MAGMA network behavior classifier for malware traffic

MAGMA
Network Behavior Classifier for Malware Traffic

Enrico Bocchi*, Luigi Grimaudo, Marco Mellia, Elena Baralis

Politecnico di Torino, Italy

Sabyasachi Saha

Cyphort Inc., United States

Stanislav Miskovic, Gaspar Modelo-Howard

Symantec Corp., United States

Sung-Ju Lee

Korea Advanced Institute of Science and Technology, Korea

Abstract

Malware is a major threat to security and privacy of network users. A large variety of malware is typically spread over the Internet, hiding in benign traffic. New types of malware appear everyday, challenging both the research community and security companies to improve malware identification techniques. In this paper we present MAGMA, MultilAyer Graphs for MAlware detection, a novel malware behavioral classifier. Our system is based on a Big Data methodology, driven by real-world data obtained from traffic traces collected in an operational network. The methodology we propose automatically extracts patterns related to a specific input event, *i.e.*, a *seed*, from the enormous amount of events the network carries. By correlating such activities over (i) time, (ii) space, and (iii) network protocols, we build a *Network Connectivity Graph* that captures the overall "network behavior" of the seed. We next extract features from the Connectivity Graph and design a supervised classifier. We run MAGMA on a large dataset collected from a commercial Internet Provider where 20,000 Internet users generated more than 330 million events. Only 42,000 are flagged as malicious by a commercial IDS, which we consider as an oracle. Using this dataset, we experimentally evaluate MAGMA accuracy and robustness to parameter settings. Results indicate that MAGMA reaches 95% accuracy, with limited false positives. Furthermore, MAGMA proves able to identify suspicious network events that the IDS ignored.

Keywords: Network traffic modeling; Malware characterization; Malicious behaviors detection; Graph networks; Automatic classification

1. Introduction

Information security over the Internet remains a primary concern for consumers, enterprise, and government alike. Malware infiltrates and spreads over the Internet using complicated methods to hide its traffic among benign activities. Cyber-attackers continuously use sophisticated schemes to create new malware (nearly 200,000 new malware samples every day [1]) to avoid detection by security

*Corresponding author

Email addresses: enrico.bocchi@polito.it (Enrico Bocchi), luigi.grimaudo@polito.it (Luigi Grimaudo), mellia@tlc.polito.it (Marco Mellia), elena.baralis@polito.it (Elena Baralis), sabyasachi.saha@gmail.com (Sabyasachi Saha), stanislav_miskovic@symantec.com (Stanislav Miskovic), gaspar_modelohoward@symantec.com (Gaspar Modelo-Howard), sjlee@cs.kaist.ac.kr (Sung-Ju Lee)

measures. Recent industry reports disclose that zero-day vulnerabilities have increased by 61% [2], and existing antivirus software's detection rate of a newly created virus is less than 5% [3].

Different approaches have been taken by security practitioners, ranging from instruction set and code analysis, to traffic characterization of infected hosts (see Sec. 2 for more details). Several methodologies have been proposed, each targeting a specific family of threats, *e.g.*, botnets [4], click fraud [5, 6], exploit kit [7], or drive-by downloads [8]. Often, the detector leverages specific features that, while effective for the targeted malicious activity, become useless when considering a different type of threat.

In this paper, we have at the ambitious goal of designing a classifier that aims at detecting *any generic malicious pattern. We do not target a specific type of threats, but any family of malware.* To do so, we follow a data-driven approach. We consider the actual traffic observed in a live network where users access the Internet. From the packets and flows, we extract and log *events* with the use of a passive monitoring tool. An event could be a HTTP request, a DNS response, or simply a TCP flow going to a remote host using an unknown protocol. Millions of events are recorded per hour.

We consider a target event under analysis, that we call the *seed*, and we build a classifier that has to return a binary answer to the question: is the seed part of a benign or a malicious activity? To provide the answer, we adopt Big Data techniques where correlation among events is extracted to produce an augmented summary of the overall activity related to the presence of the seed. We represent such summary as a *Network Connectivity Graph*, *i.e.*, a graph where nodes and edges model the subset of events tightly correlated with the seed under study. The purpose of the Network Connectivity Graph is twofold. First, it provides a set of "forensic" information for the security analyst to support her in understanding the traffic involved in an accident. Second, using a supervised learning approach, the Network Connectivity Graph is used to extract a *model* of the typical behavior of malicious or benign events.

The result is MAGMA, MultilAyer Graphs for MAlware detection. MAGMA is a system able to process the enormous amount of traffic coming from operational large-scale networks, and to identify the subset of relevant events belonging to the same activity of the event under consideration.

MAGMA employs Big Data techniques based on a filtering and enrichment processes that leverages (i) temporal and (ii) spatial repetitiveness of events generated over time by multiple hosts. MAGMA looks for common patterns across different time snapshots generated by hosts connected to the network. Intuitively, it extracts those repetitive subsets of events that appear in most of the observation windows. In practice, as few as three observations of a seed are enough to trigger the analysis.

We use a real traffic collected from an commercial network where more than 20,000 households are connected to the Internet. By using a commercial Intrusion Detection System (IDS) as oracle, we obtain a list of more than 42,000 malicious events that belong to more than 150 different threats, including exploit kits, Drive-by downloads, malicious toolbars generating click-frauds, and hosts participating in botnets. Each presents very different characteristics. Out of those, about 40,000 (95%) meet the repetitive properties required to extract the corresponding Network Connectivity Graphs, which we consider representative of malicious activities. Following the same approach, we select a subset of random benign events and extract graphs representing benign activities.

This forms a labeled dataset that we use to train and test the performance of decision tree based classifiers.[1] We follow all the best practice dictated by the machine learning community to run a thorough evaluation. Despite the heterogeneity of both malicious and benign patterns, MAGMA achieves a classification accuracy higher than 95%. In addition, our performance evaluation reveals that MAGMA is very robust to parameter settings.

The contributions of our work, MAGMA, are as follows:

- We propose a methodology that extracts and represents the activity correlated with the occurrence of a *seed*, which allows the subsequent identification of benign and malicious traffic.
- We train a classifier that explicitly targets generic malware activity, and it is not tailored to a specific threat or malware class.
- We provide augmented information to the security analyst to uncover hidden malware behaviour and provide forensic information.

[1] Interested researchers that would like to access the dataset have to contact the authors and sign a Non-disclosure agreement (NDA).

This work extends our preliminary analysis of malware traffic that appeared in [9], where the Network Connectivity Graph concept was introduced. Here, we build upon it to engineer a behavioral classifier whose performance are evaluated and discussed thoroughly.

The paper is organized as follows: Sec. 2 presents the related works. Sec. 3 provides an overview of the scenario in which we operate detailing the available dataset. Sec. 4 provides an introductory description of the intuitions behind MAGMA design. Sec. 5 provides a formal description of the graph construction processes. Sec. 6 details the characteristics of the graphs, while Sec. 7 describes the supervised classifier design. Results are discussed in Sec. 8, before drawing conclusions in Sec. 9.

2. Related work

The increased popularity of the Internet and particularly the web to spread malware and infect computers, has led to vast amounts of research that attempt to identify malware using the traffic generated by such threats. The literature suggests for a variety of techniques that can be employed in this context. We focus our attention on three macrogroups of malware traffic detection techniques, being those the most related to our work. We discuss how previous works in the fields of graph-based detection approaches, multi-protocol traffic correlation, and infection phase identification relates to our research.

Graph-based Malware Detection

Previous work has explored graph-based approaches to detect malware. In [10], the authors build a bipartite graph consisting of domain names of failed DNS queries and host issuing such queries. Given this DNS failure graph, a graph decomposition algorithm is then applied to iteratively extract dense subgraphs. The intuition is that host infected by the same malware usually query for the same, similar or correlated set of domain names. The subgraphs generated are further classified in different categories and characterized by exploring their temporal properties. Similarly, building a relationship graph based on DNS historical data is proposed [11] where suspicious structural networks are identified based on two graph measures: graph density and eigenvector centrality and ground truth labels. Recently, a malicious domain detection system [12] is proposed. It leverages homophilic properties and ground truth labels to build a host-domain graph and adapt the belief propagation to estimate an unknown domain's likelihood of being malicious. Similarly, malicious hosts are detected using a semi-supervised, score-propagation algorithm that utilizes HTTP-communication graphs [13].

All these approaches restrict their efforts to a specific protocol to identify the suspicious graph entities. Often performance is assessed using synthetic datasets or benchmarks which are now outdated. MAGMA instead uses the data gathered from multiple protocols and from real traces to create a model for generic malware patterns. Moreover, the model characterizes both benign and malicious network activity and summarizes the commonalities exposed by the involved hosts.

Multi-protocol Traffic Correlation

Many efforts have focused on the analysis of a single protocol to identify network traces and patterns displayed by malware, while others have considered a set of protocols to achieve the identification of malware. In the first case, HTTP and DNS are two of the most analyzed protocols among malware threats to communicate with victims or between malicious peers. Several detection techniques have been proposed, exploring different ways to characterize the behavior of different malware threats on HTTP [12, 14, 15, 16] or DNS [11, 17, 18]. Examples of proposals following a multi-protocol approach, such as the one presented in MAGMA, include [19, 20, 21, 22].

The popularity of HTTP on the web has made it the preferred protocol for malware creators and as such, the target for researchers to analyze and detect malware. [14] presents a system to identify malicious drive-by download activities by exposing the distribution networks necessary to distribute malware thru HTTP. Similarly, [12] and [15] propose classifiers based on features from web domains and URLs to detect malware activity. [16] proposes the use of *n-gram* techniques to filter out the majority of benign HTTP traffic and detect malicious HTTP transactions to be processed with more costly techniques. Systems that analyze the DNS protocol, usually look at failed DNS queries [17, 11], as this activity can lead to the existence of malware using domain generated algorithms (DGA). Other systems analyze the flow-level information from the DNS traffic and look for statistical patterns [18]. The problem with systems relying on a single protocol is their limited scope, as malware can switch

from protocols, and the required semantic understanding of the particular protocol considered.

In comparison, other approaches have evaluated multiple protocols to detect malicious activity. A seminal work in this area is [19], where the lifecycle of botnets is modeled according to a set of phases, with different application protocols involved. An interesting approach is used in [20, 21, 22], where traffic information is presented through generic packet information such as length sequences and encoding differences, allowing then to represent the malware activity observed in different protocols. All of these multi-protocol approaches have the limitation of targetting specific type of malware. Our approach is designed to target any malware type, whose model is extracted from actual traces rather than synthetic datasets.

Infection Phase Identification

Several types of malware usually exhibit specific phases during the infection process. For example, botnets are commonly used to distribute malware and involved several phases, including redirection of webpages and communicating to a command and control server. Multiple approaches have looked for specific phases of the infection lifecycle, in order to detect the existence of malware. Examples include DNS queries failures [17, 11, 10], HTTP connections to domains [23, 24] and command-and-control (C&C) communication [18, 25, 20]. As many malware nowadays constantly change its behavior, relying on a single phase presents a strong limitation for detection systems. In contrast, MAGMA is agnostic about possible phases exhibit by malware, by inspecting all alerts presented by a detection system and looking for common and repetitive patterns. In common with this body of work, we use traffic traces collected in the wild to obtain realistic cases of actual malware.

In summary, the consideration of a single protocol or malicious threat, as well as a single phase of the threat, presents challenges and demands for a different approach. We fill this gap by designing a flexible method that considers multiple types of threats and leverages the analysis of actual traces, to provide accurate detection along with detailed information of the malicious activity.

3. Scenario and Dataset

In this section we provide an overview of the scenario we face, detailing the actual data the system is offered and characterizing the malicious fraction of traffic.

3.1. Scenario

We consider a scenario in which a sniffer passively monitors the traffic generated by a large group of hosts, *e.g.*, hosts in an enterprise network, or households connected to a Point-of-Presence (POP) of an Internet Service Provider (ISP). The sniffer extracts information from the packets and logs them in a file where each row corresponds to a different *event*. We assume that, for each TCP and UDP connection, the sniffer logs the flow identifier (*i.e.*, a tuple made by source and destination IP addresses, source and destination ports, and protocol type), the timestamp of the first packet, the flow duration, the number of exchanged packets and bytes, etc. For some protocols, the monitor provides multiple events with detailed information. For instance, it annotates each HTTP request/response with the requested URL, user-agent, content-type, server response status code, etc. A DNS event exposes the requested hostnames along with all returned IP addresses by the resolver. We assume that an oracle (*e.g.*, an Intrusion Detection System – IDS) has processed the traffic to label malicious events.

Consider the timeline generated by a single host reported in Fig. 1. It details the events generated by Internet applications. DNS and HTTP events are reported using specific markers, while other protocols are reported as generic TCP/UDP events. The user is visiting some web pages (*e.g.*, `acme.org`), while an email client is polling a mail server for new messages. Benign events are reported in the bottom part of the timeline. Unfortunately, `acme.org` is hosting a Drive-by download page. Events on the upper part are due to the malicious activity in which the host is unknowingly fooled to download a malware from a malicious JavaScript contained in the web page. We observe the download of the JavaScript object, followed by the download of the malware executable. Once running on the host, the malware periodically contacts (via HTTP in our example) a Command and Control (C&C) server whose name is rotated among random generated names [26]. The periodic polling is visible as a sequence of failed and successful DNS requests, and some HTTP traffic to the C&C node.

The challenge is how to isolate the events that are possibly correlated with a specific malicious/benign activity from the "background" noise caused by other events. All the events are indeed exposed by

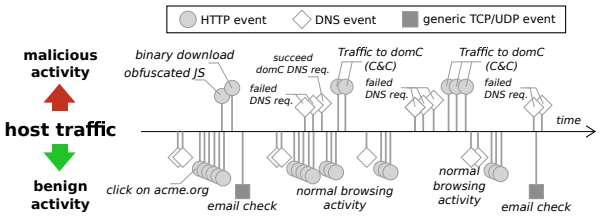


Figure 1: Example of events generated by a host as seen from the network.

the system, and, as we will see in the next section, only a handful of them are actually malicious.

3.2. Available Dataset

We consider a vantage point located in a commercial ISP where approximately 20,000 customers are connected. Most of them are residential customers, connected via ADSL modems to the monitored point. Each customer's modem is given a static IP address, which is used to identify the traffic generated/directed to all active terminals in the household. In the following, we use the term "user" to refer to traffic exchanged by a single household (IP address).[2]

We leverage a traffic trace obtained during one entire day in April 2012. A commercial monitoring tool processes the packets in real time, and extracts a text log file in which each TCP and UDP flow is logged. For each flow, a *record* is stored detailing the network flow identifier (*i.e.*, a tuple made by source and destination IP addresses, source and destination ports, and protocol type), the timestamp of the first packet, the total number of packets and bytes sent and received, and the application protocol used (*e.g.*, HTTP, BitTorrent, etc.).

In case the application protocol is HTTP, the record further reports the server hostname, object path, user-agent, content-type, response status, and content-length directly extracted from the HTTP header [27]. In case multiple HTTP transactions are present in the same TCP flow (*e.g.*, due to HTTP-persistent option), multiple records are logged. Similarly, for each DNS transaction, the tool logs the requested hostname, the set of IP addresses returned by the resolver, or the response code in case of an error (*e.g.*, *Non-Existent Domain*) [28]. IP addresses of customers are anonymized using irreversible hashing functions, and we adopt the best practices to remove any sensitive information for the current legislation.

[2]Given the popularity of NAT (Network Address Translation) at home, the ADSL modem IP address identifies traffic exchanged by all devices accessing the Internet at each customer household.

Table 1: Dataset summary.

	All Traffic		Flagged Traffic	
Class	Users (%)	Records (%)	Users	Records
HTTP	16,217 (*79.1*)	39.7 M (*11.8*)	1,308	42,007
Email	3,640 (*17.7*)	880.7 k (*0.2*)	-	-
Chat	3,045 (*14.8*)	100.8 k (*0.03*)	7	1,467
P2P	3,163 (*15.4*)	17.1 M (*5.05*)	-	-
OthTCP	18,806 (*91.8*)	22.7 M (*6.7*)	24	76
DNS	15,164 (*74.1*)	30.7 M (*9.3*)	-	-
VoIP	8,371 (*40.8*)	80.5 k (*0.02*)	-	-
OthUDP	17,664 (*86.2*)	224.6 M (*66.8*)	-	-
Total	20,486	336.1 M	1,321	43,550

P2P = (eMule, BitTorrent),
Email = (SMTP, POP3, IMAP),
Chat = (XMPP, YahooMsg, MSN, IRC)

While a characterization of the overall traffic is out of the scope of this work, we provide some statistics to show the huge volume, heterogeneity and complexity of the data that the system has to face. More details can be found in our previous work [29]. Table 1 provides a summary. Focusing on the first three columns, we observe a total of 20,486 users generating about 336 million flows over the whole day. About 20% of those are related to HTTP and DNS records, with a large majority classified as "Other TCP" due to TLS/SSL (HTTPS) traffic, and "Other UDP" due to and Peer-to-Peer applications.

Some traffic is machine generated, *e.g.*, keep-alive messages, software updates, or cloud-based applications synchronizing their status. Some use proprietary protocols and generate little traffic. Other exchange information frequently inflating the number of events. Considering user-generated traffic, we observe some heavy users that generate thousands of HTTP requests, run P2P applications, play online games, and use multiple devices at the same time. Other users, instead, just have their mobile phone periodically checking the email.

3.3. Traffic Volume of Malicious Activities

In parallel to the monitoring tool, a commercial IDS processes the packets producing alerts if a network activity matches any rule in its database. For each alert, the IDS specifies the network flow identifier it relates to and a *threat-ID*, *i.e.*, a numerical

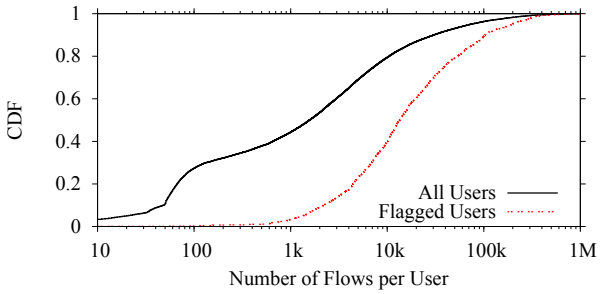


Figure 2: CDF of the total number of records per user.

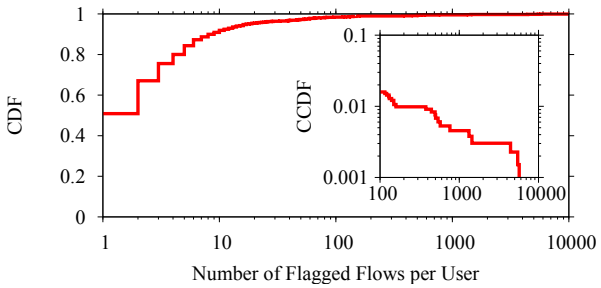


Figure 3: CDF of the number of flagged records per user.

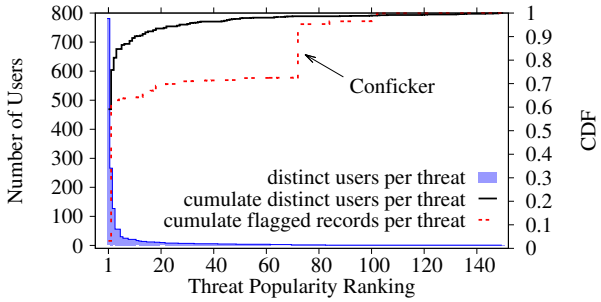


Figure 4: Overall threats statistics.

code that identifies a particular threat. For some threat-IDs, a name and description of the malicious activity is available, detailing the severity of the threat and which component of the host is vulnerable, *e.g.*, browser, operating system, etc. Other threat-IDs are instead little documented. The IDS is very conservative in triggering alerts, and hence it possible that some malicious event do not trigger any alerts (*i.e.*, false negative). Conversely, every alert raised is related to malicious activities.[3] In the following, we use the IDS as oracle, i.e., events are labeled as benign or malicious according to the IDS labels.

We consider each record in the log file as a different *event*. By matching the flow identifiers, alerts are linked to records, so that records can be flagged as *malicious*. We refer to a *flag* as a log record for which the IDS triggered an alert, and to a *flagged user* as a user exhibiting at least one flagged record. A *non-flagged user* is instead a user for which no alerts are risen in the whole day. Rightmost columns in Table 1 details the flagged events. Among all users, 1308 (6.4%) of them exhibit some malicious activity, with more than 150 different threat-IDs being reported. Only 43,550 flags are raised by the IDS. That translates to a negligible 0.013% of all traffic. Most of these records correspond to HTTP traffic, with the exception of some IRC (Internet Relay Chat) and RPC (Remote Procedure Call) activities, which are known to be commonly abused by malicious adversaries. This highlights the very stealthy and low rate activity that malware is typically generating, and also confirms the conservative design of the IDS.

We dig into more details to observe if it is possible to pinpoint differences between flagged users and non-flagged users. Specifically, we investigate on the occurrences at which flagged events appear. The intuition is that the more flagged events occur, the easier it should be to spot them in the traffic aggregate. Fig. 2 shows the Cumulative Distribution Function (CDF) of the amount of total records logged for all users, and for the subset of flagged users. Results show that the flagged users generate much more traffic than the rest of the population. One would think this is due to the extra traffic generated by the malicious application running at the infected client. However, flagged users present a very small number of flags. This is detailed in Fig. 3 that reports the number of flags per flagged user. We find that 75% (92%) of the users show less than 3 (10) flags in the whole day, and only two users show more than 1,000 flags. As such, while malicious activities can inflate traffic volume, in general the rate of malicious records is very limited.

3.4. Threat diversity

To investigate the threats diversity and the traffic they generate, Fig. 4 reports different statistics on the alarms raised by the IDS. Consider first the shaded histogram. It shows the number of users affected by each threat. Threats are sorted by popularity on the x-axis. Overall, the IDS detects 151

[3]In general, we cannot exclude that some few false positives are present. However, those appear to be marginal.

distinct threats. Their popularity is highly skewed, with the most popular affecting about 800 (61%) flagged users, and 129 threats affecting less than 10 users each. Despite the limited number of alerts, this highlights a very diversified scenario of malicious activities.

Next, consider the red dashed line of Fig. 4. It reports the CDF of the number of flagged records contributed by each threat. As expected, the majority of these records are related to the most popular threats. However, the distribution has several steps, indicating that some are more "chatty" than others and produce many alarms even when only few users are involved. This is the case of Conficker [30], which infects only two users, yet it contributes to 23.3% of all flagged records. Note that Conficker is a worm that was first detected in 2008 but is still one of the most popular threats [31].

The solid line in Fig. 4 shows the CDF of the number of distinct users involved in each threat. In other words, we progressively add the fraction of new users that were not affected by previously considered threats. Notice how the distribution presents several "plateaus", indicating that all involved users were already accounted by previous threats. We find that 23% of users are flagged with multiple threats. This is due to users being infected by different malware.

To give more insights about how diverse and heterogeneous the malicious events are in the wild, Tab. 2 offers a deeper characterization of the 15 most popular threats. It details the popularity ranking of the threat, the number of infected users and the number of flagged records it generates. For some threat-IDs, the IDS provides limited information on the malicious activity and hence we adopt generic names, *e.g.*, Threat-A. Notice how some threats presents a *type*. This corresponds to the ability of the IDS to identify different variant of the network traffic of the same threat.

Some examples of threats include Drive-by downloads and Exploit Kits (EKs), which are among the most popular threats. The *DynDNS activity* corresponds to traffic toward hostnames registered with DynDNS services that hide control messages (*e.g.*, periodic communications to check network connectivity). *Skintrim* and *Tidserv* are two popular trojans that can trigger the download of other malwares through backdoors. *Toolbar activity* threats are related to the Ask.com toolbar that are triggered by the download of unwanted advertisement objects or perform iframe injections in the browser.

Table 2: Most Popular Threats.

#	Name	Users	Flags
1	Drive-by download [type1]	781	1427
2	DynDNS activity [type1]	266	26270
3	Blackhole EK [type1]	127	158
4	Skintrim [type2]	56	301
5	Skintrim [type3]	56	301
6	Facebook plugin attack	30	31
7	Threat-A	25	27
8	Blackhole EK [type2]	25	25
9	Toolbar activity [type1]	21	105
10	Threat-B	21	23
11	Threat-C	21	22
12	Toolbar activity [type2]	17	19
13	Drive-by download [type2]	15	33
14	Tidserv	14	228
15	Threat-D	14	470

3.5. Events Popularity and Whitelisting

Fig. 5 shows the HTTP event popularity, *i.e.*, the fraction of hosts that accessed a given URL (with stripped parameters). Note the log scale on x-axis. Fig. 5 shows the classic heavy tailed popularity. Top URLs are clearly very common among most of the hosts. Those include social network buttons (*e.g.*, `www.facebook.com/plugins/like.php`), analytics services (*e.g.*, `www.google-analytics.com/ga.js`), software update check (*e.g.*, `download.windowsupdate.com/v9/windowsupdate/redir/muv4wuredir.cab`), etc. Red triangles highlight those events that are considered malicious by the oracle. The most diffused type of attack – a Drive-by Download threat – infects about 800 hosts (3.8% of hosts). The huge tail confirms the intuition that most of URLs are accessed by few hosts only.

Leveraging the stealthy nature of malicious traffic, and thanks to the fact that few users are actually infected by a given malware, we adopt whitelisting as a common technique used to both reduce the amount of information to process, and to discard data that would possibly pollute the analysis. We built a whitelist that targets very popular events, which add little information or create noise. Instead of creating a manual list of popular and benign events, we opt for a *dynamic* and *context-aware* approach. MAGMA builds a whitelist based on events popularity among clients, and selects the top-k elements to be ignored during the processing. We whitelist single HTTP events and not the entire websites, as it is known that malware can be hosted

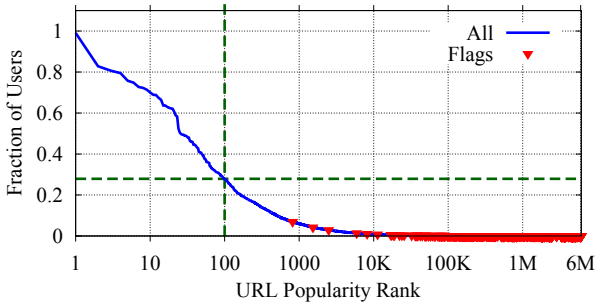


Figure 5: Popularity of HTTP objects.

and distributed also from benign services. We pick top most popular 100 HTTP events (highlighted in Fig. 5), i.e., we filter those events that are common to more than 23% of users. This is equivalent to assume that the most popular malware has infected less than 23% of population.

4. Methodology overview

Before presenting the details of MAGMA, we provide an overview on the adopted methodology and the intuitions behind its design. Through a high-level description of the overall workflow, we aim at defining conventional names that we use in the forthcoming sections of the paper. We follow a data-driven approach, where filtering and correlation phases allow us to extract information from the huge amount of data at our disposal, as is common practice in a Big Data scenario.

4.1. Single Host Connectivity Graph

Consider Fig. 6. It depicts the procedure used to extract the *Host Connectivity Graph* (Host-CG) from those hosts presenting the *seed* among the events they generate. We refer to the seed as the event we want to classify as benign or malicious.

Three steps are executed: (i) Snapshots extraction, (ii) Per-layer common patterns mining, and (iii) Host-CG creation.

Snapshots Extraction. Consider the seed event, and the timeline around it. Intuitively, events close-in-time with the seed are likely to be related to it (*e.g.*, a DNS request followed by several HTTP transactions could be identified as a typical pattern). For each instance of the seed, we extract a *snapshot* defined as the *ordered* set of events occurring in the temporal window centered on the seed.

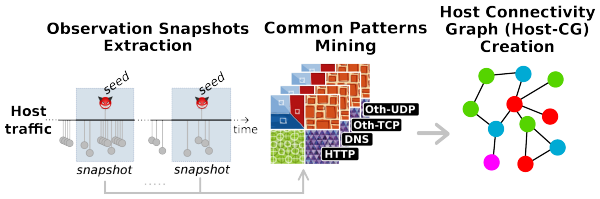


Figure 6: Host Connectivity Graph generation.

Common Patterns Mining. We then look for common *patterns* across snapshots. A pattern is defined as the unordered set of events that appear across multiple snapshots. We extract separate common patterns by processing the host traffic considering *layers* in isolation. The traffic generated on each layer corresponds to all events of a specific protocol so that HTTP, DNS, other-TCP (*i.e.*, all TCP communications except HTTP on port 80), and other-UDP (*i.e.*, all UDP communication except DNS traffic) events are separately analyzed. The choice of separately analyzing layers originates from the fact that each protocol has some peculiarities that we would leverage. For instance, in the HTTP layer, we are looking for common and repetitive patterns. On the DNS layer instead, failed DNS requests may be more interesting than successful DNS requests.

Host Connectivity Graph. For each layer, we represent the common pattern as a graph, where nodes and edges are defined considering specific layer properties. For instance, focusing on the HTTP layer, URLs are represented by separating server hostnames and object paths using two nodes. An edge between the hostname and the path represent a URL. The resulting graph captures the website structure. Similarly, in the DNS layer, requested hostnames are linked to the server IP address(es) returned by the resolver. As the last step, we collapse the per-layer graphs into a single *Host Connectivity Graph*. This is done by linking common nodes in multiple layers. For instance, the hostname in the HTTP layer is linked to the hostname node in the DNS layer graph.

4.2. Seed Connectivity Graph

We now leverage the fact that the same seed can be present in the timeline of *multiple hosts*. We exploit this to gain a broader view of the common ac-

tivity using the "spatial" diversity provided by multiple hosts. To do so, we merge multiple Host-CGs into a single *Seed Connectivity Graph* (Seed-CG). This can be done by taking the union, the intersection (or implementing more complex strategies between these two extremes) of all nodes from Host-CGs. The process of generating the Seed-CG aims at creating a rich but compact representation of the common events generated by multiple hosts, *i.e.*, combining common patterns across distinct users. A Seed-CG is thus a summary of events correlated with the seed.

4.3. MAGMA supervised classifier

Consider now Fig. 7. It shows the processes needed to train a classifier able to distinguish between malicious a benign Seed-CGs. As first step, we use the alarms raised by the commercial IDS to build two distinct sets of Seed-CGs, one containing CGs generated by malicious seeds, the other containing CGs of benign seeds. We then leverage the descriptiveness of Seed-CGs to define a set of features with which the classifier can be trained and tested. These include: (i) Graph topology properties (*e.g.*, number of nodes, node degrees); (ii) HTTP header parameters (*e.g.*, response status, distinct user-agents, content-type); (iii) Syntax properties extracted from the names (*e.g.*, string length, number of subdomains, digit–characters ratio); and (iv) Occurrence properties (*e.g.*, minimum, maximum, average recurrence of specific events). Some of these features are driven by the domain knowledge, while others are generic and considered to avoid biasing towards a specific threat. The complete set of features is described in App. Appendix A.

As last step, we run an exhaustive set of experiments to assess the accuracy of the classifier under a variety of conditions, and its sensitivity to parameter setting. The results is MAGMA, a behavioral classifier able to label Seed-CGs as malicious or benign with high accuracy.

5. Building the Connectivity Graph

This section discusses the design choices taken and the parameters to control when creating a Network Connectivity Graph. The pseudo-code in Alg. 1 details the overall approach.

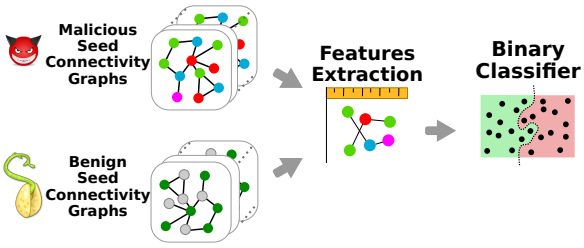


Figure 7: MAGMA classifier overview.

Algorithm 1 Create Network Connectivity Graph.

input args	s: seed H: set of hosts Δ: snapshot duration
output	Seed Connectivity Graph

```
1:  procedure graphLayer (s, S, layer):
2:    P = findCommonPattern (s, S, layer)
3:    return fromPatternsToGraph (P, layer)

4:  procedure hostConnectivityGraph (s, h, Δ):
5:    S = getSnapshots (s, h, Δ)
6:    g_HTTP = graphLayer (s, S, 'HTTP')
7:    g_DNS = graphLayer (s, S, 'DNS')
8:    g_TCP = graphLayer (s, S, 'TCP')
9:    g_UDP = graphLayer (s, S, 'UDP')
10:     return connectLayers (g_HTTP, g_DNS, g_TCP, g_UDP)

11: procedure seedConnectivityGraph (s, H, Δ):
12:   G_s = ∅
13:   foreach h ∈ H:
14:     G_s ← hostConnectivityGraph (s, h, Δ)
15:   return fuseGraphs (G_s)
```

5.1. Snapshots Extraction

The first step to process the traffic generated by each host (h) among the set of hosts (H) exhibiting the seed (s) is to extract an observation snapshot for each occurrence of the seed. We define the parameter Δ that controls the duration of the snapshots. In particular, a snapshot is composed by all events occurring in the interval $\pm\Delta/2$ centered around the seed. In case consecutive snapshots overlap, we apply two strategies depicted in Fig. 8 to solve the conflict. If the overlapping window lasts for less than $\Delta/2$, the two snapshots are merged. Otherwise, the overlap is split into two halves, each associated to a different snapshot. These operations are executed by *getSnapshots*() (Alg. 1 line:5) that receives the seed (s), a host (h) presenting at least one instance of s, and the snapshot duration (Δ) as inputs. It returns the set of snapshots found (S).

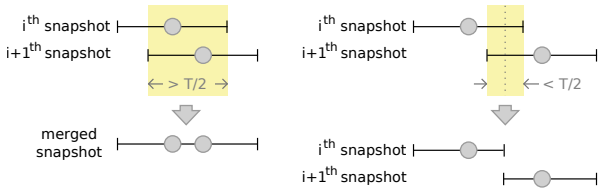


Figure 8: Snapshots creation when consecutive snapshots overlap.

Different values of Δ can lead to different results. In particular, the larger the Δ (*e.g.*, hours), the more the snapshots will merge. This results in less snapshots on which perform pattern mining, producing "noisy" data since not many events are filtered. Conversely, a small value of Δ (*e.g.*, seconds) might be too conservative. In the following, we set $\Delta = 30$ minutes. A complete sensitivity analysis is reported in Sec. 8.

5.2. Patterns Mining

We use the frequent itemset mining technique to extract common patters [32]. This technique works on unordered sets of simple objects (*e.g.*, strings). Snapshots however, correspond to ordered sequences of events that may appear multiple times. We thus map each event to an *item* based on the event properties. Specifically,

- A HTTP item is represented by HTTP URLs, *e.g.*, `http://domain.com/path/file.ext`.
- A DNS item combines the requested hostname, and either the list of returned IP addresses or the query response error code, *e.g.*, *DoesNotExists.com – NXDomain*.
- TCP and UDP items are represented by the server IP address and the server port being contacted, *e.g.*, *10.20.30.40:443*.

For each snapshot, we create a *transaction* containing the set of *distinct* items. We look for common *itemsets*, *i.e.*, sets of items common across multiple transactions. A *support* value is computed for each itemset and indicates the fraction of transactions containing the specific itemset. For a given support value, the itemset presenting the highest number of items is said to be *closed*. The closed attribute implies that there is no other itemset made by more items with the same support. An itemset is "frequent" if its support is greater than or equal to MinSup.

Itemsets with a number of items smaller than MinLen could be discarded. By setting MinLen=1, frequent itemsets are equivalent to simple frequent items. For MinLen=2, at least pairs of items are considered. For instance, consider `acme.org/index.html` and `acme.org/logo.png` that appear in 70% and 45% of snapshots, respectively. The itemset (`acme.org/index.html`, `acme.org/logo.png`) may appear from 15% to 45% of snapshots.

Looking for all itemsets is a #P-hard problem [33], but well-known algorithms efficiently compute frequent closed itemsets. Among those, we rely on the Carpenter algorithm [34], which is specifically designed for datasets made of few transactions (*i.e.*, snapshots) that have a huge number of items (*i.e.*, events). A MapReduce implementation is available [35].

MAGMA looks for *frequent closed itemsets* that, for simplicity, we call *patterns*. Patterns are extracted by *findCommonPatterns*() (Alg. 1 line:2), that receives the seed (s), the set of snapshots (S) and the layer (layer) to process. It returns the pattern (P). The pattern extraction process is guided by the definition of the value of MinSup: all events that do not appear with frequency at least MinSup are discarded. We set MinSup = 1/2, *i.e.*, for each host, we discard all events not appearing in at least half of the snapshots. Sensitivity analysis in detailed in Sec. 8.

5.3. Host Connectivity Graph

As previously discussed, we individually process each layer to create separate graphs. The *graphLayer*() (Alg. 1 line:1) extracts patterns for a specific layer and maps them into a graph. This mapping exploits a subset of the events properties, as follows:

- The HTTP layer has two types of nodes: hostnames and object paths. An edge connects the hostname and the object path to compose a URL.
- The DNS layer has three types of nodes: server hostnames, server IP addresses, and DNS error codes. An edge connects the hostname to either the IP addresses returned by a DNS response, or to an error code.
- The TCP and UDP layers have two types of nodes: server IP addresses and server ports. An edge connects the two to represent a TCP or UDP connection.

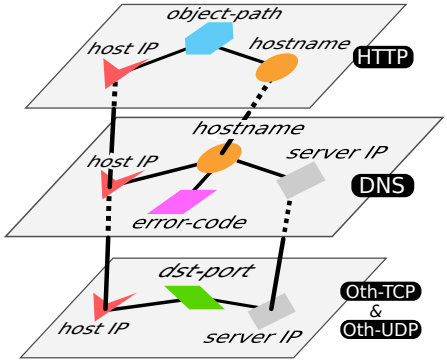


Figure 9: Graph layers nodes and multi-layer connections.

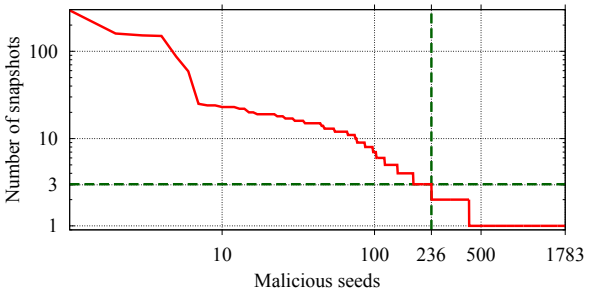


Figure 10: Number of snapshots per malicious events. Only those with frequency higher than $k = 3$ are processed by MAGMA.

Different graph layers are combined in a single Host-CG using *hostConnectivityGraph*() (Alg. 1 line:4). The function starts by extracting the snapshots (S) related to the seed. The shapshots are then processed to extract the graph layers (g_{HTTP}, g_{DNS}, g_{TCP}, g_{UDP}) through calls to *graphLayer*(). The separate graph layers are finally integrated to form the Host-CG using the *collectLayers*() function, which looks for common nodes across the layers and links them as represented in Fig. 9. Notice that each graph layers contains the host (h) IP address by construction.

5.4. Seed Connectivity Graph

To provide the global view of the common behavior gained by observing multiple hosts, we combine all Host-CGs. This operation is performed by the *seedConnectivityGraph*() (Alg. 1 line:11) function. For each host (h) among the subset presenting the seed (H), the function creates the Host-CG calling *hostConnectivityGraph*(). All the output graph are collected into the set G_{hosts}.

The graphs are finally merged using *fuseGraphs*(). This operation can consider different strategies. For instance, applying a strict intersection would retain only nodes appearing in all Host-CGs. In the worst case, this results in a Seed-CG containing only the original seed. More complex strategies can instead compute node and link frequency or popularity among hosts, and discard those below a given threshold of MinPopularity.

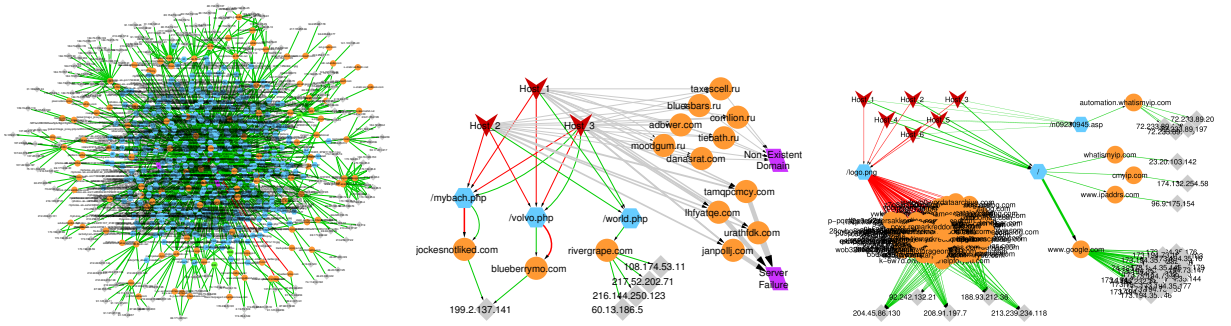
In the following, we consider the strict intersection across Host-CGs as the default choice, *i.e.*, MinPopularity=1. A detailed discussion about the impact of this choice is deferred to Sec. 8.

6. CG Characterization

We next evaluate the benefits and properties of CGs created by MAGMA. First, we identify the amount of events eligible of becoming seeds. Recall that MAGMA's CG construction requires a recurrence of seed events over time and user population. The presence of recurrent events matches the basic properties of malicious activities, such as periodic reporting to the Command and Control center, or recurrent attempts to identify new victims. We also expect that malware creators would try to disguise such repetitiveness as much as possible. In our data, indeed, we found 820 malicious hosts that had only one flagged event. If analyzed in isolation (on per host basis), these events would evade MAGMA's detection by not having any recurrence. Yet, by looking at correlation among different hosts, MAGMA is able to find commonalities between these events and tie them to a common malicious activity.

Fig. 10 reports the number of snapshots that can be associated to each unique malicious event. By considering 1,783 unique malicious events, we found that 236 events can be uniquely associated to at least three independent snapshots. Setting minSnapshots=3, these events become fully characterizable by MAGMA. In fact, looking at the absolute numbers, MAGMA can provide insights in 95% of malicious snapshots in our dataset (40k out of 42k flagged records, cfr. Table 1). We also emphasize the diversity of MAGMA's characterization capabilities, noting that the events in scope correspond to 60 different types of threats.

In summary, Table 3 details the amount of events eligible to be seeds for both benign and malicious events when minSnapshots=3. We observe that only 509,700 (8.3%) benign events are repetitive enough



(a) Initial graph generated from the traffic a single host in a single snapshot. The overall picture is unreadable.

(b) CG for the malicious seed `http://jockesnotliked.com/mybach.php`. Red edges are flagged by the oracle as malicous. Green and grey edges are instead identified by MAGMA.

(c) CG for *Cycbot Backdoor Activity* trojan. Notice the *fast-flux* domain name shuffling (orange circles in lower-left part), and the IP address detection webpages (top-right) to assess victims' reachability.

Figure 11: Examples of Network Connectivity Graphs.

Table 3: Eligible seeds with minSnapshots = 3.

	Unique seeds			Threat-IDs		
	All	Elig.	(%)	All	Elig.	(%)
Benign	6,111k	509,700	(*8.3*)	-	-	
Malicious	1,783	236	(*13.2*)	151	60	(*39.7*)

to be considered by our system. This is largely expected: benign traffic is mostly related to activities of human users, and would not access identical objects as recurrently as malware. To test our system, we next use all 236 malicious seeds and combine them with 664 randomly selected benign seeds.

6.1. Examples of CGs

In Fig. 11, we show and example of the corresponding CGs that MAGMA produces. Fig. 11(a) is an effective input element to the CG creation. It represents all the events present in a single 30 minute snapshot around the malicious seed `http://jockesnotliked.com/mybach.php`. Obviously, this graph is very difficult to interpret since the malicious activity is mixed with ordinary user-generated events due to web surfing.

Fig. 11(b) depicts the final Seed-CG generated by MAGMA after the filtering and enrichment process. Note the original malicious seed on the leftmost part of the graph. The final picture is much clearer, identifying three suspicious clients (red markers). Red edges highlight the events that are malicious according to our oracle. The richness of MAGMA's indications stems from the augmented context that we provide about the activity of these clients. First, the clients access three URLs (blue hexagons) hosted by three hostnames (orange circles) - all of which now become an indication of a suspicious infrastructure. Next, two servers use the same IP address (gray diamonds) suggesting a potential obfuscation by hostname flipping. The third host is distributed over several mirrors whose IP addresses belong to very different subnets, a hint of non-structured infrastructure or zombies that were previously infected. Finally, the rightmost part of the graph shows another layer of information, indicating multiple failures of DNS queries (purple boxes). This reaffirms our suspiciousness.

Apart from providing more context for the malicious activities, MAGMA also discovers new malicious objects and improves the flagging consistency of our oracle. For example, referring to Fig. 11(b), MAGMA consistently includes the object `bluberrymo.com/volvo.php` in the malicious graph, while the IDS occasionally missed it. Since we set MinPopularity=1, all the events are common to the three hosts, strengthening the correlation with the seed and providing to the security analysis a richer context to investigate on the incident. MAGMA also discovered a new object `rivergrape.com/world.php` for which we confirmed its maliciousness across several other security tools such as VirusTotal [36].[4]

[4]VirusTotal is a free service that scans submitted URLs to detect malicious URLs. VirusTotal is not a defense tool per se, but it leverages threat definitions of more than 65 commercially available antiviruses and IDS suites.

As second example of representative CG related to a malicious seeds, Fig. 11(c) details the results for `*/logo.png`, which is linked to the *Cycbot* botnet. Cycbot is a backdoor trojan that allows cybercriminals to access infected computers remotely. This causes victims' hosts to be exploited by malicious adversaries for large-scale attacks, and to potential leakage of personal information. The CG in Fig. 11(c) offers interesting insights. For instance, more than 80 hostnames serve the malicious file *logo.png* (cloud of orange circles in bottom left part of the graph). All those hostnames have random strings that are made of both characters and numbers, and are hard to code with regular expressions. This technique, known as *fast-flux*, allows attackers to hide malicious infrastructures by generating random hostnames. Those are registered to the DNS and lately removed with a high frequency. This makes the detection harder, circumvents blacklisting, and guarantees a longer lifetime to the infrastructure. Considering only second-level domains, they present appealing names acting as a lure for potential victims, *e.g.*, `faststorageonline.com`, `phonegamescatalog.com`, `wwwmp3archives.com`, etc. The entire set of domain names is hosted on 5 IP addresses. Those addresses are not organized in a structured CDN (*e.g.*, they do not belong to the same subnet), suggesting for the usage of infected servers acting as C&C nodes.

Moving to the right part of Fig. 11(c), we observe some benign objects. Those are indeed perfectly legitimate services, and thus any IDS would not block them. However, those are contacted as part of the malicious activity of the infected hosts. First, look at the bottom-right part of the CG and observe how the malware is checking victim's Internet connectivity by visiting the `www.google.com` homepage. This is a first test to gather connectivity properties of the victim. Look now at the top right URLs. Contacted websites host services aimed at the discovery of the public IP address of the host, and the malware is abusing of these legitimate services for its goals. Such behavior is coherent considering the intent of the malware we are facing. Being a backdoor trojan, the infected clients form a botnet. They have to be reachable by the cyber-criminals to control them. Eventual reachability issues, *e.g.*, NATs or firewalls, might preclude the access to the host. Thus the malware tests connectivity abusing of the above mentioned services.

For comparison, graphs related to benign seeds (not reported here due to lack of space) look radically different: they typically show legitimate CDNs (many IP addresses belong to only three subnets), and legitimate websites (many objects hosted on the same domain), no failed DNS events, etc.

6.2. Overall analysis of the CGs

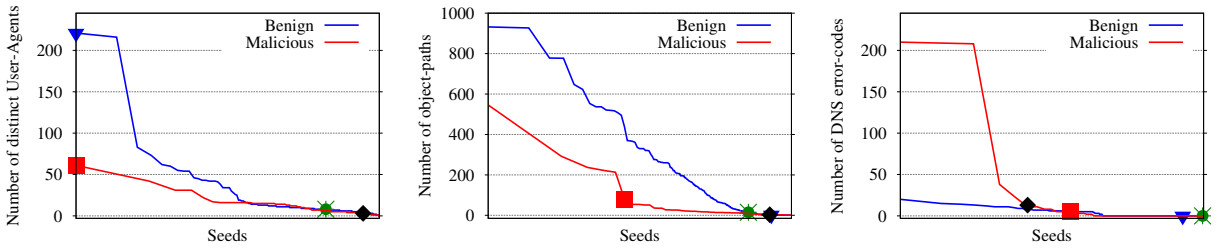
We now offer a thorough analysis of URLs MAGMA identified as belonging to Connectivity Graphs. In particular, for each of the 236 malicious seeds, we extract the corresponding CG. Overall, 5213 unique items appear, out of which 2393 are HTTP requests that the IDS oracle does not flag. For each of these requests, we investigate if they are malicious or benign. For this purpose we use again the online service VirusTotal. In addition, we also double check each of them using Snort [37]. Results show that 1580 (66%) of the discovered items are labeled as malicious by either VirusTotal or Snort, thus confirming the items in the CGs generated by MAGMA form a better and more complete picture than the one originally offered by the oracle.

CGs also include 1114 benign objects that no IDS or antivirus flags. While these may seem false positives, it is often not the case. For instance, we have seen perfectly legitimate URLs being included in a malicious graph. We have already verified that benign URLs are indeed part of malicious activities that run checks (*e.g.*, the `www.google.com` or `whatismyip.org` for the Cycbot case, Fig. 11(c)). Furthermore, infected hosts often contact legitimate servers to run DDoS attacks, or to generate fake clicks on advertisements, or in general to spread the malware and run attacks. IDSes cannot flag these events as malicious, since the corresponding services are not malicous. In contrast, MAGMA is capable of discovering such interactions between malicious and legitimate infrastructures, and to expose them to the security analyst.

In summary, GCs offer expressive information to characterize and understand activities related to malicious events. While this is useful for the analyst to understand an accident, the information they provide can be used to train classifiers and to extract signatures to spot new malicious activities.

6.3. Impact of Pattern Filtering

Before training the classifier, we first study the volume of information that Seed-CG creation process extracts from single seeds - malicious or benign.



(a) Number of distinct User Agents. (b) Number of object-paths. (c) Number of DNS error-codes.

Figure 12: Feature distributions for malicious and benign CGs. Due to the variety of patterns, the usage of simple heuristics is not suitable to separate benign and malicious CGs. This suggests for the need of a classifier able to combine different planes of information.

Table 4: Average number of nodes of different types among Seed-CG for malicious and benign seeds.

	Malicious			Benign		
Type	c1	c2	c3	c1	c2	c3
Object-path	6.6	14.9	2351.4	18.5	56.3	3320.1
Hostname	7.0	16.8	691.5	8.1	28.9	781.2
Server IP	19.9	95.9	3423.0	39.0	161.2	6260.2
Dst-port TCP	0.2	0.6	79.9	0.8	4.1	194.2
Dst-port UDP	2.0	27.8	1335.1	2.8	42.2	3129.5
DNS error	0.3	2.4	40.4	0.2	0.6	19.4
Total	36.0	158.4	7921.5	69.4	293.3	13704.6

c1 = minSup=1, minPopularity=1
c2 = minSup=0.5, minPopularity=1
c3 = minSup=0, minPopularity=0

We later use some of these indications to create features that distinguish maliciousness. Table 4 shows the average number of nodes included in the final CGs for benign and malicious seeds, and for each node type.

Three sets of parameters are considered. c1, a very selective type, sets filtering parameters to minSup=1 and minPopularity=1, resulting in the selection of objects that appear in all snapshots and for all hosts. c3, with minSup=0 and minPopularity=0, instead merges and fuses all patterns independently of their support and popularity. Finally c2, with minSup=0.5 and minPopularity=1, is the default parameter setting. It selects all events appearing in at least half of the snapshots generated by each host involved in the seed activity, and retains only those common to all hosts.

Results clearly show that MAGMA builds graphs with hundreds of nodes. Note that malicious Seed-CGs generally have fewer nodes, except for the nodes that represent DNS errors (last row of Table 4). For c2, the number of common object-paths found in benign CG is approximately four times larger than in malicious CG, suggesting that benign web pages are more complicated than malicious HTTP patterns. Note also that the number of elements in the graph grows very large for c3, where no item is discarded and several thousands of nodes are retained. This could undermine the supervised classifier accuracy (see Sec. 8), and it definitively hurts the amount of information offered to the security analyst (see Fig. 11(a) for instance). c2 offers a good trade-off between descriptiveness and richness of the final CG.

In summary, CGs are focused and descriptive characterization of common activities. Benign and malicious graphs look different, suggesting that a supervised classifier would be able to model them and distinguish between the two categories.

7. MAGMA Classifier and Features

We now design a supervised classifier and train it using the labeled dataset of graphs obtained considering malicious and benign seeds. We consider different decision tree classifiers: (i) the original J48 (an open source implementation of the C4.5 decision tree); (ii) Bagging coupled with J48; and (iii) Random Forest (RF). Decision trees are known for being scalable and offering a interpretative classification models [32]. In a decision tree, internal nodes represent tests on individual features, each branch is an outcome of the tests, and each leaf node represents a decision, *i.e.*, a class label. The paths from the root to a leaf represent classification rules. Bagging is a process that improves stability and accuracy by training m decision trees on m independent samples of the training set. The m models are combined by voting at the end. Random Forest is an extension of the bagging process such that, at each candidate branch in the learning process, a random

subset of the features are selected to avoid strong features from biasing the construction of trees.

Since MAGMA aims at the classification of malicious patterns in general and does not target specific class of malicious behaviors, we define an extensive set of features and extract them from Seed-CGs. Four different domains are covered: (i) *graph topology* (*e.g.*, the number of distinct nodes for each type, min/max/avg/std of in-degree and out-degree for each node type, graph giant connection ratio, etc.); (ii) *HTTP* (*e.g.*, the number of GET/POST events, min/max/avg/std of the length of user-agent string, etc.); (iii) URL *syntax* (*e.g.*, the number of hostname accessed directly using the IP address, or starting with `www`, etc.); and (iv) *occurences* (*i.e.*, min/max/avg/std of the number of events for each node type). The choice of features is partly driven by domain knowledge or has been previously used in the literature. Some features are instead generic, but could be useful in making the distinction. In total, 111 features are extracted from each CG, as detailed in App. Appendix A.

In the following, we consider the classifiers trained using (i) all; (ii) only HTTP; (iii) only Topology; and (iv) only Syntax features. We do this in order to compare against the previous works in which only HTTP or syntax has been used to target a specific malware. For comparison, we consider the subset of features suggested by the Minimum-redundancy-maximum-relevance (mRMR) feature selection algorithm [38] (the selected features are reported in bold in App. Appendix A). Note that mRMR selects features from all four of our domains.

7.1. Feature Characterization

Table 4 hints that CGs obtained from malicious and benign seeds contain a different amount of nodes and edges. Here, we briefly illustrate the extent of feature differences extracted from CGs. Fig. 12 compares the number of occurrences of three features in individual CGs: (i) number of distinct User-Agents; (ii) number of object-paths; and (iii) number of DNS failures. Two benign and two malicious seeds are highlighted for comparison.

Consider first the number of distinct User-Agents. The intuition is that malware could abuse the semantic associated to the User-Agent information and generate a large number of semi-random strings. This is what happens with `http://badi4net.no-ip.org/realtime.xmltmp` (red square), which is a malware that generates click-fraud and "impersonates" different browsers. Surprisingly, benign applications also abuse the User-Agent field, *e.g.*, `liveupdate.symanteclive update.com` (blu triangle) encodes the update versions in user agents, generating more than 200 different agents.

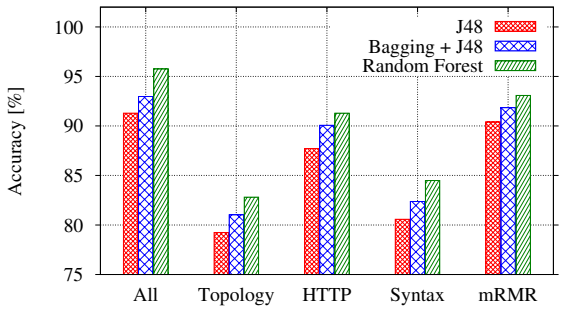


Figure 13: Comparison of MAGMA's accuracy for different classifiers and set of features.

Looking at the number of HTTP objects, Fig. 12(b) confirms the intuition that benign patterns include more objects. Yet, there are some malicious patterns that have a large number of objects, and a lot of benign CG have few HTTP objects. This is the case of the previously investigated seed in Fig. 12(a) (black diamond and green star). Finally, look at Fig. 12(c), which represent the number of DNS failures. Also in this case we expect a high number of failing DNS requests to be a characteristic of malicious activity. But there are a lot of benign CGs that exhibit a number of DNS failures very similar to benign patterns.

Two conclusions can be drawn from these examples. First, the 60 threats in the dataset that we use exhibit a wide range of patterns. Second, there are no easy means to separate malicious and benign CGs using some simple heuristic. A state-of-the-art classifier is needed to combine different planes of information and make the distinction.

8. Classification Results

To assess the performance of MAGMA, we follow best practices suggested by the machine learning community. We consider a labeled dataset, where ground truth labels are provided by the oracle, *i.e.*, the IDS, and then we train and test performance using this labeled dataset.

8.1. Cross-validation and Performance Metrics

We split the labeled dataset of N eligible seeds in two parts, one for training the classifier, and the

Table 5: Confusion matrix for Random Forest and all features.

	Predicted Class	
	Malicious	Benign
Malicious	218	18
Benign	21	643

other for testing its performance.

We employ several validation methodologies: (i) 66% split; (ii) k-fold cross-validation; and (iii) leave-one-out cross-validation methodologies. In the first methodology, we run a single experiment using 66% of our dataset for training and the remaining 33% for testing. k-fold cross validation generalizes this such that k equal size subsets are randomly generated. Then, k experiments are run, where $k-1$ subsets are used for training, and the remaining 1 is used for testing. The results are computed over all k runs. Finally, "leave-one-out" is an exhaustive cross-validation methodology in which, for each of the N elements in the labeled dataset, $N-1$ are used for training, and 1 is used for testing. The results are then computed over N independent experiments. Exhaustive cross-validation methods are preferred since they learn and test on all possible ways to divide the original sample into a training and a validation set. They are considered to be the most accurate means of testing a classifier, but they require a very large number of tests. For $N \approx 900$, we could afford the complexity of the leave-one-out in the following.

We measure the performance in terms of *accuracy*, *i.e.*, the fraction of valid results over the number of tests. We also report the confusion matrix which details the number of true positives and false negatives for each class, *i.e.*, for malicious (benign) seeds, it shows the number of events that were correctly classified as malicious (benign), and the number of events that were erroneously classified as benign (malicious).

8.2. Classifier and Feature Impact

We start by evaluating MAGMA with the default parameters setting, *i.e.*, $\Delta = 30$ min, MinSup=0.5, and MinPopularity=1. Fig. 13 reports the classification accuracy for the three classifiers, considering different sets of features (see Tab. A.6 in Appendix). Observe how Random Forest consistently provides the best results, with J48 providing the worst. Accuracy is higher when all features are used, with all

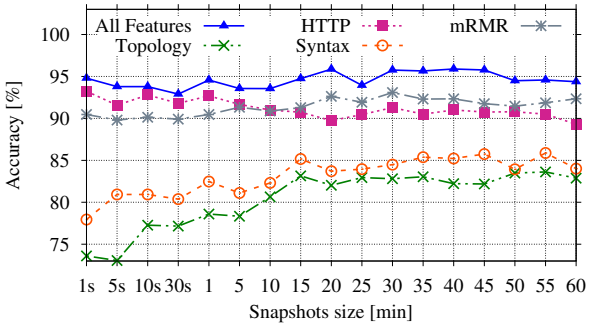


Figure 14: Classifier accuracy depending on the observation snapshot duration Δ with Random Forest classifier and different feature combinations.

subsets contributing to improve performance. The two observations confirm the richness of information offered by Seed-CGs, as well as the need to consider a wide range of generic features that do not focus on particular types of malware. We also confirm this reasoning via mRMR checks, where only few features are selected, but all of them are always from different domains.

When all features are offered to the Random Forest classifier, accuracy reaches more than 95%. This is an excellent result in general, which is confirmed by the confusion matrix reported in Table 5. The rows of the matrix specify the ground truth class, while the columns indicate the predicted class. Cells on the diagonal represent the number of true positives, while cells outside represent the false positives (or false negatives). The results confirm that the recall and precision in pattern classification is very high for both classes.

8.3. Parameter Sensitivity

We now focus on the impact of parameters settings. Fig. 14 shows the impact of the choice of Δ, which we vary from 1 s to 60 min. It considers only the Random Forest, with MinSup=0.5, and MinPopularity=1. The experiments are repeated for different feature combinations. Results show that Δ has a limited impact. The intuition is that typical activity related to an event lasts few seconds during which the application running at the host generates a burst of events. Only for very small values of Δ indeed it is possible to appreciate a generic decrease of accuracy due to a limited number of events that fall within the snapshots. Interestingly, we observe that HTTP features tend to be more significant for small values of Δ, while graph topology features

gains of importance for larger values of Δ. The drop of accuracy for a HTTP only based classifier is due to noise infiltrating into the benign graphs when large snapshots are considered. At the same time, the rich graphs are better characterized by topology features. Notice how the classifier trained considering all features is able to trade the drop of HTTP feature information with the increase of information offered by other features.

With $\Delta = 30$ min, we change the MinSup, and MinPopularity parameters. In Sec. 7 we already detailed how a different choice of parameters induces on the filtering and enrichment process, crf. Table 4. We now compare the impact on classification accuracy. Recall, that the number of snapshots and of hosts presenting a seed is rather limited. As such, we can only coarsely choose the parameters. Fig. 15 reports results. We observe that also in this case the impact of parameter settings is not crucial. However, by applying a too selective choice, *e.g.*, MinSup=1, and MinPopularity=1, or a too permissive filter, *e.g.*, MinSup=0, and MinPopularity=0, the accuracy tends to decrease. In the first case, too few events are left in the common pattern extraction. In the second case, too many events are instead accepted so that CGs appear to be noisy. Trading between minimum frequency in snapshot and minimum frequency among hosts provides a good trade-off.

Notice that the choice of MinSup, MinPopularity impacts also the number of events that appear in the Seed-CG, which is then offered to the security analyst in case he/she likes to investigate a decision returned by MAGMA. The more restrictive they are, the smaller the number of events. The choice of MinSup=0.5, and MinPopularity=1 results in a good balance between accuracy and richness of the graph, as depicted in Fig. 11.

8.4. Additional Experiment

We now aim at assessing the usage of MAGMA in the wild. To this extent, we run an experiment on a separate trace which includes only traffic from 15 hosts monitored for one day. The IDS does not flag any of the events appearing in the traffic, therefore these hosts should be considered as not infected by any malware. Overall, 43,000 distinct HTTP events are collected, of which 1868 are seen at least 3 times, and thus are valid seeds.

For each seed, we extract the CG, and run it through the MAGMA classifier that we previously trained. We let then MAGMA classify each of the

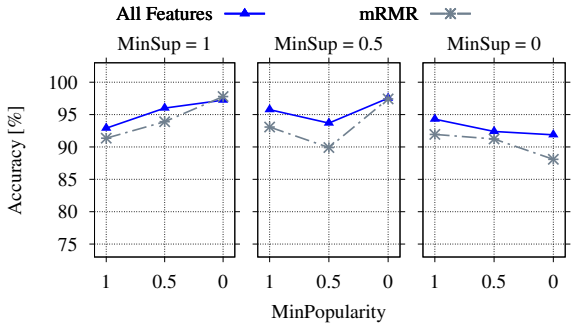


Figure 15: Sensitivity analysis on *MinSup* and *MinPopularity* with Random Forest classifier, all and mRMR features, $\Delta = 30$ min.

CGs. It returns 1852 benign tests, and only 16 malicious tests. By considering the oracle labels, the former have to be considered as true negatives (*i.e.*, benign events classified as not malicious). The latter instead have to be considered as false positives (*i.e.*, benign events misclassified as malicious). This corresponds to 99.14% of accuracy, with a mere 0.86% of false positive.

We further investigate the 16 misclassified cases using again Snort and VirusTotal. Snort detects 10 malicious events out of the 16 misclassified cases, while VirusTotal raises 4 warnings for events that were already flagged by Snort. Cross-checking VirusTotal threat descriptions and Snort detection-rules documentation, it appears that the detected events are related to *Simbar Spyware*, a redirecting toolbar affecting Internet Explorer, *Sgrunt Dialer*, a Trojan virus that limits the access to files and programs, and *AskSearch Toolbar* that is responsible for inflating clicks on advertisement to monetize traffic. In a nutshell, MAGMA identified 10 threats that the oracle ignored (but other tools have signatures for). This reduces the false positive to only 6 cases over 1868 tests (0.3%).

9. Conclusions

We presented MAGMA, a classifier for malicious network activity identification. It leverages simple events collected from the network vantage point, where both the spatial and temporal recurrences of events allow MAGMA to capture a detailed picture of the activity involved in a malicious or benign activity using Big Data approaches. MAGMA models this by means of Network Connectivity Graphs, in which multiple graphs model the common events

found by separately analyzing different protocols, and then fusing them in a single graph. A decision tree classifier is trained on a dataset where malicious and benign graphs are labeled by an oracle, which exposed a very heterogeneous set of malicious and benign activities. MAGMA thus results in a general purpose malware classifier, able to leverage common features that characterize several different families and variations of malware. We presented a performance evaluation using a real traffic trace obtained from a large ISP. MAGMA accuracy is over 95%, and its performance shows little sensitivity to parameter settings.

MAGMA model is based on the extraction of recurrent events from the traffic surrounding a given seed. We acknowledge that MAGMA applicability is limited to only those threat families that exhibit recurrent patterns over time and over multiple hosts. MAGMA is intended to facilitate the identification of previously unknown malware and to support the forensic activity of a security analyst. We have shown that the MAGMA Network Connectivity Graph provides a rich and interpretable characterization of the malicious activity.

[1] Kaspersky Lab, "Global corporate it security risks: 2013," http://media.kaspersky.com/en/business-security/Kaspersky_Global_IT_Security_Risks_Survey_report_Eng_final.pdf, 2013.

[2] Symantec, "2014 internet security threat report," http://www.symantec.com/security_response/publications/threatreport.jsp, 2014.

[3] iMPERVA, "Assessing the effectiveness of antivirus solutions," http://www.imperva.com/docs/HII_Assessing_the_Effectiveness_of_Antivirus_Solutions.pdf, 2012.

[4] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '06. ACM, 2006, pp. 41–52. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177086

[5] L. Zhang and Y. Guan, "Detecting click fraud in pay-per-click streams of online advertising networks," in *Distributed Computing Systems, 2008. ICDCS '08. The 28th International Conference on*, June 2008, pp. 77–84.

[6] N. Kshetri, "The economics of click fraud," *IEEE Security & Privacy*, vol. 8, no. 3, pp. 45–53, May 2010.

[7] C. Grier *et al.*, "Manufacturing compromise: The emergence of exploit-as-a-service," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 821–832. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382283

[8] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proc. of WWW*, 2010.

[9] E. Bocchi, L. Grimaudo, M. Mellia, E. Baralis, S. Saha, S. Miskovic, G. Modelo-Howard, and S.-J. Lee, "Network connectivity graph for malicious traffic dissection," in *Conference on Computer Communications and Networks (ICCCN), 2015 24th International*, Aug 2015.

[10] N. Jiang, J. Cao, Y. Jin, L. Li, and Z.-L. Zhang, "Identifying Suspicious Activities Through DNS Failure Graph Analysis," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. IEEE, 2010, pp. 144–153.

[11] Y. Nadji, M. Antonakakis, R. Perdisci, and W. Lee, "Connected colors: Unveiling the structure of criminal networks," in *Research in Attacks, Intrusions, and Defenses*. Springer, 2013, pp. 390–410.

[12] P. K. Manadhata, S. Yadav, P. Rao, and W. Horne, "Detecting malicious domains via graph inference," in *ESORICS 2014*. Springer, 2014, pp. 1–18.

[13] L. Liu, S. Saha, R. Torres, J. Xu, P.-N. Tan, A. Nucci, and M. Mellia, "Detecting Malicious Clients in ISP Networks Using HTTP Connectivity Graph and Flow Information," in *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*. IEEE, 2014, pp. 150–157.

[14] L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S.-J. Lee, C. Kruegel, and G. Vigna, "Nazca: Detecting Malware Distribution in Large-Scale Networks," in *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS '14)*, Feb 2014.

[15] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," in *Proc. of the 30th IEEE Int'l Conference on Computer Communications*, 2011, pp. 191–195.

[16] A. Oza, K. Ross, R. M. Low, and M. Stamp, "Http attack detection using n-gram analysis," *Elsevier Computers & Security*, vol. 45, pp. 242–254, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814000959

[17] M. Antonakakis, R. Perdisci, Y. Nadji, N. V. II, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," in *Proc. of USENIX Security Symposium*, 2012, pp. 491–506.

[18] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Conference on Security Symposium*, ser. SS'08, 2008, pp. 139–154.

[19] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: detecting malware infection through IDS-driven dialog correlation," in *Proc. of the 16th USENIX Security Symposium*, 2007, pp. 12:1–12:16.

[20] C. J. Dietrich, C. Rossow, and N. Pohlmann, "Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis," *Computer Networks*, vol. 57, no. 2, pp. 475–486, 2013.

[21] J. François, S. Wang, R. State, and T. Engel, "Bottrack: tracking botnets using netflow and pagerank," in *NETWORKING 2011*. Springer, 2011, pp. 1–14.

[22] H. Hang, X. Wei, M. Faloutsos, and T. Eliassi-Rad, "Entelecheia: Detecting P2P botnets in their waiting stage," in *IFIP Networking Conference, 2013*, 2013, pp. 1–9.

[23] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in *Proc. of the ACM SIGKDD*,

2009, pp. 1245–1254.
[24] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee, "Arrow: Generating signatures to detect drive-by downloads," in *Proc. of the 20th Int'l Conference on World Wide Web*, 2011, pp. 187–196.
[25] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proc. of the Network and Distributed System Security Symposium*, 2008.
[26] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive DNS traces," in *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, 2009, pp. 311–320.
[27] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1, 1999," *RFC 2616*, 2006.
[28] P. Mockapetris, "RFC 1034: Domain names: concepts and facilities (november 1987)," *Status: Standard*, 2003.
[29] A. Finamore, S. Saha, G. Modelo-Howard, S.-J. Lee, E. Bocchi, L. Grimaudo, M. Mellia, and E. Baralis, "Macroscopic view of malware in home networks," in *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, Jan 2015, pp. 262–266.
[30] P. Porras, "Inside risks: Reflections on conficker," *Communications of ACM*, vol. 52, no. 10, Oct. 2009.
[31] L. Seltzer, "Conficker: Still spamming after all these years," http://www.zdnet.com/conficker-still-spamming-after-all-these-years-7000031206/, 2014.
[32] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Addison-Wesley, 2013.
[33] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharma, "Discovering all most specific sentences," *ACM Transactions on Database Systems (TODS)*, vol. 28, no. 2, pp. 140–174, 2003.
[34] F. Pan, G. Cong, A. K. Tung, J. Yang, and M. J. Zaki, "Carpenter: Finding closed patterns in long biological datasets," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 637–642.
[35] D. Apiletti, E. Baralis, T. Cerquitelli, S. Chiusano, and L. Grimaudo, "Searum: A cloud-based service for association rule mining," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, July 2013, pp. 1283–1290.
[36] "VirusTotal," https://www.virustotal.com/.
[37] "Snort," https://www.snort.org/.
[38] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.

Appendix A. Classification Features

Table A.6 list all features extracted from Seed-CGs divided by categories. Most of them are self explanatory. The *CDN hostname ratio* takes into account the presence of CDN in nowadays webpages. We assume that if a hostname is linked to more than 3 distinct IP addresses, the content it refers to is hosted on a CDN. We define such hostname a *CDN hostname*.

Table A.6: Full list of features extracted from Seed-CGs. Underlined features are selected by mRMR.

Topology	
num. of	total number of nodes
num. of	total number of edges
num. of	failed DNS queries events
num. of	object-path nodes
num. of	hostname nodes
num. of	serverIP nodes
num. of	UDP-ports nodes
num. of	TCP-ports nodes
num. of	DNS error types
num. of	nodes with single edge
min/max/avg/std	in-degree object-path nodes
min/max/avg/std	in-degree hostname nodes
min/max/avg/std	in-degree serverIP nodes
min/max/**avg**/std	out-degree object-path nodes
min/max/avg/std	out-degree hostname nodes
min/max/avg/std	out-degree serverIP nodes
ratio	giant connection ratio
ratio	ratio num. of CDN hostnames† over total hostnames (*CDN hostname ratio*)

HTTP	
num. of	requests per each method [GET, POST, others]
num. of	replies per each response status [20x, 30x, **40x**, 50x]
num. of	replies per each content-type [text, image, application, binary, multipart, multimedia]
num. of	distinct user-agent strings
min/max/avg/std	user-agent strings length
min/max/**avg**/std	requests per distinct user-agent string
min/max/avg/std	user-agent strings blank chars

Syntax	
num. of	hostname nodes being IP addresses
num. of	hostname starting with `www`
min/max/avg/std	hostname string length
min/max/**avg**/std	hostname digits and alphabetic chars ratio
min/max/avg/std	for hostname up to 2nd LD, num. of distinct 3rd LD

Occurrences	
min/**max**/avg/std	object-path nodes events
min/max/avg/std	hostname nodes events
min/**max**/avg/std	DNS fail nodes events
min/**max**/avg/std	DNS succeed nodes events
min/max/avg/std	TCP port nodes events
min/max/avg/std	UDP port nodes events