Rochester Institute of Technology
RIT Digital Institutional Repository

Theses

12-2021

HeAT PATRL: Network-Agnostic Cyber Attack Campaign Triage With Pseudo-Active Transfer Learning

Stephen Frank Moskal
sfm5015@rit.edu

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation
Moskal, Stephen Frank, "HeAT PATRL: Network-Agnostic Cyber Attack Campaign Triage With Pseudo-Active Transfer Learning" (2021). Thesis. Rochester Institute of Technology. Accessed from

This Dissertation is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

HeAT PATRL: Network-Agnostic Cyber Attack Campaign Triage With Pseudo-Active Transfer Learning

by

Stephen Frank Moskal

A Dissertation Template Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy in Engineering

Supervised by

Dr. Shanchieh Jay Yang
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
December 2021

Approved By:

Dr. Shanchieh Jay Yang
Primary Advisor

Dr. Michael E. Kuhl
Committee Member

Dr. Andres Kwasinski
Committee Member

Dr. Qi Yu
Committee Member

To my parents, my wife Abbey, and Luna

Abstract

SOC (Security Operation Center) analysts historically struggled to keep up with the growing sophistication and daily prevalence of cyber attackers. To aid in the detection of cyber threats, many tools like IDS's (Intrusion Detection Systems) are utilized to monitor cyber threats on a network. However, a common problem with these tools is the volume of the logs generated is extreme and does not stop, further increasing the chance for an adversary to go unnoticed until it's too late. Typically, the initial evidence of an attack is not an isolated event but a part of a larger attack campaign describing prior events that the attacker took to reach their final goal. If an analyst can quickly identify each step of an attack campaign, a timely response can be made to limit the impact of the attack or future attacks. In this work, we ask the question "Given IDS alerts, can we extract out the cyber-attack kill chain for an observed threat that is meaningful to the analyst?"

We present HeAT-PATRL, an IDS attack campaign extractor that leverages multiple deep machine learning techniques, network-agnostic feature engineering, and the analyst's knowledge of potential threats to extract out cyber-attack campaigns from IDS alert logs. HeAT-PATRL is the culmination of two works. Our first work "PATRL" (Pseudo-Active Transfer Learning), translates the complex alert signature description to the Action-Intent Framework (AIF), a customized set of attack stages. PATRL employs a deep language model with cyber security texts (CVE's, C-Sec Blogs, etc.) and then uses transfer learning to classify alert descriptions. To further leverage the cyber-context learned in the language model, we develop Pseudo-Active learning to self-label unknown unlabeled alerts to use as additional training data. We show PATRL classifying the entire Suricata database (70k signatures) with a top-1 of 87% and top-3 of 99% with less than 1,200 manually labeled signatures.

The final work, HeAT (Heated Alert Triage), captures the analyst's domain knowledge and opinion of the contribution of IDS events to an attack campaign given a critical IoC (indicator of compromise). We developed network-agnostic features to characterize and generalize attack

campaign contributions so that prior triages can aid in identifying attack campaigns for other attack types, new attackers, or network infrastructures. With the use of cyber-attack competition data (CPTC) and data from a real SOC operation, we demonstrate that the HeAT process can identify campaigns reflective of the analysts thinking while greatly reducing the number of actions to be assessed by the analyst. HeAT has the unique ability to uncover attack campaigns meaningful to the analyst across drastically different network structures while maintaining the important attack campaign relationships defined by the analyst.

Contents

List of Figures

List of Tables

1. HeAT PATRL: Objective Background and Literature Overview

1.1 Objective Summary and Contributions

Intrusion Detection Systems (IDS) monitor a network's activity for known adversarial behavior; producing logs of "alerts" to notify security practitioners of the suspicious behavior. IDS's are widely adopted by Security Operation Centers (SOC) and enable analysts to identify abnormal behavior, investigate the causes, and close any vulnerabilities to prevent the attack again. Typically a SOC triage is triggered by an "indicator of compromise" (IoC) and the analyst will refer to prior alerts to trace each step an adversary takes, as there are typically other prerequisite steps the adversary needs to accomplish before they can achieve their final objective. These set of steps taken by the adversary is known as the "attack campaign" and the earlier an analyst can identify each step of the attack campaign results in a quicker response in mitigating the threat. However, the triage process is not straightforward. Analysts are often overwhelmed with the constant high volume of alerts produced each day, correlating cryptic alerts to other related alerts, and often have limited time resources. In this dissertation we investigate how we can leverage machine learning techniques combined with cyber-specific domain knowledge to aid in the investigation of IDS alerts to reveal meaningful attack campaigns.

We begin this work by defining how attack campaigns are used currently in practice and we identify that the definition of the stages within a attack campaign for any given scenario is dependent on the context of not only the behaviors of the adversary but also some network characteristics. Then we investigate the causes of the overwhelming number of alerts produced by an IDS, current methods in reducing the number of alerts, and how current works use IDS alerts to discover adversarial behavior. This is where we find many technical challenges as data-sets of describing attack campaigns within IDS alerts is extremely limited or outdated, making it especially challenging to develop and verify methods to extract attack campaigns. This motivated us to investigate and

innovate on techniques for identifying attack campaigns with little reliance on large labelled data-sets and instead leverage the analyst's own domain knowledge, unsupervised learning techniques, and clever network-agnostic feature engineering to overcome these data challenges. We present our work as "HeAT-PATRL" which enables analysts to leverage observations from prior triages to provide quick and meaningful representation of attack campaigns given an IoC and maintain those observations across other attack types and network infrastructures.

Our contributions for this dissertation are as follows:

1. We defined the "Action Intent Framework" (AIF) as a set of "Action-Intent Stages" (AIS) to describe stages of an attack campaign with respect to the intentions of the adversary and the ability to observe the intent from the perspective of an IDS.

2. Employed a semi-supervised learning technique known as "Pseudo-Active Transfer Learning" (PATRL) to translate *any* IDS alert description to the AIF with limited labelled data, deep-unsupervised language modeling of cyber-specific texts, and transfer learning.

3. Created an attack-stage based alert aggregation method using Gaussian smoothing to aggregate similar sets of alerts into "attack episodes" to substantially reduce the raw volume of alerts into a more meaningful and interpret-able representation based on cyber-characteristics.

4. We used prior knowledge, the opinions of analysts, and network-agnostic feature engineering to enable "Heated Alert Triage" (HeAT) to reveal meaningful and noise-free attack campaigns within a real SOC data-set with campaign-observations from a separate network.

As the basis of an attack campaign revolves around the concept of "attack stages" we now investigate the current definitions of attack stages, most commonly known as the "Cyber Attack Kill Chain", and demonstrate that there is a need for a customized set of stages for IDS alerts.

1.2 The Cyber Attack Kill Chain from the Perspective of an IDS

A necessary component to any cyber attack detection or prediction algorithm is the understanding the tactics and the techniques that adversaries use to learn about the network, compromise assets, and eventually achieve their end-objective (stealing information, disrupting services, etc.). The

techniques and tactics used by cyber adversaries are becoming more sophisticated, ironically, as defense getting stronger and the cost of a breach continuing to rise. Understanding the thought processes and behaviors of adversaries is extremely challenging as high profile or even amateur attacker actions are often complex and difficult for non-security conscious people to understand. As a result, in 2011 Lockheed Martin introduced the concept of the "Cyber Attack Kill Chain" which described a finite set of stages to describe the general process and types of actions an adversary typically takes to achieve some objective [11]. Its high level descriptions of attack stages shown in Figure 1.1 made it intuitive for anyone to understand that the chain should be "cut" as early as possible to limit the impact as much as possible. It became immediately apparent that its over-simplification of the attack process meant that its actual usefulness outside of high-level security presentations were limited, however the concept behind describing attack campaigns of meaningful and unique stages has been adopted by the community.

Figure 1.1: Lockheed Martin's original definition of the Kill Chain in 2011 [11].

Since 2011, the concept of the kill chain has become a interesting phenomenon as many different companies have developed their own version of the kill chain. Some kill chains have been developed for specific attack types such as Advanced Persistent Threats (APT) [67], insider threats [68], and other types [59]. The most notable concept added to the original kill chain was from MITRE Unified Kill Chain [50] introducing the idea that the kill chain is cyclical and the adversary may have to repeat the kill chain process multiple times on other victims before they reach their final objective. Imagine the scenario where objective is to compromise an internal company server that requires the adversary to compromise other machines to eventually have the opportunity to reach the target. MITRE's Unified kill chain is shown in Figure 1.2, where even reconnaissance-type actions may

also have its own chain of attack types.

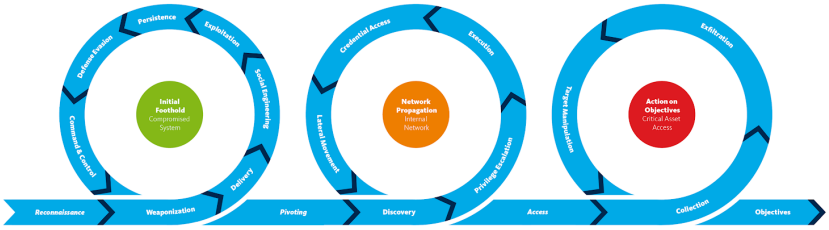

Figure 1.2: The adversary may have to repeat some attack stages before they can achieve their final goal.

While not an exhaustive list, Table 1.1 demonstrates that the number of attack "stages" defined for each kill chain differs depending on the level of abstraction desired by the author. Typical kill chains summarize the entire attack campaign typically in less than 10 stages creating a concise and easy to understand attacker description for both security professionals and the average person. When it comes to generally classifying attack actions, most kill chain descriptions are not well suited to differentiate the different tactics and techniques that an adversary may use. They instead are intended to represent milestones in the attack campaign but not the individual actions. Which leads to undoubtedly the industry leader of cyber attack descriptions, MITRE's ATT&CK Framework, which describes the adversary tactics and techniques based on real-world observations.

Table 1.1: Attack stage classifications by various organizations.

	Year	Type	# of Classes
MTIRE CAPEC [7]	2007	Attack Patterns	517
Lockheed Martin [11]	2011	Kill Chain	7
STIX [63]	2012	Attack Descriptor	9
MITRE Kill Chain	2013	Kill Chain	7
MITRE ATT&CK [39]	2013	Attack Techniques	218
Varonis [23]	2018	Kill Chain	8
MITRE Unified Kill Chain [64]	2018	Kill Chain	20

MITRE ATT&CK [64] uses 14 tactics to classify 218 attack technique classes and is an industry-leading and comprehensive attack-type description that is constantly growing each year. ATT&CK's objective is to be comprehensive and be able to capture nearly any unique attack technique and describe it as concisely as possible. If we had perfect information about every action the attacker

attempting, then the ATT&CK framework is by far the best set of attack stages to concisely describe the attack campaign. If we assume that we can only observe the adversary from the perspective of a network-based IDS then a majority of the ATT&CK techniques defined become irrelevant as most do not generate any network traffic for an IDS to observe.

This becomes our first objective in this work as we ask: "What attack stage framework is the most appropriate to identify the technique performed by the adversary as described by an IDS alert?" We find the more traditional kill chain descriptions to be too high-level to appropriate distinguish meaningful behaviors or patterns of behaviors. We instead use concepts such as the tactics and techniques from MITRE ATT&CK to focus on a set of attack stages that 1) can be observed by an IDS, and 2) describe the intentions of the adversary so that it is immediately clear of the objective of each stage in an attack campaign. We introduce the "Action Intent Framework" in Chapter 2 as our alternative to the ATT&CK framework with a specific focus of attack techniques that can be resolved through an IDS.

This leads us into our next challenge that we must address of the IDS and the numerous alerts generated from these IDS systems. IDS's have no notion of attack stage and is simply reporting if the incoming network packet matches a previously identified "signature" of an attack. These signatures are often not unique and it is very common for normal network traffic to falsely trigger these alerts, contributing to the challenge of assessing IDS alerts. Even if the detection of adversarial traffic was perfect, current definitions for alert rules and especially older legacy rules do not contain an attack stage and typically only contain a cryptic and difficult to interpret description. In the next section, we lay out these IDS challenges in more depth.

1.3 Overwhelmed by IDS Alerts?

In this section we define two core issues with the usage of IDS alerts that make the automatic extraction of attack campaigns especially challenging: 1) the volume of alerts raised per day can be overwhelming for analysts and may contain many false positives, and 2) IDS alerts do not describe the stage of the attack, only providing a cryptic description of the alert. This poses a significant challenge to the analyst as they must use their own knowledge of the network and the alert themselves to identify what the attacker is trying to accomplish and if the alert actually contributes to the

attack campaign that we are investigating. We begin this section with describing the IDS and the alerts generated by these IDS's to investigate why these challenges exist and what is currently being done to address these challenges.

IDS is simply a generic term for an intrusion detection system that monitors the activity of one or many machines on a network and raises alerts if it detects any suspicious behavior. IDS's are not actively attempting to stop or mitigate threats when detected, just report to the user that the behavior was detected. Systems that do automatically respond to the threats are defined as intrusion prevention systems (IPS), however in this work we address the challenges with IDS's as they are much more widely adopted.

There are a few different types of IDS's that a SOC may deploy within their network: 1) network-based, 2) host-based, and 3) anomaly detectors. Network-based IDS's like Suricata [2] monitors network traffic typically deployed at the "edge" of the network (before the internal network). They use a set of rules containing signatures of known suspicious behavior and raise alerts when the incoming traffic matches the signature. Host-based IDS's like Zeek [51] monitors many attributes of individual machines such as processes running, login/logout actvity, file access, and USB access to record system-level behaviors of the legitimate and non-legitimate users. Which leads into the concept of anomaly-based IDS's where a model is created to represent of the "normal" usage of either the network traffic or individual assets and the IDS will report when behaviors out side of the "norm" is seen. Each of these types of IDS's have their own value deployed on a network and each of them also have their short comings, mostly the issue of a high volume of alerts. In this work, we focus on network-based IDS's and specifically Suricata alerts as these types of IDS's integral part of nearly any SOC. Next, we define the contents of a Suricata alert to demonstrate the types of information presented to users and begin to describe how we use these alert attributes to address our aforementioned challenges with IDS alerts.

Suricata alerts are triggered when the incoming traffic through the IDS matches the signature of another known behavior defined in the alert rule. Shown in Figure 1.3 is a generic example of an alert rule where IP's, protocols, packet specific contents, etc. can be used to alert administrators of any behavior, malicious or not. This begins to layout our first challenge of why these IDS's produce so many alerts as the rules can be defined to capture a broad set of behaviors to very specific rules where packet contents must be matched. This is then compounded by the fact these IDS's require

significant tuning for each network or it may result into many false positives [60]. This is a process that many smaller SOC's may overlook as many analysts may not be an expert at configuring these complex systems and instead use the default rule set. We develop our process to keep in mind that analysts may not always be experts in configuring IDS's and that these false positives should be handled by our process.

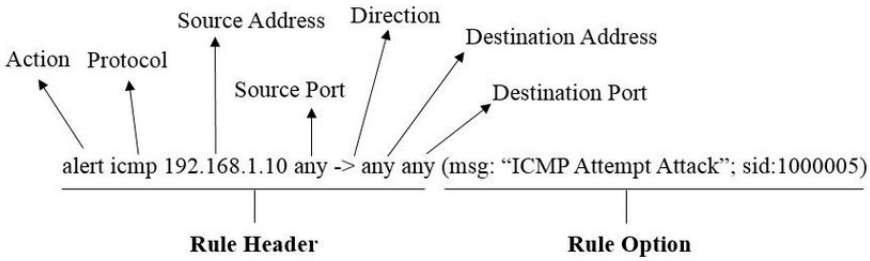

Figure 1.3: Suricata alerts are raised when the incoming traffic matches the parameters defined in this alert rule [31]

When the criteria of the rule is met, an alert is logged and is typically viewed in a Security Information and Event Management (SIEM) like Splunk or Kibana. These alerts contain specific information about the network packet such as the source/destination IP address, port numbers, timestamps, etc. A comprehensive list of all the parameters that can be recorded in an alert can be seen in [66] and we show a few key alert attributes that we will focus on in the remainder of this work in Table 1.2.

Table 1.2: A subset of the the possible parameters recorded within Suricata alerts.

Suricata Alert Parameter	Description
Alert Category	Type of activity described by the alert such as "web-application-attack" or "not-suspicious"
Alert Severity	Numeric rank (1-3) of the severity of the alert (1=highest severity, 3=lowest)
Signature Description (msg)	Text description of the behavior captured by the rule
Source & Destination IP	IP addresses of the source and destination of the packet
Source & Destination Port	Source and Destination ports of the packet
Packet Payload	The packet payload of the rule-breaking packet
Timestamp	Time at which the alert occurred

Upon initial glance it would seem that the alert category could describe the attack stage of the alert, however we found in one of our initial works that the alert category contains multiple useless

categories such as “misc” and “unknown” [44]. This is where we realized that this category was a poor representation of the attack stage and we had to turn to the signature description to get further clarification of the behavior the alert was describing. Where once again we were met with another setback; the documentation of these alert rules are poor and often even with external research it is extremely difficult to find any information about the meaning of the alert. Not only does an analyst have to deal with a high volume of alerts, its reasonable to assume that they may not be an expert using these IDS and obtaining any meaningful knowledge from the alerts may be difficult. So we ask: what is being done about this in practice?

1.3.1 Never Ending High Volume of Alerts

IDS’s producing a high volume of alerts is a well known characteristic of IDS’s [1] and it is not surprising for a SOC to record hundreds of thousands to millions of individual alerts *per day*. This is often unavoidable as the configuration of these IDS’s is typically a balance between creating a rule set that minimises the amount of false positives but not so restrictive that some observations are missed. There are techniques to tune the performance of these sensors such as proper placement of the sensor in the network, assessment for false positives, and customized rule sets. Even in their optimal configuration, certain behaviors such as scanning or brute force attempts will cause many alerts for a single action as Suricata processes network traffic at the individual packet level. This can be addressed using “alert aggregation” where similar alerts can be combined based on their similarity in attributes and further with “alert correlation” which finds other correlated alerts from other sensors to further provide evidence of an attack.

Alert aggregation describes the most basic methods to reduce the number of alerts presented to the analyst by using statistical methods to group alerts together. Zheng et al. employed clustering methods on the alert features to create “hyper alerts” [79] and time-series based methods to group alerts that occur around the same time [72, 38]. More emphasis has been put on alert correlation as external information, logs, or templates can be used to create more meaningful groupings to the alerts. Alert correlation where alerts are processed to reduce noise, extract patterns, or evaluated against other observables to help uncover the true action the attacker performed. Due to the wide adoption of IDS’s, extensive surveys and studies have been performed on the types of alert correlation types [56, 15, 57], techniques to minimize the amount of false alarms in IDS’s [25],

intrusion response systems using alert correlations [27], and multistage attack anomaly detection for advanced persistent threats (APT) [19].

Sadoddin *et al.* [56] described three categories of alert correlation algorithms: 1) Similarity-based, 2) Knowledge-based, and 3) Statistical-based. Similiarity-based methods rely on a set of rules defining the relationships between features where the rules can be simple and explicitly defined [30, 71] to self-defining feature relationships using machine learning [80] and artificial neural networks [76]. Knowledge-based algorithms uses specific context like an predetermined attack scenario to create attack graphs [55, 17] or uncovering attack scenarios using previous attacks [71, 41]. Lastly, statistically-based models focus on the statistics of each alert and develop behavioral patterns and associations with no contextual information [65]. Sadoddin *et al.* [56] admits that this classification is not encompassing and some methods may be a mixture of these three categories.

Here we find that to extract out attack campaigns using prior knowledge of an analyst we will first need to aggregate alerts that occur within a similar time frame and attack type to represent the actions performed by the adversary. Second, create a model that will represent the knowledge base of the analyst to correlate alerts together that are likely to be apart of the same attack campaign. We use the concept of statistical-based correlation approaches as our motivation to create engineered features to describe the aggregated alerts in a way they can be compared with one another. As we mentioned before, alerts have no meaningful notion of attack stage and we will need to leverage outside sources and research to obtain the attack stage given an alert.

1.3.2 Researching and Translating Alert Descriptions

Consider a Suricata IDS alert description: *'ET EXPLOIT Possible CVE-2014-3704 Drupal SQLi attempt URLENCODE 1.'* Suricata labels this as *'web-application-attack'* which provides some context but does not provide sufficient details of the potential attack impact without additional research. Additional research into the specific CVE identifier shows that the attack action involves *'Arbitrary Code Execution'* that might lead to *'Privilege Escalation*.' Providing the latter set of labels allows the analyst to assess and differentiate *what* the attacker tried to accomplish, using the intuitive and commonly referred attack campaign descriptions such as the Cyber Attack Kill Chain [11] and MITRE's ATT&CK [39] framework. The example above has the luxury of an associated CVE which is rare in the entire set of Suricata alerts (1664 out of 64k as of late-2020). One often

needs to resort to security blogs, forum posts, and other related alerts to resolve the attack impact. Once again taking up valuable analyst time and prolonging the time to respond to the threat.

Ideally the attack stage of each alert can be manually defined within the alert rule however with no agreement on the attack stage framework and that rules can be defined by anyone, this does not exist. Rule sets are defined by many entities and constantly updating meaning that we must look for a method to classify the attack stage of the alert regardless of the set of attack stages or the sensor used. As descriptions of alerts are nearly universally defined, we wonder if we can "read" the alert description and interpret it into an attack stage. Since each signature description takes a significant amount of research to discover the true impact of the alert, we also wonder if it is possible to reproduce the "research" conducted into vulnerability descriptions, blogs, and forums to classify the attack stage. This requires the understanding and interpretation of unstructured text which until the recent advancements to deep learning architectures has been extremely difficult.

To our knowledge, there exists no work that enables learning information from blogs, vulnerability descriptions, forums, *etc.* and apply that knowledge to classify data to a discrete set of labels without intervention. The closest related work involving attack action classification is MITRE TRAM [78], which uses a supervised Logistic Regression to label threat reports with MITRE ATT&CK techniques [39] and provides a user interface for administrators to define and refine labels. While TRAM does ingest unstructured data, it is only trained from the self-labeled examples and requires human intervention to correct miss-classified labels. The authors mention that due to the size of the ATT&CK framework the user may have a bias on ATT&CK techniques they are most familiar with, an aspect we consider when developing the Action-Intent Framework. Due to the lack of data customized to their application, the actual performance of TRAM is unknown and only a proof of concept is given. We see the need for not only a method to aid analysts in understanding cyber threats but also motivates us to develop a process is not constrained by the lack of labeled data.

Which leads us to our main objective which assumes we can provide meaningful reduction in alerts and be able to identify the attack stage of the alerts: "How do we now identify the attack campaigns from the IDS alerts?"

1.4 Discovery of the Attack Campaign

Throughout the 2010's the cyber security market is one of the fastest growing sectors across the world and has been accelerated into the 2020's with the reliance of remote services due to the COVID-19 pandemic [28]. Due to the extreme financial impact a successful cyber attack could have on a company, most of the contributors in this space have the same objective in mind- stop the attacker as early as possible to minimize the impact. Recall the concept of the cyber attack kill chain of "cutting the chain." The adversary has a clear advantage over the defenders as cyber attacks can come from anyone, at anytime, from anywhere in the world. The best adversaries will traverse and attack their victims as quietly as possible to remain undetected. By the time the attack is detected, typically its too late and the SOC must react quickly to minimize further damage.

In the scenario where an attack is detected (an IoC) and we could identify the complete attack campaign of the adversary from initial reconnaissance to achieving their final objective, then the response to mitigating the threat is much quicker. A detrimental challenge to research in this space however is that data describing cyber-attacks is extremely limited due to privacy reasons, rarely having any knowledge about the adversary, outdated/obsolete attack actions, and constantly changing attacker behaviors. Despite this, many researchers have been trying to identify attack campaigns within log data since the early 2000's and have been getting more sophisticated each year.

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 [52, 80, 75, 73], process-mining [13, 9], and Markovian-based approaches [17, 20] to map observables to pre-existing AG's. These approaches require a significant amount of expert knowledge to configure, create attacker scenario templates, and assumes that each vulnerability is known [4]. 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 [46] and OMMA [47] to extract context from logs, vulnerability databases like CVE and CAPEC, and analyist feedback to find malware behaviors. In 2018, we 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 [44]. This process will be explained in detail later

when we describe our HeAT work. Landauer *et al.* [32] 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.* [45] present SAGE which employs S-PDFA to extract meaningful AG's from only intrusion alerts and without prior expert knowledge. 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" [12], IBM with QRadar Advisor with Watson [26], and Centripetal with AI-Analyst [48] 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 will not position ourselves to compete with these products, their existence shows that this is a developing and more notably a valuable problem to address and document.

This leads us finally into our solution to each of these challenges with our process HeAT-PATRL. We acknowledge that the analyst's time is extremely limited and that well labeled and relevant IDS data sets are not available to create sophisticated data-driven models. We propose our methods to overcome these challenges with limited labelled data, the analyst's own self-reflection of cyber attacks, and the infinite knowledge base contained on the internet. Our work "PATRL" (Psuedo-Active Transfer Learning) translates any alert description to our own custom-defined set of attack stages known as the Action-Intent Framework (AIF). PATRL uses a language model trained on various cyber-security related text sources to enhance the classification of a small set alert descriptions, only 1% of the the total Suricata alerts are needed to classify the remaining. Then we further extract out meaningful knowledge from our language model using the novel self-learning process of Psuedo-Active Learning, creating a robust and accurate method to translate alerts into an attack stage. The PATRL process system architecture is shown in Figure 1.4

Then lastly our process for extracting attack campaigns from IDS data, HeATed Alert Triage (HeAT), focuses on capturing the analyst opinion of critical relationships between alerts within the

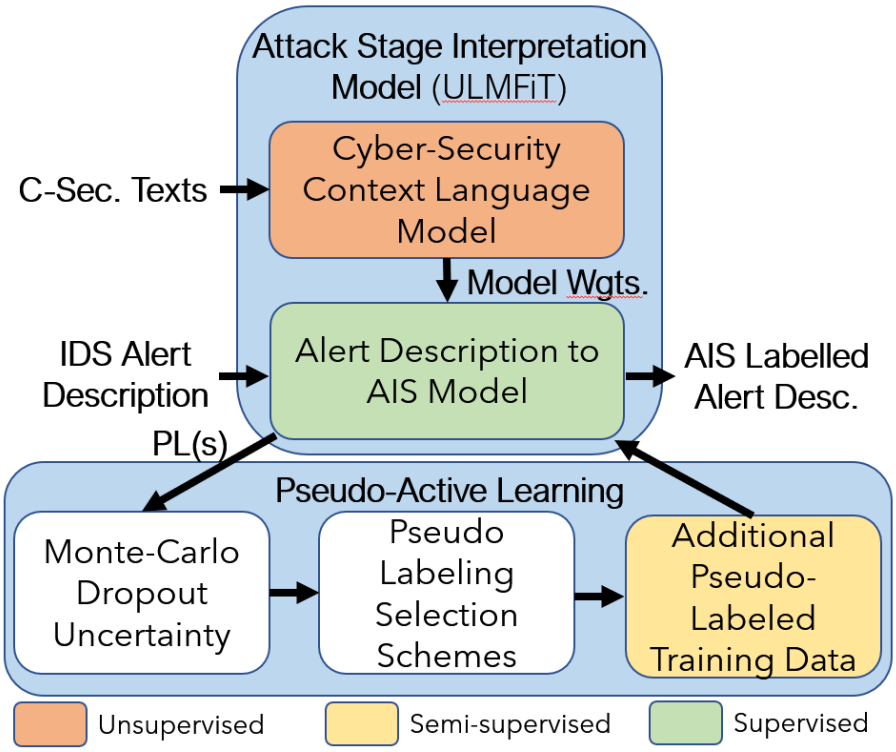

Figure 1.4: System architecture for PATRL: Users input an alert description (as text) and will translate the description to the appropriate attack stage in the AIF

same attack campaign and using that knowledge to reveal other attack campaigns. This is done by having the analyst conduct a short triage with an known IoC, label significant attack campaign contributing events, and use network-agnostic features on aggregated alerts called alert episodes, so that observations of other attacks can classify others. We demonstrate through the use of our entropy-based metric called "HeAT Gain" where we find that triages from cyber-competitions (easy to understand and find) can find meaningful attack campaigns within real-world SOC operations with minimal additional data. The overall system architecture of HeAT is displayed below in Figure 1.5.

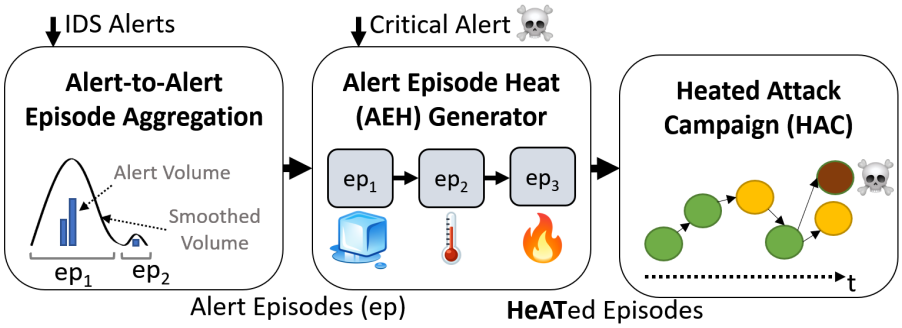

Figure 1.5: System architecture for HeAT: Extracts the Heated Attack Campaign based of the analyst's knowledge captured from prior triages.

To build up to extracting attack campaigns using HeAT, we first need to determine the attack stage of the IDS alerts given the signature description using PATRL. Suricata IDS alerts can only capture actions that generates network traffic which restricts the number of possible attack types that we can capture. To address this, we now define the Action-Intent Framework (AIF) to focus on determining the intentions of the adversary through IDS alerts.

This dissertation is organized as follows: Chapter 2 defines the Action-Intent Framework to develop a set of IDS-observable attack stages. Then in Chapter 3 we use the AIF in PATRL where we translate IDS alert descriptions into one of the micro-AIS. In Chapter 4 we describe our process called HeAT which leverages PATRL and prior campaign triages to extract out "HeATed Attack Campaigns" within real-world SOC operations. Lastly we conclude this dissertation with final remarks in Chapter 5.

2. Action-Intent Framework (AIF): Action-Intent Stages (AIS) to Describe Attack Campaign Impacts

Cyber-attacks are characteristically secretive, anonymous, and often illegal. Often the result of cyber-attacks reveals highly sensitive data and other lucrative information and many try to understand how the attack happened and what can be done to stop it. Cyber-attacks can literally come from anywhere and it is extremely unlikely that you will ever know the person on the other end or their true intentions. We are able to often detect their presence and some attributes about their actions through an IDS, which is our best opportunity to observe and then describe the attack campaign. Through classifying the actions performed by an adversary by their intentions, we believe we can describe and compare attack campaigns as specific behaviors.

We turn to the most common and abundant sources of attacker's actions, IDS logs, where network traffic passing though a sensor is analyzed against known signatures of specific behaviors raising alerts to the administrators when suspicious actions are performed on the network. Sophisticated attack action frameworks like MITRE ATT&CK are designed to describe attack campaigns. We realized that the current evolution of IDS's are not yet able to resolve to the highly specific and detailed ATT&CK techniques. We also believe that it is important to understand the specific type of exploit used in an attack campaign, we also believe we should know *why* the attacker would perform an action. This could be considered as describing the "intent" of each action as opposed to specifically the type of exploit used by the adversary, which could be contained within the IDS alert description. We propose the Action-Intent Framework (AIF) as set of Action-Intent States (AIS) that is designed to resolve the intentions of the actions performed by the adversary when observed within an attack campaign. We define the concept of Macro-AIS as a high level description of the outcome of the action such as privilege escalation or data destruction. The Macro-AIS is similar

to the higher-level tactics described of ATT&CK, including reconnaissance stages and zero-day attack stages. For each Macro-AIS, we define a set of Micro-AIS's that describe 'how' the adversary achieves the Macro-AIS. The Micro-AIS's are similar to the techniques defined by ATT&CK but with a focus on action-types that are observable by an IDS and not specific to any service, operating system, or network configuration.

2.1 Action-Intent State Selection Methodology

Taking inspiration from MITRE ATT&CK where the "techniques" are classified under a high-level description called a "tactic", we employ a similar two-tier structure with a more restrictive and specialized criteria to better capture the intent of the action performed. Our Macro AIS definitions contains commonalities to the ATT&CK Tactics but the Macro AIS focuses on resolving action-intents from the defense's perspective where tactics such as "persistence" or "defense evasion" describes intentions that may not be possible to definitively determine with just IDS logs. The Micro AIS definition also has similarities to the ATT&CK Techniques however a key difference is that the Micro AIS does not include techniques that are specific to any one system such as Kerberoasting or very specialized techniques such as exfiltration though "Audio Capture". With these restrictions, we propose a framework to define a set of AIS so that a sequence of AIS derived from IDS observables describes the type of actions specific attacker takes that is general to a network configuration. Our objectives for this section are as follows:

- Define the "guiding principals" for each the macro and micro action-intent stages to define if a stage/attack-type should be included in the AIF,
- apply the guiding principals given other attack-stage frameworks to define the action-intent stages applicable to defining attack campaigns from IDS alerts, and
- discuss the different types of micro-AIS defined and how they are used in practice.

2.1.1 Macro Action-Intent States

Table 2.1 contains the currently defined Macro AIS and their descriptions. Each of these macro-AIS describes a high level description of "what" the attacker has performed but does not describe a

specific method of to achieve the state. This is an important distinction as many actions performed by an aversary may have the same impact/objective but the means to achieve a macro-AIS may be different given an attack campaign. Our guiding principals for defining an Macro AIS is as follows:

1. The stage describes the impact or end goal of the action,
2. the stage does not describe a specific means of achieving a goal, and
    - For example, there are multiple methods to achieve "privilege escalation" which has a specific impact.
3. if an action type has technical or behavioral properties in-which other macro stages do not accurately describe the action's outcome due to different means of observability
    - "Active reconnaissance" actions (e.g. network scanning) typically can be observed using an IDS where "passive reconnaissance" like social engineering cannot be observed by traditional technical methods and are used in different situations.

States such as passive and active reconnaissance describe actions where the primary goal is to gather information about the target whether its through publically accessible means (passive) or technical approaches using scanning tools for example (active). Privilege Escalation, Targeted Exploits, Ensure Access, and Zero-Day describe some sort of exploitation of the target to allow the attacker to gain access or ensure access to the target. Although Zero-days are by nature difficult to detect, these types of actions are included as the usage of zero days is a critical differentiator of attacker behaviors and certain types of sensors like anomaly detectors do have the capability to observe zero-days. States such as Disrupt, Distort, Destroy, Disclosure, and Delivery describes a specific end impact that an attacker may perform as either a sub-goal or end-goal of the overall attack. For example attackers may choose to disrupt specific machines as they are traversing the network to draw attention away from the action where crucial information such as customer data is disclosed to the attacker. Under each Macro AIS will be a set of Micro AIS which will describe "how" the attacker chose to achieve the behavior described in as Micro AIS in the following section.

Table 2.1: The Macro Action-Intent States currently defined in the framework.

Macro AIS	Description
Passive Recon	An attempt to gain information about targeted computers and networks without actively engaging with the systems
Active Recon	An intruder engages with the targeted system to gather information about vulnerabilities
Privilege Escalation.	Act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access
Targeted Exploits	Exploits targeting a specific service, application, organizational entity, or person
Ensure Access	Actions which expand preexisting access or circumvent active defense strategies
Zero Day	Actions performed employing undocumented vulnerabilities or strategies with unknown consequences where no patch exists at the time of the attack
Disrupt	Disruption in services, usually from a Denial of Service.
Destroy	Destruction of information, usually when an attack has caused a deletion of files or removal of access.
Distort	Distortion in information, usually when an attack has caused a modification of a file.
Disclosure	Disclosure of information, usually providing an attacker with a view of information they would normally not have access to
Delivery	Actions where the intent is to place/install/deliver data that could be in the form of malware, backdoor, application, etc.

2.1.2 Micro Action-Intent States

The Micro AIS is similar to the MITRE Att&ck techniques but with the key difference is that we do not include techniques that are specific to any one service, operating system, or network. In the case of privilege escalation, MITRE Att&ck defines techniques like “Sudo” and “Bypass User Account Control” which are legitimate techniques to gain root access to Linux and Windows machines respectively however the choice of choosing these one of these techniques is situational depending on the network. The true intention of these techniques was to gain administrative access to the machine regardless of the network, thus our methodology is to combine these two techniques based on the intent creating the Micro Action-Intent state of “Root Privilege Escalation” and also “User Privilege Escalation”. These are two states with specific impacts to the target, used in different situations, and describes the intentions of the adversary without requiring information about the target network. Our defined micro-AIS for reconnaissance, exploitation, and objective-based attack stages is described in Tables 2.2, 2.3, 2.4 respectively. Our guiding principals for selecting new

states for the Micro AIS are defined below:

1. The state describes a specific and unique means of achieving an **macro** AIS,
    - Network sniffing credential access and brute force credential access can privileges, however they are used in different situations and are observed differently
2. the state is service and platform agnostic,
3. the state has a well defined impact on a type of target or yields different response from the target, and
    - For example, "End-point DoS" is used to target individual services and "Network DoS" targets an entire network, leading to a substantially different impact
    - Attackers may use different types of reconnaissance actions to reveal different characteristics of the network, services, or vulnerabilities of a network
4. if the state has observable characteristics that differentiates itself from other stages within the same macro attack stage.
    - The micro states of "End-Point DoS" and "Service Stop" both disrupt the function of a single machine however end-point DoS implies disruption by exhausting system resources. Service stop involves directly terminating the process.

Given these design principals we now define the micro-AIS so that each stage clearly defines each sub-objective of an attack campaign. Following the traditional definition of the stages of the Cyber Attack Kill Chain, we first follow our principals to define the micro-AIS for reconnaissance-type actions. Our reconnaissance-type micro-AIS is defined in Table 2.2, where we have two macro-AIS for reconnaissance for passive and active types of reconnaissance. Passive recon can be extremely challenging to observe from a network perspective as its intent is to appear as normal as possible, however the impact of these actions can be immense. Where as active recon describes actions who inherently creates network traffic for the adversary to learn details about the target network infrastructure. Distinguishing the level of active recon from host, to service, to vulnerability is so that we can track the progress of the adversary as they learn more about the network. It is

unlikely that a network-based IDS would be able to detect passive recon however it may be possible for a host-based IDS to detect the network response of passive actions. This makes it worth while to differentiate these behaviors as a stage in an attack campaign in the event there is evidence to make that distinction.

Table 2.2: The reconnaissance-based AIS defined describing different methods adversaries may use to obtain pertinent information about the target.

Macro AIS	**Micro AIS**	Description
Passive Recon	Target Identification	Determining the organization/network target
	Surfing	Using legitimate methods (websites, public documents, etc) to obtain information about the target
	Social Engineering	Non-technical strategy cyber attackers use that relies heavily on human interaction and often involves tricking people into breaking standard security practices.
Active Recon	Host Discovery	Use of technical programs to uncover the location/IP of machines in the target network
	Service Discovery	Use of technical programs to uncover the services or applications deployed on a machine
	Vulnerability Discovery	Techniques or programs to uncover vulnerabilities on machine with a specific application or OS

The usage of exploits, vulnerabilities, password cracking, and etc. is typically what most people think of when they think of an attack campaign. Given the size and detail of MTIRE ATT&CK, there are many techniques, tools, methods, and exploits one can use to gain unauthorized access to a target. The exploitation-type micro-AIS is our largest group of stages as this is our best opportunity to directly observe the effects (in the form of sensor logs) at a granularity that allows us to characterize many types of attack campaigns/behaviors. Our exploitation-type micro-AIS is shown in Table 2.3.

Each of these exploitation-type mirco-AIS have distinct characteristics and impact that an adversary under specific circumstances will choose to conduct that type of action. For example if we discovered in our attack campaign that an adversary used “Trusted Organization Exploits”, we would have to conduct mitigation strategies differently as the attack would appear from coming from a trusted company. Whereas other adversaries may find other opportunities to exploit a network and possibly use new strategies such as our “zero-day” stages. Like the passive recon cases, zero-days are obviously extremely difficult to observe, however if there is evidence of an zero-day it is still crucial to be able to distinguish that in our attack campaigns. This level of granularity but

Table 2.3: The exploitation-based AIS to describe the different methods an adversary may choose to obtain access to the network or internal machines including zero-day stages if observed.

Macro AIS	**Micro AIS**	Description
Privilege Esc.	User Privilege Esc.	Action which results in the adversary gaining user privileges
	Root Privilege Esc.	Action which results in the adversary gaining root/admin privileges
	Network-Sniffing Credential Access	Using the network interface on a system to monitor or capture information sent over a wired or wireless connection
	Brute-Force Credential Access	Brute force techniques to attempt access to accounts when passwords are unknown or when password hashes are obtained
	Account Manipulation	Modifying permissions, modifying credentials, adding or changing permission groups, modifying account settings, or modifying how authentication is performed
Targeted Exploits	Trusted Organization Exploitation	Access through trusted third party relationship exploits an existing connection that may not be protected or receives less scrutiny than standard mechanisms of gaining access to a network
	Exploit Public Facing Application	Use of software, data, or commands to take advantage of a weakness in an Internet-facing computer system or program in order to cause unintended or unanticipated behavior
	Exploit Remote Services	Exploitation of remote services such as VPNs, Citrix, and other access mechanisms allow users to connect to internal enterprise network resources from external locations
	Spearphishing	An email spoofing attack that targets a specific organization or individual, seeking unauthorized access to sensitive information
	Service-Specific Exploitation	Use of a vulnerability specific to a system OS, application, and version
Ensure Access	Defense Evasion	Techniques an adversary may use to evade detection or avoid other defenses
	Command & Control (C2)	Control over an target by establishing a communication channel between adversary and target
	Lateral Movement	Techniques that enable an adversary to access and control remote systems on a network and could include execution of tools on remote systems
Zero Day	Privilege Esc.	Undocumented action that raises the privilege level of the adversary
	Targeted Exploit	Usage of a unpatched and possibly undocumented targeted exploit
	Ensure Access	Unknown method to evade detection or controlling method

straightforward intuitive stages will enable us to define diverse and meaningful attack campaigns for the types of exploits used by the adversary. These stages are typically used as a means gain access to victims with the intent to eventually achieve some overall objective. The AIF also defines a set of objective-based AIS, shown in Table 2.4.

Table 2.4: The goal/objective-based AIS to describe adversarial actions with a specific end objective and leads significant impact to the victim.

Macro AIS	**Micro AIS**	Description
Disrupt	End Point DoS	Exhausting the system resources those services are hosted on or exploiting the system to cause a persistent crash condition
	Network DoS	Exhaust the network bandwidth services rely on
	Service Stop	Stop or disable services on a system to render those services unavailable to legitimate users
	Resource Hijacking	Leverage the resources of co-opted systems in order to solve resource intensive problems which may impact system and/or hosted service availability
Destroy	Data Destruction	Destroy data and files on specific systems or in large numbers on a network to interrupt availability to systems, services, and network resources
	Content Wipe	Erase the contents of storage devices on specific systems as well as large numbers of systems in a network to interrupt availability to system and network resources
Distort	Data Encryption	Encrypt data on target systems or on large numbers of systems in a network to interrupt availability to system and network resources
	Defacement	Modify visual content available internally or externally to an enterprise network.
	Data Manipulation	Insert, delete, or manipulate data at rest in order to manipulate external outcomes or hide activity.
Disclosure	Data Exfiltration	Techniques and attributes that result or aid in the adversary removing files and information from a target network

These objective-based micro-AIS more describe *why* the adversary is conducting their attack and we want to investigate how the attacker got to these stages within their attack campaign. Out of all the AIS defined, these are most likely to be the most impactful if observed. Once we get to the point that we can begin defining IoC's to assess for attack campaigns, we will prioritize these objective-based AIS alerts as we would expect these to be the end of a complete attack campaign. Looking back from these type of observations will enable us to describe attack campaigns from their initial reconnaissance stages, to initial exploitation/access, leading towards the achievement of impactful objectives.

2.2 Applications of the AIF

Initially our need to develop the AIF came from our realization in our previous work [44] that the attack category defined by Suricata is a poor representation of the actual attack stages as described by the alert description. We intended to use the AIF as the basis behind the definition of our "Attack Behavior Model" in our cyber attack simulator "CASCADES" [43] to simulate the impact and progress made given an attack campaign. The definition of the attack campaigns to be simulated could be either manually defined or, in our case, extracted automatically from attack observations (IDS alerts for example). Drasar et al. uses the AIF to manually define an attack campaign given a detailed description of a real-world advanced persistent threat known as "Bronze Butler" and simulate the potential paths this adversary could take through the network. Drasar et al. mentions that the AIF was selected as it enabled them to quickly define attack campaigns using either the macro or micro AIS without the need for detailed exploit definitions or network attributes [14]. Nadeem et al. describes their process, SAGE, to extract AIF-based attack campaigns without the need for prior knowledge/templates by generating attack graphs using a method called S-PDFA to bring infrequent severe alerts into the spotlight and summarizes paths leading to them. [45].

For SAGE to use the AIF to extract out attack campaigns, a pre-defined mapping between the AIF and alert signature descriptions was needed. As the datasets used in previous work only contained a couple hundred unique alert types, manually mapping these alert descriptions to the AIF was not a huge burden. It was when we wanted to generalize these processes to any network did we encounter data with significantly different unique alerts, sometimes from external alert-rule databases, did we find manually labelling challenging and time-consuming. Documentation of many alerts is poor and sometimes non-existent. We had to use our best judgement of attack stage given the alert description and the specific terminology contained to infer on the stage based upon similar and better documented alert descriptions. Given our objective of extracting attack campaigns from IDS alerts across any network, we must address that there will be differences between the defined alert-rules and our methodology relies on the realization of attack stage given a IDS alert. In the next chapter, we demonstrate our process, PATRL, to translate IDS alert descriptions into a corresponding AIS leveraging unsupervised cyber-text learning, transfer learning, and a novel application of "Psuedo-Active Learning." Once we can decipher the attack stage of each alert using PATRL, we use the attack stages in our final work HeAT as an attribute to aggregate alerts with similar impact

called “alert episodes”.

3. PATRL: Interpretation of Alert Signatures With Pseudo-Active TRansfer Learning

Our first technical contribution in this dissertation, PATRL, uses multiple machine learning techniques to interpret IDS alert signature descriptions into one of the various micro-AIS's defined in the previous chapter. Understanding the corresponding attack stage of an alert gives us additional context into what the attacker could have achieved when an alert is observed and the ability correlate attributes between similar alerts from the same action or even alerts related to the overall attack campaign. Due to the complexity of the alert descriptions, the poor documentation describing the true meaning of the alert, and the lack of standardization rule definitions (no standard attack stage framework with specific criteria), we design PATRL to not rely on a high volume of labelled data but instead we take advantage of unsupervised and semi-supervised ML/AI techniques to address these challenges. In this chapter, we describe our process of "Pseudo-Active Transfer Learning" for attack stage interpretation and demonstrate how we can achieve high accuracy of a cyber-focused natural language processing classification problem with limited labelled data. Our technical abstract for PATRL is as follows:

Intrusion alerts continue to grow in volume, variety, and complexity. Its cryptic nature requires substantial time and expertise to interpret the intended consequence of observed malicious actions. To assist security analysts in effectively diagnosing what alerts mean, this work develops a novel machine learning approach that translates alert descriptions to intuitively interpretable Action-Intent-Stages (AIS) with only 1% labeled data. We combine transfer learning, active learning, and pseudo labels and develop the Pseudo-Active Transfer Learning (PATRL) process. The PATRL process begins with an unsupervised-trained language model using MITRE ATT&CK, CVE, and IDS alert descriptions. The language model feeds to an LSTM classifier to train with 1% labeled data and is further enhanced with active learning using pseudo labels predicted by the iteratively improved models. Our results suggest PATRL can predict correctly for 85% (top-1 label) and 99% (top-3 labels) of the remaining 99% unknown data. Recognizing the need to build confidence for the

analysts to use the model, the system provides Monte-Carlo Dropout Uncertainty and Pseudo-Label Convergence Score for each of the predicted alerts. These metrics give the analyst insights to determine whether to directly trust the top-1 or top-3 predictions and whether additional pseudo labels are needed. Our approach overcomes a rarely tackled research problem where minimal amounts of labeled data do not reflect the truly unlabeled data's characteristics. Combining the advantages of transfer learning, active learning, and pseudo labels, the PATRL process translates the complex intrusion alert description for the analysts with confidence.

3.1 Introduction

Security practitioners use various intrusion detection systems (IDS) to capture suspicious and/or malicious activity on a network to prevent current and future attacks. The sensors' output, *i.e.* raised alerts, are meant to provide contextual information for preventive or remedial actions to alleviate financial and/or operational damages. In the ever-evolving landscape of hacker sophistication and networked systems, interpreting these alerts requires substantial expertise and research into vulnerability databases and security blogs. This leads to a potentially lengthy process to determine the meaning of the alerts and the results are often ambiguous due to the lack of documentation when alert rules are created. Compounded with the endless stream of alerts, this time-consuming process makes it difficult for security analysts to effectively assess and differentiate the intended consequences of observed malicious actions.

Recall from earlier the Suricata IDS alert description: *'ET EXPLOIT Possible CVE-2014-3704 Drupal SQLi attempt URLENCODE 1.'* Additional research into the specific CVE identifier shows that the attack action corresponds to the attack stage *"Arbitrary Code Execution"* which may that lead to *"Privilege Escalation*." We came to this conclusion from the description containing an associated CVE, this only is present in 1664 out of 64k rules as of late-2020. One often needs to resort to security blogs, forum posts, and other related alerts to resolve the attack impact. This work aims to alleviate the time-consuming process for security analysts to manually interpret complex alert descriptions. This work develops a novel machine learning process that maps the 64K+ Suricata alerts to a small set of Action-Intent-Stages with only $\sim$1% of expert labeled alert descriptions.

There is a number of challenges to automatically translate the large number and variety of

cryptic alert descriptions to a small number of AIS in a real-world setting. First, one needs to extract the information contained within CVE descriptions, cybersecurity forums, and other unstructured texts. Second, we can only assume the analysts are familiar with a small percentage ($\sim$1%) of alert descriptions and thus have the expertise to label them for machine to learn. Third, the system needs to provide a means for the analysts to evaluate the uncertainty associated with machine predicted AISs and thus have the opportunity to correct the prediction if needed.

To combat these challenges, we first leverage advances in LSTM-based Transfer Learning, specifically ULMFiT [24], to learn the structure and cyber-specific contexts of the unstructured cyber-security texts. The learned model yields significant interpretation accuracy for the initial small set of familiar alerts with minimal intervention and makes non-obvious inferences when classifying unseen alert descriptions. To expand the effectiveness of the machine learned model for the unlabeled alert descriptions, we build upon the idea of Active Learning. Active Learning is an iterative process where in each iteration an expert is asked to label one or few low confidence predictions as suggested by the system to further add to the training of the model for the next iteration. In our setting, we do not assume that the security analysts will have the expertise to continue label unfamiliar alert descriptions. Therefore, we propose a novel approach called Pseudo Active Learning, where the machine predicted labels are used in place of the expert labels. This approach is motivated by the concept of 'Pseudo-Labeling' [34], a semi-supervised approach that labels new data samples as training data to lower the prediction error. We experiment several options on how to select the machine predicted labels to effective enhance the accuracy for the unlabeled alert descriptions. The combination of Transfer Learning and Pseudo Active Learning gives the proposed integrated process - *Pseudo-Active Transfer Learning (PATRL)*. PATRL is able to not only predict AIS for 64K+ alert descriptions with only extremely small percentage of labeled data, but also a measure of Monte-Carlo Dropout Uncertainty (MCDU) [18] to provide analysts a means to assess the confidence of the predictions. This allows the analysts to make necessary corrections of the machine predicted labels when needed, which is an essential function to provide confidence for human analysts.

The major contributions are summarized as follows:

1. Demonstrate how cyber-relevant unstructured texts can be used to help translate complex IDS alert descriptions into Action-Intent-Stages (AIS),

2. propose the concept of 'Pseudo-Active Learning,' where new training labels are iteratively created by predicting previously unknown, unlabeled alert descriptions as Pseudo-labels, and

3. demonstrate how Pseudo-Active Learning corrects for differences in feature and label distributions between known and unknown data.

The remainder of this paper is organized as follows: Sec. 3.2 discusses related works. Sec. 3.3.1 gives the methodology of the transfer learning and language modeling portion of this work. Sec. 3.3.2 defines the concept of PATRL using MCDU. Sec. 3.4 defines the design of experiments and data used in this work. Sec. 3.5 contains the results of our experiments. Lastly, we conclude the paper in Sec. 3.6.

3.2 Related Works Addressing Limited Data

Many research fields have issues with obtaining useful labelled data, however the complexity, ambiguity, and sensitivity of cyber-attack data means that obtaining the massive amounts of labelled data that deep-learning architectures require is unlikely. A simple method to account for the lack of data is to label more data, but do we select any data to label? Selecting *any* new data to be labelled may introduce negatively impact performance as the additional labeled data introduce bias if the new data favors a particular label [78] while also being extremely time consuming. Active learning is a method used to overcome these labeling challenges by assessing the model's confidence in its label prediction and recommends the user to label the least confident samples. Given that the prediction confidence is self-reported by the model, the active learning technique has been automated. Many works follow a similar approach to the original work of Yarowsky *et al.* [77], where a weak model is used to classify unseen samples, and the uncertainty of the prediction is used to determine if the sample should be used to retrain a new model. High confidence predictions are typically added to the model, and low confidence predictions are given to the domain expert to be labeled as traditional active learning [70]. Using the uncertainty to select the additional labels is known as uncertainty sampling [35]; we use the Monte Carlo Dropout Uncertainty (MCDU) to assess uncertainty in deep learning architectures. Usage of semi-supervised active learning is limited and has been applied to situations where unlabeled data characteristics are known [22, 8]. Calma *et al.* mention that this semi-supervised active learning approach can be used in label imbalance situations between the

known labeled data and the unlabeled data, a characteristic that challenges this work [8]. There is a need to address situations where the distribution of unlabeled data is largely unknown and assess the model's performance for this unknown, unlabeled data.

An approach to account for the lack of data is to leverage the potentially infinite amount of knowledge within unstructured texts. Inherently these sources are unlabeled and it is difficult to consistently extract relevant information out of these sources as they typically involved unsupervised or semi-supervised extraction processes. We find methods using keyword-based approaches to classify cyber-threats [53, 33] and various rule-based approaches to provide IDS-like alert reporting [40, 5]. Before the age of sophisticated deep learning architectures, Joshi *et al.* [29] used a conditional random forest (CDF) to link the information contained in CVE to NVD, compiling linked data as Resource Description Framework (RDF) with little human intervention. As of 2020, the significance of deep learning approaches to the cyber-security field is growing fast however few works focus on extracting information from unstructured sources. Notably, Long *et al.* uses a multi-head self-attention model along with contextual features to report potential indicators of compromise (IOC) [37]. The authors mention that other works require extreme cyber security knowledge and are difficult to configure/maintain and the contextual features allow of IOC's to be defined without a deep cyber security background. PATRL's learning architecture is selected so that the contextual features between alert descriptions and attack stages is realized by the model and Pseudo-Active learning accounts for label imbalance or user inexperience.

3.3 PATRL: Theory and Architecture

The PATRL process comprises two core components: the Cyber-text Interpretation module with transfer learning of cybersecurity language, and the Pseudo-Active Learning module that self-labels alert descriptions for model improvement. The two modules feed to the model that translate alert descriptions to Action-Intent-Stages (AIS), a MITRE ATT&CK derived attack stage framework. Figure 3.1 shows the overall system architecture. We will discuss the two modules and the overall system below.

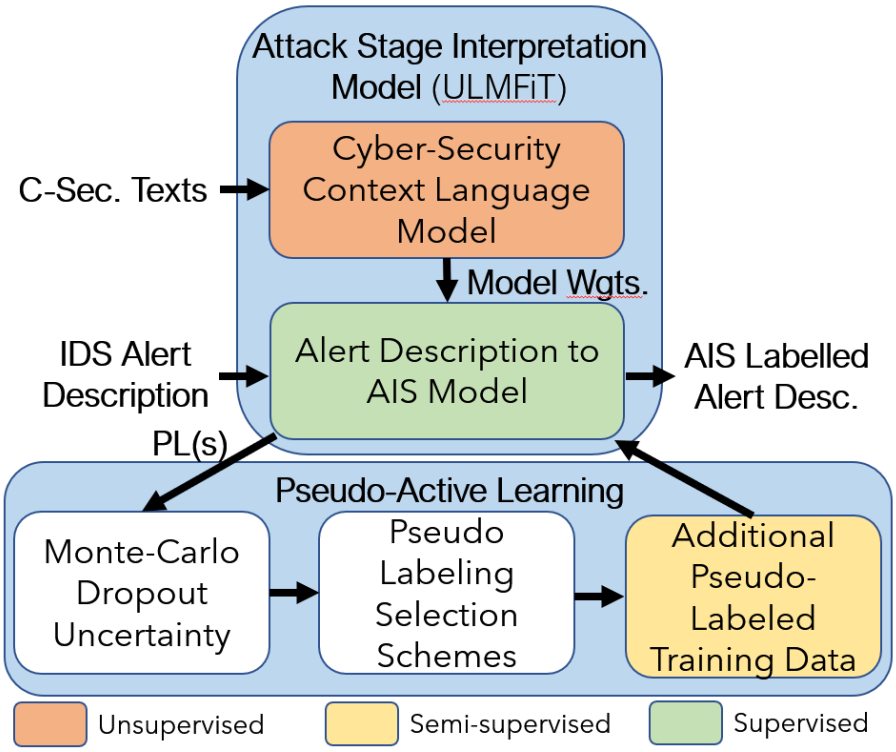

Figure 3.1: The PATRL system architecture: combining supervised, semi-supervised, and unsupervised learning to translate alert descriptions to Action-Intent Stages (AIS).

3.3.1 Attack Stage Interpretation Model via Transfer Learning

We define two challenges in developing a classifier that translates IDS alert descriptions to attack stages: 1) understanding and processing natural language (i.e., a language model), and 2) comprehending and classifying texts into an attack stage framework. The first challenge represents our objective of learning from cyber-security texts. We believe that a language model trained on cyber-specific texts can learn specific patterns and terminologies which can be related to the attacker's attack stages. Advancements in transfer learning have enabled information (i.e., model weights) from one task to be applied to another task that may or may not be related. We propose that a language model using cyber-security texts will help resolve some of the cyber-specific nuances by recognizing (in the form of neuron activation) certain patterns or keywords for specific attack stages. The formal definition of transfer learning with respect to our objective is as follows.

Pan et al. [49] give the formal definition for transfer learning as: a domain $D = \{\Lambda, P(X)\}$ defined as a two-element tuple consisting of the feature space Λ, a sample data point $X = \{x_1, \ldots, x_n\}, x_i \in \Lambda$, and the marginal probability $P(X)$ [49]. A task $T = \{\Gamma, P(Y|X)\} = \{\Gamma, \eta\}$ where Γ represents the label space, the labels $Y = \{y_1, \ldots, y_n\}$, $y_i \in \Gamma$, η is a predictive function learned through feature vector/label pairs $(x_i, y_i), x_i \in \chi, y_i \in \Gamma$ and η predicts labels such that $\eta(x_i) = y_i$. Given the source and target domains D_s and D_t and the source and target tasks T_s and T_t, respectively, the objective of transfer learning is to optimize the distribution $P(Y_t|X_t)$ in D_t with domain knowledge within D_s and T_s where $D_s \neq D_t$ or $T_s \neq T_t$. The source domain D_s and target domain D_t are shared and defined as the English language text sources where the source task T_s is to model the language structure and semantics of the source data that we provide. We use the ULMFiT deep learning architecture [24] to create a language model of cyber security texts to first learn these relationships unsupervised as our source task. We propose the semantical relationships between certain words and combinations of words can be exploited and related to the target task T_s of interpreting IDS alerts to an attack stage. The selection of the attack stage framework for the target classification task dictates the overall utility of this work.

We select our attack stage framework based upon our evaluation of the framework's ability to: describe distinct actions leading to unique outcomes, represent diverse attack actions, and, most importantly, describe actions that are observable to our IDS of choice. We consider the use of

various Cyber Attack Kill Chains [11, 67], MITRE ATT&CK framework [39], and the Action-Intent Framework [42] to label IDS alerts with. Cyber Attack Kill Chains provide an intuitive and is easy to interpret of the objective of an attacker, however in general we find kill chains to be too high level to effectively describe the specific actions contained in IDS alerts. Conversely, the MITRE ATT&CK framework contains nearly 300 techniques describing general behavior such as "Active Scanning" to service-specific techniques such as "Forge Kerberos Tickets" A significant burden would be put on the user to label data with the most appropriate label without bias to a more well understood technique [78]. This could lead to many techniques being underrepresented in our dataset, especially with our proposed limited data.

Instead, this work adopts the Action-Intent Framework (AIF) [42] to balance the intuitive nature of kill chains and the detail of MITRE ATT&CK. The Action-Intent Stages (AIS) within the AIF are similar to the ATT&CK's tactics and techniques but focus on defining behaviors to describe the intended outcomes and consequences of the attack actions observed through IDS. The AIF contains general descriptions of various types of reconnaissance such as network, service and vulnerability discovery to aid in distinguishing between key moments when specific information is obtained. The same methodology is applied to the exploitation and exfiltration phases of the attack where critical access behaviors such as "arbitrary code execution" is distinguished from "brute force credential access" since these techniques are typically conducted in different situations. When applied to IDS alerts, we propose that the AIS provides quick additional context to alerts that is useful to quickly hunt threats and further assess the progress of the attacker.

Our proposed architectural diagram for the attack stage interpretation model is shown in Figure 3.2. Given an IDS alert description in the form of an sentence, this model predicts the appropriate AIS given the a labeled alert description set and the transferred information from the language model. The different types of texts used to train the language model(s), and the AIS labeled data distributions are located in Section 3.4.

We expect language models trained with little cyber-security relevancy to perform significantly worse that those trained with rich cyber-security information. We are realistic in our expectations of the additional performance this language model can provide given that our labeled data is extremely limited when compared to all of the "unknown" IDS alerts. Consequentially, this model may not be sufficiently accurate to represent the unknown data (*e.g.* over-fitting to the labeled data). We next

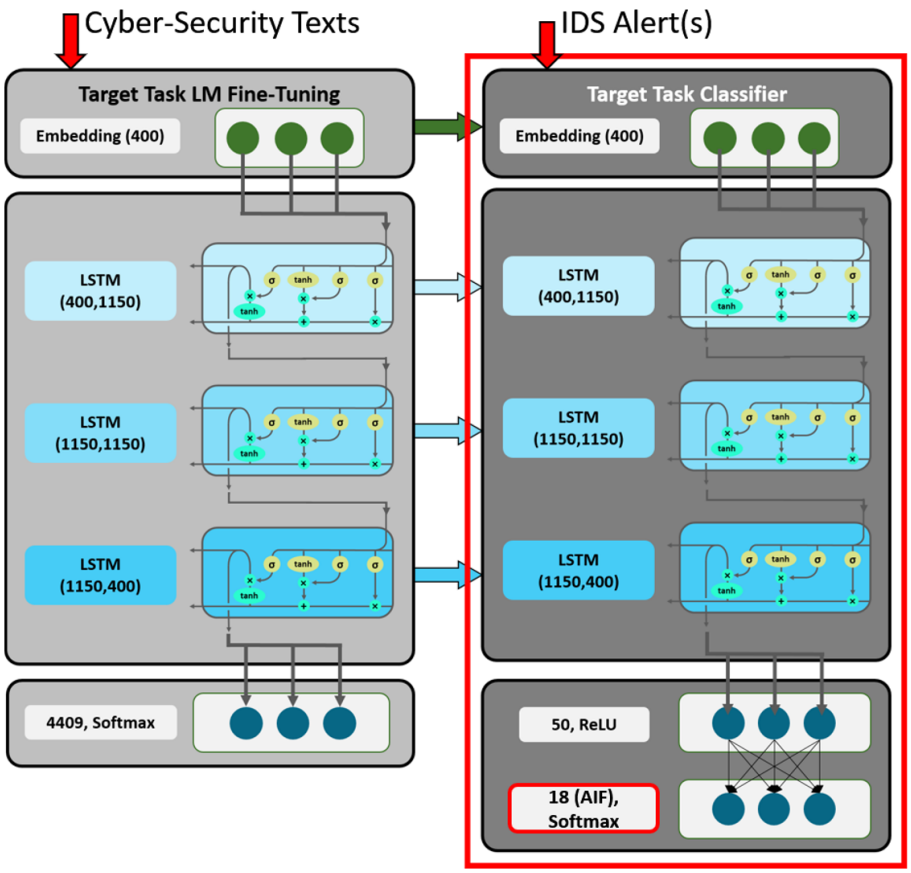

Figure 3.2: The use of ULMFiT [24] to create the initial model that translate alert descriptions to AIS labels.

define our process for improving this model by applying "Pseudo-Active Learning" to account for the disparities between the labeled data set and the unknown data.

3.3.2 Pseudo-Active Learning for Unknown Data

Our initial labeled set of alert descriptions only represents those captured on a specific network configuration and time frame; it is unlikely that this small labeled set accurately represents the characteristics of the remaining set of sophisticated and unknown alert descriptions. To combat this, the model can be refined by including additional labeled data where the model has reported uncertainty with its prediction, known as Active Learning[61]. We propose leveraging the concept of Pseudo-Labeling [34] combined with Monte Carlo Dropout Uncertainty (MCDU) [18] to estimate a traditional active learning process. We begin this process by using the MCDU metric to assess the prediction confidence of the unlabeled data and choose new pseudo-labels based upon the uncertainty metric.

Monte Carlo Dropout Uncertainty (MCDU)

The concept of Active Learning relies on the model's ability to assess the confidence or certainty of its predictions and recommend the user to apply labels to the data where the model reports low confidence. Deep learning architectures are challenging to determine its certainty as the model itself is not easily interpretable [58] and the output class probabilities are not a reliable method for assessing the uncertainty [6]. Gal *et al.* propose that the concept of "Dropout" (i.e., eliminating neurons at training time to improve generalizability) can be used at testing time to approximate a Bayesian approximation of the model using class probabilities for different "configurations" of the model [18]. The model is given the input multiple times for a single sample. Each iteration randomly drops out a percentage of the neurons with the uncertainty given as the standard deviation of the class with the highest mean probability across the dropout iterations.

The formal definition for determining the MCDU is as follows, and the implementation in Keras is shown in [62]. Given our transfer learning model as f_{nn}, a data sample as x, and the number of dropout iterations as T, the model with dropout configuration of d_i is given as $f_{nn}^{d_i}$. The ensemble of dropout models to assess the uncertainty is given as: $f_{nn}^{d_0}(x), ..., f_{nn}^{d_T}(x)$. The mean of the probabilities for each class across each of the dropout iterations represents the ensemble's prediction as

opposed to the single prediction if no dropout was applied. The predictive posterior mean for the ensemble is given as:

$$p = \frac{1}{T}\sum_{i=0}^{T} f_{nn}^{d_i}(x) \tag{3.1}$$

The uncertainty of the model predicting the class of x is defined as the prediction with the highest posterior mean (highest overall likelihood for each model). The per-class prediction uncertainty is given as follows, represented as the standard deviation of the prediction probability across all models.

$$c = \frac{1}{T}\sum_{i=0}^{T} [f_{nn}^{d_i}(x) - p]^2 \tag{3.2}$$

Samples with a low amount of variance in the prediction probability across all iterations and a high mean probability demonstrate a case with high certainty. An uncertain prediction will have a higher variance of prediction probabilities, demonstrating that different portions of the neural network have conflicting activation for a given input. This calculation of the model uncertainty provides us with the basis to implement Active Learning techniques for deep learning architectures. We use the MCDU to determine which data samples and their associated pseudo-labels should be added to the training data.

Semi-Supervised Active Learning

Semi-supervised active learning techniques use semi-supervised learning techniques incrementally to create stronger predictive models with minimal additional labeling labor costs [22]. We propose that semi-supervised learning combined with intelligent selection of the additional unlabeled data using the MCDU will provide significant interpretation improvements with minimal labeling costs. Traditional active learning typically requires an expert to assess and refine the labels of uncertain predictions to retrain a enhanced model. We use our transfer learning model and MCDU in conjunction with the concept of "Pseudo-Labeling" [34] to act as our active learning "domain expert" to refine our model with the unlabeled data of a different distribution. Pseudo-Labeling is an semi-supervised approach where unlabeled data is self-labeled and then used as training data as a form of

domain adaptation [74, 36]. We combine Pseudo-labeling with the MCDU produced by our transfer learning model to select data and then retraining the model with the self-labeled data, creating the PATRL process.

We suspect a diminishing return on the number of pseudo-labels that can be introduced into the model, and we propose the concept of "pseudo-label convergence" as a metric to determine a point at which additional pseudo-labels no longer provides significant benefit to the model. As pseudo-labels are iteratively added to the model, we expect the predicted data labels to converge as more information is received. We design our convergence metric to penalize cases where many different labels are predicted but not affect cases where uncertainty is between few labels.

The iterative pseudo-label convergence metric for a given data sample x is defined as: $C(x) = [c_1(x), ..., c_L(x)]$, where L the total number of active-learning iterations and y_l is the predicted label from the active learned model $y_l = f_l(x)$. We define the iteration-to-iteration change of the predicted label y_l as ΔY_l. $\Delta Y_l = 1$ if and only if $y_{l-1} \neq y_l$ and 0 other wise to represent if a label has changed after a training iteration. Assume the label space of the output labels to be defined as Y with length $|Y|$ and the set of unique labels from prior iterations as $Y_{l,u}$ and the number of possible unique labels at iteration l as $Y_{l,p}$ shown in (3.3)

$$Y_{l_p} = \begin{cases} l+1 & \text{if } l < |Y| \\ |Y| & \text{if } l \geq |Y| \end{cases} \tag{3.3}$$

We define our pseudo-label convergence metric at iteration l as a product of the count of label changes and the unique labels predicted across the iterations, shown in (3.4).

$$c_l(x) = (\frac{\sum_{i=1}^{l} \Delta Y_i}{l})(\frac{|Y_{l,u}|}{Y_{l,p}}) \tag{3.4}$$

We expect this metric to reveal cases where the pseudo-labels never change, quickly change after few pseudo-labels, and inconsistent decisions over iterations. This metric helps to establish when the additional pseudo-labels no longer significantly change the model's predictions and the pseudo-active learning process should end.

Pseudo-Labeling Selection Methods

In this work, we assess applying additional pseudo-labels of the most uncertain predictions (high MCDU) and the most certain (low MCDU) versus randomly selected pseudo-labels. We hypothesize that additional low confidence pseudo labels may negatively impact the model as the given label is likely to be incorrect. High confidence predictions may emphasize deterministic data samples whereas random selection may be the best to represent the unknown data distribution. Emulating the active learning training schedule, we incrementally apply additional pseudo labels, retrain train the model, and use the pseudo-label convergence to determine the optimal number of additional samples to apply.

Given our initial classifier trained on limited labeled data, we define the PATRL training process in Figure 3.3. The MCDU is calculated for all samples in the unknown data set and m_L samples are taken depending on the MCDU selection method. The m_L samples and their given pseudo-label are then used as additional training data to create a new model containing both manually labeled data and pseudo-labeled data. This process is repeated for m_i iterations resulting in a PATRL-trained model.

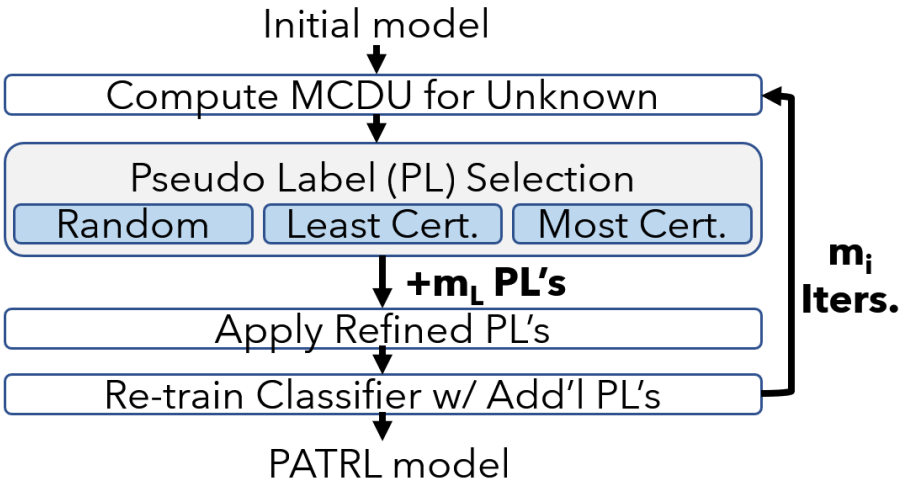

Figure 3.3: PATRL's iterative training process using pseudo-labels based upon MCDU of unknown alert description.

This process will be the basis of our experiments to measure the effectiveness of PATRL. We use the over-iteration MCDU's, the over-iteration predictions, and the label convergence to determine the PL selection method, m_L, and m_i configuration that can improve the accuracy of a model with no additional manually labeled data. In the next section, we define our design of experiments

where we define data sets used, transfer model training configurations, our Pseudo-Active Learning training schedules, and our experiments using MCDU and the Pseudo-Labeling Convergence Metric.

3.4 Design of Experiments

We design the experiments to answer three sets of questions:

1. What unstructured texts can be best transferred to interpret the Suricata alert descriptions? Is the transferred model trained with small amount (1%) of alert descriptions sufficient to interpret the remaining ones?

2. Does pseudo active learning help enhance the model's ability to interpret unseen alert descriptions? How do we select the optimal pseudo-labeled descriptions to achieve good performance for the unseen ones?

3. How do we provide a measure to reflect the confidence of the predicted AIS for a given alert description? What does the prediction confidence mean for the unseen/unlabeled alert descriptions?

To begin addressing these questions, we define two manually labeled data-sets of Suricata alert descriptions to AIS. To assess our baseline performance of our transfer learning model and provide initial training data for our pseudo-active learning model, we label alert descriptions from the security competitions CPTC [54] and CCDC [21] to approximate the types of alerts captured in a realistic network environment. A second labeled data set is created to estimate the performance of the unknown Suricata alerts. We randomly select a subset of alert descriptions from the remaining $\sim$64k defined alerts, manually apply labels, and set aside for strictly testing purposes. This data set approximates the distribution of the entire Suricata unlabeled data set and we relate our performance of this "unknown" labeled test set to the pseudo-label convergence metric and establish how to configure PATRL without this set. These two data sets and their label distribution is shown in Table 3.1.

Table 3.1: Action-Intent Stage (AIS) distributions for two labeled sets.

Action-Intent Stage	CPTC/CCDC Alert Desc.	Unknown-Random Labeled Set
Network Discovery	69	4
Vuln Discovery	52	3
Information Discovery	113	18
Credential Access	37	0
Command and Control	99	525
Code Execution	122	198
Privilege Escalation	39	1
Denial of Service	86	5
Data Exfiltration	69	15
Data Delivery	70	18
None (Non Malicious)	91	30
Total:	847	922

3.4.1 Transfer Learning with Cyber-Security Language Models

We suspect that cyber-relevancy and the volume of text data play a role in the transferable information in the language model. Table 3.2 shows the source and word counts of the data used to train the language model of the ULMFiT architecture. Leveraging no external sources, we investigate language models trained with only Suricata rules, where the language model will be finely-tuned to the structure of the sort of data being classified. MITRE's CVE database [10] is much higher in volume and abundant in information but has a much different language structure than alert signatures. We expect irrelevant language models like IMDB and Wikipedia to perform poorly, as we believe the relevancy is more important than volume.

Table 3.2: Unstructured text sources used for language model training.

Source	Word Count
Labeled Data Unique Alerts	4,626
MITRE ATT&CK Descriptions	40,808
IMDB	247,797
Entire Suricata Rule Set	482,961
CVE Descriptions	5,433,338
Pre-trained Wikipedia Set	100M+

Our first set of experiments varies the language model and uses the CPTC/CCDC set as the benchmark for accuracy. The data set is split with a train/test ratio of 80/20 with 5-fold validation on each classifier model, and we report the average accuracy over the folds. This work uses the

ULMFiT architecture within the Python Fast-AI library using the AWD-LSTM architecture for the LSTM stage, following the hyper-parameter tuning as recommended by Howard *et al.* [24].

3.4.2 Pseudo-Active Learning Experiments

To assess the differences in prediction quality for each PL selection method we follow the PATRL training process described previously in Figure 3.3, where $m_L = 250$ and $m_i = 20$. We choose 250 PL's per iteration to give sufficient iterations before and after the size of our known training set and to balance computational effort. We run this process for 20 iterations for each pseudo label selection method and use our label convergence metric to determine the optimal number of iterations needed.

At the end of retraining for each iteration, we calculate the MCDU for each sample in both the CPTC/CCDC data set and the unknown data set and we report the average MCDU on a per iteration basis. We use the unknown test set as our benchmark, referring to the ground truth to verify observations only. The accuracy of the unknown test set over iterations is used to show the Pseudo-Active training process's effectiveness and estimate accuracy over the entire unknown data set.

3.5 Experimental Results

Recall the three sets of questions laid out in the beginning of Sec 4. We begin answering these questions by assessing the contributions of transfer learning to the PATRL process.

3.5.1 Transfer Learning w/ Unstructured Texts

We hypothesize that the cyber-relevancy of unstructured text used for the language model is critical to translating intrusion alert descriptions to AIS. We consider the small set of labeled alert descriptions collected from CPTC and CCDC to test this hypothesis. Table 3.3 shows the 5-fold cross-validated accuracy for this labeled set. Ideally, a language model trained with the entire Wikipedia database would have cyber security information embedded within. As can be seen from Table 3.3, such model has a 35% Top-1 accuracy, worse than using a Multinomial Naive Bayes model without transfer learning. Adding irrelevant texts from IMDB to Wiki does not help.

On the contrary, when adding cyber-relevant texts, such as MITRE ATT&CK, CPTC/CCDC

alert description themselves, all Suricata alert descriptions, or CVE database to train the language model, we start to see higher performance for both Top-1 and Top-3 accuracy at 60∼70% and 85∼90% ranges, respectively. Note that the amount of unstructured texts from each of the cyber-relevant sources varies significantly - recall Table 3.2. The computational time needed to train the language model grows approximately linear with the word counts. Fortunately, the PATRL framework only requires to train the language model once and is fine to consume a large amount of unstructured texts to build a high-performing model. Therefore, we experiment the case where all cyber-relevant texts are used in addition to Wiki. This results in a much higher Top-1 and Top-3 accuracy at 80% and 90%, respectively.

Table 3.3: Cross-validated accuracy for the CPTC/CCDC data using unstructured texts to train the language model (LM).

Transfer LM w/ Text Source(s)	Top 1 Acc.	Top 3 Acc.
Multinomial Naive Bayes (No LM)	.5452	.8025
LM: Wikipedia (Default)	.3535	.61
LM: Wiki + IMDB	.4357	.75
LM: Wiki + MITRE ATT&CK	.5928	.8786
LM: Wiki + CPTC/CCDC Suricata	.6462	.9048
LM: Wiki + All Suricata (64k)	.6871	.85
LM: Wiki + CVE Database	.6975	.8929
LM: Wiki + All Cyber-relevant Texts	.8024	.9084
LM: Wiki + All Cyber + 1k Random PL's	**.8292**	**.98**

The accuracy results are cross-validated with respect to the small set of Suricata alerts collected through the CPTC and CCDC events. These alerts represent those an analyst might be familiar with and able to label to train the alert-to-AIS model. Under the PATRL framework, we may ask: what if we add some randomly selected pseudo-labeled alert descriptions? The last row in Table 3.3 shows the Top-1 and Top-3 accuracy when we add an additional 1,000 pseudo-labeled alert descriptions. As can be seen, this attempt further improves and reaches 98% Top-3 accuracy. These results demonstrate clearly the promise of using transfer learning with unstructured cyber-relevant texts and pseudo-labeled alerts, at least for the small expert-labeled set of alert descriptions.

We now turn our attention to what this model can do for the unknown alert descriptions. We labeled a set of randomly selected Suricata alert descriptions as described in Table 3.1. We use the 'LM: Wiki + All Cyber' language model with different combinations of training and test sets between the 'CPTC/CCDC' and 'Unknown Test' data. Table 3.4 shows the Top-1 and Top-3 accuracy

results for the various combinations. As can be seen, directly using the 'LM: Wiki + All Cyber' model with mismatched training and test sets has its limitations. Comparing to the results shown in the second to the last row in Table 3.3, we see the Top-1 (Top-3) accuracy dropped from 80% (90%) to 72% (80%), respectively. This problem is even more obvious when using the 'Unknown Test' as the training set and testing on the 'CPTC/CCDC' set with 31% (62%) Top-1 (Top-3) accuracy. This is due to that the small set ($\sim$1%) of Suricata alert descriptions from a couple given networks in CPTC and CCDC do not reflect the remaining (99%) descriptions that may show up. This brings the question on how to select the pseudo-labeled data so that the 'LM: Wiki + All Cyber' model can be improved effectively to predict the remaining descriptions. We will experiment a few Pseudo-Active Learning options next.

Table 3.4: Top-1 (Top-3) accuracy using 'LM: Wiki + All Cyber' model with different combinations of training and test sets between the 'CPTC/CCDC' and 'Unknown Test' data.

Training — Testing	CPTC/CCDC	Unknown Test
CPTC/CCDC	.9385 (.9742)	.7216 (.8001)
Unknown Test	.3116 (.62)	.9271 (.995)

3.5.2 Enhancement w/ Pseudo-Active Learning

Given the strongest model trained with all the cyber security texts, we apply Pseudo-Active Learning by iteratively adding PL's based off of the calculated MCDU of the remaining set of unlabeled data. Recall that our initial hypothesis mirrored the active learning methodology where the least confident PL's (high MCDU) should be added to the model. Figure 3.4 shows the Top-1 accuracy when PATRL is used to predict AIS for both the CPTC/CCDC labeled set and the small unknown labeled test set as more pseudo-labels are added. We found that the application of PATRL improves the unknown data accuracy across all PL selection methods to nearly 85% with a top-3 of 99%. While our top-1 performance is exceptional given our limited training data, we find the top-3 performance to be extremely desirable. A single miss-classification in a critical situation could lead to substantial impact to a network and we believe it would be advantageous to identify *when* the user should refer to the top-3. We assess the MCDU values in the next section to gain insight into this. This is also accompanied with little to no impact to the CPTC/CCDC data accuracy at 94%, further demonstrating PATRL maintaining accuracy for what we believe are the most relevant alerts for the user.

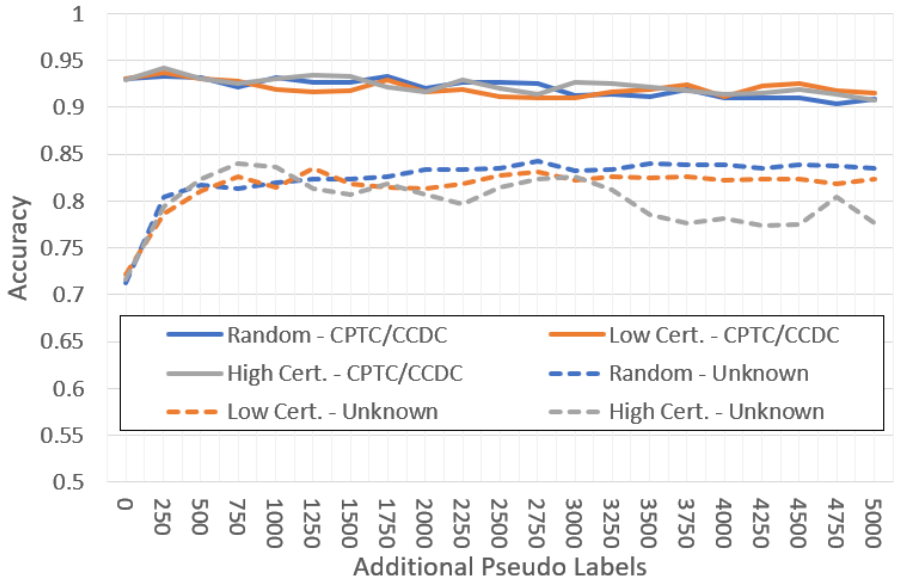

Figure 3.4: Accuracy for each PL selection method for the CPTC/CCDC and Unknown datasets. Each PL selection method in PATRL increased performance by over 10% on the unknown test set.

Each selection method achieved exceptional accuracy on the unknown test set, we found randomly selecting labels was unique in the sense that there was little variation in the accuracy once a 'sufficient' amount of PL's was added. This result goes against traditional active learning and our hypothesis using the lowest confidence samples to pseudo labeling. Recall that our test set is a random subset of the entire Suricata rule set, whereas our initial training set was extracted from 'real' cyberattack exercises. This is an important distinction because as more randomly selected pseudo labels are used for training, we fine-tune the model toward the actual distribution of the unlabeled Suricata rule set. Given the randomly selected set that represents the unknown distribution, one may use the accuracy measure with respect to this set to determine the 'sufficient' amount of PL's and iterations needed to refine the prediction model. In practice, however, one cannot expect that this test set exists. We instead use the label convergence metric to estimate the point at which the predictions do not change significantly throughout the iterations.

The results for the iterative label convergence metric is shown in Figure 3.5; this metric represents the change in predicted labels throughout the PATRL process. The steep decline in the convergence metric between 250-1000 pseudo-labels indicates that most of the changes to the predicted classes occur early on in the Pseudo-Active learning process. Naturally, we expect the labels

to eventually converge given that PATRL is self-labeled. But when compared to our unknown test set performance, we look for insights for *when* we could hit a point of diminishing returns.

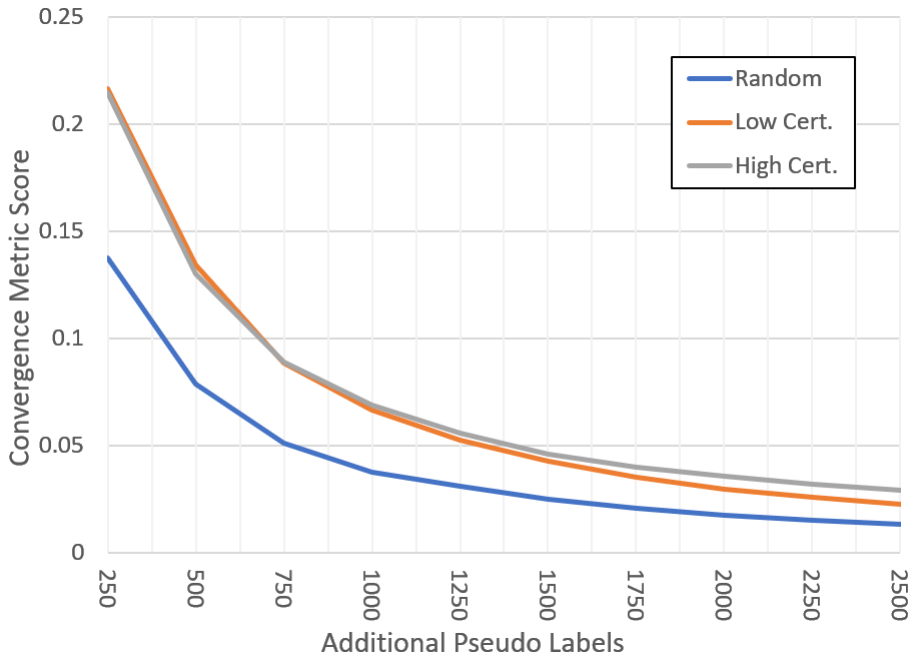

Figure 3.5: Pseudo-label convergence metric for each PL selection method. The predicted labels for the unknown data converge once the amount of pseudo labels exceed the initial training data.

Given our knowledge of the random unknown data performance in Figure 3.4 and the convergence metric in Figure 3.5, we show that there is a limit to the number of PL's that can be added to the model. Applying PATRL with randomly selected pseudo labels reaches optimal performance consistently assuming the number of additional PL's is greater than or equal to the initial training set. From a user configuration standpoint this is desirable because our main motivation of this work is to not further burden security analysts with excessive labeling and configuration. Now that we have achieved promising accuracy on our unknown test set, we now evaluate if the model can be trusted in classifying the remaining unlabeled data. We analyze the MCDU's with respect to the unknown test set to assess the models confidence when predicting the unknown test set correctly and incorrectly.

3.5.3 PATRL with MCDU Analysis

Given the high top-3 accuracy with PATRL, we believe it would be advantageous if it was possible to identify when we should consider the top-3 over the top-1. Figure 3.6 demonstrates then when there are no pseudo labels applied, the MCDU's for the positive and negatively classified samples for the unknown test set are *nearly identical*. As PATRL is applied, the MCDU decreases with correctly identified samples while the negative samples remain uncertain or, in the case of the random selection, maintain a consistent margin. A significant observation of this work: Pseudo-Active trained models can indicate probable correct and incorrect top-1 predictions through the MCDU. In practice, it would then possible to define thresholds of when the MCDU of a prediction is sufficiently high, it could recommend the user to the top-3 where the correct classification is *likely* to be contained within.

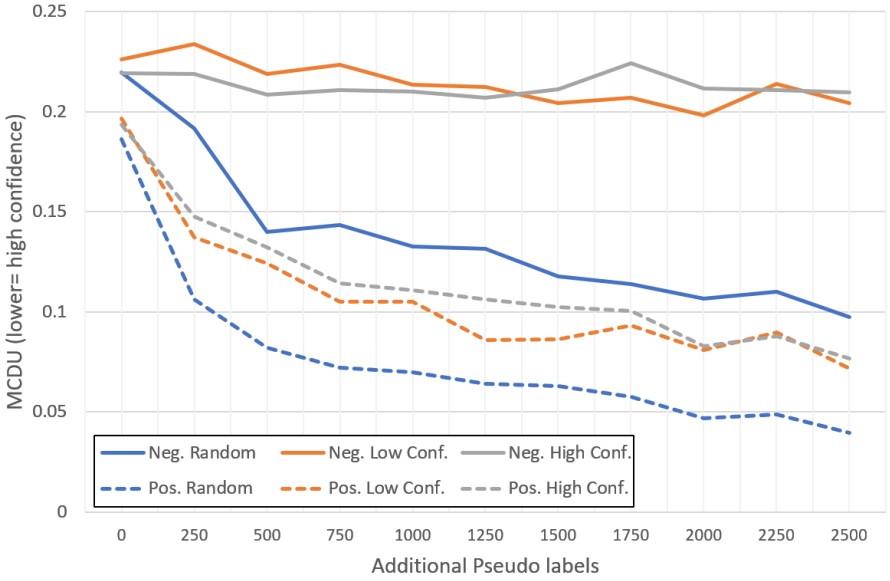

Figure 3.6: Average MCDU values for the positive and negatively classified samples for the unknown test set, for each PL selection method.

Given this observation, imagine the scenario where a security practitioner receives a new alert where they are unfamiliar with nor was it included in the initial training set. Inaccurate assessment of an IDS alert can lead to devastating consequences to a network, and gaining users' trust in applying such models is challenging. It would not be feasible to expect users of PATRL to assess every prediction of new or unknown data. Although, given the observation that the MCDU now reflects the confidence of a correct top-1, we believe we can significantly reduce the number of times the

user should intervene with the top-3. Figure 3.7 shows the distribution of the MCDU's for the remaining unknown Suricata alerts with and without PATRL applied. Assuming an MCDU threshold of .1 from Figure 3.6 (represented by the vertical red line in Figure 3.7), we find that over 75% of the unknown signatures fall below this threshold using PATRL. For those samples above this threshold, we recommend users refer to the top-3, whereas without PATRL, users will have little to no confidence their model is producing correct results as only 2% fell under our uncertainty threshold. This shift in prediction certainty with PATRL allows users to trust confident predictions while narrowing down the selection process to the top-3 for those uncertain predictions.

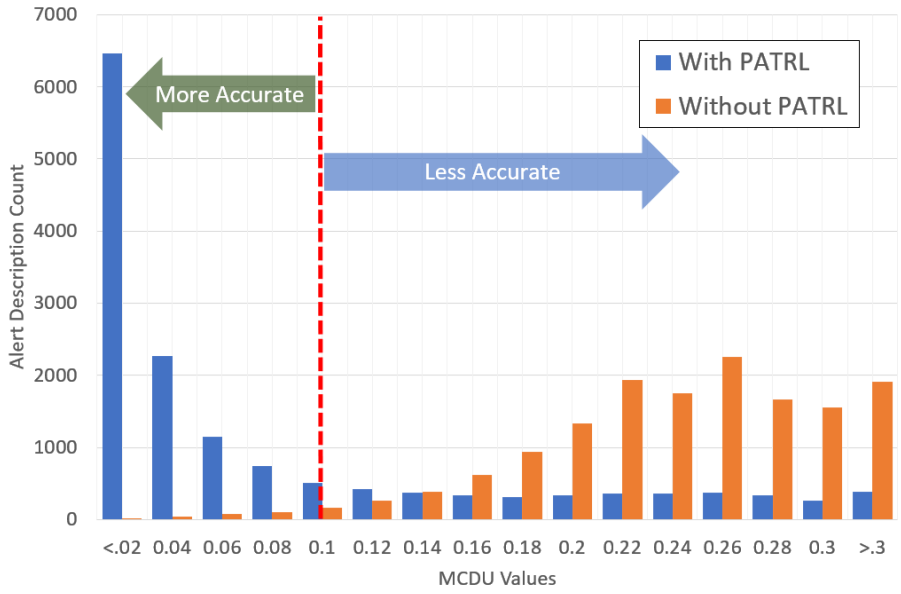

Figure 3.7: Histogram of MCDU values of the remaining unlabeled Suricata alert descriptions, with and without PATRL applied.

To demonstrate the usage of PATRL and the MCDU, Table 3.5 shows the top-3 predictions and MCDU values for a random selection of ten unlabelled Suricata alert descriptions. Using our prior observations of the MCDU, we find two of these alert descriptions to have an MCDU value significantly higher than the rest. This indicates that for those two descriptions that the top-1 value is unlikely to be correct and the 2nd or 3rd attack stage prediction should be considered. In this case, we confirm that "PHP Scan Precursor" should be considered as "Information Discovery" and the "DataCha0s" alert should be considered as "Data Exfiltration". The main distinction being that "DataCha0s" is reported to be used post-exploitation to dump data from a webserver, where a PHP

scan is clearly for information gathering purposes. For all other low-MCDU/confident predictions, we agree with the attack stages given and further demonstrates the accuracy of PATRL with significantly limited labeled data.

Table 3.5: Top-3 attack stage predictions and corresponding MCDU values for 10 random unlabeled Suricata sigantures using PATRL. Note: C2 is "Command and Control"

Signature Description	1st Pred.	2nd Pred	3rd Pred.	MCDU Value
ETPRO CURRENT_EVENTS Successful Netflix Phish 2017-12-26	PHISHING	CODE EXECUTION	CREDENTIAL ACCESS	0.000465
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 597	NON MALICIOUS	DoS	C2	0.001219
ETPRO TROJAN Trojan-Dropper.Win32. Dapato.cvia Checkin 2	C2	DATA DELIVERY	DoS	0.010048
ET WEB_SPECIFIC_APPS phpx SQL Injection Attempt – forums.php cat_id UNION SELECT	CODE EXECUTION	INFO DISC	C2	0.012862
ETPRO TROJAN Worm.Mydoom spreading via SMTP 30	C2	DATA EXFILTRATION	DoS	0.015334
ETPRO WEB_CLIENT MS15-124 Internet Explorer Memory Corruption Vulnerability (CVE-2015-6134)	CODE EXECUTION	NON MALICIOUS	DoS	0.035844
ETPRO EXPLOIT Tivoli Storage Manager Initial Sign-on Request Buffer Overflow	CODE EXECUTION	DoS	C2	0.065976
ET MALWARE Realtimegaming.com Online Casino Spyware Gaming Checkin	C2	NON MALICIOUS	CODE EXECUTION	0.082716
ET WEB_SERVER PHP Scan Precursor	CODE EXECUTION	INFO DISC	DATA DELIVERY	0.181268
ET WEB_SERVER DataCha0s Web Scanner/Robot	INFO DISC	DATA EXFILTRATION	C2	0.192619

3.6 Conclusion

In the evolving landscape of cyber-attacks and defenses, it is becoming increasingly difficult to keep up with the continuously evolving and complex IDS outputs. In practice, security analysts may not have sufficient expertise to know every single possible IDS alert or time to research every new alert as they appear. We demonstrate that Transfer Learning combined with Pseudo-Active learning, PATRL, is able to produce strong alert interpretation models in situations where familiar (labeled) alerts are extremely limited and unknown (unlabeled) alerts can exhibit very different characteristics. The introduction of the cyber-focused language model yielded 25% improvements

in translating IDS alert descriptions to AIS for the limited labeled data when compared to a baseline without external data. For new unknown alerts, however, the model requires additional refinement without labeled descriptions as these alerts may have the most substantial impact if incorrectly identified or ignored.

Combining the Pseudo Labeling and Active Learning concepts, PATRL refines the model by adding 'pseudo-labeled' alert descriptions to the model trained with known labels. We proposed various pseudo label selection methods based upon the Monte Carlo Dropout Uncertainty (MCDU), where we saw 10-15% increases in predicting a randomly selected subset from the entire Suricata rule set, achieving 83% top-1 and 99% top-3 accuracy. We show that randomly selecting new pseudo labels in PATRL best represented the unknown distribution of the unlabeled alert descriptions. Metrics such as the change in uncertainty and our Pseudo Label Convergence metric optimize the configuration of Pseudo-Active Learning for the unknown data without a labeled set. Furthermore, we demonstrate that the MCDU of Pseudo-Actively trained models reflect the correctness of the prediction. This gives a measure for the analysts to establish trust and understanding of a system predicting AIS from unknown, unfamiliar alert descriptions. We found our model's top predictions can be confidently used for 75% of the entire Suricata rule set, while recommending analysts the highly accurate top-3 AIS's for the remaining 25%.

3.6.1 PATRL Limitations

We find PATRL's attack stage interpretation accuracy to be impressive given the challenging language task and the limited labelled data for the data hungry deep learning classifier employed. We would prefer for our top-1 accuracy to be in the low 90% range as opposed to the low 80% to further reduce the amount of times an analyst should refer to the top-3 value. This will also increase the "accuracy" of our eventual extracted attack campaigns in HeAT if the first predicted attack stage is more likely to be correct. Given our experience with the usage of PATRL, we have identified some limitations that we believe are worth mentioning for others to address if adopting the PATRL process.

Deep learning architectures, like our language model, are a black-box and it is extremely difficult to understand what the model is actually learning and why. This is a very common criticism of deep learning techniques and within the computer vision research community there are efforts for

explainable AI [58]. It is evident that our language model transfers and improves our interpretation accuracy with our cyber-specific texts, however it is unclear what it is actually learning, relating, and correlating to attack stages. With traditional techniques like random forest, we would be able to assess the decision trees to understand how each word in the description contributes to the probability of being a part of each attack stage. Where the meaning of the activation of neurons within a neural net is abstract due to the non-deterministic stochastic gradient decent training. It is difficult to determine if we should prioritize improving the language model to improve accuracy of PATRL or give more high quality labelled data examples. As the language model can only contain a finite amount of "knowledge", we wonder if the language model text can be optimized even further to the cyber-domain. The text used to train the language model has a large word count and its reasonble to think that not all of the text is actually relevant. If we could interpret what the language model is actually learning we wonder if we can gain extra performance with optimized cyber-texts.

Multiple attack stages could be valid for a single observed action. Using our signature description example from the beginning of this work, this described a vulnerability with the identifier of CVE-2014-3704 which resulted in arbitrary code execution and *potentially* privilege escalation. This is an important distinction to make as not all code execution attempts lead to privilege escalation but the user should be aware that this is a possibility. Currently the labelled samples in PATRL are only mapped to a single attack stage, where in reality we may want to choose a ranking of multiple potential attack stages if needed. While for the description the top-2 attack stage predicted was indeed privilege escalation, we would prefer if the result was arbitrary code execution *and* privilege escalation as opposed to the implied "or" relationship of the reported top-3. This would lead to even more meaningful interpretations of the attack stage from the alert description.

Assessing the true accuracy of the PATRL process may be difficult depending on the context or application. The concepts behind PATRL can be applied to label other types of text data, however our IDS data has a few specific characteristics that we believe contributes to our accuracy that may not apply to other domains. First, official Suricata IDS alert descriptions are consistent in their formatting and often contain keywords such as specific services like "mySQL" that are typically only vulnerable to a specific subset of the AIF. We believe that this consistency in the format in the IDS alerts allows the model to quickly identify the important keywords related to the attack stages, where as less structured inputs may need additional data. The second, the rule set for Suricata is

relatively finite (although new rules are added daily) and thus set of unlabelled rules are known. This gives us the opportunity to create the "unknown" test set to estimate our performance on the remaining unlabelled samples. We foresee the issue of estimating accuracy for domains where the input space is unknown or even infinite. Our pseudo-label convergence metric and MCDU provides a look into if the model is performing well, however testing and verifying less structured inputs would have to be approached differently.

3.6.2 Discussion: Future of PATRL

Before moving onto our usage of PATRL in our final work HeAT, we would like to discuss briefly how PATRL can be applied in the future. PATRL can enable other researchers who use IDS alert data to have an extra feature for correlation and prediction. For example the creators of SAGE, Nadeem et al., now can apply their process to other data sets or other networks without requiring to label undefined alerts. This could further makeup some of the data-deficiencies that we face in the cyber-security research field. PATRL also can be adapted to accepting other forms cyber-security related text to apply attack stages on, similar MITRE TRAM's objective of identifying the attack stages given a threat report [78]. We could explore methods to annotate, tag, and summarize full threat reports with customized cyber-attack information without the need for massive amounts of labelled training data. We are interested in the flexibility of PATRL, as the general process of pseudo-active transfer learning could be applied to other scenarios where labelled data is limited but unstructured/unlabelled data is unlimited. Pseudo-active learning combined with a transferable deep learning model could enable other domains (i.g. domain adaptation) to leverage unlabelled data to increase accuracy with little human effort that otherwise would have been left on the table.

PATRL becomes the backbone of our next work HeAT, applying attack stage labels to each unique alert description to provide additional context for HeAT to extract attack campaigns. HeAT uses the PATRL-interpreted attack stage labels to aggregate similar alerts into alert episodes, as a feature for prediction, a metric to measure attack campaign quality, and to clearly visualize the extracted attack campaign. With out PATRL, the process described by HeAT and the resulting found attack campaigns would be less meaningful as the individual attack stages are the core of an attack campaign. In the next chapter, we describe our process "HeATed Alert Triage" to uncover attack campaigns within IDS alerts by leveraging surveys of an analyst's opinion of the criticallity

of actions within attack campaigns prior.

4. Heated Alert Triage (HeAT): Network-agnostic Extraction of Cyber Attack Campaigns

Given that we can now describe any IDS alert in terms of the AIF using PATRL, we now can process and relate alerts based on their attack stage and other alert attributes so that attack campaigns within can be described. Once again, we are faced with data-limitation challenges as IDS data sets containing labelled attack campaign ground truth is rare, limited in scope, or outdated. It is common practice for SOC analysts to triage alerts to manually compile evidences of an attack campaign where we thought: "Can we tap into the SOC analysis workflow, capture the reasons why the analyst thinks alerts are a part of an attack campaign, and then use those observations to find other attack campaigns?" HeAT is based around the concept that analysts may have different opinions of the alert severity given the attributes within the alert, their own expertise, and knowledge of the network. We look to use the experience of the analyst to find indicators of an attack campaign given evidence that an attack may have occurred and use the extracted indicators to find other potential attack campaigns. Our technical abstract for HeAT is as follows:

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.

4.1 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 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" 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 HeAT-value 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 analyst-labelled 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.

4.2 HeATing Episodes to Extract the 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 4.1.

- Introduce the labeling approach called "Alert Episode Heat" (HeAT-value) 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 the HeAT-value regardless of attack type or network configuration,
- use ML/AI to learn and predict HeAT-values for prior episodes to a critical episode,
- construct and visualize HeATed Attack Campaigns (HAC) with "HeATed episodes", and
- propose the entropy-based HeAT-gain metric to prioritize HAC's with desirable attack campaign properties.

In the subsequent sections of this chapter we define the concept of how the HeAT-value is used to represent attack scenarios, describe our process to aggregate alerts as alert episodes, define our initial triaging process, and then finally describe our application of HeAT to generate the HAC for a given critical alert.

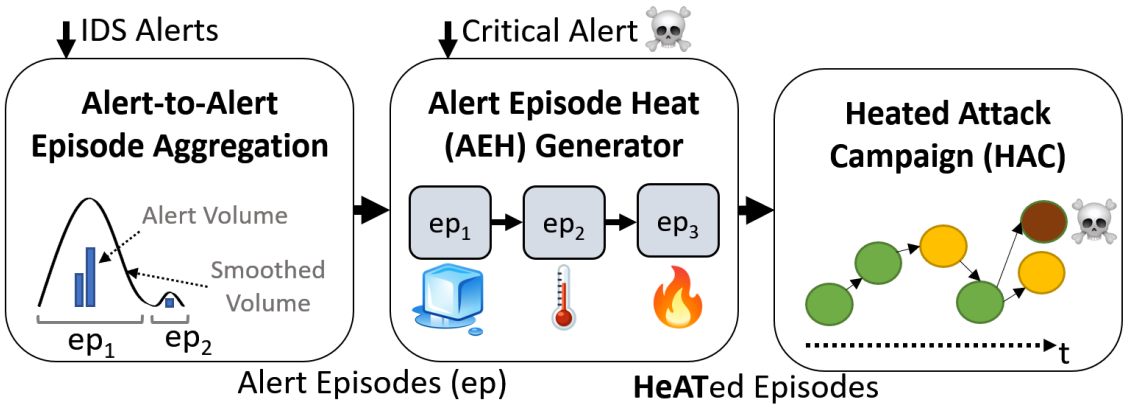

Figure 4.1: Process overview of HeAT to generate HAC from a set of IDS alerts and a given critical alert.

4.2.1 Meaning of HeAT - Progress Towards Attack Objective

The concept of Alert Episode Heat, or the HeAT-value, is a numeric ranking system (0-3) which given a critical alert episode e_c and a prior episode e_p, the HeAT-value 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. The HeAT-value 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 HeAT 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 Heat 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 HeAT-values as a small set of discrete values that signify key milestones within an attack campaign. Table 4.1 describes the characteristics of the distinct HeAT-values used to label and

create the initial triage training set. We use the high-level attack stages such as "reconnaissance", "exploitation", and "actions on objective" [11] 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 [3]. 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 4.1: Description of the the HeAT-value levels relating to attack milestones

HeAT-value	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.

4.2.2 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 our own Gaussian Smoothing approach to aggregate alerts based on source IP, attack stage, and time [44]. We describe 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. We concluded [44] mentioning that the Suricata alert "category" is a weak representation attack stages of an well established attack stage framework such as MITRE ATT&CK. We use PATRL to map all Suricata alert descriptions to the AIF so that alerts can be aggregated based on similar attack stages. 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 4.2. 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.

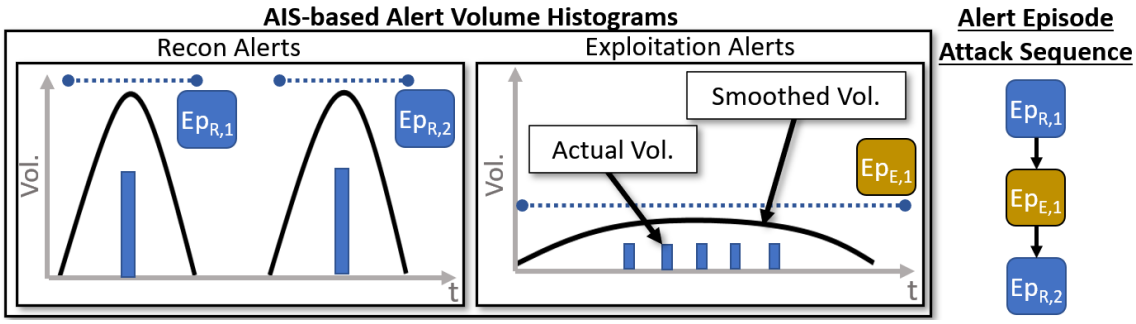

Figure 4.2: The Gaussian smoothing approach by Moskal *et al.* accounts for variations in alert arrival time to create Alert Episodes and creates sequences of episodes by sorting episodes by peak smoothed volume times.

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 HeAT-value.

4.2.3 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 HeAT-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 4.2 which are derived from the alerts contained within the episode.

Table 4.2: 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 4.3. 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.

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 HeAT-value for other attack types and be applied to other networks. In the next section we describe our methodology for creating the "Heat Generator" and how we leverage a small amount of labeled HeAT-values to determine HeATed Attack Campaigns (HAC).

Table 4.3: 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

4.2.4 Heat Generator: Learning the Analyst's Heat

The "heat generator" is our name for a machine learning model for predicting the HeAT-value given the aforementioned network agnostic features representing the relationship between two alert episodes. When defining the concept of the Heat 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 HeAT-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.

Datasets for the Initial Triage

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 [54] and use observations from this set to find other attack campaigns in a real-world SOC data set. 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 [69] and we use PATRL to provide the attack stage described by each alert. Table 4.4 summarizes the characteristics of the overall CTPC data versus the single team used for training.

Table 4.4: Characteristics of the alerts and episodes between all teams in CPTC and the team used for the initial training triage.

	Unique Sources	Unique Targets	Unique Sigs	Total Alerts	Total Episodes
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 heat generator, we ask the user to perform a short initial triage of the prior episodes with respect to a critical IoC and apply HeAT-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 HeAT-value 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. 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 HeAT-value 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 heat 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. We then manually apply HeAT-value to prior episodes to the critical IoC's and show the distribution of HeAT-values given in Table 4.5.

Table 4.5: Distribution of Team Train's HeAT-values of prior episodes given the IoC's

Training HeAT-value	HeAT-value Count
0	720
1	202
2	154
3	347

Recall that our network agnostic features describe the relationship between two alert episodes, where one of the episodes are an episode containing one of the IoC definitions above. As this competition data set has many objectives we could identify episodes with no relationship to the IoC being triaged, thus our labelled dataset is skewed towards zero heat-value observations. We are not concerned about this however as actual attacks in the real-world will be rare when compared to the number of alerts raised. Given this set of traiged pairs of episodes with analyst-supplied HeAT-values, we convert the episode pair into our network-agnostic feature space, use that to train the heat generator, and then use the heat generator to replicate the triage process with HeAT.

HeATing Episodes to Develop HAC

Given an set of traiged episodes with the analyst's labelled HeAT-value, we define the heat 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 heat generator applies a non-zero HeAT-value. Our requirements for a selecting a machine learning model for this application are bound by our non-linear features and that the HeAT-value must be a continuous value. HeAT is implemented in Python and our heat generator leverages Fast.AI's Tabular learner [16] to predict the HeAT-value. 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 heat generator is used to "HeAT" *all* prior episodes with respect to the critical episode. This process is shown in Figure 4.3. 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 HeAT-value 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 HeAT-value threshold can be applied if the user desires. We expect truly impactful episodes to have significantly higher HeAT-value levels than those with just a few similarities between features.

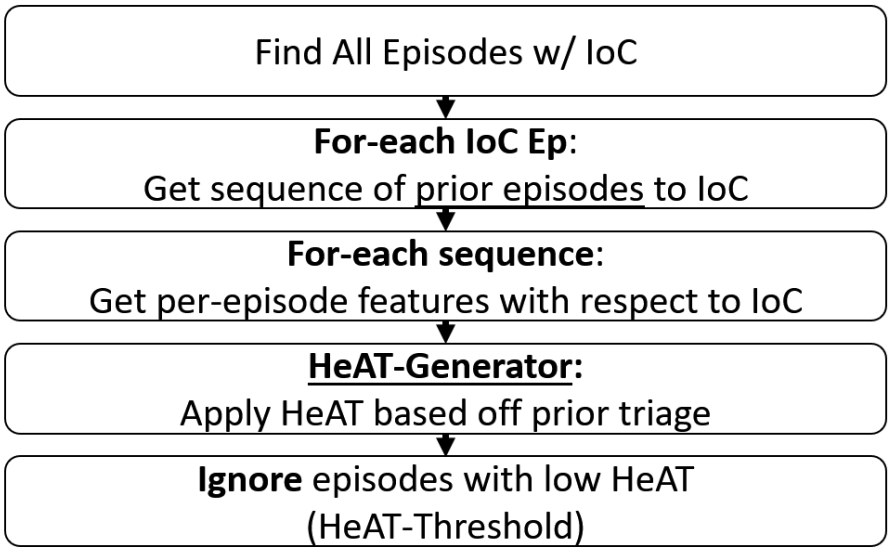

Figure 4.3: General process of HeAT to apply HeAT-values using the Heat Generator given an IoC.

4.3 Metrics to Assess HAC: HeAT-Gain

To demonstrate the additional value provided by the HeAT process and demonstrate its ability to reveal meaningful attack campaigns to the analyst, we develop a quantitative entropy-based metric called "HeAT Gain". As HeAT is intended to assess unknown/unlabeled IDS datasets, we assume that it may be impossible to determine the actual accuracy and quality of the HAC presented to the user. We propose the assessment of the HAC characteristics to infer on the "quality" of the HAC without the need for labeled ground truth data. We leverage characteristics from the generated HAC, the data being tested (history of IDS alerts), and the HeAT model training data to develop a quantitative metric to evaluate the extra knowledge the HAC provides to the user. HeAT Gain is envisioned to aid the user to quickly compare and prioritize HAC's that are most likely caused by adversarial behavior, especially in production systems with high volumes of daily traffic.

Given the attack episodes of the data-under-test E_d, the set of heat-labeled training episodes E_t, and an HAC representing a finite set of HeATed attack episodes $e \in E_h$ where $E_h \subseteq E_d$, we define HeAT Gain, $\Delta(E_h|E_d, E_t)$ as a combination of three entropy-based metrics with the following properties:

1. **AIS Coverage Gain**: E_h should contain a variety of attack stages, for example a broad coverage of the cyber attack kill chain,

2. **Noise Reduction Gain**: E_h is a subset of E_d with noisy and/or irrelevant episodes to the IoC removed, and

3. **HeAT Coherence**: the HeAT model should apply appropriate HeAT according to the analyst's initial observations in the training set E_t.

To quantify these properties using the distribution of attack stages and the predicted HeAT values, we use the standard definitions for Shannon entropy $H(X)$ and conditional entropy $H(X|Y)$ shown in Eq. (4.1) and (4.2) respectively.

$$H(X) = -\sum_{i=1}^{n} P(x_i) log_b P(x_i) \tag{4.1}$$

$$H(X|Y) = -\sum_{i,j} P(x_i, y_j) log_b \frac{p(x_i, y_j)}{p(y_j)} \tag{4.2}$$

Let A_x be the random variable of attack stages given the set of episodes E_x. Likewise let Y_x be the random variable for the predicted HeAT value given E_x where the predicted HeAT for $e \in E_x$ is given the discrete value: $\{y \in \mathbb{Z} | 0 \leq y \leq 3\}$. For both entropy definitions we calculate entropy with respect to attack stage and we use the log base b to be the count of attack stages defined in the AIF. Lastly, the subscript x represents either the HAC, the data-under-test, or training data denoted as h, d, or t respectively. Referring to the properties above, we now describe our definitions and methodology for each and then combine each property to create our HeAT Gain metric.

AIS Coverage Gain: $\delta_{ACG} = H(A_h)$. AIS Coverage Gain quantifies the diversity of the attack stages caputured in our HAC. With our assumption of a "quality" HAC containing a diverse set of attack stages, HAC's with high AIS coverage gain signifies that the HAC represents many types of attack stages and is more likely to capture each step of the cyber attack kill chain, from

reconnaissance to achieving the final attack objective. Whereas low coverage gain may indicate limited adversarial behavior, false positive alerts, or otherwise less revealing HAC's by the model. This property provides the basis of our HeAT gain metric where high coverage of attack stages is preferred. This simple metric alone may be misleading where calculated attacks that take few steps will be unfairly penalized when compared to adversaries that caused many unique alerts. Where it is completely likely the calculated attack may have a more significant impact and be harder to find. We instead use our next metric "noise reduction gain" to reward the HAC's that reduce the most amount of noise or irrelevant episodes.

Noise Reduction Gain: $\delta_{NRG} = H(A_d) - H(A_h^{'})$. The noise reduction gain measures the difference between AIS coverage gain between the entire data set E_d and the irrelevant episodes filtered out by our HeAT process. Recall that E_h is a subset of E_d where the predicted HeAT value exceeds a minimum threshold defined by the user and it is this reduction of irrelevant episodes that will provide a concise representation of the attack campaign. We define $A_h^{'}$ similarly to A_h however in this case all $e \in E_d$ and the episodes that do not meet the minimum heat threshold set are given a unique attack stage α_r, signifying that those episodes were removed from the HAC. We expect $A_h^{'}$ to have substantially lower entropy as most of $e \in E_d$ will be removed from E_h, skewing the attack stage distribution towards α_r. E_d may represent multiple days worth of episodes and we expect a vast majority of these episodes to be irrelevant. We believe this will complement the coverage gain as this rewards HAC's that only contain relevant episodes and is unique when compared to all data. Ideally we want $H(A_h)$ to have high entropy (many attack stages represented) and a near-zero $A_h^{'}$ (nearly deterministic to α_r) for the maximum amount of HeAT Gain. In the case for our calculated adversary example, $H(A_h)$ will be low as few stages are represented but $A_h^{'}$ should be very close to zero meaning the HAC is concise in the stages it is representing.

These two properties rely on a minimum heat threshold to filter out irrelevant alerts which has a significant effect on the entropy calculation. These properties do not take in account of if the predicted HeAT value by the model accurately representing the HeAT training values set by the analyst during the initial training triages. Our last metric "HeAT Coherence" will take in account the predicted heat values of E_h versus the training data E_t.

HeAT Coherence: $\delta_{COH} = abs(H(A_h|Y_h) - H(A_t|Y_t))$. HeAT Coherence measures the coherency of the relationship between the attack stage and the predicted HeAT value given from

the analyst reflected by the HeAT model. Ideally we would expect our model to apply HeAT to episodes consistently with the labeled HeAT values within the given training set. This would lead to $H(A_h|Y_h) = H(A_t|Y_t))$ and we expect the HeAT coherence to be as close to zero as possible. As our predictive features consider more than just attack stage we may experience the case where $H(A_h|Y_h) > H(A_t|Y_t))$ which may indicate that the model is applying HeAT based on other factors than attack stage which is inconsistent with the feature values given in the training set. Whereas $H(A_h|Y_h) < H(A_t|Y_t)$ would instead indicate that is deterministically applying HeAT based on attack stage, a condition that could be addressed with simple rules. We find either of these cases to be undesirable and situations where $H(A_h|Y_h)$ is constantly divergent from $H(A_t|Y_t)$ may indicate when the model is insufficient to apply HeAT given the data and additional labeled training data may be needed. Given these properties, we now define $\Delta(E_h|E_d, E_t)$ as a combination of each property to quantify the overall utility of the attack campaigns represented by our HAC's.

HeAT Gain: $\Delta(E_h|E_d, E_t) = \delta_{ACG} + \delta_{NRG} - \delta_{COH}$. The final formalization of the HeAT Gain metric rewards attack campaigns with a diverse set of attack campaigns that are unique when compared against all other episodes while penalizing and significant deviation of predicted heat values given the observations of the training data. Using entropy allows us to quantify specific properties of desirable attack campaigns and infer on the potential "quality" of the generated attack campaign without the need of labeled ground truth. Due to the volume of data typical SOC operations are faced with, HeAT may still present the user a large amount of viable HAC's to assess and metrics like this enable the user to prioritize their time on high gain HAC's. We will also show that the individual values of each of the components of the metric compared to each other can also provide interesting details about the HAC that cannot not be realized with only $\Delta(E_h|E_d, E_t)$.

Our intended usage for HeAT-gain is simple, it is to prioritize the analysis of HAC's with significant AIS coverage, free of irrelevant episodes, and is coherent with the analyst's throughout processes. In this work, we will prioritize the assessment of those HAC's that have the highest over HeAT-Gain value. There may be multiple valid HAC's for a given IoC and we will demonstrate HeAT-gain can be used to quickly focus on attack campaigns that are likely to be complete and significant to the analyst. Given our initial assumptions this entropy-based HeAT Gain metric provides an intuitive method to quantify quality attack campaigns and provide some initial impressions of the HAC before inspection of the episodes within. In the next section, we apply the HeAT model to

the other CPTC teams and a real-world SOC data set to discover and prioritize meaningful attack campaigns using HeAT-gain.

4.4 HeATed Attack Campaign Analysis

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. In this section, we will be doing case-studies on two sources of IDS data: 1) the penetration testing competition CPTC, and 2) a real-world example of alerts from a SOC. For each of these case-studies we will be using HeAT-gain to first identify the most critical HAC's for the IoC's. We begin assessing the CPTC IDS data in which adversarial behavior is abundant and easy to identify by IP address. Then we move on to our SOC dataset where there is little knowledge about the network or targets and we use HeAT to assess this network "blind." With the SOC data, we have no clear directive and we use HeAT-gain to prioritize the assessment of HAC's of a variety of IoC's to find interesting attack campaigns. We close this section by discussing our ability to capture the analyst's attack campaign knowledge using HeAT-gain.

4.4.1 Case Study: Assessment of CPTC

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 a critical alert in the past, triaged the critical alert with HeAT, 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 apply HeAT to different adversaries to demonstrate how HeAT is used to gain additional insight into attack strategies through the use of the HeAT gain metric.

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. Even given the limited scope of CPTC, manual analysis of this signature would still prove to be challenging and time consuming. Referring to Table 4.6, this CodeRed signature accounted for 38 out of the

169,448 alerts. Without HeAT, an analyst would have to manually identify which of the remaining alerts are related to each CodeRed occurrence through numerous SIEM queries. The application of our alert episode aggregation alone would make this analysis significantly less tedious as the aggregation process reduces the alert volume in this case by over 98%.

Table 4.6: Summarization of the CPTC data that an analyst would have to assess for each CodeRed observation

Unique Sources	Unique Targets	Unique Signatures	Total Alerts	Total Episodes	CodeRed Occurrences
45	81	265	169,448	3200	38

This still leaves us with 38 potential attack campaigns that will require the analyst to assess and even with the episode reduction this task may still be tedious. Due to the competitive nature of this data set and the variance in skill set, it is reasonable to assume that some of these observations of CodeRed lead to more severe outcomes than others. We use our HeAT Gain metric t to prioritize the assessment of attack campaigns and also to compare HeAT against typical methods one would use if performing a manual assessment. We consider "naive" rule-based methods for generating attack campaigns for a given IoC by collecting sets of episodes where: 1) the source IP matches with the IoC, 2) the target IP matches with the IoC, and 3) both source and target IP match the IoC.

Applying HeAT to the CPTC episodes for each observation of CodeRed, we first turn our attention to the HAC that is reporting the highest HeAT-gain overall. Figure 4.4 compares the CodeRed attack campaigns generated by HeAT versus our IP address rule-based representations of the attack campaign. Each circle represents an alert episode for a specific attack stage, the size of the episode represents the number alerts in the episode, and the color corresponds to the predicted HeAT value (Grey/Green=low HeAT, Red=High HeAT).

For the HeATed representation we found the highest gain with a minimum heat threshold of .3, resulting in a concise attack campaign with clear examples of reconnaissance, exploitation, and other high severity attack stages for this IoC. Simply searching by target IP is insufficient which is reflected by the comparatively low gain. Assessment of the breakdown of the HeAT gain metric for each attack campaign in Figure 4.5 verifies

The inclusion of searching by source IP leads to significant improvement in gain but fails to eliminate episodes in the earlier part of the campaign that we identified as apart of a prior attack

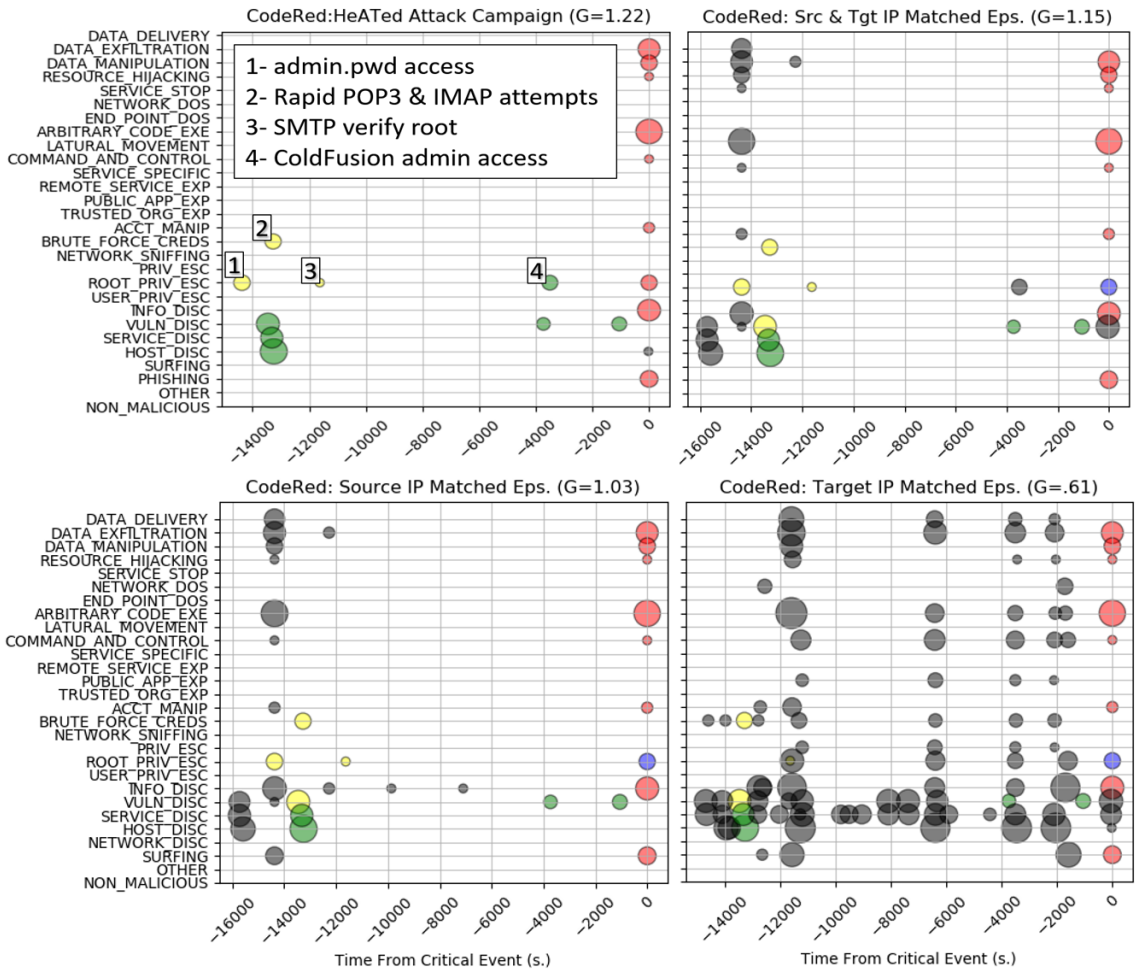

Figure 4.4: The HAC appropriately removes irrelevant episodes and enables the prioritization of the analysis of episodes with the provided HeAT level.

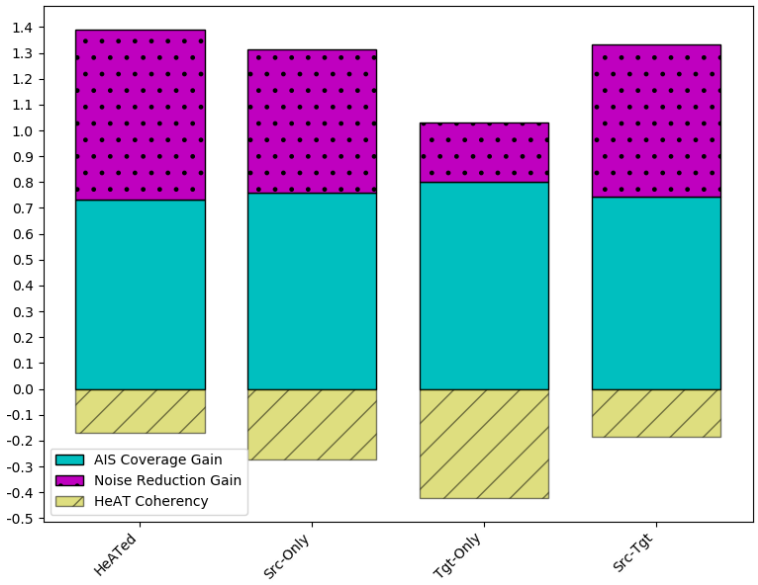

Figure 4.5: The HeAT Gain for the HAC is contributed to maintaining high AIS coverage with large amount of noise reduction.

campaign. The penalty in gain for these cases is due to the lesser amount of noise reduction when compared to the HAC representation. A benefit of HeAT is that we would not be distracted by the episodes from another campaign; where the first example of "Root Privilege Escalation" was given a significant amount of HeAT indicating that episode should also be assessed.

In many instances for the CPTC dataset, considering only the source and target IP matched may be sufficient. Consider the HAC for CodeRed in Figure 4.6 which has a significantly lower gain of .64 than the previous HAC and also is an example where the source and target matched attack campaign reports a slightly higher gain of .66. Despite the low gain, we found this HAC to be interesting as it has a low amount of noise reduction compared to other examples. We have identified that this behavior seen is the result of scripted attacks where we observe a small amount of reconnaissance at the beginning of the campaign followed numerous attempts of CodeRed on multiple targets. Our source-based aggregation benefited us greatly here as these actions were single-source, multi-target. This is evident in the breakdown of the gain of the source IP matching case being identical to the HeATed case.

Due to the simplicity of the CPTC data and the fact that it is a penetration testing competition, we find that in general the HeAT model is strongly biased towards episodes with matching source

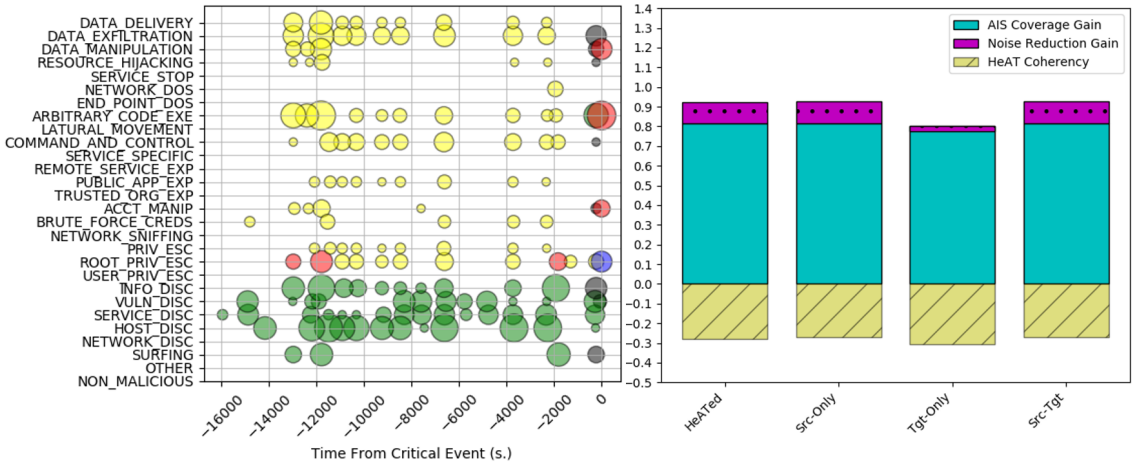

Figure 4.6: Certain behaviors such as a"script kiddie" has results in a small amount of noise reduction due to the high volume of actions from a single source, repeating the same action.

and target IP's to the IoC. There is very little benign noise within CPTC and it is safe to assume that most actions captured in this set are deliberate with no intention of going undetected. Herein lies the issue with using competition data-sets; real-world data-sets are flooded with false-positive alerts, where actual attacks are rare and there is likely effort to avoid detection. We are confident in HeAT's ability to reveal attack campaigns within an environment like CPTC, we now turn to the assessment to a real-world SOC dataset that is much more difficult to process and interpret.

4.4.2 HeATing a Real-World SOC OP

Our SOC data-set contains 30 days total worth of Suricata alerts from a medium-sized US-based university; in this work we process one week of this data. Assume the role of the SOC analyst of this network where there are 1000's of daily users generating mostly benign traffic. It is your job to identify first if there are potential threats and then second identify each step the adversary took to identify the attack campaign. Table 4.7 demonstrates the challenges SOC analysts have with assessing IDS traffic and why so many attacks go unnoticed until it is too late. Ideally SOC operations are a 24/7 job, however this is not the case in reality and the analyst is constantly attempting to catch up. If we focus on the signatures that Suricata defines as "high severity" (severity=1) as "critical", we find that a significant portion of the unique alerts per day are critical. Not all of these will be actually severe and lead to an attack campaign; distracting the analyst from discovering the true critical attack campaign. We demonstrate how HeAT can be used to aid the SOC analyst using the

observations previously seen in CPTC, a data-set where it was much more obvious to determine adversarial activity.

Table 4.7: Daily summary of alert traffic for the SOC data-set where alert traffic is at a constantly high volume. *Weekend Traffic

Day	Total Alerts	Total Episodes	Unique Sources	Unique Targets	Unique Signatures	"Critical" Signatures
1*	**894,915**	77,202	3,071	**195,553**	123	32
2*	664,754	74,396	2,946	187,452	104	33
3	620,202	94,408	5,263	171,006	145	48
4	773,622	**97,276**	5,521	189,877	145	48
5	532,182	89,919	**5,357**	163,818	**149**	**60**
6	549,498	87,929	5,072	165,265	125	37
7	646,699	83,409	4,860	186,681	130	45

Other than obvious difference in scale compared to the CPTC data, we apply some extra processing to this data set to improve scalability. First, IDS for this data is placed at the actual "edge" of the network where the "sources" in this data are from external sources as opposed to CPTC where the traffic was from all internal competitors. As an externally-facing network is subjected to constant probes from the internet, both benign and malicious, we foresee our source-based aggregation to struggle to scale to this level. Our IP addresses are also sanitized making it more challenging to know where/who is the attack is originating from. We instead use the Autonomous System Number (ASN) of the sources to perform our aggregation on as we are able to distinguish external sources from internal sources and group sources from roughly the same physical location. Also some types of malware results in internal IP's to appear as a source to the IDS to call out to a malware control server, in these cases we swap the source and target IP's of these alerts to represent the true adversary and victim IP. Table 4.8 demonstrates that using ASN for our aggregation significantly reduces the number of episodes the analyst will have to assess.

Table 4.8: Source-ASN based aggregation is recommended for SOC data as it combines the activities from sources of similar location.

Source IP Episodes	Source ASN Episodes	Total Reduction
604,537	336,822	-44.3%

This only changes how the alerts are aggregated into episodes and the real source and target IP's

are maintained in the episode. Given that we use network-agnostic features, our process can assess the attack campaigns with respect to the analyst's training triages regardless of how the episodes are represented. Using a set of critical signatures, we demonstrate how HeAT can use observations from CPTC (source IP based) to assess this real-world set.

Case Study: HeATing the SOC

Our analysis begins with the selection of our IoC's and for this experiment we choose examples of signatures that contain a CVE number as the impact of these actions is much more well documented. Demonstrated in Table 4.9, once again the scale of this data set poses a challenge for the analyst as there are 1000's of occurrences for these signatures each treated as an individual IoC. From the minimum and maximum HeAT gain for each of these signatures, we find that not all of these lead to any significant attack campaigns. Looking at the HeAT-Gain of all IoC's in Figure 4.7, we confirm that we would prioritize the attack campaigns with the highest gain as these also have a diverse set of attack stages represented. We find that the AIS coverage term of the gain enables us to further prioritize the HAC's with a high amount of attack stage coverage, where we find only a handful of the hundreds of possible HAC's have a significant amount of AIS coverage. We found that 20 out of the top-25 HeAT-gain HAC's had the highest AIS coverage which we believe is most likely to contain the most significant attack campaigns. We demonstrate this by assessing the HAC's with the highest overall HeAT-gain. Without HeAT and this metric, it would be substantially more difficult to identify which occurrence of the alert should be assessed. Each of these 1000's IoC's may still be related but the highest HeAT-Gain HAC is most likely to completely describe each stage of a successful attack scenario.

We first turn to the alert of CVE-2020-5902 which describes a remote code execution (RCE) vulnerability on F5's enterprise traffic manager, load balancer, and DNS. This is a particularly critical signature if successful as the adversary could have nearly full control over the flow of traffic within the network and could redirect traffic wherever they desire. With our insights AIS coverage, Figure 4.8 displays the HAC with the highest AIS coverage and overall gain compared to all episodes occurring 48-hours prior to the critical event.

Again it is immediately apparent of the importance of automating the triage process as the scale of the network increases, as few episodes actually contribute to the attack campaign. With HeAT we

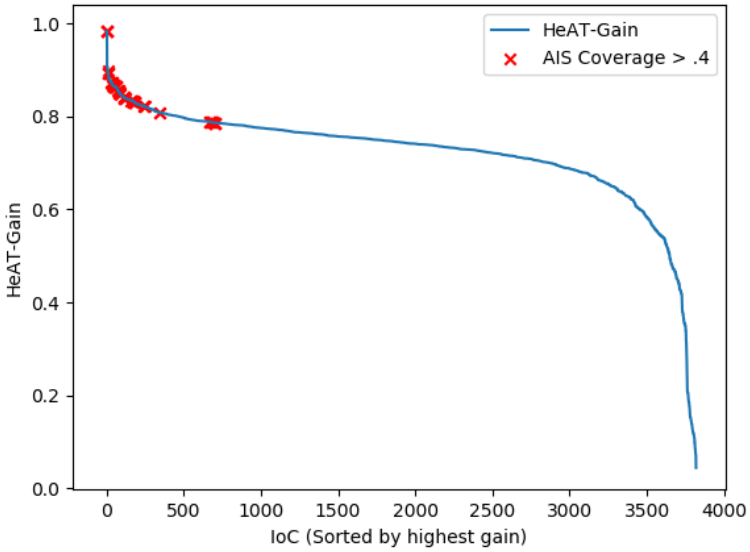

Figure 4.7: High HeAT-Gain attack campaigns also have high AIS coverage making it more likely to reveal a complete attack campaign

Table 4.9: Out of the hundreds of occurrences of an signature, only few have significant AIS coverage leading towards meaningful attack campaigns.

Signature Description	Total	Max Δ_h	Min. Δ_h	Count of HAC with AIS Coverage		
				$>.2$	$>.3$	$>.4$
ET WEB_SPECIFIC_APPS Drupalgeddon2 <8.5.1 RCE Through Registration Form (CVE-2018-7600)	808	.9834	.085	766	349	7
ET EXPLOIT F5 TMUI RCE vulnerability CVE-2020-5902	866	.9834	.076	812	360	7
ET EXPLOIT Possible CVE-2013-0156 Ruby On Rails XML POST to Disallowed Type YAML	881	.98	.044	810	355	7
ET WEB_SERVER Possible CVE-2014-6271 Attempt	379	.98	.316	360	118	5
ET EXPLOIT Cisco ASA/Firepower Unauthenticated File Read (CVE-2020-3452)	885	.906	.072	830	365	7

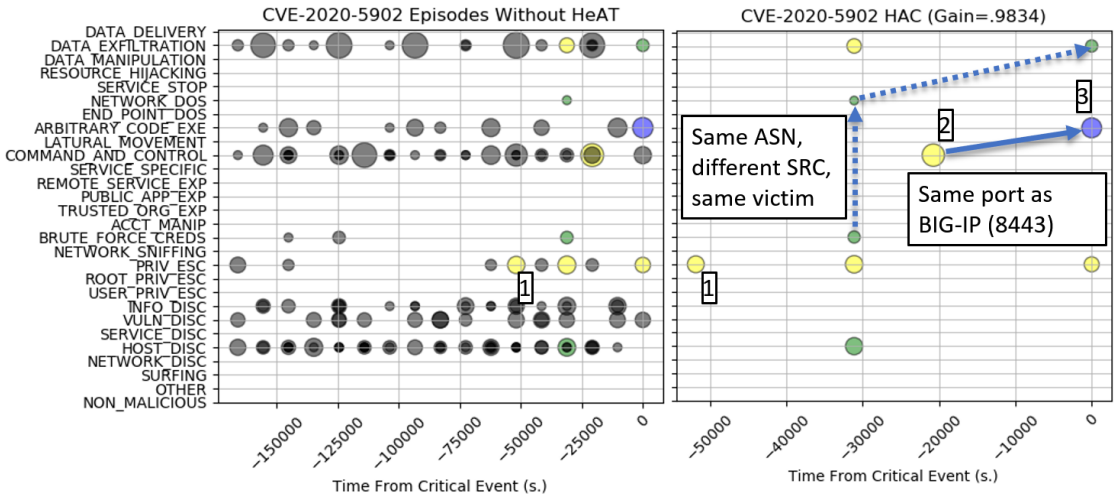

Figure 4.8: HAC with a minimum HeAT of .15 (right) substantially reduces irrelevant alerts and revealing concerning characteristics about prior episodes.

can immediately turn out focus on the HeATed episode immediately prior to the critical IoC where we find lateral movement and exploitation on the same port of alleged victim service. Following the annotations in the figure, we make the following observations immediately given this HAC:

1. The first evidence of root-privilege escalation on the victim

2. Command and control activity on the same source and victim using the same port number as the IoC. This is the only HAC for this IoC to contain this specific episode.

3. The IoC episode with the same destination port as the C2, this is likely to be compromised. There is other exfiltration activity targeting the same victim from the same ASN (different true source IP) at the same time.

This HAC provides an exceptional starting point for investigation as the C2 behavior does indicate some level of compromise and we should investigate the initial privilege escalation further. Whereas if we look at the same episode without HeAT, we see that there were 3 other episodes containing over 100 alerts occurring nearly at the same time of the C2. This would make alert-based analysis significantly more difficult. Even at the episode level, individual analysis of each episode for each IoC would be exhausting for an analyst to perform as there are many irrelevant periodic episodes. With HeAT and HeAT-gain, we were able to quickly identify suspicious episodes that would need further investigation.

We also find other interesting behaviors from the same ASN, but different true source IP, targeting our victim at the same time. These episodes were appropriately given a lower HeAT value than the episodes that did share the same true source and target IP's and we find it suspicious that the "data delivery" episode occurs at the same time as our IoC on the victim. These are observations that may have been missed or obscured if we were using the IP-based "attack campaigns" instead. Unlike the CPTC data the IP-based attack campaigns are much less useful due to volume of data and potentially-large IP-space as demonstrated by Figure 4.9.

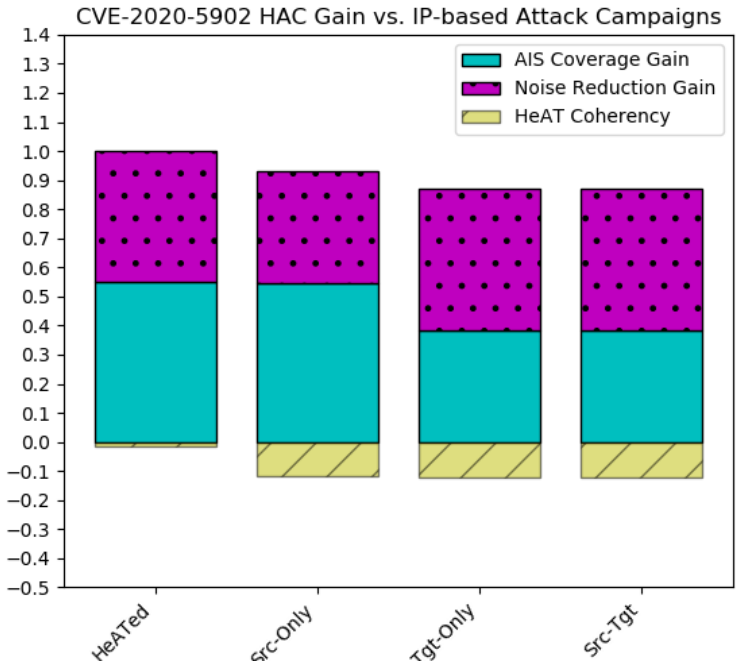

Figure 4.9: The HeATed representation provides the most amount of AIS coverage, has the most amount of noise reduction, and has an accurate HeAT coherence.

We found the IP-based attack campaigns to contain many episodes from periodic and irrelevant alerts that we believe would initially distract the analyst from the true attack campaign. This is reflected by the lower amount of noise reduction and the AIS coverage entropy being bias towards the noisy periodic alerts when compared to the HeATed example. Given the sheer number of the potential attack campaigns, the time taken addressing these distractions will significantly add to the time needed to respond to the attack.

We also find that for the HeATed representation the HeAT coherency term is near-zero meaning that the HeAT values displayed are an accurate reflection of the HeAT observed by the analyst in

prior triages. High AIS coverage and noise reduction combined with near-zero coherency means that we have a diverse and noise free attack campaign that is tailored to the thinking of the analyst. For each of the aforementioned signatures we find that as AIS coverage increases, our HeAT coherency will typically decrease as our HAC's have more meaningful attack stages. We show this in Table 4.10 where we show the average HeAT coherency for HAC's of various AIS coverage.

Table 4.10: As the AIS coverage increases for the HAC's, the HeAT model can more accurately apply HeAT from prior triages.

Signature Description	Average HeAT Coherency Given AIS Coverage:		
	$<.2$	$.2<\delta_{ACG}<=.4$	$>.4$
Drupalgeddon2 <8.5.1 RCE Through Registration Form (CVE-2018-7600)	.217	.0857	.0745
F5 TMUI RCE vulnerability CVE-2020-5902	.198	.0962	.0756
CVE-2013-0156 Ruby On Rails XML POST to Disallowed Type YAML	.225	.0861	.0607
Possible CVE-2014-6271 Attempt	.181	.0865	.0598
Cisco ASA/Firepower Unauthenticated File Read (CVE-2020-3452)	.251	.0659	.0754

Our HeAT gain metric is reporting that our most meaningful attack campaigns are also best representing the analyst's impressions of important actions to be included in the attack campaign. Given that we used observations from another network with significant differences in behavior, we wonder if the observations from CPTC are sufficient to reveal attack campaigns within the SOC data or if additional test-network specific data is required. The above attack campaigns used CPTC observations with a small amount of additional observations from the SOC network and we demonstrate that our network-agnostic features do allow for significant transferability of observations but even better results can be achieved with additional data.

Transferability of Attack Campaigns

Given that identifying and evaluating attack campaigns on CPTC is relatively straightforward, it would be advantageous if we can capture the analyst's HeAT input from CPTC and apply it to other scenarios. As mentioned before, CPTC may not be the most realistic representation of real-world attack campaigns however we believe the indications/characteristics of prior attack campaigns through network-agnostic features can aid in transferring the analyst's prior observations. Due to

the vast difference in alert volume, network architecture, and attacker behavior between the CPTC and the SOC data-set, it is we assume that additional observations from the SOC data-set will result in identifying more meaningful attack campaigns tailored for the SOC network. Assume the scenario where the observations from the CPTC set are provided to the user initially and we ask the analyst to first provide HeAT observations on the new network under test. Using the insights from the previous experiments and the HeAT Coherency term of HeAT-Gain, we demonstrate how these new additional observations enable the "fine-tuning" of the CPTC observations to the new network architecture resulting in the extraction of attack campaigns from SOC data sets.

Given the size of our SOC data, the selection of the IoC that results in the discovery of an actual attack campaign to be labelled may be unlikely. We believe additional labelled data regardless if it represents actual adversarial actions allows for the HeAT model to better distinguish between irrelevant noise and the more critical attack campaign characteristics as observed in the CPTC data. Given at the time that we also had very little knowledge about potential attacks within the SOC data, we selected a signature that we felt was important to assess and include as our own critical SOC observations. The signature "*ET EXPLOIT Possible ETERNALBLUE Probe MS17-010*" allows for arbitrary code to be executed through a SMB-share typically leading to the installation of ransomware. We triaged two instances of this signature where we labeled 125 prior episodes with our reflection of the appropriate HeAT value given the relationship between the prior alert and the IoC and our own experiences of if the episode is likely to be a part of the same attack campaign. This is a small fraction of the 1,700+ observations we made for CPTC and we now demonstrate how the new observations allows the model to more accurately apply HeAT and expose more meaningful attack campaigns with reduced noise.

Using the high AIS coverage HAC's discovered previously in Table 4.9, we compare the HeAT-Gain of each attack campaign before and after the SOC observations were added to the HeAT model. Recall that when our HeAT coherency term is near zero the heat-value given the attack stage is reported consistently between our labelled observations and the generated HAC. Figure 4.10 demonstrates for the "Drupalgeddon2" CVE-2018-7600 signature the additional SOC observations reduces the coherency to near-zero consistently and in most cases reports a higher gain than using only CPTC observations. We also find that some cases the CPTC-only observations perform comparatively and there are very few differences between the attack campaigns. So we ask: "what

attack campaign characteristics are getting transferred over?"

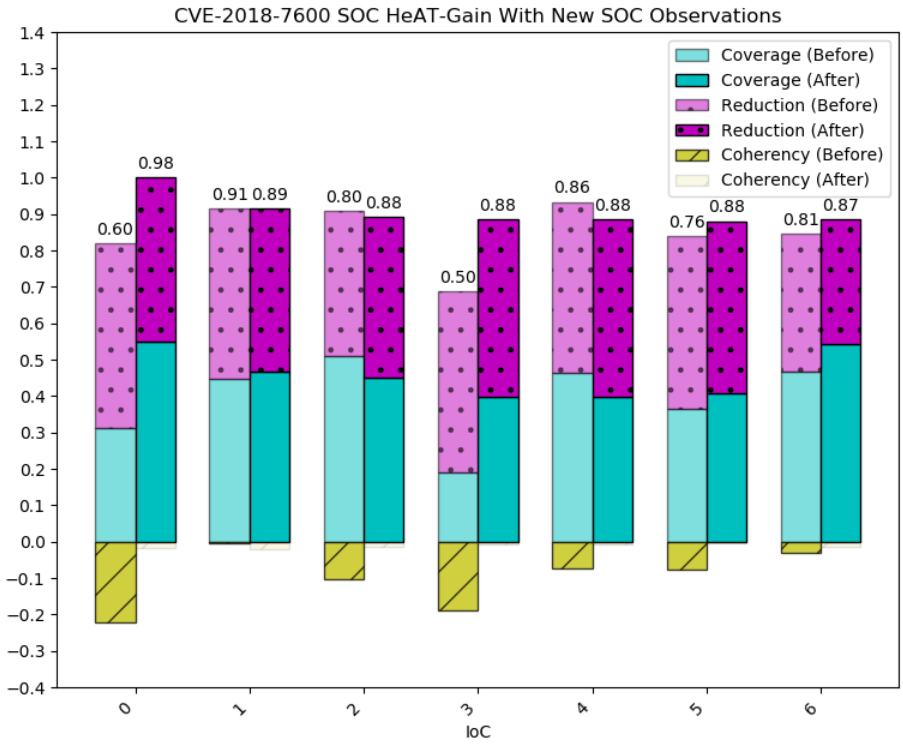

Figure 4.10: HAC's discovered with additional SOC data consistently reduces the HeAT coherency to near-zero resulting in a more accurate reflection of the true attack campaign contribution.

Recall that in our CPTC experiments we made the observation that the model was biased to episodes that had the same source and target IP address of the IoC. In CPTC it was common to see episodes with only one source and target IP whereas our SOC data it is common to observe a single episode with multiple sources and many targets. This is a result of us finding that filtering by source-ASN was much more appropriate to represent and aggregate the vast volume of SOC alerts. Remember that our network-agnostic features evaluate the *ratio* of common IP addresses. We found in cases where there was a large discrepancy between in the HeAT coherency, such as IoC 0 and 3 in Figure 4.10, the CPTC-only model would apply high heat to episodes where there was almost an exact match between the source and target IP. Where the additional SOC observations were able to bring up the HeAT-value for cases where the true victim IP address was one of many that where targeted within that episode. This would not be unusual as with a network of this scale, many adversaries may target many victims in the hopes of gaining access to just one. This result

demonstrates that a small number of additional ASN-based observations enable our IP-based observations in CPTC to be accurately applied and find attack campaigns within network data multiple orders of magnitude larger.

IoC's 1, 4, and 6 demonstrate cases where only the CPTC observations are enough to accurately capture some attack campaigns, with 4 actually reporting a higher gain than if we supplied additional data. These campaigns were a small reduction compared to the naive attack campaigns where we only consider the source-target IP matches of the IoC, meaning that there is not much more for HeAT to resolve. IoC 4 reports a higher gain using CPTC-only due to applying heat to episodes that are identified as a false positive attack response that triggers a scanning type alert at the same time as the IoC. Our additional data has made those types of alerts low heat and thus ignored as reflected by the lower coverage and slightly higher noise reduction. This led us to finding another bias, the CPTC data model is biased towards applying HeAT of episodes with close time proximity to the IoC and applying high HeAT to benign signatures.

We investigated the behavior of HeAT for all unique signatures, including non-malicious signatures, as we expect lesser critical signatures to have little HeAT. The signatures with the top-5 HeAT-Gains for the CPTC+SOC Observations are the 5 signatures shown in Table 4.9, each of which could be considered significant given that there is a unique CVE identifier. When we performed the same process using only the CPTC-observations, we found disproportionately high gain attack campaigns being identified for seemingly benign signature descriptions. For example the top-3 signatures with the highest gain given the CPTC-only observations consisted of: 1) *ET WEB_SERVER Suspicious Chmod Usage in URI (Inbound)* (Δ=1.04), 2) *ET WEB_SERVER 401TRG Generic Webshell Request - POST with wget in body* (Δ=1.01), and 3) *ET WEB_SERVER WebShell Generic - wget http - POST* (Δ=.99). Shown in Figure 4.11, we found the episodes contained within HAC's of these IoC's to all occur concurrently for a variety of attack stages indicating that this alert is triggered as a response to another action and our CPTC data is biased towards these episodes. We find that our additional SOC observations to eliminate these biases and further provide necessary noise reduction tailored for this network architecture; each of these IoC's now have a maximum gain of .6. This is a symptom of using non-realistic competition data sets as our starting point however this demonstrates that these observations from CPTC are still meaningful and can be adapted to be even more useful with a small amount of additional observations.

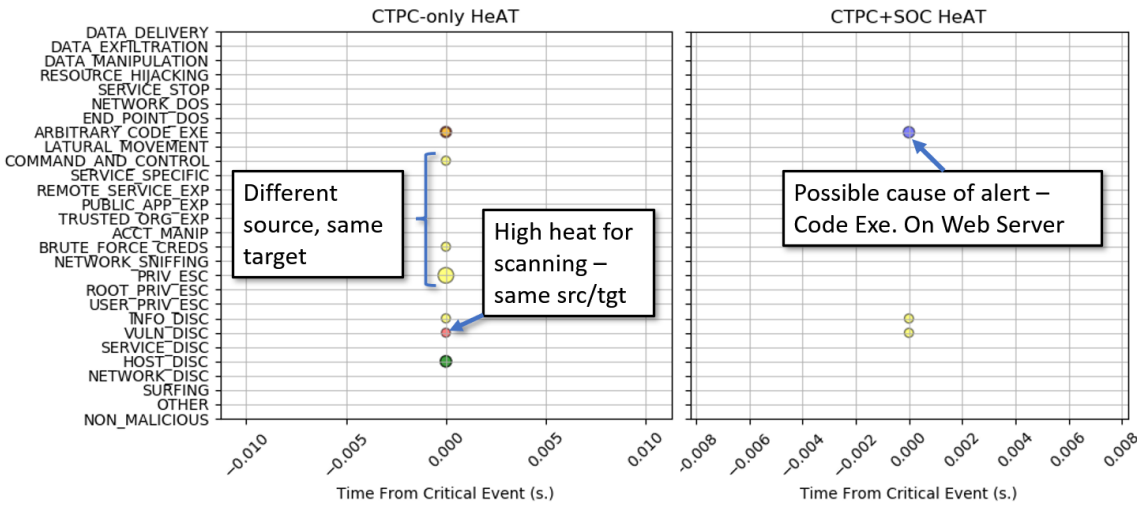

Figure 4.11: Additional network-specific observations enables HeAT to account for attack campaign biases that may be present in competition data sets like CPTC.

We demonstrate that despite the vast differences between these data sets in not only volume but also the prevalence of adversarial attack campaigns captured, attack campaigns observed in other networks have meaningful characteristics that can be used to discover other attack campaigns. It comes no surprise that attack campaigns observed within a highly-controlled competition does not entirely portray the same type of attack scenarios present within a large-scale SOC. HeAT with our network-agnostic features enables these observations in controlled environments to be leveraged and further enhanced using few SOC-specific observations. In a field where labelled data describing adversarial attack campaigns is scarce, it is extremely beneficial that data where it is significantly easier to discover attack campaigns can be used to investigate other scenarios where it may be extremely difficult to find legitimate examples of attacks.

4.4.3 Conclusion and Limitations

HeAT and its ability to capture the analysts opinion of events likely to contribute to an attack campaign and apply that knowledge to discover attack campaigns in other networks is a unique approach to combat the reliance on large labelled data sets. In a field where there is significant privacy concerns with the simply the access and usage of unlabelled data sets containing adversarial behavior, we find it advantageous that competition data like CPTC can indeed be used to find other attack campaigns within the SOC data set with minimal extra effort needed. We envision the scenario

where HeAT can provide initial professionally labelled data that is dense with actual attack campaign information and ask the end user to provide additional data to fine-tune the model for their network. This eliminates the privacy concerns when using observations from other networks as our episode features never contain any specific information either network while maintaining critical attack campaign indicators. HeAT is unique in the sense that few works demonstrate their ability to provide meaningful results in both highly controlled scenarios and in a real-world scenario where little is known. Throughout this work we have identified methodical limitations of HeAT we think are appropriate to address when developing future methods to extract attack campaigns.

Initial triage quality directly impacts the overall utility of HeAT. To combat the lack of labelled data, we chose to have the option where the analyst can self-label data using HeAT and use that information to find similar observations. This is inherently dangerous as two analysts may have a different opinion of the critically of an action depending on the experience level of the analyst or their own instinct. Issues could be compounded in the event the analyst's opinion is objectively wrong or if the attack type has not been observed previously, leading to HeAT potentially miss-representing a critical event. Ideally these initial triages would be done by multiple security professionals of various experience-levels and on many attack campaign types to obtain the best possible initial observations to train HeAT. This is why we propose the concept of providing a "pre-trained" CPTC model as a reference to actual attack scenarios and then have the analyst fine-tune to their specific network. We believe the fine tuning process be taken further and we propose using the concept of active-learning to use generated HAC's as additional training data if the analyst deems them as accurate.

Scaling to SOC has exposed some limitations in some of our aggregation process and assumptions about attack campaigns. We found that at the enterprise-level our source IP-based alert aggregation still resulted in an overwhelming number of episodes and we had to turn to ASN-based filtering to further aggregate alerts to a manageable. Certain attack types such as distributed denial of service (DDoS) and botnets along with identity obfuscation techniques such as VPN's/Tor and IP/Mac spoofing may result in the inaccurate representation of the real attack episode. Assuming that our aggregation process correctly represents an attack action correctly, our original definition of an IoC turned out to be oversimplified. In the CPTC, the more "critical" alert signatures would only

be observed about 10 times per team, which made the assessment of each of these cases individually not too challenging. As we demonstrated in the SOC set, a critical signature may be observed 1000's of times per week and the issues with choosing individual examples of signatures as the IoC became apparent. HeAT-Gain made it significantly easier to distinguish the actually important-to-assess IoC's however we think we need to evaluate: "what actually is an IoC?" Typically an IoC would be a data leak, ransomware, or some denial of service and only then the analyst would look for evidence with in the IDS. Our current definition does allow us to pre-assess for threats however for additional flexibility and utility it may be beneficial to have a more robust definition of IoC types.

HeAT-Gain describes desirable properties of attack campaigns but does not infer on the actual quality of the attack campaign. HeAT-Gain proved itself to be a helpful metric to aid the analyst to prioritize the assessment of attack campaigns based off the coverage of attack stages, noise reduction, and accurately applying HeAT to episodes. As we saw with only using CPTC observations on the SOC, the HeAT-Gain for non-malicious signatures were very high due to high attack stage coverage all occurring at the same time. We do not believe that "attack campaigns" that have no variance in time regardless of the attack stage coverage should be rewarded with high gain. HeAT-gain does not include any notion of time, which may mislead the analyst into assessing less interesting attack campaigns. The unusually high gain was due to insufficient data to represent the characteristics of the SOC set and our attack stage-based metric was negatively impacted. A time-based term could be added into the metric however aside from the aforementioned situation, time-variance does not actually play a large role in how interesting an attack campaign is. An advanced persistent threat (APT) could conduct a similar attack campaign to other attackers with the only difference being the APT took months. Instead we would build additional network evidence into our metric such as recognizing access to a machine that can communicate with the victim or identifying if the episode targets a service that is actually running on the machine. These sorts of network-specific metrics would be able to clarify if the attack campaign did actually happen and thus recognize if the HeAT model is correctly identifying real attack campaigns.

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.

5. Conclusion

HeAT-PATRL is one of the few works in the cyber-research community that considers the limited resources of analysts applying such methods while still recognizing that the individual analyst's experience and knowledge is a critical asset for identifying attack campaigns. We accepted the facts of labelled data sets being limited, IDS alerts contain a limited amount of relevant attack attributes, IDS's produce excessive amounts of alerts, and that verifying our processes in real-world applications may be extremely difficult. We took a novel approach keeping each of these challenges in mind and addressed them using advanced machine learning architectures and techniques, attack stage context-based alert aggregation, network-agnostic feature engineering, SOC analyst surveys, and created metrics to estimate our performance without ground truth. HeAT-PATRL describes an end-to-end process to take a set of raw IDS alerts, define them in-terms of attack stages, and reveal attack scenarios on real-world SOC data sets based on the thought process of what the analyst defines an attack campaign to be.

We demonstrated through our HeAT-Gain metric that HeAT-PATRL is able to leverage campaign triages from cyber-attack competition to reveal significant attack campaigns in a real-world data set and maintain the analyst's input throughout. Due to the differences in alert volume between our competition set and our real-world data set, we found that a small amount of additional "real-world" attack campaign observations were needed to adapt HeAT to the new network-context. This enabled HeAT-Gain to better reveal and prioritize attack campaigns that are most severe as the HeAT-Coherence indicated that HeAT is matching the analyst's initial input. These campaigns with also were likely to contain the entire attack scenario from start to finish given the high AIS-Coverage and noise reduction. The additional data required to transfer analyst campaign observations from a competition environment to a real-world network was less than 1% of the competition triage. We find this extremely advantageous as the attack campaigns within the competition data are significantly easier to observe and realize than in a real network; indicating that we can provide the competition-derived campaign observations and have users of HeAT-PATRL to fine-tune the model with their own network-specific observations.

Aside from the limitations described in each of the individual works, throughout this dissertation we have learned some lessons learned that may have changed our approach if realized earlier. These lessons stem from our approach were we were initially provided the IDS alerts from the competition and from there we theorized how to find attacks within it, playing the role of the SOC analyst but we wanted to automate it. Because of this, our lessons are around the limitations data itself and the finite information it can represent.

Competition data sets may mislead research into findings that may not translate into real-world applicable applications. Like many others, we found the usage of competition data sets from either academic sources or security conferences to be appealing because it is safe to assume that most of the traffic observed is likely to be the result of adversarial behavior. As we showed with simply filtering by an IoC's source and target IP, manually discovering attack campaigns within these sets was mostly straightforward and thus it was not difficult for a machine learning model to do the same. It was not until we started to assess the SOC data set did we realize significant scalability issues with our aggregation process, IoC definition, and feature descriptions. Had we started with the intentions of only using unlabelled SOC data we could have accounted for some of these scalability issues but would had a difficult initially verifying our findings as we knew little about our data. It should be recognized that these competition data sets are valuable and can be used to learn about certain types of attacks, however we advise that they are not relied on if the objective is to be applied to real-world data.

IDS alerts alone do not provide enough information about the actions of the adversary to definitively prove the attack campaign actually was successful. We started with the objective of extracting attack campaigns using only network-based IDS alerts as they are commonly deployed and often is where analysts look first if they suspect that there is an security issue on their network. These IDS's are verbose, only considers the information within the incoming packet, and has no knowledge of if the alert that is describing was actually successful or not. Other host-based IDS's or sensors along with techniques described in "alert correlation" can be used to provide additional evidence to determine if the action was successful, however this adds another layer of complexity that may or may not be able to be made network-agnostic. We could also infer on the success of the adversary if we had detailed network knowledge for each network. For example if we had access to the network's domain controller, we would be able to know the connectivity of the machines, firewall policies,

access control, services running, etc. to determine if the alert(s) being described are even possible given the configuration. Given the dynamic nature of the machines on a network are, we do not think it is ever likely that this detailed network description will be available to us or even most network administrators. Assuming we do have access to this information, we wonder how specific network information like access policies and user information can be described network-agnostically so that the information can be leveraged across networks. For the future, we look for methods to confirm each stage within the attack campaign for their likelihood to be successful given the network.

Despite the limitations in the quality our data, we demonstrate that much can be learned about the adversarial behaviors and attack campaigns with out knowing much about the adversary themselves. Cyber-attacks are not going to stop or even slow down in the foreseeable future; coming from all over the world, use extremely complex techniques, and while targeting nearly anyone who is vulnerable. It is unlikely that we will ever be able to interface with these adversaries to learn more about their methods, workflow, and thought process. We use IDS's, cloud-based security products, penetration tests, etc. to discover and mitigate threats as soon as possible, however these are still not enough. Research into detecting, understanding, and describing attacker behaviors through the lens of defense, like HeAT-PATRL, is imperative. The reality is that breaches will happen and damage can be mitigated if we can quickly identify *what* happened and *how* it happened as described by each step of the adversaries attack campaign.

5.1 Source Code

Source code for PATRL and HeAT will be made available for research purposes on: `https://github.com/smoskal`. The current iteration and description of the Action-Intent Framework can be seen in [42]. PATRL and HeAT is built on Python 3.8 and will require a GPU capable of using Tensorflow r1.15. Additional inquiry on either of these works should be directed through email (sfm5015@rit.edu) or through LinkedIn: `https://www.linkedin.com/in/stephen-moskal/`

Bibliography

[1] ”Jkurucar” ”KirstenS” ”Wichers” and ”kingthorin”. *Intrusion Detection*. https://owasp.org/www-community/controls/Intrusion_Detection. [Online; accessed 17-November-2021]. 2020.

[2] Open Information Security Foundation (OISF). *Home - Suricata*. https://suricata.io/. [Online; accessed 21-June-2021]. 2021.

[3] 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/. [Online; accessed 21-June-2021]. 2011.

[4] Faeiz M Alserhani. “Alert correlation and aggregation techniques for reduction of security alerts and detection of multistage attack.” In: *International Journal of Advanced Studies in Computers, Science and Engineering* 5.2 (2016), p. 1.

[5] Fernando Alves et al. *Processing Tweets for Cybersecurity Threat Awareness*. 2019. arXiv: 1904.02072 [cs.CR].

[6] Simon Bachstein. *Uncertainty Quantification in Deep Learning*. https://www.inovex.de/blog/uncertainty-quantification-deep-learning/. [Online; accessed 20-Nov-2020]. 2019.

[7] S Barnum. “Common attack pattern enumeration and classification (capec) schema description.” In: *Cigital Inc, http://capec. mitre. org/documents/documentation/CAPEC_Schema_Description_v1* 3 (2008).

[8] Adrian Calma, Tobias Reitmaier, and Bernhard Sick. “Semi-supervised active learning for support vector machines: A novel approach that exploits structure information in data.” In: *Information Sciences* 456 (2018), pp. 13–33.

[9] Yuzhong Chen et al. “Distributed Attack Modeling Approach Based on Process Mining and Graph Segmentation.” In: *Entropy* 22.9 (2020), p. 1026.

[10] *CVE - Common Vulnerabilities and Exposures*. https://cve.mitre.org/. [Online; accessed 5-May-2020]. 2020.

[11] *Cyber Kill Chain — Lockheed Martin Security*. http://cyber.lockheedmartin.com/solutions/cyber-kill-chain. [Online; accessed 11-April-2016]. 2016.

[12] Darktrace. *Darktrace Cyber AI Analyst: Autonomous Investigations*. https://www.darktrace.com/en/resources/wp-cyber-ai-analyst.pdf. [Online; accessed 21-June-2021]. 2021.

[13] Sean Carlisto De Alvarenga et al. "Process mining and hierarchical clustering to help intrusion alert visualization." In: *Computers & Security* 73 (2018), pp. 474–491.

[14] Martin Drasar et al. "Session-level Adversary Intent-Driven Cyberattack Simulator." In: *Proceedings of 2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 2020, pp. 1–9. DOI: 10.1109/DS-RT50469.2020.9213690.

[15] Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman. "Alert Correlation in collaborative intelligent intrusion detection systems- A survey." In: *Applied Soft Computing* 11.7 (2011), pp. 4349–4365.

[16] Fast.AI. *Fastai Python Library v1.0.57*. http://docs.fast.ai/. [Online; accessed 28-April-2020]. 2020.

[17] Ouissem Ben Fredj. "A realistic graph-based alert correlation system." In: *Security and Communication Networks* 8.15 (2015), pp. 2477–2493.

[18] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In: *Proceedings of 2016 International Conference on Machine Learning*. 2016, pp. 1050–1059.

[19] Peng Gao et al. "SAQL: A Stream-based Query System for Real-Time Abnormal System Behavior Detection." In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 639–656. ISBN: 978-1-931971-46-1. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/gao-peng.

[20] Ibrahim Ghafir et al. "Hidden Markov models and alert correlations for the prediction of advanced persistent threats." In: *IEEE Access* 7 (2019), pp. 99508–99520.

[21] Frank Hassanabad. *suricata-sample-data*. https://github.com/FrankHassanabad/suricata-sample-data/blob/master/README.md. [Online; accessed 5-May-2020]. 2019.

[22] Steven CH Hoi and Michael R Lyu. "A semi-supervised active learning framework for image retrieval." In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, pp. 302–309.

[23] Sarah Hospelhorn. *What is The Cyber Kill Chain and How to Use it Effectively*. https://www.varonis.com/blog/cyber-kill-chain/. [Online; accessed 14-Feb-2020]. 2018.

[24] Jeremy Howard and Sebastian Ruder. "Universal language model fine-tuning for text classification." In: *arXiv preprint arXiv:1801.06146* (2018).

[25] Neminath Hubballi and Vinoth Suryanarayanan. "False alarm minimization techniques in signature-based intrusion detection systems: A survey." In: *Computer Communications* 49 (2014), pp. 1–17.

[26] IBM. *IBM QRadar Advison with Watson*. https://www.ibm.com/products/cognitive-security-analytics. [Online; accessed 21-June-2021]. 2021.

[27] Zakira Inayat et al. "Intrusion response systems: Foundations, design, and challenges." In: *Journal of Network and Computer Applications* 62 (2016), pp. 53–74.

[28] Mordor Intelligence. *CYBERSECURITY MARKET - GROWTH, TRENDS, COVID-19 IMPACT, AND FORECASTS (2021 - 2026)*. https://www.mordorintelligence.com/industry-reports/cyber-security-market. [Online; accessed 17-November-2021]. 2021.

[29] Arnav Joshi et al. "Extracting cybersecurity related linked data from text." In: *Proceedings of 2013 IEEE Seventh International Conference on Semantic Computing*. IEEE. 2013, pp. 252–259.

[30] Klaus Julisch. "Mining alarm clusters to improve alarm handling efficiency." In: *Proceedings of Computer Security Applications Conference*. IEEE. 2001, pp. 12–21.

[31] Nattawat Khamphakdee, Nunnapus Benjamas, and Saiyan Saiyod. "Improving Intrusion Detection System Based on Snort Rules for Network Probe Attacks Detection with Association Rules Technique of Data Mining." In: *Journal of ICT Research & Applications* 8.3 (2015).

[32] Max Landauer et al. "A framework for cyber threat intelligence extraction from raw log data." In: *Proceedings of 2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 3200–3209.

[33] Quentin Le Sceller et al. "SONAR: Automatic Detection of Cyber Security Events over the Twitter Stream." In: *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ARES '17. Reggio Calabria, Italy: Association for Computing Machinery, 2017. ISBN: 9781450352574. DOI: 10.1145/3098954.3098992. URL: https://doi.org/10.1145/3098954.3098992.

[34] Dong-Hyun Lee. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks." In: *Proceedings of ICML 2013: Workshop on challenges in representation learning*. Vol. 3. 2. 2013.

[35] David D Lewis and William A Gale. "A sequential algorithm for training text classifiers." In: *Proceedings of SIGIR'94*. Springer. 1994, pp. 3–12.

[36] Zhun Li, ByungSoo Ko, and Ho-Jin Choi. "Naive semi-supervised deep learning using pseudo-label." In: *Peer-to-Peer Networking and Applications* 12.5 (2019), pp. 1358–1368.

[37] Zi Long et al. "Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling." In: *Proceeedings of 2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[38] Federico Maggi, Matteo Matteucci, and Stefano Zanero. "Reducing false positives in anomaly detectors through fuzzy alert aggregation." In: *Information Fusion* 10.4 (2009), pp. 300–311.

[39] MITRE. *MITRE ATT&CK*. `https://attack.mitre.org/`. [Online; accessed 23-April-2019]. 2018.

[40] S. Mittal et al. "CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities." In: *Proceedings of 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2016, pp. 860–867.

[41] Benjamin Morin et al. "A logic-based model to support alert correlation in intrusion detection." In: *Information Fusion* 10.4 (2009), pp. 285–299.

[42] Stephen Moskal and Shanchieh Jay Yang. "Framework to Describe Intentions of a Cyber Attack Action." In: *arXiv preprint arXiv:2002.07838* (2020).

[43] Stephen Moskal, Shanchieh Jay Yang, and Michael E Kuhl. "Cyber Threat Assessment Via Attack Scenario Simulation Using an Integrated Adversary and Network Modeling Approach." In: *The Journal of Defense Modeling and Simulation* 15.1 (2018), pp. 13–29.

[44] Stephen Moskal, Shanchieh Jay Yang, and Michael 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)*. IEEE. 2018, pp. 49–54.

[45] Azqa Nadeem et al. "Alert-driven Attack Graph Generation using S-PDFA." In: *Accepted to Appear in 2021 ACM KDD Workshop on Artificial Intelligence-enabled Cybersecurity Analytics (AI4Cyber)* (2021).

[46] Julio Navarro et al. "Huma: A multi-layer framework for threat analysis in a heterogeneous log environment." In: *International Symposium on Foundations and Practice of Security*. Springer. 2017, pp. 144–159.

[47] Julio Navarro et al. "OMMA: open architecture for Operator-guided Monitoring of Multi-step Attacks." In: *EURASIP Journal on Information Security* 2018.1 (2018), pp. 1–25.

[48] Centripetal Networks. *AI-Analyst*. `Centrcentripetalnetworks.com/hubfs/Data\%20Sheets/CI_AI_Analyst_Brief.pdf`. [Online; accessed 21-June-2021]. 2018.

[49] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[50] Paul Pols. *Unified Kill Chain: Raising Resilience Against Cyber Attacks*. https://www.unifiedkillchain.com/. [Online; accessed 17-November-2021]. 2021.

[51] The Zeek Project. *The Zeek Network Security Monitor*. https://zeek.org/. [Online; accessed 17-November-2021]. 2021.

[52] Xinzhou Qin and Wenke Lee. "Discovering novel attack strategies from INFOSEC alerts." In: *Proceedings of 2004 European Symposium on Research in Computer Security*. Springer. 2004, pp. 439–456.

[53] Alan Ritter et al. "Weakly Supervised Extraction of Computer Security Events from Twitter." In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 896–905. ISBN: 9781450334693. DOI: 10.1145/2736277.2741083. URL: https://doi.org/10.1145/2736277.2741083.

[54] Rochester Institute of Technology. *Collegiate Penetration Testing Competition*. hhttp://nationalcptc.org. [Online; accessed 19-July-2018]. 2018.

[55] Sebastian Roschke, Feng Cheng, and Christoph Meinel. "A new alert correlation algorithm based on attack graph." In: *Computational intelligence in security for information systems*. Springer, 2011, pp. 58–67.

[56] Reza Sadoddin and Ali Ghorbani. "Alert correlation survey: framework and techniques." In: *Proceedings of the 2006 international conference on privacy, security and trust: bridge the gap between PST technologies and business services*. ACM. 2006, p. 37.

[57] Saeed Salah, Gabriel Maciá-Fernández, and JesúS E DıeAz-Verdejo. "A model-based survey of alert correlation techniques." In: *Computer Networks* 57.5 (2013), pp. 1289–1317.

[58] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models." In: *arXiv preprint arXiv:1708.08296* (2017).

[59] Dell SecureWorks. *Breaking the Kill Chain- Knowing, Detecting, Disrupting and Eradicating the Advanced Threat*. https://www.secureworks.com/assets/pdf-store/other/article-breaking-the-kill-chain.pdf. [Online; accessed 4-April-2016]. 2015.

[60] CCIE Security. *IDS Tuning*. https://www.ccexpert.us/ccie-security/ids-tuning.html. [Online; accessed 17-November-2021]. 2021.

[61] Burr Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[62] Tobias Sterbak. *Model uncertainty in deep learning with Monte Carlo dropout in keras*. https://www.depends-on-the-definition.com/model-uncertainty-in-deep-learning-with-monte-carlo-dropout/. [Online; accessed 20-Nov-2020]. 2019.

[63] STIX. *Kill Chains in STIX*. http://stixproject.github.io/documentation/idioms/kill-chain/. [Online; accessed 4-April-2016]. 2015.

[64] Blake E Strom et al. "MITRE ATT&CK: Design and Philosophy." In: *MITRE Product MP* (2018), pp. 18–0944.

[65] Michael Stute. *Adaptive behavioral intrusion detection systems and methods*. US Patent 8,205,259. June 2012.

[66] Suricata. *10.1. Suricata.yaml*. https://suricata.readthedocs.io/en/suricata-6.0.0/configuration/suricata-yaml.html. [Online; accessed 17-November-2021]. 2019.

[67] Symantec. *Advanced Persistent Threats: A Symantec Perspective*. http://www.symantec.com/content/en/us/enterprise/white_papers/b-advanced_persistent_threats_WP_21215957.en-us.pdf. [Online; accessed 4-April-2016]. 2012.

[68] DTex Systems. *DTex and the Insider Threat Kill Chain*. https://www.dtexsystems.com/wp-content/uploads/2019/04/Dtex-and-the-Insider-Threat-Kill-Chain.pdf. [Online; accessed 17-November-2021]. 2019.

[69] Rochester Insitute of Technology. *Index of cptc2018*. http://mirror.rit.edu/cptc/2018/. [Online; accessed 21-June-2021]. 2021.

[70] Katrin Tomanek and Udo Hahn. "Semi-supervised active learning for sequence labeling." In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pp. 1039–1047.

[71] Fredrik Valeur et al. "Comprehensive approach to intrusion detection alert correlation." In: *IEEE Transactions on Dependable and Secure Computing* 1.3 (2004), pp. 146–169.

[72] Jouni Viinikka et al. "Time series modeling for IDS alert management." In: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. 2006, pp. 102–113.

[73] Chih-Hung Wang and Ye-Chen Chiou. "Alert correlation system with automatic extraction of attack strategies by using dynamic feature weights." In: *International Journal of Computer and Communication Engineering* 5.1 (2016), p. 1.

[74] Fei Wang et al. “Discriminative and Selective Pseudo-Labeling for Domain Adaptation.” In: *Proceedings of International Conference on Multimedia Modeling (2021)*. Springer. 2021, pp. 365–377.

[75] Lingyu Wang, Anyi Liu, and Sushil Jajodia. “Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts.” In: *Computer communications* 29.15 (2006), pp. 2917–2933.

[76] Shelly Xiaonan Wu and Wolfgang Banzhaf. “The use of computational intelligence in intrusion detection systems: A review.” In: *Applied Soft Computing* 10.1 (2010), pp. 1–35.

[77] David Yarowsky. “Unsupervised word sense disambiguation rivaling supervised methods.” In: *Proceedings of the 33rd annual meeting of the association for computational linguistics*. 1995, pp. 189–196.

[78] Sarah Yoder. *Automating Mapping to ATT&CK: The Threat Report ATT&CK Mapper (TRAM) Tool*. `https://medium.com/mitre-attack/automating-mapping-to-attack-tram-1bb1b44bda76`. [Online; accessed 28-April-2020]. 2019.

[79] Qiu Hua Zheng, Yi Guang Xuan, and Wei Hua Hu. “An IDS alert aggregation method based on clustering.” In: *Advanced Materials Research*. Vol. 219. Trans Tech Publ. 2011, pp. 156–159.

[80] Bin Zhu and Ali A Ghorbani. “Alert correlation for extracting attack strategies.” In: *IJ Network Security* 3.3 (2006), pp. 244–258.