DorkPot: A Honeypot-based Analysis of Google Dorks

Florian Quinkert
Ruhr-University Bochum
florian.quinkert@rub.de

Eduard Leonhardt
Ruhr-University Bochum
eduard.leonhardt@rub.de

Thorsten Holz
Ruhr-University Bochum
thorsten.holz@rub.de

Abstract—**Attackers use search engines to find vulnerable systems and interesting information such as passwords, hidden files, or other kinds of sensitive information on the Internet. Besides common search terms, they use advanced search parameters called *Google dorks* to find only results with specific strings in the URL or files with a particular extension. So far, only a few works have empirically studied Google dorks, e.g., if they are still in use, which Google dorks attackers use, and how often as well as how old the used Google dorks are.**

In this paper, we study this type of attacks from a different perspective and present *DorkPot*, a dynamic, low-interaction webserver honeypot to detect Google dork related requests and, thereby, analyze such attacks in the wild. DorkPot uses Google dorks as input and creates a website for each Google dork, which matches the Google dork's content, e.g., strings in the title field or the URL. Hence, we ensure the particular website can be found later via a Google search with the corresponding Google dork. To evaluate our prototype implementation, we deployed DorkPot with more than 4,000 Google dorks as input on ten instances of a cloud provider. Throughout more than ten months, we collected almost 9,000 clicks for 371 different Google dorks. Our analysis reveals that the top-ten Google dorks were responsible for more than 50% of the clicks, were mostly published in mid-2017 and searched for various online devices, such as IP cameras or routers, as well as passwords and database backups. In particular, three of the top-ten Google dorks targeted Internet of Things devices and another two searched for passwords and related files with authentication information.

I. Introduction

Modern attacks on IT systems consist of multiple phases [32]. In a first phase called *reconnaissance*, the attacker identifies vulnerable systems and searches for helpful information. In the case of web applications, attackers use for example vulnerability scanners like Nessus [33] or Nexpose [25] which provide an easy-to-use interface. Additionally, attackers send specially crafted URLs to a huge number of websites to fingerprint devices and detect whether certain software versions are installed. Furthermore, it is possible to use search engines to discover vulnerable websites or collect helpful information [16]. In some cases, just using common search terms and linking them with operators like *AND* or *OR* is sufficient to obtain interesting results. Back in 2005, Long discovered that advanced search parameters as an even better way to obtain interesting information than just using search terms [18]. Advanced search parameters (often referred to as *Google Hacking* or *Google Dorking* in the context of computer security) include for instance *intitle* (limit results to those having a certain string in the HTML *title* tag) or *inurl* (limit results to those having a certain string in the URL). Search strings with advanced search parameters are called *Google dorks* and allow precise search queries, e.g., searching with the Google dork *"index of" inurl:wp-content/* returns a list of websites with installed WordPress blogging software [14]. Additionally, another common use case for Google dorks is finding information not intended to be exposed publicly, such as database backups or password files. For example, the Google dork *"– Dumping data for table" ext:sql* returns SQL database backups [9]. The advanced search parameter *ext* limits search results to those with the file extension *sql*.

In 2014, Zhang et al. performed Google searches with about 1,000 Google dorks and found more than 300,000 potentially vulnerable websites, showing that Google dorks are an effective method to identify vulnerable websites [35]. However, it remains unclear whether attackers use Google dorks, which Google dorks are particularly interesting, how often they are used etc. Therefore, in this paper, we close this research gap and present DorkPot, a dynamic honeypot to detect and measure Google dork usage. According to Spitzner, a honeypot is a "security resource whose value lies in being probed, attacked, or compromised" [30]. Stoll [31] in the late 1980s and Cheswick [5] at the beginning of the 1990s first described honeypots. Spitzner categorizes honeypots based on their interaction level into low-interaction and high-interaction honeypots. A low-interaction honeypot is an emulated resource, which is easy to set up and maintain, but only provides a superficial insight into attackers' behavior and is easy to detect as a honeypot. A high-interaction honeypot is a real system, which makes it difficult to maintain and detect, but provides deeper insights into attack techniques. Additionally, it is possible to differentiate between server honeypots and client honeypots [24]. A server honeypot runs server services and passively waits for attackers. In contrast, a client honeypot emulates client services, e.g., a browser or PDF reader, and needs input data to test whether an input attacks the honeypot. According to these definitions, DorkPot is a low-interaction server honeypot. In contrast to traditional honeypots, our reason to develop DorkPot was to get a tool to measure the prevalence of a large variety of Google dorks instead of a honeypot which reveals deeper insights into attackers' behavior. Nevertheless, it is possible to use DorkPot to monitor

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2019
24 February 2019, San Diego, CA, USA
ISBN 1-891562-60-6
https://dx.doi.org/10.14722/madweb.2019.23xxx
www.ndss-symposium.org

whether a smaller number of Google dorks is used, e.g., a company could monitor Google dorks used to find systems running their software because an increased interest could indicate an unknown vulnerability.

As input, DorkPot takes a list of Google dorks, obtained for example from the Google Hacking Database (GHDB) [10]. In the second step, we create a website for each input Google dork, specifically crafted to match the Google dork, in an automated way. For example, an *intitle* parameter in the Google dork will result in a website containing an HTML *title*-tag with this specific content. We refer to the generated websites as *honeypages* and bundle all of them in a web honeypot deployable for instance on a machine rented from a cloud provider. Once a search engine crawler visits the honeypages and indexes them, the honeypot can detect requests originating from Google dorks and thus provides an overview of Google dork related requests.

In our empirical evaluation, we used more than 4,000 Google dorks from GHDB as input for DorkPot and deployed the generated honeypot on ten instances of a cloud provider. Over an evaluation period of more than ten months, we collected almost 9,000 requests from 371 different Google dorks. Our analysis showed that the top-ten Google dorks were responsible for more than 50% of the received clicks. Among the ten most popular Google dorks, which were mainly published in mid-2017, three Google dorks targeted Internet of Things (IoT) devices which is not surprising taking the emergence of IoT botnets into consideration. Additionally, two Google dorks searched for passwords, which are an easy to use resource, without necessity to exploit for example vulnerable systems. Surprisingly, our analysis revealed that even old Google dorks published between 2004 and 2006 were used.

In summary, we make the following contributions:

- We propose a method to dynamically generate honeypages based on arbitrary Google dorks and build a web honeypot called DorkPot.
- We set up DorkPot with more than 4,000 Google dorks from GHDB and deploy it on ten instances of a cloud provider to collect requests for more than five months.
- To the best of our knowledge, we are the first to analyze Google dork usage from a server perspective.

The remaining sections of this paper are structured as follows: first, we review related work in Section II. Next, we introduce DorkPot in Section III. We describe our evaluation in Section IV and discuss threats to validity of our work in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Recent research dealt with honeypots, the analysis of Google dorks, and malicious usage of search queries. In the following, we discuss related work and explain the differences compared to our approach.

a) Honeypots: Over the years, many different types of honeypots were proposed and, hence, we focus only on web honeypots. Snare is a low-interaction web honeypot written in Python and the successor of Glastopf [26]. Additionally, it provides a clone functionality to copy existing websites and use them as honeypots. In 2005, McGeehan et al. introduced Google Hack Honeypot [13] which contains a small number of hand-crafted websites. Each website belongs to a single Google dork and attracts attackers' attention when they use the particular Google dork. In contrast to our work, the generation of websites is not automatized so that a manually written website is necessary for each new Google dork, which is a time-consuming task for a high number of Google dorks. Therefore, the approach is not feasible to analyze the prevalence of a high number of Google dorks. Furthermore, the available Google dorks are more than ten years old and therefore irrelevant nowadays.

John et al. generated templates based on malicious queries from search engine logs [17]. They combined these templates with a plain web server and four manually installed web applications to obtain a honeypot, which attracted more than 44,000 attacker visits. Moreover, they provide an extensive discussion on how to distinguish between crawler, benign, and malicious visits. Both John et al. as well as this work focus on the automatic generation of honeypots. In contrast to our work, John et al. cover only a small fraction of Google dorks.

In 2013, Canali et al. set up 500 honeypot websites to discover what happens after a website has been compromised [2]. On each honeypot, they installed content management systems, web shells, and a static website to attract attacker's attention. After an evaluation period of 100 days, they had collected about 85,000 files which were created in approximately 6,000 attacks. Furthermore, they performed an analysis of the referrer headers and detected that the attackers used a variety of Google dorks to find the honeypot websites. However, in contrast to our work, the honeypot pages are not intended to be found via Google dorks and do not cover a large set of Google dorks.

The high-interaction honeypot presented by Catakoglu et al. collects Indicators of Compromise (IoCs, i.e., artifacts found on a compromised computer system, such as IP addresses or malware hashes) [3]. Eventually, they extracted 96 IoCs throughout four months. Even though they mention the importance of Google dorks, they do not describe how attackers found the honeypot.

Catakoglu et al. investigated the Tor network by deploying a high-interaction honeypot to study whether adversaries use the same attack techniques on the dark side of the web as on the traditional Internet [4]. In an evaluation over seven months, they detected a variety of attacks, such as upload of web shells, defacements, path traversal attacks as well as attacks from the traditional Internet via Tor proxies.

b) Google dorks: Recent research dealt already with the analysis of Google dorks. However, it focused on the client side and did not measure the prevalence and distribution of Google dork usage.

Toffalini et al. performed a large-scale study of existing Google dorks [34]. They proposed an additional classification based on information a Google dork uses. Additionally, they introduce two novel defense techniques: addition of a random string to a URL which is removed by an Apache module when a user visits the page and addition of non-visible random characters into words often used for fingerprinting. On the contrary, our work aims at understanding the usage of Google dorks from a server perspective.

The relationship between Google dorks and targeted websites was analyzed by Zhang et al. in 2014 [35]. They used about 1,000 Google dorks to collect a set of 300,000 potentially vulnerable websites out of which they could verify 6,000 to be vulnerable. Furthermore, they discovered most Google dorks target vulnerabilities with high severity scores but low attack complexity which can be rendered useless often by removing keywords from a website so that the website is no longer found via Google dorks. Compared to our work, which concentrates on the server side, Zhang et al. focus solely on the client side and show it is, in fact, possible to find vulnerable websites via Google dorks.

Pelizzi et al. introduced Gd0rk and searched with Google dorks for XSS vulnerable websites [23]. It uses a small list of precompiled keywords to generate Google dorks and afterwards tests whether the found websites are vulnerable to XSS attacks. During one month of evaluation, Pelizzi et al. found 200,000 vulnerable websites (0.94% of all scanned websites). They aim at generating Google dorks to find vulnerable websites whereas we target the detection of Google dork usage from a server perspective.

SearchAudit, a tool presented by John et al., identifies malicious search queries from search query logs [16]. It is based on two steps: at first, it identifies malicious search queries similar to a given seed and builds regular expressions to find more malicious search queries. In a second step, John et al. analyze the found malicious search queries from step one as well as the correlation between the search query and other attacks.

Zhang et al. performed a large-scale analysis of Bing search query logs to identify bot queries [36]. They searched for query intention, i.e., the topic of a query, and query origination, especially hosts that are part of large-scale efforts. In an evaluation, they identified more than 3 billion bot queries with 33% of them searching for vulnerabilities and 11% for user account information.

III. System Design

In the following, we describe DorkPot's structure and technical details. At first, we give a short overview of the full system, followed by a detailed description of the individual components.

Figure 1 shows the four main components *crawler*, *honeypage-generator*, *honeypot* and *management* as well as their interaction with each other and their environment. The crawler component crawls a data source, e.g., exploit-db's Google Hacking Database (GHDB) [10], to collect all available Google dorks (1). The honeypage-generator component first uses this information to generate one honeypage for each Google dork, which is later indexed by a search engine such as Google and found when someone enters the corresponding Google dork (2). Additionally, it bundles all generated honeypages with a web server and logging modules to generate the deployable honeypot. Later on, we deploy the honeypot, e.g., on a cloud service, and wait until the honeypages are indexed (3). When an attacker uses a particular Google dork and enters the corresponding honeypage (4), the honeypot generates a log message and sends it to the management component (5).

A. Crawler

It is possible to generate arbitrary Google dorks by using one or multiple keywords with any string as an argument. Since we can not guess in advance which Google dork an attacker might come up with in future, DorkPot needs Google dorks as input to generate honeypages. The Internet provides various resources to obtain Google dorks. For example, people share Google dorks on paste websites like pastebin [22] or document sharing websites like scribd [28]. In addition, news websites and blog postings publish articles about new Google dorks [21], [15]. In some cases, people even use Twitter to propose new Google dorks [29]. However, it requires a lot of effort to collect Google dorks from all these sources and usually only a small portion of blog postings, tweets or pastes deals with Google dorks. Therefore, it is more useful to focus on a data source dedicated solely to Google dorks. By far, the biggest and most comprehensive data source for Google dorks is exploit-db's GHDB, which was used as well by Zhang et al. to conduct their study of Google dorks from the client's perspective [35]. Therefore, we focus on GHDB for the rest of this paper.

In particular, we crawl GHDB to obtain a list of up to date Google dorks. GHDB groups Google dorks into 14 different categories based on the information an attacker can find, e.g., Google dorks to find files containing passwords or Google dorks to find pages containing login portals (see Section IV-A for detailed information about GHDB). As of 2017/10/06, GHDB includes 4164 Google dorks. In some cases, the same Google dork is added to multiple categories and gets a unique ID for each addition so that in total there are 3897 unique Google dorks in GHDB. For each Google dork, we collect the Google dork itself, the submission date to GHDB, the category, and the description. Furthermore, we emphasize that the rest of our infrastructure works with Google dorks from other data sources in the same way so that it is easy to integrate Google dorks from other data sources by implementing another crawler component for a different data source.

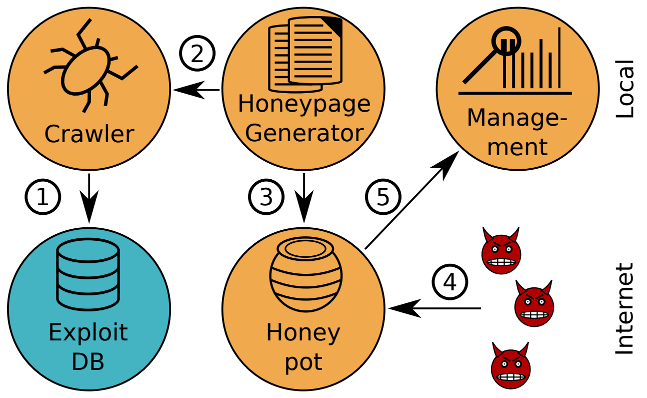

Fig. 1. Overview of main components and their interaction. *Crawler* crawls a data source like exploit-db's Google Hacking Database (1). *Honeypage-Generator* generates honeypages and bundles them with a web server and logging modules (2). *Honeypot* can be deployed to cloud service (3). Honeypot logs attacks from attacker (4) and sends the logs to management component (5).

TABLE I. Overview of Google dork keywords and their descriptions. Google dork keywords marked with * are available in a version with prefixed "all" which shows only results containing all arguments.

Keyword	Description
intitle *	Title-tag contains given argument
inurl *	URL contains given argument
intext *	Website content contains given argument
filetype	Limits results to given filetype
ext	Limit results to files with given extension
site	Limit results to a given website

B. Honeypage-Generator

The honeypot generator takes a list of crawled Google dorks as input and parses them to generate one honeypage for each Google dork. Instead of generating honeypages on our own, we could use Google dorks as search queries on Google and use the results as templates for our honeypages. However, during our research, we used lots of Google dorks and found multiple reasons to create honeypages on our own. First, Google prevents usage of Google dorks by displaying hard to automatically solve captchas after very few requests. Therefore, either a lot of manual effort or building an infrastructure with for example changing IP addresses is necessary to collect Google results automatically. Second, Google search results contain in some cases less relevant results, which are not suitable for honeypage generation. For example, searching with a Google dork for a certain configuration file can reveal forum postings about the configuration file. Thus, a complex algorithm is necessary to select proper search results. Third, some Google dorks search for sensitive information, such as passwords or confidential documents. We want neither access nor copy this information due to legal and ethical reasons. Eventually, a self-generated honeypage needs less effort, prevents legal issues, and guarantees each part of the Google dork is available.

Table I shows an overview of keywords available in Google dorks. A Google dork consists of a keyword, followed by a colon, followed by one (line 1 in Listing 1) or multiple arguments (line 2). In case of multiple arguments, we have to use quotation marks to indicate that all arguments belong to the keyword. Otherwise, the search engine interprets only the first argument as belonging to the keyword and the remaining ones as regular search terms. Additionally, it is possible to concatenate multiple Google dorks (line 3) and use Google dorks together with regular search terms (line 4). Furthermore, quotation marks before the keyword and after the last argument are possible.

Listing 1. Google dork syntax
```
1  keyword:argument
2  keyword:"argument argument"
3  keyword:argument keyword:argument
4  keyword:argument search\_term
5  "keyword:argument argument"
```

We divide the honeypage generation into two parts: first, we split the Google dork into pairs of keyword and arguments considering the various Google dork structures. Depending on the keyword type, the honeypage generation slightly differs. The keyword *intitle* searches for the HTML *title* tag and checks whether it contains the given arguments. The keyword *intext* searches for an HTML *p* tag with the given arguments. Therefore, in case of *intitle* and *intext*, we start with a generic HTML template and add an HTML *title* tag containing the given arguments or an HTML *p* tag, respectively. The keyword *inurl* checks whether a URL contains a given set of arguments or not. Therefore, we create a folder structure containing the requested arguments and place a honeypage with generic content into the folder. Hence, the link to the honeypage contains the arguments of the *inurl* keyword. The keywords *filetype* and *ext* limit search results to certain filetypes or files with the particular ending. In those cases, we generate a generic honeypage with the requested file ending.

The keyword *site* limits search results to a particular domain. Even if we generate a honeypage for all other keywords of a Google dork with the keyword *site*, we would not be able to add it to the domain specified by the keyword site because the domain is not under our control. Therefore, the honeypage would not be found when someone uses a web search with the Google dork. Hence, we omit those Google dorks. However, our set of 4164 crawled Google dorks from GHDB contains only 60 Google dorks with the keyword *site*, which is a negligible amount of Google dorks (1.4%).

Furthermore, we filter broken Google dorks and Google dorks which use keywords in an unintended way. In particular, we filter based on the following rules: first, we filter each Google dork with a space between keyword and colon (*keyword :argument*) which results in interpreting the keyword and colon plus argument as regular search terms (71 cases out of 4164 total Google dorks). Most likely, the authors of such Google dorks meant *keyword:argument*, without space. Second, we filter each Google dork with a space between colon and argument (*keyword: argument*) which causes Google to interpret keyword plus colon and argument as regular search terms (nine cases). Again, the intended Google dork should look like *keyword:argument*, without space. Third, we filter each Google dork with one or more characters before a keyword (***a**keyword:argument*) which renders the Google dork useless (four cases). Fourth, we found 11 Google dorks with one keyword and multiple arguments in parentheses (*keyword:(argument1 argument2 argument3)*) and suppose the creator of the Google dork expected each argument to be part of the Google dork. However, Google interprets only keyword and parenthesis plus argument1 as Google dork and treats argument2 as well as argument3 plus parenthesis as standard search terms. Therefore, we filtered those Google dorks as well.

This filtering is worth discussing since even broken Google dorks return search results and could be used with malicious intentions. However, search results for a broken Google dork are different from those for the corresponding correct Google dork, e.g., a space between keyword and colon will result in a search with the keyword as a regular search term and, therefore, different search results than expected. Thus, our honeypage would be found among unrelated websites. Furthermore, in most cases, search results for a broken Google dork are easily identifiable as less interesting which makes it unlikely to collect requests when including them in the honeypot. In contrast, it is possible to correct the mistyped Google dorks. But we argue that many users copy and paste the Google dorks so that it is not useful to fix them. Hence, because including and correcting broken Google dorks both has unwanted effects, we decided to remove the, in comparison with the overall number of Google dorks, small number of broken Google dorks.

Additionally, we make sure each honeypage has a different filename by composing the filename out of two randomly chosen terms from a list of terms related to topics an attacker might be interested in, e.g., business.

Traditional honeypots aim at emulating a system's behavior as close as possible so that our rather superficial honeypages might raise questions. However, our focus is on detecting Google dork usage instead of detecting what an attacker does on a compromised system. As you can see later, it is sufficient to identify the usage of a particular Google dork already when the corresponding honeypage is among the returned search results. Therefore, the degree of emulation is adequate for our purpose. In contrast, generating honeypages for each Google dork automatically in a way that they are not distinguishable from their real counterparts is not possible due to a large number of different systems targeted by Google dorks (see the low number of emulated Google dorks by Google Hack Honeypot in comparison [13]).

C. Honeypot

After we generate a honeypage for each Google dork, we bundle all generated honeypages with an Nginx web server [20] into a Docker [7] container. The use of Docker both eases the deployment and adds a level of security since a potential attacker only gets access to the Docker container instead of the full system. Instead of using a plain web server, we considered using a web honeypot, such as Glastopf [27], as the foundation. However, we are only interested in capturing the requests originating from Google dorks and do not need any further interaction capabilities. Thus, we decided to add the honeypages to a web server.

Additionally, we add logging capabilities to send received requests to the management component. Afterwards, the dockerized honeypot is deployable on an arbitrary machine connected to the Internet, e.g., an instance rented at a cloud service. Attackers will use Google dorks in a web search to find vulnerable systems or interesting information. Therefore, we request Google to index the honeypages to make sure they can be found by Google dorks. We call DorkPot a dynamic honeypot because it can take arbitrary Google dorks as input without the necessity of manual interaction in the generation of honeypages. However, adding new Google dorks on a daily basis is not feasible because it requires manual interaction to initiate Google's indexing process. Furthermore, Google limits the number of indexing requests. Also, we register a domain for each honeypot to let it look more like a real system and, therefore, attract more attention.

D. Management

The management component is responsible for collecting and storing received requests from all honeypots to provide an overview of incoming requests and the system's status. We base the management component on the Elastic Stack [8] to provide easy access to the collected data. Additionally, the Elastic Stack supports the generation of visualizations which further improves the understanding of ongoing actions.

IV. Evaluation

The following section describes our evaluation. At first, we give an overview of our test setup and dataset, followed by an explanation of how we used Google Search Console to analyze the search requests our honeypots received and how we assigned originating Google dorks to them. Afterwards, we analyze the obtained Google dork requests in detail.

A. Test setup and dataset

For our evaluation, we scraped GHDB on 2017/10/06 to collect all 4,164 available Google dorks. GHDB started to collect Google dorks in 2003. Many of the old Google dorks target very old systems and are, therefore, not interesting for state-of-the-art attacks. Additionally, they consist in many cases only of search terms without Google dork specific keywords. This results in many Google search results not expected to be found by the particular Google dork, e.g., forum discussions instead of fingerprinted services. Thus, we first focused only on Google dorks added to GHDB from the beginning of 2013 on to cover the last five years, which led to 715 remaining Google dorks. Out of those Google dorks, we removed 36 because they contain the keyword *site*, five because they contain a space between keyword and colon and two because they try to connect a keyword with multiple arguments by using parentheses. Eventually, we started our evaluation with 672 different Google dorks. A first evaluation revealed that even old Google dorks from 2013 were requested regularly. Therefore, we added the remaining Google dorks published before 2013 on 2018/02/12 to gain an even broader understanding which Google dorks are in use.

Figure 2 shows the number of Google dorks in each of GHDB's fourteen categories. The category *Advisories and Vulnerabilities* contains with more than 1,000 Google dorks the highest number, which is expected since finding vulnerable systems is one of the main reasons for using Google dorks. Other categories with high numbers of Google dorks search for juicy information, login portals, passwords or online devices, such as surveillance cameras or routers, which all provide

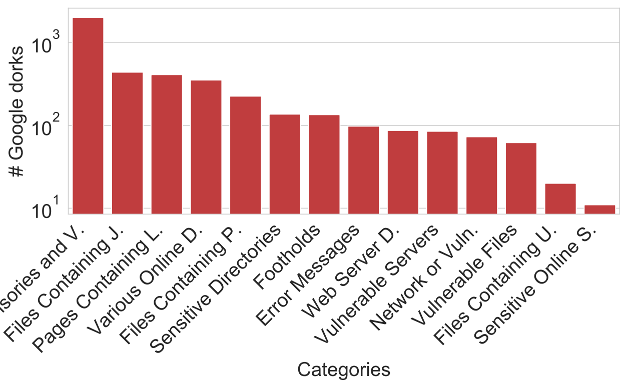

Fig. 2. Number of published Google dorks as a function of the GHDB categories

interesting and actionable information. The categories *Files Containing Usernames* and *Sensitive Online Shopping Info* contain only a small number of Google dorks. In both cases, other data sources, such as widely available data leaks, provide the same information more quickly so that Google dorks are less attractive in this area. Additionally, in a few cases, the categorization remains a bit shallow because a Google dork finding a vulnerable IoT device could belong to the the groups ”Advisories and Vulnerabilities” as well as ”Various Online Devices” and, depending on the kind of device, even to ”Vulnerable Servers”. Nevertheless, the categorization provides an overview for what purposes Google dorks are available.

We used the collected Google dorks to generate a honeypage for each Google dork as described in Section III-B. Additionally, we used domains and proper website templates from the following five industrial sectors to let the websites look more believable in case an attacker visits the main page instead of a honeypage: banking, medicine technology, insurance, consulting and architecture. However, the templates do not influence the generated honeypages. Afterwards, we registered two domains for each industrial sector and bundled the generated honeypages with the website templates to get ten deployable honeypots. For the evaluation, we rented ten instances at a cloud service provider and deployed one honeypot on each of them. In the following, we will refer to the honeypots as *hp_01* to *hp_10*. Furthermore, we requested Google via the Google Search Console [11] to crawl the honeypages of each honeypot to ensure search results from a Google search with the particular Google dork contain the corresponding honeypage. During our initial tests, we encountered that Google does not always index every honeypage on each honeypot. Therefore, we decided to run the system on multiple honeypots to increase the likelihood of Google indexing our honeypages.

B. Data filtering

We collected more than 3.2 million requests between 2017/10/06 and 2018/08/10. Of course, not all of these requests originated from web searches via Google dorks. A few years ago, the process of distinguishing between requests originating from Google and arbitrary requests was a lot easier than nowadays. Back then, Google provided not only the bare host name as referrer but additional information such as the search query string. Access to the search query string would enable us to recognize the used Google dork at our honeypots and, therefore, ease the process of assigning Google dorks to requests. However, Google stopped delivering this information in 2012 to protect users’ privacy [19].

Therefore, we need either rules to tell apart for example scanning activity, randomly sending requests, from actual Google dork usage or a reliable source providing us with information on performed search queries. Creating a set of rules to reliable detect Google dork related requests is not possible due to two main reasons. First, every part of a request can be spoofed. For example, a request to a honeypage with a Google domain in the referrer field can result from both using a Google dork on Google and sending a request with spoofed referrer field to the honeypage. Second, assuming we could reliably detect requests from Google searches, we still could not count every such request to a honeypage as originating from the corresponding Google dork. Most Google dorks consist of multiple parts, e.g., *inurl:foo intext:bar*. However, the corresponding honeypage already shows up among the search results when using only a part of the Google dork, e.g., *inurl:foo*. Therefore, without information on the search query, we would count a request originating from *inurl:foo* as originating from *inurl:foo intext:bar*. Hence, we need access to the search queries to reliable assign requests to Google dork usage.

Fortunately, Google Search Console provides information on all performed search queries, which contained one of our honeypages in the results. In particular, for each search query it gives the number of times a honeypage was clicked (*clicks*), the number of times a honeypage was among the search results, i.e., without being necessarily clicked (*impressions*), the *click through rate* (CTR), which is defined as clicks divided by impressions and the average position of a honeypage in the search results (*position*). Using Google Search Console has multiple advantages. First, it is a reliable data source to detect search queries used to find our honeypages because the search provider Google itself provides the data. Second, the impressions enable us to get information about performed search queries, even if a honeypage was not clicked but only has been displayed in the search results. In contrast, an analysis of the web server logfiles reveals only results when a honeypage was clicked. The impressions are especially useful because it is difficult to reliably place a honeypage among the top search results which increases the likelihood of being clicked as we will show later. Third, the position in the search results provides further insights when a honeypage is clicked. Fourth, we do not have to filter scanner traffic, e.g., from Google or other search engines. Therefore, we developed a script to daily collect search queries along with clicks, impressions, CTR and position for each honeypot from Google Search Console. The resulting dataset is the basis of our evaluation.

Linking both Google Search Console data and honeypot logs would enable us to get more information about attackers. However, we refrained from doing so due to the following two reasons. First, the focus of this study is the use of Google dorks in the wild rather than collecting information about attackers. Second, it is impossible to link the Google Search console data with the honeypot logs because, as described earlier, it remains unclear whether a request originates from Google dork usage or is a request with spoofed HTTP request fields.

C. Results

In the following sections, we describe and discuss our results. First, we analyze the distribution of clicks and impressions between our honeypots. Afterwards, we assign Google dorks from GHDB to the search queries and analyze which Google dorks are responsible for most clicks and impressions as well as how publication year and GHDB category influence these numbers. Finally, we examine the correlation between the position of a honeypage in the search results and the number of received clicks.

1) Distribution between honeypots: According to Google Search Console, the ten honeypots received in total 8,889 clicks and 62,054 impressions originating from 562 distinct search queries. Figure 3 shows the number of clicks (red)

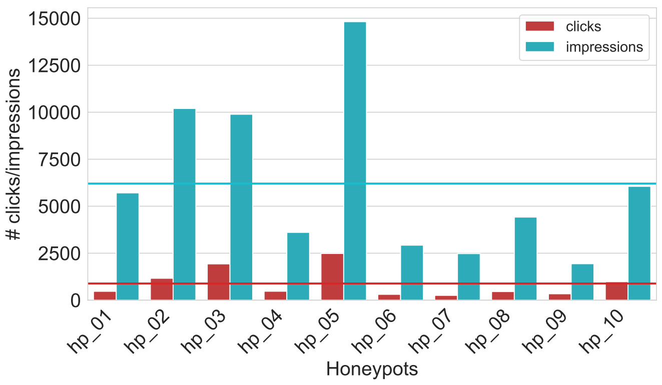

Fig. 3. Number of clicks and impressions each honeypot received according to Google Search Console along with mean number of clicks (red line) and impressions (blue line)

and impressions (blue) for each honeypot along with the mean number of clicks (red line) and impressions (blue line). The figure proofs the feasibility of our approach and shows that many search queries were used during our evaluation period to find our honeypots. Later on, we will show that a high percentage of these search queries belongs to Google dorks from our initial set.

We initialized every honeypot with the same set of honeypages and, therefore, expected an at least more comparable number of impressions for each honeypot. However, two reasons explain the high difference: first, Google uses more than 200 factors to determine whether and how relevant a result is, such as a user's browsing or search history [12], [6]. Thus, different users using the same Google dork receive different search results, which do not necessarily contain all of our honeypots. Second, Google has to index a honeypage before it is included in search results. However, the number of indexed honeypages varies for each honeypot and changes frequently. In Section V, we discuss implications of the different numbers of indexed honeypages on our approach. On average, each honeypot received 889 clicks and 6,205 impressions. The two honeypots *hp_04* and *hp_05* stick out with an above-average number of clicks. Both honeypots got a high number of impressions, i.e., they have been among the search results for a lot of search queries. Since this is a prerequisite of attracting attackers' attention, the high number of clicks is expected. On the contrary, *hp_07* and *hp_09* received among the least clicks which is explainable by the low number of impressions.

In the following, we will assign Google dorks to the obtained search queries and further analyze them. Since the number of indexed honeypages influences the number of clicks and impressions a honeypot receives, we will focus on the Google dorks itself instead of their distribution between honeypots.

2) Assigning Google dorks to search queries: Next, we further analyze the Google Search Console data and identify Google dork related search queries. In particular, we divide the search queries into three groups. The first group contains search queries equal to a Google dork in GHDB. The search queries in this group are the most interesting ones because they prove the correct operation of our approach. The second group includes search queries which are Google dorks but not in our initial set from GHDB. Some of the Google dorks are a subset of a Google dork in GHDB, e.g., the search query *inurl:foo* if the Google dork *inurl:foo intext:bar* is in GHDB. The search queries in the second group either originate from modified GHDB Google dorks, usage of Google dork lists different from GHDB or devised Google dorks. Eventually, it is not possible to resolve the origin doubtlessly. The third group contains all remaining search queries, e.g., search terms which accidentally matches one of our honeypages. The search terms in this group are less interesting because they do not belong to Google dork usage and are a side effect of our approach. However, we observed in some cases that search terms in this group fully match the arguments of a Google dork in GHDB, indicating knowledge of the particular Google dork. Potentially, an attacker avoids usage of Google dork keywords to cover his traces.

In total, Google Search Console lists 562 distinct search queries for all ten honeypots combined. After replacing different quotation marks, e.g., single or german quotation marks, with English quotation marks, 549 distinct search queries remain. We link 328 search queries to Google dorks in GHDB (59.7%), 148 search queries use Google dork keywords but are not in GHDB (27.0%) and 73 search queries consist of terms without Google dork keywords (13.3%). A further manual analysis of the 148 search queries in the second group revealed 78 search queries which differ only in quotation mark usage (*inurl:foo* vs *inurl:"foo"*), different usage of slashes (*inurl:foo* vs *inurl:/foo*) or missing space characters from a Google dork in GHDB. The latter one is a side effect of copying and pasting multi-line Google dorks. We decided to include these 78 search queries, which we could link to Google dorks in GHDB, in the first group so that this group now contains 406 search queries (74.0%). During the rest of the evaluation, we will focus on the search queries in this first group because we reliably know that these search queries are in GHDB.

3) Google dork distribution: According to Google Search Console, the 406 search queries we could link to Google dorks from GHDB are responsible for 8,736 clicks and 59,459 impressions out of 8,889 clicks and 62,054 impressions our honeypots received in total. Hence, the vast majority of clicks (98.3%) and impressions (95.8%) originates from Google dork usage, which proves the usefulness of our approach. The 406 search queries belong to 371 distinct Google dorks when disregarding different kinds of quotation marks, usage of capital and small letters and so on. Figure 4 shows a Cumulative Distribution Function (CDF) graph of the accumulated clicks (red) and impressions (blue) for one up to 371 Google dorks. The strong increase at the beginning for both clicks and impressions indicates that only a small number of Google dorks generates a majority of clicks and impressions. In particular, only ten Google dorks are responsible for about 50% of all clicks and 40% of all impressions. On the contrary, this means many Google dorks have only very few clicks and impressions, which leads to the assumption that many Google dorks are not used on a regular basis. In fact, more than 50% of the 371 Google dorks were used on less than 20 out of more than 300 days in our evaluation period. A reasonable assumption is that the more frequent Google dorks are used in more target-oriented campaigns to obtain interesting information. In the following, we will analyze the most prevalent Google dorks

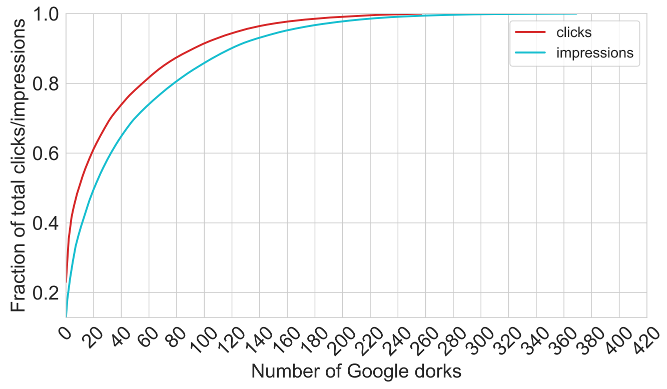

Fig. 4. Cumulative Distribution Function graph of the accumulated clicks and impressions for one up to 288 Google dorks

in more details. On the contrary, it is likely that many of the less frequently used Google dorks are used without or with at least less malicious intentions but rather to give them a try.

Table II shows information about the ten most prevalent Google dorks, responsible for 50% of all impressions. The *Dork-ID* references the ID in GHDB [10], and the *Category* refers to the category in GHDB. The most frequent keywords are *inurl* and *intitle* which are very effective at fingerprinting services because a big fraction contains unique strings in the URL or title. Five Google dorks were created in mid-2017, and two Google dorks were created in 2015 and one in 2016, respectively indicating that new Google dorks are used more often. Nevertheless, it is noteworthy that two very old Google dorks were among the most prevalent ones, released in 2004 and 2006, respectively. More than half of the Google dorks belong to the category *Various Online Devices* and search for example for surveillance cameras and routers. Two Google dorks belong to the category *Files Containing Passwords*, which is not unexpected because passwords are an easy to use resource without the necessity to take further actions. In the case of for example vulnerable systems an attacker needs to break into a system after finding it. One Google dork searches for databases of the well-known messenger WhatsApp (*Sensitive Directories*). Surprisingly, only one Google dork belongs to the three most prevalent categories in GHDB. Furthermore, the number of active days shows that the top ten Google dorks were used on average at least every third day and in case of the top two Google dorks almost every day. Hence, targeted companies or vendors of correspondent hard- and software should take further actions to secure their systems.

Overall, most of the top ten Google dorks were published within the last three years and search for IoT devices or interesting information like passwords or database backups. IoT botnets such as Mirai [1] gained a lot of attention recently. Therefore, it is not surprising when such Google dorks are among the most prevalent.

Next, we further analyze whether the category or publication date influences the likelihood of using a Google dork.

4) Publication years: At the beginning of our evaluation, we expected attackers to be less interested in old Google dorks and rather use new Google dorks. However, from the top ten Google dorks we already know that even Google dorks published in 2004 and 2006, respectively, were used very often. Therefore, we analyze in the following the correlation between publication year and used Google dorks as well as clicks and impressions per publication year. Figure 5 shows the number of Google dorks published by GHDB (red) and used during our evaluation period (blue) (top figure) along with the number of clicks (yellow) and impressions (green) (bottom figure) as a function of GHDB's publication year. Between 2007 and 2009, GHDB did not publish any Google dorks so that we omitted these years in the graphic. GHDB published the highest number of Google dorks between 2004 and 2006 and in 2010. Most likely, GHDB collected Google dorks between 2007 and 2009 as well and published them in 2010. Between 2011 and 2014, GHDB published less Google dorks. However, beginning in 2015, the number of published Google dorks increased again, indicating a growing interest in Google dorks.

For the analysis, we split the publication years into three periods: first, 2006 and before, second, between 2010 and 2014 and, third, between 2015 and 2017. Most used Google dorks were published in the third period and to a lesser extent during the first and second period. The Google dorks published in the second period between 2010 and 2014 were used less often during our evaluation period than the ones in the first period. Nevertheless, the used Google dorks from the first and second periods generated a considerable number of clicks and impressions. The Google dorks published between 2015 and 2017 were used most often when taking into consideration the clicks and impressions, which further supports the assumption of new Google dorks being more interesting because it is more likely that the targeted systems and information are still available.

In total, 126 Google dorks published between 2003 and 2006 were used during our evaluation (2003: 1, 2004: 67, 2005: 14, 2006: 18). Regarding the Google dorks from 2004, only three Google dorks searching for surveillance cameras were responsible for two-third of the clicks and impressions caused by all Google dorks published in 2004. The results for 2005 (50% of clicks/impressions caused by four Google dorks searching for surveillance cameras) and 2006 (75% of clicks and 50% of impressions caused by only one Google dork searching for surveillance cameras) further emphasize attackers' interests in such devices.

The 73 used Google dorks, which were published between 2010 and 2014, search for more diverse systems and information (2010: 28, 2011: 9, 2012: 12, 2013: 15, 2014: 9). Nevertheless, for each year there is at least one Google dork which gained more attention than others. Two Google dorks published in 2010 target the still popular forum software vBulletin and find usernames and passwords. In 2011, one Google dork searched for the web shell `k4rael`. Used Google dorks published in 2012 and 2013 gained most attention when they can find surveillance cameras, however, to a lesser extent than Google dorks published between 2010 and 2014. Additionally, one Google dork published in 2013 finds shared directories and still gets a lot of attention. Two Google dorks published in 2014, which aim at finding well-known Fritzbox routers and

TABLE II. Information about the ten most prevalent Google dorks, responsible for 50% of all clicks. The categories are: 1.) Various Online Devices, 2.) Files Containing Passwords, 3.) Advisories and Vulnerabilities, 4.) Sensitive Directories

Dork-ID	Dork	Creation date	Category	# impressions	# clicks	# active days
4068	intitle:"IPCam" inurl:monitor2.htm	2015-09-02	1	7,650	2,012	307
4548	inurl:login.cgi intitle:NETGEAR	2017-07-14	1	3,093	563	288
4498	"iSpy Keylogger" "Passwords Log" ext:txt	2017-05-29	2	1,814	520	139
1394	intitle:"BlueNet Video Viewer"	2006-06-25	1	1,771	273	155
4562	inurl:"/api/index.php" intitle:UniFi	2017-07-31	1	1,460	258	235
542	"Active Webcam Page" inurl:8080	2004-10-09	3	1,449	98	173
4073	intitle:"Index of" "WhatsApp Databases"	2015-09-07	4	1,324	83	95
4343	intitle:"open webif" "Linux set-top-box"	2016-10-24	1	1,253	109	225
4570	inurl:_vti_pvt/administrators.pwd	2017-08-03	2	933	136	128
4574	inurl:"img/main.cgi?next_file"	2017-07-31	1	874	138	127

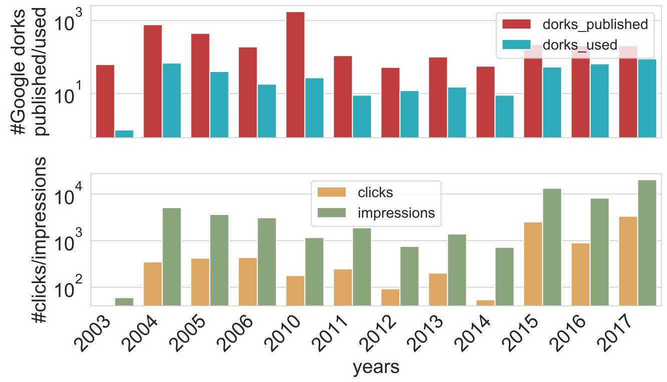

Fig. 5. Number of published and used Google dorks from GHDB with the number of clicks and impressions as a function of the publication years

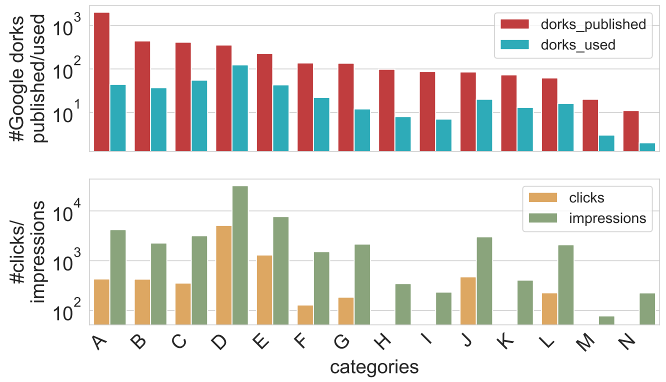

Fig. 6. Number of published and used Google dorks from GHDB with the number of clicks and impressions as a function of the category in GHDB. The categories are: A Advisories and Vulnerabilities, B Files Containing Juicy Info, C Pages Containing Login Portals, D Various Online Devices, E Files Containing Passwords, F Sensitive Directories, G Footholds, H Error Messages, I Web Server Detection, J Vulnerable Servers, K Network or Vulnerability Data, L Vulnerable Files, M Files Containing Usernames, N Sensitive Online Shopping Info

private information from shopping websites, received most of the clicks and impressions.

The most popular Google dorks published in phase three between 2015 and 2017 are at the same time the top-ten Google dorks and were already covered in the last subsection.

5) Categories: GHDB classifies Google dorks into fourteen different categories, such as *Files Containing Passwords* or *Various Online Devices*. In the following, we analyze whether Google dorks from certain categories are more interesting and, therefore, more often used. Figure 6 shows the number of Google dorks published by GHDB (red) and used during our evaluation period (blue) (top figure) along with the number of clicks (yellow) and impressions (green) (bottom figure) as a function of the category in GHDB. The category *Advisories and Vulnerabilities* contains by far the highest number of Google dorks. Even though Google dorks from this category lead to vulnerable systems, which is a primary goal of an attacker, the number of used Google dorks is rather small. An attacker has to take further actions and needs a lot of knowledge to exploit the found system, which might explain this difference. In contrast, the categories *Files Containing Juicy Info* and *Files Containing Passwords* have less published but more used Google dorks. Google dorks from both categories reveal interesting information which is easy to obtain and use, without the necessity of advanced knowledge. The category *Various Online Devices* contains a variety of devices, such as surveillance cameras, routers or printers. Taking the already mentioned growing interest in IoT devices into account, the high usage rate (both regarding the absolute number of used Google dorks and percentaged concerning the published Google dorks in this category) is expected. Notably, the category *Sensitive Online Shopping Info* has only two used Google dorks. However, these two Google dorks are responsible for a considerable number of impressions. The two Google dorks search for personal information and are, therefore, very interesting for attackers. Many categories have a considerable number of used Google dorks but a low number of clicks and impressions which further supports our assumption that many Google dorks are used without malicious intentions.

6) Correlation position and clicks: A high position in the search results is desirable because it usually increases the likelihood of people visiting a website. Google Search Console provides the average position in the search results of a website for each performed search query. Overall, our data set obtained from Google Search Console contains 29,355 entries. An entry consists of honeypot, day and search query along with the statistical information number of clicks and impressions as well as average position among the search results. Hence, the same search query can have multiple entries on different honeypots and days. Figure 7 shows standard deviation (red) and mean (blue) of the positions among the search results as a function of the number of clicks. The parentheses on the x-axis contain the number of entries for the particular number of clicks. Thus, the results are less meaningful for higher numbers of clicks because only very few entries are

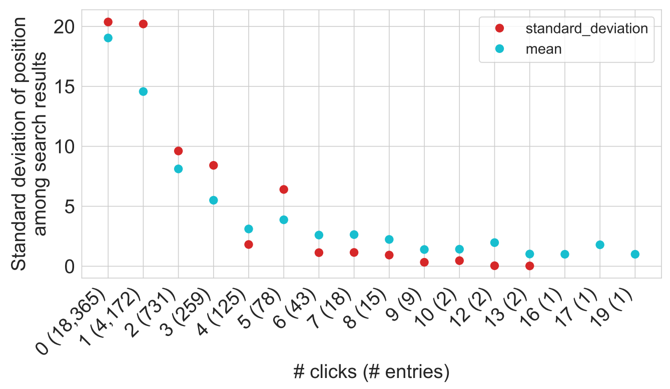

Fig. 7. Mean and standard deviation of positions among search results as a function of the number of clicks

available. Nevertheless, the graphic shows that entries with a higher number of clicks have a lower position among the search results which confirms the assumption that a better position among the search results increases the click likelihood.

V. Threats to Validity

Although we demonstrated that our system is providing useful insights into the usage of Google dorks in the wild, there are some threats to validity which we discuss in the following section. First, the Google search engine has to index our honeypages so that an attacker can find a particular honeypage when using the corresponding Google dork. According to Figure 3 in Section IV-C1, the number of indexed honeypages per Google dork varies, which we can not control. Furthermore, we detected a change in the number of indexed honeypages over time, i.e., honeypages were added or removed from the Google search index. We addressed this issue by using multiple honeypots and, thereby, increasing the likelihood for a honeypage being indexed at least once. Furthermore, we focused our evaluation on the overall obtained search queries, clicks and impressions instead of the results we got from single honeypots. Thereby, we can balance the differences in the number of indexed honeypages and still get meaningful results.

Additionally, as discussed previously, we can not link the Google Search Console information with the requests we received at the honeypots, limiting our results to the Google Search Console data. However, we argue that the focus of this study is the use of Google dorks in the wild, instead of what attackers intend to do with the collected information. Nevertheless, a connection of both data sources could reveal additional information and is interesting for future research.

Our insights are limited to the Google dorks in GHDB. We argue that it is not possible to gain any completeness because using the Google dork keywords, an attacker can generate arbitrary Google dorks. In contrast, we need the Google dork in advance to generate the corresponding honeypage. GHDB is by far the most comprehensive list of Google dorks publicly available so that the usage of this list provides a good overview of available Google dorks. Additionally, our infrastructure works with Google dorks from other data sources.

VI. Conclusion

In this paper, we presented DorkPot, a dynamic honeypot to collect requests originating from Google dorks. To the best of our knowledge, we are the first to analyze Google dork usage from a server perspective. We used Google dorks as input and generated honeypages tailored to be found via a Google search for each input Google dork. Finally, we bundled all generated honeypages into a low-interaction server honeypot.

In our evaluation, we analyzed more than 400 search queries, which we collected via Google Search console throughout more than ten months on ten deployed instances of DorkPot. We showed that 371 Google dorks were used to find our honeypots. In particular, attackers were interested in new Google dorks published between 2015 and 2017 and searched among others for IoT devices, such as surveillance cameras. Additionally, old Google dorks published between 2004 and 2006 were used to a lesser extent as well. Again, Google dorks revealing surveillance cameras were responsible for a majority of clicks and impressions. Our results show that website operator and vendors of Internet-accessible devices should pay attention to Google dorks as often ignored reconnaissance technique. To detect whether their systems are targeted or not, they can either monitor newly published Google dorks or analyze search queries used to find their websites.

VII. Acknowledgments

This project was funded by the Office of the Director of National Intelligence (ODNI) and the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C-0112. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, or the U.S. Government.

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[2] D. Canali and D. Balzarotti, "Behind the scenes of online attacks: an analysis of exploitation behaviors on the web," in *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS)*, 2013.

[3] O. Catakoglu, M. Balduzzi, and D. Balzarotti, "Automatic extraction of indicators of compromise for web applications," in *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[4] ——, "Attacks landscape in the dark side of the web," in *Proceedings of the Symposium on Applied Computing*, 2017.

[5] B. Cheswick, "An evening with berferd in which a cracker is lured, endured, and studied," in *In Proc. Winter USENIX Conference*, 1992, pp. 163–174.

[6] B. Dean, "Google's 200 ranking factors: The complete list," https://backlinko.com/google-ranking-factors, 2016, accessed: 2018-12-10.

[7] Docker, "Docker," https://www.docker.com/, 2017, accessed: 2018-12-10.

[8] Elastic, “Elastic stack,” https://www.elastic.co/products, 2017, accessed: 2018-12-10.

[9] Exploit-DB, “Google dork 4617 on google hacking database,” https://www.exploit-db.com/ghdb/4617/, 2017, accessed: 2018-12-10.

[10] ——, “google-hacking-database,” https://www.exploit-db.com/google-hacking-database/, 2017, accessed: 2018-12-10.

[11] Google, “Google search console,” https://www.google.com/webmasters/tools/home?hl=de, 2017, accessed: 2018-12-10.

[12] ——, “How google search works,” https://support.google.com/webmasters/answer /70897?hl=en, 2018, accessed: 2018-12-10.

[13] GoogleHackHoneypot, “Google hack honeypot,” http://ghh.sourceforge.net/, 2005, accessed: 2018-12-10.

[14] Hackertarget, “Google dorking wordpress,” https://hackertarget.com/google-dorking-wordpress/, 2017, accessed: 2018-12-10.

[15] Haxf4rall, “Google dorks list 2017 for sqli,” https://haxf4rall.com/2017/07/15/google-dorks-list-2017-for-sqli/, 2017, accessed: 2018-12-10.

[16] J. P. John, F. Yu, Y. Xie, M. Abadi, and A. Krishnamurthy, “Searching the searchers with searchaudit.” in *USENIX Security Symposium*, 2010.

[17] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, “Heat-seeking honeypots: Design and experience.” WWW 2011, 2011.

[18] J. Long, B. Gardner, and J. Brown, *Google hacking for penetration testers*. Syngress, 2011, vol. 2.

[19] J. Mueller, “Upcoming changes in google’s http referrer,” https://webmasters.googleblog.com/2012/03/upcoming-changes-in-googles-http.html, 2012, accessed: 2018-12-10.

[20] Nginx, “Nginx,” https://www.nginx.com/resources/wiki/, 2017, accessed: 2018-12-10.

[21] G. Pal, “Google dorks list 2017,” https://howtechhack.com/google-dorks-list-2017/, 2017, accessed: 2018-12-10.

[22] Pastebin, “Sqli dorks 2016-2017,” https://pastebin.com/7TQiMj3A, 2016, accessed: 2018-12-10.

[23] R. Pelizzi, T. Tran, and A. Saberi, “Large-scale, automatic xss detection using google dorks,” *Stony Brook University, Department of Computer Science. URL: https://www3.cs.stonybrook.edu/ rpelizzi/gdorktr.pdf (visited: 2018/03/26)*, 2011.

[24] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, 1st ed. Addison-Wesley Professional, 2007.

[25] Rapid7, “Nexpose vulnerability scanner,” https://www.rapid7.com/products/nexpose/, 2017, accessed: 2018-12-10.

[26] L. Rist, “Snare honeypot,” https://github.com/mushorg/snare, 2015, accessed: 2018-12-10.

[27] ——, “Glastopf,” https://github.com/mushorg/glastopf, 2017, accessed: 2018-12-10.

[28] Scribd, “List of google dorks for sql injection,” https://www.scribd.com/document/327673332/List-of-Google-Dorks-for-SQL-Injection, 2017, accessed: 2018-12-10.

[29] E. Shanab, “test,” https://twitter.com/alra3ees/status /988074729020928001?lang=en, 2018, accessed: 2018-12-10.

[30] L. Spitzner, *Honeypots: Tracking Hackers*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[31] C. Stoll, *The Cuckoo’s Egg: Tracking a Spy Through the Maze of Computer Espionage*. New York, NY, USA: Doubleday, 1989.

[32] C. Stoneff, “The seven steps of a successful cyber attack,” http://resources.infosecinstitute.com/the-seven-steps-of-a-successful-cyber-attack/, 2015, accessed: 2018-12-10.

[33] Tenable, “Nessus vulnerability scanner,” https://www.tenable.com/products/nessus-vulnerability-scanner, 2017, accessed: 2018-12-10.

[34] F. Toffalini, M. Abbà, D. Carra, and D. Balzarotti, “Google dorks: Analysis, creation, and new defenses,” in *Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016.

[35] J. Zhang, J. Notani, and G. Gu, “Characterizing google hacking: A first large-scale quantitative study,” in *International Conference on Security and Privacy in Communication Systems*, 2014.

[36] J. Zhang, Y. Xie, F. Yu, D. Soukal, and W. Lee, “Intention and origination: An inside look at large-scale bot queries.” in *NDSS*, 2013.