Okutan and Yang *Cybersecurity* (2019) 2:15
https://doi.org/10.1186/s42400-019-0032-0

Cybersecurity

RESEARCH Open Access

# ASSERT: attack synthesis and separation with entropy redistribution towards predictive cyber defense

Ahmet Okutan* and Shanchieh Jay Yang

**Abstract**

The sophistication of cyberattacks penetrating into enterprise networks has called for predictive defense beyond intrusion detection, where different attack strategies can be analyzed and used to anticipate next malicious actions, especially the unusual ones. Unfortunately, traditional predictive analytics or machine learning techniques that require training data of known attack strategies are not practical, given the scarcity of representative data and the evolving nature of cyberattacks. This paper describes the design and evaluation of a novel automated system, ASSERT, which continuously synthesizes and separates cyberattack behavior models to enable better prediction of future actions. It takes streaming malicious event evidences as inputs, abstracts them to edge-based behavior aggregates, and associates the edges to attack models, where each represents a unique and collective attack behavior. It follows a dynamic Bayesian-based model generation approach to determine when a new attack behavior is present, and creates new attack models by maximizing a cluster validity index. ASSERT generates empirical attack models by separating evidences and use the generated models to predict unseen future incidents. It continuously evaluates the quality of the model separation and triggers a re-clustering process when needed. Through the use of 2017 National Collegiate Penetration Testing Competition data, this work demonstrates the effectiveness of ASSERT in terms of the quality of the generated empirical models and the predictability of future actions using the models.

**Keywords:** Cyber security, Dynamic bayesian classifier, Clustering KL divergence

## Introduction

As new system vulnerabilities are discovered and attack tools become even more prevalent, cyber attackers may employ a variety of evolving strategies with a plethora of exploits. Symantec's 2017 Internet Security Threat Report (Symantec 2017) suggested that attacking tactics continued to change and many attackers used whatever tools on hand rather than focused on sophisticated techniques. This means how attackers penetrate into a network may be situational and diverse, depending on what they discover and what is available. The diverse and changing tactics pose challenges for security analysts to recognize and comprehend the ongoing malicious activities and emerging threats. Imagine a system that can process a significant volume of observables produced by intrusion detection systems, and continuously synthesize and update a manageable set of *'empirical attack models'* that reflect the different 'how', 'where', and 'what' attack activities are present in the network. Such a system will assist security analysts to prioritize and anticipate critical attacks, hence offering a robust predictive cyber defense even in the presence of evolving and diverse cyberattack tactics.

Traditional intrusion detection systems (Zhang et al. 2008; Li et al. 2012) differentiate malicious events from benign ones to alert security analysts for potentially malicious activities. It is not uncommon for analysts to be overwhelmed by the potentially significant volumes of false positives or less critical alerts due to scanning activities. Alert correlation systems (Valeur et al. 2004; Ning et al. 2004; Yang et al. 2009) attempt to match the intrusion alerts with a priori known models based on expert knowledge from previously observed incidents. An attack episode, however, may be composed of multiple stages where the scanning techniques, exploits, and targets are

*Correspondence: axoeec@rit.edu
Computer Engineering, Rochester Institute of Technology, Rochester, NY, USA

likely to change depending on the attackers' knowledge, preference, or just situational. The notion of training with ground truth or sole reliance on expert knowledge will not be sufficient to provide timely and robust attack models that can be used to predict future attacks. Complementing existing intrusion detection and alert correlation works, this paper aims at separating intrusion alerts to generate emerging attack behavior models, some more critical than others, for better analyses and predictability. This paper presents ASSERT – a novel, automated and configurable system that separates unlabeled observables of malicious cyber activities, and generates and refines attack models as they emerge. ASSERT analyzes the aggregates of alert attributes as non-parameterized histograms, a general form to represent categorical features that are prevalent in cyber observables. Information theoretical techniques are used to process the non-parameterized histograms to determine the likelihood and, thus, the posterior match between the behavior aggregates and the attack models. Each empirically generated attack model then represents the collective behavior exhibited by a subset of intrusion alerts. This allows the analysts to focus on models that contain more critical activities even if the entirety of intrusion alerts is overwhelming.

One may question the possibility to create attack models without ground truth of how cyberattacks progress, especially with the evolving and situational nature of attack behaviors. ASSERT addresses these challenges by introducing a semi-supervised Bayesian learning framework where the max-posteriors are converted to distances to the model centroid so as to assess and maximize the separation of models. More specifically, each aggregate of evidences (malicious activities occurring on the same source-target IP pair over a reasonable period of time) is either associated with the max-posterior attack model or used to generate a new model depending on which of the two gives a higher cluster validity index calculated using the max-posterior distance. Attack models created will be continuously re-assessed as new evidences emerge. The aggregates of evidences may change over time and ASSERT 'shuffles' these aggregates through re-clustering when the cluster validity index is low. The occasional shuffling allows ASSERT to recover from early mis-association of evidences to models, while allowing continuous and computationally efficient Bayesian-based separation of evidences and synthesis of attack models. Figure 1 shows an overview of ASSERT architecture, along with its connection to experimental data and external assessment modules, in terms of both Jensen-Shannon Divergence (JSD) (Lin 1991) and potential predictability using the generated attack models.

The rest of the paper is organized as follows. "Background and related work" section summarizes the previous works on extracting the intrinsic spatiotemporal patterns in cyberattacks and classifying unlabeled and streaming network attack data. "Methodology: ASSERT" section gives the details of the ASSERT system including the use of dynamic Bayesian classifier with Kullback-Leibler divergence (KLD) (Kullback and Leibler 1951), model generation and assessment with Wemmert-Gancarski Index (WGI) (Wemmert et al. 2000), and shuffling with Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996). "Design of experiments" section presents the design of experiments, and "Results" section discusses the results to demonstrate ASSERT's effectiveness in generating quality attack models over time and its potential to improve prediction of future attack actions. Finally, "Conclusion" section presents concluding remarks as well as opportunities for future works.

## Background and related work

The need to recognize different types of cyberattacks is not new. Early works created taxonomies of cyberattacks based on known attack strategies and compiled a list of categories that an attack can be associated with (Hansman and Hunt 2005). Different taxonomies categorize cyberattacks based on various criteria e.g. the vulnerability behind an attack or its observed consequences. These taxonomies serve the purpose to provide a foundational understanding of different types of attacks, allowing technical solutions to focus on detecting or differentiating specific attacks.

ASSERT aims at processing streaming intrusion alerts and separate observables of malicious events into empirically generated attack models that can better differentiate attack behaviors and thus help predict future actions. Using a dynamic Bayesian approach, it focuses on grouping intrusion alerts to create a collective behavior, not necessarily a similar one. This is different from many previous studies where similar observables are 'clustered' together to identify similar behaving end hosts (Xu et al. 2011) or traffic flows (McGregor et al. 2004; Shadi et al. 2017; Song and Chen 2007).

Much research in the past two decades on cyberattack modeling aims at enhancing the ability to detect intrusion i.e. differentiating between malicious and benign activities from packet captures or traffic flows (Subba et al. 2016; Haddadi et al. 2010; Zhang et al. 2008; Li et al. 2012). Intrusion detection ideally enables reactive measures, such as stopping the detected attacks if possible or blocking intrusion from the same entry point again. Current practices on using intrusion detection systems rely much upon security analysts to determine the proper course of action, which can be time consuming and ineffective. One way to alleviate this challenge is to automatically classify intrusion alerts to different attack models defined in a way that can help assist predict

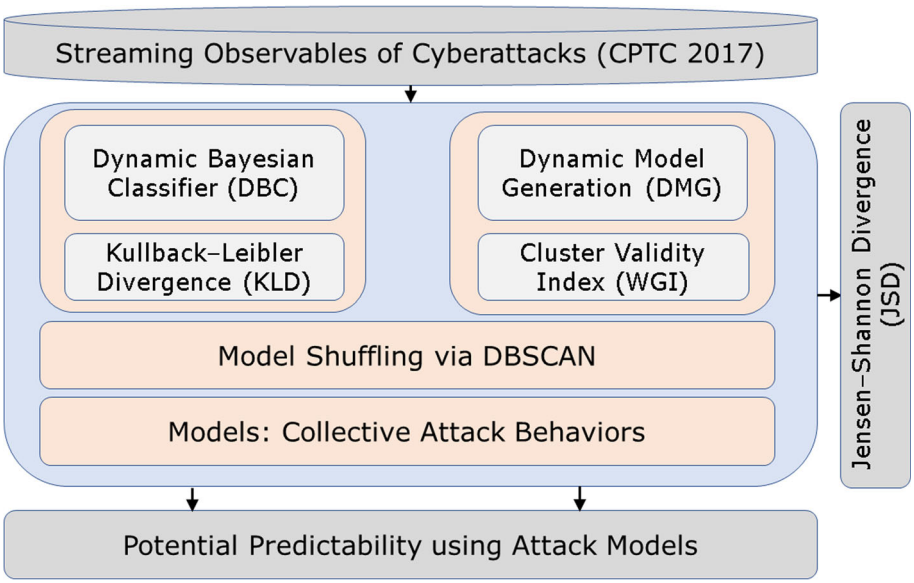


**Fig. 1** The ASSERT architecture and its connection to external modules

future actions. Past research works (Luo et al. 2016) and (Bolzoni et al. 2009) have suggested a similar concept and the need to integrate intrusion detection systems with attack classifiers.

There are several ways to model cyberattacks. Al-Mohannadi et al. provided an overview of some of the popular methods to model network attacks (Al-Mohannadi et al. 2016) , and Yang et al. summarized a number of modeling approaches used for predicting future attacks (Yang et al. 2014). One common approach used by many was using attack graphs to model the system vulnerabilities in a network and the inter-dependencies between these vulnerabilities. Also commonly referred to by the community has been the kill chain model that describes how an attack might progress as a chain of actions from reconnaissance, exploitation, lateral movements, evasions, etc. The Capability-Opportunity-Intent (COI) model characterizes cyberattacks based on the motif and capability of the attacker, the infrastructure of the network being attacked, and the defendability of the attacker (Salerno et al. 2010). These modeling approaches (attack graphs, kill chain, and COI) have been used by many researchers to develop algorithms to correlate intrusion alerts or to predict future attack actions. For example, Wang et al. proposed to use probabilistic attack graphs to correlate isolated alerts into attack scenarios, so as to efficiently identify the vulnerabilities that could be exploited in a network (Wang et al. 2006; Wang et al. 2010). Homer et al. discussed several methods to determine parts of an attack graph that would not help identify critical security threats and group similar attack steps as virtual nodes to increase the understandability of the data flow in a network (Homer et al. 68). Noel et al. introduced CyGraph as a tool for cyber warfare analytics which builds an attack graph model that maps the potential attack paths through a network (Noel et al. 2016). Aguessy et al. presented a Bayesian Attack Model (BAM) for dynamic risk assessment (Legany et al. 2006). They combined the ideas of a topological attack graph and a Bayesian network in order to represent all possible attacks in an information system. The BAM approach aimed at associating possible attacks on a network with probabilities so that an analyst could identify the major vulnerabilities in the network. Shandilya et al. presented an overview of the state-of-the-art technologies in attack graph generation (Shandilya et al. 2014).

The above attack-graph based approaches primarily focused on leveraging, and thus depended on the knowledge of the system vulnerabilities and network configurations. Other works have focused on analyzing the attackers' behaviors by examining data reflecting malicious activities. For example, Chen et al. show that spatiotemporal statistics of malicious traffic can be indicative of major attacker fingerprints and their target selection schemes (Chen et al. 2015). Fava et al. build Variable Length Markov Models based on sequences of intrusion alerts to predict future attack actions (Fava et al. 2008). Du and Yang suggest to map intrusion alerts to an Attack Social Graph (ASG) that can be used to reveal attack patterns (Du and Yang 2011). Strapp and Yang expand the work by Du and Yang and use a dynamic Bayesian analysis to find the best fit of a new malicious activity to a subgraph of ASG given the graph compactness as defined by the posterior (Strapp and Yang 2014). These

studies shed lights on the spatial and temporal modeling approaches that could capture cyberattack patterns from the attacker's perspective. This research builds upon these foundations and develops a computationally efficient dynamic Bayesian approach with Kullback-Leibler divergence and entropy redistribution to process intrusion alerts and create attack models in an online manner without a priori knowledge on either network topology or attack behaviors.

Traditional predictive analytics approaches require training data of known attacks which may not be easy to gather, due to the scarcity of the predictive indicators and the ground truth. Although some prior works use the observables of malicious activities reported by known global anti-virus programs as indicators (Yen et al. 2014; Bilge et al. 2017), most of the time it is difficult to gather the ground truth for a target. And without a priori or ground truth on the attack data, it is challenging to assess the quality of the generated empirical attack models. Are they similar? Will there be too many? Yen et al. state that they do not have ground truth for many aspects of their investigation, and use indirect indicators instead (Yen et al. 2014). A prior study (Liu et al. 2015) states that reported cybersecurity incidents that could serve as ground-truth are vastly under reported and uses the reported public incidents from the VERIS Community Database (VCDB) (VERIS 2018), Hackmageddon (Hackmageddon 2018 and Web Hacking Incidents Database (WASC 2018). Bilge et al. state that obtaining labeled data that is sufficiently comprehensive to capture all clean and risky host profiles in the wild is nearly impossible, because no malware detection solution can attain perfection due to the known arms-race with the cyber attackers (Bilge et al. 2017). When there is no labeled data, clustering could be used as an alternative unsupervised learning approach where similar instances are grouped together based on their attributes. Instances within the same cluster are expected to have a high similarity, whereas the instances of different clusters are expected to be dissimilar. A cluster validity index takes into account the separation between clusters along with the coherence of the data points within each cluster. Previous studies (Wang et al. 2009; Kovács et al. 2005) have suggested the use of several cluster validity indices to measure the quality of a clustering process. The concept of the cluster validity index coincides with the needs to assess the quality of the generated models. This work, thus, adopts Wemmert-Gancarski Index (WGI) to complement the dynamic Bayesian approach to assess the quality of the generated and updated models regularly.

Classification of streaming data is a relatively new but growing problem. Streaming data is defined as data that enters the system one observable at a time. One common method for the classification of streaming data is to use ensemble algorithms such as those described by Street and Kim (Street and Kim 2001) and Kuncheva (Kuncheva 2008). These ensemble algorithms combine the results of multiple types of classifiers on a subset of the data stream. Another common practice for classifying streaming data is to store data points in memory until a large enough subset is gathered and then perform the classification on these data chunks. The classification can either be done using an ensemble algorithm or by clustering (O'Callaghan et al. 2002). Unlike the previous works, ASSERT is based on a novel Bayesian based model generation approach which enables the classification of the streaming observables efficiently. The quality of the generated attack models is regularly assessed with WGI and a systematic density based clustering approach is used to optimize earlier attack models.

## Methodology: ASSERT

Intrusion alerts are commonly assessed by security analysts with their statistics and statistical distributions. ASSERT uses a set of non-parameterized histograms that represent the statistically aggregated behavior as exhibited by the intrusion alerts. The non-parameterized histograms generalize the modeling approach for specific attack attributes, instead of assuming the attack behaviors are known a priori. Each set of histograms, for example, target IPs, target ports, alert categories, and alert signatures, is used to represent an attack behavior model that represents a set of potentially related malicious activities. Intrusion alerts are first aggregated into 'edges' based on common source and target IP addresses. These edge-based aggregates will then be classified into an attack model that maximizes the posterior probability under the Bayesian framework, or used to create a new model. This work assumes that each edge-based aggregate in the same time proximity reflects an attack behavior. This assumption is reasonable since it is unlikely multiple attackers will use, or spoof, the same source IP to attack the same target IP in the same time frame without coordination. If there is coordination, the collective behavior is what ASSERT meant to capture. The aggregation enables the comparison of non-parameterized histograms and thus generation of empirical models.

### The overall process flow

ASSERT uses a graph $G$ to represent the observed malicious activities as exhibited by intrusion alerts. A directed edge $e$ connecting two nodes in $G$ reflects at least one evidence $v$ showing a malicious action from the source node to the target node, where the nodes are naturally indexed with the corresponding IP addresses given by the intrusion alerts. In reality, an edge typically contains a number of evidences accumulated over a period of time to reflect a

collective attack behavior exhibited by a specific combination of intrusion signatures, port numbers, *etc.*. Each edge is represented by the non-parameterized histograms, and either matched to max-posterior attack model or used to create a new model if the histograms deviate too much from the existing models. As a subset of $G$, an attack model is a collection of edges and is represented by the histograms generated from the aggregated statistics of all edges within the model. Note that the empirically generated models evolve over time since the edges added into a model contribute to the histograms associated with that model. This iterative process, as shown in Fig. 2, allows the generation of empirical models without a priori knowledge of any specific attacks or network configuration.

When a new alert or evidence $v$ is received, it is added to an existing edge $e$ on the overall graph $G$ between the source and target IP addresses of the processed observable. Note that this graph $G$ is not the same as the attack graph used in the literature; it is merely a representation of the collection of the edges. If no such edge exists, then it is created and added to the graph. The process of analyzing a new evidence is described in Algorithm 1.

**Algorithm 1** Add evidence $v$ to a new or existing edge $e$ in an attack graph $G$ where a dynamic attack model is represented by $\Omega$.

```
1:  function AddEvidence(v, G)
2:      if an e for v does not exist in G then
3:          e = G.createEdge(v.sourceIP, v.targetIP)
4:          for model Ω in G do
5:              e.updatePosterior(Ω)
6:          classify(e, G)
7:      else
8:          e = G.getEdge(v.sourceIP, v.targetIP)
9:          e.add(v)
10:         G.updatePriors()
11:         for edge e in G do
12:             for model Ω in G do
13:                 e.updatePosterior(Ω)
```

An edge $e$ (an aggregate of evidences) may be classified into an existing model or used to generate a new model. ASSERT adopts the use of Wemmert-Gancarski Index (WGI) to decide whether to create a new model. The inverse of the maximum posterior probability is considered as the distance between each aggregate and the model centroid, and used to calculate WGI and determine the separation of the attack models. The WGI associated with classifying a new edge to an existing model is compared to that associated with using the new edge to generate a new model, and the decision goes to the one with the higher WGI. The process of deciding whether to create a new model or not is summarized in Algorithm 2.

**Algorithm 2** Classify an edge $e$ to a new model $\Omega_n$ or an existing model $\Omega_e$ with maximum posterior for $e$ in an attack graph $G$ where $\Omega[\,]$ represents the list of models.

```
1:  function CLASSIFY(e, G)
2:      if |Ω[ ]| = 0 then
3:          Ω_0 = G.createModel(e)
4:          e.updatePosterior(Ω_0)
5:      else
6:          Ω_n = G.createModel(e)
7:          WGI_{Ω_n} = G.calculateWGI()
8:          G.removeModel(Ω_n)
9:          Find Ω_e for e
10:         e.setModel(Ω_e)
11:         G.updatePriors()
12:         for edge e in G do
13:             for model Ω in G do
14:                 e.updatePosterior(Ω)
15:         WGI_{Ω_e} = G.calculateWGI()
16:         if WGI_{Ω_n} > WGI_{Ω_e} then
17:             Ω_e.remove(e)
18:             G.createModel(e)
```

#### Dynamic Bayesian Classification

One of the key novelties of ASSERT is the calculation of $P(X|\Omega)_f$ that is the likelihood of a model $\Omega$ generating an evidence aggregate $X$ for a given feature $f$. Note that explicitly calculating $P(X|\Omega)_f$ by assessing a large number of combinatorial possible occurrences could lead to an extremely small likelihood, be computationally challenging and cause precision errors in standard computers. However, the use of a non-parametric histogram is needed, because there is no evidence in the community suggesting the target IP, port or exploit selections follow a certain distribution.

Given two probability distributions $\Omega$ and $X$, the Kullback-Leibler divergence (KLD) (Kullback and Leibler 1951) is a non-symmetric measure of the difference between $\Omega$ and $X$. It represents the amount of information lost when a model $\Omega$ is used to approximate an edge $X$. Therefore, instead of explicitly calculating $P(X|\Omega)_f$, ASSERT uses KLD to estimate how well a model explains the observed evidences on an edge. Using $p_i$ as the probability of a feature instance $i$ in $\Omega$ and $q_i$ as its probability in $X$, the KLD from $\Omega$ to $X$ for a feature $f$ is defined as

$$KLD_f\,(X\|\Omega) = \sum_i q_i \log \frac{q_i}{p_i} \tag{1}$$

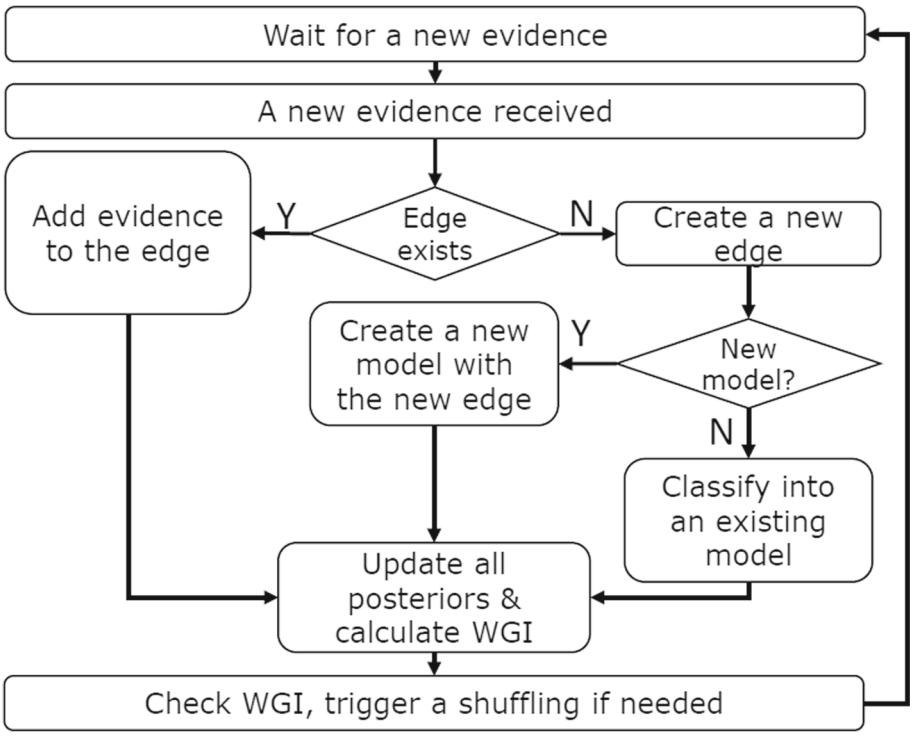


**Fig. 2** The execution flow in the ASSERT system

ASSERT uses the inverse of *KLD* to calculate the likelihood for each edge distribution $X$. Then, the likelihoods for all features are multiplied to find the overall likelihood between a model $\Omega$ and edge $X$:

$$P(X|\Omega) = \prod_f \frac{1}{KLD_f\left(X\|\Omega\right)) + 1} \tag{2}$$

In addition, we recognize that it is not uncommon to have certain feature instances not show up in the model during the dynamic classification process. This can lead to zero probability when using KLD (which causes problem if any $p_i = 0$ in (1)). In the context of cyberattack modeling, not yet observing a feature instance (e.g. a specific exploit being used or a port being targeted) does not necessarily mean that such feature will not occur in the future. In fact, an attacker is likely to target different exploits, ports, and IPs over time. Therefore, ASSERT uses the Shannon's entropy to estimate the stochastic nature of a feature histogram and distribute the entropy to the unseen feature values in the model ($\Omega$), so as to reflect the possibility of having an unobserved feature instance.

More specifically, if a certain feature value $i$ is observed on an edge but is not seen in a model ($p_i = 0$), the Shannon entropy of the feature model is used to calculate the probability of unseen feature observations in the model. The entropy of a feature $f$ within a model $\Omega$ i.e. $E_f$ is calculated by

$$E_f = -\sum_i^n p_i \log\left(p_i\right) \tag{3}$$

where $n$ is the total number of observations within the feature histogram. ASSERT evenly distributes the entropy to all feature values, by defining $p_i'$ as

$$p_i' = p_i\left(1 - E_f\right) + \frac{E_f}{|f|} \tag{4}$$

where $0 \leq E_f \leq 1$ and $|f|$ is the total number of feature values present in the model.

With this entropy re-distribution, ASSERT updates (1) by using $p_i'$ instead of $p_i$. This gives the final likelihood for ASSERT, which is used in conjunction with a uniform prior and, in turn, determines the posterior between each pair of model and edge. The maximum posterior is then found and used to calculate WGI to determine whether to associate the edge to the max-posterior model or to use the edge to create a new model.

#### Dynamic model generation

ASSERT assigns a uniform prior probability to each model and uses it and the edge likelihood to calculate the posterior probability for an edge. Note that there is no a priori model for ASSERT, and thus empirical models need to be generated. In other words, the max-posterior classification must be accomplished by a method allowing the

creation of a new model. The choice between associating with an existing empirical model versus creating a new model is accomplished by treating the models as clusters, and the max-posterior as the inverse of the distance between an edge and its model.

If the posterior probability of an edge $X_i$ for a model $\Omega$ is represented by $P(\Omega|X_i)$, the inverse of this posterior is defined as

$$d_i = \frac{1}{P(\Omega|X_i)} \tag{5}$$

and is used as a distance metric to determine WGI cluster validity index. WGI is then used to assess the quality of the model separation, decide when a new model needs to be created, and trigger a new edge shuffling process. WGI defines a coherence and separation metric $R$ for each edge $X_i$ as

$$R(X_i) = \frac{d_i}{\min\left(d_i'\right)} \tag{6}$$

where $d_i$ represents the distance between an edge $X_i$ and its max posterior model centroid and $d_i'$ shows the distance between $X_i$ and its second best model centroid. WGI takes the mean of $R(X_i)$ for each model and defines a coherence and separation metric $J_k$ for each model as

$$J_k = \max\left\{0, 1 - \frac{1}{n_k}\sum_{i}^{n_k} R(X_i)\right\} \tag{7}$$

$J_k$ takes the complement of the mean of $R(X_i)$ for model $k$ to 1 (or ignore it when it is $> 1$) where $n_k$ shows the number of edges in model $k$. Based on Eqs. 6 and 7, WGI is defined as

$$WGI = \frac{1}{N}\sum_{k=1}^{N} n_k J_k \tag{8}$$

where $N$ represents the total number of models in the system. ASSERT uses $n_k = 1$ in (8) in order to treat very large and relatively smaller models in the same way. This is needed particularly in the context of cyberattack modeling, because often times significant amount of the received alerts are scanning, and attack model generation should not be biased against non-scanning alerts.

#### Model shuffling via DBSCAN

ASSERT monitors the quality of the classification continuously by referencing the WGI. When WGI falls below a configurable threshold, a shuffling process is triggered to ensure that each edge is assigned to a model with the highest posterior and redundant models are removed. The shuffling process might be triggered several times throughout the execution flow based on the two trigger conditions:

1. **Index Threshold ($I_T$)**: If the WGI value falls below a configured threshold.
2. **Number of Evidences ($E_T$)**: If the number of evidences received after the last shuffle has reached a certain configured threshold.

To trigger a shuffling process, both $I_T$ and $E_T$ conditions must be satisfied. That means, ASSERT waits for a certain number of evidences to trigger the shuffling, even if the index falls below the configured boundary to avoid a shuffling overhead.

ASSERT uses a density-based clustering algorithm i.e. DBSCAN that groups closely packed edges together. ASSERT uses DBSCAN, because unlike traditional clustering algorithms such as $k$-Means, it does not need to know the number of clusters, and can identify outliers if any. DBSCAN uses two parameters i.e. the epsilon ($\epsilon$) and the minimum number of points ($mp$) to define a dense region. The algorithm iterates through the edges that have not been visited and retrieves their $\epsilon$-neighborhood. If the neighborhood contains enough number of edges a cluster is created, otherwise the edge is labeled as noise (outlier).

Note that DBSCAN requires pair-wise distances between edges. The pair-wise Jensen-Shannon divergence (JSD) is used as an estimate of the distance between edges, since there may not be sufficient evidences on the edges to determine the divergence. ASSERT uses the JSD, a symmetrized and smoothed version of the KLD, to calculate a distance matrix for the behavior aggregates (edges). Given two edges i.e. $X_1$ and $X_2$, and a feature $f$, the JS divergence between $X_1$ and $X_2$ is calculated by

$$JSD_f\left(X_1 \| X_2\right) = \pi_1 KLD_f\left(X_1 \| M\right) + \pi_2 KLD_f\left(M \| X_2\right) \tag{9}$$

where $\pi_1$ and $\pi_2$ weights are calculated considering the number of evidences in $X_1$ and $X_2$ and $M = \pi_1 X_1 + \pi_2 X_2$. Then the distance between two edges is found by

$$JSD(X_1, X_2) = \frac{1}{|f|}\sum_{f} JSD_f\left(X_1 \| X_2\right) \tag{10}$$

where $|f|$ denotes the number of features.

ASSERT follows a systematic approach to set the parameters of the DBSCAN algorithm. First, it uses the natural logarithm of the total number of edges to set the value of the $mp$ parameter. Second, it computes the average distance of each edge to its $k$ nearest neighbors, where $k = mp$. Then, these average distances are plotted in an ascending order and the elbow (knee) point on the plot is used to set the value of $\epsilon$.

Once DBSCAN is triggered by the aforementioned conditions, the edges are re-clustered to a set of attack models. At the end of the re-clustering (shuffling) process, DBSCAN may create noisy clusters and mark the edges

within these clusters as outliers. These outliers are not considered by DBSCAN as valid clusters, and thus not associated with any attack model. Note that an edge considered an outlier by DBSCAN could contain the first observables of a unique and emerging attack behavior. Therefore, each of these edges should be given a chance to create a new model, or associated with the closest valid model. ASSERT reclassifies each of these outliers by either associating them to an existing regular cluster or creating a new model depending on the value of the WGI, which is the same process described in Algorithm 2.

## Design of experiments

This paper demonstrates ASSERT's capability using the intrusion alerts collected from the 2017 National Collegiate Penetration Testing Competition (CPTC) (CPTC Organizing Committee 2017), where approximately 60 people from 10 teams attempting to penetrate into the same computing infrastructure to find as many vulnerabilities as possible. Suricata was installed to capture malicious activities over approximately a 9-h period, and the Suricata alerts were used as inputs for the experiments shown in this paper (Suricata 2019). ASSERT processes the Suricata alerts one by one to generate and update empirical models without any a priori knowledge on either the attackers or the infrastructure. The source and target IPs are used to define the edges. In this set of experiments, alert category, alert signature, target IP, and target port are used as features to generate and refine dynamic attack models.

### WGI and Jensen-Shannon divergence

ASSERT assesses the empirically generated models regularly using a cluster validity index. Clustering indexes are similar in nature, as they provide a measure for the intra and inter cluster quality. However, there might be differences about what they need to calculate the index value. A key point for ASSERT is that there is no natural meaning of a behavior aggregate (edge) in a feature space. Because, an edge as characterized by multiple probability distribution functions, it may not have an easy interpretation of its distance to other edges. One key novelty of ASSERT is to interpret the model as the centroid and the posterior as the inverse of distance to the centroid during the Bayesian learning phase. Therefore, ASSERT uses the Wemmert-Gancarski Index (WGI) which treats the inverse of the calculated posterior as the distance to the model centroid, so as to reflect the intra-cluster compactness with the maximum a posterior and the inter-cluster separation with the second best posterior.

To provide an independent assessment of the quality of models generated, this work considers an alternative metric - the general Jensen-Shannon divergence (JSD). The general JSD is an extension of the pair-wise JSD discussed in "Dynamic model generation" section, and aims at measuring the overall 'divergence' or coherence of multiple models. More specifically, the JSD of multiple models $(\Omega_1, \Omega_2, ..\Omega_n)$ based on a feature $f$ is defined by the entropy of the combined model $(E_f^T)$ and the weighted average of individual model entropies $(E_f^I)$.

$$JSD_f(\Omega_1, \Omega_2, ..\Omega_n) = E_f^T - E_f^I \tag{11}$$

$$E_f^T = E_f(\sum_{i=1}^{N} \pi_i \Omega_i) \tag{12}$$

$$E_f^I = \sum_{i=1}^{N} \pi_i E_f(\Omega_i) \tag{13}$$

where $\pi_i$ represents the weight of a model $\Omega_i$ and $N$ is the number of models. We use a uniform weight value of $\pi_i = \frac{1}{N}$ for each model while calculating $E_f^T$ and $E_f^I$. It is desirable to have lower individual entropies $(E_f^I)$ for better coherence within each model, and higher entropy for the combined model $(E_f^T)$ for larger separation among the models. $JSD_f$ generates a value between 0 and $\log_2 N$. For easy comparison over time where $N$ varies, this work normalizes the general JSD of each feature by $\log_2 N$ and then finds the average JSD over the multiple features by

$$JSD(\Omega_1, \Omega_2, ..\Omega_n) = \frac{1}{|f|} \sum_f JSD_f. \tag{14}$$

### Predictability indices

ASSERT aims at generating and updating attack models, each reflecting a collective and unique behavior that could be used to enhance cyberattack prediction. If ASSERT is not used, the overall statistics from the cumulative intrusion alerts i.e. all evidences from all edges in the entire graph $G$ could potentially be used by security professionals to derive what might happen next. We will consider the statistical model from the entire graph $G$ as the baseline. ASSERT focuses on how to best separate evidences of malicious activities in an online and robust manner. While there could be advanced cyberattack prediction algorithms (Fava et al. 2008; Yang et al. 2014), this work considers basic use of statistics to estimate the potential to predict which future actions might be observed. When a new evidence is received, ASSERT calculates the probability of each feature value in the evidence, based on the model $\Omega$ (that its edge is associated) versus that based on all the evidences in the graph $G$. This work considers this probability as an estimate of the potential 'predictability' that can be achieved for a given model or the overall statistics.

Let $p_i^n(\Omega)$ be the probability of a feature value $i$ given a model $\Omega$ just before the $n^{\text{th}}$ evidence arrives. Likewise, define $p_i^n(G)$ for the baseline case that considers the entire

graph $G$. For the ease of presentation, we shall refer these probability values as the 'predictability' for the remainder of this paper. The average predictability over a moving window is calculated and plotted to assess whether the generated model finds a better/higher probability of the imminent feature value compared to the baseline. More formally, the average predictabilities are calculated as

$$\tilde{p}^n(\Omega) = \frac{1}{|w|} \sum_{k=n-|w|+1}^{n} p_i^k(\Omega) \tag{15}$$

$$\tilde{p}^n(G) = \frac{1}{|w|} \sum_{k=n-|w|+1}^{n} p_i^k(G) \tag{16}$$

where $|w|$ represents the size of the moving window and $n$ shows the alert number.

Traditional cyber defense relies much on past observations or statistics. The predictability metric presented above shows the potential value of ASSERT for all imminent feature values. What if a feature value never occurs on a given edge? This means predicting an 'unseen' feature for a given source-target pair. If the generated model is effective, it should help to predict these unseen features better than using the overall graph. Note that the overall graph has a larger sample size and potentially more likely to see a feature that are previously unseen on an edge. A model needs to be unique and useful for the edges associated with the model so as to have a higher predictability for unseen features. This paper also compares the smoothed 'unseen predictabilities' ($\tilde{q}^n(\Omega)$ and $\tilde{q}^n(G)$) to assess whether ASSERT generates quality models that can help to predict the unseen features better.

## Results

Before presenting the specific results, we summarize the parameters used for the set of experiments shown in this paper. ASSERT uses DBSCAN during shuffling with an $\epsilon = 1.9$ which reflects the elbow (or knee) point in the $k$-distance chart as explained in "Model shuffling via DBSCAN" section. A shuffling process is triggered each time when $I_T = 0.70$ and $E_T = 1000$. Moreover, a moving window size of 20 is used while calculating all predictabilities i.e. $\tilde{p}^n(\Omega)$, $\tilde{p}_i^n(G)$, $\tilde{q}^n(\Omega)$ and $\tilde{q}^n(G)$. The $I_T$ and $E_T$ parameters reflect user preference to tolerate overall model quality (in WGI) versus processing time. They are used by ASSERT to decide when to trigger a shuffling process, which takes a longer time than a single Bayesian classification step but can re-assess the cumulative evidences at once. Selecting a higher $I_T$ or a lower $E_T$ will potentially lead to more shuffling processes and increase the total processing time for ASSERT but may give better WGI while evidences arrive, and vice versa. $I_T = 0.7$ is chosen to ensure a relatively high quality of clustering throughout the execution process. A lower $I_T$ could also be chosen depending on the tolerance that the user has for the quality of the model separation or cohesion. Similarly, a higher $E_T$ could be chosen, but that could mean a lower quality in WGI for a longer period of time. Depending on the tolerance of the user for clustering quality, a higher $E_T$ value might be selected. $E_T = 1000$ is chosen as a threshold value for the CPTC'17 data set, to avoid frequent shufflings and wait for 1000 intrusion alerts before triggering a shuffling process when WGI is $< 0.7$. In our experiment, we observe that except the very first shuffling process, $E_T$ is not the dominating factor to trigger the shuffling processes. The iterative Bayesian classification performs reasonably well and thus $I_T$ goes below 0.7 after more than 1000 alerts are observed since last shuffling.

Processing 32,265 intrusion alerts in the CPTC17 data, ASSERT generates 2,392 edges and 47 attack models. Table 1 shows details of each shuffling process throughout the execution. ASSERT triggers a total of 8 re-clustering processes where *NA* and *NE* show the cumulative number of alerts and edges prior to each shuffling process. *NAB* represents the number of alerts processed after the last shuffling and *NM* shows the number of models created at the end of each shuffling. *NOE* shows the number of edges in the outlier cluster generated by DBSCAN, and *NOM* represents the number of new models created from these edges. For example, 28628 alerts have been processed (5636 of which received after the $7^{th}$ shuffling) and 2341 edges have been created prior to the $8^{th}$ shuffling process. At the end of the $8^{th}$ shuffling, a total of 16 models and 39 edge outliers were generated. 28 of these 39 outliers were associated to the 5 models generated by DBSCAN and the remaining 11 were used to create new models.

The CPTC17 data set includes a total of 20, 166, 68, and 495 distinct values for the alert category, alert signature, target IP, and target port features, respectively. As the alerts come into the system, each of these features may see new values over time. Since an important challenge for ASSERT is to associate edges with models even if some feature values exist in the model are not present on the edge, it is important to review the 'entropy' of the

**Table 1** Re-clustering processes for the CPTC17 data set

| | NA | NE | NAB | NM | NOE | NOM |
|---|---|---|---|---|---|---|
| 1 | 1010 | 161 | 1010 | 14 | 17 | 11 |
| 2 | 3300 | 273 | 2290 | 7 | 14 | 4 |
| 3 | 5257 | 509 | 1957 | 8 | 10 | 6 |
| 4 | 8676 | 798 | 3419 | 12 | 22 | 9 |
| 5 | 14726 | 1214 | 6050 | 15 | 30 | 11 |
| 6 | 18871 | 1772 | 4145 | 20 | 38 | 15 |
| 7 | 22992 | 2136 | 4121 | 24 | 39 | 20 |
| 8 | 28628 | 2341 | 5636 | 16 | 39 | 11 |

features as the alerts come into the system. Figure 3 shows the average entropy of four features as more and more alerts come into the system. All four feature entropies follow similar trends, and thus not shown here. We observe that the average feature entropy fluctuates and has a general upward trend until it saturates after approximately 10,000 alerts are received. This observation coincides with the results in Table 1, where more frequent shufflings are needed prior to the 10,000 alert mark.

#### The cluster validity index

As more edges are classified into a model, the collective behavior of the model changes over time and it might not reflect the behavior of the edges classified into the model initially. Therefore, the overall separation and coherence of the generated models need to be monitored continuously. ASSERT assesses the quality of the empirically generated models regularly, using the WGI cluster validity index. The novel use of the inverse of the maximum posterior as a distance measure to the model centroids allows representing the intra-cluster compactness and the calculation of WGI in the absence of actual feature space (due to the complexity of non-parametric feature histograms). A decrease in WGI is possible, due to the divergence between the feature histograms of the edges and the model they are classified when additional evidences are observed on the edges. Therefore, a shuffling process using DBSCAN is used to re-cluster edges with the attack models. Figure 4 shows the change in the WGI as alerts come into the system. The number of steep rises in WGI, reflects the re-clustering processes triggered. While the re-clustering improves WGI, it takes on average 6.86 s to complete as opposed to only 0.11 milliseconds for each Bayesian classification. The combination of dynamic Bayesian classification and shuffling enables a good balance between computational efficiency to process streaming alerts and robustness to maintain quality models in terms of cohesiveness within each model and separations among them.

As shown in Table 1, *NAB* represents the number of alerts processed between two successive shuffling processes. This along with Fig. 4, shows more frequent shufflings occur prior to the 10,000 alert mark and less so later on. In fact, after the $8^{th}$ shuffling, WGI did not come down and there was no need to shuffle. This trend somewhat coincides with the entropy shown in Fig. 3, where upward trend was observed until the 10,000 alert mark, and slight decline towards the end.

#### Jensen-Shannon divergence

Note that WGI shows the quality of the model in terms of the separation between the models and the cohesion within each model in the feature space. As an alternative and external measure to assess the quality of the generated attack models, we calculate the general JSD among models as defined in "WGI and Jensen-Shannon divergence" section. Figure 5 shows the general JSD ($JSD_f$ in (11) with the middle red line), the entropy of the combined model ($E_f^T$ in (12) with the top blue line), and the average individual model entropy ($E_f^I$ in (13) with the bottom green line). Note that the entropy of the combined model reflects the overall uncertainty (separation) across the individual models, and, thus, the higher the better. At the same time, the lower individual model entropy reflects better cohesion within each model. Overall, Fig. 5 shows that ASSERT performs well and have high $E_f^T$ and low $E_f^I$. When $E_f^T$ become low, the shuffling triggered by WGI re-associates the edges with models and recovers $E_f^T$ without increasing too much on $E_f^I$ (or have it go down quickly afterward with the dynamic Bayesian classifier).

#### Predictability

ASSERT associates an aggregated edge histogram to an attack model, expecting that the future behavior on the edge would be similar to its maximum posterior model. Therefore, if the classification of the attack behaviors

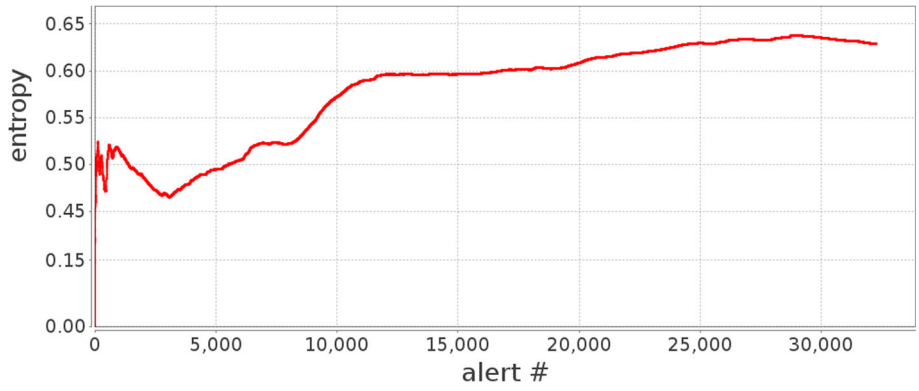


**Fig. 3** The overall entropy of *G* over time

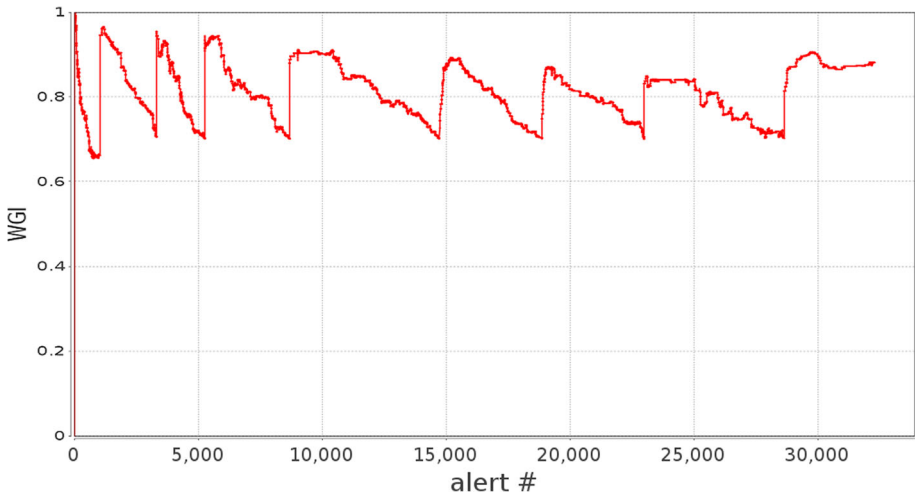


**Fig. 4** WGI over iterations with shuffling

is good, then the model statistics should be indicative of the feature values observed on an edge whether or not such features have occurred before on the edge. This section shows $\tilde{p}^n(\Omega)$ and $\tilde{p}_i^n(G)$ as defined in "Predictability indices" section and the next section discusses the unseen predictabilities.

Recall that this paper defines 'predictability' as the likelihood of a feature value as given by the attack model it is associated with, just before the feature value was observed. Note that advanced prediction algorithms e.g. (Fava et al. 2008) may take past transitions and other factors into account and likely achieve better prediction accuracy. The definition of predictability presented in this paper is meant to assess the value of the attack models generated by ASSERT when used to predict next actions based on zero-order statistics. ASSERT is compared to the baseline where the cumulative statistical model is used to determine the likelihood.

Figures 6, 7, 8 and 9 show $\tilde{p}^n(\Omega)$ and $\tilde{p}_i^n(G)$ as alerts come into the system for the target IP, target port, alert signature and alert category features, respectively. Overall, $\tilde{p}^n(\Omega)$ clearly outperforms $\tilde{p}_i^n(G)$ consistently over the entire dataset, showing the attack models created by ASSERT provide outstanding value for predicting future attack action features. To quantify this performance improvement, we define a predictability ratio $\Delta(f_n) =$

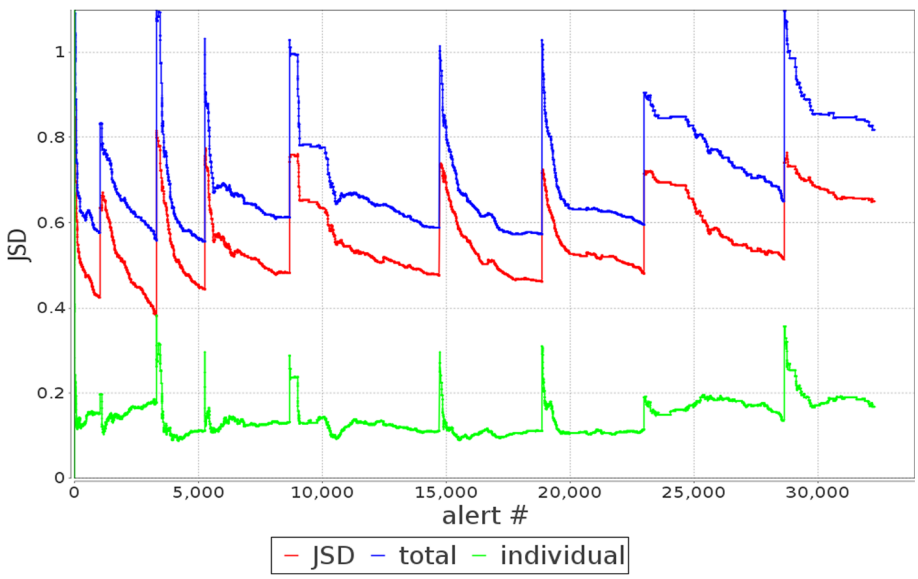


**Fig. 5** The $JSD_f$ (JSD), $E_f^T$ (total), and $E_f^I$ (individual) as the alerts come into the system

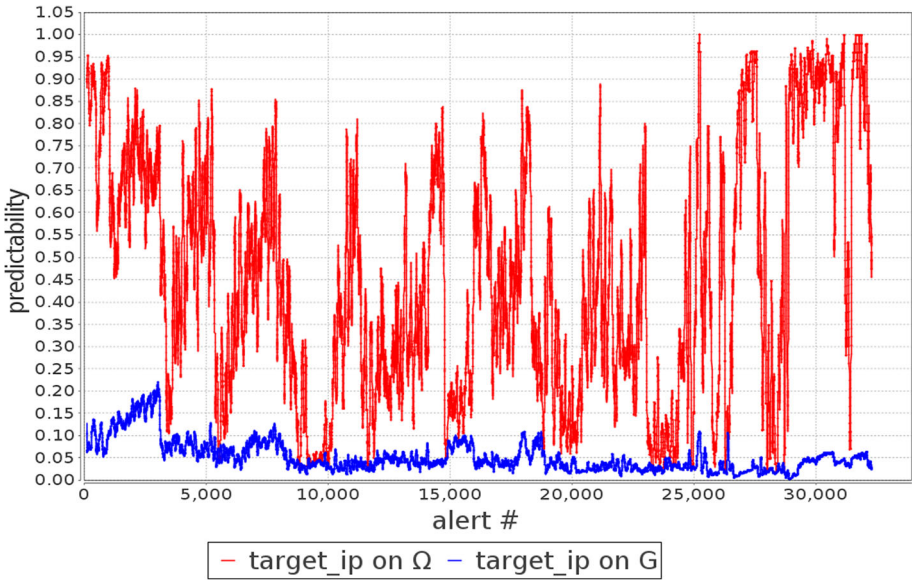


**Fig. 6** The $\tilde{p}^n(\Omega)$ and $\tilde{p}^n(G)$ for target IP as the alerts come into the system

$p_i^n(\Omega)/p_i^n(G)$ for each feature $f$. It is observed that the average $\Delta(f_n)$ for the target IP, target port, alert signature, and alert category features are 12.80, 3.64, 2.39, and 4.39, respectively.

To compare the predictability measures in more detail, we calculate the percentage of times the $p_i^n(\Omega)$ and $p_i^n(G)$ predictabilities are larger than a certain threshold $T$, ranging between 0.05 and 0.5. Table 2 shows these percentages for all four features. Note that these are based on the individual probabilities while the lines plotted in the Figs. 6, 7, 8 and 9 are based on the moving window average. Except the only case where the threshold is 0.05 for the target port feature, $p_i^n(\Omega)$ exceeds the threshold significantly more often than $p_i^n(G)$. This gap tends to increase as the threshold increases as well. This is interesting in that, it shows ASSERT is able to produce attack models that are capable of predicting specific feature values with much higher probabilities. For instance, the attack models give a probability value of 0.35 or above for target IP 40.34% of the time while the cumulative statistics is at

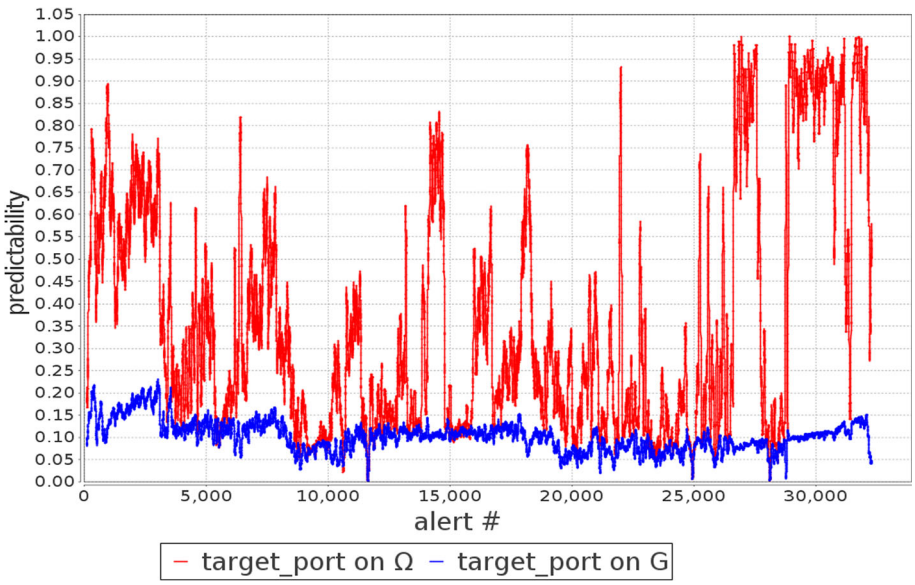


**Fig. 7** The $\tilde{p}^n(\Omega)$ and $\tilde{p}^n(G)$ for target port as the alerts come into the system

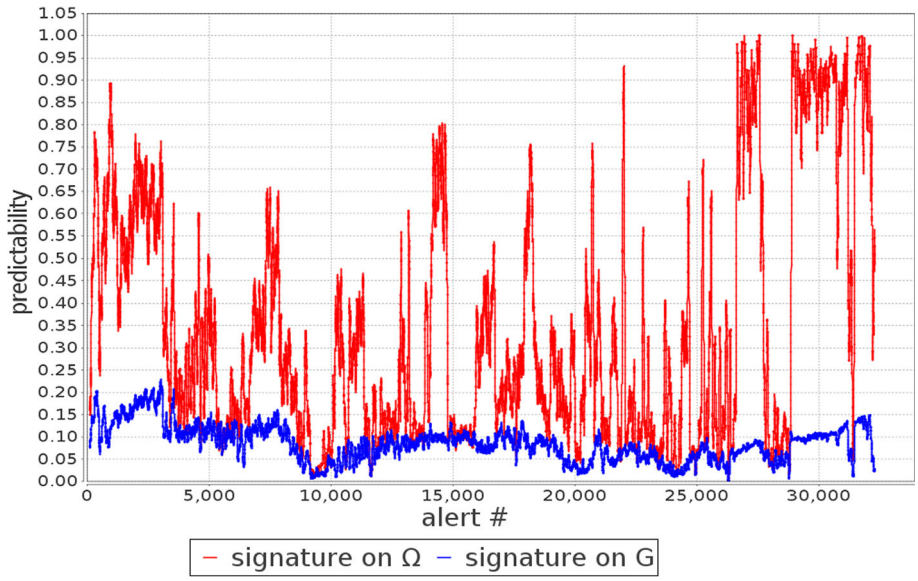


**Fig. 8** The $\tilde{p}^n(\Omega)$ and $\tilde{p}^n(G)$ for alert signature as the alerts come into the system

0.01%. Similar performance gaps are observed for target port and alert signature at the same threshold level. There are much fewer number of unique alert categories, so it is relatively easier to predict, but similar performance gap is also observed at the threshold level of 0.5.

#### Unseen predictability

Similar to the analysis for the overall predictability, the probabilities $\tilde{q}^n(\Omega)$ and $\tilde{q}^n(G)$, are calculated for each feature instance that has never been seen on an edge. It should be noted that these probabilities are calculated only if a feature instance $i$ has not been seen on its edge, but was seen in the associated model. Understandably, there are much fewer (but still significant) instances of unseens than the cases tracked in the previous section. For this section, we focus on the target port and alert signature features because these two features have a significant number of distinct values (495 and 166, respectively) than the target IP (68) and alert category (20). Having more distinct values means that there are more cases of unseens

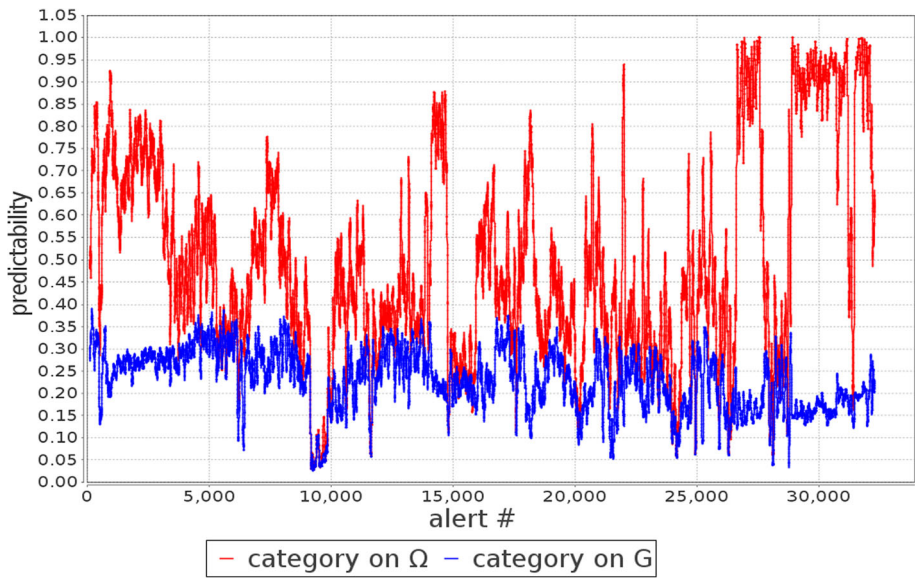


**Fig. 9** The $\tilde{p}^n(\Omega)$ and $\tilde{p}^n(G)$ for alert category as the alerts come into the system

**Table 2** The percentage of times $p_i^n(\Omega) > T$ and $p_i^n(G) > T$ for different features

| | Target IP | | Target port | | Signature | | Category | |
|---|---|---|---|---|---|---|---|---|
| $T$ | $p_i^n(\Omega)$ | $p_i^n(G)$ | $p_i^n(\Omega)$ | $p_i^n(G)$ | $p_i^n(\Omega)$ | $p_i^n(G)$ | $p_i^n(\Omega)$ | $p_i^n(G)$ |
| 0.05 | 60.25 | 29.38 | 71.71 | 72.13 | 63.15 | 58.03 | 81.08 | 80.31 |
| 0.10 | 55.62 | 21.66 | 59.13 | 42.46 | 48.23 | 28.51 | 76.08 | 72.79 |
| 0.15 | 48.71 | 9.03 | 44.51 | 19.68 | 42.14 | 19.03 | 71.68 | 59.31 |
| 0.20 | 44.18 | 2.17 | 41.61 | 10.08 | 39.85 | 9.98 | 63.56 | 43.02 |
| 0.25 | 41.35 | 1.33 | 32.38 | 1.83 | 30.72 | 1.79 | 57.77 | 34.32 |
| 0.30 | 41.03 | 0.14 | 30.33 | 0.21 | 28.81 | 0.21 | 57.3 | 33.15 |
| 0.35 | 40.34 | 0.01 | 29.65 | 0 | 28.29 | 0 | 56.44 | 30.71 |
| 0.40 | 39.81 | 0.01 | 29.12 | 0 | 27.9 | 0 | 53.32 | 20.75 |
| 0.45 | 39.47 | 0 | 28.74 | 0 | 27.66 | 0 | 41.66 | 1.95 |
| 0.50 | 38.34 | 0 | 27.35 | 0 | 26.91 | 0 | 35.16 | 0 |

and it is harder to predict these values. Moreover, there is no 'unseen' instances of target IP based on our definition, since all target IPs seen in a model must already exist on an edge in the model. Note that predicting a potentially new target IP is an interesting research question, but is beyond the scope of this paper.

In order to have a set of previously observed evidences to check against, sufficient number of evidences must be processed by the system beforehand. Therefore, this paper reports the predictabilities for the unseen feature instances after the first 200 alerts come into the system. Figures 10 and 11 show $\tilde{q}^n(\Omega)$ and $\tilde{q}^n(G)$ for the target port and alert signature features, respectively, as the alerts come into the system. Except very few cases, most of the time $\tilde{q}^n(\Omega)$ is higher than $\tilde{q}^n(G)$. Note that the range (y-axis) of unseen predictabilities is at around or below 0.1, much smaller than the cases presented in Figs. 6, 7, 8 and 9. This is expected, but also means that a more complex and careful design of prediction algorithms will be needed to leverage these lower zero-order probabilities.

We again use a set of probability thresholds ($T$) between 0.05 and 0.50 and count the percentage of times the unseen predictability measures $q_i^n(\Omega)$ and $q_i^n(G)$ are above these thresholds. Table 3 shows these percentages for the target port and alert signature features. As expected, ASSERT performs better than using the cumulative statistics to predict the unseens. In fact, using the cumulative statistics cannot produce any probabilities above 0.20 for unseen target port or alert signature features, while it is still feasible using ASSERT even at 0.50 level. Recall

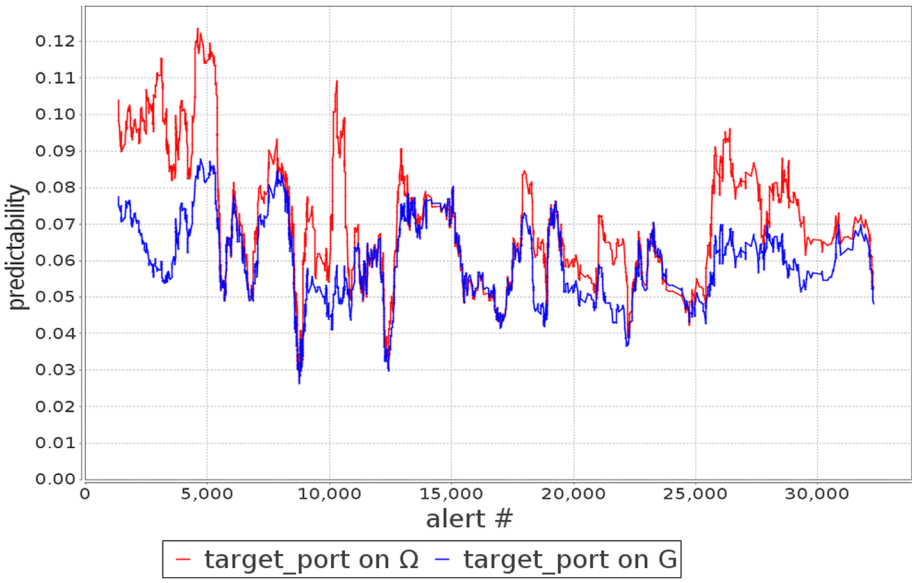


**Fig. 10** $\tilde{q}^n(\Omega)$ and $\tilde{q}^n(G)$ over iterations for the target port

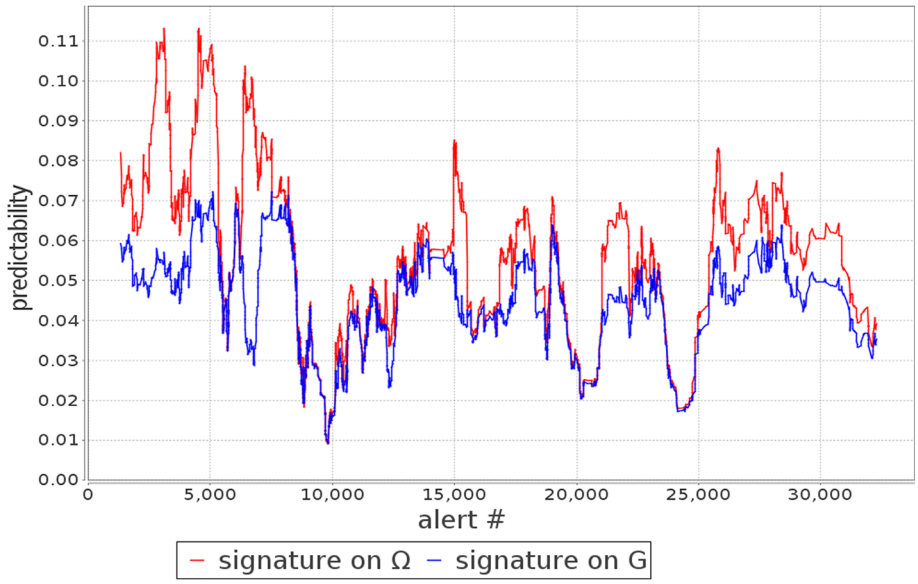


**Fig. 11** $\tilde{q}^n(\Omega)$ and $\tilde{q}^n(G)$ over iterations for the signature

again that the results shown in Table 3 are based on the individual probabilities while the plotted lines shown in Figs. 10 and 11 are based on the moving average, hence appears lower. Certainly, the benefits seem to be smaller than the overall cases presented in Table 2. In fact, the average of the predictability ratio $\Delta(f_n) = q_i^n(\Omega)/q_i^n(G)$ for the unseen target port and signature features are 1.17 and 1.22, respectively, which are much smaller than those reported for the overall predictability cases. This smaller performance gain is expected, since predicting unseens is fundamentally a daunting task. The fact that the attack models created by ASSERT can continuously produce sufficiently large probabilities for the unseens throughout the approximately 9-h CPTC event is promising. A careful design of a prediction algorithm that leverages these zero-order probabilities could potentially lead to an unprecedented capability where unseen attack actions can be actually predicted.

### Attack models: a closer look

The core value of ASSERT lies in the creation of attack models that could assist the analysts to focus on critical activities and use such more targeted analysis to predict likely attack actions. In addition to the overall predictability analysis discussed in the previous sections, this section shows two attack models extracted from the same data set and describes how an analyst might be able to use them for insightful and targeted analysis. Specifically, we will show that each of these two attack models contains related attack actions from different teams and suggests attack actions that could possibly be executed by some teams, because another team or two in the same model has done them.

Figure 12 shows the feature histograms of two attack models, Model A (left), which contains a total of 223 intrusion alerts and a smaller Model B (right), which contains 65 alerts. As a reference, recall there are a total of 32,265 alerts in our data set. For each model, the feature histograms for the alert category, alert signature, target IP, and target port are provided. As stated earlier, the CPTC 2017 data set is composed of intrusion alerts from ten different teams which are trying to penetrate into the same computing infrastructure. Therefore, each attack model includes intrusion alerts from various teams, and in each histogram alerts from different teams are represented with different colors.

Attack Model A aggregates potentially related intrusion alerts from nine different teams and indicates a

**Table 3** The percentage of times $q_i^n(\Omega) > T$ and $q_i^n(G) > T$ for the target port and alert signature features

| | Target port | | Signature | |
|---|---|---|---|---|
| $T$ | $q_i^n(\Omega)$ | $q_i^n(G)$ | $q_i^n(\Omega)$ | $q_i^n(G)$ |
| 0.05 | 26.51 | 25.17 | 22.05 | 18.88 |
| 0.10 | 12.97 | 9.8 | 6.08 | 3.22 |
| 0.15 | 4.88 | 3.44 | 3.61 | 2.77 |
| 0.20 | 3.46 | 1.49 | 2.59 | 1.23 |
| 0.25 | 1.29 | 0 | 0.86 | 0 |
| 0.30 | 0.7 | 0 | 0.41 | 0 |
| 0.35 | 0.53 | 0 | 0.3 | 0 |
| 0.40 | 0.38 | 0 | 0.14 | 0 |
| 0.45 | 0.34 | 0 | 0.12 | 0 |
| 0.50 | 0.13 | 0 | 0.05 | 0 |


**Fig. 12** Alert category, alert signature, target IP and target port feature histograms for Models A (left) and B (right) generated by ASSERT. The *x* and *y* axis of each histogram show the individual feature values or categories and the number (volume) of alerts, respectively. Different colors in each bar represent alerts coming from the different teams

serious network trojan injection. Its alert category histogram includes miscellaneous activity, unsuccessful user privilege gain, trojan detection, and potentially bad traffic categories. Its alert signature histogram indicates Remote Desktop Protocol (RDP) connection confirmations, potential FTP brute force attempts, repeated logon failures and executable script downloads. Similarly, its target port histogram includes alerts with categorized destination ports that are used by the Remote Desktop Management interfaces and its target IP histogram shows the victim IP numbers for the aforementioned exploits. The intrusion alerts of Model B are from five different teams, representing a collective attack behavior where the alert category histogram includes alerts for information leak attempts and potential bad traffic. The alert signature histogram shows executions of a critical command (curl) and the gain of critical 'root' privileges. The target port histogram shows related categorized ports through which the exploit was carried out and the target IP histogram shows the victim hosts.

We observe that each attack model aggregates potentially related exploits carried out by different attack teams. In Model A, all nine teams have RDP connection confirmation alerts for a variety of target hosts. Team 6 has RDP connection confirmation alerts for six different target hosts, one of which is followed by a set of repeated logon failures. Similarly, Team 10 carries out a set of potential FTP brute-force attempts on a specific target, after a set of RDP connection confirmation alerts. Therefore, one might anticipate to see logon or FTP brute-force attempts executed by the remaining seven teams towards the target hosts for which an RDP connection confirmation alert is already observed. In Model B, five teams have used the command 'curl' to execute a data transfer. For one of these teams, intrusion alerts are observed showing that critical root privileges were gained. Therefore, one may anticipate a similar behavior from the remaining four teams towards the target hosts for which a 'curl' command is already executed.

Each attack model represents a collective attack behavior and includes a set of potentially related malicious activities whose relationships are not trivial to be discovered via simple statistical query or clustering approaches. When multiple teams in a model have a common behavior and one of the teams exhibits an additional behavior later on, one could anticipate the additional behavior from the remaining teams. Recognizing that these are from student penetration testing competition, the examples here

merely show a potential use of ASSERT where the analysts can focus on a small number of critical alerts from various attack sources (different teams) and make prediction without being overwhelmed by the tens of thousands of alerts. The emerging threats exhibited through the empirical models do not require a priori knowledge and are adaptive as more evidences are collected.

## Conclusion

In the absence of a priori knowledge on specific attacks and network configuration, ASSERT processes aggregated non-parametric feature histograms from streaming intrusion alerts to generate attack models using a dynamic Bayesian approach with a novel likelihood calculation, regularly assessed with WGI, and complemented with DBSCAN. The key novelties of the ASSERT system are:

- The use of posterior as the inverse of distance to cluster centroid enables the integrated use of Bayesian classifier and WGI in a dynamic manner, as a foundation for the semi-supervised online learning framework that synthesizes cyberattack models.
- The use of KLD with entropy redistribution over non-parametric feature histograms enables the association of observable aggregates with empirical models even if there are emerging features that are previously unseen.
- The use of pair-wise JSD between observable aggregates allows re-clustering with DBSCAN, which enables improvement and recovery from imperfect decisions made earlier by the dynamic Bayesian classifier with insufficient evidences.

Using the intrusion alerts from the 2017 CPTC data set, this paper demonstrates the capability of ASSERT in generating and updating distinct attack models over time without a priori knowledge. Using WGI and general multi-model JSD, we show that both the cohesiveness within individual models and the separation between models remain sound as new alerts are received. These models are also shown to be promising in predicting future attack actions. In particular, we assess the 'predictability' against the baseline using total cumulative statistics, a reasonable practice in reading intrusion alert data. ASSERT is able to produce probabilities that are 12.80, 3.64, 2.39, and 4.39 times higher than the baseline on average for the target IP, target port, alert signature, and alert category respectively. Furthermore, it is able to predict unseen target port and alert signature with probability values over 0.25 or even 0.5 while the baseline cannot produce any over 0.20. These results demonstrate the value of the attack models synthesized by ASSERT in predicting the 'where', 'how', and 'what' of future attack actions. Certainly, a careful design of how to use these models and feature predictions for an actual cyberattack prediction is needed and underway. This work demonstrates a clear and consistent advantage of using ASSERT to develop such a predictor.

The overall ASSERT framework and the experiment methodology presented in this paper are also generalizable for using additional features and assessing empirical models. In addition to the novelties of ASSERT, the introduction of the general multi-model JSD and the comprehensive evaluation using 'predictability' and 'unseen predictability' is important for the community as we move towards predictive cyber defense and in need of a way to assess empirical attack models without a priori knowledge.

ASSERT separates intrusion alerts into empirical attack models where the analysts may focus on critical activities and use the aggregated statistics from selected models to potentially predict future attack actions. While the assumption of edge aggregates within a reasonable duration is sound, there is a risk of adversaries intentionally embedding one or two critical exploits into very large number of common scanning activities in some edges. In such cases, ASSERT could still reveal the critical exploits as part of the attack model, but they will be statistically insignificant to derive a high predictability or the improvement in predictability would be limited. Another limitation of the current implementation of ASSERT is the inclusion of historical alerts, where the system needs to define how and when the alerts will be treated as part of historical instead of emerging attack behaviors. Finally, additional experiments with more engineered features as well as human subject study will be beneficial to maximize the value provided by the ASSERT framework.

**Funding**

This research is supported by NSF Award #1526383.

**Availability of data and materials**

The data used during this research will not be shared, because it is protected by a confidentiality agreement.

**Authors' contributions**

AO: The lead researcher and author. SJY: Developed research ideas and helped in writing. All authors read and approved the final manuscript.

**Competing interests**

All authors declare that they have no competing interests.

## Publisher's Note



Received: 16 January 2019 Accepted: 15 April 2019
Published online: 14 May 2019

**References**

Al-Mohannadi H, Mirza Q, Namanya A, Awan I, Cullen A, Disso J (2016) Cyber-attack modeling analysis techniques: An overview. In: Proceedings of the 4th International Conference on Future Internet of Things and Cloud Workshops, Vienna. pp 69–76. https://doi.org/10.1109/W-FiCloud.2016.29/W-FiCloud.2016.29

Bilge L, Han Y, Dell'Amico M (2017) Riskteller: Predicting the risk of cyber incidents. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM, New York. pp 1299–1311. https://doi.org/10.1145/3133956.3134022
Bolzoni D, Etalle S, Hartel PH (2009) Panacea: Automating attack classification for anomaly-based network intrusion detection systems. In: Kirda E, Jha S (eds). Recent Advances in Intrusion Detection. RAID 2009. Lecture Notes in Computer Science, vol 5758. Springer, Berlin, Heidelberg
Chen YZ, Huang ZG, Xu S, Lai YC (2015) Spatiotemporal patterns and predictability of cyberattacks. PLOS ONE 10(6):1–1. https://doi.org/10.1371/journal.pone.0131501
CPTC Organizing Committee (2017) Collegiate penetration testing competition at Rochester Institute of Technology. https://nationalcptc.org/
Du H, Yang SJ (2011) Discovering collaborative cyber attack patterns using social network analysis. In: Proceedings of the 4th International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction. Springer Berlin Heidelberg, College Park. pp 129–136
Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, KDD'96. pp 226–231. http://dl.acm.org/citation.cfm?id=3001460.3001507
Fava DS, Byers SR, Yang SJ (2008) Projecting cyberattacks through variable-length markov models. IEEE Trans Inf Forensic Secur 3(3):359–369. https://doi.org/10.1109/TIFS.2008.924605
Hackmageddon (2018) Hackmageddon information security timelines and statistics. http://www.hackmageddon.com/. Accessed 6 Feb 2018
Haddadi F, Khanchi S, Shetabi M, Derhami V (2010) Intrusion detection and attack classification using feed-forward neural network. In: Proceedings of the 2010 Second International Conference on Computer and Network Technology, IEEE Computer Society, Washington, DC. pp 262–266. https://doi.org/10.1109/ICCNT.2010.28. ICCNT '10
Hansman S, Hunt R (2005) A taxonomy of network and computer attacks. Comput Secur 24(1):31–43
Homer J, Varikuti A, Ou X, Mcqueen MA (68) Improving attack graph visualization through data reduction and attack grouping. In: Proceedings of the 5th International Workshop on Visualization for Computer Security. Springer-Verlag, Berlin. https://doi.org/10.1007/978-3-540-85933-8_7
Kovács F, Legány C, Babos A (2005) Cluster validity measurement techniques. In: 6th International symposium of hungarian researchers on computational intelligence
Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat 22(1):79–86. [https://doi.org/10.1214/aoms/1177729694]
Kuncheva LI (2008) Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: 2nd Workshop SUEMA. pp 5–10
Legany C, Juhasz S, Babos A (2006) Cluster validity measurement techniques. In: Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, World Scientific and Engineering Academy and Society (WSEAS) AIKED'06. Stevens Point, Wisconsin. pp 388–393
Li Y, Xia J, Zhang S, Yan J, Ai X, Dai K (2012) An efficient intrusion detection system based on support vector machines and gradually feature removal method. Expert Syst Appl 39(1):424–430. [http://doi.org/10.1016/j.eswa.2011.07.032]
Lin J (1991) Divergence measures based on the shannon entropy. IEEE Trans Inf Theory 37:145–151
Liu Y, Sarabi A, Zhang J, Naghizadeh P, Karir M, Bailey M, Liu M (2015) Cloudy with a chance of breach: Forecasting cyber security incidents. In: 24th USENIX Security Symposium (USENIX Security 15). USENIX Association, Washington, D.C. pp 1009–1024
Luo G, Wen Y, Lingyun X (2016) Network attack classification and recognition using hmm and improved evidence theory. Int J Adv Comput Sci Appl 7(4)
McGregor A, Hall M, Lorier P, Brunskill J (2004) Flow clustering using machine learning techniques. In: Pratt I, Barakat C (eds). Passive and Active Network Measurement. Springer Berlin Heidelberg, Berlin. pp 205–214
Ning P, Xu D, Healey CG, Amant RS (2004) Building attack scenarios through integration of complementary alert correlation methods. In: Proceedings of the 11th Annual Network and Distributed System Security Symposium. pp 97–111
Noel S, Harley E, Tam K, Limiero M, Share M (2016) Chapter 4 – CyGraph: Graph-based analytics and visualization for cybersecurity. In: Gudivada VN, Raghavan VV, Govindaraju V, Rao C (eds). Cognitive Computing: Theory and Applications, Handbook of Statistics, vol 35. Elsevier. pp 117–167. https://doi.org/10.1016/bs.host.2016.07.001
O'Callaghan L, Mishra N, Meyerson A, Guha S, Motwani R (2002) Streaming-data algorithms for high-quality clustering. In: Proceedings of the 18th International Conference on Data Engineering. pp 685–694. https://doi.org/10.1109/ICDE.2002.994785
Salerno JJ, Yang SJ, Kadar I, Sudit M, Tadda GP, Holsopple J (2010) Issues and challenges in higher level fusion: Threat/impact assessment and intent modeling (a panel summary). In: 2010 13th International Conference on Information Fusion. https://doi.org/10.1109/ICIF.2010.5711862
Shadi K, Natarajan P, Dovrolis C (2017) Hierarchical ip flow clustering. In: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks. ACM, New York Big-DAMA '17. pp 25–30. http://doi.acm.org/10.1145/3098593.3098598
Shandilya V, Simmons CB, Shiva S (2014) Use of attack graphs in security systems. J Comput Netw Commun:13. http://doi.org/10.1155/2014/818957
Song S, Chen Z (2007) Adaptive network flow clustering. In: Proceedings of the IEEE International Conference on Networking, Sensing and Control, London. pp 596–601. https://doi.org/10.1109/ICNSC.2007.372846
Strapp S, Yang SJ (2014) Segmenting large-scale cyber attacks for online behavior model generation. In: Kennedy WG, Agarwal N, Yang SJ (eds). Social Computing, Behavioral-Cultural Modeling and Prediction. Springer International Publishing. pp 169–177
Street WN, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York KDD '01. pp 377–382. https://doi.org/10.1145/502512.502568 http://doi.acm.org/10.1145/502512.502568
Subba B, Biswas S, Karmakar S (2016) A neural network based system for intrusion detection and attack classification. In: Proceedings of the National Conference on Communication (NCC). https://doi.org/10.1109/NCC.2016.7561088
Suricata (2019) An open source-based intrusion detection system (ids). https://suricata-ids.org/. Accessed 15 Jan 2019
Symantec (2017) Internet security threat report. https://www.symantec.com/security-center/threat-report. Accessed 25 Apr 2017
Valeur F, Vigna G, Kruegel C, Kemmerer RA (2004) Comprehensive approach to intrusion detection alert correlation. IEEE Trans Dependable Secure Comput 1(3):146–169
VERIS (2018) Veris community database (vcdb). http://veriscommunity.net/index.html. Accessed 6 Sept 2018
Wang K, Wang B, Peng L (2009) CVAP: validation for cluster analyses. Data Sci J 8:88–93
Wang L, Liu A, Jajodia S (2006) Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. Comput Commun 29(15):2917–2933. https://doi.org/10.1016/j.comcom.2006.04.001
Wang L, Jajodia S, Singhal A, Noel S (2010) k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks. Springer Berlin Heidelberg, Berlin
WASC (2018) The web hacking incident database. http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database. Accessed 6 Sept 2018
Wemmert C, Gançarski P, Korczak JJ (2000) A collaborative approach to combine multiple learning methods. Int J Artif Intell Tools 9(01):59–78
Xu K, Wang F, Gu L (2011) Network-aware behavior clustering of internet end hosts. In: 2011 Proceedings of the IEEE INFOCOM. pp 2078–2086. https://doi.org/10.1109/INFCOM.2011.5935017
Yang SJ, Stotz A, Holsopple J, Sudit M, Kuhl M (2009) High level information fusion for tracking and projection of multistage cyber attacks. Inf Fusion 10(1):107–121. https://doi.org/10.1016/j.inffus.2007.06.002
Yang SJ, Du H, Holsopple J, Sudit M (2014) Attack projection. In: Kott A, Wang C, Erbacher RF (eds). Cyber Defense and Situational Awareness. Springer International Publishing, New York City, chap Attack Projection. pp 239–261
Yen TF, Heorhiadi V, Oprea A, Reiter MK, Juels A (2014) An epidemiological study of malware encounters in a large enterprise. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM, New York, CCS '14. pp 1117–1130. https://doi.org/10.1145/2660267.2660330
Zhang J, Zulkernine M, Haque A (2008) Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C 38(5):649–659. https://doi.org/10.1109/TSMCC.2008.923876