Design of a Crawler against Crawling Attacks (DCACA)

Saurabh Vardani[1], Vijay Maheshwari[2]
Department of Computer Engineering
Shobhit University, Meerut-250110(INDIA)
[1]saurabhvardani@gmail.com
Department of Computer Engineering
Shobhit University, Meerut-250110(INDIA)
[2]maheshwarivijay@rediffmail.com

***Abstract*—**
Web crawling is the process used by search engines to collect pages from the web. Fraudulent and malicious web sites pose a significant threat to desktop security, integrity, and privacy. This paper examines the threat from different perspectives. We harvested URLs linking to web sites from different sources and conducted a study to examine these URLs in-depth. For each URL, we extract its domain name, determine its frequency, IP address and geographic location, and check if the web site is accessible. Our study shows that users can encounter URLs pointing to fraudulent and malicious web sites not only in spam and phishing messages but in top search results returned by search engines. To provide better countermeasures against these threats, we present DCACA.
This paper describes WebCrawler, attacks on web crawler and how to prevent web crawl from attacks. WebCrawler has played a fundamental role in making the Web easier to use for millions of people. Its invention and subsequent evolution, from 1994 to 1997, helped fuel the Web's growth by creating a new way of navigating hypertext: searching. Before search engines, a user who wished to locate information on the Web either had to know the precise address of the documents he sought or had to navigate patiently from link to link in hopes of finding his destination. As the Web grew to encompass millions of sites, with many different purposes, such navigation became impractical and arguably impossible. Web Crawler as a Web service to assist users in their Web navigation by automating the task of link traversal and creating a searchable index of the Web. Conceptually, WebCrawler is a node in the Web graph that contains links to many sites on the Web, shortening the path between searchers and their destinations.
.

***Keywords*— DCACA, WWW, HTTP, URL, SSL**

I. INTRODUCTION

WebCrawler is a Web service that assists users in their Web navigation by automating the task of Link traversal, creating a searchable index of the web, and fulfilling searchers' queries from the index. Conceptually, WebCrawler is a node in the Web graph that contains links to many sites on the net, shortening the path between users and their destinations. Such a simplification of the Web experience is important for several reasons: First, WebCrawler saves users time when they search instead of trying to guess at a path of links from page to page.[1] For example, he may be viewing a page on one topic and desire a page on a completely different topic, one that is not linked from his current location. In such cases, by jumping to WebCrawler either using its address or a button on the browser the searcher can easily locate his destination page[1][2].

II. PROBLEM ASSOCIATE DURING CRAWLING

Web Crawler visits at URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit and called the crawl frontier but these hyperlinks and URLs have bad links. Once users click a bad link and land on the Malware site, they're often promoted with a fake codec installation dialog. If that doesn't get them, the site is still loaded will dozens of other tactics to infect their computer. From fake toolbars, scare ware, rogue software, and more, the sites have it all. One site that they came across even tried to install 25 different bits of Malware. Such sites are leaving people vulnerable to installations of spam bots, root kits, password Steelers, and an assortment of Trojan horses, amongst other things[3].

According to a security research firm report, Google, Yahoo, and Microsoft Live Search are currently under a large-scale organized attack campaign designed to steer online searchers towards malicious Malware websites[1][3]. Chances are, if you do any significant amount of searching, you may even run across some of these dangerous search results. Hundreds of legitimate search phrases have already been found to pull up links near the top of the results listings that lead straight to the malicious sites. According to survey, we've already found 27 different domains, each containing up to 1,499 bad pages. That's about 40,000 potential pages, which is a pretty big number.

III. DESIGN OF A DCACA

Web crawler is console application it doesn't need a rich interface. In this crawler we crawl some link from internet or WWW then we send all link to frontier queue through authentication tool, in this tool we crawl links and then retrieve page from these links and then after we compare SSL certificate of these pages with already define SSL certificates to identified original pages. In this figure 1.1 we figure out all procedure of DCACA Conclusions.

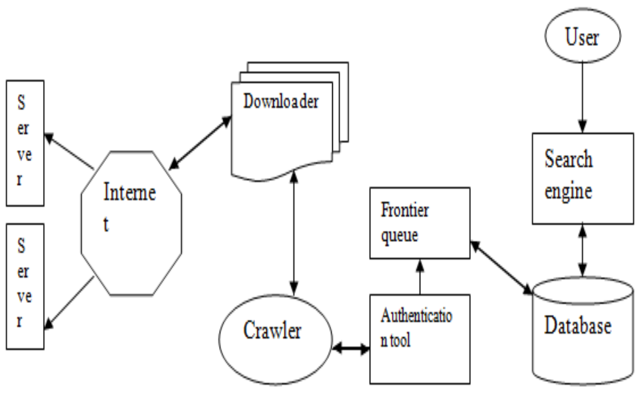

Figure 1.1: Architecture of DCACA

A crawler identifies the location of a document by its URL. From the URL, it can search and download the document as shown in fig 5.1. The crawler maintains a list of unvisited URLs called the frontier. The list is initialized with seed URLs which may be provided by user or another program. Each crawling loop involves picking the next URL to crawl from the frontier fetching the page corresponding to the URL. Through HTTP, parsing the retrieved page to extract the URLS and application specific information, and finally adding the unvisited URLs to the frontier. Before the URLs are added to the frontier they may be assigned a score that represents the estimated benefit of visiting the page corresponding to the URL[]3[4]. The crawling process may be terminated when a certain number of age have been crawled. If the crawler is ready to crawl another page and the frontier is empty, the situation signals a dead end for the crawler. The crawler has no new page to fetch and hence it stops.

FRONTIER

The frontier is a list that contains the URLs of unvisited pages. In graph search terminology the frontier is an open list of unvisited nodes. Once can expect around 60000 URLs in the frontier with a crawl of 1000 pages assuming an average of about 7 links per page. The frontier may be implemented as a FIFO (first in first out) queue. The URL to crawl next comes from the head of the new URLs are added to the tail of the queue.

HISTORY AND PAGE REPOSITORY

The craw history is a time stamped list of Urls that were fetched by the crawler. In effect, it shows the path of the crawler through the web starting from the seed pages. A URL entry is made into the history only after fetching the corresponding page. This history may be used for post crawl analysis and evaluations. For example, a value can be associated with each page on the craw path and identify significant events [7][9].

FETCHING

In order to fetching a web page, an HTTP client is needed which sends an HTTP request for a page and reads the response. The client needs to have timeouts to make sure that an unnecessary amount of time is not spent on slow servers or in reading large pages. The client may be restricted to download only the first 10-20kb of the page. In fact there is an access policy for crawler or a robot known as robot exclusion protocol. This protocol provided a mechanism for web server administration to communicate their file access policies; more specifically to identify files that may not be accessed by a crawler. This is done by keeping a file named robots.txt under the root directory of the web server [6].

PARSING

Once a page has been fetching, there is a need to parse its content to extract information that will feed and possibly guide the future path of the crawler. Parsing may imply simple hyperlink/URL extraction or it may involve the more complex process of tidying up the HTML content in order to analyse the HTML tag tree. Parsing might also involve steps to convert the extracted URL to a canonical form, remove stop words from the page's content and stem the remaining words [5].

AUTHENTICATION TOOL OF DCACA

We used authentication tool in crawling process to make crawler more efficient. With the help of this authentication tool we identify original URLs or web pages. In authentication process we retrieve web page from its URLs and then send it to comparison with its SSL (secure socket layer) certificate which identify that this page is original or not [10]. Every web page has unique SSL certificate, according to figure 1.2 we follow some procedure to retrieve original page.

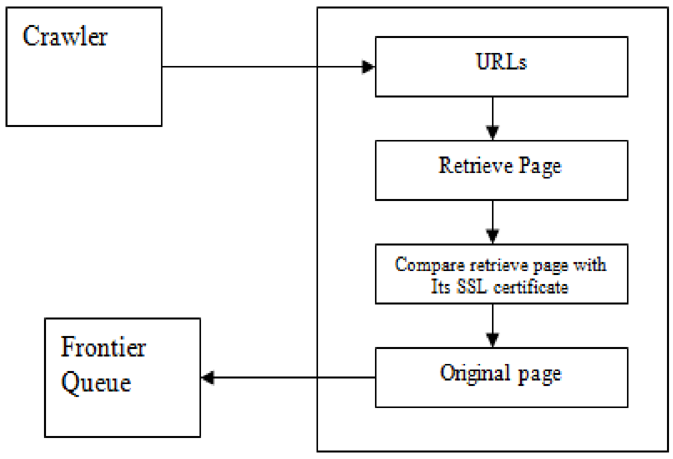

Figure 1.2 Authentication tool of DCACA

SECURE SOCKET LAYER

The Secure Socket Layer protocol was created by Netscape to ensure secure transactions between web servers and browsers. The protocol uses a third party, a Certificate Authority (CA), to identify one end or both end of the transactions. This is in short how it works .A browser request a secure page (usually https://). The web server sends it public key with its certificate. The browser checks that the certificate was issued by a trusted party (usually a trusted root CA) that the certificate is still valid and that the certificate is related to the site contacted.
The browser then uses the public key, to encrypt a random symmetric encryption key and sends it to the server with the encrypted URL required as well as other encrypted http data.
The web server decrypts the symmetric encryption key using its private key and uses the symmetric key to decrypt the URL and http data .The web server sends back the requested html document and http data encrypted with the symmetric key.
The browser decrypts the http data and html document using the symmetric key and displays the information [8] [10].

PRIVATE KEY/PUBLIC KEY

The encryption using a private key/public key pair ensures that the data can be encrypted by one key but can only be decrypted by the other key pair. The keys are similar in nature and can be used alternatively: what one key encrypts, the other key pair can decrypt. The key pair is based on prime numbers and their length in terms of bits ensures the difficulty of being able to decrypt the message without the key pairs. The trick in a key pair is to keep one key secret (the private key) and to distribute the other key (the public key) to everybody. Anybody can send an encrypted message that only we will be able to decrypt, If we have the other key pair, In the opposite, we can certify that a message is only coming from right person, because we have encrypted it with our private key, and only the associated public key will decrypt it correctly. Beware, in this case the message is not secured you have only signed it. Everybody has the public key, remember! One of the problems left is to know the public key of your correspondent. Usually you will ask him to send you a non confidential signed message that will contains his public key as well as a certificate[10].

THE CERTIFICATE

How do we know that we are dealing with the right person or rather the right web site? Well, someone has taken great length (if they are serious) to ensure that the web site owners are who they claim to be. This someone, we have to implicitly trust: you have his/her certificate loaded in your browser (a root Certificate). A certificate contains information about the owner of the certificate, like e-mail address, owner's name, certificate usage, duration of validity, resource location or Distinguished Name (DN) which includes the Common Name (CN) (web site address or e-mail address depending of the usage) and the certificate ID of the person who certifies (signs) this information. It contains also the public key and finally a hash to ensure that the certificate has not been tampered with. As you made the choice to trust the person who signs this certificate, therefore we also trust this certificate. This is a certificate trust tree or certificate path. Usually your browser or application has already loaded the root certificate of well known Certification Authorities (CA) or root CA Certificates. The CA maintains a list of all signed certificates as well as a list of revoked certificates. A certificate is insecure until it is signed, as only a signed certificate cannot be modified. we can sign a certificate using itself, it is called a self signed certificate. All root CA certificates are self signed [9][10].

GETTING THE GOOD LINK FROM AN INTERNET USING DCACA

The first crucial piece of building a crawler is the mechanism for going out and fetching the html of the web. Like so much else, c# has classes for doing this very thing built into the framework.

```
Public static void Crawl Site ()
       {
          Console Write Line ("Beginning crawl.");
          Crawl Page (Configuration Manager App Settings
["urls"]);
          String Builder sb = Create Report ();
          Write Report To Disk (sb. To String ());
          Open Report in IE ();
          Console Write Line ("Finished crawl.");

       }


Private static void Crawl Page (string url)
       {
          if (Page Has Been Crawled (url))
          {
             String html Text = Get Web Text (url);

             Page page = new Page ();
             Page Text = html Text;
             Page Url = url;
             Page Calculate View state Size ();

             pages Add (page);

if (string link in link Parser is original Urls)
             {
                String formatted Link = link;
                Link = good urls;
                {

Formatted Link = Fix Path (url, formatted Link);

if (formatted Link != String Empty)
                   {
                      Crawl Page (formatted Link);
                   }
                }
                catch (Exception exc)
                {
failed Urls Add (formatted Link + " (on page at url " + url + ")
- " + exc Message);
                }
            }
        }
```

The Http Web Request class can be used to request any page from the internet. The response (retrieved through a call to Get Response ( )) holds the data you want.

IV. Conclusion

A web crawler is a program that someone uses to view a page, extract all the links and various pieces of data for the page, which then hits all the links referenced on that page, getting all the data for those, and so on. This is how search engines, for example, get all their data. They write crawlers.
And that is exactly what we needed; something to crawl site to make sure all links are good. So we decided to write one, and we are sharing it i.e. DCACA. Between here and there is a discussion of some of the more interesting bits of features and code in the crawler.

V. References

[1] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt, pages 200–209. Morgan Kaufmann, 2000, (Cooley, Mobasher, & Srivastava 1997; Srivastava et al. 2000).

[2] Chiasen Chung and Charles L. A. Clarke. Topic-oriented collaborative crawling. In11th ACM Conference on Information and Knowledge Management, McLean, Virginia, November 2002, (Mobasher 2005; 2007).

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the 7th World-Wide Web Conference (WWW7), 1998. Online at http://decweb.ethz.ch/WWW7/1921/com1921.htm, (O'Mahony et al. 2004; Lam & Riedl 2004; Burke, Mobasher, & Bhaumik 2005; Mobasher et al. 2005)

[4] S Chakrabarti, M M Joshi, and V B Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In SIGIR, 2001, (Burke et al. 2005)

[5] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. JACM, 46(5):604–632, 1999.

[6] Junghoo Cho. Crawling the Web: Discovery and Maintenance of a Large-Scale Web Data. PhD thesis, Stanford University, 2001.

[7] Andrew McCallum.Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Software available from http://www.cs.cmu. edu/~mccallum/bow/, 1998.

[8] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. Computer Networks (Amsterdam, Netherlands: 1999), 31(11–16):1623–1640, 1999.

[9] Filippo Menczer. Links tell us about lexical and semantic Web content. Technical Report Computer Science Abstract CS.IR/0108004, arXiv.org, August 2001. Online at http://arxiv.org/abs/cs.IR/0108004.

[10] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In WWW, Hawaii. ACM, May 2002.

[11] Michael Hersovici, Michal Jacovi, Yoelle S Maarek, Dan Pelleg, Menachem Shtalheim, and Sigalit Ur. The Shark-Search algorithm-an application: Tailored web site mapping. In 7th World-Wide Web Conference, Brisbane, Australia, April 1998.
Online at http://www7.scu.edu.au/programme/fullpapers/1849/com1849.htm.