PeerViewer: Behavioral Tracking and Classification of P2P Malware

Nizar Kheir and Xiao Han

Orange Labs, Paris

Abstract. To keep pace with the rampant malware threat, security analysts operate tools that collect and observe malicious content on the internet. Since malware is robust against static analysis, dynamic environments are being used for this purpose. They use automated platforms that execute malware and acquire knowledge about its runtime behavior. Today, malware analysis platforms are powerful in characterizing the system behavior of malware. However, little research is being done to automatically charaterize malicious code according to its network communication protocols. Yet this is becoming a real challenge as modern botnets increasingly adopte hybrid topologies that use custom P2P protocols for command and control.

This paper presents PeerViewer, a system that automatically classifies malware according to its network P2P behavior. Nowadays P2P malware either uses variants of known P2P protocols, or it builds its custom P2P protocols as for Sality and zeroAccess. PeerViewer builds classifiers for known P2P malware families. Then it builds a network footprint for malicious code running in a sandbox, and compares this footprint with those for known P2P malware families. It associates malicious code with a known botnet family where possible, or it notifies the security analysts of a new or unknown P2P malware family, so it can be considered for a deeper analysis. Our experimental results prove the ability of PeerViewer to accurately classify P2P malware, with a very low false positives rate.

1 Introduction

Over the past decade, malware has infected every corner of the internet, with no signs of it abating. So far it became the root cause for many security problems such as spam, denial of service and data theft; yet it is branching on social networks and mobile devices [3]. As long as malware is growing rampant, it ecompasses a range of threats where botnets constitute the most widespread type today. These are networks of infected nodes controled by a single attacker through a common Command and Control (C&C) network. Today, botnet trackers mostly use dynamic analysis environments where they execute malware in order to learn about its malicious techniques [2,5,13,21,22]. Whichever means they use to collect malware (e.g. traffic sampling, honeypots, monitoring phishing emails), security analysts are being overwhelmed with a huge number of malware samples daily [20]. Most of these samples are polymorphic variants of known malware families, thus urging researchers to propose dynamic and automated malware classification models [25]. In fact variants of the same malware family share typical behavioral patterns that reflect their origin and purpose. Automated classification models thus discard samples that are variants of known malware families, creaming off new malware that would be further submitted to a deeper analysis. They observe malware runtime features such as system calls, registries and memory in order to build appropriate behavioral classifiers.

Current malware analysis tools mostly operate at the system level. They build behavioral patterns that apply to host-based malware detection and diagnosis. Yet they provide only a raw description of malware network behavior, usually limited to domain names, supported protocols and callbacks. While this level of information enables detecting and neutralizing malware that uses centralized botnet architectures, it has proven to be insufficient against hybrid and distributed botnets. With malware increasingly adopting hybrid C&C topologies, current systems are struggling with their fight against botnets [10,15]. For example, the recent switching of Zeus to a hybrid C&C topology made Zeustracker unable to produce exact C&C domain block lists [1].

Hybrid botnets have network patterns and behaviors that are clearly different from centralized botnets. They usually operate outside the DNS system, which makes domain block lists irrelevant. They use custom P2P protocols, as opposed to centralized botnets that mostly use `HTTP` for command and control. Hence, behavioral classification based on malware `HTTP` patterns is no more appropriate against P2P botnets [22]. Yet hybrid botnets use a wide range of P2P protocols, each one implementing its own set of messages such as keep-alive, route discovery, data search and broadcast. Besides, botnet P2P flows are usually encrypted and transmitted over TCP and UDP alike, which makes difficult to classify malware P2P flows based on the message types they are carrying. Therefore, the network behavior of P2P malware during dynamic analysis would be no more than a bunch of encrypted flows, with no clear evidence about their nature and remote destinations. To the best of our knowledge, *it is yet unclear how we can assign malicious code to a common P2P malware family while observing its network behavior during dynamic analysis.*

This paper presents PeerViewer, a system that automatically classifies P2P malware according to its network behavior. PeerViewer uses a learning set of known P2P malware families such as Sality, Zeus, TDSS and zero access [14,23,26]. It uses machine learning techniques in order to build a network based classifier for each family of P2P malware. It further builds a network footprint of P2P malware when executed in a dynamic analysis environment, and checks this footprint against known P2P malware classifiers. PeerViewer assigns the malicious code being analyzed to the appropriate P2P malware family where possible, or it notifies the security analysts of a new or yet unknown P2P malware family. In the latter case, the malware can be considered for a deeper analysis as it may reveal new trends in P2P botnet activity.

Through its automated and dynamic classification of P2P malware, PeerViewer offers two main contributions. First, it reduces overhead for malware

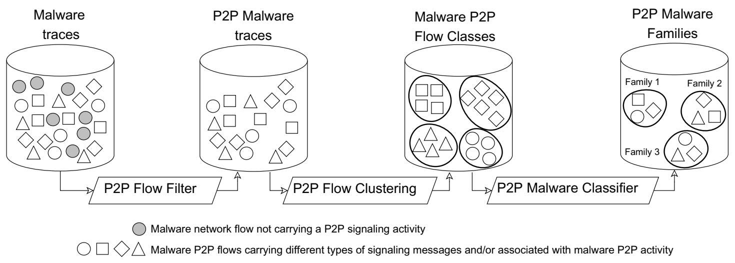
Malware traces
P2P Malware traces
Malware P2P Flow Classes
P2P Malware Families
Family 1
Family 2
Family 3
P2P Flow Filter
P2P Flow Clustering
P2P Malware Classifier
Malware network flow not carrying a P2P signaling activity
Malware P2P flows carrying different types of signaling messages and/or associated with malware P2P activity

Fig. 1. Overview and workflow of PeerViewer

analysts by pinpointing only new P2P malware to be considered for further investigation. Second, it provides behavioral detection models that can be used to detect P2P bots and associate them with known or new P2P malware families. Our experimental results prove the ability of our system to accurately classify P2P malware, with only very few false positives.

This paper will be organized as follows. Section 2 presents PeerViewer and provides an overview of its architecture. Section 3 describes more in details the different modules that constitute our system. Section 4 details our experiments and results. Section 5 discusses the limitations of our system and provides future work. Section 6 presents related work, and section 7 concludes.

2 System Overview

PeerViewer builds families of P2P malware based on their network behavior when executed into a dynamic analysis environment. It aims at automatically classifying malware that uses known P2P protocols, and to cream off malware that implements new P2P protocols so it can be considered for further analysis. PeerViewer operates in two phases, the *buildup* phase and the *detection* phase. During buildup, it builds P2P classifiers using a training set of malware belonging to several P2P families. PeerViewer has a modular framework that makes possible to add new P2P classifiers when new P2P malware families are discovered. During detection, PeerViewer observes malware traffic and builds a network P2P footprint for each malware sample. It uses this footprint in order to associate malware with a known P2P family where possible. Otherwise PeerViewer notifies the security analysts about a new malware that belongs to a yet unknown P2P family.

As in figure 1, PeerViewer includes three separate modules. The P2P flow filter implements several heuristics which aim to discard malware that does not show any P2P activity during analysis. For instance, the rate of failed connection attempts is usually used as a way to detect P2P applications. Therefore, our filter discards malware whose rate of failed connection attempts does not exceed a given threshold. It also uses other heuristics such as flows initiated after successful DNS requests, number and geographical distribution of remote contacted IPs. Our experimental results prove that our filter is indeed effective in eliminating non-P2P malware, with almost no false positives.

Remaining flows for P2P malware in our dataset are used as input to the flow clustering module. It groups together malware flows that are likely to implement the same P2P functionality and that use the same P2P protocol. The flow-size distribution for P2P signaling activity shows frequent flow sizes that are associated with specific P2P message types [7,18]. Malware that implements the same P2P protocol and belongs to the same P2P botnet topology would have the same P2P signaling activity, thus resulting in similar flows when observed at the network level. PeerViewer uses unsupervised clustering in order to group together similar malware flows that are likely to implement the same P2P activity. The flow clustering process uses high-level malware traffic features such as flow size, number of packets, bits per packet, and flow duration. The output of this process is a multiple set of clusters, each one including P2P flows triggered by multiple malware samples, but carrying the same P2P signaling activity (e.g. keep-alive, route discovery, search request, push data) and protocol.

Malware of the same family has its P2P flows grouped within the same clusters because they carry the same P2P signaling activities. The malware classifier module uses P2P flow clusters in order to build families of malware that implement the same P2P protocol. In fact, PeerViewer builds a P2P footprint $\mathcal{F}_\alpha\{c_i\}_{i=1}^m$ of size m for each malware $\mathcal{M}_\alpha$ in our initial learning set, and which specifies the rate of malware P2P flows within each cluster $\{c_i\}_{i=1}^m$. In other terms, the feature $\mathcal{F}_\alpha\{c_k\}$ would be set to 0 if malware $\mathcal{M}_\alpha$ has no flows in cluster c_k, and it will be set to 1 if it has all its P2P flows in c_k. PeerViewer uses malware footprints as a training set to build P2P malware clusters, each cluster representing a new P2P malware family. Hence, malware that belongs to the same family implements the same P2P protocols and has the same P2P botnet topology.

We evaluated our behavioral P2P malware classifier against malware signatures for three anti-virus solutions. Our experiments prove that PeerViewer builds malware P2P families with a very high accuracy. It further builds a classifier for each P2P malware family, so it can be used to classify P2P malware on-the-fly. PeerViewer assigns a malware sample to the family that best fits its network footprint. Malware that matches with any of the P2P malware families in our training set belongs to an unknown family. It is thus submitted to the security analyst for a manual inquiry.

3 System Description

This section describes the architecture and workflow of PeerViewer, including the process and tools that it uses to build and classify P2P malware families.

3.1 Malware P2P Flow Filter

Malware P2P filter uses heuristics that select P2P malware and discard flows that do not carry P2P signaling activity. These heuristics characterize a distributed

P2P activity using high-level network behaviors. In fact, P2P traffic has multiple characteristics that are clearly different from other centralized network communications. For instance, P2P networks constitute unstructured topologies where P2P nodes may constantly join and leave the network. This phenomenon results in a high rate of failed connection attempts, which is a distinctive feature of P2P activity. Our filter implements the following features.

DNS Filtering is commonly used to discard non-P2P traffic. Nodes in a P2P network operate outside the DNS system [4]. They search for other peers using routing tables in the overlay network, without prior DNS requests. Although access to a central server through DNS resolution is possible at bootstrap, nodes further communicate directly using IP addresses, and access to the DNS service is usually no longer required. Therefore, PeerViewer discards malware flows initiated after a successful DNS resolution.

Failed Connection Filter processes flows not eliminated by the DNS filter. It discards non-P2P malware using the rate of failed connection attempts, which characterizes the independent arrival and departure by thousands of peers in the network. We consider as a failed connection attempt all unidirectional UDP flows, as well as failed TCP syn attempts including both no TCP response or a TCP reset. PeerViewer uses the rate of failed connection attempts within a malware trace as a way to discard non-P2P malware.

Flow size filter keeps only flows that include P2P signaling activity, and discards P2P data flows. The flow size distribution of P2P traffic usually shows discontinuities near small flow sizes, and that characterize P2P signaling activity [12]. It also includes flows with clearly higher flow sizes, usually involving data transfer. PeerViewer uses the flow size distribution in order to discard P2P data flows. It drops all flows whose size exceeds a given threshold that we empirically set based on P2P flow size distributions in [16,12].

AS-based filtering: P2P botnets constitute overlay architectures that spans multiple autonomous systems (AS). We use the rate of distinct AS numbers within a malware trace in order to discard non-P2P malware. It is defined as the number of remote AS to the total number of flows ratio in a given malware trace. We empirically set a threshold $\tau_{as} = 0.2$ for this rate, based on our malware training set. PeerViewer discards malware whose rate of distinct AS numbers does not exceed this threshold.

Although these heuristics cannot discard all non-P2P flows, they are reliable enough to characterize the network behavior of P2P applications. They describe invariants in the P2P signaling activity, and so they cannot be easily evaded without modifying the P2P protocol implementation. The output of this filter is a set of P2P signaling flows for each malware sample. We use these flows as input to the flow clustering module. It groups P2P signaling flows for malware in our training set according to protocols and message types they are carrying.

3.2 Malware P2P Flow Clustering

PeerViewer aims at classifying malicious code based on its P2P network behavior. We define the network behavior of a P2P application through its signaling activity, and which results in a different distribution of its network flows. We proceed first with a flow clustering step that groups together malware P2P flows that implement the same protocol and signaling activity. We further use clusters of P2P flows in order to define a P2P footprint for each malware sample.

We consider as a malware P2P flow both the flow triggered by a malware and its associated peer response. We represent a bidirectional flow using the following features vector: $f_\alpha =< \mathcal{M}_\alpha, proto, B_s, B_r, Pkt_s, Pkt_r, \Delta_t >$. Features of this vector are defined as follows: $\mathcal{M}_\alpha$ is a tag that associates flow f_α with malware $\mathcal{M}_\alpha$; *proto* is a tag that designates the transport layer protocol, being either TCP or UDP; B_s and B_r are the amount of Bytes sent and received within f_α; Pkt_s and Pkt_r are the number of packets sent and received; and Δ_t is the flow duration. PeerViewer separately builds clusters for TCP and UDP flows using the *proto* tag, as these flows clearly carry different signaling activities.

We use the unsupervised incremental K-means clustering algorithm in order to build clusters of P2P flows. It starts with an initial number of clusters, and increments clusters when the distance of a flow to all existing clusters exceeds a given threshold. We use the euclidian distance in order to compute the similarity between two separate malware P2P flows, and we set different clustering thresholds for TCP and UDP flows. In fact TCP flows have a higher offset size because of their larger TCP headers and their higher number of packets compared to UDP flows, due to TCP handshake and TCP Acks. Hence, we empirically set TCP and UDP thresholds to 100 and 20 respectively. They characterize the minimal flow size (400 and 40) to the minimal packets number (4 and 2) ratio for non-empty TCP and UDP flows.

PeerViewer builds clusters of flows by comparing P2P flows that we extracted from our malware training set. P2P flows for a malware sample $\mathcal{M}_\alpha$ may span on multiple clusters. Each cluster contains flows that have similar network features, and so they are likely to carry the same P2P signaling activity and protocol, but that are triggered by different malware samples. We further build a P2P footprint for each malware in our dataset. It specifies the rate of flows for a given malware within each P2P flow cluster provided by our system. In other terms, a malware footprint $\mathcal{F}_\alpha\{c_i\}_{i=1}^{m}$ is an $m - arry$ vector of size m, where m is the number of P2P flow clusters. Attribute $\mathcal{F}_\alpha\{c_k\}$ for malware $\mathcal{M}_\alpha$ corresponds to the fraction of P2P flows for $\mathcal{M}_\alpha$ within c_k, with respect to the total number of P2P flows in the network trace of $\mathcal{M}_\alpha$. Hence, attributes of a malware footprint are real values in the $[0, 1]$ interval, with $\sum \mathcal{F}_\alpha\{c_i\}_{i=1}^{m} = 1$.

3.3 P2P Malware Classifier

The classifier module builds clusters of malware that implement the same P2P protocol and belong to the same P2P botnet family. It groups together malware that has similar P2P footprints so they are likely to use the same P2P botnet topology. We use the unsupervised hierarchical clustering algorithm to obtain P2P malware families. It builds different families of malware according to the initial malware training set. As opposed to incremental K-means, the hierarchical clustering algorithm does not require a threshold for adding a new cluster. In fact, it is possible to set a threshold for flow clustering because of the ground truth provided by malware network traces. However, malware clustering does not have a reliable ground truth as AV solutions usually provide conflicting malware classifications (section 4 provides a detailed comparison between our system and AV signatures). Hierarchical clustering creates a dendrogram where leaf nodes are elementary P2P malware, and the root node is a set of all malware samples. We use the Davies-Bouldin index [8] to find the optimal cut in the dendrogram, and thus to obtain our set of malware P2P families. Each family includes a set of malware aggregated within a single node in the dendrogram.

Malware families provided by our system are used to classify unknown malicious code on-the-fly while executed in a dynamic analysis environment. We build a *one-class classifier* for each family of malware provided by our system [17]. It characterizes the P2P footprints for malware samples within this family. During detection, PeerViewer collects the network trace for unknown malicious code running in a sandbox. It applies P2P filtering and flow clustering, which provide clusters of P2P flows triggered by a given malware sample. These clusters constitute a network footprint that we use to associate malicious code with a known malware family. PeerViewer tests this footprint against the one-class classifiers for all malware families in the training set. It associates a malicious code with a given malware family when its P2P footprint matches the one-class classifier of this family. Yet PeerViewer is unable to classify a malicious code when its P2P footprint matches any, or more than a single malware family. It notifies the security analysit of a new or unknown P2P malware, so it can be submitted to a deeper analysis.

4 Experimentation

This section presents the malware dataset that we used in order to build and validate our system. It describes the design of our experiments, including tests and results of PeerViewer when applied to the malware dataset at our disposal.

4.1 P2P Malware Dataset

In order to validate our system, we obtained malware samples from a security company that implements its own collection and analysis platforms. Our dataset includes thirty minutes of network traffic for malware executed in a dynamic analysis environment. Malware was granted open access to its command and control infrastructure, including updates and command execution. Malware traffic was provided in separate pcap files. In fact we do not have access to malware binaries, but only to their network traces, associated each with the md5 hash for the originating malicious code. The dataset at our disposal includes network traffic for almost twenty thousand distinct malware samples collected during a three months period, between March and June 2012.

We use the virusTotal API in order to qualify P2P malware in our dataset and to validate the results of our experimens. We searched in virusTotal for md5

Table 1. Malware samples by families of malware

Malware Family	P2P protocol	Samples	Training	Evaluation	Flows	P2P flows
Sality v3 and v4	Custom	386	335	51	105178	28071
Zeus v3	Kademlia	35	27	8	8523	4227
ZeroAccess	Custom	33	24	9	14328	5676
Kelihos	Custom	41	34	7	12906	4440
TDSS	Kademlia	40	30	10	17680	4368

hashes in our dataset that match with existing P2P malware families. In order to obtain a valid ground truth for our experiments, we pick-up network traces only when their md5 labels match with more than 10 known signatures for the same P2P malware family in virusTotal. Note that AV scanners usually assign conflicting signatures for the same malware sample. For example, a same Sality malware has a `kaspersky` signature of `Virus.Win32.Sality.aa` and a `trendMicro` signature of `PE_SALITY.BU`. Therefore, we build our ground truth malware classes by matching keywords associated with known P2P malware families, as shown in table 1. We further compare in section 4.3 our malware families with signatures provided by three distinct AV scanners. Table 1 summarizes the six distinct P2P malware families that we identified within our malware dataset. Although 60% of our dataset consists of Sality (v3 and v4), it also includes significant flows for other P2P malware families. Yet we aim at experimentally validating three properties of PeerViewer using our P2P malware dataset.

First, PeerViewer identifies small malware families into a larger set of P2P malware. For instance, it accurately identifies the zeroAccess family, although it only constitutes 5% of our initial learning set. Second, PeerViewer identifies variants of the same malware family that have different implementations of P2P protocol. We validate this property using the example of Sality versions 3 and 4, that were correctly classified by our system. Third, PeerViewer separates families that use the same P2P protocol, but having different P2P signaling activity. Our system efficiently classifies samples of Zeus v3 and TDSS malware, although they are based on the same kademlia protocol.

We tested our P2P filter against the initial malware dataset, using the ground truth provided by P2P malware signatures in table 1. The DNS filter reduces up to the third the initial number of malware samples. Indeed, it cannot discard all non-P2P malware because there is multiple other reasons for malware to operate outside the DNS system, including hard coded C&C addresses, scan attempts and spam. Yet the flow size filter had little impact with only few flows being discarded, and that mostly carry malware spam activities. We believe this is mainly because of the short dynamic analysis time (30 minutes), which was not long enough for malware to trigger P2P data flows. On the other hand, the P2P filter uses two distinct thresholds, that are associated with the failed connection (τ_{fc}) and AS-based (τ_{as}) filters. We experimentally configured the values for these thresholds using our ground truth malware dataset. We were able to achieve 100% detection accuracy for values of τ_{as} in the interval $[0.1, 0.3]$

Table 2. Examples of malware P2P flow clusters detected by PeerViewer

Clstr Id	**Nb of flows**	**Tr. proto**	**P2P proto**	**P2P activity**	B_s	B_r	Pkt_s	Pkt_r
1	205	UDP	Gnutella	Query	35	130	1	1
2	937	UDP	Custom (Sality P2P)	Peer exchange	34	610	1	1
3	11944	UDP	Custom (Sality P2P)	Peer announcement	20	600	1	1
4	1674	UDP	uTorrent	find_node	450	970	3	3
5	1427	UDP	uTorrent	find_node	300	630	2	2
6	1164	UDP	Custom (Zeus P2P)	version request	200	645	2	2
7	5778	TCP	Gnutella	push	367	0	4	4
8	504	TCP	Gnutella	push	882	0	8	7

and τ_{fc} in the interval $[0.14, 0.6]$. We thus conservatively set these thresholds to the values 0.2 and 0.3 respectively, using our ground truth dataset in table 1.

We collected a total number of 541 malware samples that are classified into six distinct P2P malware families, as shown in table 1. In the remaining of this section, we validate our malware classification module using the set of P2P malware samples extracted from our dataset. We compare PeerViewer with P2P malware families provided by AV scanners. We also demonstrate its ability to efficiently classify P2P malware with high precision and recall.

4.2 Malware Classification

We split the dataset at our disposal into two separate groups. The first one includes 85% of our P2P malware learning set, and that we used to build P2P malware families. The second group includes the remaining 15% malware samples, and that we further used to test and validate our system. For the purpose of this paper, we randomly extracted the malware validation set from each P2P malware family using the ground truth families in table 1. Our validation set thus included 85 samples extracted from all six P2P malware families. We use the remaining 450 malware samples in order to build our malware classification system. The fourth column in table 1 summarizes the number of samples for each malware family that we use to build our malware classification system.

PeerViewer builds clusters of flows in order to group together malware flows that implement the same P2P protocol and signaling activity. It applies incremental k-means to the entire set of malware P2P flows, using the features vector f_α presented in section 3. The flow clustering module, applied to the 450 malware samples in our dataset, provided a total number of 28 P2P flow clusters, including 22 clusters of UDP flows and 6 clusters of TCP flows. Because of space limitations, table 2 illustrates only examples of network features for a subset of 8 P2P flow clusters identified by PeerViewer.

As shown in table 2, different signaling activities for the same P2P malware were indeed classified into different clusters. For example, clusters 2 and 3 in table 2 included two separate signaling messages (Peer exchange and Peer announcement) for the same Sality malware. As shown in table 2, Sality has different average request sizes for its two P2P signaling activities (34 vs 20), and so they

Table 3. Examples of P2P footprints for Zeus v3 and ZeroAccess families

Malware	Cltr 1	Cltr 2	Cltr 3..15	Cltr 16	Cltr 17..24	Cltr 25	Cltr 26	Cltr 27	Cltr 28
Zeus 1	**0.07**	0	0	**0.93**	0	0	0	0	0
Zeus 2	**0.085**	0	0	**0.914**	0	0	0	0	0
Zeus 3	**0.03**	0	0	**0.97**	0	0	0	0	0
Zeus 4	**0.037**	0	0	**0.962**	0	0	0	0	0
Zeus 5	**0.071**	0	0	**0.928**	0	0	0	0	0
Zeus 7	**0.098**	0.02	0	**0.87**	0	0.01	0	0	0
ZA 1	0.035	0.014	0	0	0	**0.577**	0.04	**0.3**	0
ZA 2	0.078	0.022	0	0	0	**0.592**	0	**0.3**	0.011
ZA 3	0.102	0.011	0	0	0	**0.606**	0.02	**0.27**	0
ZA 4	0.019	0.015	0	0	0	**0.62**	0.06	**0.29**	0.013

were classified into different clusters. Clusters 1, 7 and 8 provide yet another example for the Gnutella protocol. We obtained separate clusters for the query and push signaling activities of the same Gnutella P2P protocol. They respectively use UDP and TCP protocols, and they have different network features so they were classified into separate clusters. Note that we may still obtain clusters that implement the same P2P signaling activity and protocol (e.g. clusters 4 and 5 for the same uTorrent protocol). Nonetheless, these clusters show different P2P network features that characterize different implementations of the same P2P protocol by different malware families.

We use the 28 flow clusters in order to build P2P footprints for malware in our dataset. Malware footprints indicate the proportion of P2P flows for a given malware that belong to each of the 28 P2P flow clusters. Due to space limitations, table 3 illustrates examples of P2P footprints from only two P2P malware families, Zeus v3 and ZeroAccess. As shown in this table, malware of the same family has almost identical P2P footprints and so it would be grouped within the same clusters. For example, malware of the Zeus v3 family has almost all of its P2P signaling flows in cluster 16, while the few remaining flows belong to cluster 1. On the other hand, malware of the zeroAccess family has almost a third of its P2P signaling flows in cluster 27, and the remaining two thirds in cluster 25. These two malware families would be clearly separated into two clusters by the malware classifier.

The classifier module uses malware footprints in order to build families of P2P malware. We implement the hierarchical clustering algorithm using Python, and we use the Davies-Bouldin index to obtain the optimal set of clusters. The malware classifier module identified a total number of 8 clusters, associated with 8 distinct P2P malware families. We validate our P2P malware families using the ground truth classification in table 1.

Six P2P malware clusters were clearly associated with each of the six malware families in table 1. In fact all Zeus v3, ZeroAccess and Kelihos malware samples were classified into separate clusters respectively. We thus consider our clusters to characterize the P2P network footprint of these distinct malware families. On the other hand, samples of Sality malware were split into two separate clusters, including 295 and 37 samples in each cluster respectively. These clusters are likely to include malware that respectively belong to versions v3 and v4 of the Sality family. Yet we couldn't validate this assumption using our ground truth in table 1 because of the conflicting AV signatures for versions of the Sality malware. Therefore, in order to refine our ground truth, we checked the update time for AV signatures that were matching each of the malware md5 hashes associated with the Sality malware. We would expect samples for the version v4 of this malware to be more recent *in general* than samples of version v3. We admit that AV update times do not formally validate our classification because we cannot rule out the possibility of newer malware samples implementing P2P protocol for version 3 of this malware. However, signature update times still provide evidence of different version implementations for this same malware family. Yet we observed that 80% of malware in the smaller Sality P2P cluster has newer update times than all other samples in the larger P2P cluster. We believe this is a clear evidence of two families of the Sality malware, that we associate with versions 3 and 4 of this malware family. In fact, versions v3 and v4 of the Sality malware have different implementations of their P2P signaling protocols, and so AV signatures cannot correctly classify these two malware families based on their system behavior. PeerViewer thus offers a complementary approach that classifies P2P malware based on its network-level behavior, which cannot be easily characterized by host-based signatures.

Finally, we obtained two additional clusters, both including two malware samples that belong to different malware families in table 1. These are clearly outliers and so they were misclassified by our system. PeerViewer was indeed able to correctly classify 446 out of 450 malware samples in the initial training set. It clearly outperforms current AV signatures as it achieved near 0.8% misclassification rate.

4.3 Comparison with AV Signatures

This section analyzes the validity of our P2P malware families by comparing them with signatures from three AV scanners, including `McAfee`, `kaspersky` and `Trend Micro`. In fact, our system proposes a behavioral approach that classifies P2P malware on the fly while executing in a dynamic analysis environment. We need to verify the cohesion of our P2P malware families using a learning set of known and already qualified malware dataset. For each malware family created by our system, we collect AV signatures for all samples of this family. We compute the precision and recall of our system in order to validate the consistency of our malware classification with respect to all three AV scanners. Our experiments prove the ability of PeerViewer to accurately classify P2P malware using only network level information, with no *a-priori* knowldge about the system behavior of malware.

Table 4 compares malware families provided by our system with signatures from three AV scanners. As shown in this table, AV scanners assign different signatures for samples of the same malware families. These signatures usually constitute different aliases for the same malware family. In order to have common evaluation criteria for all three AV scanners, we used the spyware remove

Table 4. Comparison with `kaspersky`, `McAfee` and `TrendMicro` signatures

Family Id	Samples	Kaspersky	McAfee	TrendMicro
1	295	**win32.Sality: 193** **Win32.Spammy: 29** Unknown: 23	**W32/Sality: 219** Downloader-CPY: 22 Unknown: 4	**PE_SALITY: 223** WORM_KOLAB: 9 Mal_Odra-5: 2 Unknown: 11
2	27	**win32.Zbot: 25** Unknown: 2	**PWS-Zbot**: 27	**Tspy_Zbot**: 27
3	24	**Win32.Sefnit: 17** **Win32.ZAccess: 7**	**Sefnit: 24**	**Troj_Kazy: 13** Troj_Sirefef: 7 Unknown: 4
4	32	**Win32.Kelihos: 27** unknown: 5	**Win32/Kelihos: 23** GenericBackDoor.xf: 8 unknown: 1	**TROJ_FAKEAV: 29** TROJ_INJECTER: 3
5	37	**win32.Sality: 37**	**W32/Sality: 37**	**PE_SALITY: 37**
6	30	**Win32.TDSS:19** Win32.FakeAV: 11	**FakeAlert-JM: 26** **Trojan.Alureon:4**	**BKDR_TDSS: 30**
7	2	win32.Sality, win32.killAV	Win32/Nimnul, win32/Zbot	PE_fujacks, PE_nimnul
8	2	win32.Sality, unknown	unknown: 2	PE_fujacks, PE_down

	Kaspersky	McAfee	TrendMicro
Precision	83.16%	88.45%	86.5%
Recall	90.8%	89.31%	94.85%

Fig. 2. Precision and recall against the three AV scanners

	Sality v3	Sality v4	ZA	kelihos	TDSS	Zeus v3
Accuracy	99.3%	99.1%	94.2%	95%	98%	100%

Fig. 3. Classification accuracy by malware family

website[1] in order to associate all aliases of the same malware families. For example, the signature `win32.spammy` for the first malware family in table 4 is identified by spyware remove as a `kaspersky` alias of `spammer.sality.a`, and so we consider it as part of the sality family.

Figure 2 summarizes the classification accuracy and recall of PeerViewer against the three AV scanners. Classification accuracy is computed as the average precision rate for all six P2P malware families identified by PeerViewer. We introduce the precision rate for a P2P malware family as the ratio of malware samples that have the same predominant AV signature with respect to the total number of samples in this family. The classification recall is computed the same as for the precision rate, excluding samples that are unknown for AV scanners. As in figure 2, PeerViewer has almost stable precision and recall against all three AV scanners. It enhances by at least 11.5% the malware classification for AV scanners (in case of McAfee which provides the highest precision rate), based on our ground truth in table 1. It also differentiates samples of the same malware family that implement different variants of the same P2P protocol, as for the sality malware which is indeed represented by the same signature by all three AV scanners. Yet it replaces current AV signature aliases with a common behavioral malware classification, as in the example of the third malware family provided by PeerViewer in table 4. The latter provides a common classification for multiple aliases of the same zeroAccess malware family, including `win32.ZAccess`, `win32.sefnit`, `troj_Kazy` and `troj_Sirefef` aliases.

[1] `http://www.spywareremove.com/`

4.4 Classification and Detection

This section demonstrates the detection phase of PeerViewer, which classifies P2P malware on-the-fly during dynamic analysis. We implement the cross-validation method that consists of extracting an evaluation dataset prior to building malware classifiers, and then to use this dataset in order to test and validate our classifiers. We reiterated the cross-validation process using different evaluation sets, each time randomly extracting 15% of our malware dataset before we build our one-class classifiers. In order to guarantee the soundness of our experiments, our evaluation set had always the same malware composition, as shown in the fifth column of table 1.

We apply the P2P flow filter and we build clusters of P2P flows using the network traces for each sample in our malware validation set. Then we build a P2P footprint for each sample using its P2P flow clusters. We use malware footprints as input to the one-class classifiers for each of our six malware families. Our system achieved near 97.6% classification accuracy, based on the ground truth classification in table 1. The detailed results of our experiments for each malware family are illustrated in the table of figure 3.

Samples of Zeus v3 and TDSS malware families were accurately classified with almost no false positives. False positives in case of Sality malware were all due to mis-classifications between the different versions of this family. Note that 100% of Sality malware in our dataset was correctly classified by PeerViewer, and almost 99.2% of these samples were classified with the appropriate version of this family. In fact we couldn't formally validate our classification of Sality versions v3 and v4 because AV scanners do not constitute a reliable ground truth. Hence, we used update times for AV detection signatures in order to separate between different versions of Sality malware. On the other hand, PeerViewer has correctly classified only 94.2% of kelihos malware mostly because of the small number of samples in our learning set. Yet PeerViewer outperforms most AV scanners with an overall classification accuracy of 97.6%, while only relying on network features with no need of malware binary analysis.

5 Discussion

PeerViewer classifies malware using statistical network features such as flow size, IP distribution and traffic rate. First, it classifies flows for a given malware sample into categories where they implement the same signaling activities. Then it builds malware families using similarities in their P2P network footprints. PeerViewer would be thus unable to accurately classify P2P malware that modifies its P2P communication rounds, contacts a larger set of peers, or uses random paddings in its P2P traffic. These maneuvers modify statistical consistency in malware P2P flows and so it makes malware classification more difficult using our features. Although they are technically possible, these techniques require a malware developper to modify its P2P C&C toolkit. They also increase overhead and reduce botnet stability, which makes botnet management more difficult. Yet botnets that adopt these techniques would no longer be able to dissimulate within benign P2P flows, and so they will be exposed to other malware detection techniques.

On the other hand, PeerViewer classifies malware that uses P2P protocols only as a primary C&C channel. In fact, malware may use P2P protocols as a failover mechanism in case where it cannot access its primary C&C channel. This malware does not trigger P2P flows during analysis, and so it would not contribute to building P2P malware classifiers. Authors in [21] propose an approach that detects primary C&C channels during malware dynamic analysis. This approach dynamically intercepts primary C&C channels and forces malware to engage in a failover strategy. Using techniques such as [21] enables to trigger P2P failover strategies, and so PeerViewer will be able to take these into account during its processing of malware P2P flows. Nonetheless, these techniques apply during malware dynamic analysis and so they are out of scope in this study.

Future work will explore techniques to integrate PeerViewer into a more comprehensive malware detection system. In fact, PeerViewer classifies malware samples with a high detection accuracy. Nonetheless, it is yet unclear how PeerViewer would be a able to separate P2P flows triggered by multiple P2P applications running on the same terminal. Although it is out of scope in this paper, we experimented with PeerViewer in order to detect P2P infected nodes within live network traffic. PeerViewer efficiently *detects and characterizes* infected nodes when they do not concurrently implement other benign P2P applications. Therefore, future work will adress this issue by proposing appropriate methods that tell apart malware and benign P2P applications when they are running on the same network terminal.

6 Related Work

Several approaches detect P2P malware through behavioral analysis of network traffic, without deep packet inspection. They usually propose a binary classification of P2P nodes, that is whether being infected or benign [6,9,11,19,27,28]. Yet there is only few approaches that build families of P2P malware based on its P2P network behavior [12,24].

The first category includes solutions such as BotTrack [9], BotMiner [11] and BotGrep [19]. They correlate network flows and detect P2P bots based on their overlay C&C topologies. First, they build clusters of terminals and isolate groups of hosts that form P2P networks. Then they separate malicious P2P groups using lists of infected P2P nodes provided by sources such as honeypots. These techniques mostly rely on IDS signatures and IP blacklists to detect P2P bots. However, botnet activity is becoming stealthier and difficult to detect using IDS signatures, thus limiting the coverage of these solutions. Bilge et al. propose an alternative approach that detects botnets using large scale netflow analysis [6]. It observes traffic at large ISP networks and detects botnets through the coordinated activity for groups of infected nodes. However, this approach detects only centralized botnet architectures, and cannot accurately detect distributed P2P botnets. Another trend of research aims at detecting infected P2P bots inside a given network perimeter [27,28]. These studies propose to first discard non-P2P traffic using heuristics such as DNS traffic and failed connection attempts. They build groups of P2P nodes that have the same network behavior or that connect to a common set of remote IP addresses. Further they compute a similarity degree between network nodes in order to detect those that are likely to be part of a same botnet. However, these studies can only detect P2P bots when there is multiple infected nodes inside the same network perimeter. Yet they only provide a binary classification, without being able to identify a common malware family or a given P2P protocol.

The second category classifies P2P flows and identifies specific P2P protocols or applications [12,24]. PeerRush [24] uses features such as inter-packet delays and flow duration in order to classify P2P applications. These features achieve good detection accuracies against benign P2P applications. However, it is not clear how they will contribute to classifying P2P botnets. For exemple, inter-packet delays can be easily evaded and these are weak indicators of P2P activity. Yet PeerRush deals with all P2P signaling flows as a whole. It does not classify flows according to their embedded message types and the rate of each signaling activity. PeerViewer thus provides a better alternative as it builds specific malware P2P footprints that take it account the P2P signaling rounds and categories of message types. On the other hand, Hu et al. [12] use flow statistics to build behavior profiles for P2P applications. They experimented only with two P2P applications (PPLive and BitTorrent), and did not consider malicious P2P activity. Yet, they do not separate P2P control and data traffic. In fact data flows do not clearly characterize P2P botnet C&C activity as they depend on the content being shared. PeerViewer thus classifies P2P signaling flows and use only these as a basis for P2P botnet classification.

7 Conclusion

This paper presented PeerViewer, a system that automatically classifies P2P malware based on its P2P network behavior. It does not use system-level information, nor does it use flow content signatures during its processing of malware traffic. Indeed PeerViewer classifies P2P flows for a specific malware into categories where they implement the same P2P signaling activity. It further builds a footprint that characterizes the P2P network behavior of malware, using the different categories of signaling flows triggered by this malware. To the best of our knowledge, PeerViewer is the first to propose a fully behavioral approach that detects *and* classifies P2P malware into specific malware families. We tested our system against signatures of malware families provided by several anti-virus solutions. Experimental results prove our ability to accurately classify P2P malware with a very low false positives rate.

References

1. Zeus tracker, `https://zeustracker.abuse.ch/` (accessed at June 2013)
2. Anubis: Analyzing unknown binaries (2011), `http://anubis.iseclab.org`
3. Blue coat - exposing malnet strategies and best practices for threat protection. In: 2012 Web Security Report (2012)
4. Aberer, K., Hauswirth, M.: An overview on peer-to-peer information systems. In: Proceedings of the 4th Workshop on Distributed Data and Structures (2002)
5. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable behavior-based malware clustering. In: Proc. 19th NDSS (2009)
6. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis. In: Proc. 18th NDSS (2011)
7. Bolla, R., Canini, M., Rapuzzi, R., Sciuto, M.: Characterizing the network behavior of p2p traffic. 4th International Workshop on QoS in Multiservice IP Networks (2008)
8. Davies, D.I., Bouldin, D.W.: A cluster seperation measure. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (1979)
9. François, J., Wang, S., State, R., Engel, T.: Bottrack: Tracking botnets using netflow and pagerank. In: Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C. (eds.) NETWORKING 2011, Part I. LNCS, vol. 6640, pp. 1–14. Springer, Heidelberg (2011)
10. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B.: Peer-to-peer botnets: Overview and case study. In: Proceedings of USENIX HotBots (2007)
11. Gu, G., Perdisci, R., Zhang, J., Lee, W.: Botminer: Clustering analysis of network traffic for protocol and structure independent botnet detection. In: SSP (2008)
12. Hu, Y., Chiu, D.-M., Lui, J.C.S.: Profiling and identification of p2p traffic. In: Computer Networks, vol. 53, pp. 849–863 (2009)
13. Jacob, G., Hund, R., Kruegel, C., Holz, T.: Jackstraws: Picking command and control connections from bot traffic. In: 20th Usenix Security Symposium (2011)
14. Kapoor, A., Mathur, R.: Predicting the future of stealth attacks. In: Virus Bulletin (2011)
15. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is p2p dying or just hiding? In: IEEE GLOBECOM, vol. 3, pp. 1532–1538 (2004)
16. Karagiannis, T., Broido, A., Brownlee, N.: k claffy, and M. Faloutsos. File-sharing in the internet: A characterization of p2p traffic in the backbone. In: UC Riverside technical report (November 2003)
17. Khan, S.S., Madden, M.G.: A survey of recent trends in one class classiffication. In: Coyle, L., Freyne, J. (eds.) AICS 2009. LNCS, vol. 6206, pp. 188–197. Springer, Heidelberg (2010)
18. Lua, C.-N., Huang, C.-Y., Lina, Y.-D., Lai, Y.-C.: Session level flow classification by packet size distribution and session grouping. International Journal on Computer Networks 56, 260–272 (2012)
19. Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., Borisov, N.: Botgrep: Finding p2p bots with structured graph analysis. In: Proc. 19th USENIX Security (2010)
20. Neugschwandtner, M., Comparetti, P.M., Jacob, G., Kruegel, C.: Forecast: skimming off the malware cream. In: 27th Annual Computer Security Applications Conference, ACSAC (2011)
21. Neugschwandtner, M., Comparetti, P.M., Platzer, C.: Detecting malware's failover c&c strategies with squeeze. In: Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC (2011)

22. Perdisci, R., Lee, W., Feamster, N.: Behavioral clustering of http-based malware and signature generation using malicious network traces. In: USENIX Symposium on Networked Systems Design and Implementation (2010)
23. Porras, P., Saidi, H., Yegneswaran, V.: Conficker c p2p protocol and implementation. Technical report, Computer Science Laboratory, SRI International (2009)
24. Rahbarinia, B., Perdisci, R., Lanzi, A., Li, K.: Peerrush: Mining for unwanted p2p traffic. In: Rieck, K., Stewin, P., Seifert, J.-P. (eds.) DIMVA 2013. LNCS, vol. 7967, pp. 62–82. Springer, Heidelberg (2013)
25. Rieck, K., Holz, T., Willems, C., Düssel, P., Laskov, P.: Learning and classification of malware behavior. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 108–125. Springer, Heidelberg (2008)
26. Tenebro, G.: W32.waledac threat analysis. In: Symantec Technical Report (2009)
27. Yen, T.-F., Reiter, M.K.: Are your hosts trading or plotting? telling p2p file-sharing and bots apart. In: 30th Conf. Distributed Computing Systems (2010)
28. Zhang, J., Perdisci, R., Lee, W., Sarfraz, U., Luo, X.: Detecting stealthy p2p botnet using statistical traffic fingerprints. In: Proc. 41st DSN (2011)