International Journal of Scientific Research & Engineering Trends
Volume 6, Issue 1, Jan-Feb-2020, ISSN (Online): 2395-566X

Web Application Vulnerability Exploitation using Penetration Testing scripts

Shubham Rawat , Tushar Bhatia, Eklavaya Chopra
Under guidance of : Ms Akanksha Dhamija(Assistant Professor)
Bhagwan Parshuram Institute of Technology
Guru Gobind Singh Indraprastha University
Delhi, India

***Abstract* – Now a day many Webapps are being developed which on the one hand are beneficial but on the same part contains a lot of vulnerabilities. Most of the work which remains untouched is web security. Online shopping and web services are increasing at rapid rate. Cross Site Request forgery(CSRF) and Cross side scripting(XSS) are some of the top vulnerabilities. Going through this paper, we will cover a lot of vulnerabilities that are present in webapps and will be presenting some real world threats to the web apps. The vulnerabilities will be found out by the help of penetration testing. Various threat models for the vulnerabilities are also mentioned to give a good understanding about them.**

***Keywords*- Cross Site Scripting, Penetration testing, Cross Site Request Forgery.**

I. INTRODUCTION

The World Wide Web has changed the life of people very drastically.All individuals as well as large organizations everyone are using Web. Web application like personal websites, forums, e commerce application is spread all over the world. Looking at the present Scenario Most of the infrastructures like banks, stock market, communication, defence all are using Web Application. As the data which is managed by the site web application increases and the web apps expand,so does the risk of attack by malicious minds also increases. Several WebApps are being developed lately and with the growing amount of work the need of we applications are also increasing. Most of the webapps are contains a considerable amount of vulnerabilities. Some of the common vulnerabilities are as follow:-

1. Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) is one type of an attack over a Web application in which an adversary causes a victim's browser to perform an unwanted action on a trusted website via a malicious link or other content. It is an old type of application attack. It was discovered by Norm Hardy in 1988, application level trust issue and called it confused deputy [1].
It has appeared in the top 10 Web application threats.[2]

2. Cross-Site Scripting

Cross-site Scripting (XSS) in web applications emerged as one of the most frequent types of security vulnerabilities in The last years [3]. Unlike related problems, such as SQL injection, XSS attacks do not affect the server-side but clients: The actual exploitation is within the victim's web browser. Therefore, the operator of a web application has only very limited Evidence of successful XSS attacks.XSS related problems are therefore often overlooked or recognized rather late.

3. SQL Injection

SQL Injection vulnerability may affect to dynamic web application which stored data in the associated database. Through SQL Injection, attacker passes malicious code to SQL Server through inserting it in the strings. This malicious code is commonly known as payloads that instruct the database server to retrieve specific information from database.

Inspite being aware of all the vulnerabilities, web application are still being deployed with the vulnerabilities which make them an easy target to attack.Developers still need to have more knowledge about the vulnerabilities and attacks so that they can handle it during the development phase itself.In this paper, by the help of Penetration Testing we will be finding out the some of the above mentioned vulnerabilities and after finding those vulnerabilities solution will be provided on how we can protect our Web Apps from such type of web attacks.The Information that will be covered through out the research paper will help the developers and researchers to get knowledge about security related scenarios so that they can make their sites free from the vulnerabilities.

II. RELATED WORK

In a research work, there was some development in finding XSS vulnerability. [3] They evaluated where the attack will take place in the JS script and targeted their focus on it. They analysed that on which section of code is targeted by attacker and discussed how the attack

would take place. Also, there is this research paper , which shows how XSS works in detail. They have made few scripts that are capable to perform Generic as well as Reflected XSS.They have discussed full mechanism on how a string is accepted by system through automata and have shown that how their exploit will help them retrieve the information of the system.They are testing the web app manually and are not using any tool. [5]

Another work shows that how cross site request forgery (XSRF) works. In this , they have illustrated how the XSRF attack works and have given some methodologies on how to prevent it using a threat model . They have shown various Data Flow Diagrams(DFD) to illustrate how data flows through a web app and how they will forge a hoax request , so that they get access to their data. They have shown some common XSRF attacks and their preventive measures.They also created malicious script and exploits which depicted on how much vulnerable the web app is to the XSRF attack. [4] This work shows preventive measures on XSRF attack only.

There is also a research which shows how to prevent and protect the web app from these attacks. It shows various attacks and how they are used to exploit vulnerabilities like XSS, XSRF, SQL Injection,etc. They have also discussed about the Security

measures to prevent these attacks , like applying network firewall, Intrusion Detection System (IDS), Intrusion Prevention System (IPS),etc .[8] The goal of that research paper was to illuminate the preventive measures for a web app.
There is also this intriguing research , which shows how an attacker thinks while attacking on a web app . They have statistically supported their research on how much damage these web exploits and payload can cause while exploiting the common vulnerabilities. This research helped in getting a gist about how an attacker may proceed on finding basic vulnerabilities and how much risk these vulnerabilities are to the system.[9]

This research paper is exhilarated by these researches and how they proceed on a web app. Web apps are usually vulnerable as the developer were unaware of the vulnerabilities that are very common. This research is distant from the rest as here the developers created the web app keeping in mind on how much vulnerable can their web app is. Web app are usually tested for vulnerability assessment, but this research shows building a web app from scratch and then exploiting it with exploits and malicious scripts. The developers working on this research scripted a web app and demonstrated the the techniques to extract data from it.[2] This is a stand-alone approach used to depict how much secure the data is on a basic web app implemented on a website. Showing vulnerability report after assessment will not solve the purpose of Penetration testing, but making a schematic model on how much it is easy for an attacker to deploy few exploits to extract user credentials , will illuminate the need of the hour,i.e., deploying preventive measures against exploitation of common vulnerabilities.

1. Work Done:

In this research, we took a web app and tried to demonstrate the common web application attacks, i.e. , cross site scripting, sql injection and cross site request forgery. Firstly, we took three tools which are, Nikto, Sqlmap and XSStrike. These three tools are used to access these vulnerabilities and help in exploiting them. In this research, these tools were altered and an exploit was attached at the end so that after running the script , the exploit can attack on the vulnerability after scanning them

2. Analysing the most Appropriate type of attack:-

Website today consist of numerous type of vulnerabilities. Some of the vulnerabilities which are mostly being popped out in webapps are Sql injections, Xss forgery and cross site scripting. These vulnerabilities need to be handled. If they are left unhandled then a lot of consequences needs to be faced in the future.

3. Analysing process has been done on the listed attacks:-

Sql injection is a code injection technique where malicious queries of Sql are used to control a web application database; It scans the database and try to do SQL injection. If the database supports sql and the encryption is weak , then it will notify that the attack is successful and will display the login credentials.The script will run common exploits like basic sql query for blind sql, obfuscated queries and few more till we can confirm that sql injection is possible. If the database doesnt exist , then it will simply not run and will give no result on the console. Cross-site Scripting (XSS) in web applications emerged. As one of the most frequent types of security vulnerabilities in the last years [3]. Unlike related problems, Such as SQL injection, XSS attacks do not affect the server-side but clients: The actual exploitation is within the victim's web Browser. Therefore, the operator of a web application has Only very limited evidence of successful XSS attacks.

4. Cross-Site Request Forgery (CSRF) is one type of an attack:- over a Web application in which an adversary causes a victim's browser to perform an unwanted action on a trusted website via a malicious link or other content. It is an old type of application attack. It was discovered by Norm Hardy in 1988, application level trust issue and called it confused deputy [1].It has appeared in the top 10 Web application threats[2].

5. Scriptsthat are being used for Penetration Testing:-

5.1 NIKTO

Nikto is a script which scans the website and finds the vulnerabilities that are present in the website. If the web server is hosting multiple sites using virtual hosts then all hosted sites need to be scanned. It takes about 45 minutes

for the script to run on the Website.As it is perl based security tool it will execute on most of the platforms with the necessary perl interpreter.It can be use to used to find every type of obscure issues that are present in the website.

5.2 Nikto Process

Firstly, in Nikto , the script scans the whole web app and its database and finds all the vulnerabilities in them . Then , after finding the php file or jwt tokens , the exploit which is attached in the script will run and try to exploit the xsrf(cross site request forgery) vulnerability. It will try few combinations of exploit and then it will report if the attack was successful or not. It will work simply by getting the hostname and the perl command to run it. Ti saves the retrieved data in TXT, CSV,etc. If the attack was successful, it will notify and the user id and password will be defined. Before performing any scan we need to update the nikto database packages using the following command

/usr/local/bin/nikto.pl –update.

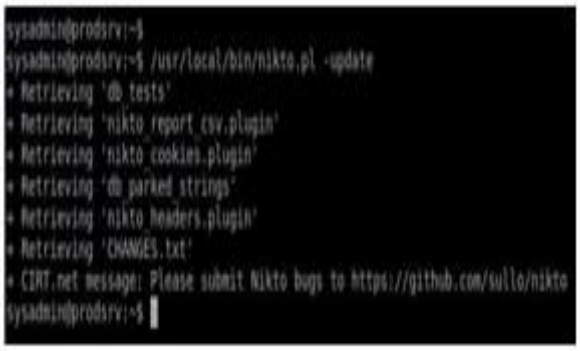

Fig.1. Above command on system's terminal.

To list the available Plugins for nikto we can use the below command.

nikto.pl -list-plugins

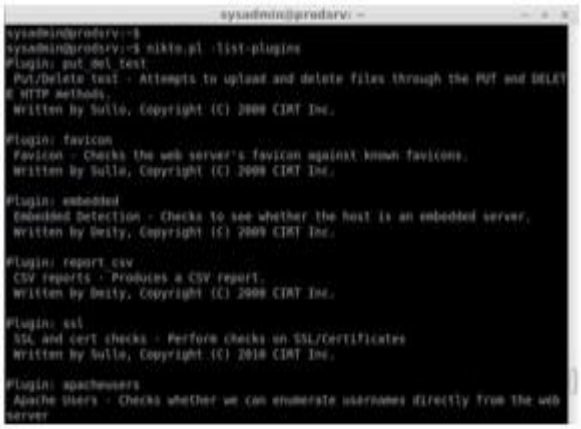

Fig.2. Above command on system's terminal.

To Scan for a website using hostname we can use the option -h followed by niktop command.

nikto.pl -h www.google.com

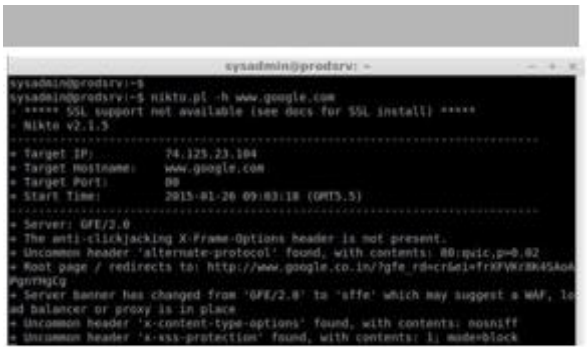

Fig.3. Above command on system's terminal.

Scan for a hostname using multiple ports we can use -p option followed by nikto.pl.

nikto.pl -h www.google.com -p 80,443

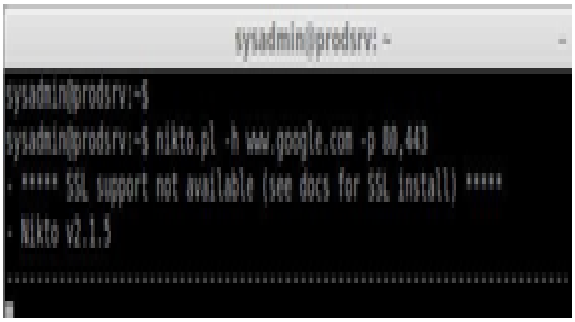

Fig.4. Above command on system's terminal.

While scanning for vulnerabilities we can see the process, If we need to see the live process we need to use option Display.

nikto.pl -D v -h www.google.com

Where,
-D = Display
v = Verbose
-h = hostname

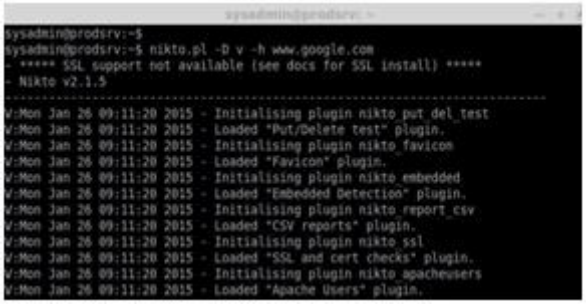

Fig.5. Above command on system's terminal.

While Tuning options used we can specify which test need to made, Using x option we can exclude the tests which we don't need.

Below Options are available for specific scan's.

0 – File Upload
1 – Interesting File // we will get in logs 2 – Misconfiguration / Default File
3 – Information Disclosure
4 – Injection (XSS/Script/HTML)
5 – Remote File Retrieval – Inside Web Root
6 – Denial of Service // Scan for DDOS 7 – Remote File Retrieval – Server Wide
8 – Command Execution // Remote Shell
9 – SQL Injection // Scan for mysql vulnerabilities
A– Authentication Bypass b – Software Identification

c – Remote Source Inclusion x – Reverse Tuning Options
Now here it scans for SQL vulnerabilities for a website. A single test will finish in short time if we Have not specified for a single scan it will take the full scan and take hours to complete.

nikto.pl -Tuning 9 -h www.isanalytics.com

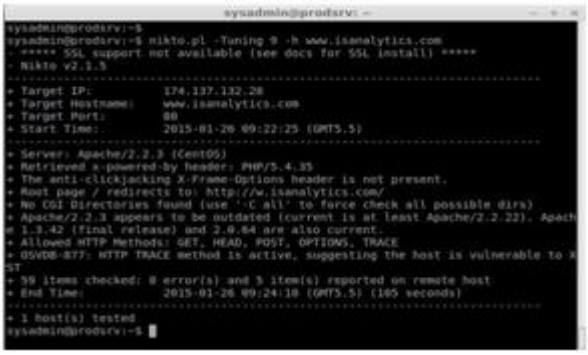

Fig.6. Above command on system's terminal.

Scan and save the result to a file using below command to find the vulnerabilities:

nikto.pl -Display V -o nikto_scan_result.html -Format html -h 192.168.0.166.

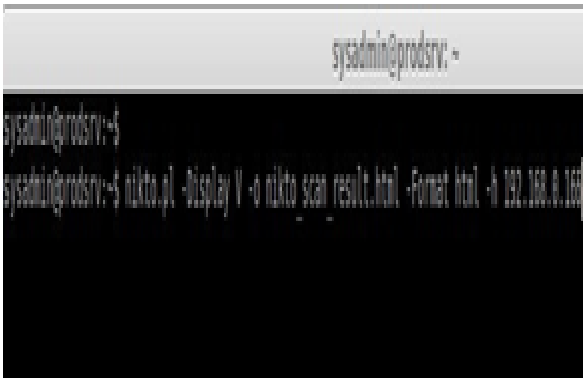

Fig.7. Above command on system's terminal.

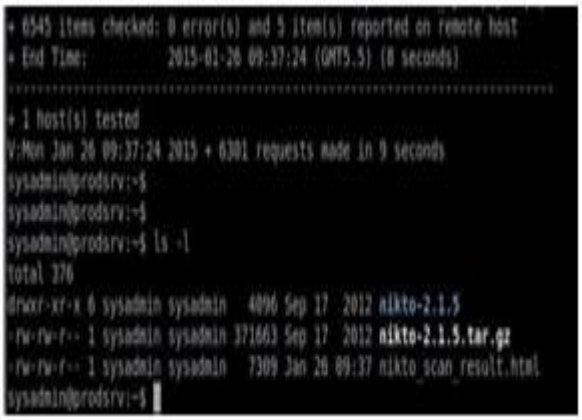

Fig.8. Result after the scan is complete.

Fig.10. Output of the vulnerabilities in nikto_scan_result.html.

1. Sql Map

Sql injection is a code injection technique where malicious queries of Sql are used to control a web application database; Sqlmap scans the database and try to do SQL injection. If the database supports sql and the encryption is weak, then it will notify that the attack is successful and will display the login credentials.The script will run common exploits like basic sql query for blind sql, obfuscated queries and few more till we can confirm that sql injection is possible. If the database does not exist , then it will simply not run and will give no result on the console.

2. Sql Map Process

In SqlMap, the first thing which happens is that we enter the web URl that we want to check along with the -u parameter. -tor can be used to test the website with proxies.

Fig.11.Example of SQLi Errors from Different Databases and Languages.

Now it test whether the database is being accessible or not. If the site is filled with vulnerabilities. If the database is accessible now, it can easily list the tables present in the database as well as it can be use to list down the columns of the tables. - dump command is used to retrieve the data. To list down the vulnerable databases, Run the following command :-
sqlmap -u
http://www.gbhackers.com/products_showit
em_clemco.php?item_id=28434 --dbs
Screenshot after running the commad is given below :-

Fig.12. Demonstration of how the terminal will display output when above command will be executed.

The command that is used for listing the tables in the database is as follow :-
sqlmap -u
http://www.gbhackers.com/cgi-bin/item.cgi?i tem_id=15 -D
clem co industries --tables

It will list down the tables as follow :-

[10:56:25] [INFO] retrieved: item
[10:56:27] [INFO] retrieved: link
[10:56:30] [INFO] retrieved: other
[10:56:32] [INFO] retrieved: picture
[10:56:34] [INFO] retrieved: picture_tag
[10:56:37] [INFO] retrieved: popular_picture
[10:56:39] [INFO] retrieved: popular_tag
[10:56:42] [INFO] retrieved: user_info

Fig.13. Demonstration of how the terminal will display output when above command will be executed.

Now list the columns of the table using the command :-

sqlmap -u
http://www.gbhackers.com/cgi-bin/item.cgi?i tem_id=15 -D
gbhackers-T user_i nfo --columns

This returns 5 entries from target table user_info of clemcoindustries database.

[10:57:16] [INFO] fetching columns for table
'user_info' in database 'gbhackers '
[10:57:18] [INFO] heuristics detected web
page charset 'ISO-8859-2'
[10:57:18] [INFO] the SQL query used
returns 5 entries
[10:57:20] [INFO] retrieved: user_id
[10:57:22] [INFO] retrieved: int(10) unsigned
[10:57:25] [INFO] retrieved: user_login
[10:57:27] [INFO] retrieved: varchar(45)
[10:57:32] [INFO] retrieved: user_password
[10:57:34] [INFO] retrieved: varchar(255)

[10:57:37] [INFO] retrieved: unique_id
[10:57:39] [INFO] retrieved: varchar(255)
[10:57:41] [INFO] retrieved: record_status
[10:57:43] [INFO] retrieved: tinyint(4)

Fig.14. Demonstration of how the terminal will display output when above command is executed.

Now the next step is to list the usernames from the targeted table form targeted database, to do the same, the command used is :-

sqlmap -u
http://www.gbhackers.com/cgi-bin/item.cgi?i tem_id=15 -D
gbhackers-T user_info -C user_login –dump

Fig .15. Demonstration of how the terminal displays output after the above command will be executed

Now the work for extracting password is required. It can be through the following command:-

sqlmap -u
http://www.gbhackers.com/cgi-bin/item.cgi?i tem_id=15 -D gbhackers-T
user_info -C user_password --dump

Fig .16. Demonstration of how the terminal will show output after command will be executed.

The password extracted by this process is a hashed password. This hashed password can be easily decrypted by a hash Identifier hence giving us the password for an extracted username.

3. XSSTRIKE

XSStrike is a python based Script with fuzzing which can bypass the systems FireWall and can detect the Xss vulnerabilities. The Scripting payloads are being stored in the SQL Lite Database. GET as well as POST Http request both are being supported by this tool. It fuzzeez the parameters with payload . It is possible because of the fuzzer module. It consist of an Striker module which is used to provide brute force parameters. Crawler functionality is also present with the script due to the presence of the spider module.

4. XSStrike process

In Xsstrike, the script will check that the web app can be exploited by XSS(cross site scripting) or not. It will crawl through the code and will check that we can input the command into it or not. It will try to look for the website and would try to sneak in the script command using javascript. If the script can be inserted , the it will notify and we can manually insert the script using console of the browser.

To start the process:-

Command to execute the XSS script is as followed :-

Python3 xsstrike.py

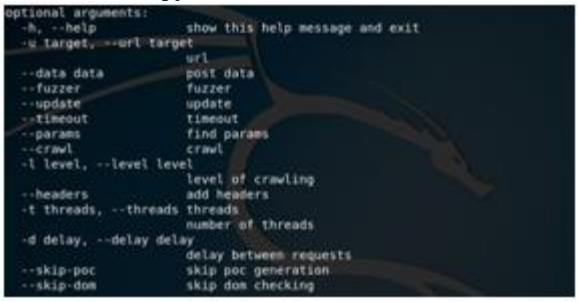

Fig.17. Above command in system's terminal.

The next step is to find out the vulnerabilities in the website using the XSS commands:- python3 xsstrike.py –u <target web application>

Above command is used to find the vulnerabilities using the XSScript To run XSStrike on a test web application, the tool searches for both, DOM and reflected XSS vulnerabilities, starting with DOM XSS.

python3 xsstrike.py –u
http://testphp.vulnweb.com/listproducts.php
?cat=1

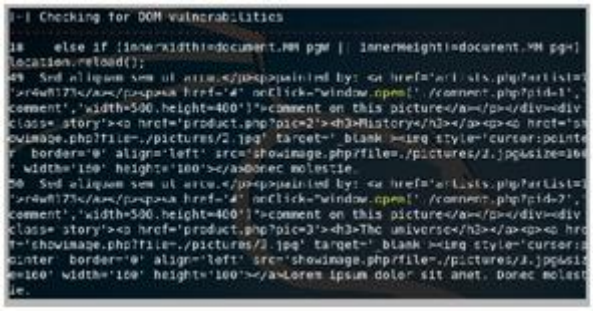

Fig.18.Output on terminal after running the command.

The above output shows the scripts were the scripts can be altered and these DOM vulnerabilities can be extracted on the console of the web browser. This script can be altered on the website source code and the website gets altered and everyone can see the modified script.

REFERENCE

[1]. R. A. Baeza-Yates and G. H. Gonnet. Fast text searching for regular expressions or automaton searching on tries. Journal of the ACM, 43(6):915 – 936, November 1996.
[2]. Blwood. Multiple xss vulnerabilities in tikiwiki 1.9.x. mailing list Bugtraq, http://www.securityfocus. com/archive/1/435127/30/120/threa ded, May 2006.
[3]. S. Christey and R. A. Martin. Vulnerability type distribu-tions in cve, version 1.1. [online], http://cwe.mitre. org/documents/vuln-trends/index.ht ml, (09/11/07), May 2007.
[4]. K. Fernandez and D. Pagkalos. Xssed.com - xss (cross-site scripting) information and vulnerabile websites archive. [on-line], http://xssed.com (03/20/08).
[5]. D. Gusfield. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cam-bridge University Press, New York, USA, 1997. ISBN 0521585198.
[6]. W. G. Halfond, A. Orso, and P. Manolios. Using positive tainting and syntax-aware evaluation to counter sql injection attacks. In 14th ACM Symposium on the Foundations of Soft-ware Engineering (FSE), 2006.
[7]. O. Hallaraker and G. Vigna. Detecting malicious javascript code in mozilla. In Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), pages 85–94, June 2005.
[8]. R. Hansen. XSS (cross-site scripting) cheat sheet - esp: for filter evasion. [online], http://ha.ckers.org/xss. html, (05/05/07).
[9]. O. Ismail, M. Eto, Y. Kadobayashi, and S. Yamaguchi. A proposal and implementation of automatic detection/collec-tion system for cross-site scripting vulnerability. In 8th In-ternational Conference on
[10]. Advanced Information Network-ing and Applications (AINA04), March 2004.
[11]. T. Jim, N. Swamy, and M. Hicks. Defeating script injection attacks with browser-enforced embedded policies. In 16th International World Wide Web Conference (WWW2007), May 2007.
[12]. A. Klein. Cross site scripting explained. White Paper, Sanc-tum Security Group, http://crypto.stanford. edu/cs155/CSS.pdf, June 2002.
[13]. Klein. Dom based cross site scripting or xss of the third kind. [online], http://www.webappsec.org/ projects/articles/071105.shtml, (05/05/07), Sebtember 2005.
[14]. J. Kratzer. Jspwiki multiple vulnerabilitie. Posting tothe Bugtraq mailinglist, http://seclists.org/ bugtraq/2007/Sep/0324.html, September 2007.
[15]. C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03), pages 251–261. ACM Press, October 2003.
[16]. G. Maone. Noscript firefox extension. Software, http://www.noscript.net/whats, 2006.
[17]. Misc. New xss vectors/unusual javascript. [on-line],http://sla.ckers.org/forum/read.php ? 2,15812 (04/01/08), 2007.
[18]. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans. Automatically hardening web applications using precise tainting. In 20th IFIP International Information Security Conference, May 2005.
[19]. T. Pietraszek and C. V. Berghe. Defending against injec-tion attacks through context-sensitive string evaluation. In Recent Advances in Intrusion Detection (RAID2005), 2005.
[20]. A. Pigrelax. Xss in nested tag in phpbb 2.0.16. mailing list Bugtraq, http://www.securityfocus.com/archi ve/1/404300, July 2005.
[21]. D. Scott and R. Sharp. Abstracting application-level web security. In WWW 2002, pages 396 – 407. ACM Press New York, NY, USA, 2002.
[22]. P. Sowden. rbnarcissus. Software, http://code. google.com/p/rbnarcissus/ (04/01/08), 2008.
[23]. E. Ukkonen. On-line construction of suffix trees. Algorith-mica, 14:249 – 260, 1995.