Heated Alert Triage (HeAT): Network-Agnostic Extraction of Cyber Attack Campaigns

Stephen Moskal*[1] and Shanchieh Jay Yang[1]

[1]Rochester Institute of Technology

Abstract

With growing sophistication and volume of cyber attacks combined with complex network structures, it is becoming extremely difficult for security analysts to corroborate evidences to identify campaigns and threats on their network. So much so that organizations employ teams of security professionals just to keep up with vast amount of data presented to the analysts each day. This work develops HeAT (Heated Alert Triage): given a critical indicator of compromise (IoC) such as a severe IDS alert, HeAT produces a HeATed Attack Campaign depicting the actions that led up to the critical event including reconnaissance and initial exploitation stages. We define the concept of "Alert Episode Heat" to represent the analysts opinion of how much an event contributes to the attack campaign of the critical IoC given their own knowledge of their network context and security expertise. Leveraging a network-agnostic feature set and a short but targeted training process, HeAT is able to realize insightful and concise attack campaigns for IoC's not observed before, compare attack strategies of different attackers with the same IoC, and also be applied across networks with the same degree of fidelity. HeAT maintains the analysts original assessment of the specified "HeAT" regardless of the critical event being assessed or the network topology. We demonstrate the capabilities of HeAT with case studies using cyber-competition datasets to mimic how HeAT would be deployed in practice and assess the HeATed attack campaign from the analyst's perspective. With the goal of aiding the analyst in quickly finding further evidence of an attack, we show that HeAT immediately reveals each attack stage of an attack campaign embedded deeply within millions of alerts that may have needed a whole team of analysts to achieve otherwise.

1 Introduction

Threats of sophisticated and highly impactful cyber attacks have become so common that many organizations have implemented "Security Operations Centers" (SOC) to investigate, respond to, and hunt potential threats within networks. SOC's typically implement a tiered structure where a tier 1 analyst triages the network for critical events which may be escalated to a tier 2 analyst who will respond to the incident. Assume the role of a tier 1 SOC analyst and you observe a critical alert, "*GPL EXPLOIT CodeRed v2 root.exe access*", targeting a customer database. While occurring on a critical asset, a single alert may not be enough evidence to escalate to the tier 2 analyst and you must now look for other "Indicators of Compromise" (IoC) to develop more evidence that the alert was indeed caused by an adversary. This is known as a "triage" and is typically a time consuming and mostly manual process sometimes involving multiple analysts to comb through lengthy log files to find other IoC's related to the initial IoC. With the inflation of network sizes and the general increase of foreign threats broadly targeting any type of organization, many SOC analysts are overwhelmed with the amount of log data from

*sfm5015@rit.edu

Intrusion Detection Systems (IDS) which hampers their ability to quickly assess their network for threats.

Given a critical alert (an IoC) and IDS alert logs, we ask if we can leverage machine learning techniques to aid the analyst in the triage process and automatically reveal other steps the adversary took to “arrive" at their goal. The compilation of the actions detailing each “stage" of the attack is called an “attack campaign" which would describe how, when, and where the attacker learned about the network, gained initial access, and then eventually achieving their goal. Developing this attack campaign from IDS alert logs can be extremely difficult as the analyst must consider for each alert: the network context, related attributes between the alerts, and their own expertise to determine the relationship between the critical alert and prior alerts. These considerations sometimes leads to subjectivity of the actual contribution of the alert to the attack campaign. We envision an automated triage system to reflect the analyst’s opinion on the types of events that they believe are a part of an attack campaign and ability to apply that “thinking" to other triages in the future.

We propose a system, HeATed Alert Triage (HeAT), to perform automated triaging of IDS alerts. Given a critical IDS alert, HeAT creates a “HeATed Attack Campaign" (HAC) using a set of network agnostic features and a small set of analyst defined critical alert episode relations. In the form of aggregated alerts defined as “Alert Episodes," the HAC’s generated by HeAT tells the story of the attacker’s progression leading to a critical event. HeAT estimates the "Alert Episode HeAT” (AEH) for each alert episode with respect to the critical alert to describe the episodes contribution to the attack campaign given how the analyst has interpreted AEH previously. We have developed HeAT with reusability and transferability in mind; we use network agnostic features so that HeAT can uncover attack campaigns for other critical alerts, adversaries, or networks. We envision HeAT to be used by SOC analysts to display the HAC once they observe the first IoC so that they can quickly determine if further action is needed for not only one attack type but for many. Note that we demonstrate the methodology and capability of HeAT with one specific IDS, Suricata, in this work, while the network agnostic features are generalizable to treat heterogeneous alerts and event logs.

Using a set of targeted case studies by processing data collected through cyber-competitions, we demonstrate, in close to real “deploy-able" scenarios, HeAT’s ability to:

1. Leverage a small amount of self-labeled data to discover meaningful insights into attack campaigns for critical alerts,
2. compare attack strategies for the same critical event and quickly determine key milestones such as discovery, initial access, and other events leading up to the critical event, and
3. identify attack campaigns under different network settings, and reveals non-coincidental patterns in attack strategies across networks.

2 Related Work

Extraction and assessments of attack campaigns has been studied in depth in the form of Attack Graphs (AG) and have the capability to provide detailed insights into *how* attackers can traverse a network. AG’s use network topology and vulnerability assessments to define potential paths through a network an adversary can exploit. AG works employ techniques such as alert correlation [20, 25, 23, 22], process-mining [5, 3], and Markovian-based approaches [7, 8] to map observables to pre-existing AG’s. However these approaches require a significant amount of expert knowledge to configure, create attacker scenario templates, and assumes that each vulnerability is known [2]. If we constrain our research to find approaches that give AG-like insight without intimate knowledge of the network and vulnerabilities, we find significantly less works within academia. Navarro *et al.* presents HuMa [16] and OMMA [17] to extract context from logs, vulnerability databases like CVE and CAPEC, and analyst feedback to find malware behaviors. Moskal *et al.* used a suffix-based Markov chain to derive sequences of aggregated alerts based on their alert characteristics called *attack episodes* so that sequences of episodes could be compared [14]. Landauer *et al.* [11] extracts from cyber threat intelligence (CTI) reports and applies the knowledge to raw log data to report actionable multi-stage scenarios. Lastly, Nadeem *et al.* [15] present SAGE which employs S-PDFA to extract meaningful AG's from only intrusion alerts and without prior expert knowledge. However, an issue that plagues these works is the lack of high quality labeled attack scenario data to comprehensively assess, compare, and validate the identified attack strategies.

In the private sector, where data is more abundant, the concept of AI-driven products to assess and automatically triaging a network is an extremely fast growing sector. As of 2021, the adoption of AI/ML techniques to solve cyber security problems has exploded. To name a few, companies such as DarkTrace with their "Cyber AI Analyst" [4], IBM with QRadar Advisor with Watson [10], and Centripetal with AI-Analyst [18] all advertise their capabilities to leverage AI specifically to aid analysts in the triaging process. While these products are undoubtedly extremely sophisticated due to their substantial resources, it is impossible to assess their true capabilities due to the proprietary nature of the method and data. While we do not claim to compete with these products, their existence shows that this is a developing and more notably a valuable problem to address and document.

3 HeAT: Extraction of a Heated Attack Campaign (HAC)

Given the IDS alert logs from a network and an IoC such as a critical IDS alert, our objective is to develop of sequence of alerts likely to be related to the IoC, forming the attack campaign of the adversary. We define an "Attack Campaign" as the collection of actions in time which describe each stage of an attack conducted by an adversary leading to some objective. As there will be no ground truth describing the real attack campaign, we rely on an initial triage to establish characteristics of an actual attack campaign first. Then we address other technical challenges such as high alert volume, network-specific attack characteristics, and limited analyst data-labeling resources to extract meaningful and concise attack campaigns quickly. The summary of the methods used in HeAT are described below and the system overview is shown in Figure 1.

- Introduce the labeling approach called "Alert Episode Heat" (AEH) as a numeric ranking system (0-3) representing key milestones of an attack campaign leading towards an IoC,
- propose a short and efficient labeling process to capture the analyst's reflection of meaningful relationships between alert episodes contributing to the same attack campaign,
- use an attack stage-based Gaussian smoothing approach to alert aggregation to create alert episodes indicative of actions performed by adversaries,
- use alert episodes to derive network agnostic features relating characteristics between episodes, enabling prediction of AEH regardless of attack type or network configuration,
- use ML/AI to learn and predict AEH values for prior episodes to a critical episode, and
- construct and visualize HeATed Attack Campaigns (HAC) with "HeATed episodes".

In the subsequent sections in this work we define the concept of AEH, describe our process to aggregate alerts as alert episodes, define our AEH labelling process, and then finally describe our application of AEH to generate the HAC for a given critical alert.

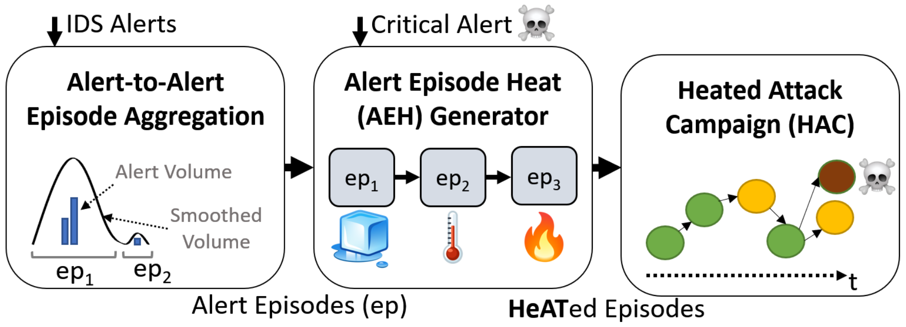


Figure 1: Process overview of HeAT to generate HAC from a set of IDS alerts and a given critical alert.

3.1 Alert Episode Heat - Progress Towards Attack Objective

The concept of Alert Episode Heat (AEH) is a numeric ranking system (0-3) which given a critical alert episode e_c and a prior episode e_p, AEH ranks the contribution of e_p to the attack campaign of e_c. We use the concept of “alert episodes" to represent groups of alerts that are indicative of action(s) with a specific impact. Each alert episode may contain one or many alerts sharing similar attributes, such as attack impact, which may or may not be related to the campaign of e_c. AEH is intended to capture the attacker's progression towards e_c given the alerts of e_p.

While many IDS's already have some notion of severity embedded within the alert (Suricata's severity attribute), these are typically static and independent from all other alerts that have occurred. IDS's such as Suricata have no notion of correlated alerts but simply report suspicious behavior based on signature matches of known adversarial actions and additional information is needed to determine if two events are correlated. Additional factors such as the network topology, the assets contained on specific machines, and the analyst's own expertise is considered when correlating the true severity between security events. The concept of Alert Episode Heat (AEH) is to create these correlations between a critical episode and the episodes prior. Given e_c and e_p, our objective is to define an AEH Generator as: $h(e_p|e_c) = f(e_p, e_c)$ where $\{h \in \mathbb{R} | 0 \leq h \leq 3\}$.

We design the AEH values as a small set of discrete values that signify key milestones within an attack campaign. Table 1 describes the characteristics of the “HeAT levels" which are used to label and create the initial AEH training set. We use the high-level attack stages such as “reconnaissance", “exploitation", and “actions on objective" [12] to represent heat levels 1, 2 and 3, respectively, to reflect their progressive impact on a network. We choose a “less-is-more" approach as we embed specific attack stage information within our labels and human studies show that 3 to 4 options is optimal reduce error for human surveys [1]. With HeAT level representing a small number of mutually exclusive attack stages, we believe the analyst can quickly determine an appropriate HeAT level and we believe there will be less ambiguity between HeAT levels.

Table 1: Description of the AEH levels relating to attack milestones

AEH	Description
0	No relation to critical event
1	Recon. actions that may provide info. about e_c
2	Exploitation of assets giving access required to achieve e_c
3	Exfiltration/DoS/Access to info. directly relevant to e_c

Given our focus on episodes we now describe our process for converting individual alert streams into aggregated alerts to “Alert Episodes" and then features that describe our episodes.

3.2 Alert Episodes with the Action-Intent Framework (AIF)

When discussing IDS's it is common knowledge that they are plagued with generating a high volume of alerts due to false positives, vague signatures, or actions that cause excessively repeated alerts. Alert aggregation is used to group alerts of similar attributes such as time proximity or impact to reduce the number of events presented to the analyst but also could represent an action performed by the adversary. Our main limitation is that we only have the attributes defined within the Suricata IDS alerts, which is limited in scope, however we use the alert signature description to deduce the "type" of action the adversary could be performing. Given the alert attributes, we define an "Alert Episode" to be the set of aggregated alerts for a single source IP and same attack stage across multiple target IP's within a similar time proximity. We accept that source IP is not a totally reliable attribute, we believe it is the best opportunity to capture alerts caused by one adversary.

We adopt the Gaussian Smoothing approach as described by Moskal *et al.* to aggregate alerts based on source IP, attack stage, and time [14]. Moskal *et al.* [14] describes a process where alerts are aggregated based on the fluctuations of alert volume within a time window for specific IP addresses and Suricata categories to uncover common sub-sequences of attack patterns. We choose this process due to its effective application of Gaussian smoothing to represent aggregate alerts whose the alert arrival time may be inconsistent, sporadic, or periodic. Moskal *et al.* concludes [14] mentioning that the Suricata alert "category" is a weak representation attack stages of an well established attack stage framework such as MITRE ATT&CK. The Action-Intent Framework (AIF), also defined by Moskal *et al.* [13], was created as an IDS-focused version of MITRE ATT&CK and other kill chain frameworks. A mapping between Suricata signatures and a Action-Intent Stage (AIS) is given. Gaussian low-pass filtering is applied to histograms in time of alert volume for single IP and AIS, where the LPF filter parameter is set based on the expected duration of the action on a per AIS basis. Certain types of attacks may have longer duration than others and thus different filter sizes are used.

Our Alert Episodes are derived by evaluating each peak of the AIS-based filtered histograms and the collection alert(s) contained in-between the two local minima of the corresponding peak make the episode. An example of this process can be seen in Figure 2. Conducting this process over each attack stage for each source IP, combining the derived episodes, and sorting the by the peak episode time gives an abbreviated view of the sequence of "actions" performed by that adversary.

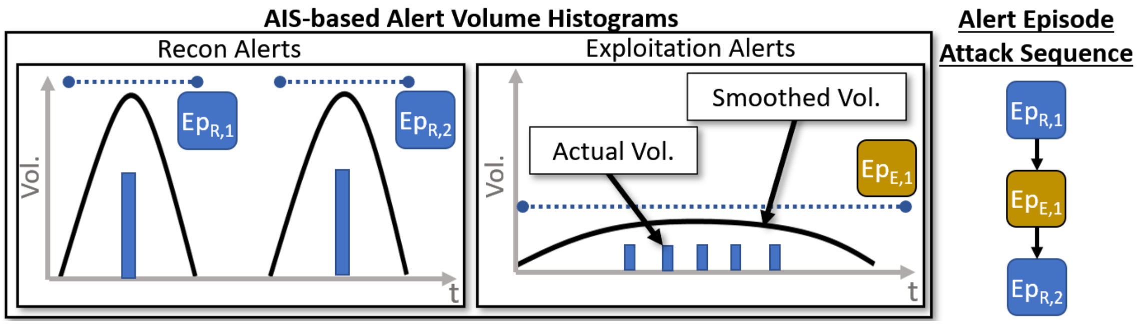


Figure 2: The Gaussian smoothing approach by Moskal *et al.* accounts for variations in alert arrival time to create Alert Episodes and creates sequences of episodes by sorting episodes by peak smoothed volume times.

This episode representation not only summarizes the alerts but also enables us to define network-agnostic features to compare episodes. Next, we define our network-agnostic features used to represent the relationship between two alert episodes and the AEH.

3.3 Network Agnostic Features between Alert Episodes

We engineer our features with two elements in mind: 1) the features describe relations between two episodes so that the AEH value can be determined with respect to a critical episode and 2) the features are network agnostic so that the model does not learn network specific HeAT relations that cannot be applied to other attack types or network configurations. As the episodes contain set of alerts with a wide variety of complex data types such as IP addresses, alert signatures, *etc.*, we manually define a set of episode features to represent each of these data types. Each alert episode contains the attributes shown in Table 2 which are derived from the alerts contained within the episode.

Table 2: Definitions of the attributes contained within an alert episode.

Name	Symbol	Description
Ep. Peak	e_{peak}	Time of peak alert volume
Ep. Start	e_{start}	Time of earliest alert
Ep. End	e_{end}	Time of latest alert
Distinct Source(s)	e_{src}	Set of distinct source IP(s)
Distinct Target(s)	e_{tgt}	Set of distinct target IP(s)
Distinct Sig(s)	e_{sig}	Set of distinct signatures
Distinct Dest. Port(s)	e_{port}	Set of distinct dest. ports
AIS	e_{ais}	AIS of the episode

We define three types of features to capture different aspects of common characteristics between episodes: 1) Time, 2) IP, and 3) Action based features, shown in Table 3. The time-based features capture the differences between the critical alert episode and the prior episodes. Our IP-based features compare if there are similarities between IP addresses of the two episodes without defining any details of the IP addresses themselves. Lastly, our action-based features capture similarities between attack stages, signatures, and port numbers to determine if the two episodes have a similar network impact.

Table 3: The set of network agnostic features relating the attributes of two alert episodes.

Type	Feature	Description
Time	Ep. Interval Overlap	Overlap between the start & end times of e_c and e_p
	Ep. Peak Time Diff.	$e_{c,peak} - e_{p,peak}$
	Ep. Start Time Diff.	$e_{c,start} - e_{p,start}$
	Ep End Time Diff.	$e_{c,end} - e_{p,end}$
IP	Has Matching Source	1 if $e_{c,src} \cap e_{p,src}$ else 0
	Has Matching Target	1 if $e_{c,tgt} \cap e_{p,tgt}$ else 0
	Matching Source Ratio	Ratio of matching source IPs
	Matching Target Ratio	Ratio of matching target IPs
	Crit. Source as Target	1 if $e_{c,src} \cap e_{p,tgt}$ else 0
	Crit. Target as Source	1 if $e_{c,tgt} \cap e_{p,src}$ else 0
Action	Critical Ep. AIS	1-hot encoded $e_{c,AIS}$
	Prior Ep. AIS	1-hot encoded $e_{p,AIS}$
	Has Matching Sigs.	1 if $e_{c,sig} \cap e_{p,sig}$ else 0
	Matched Sig. Ratio	Ratio of matching signatures
	Matching Dest. Port	1 if $e_{c,port} \cap e_{p,port}$ else 0

Our hypothesis is that these network agnostic features will allow us to uncover a variety of attack campaigns without detailed network topology or system vulnerabilities. We propose that these network agnostic features can be used to predict the AEH for other attack types and be applied to other networks. In the next section we describe our methodology for creating the AEH Generator and how we leverage a small amount of labeled AEH values to determine HeATed Attack Campaigns (HAC).

3.4 Alert Episode Heat Generator

When defining the concept of the AEH generator significant challenges arise as labeled data describing an attack campaign with respect to IDS alerts generated is few and far between. Data sets that do exist within the research community are typically either outdated (irrelevant attack types), unlabeled, and/or represented in a different domain (*i.e.,* packet captures) than IDS alerts. Instead we have the user conduct an initial “triage" of their IDS alerts, label AEH values to episodes related to a known IoC, and then use the network-agnostic features to create a predictive model to “generate" heat given other IoC’s.

3.4.1 HeATing Episodes to Develop HAC

Given an AEH-labeled data set, we train a AEH predictive model known as the AEH generator as $h(e_p|e_c) = f(e_p, e_c)$ where $\{h \in \mathbb{R}|0 \leq h \leq 3\}$. We define a HAC for given a critical episode e_c as the set of all prior episodes $e_p \in E$ where the AEH generator applies a non-zero AEH. Our requirements for a selecting a machine learning model for this application are bound by our non-linear features and that the AEH value must be a continuous value. HeAT is implemented in Python and our AEH generator leverages Fast.AI’s Tabular learner [6] to predict the AEH. All features within our data are standardized to have a zero mean and unit variance and we report our 5-fold cross-validated mean squared error (MSE) for the training data.

The process of extracting the HAC from the our data is similar to our training process where a critical alert is given by the user, HeAT finds the corresponding episode containing the critical alert, and then the AEH generator is used to “HeAT" *all* prior episodes with respect to the critical episode. We apply HeAT to all prior episodes to give our model the opportunity to discover episodes that may have significantly contributed to the attack campaign that may not immediately obvious. The set of HeATed episodes with a non-zero AEH are then considered to be apart of the HeATed attack campaign of the critical alert. As the episodes may contain many alerts, we foresee the generator finding small relations to the critical episode and apply a small amount of HeAT to episodes that may not contribute much to the overall campaign; a minimum AEH threshold can be applied if the user desires. However, we expect truly impactful episodes to have significantly higher AEH levels than those with just a few similarities between features.

4 Datasets for Training and Testing

To demonstrate HeAT’s ability to uncover insightful attack campaigns within IDS alerts, we choose to use data that is known to have actual examples adversarial behavior. In this work we use publicly available data from penetration competitions such as CPTC ‘18 [21] and CCDC ’18 [9] as it contains the impacts of multiple different adversaries in a controlled and isolated network scenario. Competition sets like this give us the opportunity to use HeAT to discover minute differences between attacker strategies with the same critical alert. We set-aside the alerts of one of the 10 teams as “team train" to create our initial triage data and compare the results of HeAT against the remaining teams as "team test." The Suricata IDS alerts from the CPTC ’18 event are publicly accessible from [19] and we leverage the Suricata-to-AIS mapping provided by Moskal *et al.* [13] to gain a better representation the attack stage of each alert than what is provided from only Suricata. CCDC data is only available as packet captures from [24] and we convert the PCAP into Suricata alerts using the PCAP mode of Suricata using the default Suricata rule set. Table 4 summarizes the characteristics of each of these datasets demonstrating the difference in the number of sources and targets, the large disparity in alert volume, and reduction of alerts when represented as alert episodes.

Table 4: Characteristics of the alerts contained within the CPTC and CCDC data including the total count of alert episodes.

	Unique Sources	Unique Targets	Unique Sigs	Total Alerts	Total Episodes
CCDC	485	2903	348	8,738,994	2852
CPTC (All Teams)	45	81	265	169,448	3200
CPTC “Team Train"	29	49	171	53,362	529

To create the training data for the AEH generator, we ask the user to perform a short initial triage of the prior episodes with respect to a critical IoC and apply AEH values representing their opinion of the attackers progress contributing towards the IoC. For each prior episode to the IoC, we compare attributes between the two episodes such as: time difference, IP address similarities , AIS, and signatures to derive an appropriate AEH given our experience of the network, data characteristics, and our own security knowledge. In practice we recognize that analysts could be faced with an excessive amount of data to be labeled. However in this case, we assume our network agnostic feature set can lessen the reliance on massive amounts of labeled data typically needed for machine learning applications along with our intuitive AEH definition to create a more efficient labeling process.

To test this assumption, we use the CPTC team set aside, “Team Train", and use those episodes to create our initial AEH Generator training data. We selected three critical IoC’s to initially manually triage: 1) “*ETPRO ATTACK_RESPONSE MongoDB Database Enumeration Request*", 2) “*ET EXPLOIT Possible ETERNALBLUE MS17-010 Heap Spray*", and 3) “*GPL EXPLOIT CodeRed v2 root.exe access.*" These signatures describe data exfiltration, arbitrary code execution, and root privilege escalation actions respectively and can lead to significant access and impact if successful. These were selected for the purposes of our intended case studies where: CodeRed was observed in both CPTC and CCDC; Mongo is exclusive to CPTC; and Eternal Blue is exclusive to Team Train. We then manually apply AEH to prior episodes to the critical IoC’s as shown in Table 5 with the distribution of AEH values given in Table 6.

Table 5: Count of alerts and episodes containing each IoC for CPTC “Team Train" and the entire CCDC dataset along with the distribution of episodes with AEH manually applied.

IoC Signature	# IoC Alerts		# IoC Episodes		# Prior Eps. AEH Applied
	Train	CCDC	Train	CCDC	
Mongo	125	0	69	0	756
Eternal Blue	4	0	4	0	231
CodeRed	146	27	63	4	436

Table 6: Distribution of Team Train’s AEH values of prior episodes given the IoC’s from Table 5

Training AEH	AEH Count
0	720
1	202
2	154
3	347

This training data relating two episodes together through the AEH value given is then converted into our feature space and used to create the AEH Generator to be used to reveal future attack campaigns with a small portion of the original alert set. As we show in the next section, our HeAT process is able to extract key characteristics of the attack campaigns given the limited amount of labeled data. While we show that the capabilities with just this amount of data is impressive, we plan on conducting another study in the future with multiple analysts and across other scenarios to assess how the quality of the initial triage affects the resulting HAC.

5 HeATed Attack Campaign (HAC) Analysis Using AEH

Demonstrating the effectiveness of a process like HeAT is extremely challenging due to the lack of standardized training sets. Instead we take an alternate approach by performing a set of case studies that mimics the type of analysis we would expect to see if HeAT was deployed in a real world scenario. First we perform standard ML techniques to assess our performance of predicting AEH values given the training data described prior. Performing 5-fold cross validation and using mean-squared error as our performance metric, we report average MSE across the 5 folds as **.175** with a maximum error of .335. This is promising as it seems that our model can capture the AEH values of our training data appropriately and can be applied to other IoC's.

Leveraging this model to predict the HeAT of other attack campaigns, we apply HeAT to the remaining CPTC teams, "Team Test", using Mongo and CodeRed as our IoC's. We measure our reduction of relevant events by comparing the number of HeATed episodes to the total number of "related" alerts and episodes. A related alert (or episode) in this context is any alert that shares an IP address (source or destination) with the critical IoC. For the two IoC's there was 16 HAC's generated and the summary of those HAC's are given in Table 7.

Table 7: Summary of the HAC's generated for Team Test given the latest in time Mongo and CodeRed IoC's.

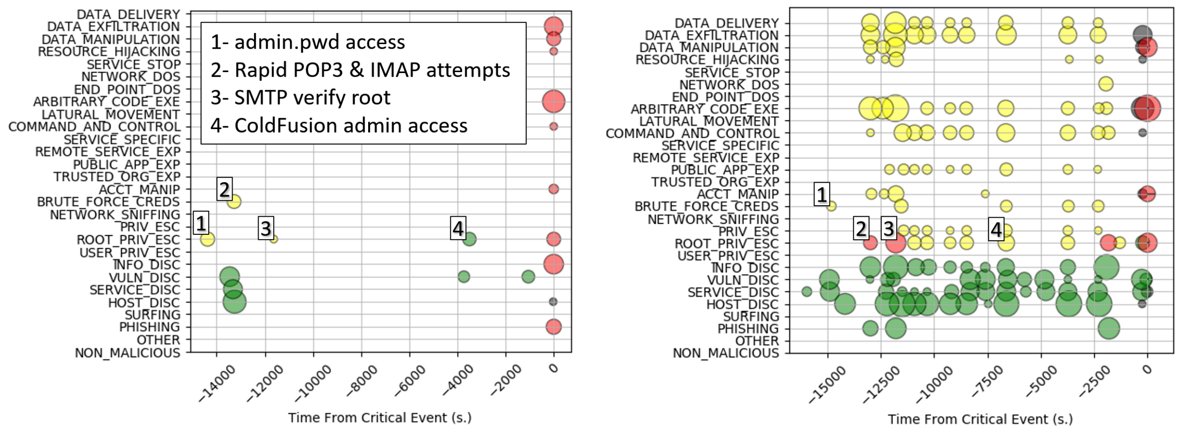
	"Related" Alerts	"Related" Episodes	HeATed Alerts	HeATed Episodes	Reduction of Episodes
Average	4,228	196.89	668.33	22.28	**174.62 (-88.7%)**
Min	73	83	150	5	
Max	32741	389	4470	149	

From the perspective of a SOC analyst, this 88% reduction of events needed to be triaged may lead to more timely response to threats as less time will be spent going through irrelevant alerts. However this will only be true if the HeATed episodes realized by HeAT are an accurate representation of the attack campaign conducted by the adversary. We now conduct a set of case studies to assess the fidelity of the HAC's generated by HeAT.

5.1 Case Study: Tactics towards CodeRed

Our CPTC dataset gives us the unique opportunity to investigate differences in attack strategies of different adversaries given the same objective, or in this case the same critical alert. We propose the scenario where a user has identified this critical alert in the past, gone through the HeAT training process for this critical alert, and then re-observes the same critical alert in the future. Although the impact of the vulnerability is the same, we expect the approach taken could be different and thus a different mitigation strategy would be required. We use apply HeAT to different adversaries to demonstrate how HeAT is used to gain additional insight into attack strategies. We find through this case study that HeAT can reveal attacker strategies of adversarial behavior unknown to HeAT.

For this case study, we used the alert "*GPL EXPLOIT CodeRed v2 root.exe access*" as it was prevalent across many adversaries and it leads to a significant amount of access if successful. To demonstrate the diversity of the HAC's that can be extracted, we begin by comparing two particularly interesting HAC's who's approaches were vastly different from each other. Figure 3a shows the resulting HAC for a "calculated" adversary, where we have discovered four key steps that leads to the CodeRed signature out of a total of 19 HeATed episodes. Whereas the HAC for a "script kiddie", shown in Figure 3b, demonstrates a significantly different approach where we show a HAC consisting of 144 episodes to achieve the same objective. Due to the complexities of the script kiddie HAC, our annotations are as follows: 1) Rapid POP3 & IMAP attempts, 2) CVE-2014-6271 'Shellshock', 3) CodeRed First Attempt, and 4) CodeRed on other targets, same source. The HeATed episodes of the calculated and script kiddie represents 1458 and 29,015 individual alerts respectively, thus our HAC's are a substantial reduction of events an analyst needs to assess.



(a) "Calculated" HAC (b) "Script Kiddie" HAC

Figure 3: HAC's of two separate CPTC adversaries given the CodeRed IoC. Each circle is an episode where the size corresponds to the number of alerts within the episode. Y-axis represents AIS and color corresponds to the AEH value of the episode (red = high AEH)

Our rationale for distinguishing the "calculated" adversary versus the "script kiddie" is from our analysis of the HeATed episodes. The "calculated" episodes had a single target throughout this HAC and significant time between episodes. Whereas the script kiddie maintained a consistent time between actions and targeted many different IP's with the same action which is indicative of a scripted process. Without HeAT the count of all episodes occurring within these time-frames for the calculated and script kiddie adversary was 198 and 498 episodes, HeAT provides a reduction of the number of episodes to be considered by 92% and 71% respectively. By investigating the alerts within the smaller amount of HeATed episodes, we were able to gain insight into the individual campaigns as shown by our annotations.

Due to our alert episodes representing groups of alerts occurring within a similar time frame and AIS, we find that interfacing with the individual alerts to be much more manageable compared to the raw alerts. Here we found indications that vulnerabilities within a mail server is likely be targeted when CodeRed is seen as both of our adversaries have early episodes targeting POP and IMAP. By looking further into the HAC we see that the mail server maybe initial access and both adversaries used other actions such as the ColdFusion vulnerability and the Shellshock attack to further position themselves within the targeted asset. While other evidence is needed to definitively prove these observations, we believe the resultant HAC's provides substantial evidence for an SOC analyst to escalate investigations to the next tier of analysis. Given that a network may produce seemingly endless volume of alerts, the realization of attack campaigns becomes overwhelming to conduct manually and we believe HeAT alleviates the strain so that prevention measures acted on quicker.

5.2 Case Study: HeATing a Different Network

We now ask, can HeATed episodes from one network explain an attack campaign on another network? For this case study we use the CCDC dataset, which was conducted over a shorter time period, contains orders of magnitude more alerts, and we have little to no knowledge about the real network topology or the adversaries themselves. We find this to be a realistic use-case where a user may use HeAT from another network to gain insight into new network that they may be unfamiliar with. Using the exact same AEH generator, CodeRed as our critical alert, and directly applying HeAT to the CCDC dataset with no extra configuration or training, we present the HAC for this case study in Figure 4

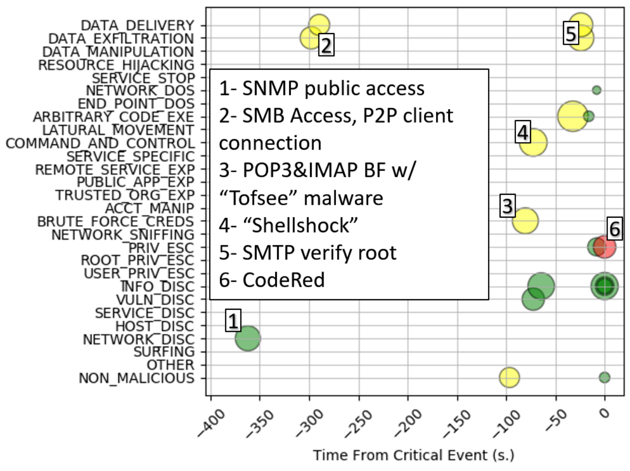


Figure 4: The HAC of CodeRed on the CCDC set using the AEH Generator trained on CPTC data

While visually the HAC's of the CPTC and CCDC do not seem to be similar, upon deeper inspection into the alerts contained within the episodes, we do find some striking similarities between the HAC's. Initially, we see step 1 making the initial discovery of the critical asset and step 2 describes a Peer-to-Peer connection made to a SMB network share. The corresponding signatures within this episode report that data was transferred to and from this asset. This is a substantial discovery as it indicates that the SMB share is vulnerable which enabled the adversary to deliver malware as shown in Step 3. Steps 3-5 in Figure 4 are interestingly similar episodes that we have seen previously including references to POP3 & IMAP brute force, Shellshock, and the SMTP verification of root access. Without HeAT we believe that finding such campaign similarities would be much more difficult as the CCDC data contained two orders of magnitude more alerts. However our HAC does not tell the whole store and further research would be needed to determine if these alerts were caused by similar tools used by the adversaries or purely coincidental.

The capabilities of HeAT to reveal convincing attack campaigns across multiple networks seems to be impressive and we demonstrate that in some cases such as CodeRed the campaigns are similar regardless of the network. These observations indicate that prior observations of attack campaigns can be generalized using HeAT's network-agnostic features and can be used to find similar strategies in other networks. This sort of capability could provide immense value to analysts hunting threats as HeAT could be applied regardless of the network context while still providing high quality automated triages. In our future work, we investigate the validity of this observation through a comprehensive study of many more critical events across more networks. Given our observations with our case studies however, we have confidence in HeAT's ability to provide users the first round of insight into attack campaigns occurring in their network with simply a single observed alert and a straightforward training process.

6 Limitations

Throughout our research of HeAT we have identified several limitations, some technical and others are methodological limitations. First for technical limitations, we only evaluate HeAT as a post-mortem analysis tool and do not initially intend for HeAT to be used in real-time applications. Also the HeAT process only considers network-based IDS Suricata alerts which does not have the ability to resolve actions that do not produce network traffic nor does it handle encrypted traffic well. For extended detail in HAC's we would consider the addition of other security monitors such as the host-based IDS Zeek or ther vulnerability scanner Nessus to further enhance HeAT.

Our methodological limitations stems from our training process: *i.e.,* the data used, the quality of the analyst's AEH assessments, and the verification of the produced HAC's. HeAT relies on the analyst to provide accurate AEH values but you may ask: 1) What if the AEH values given are incorrect?, 2) What if you have multiple analysts?, or 3) What if the data is a bad representation of attack campaigns? Each of these questions could have detrimental impact to the utility of HeAT, especially in a high-stakes field such as network security. We propose to conduct human studies of different analysts under controlled scenarios to answer these questions. A future iteration of this work will conduct significantly more case studies of attacker behaviors, provide more statistical analysis of the completeness/utilities of the HAC's, and be applied to a real SOC operation to determine the true real-world capabilities of HeAT. In this case we have the luxury of knowing our data well and certainly is abundant with adversarial behavior, something that cannot always be said about real-world data.

7 Conclusion

The case studies presented in this paper demonstrated how an analyst could use HeAT to identify attack campaigns leading to a critical IoC with minimal expert inputs. Our HeATed Attack Campaigns represents the attack stages in time related to a given critical IoC as set of Alert Episodes with associated Action-Intent Stage. Defined as the "Alert Episode Heat" (AEH), the AEH captured the analysts reflection of an episode's contribution to a specified attack campaign. We found that our implementation of HeAT with network agnostic feature descriptions was able to extract attack campaigns from alerts novel to HeAT, show the difference between diverse attacker strategies, and be applied to multiple networks to perform a HeATed triage. We demonstrated through analysis of our cyber-competition datasets that HeAT can quickly show the difference between attacker strategies with the same objective and do so in a concise alert episode representation without ever interfacing with alerts directly. We also demonstrate HeAT's ability to maintain the important attack campaign characteristics defined by the analyst for multiple networks; where we captured non-coincidental similarities in attacker strategies between our CPTC and CCDC datasets.

For the foreseeable future, triaging networks to assess attack campaigns is only going to get more difficult and time consuming to the point that manual triaging becomes infeasible. As seen in the related works within the private sector, we believe that automated triaging with AI/ML will be a necessary component to all SoC operations. We believe that HeAT is a viable solution that requires minimal expertise and analyst' effort, and can inspire more research into AI-based automated triage. We are in the process of making HeAT available and open sourced. Aside from the future works proposed within our case studies, we plan for HeAT to become integrated as a plug-in with Splunk or other SIEMs. We also plan to extend HeAT to account for a variety of intrusion alerts and event logs. Particularly, we consider integrating with Zeek logs and phishing email detection with HeAT to broaden the insights and address more complex attack types.

References

[1] Alchemer. Number of choices in survey questions: How much is too much? `https://www.alchemer.com/resources/blog/survey-questions-how-much-is-too-much/`, 2011. [Online; accessed 21-June-2021].

[2] 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.

[3] 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.

[4] Darktrace. Darktrace cyber ai analyst: Autonomous investigations. `https://www.darktrace.com/en/resources/wp-cyber-ai-analyst.pdf`, 2021. [Online; accessed 21-June-2021].

[5] 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.

[6] Fast.AI. Fastai python library v1.0.57. `http://docs.fast.ai/`, 2020. [Online; accessed 28-April-2020].

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

[8] 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.

[9] F. Hassanabad. suricata-sample-data. `https://github.com/FrankHassanabad/suricata-sample-data/blob/master/README.md`, 2019. [Online; accessed 5-May-2020].

[10] IBM. Ibm qradar advison with watson. `https://www.ibm.com/products/cognitive-security-analytics`, 2021. [Online; accessed 21-June-2021].

[11] M. Landauer, F. Skopik, M. Wurzenberger, W. Hotwagner, and A. Rauber. A framework for cyber threat intelligence extraction from raw log data. In *Proceedings of 2019 IEEE International Conference on Big Data (Big Data)*, pages 3200–3209. IEEE, 2019.

[12] L. Martin. Cyber kill chain | lockheed martin security. `http://cyber.lockheedmartin.com/solutions/cyber-kill-chain`, 2016. [Online; accessed 11-April-2016].

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

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

[15] A. Nadeem, S. Verwer, S. Moskal, and S. J. Yang. Alert-driven attack graph generation using s-pdfa. *Accepted to Appear in 2021 ACM KDD Workshop on Artificial Intelligence-enabled Cybersecurity Analytics (AI4Cyber)*, 2021.

[16] J. Navarro, V. Legrand, S. Lagraa, J. François, A. Lahmadi, G. De Santis, O. Festor, N. Lammari, F. Hamdi, A. Deruyver, et al. Huma: A multi-layer framework for threat analysis in a heterogeneous log environment. In *International Symposium on Foundations and Practice of Security*, pages 144–159. Springer, 2017.

[17] J. Navarro, V. Legrand, A. Deruyver, and P. Parrend. Omma: open architecture for operator-guided monitoring of multi-step attacks. *EURASIP Journal on Information Security*, 2018(1):1–25, 2018.

[18] C. Networks. Ai-analyst. `Centrcentripetalnetworks.com/hubfs/Data%20Sheets/CI_AI_Analyst_Brief.pdf`, 2018. [Online; accessed 21-June-2021].

[19] R. I. of Technology. Index of cptc2018. `http://mirror.rit.edu/cptc/2018/`, 2021. [Online; accessed 21-June-2021].

[20] X. Qin and W. Lee. Discovering novel attack strategies from infosec alerts. In *Proceedings of 2004 European Symposium on Research in Computer Security*, pages 439–456. Springer, 2004.

[21] Rochester Institute of Technology. Collegiate penetration testing competition. `hhttp://nationalcptc.org`, 2018. [Online; accessed 19-July-2018].

[22] C.-H. Wang and Y.-C. Chiou. Alert correlation system with automatic extraction of attack strategies by using dynamic feature weights. *International Journal of Computer and Communication Engineering*, 5(1):1, 2016.

[23] L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer communications*, 29(15):2917–2933, 2006.

[24] WRCCDC. Wrccdc public archive. `https://archive.wrccdc.org/`, 2021. [Online; accessed 21-June-2021].

[25] B. Zhu and A. A. Ghorbani. Alert correlation for extracting attack strategies. *IJ Network Security*, 3(3):244–258, 2006.