Vulnerability Scanner for SQL Injection based on Deep Web Harvesting

Vrushali Narendra Bora[1]
[1] *Vrushali Narendra Bora, Student , Computer Department, MCOERC, Maharashtra, India*

ABSTRACT

Deep harvesting is a focused area of research these days. In deep websites contents lie behind search-able web interfaces. The deep websites are not registered to any search engine. In deep web harvesting framework site location and balanced in-site exploration carried out. This leads to wide coverage of data on sites and high efficiency for interfacing websites. In the proposed work, a web vulnerability scanner for SQL injection based on deep web harvesting is designed. In this system vulnerabilities like SQL injection, directory access, injection of vulnerabilities and attacks is framed and analysed by security testing tool of web application. The system will also focus on retrieval of hidden websites from large scale sites and on improving the performance of harvesting by increasing its rate.

Keyword: - *Security, SQL injection, internet application, Deep web harvesting, web crawler.*

1. INTRODUCTION

The content which can not be indexed by searching engines is called as deep or hidden websites. It is provoke to stimulate the hidden web databases, because they are not enrolled with any search engines, are usually comparatively scattered, and are unstable. The Deep Web has been acknowledged as a important division in the coverage of search engines because web crawlers employed by search engines rely on hyperlinks. Various accounts have assumed that the Deep Web has an structure of magnitude more data than the currently searchable World Wide Web Furthermore, the Deep Web has been a deep-standing challenge for the database community because it represents a large fraction of the structured data on the Web.

Website security is most important need in today's world ,an enterprise and should be a priority in any organization. progressively, hackers are concentrating their intensions on web-based applications: shopping carts, forms, login pages. Accessible anytime from anywhere in the world, insecure web applications provide easy approach to backend corporate databases and allow hackers to perform illegal activities using the attacked sites. A victim's webpage can be worn to fire acts.

Hackers already have a wide repertoire of attacks that they regularly fire across organizations including SQL Injection, Cross Site Scripting, Directory Traversal Attacks, Authentication Attacks, Directory Enumeration and other exploits.

2. LITERATURE SURVEY

2.1 Locating deep web content sources

The output rate of hidden web is less only six lakhs forty seven thousand different web forms were searched by sampling twenty five million pages from Google index. Generic crawlers are mainly developed for characterizing deep web and directory construction of deep web resources, that do not limit search on a specific topic, but attempt to fetch all searchable forms. The Database Crawler in the Meta Query is invented for accordingly inventing query interfaces. By an IP address sampling the database crawler, first finds root pages and then to crawl pages it performs shallow crawling within a web server from a given root pages. One IP address can have one or more virtual hosts, thus missing of many websites due to IP address sampling. To conquered the defect of IP based sampling in the Database Crawler, Denis invented a stratified random sampling of hosts to characterize national deep web, using the Host graph provided by the Russian search engine Yandex. I-Crawler mixed prequery and post-query access for classification of searchable forms [1].

2.2 Selecting Relevant sources

Previous system normally have small number of relevant databases of deep web directories [3],[4],[7]. Which are only used for satisfying the access needs. Focused crawler is invented for focusing on relevant topic links visit it and avoid the links which are irrelevant [1], [2], [3], [4]. Best-first focused crawler is used for classifying the pages [4]. The page classifier determine to classify pages as topic-relevant or irrelevant and gives priority accordingly to links in topic relevant pages. However, a focused best-first crawler harvests only 94 movie search forms after crawling 100,000 movie related pages [3]. An improvement to the best-first crawler is proposed in [2], where instead of following all links in relevant pages, the crawler used an additional classifier, the apprentice, to select the most promising links in a relevant page. The baseline classifier gives its choice as feedback so that the apprentice can learn the features of good links and prioritize links in the frontier. The FFC [2] and ACHE [3] are focused crawlers used for searching interested deep web interfaces. The FFC contains three classifiers: a page classifier that scores the relevance of retrieved pages with a specific topic, a link classifier that prioritizes the links that may lead to pages with searchable forms, and a form classifier that filters out non-searchable forms. ACHE improves FFC with an adaptive link learner and automatic feature selection. Source Rank [2], [3] assesses the relevance of deep web sources during retrieval. Based on an agreement graph, Source Rank calculates the stationary visit probability of a random walk to rank results. Different from the crawling techniques and tools mentioned above, Smart Crawler is a domain-specific crawler for locating relevant deep web content sources. Smart Crawler targets at deep web interfaces and employs a two-stage design, which not only classifies sites in the first stage to filter out irrelevant websites, but also categorizes searchable forms in the second stage. Instead of simply classifying links as relevant or not, Smart Crawler first ranks sites and then prioritizes links within a site with another ranker.

2.3 Security mechanisms using vulnerability

2.3.1 Detection of Vulnerabilities by Security Scanner

By a collection of signatures of known vulnerabilities security scanners identify defects and weaknesses. As new vulnerabilities are discovered, Signatures are updated regularly. The scanners execute lots of pattern variations adapted to the specific test in order to detect the vulnerability, like XSS and SQL injection are discover in search of vulnerabilities. There are two main types to test web applications for vulnerabilities. The ”white box” focuses on consists of the testing of the source code of the web application. Static code analysis is a type of white-box analysis[3][4]. This can be done manually or by using code testing tools like FORTIFY [3], Code Secure, etc. These static analyser tools analyses source code to detect vulnerabilities, such as SQL injection and cross site scripting. The black-box vulnerability scanner without knowing the internal design of the web application uses fuzzy techniques over the web HTTP requests, simulates numerous scenarios such as hackers’ intentional attacks or general users’ inadvertent attacks, and provides an automatic way to find for, neglect the repetitive and monotonous task of doing hundreds or even thousands of tests by hand for each vulnerability type. There are many commercial web vulnerability scanners for black box testing such as Acunetix Web Vulnerability Scanner [1], HP Web Inspect [4], IBMAppScan [5]. The testing results of vulnerabilities for web applications are quite different from scanner to scanner. According to the survey presented in [3], black-box testing is the second most used technique to calculate the effectiveness of security. In our experiments, we use our proposed approach to calculate four famous ”black box” commercial scanners, App Scan, Web Inspect, Paros, and Acunetix.

2.3.2 Web Vulnerability Scanner Evaluation Criteria

Vulnerability scanners are considered as a solution for detecting vulnerabilities and security threats in web applications. Among the studies focusing on tools evaluations [2][3], the Web Application Security proposed ”Web Application Security Scanner Evaluation Criteria project to come up with a set of detailed evaluation criteria and a framework for conducting a formal scanner calculation. The goal of the WASSEC is that for the tools given to users they need to conduct a solid evaluation and make their own informed resolve on which scanner(s) best meet their needs. P.E. Black. proposed guidelines for describe the functional specifications of Source Code Security Analysis Tool and Web Application Security Scanner in NIST Special Publication 500-268[7] and 500-269[6]. Through the development of tool functional specifications, project aims to better quantify the state of the art for contrast classes of software security assurance tools. The documents constitute a specification for a particular class of software assurance tool, which is referred to here as a web application security scanner. By examining the steps in scanning processes, we can reasonably assume that costs of vulnerability scanner include construction cost, functioning cost, and analysis cost, with operation and analysis ones being the main parts in cost evaluation. These processes are

generally labour-intensive and often involve substantial human resources, including developers, domain experts, and security experts. There are several studies focusing on reducing cost in vulnerabilities detection, such as [5][6]. However, the issue of redundant alerts have not been considered in the previous research. In general, when a certain defect is found repeatedly, the developers would spend double effort to solve it. In this paper, we proposed a cost-effective evaluation approach to evaluate vulnerability scanner by considering issue of redundant alerts.

3. SYSTEM ARCHITECTURE

Checking vulnerability manually is huge and complex task, since it involves processing a large volume of data. It also demands a high level of expertise and the ability to keep track of considerable volumes of code used in a web application. In addition, hackers are constantly finding new ways to exploit your web application, which means that you would have to constantly monitor the security communities, and find new vulnerabilities in your web application code before hackers discover them.

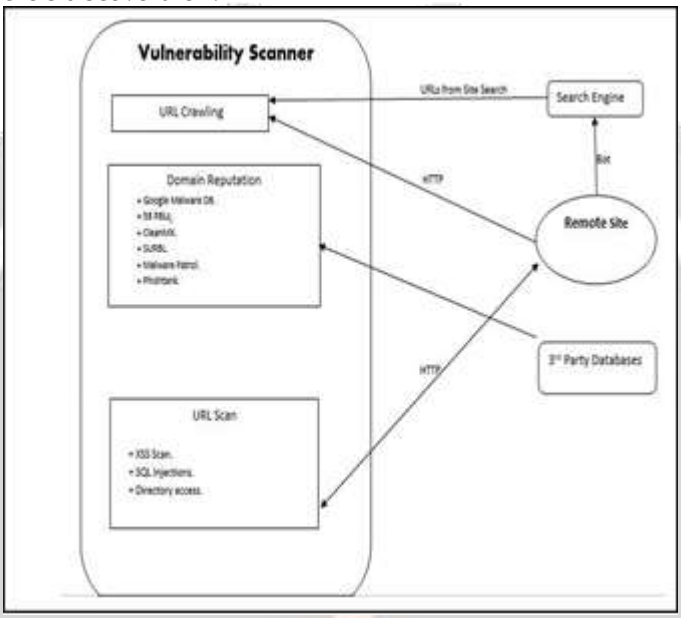

Fig:1 System Architecture

3.1 URL Crawling

URL crawler is mainly used to crawl the URL's from the search engine. If we search any keyword using search engine the crawler will find the number web pages for that particular search. When we search any One keyword it can contains number of web pages and each web page has its URL but when we click on that URL it will again number of URL's for that one web page so we can say that it is been call recursively for one single search. So basically the URL Crawler is used to crawl the web pages for the particular search it will automatically crawl the URL's and will show limited URL's or web pages when we search any keyword the limit is given up to 6-7 web pages will show after we will search for the particular keyword.

3.2 Search Engine

Search engine is mainly used for searching contents on the network here the system also used search engine for searching content various search engine are available now a days we are also using Bing search engine in the system for searching the contents so that the user can get the required data easily which they want. The search engine is mainly used for getting the limited data instead of getting huge number of data after applying the search.

3.3 3rd Party Database

The 3rd party database is mainly used to check reputation of site and to check whether the site is corrupted. It checks if any site is black listed or not and if the site is black listed it gives the message for that it is mainly to check Vulnerability of black listed sites.

3.4 Domain Reputation

Domain reputation is mainly used to check the black listed sites so that reputation for that site will check. Various domains are available for checking the reputation of sites different domains are listed in the system architecture. Domain reputation will check the vulnerability of sites using different domains mainly the RBL's are used to check mail server's IP and it checks whether the server's IP is black listed or not.

3.5 CMS Scan

Many tools are available to design the website and due to this easily available tool more changes are to cause Vulnerability. Tools like Joomla, WordPress are some example of CMS. In this the hackers check for the loopholes to detect Vulnerability. This tool are open source so it is easy for hackers to detect code also this kind of tools used specific type of format and this formats are easily available for the hackers and they detect Vulnerability easily.

3.6 URL Scan

The URL Scan is mainly done using two types of methods GET and POST. Using these two types of methods we check the Vulnerability for URL. For GET method a specific pattern is been define and for every URL these specific pattern is applied to check Vulnerability while in POST method no specific type of pattern is applied in this the whole URL is applied to check Vulnerability. Rather than only checking for Vulnerability the system is a product which can be applied to detect Vulnerability and to solve that Vulnerability the advance part in the system is that the system is fully automated no manual work is required to do the system fully develop to be automated and anyone can understand it easily and use it. Due to automated design of system it will work fast and results will generated quickly.

4. ALGORITHMIC STRATERGIES

4.1 Algorithm for Searching more site

1) Input: seed websites and harvested deep websites
2) While of cansite less than a threshold do
3) Pick a deep website
4) Site= get Deep Websites
5) Result page= reverse search (site)
6) links= extract links (result page)
7) for each link into links do
8) page= download page (link)
9) relevant = classify (page)
10) if relevant then
11) relevant sites= extract unvisited sites (page)
12) output= Relevant sites.

4.2 Algorithm for Incremental Site Prioritizing

1) Input: SiteFrontire
2) HPQ= Create queue
3) LPQ= Create queue
4) if HPQ is emp then
5) HPQ.addAll(LQueue)
6) LPQ.clear()
7) end
8) site = HPQ.poll()
9) relevant = classify Site(site)
10) if relevant then
11) performInSiteExploring(site)
12) Output forms and OutOfSiteLinks
13) siteRanker.rank(OutOfSiteLinks)
14) if forms is not empty then
15) HQueue.add (OutOfSiteLinks)
16) end
17) else
18) LQueue.add(OutOfSiteLinks)
19) end

20) output: Out-of-site links and searchable forms .

5. STATE OF RESULT

In proposed systemuser will enter the URL and it will give us more specific result that is relevant result using deep web harvesting. After clicking on go button it will again display hidden websites.

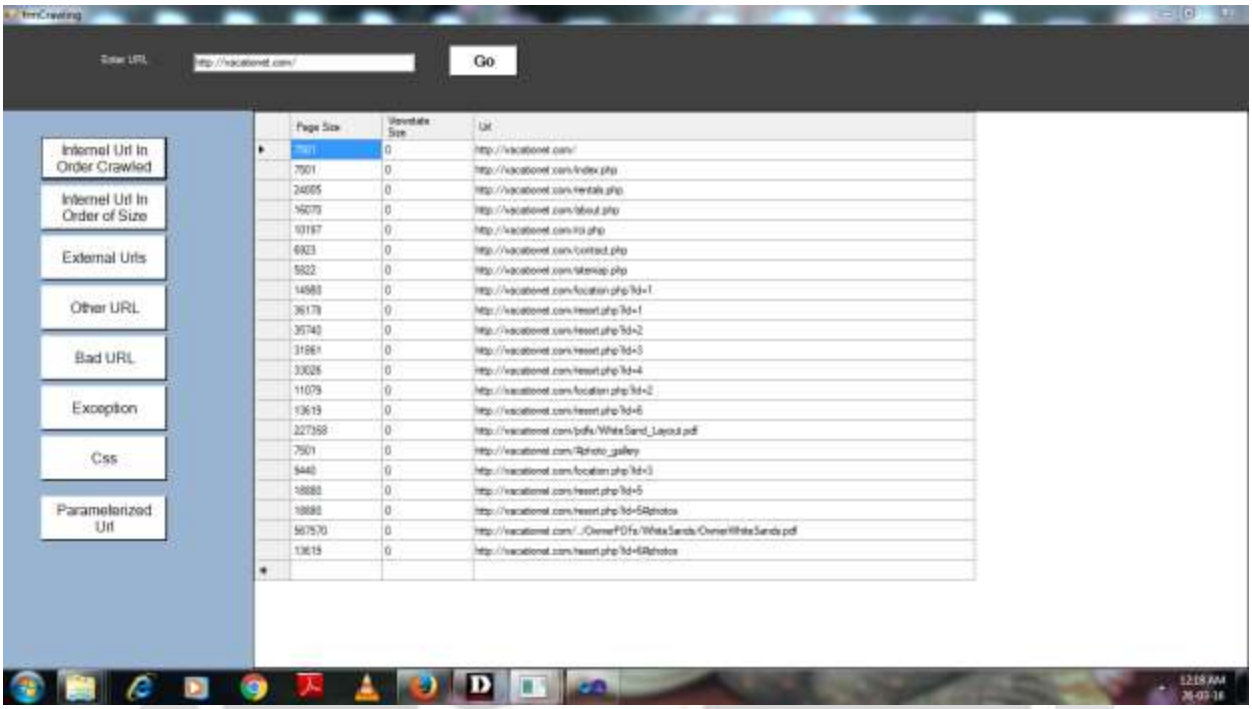

Fig. 2. URL Crawling stage1

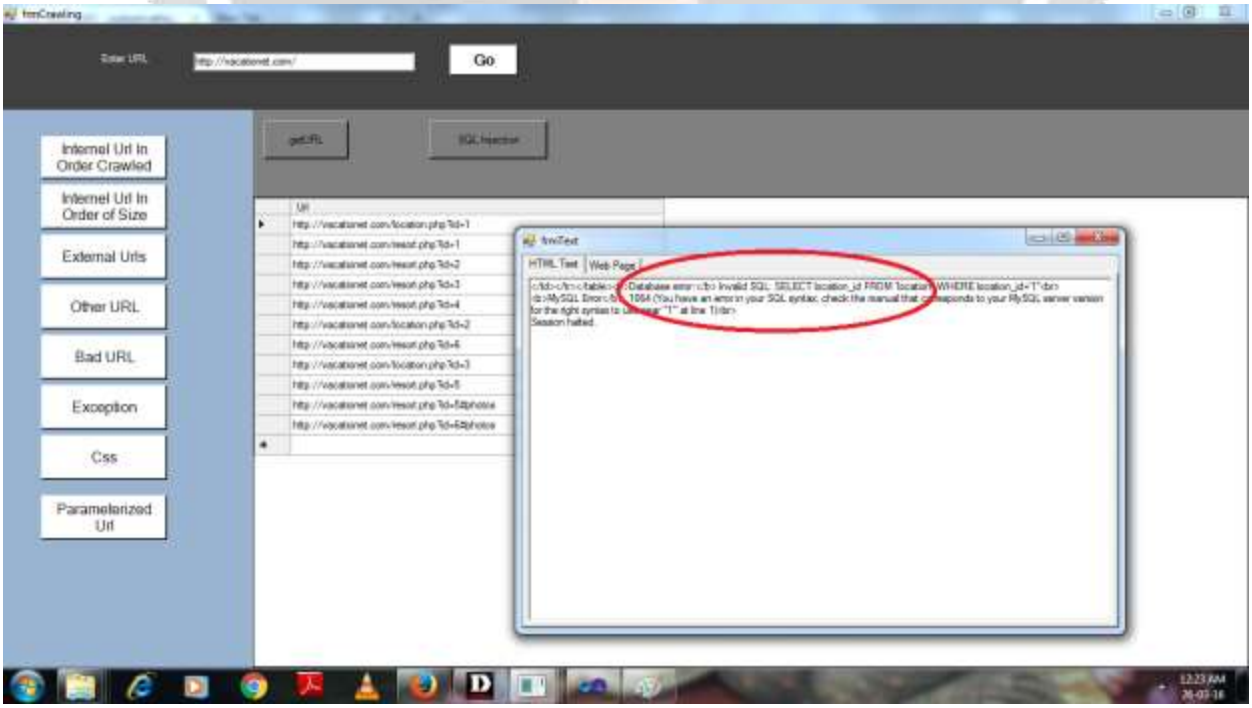

Fig. 2. SQL Injection

5. CONCLUSION

The proposed work, a web vulnerability scanner for SQL injection based on deep web harvesting is designed. In this systema web application security testing tool that audits web application by checking for vulnerabilities like SQL injection, directory access, injection of vulnerabilities and attacks is framed and analysed. The systemwill also focus on retrieval from large scale sites of deep web interfaces and on improving the performance of harvesting by increasing its rate. Thus, this application is useful for removing the vulnerabilities of websites.

6. REFERENCES

[1]. Feng Zhao, Jingyu Zhou, Chang Nie, Heqing Huang, Hai Jin SmartCrawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces,IEEE Transactions on Services Computing, Volume 99 Year: 2015

[2]. Luciano Barbosa and Juliana Freire, An adaptive crawler for locating hidden-web entry points,In Proceedings of the 16th international conference on World Wide Web,ACM:2007.

[3]. Denis Shestakov and Tapio Salakoski, On estimating the scale of national deep web,In Database and Expert Systems Applications, Springer:2007.

[4]. Balakrishnan Raju, Kambhampati Subbarao, and Jha Manishkumar, Assessing relevance and trust of the deep web sources and results based on inter-source agreement ACM Transactions 2013.

[5].] Olston Christopher and Najork Marc, Web crawling Foundations and Trends in Information Retrieval,2010.

[6] Larry Suto Analyzing the Accuracy and Time Costs of Web Application SecurityScanners,SanFrancisco, 2010.

[7] J. Fonseca, M. Vieira, and H. Madeira, Testing and comparing web vulnerability scanning tools for sql injection and xss attacks, 2007