Sharing and Automation for
Privacy Preserving Attack Neutralization

(H2020 833418)

D4.5 Algorithms to recommend response and recovery actions to human operators, final verison (M30)

Published by the SAPPAN Consortium

Dissemination Level: Public

H2020-SU-ICT-2018-2020 – Cybersecurity

Document control page

Document file: Deliverable 4.5
Document version: 1.0
Document owner: Alexey Kirichenko (F-Secure)

Work package: WP4
Task: T4.3
Deliverable type: Other
Delivery month: M30
Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document History:

Version	Author(s)	Date	Summary of changes made
0.1	Andrew Patel (FSC) Teppo Ahonen (FSC) Paul Blomstedt (FSC) David Karpuk (FSC) Martin Zadnik (CESNET)	2021-10-06	Started writing draft
0.2	Willie Victor (FSC) Alexey Kirichenko (FSC)	2021-10-25	Revisions, Section 6 added
0.3	Andrew Patel (FSC)	2021-10-27	Fixed Table and Figure numbering, reference numbering.
0.4	Alexey Kirichenko (FSC) Willie Victor (FSC)	2021-10-29	Clean-up, improvements to Section 6, addressing review comments, Conclusion and Exec. Summary
1.0	Alexey Kirichenko (FSC)	2021-10-30	Integrated final revisions

Internal review history:

Reviewed by	Date	Summary of comments
Martin Laštovička (MU)	2021-10-27	Technical review
Mehdi Akbari Gurabi (FIT)	2021-10-30	General editorial comments

Authors

Teppo Ahonen, Data Scientist, F-Secure Corporation (teppo.ahonen@f-secure.com)
Paul Blomstedt, Senior Data Scientist, F-Secure Corporation (paul.blomstedt@f-secure.com)
David Karpuk, Senior Data Scientist, F-Secure Corporation (david.karpuk@f-secure.com)
Willie Victor, Senior Security Consultant, F-Secure Corporation (willie.victor@f-secure.com)
Martin Zadnik, Ing., Ph.D., Project Manager and researcher at Traffic Monitoring and Configuration Department, CESNET (zadnik@cesnet.cz)
Andrew Patel, Researcher, F-Secure Corporation (andrew.patel@f-secure.com)
Alexey Kirichenko, Research Collaboration Manager, F-Secure Corporation (alexey.kirichenko@f-secure.com)

1 Executive Summary

Defending against cyberattacks remains a challenging task, especially given the lack of experts in the cybersecurity field. Organizations are attempting to solve this problem by deploying tools that enable less experienced security analysts to perform at a higher level of expertise. When working with incident response systems, analysts often deal with a large number of false alerts and the lack of key contextual information. In an attempt to address these challenges, the main focus of this report is capabilities for supporting security analysts – in Computer Emergency Response Teams (CERTs), Security Operation Centers (SOCs) and similar teams – in the planning and carrying out incident response activities. Such capabilities can also serve as a foundation for automating as many response tasks and procedures as possible (which is a key theme in Task 4.4 of the SAPPAN project). Understanding the type, severity and other relevant characteristics of a security incident that triggered an alert can be used to choose appropriate response actions, which can be either suggested to security personnel or, in certain cases, even carried out automatically. This report presents several methods and approaches, developed in Task 4.3 of the SAPPAN project, which contribute to more effective and efficient incident response, including: clustering of security incidents detected by an endpoint detection and response solution; generating denial of service (DoS) attack mitigation rules; predicting future attack steps in order to support security analysts in choosing appropriate response actions; estimating the 'suspiciousness' level of endpoint behaviour for better response preparedness. The report also presents and discusses the methodology for producing datasets that can be used to automate response actions via machine learning-based techniques.

2 Introduction

There is a clear need to assist cybersecurity personnel in incident response activities. This is the main objective of WP4 of the SAPPAN project, with the efforts of Task 4.3 going to providing incident responders with machine learning-generated recommendations and relevant contextual information for response activity planning and with Task 4.4 focusing on intelligent automation of the incident response process.

Based on the results of the first phase of Task 4.3, SAPPAN Deliverable 4.4 (D4.4) presented an overview of cybersecurity tools and solutions currently available for incident responders. The capabilities of a range of commercially available response support and automation offerings were described, and a review of related academic literature and ongoing Horizon 2020 projects was presented. D4.4 also presented the SAPPAN Task 4.3 research including (i) an incident similarity model (Section 3.1), (ii) a host aggregation model (Section 3.2), and (iii) a false alert recognition mechanism (Section 3.3). The

research results in this deliverable D4.5 come from the second phase of Task 4.3, and some of them continue the work presented in D4.4. Specifically, the research presented in Section 3 of D4.5 builds upon (i), and the work in section 7 – upon (ii) and (iii).

The techniques and methods covered in D4.5 can be helpful for the decision logic of response automation, which is a key theme in Task 4.4 and discussed in D4.7, together with specific examples of automated response, such as based on Apache Airflow and Intel Owl.

The remainder of this document is structured in sections, each presenting new research in the SAPPAN Task 4.3 scope. The sections are as follows.

Section 3 presents a method for clustering security incidents detected by an endpoint detection and response solution. This method contributes to the facilitation, and possibly automation, of incident response handling in the following ways: (a) it can be used to enable the execution of bulk processing actions, allowing security analysts to resolve many more incidents than they would without such a technology, (b) the output of the method can be used to improve and fine-tune attack detection engines to generate fewer false positive detections, and (c) by examining the evolution of incident clusters over time, it has the potential to be used to study the emergence and popularity of attack techniques and may even support the discovery of new attacks.

Section 4 presents an algorithm to recommend denial of service (DoS) attack mitigation rules. This section describes an innovative mechanism for analysing an attack and generating rules for filtering volumetric DoS while the attack is underway. It also presents a thorough evaluation of the proposed mechanism, that assesses its performance under conditions present in real deployment scenarios, including validation by CESNET SOC. An important contribution of the research effort is making the datasets used during the research publicly available.

Section 5 presents a contextual attack chain modelling method designed to predict future attack steps and to support security analysts in choosing appropriate response actions. When trained on well-scoped data, such as attack data from a single organization or a specific attack type, the method can highlight, with high precision, likely future adversarial actions. Outputs from this method can be used in downstream tasks, including those intended to automate response actions.

Section 6 presents methodology for building datasets that can be used to automate response actions. The main research goal was to investigate the feasibility of machine learning-based technologies, where previous response actions that led to successful resolutions of incidents are leveraged to guide the actions of responders when faced with occurrences of attacks similar to past ones. The first step in developing such technologies is clearly gathering data related to the actions of attackers and responders, establishing causal links between detection data and response actions, and presenting it in a format suited to machine learning applications.

Finally, Section 7 presents a method for measuring suspicious behaviour using host aggregation data (introduced in D4.4). Host aggregation is the process of profiling the behaviour of a monitored computer over a defined period of time, based on both current and historical events witnessed on that host. The research introduces a model that, when trained on historical data of known true positive security incidents, can estimate how likely a behaviour observed on a host is indicative of malicious activity. Such predictions can be a valuable first step in the planning of response activities.

3 Incident Clustering

3.1 Introduction

This section describes a mechanism designed in SAPPAN to cluster security incidents. It builds on the ideas and techniques presented in Section 3.1 (Incident similarity model) of SAPPAN D4.4. The proposed mechanism outputs information of interest to security analysts, partners, and customers of endpoint detection and response (EDR) solutions for planning attack response actions. In addition to describing implementation details and the usefulness of the results the proposed method produces, future directions will also be considered.

Clustering is used in a variety of cybersecurity applications. Anomaly detection uses clustering in a straightforward way – existing data is grouped into clusters and incoming data that does not fit nicely into an existing cluster is deemed an anomaly. Examples exist in the mobile data mining domain [1], and the cloud infrastructure domain [3]. Clustering is also used for the analysis of system log data [2]. Perhaps most similar to the research described in this section are mechanisms to cluster incidents for rapid processing by Security Operations Centers [4]. The most striking difference between [4] and this research is the opacity of vectorization procedures, distance metrics, and clustering algorithms used. Our use of simple vectorization techniques, the cosine distance metric, and agglomerative clustering gives our algorithms a satisfying level of transparency and explainability.

3.2 Motivation

The motivation for constructing clusters of similar incidents is at least three-fold. Firstly, the clustering of security incidents allows for bulk processing actions, allowing security analysts to resolve many more incidents than they would without such a technology (in particular, to plan response actions). Secondly, large clusters of highly similar unresolved incidents, especially if they are spread out across many organizations, are indicative of a detection technology that is generating too many false positives. This allows examination and subsequent pruning and fine-tuning of detection engines in order to reduce the number of false positive detections, saving analysts the pain of processing unnecessary incidents, and storing extraneous data. Thirdly, clustering allows the evolution of large clusters to be tracked over time. Paired with cluster 'explainability' and some expert human analysis, clustering technology can be used to study the emergence and popularity of attack techniques and may even enable the discovery of new attacks and advanced persistent threats (APTs).

3.3 Incident similarity – a review

The following describes how the proposed incident similarity clustering mechanism works. Given two incidents inc_1, inc_2, the incident similarity model allows one to calculate a similarity score:

$$0 \leq s(inc_1, inc_2) \leq 1,$$

where a similarity score of 1 means the two incidents are functionally identical, and a similarity score of 0 means they are completely dissimilar. The similarity score itself is calculated as a modified cosine similarity between vector representations of each incident. The vector representations themselves are constructed by one-hot encoding relevant features within the incident and then applying a modified TF-IDF transform.

```
model.transformed_train_data
```

```
<2922239x430707 sparse matrix of type '<class 'numpy.float32'>'
        with 52325066 stored elements in Compressed Sparse Row format>
```

Figure 1: Vectorized incidents stored in a sparse matrix

Vectorized incidents are stored in a sparse matrix as depicted in Figure 1. Given a specific incident to be investigated, a security analyst can query the model for several similar incidents. This provides a broad context that enables the security analyst to make a more informed decision when resolving incidents.

3.4 Incident Clustering – Algorithm Design

To design a clustering algorithm for security incidents that serves the use cases mentioned above, a set of conditions must be imposed which an algorithm must satisfy. The proposed clustering algorithm must satisfy the following conditions:

1. If $s(inc_1, inc_2) = 1$ then the two incidents belong to the same cluster.

2. If inc_1 and inc_2 belong to the same cluster, then $s(inc_1, inc_2)$ is large (above some pre-determined threshold ε).

The first condition simply states that if two incidents have the same vector representation then they must belong to the same cluster. This guarantees that batch processing a cluster will allow one to resolve all identical incidents simultaneously. The second condition states that any pair of incidents within a cluster are highly similar. Thus, the cluster is dense and all incidents within the cluster are likely to have the same, or a very similar resolution. Agglomerative clustering, which is a specific form of hierarchical clustering, is well-suited for this task. Agglomerative clustering works by constructing clusters recursively. Each point is initially assigned its own cluster, and clusters are subsequently merged if the maximum distance between every pair of points in a pair of clusters falls below a pre-determined threshold. Here the distance between two incidents is defined to be:

$$d(inc_1, inc_2) = 1 - s(inc_1, inc_2)$$

meaning that the second condition is guaranteed to be satisfied. Agglomerative clustering is performed on a deduplicated version of a sparse matrix containing vectorized incidents, which greatly reduces run time and guarantees that the first condition is also satisfied.

Figure 2 presents a sample of a few very large clusters obtained by the incident clustering algorithm. In addition to the information presented in the figure, analysts are also presented with incident counts per customer within each cluster, as well as identifiers of several recent example incidents per incident resolution in the cluster. Security analysts can apply the developed mechanism to perform incident investigations as an ongoing process involving both artificial intelligence and security experts, with the immediate goal of reducing false positive detections, and the ultimate goal of allowing certain slices of cluster data to become available to customers and partners.

```
[{'number_of_incidents': 495624,
  'max_distance_within_cluster': 0.0,
  'resolution': {'UNCONFIRMED': 495624}},
 {'number_of_incidents': 137683,
  'max_distance_within_cluster': 0.05674659460783005,
  'resolution': {'UNCONFIRMED': 136684,
   'AUTO_FALSE_POSITIVE': 998,
   'FALSE_POSITIVE': 1}},
 {'number_of_incidents': 93504,
  'max_distance_within_cluster': 0.07391102612018585,
  'resolution': {'UNCONFIRMED': 93504}},
 {'number_of_incidents': 66147,
  'max_distance_within_cluster': 0.07960761338472366,
  'resolution': {'UNCONFIRMED': 66146, 'AUTO_FALSE_POSITIVE': 1}}]
```

Figure 2: Large incident clusters

3.5 Resolution propagation between incidents

As illustrated in Figure 2, all the four clusters contain incidents that are UNCONFIRMED, meaning they have not been resolved by a human. Also present are FALSE_POSITIVE incidents, which were verified to be so by manual inspection. Incidents were marked as AUTO_FALSE_POSITIVE by an automated component in a detection engine. This was done by comparing the incident fingerprint with that of a known false positive. Such a comparison requires incidents be identical even before the vectorization procedure and is thus a much stricter notion of equivalence.

To implement improved security incident resolution functionality, a logic which enables the labelling of all UNCONFIRMED incidents sufficiently similar to known false positive incidents was developed. This (propagation) logic labels an UNCONFIRMED incident by comparing it to all incidents in the database within a radius ε around the queried incident. If a single CONFIRMED incident falls within the radius, the queried incident is automatically labelled as a CONFIRMED incident. If not, and there are FALSE_POSITIVE incidents within the radius, the incident is labeled as a FALSE_POSITIVE. If neither of these conditions is met, the incident label is kept as UNCONFIRMED. This process is visualized in Figures 3 and 4.

In Figure 3, the circle centered around the UNCONFIRMED query incident contains FALSE_POSITIVE incidents, and some UNCONFIRMED incidents, but no CONFIRMED incidents. It is therefore assigned the label FALSE_POSITIVE. In Figure 4, since the circle around the UNCONFIRMED query incident contains at least one CONFIRMED security incident, the query incident is labelled CONFIRMED.

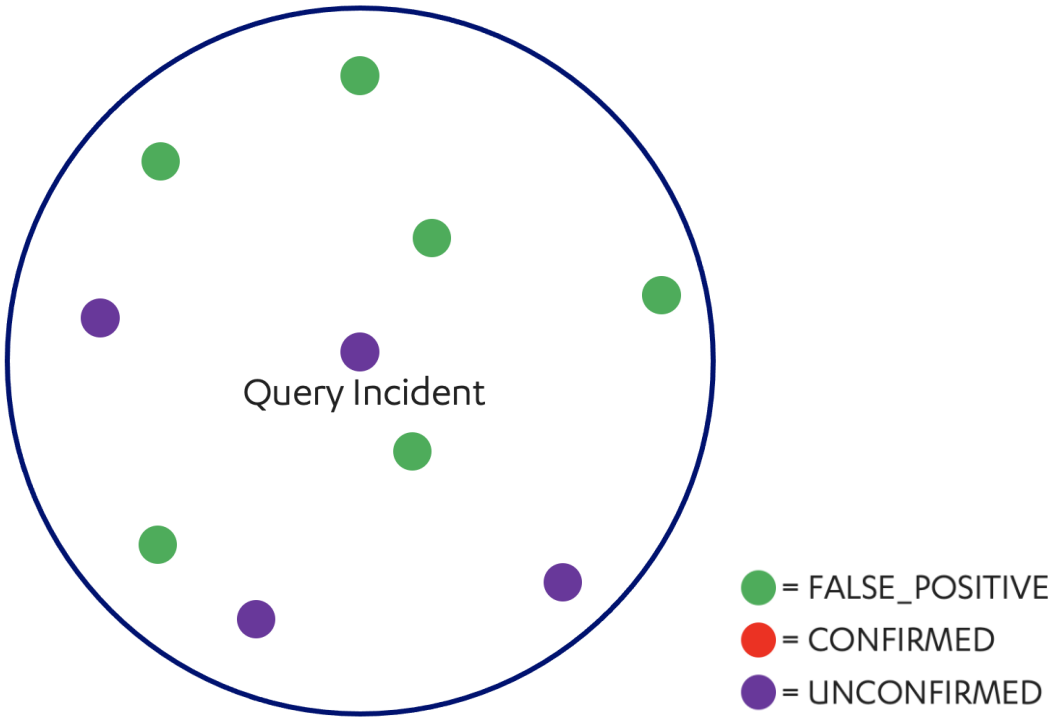

Figure 3: Labeling an UNCONFIRMED incident as FALSE_POSITIVE

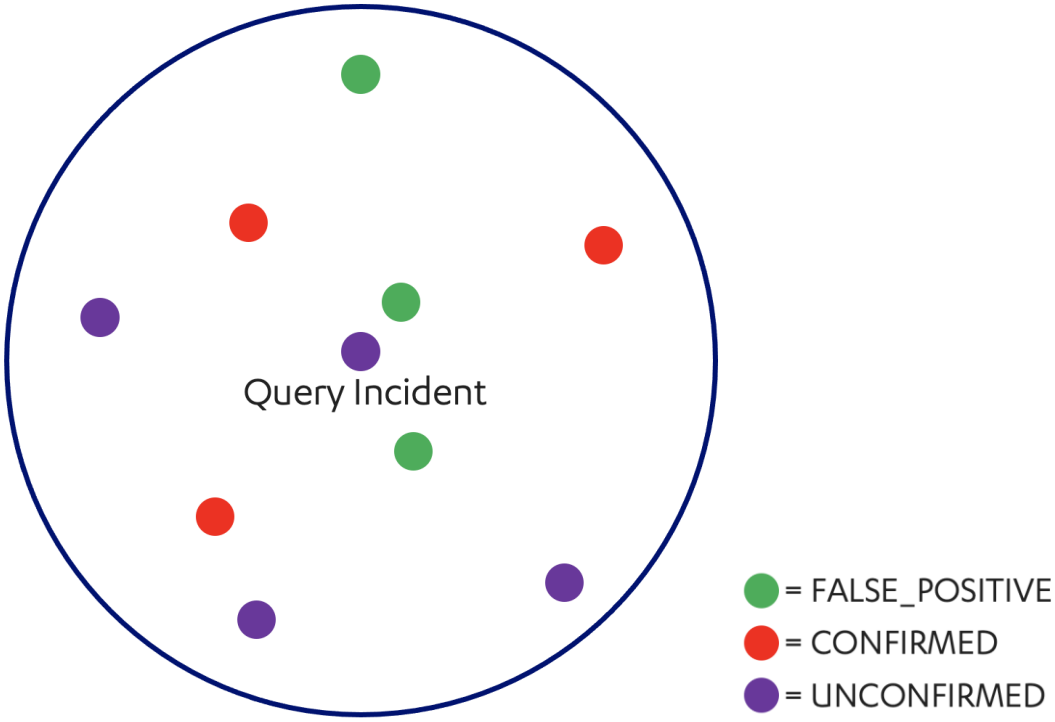

Figure 4: Labeling an UNCONFIRMED incident as CONFIRMED

The automatic labelling logic can be thought of as based on weighted majority voting of all incidents within a circle centered on the query incident, but with CONFIRMED incidents receiving an 'infinite' weight (it can also be considered a special form of nearest neighbor classification). This accounts for the class imbalance in the classifier problem, as almost all labeled incidents are false positives. From a more practical point of view, mislabeling a potential true security incident as a false positive is much worse than doing the reverse, hence the classifier must be quite liberal when labeling incidents as CONFIRMED.

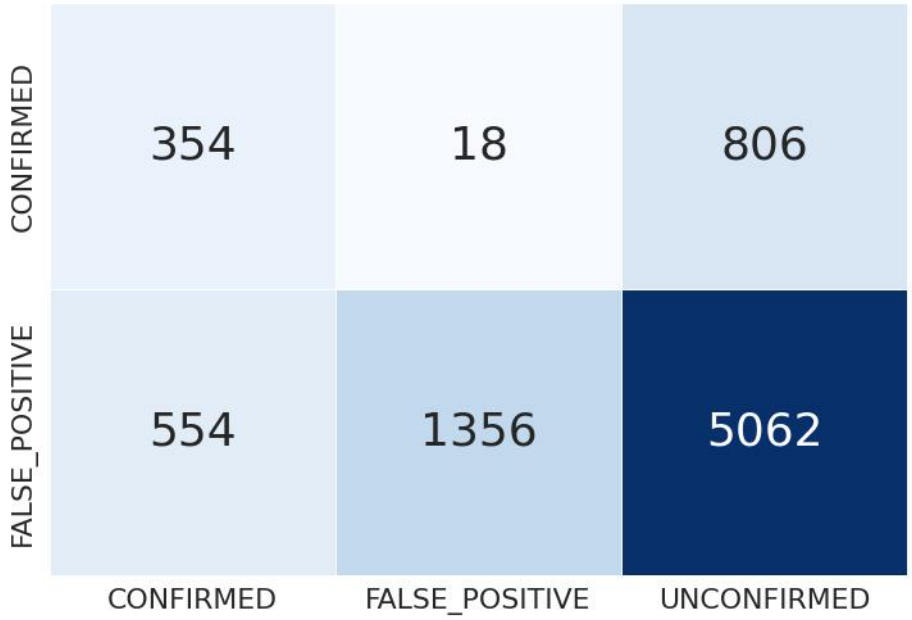

Figure 5: Incident labeling logic performance on validation set

Figure 5 illustrates how the labeling logic performs on a validation set (of 8,150 incidents not present in the reference database). The vertical axis represents the true label of the incident, and the horizontal axis represents the label predicted by the labeling logic. Because many incidents are somewhat unique (no similar incidents appear in this incident database), most incidents in the validation set are assigned the UNCONFIRMED label (so, the recall is not very high). However, this mislabeling has little practical consequence, as it simply means the resolution decision must be left to a human. On the other hand, very few CONFIRMED incidents are labeled as FALSE_POSITIVE, which is the mislabeling most likely to cause severe real-world consequences (the logic precision in predicting false positives is 0.987). While these results are encouraging, more work to fine-tune hyperparameters and acquire a larger training set (which can significantly increase the recall from the current level of 0.2) to improve the model is ongoing.

3.6 Conclusion and future directions

To conclude this section, the incident classifier essentially propagates resolution information throughout an incident cluster. Future research in this area will include: (i) investigation into passing other types of information between incidents, and (ii) research into making the clustering algorithm more dynamic, such that incoming incidents can be assigned a cluster as they are created, and information can be propagated directly to new incidents instead of having to wait for model retraining.

4 Algorithm to recommend denial of service mitigation rules

4.1 Summary

Denial of service (DoS) and distributed denial of service (DDoS) attacks are commonplace on the Internet and continue to represent a severe threat to online services. As part of the SAPPAN project, we have conducted research into mitigations against this threat. The contents of this section detail a novel mechanism, constructed as part of the SAPPAN project, to analyse a DoS attack as it is happening and automatically generate relevant filtering rules that can be used to mitigate the attack. The proposed mechanism is designed to significantly decrease the time it takes for network engineers to mitigate DoS attacks as part of their attack response activities. Our proposed methodology uses machine learning techniques to create a model of the traffic mix based on observing network traffic during an attack and during normal operation. Our approach is evaluated against several datasets. We experiment with various sets of hyperparameters as well as different intensities and types of attack traffic. The results of our experiments show that the proposed approach can successfully generate good filtering rules within a reasonable timeframe.

4.2 Introduction

Volumetric denial of service attacks (both regular DoS and DDoS attacks) prevent users and applications from accessing services provided over a network by exhausting available bandwidth or resources. The Internet does not have sufficient built-in mechanisms to prevent DoS attacks from happening. Many DoS mitigation mechanisms have been proposed, but few (e.g. reverse path filtering) have been deployed to any extent. As such, volumetric DoS attacks are still prevalent, and have evolved to become more efficient, a fact that makes defending against these attacks harder. Recent high-profile attacks utilized numerous compromised IoT devices to launch massively distributed denial of service attacks [5]. These attacks are asymmetric – they're cheap and easy to run but defending against them is difficult and costly. An attacker need not have their own infrastructure to run an attack, since DoS-as-a-service providers exist [6]). Defenders must use either significantly over-provisioned distributed infrastructure or utilize sophisticated mechanisms to detect and mitigate attacks.

When a single attack is sustained for a long enough time, it is possible for network engineers to manually analyse attack traffic in order to derive corresponding countermeasures. Investigations of this kind are lengthy and not all that affordable or scalable. DoS attacks are trending towards being frequent, multi-vector, and short-lived [7]. For these new breeds of attacks, defenders must be able to respond in an order of seconds, not minutes or hours. As such, automation is preferred over manual analysis.

In this section, we propose a method to (i) automate analysis of network traffic during an attack and (ii) automate the creation of rules to filter out the attack. Our proposed mechanism requires a sample of traffic captured during normal (non-attack) operation and a sample of the traffic seen during an attack. A machine learning algorithm, tree induction, is used to create a model of the attack traffic which is then converted into packet filtering rules. Our method differs from traditional approaches in which a classifier is trained using an annotated dataset before any attack happens and subsequently used to classify traffic during an attack.

Our research presents the following contributions:

- An innovative mechanism for analysing an attack and generating rules for filtering volumetric DoS while an attack is underway.
- A thorough evaluation of the proposed method to assess its feasibility under conditions present in a real deployment scenario.
- An assessment of our demonstration by CESNET SOC.
- Datasets used during the evaluation have been made publicly available.

4.3 Related work

Although our approach is more related to published DoS mitigation techniques, this section will also review detection techniques, since they are designed to work on information about incoming attacks, at the time they are happening, which is a vital input for our method.

MULTOPS [8], which represents early research in the field of DoS detection, was proposed to detect bandwidth attacks based on a deviation from a communication proportional symmetry using just byte and packet counters from routers. More recent methods in the area of DoS detection utilize data obtained from sampled packets or IP flows which provide better observation detail. As an example, most source IP addresses participating in a DoS attack will be new to the victim [9], so monitoring the number of new source IP addresses seen by the victim's systems [10] is a good way to build a detection mechanism.

Du and Abe [11] proposed an attack detection scheme based on packet size entropy for each application (identified by transport port number). Their assumption was that the entropy of normal traffic is higher than the entropy of attack traffic. Attack traffic often consists of similar packets as compared to legitimate traffic, where packet sizes vary according to each application. Their detection method is based on the deviation of entropy from a mean value. In general, other proposed schemes also utilize entropy calculations on selected traffic characteristics such as (i) randomness of flows at routers, (ii) distribution of source IP addresses in dependence on destination port numbers in the flows, and (iii) time series. Time series deviations are detected using simple methods such as EWMA or Holt-Winters, and more complex methods such as Wavelet analysis.

Machine learning algorithms have been proposed to detect DoS attacks in both supervised and unsupervised fashions, such as [12]. From more recent publications related to our work, the authors of [13] proposed a set of 27 features and compared the use of multi-layer perceptrons, random forests,

and Naive Bayes methods for classifying DoS attacks. In comparison to our methodology, their selected features were not packet based, and the classifiers were trained offline on annotated datasets. Furthermore, their classifiers were designed to recognize specific attack types and not to determine which packets were legitimate and which belonged to the attack.

The previously discussed machine learning approaches can tell when an attack takes place, but they do not deal with the mitigation of the DoS traffic itself.

When considering Intrusion Prevention Systems (IPS), which are designed to perform various types of attack mitigation, it can be noted that they often fall short when it comes to volumetric DoS protection [14] Intrusion prevention systems are designed to analyse incoming data in great detail, and hence are vulnerable to DoS attacks themselves. In our own experience, they are indeed the targets of volumetric attacks, resulting in a peculiar situation – the service is up and running, waiting for users, but the users cannot access it since the IPS is overwhelmed. To mitigate this problem, dedicated anti-DoS devices or cloud services are on offer. Unfortunately, the vendors of these solutions and services do not elaborate on how they work, and only provide vague descriptions such as “statistical anomaly detection”, “protocol anomaly detection”, “fingerprint matching”, and “profiled anomaly detection”. Based on our experience these products mostly automate the mitigation of DoS attacks via deny-listing IP addresses and/or regular expressions (delivered by intelligence feeds).

Surveying the existing research literature, we observe that many mitigation strategies are designed to prevent spoofing of IP addresses. One such basic preventive method suggests ingress filtering [15] in customer or source ISP networks using a pool of legitimate, well-known source IP addresses. In order to allow filtering in transit or destination networks, information about legitimate source IP addresses must be passed from source to destination networks. This can be achieved using source address validity enforcement protocol [16]. The authors of [16] propose a new protocol to spread information to routers along the path, which builds a filtering table for each ingress interface accordingly. Xie et al. [17] proposed authentication of a host connecting to the Internet using an established authentication protocol. TCP SYN cookies, improved in [18], may also be considered as an IP-spoofing prevention although the method only works to prevent TCP SYN-flood attacks.

Research into the detection of spoofed packets is also covered in [19]. The methods are based on detecting variances in TTLs (Time To Live). In [19] the authors discuss TTL issues which constitute a problematic estimation of initial TTL (consider NAT, change of routes, etc.) and the possibility of spoofing TTL values.

Finally, Xu [20] designs a method to reveal spoofing of source IP addresses by a statistical analysis of their distribution. Xu assumes that an attacker will spoof IP addresses randomly with uniform distribution. But the attacker may choose to spoof IP addresses from a given subnet or from a certain subnet within other distributions, hence violating the assumption of a uniform random distribution. In comparison to described spoofing detection methods, we consider all network and transport header fields to be relevant for the identification of packets belonging to an attack and we let a machine learning algorithm decide which fields are relevant under the given circumstances (no matter the attack type, IP addresses spoofing or TTL issues).

4.4 Problem statement

Consider two separate datasets containing captured network traffic in the form of raw packet captures (pcap files). The first dataset contains legitimate network traffic and corresponds to periods of normal operation for a service. The second dataset contains volumetric DoS traffic as well as legitimate traffic and corresponds to a period when a service or infrastructure is under attack.

Our goal is to create an algorithm capable of generating mitigation rules that can filter traffic belonging to the volumetric DoS attack. The algorithm is shown both normal and attack datasets and it is aware of which one is which. However, it has no prior knowledge about the legitimacy of packets in either dataset. After observing both datasets, the algorithm should generate a set of rules that will filter or block attack packets while ignoring or letting through legitimate packets, preferably using a very small number of filter rules. The need for a small number of rules is crucial with respect to saving mitigation resources – we don't an algorithm that outputs a rule for each offending packet (since that would be a very long list of rules and would be almost impossible for human operators to review or even understand). The rules should block all attack traffic but may also block a small amount of legitimate traffic. Blocking a small portion of legitimate traffic is considered an acceptable price for preserving the availability of a service for the rest of the user base during a DoS attack.

4.5 Assumption

Two assumptions about the characteristics of the network traffic are made in order to mirror real deployment scenarios and allow the algorithm to find a reasonable solution. The first assumption relates to the ratio of legitimate to attack traffic in each dataset. It is assumed that the legitimate dataset contains mostly normal operational traffic but may contain a small amount of attack traffic (such as packets belonging to scans, brute-force attacks, or residuals of DoS attacks, known as backscatter traffic). The attack dataset, on the other hand, is assumed to contain mostly attack packets, along with a small portion of legitimate traffic. We argue that it is realistic to collect and identify such datasets on the fly even in real deployments. For example, utilizing outputs of network behavioral analysis (NBA) systems or approaches presented in Sec. 4.3, when there is no alert issued by the NBA, the dataset is considered legitimate and when there is an alert about DoS traffic, the dataset is considered to be in the attack category.

Our second assumption is that volumetric DoS traffic exhibits a degree of self-similarity – packets belonging to the attack are somewhat similar to each other. The similarity may appear at the network layer (e.g. the same specific size of packets), at the transport layer (e.g. the same specific TCP window size) or at the application layer (e.g. the same specific value of HTTP agent or same content of the payload). We do not consider the application layer in this article, but plan on including it in future work.

4.6 Approach

We chose to use decision trees as our algorithm since (i) they have been successfully utilized in network traffic analysis [21], and (ii) trained decision tree models can be easily converted into filtering rules that follow packet filter specifications (e.g. a set of AND/OR expressions). Generated rules can be directly plugged into existing mitigation solutions and are, importantly, human readable. As such, network administrators can verify rules generated by our system and decide whether to deploy them as-is, hand-edit them before deployment, or disregard them completely.

Our algorithm is trained using a supervised approach. As mentioned earlier, training requires two datasets – one collected during normal operation, and another collected during an attack. All packets in the attack dataset are considered as positive samples and all packets in the legitimate dataset are considered as negative samples. While such an approach to annotation introduces some errors (legitimate traffic in the attack dataset will be marked as offending and vice versa), a majority of packets will be correctly labelled in each dataset. Our second assumption about the self-similarity of attack packets should mean that more weight is put on truly positive samples while the truly negative samples in the offending dataset will be outweighed by the negative samples in the legitimate dataset.

The machine learning pipeline consists of well-known steps of feature extraction, training and classification. The feature extraction phase parses packets, one by one, and extracts the header fields. The list of the currently utilized header fields is depicted in Table 1.

Proto	Fields
IPv4	Type of Service, Total Length, Identification, Dont fragment, More Fragment,Fragment Offset, Time to Live, Header Checksum, Source Address, Destination Address
TCP	Source Port, Destination Port, Sequence Number, Acknowledgement Number, Data offset, Flags, Window, Checksum, Urgent Pointer
UDP	Source Port, Destination Port, Length, Checksum

Table 1: Overview of the packet fields used as features

Decision trees are built using scikit learn's DecisionTreeClassifier implementation [22]. We omit detail on how the decision trees are constructed, since we consider the methodology to be straightforward and well-documented. Filter rules are created using all paths from the root to positive leaf nodes in the generated decision tree model. These rules are extracted from the decision tree using a depth-first search (recursive version) which creates the disjunctive form of a ruleset.

```
dfsb(node=root, brule="")
  if node is leaf and node is positive:
    print(brule +" | ")
  else:
    dfsb(node.left,brule +" & "+ condition)
    r=reverse(condition)
    dfsb(node.right,brule +" & "+ r)
```

The above algorithm constructs a rule ($brule$) corresponding to a given branch and depth of recursion derived from a trained decision tree model. It appends $brule$ with the condition of a current node delimited by \& symbol (logical and). When the algorithm encounters a positive leaf node it prints the $brule$ delimited by $|$ symbol (logical or). The above description of the algorithm is simplified (it produces a ruleset with extra leading \& and trailing $|$ symbols).

4.7 Datasets

Our datasets were constructed using three different sources:

1. A publicly available DDoS Evaluation Dataset (CIC-DDoS2019) [23] from the Canadian Institute for Cybersecurity. It provides variants of DDoS attacks, including SYN flood, UDP flood, DNS amplification and NTP amplification.

2. A dataset generated using publicly available stress-test tools – LOIC, HULK and Torshammer. Some generated attack samples were modified to simulate random spoofing of source IP address in the packet level to test that our inference algorithm would not simply use the source IP address as an identifier in a mitigation rule. Generated attack samples were mixed to create four multi-vector attack types - ALL, ALLTCP, ALLUDP, and SYNDNS. Only multi-vector attack samples (instead of single vectors) were used during evaluation since those samples realistically reflect the current DoS attack landscape. Additionally, if our inference algorithm is able to handle multi-vector attacks, it will, by default, handle single vector attacks as well.

3. A legitimate traffic mix captured between Austrian and Czech National Research and Educational Network (ACONET and CESNET respectively).

4.8 Evaluation

Our evaluation methodology was designed to empirically discover whether our proposed approach can generate filtering rules by simulating conditions that the algorithm would encounter in an operational environment. Our experiments were also designed to investigate how well our inference algorithm performs with different decision tree hyperparameters and their setups.

Grid search was used to find optimal hyperparameter values. The respective ranges and granularity of parameters tested during grid search were derived based on familiarity gained during initial experimentation. Hyperparameter settings were explored in order to find non-trivial, well-performing results. An exaggerated example of a trivial result is a decision tree containing a dedicated leaf for each distinct packet to be filtered. Such a decision tree would likely perform well in the environment where it was generated but would be useless in any other environment, in addition to being huge and unreadable.

Dataset	Max depth	Max nodes	Min leaf	Min split	Results (TP, FP)
SYNDNS	>5	10	0.005	0.005	99.94%, 0.93%
ALLUDP	>7	15	0.005	0.01	98.31%, 0.64%
ALLTCP	>5	10	0.005	0.06	100%, 0.94%
ALL	>7	12	0.005	0.005	98.32%, 2.86%

Table 2: Overview of the success rates (TP – true positive rate, FP – false positive rate)

Table 2 presents the best results found by during hyperparameter search for each dataset. The last row in the table represents a set of values that worked well across all datasets. The best performing parameters were different for each dataset. We confined the depth of a tree to between 5 and 7 and the maximum number of leaf nodes to be between 10 to 15 (in particular, because we want our rules to be simple). In all the cases, our models achieved a 97% or greater true positive rate and a 3% or lower false positive rate.

A second experiment was run to test the assumption that the majority of packets in the attack dataset are attack packets. During this experiment, the percentage of legitimate traffic in the attack dataset was gradually increased in order to determine how large the attack would need to be to generate successful mitigation rules. The aim was to simulate a situation where an attack is detected and reported but only a small portion of the aggregated traffic is being forwarded to the victim. Therefore, the majority of the mix during the attack period is composed of legitimate traffic.

The reported results are based on the general setup of hyperparameters presented in the last row (ALL) of Table 2. The graphs in Figure 6 illustrate true positive and false positive rates with respect to the percentage mix between legitimate and attack samples. The higher the percentage of legitimate samples the more we confuse the training process with incorrectly labelled samples. In the worst case, there is only 20\% true DoS traffic, while the rest of the traffic labelled as DoS is legitimate.

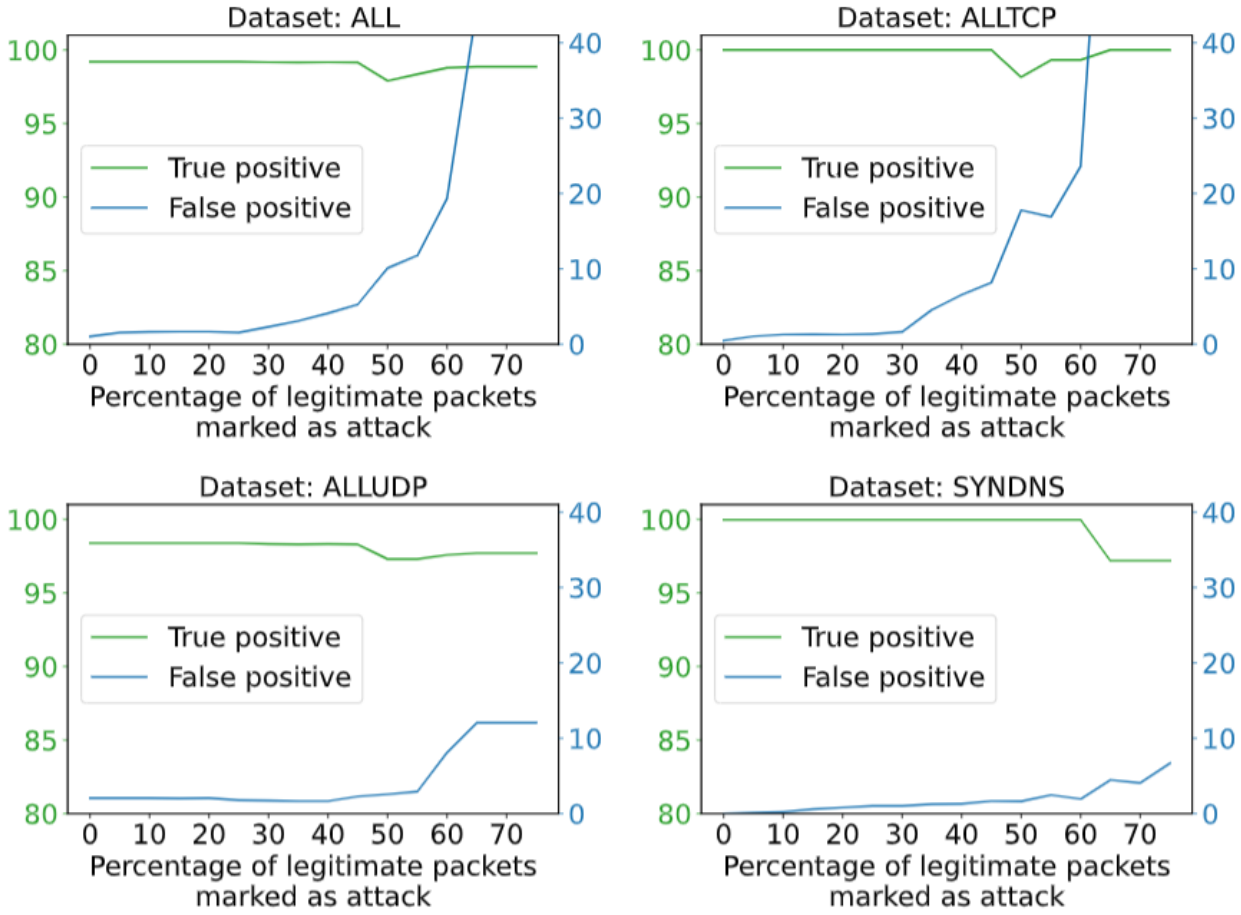

Figure 6: True and false positive rates with respect to the increasing portion of DoS traffic in the dataset.

4.9 Demonstration

In this section, we present a decision tree and the respective mitigation rule representation to demonstrate outputs generated using our mechanism. The illustrated decision tree blocks DoS traffic contained in ALLTCP dataset. Its graphical representation is depicted in Figure 7. A blue-coloured node indicates prevalence of positive (attack) class while orange-coloured nodes depict negative (legitimate) classes. White nodes indicate an equal share of classes. The first row in a node describes a condition used for decision (if the condition is met follow left arrow), Gini indicates impurity (Gini=0 means pure) and the last row assigns a class to the node. We can see that the tree uses only five features. These are the most important features to differentiate the DoS packets from the legitimate ones, but only for this particular dataset. Other attacks are identified by the same or other features. This depends on how the attacker crafts DoS packets and whether there are some significant packet fields that can differentiate those DoS packets. Moreover, we do not want a rule to contain many features as it would make the rule more complex.

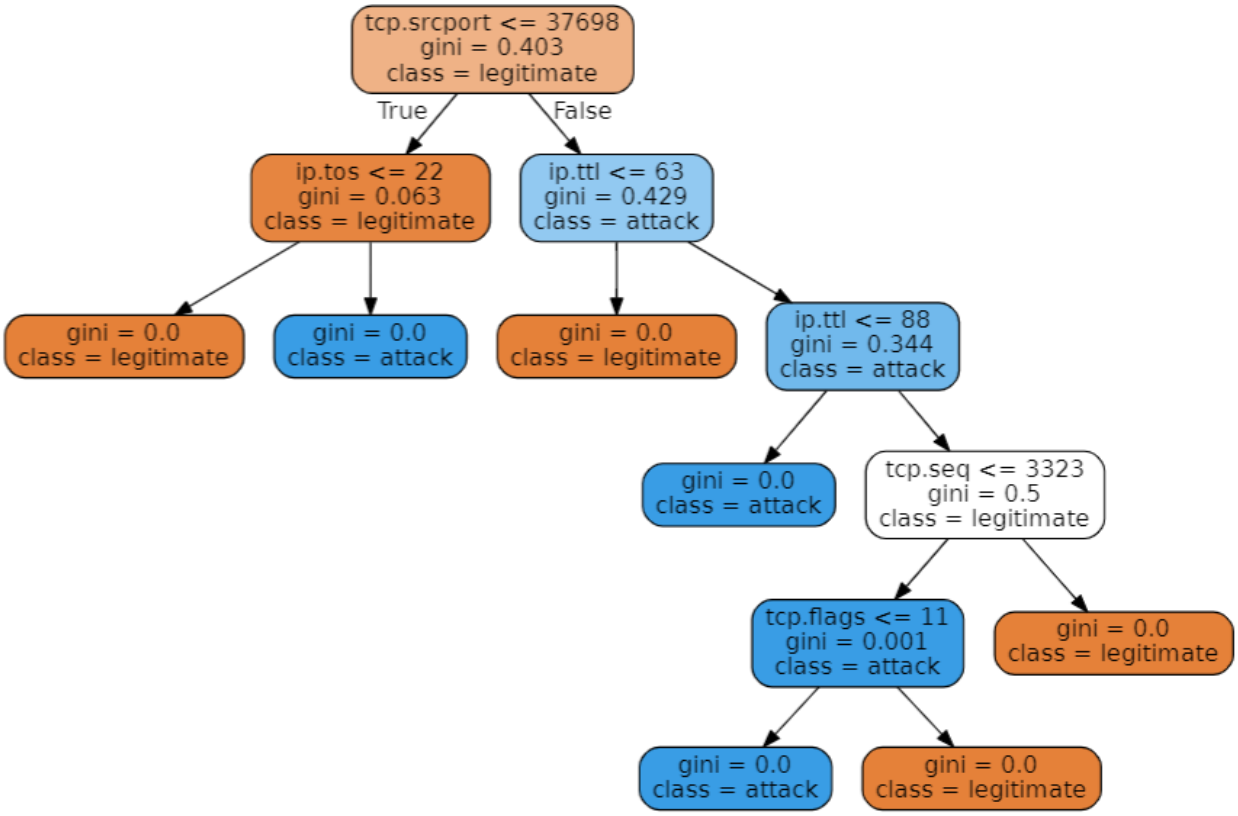

Figure 7: Example of a decision tree generated by the proposed approach

We transform this tree into a respective filtering rule set by the algorithm proposed in the Approach section. The generated rule achieves 100% true positive and 1% of false positive rate. The rule is as follows:

((tcp.srcport<=37698 & ip.tos>22)|
(tcp.srcport>37698 & ip.ttl>63 & ip.ttl<=88)|
(tcp.srcport>37698 & ip.ttl>63 & ip.ttl>88 &
tcp.seq<=3323 & tcp.flags<=11))

As a part of our evaluation, we asked our CESNET Network Operation Center to assess the proposed method. They retrospectively compared their workflow with and without using the proposed method on historical DoS cases. We received positive feedback that the method sped up their ability to quickly identify packets that are part of an attack, especially in cases when they have not seen that attack before. They also identified some scope for further research. One suggestion they presented was a need for our mechanism to prioritize certain packet fields over others – for example, to force the training algorithm to prioritize the use of source IP prefixes, enabling them to quickly identify and implement filters only on external IP prefixes (prefixes from abroad).

4.10 Conclusions

This section detailed a novel mechanism for automatically generating packet filtering rules designed to mitigate DoS attacks based on analysis of network traffic. The method utilises decision trees to learn which packet fields are part of a given attack and how to combine them into hierarchical logic

that can be used to generate filtering rules via simple conversion logic. Four multi-vector DoS datasets were evaluated to assess the effectiveness of the proposed anti-DoS mechanism under various real-world-like conditions. The results demonstrated that the proposed approach was able to generate successful filtering rules against a range of attack scenarios. Although optimal hyperparameter values varied between datasets, a set of hyperparameter values that resulted in reasonable efficiency against all tested scenarios were found to suffice if a process relabelling of impure nodes was also applied to the process.

The results of these research efforts were submitted to the IEEE Transactions on Network and Service Management journal.

Planned future work in this area includes (i) automating the application of high confidence generated rules, and (ii) automating the recognition of changes in attack vectors in order to trigger retraining or modify applied filters. The former is closely related to SAPPAN Task 4.4 where the goal is to perform response actions without a human operator.

5 Contextual attack chain modelling

5.1 Introduction

Experienced security analysts, threat hunters and incident responders, are often able to relate current observations to past experience. Relevant insights could include indicators of compromise, e.g. IP addresses, malware signatures and toolsets, or more generally, behavioural patterns observed through specific sequences of detections. This kind of knowledge gives context to current observations and may help in anticipating future adversarial actions, which in turn enables an optimal response action to be taken in a timely manner. It is clear that efficient response handling crucially depends on an analyst's expertise and acquired experience. Even a highly experienced analyst might not have been exposed to the particular kind of attack they are currently handling. Furthermore, manually inspecting even a moderate number of potentially related past attacks or incidents is often time-consuming and may not be feasible when a quick response action is of critical importance.

The goal of this work is to develop a machine learning-based approach to support the process of projecting past experience to potential future adversarial actions. This approach is built on the following observations and assumptions. First, the vast majority of attacks utilize a collection of well-understood techniques. To make this approach less dependent on a specific product or technology, the ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) framework is utilized [24], which is a well-established taxonomy to identify attack techniques at a level to which more fine-grained and product-specific detection signatures can be mapped. Second, the number of techniques used in any given attack, tends to be fairly limited and depends on the type of attack in question and the adversarial actor behind it. Finally, on a high level, attacks tend to follow a logical sequence of steps. This assumption conforms with the often-cited *cyber kill chain* framework and its many variants, which stipulates that adversaries must follow a certain chain of attack phases in order to achieve their objective [25, 26].

The above assumptions imply that certain attack techniques can be expected to appear together more often than others. Moreover, the actions taken up to the current point of an attack influence the likelihood of actions likely to be observed in future steps of the attack. These implications are used to develop an approach for predicting attack steps. More concretely, given a set of detections made within an ongoing attack or incident, the proposed methodology aims to highlight a small number of techniques that are statistically likely to occur in later stages of the attack (see Figure 8). In addition to providing contextual information to aid in choosing an optimal response action, such predictions can also be useful for incident prioritization.

The rest of this Section is structured as follows. Section 5.2 describes the data and the proposed approach. In Section 5.3, predictive performance of the approach is evaluated on incident data from a production environment. A discussion of the findings is presented in Section 5.4.

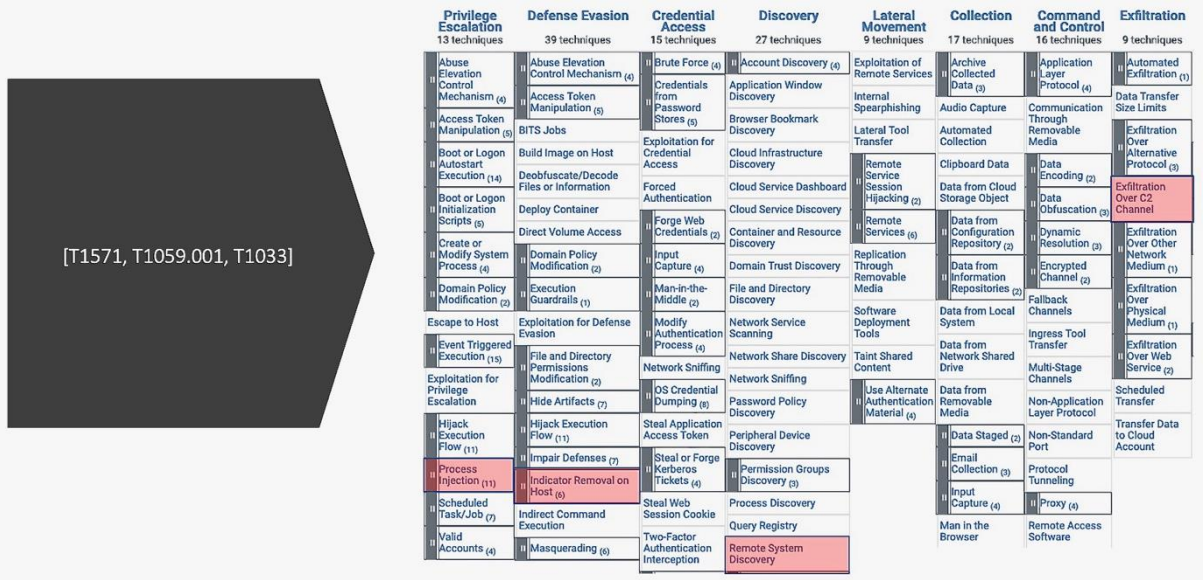

Figure 8: Given a sequence of detections with associated ATT&CK IDs (left), highlight techniques likely to occur in later stages of the attack (right). The matrix on the right is a subset of the ATT&CK matrix available at https://attack.mitre.org/.

```
[T1571, T1059.00,1 T1033, T1055, T1027.004, T1548.002, T1059.001,
T1033, T1059.00,1 T1055, T1059.00,1 T1569.002, TA0008, T1218.011,
T1059.001, T104,1 T1059.00,1 T1055, T1087.002, T1018, T1055, T1571,
T1136, T1033, T1055, T1021.006, T1055, T1204, T1055, T1204, T1055]
```

Figure 9: ATT&CK ID sequence of a lab-generated attack.

5.2 Methods

5.2.1 Data representation and characteristics

In this work, it is assumed that an attack can be represented as a sequence of detections, each of which can be mapped to an ID in the ATT&CK framework. Figure 9 displays a representative example of an ATT&CK ID sequence. Note that the term *incident* is used to denote the collection of detections triggered by an *attack*, and these terms will be used interchangeably. The current version of the ATT&CK matrix contains 552 IDs for attack techniques. Data used during this research additionally contains 7 tactic IDs, making the total number of available IDs 559. Most sequences are typically made up of only a small number of IDs with partly repeating patterns. A further characteristic is that two sequences with similar high-level structures may often have considerable local differences in how IDs are ordered. The number of confirmed incidents available for training is, depending on the scope of the model, typically small or moderate at most.

5.2.2 Modelling approach

5.2.2.1 Prediction task

The following formulates an approach for making predictions on the unobserved, remaining part of a sequence, conditional on the IDs observed up to the current time point. This is as much a question of *what* to predict than it is of *how* to do it. As a first intuition, the problem can be approached as a

sequence prediction task, i.e. directly predicting the next ID in the sequence. However, even if accurately predicted, the next ID is often a repetition of a previously observed pattern, and as such not informative.

Instead, the proposed solution is to predict which of the IDs *not yet observed*, are still likely to be observed in the remaining sequence. More formally, let $\mathcal{U}$ be the set of all available ATT&CK IDs. Furthermore, let $\boldsymbol{x} = (x_1, \dots, x_k, x_{k+1}, \dots, x_L)$ be a sequence of IDs representing a complete attack, with only the subsequence $\boldsymbol{x}_{1:k} = (x_1, \dots, x_k)$ observed. The set of all IDs observed in $\boldsymbol{x}$ is denoted as $\mathcal{T}$ and the set of IDs in the observed subsequence as $\mathcal{T}_{1:k}$. The task is then to estimate, for each $t_m \in \mathcal{U} \setminus \mathcal{T}_{1:k}, m = 1, \dots, M$, the probability

$$p_m := Pr(t_m \in \bar{\mathcal{T}}_{1:k} | x_1, \dots, x_k), \qquad (1)$$

where $\bar{\mathcal{T}}_{1:k} := \mathcal{T} \setminus \mathcal{T}_{1:k}$. Note that in general $\sum_{m=1}^{M} p_m \neq 1$.

5.2.2.2 Formulation as multilabel classification

To estimate the probabilities (1), the problem is formulated as multilabel classification (see e.g. [27, 28]). A *binary relevance method*, a popular approach for multilabel classification, is used which transforms the problem into a series of independent binary classification problems. The method can be combined with any binary classification algorithm.

For each classifier, a bag-of-words features is extracted from the observed sequence $\boldsymbol{x}_{1:k}$, and all available IDs *not* included in $\boldsymbol{x}_{1:k}$ are used as the set of target labels. Given a training set of N complete sequences $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\}$, input-target pairs are composed for training from each sequence as

$$\left(\boldsymbol{x}_{1:k}^{(n)}, \bar{\mathcal{T}}_{1:k}^{(n)}\right), \qquad k = 1, \dots, L-1.$$

5.2.2.3 Relevance thresholding

The final component of the proposed approach is to use the estimated probabilities to determine which IDs to regard as relevant in the context of the observed IDs. Unfortunately, the probability estimates given by the binary relevance model are not consistent across sequences, and therefore cannot be directly used as a decision criterion. Instead of using a fixed probability threshold or naively returning a fixed number of predictions, the probability threshold is set adaptively.

To this end, *relevance score* is defined using a two-step procedure, where probabilities are first normalized over all target classes and the ratio between the normalized probabilities and their mean are computed. The resulting expression can be written as

$$r_m := \frac{p_m}{\sum_{m=1}^{M} p_m} \Big/ \frac{1}{M} = \frac{M \cdot p_m}{\sum_{m=1}^{M} p_m},$$

where M is the number of target classes and $1/M$ is the mean of the normalized probabilities.

The score r_m can roughly be interpreted as an expression of how many times more likely the mth target ID is to occur than an ID chosen uniformly at random. Note, however, that since the underlying probability estimates cannot be guaranteed to be well-calibrated (see e.g. [29]), r_m is used only to set the relevance threshold. Further interpretation of its numerical value is forgone.

5.3 Numerical evaluation

In this section, the predictive performance of the approach described in the previous section is evaluated. We call it here the *relevance model* in reference to both the objective of predicting relevant adversarial actions, as well as the binary relevance method at the core of the approach.

5.3.1 Experimental setting

5.3.1.1 Data

The evaluation is performed on data consisting of ATT&CK ID sequences extracted from a production environment over the course of 16 weeks. Each sequence is a temporally ordered series of tagged detections, representing a single incident. The scope of an incident (i.e. the collection of detections that make up a sequence) is automatically determined by the detection system running in a production environment. The results are obtained using 8-fold time-series cross-validation, where each fold is composed of 8 weeks of training data, followed by one week of data for evaluation. Thus, the folds are time windows of fixed size, moved forward by increments of one week.

The current model primarily targets use-cases with organization-specific focus. To ensure that each model gets trained on enough data, within each fold, only organizations with at least 5 sequences available for training are considered. For both training and evaluation, sequences containing three IDs or more are used. In addition to organization-specific, i.e. *local* models, performance of *globally* trained models, which use data from all available organizations, are also evaluated.

5.3.1.2 Models

The evaluation includes the proposed relevance model, and additionally, two types of baselines:

- **Relevance model** Proposed approach using three different classification algorithms: logistic regression (LR), naive Bayes (NB) and random forest (RF).
- **Sequence model** Baseline that predicts the next unobserved ID, i.e. estimates the probability $Pr(x_{k+1}|\boldsymbol{x}_{1:k})$ using multiclass classification. The same three classifiers are used as above.
- **Naive baseline** Baseline model that makes predictions based on the marginal distribution of all target labels, estimated from the training data.

For all the models, prediction probabilities are transformed into relevance scores for thresholding.

5.3.1.3 Metrics

The following metrics are reported for all models:

- Precision: the fraction of IDs predicted to be relevant that eventually occur in the remaining sequence.
- Hit rate: accuracy w.r.t. the next unobserved ID being included in the predicted relevant set.
- Number of IDs predicted to be relevant, i.e. the size of the highlighted set of items with relevance scores above a given threshold.

The reported metrics are averaged over the steps of each evaluation sequence and over each cross-validation fold.

5.3.2 Results

The results for local and global settings are summarized in Tables 3 and 4, respectively. In both settings, metrics for two different relevance thresholds are reported. In the local setting, these are $r = 1$ and $r = 2$. In the global setting, slightly higher thresholds were found to be more appropriate and therefore used $r = 2$ and $r = 3$.

Precision is the most important metric, since it directly measures the quality of the predicted relevant set. Hit rate, on the other hand, measures the ability of the model to predict the immediate future, which may not always be crucial if the model is able to predict important steps further into the future.

There is also a trade-off between precision and hit rate: increasing the relevance threshold leads to fewer IDs being selected, which is likely to increase precision and decrease hit rate. An important consideration when interpreting the metrics, in particular the precision, is that the number of IDs per sequence is typically very small (see distribution in Figure 10). Accordingly, we also strive for a small set of highlighted IDs. This combination makes the precision very sensitive to false positives. For example, with a predicted set of size three, already a single false positive will result in a precision as low as 0.67.

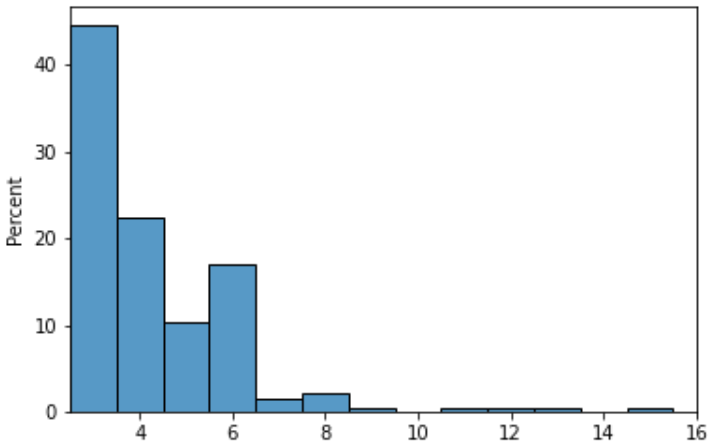

Figure 10: Distribution of the number of IDs per sequence in the data used for the evaluation.

The results show a dramatic difference between the local and the global settings, in favour of the former. This confirms our observations from preliminary experiments that the approach is of greatest value when using models trained with a suitably narrow scope. Overall, the results suggest that the relevance model with a random forest classifier is the preferred choice. In all of the evaluated settings, this model consistently achieves the best precision. Since the sequence models were optimized for hit rate, they are expected to outperform the relevance models in this metric. In the local setting this is indeed the case, while in the global setting, the best hit rate is achieved using a relevance model with logistic regression.

	Threshold $r = 1$			Threshold $r = 2$		
Model	**Prec.**	**Hit rate**	**# pred. relevant**	**Prec.**	**Hit rate**	**# pred. relevant**
Relevance LR	0.83	0.90	2.99	0.93	0.74	2.25
Relevance NB	0.81	0.54	1.86	0.82	0.49	1.72
Relevance RF	**0.97**	0.89	2.34	**0.99**	0.57	1.21
Sequence LR	0.59	**0.93**	3.48	0.65	0.68	1.83
Sequence NB	0.47	0.60	2.81	0.44	0.33	1.30
Sequence RF	0.86	**0.93**	1.44	0.90	**0.83**	1.04
Naive baseline	0.79	0.85	2.80	0.84	0.74	2.03

Table 3: Results on local models (the best result per column is bolded).

	Threshold $r = 2$			Threshold $r = 3$		
Model	**Prec.**	**Hit rate**	**# pred. relevant**	**Prec.**	**Hit rate**	**# pred. relevant**
Relevance LR	0.25	**0.59**	5.21	0.28	**0.50**	4.02
Relevance NB	0.27	0.32	2.76	0.28	0.30	2.36
Relevance RF	**0.37**	0.41	2.37	**0.41**	0.32	1.52
Sequence LR	0.22	0.53	4.84	0.26	0.47	3.63
Sequence NB	0.22	0.42	3.90	0.23	0.38	3.23
Sequence RF	0.34	0.28	1.53	0.35	0.25	1.17
Naive baseline	0.18	0.50	5.82	0.23	0.44	3.97

Table 4: Results on global models (the best result per column is bolded).

5.4 Discussion

An approach for predicting future attack steps to support analysts in choosing appropriate response actions was developed. When trained on well-scoped data, e.g. for a single organization or a specific attack type, the model is able to highlight with high precision likely future adversarial actions, at the level of attack techniques. A benefit of the model is that it can be trained on very limited data. Although not explored in the current work, models trained on specific attack patterns could also be shared and used to recognize patterns across several domains or organizations (which is relevant in the scope of SAPPAN Task 5.2).

The current model is still a prototype and could be improved in many respects. More accurate determination of the relevant set could be approached using two complementary strategies. One strategy would be to try to improve the relevance score by utilizing recent developments in classifier probability calibration [30]. Another strategy would be to explicitly estimate the length of the remaining sequence. The model could also be improved by accounting for richer contextual information in addition to only using ATT&CK IDs. This could in particular benefit models trained on global data. A further interesting direction is to combine the current model with an additional component for estimating higher level attack phases [31].

6 Data sets for constructing response recommendation engines

6.1 Introduction

The current state of endpoint security solutions allows for the collection of more incident data than ever before. While analyzing event data collected during a security incident, responders can usually reconstruct a detailed view of the attacker's actions from start to finish. The depth of security event data collection also allows for potential attacks to be identified at their very early stages using rules, heuristic engines, and machine learning models. Analysts and responders can thus be alerted and provided with data relevant to a potential ongoing attack early in the attack chain. The process of interpreting such data and deciding on an appropriate course of action requires specialist skills that are in short supply – especially when facing an advanced attacker. It is thus critical that responders and analysts' time and attention are optimally utilized, so that potential high-impact security incidents are prioritized. While there might be scope for the automation of some response actions over time, entire classes of attacks will always require an expert's experience and judgement. The development of assistive technologies is thus imperative to maintain a realistic workload on detection and response teams.

Naïve responders will generally perform thorough analysis of detection events to determine whether a security incident has occurred, gauge its severity and potential impacts, and make an informed decision on appropriate actions to take. The same generally holds true for experienced responders facing novel attacks. As responders are faced with detections or attacks that they've already faced multiple times, their experience tends to guide them to the appropriate response almost intuitively. And herein lies the value of retaining experienced experts in a detection and response team. The reality is though that such experts are in short supply, and their expense is often not justified in contexts where cybersecurity is not the core business of an organization. Another scenario to consider is where a novel attack emerges that affects multiple detection and response teams across multiple organizations, sectors, and geographies. Expert responders will tend to perform thorough investigations initially and use their findings to identify and deal with further occurrences of the attack.
The above raises the question of whether the actions taken by expert responders during an incident can be captured and used to guide the response of others during further occurrences. Furthermore, can data associated with the actions of experts across multiple organizations and environments to an identical attack be combined and distilled down to the core parts that resulted in it being resolved? Finally, can this be done in a way that is portable across organizations and endpoint detection and response technology stacks, allowing for actions that led to successful resolutions to specific attacks to be shared with the wider security community?

The goal of this research was to investigate the feasibility of machine learning-based assistive technologies for detection and response teams, where previous response actions that led to successful resolutions of incidents are leveraged to guide the actions of responders when faced with occurrences of known attacks. The first step in developing such technologies is gathering data related to the actions of attackers and responders, establishing causal links between detection data and response actions, and presenting it in a format suited to machine learning applications.

6.2 Motivation

As previously mentioned, current endpoint security products have access to more and deeper attack-related data than ever before. This has enabled the development of better rule and heuristic detection engines, as well as advanced machine learning models that can find patterns and anomalies in data that are simply not feasible for humans to do. While this resulted in earlier detection of security incidents, it has also led to more security events that may require the attention of detection and response teams. Such events can be categorized as follows:

- False positives, where an action or event was flagged by a detection engine as potentially malicious, when it was in fact benign.

- Incidents where an endpoint protection engine took automatic actions to neutralize the threat and that did not necessarily require human intervention, such as when a malicious document was blocked. Such events will likely still generate security notifications that need to be acknowledged by an analyst or responder.

- Detection of low-impact malicious activity that required human attention but was often resolved through one or two actions. An illustrative example would be one where a user executed a piece of malware that attempted to contact external infrastructure known to be associated with malicious activity. In such a case a responder may have instructed the protection agent to isolate the endpoint from the network and filed a service request for the machine to be retrieved and disinfected or rebuilt.

- Incidents that required some investigation, but where the scope was limited to a single endpoint and a single data source sufficient for analysis. For example, a heuristics engine may have raised an alert for a potential incident after an anomalous remote logon attempt

within the network, indicative of a lateral movement attempt occurred. A responder may have started by issuing a network isolation command to the source endpoint's response agent, followed by analysis of recent event data from the endpoint. Such an analysis may have included checking for connections from the endpoint to unknown external IP addresses or unexpected internal ones, recent successful incoming and outgoing logon events, whether privileged accounts were cached on the endpoint, and so forth. The analyst would have then collected these datapoints to decide whether the incident was resolved by removing the machine in question from the network, or whether the incident should have been escalated for further investigation.

- Some incidents might have required manual analysis of data from multiple sources, organizational context, and an iterative process of investigation, action, verification, and further investigation. These incidents tend to be advanced attacks driven by skilled human attackers, or novel advanced malware that was designed to rapidly spread across networks. In most cases, this category of incidents requires the intervention of experienced incident responders that possess judgement, experience, and lateral thinking capabilities.

The main priority for detection and response teams should undoubtedly be the last category, where time and attention are required. Conversely, such teams should spend as little time as possible on identifying false positives and resolving low-impact, low-risk incidents such as automated "spray and pray" attacks and unsophisticated predictable attacks performed by low-skilled attackers.
The goal of this research is to build sufficiently detailed datasets designed to enable machine learning techniques to be applied in aid of detection and response teams. Initial applications may include recommender models that provide responders with suggested actions based on observed attack indicators, preemptive retrieval and processing of data that will be needed by an analyst for their investigation, and introspective models that could provide insights on effective responses to specific incidents across organizations. Reliable recommender models appear most promising at this point due to their benefits in the field, and for their potential as a basis for "human on the loop" automated responses to certain classes of incidents.
This section deals with what such datasets should look like, how they may be collected in practice, and the challenges that will need to be resolved to make collection feasible at the scale required for data science and machine learning research.

6.3 Challenges

The initial goal of the project was to gather or generate a sample dataset that could be used for research into machine learning models that relate attack detection events to the actions of human responders. Prior research found no such datasets in the public domain, nor any concrete methodology for generating these. A framework and methodology thus needed to be developed.
In order to create a dataset suited to the goals of the research, data would need to be sourced from systems that contain detection event data, as well as the actions human operators took in response. Crucially, a clear causal link would need to exist between specific detection events (or collections of events) and responder actions – i.e., response action X was based on observations from detection datapoints A, B, and C.
Modern endpoint detection and response (EDR) products gather vast amounts of security event data from endpoints, and extensive processing, grouping, and automated analysis is performed to detect potentially malicious activity and assist human operators. Detailed records are also kept of investigation and response actions taken by operators, and these are often associated with specific incidents – broad groupings of events and detections that appear to be part of the same attack. What is generally missing is a strong direct association in datasets between the actions of a responder and the specific detection datapoints from which the action and its parameters were derived. Such associations can generally be inferred by a human expert looking at a timeline of events and response actions but going through this manual or semi-manual process is simply not feasible at the required scale.

Data gathered from endpoints are represented in schemas that are specific to the technology stack used by a vendor or product. These tend to be optimized for machine readability, since extensive processing needs to happen to identify "interesting" behavior and present it to human operators in a form that can be interpreted and used to make decisions. This presents an interoperability challenge when considering a reference dataset for research – outputs from such research, such as machine learning models, would only be applicable to the product that generated the data. Furthermore, detection event data tend to contain fields that are not relevant to the event itself but are artefacts of the technology stacks internal functions. Such parts should be considered noise and be excluded from any datasets.
Finally, if reference datasets are to be generated and shared on a limited or open basis, great care should be taken to ensure that any identifiable information is removed. This may include internal and external domain names, account and machine names, document names, specific IP addresses, usernames, and so forth. It should not be possible to associate any part of a reference dataset with any entity using any sort of inspection or analysis. Security vendors and managed security service providers have a responsibility to their clients to keep such information confidential and are also contractually and legally compelled to take all possible measures to this end.

6.4 Data Representation

In order to capture detection-response datasets suitable for machine learning research, datapoints need to be represented in a structure that accurately represents detection events and response actions. The level of detail should be sufficient to reconstruct a detailed timeline of events but should not contain any extraneous information such as vendor-specific metadata. The format and structure should ideally also be standardized in a way that would enable sharing of datasets and results between researchers.
In order to satisfy the above criteria, the following minimum information would need to be represented in a record:

- ***Timestamp***: The time at which the detection was made, or a response action was performed. For the purposes of this research, the specific time is not material, but rather the *relative* time between events. As such, the timestamp of the first event in the incident may be represented as zero, and that of subsequent events as the time elapsed since the first event.

- ***Event source***: The location where the event *occurred*. This may be different than the origin of the detection. For example, malicious network traffic from an endpoint may be detected on an organization's outbound proxy, yet the information that should be represented was that the *endpoint* generated malicious network traffic. The name or identity of the endpoint would also not be material – it is only important that it can be determined that some events occurred on one endpoint, while others occurred on a different endpoint. In practice, a vendor compiling a machine learning dataset from internal detection and response data may elect to replace all host identifiers with pseudo-anonymized strings, such as a salted hash of the host name. A researcher using the dataset may then elect to replace these identifiers in some other format suitable to the application.

- ***Reference identifier***: A unique identifier whereby the event can be uniquely referenced.

- ***Context tags***: A list of references to other events that form the context of this event. This field is especially relevant for response actions, as this is where a direct link is drawn between these and detections and observations. Other uses may include compound detections that were made based on other detection or observation events, grouping of incidents, etc. Context may be built automatically in some cases, while in others a human operator may need to provide it. For example, when a response job is created while a specific detection

event is being viewed, the reference identifier of that event may be added as a default context, but the responder may link additional identifiers to create a more accurate reflection.

- ***Event type:*** The type of event that occurred selected from a defined set. At this time a set of "detection", "observation", and "response" is proposed

- ***Taxonomy:*** The taxonomy according to which the event is identified. It is proposed that a specific type of attacker action detection or response event should only be described using a single taxonomy – multiple taxonomies should not be selected to describe the same action space. This field should ideally reference a well-defined and standardized taxonomy that enjoys acceptance among security product vendors and the security community at large, or a derivative thereof.

- ***Schema:*** The structure of event-specific data. Candidate taxonomies for this research do not all define a specific format in which the parameters of an event should be conveyed, so schemas for some event types may need to be defined.

- ***Event description:*** The name of the detection, observation, or response action according to the selected taxonomy. This field *names* the event and perhaps provides a description suitable for human consumption but does not provide any information about the parameters of this *specific* event.

- ***Event parameters:*** Information related to the specific detection or action. Where the *description* fields described above identify the event or action, this field provides information related to the specific event. For example, the detection may have a description of "process injection" (or a subcategory thereof in the selected taxonomy), while this field would provide specific information for this detection instance such as the names of the processes involved. Another example may be where a malicious file is identified by hash (observation event based on a specific indicator of compromise), where this field would contain information such as the file hash, name, location, etc. The schema for this field depends on the detection or action fields described earlier. It should be noted that some detections or actions may not have any parameters associated with them. It is important to note that this field has the potential to contain information that may identify the source of the dataset, and care should be taken in schema selection or creation to specifically guard against this. For example, the name of a malicious document may contain the name of the targeted organization, or the path to a file may contain the local user's account and domain name.

Several open taxonomies were evaluated for the purposes of this research, but none were identified that would be singularly suitable for the structure proposed above. Mature taxonomies that were not associated with specific vendors tended to be focused on one of the proposed event types (detection, observation, response), with little or no provision for the others. This is to be expected, since each serves a specific purpose in this space. For example, the Mitre ATT&CK framework [32] provides for extensive and detailed naming of attacker actions, while the STIX framework [33] provides for detailed descriptions of incident observables. The former, however, does not define a formal schema according to which event parameters should be conveyed, while the latter includes extensive provision for this. As such, additional work may be required to define such schemas as part of this work. While the creation of a new single taxonomy and associated data schemas to fulfil the needs of this research may be possible, the efforts involved in its creation and long-term maintenance become impractical.
Currently, the most suitable taxonomies for attack detection and observation event descriptions are the Mitre ATT&CK and STIX frameworks respectively. In the case of the former, a schema to carry

specific parameters of events will need to be defined. An existing open and mature response action taxonomy directly suited to this research could not be found, although some candidates used in response workflow and orchestration showed some promise. A new taxonomy may need to be defined, preferably as a supplement to an existing response definition framework.

6.5 Data Collection

A well-defined structure for representing attack-response datasets is only one part of the process. Once this is in place, operational data needs to be harvested from the field, suitably anonymized, and converted into a suitable structure. This, however, presents several challenges that make this infeasible on a large scale at this time.
Security product vendors and managed security service providers are the only realistic source for large and diverse datasets that will yield meaningful research results. It is however believed at this time that significant work will be required for such data to be collected and converted into a form that would be useful to research on this topic. Two specific challenges should be considered:

- Each vendor has their own formats and structures for representing detection data, whether this be raw data received from endpoints or supplementary outputs from back-end processing. Vendors will need to create translation processes to convert from their internal proprietary formats to a standardized form. Some aspects of this are however already in place – many vendors already link some of their detection rules to references in the Mitre ATT&CK taxonomy. In such cases, translation would entail only adding the parameter data to converted events using the selected schema. In many cases, incident reporting, both automated and manual, also already makes use of the STIX format.

- Each response action will need to be linked to one or more specific detections or observations. In this regard, there is a lot more diversity in vendor readiness, and a lot depends on their approach to case management. Response actions are usually logged in the context of an incident, which is also linked to a set of detections and observations. A direct relationship between specific detections and responses is however not always in place. This is fit for purpose in the context of incident management and traceability, especially when supplemented with human notes. It is however not ideal in the context of this research. What is required is a more granular association, where specific detections are explicitly marked as being part of the context when a response action is issued. In reality, a response action would often be defined by information beyond just detection or observation events present in the EDR platform or SIEM, such as forensic artefact analysis results, data from other security systems, and the responder's previous experience with similar or identical incidents. Such cases would not be suitable for this type of research at this time.

6.6 Examples

This section provides three illustrative examples of incident data captured and converted into the scheme proposed above. The term *incident* is used here to refer to a collection of datapoints that represent specific malicious actions which led to an attack detection event. Attacks will generally consist of multiple such actions and detections, with smaller (atomic) incident detections being merged over time into larger incidents that contain many detections, observations, and response actions. To avoid confusion, the term *detection incident* will be used to represent the more atomic version, while *managed incident* will refer to the larger incident context being managed by responders.

6.6.1 Scheduled Task

In this example, potentially malicious code was executed on an endpoint through some unknown infection vector. While this event was not detected, the malware's attempt to establish persistence through the creation of a suspiciously-named Windows scheduled task was:

```
{
    "timestamp": "T+0",
    "ref_id": "DET034",
    "context_ref": [],
    "source": "ACMEWS0703",
    "event_type": "detection",
    "taxonomy": "ATTACK_v10",
    "schema": "det_job_artefact",
    "description": "T1053.005",
    "parameters": {
        "name": "AhyRtZXp",
        "execute": "c:\\Windows\\System32\\sectsk.exe -k ad67e4f6bb04674c"
    }
}
```


The detection and response team was alerted, and the analyst on duty decided that the best course of action was to delete the task in question:


```
{
    "timestamp": "T+1",
    "ref_id": "RES004",
    "context_ref": ["DET034"],
    "source": "ACMEWS0703",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "delete_schtsk",
    "parameters": {
        "task_name": "AhyRtZXp",
    }
}
```


The analyst further decided to retrieve a sample of the executable referenced by the scheduled task before deleting it:

```
{
    "timestamp": "T+2",
    "ref_id": "RES005",
    "context_ref": ["DET034"],
    "source": "ACMEWS0703",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "get_file",
    "parameters": {
        "file_path": "c:\\Windows\\System32\\sectsk.exe",
    }
}
{
    "timestamp": "T+3",
    "ref_id": "RES006",
    "context_ref": ["DET034"],
    "source": "ACMEWS0703",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "delete_file",
    "parameters": {
        "file_path": "c:\\Windows\\System32\\sectsk.exe",
    }
}
```


Finally, a list of all the running processes was retrieved for further investigation:


```
{
    "timestamp": "T+4",
    "ref_id": "RES007",
    "context_ref": ["DET034"],
    "source": "ACMEWS0703",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "get_proclist",
    "parameters": { }
}
```


6.6.2 IOC Sighting

This example is set during an ongoing managed incident where a sample of a tool being used by the attacker was successfully retrieved before it could be deleted. The incident response team decided to push a hash-based file indicator to all the endpoints (response job reference IR881) to alert them to any use of the tool. Multiple sighting alerts were received in short order, one of which is shown below:

```
{
    "timestamp": "T+0",
    "ref_id": "ALRT45881",
    "context_ref": ["IR881"],
    "source": "DESKTOP0051",
    "event_type": "observation",
    "taxonomy": "STIX_2v1",
    "schema": "stix_object",
    "description": "indicator",
    "parameters": {
        "pattern": "[file:hashes.'SHA-256' = '...']",
    }
}
```


The SOC team was instructed to isolate any endpoints where the sample was observed from the network. The following entry was reflected in the incident log for the SOC team's instruction to isolate the endpoint referenced above:


```
{
    "timestamp": "T+1",
    "ref_id": "IR893",
    "context_ref": ["IR881", "ALRT45881"],
    "source": "DESKTOP0051",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "net_isolate",
    "parameters": { }
}
```


6.6.3 Credential Hunting

In the final example, the security service provider of an organization was alerted to the potential use of the PsExec tool on the workstation of an IT support agent:

```
{
    "timestamp": "T+0",
    "ref_id": "DETEVT00265332",
    "context_ref": [ ],
    "source": "HELPDESK29",
    "event_type": "detection",
    "taxonomy": "ATTACK_v10",
    "schema": "admin_share_schema",
    "description": "T1021.002",
    "parameters": {
        "filename": "PSEXESVC.exe",
        "source": "RECPTDSK03",
        "account": "Minion.Bob"
    }
}
{
    "timestamp": "T+1",
    "ref_id": "DETEVT00265401",
    "context_ref": [ ],
    "source": "HELPDESK29",
    "event_type": "detection",
    "taxonomy": "ATTACK_v10",
    "schema": "service_schema",
    "description": "T1569.002",
    "parameters": {
        "name": "PsExec",
        "execute": "c:\\Windows\\PSEXESVC.exe"
    }
}
```


Shortly after these events were ingested, the security product's backend identified the combination of events as a likely lateral movement detection incident originating from RECPTDSK03 and targeting endpoint HELPDESK29:


```
{
    "timestamp": "T+2",
    "ref_id": "RTDET0176",
    "context_ref": ["DETEVT00265332", "DETEVT00265401"],
    "source": "HELPDESK29",
    "event_type": "detection",
    "taxonomy": "ATTACK_v10",
    "schema": "lateral_move_schema",
    "description": "TA0008",
    "parameters": {
        "source": "RECPTDSK03",
        "destination": "HELPDESK29",
        "account": "Minion.Bob",
    }
}
```

As the response team started analyzing these events, another detection was made on the targeted endpoint – this time for a credential dumping attempt. In such cases, the product automatically retrieves a list of domain accounts that may have credentials cached on the target endpoint, as these are likely at risk of compromise:


```
{
    "timestamp": "T+3",
    "ref_id": "DETEVT00265401",
    "context_ref": [ ],
    "source": "HELPDESK29",
    "event_type": "detection",
    "taxonomy": "ATTACK_v10",
    "schema": "domain_sec_schema",
    "description": "T1003.001",
    "parameters": {
        "cached_accounts": ["Minion.Bob", "Minion.Kevin", "Minion.Stuart"]
    }
}
```


At this point, it was apparent that the attacker was moving on the network and had reached the workstation of a privileged user. The response team started the containment sequence for such incidents from their standard operating procedures playbook. The first step was to immediately isolate all the endpoints involved. The incident log reflected the issuing of the isolation instruction targeting both endpoints, as well as the job being executed on each. The detection events related to the lateral movement and credential dumping detections were added as references by the incident responder:

```
{
    "timestamp": "T+4",
    "ref_id": "RJB01453",
    "context_ref": ["RTDET0176", "DETEVT00265401"],
    "source": "RESPONSE_CONSOLE",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_contain_v2021.2",
    "description": "net_isolate",
    "parameters": {
        "targets": ["HELPDESK29", "RECPTDSK03"]
    }
}
{
    "timestamp": "T+5",
    "ref_id": "RJ11967",
    "context_ref": ["RTDET0176", "DETEVT00265401", "RJB01453"],
    "source": "HELPDESK29",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "net_isolate",
    "parameters": { }
}
{
    "timestamp": "T+6",
    "ref_id": "RJ11968",
    "context_ref": ["RTDET0176", "DETEVT00265401", "RJB01453"],
    "source": "RECPTDSK03",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_job_v2021.2",
    "description": "net_isolate",
    "parameters": { }
}
```


The next step in the playbook was to suspend all the accounts that may have been compromised, followed by the physical removal of the affected endpoints from the network and their preservation for forensic analysis. These are, however, not actions that the team could perform themselves, so high-priority service requests were logged with the relevant IT teams:

```
{
    "timestamp": "T+7",
    "ref_id": "IIR00461",
    "context_ref": ["RTDET0176", "DETEVT00265401"],
    "source": "RESPONSE_CONSOLE",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_meta_v2021.2",
    "description": "service_request",
    "parameters": {
        "request": "suspend_account",
        "targets": ["Minion.Bob", "Minion.Kevin", "Minion.Stuart"]
    }
}
{
    "timestamp": "T+8",
    "ref_id": "IIR00462",
    "context_ref": ["RTDET0176", "DETEVT00265401"],
    "source": "RESPONSE_CONSOLE",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_meta_v2021.2",
    "description": "service_request",
    "parameters": {
        "request": "quarantine_asset",
        "targets": ["HELPDESK29", "RECPTDSK03"]
    }
}
```


The final step in this part of the team's documented standard operating procedure called for a formal managed incident to be logged and escalated, allowing for the organization's incident response and management plans to be put into action:


```
{
    "timestamp": "T+9",
    "ref_id": "IIRM0015",
    "context_ref": ["RTDET0176", "DETEVT00265401"],
    "source": "RESPONSE_CONSOLE",
    "event_type": "response",
    "taxonomy": "IIR_ACT_SPACE_v01",
    "schema": "response_meta_v2021.2",
    "description": "declare_incident",
    "parameters": {
        "severity": "medium",
    }
}
```

6.7 Conclusion

We have discussed the need for developing assistive technologies for detection and response teams, in order to enable them to focus their efforts on situations that require their rare expertise. Providing teams with insights on how similar incidents were handled previously by their peers is one promising direction of research. Such research will however require sufficiently large and diverse datasets representing observations and human responses for analysis and model learning.

We further discussed the challenges involved in assembling the required datasets, the most significant of which was the lack of fine-grained causal links between observed security events and human actions. Without this context, it is hardly possible to construct robust response recommender models.

In the final part of this Section, we proposed a scheme for representing detection and response events in a suitable form, discussed some of the essential elements that need to be in place, and potential data collection strategies. These recommendations are not intended as a comprehensive blueprint for production implementation, but are rather meant as a conceptual guideline for security detection and response system builders. Vendors may choose different engineering approaches to reaching the goals set out here in their specific technology stacks. It is however crucial that the end results can be represented in a standardized format to enable interoperability between products and vendors. Without such standardization, it is unlikely that a comprehensive corpus of data can be assembled from multiple sources to enable meaningful research in this field.

7 Measuring suspicious behaviour using host aggregation data

7.1 Introduction

Endpoint detection and response solutions typically consist of endpoint sensors (software designed to collect local events on a monitored computer and send them for analysis) and back-end systems (software running on servers or in the cloud that receive and process events from monitored computers). A vast number of events are generated per second on each monitored system. Some combination of these events, when considered in the context of historical events witnessed on that particular system, can indicate that suspicious activity is occurring, which may indicate that a security breach is ongoing. The ratio of true positive suspicious or malicious events to benign events is typically very low.

H*ost aggregation* is the process of profiling the behaviour of a monitored computer over a defined period of time, based on both current and historical events witnessed on that host. Based on historical data of known true positive security incidents, it is possible to estimate how likely a host aggregation provides information that would require closer attention, which is essentially the first step in the incident response process. In this section, an approach for this task is proposed that builds on earlier work presented in Sections 3.2 and 3.3 of SAPPAN D4.4, where a supervised model was trained to estimate whether a security incident is a true or false positive (which is also conceptually similar to the setting of Section 3 of this document, though the data we deal with there is different). A host aggregation contains similar features as the incident data, but here the focus is moved from incidents to hosts, and instead of trying to determine a true or false positive incident, the proposed methodology estimates how suspiciously a host has behaved in the latest aggregation time frame.

7.2 Objectives and use cases

The motivation for this research is to create a mechanism to measure how similar a host's behaviour is compared with other hosts that have historically caused true positive incidents. This similarity, represented as a score value, can then be examined by human analysts or machine heuristics for decision making purposes and response activities planning. For initial use cases, the score values will be used as additional information for manual threat hunter work.

7.2.1 Filtering of hosts under investigation

Scores are obtained for all hosts, over a single day, and used to determine cases that require closer attention. This is done by constructing an allow-listing tool. Hosts with scores below a certain threshold are ignored, and triage occurs on all hosts with scores above that threshold.

7.2.2 Prediction of alerts

Daily host aggregation scores are collected and used to construct a time series illustrating how each host behaves over time. This allows sudden changes and gradual drifts in scores to be detected automatically. Score drift can indicate that something might have changed in a host, warranting further investigation even before an actual incident takes place.

7.3 Approach

This research is based on previous work, described in SAPPAN D4.4, where a model, designed to estimate the likelihood of an incident being a true or false positive, was trained on labelled historical incident data. This proposed host aggregation approach differs in the fact that the method does not have access to ground truth values indicating whether a host is behaving in a suspicious manner or not, and so the model – in principle – cannot be trained in a supervised manner. However, as the features in the incidents are the same as in the host aggregations, we can do predictions with a model trained on incident data, but with host aggregation data as input. The predicted score as returned by the model is not technically the likelihood of a host aggregation being a true positive incident, but the interpretation of the score is similar: the higher the value is, the more suspicious the behaviour of the host is.

7.4 Data

The incident classification model was trained using tag count features described below. For host aggregations, additional feature set designed to improve the distinguishing power of the model was used.

7.4.1 Tag counts

A *tag* refers to an interesting event that a sensor detected on a host. Tags are defined by experts and are labelled with severity levels ranging from informative to severe. A host aggregation contains counts of each tag seen during the allotted time frame. The least severe tags are filtered out since they are highly prevalent and do not contribute enough to detection of incidents worthy of investigation. Just over 1000 tags are used as features for the model. Tag counts are not scaled – it is assumed that, for certain tags, the magnitude of counts is more interesting than their relative proportion in the data.

7.4.2 Additional features

The following new features are also added as inputs to the model:

- **Alert score**: A score that describes how suspiciously a host has been behaving recently.
- **Anomality score**: Score that combines both a measure of host's suspicious behaviour and a measure of how unusual the host's behaviour has been in comparison to other hosts.
- **Number of unique tags seen by the host**: Even with less severe tags, an increased number of tags suggests more suspicious behaviour on a host.
- **Tag severity level count**: As stated, each tag has a severity level set by an expert. For each severity level used (low, medium, high, critical), the sum of the tags seen is calculated; more severe tags suggest more weight on that tag combination.
- **Client information**: The organization where the host is assigned. Hosts in different organizations behave differently, and behaviours differ between organizations.

Combining the novel features above with tag counts brings the total number of features for the model to around 1200.

7.5 Model choice

The model selection was conducted as described in Section 3.3 of SAPPAN D4.4, using the incident data which we have verdicts (ground truth) for. Several different supervised machine learning algorithms were tested, including logistic regression, random forest, and gradient boosting. The performance of all models was reasonably similar, suggesting that detecting a signal from the data was not dependent on model choice. Random forest was chosen, as it outperformed the others with a minor margin, required less feature pre-processing, and allowed straightforward analysis of feature importance values. In addition, instead of hard classification, the chosen model outputs a value describing the likelihood of an incident being a true positive. With ensemble models like random forest, this can be computed as the portion of individual weak models predicting a class.

7.5.1 Hyperparameter tuning

Hyperparameter optimization was carried out using exhaustive grid search cross validation. Parameter tuning was conducted over: number of trees, number of features tried for each split, maximum tree depth, minimum number of samples required for a leaf node, and minimum number of samples required for training a single tree. The loss function was the out-of-bag error of the forest.

With an optimal hyperparameter configuration, the random forest model was evaluated using area under ROC curve (AUC) as the metric for incident verdict classification. The result, 0.84, was considered satisfactory for the purpose.

7.6 Evaluation

Without ground truth information, it is difficult to derive a straightforward indicator of how this approach performs for host aggregation threat scoring. The distribution of scores for hosts on a single day is presented in Figure 11. The distribution shows very much what would be expected, with most of the hosts receiving very low scores.

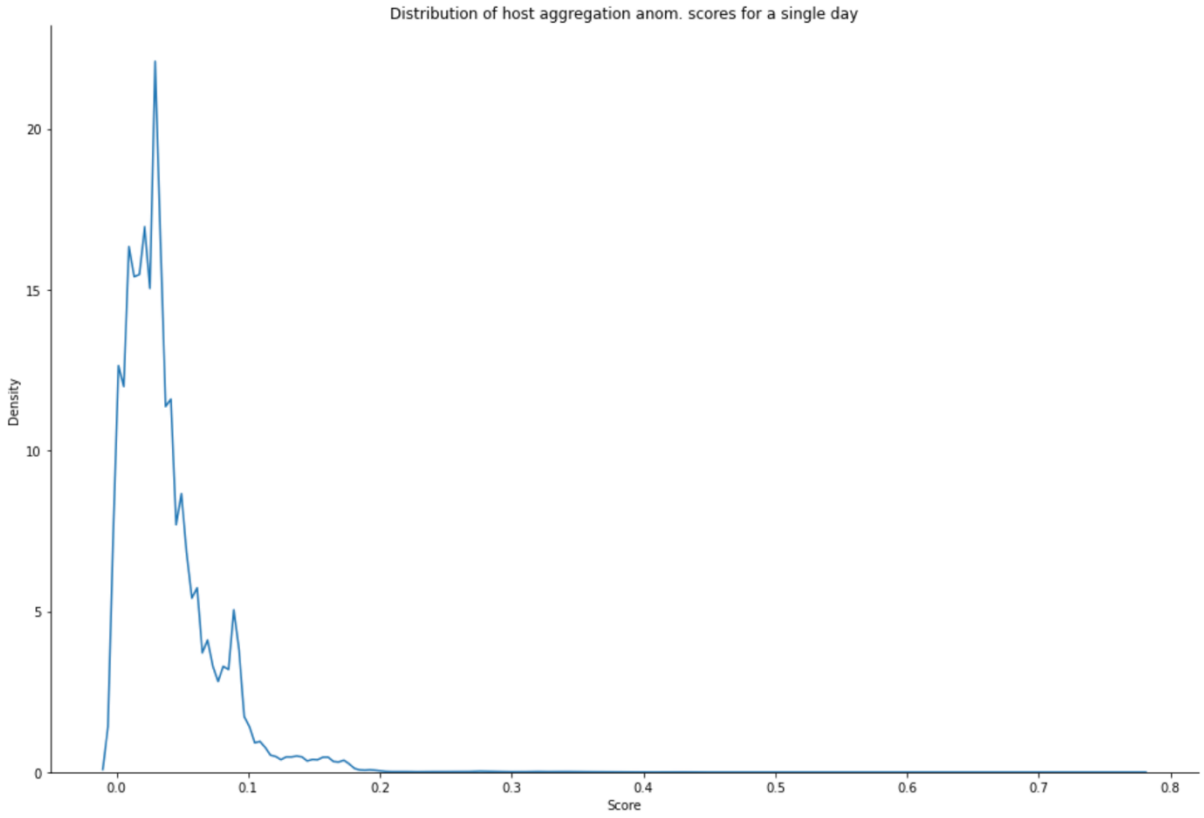

Figure 11: Example of score distribution

For evaluation purposes, the performance of the proposed model was simulated for a single week of real data. The model was retrained daily and scores for each day's host aggregations were calculated. Observations of how many of the monitored hosts caused incidents in the following days were recorded in order to determine whether potential hosts with a reasonable score threshold could be ignored. Note that a one-day time window is very coarse – in practice scores would be calculated far more often, allowing a more fine-grained time series.

In order to determine how many true positive incidents appeared in the next seven days for hosts that might have been filtered out, three threshold values – scores of 0.10, 0.20 and 0.30 – were considered. The results are depicted in Table 5.

Threshold	Percentage of hosts filtered (averaged over all days)	Filtered hosts with one or more incidents in next seven days	Unfiltered hosts with incidents in next seven days
0.10	99.293	**14**	**8**
0.20	99.743	21	1
0.30	**99.885**	22	0

Table 5: Results of the filtering experiments

It should be noted that out of the 14 possibly missed cases, a total of eight obtained continuously very low scores of under 0.05, suggesting that, for these cases, the features used did not contain a signal that could be applied to determine an upcoming incident. This might further suggest that the incidents were results of sudden changes in the hosts and could not have been predicted from the obtained scores.

Figure 12 depicts a visualization of scores for one week for a single host with very stable, predictable behaviour. Here, one can see that for such hosts detecting notable changes would be very straightforward. Behaviour like this was common amongst all the hosts in our data.

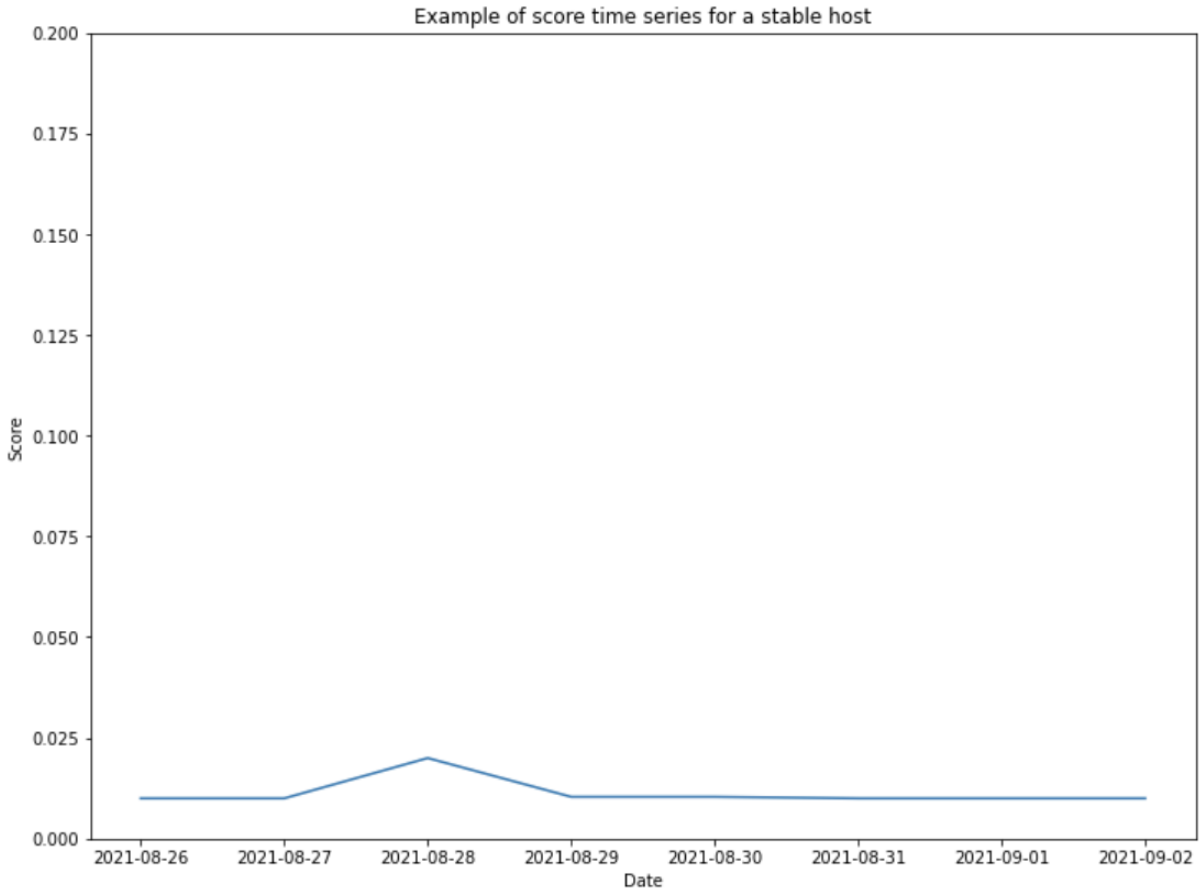

Figure 12: One week of scores for a stable host

Figure 13 depicts a similar time series for a host with a more eventful history. The host underwent notable changes between successive days. However, this host also produced constantly higher scores than most of the other hosts, with notable changes in its own history occurring multiple times during the one-week analysis period. The depicted host did not produce any actual incidents during our analysis period.

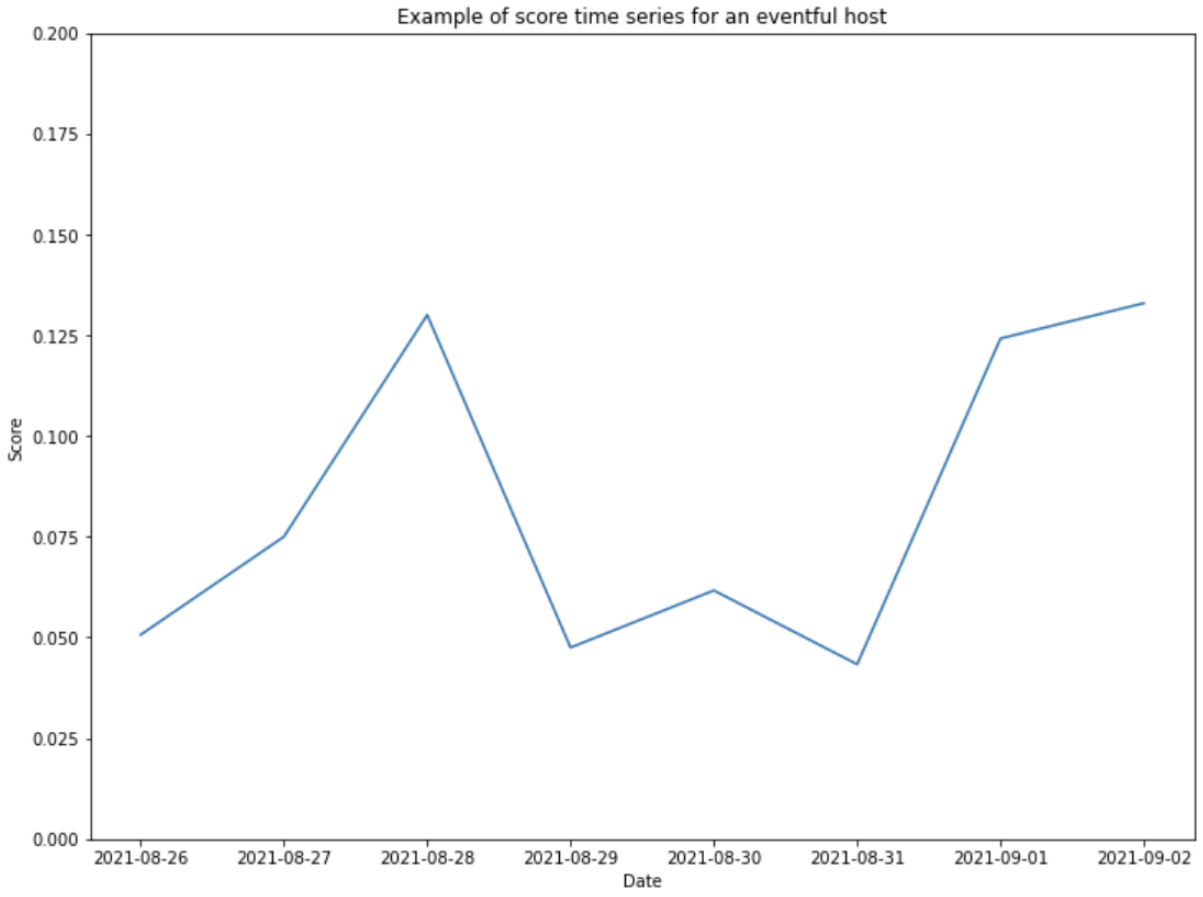

Figure 13: One week of scores for an eventful host

All in all, determining an upcoming incident using this methodology has potential, but as illustrated in this report, it remains difficult to estimate the relationship between host aggregate scores and later incidents. For the illustrated use case, this method still seems highly applicable – by manually checking any host aggregation receiving a score above a pre-selected threshold, a security analyst might catch potential incidents before they take place or detect incidents that might have otherwise gone unnoticed. Naturally, setting a suitable threshold is crucial to prevent an excessive amount of manual work. Studies for setting this threshold value optimally are currently ongoing.

7.7 Challenges and Future Work

The illustrated use case for filtering a majority of monitored hosts and thus reducing analysis workload seems to perform reasonably well. This approach is currently under evaluation in real-world scenarios. It should be stressed that the decision to filter hosts for signs of potential incidents would never rely on just host aggregation score – other information not in the scope of this report would be included. Thus, the few cases in our evaluation that would have been missed are not of concern.

A greater feature engineering effort would likely improve model performance. The current number of features is vast, and a majority of them contribute very little. More than the curse of dimensionality, we are concerned that some features carry too much weight, causing some other potentially interesting features to have little effect. This line of analysis is especially relevant in cases where the model predicts low scores for hosts that later cause incidents.

Observing scores as a time series suggests that most hosts behave in a stable manner, with very limited fluctuations in daily score values. After collecting scores for a longer period, it should be possible to run further time series analyses and identify changes or trends in score values worth flagging to an analyst.

8 Conclusion

Both security vendors and the European Union are putting a great deal of effort and resources into researching mechanisms to support security professionals and to intelligently automate tasks in breach detection and incident response workflows. Understanding the type, severity and other relevant characteristics and context of a security incident that triggered an alert can be used to choose appropriate response actions, which can be either suggested to security personnel or, in certain cases, even carried out automatically. This report presented several methods and approaches, developed in Task 4.3 of the SAPPAN project, which contribute to more effective and efficient incident response. In particular, Section 3 described clustering of security incidents detected by an endpoint detection and response solution in order to enable the execution of bulk incident processing actions, allowing security analysts to resolve many more incidents than they would without such a technology. Section 4 presented an algorithm to recommend denial of service (DoS) attack mitigation rules for filtering volumetric DoS while the attack is underway. Section 5 presented a contextual attack chain modelling method designed to predict future attack steps and to support security analysts in choosing appropriate response actions. In Section 6, we presented and discussed the methodology for building datasets that can be used to automate response actions, as the first step in investigating the feasibility of practical machine learning-based technologies for leveraging previous successful response operations in handling occurrences of new similar attacks. Finally, in Section 7, a method for estimating the 'suspiciousness' level of endpoint behaviour for better response preparedness was described.

Given the overall complexity of the incident response task, we believe that it should be approached from multiple directions and with an assortment of techniques. We also strongly believe, and attempted to provide evidence of that in this report, that machine learning – and more generally data-driven – methods can be very valuable in supporting security professionals in incident response.

9 References

[1] S. Guizani. A K-means clustering-based security framework for mobile data mining. Wireless Communications and Mobile Computing, 16:3449-3454, 2016.

[2] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber. System log clustering approaches for cyber security applications: A survey. Computers & Security, 92, May 2020.

[3] C. Raj, L. Khular, and G. Raj. Clustering Based Incident Handling for Anomaly Detection in Cloud Infrastructures. 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020, pp. 611-616.

[4] D. Silva, M. Dell-Amico, M. Hart, K. Roundy, and D. Kats. Hierarchical Incident Clustering for Security Operation Centers. Interactive Data Exploration and Analytics, August 2018.

[5] R. Hummel and C. Hildebrand, Crossing the 10 million mark: Ddos attacks in 2020, 2021. Available: https://www.netscout.com/blog/asert/crossing-10-million-mark-ddos-attacks-2020

[6] D. Douglas, J. Santanna, R. de Oliveira Schmidt, L. Granville, and A. Pras, Booters: can anything justify distributed denial-of-service (ddos) attacks for hire? Journal of information, communication and ethics in society}, vol. 15, no. 1, pp. 90--104, Jan. 2017.

[7] Netscout threat intelligence report shows a dramatic increase in multivector ddos attacks in first-half 2020, 2020. Available: https://www.netscout.com/netscouts-threat-intelligence-report-1H2020

[8] T. M. Gil and M. Poletto, Multops: a data-structure for bandwidth attack detection, in Proceedings of the 10th conference on USENIX Security Symposium - Volume 10, ser. SSYM'01. Berkeley, CA, USA: USENIX Association, 2001, pp. 3—3. Available: http://dl.acm.org/citation.cfm?id=1267612.1267615

[9] T. Peng, C. Leckie, and K. Ramamohanarao, Detecting distributed denial of service attacks using source ip address monitoring, In Proceedings of the Third International IFIP-TC6 Networking Conference.

[10] J. Jung, B. Krishnamurthy, and M. Rabinovich, Flash crowds and denial of service attacks: characterization and implications for cdns and web sites, in Proceedings of the 11th international conference on World Wide Web, New York, NY, USA: ACM, 2002, pp. 293--304. Available: http://doi.acm.org/10.1145/511446.511485

[11] P. DU and S. ABE, Ip packet size entropy-based scheme for detection of dos/ddos attacks, IE-ICE Transactions on Information and Systems, vol. E91.D, no. 5, pp. 1274--1281, 2008.

[12] D. Gavrilis and E. Dermatas, Real-time detection of distributed denial-of-service attacks using rbf networks and statistical features, Comput. Netw. ISDN Syst., vol. 48, no. 2, pp. 235--245, Jun. 2005. Available: http://dx.doi.org/10.1016/j.comnet.2004.08.014

[13] P. S. Saini, S. Behal, and S. Bhatia, Detection of ddos attacks using machine learning algorithms, in 2020 7th International Conference on Computing for Sustainable Global Development INDIACom, 2020, pp. 16--21.

[14] Why firewalls and intrusion prevention systems (ips) fall short on ddos protection, 2021. Available: https://techdata.ca/arbornetworks/files/ARBOR_TB_IPS_EN.PDF

[15] P. Ferguson and D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing}, RFC 2827 (Best Current Practice), Internet Engineering Task Force, May 2000, updated by RFC 3704. Available: http://www.ietf.org/rfc/rfc2827.txt

[16] J. Li, J. Mirkovic, T. Ehrenkranz, M. Wang, P. Reiher, and L. Zhang, Learning the valid incoming direction of ip packets, Comput. Netw., vol. 52, no. 2, pp. 399--417, Feb. 2008. Available: http://dx.doi.org/10.1016/j.comnet.2007.09.024

[17] L. Xie, J. Bi, and J. Wu, An authentication based source address spoofing prevention method deployed in ipv6 edge network, in Proceedings of the 7th international conference on Computational Science, Part IV: ICCS 2007, ser. ICCS '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 801--808. Available: http://dx.doi.org/10.1007/978-3-540-72590-9_121

[18] P. Goldschmidt and J. Kučera, Defense against syn flood dos attacks using network-based mitigation techniques, 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2021, pp. 772--777.

[19] H. Wang, C. Jin, and K. G. Shin, Defense against spoofed IP traffic using hop-count filtering, IEEE/ACM Trans. Netw., vol. 15, no. 1, Feb. 2007.

[20] X. Xu, Y. Sun, and Z. Huang, Defending ddos attacks using hidden Markov models and cooperative reinforcement learning, Intelligence and Security Informatics, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, vol. 4430, pp. 196--207. Available: http://dx.doi.org/10.1007/978-3-540-71549-8_17

[21] Z. Yuan and C. Wang, An improved network traffic classification algorithm based on hadoop decision tree, 2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS), 2016, pp. 53--56.

[22] Decision trees, online. Available: https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart

[23] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, Developing realistic distributed denial of service (ddos) attack dataset and taxonomy, in 2019 International Carnahan Conference on Security Technology (ICCST), 2019, pp. 1--8.

[24] Blake E Strom, Joseph A Battaglia, Michael S Kemmerer, William Kupersanin, Douglas P Miller, Craig Wampler, Sean M Whitley, and Ross D Wolf. Finding cyber threats with ATT&CK-based analytics. Technical Report MTR170202, The MITRE Corporation, Bedford, MA, 2017.

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

[26] Tarun Yadav and Arvind Mallari Rao. Technical aspects of cyber kill chain. In International Symposium on Security in Computing and Communication, pages 438–452. Springer, 2015.

[27] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. Machine learning, 85(3):333–359, 2011.

[28] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. International Journal of Data Warehousing and Mining (IJDWM), 3(3):1–13, 2007.

[29] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 694–699, 2002.

[30] Meelis Kull, Telmo M Silva Filho, and Peter Flach. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. Electronic Journal of Statistics, 11(2):5052–5080, 2017.

[31] Yussuf Ahmed, A Taufiq Asyhari, and Md Arafatur Rahman. A cyber kill chain approach for detecting advanced persistent threats. CMC-COMPUTERS MATERIALS & CONTINUA, 67(2):2497–2513, 2021.

[32] https://attack.mitre.org/

[33] https://oasis-open.github.io/cti-documentation/