MalFlow: Identification of C&C Servers through Host-based Data Flow Profiling

Tobias Wüchner
Technische Universität München
wuechner@in.tum.de

Martín Ochoa
Singapore University of Technology and Design
martin_ochoa@sutd.edu.sg

Mojdeh Golagha
Technische Universität München
golagha@in.tum.de

Gaurav Srivastava
Technische Universität München
ga79daw@mytum.de

Thomas Schreck
Siemens CERT
t.schreck@siemens.com

Alexander Pretschner
Technische Universität München
pretschn@in.tum.de

ABSTRACT

Modern malware interacts with multiple internet domains for various reasons: communication with command and control (C&C) servers, boosting click counts on online ads or performing denial of service attacks, among others. The identification of malign domains is thus necessary to prevent (and react to) incidents. Since malware creators constantly generate new domains to avoid detection, maintaining up-to-date lists of malign domains is challenging. We propose an approach that automatically estimates the risk associated with communicating with a domain based on the data flow behavior of a process communicating with it. Our approach uses unsupervised learning on data flow profiles that capture communication of processes with network endpoints at system call level to distinguish between likely malign or benign behavior. Our evaluations on a large and diverse data set indicate a high detection accuracy and a reasonable performance overhead. We further discuss how this concept can be used in an operational setting for fine-grained enforcement of risk-based incident response actions.

CCS Concepts

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

Keywords

data flow analysis; malware; command and control server

1. INTRODUCTION

Malware remains one of the biggest IT security threats. Due to increasingly sophisticated measures like encryption or obfuscation [22], malware detection companies are significantly challenged in keeping up the pace with a professionalized malware industry. As a consequence, research on advanced malware detection techniques has recently gained momentum. In particular, state-of-the-art behavior-based malware detection approaches have shown to be effective in detecting unknown or highly obfuscated malware samples where purely signature-based mechanisms fail [5, 15]. But even the most sophisticated malware detection mechanisms always leave a certain chance to malware to avoid detection. This is a consequence of the use of obfuscation techniques, exploitation of monitoring "blind-spots", or outdated behavior profiles. Besides sophisticated detection mechanisms it is important to also consider *risk mitigation* in terms of detecting and dealing with already infected computers.

Most modern malware is internet-based, i.e. its prime attack vectors are drive-by downloads through exploit kits on compromised websites or direct infections through malicious domains embodied in scam or phishing emails. Moreover malware includes functionality to communicate with command-and-control (C&C) servers to receive new instructions, download additional malicious payload, update itself, or send home harvested and stolen sensitive data. On the other hand, many modern malware families communicate with *benign* web sites to perform malicious activities like boosting click counts on advertisements, performing distributed denial of service attacks, or spoofing ratings.

Detecting and evaluating the domain names or IP addresses related to C&C servers is thus important to prevent (new) infections, contain damages, or identify infected hosts. Typical risk mitigation measures include marking a host as infected with a specific malware upon detection of a respective request to a known C&C server or block certain IPs with a firewall. Unfortunately, lists from sources like Google's SafeBrowsing API are known to only cover a small percentage of actual malicious domains used by malware and are challenged by domain generation algorithms [10].

In such cases it is often necessary for organizations to maintain their own blacklists. To do so, on-site malware analysis is done to extract potentially malicious domains and IPs from malware captured in the wild. This is done in a semi-automated way, e.g. with malware sandboxes [12]. While feeding these domains into respective blacklists seems

SAC 2016, April 04-08, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2851613.2851802

tempting, doing so in many cases is not advisable. As discussed before, malware communicate with a variety of benign web sites to e.g. perform internet accessibility checks, click count spoofing, or to intentionally hide the malicious communication within a cloud of benign web activities. Simply passing all extracted domains would thus often lead to severe interference with benign business processes.

To counter this risk of false positives, security experts then manually perform "sanity checks" on the domains extracted from malware. This quickly becomes tedious and costly. Moreover, modern malware families like Conficker or Zeus use DNS techniques like fast flux or domain flux to hide their C&C infrastructure. This results in the dynamic creation of hundreds to thousands newly registered and used domains per day per malware family; manual analysis does not scale.

Automating the domain classification process of extracted domains thus could help to reduce costs and improve effectiveness by compensating error-prone human analysis. This paper proposes an automated classification and risk categorization approach for domains extracted from malware samples. By capturing the interaction of malware with the extracted domains at *host-level* in form of induced quantifiable data flows we establish a notion of host-based activity profile for specific classes of domains.

Coarsly adopting a quantitative data flow analysis based malware detection concept from the literature [17], we then apply a clustering scheme on these profiles to obtain clusters of domains with similar data flow characteristics. These clusters get assigned a numeric threat level, roughly capturing the ratio and threat potential of different kinds of known malign and benign domains associated to the clusters. With this technique we can assign risk levels to unknown domains that then can be used to decide upon triggering fine-grained risk-dependent incident response actions. The risk is calculated based on the distance of the feature vectors to the centroids of the clusters and their respective threat levels.

Opposite to related work [1,4,6,7,11,20] our approach is entirely based on *quantitatively* profiling network-related activity captured at *host-level*. In contrast to more coarse-grained network-based approaches this allows us to analyze the interactivity of *individual processes* with potentially malicious domains and remote resources independent of specific communication protocols or network topologies and can be seamlessly integrated into standard dynamic malware analysis sandbox environments. Finally, our approach allows to conduct fine-grained incident response strategies based on assessed risk of identified potentially malign domains, independent of concrete communication protocols.

Seeing more regular and predictive patterns in malware induced data flows than in benign communication we show that host-level data flow profiling is useful for C&C server detection. We further consider our approach to be more precise and targeted than approaches that use network-level monitors, since it allows us to inhibit communication at a per-processes, or even per-socket level if malign behavior is detected. In contrast, network-based approaches reason about entire hosts, IP address ranges, or ports in such cases and thus are considerably less targeted and more intrusive.

Problem: We tackle the problem of identifying C&C servers based on data extracted from malware samples. That is, we want to solve the problem of partitioning the set of endpoints contacted by malware into malign and likely benign ones with good sensitivity and specificity. Note that we do not target malware classification but rather the discrimination of respective malign servers from likely benign ones from a host-based perspective.

Solution: We establish host-based behavioral profiles for the communication of malware with C&C servers based on characteristic quantitative data flow patterns between processes and contacted endpoints. Using an unsupervised machine learning approach we then create classification models to assign risk levels to unknown samples based on their similarity to learned data flow profiles.

Contribution: To the best of our knowledge, a) we are first to leverage host-based quantitative data flow analysis for C&C server identification; b) to perform risk-based classification, we are first to propose machine learning based clustering on features extracted from *host-based data flows* to the network; c) our system can automatically conduct appropriate risk mitigation steps based on predefined risk-to-incident-response profiles at a per-process granularity.

Organization: We recap the concept of modeling system calls as quantifiable data flows from the literature and briefly discuss a generic domain harvesting, assessment, and risk mitigation workflow in §2. We then present the individual steps of our approach. We evaluate clustering and risk-classification effectiveness and efficiency in §4. We put our approach in context in §5, and conclude in §6.

2. BACKGROUND

To profile communication with malign servers we adopt a concept from literature [17] of interpreting system calls issued by a process as quantifiable data flows whose basic ideas we recap in the following. We then briefly describe a typical workflow for C&C identification to highlight the constituents targeted by our work.

2.1 WinSock Calls as Quantitative Data Flows

Just like Wuechner et al. [16, 17] we capture and interpret network-related WindowsAPI calls as quantifiable data flows between a process and a network endpoint. By tracking and interpreting all network-related WindowsAPI calls issued by a process, i.e. all invocations of the WinSock *Recv* or *Send* functions, and monitoring the size of the respectively transmitted or received data, we obtain a model of the communication behavior of this process with the respective endpoint as sequences of quantitative data flows.

As already shown in literature [16], such quantitative data flow profiles obtained from interpreting system calls of malign and benign processes in principal are sufficiently discriminative to allow for highly accurate malware detection. In addition to that, a considerable amount of work already successfully leveraged data flow analysis at network level [1, 6, 7], i.e. netflows, for the detection of malign network endpoints. Combining these ideas we expect host-level data flow profiles obtained from intercepting network-related API calls of malign and benign processes to also be sufficiently

characteristic to be leveraged for C&C server detection.

In many cases such communication is bidirectional, i.e. consists of phases where data is send and ones where data is received. We thus define a data profile as tuple of an incoming and an outgoing data flow function that maps the absolute amount of received or sent data to discrete points in time. In essence, a data flow profile captures the characteristics of the communication between a process and an endpoint through the individual courses of the in- and outflow functions over time and their relation to each other.

To define the functions and thus profile the actual communication between a process and a server we first analyze the size of the buffers associated with the intercepted WinSock functions and record the respectively received or sent amount of data over time which yields the raw points of our incoming and outgoing data flow functions. By fitting polynomials of a defined degree on these points we then obtain the actual data flow functions that together finally define the actual data flow profiles.

As we will later see data flow profiles of similar communications, i.e. communication of different malware samples from the same malware family with their C&C servers, in majority are sufficiently similar and at the same time distinct enough to unrelated communications that we can use them for the subsequently described analysis tasks.

2.2 Domain Analysis and Risk Mitigation

Computer Security Incident Response Teams (CSIRTs) analyze malware on a daily basis in order to extract characteristic indicators from malware to later allow detection of compromised systems. In large organizations, due to the large scale of such infrastructures, the detection of infected systems often takes place in the network. A typical task for an analyst is to manually analyze and extract network indicators such as domain names from intercepted malware samples and to rate them as benign or malign. This manual analysis is time-consuming and error-prone because of the nature of today's malware communication: malicious software starts many connections to various remote systems, many of them legitimate servers.

CSIRTs typically obtain network indicators through executing suspicious samples within a controlled environment, analyzing their behavior, and extracting relevant network-related information. After the analysis of network indicators, CSIRTs maintain blacklists to block malign domains on the network perimeter, search in the perimeter logs for connection attempts, or define IDS signatures. Another use of the indicators within an organization's network is to *sinkhole* malign IP addresses or domain names to redirect network traffic of potentially compromised systems to a controlled system for further investigations. Depending on the initial risk ratings the CSIRT decides whether to further analyze the respective compromised systems or to directly re-install it to contain the potential threat.

An indicator mistakenly rated as malign can impact the organization's business if, e.g., benign websites, necessary for business-critical transactions, are blocked. Therefore some sort of automated risk ranking of unknown domains supports CSIRTs to classify obtained indicators in a more reliable and fine-grained way and e.g. plan and conduct appropriate incident response strategies.

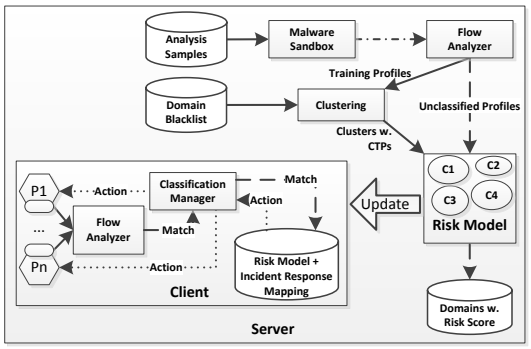

Figure 1: Architecture

3. APPROACH

In the following we describe the different analysis and risk estimation steps of our approach for C&C server identification. The upper part of Figure 1 shows our server-side architecture, the lower left part depicts our client-side components. Solid lines depict the training steps, whereas dashed lines represent classification steps. The semi-dashed lines denote processing steps used for both, training and classification.

Step 1: Network Behavior Extraction

To obtain the raw data needed for profiling malware interaction with C&C servers, we execute malware, i.e. executable PE binaries, infected PDF or Microsoft Word documents within a customized malware sandbox [12].

Our approach interprets interaction between processes and contacted domains at system call level as data flows. We hence only capture networking-related Windows API calls issued by the executed samples, i.e. calls to the WinSock *Recv* and *Send* functions and their derivatives. We decided to directly profile the samples at the Windows API level for two reasons: first, directly capturing network activity by intercepting system calls allows us to selectively capture network activities of specific processes and thus significantly reduce to-be-processed data and noise. This is hard to achieve with monitors at the network level. Second, if the monitor is deployed at the host level, the interception of network-related system calls for individual processes allows us to inhibit further malign network activity at a per-process or even per-socket rather than at a coarse per-machine level and thus limits the intrusiveness of our approach.

We use the open-source malware sandbox Cuckoo [12] to capture network-related system call traces from executed samples for large-scale malware collections. As our approach relies on the availability of quantitative system call information such as the amount of data read or written to a socket we needed to substitute Cuckoo's WindowsAPI monitor with a custom built monitor that can deliver the required information. Specifically, we obtain fine-grained networking

information like the domain of a remote endpoint by having our monitor inspect the buffers of the respective *send* or *recv* WinSock function calls. After submitting a sample to the sandbox, its network-related WindowsAPI calls are recorded for a fixed period of 5 minutes. The obtained system call traces are input to the subsequent processing and classification steps.

Step 2: Data Flow Profiling

After the raw data retrieval we can now extract and constitute the actual data flow profiles. To this end we split the set of intercepted events into events that lead to outgoing and events that lead to incoming data flows. Each inflow or outflow event is then interpreted as a point on a incoming or outgoing data flow function, mapping the absolute amount of transmitted or received data to discrete points in time. Taken together the functions build a mathematically convenient and simple model of the underlying low-level communication aspects.

The intuition behind using such data flow profiles, i.e. in- and outflow functions, is that we expect the data flow profiles of interaction with malign endpoints to look substantially different from communication with benign endpoints and very similar for samples belonging to the same malware family. An example for such a behavioral difference is connectivity tests of a malware sample with a benign website like google.com. After a successful connectivity test, the malware would then periodically communicate with its C&C server to receive new instructions or upload stolen data.

The respective data flow profiles for the interactivity with the benign website would then likely indicate a steep increase of the transferred data to the respective domains in the beginning, followed by a long period of silence, as the benign domain only needs to be contacted once. In contrast, the malign C&C communication would require periodic interaction and thus would lead to a more "spiky" data flow profile. Furthermore, specificities of the communication protocols used by different malware families are likely to yield characteristic family-specific data flow profiles.

Figure 2 substantiates this assumption by showing data flow profiles obtained from the real-world execution traces of several samples from two different malware families. For sake of brevity we only show the graphs of the outflow function part of the data flow profiles, where the y-axis relates to the total amount of transmitted data in bytes and the x-axis to time w.r.t. equidistant intervals of about 700ms. The left part of the figure shows the graphs of the outflow functions of 117 samples from the Zeus malware family communicating with their C&C server, the right part shows the outflow functions of 13 Kraken malware samples.

As we can see from the figure the function shape for both families differs quite significantly with most of the samples of one family following the same basic function shape.

Step 3: Training Phase

After having generated data flow profiles for captured event traces of a large body of executables, we are now ready to train the machine learning core of our approach.

However, before doing so we first need to transform the data flow models into a form that is more handy for machine learning purposes, i.e. into vectors of real numbers. To this extent we fit polynomials of fixed degree to each function to approximate its shape while at the same time ignoring insignificant variations. This approximation allows us to express data flow models, i.e. their flow functions, through the coefficients of the polynomials that we fit on them. By varying the degree of the fitted polynomials we can control the level of function approximation and thus to some extent the tolerance to noise in the training data.

Finally, we use the so obtained function coefficients as elements of a vector. Each vector thus consists of the coefficients of the polynomials fitted on the inflow and outflow function of a data flow profile. Each feature vector f thus encodes exactly one data flow profile with its dimensionality depending on the degree of the fitted polynomials.

In principle, one could now simply use these feature vectors for training a standard supervised machine learning classifier, e.g. using domain blacklists for labeling the vectors according to the domains or IP addresses the data flow profiles correspond to.

Unfortunately, such blacklist are inherently incomplete [10], quickly become outdated, and thus would often provide incorrect labels to the feature sets. This is for instance caused by malign domains taken down by authorities, domains not anymore used by malware, or blacklists not being able to keep the pace with malware developers that register hundreds to thousands new websites every day. Trained on such imprecise labels for the training feature sets, a classifier would likely incorrectly classify unknown feature samples.

Therefore, our approach instead relies on an unsupervised clustering technique that takes the unlabeled feature vectors as input and outputs clusters of samples with approximately similar data flow profiles. More specifically, we use Mini Batch k-means, a faster approximate version of the more expensive k-means clustering [14], to obtain clusters based on aggregation of instances that are located at a similar distance around a cluster centroid. This notion of distance of an instance, i.e. a set of features corresponding to a domain instance, to a certain cluster is later needed as the first component (i.e. likelihood) for calculating individual distance-based risk scores.

Step 4: Classification and Threat Potential

We want to assign a so-called *threat potential* to domain clusters. This cluster threat potential represents the weighted potential security threat that arises from communicating with the members of this cluster and will thus be later used as the second component (i.e. severity) of the per-sample risk score calculation.

We first need to obtain additional information for a cluster. We do so by matching the cluster members' domains against various malware domain blacklists. For our prototype, we use the public API offered by VirusTotal. We differentiate between two classes of domains: domains that had a match in a blacklist, denoted by *malign*, and *benign* domains that we did not find in a blacklist.

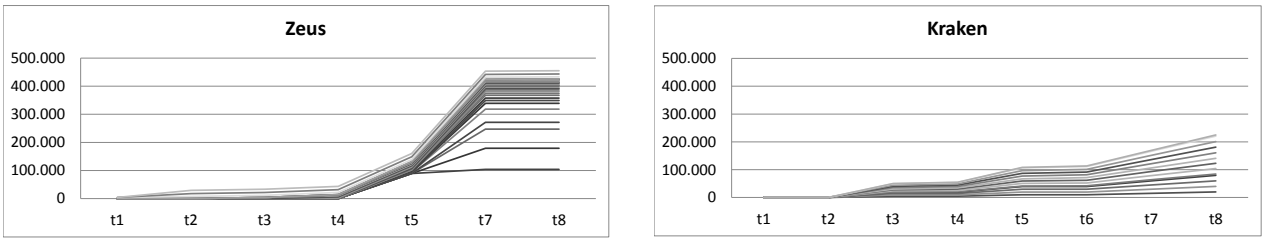

Figure 2: Data Flow Profiles of different Malware Families

Formally, let $\mathcal{D}$ be a set of domains. Domains $d \in \mathcal{D}$ have a type $\gamma \in \Gamma = \{malign, benign\}$. $\tau : \mathcal{D} \to \Gamma$ returns the type of a given domain. Let $c \in C$ be one specific cluster drawn from a set of clusters, C. Each cluster is a set of m pairs of domains and their respective feature vectors (of dimensionality n) $c \in (\mathcal{D} \times \mathbb{R}^n)^m$. The *threat level* of a domain captures the estimated threat that arises in a specific deployment context (i.e. one company or organization) if communication with a potentially malign or benign domain is not controlled or blocked, respectively. For known *malign* domains we assign a positive threat level and for *benign* domains a zero threat level. We thus model the threat level by $tl := \Gamma \to \mathbb{Z}$. One simple threat level function is $tl(malign) = 1$ and $tl(benign) = 0$.

The exact threat levels for the different classes manually need to be set once to match the protection goals within a specific context. They can substantially differ for different operational contexts. In some companies, where undetected security incidents are considered significantly worse than the denial of legitimate actions, one might assign a very high threat level to known malign sites with the risk of producing false positives (i.e. blocking legit actions). In contrast, in companies where even slight interference with legitimate workflows is considered prohibitive, reducing false positives would be an important goal which can be achieved by assigning smaller threat levels to malign domains.

We then calculate the threat potential for each cluster $c \in C$ as the average of the cluster members' individual threat levels. We compute the average in order to explicitly address the problem that the individual domain class labels and thus their individual threat levels might not be up-to-date and reliable and thus, when taken alone, do not provide enough reliable training information.

This definition of a cluster's threat potential means that the threat potential gets higher the more known malign and less benign members it has. To project the cluster threat potential levels to an $[0, 1]$ interval, we divide each individual cluster threat potential by the size of the cluster. The *cluster threat potential*, $ctp := C \to [0, 1]$, is thus defined as $ctp(c) := \frac{1}{|c|} \cdot \sum_{(d,f)\in c} tl(\tau(d))$.

This way, in contrast to directly using the not fully reliable labels for supervised learning, we distribute the risk induced by potentially incorrect labels among the different clusters. This makes the corresponding predictions less sensitive towards noisy and partially incorrect training data.

Step 5: Risk Assessment Phase

With the trained classifier and the *ctp*-annotated clusters we can now assign meaningful risk scores to unknown samples. To classify an unknown sample, i.e. calculate its risk level, we determine the cluster with the highest similarity of its member's data flow profiles and the test sample's profile.

For this we compute the distance of each cluster's centroid to the feature set of the sample to be classified. As we use a variant of the k-means algorithm for clustering, centroids in our case refer to the mean of the feature vectors belonging to a cluster. As a consequence, we can use the Euclidean distance between the samples feature vector and the clusters' centroids. More precisely, we define the distance $\|\cdot\|_c : \mathbb{R}^n \times C \to [0, 1]$ of a sample, represented by its feature vector $s \in \mathbb{R}^n$, to any cluster $c \in C$ as follows:

$$\|s\|_c := min\Big(1, \frac{\|s - \text{centroid}(c)\|_2}{N}\Big)$$

where centroid : $C \to \mathbb{R}^n$ is the centroid vector of a cluster, and $\|\cdot\|_2$ is the Euclidean distance. As for *ctp*, we want to project the distance to an interval between 0 and 1. We do so by normalizing each distance by the diameter N of the smallest sphere containing all clusters. Since there could be samples whose distance to a centroid is greater than N, we set 1 as the maximum possible distance. As an example, the point s in the 2D projection of Figure 3 is closer to the centroid of c_2 but further away from the centroid of c_1.

The actual risk score is then classically computed as "likelihood times severity". Intuitively, the smaller the distance to a high risk cluster, the higher is the overall risk. This leads us to using the complement of the normalized distance to represent the likelihood. It is also intuitive to choose the risk associated with the *closest* cluster to the sample. The risk assessment $risk : \mathbb{R}^n \to [0, 1]$ is thus computed as $risk(s) = (1 - \|s\|_{\hat{c}}) \times ctp(\hat{c})$, where $\hat{c}$ is the cluster closest to sample s, i.e., $\forall\ c \in C\ \ \|s\|_{\hat{c}} \leq \|s\|_c$.

By applying this classification strategy we are now able to determine individual risk scores for unclassified domain communication data flow profiles that we later use for reasoning about appropriate incident response actions.

Step 6: Deployment and Incident Response

Our approach can now classify unknown endpoint samples and annotate them with risk scores. We need to map this risk to specific incident response actions.

For this we define so-called *incident response profiles* (IRP) that map risk score intervals to incident response actions,

e.g. reporting a potential threat, forwarding malign traffic to a sinkholing server, blocking a domain, or even killing the respective process. We describe IRPs as partial functions $irp \in IRP : \mathbb{R} \rightarrow A$ that return a specific incident response action $a \in A$ if a provided risk score r is within a certain interval. An example for such a profile $irp_1 \in IRP$ would be: $irp_1(r) = \textsf{Ignore}$ if $0.0 \leq r \leq 0.2$, $irp_1(r) = \textsf{Notify}$ if $0.2 < r \leq 0.5$ and $irp_1(r) = \textsf{Block}$ if $0.5 < r \leq 1.0$. Note that the thresholds in real-world deployment settings need to be instantiated in congruence to concrete protection goals.

To operationalize this process, MalFlow supports two distinct deployment modes: *server-only* and *client-server* deployment. For server-only deployment, we perform all previously discussed training and classification steps directly at the server where the malware sandbox is deployed. The obtained classification results are then used to e.g. populate proxy blacklists to a-priori block the identified malign domains or notify security responsibles upon detected connection attempts to these domains. This deployment model bears the benefit of full centralization and easy integration into a typical dynamic malware analysis process. The downside is that it does not allow to conduct precise per-process incident response actions at the client side.

This is overcome by our *client-server* deployment model where domain classification is pushed to the clients. Rather than centrally blacklisting domains, the server merely generates and distributes the classification models to the clients – classification of unknown endpoints is done by the clients.

To this end, clients periodically receive updated classification models from the MalFlow server along with the risk mitigation mappings. The clients then monitor the system for potential malign network communication using an IAT-patching based syscall monitor [18] that gets injected into each process before its startup. After injection, the monitor intercepts all WinSock calls and forwards them to the flow analysis component. The classification manager matches these feature sets against the classification model, received from the MalFlow server, and calculates the respective risk score for the contacted remote endpoint. Together with the pre-defined IRPs, the risk score then determines the incident response action for the identified potential threat.

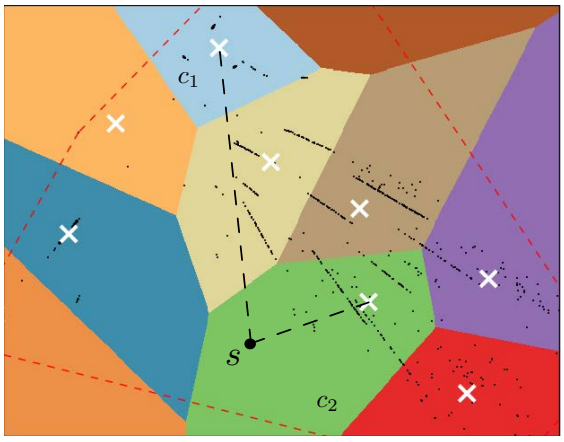

Figure 3: PCA projection of Clusters and their Centroids

4. EVALUATION

Our server-side experiments were conducted on an Intel Xeon 12-core 3.5GHz Ubuntu 12.10 machine with 64GByte of RAM; the client side was evaluated on a dual-core 2.8GHz Intel i7 Windows 7 machine with 8GByte of RAM.

We executed about 27.000 samples in our sandbox that we obtained between February and September 2015 from malshare.com and from an email server under our control, encompassing of 47 distinct malware families including Zeus, Conficker, Kraken, or Virut as evaluation baseline.

For approximating the generated data flow profile functions we least-square-fitted polynomials of degree 5. In total we obtained a set of 46.312 distinct data flow profile vectors out of which the matching of the corresponding IP addresses and domains with VirusTotal revealed about 60% of them referring to known malign servers, 38% relating to known benign servers, and 2% being completely unknown to VirusTotal.

4.1 Effectiveness

To evaluate our ability to separate malign from (likely) benign endpoints we performed a 10-fold cross validation experiment where we repeatedly trained with 90% parts of the data set (excluding the samples unknown to VirusTotal) and used the resulting risk model to calculate the risk for the remaining 10% of the data set (including the unknown ones).

To investigate the effects of different risk thresholds θ we calculated the area under receiver operator curve (AUC) which is a composed measure to capture false positives and false negatives. For this we consider benign domains incorrectly classified as malign as false positives and benign labels wrongly classified as malign as false negatives. To obtain an ROC we evaluated the performance of our classifications as congruence with the respective VirusTotal classification information. Given that corresponding blacklists are not always fully reliable and inherently incomplete [10], there is a chance that by this some of the samples get mislabeled.

This has some consequences for the interpretation of our evaluation results. As we cannot say with absolute certainty that a domain in fact is malign or benign, a disagreement between predicted class and ground-truth class does not necessarily constitute an incorrect prediction. Correspondingly, the subsequently presented effectiveness measures need to be interpreted as lower bounds of the actual effectiveness.

As we employ a unsupervised learning scheme, the effectiveness of our approach likely dependents on the amount of generated clusters. The used clustering algorithm demands the number of clusters to be a-priori fixed, we thus repeated the cross-validation experiments for different numbers of clusters. Figure 4 gives an overview of the respectively achieved effectiveness in terms of AUC for different numbers of clusters and the respectively achievable true positive rate (TPR) when fixing the false positive rate to 2%.

As we can see the maximum effectiveness is achieved with around 80 clusters after which increasing the cluster count seems to not significantly improve effectiveness. Considering the ROC of the best-performing cluster count experiment we are thus able to correctly identify at least 70% of the malign servers within the evaluation data set with less than 2% false

Figure 4: Effectiveness vs. Cluster Count

positives. Again note that these numbers are lower bounds due to the unreliable ground truth [10].

Finally, a more thorough analysis of the clustering results revealed that 410 of the 926 samples that were not known to VirusTotal had almost identical data flow profiles as known malign ones. Considering the comparably low false positive rate of our approach we thus assume that we were indeed able to identify unknown C&C servers.

4.2 Efficiency

For the efficiency evaluation we separately analyzed the server (training) and the client side components (detection).

To get a realistic estimate of induced performance overheads we conducted our experiments in a typical client environment (clients are more resource-limited than servers). For this we simulated typical networking scenarios like browsing or downloading files from a server and measured the relative overhead induced by our client-side component. To this end we used *wget* to download files between 0.1 and 100 MByte of size from a big webspace hoster. To weed out environmental influences we repeated each experiment 100 times. While the results indicated overheads of up to 7 times the normal execution time for very short-living processes, the relative overhead for longer-living processes with continuous networking activities at average remained below 1.2.

For evaluating the training phase costs we measured the total training time spent on different-sized training sets and varying cluster counts. Interestingly, our experiments only revealed a measurable correlation between computation time and cluster count, but not w.r.t. sample count. We explain this with the relatively small number of clustering instances in our setting in comparison to the several orders of magnitude bigger sample sizes that the employed clustering algorithm was originally designed for [14]. For the cluster counts considered in the effectiveness evaluation, training our approach on the full data set at average took about 380s. As the bulk of the risk calculation consists of calculating the Euclidean distance of a sample to all cluster centroids, the effort is linear with respect to the number of clusters. Considering the small number of clusters and the comparably low dimensionality of the vectors in our setting, the resulting overhead at average was below 5ms. When compared to the monitoring overhead, this is negligible.

4.3 Threats to Validity and Limitations

While we could show that our approach performs well in discriminating domains with reasonable computational overhead, we are aware of some limitations of our evaluation. As usual with machine-learning approaches to malware, we cannot claim that our results generalize to malware found in the wild or to other data sets. An objective evaluation of this generalizability is very hard to achieve.

Also, the usefulness of the effectiveness results highly depends on the reliability of the ground truth database, i.e. the coverage of the employed domain blacklists. A bad coverage possibly leads to effectiveness reported to be lower than it actually is. Moreover, the usage of VirusTotal as fixed source of ground truth in itself is not entirely unproblematic, as malware authors are known to actively prevent getting listed by VirusTotal which yields domain blacklists to quickly become outdated. However, we argue that reliable ground truth is a common issue of malware and C&C server classification in general and we showed that our approach to some extent can cope with unreliable ground truth.

Our prototype profiles malware network interaction with a user-mode WinSock hooks. Kernel-mode malware, or more generically, malware that does not use the WinSock library can thus not be profiled by our current prototype.

Furthermore, our current prototype only considers the first 5 minutes of activity of a malware for training. There is thus a risk of malware delaying the actual malign behavior and thus subverting our training scope. One can to some extent counteract this threat by extending the monitoring period or utilizing stimulus-response techniques [3].

Finally, we are aware that our approach would be challenged by malware that highly randomizes its network activity. However our evaluations indicate a good effectiveness on state-of-the-art malware and we are not aware of current commodity malware actively obfuscating its network behavior from a quantitative data flow perspective.

5. RELATED WORK

Identification of malign endpoints has been a very active area of research in the past decade. Traditionally the focus has been on identifying suspicious traffic at network level, e.g. in form of netflows [1, 6, 7]. In [7], Gu et. al. identify botnet activity by analyzing spatio-temporal correlations in network traces, e.g. bots reporting the results of a command to the C&C server at the same time. Wurzinger et al. [19] derive models of network communication, i.e. sequences of messages between bots and C&C servers, that allow to detect botnet related traffic. By analyzing a large dataset of DNS traffic, Bilge et al. [2] identify 15 features that are useful to discriminate benign from malign domains, including time-based and domain name-based features. Focusing on domain name-based features, Yadav et al. [21] propose to identify algorithmically generated malicious domains. Jacob et al. [8] propose to use host-level resource dependency graphs to relate malign behavior with associated endpoints.

Another line of research, started by Caballero et al. [4], uses machine learning and active querying to fingerprint (and thus classify) remote servers based on their responses. Following this idea, Nappa et al. [11] used probing to detect malicious hosts, and were able to identify several C&C servers and P2P bots on the wild. Xu et al. [20] followed up by leveraging binary analysis to generate fingerprints even if

C&C serves where off-line at the time of analysis.

We mainly differ from these approaches in that we perform *quantitative* data flow analysis instead of protocol- or query-based *qualitative* analysis. Following the evidence and arguments in [17], we believe that our approach is more robust towards changes in C&C communication protocols, remote resources, or query structures, but leave a thorough empirical analysis of robustness to future work. Moreover, we introduce an active risk-based incident response approach whereas the above approaches concentrate on detection.

Unsupervised learning for malware detection and classification has been extensively studied, based on both static and dynamic features. For instance, Rafique et al. [13] cluster malware based on network signatures obtained by executing the malicious samples. Kheir et al. [9] propose to cluster malware into families with similar HTTP communication patterns, ignoring the contacted domains. We differ in that we use unsupervised learning to define a robust classifier *despite unreliable ground truth* whereas the mentioned approaches do pure clustering.

In sum, to our knowledge, and in contrast to related work, we are the first to do clustering on *host-level data flow profiles* for C&C detection and risk estimation.

6. CONCLUSIONS AND FUTURE WORK

By analyzing similarities in data flows induced by processes communicating with malign and benign servers at host-level our MalFlow tool can assesses the risk presented by processes communicating with unknown network endpoints. On a large and diverse dataset we were able to identify at least 70% of the malign endpoints when fixing the acceptable false positive rate to a maximum of 2%. Furthermore, using our tool we were able to identify servers unknown to VirusTotal with data flow profiles very similar to known malign ones which we thus consider to be most likely also malicious.

In comparison with the currently practiced costly, slow, and failure-prone alternative – mainly manually categorizing domains extracted from malware samples with help of various black- and whitelists – we consider our automated approach a significant improvement. Especially our approach's capability of automatically assigning meaningful risk scores to unknown domains based on a semantically justified decision model allows companies to some extent to reduce manual labor of expensive security analysts with our tool.

We also showed how to operationalize our approach by means of a client component that can issue fine-grained per-process incident response actions based on our domain risk classification, which induces an average relative overhead of 1.2 times in real-world networking settings.

7. REFERENCES

[1] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: Detecting botnet command and control servers through large-scale netflow analysis. In: ACSAC (2012)

[2] Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis. In: NDSS (2011)

[3] Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., Yin, H.: Automatically identifying trigger-based behavior in malware. In: Botnet Detection (2008)

[4] Caballero, J., Venkataraman, S., Poosankam, P., Kang, M.G., Song, D., Blum, A.: Fig: Automatic fingerprint generation. CMU TechReport (2007)

[5] Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. CSUR (2012)

[6] Gu, G., Perdisci, R., Zhang, J., Lee, W., et al.: Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: USENIX Sec (2008)

[7] Gu, G., Zhang, J., Lee, W.: Botsniffer: Detecting botnet command and control channels in network traffic. In: NDSS (2008)

[8] Jacob, G., Hund, R., Kruegel, C., Holz, T.: Jackstraws: Picking command and control connections from bot traffic. In: USENIX Sec (2011)

[9] Kheir, N., Blanc, G., Debar, H., Garcia-Alfaro, J., Yang, D.: Automated classification of c&c connections through malware url clustering. In: SEC (2015)

[10] Kührer, M., Rossow, C., Holz, T.: Paint it black: Evaluating the effectiveness of malware blacklists. In: RAID (2014)

[11] Nappa, A., Xu, Z., Rafique, M.Z., Caballero, J., Gu, G.: Cyberprobe: Towards internet-scale active detection of malicious servers. In: NDSS (2014)

[12] Oktavianto, D., Muhardianto, I.: Cuckoo Malware Analysis. Packt Pbl. Ltd (2013)

[13] Rafique, M.Z., Caballero, J.: Firma: Malware clustering and network signature generation with mixed network behaviors. In: RAID (2013)

[14] Sculley, D.: Web-scale k-means clustering. In: WWW (2010)

[15] Wressnegger, C., Schwenk, G., Arp, D., Rieck, K.: A close look on n-grams in intrusion detection: anomaly detection vs. classification. In: AISec (2013)

[16] Wüchner, T., Ochoa, M., Pretschner, A.: Malware detection with quantitative data flow graphs. In: ASIACCS (2014)

[17] Wüchner, T., Ochoa, M., Pretschner, A.: Robust and effective malware detection through quantitative data flow graph metrics. In: DIMVA (2015)

[18] Wüchner, T., Pretschner, A.: Data loss prevention based on data-driven usage control. In: ISSRE (2012)

[19] Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., Kirda, E.: Automatically generating models for botnet detection. In: ESORICS (2009)

[20] Xu, Z., Nappa, A., Baykov, R., Yang, G., Caballero, J., Gu, G.: Autoprobe: Towards automatic active malicious server probing using dynamic binary analysis. In: CCS (2014)

[21] Yadav, S., Reddy, A.K.K., Reddy, A., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: IMC (2010)

[22] You, I., Yim, K.: Malware obfuscation techniques: A brief survey. In: BWCAA (2010)