Spotlight: Malware Lead Generation at Scale

Fabian Kaczmarczyck
kaczmarczyck@google.com
Google

Bernhard Grill
bgrill@google.com
Google

Luca Invernizzi
invernizzi@google.com
Google

Jennifer Pullman
jpullman@google.com
Google

Cecilia M. Procopiuc
mpro@google.com
Google

David Tao
dtao@google.com
Google

Borbala Benko
bbenko@google.com
Google

Elie Bursztein
elieb@google.com
Google

ABSTRACT

Malware is one of the key threats to online security today, with applications ranging from phishing mailers to ransomware and trojans. Due to the sheer size and variety of the malware threat, it is impractical to combat it as a whole. Instead, governments and companies have instituted teams dedicated to identifying, prioritizing, and removing specific malware families that directly affect their population or business model. The identification and prioritization of the most disconcerting malware families (known as *malware hunting*) is a time-consuming activity, accounting for more than 20% of the work hours of a typical threat intelligence researcher, according to our survey. To save this precious resource and amplify the team's impact on users' online safety we present Spotlight, a large-scale malware lead-generation framework. Spotlight first sifts through a large malware data set to remove known malware families, based on first and third-party threat intelligence. It then clusters the remaining malware into potentially-undiscovered families, and prioritizes them for further investigation using a score based on their potential business impact.

We evaluate Spotlight on 67M malware samples, to show that it can produce top-priority clusters with over 99% purity (i.e., homogeneity), which is higher than simpler approaches and prior work. To showcase Spotlight's effectiveness, we apply it to ad-fraud malware hunting on real-world data. Using Spotlight's output, threat intelligence researchers were able to quickly identify three large botnets that perform ad fraud.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**.

KEYWORDS

Malware hunting, Malware classification, Malware clustering, Malware prioritization

ACM Reference Format:
Fabian Kaczmarczyck, Bernhard Grill, Luca Invernizzi, Jennifer Pullman, Cecilia M. Procopiuc, David Tao, Borbala Benko, and Elie Bursztein. 2020. Spotlight: Malware Lead Generation at Scale . In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3427228.3427273

1 INTRODUCTION

As the sophistication of anti-malware defenses keeps increasing with each passing year, malware authors have been improving their malware generation pipelines to reduce their detection surface. To do so, they have introduced techniques that make every binary unique, such as obfuscation, packing, polymorphism, and metamorphism. These techniques have become so commonplace in the malware industry that over one billion unique malware samples have been identified in 2019 by AV-Test [2], an increase of over 1500% since 2011.

The sheer number of malware samples being produced every day has made the threat researchers' choice of which sample to investigate, the *malware hunter dilemma*, even more crucial. Researchers face a trade-off as they invest a scarce resource, their limited time, for the potential payoff of identifying a malware family that is (1) novel and (2) relevant to their team's mission. Unfortunately, this choice is often difficult as researchers have limited a priori information about the malware they face, and a sample chosen at random will likely be a variation of known malware families and thus ultimately of little novelty. Investigating such a sample is a bad payoff for their time as it produces little tangible benefit to their team's mission. On the other hand, finding novel malware is not sufficient; its investigation also needs to have a significant impact on their stakeholders. For example, a bank's threat intelligence team may only be interested in malware that targets the bank's clients; an advertising company focuses on ad fraud; a cloud storage company wants to root out instances of ransomware, etc. Relying solely on AV labels for malware hunting is challenging as they are often inconsistent, generic and there is no single naming convention for malware specimens. [14]

This paper aims to improve a researcher's odds against the malware hunter dilemma. Specifically, we propose Spotlight, a framework to identify and prioritize major unknown threats that affect a business so to maximize the researcher's return on investment. Spotlight is designed to operate at scale, processing tens of millions of malware samples using limited computing resources. Spotlight first leverages a continuously-trained machine learning model to sift through malware to separate samples that belong to known families from malware that does not. The unknown malware samples are further clustered based on their semantic similarity to identify potential families, and each cluster is then prioritized with an application-specific scorer to quantify the cost to the business and thereby identify the top clusters that should yield the best payoff from a manual investigation.

An application-specific scorer is inherently dependent on the team's mission; for example, given clusters of potentially-damaging malware, a bank may want to dynamically execute a few samples per cluster to see whether they target the bank's online presence or the honeypot's user credentials for the bank, whereas a botnet fighting team would prioritize clusters which have the largest victim base. In this paper, we present a use case where Spotlight is configured to surface ad fraud, using a scorer that dynamically executes samples to collect evidence of ad fraud through network traffic signatures and honeypot signals. To illustrate Spotlight's usefulness in our use case, we evaluate it on a data set of 67M malware samples to identify three ad fraud botnets. To our knowledge, this data set is the largest to date ever to be used on a malware family classification or clustering task, exceeding prior work [13] by one order of magnitude.

Our contributions are summarized as the following:

- We present Spotlight, a system that identifies and prioritizes unknown malware families that are relevant to a threat-intelligence team's mission, thus making malware hunting less time consuming. Similar to active learning, Spotlight' feedback loop uses investigation results to improve its results.
- We evaluate Spotlight on a 67M malware data set to show that it outperforms previous work on malware family classifiers, and it identifies the malware family as unknown with a 0.9994 precision and 0.992 recall.
- We apply Spotlight to a malware hunting use case for an ad company to identify three large ad fraud botnets, one of which were not detected on VirusTotal.

2 MOTIVATION

Malware hunting is known to be a time-intensive yet critical activity in the cybersecurity industry. This topic is often debated in security news sources (e.g., [8, 9, 11]), and companies are offering it as a business-to-business service (e.g., CrowdStrike [6], Carbon Black [5]).

We conducted a qualitative survey in May 2020 to verify that malware hunting is indeed essential and time consuming. We submitted this survey to 37 malware experts, working in eight independent teams of a single top-100 company per market capitalization. Each of these teams work on a specific facet of threat intelligence, ranging from securing browsers, to mobile devices, ads, and cloud computing.

The responses for this survey, shown in Table 1 and 2, suggest that malware hunting is a critical activity (very or extremely important for 90% of the respondents) that often incurs a significant time investment (>20% of work time for 70% of respondents). We

Table 1: Survey results for question 1: *"How important is lead generation (i.e., malware hunting) in your current position?"*

Response	Response count	Response ratio
Not important (1/5)	1	5%
Mildly important (2/5)	0	0%
Important (3/5)	1	5%
Very important (4/5)	8	**38%**
Extremely important (5/5)	11	**52%**
Survey not completed	18	N/A

Table 2: Survey results for question 2: *"How much of your working time did you spend on lead generation related tasks during the last 6 months (Dec 2019-May 2020)?"*

Response	Response count	Response ratio
$0\% \le t < 20\%$	5	24%
$20\% \le t < 40\%$	10	48%
$40\% \le t < 60\%$	3	14%
$60\% \le t < 80\%$	2	10%
$80\% \le t \le 100\%$	1	5%
Survey not completed	18	N/A

note that we only claim this survey to be qualitative due to its sample size, non-response bias, and sampling bias. Nonetheless, this result is in agreement with the general sentiment of the industry, and further highlights the need for better tooling to automate the hunting.

3 SYSTEM OVERVIEW

We now describe Spotlight's overall structure, as shown in Figure 1, to then provide additional details in the following sections.

Goal. Spotlight's goal is to make malware hunting more efficient by automatically identifying and prioritizing malware families that have yet to be reverse engineered (*unknown* malware) *and* that are in scope with the threat intelligence team mission, be it to defend a bank, online advertising, a government entity, or other business. We aim to achieve this goal without relying solely on AV labels due to their inconsistencies, generic names and lack of a common naming convention [14].

Processing steps. Spotlight's input is exclusively malware. We note that Spotlight's goal is *not* to distinguish malware from benign binaries; instead, it is meant to run on the cumulative knowledge that other malware-analysis systems have collected about each malware sample. Known malware may be interesting, e.g., to assess its impact by reverse engineering it.

Each malware sample is first encoded in a vector of input features, as described in Section 3.1. It is then processed by a deep-learning classifier that labels the sample as belonging to one of the malware families that have been previously studied by the team, or marks it with the "unknown malware" label (details in Section 3.2). In our operational setting, the samples attributed to a known family will be passed on to malware family tracking systems, which provide statistics on the prevalence of each malware family based on telemetry not in scope for this paper. The main output of this first step is

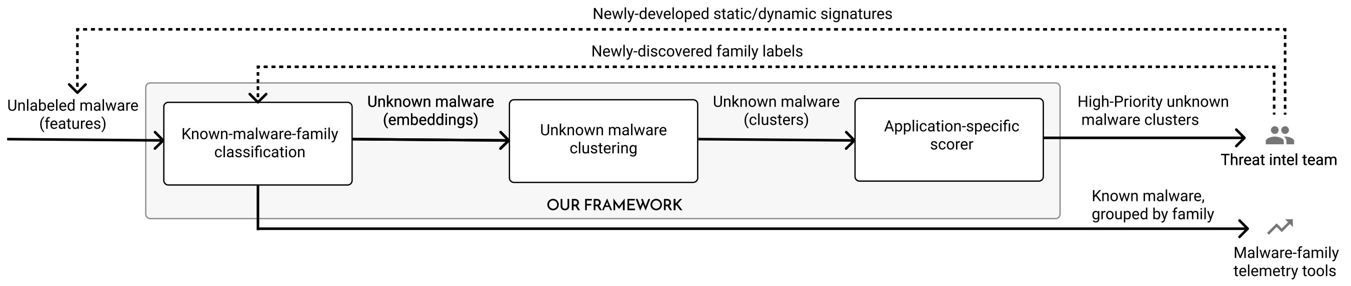
Newly-developed static/dynamic signatures
Newly-discovered family labels
Unlabeled malware (features)
Known-malware-family classification
Unknown malware (embeddings)
Unknown malware clustering
Unknown malware (clusters)
Application-specific scorer
High-Priority unknown malware clusters
Threat intel team
OUR FRAMEWORK
Known malware, grouped by family
Malware-family telemetry tools

Figure 1: Overview of our framework.

a malware embedding, computed by the classifier, that represents the malware in a compact form, reducing the large initial feature space to 32 floating point numbers.

Samples classified as *unknown malware* are passed on to a clustering component, which groups malware into clusters based on the embedding vectors generated by the classifier (see Section 3.3). Finally, an application-specific scorer takes in all the clusters and ranks them by giving them a priority that reflects the threat intelligence team's mission (details in Section 3.4).

Finally, a threat intelligence researcher manually investigates a subset of the top-priority clusters and takes appropriate action. The researcher also optionally develops new static or dynamic signatures to detect the malware, which are added to the input features described in the next section. The researcher can also attribute a name to a new malware family, in which case Spotlight adds a corresponding output to the known-malware-family classifier.

Hence, Spotlight removes samples from this now known malware family in the classification step on the next run. The classifier drives the reduction of the number of unknown malware samples. Also, more relevant features can improve embedding representations and allow better clustering results on later iterations.

3.1 Input Features

Spotlight hunts for novel malware families by leveraging the accumulated intelligence extracted by automated malware detectors, such as malware analysis sandboxes, or antivirus software.

Specifically, each sample is represented by a sparse vector of metadata extracted by those detectors, including 1) features from static analysis, 2) features from dynamic analysis, and 3) antivirus verdicts. Combining those 3 feature sets allows us to leverage the individual benefits of each approach. We obtain antivirus verdicts through VirusTotal to simulate an operational setting in which the threat intelligence team has access to antivirus verdicts through service contracts with antivirus companies or by running static scanners and honeypots. The raw labels obtained from antivirus engines, before the preprocessing described below, look like this: `Spyware.Banker.Dridex`, `Packed.Generic.525`.

We note that some of the features we use come from non-public analysis platforms, whose implementation details we cannot share as this would give a tangible advantage to malware authors. In the spirit of open science, we also run Spotlight exclusively on publicly available VirusTotal labels in section 4.6, so that our results can be replicated. We believe that including the performance of the full framework (including the proprietary features) gives a more comprehensive view of what is achievable with this system.

We further refine the 212,361 (211,246 + 1,115) one-hot-encoded categorical features by normalizing their names through standard text preprocessing (tokenization, lowercasing, removal of common delimiters, deduplication) and discarding overly prevalent (over 20% of the data set matches the feature) or extremely rare features (10^{-5}% of the data set - in our case, a max of 100 hits for the feature). This shrinks our sparse feature space nearly in half, resulting in 121,947 final input features.

3.2 Known-Family Classification

As shown in Figure 1, Spotlight first processes all input malware through a deep-learning classifier to identify malware families that have already been identified by the threat intelligence team. The output of this step is twofold: known malware gets attributed to the correct malware family, whereas the rest of the malware is associated with a compact embedding comprising 32 floats, which will be passed on to the clustering stage.

Motivation. This compact embedding has the dual benefit of 1) reducing the compute and memory requirements of the subsequent clustering, as the points to be clustered are of modest dimensionality (see Section 4.4), and 2) substantially improving the quality of Spotlight's output clusters (see Section 4.7). Intuitively, this family classifier acts both as a filter to sift out malware belonging to known families, and as a preprocessor that discards irrelevant input features, and projects the relevant ones into an embedding space that places semantically-similar samples (that is, belonging to the same family) close to each other and away from other families. This learned separation eases the job of the subsequent clustering step as testified by the boost in performance in our experiments in Section 4.7. Effectively, this classification step gives threat researchers a way to train Spotlight to semantically recognize what they consider to be malware belonging to different families, and complete the training feedback loop of the pipeline.

Structure. We detail the internal structure of the malware family classifier, a multilayer perceptron, in Figure 2. Specifically, the sparse one-hot-encoded categorical inputs are first bucketed via hash buckets to reduce their dimensions and increase density, as per standard practice when using sparse inputs to deep-learning classifiers. Then, they pass through a series of dense layers to finally produce the family classification. As part of this process, an

Table 3: Overview of input feature categories. Note that a single malware can match many features, thus ratios do not add up to 100%.

		Internally authored signatures		VirusTotal signatures	
Feature categories	**Description**	Feature count (ratio)	Hits on malware data set (ratio)	Feature count (ratio)	Hits on malware data set (ratio)
Adware	Ad fraud, such as click fraud	5 (0.4%)	198,701 (0.3%)	4,263 (2.6%)	6,911,927 (1.5%)
Crypto	Informational crypto-detection signatures, e.g. AES s-boxes, public and private keys	57 (5.1%)	7,895,063 (12.5%)	0 (0%)	0 (0%)
Dropper	Dropper or backdoor capabilities	18 (1.6%)	302,095 (0.5%)	14,165 (6.7%)	43,417,909 (6.5%)
Exploit	Known exploits, shellcode and CVEs	29 (2.6%)	568,034 (0.9%)	195 (0.1%)	709,677 (0.2%)
Family	Specific malware families, e.g. WannaCry, Mirai	241 (21.7%)	3,459,410 (5.5%)	7,120 (4.4%)	43,891,534 (9.3%)
Generic	Generic malicious behavior and heuristics	0 (0%)	0 (0%)	50,209 (31.2%)	169,884,389 (36.2%)
Info	Purely informational features, e.g. file type, compiler used	105 (9.4%)	14,184,790 (22.5%)	0 (0%)	0 (0%)
Info stealer	Info stealing behavior, e.g. banking trojans or keyloggers	19 (1.7%)	3,234,751 (5.1%)	3,644 (1.7%)	12,597,007 (1.9%)
Meta	Meta-signatures, i.e. signatures derived from the presence of groups of other signatures	151 (13.6%)	2,500,791 (4%)	0 (0%)	0 (0%)
Miner	Cryptocurrency mining detection	19 (1.7%)	324,760 (0.5%)	910 (0.6%)	7,087,959 (1.5%)
Packer	Packers, obfuscators, cryptors	26 (2.3%)	3,297,566 (5.2%)	955 (0.6%)	3,042,416 (0.6%)
Ransom	Ransomware routines	8 (0.7%)	287,271 (0.5%)	3,557 (2.2%)	22,743,700 (4.8%)
Suspicious	Suspicious but not malicious behaviour, e.g. VM and debugger checks, disable OS features	36 (3.2%)	12,230,903 (19.4%)	794 (0.4%)	3,824,586 (0.6%)
Trojan	Malicious software disguising as legitimate	30 (2.7%)	63,653 (0.1%)	62,818 (39.0%)	170,446,401 (36.3%)
Unwanted	PUP (potentially unwanted programs), which are not malicious per se	128 (11.5%)	2,661,163 (4.2%)	632 (0.4%)	961,310 (0.2%)
Worm	Malware spreading via worm-like techniques	0 (0%)	0 (0%)	7,448 (4.6%)	32,111,647 (6.8%)
Other	Any signatures not fitting in any other category	243 (21.8%)	11,943,016 (18.9%)	54,536 (25.8%)	153,717,390 (22.9%)
Total		1,115	63,151,967	211,246	671,347,852

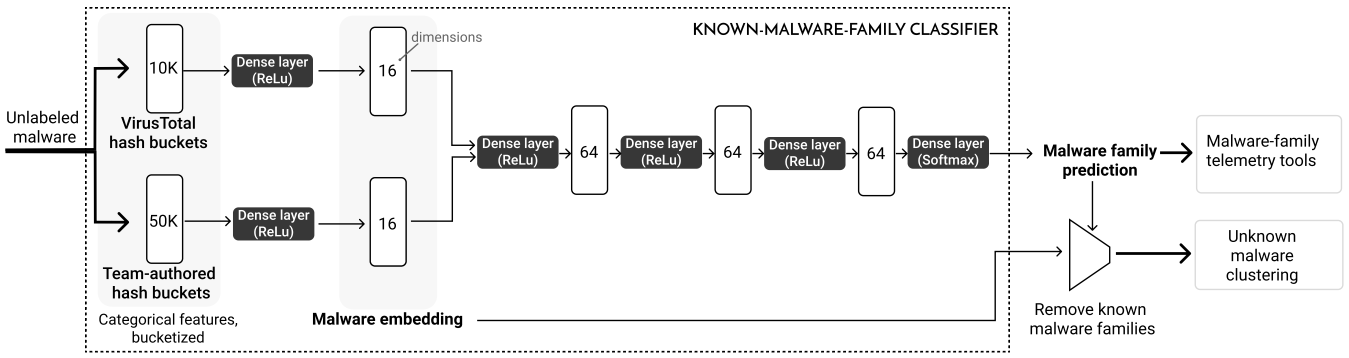
KNOWN-MALWARE-FAMILY CLASSIFIER
Unlabeled malware
10K
VirusTotal hash buckets
50K
Team-authored hash buckets
Categorical features, bucketized
Dense layer (ReLu)
dimensions
16
Malware embedding
64
Dense layer (Softmax)
Malware family prediction
Malware-family telemetry tools
Remove known malware families
Unknown malware clustering

Figure 2: Architecture of the known-malware-family classifier

embedding is computed at the most dimension-constrained section of the neural network.

The output of the classifier comprises $n_f + 1$ floats corresponding to the likelihood of the malware being part of one of the n_f families, or of an “unknown malware” category.

This classifier, implemented in TensorFlow [1], is trained in a fully-supervised fashion on a training set, as detailed in Section 4.1. To account for the class imbalance among the malware families, we use a weighted cross-entropy loss where the weights are inversely proportional to the family prevalence in the data set. By doing so, the classifier will be penalized more for a classification error on a malware sample belonging to a rarer malware family. We evaluate the performance of this classifier in Section 4.3.3

The various parameters of the classifier have been chosen via hyperparameter tuning on a similar data set. The data set for hyperparameter tuning was collected three months prior to the one we describe in our experiment. Details of the tuning are available in Section 4.3.

3.3 Unknown Malware Clustering

As shown in Figure 1, Spotlight performs clustering on all embeddings of malware binaries that have been classified as unknown. Specifically, we perform a highly parallel version of hierarchical agglomerative clustering [16], which has been recently given theoretical support on its ability to produce quality clusters [15]. We use Euclidean distance in the embedding space.

This clustering algorithm requires the setting of a parameter, ε, a distance threshold dictating when to cut the tree of clusters (i.e., when to stop creating smaller and smaller subclusters). As this choice depends on the semantics of what constitutes a malware family, ε was set manually by the threat intelligence team (details in Section 4.3.1). Higher values of ε generate fewer, larger clusters, whereas smaller values yield more, smaller clusters.

The astute reader will note that we could have applied clustering on the input malware sparse features directly. We have indeed attempted to do so, but the cluster quality was subpar due to the sparsity and noise of the features. Furthermore, the operational requirements to run our clustering tool on the larger input data were more demanding. We provide an empirical analysis of this option in Section 4.7.

Preclustering. To speed up clustering, as it is the most resource-intensive component of Spotlight (see details in Section 4.4), we first precluster embeddings which are very similar to each other, i.e. remove near duplicates. These embeddings typically consist of near duplicates due to polymorphic malware. To precluster them, we reduce the numerical precision of the embeddings by rounding each coordinate to 4 digits in its decimal representation. All samples that are represented by the same quantized embedding are then marked as belonging to the same precluster. These preclusters are then fed to the clustering system.

3.4 Prioritization

As shown in Figure 1, the clusters are then scored by an application-specific scorer which ranks clusters according to their business impact, and then passes the top ranking ones on to threat researchers for analysis. Those application-specific scorers are highly use-case dependant and have to be customized according to the specific scenario.

This component's inner workings depend on what the threat intelligence team is looking for. In our ad fraud context, for example, the scorer selects samples in each cluster and executes them in a honeypot that attempts to elicit the ad fraud behavior from the malware by simulating a user browsing the internet. In a banking application, instead, a reasonable approach is having the honeypot synthetic user sign into the bank site, in order to elicit a potential credential harvesting component of the malware under analysis. If dynamic analysis is to costly, one can also use static scorers only instead, e.g. searching for suspicious strings within the binaries. If the application is instead to take down the largest botnets by victim population size, a simple size estimator based on DNS lookup logs would be appropriate as scorer.

Ultimately, the application specific scorers act as a scoring function to sort clusters based on their relevance. Therefore, their implementation strongly depend on the specific use-case and goal, and have to be adopted accordingly.

In this paper, we use two scorers: a simple botnet-size scorer, meant to discover the larger malware families by simply giving higher scores to larger clusters, and an ad fraud scorer, described below and put to the test in a case study in Section 5.

Ad fraud scorer. This sub-section describes an application-specific scorer we used to hunt for ad-fraud malware. This is one way to do it, however one can use different techniques to hunt for ad-fraud, e.g. only rely on static analysis, target very specific forms of ad-fraud, etc.

Our scorer operates as follows. First it executes Yara signatures on unpacked samples. The Yara signatures check for suspicious strings, URLs and API calls. Afterwards, it executes every sample having at least 1 Yara signature hit in a high-interaction Windows honeypot and we monitor its network behavior to find evidence of ad fraud.

Afterwards we compute a *fraud score* for each sample, which represents the share of signals triggered by this sample during static or dynamic analysis, e.g. 2 hits and 8 negative signals would result in a fraud score of 0.2 (2 / 10). The clusters' fraud score is the average of the samples' fraud score in the same cluster - the higher this score, the more relevant the cluster and the higher its rank for malware analysts to investigate.

This is just one example how to hunt for ad-fraud malware. If dynamic analysis is too costly, one can also replace it by static analysis only, e.g. searching for suspicious strings or library calls.

3.5 Investigation and Feedback

Finally, threat intelligence researchers investigate the top ranking clusters provided by the scorer. The methodology that the researchers use to investigate clusters is their standard operating procedure, unchanged with the arrival of Spotlight, and it is not in scope of this paper. For each inspected cluster, researchers can optionally create new signatures and, if a novel family has been found, assign a new label to it. These new signatures and labels are taken into account in future iterations of the feedback loop.

4 EVALUATION

We claim that Spotlight can produce quality clusters of unknown malware, where "quality" indicates that the clusters have have high purity (as defined in Section 4.2) and business impact. These two factors are our key metrics as, when combined, they can scale a threat intelligence team impact. On the other hand, impure clusters would effectively be as useless as low-impact clusters, as no generalized conclusion about a cluster can be drawn by analyzing a few samples. While high completeness (meaning, a malware family is not split into too many clusters) is desirable, it is not critical to the mission. In this section, we show how Spotlight performs on a large (67M) data set of malware samples, and how its performance and computational cost is affected in a variety of deployment contexts. Then, we configure Spotlight to hunt for ad fraud malware, and we describe a case study where we identify three large ad fraud botnets. Table 4 provides an overview of the experiments we conducted.

4.1 Data sets

We produce a train/test data set pair of malware samples, which for brevity we call our *training data set* of 11.8M malware samples. 1.8M of which included a family label, from 26 malware families. We assigned to the remaining 10M samples the *unknown* family label. The set is split 80/20 for training and testing. The malware samples were collected between March 2018 and March 2020.

The distribution of malware families in the training data set are outlined in Table 5. We note that no benign samples are part of this

Table 4: Experiments overview.

Experiment	Data set		Section
	Training	Hunting	
Cluster quality and hyperparameter sensitivity	✓		4.3.1
Spotlight's workflow effectiveness	✓		4.3.2
Family classification performance	✓		4.3.3
Resource consumption	✓	✓	4.4
Manually investigating cluster quality	✓	✓	4.5
Removing proprietary malware features	✓	✓	4.6
Benefit provided by the malware embedding step		✓	4.7
Case study on hunting real-world ad fraud botnets	✓	✓	5

Table 5: Malware families used in the training set.

Malware family	Description	Training set ratio
Pony	Dropper and info stealer	28.5%
Shylock	Banking trojan	25.6%
Asprox	Dropper and spambot	17.9%
Lethic	Dropper and spambot	9.4%
Tinba	Banking trojan	4.6%
WannaCry	Ransomware	3.9%
Simda	Dropper and info stealer	3.2%
Nymaim	Ransomware and dropper	2.2%
Buterat	Trojan	2.0%
PlugX	Backdoor and info stealer	0.8%
Tofsee	Spambot	0.7%
Necurs	Dropper	0.6%
QakBot	Banking trojan and dropper	0.4%
DreamBot	Banking trojan and dropper	0.10%
Shifu	Banking trojan and dropper	0.05%
Sinowal	Banking trojan	0.04%
Geodo	Banking trojan	0.03%
EvilGrab	Credential harvesting	0.02%
Kronos	Banking trojan	0.02%
Elise	Dropper	0.01%
WannaCryWorm	Ransomware (worm variant)	0.01%
Cobian	Dropper	0.01%
Corebot	Banking trojan	0.01%
Cutwail	Dropper and spambot	0.003%
Slave	Banking trojan	0.001%
Total		100%

(or any) data sets, as Spotlight operates exclusively on malware. All malware in our data sets targets Windows. The malware family labels are based on a combination of in-house and external [17] signatures.

The training data set is built to mimic the distribution of labeled/unlabeled samples we find in practice. The same is valid for family distributions.

On the same day, we also collect a larger *hunting data set*, comprising 67M malware samples.

4.2 Performance Metrics

As the final output that Spotlight produces is a ranked list of malware clusters, we choose to measure Spotlight's effectiveness with well-established metrics: homogeneity and completeness, which combine to form the single metric v-measure [18] as their harmonic mean. Spotlight's goal is to provide high-purity (i.e., high homogeneity) top-ranked clusters. On the other hand, high completeness is a secondary objective, as a malware family can be broken into a reasonable number of clusters without overly affecting a threat intelligence researcher's job, especially because the hierarchical nature of our clustering helps tie these related clusters together.

4.3 Pipeline Performance

In this section, we conduct a series of experiments to evaluate Spotlight's effectiveness in producing high-quality clusters, and its sensitivity to hyperparameter variations.

In these controlled experiments, we use the training data set and exclude all unlabeled malware. This allows us to precisely measure Spotlight's performance, as we exclude the malware for which we do not have a family attribution. Instead, in each experiment we select some families and strip their labels from the corresponding samples, to later check if Spotlight correctly grouped this unlabeled malware into disjoint clusters. This provides ground truth to evaluate cluster quality.

4.3.1 Embedding quality and hyperparameter sensitivity. In this first experiment we evaluate Spotlight's performance when two key parameters change: the number of labeled families, and ε. This allows us to see how Spotlight's output is affected by the manual choice of its hyperparameter ε (see Section 3.2), and whether the pipeline benefits from being aware of more malware families during its training, thus potentially producing more expressive embeddings.

Experimental setting. To perform this experiment, we randomly choose half of the 26 malware families for our training set and the remaining half for the test set. We then create multiple test configurations, each of which picks a value for $\varepsilon \in [0.03, 0.13, 0.3, 3.0]$ and a number of known malware families $\widetilde{n}_f \in [1, 2, ..., 13]$ (13 being half of the families), for a total of 52 $(4 \cdot 13)$ configurations.

We train and test each configuration by setting the ε value and revealing to the classifier only the first n_f family labels, whereas the samples for the remaining $13 - \widetilde{n}_f$ are labeled as *unknown* malware. In doing so, we can measure how Spotlight performance varies when the number of labeled malware families increases, which is expected when the threat intelligence team identifies more families.

Embedding quality. We show the result of this experiment in Figure 3 and Figure 4. We note that Spotlight produces high-purity clusters when having access to the label of five or more families. We stress that the families that the classifier has seen during the training phase are not the ones that the classifier is processing during the evaluation phase, making this the worst-case scenario to evaluate the pipeline performance. Having high-purity clusters is fundamental for threat intelligence researchers, as it allows them to only investigate one sample of a cluster in order to draw conclusions for the behavior of all samples in the cluster.

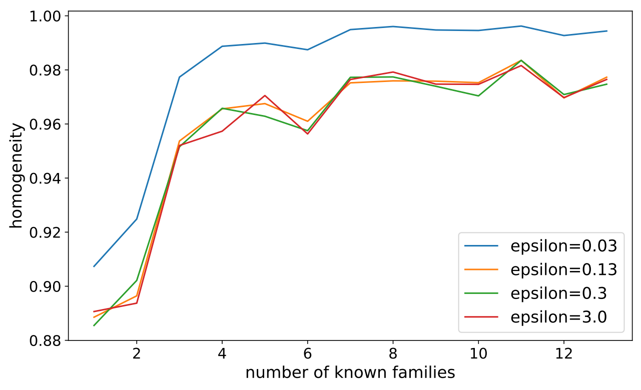

Figure 3: Clustering homogeneity for different ε, given more or fewer training labels

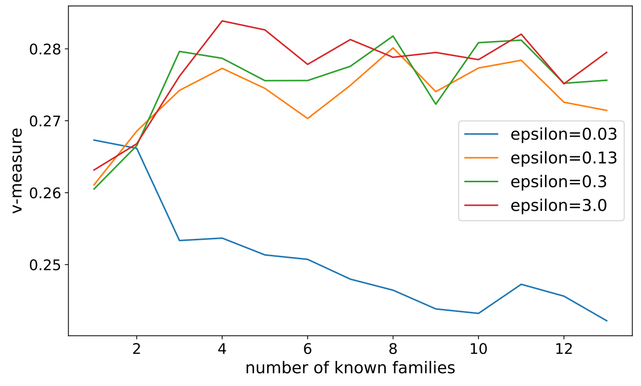

Figure 4: Clustering v-measure for different ε, given more or fewer training labels

The scale of this conclusion is instead dictated by completeness (or v-measure, which is a combination of completeness and homogeneity), which measures the fragmentation of a family into multiple clusters. Whereas low values of completeness are undesirable since they indicate that each sample is clustered by itself, mid to low values of completeness (0.1-0.5) are acceptable as they indicate that a family is split across a reasonable number of clusters (tens to hundreds). This is expected, as different strains and evolutions of the same malware families are clustered separately and allow researchers to quickly identify variants.

As an example, we show in Figure 5 the size distribution of clusters that contain a randomly-picked malware family, Asprox. The sizes of the top 10 clusters range from 947 to 693, and cumulatively they account for 12% of the total samples in the family. Asprox is a long-running botnet, first detected in 2007, and encompasses many variants. In practise, how researchers deal with the tail of clusters depends on the application. Investigating the top cluster in one iteration might help correctly classifying or clustering samples in later iterations of the pipeline.

Hyperparameter sensitivity. 's performance is fairly consistent when varying ε. This indicates that the clustering distance

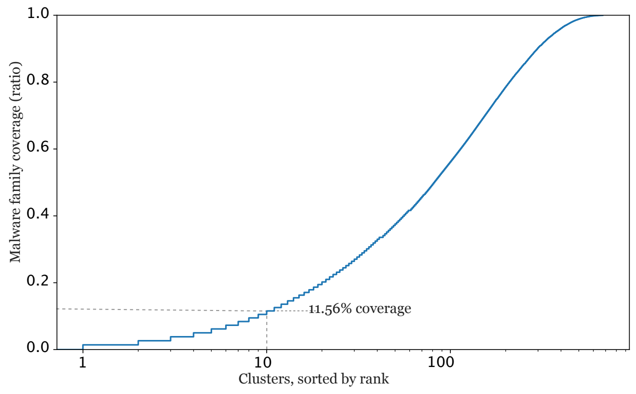

Figure 5: Cluster size distribution for the Asprox family.

among families is much higher than ε, and thus this parameter does not need to be manually set with high accuracy.

The hyperparameters used in the classification model (such as the number and size of neural network layers, dropout) have been tuned on a similar data set source three months prior to this experiment and have been kept constant across this work. At that time, our team also chose to set ε to be 10^{-4} (see Section 3.3), as it creates pure clusters that encompass polymorphic malware belonging to the same family.

4.3.2 Workflow efficiency. In this second experiment, we simulate how a threat intelligence team uses Spotlight, in order to quantify the benefit to the team's workflow.

Experimental setting. The data set we employ is the training data set (see Section 4.1), with all labels initially removed to simulate the starting state of a threat intelligence team that has yet to discover a single malware family of interest. We discard unlabeled malware, as we do not have measurable ground truth for it.

To simulate how a researcher would use Spotlight, we execute several rounds of researcher investigations, each of which produces a malware signature that is fed back into our pipeline. A pseudocode version for this procedure is described in Algorithm 1.

In the first iteration, as the researchers do not have any labeled malware family and thus cannot train the malware family classifier, we draw the classifier weights from a random distribution. We then cluster all malware samples, and rank them by size: in this scenario, our scorer is a simple counter of binaries, as the researcher is aiming to find the largest botnets.

We then simulate a researcher investigating the clusters as follows. Starting from the top cluster, the researcher is discarding clusters that have less than 95% homogeneity, and stopping at the first pure cluster. Impure clusters, as identified through our ground truth labels, are discarded since they require excessive manual investigation and would defeat the purpose of using Spotlight. Once the researcher finds the first pure cluster, they write a malware signature that identifies that malware family in the cluster. The signature is then run against all malware in the data set, and samples that match it are labeled with the family label, which we initially had removed.

Algorithm 1: *Workflow efficiency* experiment procedure pseudocode.

```
let iterationCounter = 0;
Hide all family labels from data set;
Initialize family classifier with random weights;
while iterationCounter < 10 do
    iterationCounter += 1;
    Run Spotlight to produce clusters for unknown malware;
    let pureClusterFound = false;
    for cluster in rankedClusters do
        Calculate cluster purity (Simulates investigation);
        if cluster purity is over 95% then
            pureClusterFound = true;
            break;
        end
    end
    if not pureClusterFound then
        return
    end
    Reveal family label (Simulates signature generation);
    Retrain Spotlight with the newly-discovered family label;
end
```

Subsequent iterations are performed in a similar fashion, except that now we are able to train the known-malware-family classifier using the family labels that were uncovered in previous rounds.

Results. We run a total of 10 iterations, to see which malware family is discovered. During the first iteration, where the classifier was untrained, the researcher discards the top cluster to find a 95+% pure cluster in the second position in the ranking. For all subsequent iterations, the top cluster surpasses the 95% purity threshold, so no clusters are discarded. The first two iterations remove more than half of the data set from the simulated unknown data set (see Figure 6). After 7 and 10 iterations, we respectively capture 90% and 98% of all malware samples in the data set.

Baseline. As a baseline for this experiment, we also simulate a researcher that picks malware samples at random from the data set (i.e., without Spotlight) and writes a family signature once they have analyzed T samples from the same family.

We show the results in Table 6. As expected, a hypothetical researcher that is able to generate a family signature after analyzing a single malware sample can effectively match Spotlight's performance and does not benefit from clustering. In reality, however, a researcher has to be exposed to the variations among samples of the same malware families in order to identify a common structure among them and write a precise signature. As the table shows, the more samples are needed for the researcher, the more they would benefit from Spotlight. This is also true in a team setting, as using Spotlight allows the team lead to ensure that each researcher is working on a different malware family, improving efficiency.

4.3.3 Family classification performance. In this experiment, we train the known family classifier on the training data set to obtain performance metrics about its classification performance. We find that classification precision is 0.997 or higher for 18 families, in the 0.95-0.997 range for four families (Elise - 0.99, Buterat - 0.98, EvilGrab - 0.97, Cobian - 0.95), and under 0.95 for four families (ReactorBot 0.92, Corebot 0.86, Wannacry - 0.58, Simda - 0.56). These last four families have similar traits of other families in the data set, which confuses the classifier (such as WannaCry and WannaCry-Worm). The *unknown family* class obtains 0.99 accuracy. As the latter class is predominant in the data set, the overall weighted accuracy is also over 0.996.

The recall distribution mimics the accuracy one, with most families with over 0.997 recall, except for three in the 0.98-0.997 range (Kronos, Elise, Buterat), ReactorBot at 0.92, and two families at low recall (Simba - 0.57, WannaCry - 0.59). The unknown family class has a recall of 0.99.

Table 6: Malware family signatures generated by a researcher without Spotlight (10,000 simulations per each T).

T	Family signatures		Data set coverage	
	Mean	Median	Mean	Median
1	10	10	96.5%	96.8%
2	3.62	4	70.5%	72.0%
3	1.67	2	39.9%	43.5%
4	0.68	1	17.5%	25.6%
5	0.22	0	5.8%	0%

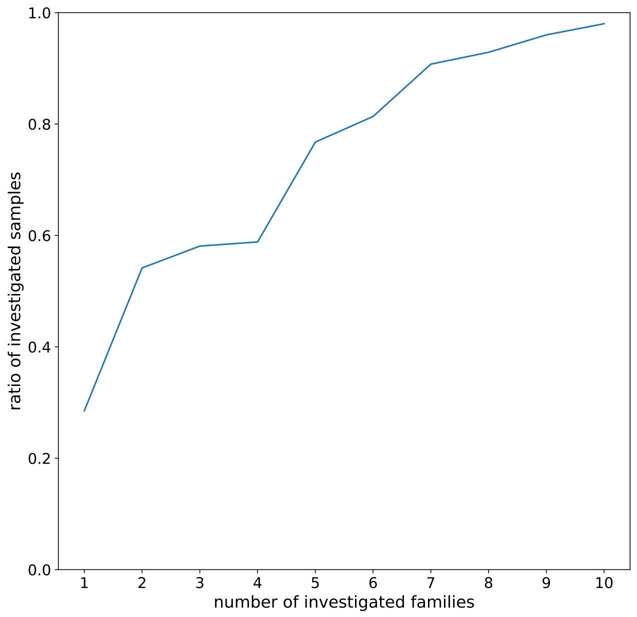

Figure 6: Ratio of malware samples associated with a malware family at each iteration.

4.4 Efficiency at Scale

We designed Spotlight to be run on a daily basis at minimum - as such, efficiency is key. After each cluster investigation by a threat intelligence researcher we also rerun the pipeline to include the newly produced signatures and labels and to automatically remove newly discovered malware families from the cluster. Frequent retraining also allows Spotlight to continuously surface emerging and evolving threats.

For these reasons, it is imperative that Spotlight's resource demands are reasonable. We measured the computational cost of running Spotlight (minus its human component) on the hunting data set. Spotlight took 34 minutes to process 67M malware samples, running on 7 CPU cores (231 CPU minutes) and 12GB of memory.

The most resource-intensive portion of Spotlight is undoubtedly the clustering, as hierarchical agglomerative clustering has a quadratic worst-case run time. We keep our system efficient through two strategies: clustering on embeddings, which yields a 3811-fold dimensionality reduction (121,947 normalized features are encoded into a 32-float embedding), and sample deduplication, which condenses the 67M malware samples into 2.8M (2,726,130) preclusters. See Section 3.3 for details. We find that without deduplication the clustering would be effectively be infeasible due to the excessive resource requirements, whereas the embedding main contribution is to improve cluster quality (see Section 4.7).

4.5 Manually investigating cluster quality

In this section we execute Spotlight on our 67M *hunting* data set, after training it on the training data set. As in the previous experiment, the ad-hoc scorer simply ranks larger clusters first, to emulate hunting for large botnets. We will discuss a case study with a different scorer in Section 5.

The largest cluster contains 872,483 malware samples. The top-10 clusters' average size is 547,164 samples. Over the whole data set, the mean and median cluster size are respectively 38 and 2. In other words, the distribution of cluster sizes comprises an initial set of large clusters and a long tail of tiny clusters, for a total of 1,561,047. This is not detrimental to Spotlight's purpose, as the researchers only investigate the top clusters, and it is expected that the data set contains some large polymorphic malware families and some one-off "families" that include a single malware sample.

Since homogeneity and v-measure cannot be evaluated in the absence of ground truth for this data set, we manually investigated the top 10 clusters, analyzing 10 random malware samples per cluster. In nine of these clusters, all malware samples belonged to the same cluster-specific family. In the tenth cluster, we found a single sample belonging to a different family than its cluster peers. This qualitative analysis indicates that the top clusters are 99% homogeneous, which is in line with what we found in our controlled experiments described above.

4.6 Limiting to VirusTotal features

In the spirit of reproducibility, we run Spotlight without any proprietary signatures, instead relying exclusively on the signatures provided by VirusTotal. This scenario allows the research community to compare other approaches to this problem to ours.

Table 7 shows how the performance changes with fewer features. The drop in classifier accuracy can be explained by some malware families that are not represented with expressive features in VirusTotal. The clustering performance seems to be only slightly affected for the important bigger clusters. With fewer features to differentiate, the number of smaller clusters decreases. The top 10 clusters for both feature sets identify the same malware families 6 times.

Table 7: Comparison of performances after excluding private features

Metric	All features	VirusTotal
Classifier accuracy	0.996	0.875
Mean per class accuracy	0.853	0.716
Precision for *unknown* class	0.9994	0.9988
Recall for *unknown* class	0.992	0.937
Investigated purity	99%	98%
Top cluster size	872,483	883,597
Top 10 cluster size average	547,164	644,871
Number of clusters	1,561,047	337,225
Median cluster size	2	6

4.7 Removing the Family Classifier

In this section, we provide a data-driven justification for our choice of introducing a family classifier in front of our clustering component. To do so, we remove the known-malware-family classifier in Figure 1, and instead opt to directly cluster the 121,947-dimensional input features representing each malware sample. Comparing the results of this experiment with the reference implementation of Spotlight will quantify the benefit that the family classifier brings to the table.

We run this modified framework on the *hunting* data set. As this approach is completely unsupervised, we drop the family labels, and we one-hot encode the input features that we pass on to the clustering component. The clustering then executes as usual.

We then manually investigate the top 10 clusters, as we did with the reference framework in Section 4.5. We find that one of the clusters contained samples belonging to at least three malware families, and seven clusters had at least one false positive. These seven clusters were also present in the reference framework output, though with higher purity. We estimate that the purity degraded from 99% to 93%, which would severely impact the usefulness of Spotlight. This data indicates that there is a statistically significant drop in cluster homogeneity when using the unsupervised approach. Additionally, we note that the average cluster size grew by 3%, and the computational advantage of clustering on compact embedding is lost, as well as the effectiveness of preclustering.

We can then conclude that the known-malware family classifier improves the quality of the results and the efficiency of the pipeline.

5 CASE STUDY: HUNTING AD FRAUD BOTNETS

To conclude the evaluation section, we present a case study in which we use Spotlight to hunt for ad fraud botnets. The application-specific scorer we adopt for this task is described in Section 3.4.

We ran this pipeline on the hunting data set (67M samples), and we investigated the top 20 clusters produced by the pipeline. Our team discovered that these 20 clusters exclusively contained various versions of three botnets. Upon further investigation, we determined that two of these botnet families did not match other botnets known to our team. Despite their novelty, further analysis enabled us to determine that the impact from these botnets to our invalid-traffic defenses was minimal.

Its presence in our malware data set was due to its binaries being flagged as suspicious by proprietary static and dynamic analysis systems.

Our analysis of random binaries from each of the top 20 clusters found that all binaries belonged to a specific version of one of the three botnets, with no clusters containing heterogeneous binaries. We did not find other botnets in those top 20 clusters.

Specifically, the three ad fraud botnets surfaced by Spotlight are:

- An ad fraud botnet using Embedded Internet Explorer to perform impression-fraud on websites. Most infections have been in South Korea and the USA.
- A traffic-selling botnet hijacking browsers' search engine settings to replace them with their websites. We have identified more than 1.3k similar looking search engine websites, which are used as redirection targets.
- A traffic-exchange botnet which also offers SEO functionality, allowing its customers to write scripts to search for a site to promote on search engines, identify keywords on the result page, and click on links. Most infections have been in China.

6 DISCUSSION

In this section we discuss a few limitations of Spotlight.

Evasion techniques. Spotlight relies on metadata from static (e.g., YARA signatures) and dynamic analysis systems (e.g., honeypots). As such, it is vulnerable to the evasion attacks that affect those systems, such as obfuscation, cloaking, or fingerprinting. If a piece of malware manages to evade all the detection systems that feed into , then our pipeline will not yield any results either. Fortunately, relies on the aggregated metadata extracted by these tools, so if some detection systems successfully identify the malware, then can process it. This happened, for example, for the botnet identified in Section 5.

Training data. As shown in Figure 3, cluster quality benefits from the availability of family labels, as they allow the family classifier to produce more expressive embeddings. Obtaining these family labels is a time-intensive activity, and it might be challenging in applications where malware samples are scarce, such as hunting for APTs. Fortunately, as shown in Section 4.3.1, just a few family labels are needed to produce an embedding that allows to produce high-quality clusters.

Application-specific scorer. As our approach relies on a scorer to rank clusters, it has to be possible to express the team objective (e.g., finding ad fraud) in code. This is not always the case: finding novel exploits used by malware, for example, is a tricky objective to express as a scorer.

Small malware families. As leverages clustering to identify malware families, it works particularly well to identify polymorphic malware with many variants. While most successful botnets fit this description perfectly, highly targeted attacks or APTs do not. In those cases, clustering does not provide the benefit of scaling the researcher impact (e.g., looking at a few samples to label a few clusters with hundreds of binaries), as the cluster size for these attacks will be extremely small. In these cases, 's only contribution is to remove botnets aimed at the general public from the data set, to leave only uncommon malware samples, which might include APTs.

Manual analysis required. Despite automating the malware prioritization process with our approach, manual analysis capacity and knowledge is still required. For example, to reverse engineer the binaries and decide whether specific malware strains are an actual threat.

7 RELATED WORK

Malware family classification. Huang and Stokes [13] has some similarity to our approach, as it proposes a fully-supervised deep-learning classifier to label malware to known families as an auxiliary task on 4.5M training and 2M test samples. These researchers find that a classifier focusing only on family accuracy achieves an accuracy of 97.06%. However, training a classifier to simultaneously learn to identify malware and to attribute it to a family raises accuracy to 99.64%, which matches our single-task classifier. We note that the goal of this prior research differs from ours, as it solely focuses on assigning a known-family label to malware, instead of hunting for novel families.

Encoding malware features. The above paper uses random projections, as introduced by Dahl et al. [7], to reduce the dimensionality of input features in an unsupervised fashion (projecting 50,000 features into 4,000). Wojnowicz et al. [20] improved on this approach by introducing randomized principal component analysis. To our knowledge, their data set (11.7M malware samples) was the largest to date used in malware classification or clustering research. On a binary malware-or-not classification task, they reduce the input feature space from 100,000 to 5,000 dimensions with minimal loss of accuracy (98,83%). In our approach, we opted to reduce our 200,000 input features with hash buckets. As we already achieve 0.9994 precision for the "unknown malware" class, we have not explored alternative approaches to hash buckets.

Malware clustering. Sebastián et al. [19] propose a tool to label malware at scale, which automatically tokenizes, deduplicates and normalizes malware labels from antivirus engines. Effectively, the processed labels can act as cluster identifiers. They demonstrate the effectiveness of their system on 10 data sets comprising 8.9M samples. This approach shares some similarities with ours as it produces malware clusters in a mostly-unsupervised fashion, though it does not focus on identifying unknown malware or rank it according to business needs. Comparing the performance of the two approaches is unfeasible due to the difference in scope of the two systems - the precision of this approach on known families ranges from 67.5% to 96.3% depending on the data set, with a recall from 46.3% to 98.8%.

Bailey et al. [3], Hu et al. [12] and Bayer et al. [4] describe semantic behavior-based clustering approaches using artifacts of static and/or dynamic analysis as features. They run their experiments on ten thousands to hundreds of thousands of samples.

Graziano et al. [10] combine dynamic and static analysis with features based on the file submission of public malware sandboxes to automatically track and identify malware development. They show-case their approach to targeted attacks and malware trying to evade detections from known malware analysis sandboxes.

Zhang et al [21] proposed an ensemble technique to combine different features from static or dynamic analysis after clustering. To make sure that all included feature sources improve the clustering quality, they selectively add them to their ensemble. Our approach instead combines different feature input sources using an embedding. All sources contribute to that embedding before clustering. Quality is controlled more implicitly, training the embeddings on a classification task.

8 CONCLUSION

In this paper, we proposed a novel framework to identify and prioritize relevant unknown threats, with the goal of saving threat intelligence researchers' time, a scarce resource. We evaluated on multiple large data sets, up to 67M malware samples, to show that the proposed approach can scale and produce top-priority clusters with high purity. To our knowledge, this is the largest data set to date to be used in a peer-reviewed malware clustering publication. To showcase 's effectiveness, we applied it to ad fraud malware hunting on real-world data. In doing so, we identified three large botnets that perform ad fraud (one doing impression fraud with an embedded browser, a traffic-selling botnet, and a traffic exchange botnet). Our experiments demonstrate that is a valuable addition to the toolkit available to a threat intelligence researcher.

ACKNOWLEDGMENTS

We would like to thank Google's Safe Browsing team for their significant signature contributions used in this work.

REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

[2] Av-Test. [n.d.]. Malware Statistics and Trends. https://www.av-test.org/en/statistics/malware.

[3] Michael Bailey, Jon Oberheide, Jon Andersen, Z Morley Mao, Farnam Jahanian, and Jose Nazario. 2007. Automated classification and analysis of internet malware. In *International Workshop on Recent Advances in Intrusion Detection*.

[4] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. 2009. Scalable, behavior-based malware clustering.. In *NDSS*.

[5] VMWare Carbon Black. [n.d.]. Threat Hunting. https://www.carbonblack.com/products/solutions/use-case/threat-hunting/.

[6] Crowdstrike. [n.d.]. Threat Hunting. https://www.crowdstrike.com/epp-101/threat-hunting/.

[7] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*.

[8] Chuvakin DarkReading, Anton. [n.d.]. Threat Hunting Is Not for Everyone. https://www.darkreading.com/threat-intelligence/threat-hunting-is-not-for-everyone/a/d-id/1336877.

[9] SANS Institute David Szili. [n.d.]. Building and Maturing Your Threat Hunting Program. https://www.sans.org/media/analyst-program/building-maturing-threat-hunting-program-39025.pdf.

[10] Mariano Graziano, Davide Canali, Leyla Bilge, Andrea Lanzi, Elaine Shi, Davide Balzarotti, Marten van Dijk, Michael Bailey, Srinivas Devadas, Mingyan Liu, et al. 2015. Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*.

[11] Chuvakin HelpNet Security, Anton. [n.d.]. What hinders successful threat hunting? https://www.helpnetsecurity.com/2020/05/26/successful-threat-hunting/.

[12] Xin Hu, Kang G Shin, Sandeep Bhatkar, and Kent Griffin. 2013. Mutantx-s: Scalable malware clustering based on static features. In *USENIX Annual Technical Conference (USENIX)*.

[13] Wenyi Huang and Jack W. Stokes. 2016. MtNet: A Multi-Task Neural Network for Dynamic Malware Classification. In *Detection of Intrusions and Malware, and Vulnerability Assessment*.

[14] Federico Maggi, Andrea Bellini, Guido Salvaneschi, and Stefano Zanero. 2011. Finding non-trivial malware naming inconsistencies. In *International Conference on Information Systems Security*. Springer, 144–159.

[15] Benjamin Moseley and Joshua Wang. 2017. Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search. In *Advances in Neural Information Processing Systems*.

[16] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. In *arXiv preprint*.

[17] Daniel Plohmann and Steffen Enders. [n.d.]. Malpedia. https://malpedia.caad.fkie.fraunhofer.de/.

[18] Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

[19] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*.

[20] Michael Wojnowicz, Di Zhang, Glenn Chisholm, Xuan Zhao, and Matt Wolff. 2016. Projecting" better than randomly": How to reduce the dimensionality of very large data sets in a way that outperforms random projections. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.

[21] Yunan Zhang, Chenghao Rong, Qingjia Huang, Yang Wu, Zeming Yang, and Jianguo Jiang. 2017. Based on multi-features and clustering ensemble method for automatic malware categorization. In *2017 IEEE Trustcom/BigDataSE/ICESS*.