE CREDENTIALS unknown* and *ACCOUNT MANIPULATION snmp* in the top side of the AG, without merging nodes such as *BRUTE FORCE CREDENTIALS unknown — ID 879*, which is in the lower side of the AG. Also, the *DATA DELIVERY unknown* nodes have not been merged for the same reason. This leads to no loss of context according to the definition in Subsection 3.4 and, therefore, no loss in interpretability.

	Baseline	MergeSinks	Proposed
Global Density	0.081	0.153	0.116
Local Density	0.494	0.421	0.505

Table 7: Local and Global Density for Victim 10.0.0.22 DATA EXFILTRATION unknown (CPTC-2017) for the baseline, merge sinks and proposed implementations.

6 Conclusions and Future Work

In conclusion, this research showcases the effects of allowing the sink states to merge with other sink states on individual attack graphs and the overall generated suits. The key discoveries were:

- All sink states transformed into normal states.
- A small overall deficit in the average size of attack graphs has been observed.
- The baseline implementation was consistently less or equally complex to the merge sinks implementation.
- Interpretability has decreased substantially in all AGs affected by the change due to the loss of context, increased global density, decreased local density and a negative difference in the ability to follow paths correctly and consistently.
- Completeness remained consistent at a level of approximatively 80% because the extra merges happening are not affecting the episodes processing, and the number of edges remains consistent.
- The proposed merging criteria performs clearly better according to the metrics than the merge sinks implementation.
- The proposed merging criteria performs at a similar level to the baseline implementation, having very similar AGs, which showcase no extra loss in context and similar readability values.

These showcased effects help validate the current modelling assumptions of *SAGE* since they prove that just allowing the sink states to merge with other sink states will lead to worse individual and holistic results for a security analyst. We also noticed that merging only the sink states, which are at an equal distance to the victim node, will lead to a similar performance to the current *SAGE* implementation and improves the interpretability for certain AGs. Such ideas should be considered in the future work of validating and improving *SAGE*.

Lastly, the research successfully helped validate the current implementation of *SAGE* and even proposed a merging criterion which possibly improves the current implementation in certain regards. Additionally, it helped in improving the correctness of the results generated by fixing five minor bugs, which had multiple negative effects on the results.

7 Responsible Research

The results obtained in this research are entirely reproducible by following the steps presented in section 4. By running *SAGE*, one should obtain the same AGs regardless of the machine it runs the algorithm on due to its deterministic nature. The only thing that needs to be kept in mind is that both *SAGE* and *FlexFringe* need to be run on the latest versions available in June 2023. Afterwards, the results mentioned in section 5 are obtained by doing the analysis described in section 3 with the help of multiple ipynb files, which can be found on GitHub [20].

Additionally, regarding the ethics involved in this research, we made decisions intending to uncover truths about SAGE. Each evaluation was done critically and supported with proper arguments. All results obtained were evaluated to eliminate any bias that might have occurred. However, some aspects of the qualitative analysis, such as the interpretability results, are of such a nature that the cognitive bias could only be limited [22].

Another element of ethics encountered was the discovery and fixing of five bugs in *SAGE*, leading to incorrect results. Such results can have severe consequences because security analysts can interpret them wrongly and not fix major system breaches. Such problems might still be present in *SAGE*, but detecting and solving these correctness issues is outside the scope of this research. For all problems of this nature that we have seen throughout this research, we let the authors of the code know of their existence and proposed fixes.

8 Acknowledgements

The improvements in the codebase of *SAGE* have been detected and created as a team effort, and I would like to acknowledge the role of *Ioan-Cristian Oprea, Jegor Zelenjak, Senne Van den Broeck* and *Vlad Constantinescu*. The methodology creation was also part of a collaboration with *Ioan-Cristian Oprea* and *Jegor Zelenjak* to ensure consistent metrics between our experiments presenting different modelling assumptions. Finally, we want to acknowledge the responsible professor, *Sicco Verwer*, and supervisor, *Azqa Ndeem*, for the help and guidance throughout the research project.

References

[1] A. Nadeem, S.E. Verwer, Stephen Moskal, and Shanchieh Jay Yang. Alert-driven attack graph generation using s-pdfa. *IEEE Transactions on Dependable and Secure Computing*, 19(2):731–746, 2022.

[2] Wajih Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. 01 2019.

[3] Steven Noel, Matthew Elder, Sushil Jajodia, Pramod Kalapa, Scott Hare, and Kenneth Prole. Advances in topological vulnerability analysis. pages 124–129, 03 2009.

[4] A. Nadeem, S.E. Verwer, Stephen Moskal, and Shanchieh Jay Yang. Enabling visual analytics via alert-driven attack graphs. In *CCS 2021 - Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, Proceedings of the ACM Conference on Computer and Communications Security, pages 2420–2422, United States, 2021. Association for Computing Machinery (ACM).

[5] A. Nadeem, S.E. Verwer, and Shanchieh Jay Yang. Suffix-based finite automata for learning explainable attacker strategies. 2022.

[6] Sicco Verwer and Christian Hammerschmidt. Flexfringe: Modeling software behavior by learning probabilistic automata, 2022.

[7] Oleg Sheyner, Joshua W. Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284, 2002.

[8] Cynthia A. Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *New Security Paradigms Workshop*, 1998.

[9] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130, 2006.

[10] Azqa Nadeem, Sicco Verwer, Stephen Moskal, and Shanchieh Yang. Sage: Intrusion alert-driven attack graph extractor. 08 2021.

[11] RIT. CPTC dataset. Available: https://mirror.rit.edu/cptc/. 2018.

[12] Sean Carlisto de Alvarenga, Sylvio Barbon, Rodrigo Sanches Miani, Michel Cukier, and Bruno Bogaz Zarpelão. Process mining and hierarchical clustering to help intrusion alert visualization. *Computers Security*, 73:474–491, 2018.

[13] David Meunier, Renaud Lambiotte, and Edward Bullmore. Modular and hierarchically modular organization of brain networks. *Frontiers in Neuroscience*, 4, 2010.

[14] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. *InfoVis'04 - 10th IEEE Symposium on Information Visualization*, pages 17–24, 2004.

[15] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

[16] M. E. J. Newman. Networks: An introduction. page 224, 2011.

[17] Clauset Aaron, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. 2004.

[18] Jennifer Golbeck. Chapter 3 - network structure and measures. In Jennifer Golbeck, editor, *Analyzing the Social Web*, pages 25–44. Morgan Kaufmann, Boston, 2013.

[19] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3:186–205, 01 2000.

[20] Dumitriu Alexandru. research-project. https://github.com/alexdumitriu2001/research-project, 2023.

[21] Zelenjak Jegor. research-project [get-merges.sh]. https://github.com/jzelenjak/research-project, 2023.

[22] Tomáš Kliegr, Štěpán Bahník, and Johannes Fürnkranz. A review of possible effects of cognitive biases on interpretation of rule-based machine learning models. *Artificial Intelligence*, 295:103458, 2021.