Smoke Detector: Cross-Product Intrusion Detection With Weak Indicators

Kevin A. Roundy
Symantec Research Labs
kevin_roundy@symantec.com

Acar Tamersoy
Symantec Research Labs
acar_tamersoy@symantec.com

Michael Spertus
Symantec Corporation
mike_spertus@symantec.com

Michael Hart
Symantec Research Labs
michael_hart@symantec.com

Daniel Kats
Symantec Research Labs
daniel_kats@symantec.com

Matteo Dell'Amico
Symantec Research Labs
matteo_dellamico@symantec.com

Robert Scott
Symantec Corporation
rscott@sentryds.com

Abstract

The central task of a Security Incident and Event Manager (SIEM) or Managed Security Service Provider (MSSP) is to detect security incidents on the basis of tens of thousands of event types coming from many kinds of security products. We present *Smoke Detector*, which processes trillions of security events with the Random Walk with Restart (RWR) algorithm, inferring high order relationships between known security incidents and imperfect secondary security events (smoke) to find undiscovered security incidents (fire). By finding previously undetected incidents, Smoke Detector's RWR algorithm is able to increase the MSSP's *critical* incident count by 19% with a 1.3% FP rate.

Perhaps equally importantly, our approach offers significant benefits beyond increased incident detection: (1) It provides a robust approach for leveraging Big Data sensor nets to increase adversarial resistance of protected networks; (2) Our event-scoring techniques enable efficient discovery of primary indicators of compromise; (3) Our confidence scores provide intuition and tuning capabilities for Smoke Detector's discovered security incidents, aiding incident display and response.

1 INTRODUCTION

The massive volume and diversity of security events collected by security products today present challenges to traditional approaches to intrusion detection. Among others, email, host, and network security products are logging an increasing variety and volume of security events while analysts are already overburdened with a glut of security-related information. *Security Incident and Event Managers* (SIEM) have been developed to distill massive amounts of security data from multiple products into modest numbers of actionable security incidents. *Managed Security Service Providers* (MSSPs) address the same task, but on behalf of multiple customers, and therefore have even more security events to manage, from more products, and with greater scalability challenges. We studied these challenges in the context of a leading Managed Security Services Provider, which receives nearly two trillion security event instances per month, with more than 70 thousand distinct types seen in a typical month. These events are sent from tens of thousands of security devices, representing one hundred distinct security products.

ACSAC 2017, San Juan, PR, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
978-1-4503-5345-8/17/12…$15.00
DOI: 10.1145/3134600.3134645

When a SIEM or MSSP raises a security incident, it is usually based on the existence of a *Primary Indicator of Compromise* event, such as an anti-virus detection or blacklisted URL in the company's security logs [19]. The presence of a primary indicator furnishes sufficient evidence to directly warrant the creation of a security incident notification when it fires under pre-specified conditions. The remainder of security events are treated as unreliable *Secondary Indicators*, which include generic security and system events that are not necessarily attack-related. This classification can be made by the vendor of the security event, the security device, the SIEM, or by expert analysts in the *Security Operations Center* (SOC) of the MSSP or enterprise, typically with significant manual effort. In many SIEMs and MSSPs, large collections of expert rules are created and tuned (at significant cost) to identify combinations of primary indicators that should trigger an incident.

The identification of primary indicators is a major undertaking that is prone to mistakes because the number of different event types is so large and evolves so rapidly. Furthermore, attackers actively work to evade primary indicators by purchasing security products and tweaking their malware or attack until the product no longer detects it. For these attacks, any detection must be on the basis of secondary indicators. The mature sub-disciplines of intrusion detection, which include network anomaly detection, alert fusion [2, 10], alert correlation [33, 34], and root-cause analysis all focus on identifying intrusions by identifying instances of secondary indicators that are malicious or anomalous. The data sets for which the prior art was developed have either consisted of collections of homogeneous events coming from a single product, or have normalized the alerts so that a single model could be applied across different events [34]. By contrast, Smoke Detector is designed for the status quo in the SIEM and MSSP, whose events are in many cases produced by the intrusion detection systems

Instance Count	Confidence	Signature Name
44,104,172,561	0.01	TCP Connection
21,396,843,738	0.00	Traffic
19,493,074,472	0.00	UPD Connection
15,161,094,870	0.00	Firewall
14,586,679,865	0.02	Teardown TCP Connection
6,905,911,034	0.00	Teardown UDP Connection
6,762,250,252	0.00	Flow Session Close
4,350,063,989	0.01	PIX-6-305012
3,790,388,695	0.01	Connection Discarded
2,626,258,118	0.02	TPC Cache Miss
2,381,224,704	0.00	HTTP Get
2,109,964,562	0.02	Packet Permitted
1,676,587,290	0.02	Connection Allowed

Table 1: Instance counts for the most frequent events seen by an MSSP over a 30-day period.

mentioned above. Security events differ greatly in the reliability with which they identify infections and are so different in nature that much would be lost in an attempt to treat all events uniformly, as most of the prior art has done. Rather, we use the relationships between these distinct event types to each other, and to known security incidents (when available), in building Smoke Detector, a scalable graph-based security-incident detection framework that makes the following contributions:

(1) Smoke Detector replicates 96.4% of the primary-indicator-driven detection capabilities of a leading MSSP at a 1% False Positive (FP) rate, and discovering 19% more critical incidents at a 1.3% FP rate.

(2) We adapt the Random Walk with Restart (RWR) algorithm [9, 21], originally proposed for image segmentation, to cross-product incident detection with good adversarial resistance properties. To the best of our knowledge, RWR has not been applied to security problems outside of a social network context.

(3) We automatically score the confidence with which an event type indicates the existence of serious security incident. These scores provide security analysts with intuition and enable them to tune Smoke Detector and suppress false positives.

2 BACKGROUND

In MSSP and SIEM data, machines with unusually high event counts rarely exhibit serious security problems and are instead symptomatic of issues such as network outages and product misconfigurations. This occurs largely because the most frequent events produced by security products are generic traffic logs and status messages, as seen in Figure 1. While these generic events are useful in that they contain metadata that can be judged anomalous and lead to additional detections, by the time they are observed by a SIEM or MSSP, they have typically also passed through the intrusion detection systems deployed by various products, which also attempt to detect anomalous behaviour and report those detections to the MSSP or SIEM.

We therefore designed Smoke Detector to track the relationships between known security incidents and security events such that useful events are automatically identified, enabling us to rank possibly infected machines to detect novel incidents. Smoke Detector consists of two key components. First, it builds a graph that captures the relationships between events and the machines on which those events appear, using the Random Walk with Restart algorithm to propagate information from known-infected machines to the rest of the network. Smoke Detector's second component identifies the conditional probability with which an event indicates that a machine is infected as its confidence score, and uses these confidence scores to provide intuition, but also to weight and tune the graph used by Random Walk with Restart.

We proceed by describing our dataset in Section 3, our algorithm's graph-based techniques in Section 4 our confidence scoring algorithm in 5. Our implementation and its scalability properties are described in Section 6, for which we present qualitative results in Section 7 and a discussion of adversarial resistance in Section 8. We discuss Related Work in Section 9, and conclude in Section 10.

3 DATA DESCRIPTION

In this work, we assume a dataset that consists of records of the form $< m_{[t_i,t_j]}, s, l >$. In this formulation, $m_{[t_i,t_j]}$ is a window of time between t_i and t_j during which machine m was observed. We will henceforth refer to $m_{[t_i,t_j]}$ as a *machine-window* wherever appropriate for ease of explanation. The *security event* that was observed in relation to the machine-window is represented by s, which we elaborate on below. Finally, l indicates the location of the source of the event, that is, whether it originated on a machine that is internal or external to the enterprise. As noted by Lindqvist and Porras [18], the source-location provides important context for network attack events, such as port scans, which may be exceedingly commonplace when externally sourced, but indicative of a dangerous post-compromised machine attempting lateral internal movement otherwise. Since most network events involve two machines, they manifest as two records, one for the source machine and the other for the destination machine.

Security events represent an security-relevant action recorded by a security product, along with metadata that include a reference to the machine on which it occurred, or, in the case of a network event, the machines on which it occurred. Each security event represents either a *primary indicator* or *secondary indicator* of compromise. Primary indicators indicate the presence of a serious security issue on a machine that may warrant remediation or preventative action. These events are typically identified and vetted by processes that are manual-labor intensive. Examples of primary indicators include anti-virus detections, post-compromise command and control traffic, buffer overflows, and network attacks, which may be conditioned to have come from within the enterprise. All remaining security events are classified as secondary indicators, which vary dramatically in their ability to indicate security issues. The spectrum ranges from serious events that ought to be re-classified as primary indicators, to events that raise no suspicion, but may serve the purpose of providing context to a forensic investigation.

Some machine-windows in the dataset are known to contain *security incidents* that have been verified as representing serious

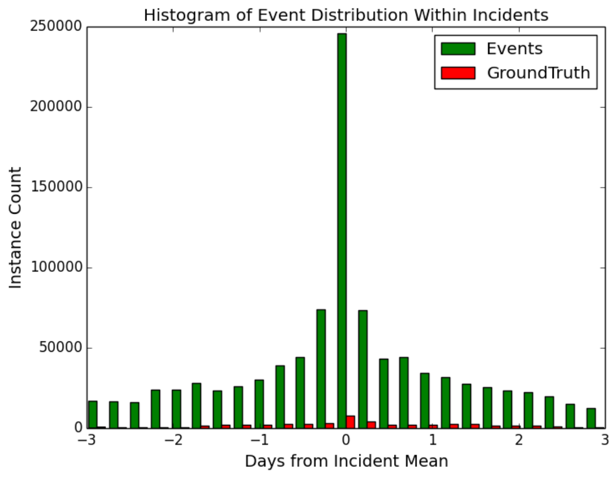


Figure 1: We plot the distribution of events corresponding to known security incidents relative to the mean timestamp of the incident's events. The distribution of primary indicators is also shown at 5x their actual rate to make them visible.

Time window each machine-window spans	1 day
Primary indicator security events	5,654
Secondary indicator security events	72,303
Machine-windows with security incidents	1,086
Unknown machine-windows	53,223,857
Earliest machine-window observed on	March 9, 2017
Latest machine-window observed on	April 24, 2017

Table 2: Summary statistics for the anonymized dataset provided by a Managed Security Services Provider for this work.

security problems. Each incident is then associated with some machine m between time window t_i and t_j. Most serious incidents are raised because of the presence of a primary indicator within this machine-window, but all secondary indicators that occur in this machine-window are included in incident machine-windows as well. The remainder of the security events in the dataset correspond to *unknown machine-windows*, that is, machine-windows that do not correspond to a known security incident, but that may contain security event data that analysts would classify as a security incident if it were brought to their attention.

What is the best width for a machine-window? Since security alerts appear in bursts corresponding both to benign and malicious actions, it is desirable for machine-windows to be small enough to prevent the admixture of too much benign behavior in security-incident windows, while keeping the windows large enough to capture the majority of malicious behavior that occurs around an incident. To answer this question, we studied the distribution of events around known security incidents and present this in Figure 1. We observe a large spike in event activity in the central 2 hours of a security incident, the majority of events fire in the central 24-hour period, and by the time 2 days have passed,

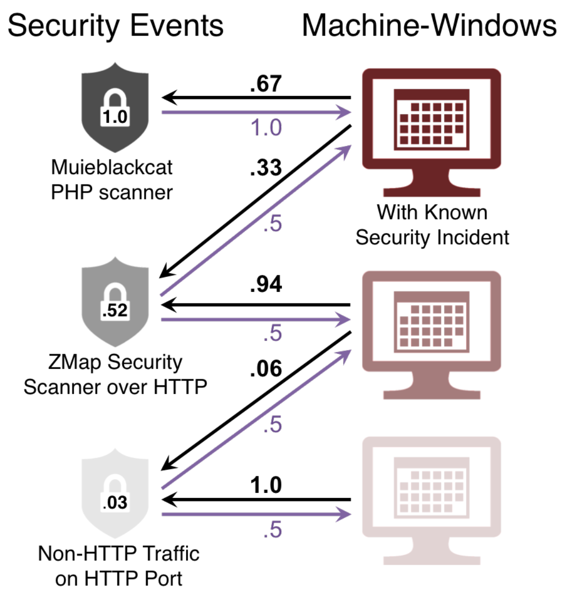


Figure 2: Example bipartite graph between security events (left side) and machine-windows (right side). The values inside the security event nodes are their confidence scores, which indicate the strength of their association with prior incidents. The confidence scores are used to assign normalized weights to the edges to reduce false alarms and for adversarial resistance. Our Smoke Detector algorithm employs Random Walk with Restart on this graph to rank the unknown machine-windows (bottom two in the figure) based on their relevance to the known security incidents for analyst review towards undiscovering new incidents.

the event volume is dominated by background noise. Accordingly, in obtaining a dataset from a leading MSSP we requested that the length of each machine-window be set to 1 day as providing us with a good tradeoff. Table 2 reports this and other summary statistics of the anonymized dataset that we obtained.

4 INCIDENT RANKING & PRIORITIZATION

Given millions of unknown machine-windows, most of which contain no primary indicators, how can an analyst identify the tiny fraction of them that represent undetected security incidents? In this section, we discuss our core methods for ranking and prioritizing the unknown machine-windows based on their likelihood of containing an undiscovered incident. Toward this goal, we aimed for a technique that can (i) operate in an one-class setting like ours where the only class we have is that of known security incidents (i.e., the unknown machine-windows do not constitute a class of their own, as each might well contain an incident), (ii) capture high-order, indirect relationships between security events and machine-windows, such as that between the top and bottom machine-windows in Figure 2, (iii) produce a ranking of potential incidents based on intuitive principles that can be easily consumed by the analysts, and (iv) scale to large amounts of data. We considered a number of machine learning approaches, ranging from traditional classification and regression techniques to more complex deep learning techniques. While some traditional machine learning methods perform single-class classification, output interpretable results, and are scalable, they typically consider each data point in isolation and hence cannot easily capture high-order relationships. Deep learning techniques, on the other hand, can handle a single class and capture high-order relationships, but they produce results that are often hard to interpret [12, 15] and scaling these techniques to large amounts of data is still an active area of research [6, 7, 13, 32].

These considerations led us to pursue a graph-based approach as a method that satisfies our solution criteria and provides effective solutions. Accordingly, we model the relationships between security events and machine-windows as a bipartite graph, and use scalable algorithms to propagate information from machine windows that contain security incidents to the rest of the graph. This formulation allows us to identify and rank unknown machine-windows that are likely to contain a security incident. Our bipartite graph represents each security event and machine-window as a node, and contains directed edges between those security events and machine-windows that appear together in the same record in our dataset (one edge in each direction, see Figure 2).[1] Overall, our graph has 53,302,900 nodes and 183,610,432 edges.

The intuition behind our approach is that the unknown machine-windows in the graph that have the closest relationships to machine-windows with known security incidents are most likely to contain security incidents themselves. By adopting algorithms that measure the relevance of machine-windows to known security incidents, we are able to rank undiscovered incidents highly. More precisely, we employ the Random Walk with Restart algorithm [9, 21] to identify undiscovered incidents from among tens of millions of unknown-machine-windows. To the best of our knowledge, the application of this algorithm to incident ranking and prioritization is novel.

4.1 Random Walk with Restart

Informally, a random walk with restart (RWR) over a graph can be described as follows [28]. Consider a random particle that starts from node i in the graph. The particle iteratively transmits to its neighboring nodes with a probability proportional to the corresponding edge weights. Additionally, at each step, there is a chance that the particle transports back to node i with probability c. The RWR score of node j with respect to node i is defined as the steady-state probability that the particle will be found at node j. The RWR score of node j indicates its *relevance* to node i. For graphs with multiple start nodes, the relevance scores of the nodes in the graph are computed with respect to all the start nodes. The restart mechanism used by RWR ensures that the nodes that are most proximate to start nodes have higher rank than distant nodes.

Formally, RWR can be defined as the following linear system:

$$\vec{r} = (1 - c)\tilde{W}\vec{r} + c\vec{e} \qquad (4.1)$$

where, assuming that the graph has n nodes, $\vec{r}$ is the $1 \times n$ RWR vector that contains the relevance scores for all the nodes, $\tilde{W}$ is the column-normalized $n \times n$ adjacency matrix that contains the weights of the edges between the nodes, c is the restart probability, and $\vec{e}$ is the $1 \times n$ starting vector with 1's for the start nodes and 0's for the remaining nodes. Directly solving Equation 4.1 involves an expensive matrix inversion, with a total time complexity of $O(n^3)$ [36]. In practice, the solution is approximated with the power iteration method, where Equation 4.1 is initialized with an RWR vector of all 1's and solved iteratively with the RWR vector from the $(t - 1)$th iteration used to update the RWR vector in the tth iteration. For t iterations, this method costs $O(tm)$, where m is the number of edges in the graph [36], improving the scalability of the approach when it is run for a small number of iterations.

In our application of RWR to cross-product intrusion detection, we select those machine-windows that correspond to known security incidents as our start nodes, and we calculate the relevance of all other unknown-machine windows and security events to these start nodes. These relevance scores form a natural ranking over the 52 million unknown machine-windows in our dataset, the most relevant of which are highly likely to represent previously undiscovered security incidents. Detection systems cannot be tuned to suppress false-alarms cause frustration and are likely to cease to be used. In designing Smoke Detector, therefore, it is vital that our system be responsive to feedback from security analysts, but how to do so? The structure of the graph is entirely determined by event data and should not be tampered with, and RWR measures relevance to a single class of true positive nodes, with all other nodes unknown. Neither does it make sense to associate a negative class with false positives, as these will have been selected from among the most suspicious machine-windows and are not as negative in their nature as the average unknown machine-window.

The way to achieve a tunable system lies in RWR's use of edge-weights. As mentioned above, when a random particle leaves a machine-window, it selects an outbound edge to an attached security event with probability proportionate to the edge's weight. Future false-positive detections can therefore be prevented by adjusting edge weights in response to False Positive reports and other forms of analyst feedback. For relevance scores to have a probabilistic interpretation, RWR assumes that the adjacency matrix of the input graph is column-normalized, meaning that the sum of the weights of each node's outgoing edges is 1. Then, the simplest initial policy for setting the edge weights is to assign them uniformly (i.e., each outgoing edge of a node with o out-neighbors receives the weight $1/o$), and while this choice does produce good results, it is possible to do better by associating edge weights with measurements of security-event quality. Accordingly, we assign the edge weights associated with a security event based on the conditional probability that a security incident will be raised if that the event is observed in a machine-window, which we refer to as the confidence score for the event and describe in Section 5. An example application of this edge weighting policy can be seen in Figure 2, where, for each security event node, we set the weights of all the outgoing edges to the confidence score for the event, which are then normalized, and for each machine-window node, we sum the confidence scores of all the in-neighbor events, which is then distributed among the outgoing edges with weights in proportional to the in-neighbor events' scores.

[1] We generate two directed edges between a pair of security event and machine-window to be able to assign different edge weights depending on the source node (see Section 4).

5 CONFIDENCE SCORING

Smoke Detector provides *confidence scores* for security events, which represent the conditional probability that a machine-window contains a security incident given that the event was observed in relation to it. We seek to fulfill three goals in providing these confidence scores. First, since novel security incidents identified by Smoke Detector may consist of many events, these scores serve to bring the most important events to the attention of the analyst. Second, these conditional probabilities have a clear interpretation that analysts can readily understand. Third, these scores provide an intuitive means by which the RWR algorithm can be tuned, and are used as edge-weights in that algorithm.

While RWR can score the importance of security events in the graph, its "relevance" scores do not fulfill the three goals listed above. Since RWR's relevance score for an event is the steady-state probability of a particle being found at the event in question, these scores are extremely low and are furthermore, they have a skewed distribution, bearing hardly any relation to the more intuitive conditional probabilities that we provide as confidence scores through the techniques of this section.

5.1 Modeling Event Confidence on Correlation with Primary Indicators

To measure the conditional probability with which an event indicates a serious compromise, we measure each security event's correlation with primary indicators. As defined in Section 3 primary indicators are those security events that should warrant the raising of a security incident whenever they are observed. A set of reliable primary indicators is typically easily obtained from any MSSP or SIEM, as it these are fundamental to their incident detection methods [19]. Should such a set not be available, an adequate set of primary indicators can be identified with some manual effort thanks to the naming conventions used by many security vendors for high-fidelity events, and through the numeric severity scores that many of them provide. Since primary indicators trigger only in the context of serious security issues, any instance of a security event that happens in the same machine-window as a primary indicator is likely to have triggered in response to malicious behavior.

Measuring confidence on the basis of co-occurrence with primary indicators is appealing because high scores can be explained and justified on the basis of primary indicators of known reliability. More precisely, we score events based on the fraction of the time in which they appear in the same machine-window as at least one primary indicator. The selection of machine-windows that are 24-hours in length offer a good trade-off between capturing the majority of incidents associated with the incident and eliminating background noise, as seen in Figure 1.

5.2 Confidence Estimation With a Prior Distribution

While MSSPs may record trillions of event instances per month, rare event types may be observed only a handful of times. To ensure that we do not overestimate the confidence of rare events, we apply a skeptical prior belief of event confidence into our scoring model. This ensures a low false positive rate, and that high confidence scores are earned on the basis of sufficient evidence, which promotes trust on the part of incident responders.

We model each event's instances as a Bernoulli random variable. An individual instance can be thought of as a trial, where a successful trial is one that occurs within a compromise-related context. We can formalize as:

$$\theta \sim Beta(\alpha, \beta) \tag{5.1}$$

$$y_{event} \sim Bern(\theta) \tag{5.2}$$

Therefore, θ is drawn from skeptical prior belief of the distribution of event confidence, which in turn influences the likelihood of the event occurring in a compromise-related context. But how do we choose α and β?

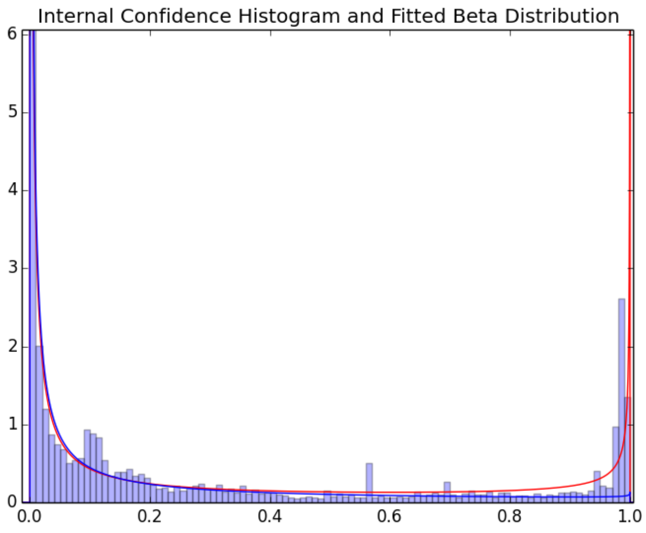
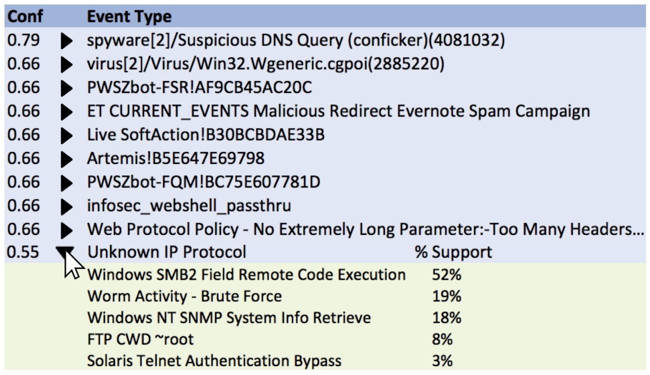


Figure 3: We plot a histogram of confidence scores measured via maximum likelihood estimation for events that appear 10 or more times, and fit a beta distribution to the data (the red curve). As this distribution is insufficiently skeptical for use as a prior probability distribution, we fit a second beta distribution to the data, constraining its minimum value to occur at confidence = 0.9 (the blue curve).

We adapt Empirical Bayes techniques [4] to our purposes, modeling the prior probability distribution based on the data, and applying Bayesian inference to arrive at the posterior probability distribution. Our prior is a Beta distribution that we fit to events that have been observed at least ten times. Since we desire our prior distribution to be skeptical, rather than applying the best-fit Beta prior, which is shown as the red curve in Figure 3 and has a minimum value at roughly 60%, we solve for a more skeptical prior with minimum value of 0.9, resulting a beta distribution prior with parameters $\alpha = 0.0499$ and $\beta = 0.8944$, which is shown as the blue curve in the same figure.The resulting empirically motivated prior distribution is generally skeptical about the confidence of most events, but does not rule out the existence of a small but important minority of events that are highly indicative of machine compromise.



Conf		Event Type	
0.79	▶	spyware[2]/Suspicious DNS Query (conficker)(4081032)	
0.66	▶	virus[2]/Virus/Win32.Wgeneric.cgpoi(2885220)	
0.66	▶	PWSZbot-FSR!AF9CB45AC20C	
0.66	▶	ET CURRENT_EVENTS Malicious Redirect Evernote Spam Campaign	
0.66	▶	Live SoftAction!B30BCBDAE33B	
0.66	▶	Artemis!B5E647E69798	
0.66	▶	PWSZbot-FQM!BC75E607781D	
0.66	▶	infosec_webshell_passthru	
0.66	▶	Web Protocol Policy - No Extremely Long Parameter:-Too Many Headers...	
0.55	▼	Unknown IP Protocol	% Support
		Windows SMB2 Field Remote Code Execution	52%
		Worm Activity - Brute Force	19%
		Windows NT SNMP System Info Retrieve	18%
		FTP CWD ~root	8%
		Solaris Telnet Authentication Bypass	3%

Figure 4: Partial display of events and metadata contributing to an incident detected by Smoke Detector. The analyst has clicked on an event to cause its confidence score to be explained in terms of its correlated ground-truth events.

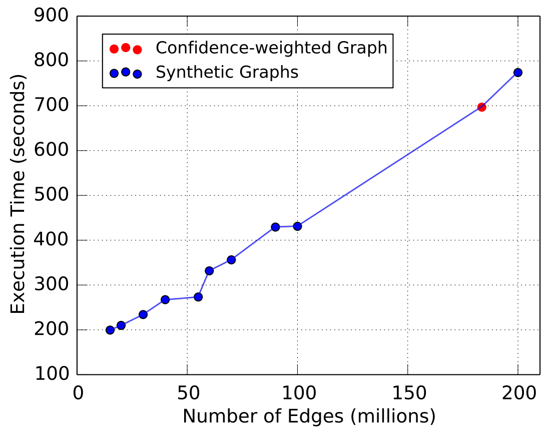


Figure 5: Performance characteristics of our RWR implementation on both our confidence-weighted and synthetic graphs.

5.3 Feedback Through Transparency

By basing Smoke Detector's confidence scoring on correlation with primary-indicator events, we are able provide intuition and transparency into the relationship-based nature of our overall approach and to prevent the mistrust that opaque models tend to engender. Figure 4 shows how this transparency can be used in presenting a Smoke Detector incident to an incident responder. To facilitate feedback, we explain the confidence of an event by exposing the degree of support that each primary indicator contributes to its score, based on its co-occurrence rate. In this particular case, the "Unknown IP Protocol" event is shown to bear strong relation to primary indicators that are more obviously severe.

This window into our algorithm provides insight into the context in which the event typically triggers, which may hint at the possibility of a more serious underlying problem. Third, when trained in an MSSP, Smoke Detector a customer with few security products can learn of alerts that non-deployed product likely would have identified had it been deployed. Finally, this transparency allows analyst to reduce false positives by removing unreliable events that should not be considered primary indicators for purposes of confidence scoring. Similarly analysts may increase true positives by identifying events that should re-classified as primary indicators.

6 IMPLEMENTATION

Smoke Detector consists of two parts: incident detection based on RWR and confidence scoring. We focus the implementation description of this section on the RWR algorithm, which dominates the performance cost, since confidence-score computations need to be frequently computed and can be efficiently calculated over the RWR graph using the algorithm of Section 5.

To scale with the size of our workloads (upwards of 150 billion events daily), we implemented RWR in Apache Spark, a framework for running distributed computation on multi-node clusters, with the aim of facilitating horizontal scaling [35].

In implementing RWR, we perform the power iteration method for 10 iterations, which is sufficient to achieve an accurate estimation of steady-state probabilities [11]. We set its restart probability c to 0.15, a value commonly used in practice [27].

As discussed in Section 4, the power-iteration method has a time complexity linear in the number of edges [36]. To confirm that our implementation conforms to this expectation, we evaluated it on Amazon EMR using a cluster of 15 machines, each of type r4.2xlarge with eight cores and 61GB RAM. To that end, we modeled several synthetic graphs of increasing edge counts on the structural properties of our dataset. These synthetic graphs were also bipartite, with the size of each partition and ratio of total edges proportionally preserved. The graphs were stored in an adjacency list format in an object store [1]. Scalability results are shown in Figure 5, confirming that the algorithm scales linearly with the number of edges in the graph, in accordance the theoretical bound. Runtime performance on the real dataset is also shown.

7 EXPERIMENTS

In this section, we evaluate Smoke Detector based on its ability to detect security incidents, and follow this with a discussion and evaluation of its adversarial resistance in Section 8. We begin by showing that Smoke Detector provides a straightforward, low-maintenance method for replicating the critical incidents detected by a leading MSSP with high coverage and accuracy. We then study the machine-windows to which Smoke Detector gave the highest rank, and present these to analysts to identify the percentage of these detections that represent undiscovered security incidents. Finally, we assess the ability that Smoke Detector's confidence scores have in assisting analysts to identify primary indicators of compromise that had previously been classified as secondary indicators.

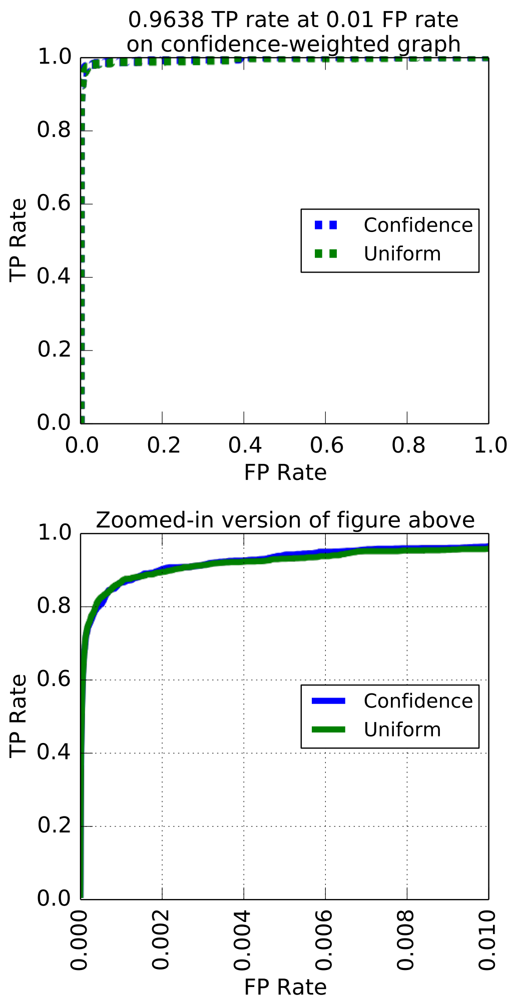


Figure 6: We show Receiver Operating Curves (ROC) to describe Smoke Detector's ability to reconstruct critical MSSP incidents triggered by primary indicators, achieving more than 95.6% coverage (true positives, TP) at a 1% false positive (FP) rate with uniform edge weights and 96.4% coverage with edge weights informed by event confidence scores.

7.1 Detecting New and Existing MSSP Incidents

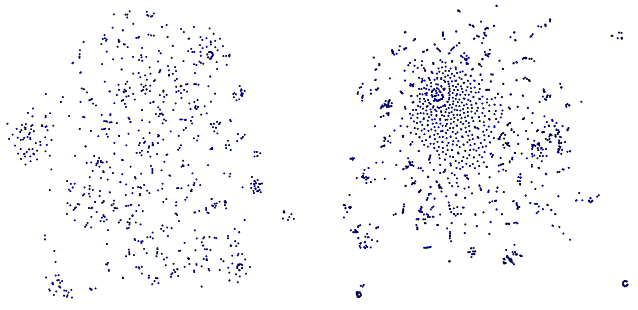
We commence our evaluation by examining Smoke Detector's suitability as a less expensive replacement for a leading MSSP's identification of critical severity incidents based on rules conditioned on the presence of primary indicators. Freeing MSSPs and SIEMs from such techniques is desirable given that the identification of primary indicators involves and the conditions under which they should raise security incidents is a costly and largely manual process. In replicating the MSSP's incidents, it is not sufficient for Smoke Detector to learn what the primary indicators are, as these are quite prevalent in non-incidents, it is also necessary to identify the conditions set by the MSSP for raising an incident when a primary indicator is present. To demonstrate that the RWR algorithm can learn both the primary indicators and the conditions under which they raise incidents, we compare the results of RWR with confidence-weighted edges to RWR with uniformly weighted edges, for which no knowledge of primary indicators is granted to the model.

Accordingly, we perform a 10-fold cross-validation experiment in which the positive class represents machine-windows corresponding to incidents of critical severity raised by of a primary indicator, while the negative class consists of all unknown-machine windows. The results of this experiment are shown in Figure 6. The uniformly-weighted RWR model achieves 95.6% coverage of True Positive (TP) incidents at 1% False Positive (FP) rate with no prior knowledge of primary indicators or event quality. While the weighted graph's coverage is only modestly higher, 96.4% coverage at the same FP rate, its tunable weights allow us to obtain a larger yield of valuable undiscovered security incidents in the 1% of unknown-machine days that are treated as "False Positives" in this experiment. The use of uniform weights results resulted in a surprisingly modest drop in classification accuracy. These weights do serve an important purpose however, in that the relevance rankings produced by the weighted graph prioritize novel detections that are relevant to known incidents through high-confidence events rather than on the basis of more prevalent lower-confidence events, as in the case of the uniformly weighted graph.

Since Smoke Detector's primary goal is to find undiscovered security incidents, we had a professional incident responder employed by the MSSP evaluate the top-ranked unknown machine-windows identified by Smoke Detector to determine whether they contained legitimate security incidents. This involved applying RWR with confidence-weighted edges to the 52 million unknown machine-windows in our dataset. We presented the incident responder with a sample of 100 of the top 708 machine-windows identifed by RWR. To maximize the coverage of the analyst's feedback, we clustered the 708 machine-windows with the DBSCAN algorithm, which yielded 25 clusters covering 30% of the machine-windows (the rest were unclustered), and sent the analyst a representative machine-window for each of these clusters, along with a stratified sample of 75 additional machine-windows. By weighing the incidents based on cluster size, we arrive at the results shown in Table

Severity Score	Count	Weighted Count	Weighted %
Emergency	2	2	0.6%
Critical	74	234	80.4%
Warning	17	48	16.5%
Informational	3	3	1.0%
False Positive	4	4	1.4%
Totals	100	291	100%

Table 3: Result of having an MSSP's incident responder evaluate 100 machine-windows sampled from the top 708 that were identified by Smoke Detector as containing probable security incidents. These results indicate that Smoke Detector increases the MSSP's volume of incidents that are critical or above in severity by 19% at a 1.3% False Positive rate, while also finding some security incidents of lower value.



(a) Smoke Detector Incident Clusters (b) MSSP Incident Clusters

Figure 7: A two-dimensional representation of clustering results applied to novel incidents identified by Smoke Detector, and to pre-existing incidents detected by the MSSP. Smoke Detector incidents are more diverse than the MSSP incidents, which are dominated by a large cluster.

3. Of the 291 incidents that are represented in this sample, 236 were judged to be of critical severity or higher. Extrapolating from this ratio, we expect that approximately 574 of the 708 top-ranked machine-windows in our dataset represent critical incidents, which represents a 19% increase in the number of critical incidents detected by the MSSP over this time period, at a 1.4% False Positive rate.

Next, we evaluate the diversity of the previously undiscovered security incidents identified by Smoke Detector in comparison to the critical incidents detected by the MSSP. We were pleased to find that the novel incidents found under these conditions are highly diverse, even when compared to the MSSP's incidents. To illustrate this, we used t-Distributed Stochastic Neighbor Embedding (t-SNE) [30] to produce a 2-dimensional clustering of 1000 Smoke Detector incidents, and 1000 MSSP incidents from the same time-period. Figure 7a shows that Smoke Detector produces many outliers and small clusters, in clear contrast to the MSSP's incidents, shown in Figure 7b, which feature a dominant cluster.

7.2 Applying Smoke Detector to Primary Indicator Classification

In addition to incident detection, we wanted to examine whether Smoke Detector's confidence generation algorithm could assist with the important problem of detecting primary indicators in an ecosystem of heterogeneous security devices from many vendors. Traditionally, security events manually classified into primary indicators, and secondary events are classified as supporting events, and noise. Events labeled as noise are not considered to be useful, and are ignored unless a customer specifically requests to see all events associated with a machine. Supporting events show up alongside primary indicators whenever there is an infection and provide important context, while primary indicators are used to raise new security incidents.

Keeping current accurate classifications is a critical challenge for MSSPs and SIEMs for several reasons:

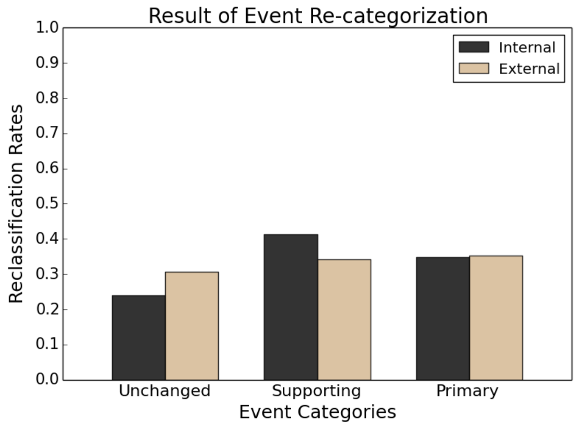


Figure 8: Result of re-classification effort for events, fed by candidate events identified by Smoke Detector as having confidence above 0.9.

(1) MSSPs and SIEMs encounter many previously unseen event types every day, making for a daunting manual classification problem. For example, in a 30-day period, the MSSP we studied encountered an average of 957 new event types per day.
(2) With over a million event types actively tracked, even a small error rate will lead to a large number of misclassified events over time.
(3) As the security landscape changes, the proper classification of an existing event type may change. Regular reexamination of all event types is impractical at the scale that MSSPs and SIEMs have to deal with.

We posited that Smoke Detector's confidence scoring algorithm could identify indicators with incorrect classifications. To this end, MSSP analysts began a regular review process that has so far resulted in the review of 840 event types that had attained a confidence score greater than 0.9 and that had been previously been ignored as noise. This reexamination resulted in 280 events previously regarded as noise being reclassified as primary indicators, and another 342 noise events reclassified as supporting events (see Figure 8 for the reclassification rates of internally- and externally-sourced events). As only 0.06% of events manually classified over the preceding 30 days had been re-classified as primary indicators, with 1.85% reclassified as supporting events, Smoke Detector's identification of a set of candidate events that consist of 33% primary indicators, and 41% supporting events was clearly a major improvement over generic reexamination.

The impact of these re-classified events was tracked for 10 days, resulting in enhancements to 224 validated security incidents being published to 49 distinct monitored organizations. Of these incidents, 10 incidents occurring at nine distinct organizations were validated as critical incidents on severity criteria closely tracking NIST-800-601 [5], including seven Unauthorized Vulnerability Scans, two Trojan Horse Infections, and one hack tool attack. Notably, for the two Trojan Horse infections, Smoke Detector was responsible for

	Detection rates	
Indicators evaded	**MSSP**	**Smoke Detector**
One primary	48.35%	67.21% TP rate at 0.1% FP rate
		82.73% TP rate at 1% FP rate
Two primary	33.01%	56.1% TP rate at 0.1% FP rate
		80% TP rate at 1% FP rate

Table 4: Incident detection rates in an adversarial setting where the attacker evades primary indicators. The MSSP relies on primary indicators, hence their detection rates suffer significantly, where Smoke Detector manages to retain high coverage by leveraging secondary indicators.

labeling the primary indicator, without which the infections would have remained unidentified.

8 ADVERSARIAL RESISTANCE DISCUSSION

For any detection system to maintain its effectiveness over time, it is must be resistant to adversarial attacks. In this section, we discuss the adversarial methods that we believe are the most likely to have a negative impact on Smoke Detector.

Attack 1: Evasion of primary indicators. We consider the evasion of primary indicators to be the most probable attack against Smoke Detector, since it is obviously beneficial to the attacker regardless of whether Smoke Detector is deployed as a backup to the individual security products that are in place to monitor the attack. Such attacks against primary detections, such as attacking antimalware software by obfuscation are both practical and prevalent [20]. Likewise, it is reasonable to expect malware writers to purchase an antivirus and not release their malware to the wild until they have developed a version that is not detected. Such attacks are evidently costly, however, to the point of being beyond the means of many attackers, since SIEMs and MSSPs are able to detect many serious attacks on the basis of primary indicators. We therefore assume an adversarial model in which the attacker has a constrained budget.

We run experiments to measure the extent by which primary-indicator evasion degrades Smoke Detector's ability to alert on incidents relative to the MSSP, which detects incidents based on the existence of one or more primary indicators. In the case of an attack that triggers a single primary indicator and raises no other security events at all in this setting, it is obvious that both Smoke Detector and the MSSP perform equally badly, since there is no evidence left on which to base a detection. The more interesting case is that of an incident involving multiple events, of which the attacker is able to remove one or more primary indicators, since machines that provide a single primary indicator are typically poorly monitored and reporting of useful additional secondary indicators can be easily accomplished through the reporting of Windows Security Event Logs and Unix-system equivalents. As shown in Table 4, when the attacker evades one primary indicator but secondary events remain, the MSSP's detections are reduced to the 48% of the original haul of incidents (those that were raised by multiple primary indicators remain), while Smoke Detector's incident coverage retains 83% coverage at a 1% FP rate, thanks to its ability to detect security incidents entirely on the basis of secondary indicators. In the case of a more powerful attacker capable of evading two primary indicators (but not all secondary indicators), Smoke Detector again outperforms traditional primary-indicator-based detection techniques, retaining 80% coverage at a 1% FP rate, while the MSSP's coverage drops to 33%.

Attack 2: Drowning the signal in noise. It may prove less expensive for an attacker to drown out the signal provided by strong indicators in noise than to prevent these events from triggering. Such attacks are not restricted to adversaries that have Smoke Detector in mind, since human security analysts can easily miss vital details in incidents that at times consist of millions of event instances and hundreds of distinct event types, as in the case of Denial of Service attacks that mask the presence of a more severe targeted attack [16]. This aspect of the problem is addressed by Smoke Detector's confidence scores, which allow the events that compose a security incident to be listed in decreasing order of confidence, as shown in Figure 4.

To assess the impact of a noise-based attack on Smoke Detector's core detection capabilities, we added noise to known incidents, both on confidence-weighted and uniformly-weighted versions of Smoke Detector's graph, under the assumption that the confidence weights would provide increased resistance to an attacker. The adversary attempts to reduce the relevance rank of an attack by triggering those security events that have the least relevance to known security incidents, by establishing new edges between them in the graph. We attempted this attack under many conditions, adding anywhere from 10 to 100 edges, and selecting the events to which we connected the incident according to both black-box policies (random selection of secondary indicators) and white-box policies (attacker has knowledge of event confidence and relevance rankings), and both on the confidence-weighted and uniformly-weighted graphs.

The counter-intuitive result to our experiments was that adding noise to security incidents has the unintended effect of increasing its overall ranking, regardless of the graph's weighting scheme. The result of one such adversarial experiment is shown in Figure 9, wherein we assumed adversaries that attach their incident to 100 security events, selected at random from the 10% least relevant events in the confidence-weighted graph. This figure contrasts the results of doing so on Smoke Detector's classification accuracy, finding that adding edges results only increases performance relative to the non-adversarial setting. Our experiments have shown that this effect is not due to the fact that multiple nodes are attacked at once in this evaluation, as adding edges to a single incident also increases its rank relative to other machine-windows.

The reason for which the noise-based attack fails is that adding edges to a machine-window node increases its number of its incoming edges, along which event nodes communicate their relevance to the machine-window. This has a net positive effect on the node's ranking as there is no normalization amongst a machine-window's incoming edges, yielding a net positive effect on the machine's relevance even if these additional edges come from events that have low, but positive relevance.[2] This consideration more than

[2]There is a path between each security event and at least one incident in the graph, hence all the events receive a positive relevance.

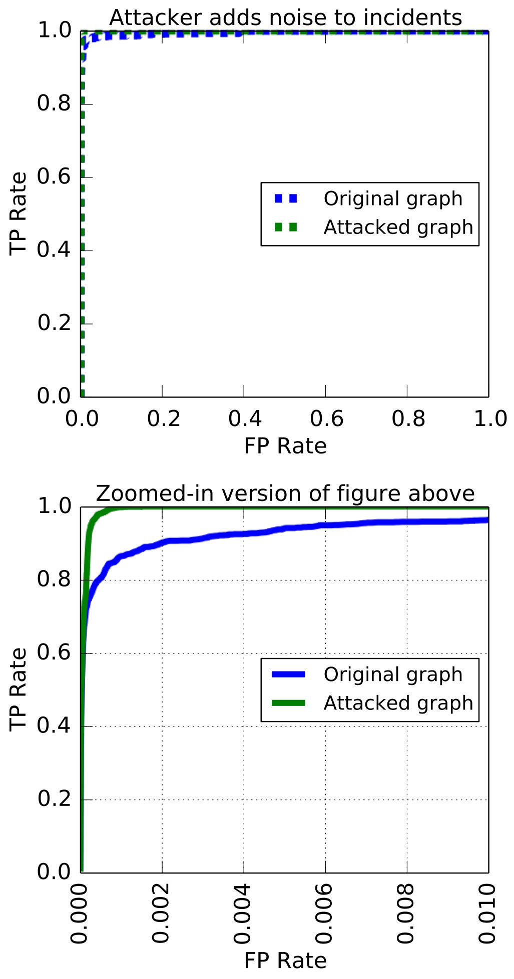


Figure 9: Smoke Detector's detection rates in an adversarial setting where the attacker adds noise to the incidents in the confidence-weighted graph. The attack fails as new edges further increase the relevance of the incidents.

counterbalances the diminished outgoing relevance of the machine-window to high-value events, as the relevance of these events is supported by other machine-windows that are not attacked.

Attack 3: Altering the graph and its weights. We now consider adversaries that control at least one machine in addition to the machine-window on which they hope to evade detection. That is, they seek to modify Smoke Detector's training data, which is a daunting task given that MSSPs use big data systems comprising trillions of event instances across tens of millions of machine windows, each protected by its own sensor net of security devices. Attacks are most likely to try to cause the rank of the primary attack's machine-window to drop into a range that has many false positives by lowering the relevance of the events connected to the primary attack's machine-window. Consider a primary attack consisting of events s_1 and s_2. The adversary divides controlled machines into two pools, one of which triggers event s_1, the other triggering s_2, to avoid triggering both events from a single machine as this would raise an incident that would increase the relevance of events s_1 and s_2. The broad class of attacks that includes this one has several limitations:

First, Smoke Detector treats a single instance of an event in a machine-window no differently than millions of instances of the same event in that window, so successful attacks require the control of many machines on multiple networks over a long period of time, which is costly and increases the probability of detection.

Second, model inversion attacks depend upon feedback from the model, but this risk can be mitigated by providing limited precision in the confidence scores that Smoke Detector shares with customers [8], and by sorting events alphabetically by name rather than by rank when they have tied limited-precision score.

Third, an effective training-set attacks on Smoke Detector would have to reduce the score of highly indicative events by triggering them many times without raising a security incident, which is difficult and could culminate in the blacklisting or lost control of attacker-controlled machines.

Lastly, the adversary is more likely to have control of external machines than enterprise-owned machines in sufficient numbers for an attack, yet only internal machines can trigger events produced by endpoint-protection products, rendering training-set attacks on anti-virus events, for instance, difficult to carry out.

In summary, while these attacks are surely not comprehensive of all possibilities, Smoke Detector has important properties that render it a difficult target for an adversary, such as its ability to capture high-order relationships, the relative inaccessibility of its model to concerted model inversion attempts, and its basis in big data, which forces attackers to perform large-scale attacks.

9 RELATED WORK

Much of the prior art in intrusion analyzes large collections of similar events to alert on a subset of malicious instances. By contrast, Smoke Detector consumes raw event data that is intermixed with the alerts generated by the prior art, and therefore treats events of different types as distinct entities of varying reliability. The prior art we discuss in this section includes techniques for event scoring, alert fusion, alert correlation, and root-cause analysis, among others.

M-Correlator [23] assigns *priority* scores to security events based on context, such as the importance and vulnerability of targeted assets. When such data is available it should be used and would enable an approach based on M-Correlator to prioritize Smoke Detector incidents, whose event scores represent infection confidence rather than severity.

Beehive [34] provides important techniques for normalizing and correlating the events provided by diverse collections of security products (which are central challenges for MSSPs and SIEMs) so that metadata is not lost, allowing anomaly detection algorithms and blacklists to be applied effectively. Smoke Detector makes no attempt to identify anomalies in event types that would otherwise not attract attention, as Beehive does, and would instead consume Beehive's detected anomalies as separate events.

Alert correlation and alert clustering techniques bear similarity to Smoke Detector in their study of the relationships between different types of alerts. However, many of these methods strive to promote understanding and root-cause analysis for existing incidents [22, 24, 31] rather than detect new incidents. Julisch [14] summarizes large volumes of event data for human consumption, so that anomalous events stand out to an analyst. Viinika et al. [33] and Valdes and Skinner [29] reduce alerts into aggregate "meta-alerts" for ease of consumption. These technique are valuable but are not designed for a SIEM or MSSP setting, where events vary widely different nature and quality.

Alert Fusion refers to the problem of determining whether to output an alert based on the result of multiple IDS models trained over a common data set of event logs [2, 10, 25, 26], such as the Darpa 2000 dataset [17]. These techniques are not designed for underlying datasets that are as diverse as those of SIEMs and MSSPs.

Many expert systems for intrusion-detection build on P-BEST's seminal algorithm [18], which provides an inference language to detect misuse and anomalies through production rules. While similar techniques are deployed by SIEMs and MSSPs today, these techniques are costly to maintain because of constant changes to the underlying events that are being collected and monitored. In Section 7 we evaluate Smoke Detector against an MSSP of this type. Buckzak and Guven [3] describe many applications of Machine Learning and Data Mining to intrusion detection, from which Smoke Detector differs in its use of a graphical model, which models relationships between known incidents, events and candidate incidents.

10 CONCLUSION

We presented Smoke Detector, an intrusion detection system designed for the challenges confronted by MSSPs and SIEMs, including huge data volumes, diverse event types of greatly varying quality, and continual event churn. We showed that Smoke Detector increases the volume of detected critical incidents by 19% at a 1.3% False Positive rate. Its confidence scores provide intuition and a tuning mechanism for Random Walk with Restart (RWR). Our implementation of the RWR algorithm scales linearly with the size of the data and works well in a distributed system on commodity hardware. To the best of our knowledge, our use of the RWR represents the first use of this algorithm for computer security applications other than in the context of security for social networks [37].

11 ACKNOWLEDGMENTS

Thanks to Lenore Zuck and Zhongkai Wen for helpful discussions on improving this paper. Thanks to Daniel Whalen for provide feedback on the Smoke Detector algorithm and its detections.

REFERENCES

[1] Amazon. 2017. Amazon S3. https://aws.amazon.com/s3/. (2017). Accessed: 2017-06-08.

[2] Tim Bass. 2000. Intrusion Detection Systems and Multisensor Data Fusion. *Commun. ACM* 43, 4 (April 2000), 95–105.

[3] A. L. Buczak and E. Guven. 2016. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys Tutorials* 18, 2 (Secondquarter 2016), 1153–1176. DOI: http://dx.doi.org/10.1109/COMST.2015.2494502

[4] George Casella. 1985. An Introduction to Empirical Bayes Data Analysis. *The American Statistician* 39, 2 (May 1985), 83–87.

[5] Paul Cichonski, Tom Millar, Tim Grance, and Karen Scarfone. 2012. Computer Security Incident Handling Guide. *NIST Special Publication 800-61 Rev 2* (August 2012).

[6] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. 2011. Flexible, high performance convolutional neural networks for image classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[7] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. 2014. Scalable object detection using deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2147–2154.

[8] Matthew Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *ACM Conference on Computer and Communications Security (CCS)*. Tokyo, Japan.

[9] Leo Grady. 2006. Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 28, 11 (2006), 1768–1783.

[10] Guofei Gu, Alvaro A. Cardenas, and Wenke Lee. 2008. Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems. In *ACM Symposium on InformAction, Computer and Communications Security (ASIACCS)*. Tokyo, Japan.

[11] Taher Haveliwala. 1999. *Efficient computation of PageRank*. Technical Report. Stanford.

[12] Fred Hohman, Nathan Hodas, and Duen Horng Chau. 2017. ShapeShop: Towards Understanding Deep Learning Representations via Interactive Experimentation. In *2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1694–1699.

[13] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, and Kurt Keutzer. 2016. FireCaffe: near-linear acceleration of deep neural network training on compute clusters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2592–2600.

[14] Klaus Julisch. 2003. Clustering Intrusion Detection Alarms to Support Root Cause Analysis. *ACM Transactions on Information Systems Security* 6, 4 (Nov. 2003), 443–471. DOI: http://dx.doi.org/10.1145/950191.950192

[15] Minsuk Kahng, Pierre Andrews, Aditya Kalro, and Duen Horng Chau. 2017. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. *arXiv preprint arXiv:1704.01942* (2017).

[16] Kaspersky Lab. 2015. Denial of Service: How Businesses Evaluate the Threat of DDoS Attacks. *IT Security Risks Special Report Series* (September 2015), 7.

[17] MIT Lincoln Laboratory. 2000. DARPA Intrusion Detection Scenario Specific Data Sets. https://www.ll.mit.edu/ideval/data/2000data.html. (2000).

[18] Ulf Lindqvist and Phillip A. Porras. 1999. Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST). In *International Symposium on Security and Privacy (SP)*. Oakland, CA.

[19] David Miller, Shon Harris, Allen Harper, Stephen VanDyke, and Chris Blask. 2010. *Security Information and Event Management (SIEM) Implementation* (1st ed.). McGraw Hill Education.

[20] P. O'Kane, S. Sezer, and K. McLaughlin. 2011. Obfuscation: The Hidden Malware. (May 2011), 41–47.

[21] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. 2004. Automatic multimedia cross-modal correlation discovery. In *ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 653–658.

[22] Roberto Perdisci, Giorgio Giacinto, and Fabio Roli. 2006. Alarm Clustering for Intrusion Detection Systems in Computer Networks. *Engineering Applied Artificial Intelligence* 19, 4 (June 2006), 429–438. DOI: http://dx.doi.org/10.1016/j.engappai.2006.01.003

[23] Phillip A. Porras, Martin W. Fong, and Alfonso Valdes. 2002. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*. Zurich, Switzerland, 95–114.

[24] Alireza Sadighian, José M. Fernandez, Antoine Lemay, and Saman T. Zargar. 2014. *ONTIDS: A Highly Flexible Context-Aware and Ontology-Based Alert Correlation Framework*. Springer International Publishing, Cham, 161–177. DOI: http://dx.doi.org/10.1007/978-3-319-05302-8_10

[25] A. Sadighian, S. T. Zargar, J. M. Fernandez, and A. Lemay. 2013. Semantic-based context-aware alert fusion for distributed Intrusion Detection Systems. In *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*. 1–6. DOI: http://dx.doi.org/10.1109/CRiSIS.2013.6766352

[26] Mallikarjun Shankar, Nageswara Rao, and Stephen Batsell. 2003. Fusing Intrusion Data for Detection and Containment. In *IEEE Military Communications Conference (MILCOM)*. Ontario, Canada.

[27] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. 2005. Relevance search and anomaly detection in bipartite graphs. *ACM SIGKDD Explorations Newsletter* 7, 2 (2005), 48–55.

[28] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *International Conference on Data Mining (ICDM)*. IEEE Computer Society, Washington, DC, USA, 613–622. DOI: http://dx.doi.org/10.1109/ICDM.2006.70

[29] Alfonso Valdes and Keith Skinner. 2001. Probabilistic Alert Correlation. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*. Davis, CA.

[30] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. 2004. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE*

Transactions on Dependable Secure Computing 1, 3 (July 2004), 146–169. DOI: http://dx.doi.org/10.1109/TDSC.2004.21

[31] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. 2004. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Transactions on Dependable Secure Computing* 1, 3 (July 2004), 146–169. DOI: http://dx.doi.org/10.1109/TDSC.2004.21

[32] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. 2011. Improving the speed of neural networks on CPUs. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Vol. 1. 4.

[33] Jouni Viinikka, Hervé Debar, Ludovic Mé, and Renaud Séguier. 2006. Time Series Modeling for IDS Alert Management. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*. ACM, New York, NY, USA, 102–113. DOI: http://dx.doi.org/10.1145/1128817.1128835

[34] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. Beehive: Large-Scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks. In *Annual Computer Security Applications Conference (ACSAC)*.

[35] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *USENIX conference on Networked Systems Design and Implementation (NSDI)*.

[36] Chao Zhang, Shan Jiang, Yucheng Chen, Yidan Sun, and Jiawei Han. 2015. Fast inbound top-k query for random walk with restart. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Springer, 608–624.

[37] Ziming Zhao, Gail-Joon Ahn, Hongxin Hu, and Deepinder Mahi. 2012. SocialImpact: Systematic Analysis of Underground Social Dynamics. In *European Symposium on Research in Computer Security (ESORICS)*.