Waves of Malice: A Longitudinal Measurement of the Malicious File Delivery Ecosystem on the Web

Colin C. Ife[★], Yun Shen[†], Steven J. Murdoch[★], and Gianluca Stringhini[‡]

[★]University College London, [†]Symantec Research Labs, [‡]Boston University
{colin.ife,s.murdoch}@ucl.ac.uk,yun_shen@symantec.com,gian@bu.edu

ABSTRACT

We present a longitudinal measurement of malicious file distribution on the Web. Following a data-driven approach, we identify network infrastructures and the files that they download. We then study their characteristics over a short period (one day), over a medium period (daily, over one month) as well as in the long term (weekly, over one year). This analysis offers us an unprecedented view of the malicious file delivery ecosystem and its dynamics.

We find that the malicious file delivery landscape can be divided into two distinct ecosystems: a much larger, tightly connected set of networks that is mostly responsible for the delivery of potentially unwanted programs (PUP), and a number of disjoint network infrastructures that are responsible for delivering malware on victim computers. We find that these two ecosystems are mostly disjoint, but it is not uncommon to see malware downloaded from the PUP Ecosystem, and vice versa. We estimate the proportions of PUP-to-malware in the wild to be heavily skewed towards PUP (17:2) and compare their distribution patterns. We observe periodicity in the activity of malicious network infrastructures, and we find that although malicious file operations present a high degree of volatility, 75% of the observed malicious networks remain active for more than six weeks, with 26% surviving for an entire year. We then reason on how our findings can help the research and law enforcement communities in developing better takedown techniques.

ACM Reference Format:
Colin C. Ife[★], Yun Shen[†], Steven J. Murdoch[★], and Gianluca Stringhini[‡]. 2019. Waves of Malice: A Longitudinal Measurement of the Malicious File Delivery Ecosystem on the Web. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9-12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, Article 4, 13 pages. https://doi.org/10.1145/3321705.3329807

1 INTRODUCTION

Malware delivery has undergone an impressive evolution since its inception in the 1980s, moving from being an amateur endeavor to a well-oiled criminal business where, rather than being a cottage industry, it became specialized with skills traded on underground markets. In pursuing larger and larger populations of victims, malware authors moved from using floppy disks as their infection vector [13] to delivering malware as attachments in spam emails [28], enticing users into opening them through social engineering [23]. Eventually, malware authors started compromising user machines without the need for explicit user interaction, by exploiting vulnerabilities in the victim browser once it visited a malicious web page (a so-called *drive-by download attack* [24]). To streamline the exploitation process, miscreants developed so-called *exploit kits*, which are software packages that contain exploits for multiple software configurations and can infect as many victims as possible by delivering the correct exploits based on the victim's software configuration [12]. Miscreants also developed *pay-per-install* (PPI) schemes [7], in which a specialized actor sets up a network of infected computers (commonly known as a botnet [4]) that are later rented out to other criminals.

More recently, researchers uncovered a parallel economy that shares many traits with the malware ecosystem, while being primarily controlled by different actors: *potentially unwanted programs* (PUPs) [17, 18, 32]. This category of programs includes software that is not willingly installed by users and that typically is an annoyance more than a direct threat to the safety of victims – examples include adware and browser toolbars. While malware delivery mostly happens through drive-by downloads, PUP victims are usually tricked into installing a downloader through social engineering [17]. After such a downloader is installed, additional components are dropped through a PPI service [32]. For this reason, files that belong to PPI services are commonly known as *droppers*.

Previous research has suggested that, although mostly disjoint, a consistent number of malicious actors (e.g., PPI operators) serve both malware and PUP samples. Kwon *et al.* [20] show that 36.7% of the droppers that they observed downloaded both malware and PUPs. Despite this finding, many questions remain unanswered on the structure, workings, and dynamics of malware delivery networks. What does the malicious file delivery network look like? Are there differences in the network structure of infrastructures that solely download malware, PUP, or both? How do these infrastructures change over time? Answers to such questions could help the security community better understand this malicious ecosystem, and could expose weak points in these infrastructures for takedowns.

In this paper, we adopt a data-driven approach to provide a *longitudinal characterization* of the malware and PUP delivery ecosystem on the Internet. First, we collect 129 million download events from millions of real users who downloaded unwanted software over one year. Our data contains information on the files downloaded,

AsiaCCS'19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329807

on the network servers that they were downloaded from, and on the file that initiated the download. We subsequently model these download relations as a graph and apply graph analysis techniques to identify the related network and file components. We then look at the types of files that these components download, and study their temporal behavioral characteristics over a short period (one day) as well as in over a medium period (every day for a period of one month) and the long term (one day a week for a period of one year).

Overview of results. We find that the malicious file delivery landscape can be partitioned into two disjoint ecosystems: a tightly connected set of network infrastructures that are mostly responsible for downloading PUPs, and a set of isolated infrastructures that are mostly responsible for downloading malware. We also find that the PUP Ecosystem is stable over the long-term (i.e., one year). In raw numbers, the PUP Ecosystem is responsible for 80% of suspicious file downloads worldwide. Although previous research found that PUPs are pervasive in the wild [17], we are the first ones to compare the prevalence of these type of malicious files to malware. We estimate the proportion of PUP-to-malware in the wild – roughly 5:1 in # of SHA-2s, and 17:2 in # of downloads – and we analyze the characteristics and distribution patterns of their ecosystems. Confirming results from previous work [20], we show that these delivery infrastructures are often not responsible for delivering a single type of malicious files (i.e., PUP or malware), but instead often deliver both. Over time, we observe the activity patterns of distribution infrastructures and their lifespans. This paper provides the security community with an unprecedented view of the characteristics of the malware and PUP delivery ecosystems. Also, we provide a methodology, with initial results, that identifies elements (IP addresses, domain names) in a delivery infrastructure that do not change over time. These can be used to direct takedown efforts towards those elements that are not volatile and therefore could have an impact if taken down.

2 DATASET

In this paper, we leverage a dataset from Symantec's data sharing platform. This allows us to access the anonymized data collected by its anti-virus and intrusion detection/prevention products on millions of end-hosts around the world. These datasets are collected from users who explicitly opt-in for the data sharing program and does not include personally identifiable information (PII).

Our dataset contains download activity information from real hosts for one year between 1 October 2015 and 29 September 2016. The users that have explicitly opted into this company's data-sharing program periodically report metadata information on the binaries that they download. This dataset offers rich information regarding the time at which a binary is downloaded, which server it is downloaded from, and which program initiated the download activity. If a malicious file `4.exe` is downloaded from a website `http://example.org/landing/page`, for example, the data will contain information about the file, as well as the website URL that `4.exe` was downloaded from, and the IP address of the server `93.184.216.34`. Note that if this malicious file `4.exe` downloaded other malicious files, we define `4.exe` as a *dropper*. Additionally, if

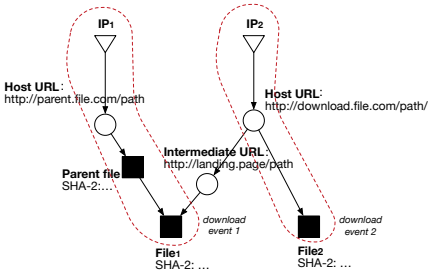

Figure 1: Example of a download graph with two download events highlighted.

a dropper malware sample downloads a second malicious file, the dataset will record information about both the server that the file is downloaded from and the dropper that initiated the download.

To be more precise, for each download event, our dataset contains the following information: the timestamp of the download event, the name, SHA-2 (256 bits) and size (in bytes) of the downloaded file, the landing page URL and IP address of the server the file was downloaded from, the SHA-2 of the parent file which initiated the download, and the referral URL that this parent was originally referred from (if available). We collect data on a daily basis in October 2015 (31 days) and, from then on, every Thursday on a weekly basis from November 2015 to September 2016 (47 days). In total, the dataset contains 129 million download events consisting of 21,398,564 unique binaries. These binaries are downloaded from 12,394,454 unique URLs, hosted on 557,429 unique IPs. After IP filtering (see Section 3.1), these are reduced to 21,388,521 unique binaries, 12,390,735 unique URLs, and 553,812 unique IPs.

Note that this paper only focuses on malicious file downloads. To this end, we leverage the reputation score that Symantec associates to files. We discard any file that has a high reputation score, and only keep files that are either known to be malware or PUP (e.g., using the ground truth maintained by the security company) or confirmed as malicious by VirusTotal. Note that we consider a file to be malicious if at least one of the top five AV vendors by market share (in no particular order, Avast, AVG, Avira, Microsoft, and Symantec) and a minimum of two other AVs detect it as malicious. Nelms *et al.* [22] used a similar technique.

3 METHODOLOGY

This study leverages two stages of analysis: (i) a 24-hour snapshot analysis, and (ii) a longitudinal analysis. In the first stage, we group related hosts and files observed over 24 hours, and map the network infrastructures involved in the delivery of malicious files. In the second stage, we track the evolution and behaviors of these infrastructures. In this section, we describe these stages in detail.

3.1 Snapshot Analysis

The data processing pipeline for a 24-hour snapshot is as follows: i) IP filtering, ii) building the graph, iii) separating components, and iv) file classification.

IP Filtering. Since this dataset presents a global outlook on download data, files that appear to be generated from the host machine

(localhost) or private IP addresses could be incorrectly inferred as being part of the same infrastructure. Consequently, we remove IPv4/v6 addresses that are not valid for public use on the Internet [2]. As a result, the graph-building stage ignores files and URLs that are downloaded *only* from and hosted on these IP addresses.

Building the Graph. For each day of our analysis, we take the download events that happened during that day to build a directed *download graph*, which is the basis of our analysis. In particular, we combine the information contained in the download event in a directed graph $G = (V, E)$, where V is the vertex list of file nodes and network nodes. Figure 1 shows an example of this graph structure. Network nodes are host URLs, referrer URLs, and IP addresses. E is the respective edge list. A directed edge (v, w) indicates that node v is related to node w under one of the following conditions: (i) a parent-file v drops a file w; (ii) a file w is downloaded from (by precedence as indicated in Figure 1) a referrer URL (landing page), a host URL, or an IP address v; (iii) a parent-file w is downloaded from a host URL v; (iv) a host URL v is associated with a referrer URL w; and (iv) an IP address v is associated with a host or referrer URL w. Take *download event 1* in Figure 1, for example, $File_1$ is dropped by *Parent file*, which was downloaded from host URL `http://parent.file.com/path`, hosted on IP address IP_1. In *download event 2*, it is straightforward to see that $File_2$ was downloaded from host URL `http://download.file.com/path/` hosted on IP_2. These two disconnected graphs are connected when we observe the third download event when $File_1$ was downloaded via referrer URL http://landing.page/path leading to host URL http://download.file.com/path/. All downloaded files with the same SHA-2 are assigned to the same file node.

Separating Components. The primary step towards attributing files, hosts, and their activities to actors is to separate the directed download graph into *weakly (undirected) connected components*, or *CCs*. This enables us to identify distribution networks of files and hosts that have direct interactions with each other, and characterize them as independent structures for a given 24-hour period. We divide the graph structure into file-only and network-only (sub)components, which are the connected components derived from the file-only and network-only sub-graphs. We define (i) a *network infrastructure* as a component in the network-only subgraph, while in the case of a file-only subgraph, (ii) a *file infrastructure* as a component consisting of *at least* two file nodes, and (iii) a *lone file* as an isolated node in this subgraph. For example, Figure 1 shows two network infrastructures, {IP_1, HostURL} and {IP_2, HostURL, IntermediateURL}, one file infrastructure, {ParentFile, $File_1$}, and a lone file, $File_2$. This separation into sub-graphs helps us later in the task of attributing infrastructures to independent actors and tracking these over time.

File Classification. To further understand the malicious file delivery ecosystem, we are interested in labeling graph components as "malware," "PUP," or "unclassified," based on their most common types of files delivered. VirusTotal [3] is a freely accessible site that analyzes file submissions across dozens of antivirus engines and produces detailed reports and detection statistics. Amongst these statistics are the family labels by which each antivirus engine classifies the file (e.g., a prominent malware or PUP family).

Simple majority voting could be applied to all labels produced in a VirusTotal report. However, an issue with this approach is that antivirus vendors use inconsistent labels for positive samples, even when the same malware families are detected. For example, two engines may generate labels of `Adware.Rotator.F` and `Adware.Adrotator.Gen!Pac` for the same instance of the AdRotator PUP family. These inconsistencies lead to unreliable majority votes. As a result, Sebastian *et al.* [26] design and evaluate the AVClass malware labeling tool to overcome this problem.

In this work, we use the AVClass tool to label each file SHA-2 that generates a VirusTotal response with a family name, and as likely malware or PUP. Each graph component is then assigned a malware, PUP, or unclassified label, based on a majority vote on the most common family it distributes. If VirusTotal classifies a sample as malicious, but AVClass does not contain its label in its database of aliases, we label it as a singleton cluster named after its SHA-2.

3.2 Longitudinal Analysis

After mapping the actors involved in malicious file delivery over one day, we want to understand how stable these distribution infrastructures are over time. To this end, we track file-only and network-only components on a daily and weekly basis (working from the same day of the week) over an entire year. We also track the lifespans of these infrastructures over a year, using a weekly sampling frame, with respect to the first day of our dataset. More precisely, we do the following:

Snapshot Processing. For each day of data, we generate file-only and network-only connected components. To achieve this, we repeat the steps from *Snapshot Analysis* in Section 3.1 to build components from the overall graphs. We also generate file-only and network-only sub-graphs and build components from these.

Optimal Signature Selection To track distribution infrastructures across different days, we need to first characterize each graph component with a *signature*: a set of nodes within these components which are likely to be temporally stable. Therefore, we need to determine (i) a good criterion for node stability, and (ii) a suitable signature length. We conduct the following experiments to establish suitable signature characteristics:

(1) **Node Centralities.** We pursue a suitable criterion for identifying stable nodes through graph percolation, i.e., the breaking down of a graph component by systematically removing nodes. Graph percolation [8] is useful in showing how resilient a network is to disruption, and by what method. We utilize different *node centralities* as the criteria for selecting the node to be removed at each iteration, with the idea that stable 'root' and 'branch' nodes (e.g., IP addresses, hosts, droppers) are likely to be more "influential" than ephemeral 'leaf' nodes (i.e., end-user downloads). In this case, we use node centralities as proxies for "influence." We then conduct graph percolation on a graph component, via centrality criteria, until it completely disintegrates. We compare the rates of graph percolation under different node centralities and select the one with the highest rate.

(2) **Sensitivity Analysis.** Besides identifying the ranking of the nodes most likely to be stable, we also need to determine a suitable number of nodes to include in the tracking signature when we attempt to trace infrastructures across days. Intuitively, it is unlikely that we would need to consider every single node in a given infrastructure in this matching process. To this end, we conduct a

sensitivity analysis, using our node selection criterion, along with a range of signature lengths as we measure the number of infrastructures that we can track across a pair of days. We then select a maximum signature length based on the principle of diminishing returns, i.e., when the increase in tracking accuracy is insignificant in comparison to the increase in signature length. We present the results of these experiments in Section 4.2.

Component Tracking. We have defined how we generate the signature of each component. Now, we explain how we track these in time. For any pair of consecutive days, i.e., day i and day $i+7$, we generate a bipartite graph: a vertex set V_i, representing components from day i, and a vertex set V_{i+7}, representing components from day $i+7$. Each component is represented by a single vertex, v, with an associated component signature, s. For example, component $v_{i,j}$ represents the jth component from day i and has signature $s_{i,j}$.

Edges represent matches between component signatures when their intersection is a non-empty set (e.g., $s_{i,j} \cap s_{i+7,k} \neq \emptyset$). This representation enables us to generate a simplified, one-to-one mapping of matched components via the following rules (in order of priority):

(1) If $v_{i,j}$ and $v_{i+7,k}$ share an edge, and they have no other incident edges, we retain this edge as a *simple transition.*
(2) If $v_{i,j}$ shares edges with multiple vertices from V_{i+7}, the "best match" is chosen (see below).
(3) If $v_{i+7,k}$ shares edges with multiple vertices from V_i, the "best match" is chosen.

The "best match" algorithm works as follows (before choosing randomly):

(1) Retain edge with the smallest difference in component size.
(2) If multiple edges retained, retain edge with the greatest overlap of leaf nodes between components (i.e., the same payloads).

Forward-facing transitions are prioritized over backward-facing ones, trading-off a little tracking reliability for simplicity. The "best match" algorithm assumes that there is more stability in how many files a dropper distributes over which files it distributes. This assumption is supported by the observation that malware can undergo rapid polymorphism [5]. Note that this tracking technique is also limited in that it oversimplifies the splitting or joining of infrastructures across days as straightforward transitions. Nonetheless, this is sufficient in estimating the activities and lifespans of these delivery infrastructures, giving lower bounds for such.

In Section 4.2 we provide a longitudinal analysis of our data. In particular, we focus on the *churn* of components over time. This aspect is indeed interesting to understand how ephemeral malicious file operations are and to better understand which mitigation techniques are more promising against these phenomena.

4 DATA ANALYSIS

As we explained in the previous section, our analysis is in two stages: first, we look at a single day of data, to better understand the network and file infrastructures involved in the malicious file delivery landscape. We then look at multiple days, to see how the network and file infrastructures evolve. In this section, we illustrate the results of our analysis in detail.

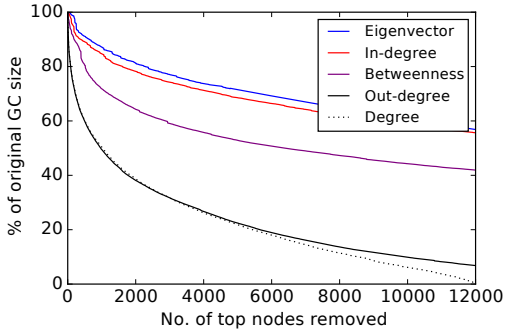

Figure 2: Decay of the GC by graph percolation under different selection criteria. N.B. line order follows graph legend.

4.1 Snapshot Analysis

We build the graph for the first day of our collection period, 1st October 2015. After the pre-filtering operations described in Section 3.1 we obtain a graph G with 1,661,636 nodes and 1,930,648 edges. These nodes consist of 964,998 file nodes (SHA-2s), 385,861 host URL and 218,530 referrer URL nodes (130,630 domains), and 92,247 IP nodes. Each file node represents all download events relating to a unique file, identified by its SHA-2, with a total of 1,644,906 download events recorded for the first day. We separate the graph into weakly connected components (see Section 3.1). Consequently, we generate 58,173 CCs.

We find that a Giant Component (GC) emerges, which accounts for 80% of download activity, comprising of 786,240 unique files (1,345,586 download events) distributed through 89,550 domains, 480,110 URLs, and 51,436 IP addresses. The GC comprises network components and file components interconnected with each other, such as multiple network infrastructures dropping the same set of files. To put this into perspective, the next largest non-Giant component consists of only 2000 nodes. The remainder of download activity (which we refer to as the Non-Giant Component or NGC) is attributed to 58,172 independent distribution infrastructures.

Table 1: Top 10 countries by # of GC articulation IP nodes.

Region	Art. IP nodes	Region	Art. IP nodes
United States	1419	Russian Federation	39
China	268	Canada	31
Netherlands	147	United Kingdom	31
France	114	Luxembourg	28
Germany	53	Brazil	26

Graph structural characteristics. We want to verify whether the GC identified in our analysis is indeed a well-connected set of network infrastructures, or if it is an artifact of our methodology. To this end, we conduct graph percolation as described in Section 3.2, shown in Figure 2. We find that the GC is fairly strongly connected. For instance, it is required to remove over 1k (0.08%) of the highest degree nodes to reduce the size of the GC by more than 50%, and at least 6k (0.46%) nodes – 5.5k of which are network nodes – to reduce the size by 80%. This ratio is an extreme example of the

Table 2: Top second-level domains ranked by # of GC network nodes.

Rank	e2LD	% of hosts	Rank	e2LD	% of hosts
1	mediafire.com	2.80%	11	d3s8yh4ki1ad1i.cloudfront.net	0.67%
2	msecnd.net	2.40%	12	drp.su	0.64%
3	uploaded.net	1.70%	13	crusharcade.com	0.62%
4	magnodnw.com	1.56%	14	doff.info	0.58%
5	mysimplefile.com	1.03%	15	4shared.com	0.53%
6	softonic.com	1.00%	16	zz-download-zz8.com	0.51%
7	clipconverter.cc	0.84%	17	zz-download-zz10.com	0.50%
8	google.com	0.77%	18	zz-download-zz7.com	0.49%
9	file8desktop.com	0.73%	19	mountspace.com	0.47%
10	up1004.info	0.72%	20	zz-download-zz9.com	0.48%

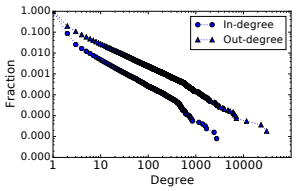

Figure 3: Giant Component degree distribution (complementary cumulative distribution function).

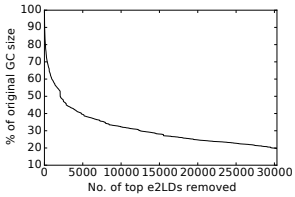

Figure 4: Decay of the GC (no IPs) by removal of top e2LDs.

Pareto principle, which itself states that for many real-world events, roughly 80% of effects come from 20% of causes.

Following from the graph percolation experiment, we identify the articulation nodes which form the structural backbone of the GC. Table 1 shows that, when we focus on IP addresses, the United States is the biggest regional contributor to this massive distribution infrastructure. This ranking could indicate where ISP takedown efforts would be most effective in dealing with unwanted software distribution. The GC is an approximate *scale-free network*: Figure 3 shows its degree distribution approximately following a power-law distribution. It contains 1.3M nodes and 1.6M edges. The diameter of the GC is 20, meaning that there are 20 hops along the longest chain of IPs, URLs, and dropped files. The average path length of the GC is 6.20 (average number of hops between any pair of nodes). The GC also has a global clustering coefficient of 3.6×10^{-5}. This property could be an indication of a tree-like structure for the GC, with a relatively small number of highly interconnected root nodes, but many branches and leaf nodes. This conclusion is supported by the fact that only a very small proportion of nodes – most of which are hosts – are responsible for the connectivity of most of the GC.

Significance of the GC. Though our initial findings showed that the GC is a well-connected ecosystem of files and network infrastructures, this component could still be an artifact of, for example, the shared use of IP addresses by different malicious operations, due to their use of popular hosting providers and content distribution networks (CDNs). This classification would result in a false connection of services that are effectively independent in the real world, e.g., separately owned Amazon EC2 instances being linked to the same IPs and/or domains. To rule out these scenarios, we conduct two experiments: first, we rebuild the graph without IP addresses, and second, we blacklist the most popular effective second-level domains (e2LDs). We use the Mozilla Public Suffix List[1] to identify the e2LDs in our dataset. Note that this list includes common CDN resources as suffixes (e.g., `ca-central-1.amazonaws.com`). We included `amazonaws.com` as a suffix to separate its service users.

For these verification experiments, we rebuild the graph G without any IP address nodes or URLs with IP addresses as domain names (e.g., *http://119.147.227.164/path/to/file*). This results in a graph of 1,544,062 nodes (7% reduction) and 1,578,585 edges (18% reduction). After computing the weakly connected graph components, we find that the GC is considerably smaller, but remains stable, with 908,029 nodes (31% reduction) and 1,102,300 edges (32% reduction). Evidently, IP addresses help form a significant part of the GC, connecting about 31% of this component. In real terms, shared IPs connect a significant proportion of the unwanted software distribution market – potentially an indication of shared or repeated use of network infrastructure, or these services being illicit, thus not appearing on the public suffix list. However, these results show that there is also a strong interconnection between distribution services through URL-to-URL redirections between hosts, and the distribution of multiple software per service (one-to-many) and common software between multiple services (many-to-one).

Next, we categorize the most popular e2LDs by grouping the network nodes (hosts and referrals) that share the same e2LD and rank them by the number of associated network nodes. Table 2 shows the top 20 e2LDs. We find that the top GC domains predominantly belong to popular CDNs, such as *MediaFire* (7.4k nodes), *Windows Azure* (under *msecnd.net*, 6.4k nodes), *Softonic* (2.7k nodes), and *Google* (2k nodes). An apparent *zz-download-zz* CDN is also very prominent, consisting of 2.86% (7.6k nodes) of hosts. As later results suggest, the unwanted software distribution economy may be leveraging, if not directly using, the infrastructures of benign and popular CDNs.

Figure 4 exhibits the exponential decay of the GC as the top e2LDs are removed in order of decreasing rank. We find that the GC structure remains resilient to percolation, even after the total removal of all 30,330 e2LDs in the GC. This result shows that both IP addresses and popular domains are important for the connectivity of the GC, but there is still a strong and resilient interconnection between the files that are distributed within it, as evidenced by 20% of the GC (180k nodes) remaining after removing all domains and IPs. That is, droppers also contribute significantly to the proliferation of unwanted software and are core to the malicious file delivery ecosystem. Taking into account that the smallest estimate of the GC is still *90 times the size* of the largest non-GC infrastructure (2k nodes), this evidence strongly suggests the presence of the GC structure in the malicious software delivery ecosystem, regardless of potential measurement artifacts. We then study the differences between the GC and NGC infrastructures.

File distribution of the GC and NGC. After identifying the presence of two distinct groups of infrastructures, the GC and the NGC (composed of 58k independent infrastructures), we aim at better understanding what kind of files are installed as part of the two ecosystems. We apply the file classification process as described in Section 3.1. Of the 965k unique files downloaded in this day of

[1]The Public Suffix List is a cross-vendor initiative to provide an accurate list of domain suffixes – https://publicsuffix.org/

data, VirusTotal generates analysis reports for only 80k file SHA-2s. AVClass [26] then processes the VirusTotal results to produce 61k family labels: 42k are from known families, while 19k are from unclassified families, which are labeled as "singletons" (see File Classification in Section 3.1). The remaining 19k of SHA-2s analyzed by VirusTotal had not been classified as malicious at the time.

The attrition in ground-truth data is undesirable but expected. Only a small proportion of files are actually submitted to VirusTotal for analysis, hence the considerably small record size compared to the total number of files. Of the files for which VirusTotal has analysis records, some attain no AV detections, hence leading to AVClass producing no family labels for these SHA-2s. Even for files that have been detected as potentially malicious, some of them are only given generic labels by the detecting AV vendors (e.g., `Trojan.Dropper.Gen`). These generic labels are stripped away by AVClass, leaving only family-specific labels, or when none such labels exist, singleton SHA-2 labels are used to indicate unclassified families. Because of this attrition in ground-truth, we characterize clusters of files that exhibit dropping behaviors using the available AVClass labels where they are observed. In particular, we use a majority voting scheme to label each file-only graph component with its most common family, as well as whether it is likely 'malware,' 'PUP,' or 'unclassified.' This estimation helps to characterize the remaining unlabelled but related files.

Figure 5 shows various family distributions for the GC and NGC ecosystems. A key finding here is that there is a clear difference in the presence of unwanted software within these two ecosystems: the GC is primarily dominated by PUP, while the NGC is dominated by malware distribution activities. For the GC, PUP such as `convertad`, `amonetize`, and `opencandy` conduct the lion's share of download activity. Similarly, these families act as prominent droppers, installing other malicious files on infected computers, though there is also a considerable malware presence, particularly with `zusy`. In the NGC infrastructures, `gamarue`, for example, is very effective in both download and dropping activities, as are other malware families. Also, note that `extcrome` is labeled as malware, while this family is actually adware and should, therefore, be classified as PUP [1]. This is a false positive result of AVClass, highlighting the imperfection of the AVClass labeler, although such misclassifications are generally rare in this dataset.

Another interesting observation is in the mixed presence of PUP and malware droppers and payloads within the GC. Given that the GC is a single connected component or download infrastructure, this alludes to a mixed distribution mechanism for PUP and malware, though it is still PUP-dominated. By majority voting on the most common family for a given file component (see Section 3.1), we estimate that the numbers of independent PUP and malware file delivery operations (i.e., file components) in the GC are roughly 1.5k and 360, respectively (3.2k unclassified), and for the NGC, 190 and 250, respectively (2.9k unclassified). Note that we do not consider lone files as file delivery operations (i.e., singleton file components that do not engage in any dropping activities). 82 (1% of) file delivery operations involve both PUP and malware, which is in alignment with Kotzias *et al.* [17] that refer to PUP distribution and malware distribution as being largely disjoint. However, we find that a single massive file delivery operation involves both PUP and malware, and is responsible for the distribution of 61k SHA-2s (7.7% of the GC) and 394k raw downloads (29% of the GC). This is in line with the work by Kwon et al. [19], who found that 36.7% of the droppers that they observed were downloading both malware and PUPs. To provide context, the next largest delivery infrastructure only distributes 2k SHA-2s. We provide a case study of the Opencandy operation in Appendix A.

We also compute estimates of the proportions of PUP-to-malware in the wild by identifying SHA-2s of known families, and whether they are likely malware or PUP. In the overall graph G, the PUP-to-malware ratios are roughly 5:1 (SHA-2s) and 17:2 (raw downloads). The proportions of PUP-to-malware in the GC are roughly 8:1 in # of SHA-2s (95.3% unclassified) and 11:1 in # of raw downloads (74.2% unclassified). In the NGC, the PUP-to-malware ratios are 1:1.78 in # of SHA-2s (97.4% unclassified) and 1:2.15 in # of raw downloads (96.1% unclassified). Despite previous work already highlighting that PUP is more predominant in the wild than it was previously thought [17], we are the first ones to quantify the ratio of malware and PUP in the wild.

To summarize, we discovered two file delivery ecosystems. The GC consists of interconnected file and network infrastructures and mostly drops PUP, while the NGC is composed of independent components and mostly drops malware. Because of the GC predominantly dropping PUP and the NGC mostly being responsible for malware downloads, for the remainder of this paper, we will refer to the GC as *PUP Ecosystem* and the NGC as *Malware Ecosystem*.

Network and file characteristics of the two ecosystems. Figure 6 shows a structural comparison of the PUP Ecosystem and Malware Ecosystem sub-graphs. The file in-degree and out-degree distributions for the PUP and Malware Ecosystems are very similar. This could be indicative of largely similar distribution patterns being employed by malware and PUP authors, e.g., the common use of PPI services. However, the PUP Ecosystem generally has higher in-degree and out-degree distributions for network nodes. This result suggests several notions. First, hosts in the PUP Ecosystem are typically more interconnected (i.e., redirections between hosts) and/or utilize more IPs than hosts in the Malware Ecosystem. Also, hosts in the PUP Ecosystem are likely to be more prolific distributors (e.g., CDNs) than in the Malware Ecosystem, as also shown in the long-tails. This is likely due to the larger volume of traffic that these services can attract.

The file SHA-2s dropped per domain distribution shows that domains in the Malware Ecosystem download significantly fewer unique files onto victim systems than those in the PUP Ecosystem. However, the actual number of raw files downloaded by PUP domains is only slightly more. There are several possible explanations for this. Sites hosting malware could be used by malware authors to only distribute their own binaries, or by illegitimate PPI infrastructures that serve fewer malware customers per domain. The malware sites could also be distributing a few file SHA-2s before changing domain names in order to evade detection. On the other hand, while many of the sites in the PUP Ecosystem may be CDNs that are accessed explicitly by users to download different types of software (hence its larger distribution of SHA-2s), more of the malware-hosting sites could be benign sites that are compromised and unknowingly hosting exploit kits. In this case, victims would be

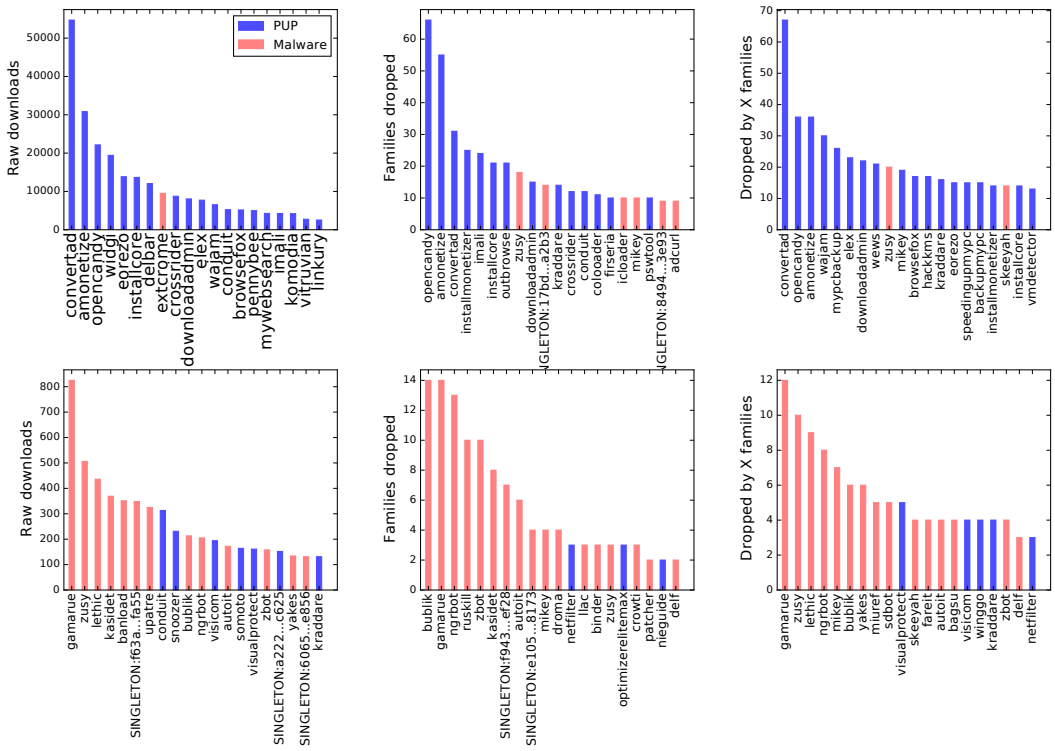

Figure 5: Malware/PUP family distributions. From left to right, figures show: i) top families by # of raw downloads; ii) top droppers by # of known families dropped; and iii) top known families dropped. The top row is for the Giant Component, while the bottom row is for Non-Giant Components.

infected without consent through silent drive-by downloads (hence the fewer SHA-2s distributed by Malware Ecosystem domains).

Over 98% of SHA-2s are *lone* files, as shown by the file component distributions. Lone files do not engage in any file dropping activities, nor are they dropped by any other file SHA-2 – they are observed to be downloaded only directly from hosts. Though component sizes vary, a majority of file components in both the PUP and Malware Ecosystems have diameters and average path lengths between 0 and 2 (>99.9% for both), although the file component sizes, diameters, and average path lengths in the Malware Ecosystem are slightly larger in general. This explains the very low clustering coefficient of the PUP Ecosystem (GC) and supports the notion that downloader graphs are generally very sparse and tree-like, with the Malware Ecosystem having similar, albeit unconnected, distribution patterns.

Evasion tactics. The distribution of IP addresses per domain provides an interesting result. While there is evidence of over 90% of domains having only one IP address each, far more IPs per domain are used by a significant proportion of the PUP Ecosystem than the Malware Ecosystem. The high usage of IPs per domains in the PUP Ecosystem could be evidence of increased use of the fast flux technique in this ecosystem. However, this could also be attributable to the significant presence of various CDNs in this ecosystem.

Rossow *et al.* [25] state that rather than using servers with fast flux, some pay-per-install operators opt to distribute their dropper malware through multiple servers, each hosted on a different autonomous system (AS). Figure 7 shows the distributions of IPs and ASes being used to serve droppers. We find a fundamental difference in the dropping modus operandi between large portions of the PUP and Malware Ecosystems. While only less than 10% of droppers in the Malware Ecosystem are distributed across more than one IP or AS, over 70% of droppers in the PUP Ecosystem are distributed across more than one IP address, while over 45% are distributed across more than one AS, indicating the use of this tactic described by [25]. Servers with this abundance of resources are very likely to be constituents of CDNs. In fact, many of these malicious network infrastructures appear to congregate on well-known ASes, as shown in Table 3. Note that a single network distribution infrastructure may operate across multiple ASes.

PPI estimation. We also estimate the number of Pay-per-Install (PPI) services active during this single day. We define a PPI service as a network-only component that directly drops more than one type of malware or PUP family. We only consider known families, as the families of files with singleton AVClass labels could not be determined. We also aggregate network components with shared e2LDs to reduce the number of potential PPI services. As a result, we estimate a potential lower bound of 215 PPIs operating in the PUP Ecosystem and 179 PPIs operating in the Malware Ecosystem. We note that the largest "PPIs" in the PUP Ecosystem and Malware Ecosystem involve about 99% and 24% of all e2LDs and IP addresses in their ecosystems, respectively. In real terms, this could further

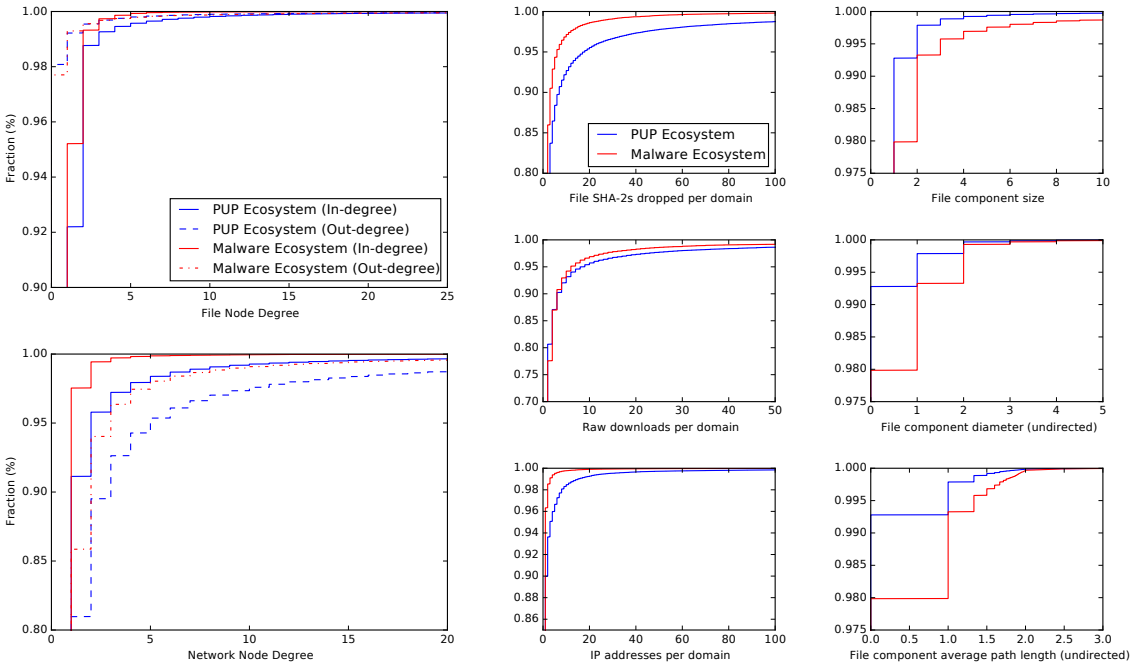

Figure 6: Structural comparison of PUP and Malware Ecosystems.

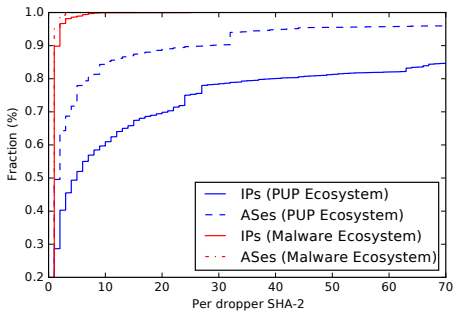

Figure 7: Distribution of IP addresses/autonomous systems serving each dropper. Droppers with no traceable IPs or ASes are omitted.

indicate that PPIs, as we know them, are more highly connected than once thought, either through shared use of infrastructure or from one service reselling to another. Note that other inter-host relationships (such as web links between pages) are not considered.

Finding such a high level of connectivity raises the question: why does the GC exist? Other works suggest that many different companies engage in unwanted software distribution and that it is unlikely that they are in close collaboration. However, arguably, our data suggests otherwise. It is possible that different affiliates are distributing the same binaries, or that software authors are running the same auction systems, leading to the downloads of these same binaries. For instance, at least 4% of file SHA-2s in the GC are distributed by more than one host. Alternative hypotheses are that multiple companies that distribute unwanted software are actually controlled by a single company, and/or that many CDNs are acting as resellers unto other resellers, and so on. For example, a particular network infrastructure consists of 2 IPs and 30 different e2LDs, including *downloadopencloud.com*, *opencloudsafe.com*, *setupfreesoftware.com*, and *thesafedownload.com*, and, within the data, most major CDNs are structurally connected in one way or the other. However, as the data suggests, we can confidently rule out malware delivery mostly being a set of vertically integrated operations. Instead, it is either one big organized operation (unlikely), or a well-connected marketplace of infrastructure providers.

Summary of results. In this 24-hour snapshot analysis, we showed that the malicious file delivery landscape could be partitioned into two disjoint ecosystems: a tightly connected ecosystem that is mostly responsible for downloading PUP, and a set of isolated infrastructures that are mostly responsible for downloading malware. We found that the PUP Ecosystem is responsible for 80% of the total number of suspicious file downloads worldwide and we reckon that it is likely a well-connected marketplace of infrastructure providers. We calculated the ratio of malware and PUP appearing in the wild, and we find that PUP dominates over malware by 17:2 in the number of files downloaded worldwide. We compared the structures and distribution techniques of the two ecosystems, finding that PUP operators are more likely to distribute the delivery of their malicious files across more IP addresses and autonomous systems. We also found that IPs from the U.S. are core to the PUP Ecosystem, which could be the most effective target for ISP takedowns. Using our longitudinal analysis technique, one could go further in identifying the most stable of these IPs such that those that are

Table 3: Top 10 autonomous systems by # of network infrastructures hosted (i.e., CCs from network-only graph).

AS No.	Organization	Region	Network Infrastructures Hosted
16509	Amazon.com Inc.	US	2901
15169	Google Inc.	US	2508
14618	Amazon.com Inc.	US	1425
16276	OVH SAS	FR	1289
4134	China Telecom	CN	999
13335	CloudFlare Inc.	US	788
20940	Akamai Technologies	EU	755
24940	Hetzner Online	DE	600
4837	China Unicom	CN	567
26496	GoDaddy.com LLC	US	563

Table 4: Sensitivity analysis. $\lfloor log_2(X) \rfloor$ is the variable length signature with the size being the rounded-down logarithm of the component size X.

Maximum Signature Length	Day-Pair 1	Day-Pair 2
1	32.5%	38.1%
$\lfloor log_2(X) \rfloor$	41.1%	46.7%
2	46.5%	51.7%
3	48.0%	53.6%
4	48.2%	53.7%
5	**48.2%**	**53.8%**
10	48.3%	53.8%
20	48.3%	53.9%
50	48.3%	53.9%
100	48.3%	53.9%

purely illicit are targeted for ISP takedowns, while the benign ones (e.g., CDNs) are advised to improve their security practices.

4.2 Longitudinal Analysis

So far, we have looked at the malicious file delivery ecosystem over 24 hours. However, many questions remain unanswered on how such delivery ecosystems evolve. Therefore, in this section, we analyze the temporal evolution of file delivery networks.

PUP Ecosystem persistence. First, we build a graph for each day. We find that the PUP Ecosystem (i.e., Giant Component) is stable over the entire year. This result is important, as prior work [17, 20] only characterizes PUP and malware ecosystems in the short-term. As described in Section 3.2, we then compute the network-only and file-only components from the overall graph, which represent the network-based and file-based delivery infrastructures.

Infrastructure tracking. We aim to develop robust signatures to track infrastructures in time. We conducted a graph percolation experiment (see Figure 2), where we measure how quickly the GC breaks down using a number of graph influence measures, i.e., eigenvector, betweenness, in-degree, out-degree, and overall degree centralities. Following this experiment, we select out-degree as the criteria to select influential nodes for infrastructure signatures (see Section 3.2). In practice, degree and out-degree perform identically, but out-degree is more efficient memory-wise as it does not include leaf nodes in the tracking signature, which are redundant.

We conduct a sensitivity analysis of tracking performances with different maximum signature lengths. Here, we select infrastructures from two randomly selected pairs of consecutive days in the data series (i.e., Day-Pair 1: 2015-Oct-22 and 2015-Oct-29, and Day-Pair 2: 2016-Feb-02 and 2016-Feb-09). A match is defined as an intersection of a pair of signatures across two days. We then compute the percentage of component signatures matched across these day-pairs using different signature lengths. Finally, by diminishing returns, we select a maximum signature length of 5 (see Table 4).

This result means that a graph component can be characterized by *up to* five of its top out-degree nodes. An example of a network component signature is {*'http://groupsetzipmyjob(dot)org/hp/'*, *'107.21.97.98'*, *'54.225.102.164'*, *'68.232.34.200'*, *'74.120.16.179'*}. Besides making tracking computationally feasible, this also points out elements (e.g., IP addresses, DNS domains) that are stable over time and can, therefore, constitute potential intervention points by law enforcement agencies (LEAs) and security companies (e.g., for takedowns).

In our tracking analyses, we only consider file clusters that exhibit dropping behavior as file-side distribution infrastructures. The churn of the remaining *lone* files is considered separately as these are less easily attributable to individual actors. We also track infrastructures using two temporal granularities: daily (over a month) and weekly (same weekday sampled over a year). This approach allows us to observe in detail the delivery network life-cycles in the medium-term, while also enabling us to monitor their long-term trends efficiently.

Churn of infrastructures. Figure 8 shows the daily churn of network and file delivery infrastructures, i.e., the number of infrastructures that are detected from one day to the next. The daily churn reveals cyclicity in the network and file distribution infrastructures, both that are active in download activity (total) and that are tracked, with a cyclic period of seven days. As 1st October 2015 was a Thursday, the results show that more distribution infrastructures are active across weekdays (i.e., Mon-Tue, Tue-Wed, Wed-Thu, and Thu-Fri) and less across weekends (i.e., Fri-Sat, Sat-Sun, Sun-Mon). The cyclic download activities during the week could show that the file distribution patterns of cybercriminals and legitimate providers alike mirror the network use, download, and work-rest patterns of people and organizations. In other words, infections increase during business hours because more potential victims have their computers on, as already observed by previous work [11, 27, 28].

Figure 9 shows the daily churn for lone files. A lone file is a file that is dropped directly from network hosts and that does not engage in any further dropping behavior. On the other hand, we defined a file-side distribution infrastructure as a file-only component that exhibits dropping behavior between files. In comparison with the churn of file infrastructures (Figure 8), the fluctuations in the presence of lone files (Figure 9) and network infrastructures (Figure 8) appear more pronounced. This is due to there being many more network infrastructures and lone files than file infrastructures, e.g., lone files constitute 98% of file-components. It should be noted that the weekly fluctuation in the total and tracked network infrastructures may not specifically represent hosts going down or coming up: it only means that the sensors used in this dataset do not observe downloads from these hosts.

As shown in Figure 10, the weekly churn of delivery infrastructures (sampled every Thursday for a year), omits this weekly periodicity. However, we observe a large drop in download activity from 19th November 2015 until 14th January 2016, with a small

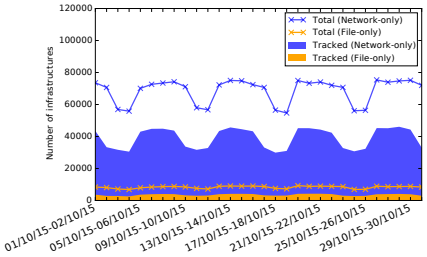

Figure 8: Daily churn of delivery infrastructures over a month.

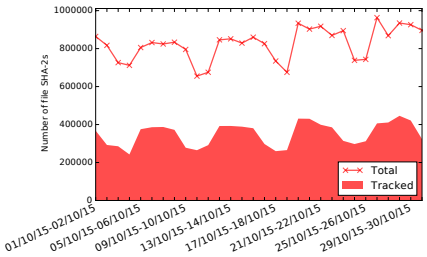

Figure 9: Daily churn of lone file SHA-2s over a month.

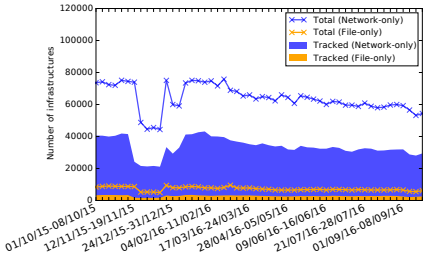

Figure 10: Weekly churn of delivery infrastructures over a year.

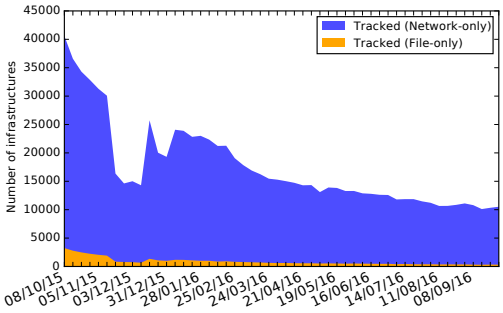

Figure 11: Lifespan of delivery infrastructures tracked from 1st October 2015, over a year.

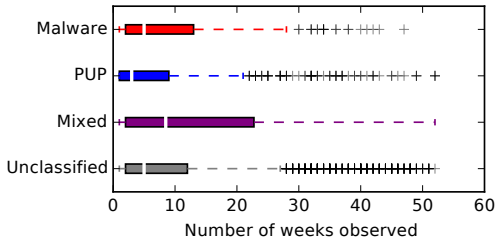

Figure 12: Box plots showing the lifespan of file delivery infrastructures.

peak in activity on the week of 17th to 24th December 2015. We later investigate this anomaly (see ensuing case study).

Lifespan of infrastructures. Figure 11 is the lifespan plot of distribution infrastructures observed since our first day of analysis (1st October 2015), with a weekly granularity, showing the activity decay of these infrastructures over a year. That is, infrastructures observed on each sampled day are matched with infrastructures observed on 1st October 2015, where the sampling frame is seven days. We initially track 40.6k network infrastructures and 3.2k file infrastructures and find that, of these, at least 30k network infrastructures (75%) remain active for over 6 weeks, while 10.5k network infrastructures (26%) and 320 file infrastructures (10%) remain active over a year. We also observed a dip in activity beginning from between 12th and 19th November 2015. However, the rise in tracked infrastructures between 17th and 24th December 2015 indicates the re-emergence of some of the same network and file infrastructures. This volatility in network and file delivery infrastructures could be due to these infrastructures going in and out of service (e.g., server take-downs, ceasing activities, new actors entering the ecosystem). However, this could also be hosts utilizing fast flux or DGA and/or prolific droppers polymorphing.

We then look at the lifespan of file delivery infrastructures that drop only PUP, only malware, or both. Figure 12 shows these observations. We identify and track 344 confirmed malware-only delivery infrastructures, 805 PUP-only ones, and 50 infrastructures that deliver both PUP and malware. We find that file-side (not host-level) delivery operations involving malware appear to be longer-lasting than PUP ones, i.e., file-dropping networks are active for a median of 5 weeks for malware vs. 3 weeks for PUP. This could be due to the likelihood that malware is stealthier in their installation and operation on a victim computer, and/or more resilient to removal than that of PUP. Mixed operations involving both PUP and malware appear to be the most enduring. However, the validity of this result is still questionable due to its sampling biases.

The anomalous drop in download activity. We observe a substantial drop in download activity between 12th and 19th November 2015. Symantec [30] reports the cessation of activities of the cybercriminal group behind the Dyre financial fraud trojan, following a Russian LEA operation in November 2015. We also observe significant drops in the presence of other families including `upatre`, `amonetize`, `installcore`, `eorezo`, and `convertad`.

Summary of results. In this longitudinal analysis, we found that the PUP Ecosystem is stable in the long-term. We found periodic download patterns over a week, perhaps in accordance with the Routine Activities Theory from criminology [10]. We also found that network infrastructures tend to be quite short-lived, where 75% survive for over 6 weeks, while 26% survive for over a year. Finally, we presented a case study which denudes the possibility of common distribution backbones between malware, such as Dyre, and popular PUP PPIs, such as Amonetize.

5 DISCUSSION

In this paper, we presented a data-driven analysis of the delivery of malicious files on the Web. Our findings shed some light on malware and PUP operations more comprehensively than previous work. In this section, we take a step back and reason over what our findings mean, and how they could be applied for mitigation purposes. We then highlight some limitations to our approach.

5.1 Implications of findings

In our study, we found two largely disjoint ecosystems, one responsible for the delivery of PUP and one dedicated to installing malware on victim computers. We find that the malicious file delivery ecosystem makes considerable use of CDNs, which can make takedown operations difficult. On the other hand, we identified ASes in which malicious network infrastructures congregate. This result is consistent with previous research [29] and suggests that ISP-based interventions can still be a valid method to disrupt malware operations. In the paper, we presented a methodology to identify network elements (DNS domains, IP addresses) that do not change over time. This methodology could be further developed to identify optimal intervention points that LEAs could target to perform disruption, solving the fundamental problem of identifying the right elements to target when performing takedowns, as highlighted by previous work [21]. Other future works include repeatability experiments with other (open-source) datasets, and identifying more real-world stimuli (e.g., ISP takedowns) and MDN adaptations within the data.

5.2 Limitations

As we mentioned, our data-driven analysis has limitations. By applying graph analysis techniques to the download graph, we obtain a proxy to what is happening on the victim computers. The type of analysis that we perform allows us to characterize the operation of PUP and malware delivery networks, but we cannot be certain about some of the details of malware operations that go beyond our study data. For example, by looking at the dropping behavior of hosts, we may estimate whether they belong to exploit kit infrastructures or not. However, we do not observe the actual vulnerabilities being exploited on the host as part of a drive-by download attack. For this reason, it could be that some of the infrastructures that we may consider as exploit kits are just relying on social engineering. In a similar sense, we are not able to see auxiliary connections between hosts, such as direct web links.

We identify files using their SHA-2 (256 bits) hash function. This allows us to reliably distinguish between unique files, e.g., variants of the same malware family, or to identify the same file in the wild under different guises, e.g., a malware binary using different file names. However, this method of identification still presents complications for packed files. Packing alters the SHA-2 of a file and so the same binary that is re-packed multiple times would appear as different unique files. This may manifest in our technique as a host or dropper delivering multiple files when they are actually repacked versions of the same file binary.

As an additional limitation, some of our analysis relies on third-party information such as VirusTotal and the AVClass tool. This information, however, is not perfect (e.g., some false positive indications), and, as we have shown, is often incomplete. For this reason, some of the file components that we identified may have been misclassified. We focus our analysis on files with known families. This helps to mitigate false positive indications from VirusTotal, as each binary in this dataset that is assigned a family name by AVClass has at least 2 different AV engines agreeing on the associated malware/PUP family. This classification excludes AV engines that may also assign positive indications, but are not taken into account by AVClass due to them only assigning a generic family name.

6 RELATED WORK

This section aims to provide the reader with an overview of the different aspects explored by previous research in this area.

6.1 Malicious Payloads

Malware. Malware has been a growing problem for the past three decades. Previous research focused on studying how malware obfuscates itself to avoid easy detection [9]. Over the years, malware has been used for many reasons: sending spam emails [28], stealing banking credentials from infected computers [6, 27], encrypting victim data and asking for a ransom [16]. The research community showed that *droppers* that belong to PPI services often download prominent malware families [27, 28].

PUP. Recent research shows that PUP is rapidly becoming a critical problem. For example, two recent papers show that rogue browser extensions that contain hidden functionalities are on the rise [14, 15]. Thomas *et al.* [31] report that 5% of Google users have installed browser extensions that substitute the advertisements that they see. Such extensions can be particularly dangerous as rogue ad networks can be used to infect users with malware through drive-by download attacks [33]. Thomas *et al.* [32] provided a systematic study of PUP prevalence and its distribution through *commercial* pay-per-install (PPI) services, mainly focusing on four prominent downloaders from Amonetize, InstallMonetizer, OpenCandy, and Outbrowse. Their research results claimed that commercial PPIs drive over 60 million download attempts per week and knowingly attempt to evade user protections (e.g., antivirus software). Kotzias *et al.* [17] take a different approach, identifying dominant PUP publisher names from code signing certificates, to study PUP prevalence and its distribution through PPI services. The authors claim that the fundamental difference between malware and PUP is the distribution mechanism. They argue that malware distribution is dominated by *silent* installation through vulnerability exploitation, while PUPs are installed with the consent of the users.

6.2 Payload Delivery Techniques

The research community identified two main malware delivery methods: *exploit kits* and *pay-per-install services*.

Exploit Kits. Exploit kits have been used for many years to spread malware. In a nutshell, exploit kits collect a large number of exploits targeting many versions of operating systems, browsers, and browser plugins, to make sure that criminals can infect as many victim computers as possible [12]. One of the earliest exploit kits is MPack, a PHP-based kit released in late 2006 [12]. The main functionality of these kits is to gather information on the victim machine, find vulnerabilities, determine the appropriate exploit, and, finally, deliver it (e.g., drive-by downloads) and execute the malicious payload. Grier *et al.* [12] focused on malware installed upon a successful browser exploit, and investigated the emergence of the exploit-as-a-service model for drive-by browser compromise. They did so by analyzing over 10,000 distinct binaries extracted from 77,000 malicious URLs. Their study showed that 9 exploit kits, though a small number, account for 92% of the malicious URLs in their dataset, 29% of which belong to the Blackhole exploit kit.

Pay-Per-Install (PPI) Services. PPI services have been existing for years. They originated as services to facilitate the distribution

of advertisements, but have seen significant (malicious) changes over the years by centering on pushing malware and spyware to unsuspecting users [28]. A typical PPI ecosystem has three main actors: a client, a service provider, and an affiliate. Caballero *et al.* [7] provided the first large scale measurement of blackmarket pay-per-install services in the wild. They achieve this by harvesting over a million client executables using vantage points spread across 15 countries. This work found that 12 out of 20 of the most prevalent malware families at the time employed PPI services to buy infections. Kotzias *et al.* [17] leveraged file dropping graphs to build a *publisher graph* and identify specific roles in the ecosystem. The authors tag roles (e.g., client, service provider, and affiliate) to each publisher by measuring the in-degree and out-degree of each cluster in the publisher graph. Publishers with both high in-degree and out-degree behave like PPI service providers; publishers with high in-degree but low out-degree are likely advertisers; publishers with low in-degree and high out-degree are likely affiliates.

7 CONCLUSION

In this paper, we presented the first comprehensive data-driven analysis of malicious file distribution on the web. We discovered that there are two disjoint ecosystems responsible for the delivery of PUP and malware, respectively, and that the PUP ecosystem is particularly stable over the long-term. Studying the characteristics of these ecosystems in detail, together with their temporal dynamics, we found that the PUP ecosystem is responsible for 80% of suspicious downloads worldwide. We estimated the ratios of PUP-to-malware in the wild to be 17:2 and differentiated the modus operandi of file distribution between the two ecosystems. We also tracked these distribution infrastructures over a year, finding that 75% of malicious network infrastructures survive for over six weeks, while 26% survive for over a year. Our findings help researchers gain a better understanding of this ecosystem, and allow us to identify promising routes for more effective mitigation against the distribution of malicious software. For instance, we devised a methodology to identify those elements in a delivery infrastructure that change slowly over time. In future work, we will explore the possibility of using such elements (IP addresses, Autonomous Systems, Domain Names) for performing effective takedown operations.

8 ACKNOWLEDGMENTS

We would like to thank all of our anonymous reviewers. Colin C. Ife was funded by EPSRC under grant EP/M507970/1. Steven J. Murdoch is supported by The Royal Society under grant UF160505.

REFERENCES

[1] Adware/ExtCrome.syek. https://www.avira.com/en/support-threats-summary/tid/143973/threat/Adware.ExtCrome.syek.
[2] IPv6 martian and bogon filters. https://6session.wordpress.com/2009/04/08/ipv6-martian-and-bogon-filters/. Accessed: 2018-05-24.
[3] VirusTotal. https://www.virustotal.com.
[4] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Internet Measurement Conference (IMC)*, 2006.
[5] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel. A view on current malware behaviors. In *LEET*, 2009.
[6] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *Privacy Security and Trust (PST)*, 2010.
[7] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *USENIX Security Symposium*, 2011.
[8] D. S. Callaway, M. E. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. *Phy. rev. letters*, 85(25), 2000.
[9] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant. Semantics-aware malware detection. In *IEEE Symposium on Security and Privacy*, 2005.
[10] L. E. Cohen and M. Felson. Social change and crime rate trends: A routine activity approach. *American sociological review*, 1979.
[11] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. *SRUTI*, 5:6–6, 2005.
[12] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, et al. Manufacturing compromise: the emergence of exploit-as-a-service. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.
[13] H. J. Highland. The BRAIN virus: fact and fantasy. *Computers & Security*, 1988.
[14] N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas. Trends and lessons from three years fighting malicious extensions. In *USENIX Security Symposium*, 2015.
[15] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson. Hulk: Eliciting malicious behavior in browser extensions. In *USENIX Security Symposium*, 2014.
[16] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda. Cutting the gordian knot: a look under the hood of ransomware attacks. In *Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2015.
[17] P. Kotzias, L. Bilge, and J. Caballero. Measuring PUP Prevalence and PUP Distribution through Pay-Per-Install Services. In *USENIX Security Symposium*, 2016.
[18] P. Kotzias, S. Matic, R. Rivera, and J. Caballero. Certified PUP: Abuse in authenticode code signing. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.
[19] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitras. The dropper effect: Insights into malware distribution with downloader graph analytics. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.
[20] B. J. Kwon, V. Srinivas, A. Deshpande, and T. Dumitraş. Catching worms, trojan horses and pups: Unsupervised detection of silent delivery campaigns. *arXiv preprint arXiv:1611.02787*, 2016.
[21] Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee. Beheading hydras: performing effective botnet takedowns. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 121–132. ACM, 2013.
[22] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad. Webwitness: Investigating, categorizing, and mitigating malware download paths. In *USENIX Security Symposium*, 2015.
[23] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad. Towards measuring and mitigating social engineering software download attacks. In *USENIX Security Symposium*, 2016.
[24] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, N. Modadugu, et al. The ghost in the browser: Analysis of web-based malware. In *HotBots*, 2007.
[25] C. Rossow, C. Dietrich, and H. Bos. Large-scale analysis of malware downloaders. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. 2013.
[26] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, 2016.
[27] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *ACM conference on Computer and communications security (CCS)*, 2009.
[28] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. In *Workshop on lage-scale exploits and emerging threats (LEET)*, 2011.
[29] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding rogue networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2009.
[30] Symantec. Dyre: Operations of bank fraud group grind to halt following takedown. https://www.symantec.com/connect/blogs/dyre-operations-bank-fraud-group-grind-halt-following-takedown, 2016. [Online; accessed 11-August-2017].
[31] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, et al. Ad injection at scale: Assessing deceptive advertisement modifications. In *IEEE Symposium on Security and Privacy*, 2015.
[32] K. Thomas, J. Crespo, J.-M. Picod, C. Phillips, C. Sharp, M.-A. Decoste, A. Tofigh, M.-A. Courteau, L. Ballard, R. Shield, N. Jagpal, M. Abu Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy. Investigating Commercial Pay-Per-Install and the Distribution of Unwanted Software. In *USENIX Security Symposium*, 2016.
[33] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna. The dark alleys of Madison avenue: Understanding malicious advertisements. In *Internet Measurement Conference (IMC)*, 2014.

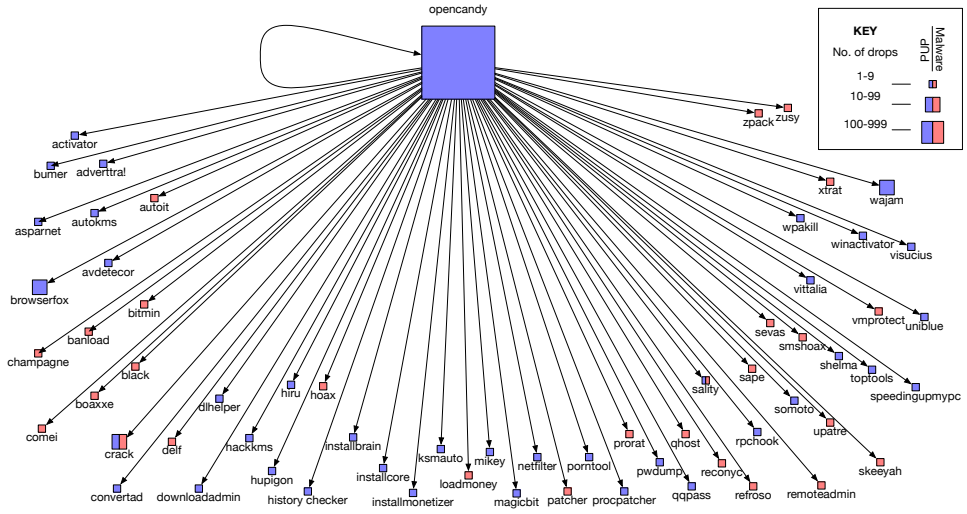

Figure 13: Known families dropped by Opencandy. Note that unknown families are omitted from this diagram.

A OPENCANDY OPERATION

Figure 13 shows the known families dropped by the prevalent opencandy PUP, a commercial pay-per-install (PPI), within this 24-hour snapshot. This figure indicates that PUP-malware relationships and mixed distribution infrastructures may be a bigger problem than first thought. Opencandy seems to drop malware and PUP by similar proportions: 26 malware families (63 file SHA-2s) versus 37 PUP families (132 file SHA-2s), excluding the 288 Opencandy self-dropped SHA-2s. It is also interesting to see the dropping behaviours of this PPI. In particular, some of its customers include other installer software such as `convertad` and `installmonetizer`. This could be evidence of business-to-business relationships and shared distribution infrastructures between these competing PPI brands.

Opencandy also directly drops instances of its own binaries. We find that the longest chain of Opencandy dropping its own binaries is a length of 2 sequential drops. For instance, a drop-chain of Opencandy binaries (same SHA-2) have the file names `PowerISO5 X64.exe`, `ADV_35.EXE`, and `spstub[1].exe`. `PowerISO5 X64.exe` is the brand name of a CD/DVD image processing tool, while `spstub[1].exe` is the name of software developed by Conduit, most often with the description "Search Protect by Conduit". This could simply be the result of affiliate tracking, but, given that Opencandy has been found to distribute malware, one cannot completely rule out the possibility foul play in the use of this mechanism.