Whitepaper – SNS08-01

Paranormal Fallacy, SE Automated Scanning Anomaly

Inadvertent issue in Google Dorking

Date: 6 July 2008

Aditya K Sood aka 0kn0ck , http://www.secniche.org | adi_ks [at] secniche.org

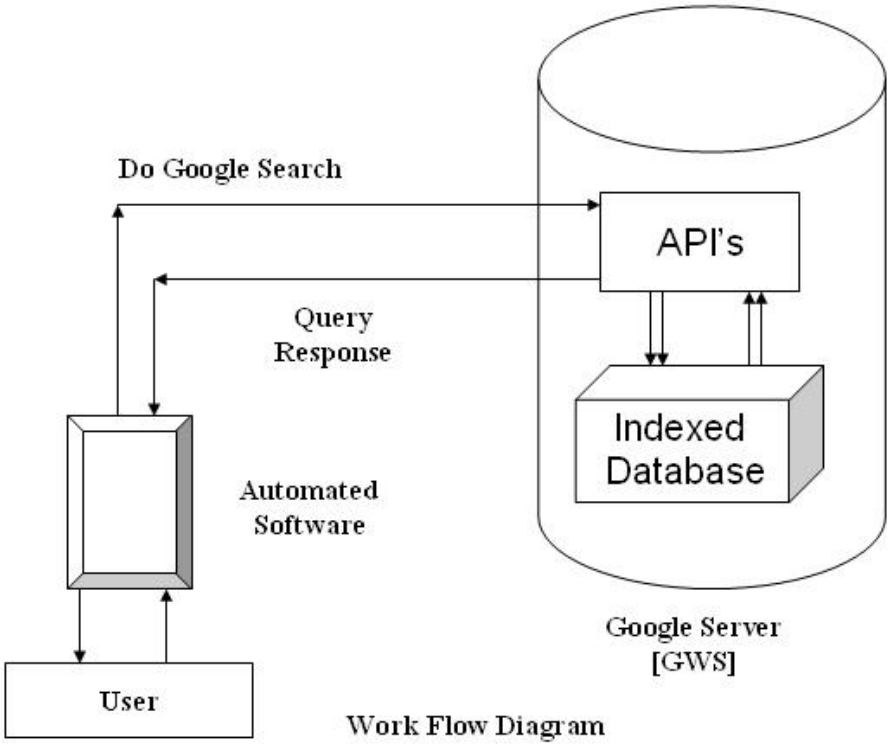
[1] Abstract:

This paper will discuss the anomaly behavior of Google search engine that affects the working of automated scanning tools. This anomaly can be considered as a security mechanism implemented by Google to prevent number of search queries to be executed by a single host within a specific time limit. Due to this factor the scanning functionality of number of tools is disrupted. The generic reason is Google considered those queries to be rogue or worm driven. In order to circumvent this CAPTCHA is implemented. The issue is very critical from view point of automated scanning. This not only disrupts the scanning procedure but also lowers the functionality of Google search engine to be used in an automated manner by third party software's. The vulnerability scanning through Google dorks become hard. The point of discussion is to understand the root cause and find a solution to implement automation functionality in a right context. This process helps in vulnerability scanning through Google search by automated queries.

[2] Automated Google Search Paradigm:

The web is not a static environment. There are lots of changes taking place in day to day life. The Google tries to manage this extensibility pattern of web but it can not be controlled fully. Due to this factor the website information changes time to time. That's why it has been noticed that search at one time produces different results and so on. This is one of the critical factors in search engine optimization too. The web is getting divergent day by day.

The Google search engine has been used for various automation purposes. As we all know Google functionality is based on collecting number of resources from websites and rearranging them in a perfect index order. Google has highly effective algorithm based programs that traverse through all of the databases taking into account the index number. The Google has largest website database generated through Google bots and various crawlers. These are considered as backend intelligence programs that collect information form specific website without any hindrance. On the contrary search engine optimization process is undertaken in which search should be performed in a desired time period. This is possible only by designing efficient algorithms that perform the work faster. The automation of search can be used for relative purposes. Let's look into the working process:



Work Flow Diagram

The basic structure of working is presented above. A user has a direct interface with automated software. User can directly issue a specific query which is to be executed through Google. The automated software injects that query as a parameter in the calling function and request is sent to the server with API call. The crawlers traverse through whole of the database to match the desired pattern. The suitable matches are arranged in an order to be displayed. The response is sent back to the software which renders the output in a proper way to be presented to user. The received data can be raw or XML based. It depends on the calling functions and the method used. Like RSS feeds use XML for data gathering and transmission.

This is a general working flow of automated software's that use Google search engine as a scanner. The automated search is one of the highly acclaimed mechanisms of using search engine for vulnerability scanning of web applications. The number of dorks are designed in such a way to automate the scanning through Google.

[3] GHDB and Google Dorks:

Google search power is used for gathering information encompassing vulnerability checks. Lot of web vulnerabilities can be checked by effective search. The Google Hacking Database is a generic example of this. This database consist of specific search signatures as strings that should be searched through Google engine to find vulnerability behavior in number of websites. This is properly implemented through Google Keywords [intext ,inurl ,allinurl, site , etc].The GHDB comprise of Google keywords with signature checks. This database is arranged in a hierarchical order and properly classified. The objects used in GHDB are termed as Google Dorks. Basically it is kind of standard specific database that is used by number of automated software's. Two types of functional check:-

1. The coder can use the static database as such by calling every single dork from GHDB.
2. Other procedure is to design the Google dork itself and add it in a requisite database.

The automated software for a single dork work efficiently. In this when ever a single dork is constructed and executed a browser is launched directly. The browser is directed with the Google search URL to display the result. This type of automation work properly. One can design its own dork for a specific vulnerability or information gathering.

For Example:- Scroogle tool. This is a very general phenomenon of Google functioning. On the side of using Google as a vulnerability scanner number of dorks are used simultaneously to automate the process. The Dorks are classified as:

Dorks can be classified as:

- Web Server Detection.
- Vulnerable Servers.
- Vulnerable Files on Remote Targets.
- Extracting Online Shopping Information.
- Remote Sensitive Directories.
- Online Devices Information.
- Web Pages Containing Login Portals Data.
- Web Pages Containing Network Vulnerability Data.
- Files Containing Username and Passwords.
- Error Messages from Remote Targets.
- Vulnerabilities and Advisories.
- Checking Footholds.

Example:-

- inurl:sysinfo.cgi ext:cgi
- inurl:technote inurl:main.cgi*filename=*
- inurl:tmssql.php ext:php mssql pear adodb -cvs -akbk
- inurl:ttt-webmaster.php
- inurl:wiki/MediaWiki
  inurl:wp-login.php +Register Username Password "remember me" -echo -trac -footwear

The information flow is very fast and it is very crucial to control that flow. The automated scanning through Google faces a problem now days that disrupt the behavior of software to high extent.

[4] The Automation Metrics and Google Behavior:

It is necessary to understand the query automation procedure in the form of various metrics. The metrics define the automated scanning procedure through Google search and the way Google responds. We are specifically relates to automated search issues. Google shows a versatile response in many different situations. But our talk is restricted to the automated search tools for testing and scanning targets. Let's have a look at functional metrics:

Let's say software is automating search through Google dorks for scanning targets:

F(a) : automated query function applied by the software.

Google(d) = { d1 , d2 , d3, d4, ... dn} : Dork database for Google querying.

g(u) : Google standard query URL

Google(tq) = { tq1 ,tq2 ,tq3 ,tq4 .....tqn} : Time parameter for specific query.

Boolean aq = true: Automated query value whether true or false.

Q(u) = { g(u) , Google(d) } : The full automated query with Search URL and Google dork.

Browser (Q(u)) : Back end Request through browsers by automated software.

F(a) = {Browser(Q(u)) , Google(tq)}

The automated function by scanning software uses a browser based request at backend without launching browser. The query is properly constructed with search URL and Dork added to it. This will become as a full automated Scan String to be searched by Google Engine. The Browser (Q(u)) function perform the query part. The Google(tq) function define the time taken by Google Search to respond back for a particular query. The function F(a) define the full automated procedure by a designed tool.

We will be discussing some of the case and then an example to understand the real time problem posed by Google search engine.

Case 1: When automated query is a success.

```
Function automate_query_test()
{

        If (F(a) == TRUE )
                {
                        Return XML_Response ;
                        /* The scanning software update the response in software in XML format.*/

                }
}
```

Case 2: When automated query fails.

```
Function automate_query_test()
{

        If (F(a) == FALSE )
                {
                        Return XML_Response ;
                        /* The scanning software update the response as Error in software.*/

                        /* Blocked by Google. CAPTCHA Implemented */
                }
}
```

[5] SE Real Anomaly or Real Working

The Google follow certain mechanism to scrutinize the query requested. It is a kind of security feature undertaken by Google itself. It has been properly differentiated between queries generated by automated software and by humans. When ever Google finds a hint that a query is generated in an anomalous manner it asks for HUMAN interface through CAPTCHA. This feature works well through browsing

interface but when it comes to automated scanning it affects the functioning of software to a large extent. Usually the software is stuck during scanning due to secure check that comes dynamically after looking at queries. In that case the browser is launched to prove the Google Engine that human is initiating a query not any software. Actually it is basically implemented for Search Engine Worms that affect the networks by assembling as Bots. In order to circumvent the functioning, the CAPTCHA is implemented which stops the worm activity through Google Search. Basically a reverse Turing Test is conducted by Google. But the vector affects the other facets of search engine functioning.

The search engine query generation is related to URL designing. The Google implemented pre verification phase in which URL is scrutinized properly based on the metrics defined. If the URL is found to be rogue or improper like in case of automated scanning, it is redirected towards to Turing Test Machine. There are number of proxies and virtual machines are deployed in the environment to monitor the incoming queries. If a virtual machine finds a improper URL then the query mounted with that URL is not allowed. This process is termed as Virtual Machine Verification. Everything is automated from security perspective. Google has deployed well defined algorithm based heuristics to track and monitor continuously. On the lower level i.e. HTTP protocol state checks are performed. The HTTP request is scanned properly to extract the parameters used and the values passed through it. In this phase the malicious signatures are checked. All this process is implemented through a Honey NET structured there. There are number of passive and active proxies are used for dissecting the incoming traffic for better analysis. The parameters which are undertaken to deduce the query nature are:

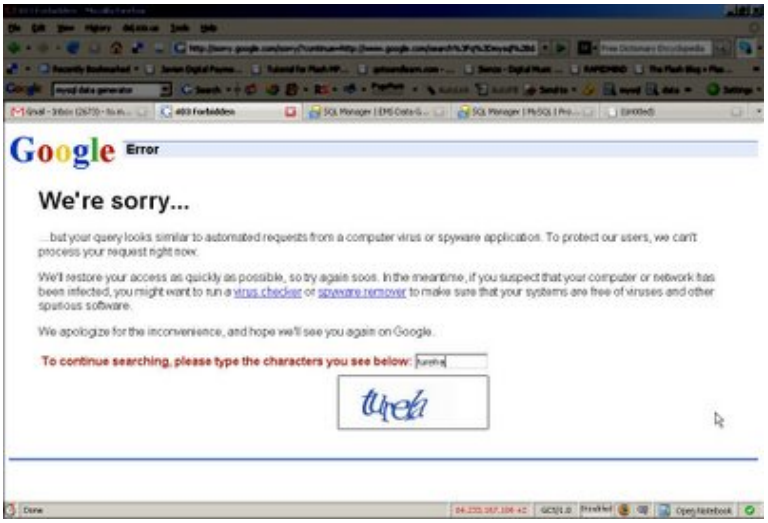
- IP Concentration defines the volume of queries generated by a specific IP address.
- Time Span defines the time gap between numbers of queries requested.
- Time Constraint function is applied on the server side for incoming query check.
- Domain Age Check is produced to find the existence status of the sender.
- Passive proxies are placed to extract the WHOIS information prior to produce results.
- IP Space Locality and Geographical checks.
- HONEYPOT traps.

There are number of other checks are defined for effective working usage of search engine.

[6] False Positive Paradox:

The automated scanning problems can be the result of FALSE POSITIVES. But due to result of these false responses the automated scanning is affected. Usually it is noticed that if an entity shows an incidence which is lower than the false positive and if the issue occurs it will be considered as false positive too. So if a search engine found that requested query is somewhat close to the false limit it will be marked as false query. Again it is a kind of anomalous behavior.

The Google simply blocks the query until Turing Test is completed. Once the query is blocked the scanning is halted or cannot be resumed till the browser is launched and CAPTCHA code is provided by the human itself. The screen appears as:



Of course this happens.

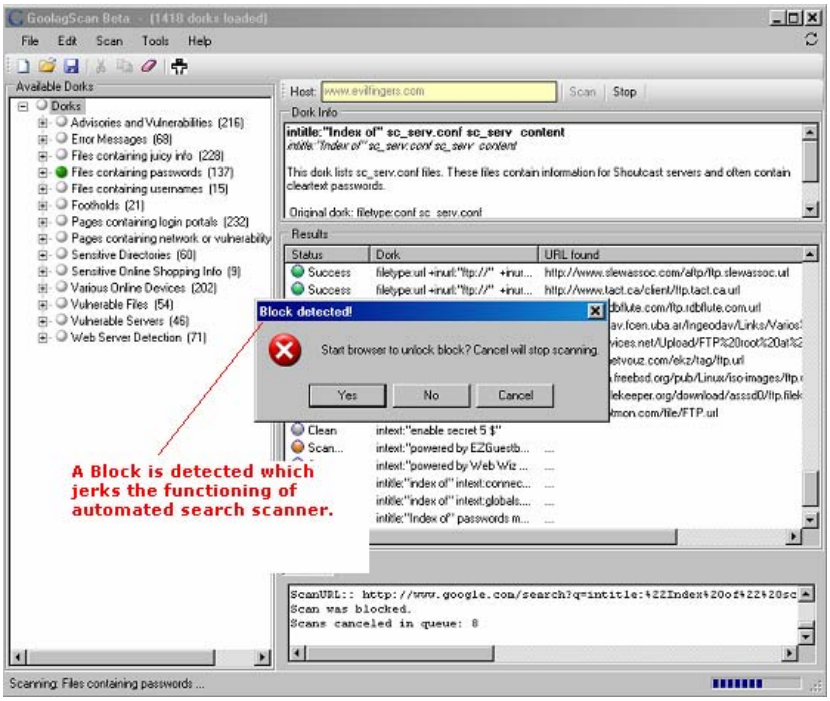
The HTTP query works as:

GET /sorry/image?id=11974966514520941203&hl=en HTTP/1.0
Host: sorry.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.15) Gecko/20080623 Firefox/2.0.0.15
Accept: image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer:
http://sorry.google.com/sorry/?continue=http://www.google.com/search?q=filetype:mdb%2520wwforum%2520+site:www.test.com
Cookie:
PREF=ID=6b05d9f3169a5304:TM=1215192351:LM=1215192351:S=hhr62k0bZkUUExx-; S=sorry=uB6tUSej1dCNgg8CaYO8-g

The HTTP raw output is shown above. So that's how the automation is affected due to security checks implemented by Google search engine.

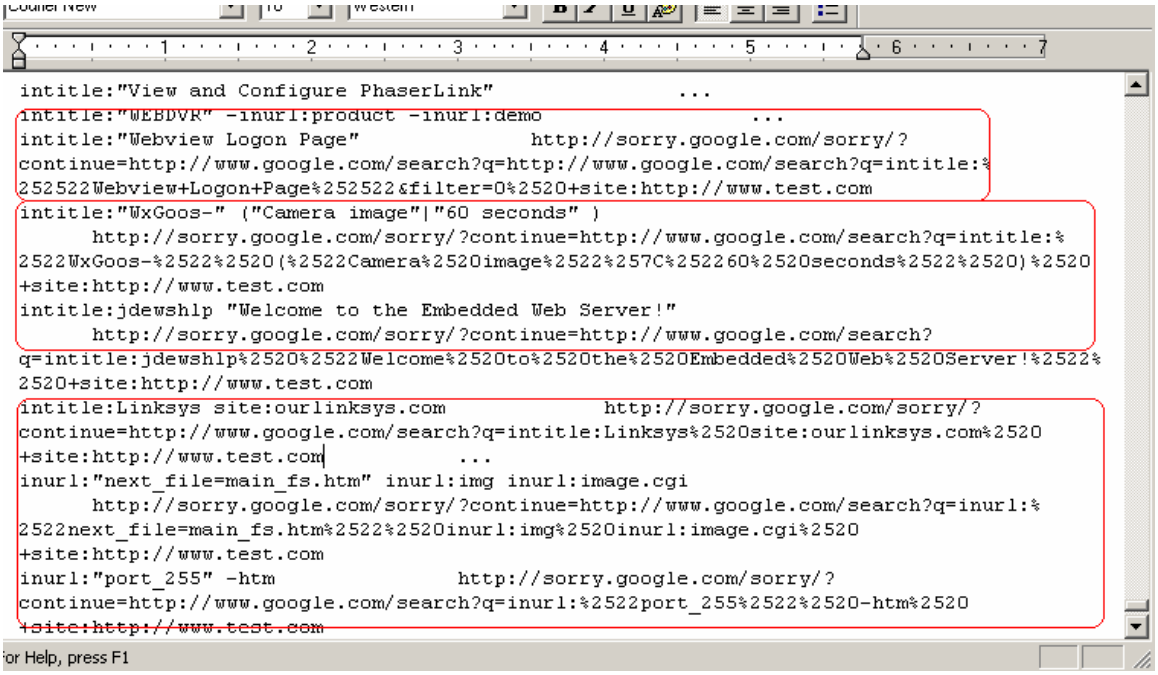
[7] Case Study: Goolag Scanner

Goolag Scanner is a kind of vulnerability scanner based on Google Dorks. This tool works efficiently in scanning the properties of the website. It is developed by CDC Group. The scanner works fine but the above stated issue really disrupts the functioning to a large extent. The Google dorks are listed in XML file which act as a scanning database. Let's see the problem again:



The above layout shows a block is occurred while scanning the target. For this specific dork the user has to manually add the required pattern which will be displayed by the sorry.google.com. Sometimes a full scanning makes the process really hard because of number of times block is detected. As I stated above this can be the cause of false positives. But it affects the nature of scanning nature of tool.

Let's see the failed dorks of one of the scan in a log file:



Again number of dorks failed. This specific problem really hits the testing. But it persists.

[8] Conclusion:

The automation is desired for testing in a time limit manner. The search engine plays a crucial role in gathering information. But it is not possible to run every single query with human intervention when a large environment is concerned. No doubt security implications are required in organizations. But false positives need to be tackled. Since the technology is getting interdependent. The weak behavior of one entity affects the working functionality of other. The anomalous behavior of search engine disrupts the automation. Stringent messages are being displayed with alarming errors. So this is some what against the rule of effectiveness. But security has its relative impacts. So this issue needs to be scrutinized and proper solution should be applied in order to prevent the future automated search engine tools.

[9] References:

[1] http://www.usenix.org/event/leet08/tech/full_papers/polychronakis/polychronakis.pdf

[2] http://www.usenix.org/events/hotbots07/tech/full_papers/provos/provos.pdf

[3] http://taviso.decsystem.org/virtsec.pdf

[4] http://www.goolag.com

[5] http://googleonlinesecurity.blogspot.com/2008/02/all-your-iframe-are-point-to-us.html

[6] http://googleonlinesecurity.blogspot.com/2007/07/reason-behind-were-sorry-message.html

[7] http://googleonlinesecurity.blogspot.com/2008/07/meet-ratproxy-our-passive-web-security.html