Characterizing Google Hacking: A First Large-Scale Quantitative Study

Jialong Zhang, Jayant Notani, and Guofei Gu

SUCCESS Lab, Texas A&M University
{jialong,guofei}@cse.tamu.edu,jayant.notani.93@gmail.com

Abstract. Google Hacking continues to be abused by attackers to find vulnerable websites on current Internet. Through searching specific terms of vulnerabilities in search engines, attackers can easily and automatically find a lot of vulnerable websites in a large scale. However, less work has been done to study the characteristics of vulnerabilities targeted by Google Hacking (e.g., what kind of vulnerabilities are typically targeted by Google Hacking? What kind of vulnerabilities usually have a large victim population? What is the impact of Google Hacking and how easy to defend against Google Hacking?).
In this paper, we conduct the first quantitative characterization study of Google Hacking. Starting from 997 Google Dorks used in Google Hacking, we collect a total of 305,485 potentially vulnerable websites, and 6,301 verified vulnerable websites. From these vulnerabilities and potentially vulnerable websites, we study the characteristics of vulnerabilities targeted by Google Hacking from different perspectives. We find that web-related CVE vulnerabilities may not fully reflect the tastes of Google Hacking. Our results show that only a few specially chosen vulnerabilities are exploited in Google Hacking. Specifically, attackers only target on certain categories of vulnerabilities and prefer vulnerabilities with high severity score but low attack complexity. Old vulnerabilities are also preferred in Google Hacking. To defend against the Google Hacking, simply modifying few keywords in web pages can defeat 65.5% of Google Hacking attacks.

1 Introduction

Web and web applications have become a necessary part of our daily lives. Every day, we interact with a large number of web applications for communication, education, and entertainment. Unfortunately, the diversity and complexity of web implementations make it hard for web developers to build bug-free web applications. Thus, these bugs/vulnerabilities give attackers a chance to compromise these benign websites. In [30, 22, 19], a large number of websites with high reputation were reported to have been exploited by attackers to redirect visitors to spam websites.

To effectively find those vulnerable websites, attackers began to explore search engines as their tools. Google Hacking refers to the practice of searching elaborate terms in search engines to find vulnerable websites. Based on a study

from [18], 33% of collected bot queries are searching for vulnerable websites. Another recent study [20] also showed that most of attackers submitted queries to search engines to look for vulnerable websites with known vulnerabilities.

There are several benefits for launching Google Hacking attacks:(1) Google Hacking can help attackers easily and efficiently find a large number of vulnerable websites with almost zero cost. (2) There exist many exploit toolkits in underground markets, which can automatically test and exploit those vulnerable websites. Thus, attackers can easily find and compromise those vulnerable websites in a large scale.

During the past 10 years, a large number of web vulnerabilities have been discovered, disclosed by researchers and software vendors, and are published on Common Vulnerabilities and Exposures database (CVE) [1]. This gives a chance for attackers to easily launch Google Hacking attacks. Attackers can easily choose their target vulnerabilities and generate corresponding search terms. Existing work has already conducted comprehensive studies either on a set of vulnerability databases in terms of the evolution of vulnerabilities, the life cycle of vulnerabilities, and the risk analysis of vulnerabilities [15, 28], or on the characteristics of specific type of vulnerable websites such as search poisoning attacks [30], HTTP parameter pollution [23]. However, not all of those vulnerabilities can be exploited in Google Hacking, thus the characteristics of vulnerabilities targeted in Google Hacking attacks are unfortunately still not clear to us.

In this paper, we conduct a first quantitative study on the Google Hacking attacks. Starting from a set of representative Google Dorks (search terms that can be used to easily find out websites with corresponding vulnerabilities) used in Google Hacking, we study the characteristics of Google Hacking targeted vulnerabilities through analyzing relationship among vulnerabilities targeted by Google Hacking, known web related vulnerabilities, potentially vulnerable websites (websites that have installed with vulnerable web applications), and victims (vulnerable websites that have been reported to be compromised).

We collect a large number of representative Google Dorks used in Google Hacking from a largest online public google hacking database [6], and a large number of known vulnerabilities from a public vulnerability database CVE [1]. We collect a total of 2,101 Google Dorks used for Google Hacking, 997 of them can be automatically matched with vulnerabilities in the CVE database. We further search those Google Dorks in Google and collect 305,485 potentially vulnerable websites. To evaluate the quality of these potentially vulnerable websites, we also collect 21,386 websites that have been successfully compromised through cross-site scripting attacks in the past from an online public XSS attack database [13]. We then cross check these XSS victim websites that also appear in our collected potentially vulnerable websites, which we term as victim-vulnerable websites. We find 6,301 websites belong to victim-vulnerable websites.

Then we study the characteristics of Google Hacking from four perspectives: targeted vulnerability, vulnerability-victim relationship, attack impact, and attack robustness.

- For the targeted vulnerability study, we study the difference between vulnerabilities in Google Hacking and all known web related vulnerabilities in the CVE database, we find that the distribution of vulnerability categories are quite different between the web related CVE vulnerabilities and targeted vulnerabilities in Google Hacking. Further study shows that vulnerabilities targeting on SQL injection attacks and the vulnerabilities with high severity and low attack complexity are frequently exploited in Google Hacking. Interestingly we also find that most of relatively old vulnerabilities are also frequently exploited in Google Hacking. In addition, although Google Hacking does target on some certain popular web applications, it also exploits the vulnerability from a variety of web applications, even for the applications that only have one vulnerability in CVE database.
- For the vulnerability-victim relationship study, we investigate the key factor to the different populations of vulnerable websites. We find vulnerable application itself could be a key factor to different population of vulnerable websites.
- For attack impact study, we investigate the impact of Google Hacking by evaluating the quality and popularity of victims of Google Hacking attacks. Our results show that both high-reputation and low-reputation websites could be victims of Google Hacking. For example, 87.6% of them have page rank higher than 3 [1]. 14 of them are in top 1,000 Alexa ranks. This again indicates that Google Hacking can be a good way to find high quality vulnerable websites.
- For the attack robustness study, we check the robustness of Google Hacking attacks. We design a new metric to evaluate the robustness of Google Hacking. Our results show that 65.5% of Google Hacking can be easily defeated by simply modifying few keywords of web pages.

2 Background

2.1 Google Dork

As we know, search engines are designed for efficiently finding information on Internet. Usually, users simply input search terms (keywords) and search engines will return relevant websites that contain corresponding information. However, search engines also support some special operators for relatively complex searching, such as *inurl, intitle, and intext*. Search queries with these special operators are called Google Dorks. With the help of Google Dorks, users can easily and quickly find more accurate search results.

In recent years, Google Dorks have also been abused by attackers to launch Google Hacking [20]. For example, *inurl:"search_results.php?browse=1"* is a Google Dork that can reveal websites with the SoftBiz Dating Script SQL Injection vulnerability, a vulnerability that allows remote attackers to execute SQL commands. Figure 1 shows some google search results of such Google Dork. In this paper, we also use such Google Dorks as input to find vulnerable websites targeted by Google Hacking.

[1] 3 is the average PageRank score based on [12]

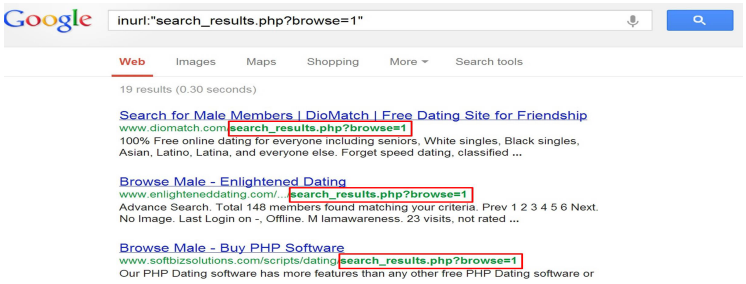


Fig. 1. Google Dork search results

2.2 Web Vulnerability

As more and more applications now can be interacted through web interface, such as online banking, online shopping, and online social networking, remote attacks on web applications are on the rise due to the large profits and scalability. Thus, web related vulnerabilities attract much more attention from attackers than traditional local exploit software vulnerabilities do. To find all known web vulnerabilities in the CVE database, we first extract vulnerabilities with "network" as access vector, which are considered to support remote exploit. Among all remote exploit vulnerabilities, we further extract web related vulnerabilities by checking keywords in their descriptions. Google Hacking usually targets on the following certain types of web vulnerabilities. All these four categories represent more than 90% of targeted vulnerabilities of Google Hacking in our database.

- **SQL Injection** [10] is done by injecting strings into database queries to change the database content or dump the database information such as passwords.
- **Cross-site scripting(XSS)** [4] is done by injecting JavaScript into web applications to bypass access controls such as the same origin policy.
- **Remote Execution** [9] allows attackers to run arbitrary code in target servers to execute their own commands.
- **Path Traversal** [5] allows attackers to access files that are not intended to be accessible.

Fig. 2. Google Hacking

2.3 Google Hacking

Vulnerability databases and existing studies have already published the details of known vulnerabilities and corresponding exploit methods. However, one important question for attackers is how to automatically find vulnerable websites with those vulnerabilities in a large scale. Google Hacking is one way to exploit search engines to find vulnerable websites. Figure 2 is a general Google Hacking

procedure. In this attack, attackers first need to choose their target vulnerabilities and generate corresponding Google Dorks as shown in ①. Then they can collect potentially vulnerable websites by directly searching Google Dorks in search engines ②. In this case, since not all of search results are actual vulnerable websites, attackers need to further scan and exploit those potentially vulnerable websites ③. They can use the exploit methods provided from the vulnerabilities databases or exploit tools from underground markets to automatically exploit those vulnerable websites. Since not all of vulnerable websites can be successfully exploited due to patching or personalized configuration, only the websites that can be successfully exploited become victims, which can be further abused by attackers to host spam or steal sensitive information.

In this paper, we conduct a comprehensive study of characteristics of Google Hacking from the following 4 perspectives. (i)Targeted vulnerability (labeled ①), e.g., what kind of vulnerabilities are typically targeted by Google Hacking? (ii)Vulnerability-victim relationship (labelled ②), e.g., what kind of vulnerabilities usually have a large population? (iii)Attack impact (labeled ③), e.g., what is the impact of Google Hacking? (iv)Attack robustness, e.g., how easily to protect vulnerable websites from being searched out through Google Hacking?

3 Data Collection

In this section, we describe the data sources that we used for our research.

3.1 Vulnerabilities

Common Vulnerability and Exposures Database (CVE) is an online public vulnerability database, which represents currently publicly known information of security vulnerabilities. To gain the knowledge of currently known web vulnerabilities, we first crawled all CVE vulnerabilities from National Vulnerability Database [8], which contains 53,611 CVE vulnerability entries reported from 1999 to 2012. For these CVE vulnerabilities, we crawled their CVE entry IDs and associated information such as CVSS scores, vulnerability summaries, and vendors. We further extracted web related vulnerabilities based on the method mentioned in Section 2. In this way, we collect a total of 26,453 such vulnerabilities. We denote this dataset as **Web_CVE** in this paper.

3.2 Google Dorks

Google Hacking Database [6] is the largest and most representative online public exploit database as we know, which contains Google Dorks relating to known vulnerabilities and threats. These Google Dorks can be used for Google Hacking to search out vulnerable websites that have corresponding vulnerabilities. Since we try to study Google Dorks that can be used to find vulnerable websites rather than collect some sensitive information such as password files, we only crawl the Google Dorks in "Vulnerable Files(60 Google Dorks)", "Vulnerable Servers(71 Google Dorks)", and "Advisories and Vulnerabilities(1,970 Google Dorks)" directories, which are usually related to certain vulnerabilities. In this way, we collect a total of 2,101 Google Dorks with associated information such as the hit number, submit time, and description.

To further understand how these Google Dorks are used to exploit vulnerabilities, we automatically match Google Hacking database with CVE database based on their descriptions. Among these 2,101 Google Dorks, 997 of them have CVE entries in their descriptions, thus we can automatically match them to CVE database, and term this dataset as **Dork_CVE**.

3.3 Potentially Vulnerable Websites

To collect vulnerable websites, we searched all the Google Dorks in Google and recorded all the search results as "potentially vulnerable websites". These potentially vulnerable websites can be more exactly described as the ones that match the conditions of specified vulnerabilities (e.g., specific version of specific installed web applications/scripts). However, at the time of our searching, some of these websites may have already been patched, cleaned, or security enhanced, thus no longer exploitable. Thus, it is true that not all of the potentially vulnerable websites we found are actual vulnerable.

3.4 Victim Websites

XSSed Database [13] is an online public XSS attack database, which contains websites that have been *actually* exploited through cross-site scripting attacks in the past. In this database, attackers have injected malicious JavaScript on at least one page of each domain. We collect a total of 21,368 unique victim domains and used these victim domains to evaluate the quality of Google Hacking. We assume that the websites on these domains did not change significantly from where they were XSSed and the time when they were found in the potentially vulnerable websites. Thus, the websites appeared in the intersection of XSSed database and our potentially vulnerable websites should be victims of Google Hacking. We cross check these victim websites with potentially vulnerable domains, which we term as victim-vulnerable websites. 6,301 websites belong to victim-vulnerable websites.

Table 1 is a short summary of our collected data.

Table 1. Data summary

Google Dork	Dork_CVE	Web_CVE	Potentially Vulnerable Webs	Victim Webs	Victim-Vulnerable Webs
2,101	997	26,453	305,485	21,368	6,301

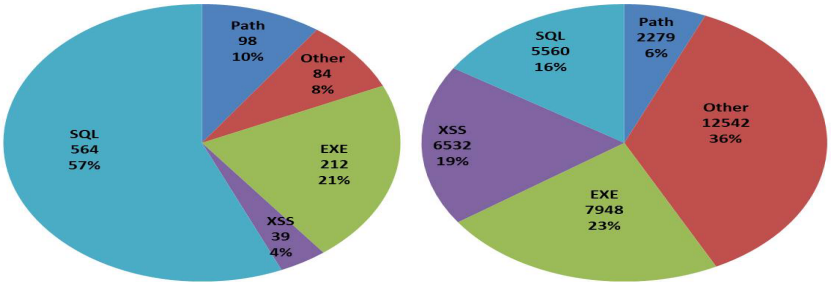
4 Measurement Methodology and Results

In this section, we study the characteristics of Google Hacking from different perspectives.

4.1 Targeted Vulnerability Study

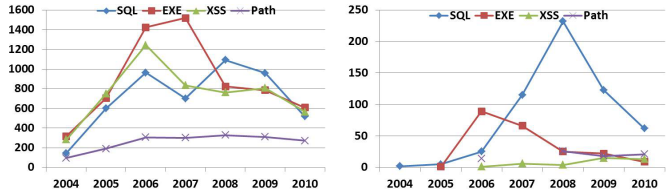
As we know, most of Google Dorks used in Google Hacking are generated based on vulnerabilities. However, not all of web vulnerabilities can be represented in the form of Google Dorks, and not all of such vulnerabilities are interested to attackers. In this part, we try to study what kind of vulnerabilities are typically targeted by Google Hacking through examining the following characteristics of vulnerabilities.

Attack Categories. To verify if Google Hacking targets on some specific attack categories, we compare the categories of vulnerabilities targeted by Dork_CVE with categories of all web related vulnerabilities in Web_CVE database. We categorize each type of vulnerability by examining the keywords in their descriptions. Figure 3 shows the category distribution for vulnerabilities in Dork_CVE and Web_CVE.



(a) Google Hacking (b) Web_CVE

Fig. 3. Vulnerabilities category distribution



(a) Web_CVE (b) Google Hacking

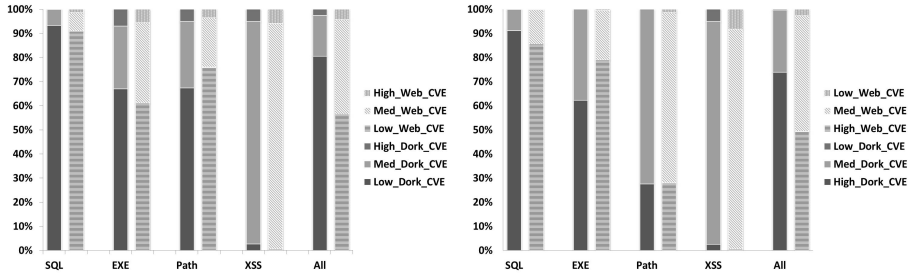
Fig. 4. Vulnerabilities category trends

We can see that the categories of vulnerabilities targeted by Google Hacking are very different with that of web related vulnerabilities in Web_CVE. Specifically, SQL, EXE, XSS, Path account for 92% Google Hacking targeted vulnerabilities while they only contribute 64% in Web_CVE. In addition, SQL injection vulnerability is exploited by most Google Hacking (57% in Dork_CVE) but only 12% in Web_CVE, which reflects that most of Google Hacking will lead to SQL injection attacks. From this perspective, only studying vulnerabilities in Web_CVE can not truly reflect attackers' interests. We further compare the trends of vulnerability category in both Google Hacking and Web_CVE database. Figure 4 are the trend distribution for vulnerabilities in Dork_CVE and Web_CVE. We can see that by the end of 2010, EXE and XSS vulnerability became top vulnerability in Web_CVE as shown in Figure 4(a). However, for Google Hacking, SQL is still the top one vulnerability as shown in Figure 4(b). In addition, although the number of XSS vulnerabilities begun to decrease since 2008 in Web_CVE, it started increasing in Google Hacking.

To further understand why these vulnerabilities are chosen to be targeted in Google Hacking, we examine them in terms of the exploit complexity, potential damage, and the age of these vulnerabilities targeted in Dork_CVE. Intuitively the vulnerabilities reported recently with high damage and low attack complexities should be good candidates for Google Hacking. We also examine the vendors

of these vulnerabilities to verify if Google Hacking only targets on vulnerabilities of certain web applications.

Attack Complexity. Ideally attackers prefer vulnerabilities that can be easily exploited so that they can launch attacks automatically in a large scale. To study how easily these vulnerabilities can be exploited, we check the complexity of exploiting these vulnerabilities. We use the feature "Access Complexity" provided in CVSS[2] to evaluate attack complexity. High access complexity means that attackers need specialized access conditions to launch attacks while low access complexity means that it is relatively easy to launch attacks. Figure 5(a) shows the access complexity distribution.



(a)Attack complexity distribution (b) Attack severity distribution

Fig. 5. Attack category distribution

We can see that most of vulnerabilities (e.g., SQL, EXE, Path) targeted by Google Hacking have relatively low access complexities, which means that attackers can easily launch attacks automatically at scale when they collect vulnerable websites. For each category, since the category itself already has low access complexity (e.g., SQL injection vulnerability is easy to attack), the percentage of vulnerabilities with low access complexity in Dork_CVE is similar to vulnerabilities in Web_CVE. However, in total, about 80% of the vulnerabilities of Dork_CVE have low access complexity while only about 55% of the vulnerabilities in Web_CVE have. In addition, attack complexity distribution is similar to the attack category distribution for Dork_CVE in Figure 3, which also reflects that complexity is a candidate consideration for Google Hacking attacks.

Attack Damage. Ideally attackers prefer vulnerabilities that have huge damage such as getting the full privilege of a vulnerable website. To study the damage of these vulnerabilities, we check the attack severity of these vulnerabilities. We use the feature "CVSS Severity Score" provided in CVE database to evaluate the damage. Figure 5 (b) shows the attack severity distribution.

We can see that most of vulnerabilities targeted by Google Hacking have high severity levels, which may cause serious damage if these vulnerabilities are exploited successfully. In total, about 74% vulnerabilities in Dork_CVE have high severity level while only 47% vulnerabilities in Web_CVE have. In addition, the attack severity distribution is also similar to attack category distribution for Dork_CVE in Figure 3. Thus, attack damage is also a good candidate consideration for Google Hacking attacks.

We further cross check the attack damage and attack complexity of vulnerabilities, only 2 vulnerabilities (cve-2006-3571 and cve-2010-0971) in Dork_CVE out of 815 such vulnerabilities in Web_CVE have low attack damage with high attack complexity. We then check the details of these two vulnerabilities, both of them belong to XSS vulnerability and allow remote attackers to inject arbitrary web scripts, which are essentially severe vulnerabilities .

Vulnerability Age. Older vulnerabilities usually have more mature attack tools, which can be easily exploited. However, newer vulnerabilities may not be widely patched so that they may have a large victim population. To further check whether Google Hacking targets on old vulnerabilities or recent vulnerabilities, we use the metric "Age", the time difference between the report time of the vulnerabilities and the submission time of the Google Dorks, to evaluate it. A vulnerability with a large age means that it is a relatively old vulnerability. Figure 6 shows the age distribution of vulnerabilities in Dork_CVE.

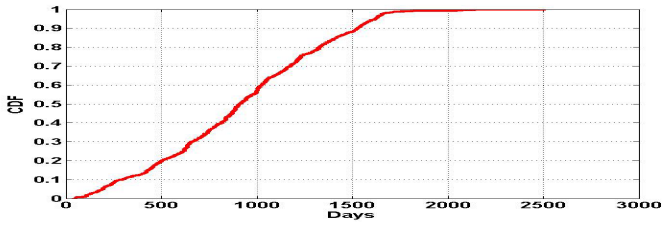


Fig. 6. Age distribution

We can see that most of these Google Dorks target on older vulnerabilities, only 10% Google Dorks target on vulnerabilities exposed in the same year. It is probably because that the techniques exploiting older vulnerabilities are more mature and most users do not patch their servers on time. Thus they are still lucrative for criminals [3]. We acknowledge that our results may have some bias since the submission time of those Google Dorks may not accurately characterize the attack time. However, the submission time somehow reflects the observation of such attacks, which can be used to estimate the trend of attackers' tastes.

Table 2. Variety of vendors and applications

Rank	Vendor		Application	
	Dork_CVE	Web_CVE	Dork_CVE	Web_CVE
1	joomla (65)	joomla (226)	joomla (9)	wordpress (110)
2	mambo (20)	novell (196)	cms_made_simple (5)	moodle (105)
3	xoops (12)	wordpress (154)	mambo (4)	php-nuke (102)
4	yourfreeworld (10)	drupal (141)	kwsphp (4)	phpmyadmin (98)
5	wordpress (8)	apache (123)	adodb_lite (3)	weblogic_server (97)

Application. Intuitively, famous web applications usually have a large number of customers, which could be a good target for Google Hacking. To verify if these Google Dorks are created to target on some specific famous applications/vendors, we check the variety of applications of these vulnerabilities. There are totally 899 web applications affected by these 997 Dork_CVE vulnerabilities, which shows that Google Hacking could target on a variety of web applications, not limit to certain applications.

Table 2 shows the top 5 vendors/applications for both Dork_CVE and Web_CVE vulnerabilities, the numbers in the bracket shows the number of vulnerabilities. For example, there are 65 dorks in Dork_CVE targeting on vulnerability of joomla while there are 226 vulnerabilities related to joomla in Web_CVE. Although the application distribution is not strongly consistent between Dork_CVE and Web_CVE, we can see that Joomla [2] and WordPress appear top in both Dork_CVE and Web_CVE. From [18], Joomla and WordPress are two popular applications that are frequently queried by bots through Google Hacking.

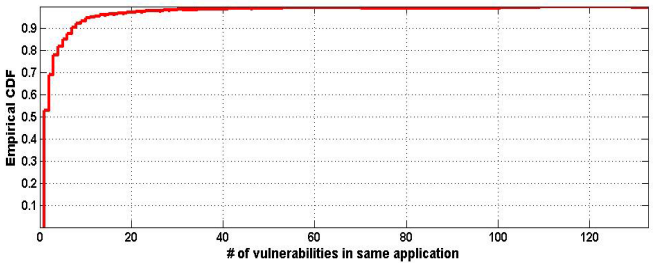


Fig. 7. Distribution of the number of vulnerabilities for web applications

To further check whether it is because WordPress and Joomla have many vulnerabilities that lead to be exploited by Google Hacking, we extract all web applications targeted by Google Hacking and check the number of vulnerabilities in Web_CVE for the same application. The high number of vulnerabilities in Web_CVE means that these applications are much more vulnerable and have a higher chance to be exploited. Figure 7 shows the vulnerability number distribution of these web applications. Interestingly, we find that more than 50% web applications targeted by Google Hacking have only one vulnerability in Web_CVE, which means that the choice of Google Hacking targeted applications is not strongly correlated with the numbers of vulnerabilities for this application.

Lessons: Most Google Hacking attacks target on certain categories of vulnerability (e.g., SQL, XSS, EXE, Path), which usually have high attack damage with low attack complexity. Thus, launching Google Hacking on them makes it easy for attackers to compromise vulnerable websites. In addition, most of Google Hacking attacks target on relatively older vulnerabilities, probably because exploitation techniques are more mature. Furthermore, both the trend of vulnerability category and application distribution of vulnerabilities are quite different between Dork_CVE and Web_CVE, and the target applications of Google Hacking are not strongly consistent with their vulnerabilities number. Thus, only studying the characteristics of Web_CVE vulnerabilities may not fully represent the taste of Google Hacking.

[2] Joomla is an open source content management system which is estimated to be the second most used CMS on the Internet after WordPress.

4.2 Vulnerability-Victim Relationship Study

Through searching Dork_CVE in Google, we collect a large number of potentially vulnerable websites. With a large number of potentially vulnerable websites, we further investigate the relationship between vulnerabilities and potentially vulnerable websites. As we know, the goal of attackers is trying to find a large number of possible vulnerable websites through Google Hacking. So what is the possible cause for a large population of vulnerable websites? To answer this question, we try to study what kind of characteristics of vulnerabilities may lead to a large population.

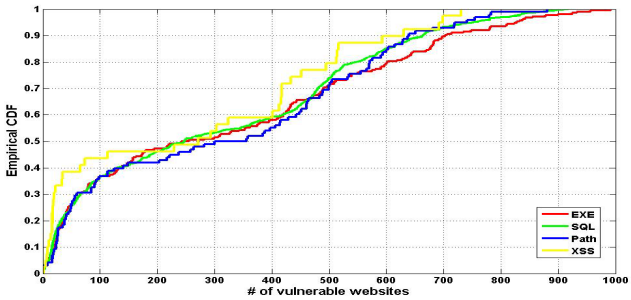


Fig. 8. Potentially vulnerable websites distribution in different vulnerability categories

Attack category. Intuitively, different attacks targeting on different vulnerabilities are likely to have different numbers of potentially vulnerable websites. To verify if the attack category may lead to different numbers of potentially vulnerable websites, we compare the distribution of the number of potentially vulnerable websites among different vulnerability categories. Figure 8 shows the distribution results.

Table 3. Top 10 vulnerabilities with large number of potentially vulnerable websites

CVE	Category	# of potentially vulnerable websites
2007-6649	EXE	991
2007-6139	EXE	956
2008-0502	EXE	932
2007-0233	Other	930
2007-0232	EXE	924
2008-5489	SQL	917
2007-1776	SQL	909
2007-6057	EXE	899
2007-5992	SQL	898
2009-0451	SQL	894

We can see that all of the four attacks have very similar distribution although they have quite different vulnerability numbers. We further run T-test [11] to determine if these distributions are significantly different from each other. T-test is a statistical hypothesis test that can be used to determine if two sets of data are significantly different from each other. In our experiment, we chose statistical

significance as 0.05, thus, if the calculated p-value is below 0.05, the null hypothesis is rejected and the two distribution are significantly different. T-test for all pairs of attacks are higher than 0.05, which further demonstrates the four attacks have very similar distribution. We then check the category of vulnerabilities with the highest number of potentially vulnerable websites. Table 3 is the Top 10 vulnerabilities with a large number of potentially vulnerable websites. We can see that vulnerabilities in "EXE" category have the highest number of potentially vulnerable websites. However, it still has a similar population distribution with vulnerabilities in other categories. Thus, the population of potentially vulnerable websites does not have a strong correlation with vulnerability categories.

Application. Intuitively, popular/famous applications should have a large population. To verify that whether it is the vulnerable applications that lead to different numbers of potentially vulnerable websites or not, we compare the average number of potentially vulnerable websites among different vendors.[3]

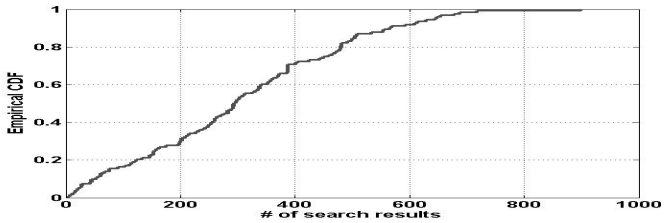


Fig. 9. Potentially vulnerable websites distribution with vendors

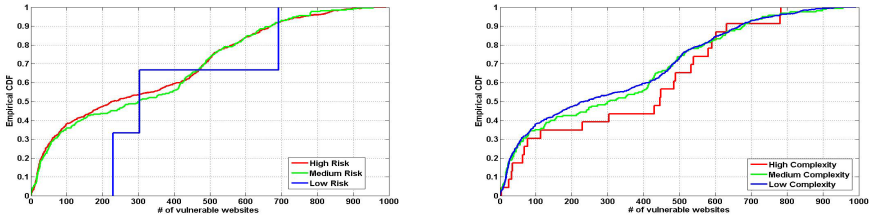
Table 4. Top 5 vendors of potentially vulnerable websites

Vendors	Avg # of potentially vulnerable websites
social_networking_script	898.5
skadate_online_dating	718.5
frontaccounting	687
minitwitter	677
minerva	654

Figure 9 is the cumulative distribution of the average number of potentially vulnerable websites for different applications. We can see that the overall distribution is almost linear. Less than 20% vulnerabilities have the number of potentially vulnerable websites larger than 600. Table 4 shows the top 5 vendors with the largest average number of potentially vulnerable websites. Interestingly, the top vulnerable applications are different in that of Dork_CVE and Web_CVE shown in Table 2 . However, they are all popular web applications or applications containing sensitive information. Social_networking_script is a datecomm social network web application, which allows remote attackers to execute arbitrary SQL commands. FrontAcoutning is a web-based accounting system that also allows remote attackers to execute arbitrary SQL commands, which will lead to sensitive information exposure. Thus, the popularity of these applications could be a key cause to the size of potentially vulnerable websites population.

[3] We ignore vendors with only 1 vulnerability, because the number of potentially vulnerable websites of them could be easily oscillated and might not be reliable.

Attack severity. To verify whether different risk levels of vulnerabilities will lead to different numbers of potentially vulnerable websites, we compare the distribution of the number of potentially vulnerable websites among vulnerabilities with different severity levels. Figure 10 (a) shows the cumulative distribution of the population of potentially vulnerable websites for vulnerabilities with different risk levels. Since we only have few low-risk vulnerabilities, its distribution is not continuous. However, both high-risk and medium-risk vulnerabilities have very similar distributions. Thus, attack severity maybe not be a cause for large population of potentially vulnerable websites.



(a) different risk levels (b) different complexity levels

Fig. 10. Potentially vulnerable websites distribution

Attack complexity. To verify whether the attack complexity will lead to different numbers of potentially vulnerable websites, we compare the distribution of the number of potentially vulnerable websites among vulnerabilities with different complexities.

Figure 10 (b) shows the cumulative distribution of vulnerabilities with different attack complexities. Although we only have few vulnerabilities with low attack complexities, their distribution is still very similar to other vulnerabilities with high or medium attack complexities. Thus, attack complexity may not contribute a lot to the population of potentially vulnerable websites.

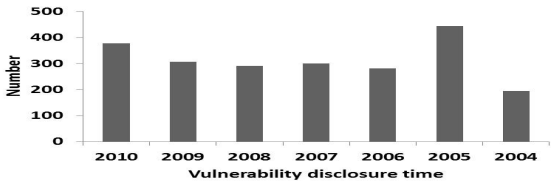


Fig. 11. Potentially vulnerable websites distribution with exposure time

Exposure time. To verify if the exposure time of vulnerabilities will lead to different numbers of potentially vulnerable websites, we compare the distribution among vulnerabilities with different exposure time. Figure 11 shows the distribution of the average number of potentially vulnerable websites in different exposure time. We can see that the number of potentially vulnerable websites does not decrease much along with time, this is possible because people are usually lazy to patch their systems [3]. The exception of 2005 is because there are only few vulnerabilities disclosed in 2005, which makes the average number of potentially vulnerable websites not reliable. Thus, the exposure time seems not to be a good indicator of large potentially vulnerable website population.

Lessons: Although most Google Hacking attacks target on SQL vulnerability, Google Hacking targeting on EXE vulnerability usually has a large number of population. And vulnerable applications could be a key factor accounting for the different population of vulnerable websites.

4.3 Attack Impact Study

To measure the impacts of Google Hacking, we essentially check the quality and popularity of those victim-vulnerable websites.

Fig. 12. Pagerank distribution

Quality. Since the final goal of attackers is trying to compromise benign websites through Google Hacking, thus the higher quality websites have, the more value attackers can gain (e.g., website reputation, sensitive information, and a large number of visitors). PageRank score is widely used by search engines to rank the importance of websites. A higher PageRank score indicates a better reputation of the website. To evaluate the overall quality of these victim-vulnerable websites, we use PageRank score as an indicator of the website quality. Figure 12 shows the PageRank score distribution. We also compare it with randomly chosen 1,000 domains from the XSSed database (Victim websites).

Table 5. Top 5 Top Level Domains of vulnerable and victim websites

Vulnerable Websites	Percentage	Victim Websites	Percentage
com	53.91%	com	44.12%
org	8.68%	org	6.00%
net	6.49%	net	5.18%
de	4.15%	de	3.36%
uk	2.29%	uk	3.17 %

Table 6. Top 5 country of vulnerable and victim websites

Vulnerable Websites	Percentage	Victim Websites	Percentage
United States	61.09%	United States	46.17%
Germany	8.43%	Germany	6.78%
United Kingdom	3.25%	France	5.36%
France	3.09%	United Kingdom	5.19%
Netherlands	2.92 %	Turkey	3.59 %

We can see that victim-vulnerable websites have relatively high reputations compared with victim websites. 87.6% of victim-vulnerable websites have page rank scores higher than 3 while only 68.5% for victim websites. We also cross check vulnerable and victim-vulnerable websites with Alexa ranks. 14 of them belong to top 1,000 Alexa ranks. which also reflects that Google Dorks could be a good way to find high quality vulnerable websites.

Popularity. Those vulnerable and victim-vulnerable websites are widely distributed over 367 Top Level Domains (TLD) in total. Table 5 presents Top 5 TLDs. We can see that more than half of them hosted in .com domain, which is also the largest domain [7] on current Internet.

We further check the country code of these websites based on their IP addresses. The vulnerable websites and victim-vulnerable websites are distributed over 153 countries. From Table 6, we can see that about 60% of them are located in United States, this is possible because our query location is in United States, thus more local websites are likely returned by search engines.

Lessons: The vulnerable websites of Google Hacking attacks are widely distributed on current Internet in terms of their popularity and quality, which makes Google Hacking attacks become a popular way to find vulnerable websites.

4.4 Attack Robustness Study

To defend against Google Hacking attacks, the best way is to patch/fix all the vulnerabilities, which are usually expensive and impractical. One alternative way is to prevent attackers from finding those vulnerable websites. To achieve this goal, we study the structure of Google Dorks. There are totally 6 operators abused by Google Hacking attacks as shown in Table 7. We define three robust levels of those dorks based on the cost for defenders to modify their websites' content to defeat Google Hacking attacks. For example, *intext*/*doublequote* operators try to find keywords in webpages. In this case, administrators can easily replace these keywords in the content with synonyms or images to avoid being searched out. Thus, *intext* and *doublequote* operators will have lowest robustness. *intitle*/*allintitle* operators try to find keywords in the title of webpages. Although it is easy to replace these keywords in titles, however, these titles usually reflect the function of these pages which is important for normal operation usage. Thus, they will have medium robustness. *inurl*/*allinurl* operators try to find certain files/scripts in the web server. These files are usually associated with other files. Directly modifying these files may lead to dependent errors of other files. Thus *inurl*/*allinurl* will have highest robustness.

Table 7. Google Dork Structure

Operator	# of dorks	robust level
double quote	610	low
intext	43	low
intitle/allintitle	22	medium
inurl/allinurl	322	High

We also noticed that some dorks may use multiple operators. Thus, their robustness should be the minimal level among all operators because modifying the keywords with minimal robustness level is enough to protect the web server from being searched out. In this case, 65.5% of dorks have low robustness and can be easily defeated by careful website administrators. For example, google dork "Powered by NovaBoard v1.1.2" tries to find websites with vulnerable application NovaBoard installed. In this case, administrator can easily remove such

content in the webpage or replace such information with a picture, which can successfully evade such attack without leading to any malfunctions.

Lessons: Although Google Hacking is efficient for attackers to find high quality vulnerable websites, simply modifying the web content of a server can help administrators defeat more than half (65.5%) of Google Hacking attacks.

5 Related Work

In this section, we discuss related research from three perspectives.

Large scale vulnerability analysis. Vulnerabilities have been widely studied by [15, 28] in terms of vulnerability evolution, life cycle, vulnerability category, vulnerability priority analysis, etc. Frei *et al.* [15] presented a comprehensive study on the life cycle of general vulnerabilities in terms of the discovery, disclosure, exploit and patch time of vulnerabilities on more than 14,000 vulnerabilities. Their results show that acquiring exploits is always faster than getting patches. Shahzad *et al.* [28] extended this work by considering vendors and types of vulnerabilities. Their results supported the previous study and presented interesting trends on vulnerability patching and exploitation. Scholte *et al.* [29] performed an empirical analysis of a large number of web related vulnerabilities. Their results show that the complexity of XSS and SQL injection exploits has not been increasing, and many web problems are still simple in nature. Edwards *et al.* [14] conducted a study on the vulnerabilities history of various popular open source software using a static source code analyzer and the entry rate in CVE database. They demonstrated a correlation between the change in the number and density of issues and the change in the rate of the discovery of exploitable bugs for new releases. An analysis of CVSS score has also been conducted by Scarfone *et al.* [26], while Fruhwirth *et.al* [16] and Gallonc [27] attempted to prioritize the vulnerabilities based on the CVSS framework.

Most of these studies only focus on vulnerabilities themselves. However, the characteristics of these vulnerabilities themselves can not *fully* represent the interests of attackers'. Thus, through studying the Google Hacking, our work complements existing research by understanding the connections among the vulnerabilities with Google Dorks, vulnerable websites, and victim websites.

Studies using Google Dorks. Moore *et al.* [24] showed that at least 18% of website compromises are triggered by Google dorks. John *et al.* [20] found that some bots explored Google Dorks to find target websites and built an automated detection tool by generating regular expressions for query dorks. Their results show that at least 12% of search results are vulnerable to SQL injection attacks. Later, John *et al.* [21] further exploited those malicious query dorks to find vulnerable websites and built honeypots of these vulnerable web pages to collect attack patterns. In [25], Pelizzi used Google Dorks from online hacking database to find seed vulnerable websites and then automatically generate Google Dorks from these vulnerable websites. Recently, Invernizzi *et al.* [17] used Google Dorks to locate more malicious websites by starting from an evil seed set.

Different from existing work using Google Dorks to find more malicious websites, we start form a new angle by studying what kind of vulnerabilities are usually exploited as Google Dorks and the quality of these Google Dorks.

Large-scale victim websites analysis. Research [19, 22] conducted a study on search poisoning attacks in terms of detection and measurement. They collected a large number of victim websites compromised by attackers to either redirect user traffic to some malicious websites or host spam directly. Then they also presented basic measurement of these victim websites. Zhang *et al.* [30] further extended their work to automatically find more victim websites, and conducted a comprehensive measurement of these victim websites in terms of distribution and quality. Balduzzi *et al.* [23] presented an automated approach to discover HTTP parameter pollution vulnerabilities. With their proposed method, they conducted a large-scale analysis on more than 5,000 popular websites and showed that 30% of them have vulnerable parameters and 14% of them suffer from HTTP parameter pollution attacks. Unlike these existing work, we focus on the relationship between victim websites and vulnerable websites rather than victim websites themselves, and we target on more generic web attacks.

6 Conclusion

In this paper, we have conducted the first quantitative study of Google Hacking. Through analyzing the relationship among vulnerabilities targeted by Google Hacking, the general web exploit vulnerabilities in Web_CVE, potentially vulnerable websites, and victim websites, we conclude that Google Hacking only targets on a few specially chosen vulnerabilities. Thus existing studies on generic vulnerabilities in Web_CVE may not truly reflect the tastes of Google Hacking.

To defend against Google Hacking attacks, we investigate the robustness of Google Hacking. Our study shows that most Google Hacking can be easily defeated through modifying a few web content without leading to any malfunctions.

In our future work, we will perform a deeper study with more data, and prioritize web vulnerabilities based on the attackers' tastes.

7 Acknowledgments

This material is based upon work supported in part by the National Science Foundation (NSF) under Grant No. CNS-1314823. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

References

1. The common vulnerabilities and exposures dictionary. `http://cve.mitre.org/`.
2. A complete guide to the common vulnerability scoring system version 2.0. `http://www.first.org/cvss/cvss-guide.html`.
3. Crims prefer old exploits: Microsoft. `http://www.theregister.co.uk/2011/10/11/zero_day_overrated_says_ms/`.
4. Cross-site scripting. `http://en.wikipedia.org/wiki/Cross-site_scripting`.
5. Directory traversal attack. `http://en.wikipedia.org/wiki/Directory_traversal_attack`.
6. Exploit database. `http://www.exploit-db.com/google-dorks/`.

7. Host distribution by top-level domain. http://userpage.fu-berlin.de/~mr94/dns/node8.html.
8. Nvd,"national vulnerbality database". http://nvd.nist.gov/.
9. Reomote code execution. http://en.wikipedia.org/wiki/Arbitrary_code_execution.
10. Sql injection. http://en.wikipedia.org/wiki/SQL_injection.
11. T-test. http://en.wikipedia.org/wiki/Student's_t-test.
12. What does your google pagerank mean. http://www.redfusionmedia.com/google_pagerank.htm.
13. Xss attacks information and archive. http://www.xssed.com/archive.
14. N. Edwards and L. Chen. An historical examination of open source releases and their vulnerabilities. In *Proceedings of the 2012 ACM conference on CCS*, 2012.
15. S. Frei, M. May, U. Fiedler, and B. Plattner. Large-Scale Vulnerability Analysis. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, 2006.
16. C. Fruhwirth and T. Mannisto. Improving CVSS-based vulnerability prioritization and response with context information. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009.
17. L. Invernizzi, P. Comparetti, Stefano Benvenuti, C. Kruegel, M. Cova, and G. Vigna. EVILSEED:A Guided Approach to Finding Malicious Web Pages. In *IEEE Symposium on Security and Privacy (Oakland)*, 2009.
18. F. Yu D. Soukal J. Zhang, Y. Xie and W. Lee. Intention and Origination: An Inside Look at Large-Scale Bot Queries. In *Proceedings of the 20th NDSS*, 2013.
19. J. John, F. Yu, Y. Xie, M. Abadi, and A. Krishnamurthy. deSEO: Combating search-result poisoning. In *In Proceedings of the 20th USENIX Security*, 2011.
20. J. P. John, F. Yu, Y. Xie, M. Abadi, and A. Krishnamurthy. Searching the searchers with searchaudit. In *Proceedings of the 19th USENIX conference on Security*, 2010.
21. J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. Heat-seeking honeypots: design and experience. In *Proceedings of the 20th WWW*, 2011.
22. N. Leontiadis, T. Moore, and N. Christin. Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade. In *In Proceedings of the 20th USENIX Security*, 2011.
23. D. Balzarotti M. Balduzzi, C. Gimenez and E. Kirda. Automated discovery of parameter pollution vulnerabilities in web applications. In *Proceedings of the NDSS*, 2011.
24. T. Moore and R. Clayton. Evil Searching: Compromise and Recompromise of Internet Hosts for Phishing. In *Proceedings of the 13th International Conference on Financial Cryptography and Data Security, Barbados*, 2009.
25. R. Pelizzi, T. Tran, and A. Saberi. Large-scale, automatic xss detection using google dorks. 2011.
26. K. Scarfone and P. Mell. An analysis of cvss version 2 vulnerability scoring. In *Proc. of ESEM'09, pages 516-525*, 2009.
27. K. Scarfone and P. Mell. Vulnerability discrimination using cvss framework. In *Proc. of NTMS'11, pages 1-6*, 2011.
28. M. Shahzad, M. Z. Shafiq, and A. X. Liu. A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles. In *34th International Conference on Software Engineering (ICSE)*, 2012.
29. D. Balzarotti T. Scholte and E. Kirda. Quo Vadis? A Study of the Evolution of Input Validation Vulnerabilities in Web Applications. In *Proceedings of the 15th RAID*, 2012.

30. J. Zhang, C. Yang, Z. Xu, and G. Gu. PoisonAmplifier: A Guided Approach of Discovering Compromised Websites through Reversing Search Poisoning Attacks. In *Proceedings of the 15th RAID*, 2012.