Evaluation of a Multi-agent Anomaly-based Advanced Persistent Threat Detection Framework

Georgi Nikolov
Royal Military Academy
Brussels, Belgium
Email: g.nikolov@cylab.be

Thibault Debatty
Royal Military Academy
Brussels, Belgium
Email: t.debatty@cylab.be

Wim Mees
Royal Military Academy
Brussels, Belgium
Email : w.mees@cylab.be

***Abstract*—Cyber attacks have become a major factor in the world today and their effect can be devastating. Protecting corporate and government networks has become an increasingly difficult challenge, when new persistent malware infections can remain undetected for long periods of time. In this paper, we introduce the Multi-agent ranking framework (MARK), a novel approach to Advanced Persistent Threat detection through the use of behavioral-analysis and pattern recognition. Such behavior-based mechanisms for discovering and eliminating new sophisticated threats are lacking in current detection systems, but research in this domain is gaining more importance and traction. Our goal is to take a on-hands approach in the detection by actively hunting for the threats, instead of passively waiting for events and alerts to signal abnormal behavior. We devise a framework that can be easily deployed as a stand-alone multi-agent system or to compliment many Security Information and Event Management systems. The MARK framework incorporates known and new beyond state-of-the-art detection techniques, in addition to facilitating incorporation of new data sources and detection agent modules through plug-ins. Throughout our testing and evaluation, impressive true detection rates and acceptable false positive rates were obtained, which proves the usefulness of the framework.**

Keywords–anomaly-based analysis; command & control channel; advanced persistent threat; aggregation.

I. INTRODUCTION

Corporate, government and military networks have often been prime targets for malicious actors and the current security solutions have proven not to be sufficient any longer. In recent years, these types of attacks have become more frequent and more sophisticated; using zero-day vulnerabilities and social engineering, the attackers can set a foothold in a network and work unnoticed for long periods of time. A recent example of a cyber-attack on a major scale is the one orchestrated on computer systems from Ukraine to the United States in 2017 [1]. The attack hijacked a tax accountant package widely used in Ukraine and distributed the malware via its update mechanism, targeting the supply-chain, a common Advanced Persistent Threat (APT) attack technique.

Currently major networks are protected via Security Information and Event Management (SIEM) systems, collecting log events and alerts and then correlating them; Splunk [2] or IBM QRadar [3] are some prominent examples of such systems. Useful as they may be, these detection solutions often focus on the initial attack and less on the possible persistency of a threat as they lack understanding of (1) the complex behavior of Advanced Persistent Threats (APT) and (2) the precision needed to correlate prolonged malicious activity that may take place over multiple hosts over prolonged periods of time. Our paper introduces the MARK framework, with the goal to detect APTs once they have established a foothold in a network. Contrary to the majority of currently established Intrusion Detection Systems (IDS) and their use of signature-based detection, focused on passive detection through the correlation of events and alerts, our system takes a more active approach by automating the data-driven threat hunting process. This type of detection approach has become more and more relevant, a lot of new research has been invested in new threat hunting methods [4]. The MARK framework accomplishes this by analysing data from multiple sources and searching for abnormal or suspicious behavior, through pattern recognition.

The rest of the paper is arranged as follows: Section II describes the design and methodology used for the implementation of the MARK framework. The detection agents that are used for the analysis are presented together with the aggregation and scoring system set in place. Section III presents the different steps in the evaluation of our framework and the results produced. Finally we conclude in Section IV and offer possible future avenues to advance our framework in Section V.

II. THE MULTI-AGENT RANKING FRAMEWORK

In this section, an overview of the Multi-Agent Ranking Framework is provided. The general goal and design of the framework is discussed, together with an explanation of the methodology used for the different agents and the score aggregation mechanism.

A. Goal

The MARK framework uses multiple agents to detect possible APTs, using behavioral analysis instead of the common knowledge-based approach, focused on signature analysis. Research into the subject of creating a modular behavior-based analysis has been conducted, such as the one proposed by [5] and [6]. In our project we combine domain knowledge with information collected from multiple agents about the behavior of possible threats and apply fuzzy logic to determine the possibility of malicious intent. We designed the framework as a multi-agent system that can be deployed on a centralized server, collecting raw data from multiple sources and correlating the findings.

 ISBN: 978-1-61208-796-2

Our framework is developed with the goal to be deployed as a stand-alone detection system, or complement currently available off-the-shelf SIEM systems, working in parallel with other detection tools, providing exponential benefits over an extended period of time. The implementation and integration of the MARK framework are shown in Figure 1.

The solid red arrow represents how the malicious actor can set-up the instructions which should be relayed to the infected machine inside the compromised network via a Command & Control (CnC) channel, represented by the dashed red arrow. This channel can be used by a malicious actor to send commands to the infected machine and receive responses, for example network reconnaissance information or exfiltrated data. In the majority of protected networks, the outbound traffic is restricted to only layer 7 channels, such as HTTP(S), SMTP or DNS. This means that any CnC channel must pass through the proxy choke point, denoted in blue. This is an important reason why our system focuses primarily on analysis of HTTP and SMTP proxy logs, as those are the most likely means to detect the CnC channel communication with the malicious server. The MARK framework continuously collects data from the proxy, alongside netflow data and end-point data, shown in green. All this information is fed to the MARK framework, analyzed, aggregated and then, using visualisation techniques, displayed to the domain expert for analysis. For a more in-depth look at how the system is developed, the code for the MARK framework is available at [7]. The agents, described

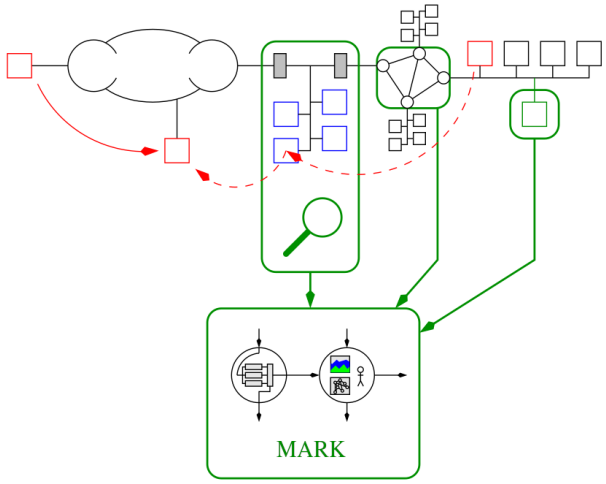


Figure 1. MARK framework diagram

in Section II-B, work independently from each other and their results are aggregated. Afterwards a ranking of the possible threats is created, as shown in Section II-C. Our goal is to observe the behavior of the different clients in the monitored network and based on a list of characteristics, the framework will decide the degree of suspiciousness a given connection has. The MARK framework is not intended to classify by itself what is a threat and what is benign, but to combine the detection system with the knowledge of a domain expert in analysing and filtering the results to come to these conclusions. This is enabled through:

- new agents can be configured for different precision through parameterization and new ones can be added as plug-ins
- using ”detection through visualization” implementing Visual Analytics techniques [8]
- a whitelisting option is present so the analyst can eliminate known harmless domains that may have been flagged as false positives

B. Agents

The MARK framework is designed to be data agnostic; different data sources can be added with minimal difficulty, and the analysis capabilities can be extended, via a plug-in system, with new detection techniques. In our current implementation, we have multiple detection agents for analysis of HTTP logs and SMTP data. These agents are not meant to run on the client machines, but act as independent modules, that focus on specific behavioral characteristics of an APT. Through the analysis of these different characteristics, the agent's findings are aggregated so significant patterns can be discovered. Each agent can also be adapted through the use of parameters, specific for each detection technique. The characteristics that we focus on are the following:

- number of unique domains per IP and visa versa
- frequency and periodicity of the connection
- geo-positioning of connection's server
- upload size and POST count
- ”lonely” single connections and time anomalies that signify abnormal behavior
- unreachable server connections

C. Scoring and Aggregation

In the area of intrusion detection, aggregation can be applied for a number of reasons. A first motivation can be to obtain a condensed view of the outputs from a number of IDS sensors located at different positions in the network [9]. Another motivation can be to reduce the false alarm rate by modeling attacks and correlating observed events with known attack scenarios or intrusion objectives [10]. In our APT detection system however, potential evidence about a single event needs to be accumulated, evidence that is produced by a number of independent agents, each verifying some a priori defined hypothesis. Such a malware behavior hypothesis expresses a specific part of the domain knowledge of a human network security expert, who will typically use vague natural language terms when expressing his knowledge. For that reason the knowledge is represented in the form of fuzzy sets and fuzzy expert system rules. The fuzzy set is defined as a pair *(U,m)* with membership function:

$$m : U \rightarrow [0, 1] \tag{1}$$

For each evidence score produced, the suspiciousness of the evidence x is defined by:

$$0 < m(x) < 1 \tag{2}$$

In order to combine the fuzzy evidence, produced by the different agents, we use the Ordered Weighted Averaging (OWA) operator, introduced by Yager [11], that has been used successfully for a number of evidence aggregation problems

 ISBN: 978-1-61208-796-2

in the area of network security [12]. The OWA operator is defined by the function:

$$F(a_1, ..., a_n) = \sum_{j=1}^{n} w_j b_j \quad (3)$$

for a collection of weights $W = [w_1, ..., w_n]$ and where b_j is the largest j^{th} of the scores a_n.

At the end of the aggregation, a ranked list of possible threats is compiled, where false positives are ranked lower and the true positives are elevated.

III. Evaluation through simulated scenarios

To determine the detection capabilities of the MARK framework, three scenarios were developed and simulated log files were generated to be analyzed by the system. This section discusses how we simulate scenarios using real world data and documented APTs, that are modeled and injected into the real world log files. The flow of the aggregation and evaluation is shown in Figure 2.

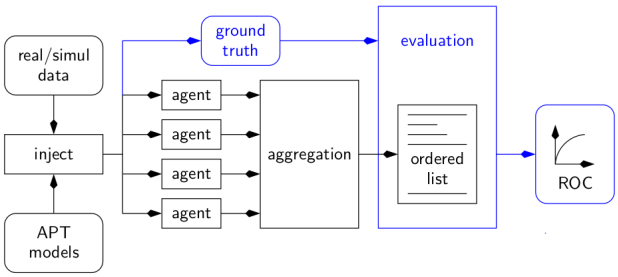


Figure 2. MARK evaluation diagram

A. Scenario configuration file

We test and validate the MARK framework through the use of simulated scenarios and each scenario is defined through a configuration file with the following parameters:

- predetermined duration of the scenario
- predetermined number of clients and respective IPs
- real world data used to generate the logs to be analyzed
- set number of attacks, which have:
  - predetermined victim (internal IP that is considered infected)
  - type of attack
  - characteristics of the attack

When each scenario is generated, a ”ground truth” file is also created that holds information about the attacks that have been injected. This ”ground truth” file is used during the evaluation to determine if the attacks have been successfully detected and placed high in the ranked list.

1) Dataset used: Recent real world log files, provided by various agencies, are used to simulate our scenarios and keep them as close as possible to real world situations. Each log file, originally in JSON proxy format, consists of all the HTTP connections from a single day. The proxy logs are transformed from JSON format to SQUID format, but other formats can easily be supported by the framework. The specifications of the real world datasets used for evaluation, are shown in ”Table I”. The scenario configuration file is read to determine the start

TABLE I. REAL WORLD DATASET SPECIFICATIONS

Original Format	real world proxy logs
Transformed Format	Squid logs
Number of Log Files	106
Size	336.3 GB

date, end date and the IP addresses of the network. A mapping is done between the real world IP addresses from the proxy logs and the IP addresses to be used in the scenario.

2) Whitelisting: While running the scenarios, whitelisting is used to remove known no-threat servers from the results. This is done for two reasons:

1) Known servers such as facebook/google/etc. have services that constantly send requests to their servers in a periodic manner, which can cause false detection (ex. facebook groups, google analytics, windows live)
2) Known adwares have similar behaviors to APTs, where periodic connection is established to the ad-server, which can also be regarded as a false positive detection.

For the preliminary analysis we use two whitelists:

- **Whitelist.txt** - holds a regex of known non-threat domains which have produced an evidence. After analysis they have been considered harmless
- **1Hosts_Pro_Domains.txt** - a compilation of 217.530 known adware domains [13]

B. Simulated Scenarios

A number of attacks are simulated and injected in the real world log files to act as our simulated network activity. These logs serve to simulate the background traffic that occurs daily in any given network and is used by malicious actors to obfuscate their actions. The injected attacks range from basic periodic attacks to high complexity real world APTs such as the Trojan Nap APT [14], the Regin APT [15] and Careto APT [16]. These APTs are used as baseline and modified to generate synthetic APTs with variable behavior through parameterization, where a variable density of attacks is defined for each scenario. In such a manner the framework’s detection capabilities can be tested with varying degrees of difficulty.

For the sake of brevity we will showcase the characteristics of one of the scenarios used.

1) Scenario 1: The first scenario consists of multiple hosts inside a network that have been infected and a static, periodic CnC channel has been established between them and an outside server.

2) Scenario 2: The second scenario uses state models that simulate real world APTs, with high density through the use of outbound connections with high periodicity.

 ISBN: 978-1-61208-796-2

3) Scenario 3: In the third generated scenario, the duration is doubled and a larger number of clients are used. State models of real world APTs are used, but lower APT density is defined through lower and varying periodicity. The characteristics of Scenario 3 are shown in "Table II" and the generated scenario variations are shown in "Table III".

TABLE II. SCENARIO 3 SPECIFICATIONS

File Format	text file
Logline Format	Squid logs
Subnet	10.0.0.1 - 10.0.0.249
Number of Attacks	10
Average Running Time	90 hours

TABLE III. SCENARIO 3 GENERATED LOGFILES

Name	Lines	Unique Client-Server pairs
scenario3.1	2302306	97230
scenario3.2	2208271	91528
scenario3.3	2619919	93360
scenario3.4	2867952	98962
scenario3.5	2590393	96000
scenario3.6	2994435	106286
scenario3.7	2453249	89857
scenario3.8	2697416	102319

C. Iterative Evaluation

To evaluate our findings, each scenario is run multiple times, where each time we run a different variant of the scenario with different modeled attacks and varying parameters. The evaluation happens in multiple:

1) review the results generated by the individual agents
2) evaluation of the precision of the OWA operator weights
3) generate the Receiver Operand Characteristics (ROC) and Area Under the Curve (AUC) for each scenario variation using the produced ranked list
4) compute the amount of data a domain specialist has to manually analyze to discover all true threats

1) Agent Evaluation: Each agent is responsible for a specific characteristic analysis and presents different paradigms. We can evaluate the results generated by comparing them to the already known behavior of the generated APTs and further analyze other highly ranked connections that might be of importance. To showcase how the different agents are evaluated, the output of the Frequency Agent is presented.

Frequency Agent: As with other agents, a set of specific parameters exists that can be adapted to fine-tune the detection rate of the agent. The Frequency agent is defined through the following parameters:

- time window of analysis, by default set to 24 hours
- sampling interval, set to 2 seconds
- minimum coverage by the peaks detected, modeled using fuzzy logic parameters set to $[0.1, 0.5, 0, 1]$
- peak threshold, set to 1.3
- suspiciousness score, modeled using fuzzy logic parameters set to $[1.3, 2, 0, 1]$

First the periodicity is determined through the computation of a Frequency Spectrum using a Fast Fourier Transformation (FTT) and further removing unnecessary noise data as shown in Figure 3. Through this type of visualization, it is clear that a periodic frequency is present. Any peaks under the threshold (blue line) are considered noise and disregarded. As with all detection agents, the threshold selected is adapted manually for the highest performance. A Time Sequence is also generated as shown in Figure 4, to better visualise if a periodicity is present. Comparing the results to the "ground truth" can show if adjustment of the parameters is needed for better precision.

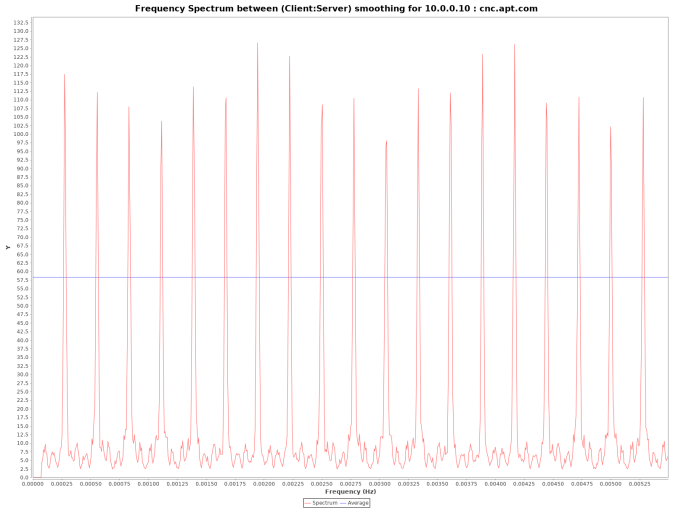


Figure 3. Frequency Spectrum between (Client:Server) smoothing for 10.0.0.10 : cnc.apt.com

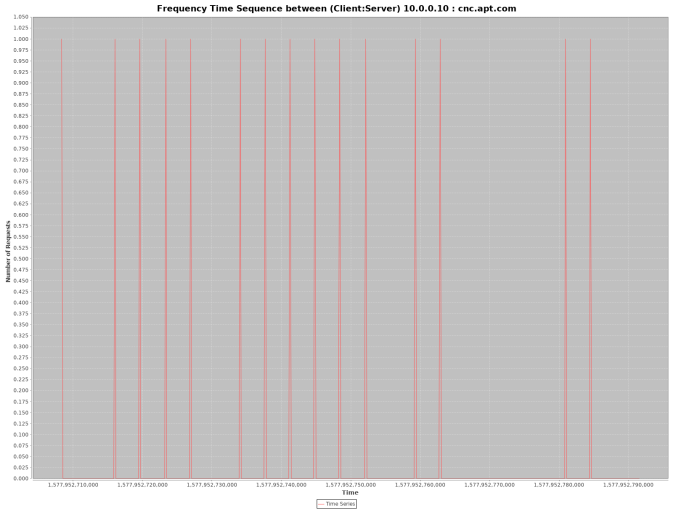


Figure 4. Frequency Time Sequence between (Client:Server) 10.0.0.10 : cnc.apt.com

2) Scatterplot Evaluation: By plotting the highest and the second highest scores for each tuple client-server, using a scatterplot, we can evaluate if the weights used in our OWA aggregation are precise or if they need to be adjusted. A scatterplot generated for one simulated scenarios is shown in Figure

5. The weights used for the evaluation are $[0.2, 0.4, 0.3, 0.1]$, where the highest produced score is assigned a lower score to prevent agents that are more active and produce a large amount of evidence to generate high quantities of false positives. The

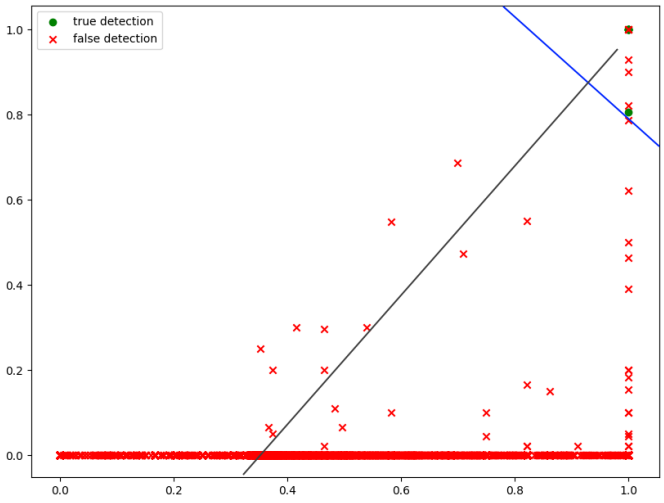


Figure 5. Example Scatterplot Scenario 2.6

true detection scores are all situated in the top right corner of the graph, signifying that both the highest and the second highest scores given by the detection agents were high. The majority of false detection are placed on the bottom, as a score for them has been generated only by one of the agents, where the other one didn't qualify them as suspicious.

The goal is to find a separation line and its slope in such a way that all true detection is above the line and as many possible false detection are situated under the line. To do that, we first need to calculate the blue line and rotate it 90° to get the function describing the black line and its slope. A number of false positives will always be included above the separation line, but that number can be minimized.

3) Boxplot and ROC: To evaluate our ranked list and the precision of our detection, we compute ROC and calculate the AUC. The ROC is a graphical representation that illustrates the diagnostic capabilities of a detection system. It is created by plotting the true positive rate (true detection) against the false positive rate (false alarm). The goal is to have a ROC that climbs as fast as possible, which would signify that the true positives are encountered earlier in the list, ideally at the top of the ranked list.

The Area Under the Curve gives a score that allows us to easily evaluate and compare the performance of the framework across multiple scenarios. The closer to 1 the AUC is, the better is the performance. All the calculated ROC and AUC for Scenario 3, described in Section III-B3, are shown in Figure 6. Sub-figure (a) shows the results when no whitelisting was used. For comparison, in sub-figure (b) the results when whitelisting has been applied to the generated ranked lists are shown. It is important to note that the results become better as we embody the role of the domain expert and analyze the entries of the ranked list.

In the Boxplots presented in Figure 7, the distribution of the AUC values calculated for our scenarios is presented. The results show a high detection rate, though the results are not tightly grouped. By applying whitelisting the resulting median is higher and lower variation in results is present. This signifies that our results become more homogeneous and even though in each scenario and its variations the attacks were parameterized differently, generally all attacks were ranked high and close together.

4) Manual analysis of the generated data: The final step is to consider the amount of entries a domain expert has to go through manually, to be able to detect all possible infections. As we see in the results presented in Table IV, with some exceptions, 100% of all attacks have been ranked in the top 20. This means that an analyst would need to analyze 20 entries, to be sure to go through all possible attacks in these particular scenarios. T his is highly important as our framework is

TABLE IV. SCENARIO 3 NUMBER OF ENTRIES TO ANALYZE FOR 100% TRUE DETECTION

# List Entries	PD score Scenario 3 variants							
	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8
10	0.4	0.9	0.9	0.9	0.6	0.4	0.9	0.9
20	0.9	1.0	1.0	1.0	1.0	0.9	1.0	1.0
30	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
40	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
50	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

designed to work hand in hand with a real world domain expert through the use of Visual Analytics and domain knowledge. It is imperative to minimize the time spent by the analyst on reviewing insignificant data and instead focus on information that is of greater importance.

IV. Conclusion

Throughout the research and development of the MARK framework, it became evident that the need for a robust system for APT detection is apparent. The system we developed offers a novel solution for the detection of attacks that typically offer greater challenge to detect, because of their unique nature.

During the evaluation we discovered the importance of setting the OWA operator weights correctly greatly benefices the correct ranking of the threats. Furthermore, by judiciously choosing the assigned weights, the amount of false positives can be lowered drastically.

The efficiency of the framework is demonstrated via the ROC and AUC computed for the different generated scenarios. It can be observed that the AUC is close to 1, signifying that all malicious connections have been discovered and placed at the top of the ranking list. This also leads to better visibility for the domain expert and significantly less time spent on analysis of large amounts of data.

V. Future Work

There are multiple avenues that can be examined for the future development of the MARK framework. First we will research and implement new detection techniques such as the use of graph theory for APT detection, as showcased in [17].

The aggregation of our evidences can be fine-tuned by incorporating the Weighted Ordered Weighted Averaging (WOWA) operator [18] and combine it with Machine Learning algorithms to augment the multi-criteria decision system [19].

 ISBN: 978-1-61208-796-2

Using WOWA we are not limited to assigning weights to the different scores produced by the agents, but also to the agents themselves.

Different network infrastructures have different specifications. As a future work we intend to implement Machine Learning algorithms and semi-supervised learning techniques [20], that will help configure the parameters used by the different detection agents depending on the environment wherein the MARK framework is deployed.

REFERENCES

[1] "Cyberattack hits ukraine then spreads internationally," https://www.nytimes.com/2017/06/27/technology/ransomware-hackers.html, retrieved: January, 2020.

[2] SPLUNK, "Siem, it operations, security, devops," https://www.splunk.com/, retrieved: January, 2020.

[3] IBM, "Ibm qradar siem," https://www.ibm.com/products/qradar-siem, retrieved: January, 2020.

[4] E. C. Thompson, "Threat hunting," in Designing a HIPAA-Compliant Security Operations Center. Springer, 2020, pp. 205–212.

[5] W. Mees, "Multi-agent anomaly-based apt detection," in Proceedings of Information Systems Technology Panel Symposium, 2012, pp. 1–10.

[6] P. Panero, L. Vlsan, V. Brillault, and I. C. Schuszter, "Building a large scale intrusion detection system using big data technologies," PoS, 2018, p. 014.

[7] RUCD. Mark framework. [Online]. Available: https://gitlab.cylab.be/cylab/mark

[8] R. F. Erbacher, "Intrusion behavior detection through visualization," in SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483), vol. 3. IEEE, 2003, pp. 2507–2513.

[9] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in International Workshop on Recent Advances in Intrusion Detection. Springer, 2001, pp. 85–103.

[10] F. Cuppens, F. Autrel, A. Miege, and S. Benferhat, "Correlation in an intrusion detection process," in Internet Security Communication Workshop, 2002, pp. 153–172.

[11] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," IEEE Transactions on systems, Man, and Cybernetics, vol. 18, no. 1, 1988, pp. 183–190.

[12] O. Thonnard, W. Mees, and M. Dacier, "Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making," in Proceedings of the ACM SIGKDD workshop on CyberSecurity and intelligence informatics, 2009, pp. 11–21.

[13] C. M. Barrett. Filterlists is the independent, comprehensive directory of filter and host lists for advertisements, trackers, malware, and annoyances. [Online]. Available: https://filterlists.com/

[14] M. Parkour. Trojan Nap aka kelihos/hlux - feb. 2013 status update. [Online]. Available: http://www.deependresearch.org/2013/02/trojan-nap-aka-kelihoshlux-feb-2013.html (2013)

[15] C. . I. S. Agency. Regin malware. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/TA14-329A (2014)

[16] K. Lab. Unveiling "Careto"-The Masked APT. [Online]. Available: https://d2538mqrb7brka.cloudfront.net/wp-content/uploads/sites/43/2018/03/20133638/unveilingthemask_v1.0.pdf (2014)

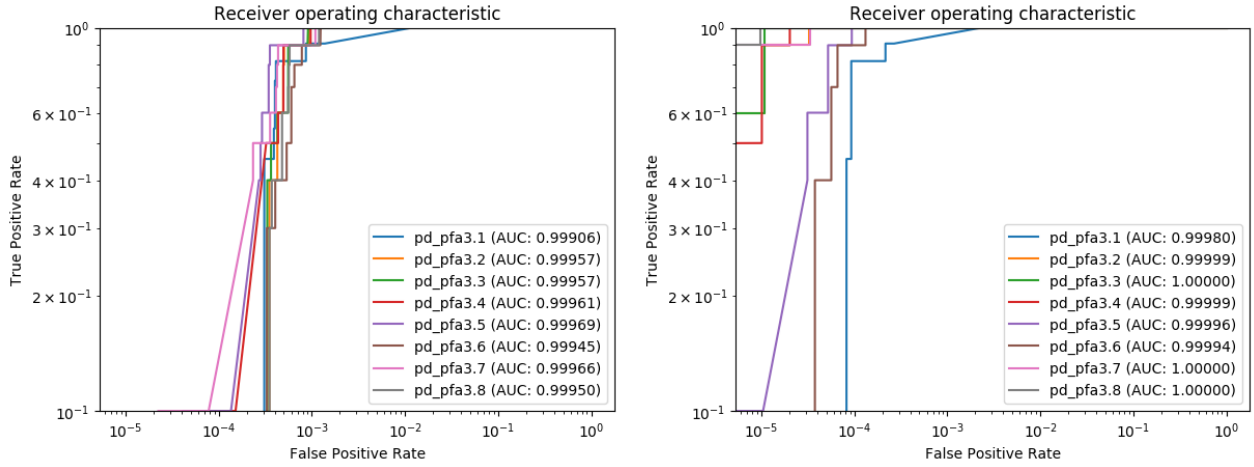
[17] T. Debatty, W. Mees, and T. Gilon, "Graph-based apt detection," in 2018 International Conference on Military Communications and Information Systems (ICMCIS). IEEE, 2018, pp. 1–8.

[18] V. Torra, "The wowa operator: a review," in Recent developments in the ordered weighted averaging operators: theory and practice. Springer, 2011, pp. 17–28.

[19] A. Croix, T. Debatty, and W. Mees, "Training a multi-criteria decision system and application to the detection of php webshells," in 2019 International Conference on Military Communications and Information Systems (ICMCIS). IEEE, 2019, pp. 1–8.

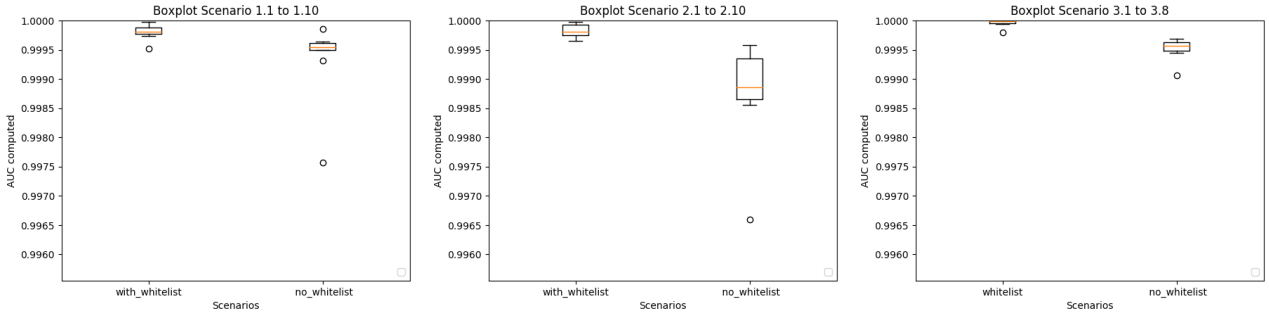
[20] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," Information Sciences, vol. 378, 2017, pp. 484–497.

 ISBN: 978-1-61208-796-2



(a) (b)

Figure 6. (a) Scenario 3 no whitelisting (b) Scenario 3 with whitelisting



(a) (b) (c)

Figure 7. Boxplots for the computed AUC, without and with the use of whitelisting, for Scenario 1, Scenario 2 and Scenario 3