Unsupervised Detection of APT C&C Channels using Web Request Graphs

Pavlos Lamprakis[1], Ruggiero Dargenio[1], David Gugelmann[1], Vincent Lenders[2], Markus Happe[1], and Laurent Vanbever[1]

[1] ETH Zurich, Zurich, Switzerland
[2] Armasuisse, Thun, Switzerland

Abstract. HTTP is the main protocol used by attackers to establish a command and control (C&C) channel to infected hosts in a network. Identifying such C&C channels in network traffic is however a challenge because of the large volume and complex structure of benign HTTP requests emerging from regular user browsing activities. A common approach to C&C channel detection has been to use supervised learning techniques which are trained on old malware samples. However, these techniques require large training datasets which are generally not available in the case of advanced persistent threats (APT); APT malware are often custom-built and used against selected targets only, making it difficult to collect malware artifacts for supervised machine learning and thus rendering supervised approaches ineffective at detecting APT traffic.

In this paper, we present a novel and highly effective unsupervised approach to detect C&C channels in Web traffic. Our key observation is that APT malware typically follow a specific communication pattern that is different from regular Web browsing. Therefore, by reconstructing the dependencies between Web requests, that is the Web request graphs, and filtering away the nodes pertaining to regular Web browsing, we can identify malware requests without training a malware model.

We evaluated our approach on real Web traces and show that it can detect the C&C requests of nine APTs with a true positive rate of 99.5-100% and a true negative rate of 99.5-99.7%. These APTs had been used against several hundred organizations for years without being detected.

Keywords: Malware detection, Web request graph, command and control channel, click detection, graph analysis, advanced persistent threat

1 Introduction

An increasing number of high-profile cyber attacks against companies and governments were reported in the last years. In contrast to untargeted attacks that aim at infecting as many hosts in the Internet as possible, these so called Advanced Persistent Threats (APTs) target a certain organization over long periods of time, focus on a specific objective and are conducted by adversaries with significant resources in a stealthy way [11,28]. Because these APT campaigns are

supposed to run for a long time, the malware used is often tailored-made and attackers take great care in hiding its traces. This makes it difficult to obtain APT malware samples for analysis — in contrast to general purpose malware that can, due to their widespread presence[3], easily be collected and analyzed. As a result, traditional signature-based threat protection solutions and supervised learning techniques struggle to identify APT malware. As an example, the Swiss defense contractor RUAG had been compromised for at least one year until an external organization provided information that lead to the detection of the HTTP C&C channel [1].

Once in place, APT malware typically rely on HTTP-based Command & Control (C&C) channels [1,12,13,14,21,24,35,39]. Using HTTP provides the attacker with two main advantages. First, this C&C channel is widely available as most organizations allow their employees to browse the Web. Second, normal Web browsing generates a huge amount of requests destined to a large number of servers. This makes it very difficult to tell apart benign HTTP requests caused by employees' browsing from malicious activity, allowing attackers to hide their communication in plain sight.

Detecting and blocking C&C channels under these constraints is challenging. Indeed, the large number of Web servers contacted daily makes it impractical to operate with a default-block policy and a whitelist for Web browsing. Therefore, most organizations use a default-accept policy in combination with a blacklist to detect C&C channels in the Web traffic of internal clients. The employed blacklists typically combine the Indicators Of Compromise (IOC) from different commercial and freely available intelligence feeds, such as abuse.ch, cymon.io, autoshun.org, and www.openbl.org. Unfortunately, since the target scope of APT malware is very narrow, traces of APT samples are often only detected by accident and it can take years until corresponding malware samples are recovered and IOC are added to intelligence feeds. Furthermore, the fact that there are only few APT malware samples available makes it difficult to apply supervised learning techniques for the detection of APT campaigns.

In this paper, we propose an unsupervised detection approach that does not need any malware samples for training. Our one-class classifier only requires labeled benign traces for training. Our approach is built around the observation that C&C channels typically follow a specific communication pattern that is unrelated to regular Web browsing. Therefore, after analyzing and reconstructing all the artifacts caused by human Web browsing, i.e., creating the Web request graph of the recorded Web traffic [25,42], the malicious requests to C&C servers stand out because they do not have any dependency or interaction with other Web requests; they are so-called unrelated nodes in the Web request graph. The key challenge behind this approach is that simply relying on the HTTP referrer for Web request dependency reconstruction, such as done by [17,42], results in many unrelated benign nodes. For this reason, we studied the Web traffic caused by benign browsing in detail and introduce several new heuristics

[3] 430 million malware samples have been released in 2015 according to Symantec's Internet security threat report [37]

to reconstruct missing links in the request graph. For instance, if the requested URL of an unrelated node can be found in the HTML source code of a recently accessed Web page, we can connect both requests. In combination with a small whitelist of benign services causing unrelated requests, such as OCSP servers and software update services, this approach allows us to identify C&C requests with high accuracy — after running the link completion process and applying the whitelist, all remaining requests are considered as suspicious.

This paper provides the following key contributions:

- *Link completion heuristics* that extend and complete the request graph generated in our previous work Hviz [17] by linking unrelated Web requests to their most likely parent. Link completion reduces the number of unrelated nodes in benign Web traffic by a factor of 8-30.
- *A malware detection approach* that marks non-whitelisted, unrelated Web requests in (completed) request graphs as malicious. Our whitelist only contains certificate authority domains and the update server of the operating system.
- *A comprehensive evaluation* in which we evaluate the performance of our approach by randomly inserting C&C traces covering the activities of trojan horses, exploit kits, botnets, ransomware and APTs into benign Web traffic traces[4]. We detect 99.5% of all malicious C&C requests (true positive rate) while falsely labeling 0.3-0.5% of the benign requests as malicious (false positive rate).

The rest of this paper is structured as follows. Section 2 presents our malware detection approach that applies a click detection classifier and link completion heuristics to connect unrelated, benign Web request to their most likely parent. Section 3 evaluates our approach and Section 4 discusses our results. Section 5 compares our approach to related work and Section 6 concludes the paper.

2 Approach

Our three-step malware detection approach is shown in Figure 1. It detects C&C channels of APT malware (used in targeted attacks) and 'general purpose' malware (used in untargeted attacks). In a first step, we extract the (incomplete) request graph from Web traffic logs. In a second step, we complete the request graph by (i) click detection (see Section 2.3) and (ii) link completion (see Section 2.4). In a third step, we filter the remaining unrelated requests. The remaining unrelated requests are considered as suspicious unless the contacted server is whitelisted. We use Bro IDS [31] and Hviz [17] to create the request graph.

In the following we first give an overview on request graphs in Section 2.1, which are the base for our detection approach. Section 2.2 describes the idea behind our approach in more detail and the applied click detection and link completion are discussed in Section 2.3 and Section 2.4, respectively.

[4] We use benign Web traffic generated by scripts accessing the top 250 Web sites for Switzerland and user traffic logs from ClickMiner [25].

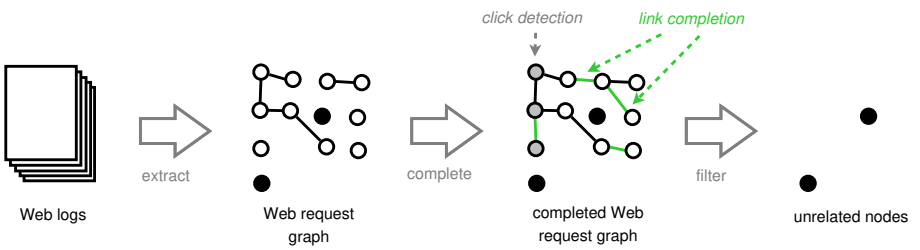

Fig. 1. Our malware detection approach: First, we extract the request graph from Web traffic logs. Second, we complete the request graph by (i) click detection and (ii) link completion. Third, we filter the remaining unrelated requests that are not whitelisted.

2.1 Background on Web request graphs

Web traffic logs store HTTP requests and corresponding responses. The requests can be connected to a request graph. In a request graph, a node corresponds to an HTTP request and its response. Two nodes i and j can be connected using a directed edge (i, j) if the request j has been issued by the response of i. For most HTTP requests, these links can be derived from the referrer field in request j, which points to i. If there is a directed edge (i, j) from i to j, then i is the parent of j and j is the child of i. If there are two edges (i, j_1) and (i, j_2) then j_1 and j_2 are siblings. Unfortunately, the referrer is not always set. Therefore, request graphs are often incomplete if they are constructed solely based on the referrer information.

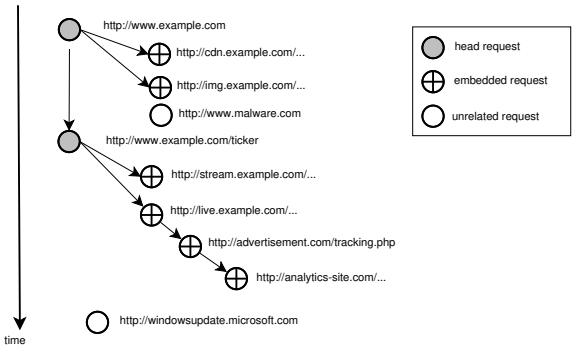

Fig. 2. Example request graph

We distinguish between three types of requests: 'head', 'embedded' and 'unrelated' requests, as can be seen in Figure 2. Head requests are requests that have been issued by the user directly, for example by typing an URL into the browser, clicking on a link, a browser bookmark or submitting a Web form. Embedded requests are generated as a result of head requests. For instance, accessing a Web

page triggers embedded requests to content delivery networks and analytics services. Unrelated nodes have no dependency to previous requests. They do not have any parent or children. Figure 2 shows an example request graph where the user has directly accessed two URLs, marked as head requests. Both head requests trigger further requests, marked as embedded requests. Figure 2 also contains two unrelated requests.

2.2 Malware detection in Web request graphs

Our detector is based on the idea that any HTTP request must have one of the following root causes:

1. *Triggered by users' Web browsing:* The request is directly or indirectly triggered by a user's Web browsing. These requests are part of a larger graph component that represents Web browsing.
2. *Triggered by benign software applications:* Many benign software applications running on end hosts issue HTTP requests, for example to check for updates or load information. These requests are unrelated to a user's Web browsing and thus classified as "unrelated". End hosts in larger organizations typically run a pre-build image that contains a limited number of benign software applications. Thus the Web services that are contacted by valid application software can easily be whitelisted.
3. *Triggered by malicious software:* Any request not being part of one of the previous categories falls into this category.

The assumption behind this scheme is that regular Web browsing results in perfectly connected request graphs. However, as we will show in Section 3.2 between 2.6 % and 9 % of the links in the request graph are typically missing. Therefore we introduce an heuristical approach that adds the missing links in the request graph. After applying the graph completion heuristic, our detector considers all remaining unrelated nodes as either being triggered by a benign software that accesses a server, such as the Windows update server, or by a malware accessing a command and control server.

2.3 Click detection

The goal of click detection [42,25,17,40] is to distinguish between *user clicks* and *other requests* (embedded and unrelated requests). We use the features shown in Table 1 as input for the machine learning. We use labeled data for training, but we only train on benign traces. Hence, we do not require any labeled malware trace. After evaluating different machine learning classifiers using Python scikit-learn [32], we found that a random forest classifier performs best, which is inline with the work of Vassio et al. [40]. The detailed results are shown in our evaluation in Section 3.1.

2.4 Link completion

The goal of our link completion algorithm is to add missing edges to the request graph. Referrer-based request graphs of benign Web browsing contain many un-

Table 1. Feature set for click detection

#	feature	description
F1	content type	content type such as text/html or image/jpeg
F2	response length	number of bytes of the HTTP response body
F3	number of referrals	number of children in request graph
F4	time gap	time gap between current and parent request
F5	URL length	number of characters of the URL
F6	advertisement	Is the request an advertisement (in EasyList)?
F7	presence of parent	Does the node have a parent node (referrer)?

related nodes. In the following we discuss the primary reasons we have observed in our traces:

- *Certificate status checks*: The Online Certificate Status Protocol (OCSP) [18] is an Internet protocol which is used as an alternative to certificate revocation lists (CRLs). It allows applications to determine the validity of a digital certificate. An OCSP client (e.g., the browser) issues a status request to the Certificate Authority (CA). The browser suspends the acceptance of the certificate until it receives a response from the CA. Those requests/responses do not have a referrer header set and do not cause any embedded requests. Thus, the nodes corresponding to them are unrelated. The same is happening with the usual transfer of certificates and CRLs. These requests can be identified by their content type which is *application/ocsp-response*, *application/pkix-cert* and *application/pkix-crl*. However, note that we can not simply whitelist all requests with these content types, as this would make it very easy for attackers to hide their HTTP requests by including a corresponding (fake) header.
- *Favicons*: We observed that whenever Firefox sends an HTTP request to retrieve the favicon of a website, it does not include the referrer field in the HTTP request headers. As it turned out, this happens due to the *link rel='icon'* tag found in the HTML source code of web pages. There is a known bug associated with the above behavior [9] which has been resolved but not fixed yet. The same bug is not present in Google Chrome.
- *Privacy*: There are cases where the referrer header can affect the user's privacy. For instance, a URL might contain personal information in its query strings in case of a GET request. For example, this was the case with Facebook in 2010 [20]. More specifically, advertisers could identify users who clicked on their advertisement since their user ID was contained in the referrer header. Thus, security-aware developers remove this information from the referrer by specifying referrer policies [41], which were recently developed by the World Wide Web Consortium (W3C). These referrer policies allow developers to limit the referrer to only the visited domain of the origin website or to even remove referrers completely. Another case which results in a missing referrer is the transition from an object loaded via HTTPS to

an HTTP object (downgrade). The main reason for this behavior is to avoid leaking sensitive information in the plain-text HTTP request.
- *Cross-Origin Resource Sharing (CORS)* [3]: When browsers make cross-domain HTTP requests, the referrer can be missing while the origin header is set. For example, this can happen when an OPTIONS preflight request is being sent, in order "to determine the options and/or requirements associated with a resource before performing the actual HTTP request" [4]. Firefox does not set the referrer header when performing this kind of requests, in contrast to Chrome. As a result, nodes that relate to this HTTP method become unrelated in Firefox.
- *Invalid Referrer*: The referrer header can have an invalid value which means that it does not correspond to a request URI of any previous node in the graph. A possible reason for this behavior could be bugs in the software.
- *Redirect Implementation*: There are several different ways for a user to be redirected from a source to a destination website. Firstly, the recommended way is to provide a 302 HTTP status code combined with the Location value in the HTTP response headers. Another way is to send a regular 200 HTTP status code and set the Refresh header or an HTML meta tag. In addition, a user can be redirected using Javascript. Depending on the implementation of the redirection, there are different behaviors of browsers to either keep or suppress the referrer [20].

The link completion algorithm completes the request graph to reduce the number of unrelated nodes in benign traffic. Our algorithm, which is depicted in Figure 3, takes as main input an unrelated node n and a request graph G. The output is the most likely parent in the graph or False if the node does not fit into the graph sequence.

Firstly, the algorithm uses a whitelist in order to filter requests from benign software that can be running on the host (step 1). The whitelist consists of 37 entries and includes OS update domains as well as Certificate Authorities. The latter domains can be reduced since companies usually set up their own OCSP responder which acts as an OCSP proxy server. Further, domains and IP addresses contacted by deployed software can be added to this list. We argue that the overhead for maintaining a corresponding whitelist is small, primarily because of two reasons: *(I)* Even if an organization does not use an OCSP proxy server, the number of contacted OCSP servers is limited as certificate issuers typically only operate few OCSP servers and there are publicly available lists of these servers. *(II)* Security-aware organizations should already be aware of the software deployed in their network and the corresponding external servers contacted by the software, which allows them to add the corresponding domains and IP addresses either proactively or reactively to the whitelist. In fact, our approach can be helpful to identify software that has been installed without authorization because most software includes an update process that operates over HTTP(S). The corresponding requests will most likely be unrelated such that our system will trigger an alert when the software contacts its update server.

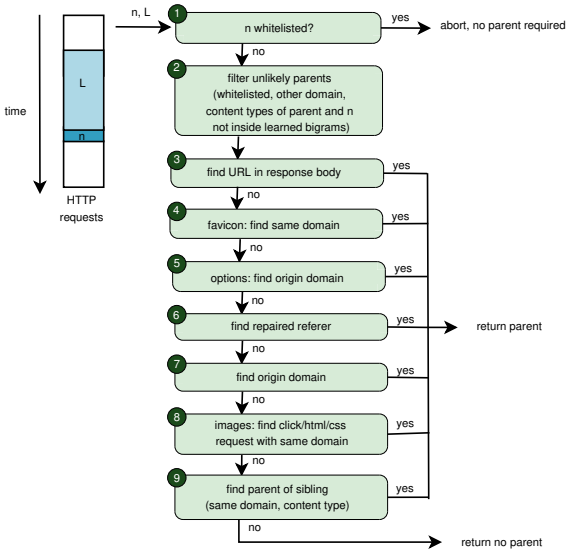

Fig. 3. The link completion algorithm tries to find the most likely parent of a node n inside a request list L. L is sorted by time and only contains the previous requests of a given time window. Steps 3-5 are processed twice, first for the click requests and second for the embedded requests in L.

If the node is not whitelisted, its possible parent is predicted as follows (step 2): Based on the fact that the HTTP requests of an unrelated node's possible parents were performed before it, we create a list with all the candidate nodes falling into a time window covering few seconds before the analyzed request. The time window's length is not fixed and can be provided as input to the algorithm. Before adding a possible parent node in the list, the algorithm confirms that it is a candidate depending on its content type. There are certain content types which have much more embedded objects (and therefore cause child requests) than others. For instance, an HTML document is more likely to perform more requests to load additional content (e.g., third-party content) than a Javascript file. In contrast, a node representing a request to a PNG image should not have any children since it is not rational for this type of content to make additional requests. The algorithm encodes the knowledge on likely and unlikely parent-child relations as bigrams of content types. For instance, a node whose content type is *text/html* will usually have children with content types *image/jpeg*, *text/css*, *application/javascript*, etc., whereas a *text/css* object is more likely to have children with *image/png*, *image/gif*, *application/font-woff* etc. content types. The bigrams are constructed by traversing all the graphs of the network traces in the training set and counting the top length-two sequences of the content types with most children.

For each candidate parent node in the list, its response body is examined (step 3). The idea behind this step is that the absolute or relative URLs of child

objects are often contained in the parent's response body. For example, the URL of a displayed image is typically contained in an *src* attribute in the webpage embedding the image and if the user clicked on a link, then the corresponding URL often previously appeared as *href* attribute. While this method is quite accurate, it requires complete response bodies to be stored, which can be large – especially if users consume videos. Therefore, we only apply this approach to response bodies for content types that have been found to have the most children, such as *text/html* responses.

Favicons can partially be linked using the above methods, but there is also a more accurate way (step 4): If the unrelated node's request URI is `www.example.com/favicon.ico` then, the parent's should be `www.example.com`. By default the favicon is placed in the root directory of the web page and browsers know where to find it. However, it is a common practice that developers place their favicons in other directories. In that case, the algorithm finds the parent based on the domain name and the content type of the possible parent nodes. Further, if a request's HTTP method is *OPTIONS* and the origin header value is set, then the parent is identified based on this value (step 5).

The steps 3-5 are run twice. In a first run, only nodes in the time window L that the click detection identified as head nodes are considered. If no parent has been found in the first run, then a second run is started which considers all nodes in the time window L. This way, identified clicks have a higher priority.

In order to handle requests with invalid referrers, the algorithm extracts the domain of the invalid referrer and searches for the parent that is closest in the time domain (step 6). The algorithm connects requests according to the origin header field, if the header field is available (step 7). If a request's content type relates to an image, then the time windows L is traversed and the first parented node that is either a head node, a node with *text/html* or *text/css* content is returned (step 8).

Finally, the algorithm matches nodes of the same content type and domain to the same parent (step 9). In other words, the algorithm tries to find the closest sibling s, which has the same content type and domain as the analyzed node n, and connects n to the parent of its closest sibling s.

3 Evaluation

For our evaluation we have merged benign Web browsing traces with malicious C&C requests. We have two types of benign Web traces, script-generated traces and user traffic collected by Neasbitt et al. [25]. We collected C&C requests from general purpose malware and APT malware samples from Weblogs [2,5,6,7]. We only use the post infection traffic of that general purpose/APT malware samples.

Table 2 shows our benign datasets. We have generated datasets S_1 and S_2 with a python script that emulates the Web browsing behavior of users by accessing the Alexa top 250 websites of Switzerland. Our script is based on the Selenium WebDriver [36]. It visits each of the top 250 websites in random order. We have removed websites with adult content from that list. The script makes

five clicks per average on each website and stays on each resulting page for a random time interval. The time spent on each page has an upper bound of 30 seconds. We record only unencrypted HTTP traffic. This is achieved by visiting the HTTP versions of the websites included in the input list. In case the website is forcing SSL by redirecting the client to its secure version, the connection is terminated and the next URL in the list is fetched.

Table 2. Benign Web traffic: S_1 and S_2 have been generated by a script and C_1 has been taken from the ClickMiner dataset [25]

id	data source	browser	#traces	# train requests	# test requests
S_1	script	Firefox 46.0.1	10	132k	278k
S_2	script	Chrome 54.0.2840.71	10	112k	257k
C_1	ClickMiner	Firefox 14.0.1	24	-	74k

We recorded 10 browsing traces using Mozilla Firefox as a browser and 10 browsing traces using Google Chrome. The user clicks have been recorded in order to train and evaluate the click detection classifier. Three out of the ten traces are used for training the click detection classifier. The other seven traces are used for testing in click detection, link completion and malware detection.

For evaluation we additionally used a third benign dataset C_1 that contains traffic from real users. The dataset has been published together with the ClickMiner paper [25] and contains 24 traces. These traces were accumulated from a user study with 21 participants. Each participant was requested to browse any website they wished for twenty minutes while preserving their privacy.

Table 3 summarizes the general purpose malware samples that have been collected from Contagiodump [2], Malware-traffic-analysis [6], the malware capture facility project [5] and pcapanalysis.com [7]. We labeled the C&C requests of these 49 malicious traces manually. We used a variety of general purpose malware that can be categorized in five malware families: botnets, exploit kits, trojan horses, sality and ransomware.

Table 3. C&C requests from published general purpose malware traces [2,5,6,7].

id	malware type	#traces	#C&C requests
M_1	Botnet	6	478
M_2	Exploit Kit	13	357
M_3	Trojan	25	274
M_4	Sality	3	155
M_5	Ransomware	2	3

Table 4 lists our APT malware samples. Again, we labeled the C&C requests manually. Section 3.4 explains the APTs in more detail. Unfortunately, some of the APT malware traces only consist of few HTTP samples. We decided to include these traces in our evaluation in order to investigate whether our approach mistakenly connects these requests to benign traffic or not.

Table 4. C&C requests collected from published APT malware traces [2].

id	APT type	#traces	#C&C requests	APT report
A_1	TrojanCookies	1	720	[24]
A_2	Lagulon	1	561	[12]
A_3	Taidoor	1	35	[39]
A_4	Netraveler	1	11	[21]
A_5	Tapaoux	1	8	[23]
A_6	Sanny	1	6	[14]
A_7	Taleret	1	1	[13]
A_8	Likseput	1	1	[24]
A_9	Darkcomet	1	1	[35]

3.1 Click detection

The information gain of each feature F1-F7 for click detection is depicted in Figure 4 for the datasets S_1 and S_2 separately. It can be seen that the results are similar for both tested browser types. The content type (F1) has the highest information gain. Most user clicks are performed on a text/html content type, while embedded requests often contain images, scripts, style sheets but also text/html. The response length contains more bytes for user clicks than for embedded requests (F2). The number of referrals (F3) is higher for user clicks since the accessed websites often trigger many embedded requests that refer to the clicked website. The time gap (F4) between a request and its parent is usually longer for user clicks since they are manually triggered as compared to embedded requests that are automatically generated. The URL length (F5) of user clicks are longer than the one of embedded requests. Users do not often directly access servers that are listed on EasyList as advertisement sites (F6). The presence of a parent feature (F7) provides only limited information gain.

We have evaluated click detection for the datasets S_1 and S_2 separately. A separate investigation shows us, how well the approach works for different browsers. We have tested Mozilla Firefox in S_1 and Google Chrome in S_2, which are two of the most popular Web browsers. For both datasets, we used the same testing methods. We randomly selected three out of ten traces for training and used the remaining seven traces for testing. The actual user clicks have been recorded while capturing the network traffic. We assume that every access to a Web server that is listed in our recorded click database is a user click. We have

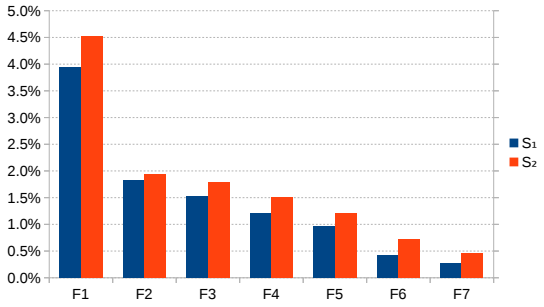

Fig. 4. Information gain per feature.

evaluated various machine learning approaches and found that a random forest classifier with 1000 estimators shows the best performance. A decision tree was the runner up.

The results of the random forest classifier are listed in Table 5. The trained random forest classifier has a recall of 0.96 for both browsers. The precision is higher for Google Chrome with 0.96 as compared to Mozilla Firefox with 0.94. The resulting f1 score is therefore slightly better for Google Chrome. Our click detection performance is comparable to the one published by Vassio et al. [40]. We refer to Vassio et al. [40] for a more detailed analysis of click detection based on a random forest classifier.

Table 5. Click detection classification results

dataset	**recall**	**precision**	**f1 score**
S_1	0.96	0.94	0.95
S_2	0.96	0.96	0.96

In the following, all presented results are based on Web request graphs that have been updated by the click detection classifier, such that only requests that have been classified as user clicks are marked as head nodes. The remaining results are only based on the test datasets of S_1, S_2 and C_1. We have also applied click detection to the Web request graphs of the ClickMiner dataset. We have used the trained classifier of S_1, since ClickMiner has used (a previous version of) Mozilla Firefox.

We have merged the malicious datasets M_{1-5} randomly into the benign test datasets S_1, S_2 and C_1 to evaluate our approach for general-purpose malware. The merged datasets are labeled as $\{S_1, M\}$, $\{S_2, M\}$ and $\{C_1, M\}$. Similarly, we have merged the malicious traces A_{1-9} randomly into the benign test datasets S_1, S_2 and C_1 to evaluate our approach for APT malware. The merged datasets are named $\{S_1, A\}$, $\{S_2, A\}$ and $\{C_1, A\}$.

3.2 Link completion

Table 6 gives an overview of our results for link completion and malware detection for all merged datasets. It can be seen that the malicious requests are mostly unrelated. Only six requests from general-purpose malware are related. When we only rely on the referrer, we see that the vast majority of the unrelated requests is benign (72-95%). After applying click detection and link completion, only 23-39% of the remaining unrelated requests are benign.

Table 6. Statistics on the number of related/unrelated and benign/malicious requests for all merged datasets. The majority of the requests are benign for all datasets. Our approach significantly reduces the number of unrelated benign requests without reducing the number of unrelated malicious requests.

datasets	$\{S_1, M\}$	$\{S_2, M\}$	$\{C_1, M\}$	$\{S_1, A\}$	$\{S_2, A\}$	$\{C_1, A\}$
# benign	278 367	257 123	74 037	278 367	257 123	74 037
# malicious	1 267	1 267	1 267	1 344	1 344	1 344
referrer-based approach						
# related	253 239	250 490	70 851	253 233	250 484	70 845
# unrelated	26 395	7 900	4 453	26 478	7 983	4 536
# related malicious	6	6	6	0	0	0
# unrelated malicious	1 261	1 261	1 261	1 344	1 344	1 344
# related benign	253 233	250 484	70 845	253 233	250 484	70 845
# unrelated benign	25 134	6 639	3 192	25 134	6 639	3 192
our approach: click detection and link completion						
# related	277 551	256 411	73 650	277 545	256 405	73 644
# unrelated	2 083	1 979	1 654	2 166	2 061	1 737
# related malicious	6	6	6	0	0	0
# unrelated malicious	1 261	1 261	1 261	1 344	1 344	1 344
# related benign	277 545	256 405	73 644	277 545	256 405	73 644
# unrelated benign	822	718	393	822	718	393

We evaluated our link completion algorithm with the test datasets S_1, S_2 and C_1. The results are depicted in Figure 5. We can see that the S_2 dataset has less unrelated nodes as compared to dataset S_1 before applying link completion. Chrome produces a more complete request graph as compared to Firefox since it sets the referrer header more often than Firefox as explained in Section 2.4. Our link completion algorithm decreased the number of unrelated nodes by an average factor of 30 for S_1 dataset, nine for S_2 and eight for ClickMiner. After applying link completion, each test dataset contains between 0.28-0.53% unrelated nodes.

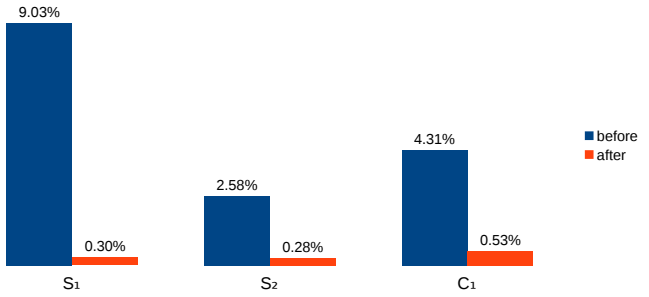

Fig. 5. Share of unrelated nodes before and after link completion. Link completion decreases the number of unrelated nodes by an average factor of 30 for S_1, nine for S_2 and eight for C_1.

3.3 General purpose malware detection

We evaluated the ability of the algorithm to perform malware detection over the above metrics using the merged datasets in the following way. For each data source we randomly merged its traces with malicious ones by injecting the whole malicious graphs inside the benign graph at random timestamps. This would simulate a real case scenario where a user browses the Web and at the same time a general purpose malware is running in the background (e.g., exfiltrating data to the C&C server). We run the algorithm on each merged trace for each data source and the results can be seen in Table 7. 99.5% of the C&C requests were successfully detected while 0.5% were missed, independently of the benign dataset used. The true negative and false positive rates are the same as shown in Figure 5. Hence, we can see that the link completion algorithm does not falsely connect benign and malicious nodes. The false positive rate is a bit different for each dataset and relates to the benign nodes (requests/responses) that the algorithm was not able to connect in the graph.

Table 7. Malware detection results.

data set	TPR	FPR	TNR	FNR
$\{S_1, M\}$	0.995	0.005	0.995	0.005
$\{S_2, M\}$	0.995	0.003	0.997	0.005
$\{C_1, M\}$	0.995	0.003	0.997	0.005

3.4 Advanced persistent threat malware detection

Advanced persistent threats employ targeted malware. They are hard to detect, because they only attack selected high-profile targets, such as governments, military, diplomats and research institutes. The attackers use advanced methods to

infect the target's computers, because their targets are often better protected against malware than the average user. APTs can operate for years without being noticed by the victims. When an APT has successfully infected a high-profile target, it is often reused to attack other high-profile targets.

Nettraveler is an APT that has been in operation since at least 2005. It automatically extracts large amounts of private data over long time periods. The APT malware compresses the private data and sends it to C&C servers in HTTP requests. Kaspersky Labs [21] revealed this cyber espionage campaign in 2013. More than 350 high-profile targets have been attacked in 40 countries during this campaign. When Kaspersky revealed the campaign, 22 GB of stolen data was still on the C&C servers. However, it is likely that stolen data had been removed from the servers during the campaign. Therefore the total amount of stolen data cannot be estimated.

The attackers send spear phishing e-mails to selected users. The Nettraveler APT malware is hidden inside a Microsoft Office document. The APT malware takes advantage of one of two vulnerabilities in Microsoft Office that can lead to remote code execution. Both vulnerabilities have been patched in the mean time, the vulnerability CVE-2010-3333 in 2010 and the vulnerability CVE-2012-0158 in 2012. Interestingly, this APT has been recently used to attack high-profile targets in Russia, Mongolia, Belarus and other European countries in 2016 [34]. This indicates that even high-profile targets do not continuously and consistently apply critical software updates on their computers. Hence, attackers can still find a machine that can be attacked in order to get access to the corporate network.

We have also investigated malicious samples of the following APTs.

- *Likseput* (trace A_8) is an APT malware which was used by a government-sponsored Chinese APT group, called APT1, in order to control compromised systems in cyber espionage campaigns that took place since at least 2006. APT1 has already extracted hundreds of terabytes from at least 141 organizations according to the Mandiant report [24].
- *TrojanCookies* (trace A_1) is another APT malware used by APT1. It communicates with the C&C server by encoding the commands as well as the responses in the cookie using base64 and a single-byte xor obfuscation.
- *Lagulon* (trace A_2) was used in several targeted campaigns performed by an Iranian group, named Cleaver, in 2013. The APT malware can log the user's keystrokes, download and execute code, take screenshots and periodically exfiltrate data to a remote HTTP-based C&C server. The attackers gained highly sensitive information from government agencies and infrastructure companies in many countries [12].
- *Sanny* (trace A_6) was used in targeted attacks primarily against major industries in Russia. It was detected in 2012. The attackers sent a malicious Microsoft Word document via spear phishing emails. The APT malware profiles the victims regarding their region and language. It extracts credentials such as saved passwords in applications [14].

- *Taidoor* (trace A_3) APT malware, a remote access trojan, was used to compromise targets since at least 2008. The threat actors sent out spear phishing emails to Taiwanese government email addresses [39].
- *Taleret* (trace A_7) APT malware was also used in the Taidoor campaign. Unlike Taidoor, it connected to Yahoo blogs to retrieve a list of C&C servers [13].
- *Tapaoux* (trace A_5) is an APT malware used by the Darkhotel APT campaign which appeared to have been active for seven years since 2007 [23]. The attackers also used spear phishing with advanced zero-day exploits.
- *Darkcomet* (trace A_9) is a remote access trojan, which was developed in 2008. The Syrian government used it to spy on dissidents during the Syrian Civil war in 2014 according to Fidelis Security [35]. It was also associated with operation hangover, a cyber espionage campaign against Pakistani organizations that took place from 2010 until 2013 [29].

We have evaluated our malware detection approach on malicious traces for all nine mentioned advanced persistent threats, see Table 4. All of these APTs have lead to severe damages on high-profile targets as described before. We randomly integrated the C&C requests and responses of A_{1-9} to the benign test data sets S_1, S_2 and C_1. Our malware detection approach successfully detects all C&C requests, as can be seen in Table 8. Therefore, the true positive rate is one and the false negative rate is zero for all tested data sets. The false positive rate has not changed as compared to Table 7.

Table 8. Advanced persistent threat detection results.

data set	TPR	FPR	TNR	FNR
$\{S_1, M\}$	1.000	0.005	0.995	0.000
$\{S_2, M\}$	1.000	0.003	0.997	0.000
$\{C_1, M\}$	1.000	0.003	0.997	0.000

4 Discussion

In total 2605 of 2611 C&C requests (99.8%) are unrelated in the Web request graphs of the malicious datasets of general purpose malware M_{1-5} and APT malware A_{1-9}. Only six C&C requests are related. There are three traces that each contain a pair of related C&C requests. Each pair of C&C requests happens due to an URL redirection. Table 9 shows the three traces that contain related C&C requests. It can be seen that the HorstProxy trace is the only tested trace without any unrelated C&C request. All other 57 traces contain unrelated C&C requests, which are identified by our approach. This means that our approach detects C&C traffic in 57 out of 58 malicious traces (including APT traces).

Our experiments show that most C&C requests are indeed unrelated and are correctly identified as malicious. Only HTTP redirects of malicious requests

Table 9. Malicious traces with false negatives

set	trace	# related	# unrelated
M_3	HorstProxy_EFE5529D697174914938F4ABF115F762-2013-05-13 [7]	2	0
M_5	BIN_sality_CEAF4D9E1F408299144E75D7F29C1810 [7]	2	6
M_5	InvestigationExtraction-RSA_Sality [2]	2	8

are not identified. Such redirects could be merged in a request graph to single nodes. In this case, we would have identified the redirected C&C requests as malicious. However, we did not combine redirections and redirection targets into single nodes in this work as this could result in additional false positives if the benign requests have no relation to other nodes in the Web graph.

Our approach works on single clients and equally good for general purpose and APT malware. This is a strong result considering the fact that some of the considered APTs were active for years without being noticed. Our approach would have detected the general purpose/APT malware in few minutes. We consider 30 second time windows, this means that a real-time implementation of the detection approach can react in a granularity of 30 seconds to a C&C request. Our C&C detection approach can significantly improve the response time to attacks, which might last for several years until the vulnerability has been identified and patched.

As with any malware detection approach, attackers might change their behavior in order to better obfuscate their activities and circumvent detection. However, the fact that the investigated malware traces caused considerable damage, clearly shows that there is need for an approach like ours. We see several future challenges for our approach. Firstly, the C&C traffic can be adapted such that it sends related requests that mimic benign Web browsing traffic. Fake referrers could be detected by analyzing the popularity of links in the request tree, as outlined in our previous work [17]. Further, click detection could be used to analyze the sites visited by users. In case malware builds its own sequence of related requests, this could still identify the C&C channel since it is a Web site that is visited repeatedly.

Secondly, C&C requests can set a referrer to a benign request in order to better hide inside Web browsing. In this scenario, one has to take the referrer field into question. One will look at other features such as the timing behavior between related requests in order to see, whether the general purpose/APT malware performs the requests in the same manner as a Web browser.

Thirdly, the number of false positives might increase in future due to a growing complexity of Web request graphs and removal of referrers due to privacy constraints. In this case, one can develop further heuristics to improve the link completion. Furthermore, one could reduce the number of detection alerts by summarizing the unrelated requests on domain level. One can send a detection alert whenever a server has been contacted with unrelated requests for at least a given number of times. This should significantly reduce the number of alerts.

Finally, benign and malicious Web traffic might mostly consist of HTTPS connections instead of HTTP. This challenge can be overcome by using man-in-the-middle proxies, which allows a network application to inspect the otherwise encrypted traffic. This is already done in many companies and other high-profile targets.

5 Related Work

Supervised malware detection: Most related approaches that detect C&C channels in network traffic use supervised machine learning that trains on labeled malware samples. For instance, Perdisci et al. [33] propose a scalable malware clustering system for HTTP-based malware, which has a detection rate of 65%-85%. BotFinder [38] creates botnet traffic inside a controlled environment in order to learn bot detection models with a detection rate of 80%. ExecScent [26] learns adaptive control protocol templates in order to determine a good trade-off between true and false positive rates. DISCLOSURE [8] detects C&C traffic using a large-scale NetFlow analysis that relies on labeled training samples. Their detection rate is about 90%. They use external reputation scores to reduce the false positive rate.

HAS-Analyzer [22] uses a random forest classifier with an accuracy of 96% and a false positive rate of 1.3% without using any whitelist. JACKSTRAWS [19] correlates host behavior with network traffic to detect C&C channels of botnets. The authors use machine learning with labeled malicious samples and achieve a detection rate of 81.6% at a false positive rate of 0.2%. In contrast to [8,19,22,26,27,33,38], our detector works without learning from malware samples while providing a very high true positive rate and a low false positive rate.

Our link completion algorithm uses similar techniques as the ones developed by Nelms et al. [27] for the WebWitness system. WebWitness classifies the infection method as malicious drive-by, social engineering and update/drop downloads. However, in contrast to our approach, Nelms et al. do not build a malware detector, instead they focus on enabling forensic investigations of already identified malware infections.

Unsupervised malware detection: BotSniffer [16] presents an unsupervised network-based anomaly detector without prior knowledge of the malware. The detection rate is 100% with a false positive rate of 0.16%. BotMiner [15] clusters communication patterns and malicious activities traffic and identifies botnets due to a cross cluster correlation. The detection rate is 100% for HTTP bots with a false positive rate of 0.003%. However, BotSniffer and BotMiner only work on network level with multiple infected hosts. BotSniffer takes advantage of the observations that bots communicate in a similar spatial-temporal behavior to the malicious server. However, these approaches do not work for APTs, which only infect a single computer.

Burghouwt et al. [10] and Zhang et al. [43] correlate Web request graphs with user interactions to detect malware. Burghouwt et al. achieve a detection rate of 70%-80% with a false positive rate of 0.17%, Zhang et al. achieve an accuracy

of 96%. In contrast to our approach, they continuously record user interactions such as clicks and keystrokes. We only record the accessed domains in an initial training stage for click detection. After training is completed, we do not need to record any user interaction. Furthermore, [10,43] do not employ link completion.

Oprea et al. [30] propose a graph-theoretic framework based on belief propagation to detect early-stage APT campaigns. Unlike our approach, they do not fully rely on the network traffic but also collect registration information of the accessed domains. Furthermore, their approach only works on the enterprise network level, while our approach also works for single hosts.

6 Conclusion

We propose a novel APT and general purpose malware detection approach that identifies command and control channels in Web request graphs. Our approach relies on the observation that malware communicates to malicious servers periodically with single, unrelated HTTP requests. This communication pattern is different from Web browsing where page requests usually result in several requests that are related to each other. Software applications and the operating system also send single, unrelated request to dedicated servers. Their traffic patterns are similar to C&C requests. However, we assume that these servers are well known and can be whitelisted. In our experiments, we whitelist 37 update servers and certificate authorities.

Our malware detection approach improves the request graphs of related work by automatically detecting user clicks (click detection) and restoring dependencies between unconnected requests (link completion). In a first step, we use a random forest classifier to detect user clicks inside request graphs. Our classifier relies on seven features on node and graph level, such as content type (node level) and number of children (graph level). We evaluate our click detection classifier with generated benign browsing traffic that accesses the Alexa top 250 of Switzerland. Our classifier has a f1 score of 95% for Firefox and 96% for Chrome.

In a second step, we connect unrelated nodes to their most likely parent. We evaluated link completion on the script-generated traffic and real user traffic provided by ClickMiner [25]. We find that 91-97% of the HTTP requests in Web browsing are already connected to other requests after extracting the request graphs with Hviz [17]. Our heuristic algorithm, called link completion, connects unrelated requests to their most likely parent. Our experiments show that link completion adds many missing links to the request graph. Between 99.5% and 99.7% of the benign nodes are connected after link completion.

We have evaluated our detection approach for 49 general purpose malware and nine APT malware packet traces, which are publicly available. The post-infection C&C traffic of these traces consists of 99.8% unrelated requests and 0.2% related requests. When we randomly merge these requests to benign Web browsing traffic, we can detect 99.5% of the malware and 100% of the APT C&C requests while having a false positive rate of 0.3-0.5%. Our approach can be applied in real-time at the granularity of single clients. This means that C&C

traffic can be recognized in the order of 30 seconds. The Web request graphs can be tracked and completed on a per client base. This allows the proposed algorithm to scale horizontally as well as vertically. We hope that our findings will help to significantly shorten the timespan until new pieces of malware, especially those used by APTs, are discovered.

References

1. APT Case RUAG. Technical Report. GovCERT.ch (2016-05-23), https://www.melani.admin.ch/dam/melani/en/dokumente/2016/technical%20report%20ruag.pdf.download.pdf/Report_Ruag-Espionage-Case.pdf
2. Contagiodump Blog. http://contagiodump.blogspot.com, [Online; accessed January-2017]
3. HTTP Access Control. https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS, [Online; accessed January-2017]
4. HTTP Method Definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html, [Online; accessed January-2017]
5. Malware Capture Facility Project. http://mcfp.weebly.com, [Online; accessed January-2017]
6. Malware-Traffic-Analysis Blog. http://www.malware-traffic-analysis.net, [Online; accessed January-2017]
7. pcapanalysis. http://www.pcapanalysis.com, [Online; accessed January-2017]
8. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: Detecting botnet command and control servers through large-scale netflow analysis. In: Proc. Annual Computer Security Applications Conference. pp. 129–138. ACSAC '12, ACM (2012)
9. Bugzilla: Bug 1282878. https://bugzilla.mozilla.org/show_bug.cgi?id=1282878, [Online; accessed February-2017]
10. Burghouwt, P., Spruit, M., Sips, H.: Detection of Covert Botnet Command and Control Channels by Causal Analysis of Traffic Flows, pp. 117–131. Springer International Publishing (2013)
11. Chen, P., Desmet, L., Huygens, C.: A study on advanced persistent threats. In: IFIP International Conference on Communications and Multimedia Security. pp. 63–72. Springer (2014)
12. Cylance: Operation cleaver report. http://cdn2.hubspot.net/hubfs/270968/assets/Cleaver/Cylance_Operation_Cleaver_Report.pdf, [Online; accessed February-2017]
13. FireEye: Evasive Tactics: Taidoor. https://www.fireeye.com/blog/threat-research/2013/09/evasive-tactics-taidoor-3.html, [Online; accessed February-2017]
14. FireEye: To Russia With Targeted Attack. https://www.fireeye.com/blog/threat-research/2012/12/to-russia-with-apt.html, [Online; accessed February-2017]
15. Gu, G., Perdisci, R., Zhang, J., Lee, W.: Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proc. USENIX Security Symposium. USENIX Security '08 (2008)
16. Gu, G., Zhang, J., Lee, W.: Botsniffer: Detecting botnet command and control channels in network traffic. In: Proc. Network and Distributed System Security Symposium (NDSS '08) (2008)

17. Gugelmann, D., Gasser, F., Ager, B., Lenders, V.: Hviz: Http(s) traffic aggregation and visualization for network forensics. Digital Investigation 12, Supplement 1, pp. 1–11 (2015), Proc. DFRWS Europe (DFRWS 2015 Europe)
18. IETF: Online Certificate Status Protocol - OCSP. https://tools.ietf.org/html/rfc6960, [Online; accessed February-2017]
19. Jacob, G., Hund, R., Kruegel, C., Holz, T.: Jackstraws: Picking command and control connections from bot traffic. In: Proc. USENIX Security Symposium. USENIX Security '11 (2011)
20. Jones, M.: Protecting privacy with referrers. https://www.facebook.com/notes/facebook-engineering/protecting-privacy-with-referrers/392382738919/ (2010), [Online; accessed February-2017]
21. Kaspersky Lab: The Nettraveler (aka 'Travnet'). https://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/kaspersky-the-net-traveler-part1-final.pdf, [Online; accessed January-2017]
22. Kim, S.J., Lee, S., Bae, B.: Has-analyzer: Detecting http-based c&c based on the analysis of http activity sets. TIIS 8(5), pp. 1801–1816 (2014)
23. Lab, K.: The Darkhotel APT, a story of unusual hospitality. https://securelist.com/files/2014/11/darkhotel_kl_07.11.pdf, [Online; accessed February-2017]
24. Mandiant: APT1 - Exposing One of China's Cyber Espionage Units. https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf, [Online; accessed February-2017]
25. Neasbitt, C., Perdisci, R., Li, K., Nelms, T.: Clickminer: Towards forensic reconstruction of user-browser interactions from network traces. In: Proc. ACM CCS '14. pp. 1244–1255. ACM (2014)
26. Nelms, T., Perdisci, R., Ahamad, M.: Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates. In: Proc. USENIX Security Symposium. pp. 589–604. USENIX, Washington, D.C. (2013)
27. Nelms, T., Perdisci, R., Antonakakis, M., Ahamad, M.: Webwitness: Investigating, categorizing, and mitigating malware download paths. In: Proc. USENIX Security Symposium. pp. 1025–1040. USENIX (2015)
28. NIST: Managing Information Security Risk. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-39.pdf, nIST Special Publication 800-39. March 2011.
29. Norman: Operation Hangover. http://enterprise-manage.norman.c.bitbit.net/resources/files/Unveiling_an_Indian_Cyberattack_Infrastructure.pdf, [Online; accessed February-2017]
30. Oprea, A., Li, Z., Yen, T.F., Chin, S.H., Alrwais, S.: Detection of early-stage enterprise infection by mining large-scale log data. In: Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks. pp. 45–56. DSN '15, IEEE Computer Society (2015)
31. Paxson, V.: Bro: A system for detecting network intruders in real-time. Computer Networks 31(23-24), 2435–2463 (Dec 1999)
32. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
33. Perdisci, R., Ariu, D., Giacinto, G.: Scalable fine-grained behavioral clustering of http-based malware. Computer Networks 57(2), 487–500 (Feb 2013)

34. Proofpoint: Nettraveler apt targets russian, european interests. https://www.proofpoint.com/us/threat-insight/post/nettraveler-apt-targets-russian-european-interests, [Online; accessed January-2017]
35. Security, F.: Looking at the Sky for a DarkComet. https://www.fidelissecurity.com/sites/default/files/FTA_1018_looking_at_the_sky_for_a_dark_comet.pdf, [Online; accessed February-2017]
36. SeleniumHQ: http://www.seleniumhq.org, [Online; accessed January-2017]
37. Symantec: Internet security threat report. Tech. Rep. 21, Symantec (Apr 2016), https://www.symantec.com/security-center/threat-report
38. Tegeler, F., Fu, X., Vigna, G., Kruegel, C.: Botfinder: Finding bots in network traffic without deep packet inspection. In: Proc. of the Int. Conference on Emerging Networking Experiments and Technologies (CoNEXT). pp. 349–360. ACM (2012)
39. TrendMicro: The Taidoor Campaign. https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp_the_taidoor_campaign.pdf, [Online; accessed February-2017]
40. Vassio, L., Drago, I., Mellia, M.: Detecting user actions from HTTP traces: Toward an automatic approach. In: Int. Wireless Communications and Mobile Computing Conf. (IWCMC). pp. 50–55 (2016)
41. W3C: Referer Policy. https://w3c.github.io/webappsec-referrer-policy, [Online; accessed February-2017]
42. Xie, G., Iliofotou, M., Karagiannis, T., Faloutsos, M., Jin, Y.: Resurf: Reconstructing web-surfing activity from network traffic. In: Prod. Int. Conf. on Networking. IFIP (2013)
43. Zhang, H., Banick, W., Yao, D., Ramakrishnan, N.: User intention-based traffic dependence analysis for anomaly detection. In: IEEE Symposium on Security and Privacy Workshops. pp. 104–112 (May 2012)