WEBSITE PENETRATION VIA Google SEARCH

Azam Zia Muhammad Ayaz
Email: azazi022@student.liu.se, muhay664@student.liu.se
Supervisor: Juha Takkinen, juhta@ida.liu.se
Project Report for Information Security Course (TDDD17)
Linköpings universitet, Sweden

Abstract

By using a search engine such as Google, a security breach in a website can be located. There are two types of vulnerabilities, software vulnerabilities and misconfigurations. Majority of the hackers try to exploit already known misconfigurations and locate a system that has a similar security flaw. The aim of this report is to examine such vulnerabilities by practically trying to penetrate a website and then suggest solutions to tackle them. We also implemented those solutions and tested them again to find how effective they are. We found that the implemented solutions were very effective and achieved success in preventing unauthorized access of information.

1. Introduction

This course project is about penetration testing of a website. Penetration testing or ethical hacking is a method to discover how vulnerable a website is.[1] We have selected to use the Google search engine as a means to penetrate a website.

1.1 Problem and Motivation

There is an ever increasing need of information security these days because the internet is a major source of information transfer. The core issue of our project is that we should suggest methods to prevent information leakage in a website. We are focusing on just one aspect of information leakage which is unauthorized access of user information. This is information that later can be used to blackmail the user this information belongs to. We chose this project because information security is a basic need of every organization.

We are using penetration testing because most of the hackers try to penetrate a system using commonly known misconfigurations in the website. [16] We are using Google as a means to penetrate because it is the most popular search engine. It also provides mechanisms to easily search for unsecured files using this search engine.[16]

1.2 Objectives of project

Our primary objective for doing this project is to learn about penetration testing and understand the significance of web security. The purpose of this project is to analyse a website's vulnerabilities related to hacking via Google. Also we will suggest solutions against some of these vulnerabilities. Because the vulnerabilities are of many types e.g. unauthorized access of data, denial of service, execution of script etc. We focused only on unauthorized access of data

because Google hacking is mainly used to exploit this kind of vulnerability.

1.3 Method

The main method we used for this project is to experiment. We chose this method because in this way we hoped to be able to have a closer look at the subject. We expected it to give us practical experience with how to perform penetration testing and also how to secure information.

We developed a website to perform these penetration tests and then analyzed the results to find security susceptibilities. We focused to a few of these vulnerabilities to find their solutions and implemented them on the demo site. After implementation we reran the tests and finally compared the two results to find the effectiveness of our solutions.

1.4 Report structure

We start with describing the background of the project in chapter 2, including why we need website security and also its implementation issues. The background also contains a description of the workings of the Google search engine and query building techniques.

Third chapter is the analysis where we explain the methodology we adopted and how we performed the experiment. Chapter 4 contains the results from the project. Chapter 5 contains our suggestions and recommendations. Lastly we list the conclusions and problems faced during the project.

2. Background

Penetration testing or ethical hacking is used for performing a security check on a website. [1] Its aim is to try and find as many security loopholes as possible. In our project we are using Google search as a tool for penetration.

Date revised: 28th, April 2009.

2.1 Why we need security

In today's world that is also known as a global village, internet security is becoming a major issue, especially when software development is fast, internet connections are becoming inexpensive, and the internet is constantly changing. Website security is now a basic requirement because data is transferred from one point to another using internet as a channel. Data may pass through several points, it may give opportunity to capture or even alter it.[6] According to the National Opinion Poll in UK, as of January 2007 *"With the ever growing use of home computers, the spread of broadband, and the rise in internet banking and commerce the importance of proper internet security measures has never been greater."*[7]

2.2 Issues related to website security

When a computer connects to the internet and starts communicating it is at risk. The website security involves the protection of website files and database, protecting information from an intruder. Website security is mainly related to securing user information, illegal access of files, illegal use of services, copying, modification, and locking of the data. [8][9] Some issues related to website security are:

1. If a website uses a third party plug-in then it cannot be assured that it will not establish direct independent connection from a server to remote user. [10]

2. Many websites do not keep record of access logs. They do not keep record of connections between the accessed address and the website itself. [10]

3. ActiveX technology is used to sign plug-ins that installs additional software while accessing a website.[17] These are seemingly safe and all authorized Active X applications are marked with a certified number.[10] These applications have more or less limitless access to the file system. They can access names and email

addresses and can access the file system to create, modify or delete files. An Active X component that causes damage to one's website can only be tracked through the certification number. [10]

4. Most browsers have java script enabled by default and this can be used to reveal data to perform vicious acts to one's website.[10]

2.3 The Google search engine

Every internet user is familiar with the accuracy and speed of Google search results. Pigeon Rank is one of the main component of Google search.[11] This system is used to rank the pages and it was developed by founders of Google, Larry Page and Sergey Brin at Stanford university. [11] Although Google engineers are working on every aspect of Google services for better improvement on daily basis but Pigeon Rank is the basis for all web search tools. [11]

As shown in figure 2.1 Google has a complex architecture for crawling the web pages and caching them via several web crawlers. A Universal Resource Locater (URL) server sends a list of URLs to the crawlers to fetch. The fetched pages are then send to storeserver which compresses and stores them into a repository. All web pages have a unique ID which is assigned when a new URL is parsed out of a web page. Indexer performs indexing and sorter performs sorting of these stored web pages. Indexer uncompresses documents while reading from repository and parses them. Each webpage (document) is converted into a set of words that is called hits. A hit record contains information about the position, font and capitalization of words.[12] Indexer distributes these hits to "barrels". This creates a partially sorted forward index. Indexer also parses for all links and stores them in an anchors file. This file contains information about all links, where each link points to and from. It also contains display text of links.
The URL resolver reads this anchors file and converts the links into absolute URLs. It puts anchor text into forward index that is associated with DocID in anchors file. It generates a database of these links that is used to compute page ranks by using PigeonRank.
Sorter takes the data from barrels sorted on DocID and generates inverted index. WordIDs and their offsets are also produced by sorter. DumpLexicon takes this list with the lexicon produced by the indexer and generates new lexicon for searcher. Searcher is a web server that uses this lexicon with inverted index and page ranks to answer search queries. [12]

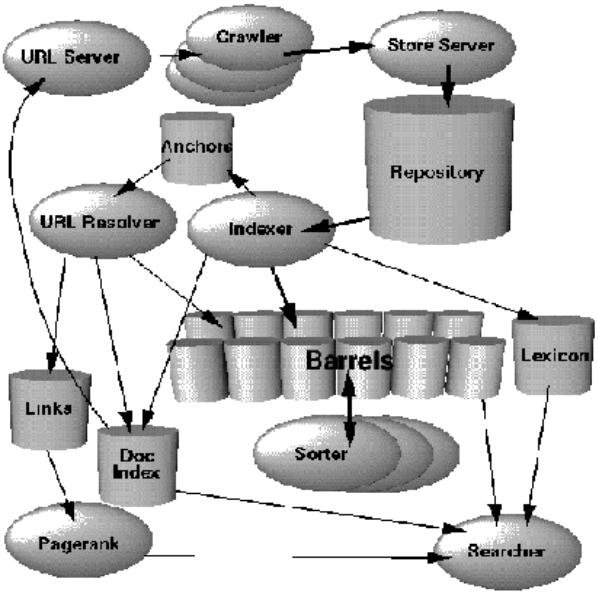

Figure 2.1 High Level Google Architecture [12]

2.4 Query building

Google provides a number of operators known as advanced operators [1] to build advance search queries. These advance operators can be used to refine search results. Google advanced operators are as follows: [13]

intitle, allintitle, inurl, allinurl, filetype, allintext, site, link, inanchor, daterange, cache, info, related, phonebook, rphonebook, bphonebook, author, group, msgid, insubject, stocks, define.

We will take a look at some of these operators but first we discuss the syntax for these operators.[13]

Advanced operators can be part of search queries. These can be passed to Google just like other search terms; however advanced operators have their own writing style and rigid syntax that must be followed. The basic syntax is *operator:search_term* and the following should be kept in mind while using these operators. [13]

1. There is no space between operator, colon and search term.
2. The search term can be provided in a phrase or a single word.
3. Boolean operators (OR, AND) can also be applied.
4. More than one operator can be combined in a single query.
5. The operator that begins with the word "all" generally cannot be mixed with other operators and can be used once per query.

Here are some examples of Google search operators.

intitle, allintitle

Title is the text that is written on top of most browsers while viewing a page. A title can be static or dynamic (can be generated by the web browser) or in some cases a page might not even have a title. While using *intitle* the word or phrase following this is considered as the search term while *allintitle* tells Google that every word or phrase that is to be found in a title. [18] For example *"intitle:All your FX needs filled here"* will return search results for all those pages having the text "*All your FX needs filled here"* in their title. [13]

inurl, allinurl

The beginning of a URL consists of the communication protocol followed by "://", which is followed by the path to the page, with directories separated by a forward slash (/). A simple URL example is as follows:

"http://digitalalgo.com/Contact.aspx".

inurl is one of more tricky operator for advanced users. It looks for a word or a phrase in the URL of a webpage like for instance, "*inurl:digitalalgo.com/Contact.aspx*". This query returns the search results with a complete address for that given page name. This operator searches for followed search terms as a complete url. *allinurl* searches for followed terms anywhere in url like *"allintitle"*. *allinurl* do not play very well like *allintitle*. If you need to find many words then it's better to put several *inurl* queries together. [13]

allintext

allintext is the simplest operator and which search engines are most known for which is to locate a text string within the body of the page. It is useful when we want to find something in the text of page. *"allintext"* can found string anywhere in the page except title, url and internal/external links. Due to its functionality *allintext* should not be mixed with other advanced Google operators. For example *intext:"Advertisement on TruthFX"*. This query returns search results with all pages having text *"Advertisement on TruthFX"*. [13]

site:

By using *site* operator a domain name can be best searched. [18] This search can only retrieve those pages that are hosted on a specific domain. Syntax for this operator is fairly straight forward i.e. *site:domain_name.* For instance, consider a domain name, such as *www.digitalalgo.com.* To locate all pages hosted in this domain the query will be *site:digitalalgo.com.* If we look at the results they will show all the pages hosted by *digitalalgo.com* that exist in the Google cache. This operator can show unpredictable results if top level domain name is not provided in search terms.[18] *site* operator can be used with other advanced operators in a single query.

2.5 Summary

To summarise the discussion in chapter 2, it was necessary to emphasize why web security is so important and issues that may arise. Also we gave

reader a background about Google search engine and page ranking because normally the hacker would look through only first few pages of his query result. The techniques used to build simple/advanced search queries are given because they are the core of our experiment.

3. Analysis

3.1 Specifications of our demo site

As we could not find an existing suitable site where we could freely perform our experiment, we first developed a little website where we had administrative rights. The name of this new site was *www.digitalalgo.com*. This site was developed with Microsoft .net technologies using ASP.net in C# and based on Framework 2.0. [21] This is basically an online music shopping center where registered users can upload and download audios/videos. Our website was also made accessible by the Google search engine. Here are some details of the website.

1. The home page is designed to show the main contents of the website and its purpose. This gives the complete introduction about the website.
2. The "about us" page displays some text about the authors of the website. It will show the experience and education information of the designers.
3. The "contact us" page displays all contact information of the website authors like their email address, phone numbers and postal address.
4. The advertisement page displays some advertisements on our website. We show Google ads on this page. [19] These ads are about different kinds of albums, music in different languages and of different genres.
5. The link page shows external links to our site digitalalgo.com. These links are to other popular online music stores with some of their details.

3.2 Experimental set-up

To find the vulnerabilities in a website by using Google search engine, we performed a penetration test on our newly created website. In the first step we created an xml file *sitemap.xml* that we submitted to Google bots. Google bots are special kind of agents who are responsible for crawling websites. When these bots crawled our website they took those pages also that we do not want to add in Google cache because those pages contains some sensitive information and only registered users can view them. We analyze this by using advance operators in our search queries. After our analysis we implemented a few solutions to this problem. We also did a retest to verify our solutions.

3.3 Test process

We tried to penetrate the site and access user information by using different query strings for Google search. We here describe the test cases that we developed from these query strings.

Test case 1:- site:digitalalgo.com.

This test case was built to access all cached pages of *digitalalgo.com*. It contains a Google advanced operator and search string.
After successful execution of this test case we found that our search result contains all pages that are cached by Google. Search results also contain those pages that are not allowed to view without registration. Google bots added all pages in Google cache.

Test case 2:- inurl:digitalalgo.com

This test case contains *inurl* operator which is used to search given string in URL addresses of web pages of provided website.

This search returns all those pages that have this word digitalalgo.com in their address link. This operator and it's another version *allinurl* returns

the same results. It shows nothing when we use it with *intext* operator. i.e.

allinurl:"digitalalgo"+*intext*:"Advertisement on TruthFX".

It show a Google generated message when we use this with *allintext* operator that "Your query looks similar to automated requests from a computer virus". The query was

inurl:"digitalalgo"+*allintext*:"Advertisement on TruthFX".

Test case 3:- intitle:"All your FX needs filled here"+site:digitalalgo.com.

This test case contains two operators each followed by a search string. First is *intitle* which is used to search in title of web pages and the second is *site* operator which is used to search all cached pages.

Although *site* operator returns all cached pages but it shows less results when it is used with other operators like *intitile* even though all pages have the same title. Also this shows wrong snippet in search results. The text shows in snippet is of different page and given link is of different page. They should be the same.

Test case 4:- intitle:"All your FX needs filled here!"+site:digitalalgo.com+*intext*:"Advertisement on TruthFX"

This test case consists on three Google operators. Purpose behind this query was to search for all those pages that are cached in Google cache, having a specific text in title and body text.

site operator alone shows all the cached pages and all pages have the same title. When these two operators are used in a single query then search results shows very less results. And when we used these operators with *intext* operator then Google shows wrong results in return.

In some of our test cases we got unauthorized access to user information we then examined the web page responsible for that data and tried to analyse why authentication was bypassed.

3.4 Retest

After our analysis we thought about the solutions to the security flaws we found. We implemented those solutions and tested again. We disabled Google caching for a few pages by using robots.txt file.[20] This file is used to restrict Google bots not to cache all pages of this domain. We mention those pages in robots.txt file who has sensitive user information or required user registration. This file is usually placed in root directory of website otherwise Google bots do not access it.

4. Results

In our first experiment the Google search results also contained those pages which should only be shown to user after successful login. We implemented robots.txt file and added meta tags on pages having sensitive information in order to disable Google caching. We also used Google removal tool to immediately remove sensitive pages from Google cache. In our retest we found that those pages were removed from Google search results. This time there was much less access to confidential information.

5. Discussion

Today (2009) it is very important for a website administrator to have full control over the security. Google can pick all sensitive information from a site or domain which was not hosted or administered carefully. If the administrator does a little study on Google webmaster tools, [14] Pigeon Rank and the Google bots crawling mechanism, then unwanted access to website information via Google can be stopped completely. The administrator can also use the knowledge gained to improve the ranking of the administered site and get more hits.

To build advanced Google search queries by using advanced operators requires knowledge. Chapter

two of the book written by Johnny Long "Google Hacking for Penetration Testers" [18] has a good explanation of search operators. Without administrative rights on a website good penetration testing cannot be performed because to play with Google bots different kinds of files are used e.g. some html files to verify the website, xml files about the pages of website and text files to allow/block Google bots to cache the website.

5.1 Suggested solutions

While design a website the following points should be kept in consideration:

- The administrator should know which pages should be given for crawling to Google bots and which pages should be kept secret.
- We can add a text file named robots.txt in the public folder. All the links mentioned in this file will not be crawled or cached by Google. We can also list which bots should not crawl our website.
- We can add a meta tag on each page to stop Google bots from caching the page.
- We can also remove cached copies, snippets, images etc. by using robots meta tags. [15] Robots meta tags are usually written in html of page by programmers or developers to control access of search engine bots.

5.2 Implemented solutions

We changed the code of our website to stop Google caching. We also constructed a robots.txt file to make sure that only those pages are crawled by Google agents which we want to have indexed by Google. We also uploaded a sitemap which helped the crawlers to find all relevant pages. Sitemap.xml file usually contains all the links to website, last modification date, priority and location of web pages under root directory. An example of a meta tag is as follows.
<meta name="robots" content="noarchive">.
This tag instructs Google bots not to crawl this page.

6. Conclusions

Our focus was to find vulnerabilities in a website. We started our experiment by executing test cases which consists of Google search queries. These queries were built by using advanced Google operators which helps to search for given string in Google cache. From our experiment we can conclude that Google hacking can be prevented if the website developers and administrators (they may the same) know a little about how Google search works and how they can stop Google from caching sensitive data. Also the website administrators should know how to limit Google bots to crawl on only those pages where the administrator wants them to.
We have only checked the information that is available on web pages. However, there should be done penetration testing on databases as well.

7. Problems faced

We contacted many people who are administering websites but none of them allowed us to perform our experiment on their website. As a result we had to spend a few days in just developing a website.

Google bots take a lot of time approximately 15 days, to crawl through a website. Also the crawling rate depends on the website ranking and our newly uploaded website did not have a good rank.

While we were performing penetration test, for some queries we got an error from Google saying that this seems to be a computer generated query. This is because Google has also implemented few solutions to stop Google hacking. This issue can be resolved by breaking the query into parts.

We first registered our site with a company which had implemented policies to stop Google hacking. So we had to shift our site to another provider.

References

[1] Johnny Long CS, (2005) *Google hacking for penetration testers*, http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-long.pdf, 19th April 2009.

[2] *Google Hacking*, http://en.wikipedia.org/wiki/Google_Hacking, 2 March 2009

[3] Acunetix,(2008) *Web application security*, http://www.acunetix.com/websitesecurity/google-hacking.htm, 25 March 2009

[4] *"Google search API 2.0",* http://googlesystem.blogspot.com/2006/06/Google-search-api-20.html, April 1, 2009

[5] *"Google AJAX search API"*, http://code.google.com/apis/ajaxsearch/, 25 March 2009

[6] Guardian Digital, Inc. (2000-2009), *Secure community,* http://infocenter.guardiandigital.com/manuals/SecureCommunity/node9.html, 2002-12-16.

[7] Lilian Edwards (2007), *The internet and security* http://www.law.ed.ac.uk/ahrc/SCRIPT-ed/vol4-1/editorial.asp, April 10, 2009.

[8] Wikipedia, *Internet security,* http://en.wikipedia.org/wiki/Internet_security, April 17, 2009.

[9] Federation Council, (1996) *Criminal code of the Russian Federation*, Group-IB 2008, 18 April 2009.

[10] William Stewart, (1996-2007), *Anonymizer Limitations*, http://www.livinginternet.com/i/is_anon_limits.htm, 25th March 2009.

[11]PigeonRank (2009), http://www.google.com/technology/pigeonrank.html, 18 April 2009.

[12]Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, 18th April 2009.

[13]Johnny Long CS, Google hacking for penetration testers, 18th April 2009.

[14]Google webmaster tools, https://www.google.com/webmasters/tools, 17th April 2009.

[15]Google webmaster tools, http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=35301, April 20, 2009.

[16]Stuart, Joel, George, (2005) *Hacking Exposed fifth edition,* McGraw-Hill Osborne Media, May 27 2009.

[17]Microsoft Corporation, 2009, ActiveX controls, Microsoft, http://msdn.microsoft.com/en-us/library/aa751968.aspx, 27 May 2009.

[18]Johny Long, 2005, Google hacking for penetration testers, Syngress Media, 27 May 2009.

[19]Google advertisements, 2009, http://www.google.com/intl/en/ads/, 27 May 2009.

[20] Katherine Nolan, 2009, "Creating and Using a robots.txt File", http://www.inkkdesign.com, 27th may 2009.

[21] Microsoft Corporation, *Official site of ASP.NET*, http://www.asp.net/, 27th may 2009.