Security Mutation Algorithm for Google Print and Google's Search Security Deficiency

Amar Akshat

***Abstract*—The research work prove the fact that Google® installs its cookie into the user's system for more than thirty years and they expire on 17th January 2038. Google® recognized the efforts of all the researchers throughout the world and as update Google® has informed us about their changed cookie policy. Google® is modifying how it keeps track of users via cookies, by setting cookies to expire in two years if a user doesn't return and auto-extending cookie length for active users, according to a policy change announced by Google's Global Policy Counsel. However the algorithm discussed in the paper was developed at the time the when Google's® cookie expired on the date above mentioned. The algorithm emphasizes upon the web session management techniques of Google® and ways to use the services to attack the security of Google® Print cookie, which enables Google Book Search, which is a tool that searches the full text of books that Google scans and stores in its digital database. Further the paper relates to various search syntaxes provided by Google® for efficient searches and how they are incapable to maintain searcher's and searched security.**

***Index Terms* — Cookie , Google Print , Security**

I. Introduction

Analysing the Google® cookie.
GPREF=ID=61s3149f117f3033:TM=1102333418:LM=1129745420:S=gbbDQwe8rVmtrGK [4]
The Google® cookie which is in our system has four distinct parameters namely:

1. 'ID' which is hopefully unique.
2. 'TM' **is a time stamp of some sort at which Google® baked our cookie and injected into our system.**
3. 'LM' again a time stamp.
4. 'S' is a signature or checksum of some sort. **It could be a cryptic, a hash for instance. In my experience, the signature only varies with different ID and/or TM values. [5]**

Manuscript submitted April 23, 2008. The work was supported by Dept. Of CSE , Sikkim Manipal Institute of Technology , Mazhitar , Sikkim.
A.Akshat , Dept. Of CSE , 5th Semester, Sikkim Manipal Institute of Technology , Mazhitar , Rang-Po , E-Sikkim , 737132 , India.(Phone No: +919932389841. E-mail : amar.akshat@gmail.com)
The algorithm discussed in the paper relates to identifying the Google® print [3] cookie and then parsing its parameters to relate to a new parameter baking scheme.

The fact that Google® does not recognize users on a unique level helps us to iterate the procedure to generate user defined cookies. The user defined cookies would be generated by the PRNG (pseudo random number generator) [5],[1] algorithm used by Google®.

II. Analyzing Google® Print URLs

Google® Print URLs are of form:

http://print.google.com/print?id=KvBRxoA2icQW&pg=1&sig=hoLj_8Gt12aG2cSj Rxr741sbP7E

We notice a signature parameter again in the URL which actually interacts with the parameter of signature in the actual Google® cookie and generates a 4 + 22 lettered character parameter by simply modifying the parameters in signature parameter of the actual Google® cookie by some algorithm. ID in the URL points to the book that you are reading and PG points to the page number (may be). Now click the "Next Page" arrow. The URL now becomes like this.

Though Google Global Policy Counsel claims to have reduced the time limit [2], the assurance of the point can not be provided that tracking and monitoring of web activity to enhance search accuracy is discouraged by Google.

http://print.google.com/print?id=KvBRxoA2icQW&pg=2&sig= *gBBbI6T0Fz HxgVeJJQKQqmZ_MNk*

What changes here is the signature and the page number, and pg now becomes 2 , so now its sure enough that pg is the page number. **The signature changes when you change pages, and PG points to the page you started from! Google wants to limit you on reading the number of pages, because if you read the whole book then it would make the publishers unhappy. Now try removing the pg from the URL, the resulting URL results in page not found error. So we may say that the signature allotted to every URL depends upon the page we entered on, the page we are currently on and the book id.**

III. The Algorithm

// HTTP GET a page from a URL string getPage(string url)

By using some function or the getPage() function provided by Java® libraries we get the page provided as the string and download into the local memory device.

// search the book for pageNumber and return first (if any) URL found with pg = pageNumber string searchForPageURL(string bookid, int pageNumber)

Searching for the page number (which the user wants to read) in the victim book provided by Google® print pages (using a for loop) and if any such page exists then simply search for the URL string which actually points to the page, which contains any such parameters.

// extract the URL from the "next page" link on the book viewing page string extractNextPageURL(string page)

Now the function implementing the algorithm tries extracting the next page link on the same page provided to surf further on the book. Java® libraries provide us with such function which searches for all the links going from the page and returns which is closest to the matching all the parameters and some parameter changing in ascending order.

// extract the URL from the "previous page" link on the book viewing page string extractPreviousPageURL(string page)

The similar function extracts the URL for the previous page , as the link which has most of the parameters as same , except one which descends by lowest.

// from the oven if we get null ... then keep the new cookie if it works,

The “Cookie Baker” comes into action at this particular stage when method has the hard link hit and the function has older cookie overhauled. The cookie baker using the PRNG algorithm used by Google® bakes a new cookie. The Algorithm then checks for the validity of the cookie and decides its usage criteria.

// that way we can get earlier parameters of the BOOK: limits string bookid, integer pageNumber

{ // first try directly searching for the page String page = getPage(searchForPageURL(bookid, pageNumber));

The function now searches for the target page which the user is looking for the page. The getPage method from Java® libraries searches for the page having the parameters **“bookid”** and **“pageNumber”** same as provided.

// if we found the page, return the image URL from the page if (page ISNOT null)

//return extractImageURL(page);

The algorithm stops working if the target URL is achieved without violating validity checksums. The above condition is theoretically impossible to attain but in practical situations due to inefficient server load balancing features the pages outside hard link may be retrieved.

// do this for up to 2 pages, forward and backwards for (integer i = 1; i <= 2; i++)

{ // search for the i'th page after the one we want page = getPage(searchForPageURL(bookid, pageNumber + i));

The theoretically possible situation arrives here; the algorithm now gets two pages from the valid page backwards.

// if we found this one, then "click" the "previous page" button until we get to the page we want, then return the image URL from it if (page ISNOT null)

If the page returned is not null then surf to previous page links until we reach the page we want. The previous page surfing may be cumbersome so an automatic page surfer may be developed to grab the links and extract them into the page.

{ // "clicking" the previous page button for (integer j = 0; j < i; j++) { page = getPage(extractPreviousPageURL(page)); }
return extractImageURL(page); }

The method extracts the previous page URL for each iteration of ‘i’ upon ‘j’ and returns the image URL to the calling function. The page in case returned is NULL then the iteration looks for another such iteration.

// search for the i'th page before the one we want page = getPage(searchForPageURL(bookid, pageNumber - i));

The theoretically possible situation arrives here; the algorithm now gets two pages from the valid page forwards.

// if we found this one, then "click" the "next page" button until we get to the page we want, then return the image URL from it if (page ISNOT null)

The page is then surfed manually or by any tool which actually surfs the page for the user. The page which is not NULL is returned by the function.

// "clicking" the next page button for (integer j = 0; j < i; j++) { page = getPage(extractNextPageURL(page)); }
return extractImageURL(page); } }

// null }

Since we got nothing the algorithm returns NULL.

IV. Search Security Deficiency

Google is world's most popular and powerful search engine which has the ability to accept pre-defined commands as inputs and produce unbelievable results. This enables malicious users like hackers, crackers, and script kiddies etc to use Google search engine extensively to gather confidential or sensitive information which are not visible through common searches.

V. Google's® Advanced Search Query Syntaxes [6].

Below discussed are various Google's special commands and I shall be explaining each command in brief and will show how it can be used for critical information digging.

- The "**intitle:**" syntax helps Google restrict the search results to pages containing that word in the title.
- The "**inurl:**" syntax restricts the search results to those URLs containing the search keyword.
- The "**site:**" syntax restricts Google to query for certain keywords in a particular site or domain.
- This "**filetype:**" syntax restricts Google search for files on internet with particular extensions (i.e.doc, pdf or ppt etc).
- **"link:"** syntax will list down web pages that have links to the specified webpage.
- The "**related**:" will list web pages that are "similar" to a specified web page.
- The query "**cache:**" will show the version of the web page that Google has in its cache.

- The "**intext:**" syntax searches for words in a particular website. It ignores links or URLs and page titles.
- "**phonebook**" searches for U.S. street address and phone number information.

In the last few years a number of news articles appeared that warned of the fact that hackers (or crackers if you will) make use of the google search engine to gain access to files they shouldn't be allowed to see or have access to. This knowledge is nothing new to some people but personally I have always wondered how exactly a thing like this works. VNUnet's James Middleton wrote an article in 2001 talking about hackers using a special search string on google to find sensitive banking data:

"One such posting on a security newsgroup claimed that searching using the string 'Index of / +banques +filetype:xls' eventually turned up sensitive Excel spreadsheets from French banks. The same technique could also be used to find password files"[6]

An article on wired.com informed us about how hackers like Adrian Lamo used Google® as severe tool to get into boxed information of many such companies. Adrian tells us:

"For example, typing the phrase "Select a database to view" -- a common phrase in the FileMaker Prodatabase interface -- into Google recently yielded about 200 links, almost all of which led to FileMaker databases accessible online."[7].

The tabulation in figure (Fig 1) shows the results of query formulation for different cases. The advanced search query formulation successfully works as shown in the fig for attining sensitive information which has been made available on the web by simple means of tricky Google search syntaxes. There are six categories listed in the table which have been tested with syntax searches and the basic site indexing is exposed when the search is actually executed. The "Search String" column provides a basic description of the subject of search. The "Supported File Types" column lists the format of file or document for which search is targeted. "Search Query" column is for actual query with syntax which refines our search.

The category of interest lies is "Password" where the result is very sensitive for the sites who have their configuration files just listed in index without a protection layer. Two techniques have been analyzed , which is the most common way many site administrators prefer to store passwords in.

- auth_user_file.txt : a file used to store information about the authorized users , their passwords and information about the permissions awarded to them.
- config.php : a very common file to store the configuration information which is very often detailed with the password information.

Category	Search String	Supported File Types	Search Query (Google)	Remarks
1. Applications	N/A	*.exe , *.rar	-inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.exe|.rar) "video editor"	softsea.com dotnetthis.com
2. Books	Fiction Romance	*.pdf , *.doc *.pdf, *.doc	-inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.pdf) "fiction" -inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.pdf) "romance"	ibiblio.org etext.org jeffcovey.net
3. Music	Rock Jackson	*.mp3, *.wma *.mp3, *.wma	-inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.mp3|.wma) "rock" -inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.mp3|.wma) "jackson"	oshacker.free.fr rock.test.au vmp3 web portal
4. Password	Technique 1.auth_user_file.txt 2.config.php	N/A	allinurl:auth_user_file.txt intitle:"Index of" config.php	owebu.cz gray_world.net /etc/passwd file jengarner.free.fr
5. Tools	Hack Morph	N/A	related:hack related:morph	No Results No Results
6. Videos	Horror Porn	*.avi, *.flv *.avi , *.flv	-inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.avi|.flv) "horror" -inurl:(htm|html|php) intitle:"index of" +"last modified" +"parent directory" +description +size +(.avi|.flv) "porn"	avicollector.com lafferty.ca

Fig 1. Query Formulation.

The theory behind this is very simple in actual practice. The attacker only collects the maximum information he can collect about the target file and then biases the search result according to them. For example using wildcard characters we search for particular type of files (e.g.*.doc), then instead of mixed up result we get the procedure to extract the best match with that type of file executed. This is what the attacker actually hits. General attacks with open index pages and interesting hints to inside sensitive pages. A very good example is of attack on HMAC and NMAC at sensitive information with complexity low.[9]

A web server which allows browsing through indexes and directories may be visited and hunted for through Google®. The 'index of 'syntax has been doing this job for hackers throughout the world. This syntax with dangerous combinations such as:

- index of /admin
- index of /mail
- index of /passwd
- index of /confidential
- index of /root

A case study shows that when the query "index of /confidential" was issued in Google® search prompt it showed an in figure (Fig 2).

The security of navref.org is straightforward put to question. The PDF therein contains a list of personal and official letters. The case is actually not surf onto a number of pages in search, this happens to be the first result.

The security question is actually not into the parameter for one combination. A combination using other syntaxes like "**allinurl:** winnt/system32/" lists down all the sites which gives access to sensitive directories like system32 which is the by default option in server managers. Being lucky enough we may get an access to the "cmd.exe". A combination like "**inurl:** config.txt" may list up links which may have left an unchecked link to "config.txt" open.

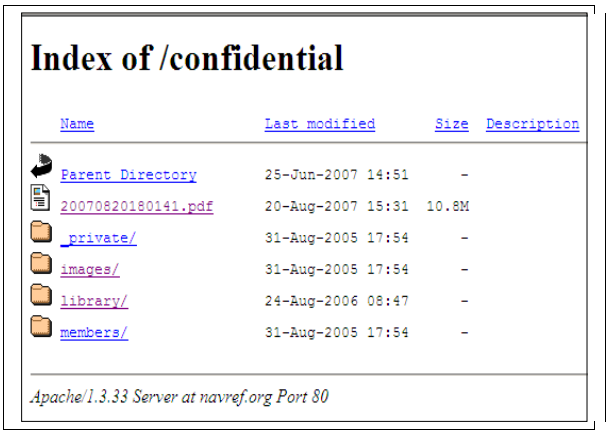

Fig 2. Screen Shot Of Google Directed Search.

Checking for the sites vulnerable to SQL injection techniques a trial like “allinurl:/privmsg.php” or “allinurl:/errmsg.php” may be efficient enough for the hacker to try the buzzer on them. Attacking has always been a combination of several hits and trials. So a much thought combination with such syntaxes may generate pages not to be viewed by general user. The link: http://wacker-welt.de/webadmin/ explains about webadmin , which is a piece of software which allows the user to remotely access an edit parts of websites. The workhouse for webadmin starts from the PHP : “webeditor.php”. Simply searching for links which have in their URL “webeditor.php” does the job for the malicious user.
“inurl: webeditor.php”
The results of the above search query :
www.namo.com/products/webeditor.php [8]
http://artematrix.org/webeditor/webeditor.php
http://www.directinfo.hu/kapu/webeditor.php
www.directinfo.hu/kapu/webeditor.php

The Freesco router software for Linux as a default, installs a small web browser which allows owners to control the router through the http protocol. In other words, a website automatically gets setup that allows you to control the router. The default password and login for this control panel is “admin” and “admin”. Many people who use freesco don’t know this. Planning an attack against it may be fruitful and syntax queries like:
“**intitle:** Freesco System” or “**intitle:** Freesco Control”

VI. Conclusion

The checks kept upon the books and the limit described by the servers or authorities is very much legal and the society respects that as well. However the cookie conspiracy which Google® was planning by keeping a track of all the web pages which every client is searching for is the closest to defacing public information security. The security regarding Google® print is at a stage that only planned attack may be able to crucify its conditions. This would not have been possible until I.T professionals throughout the world felt that they were being cheated by one way or the other.

Software designers and end users should pay more attention to default installation security configuration and security policy. In the end, there are always going to be people who make mistakes, use default installs, use poorly secured software or just don’t care or still believe there’s no danger in putting this kind of data online.

Acknowledgment

The research work is an idea out of curiosity towards the reason why Google cookies dwell in one's system by default for so long.
A.Akshat thanks Mr. C.T.Singh (Reader , SMIT) , Mr. S.Borah (Lect. SMIT) , Mr. Mohan Pradhan(Lect. SMIT) and Mr D. Mohanty's research works for Google's unsecure search feature. A vote of thanks to Dept. Of C.S.E , SMIT for encouraging such work in good faith.

References

[1] Unveiling Google Cookie Secrets by Amar Akshat. www.ethical-hackers.741.com/googly.html

[2] Google Changes Cookie Policy But Privacy Effect is Small http://blog.wired.com/27bstroke6/2007/07/google-changes-.html

[3] Google Book Search – Wikipedia. http://en.wikipedia.org/wiki/Google_Print

[4] Unveiling Google Cookie Secrets by Amar Akshat. www.ethical-hackers.741.com/googly.html

[5] Stompy - The Web Application Session Analyzer Tool http://www.darknet.org.uk/2007/03/stompy-the-web-application-session-analyzer-tool/

[6] Google not 'hackers' best friend',James Middleton, VNUnet.com, 2001 http://www.vnunet.com/News/1127162

[7] Google: Net Hacker Tool du Jour http://www.wired.com/news/infostructure/0,1377,57897,00.html

[8] Google Search: “inurl: webeditor.php” http://www.google.co.in/search?hl=en&q=inurl%3Awebeditor.php&btnG=Google+Search&meta

[9] General Distinguishing Attacks on NMAC and HMAC with Birthday Attack Complexity by Donghoon Chang and Mridul Nandi.