

1 Defensive Approaches on SQL Injection and Cross-Site Scripting 2 Attacks

3 icom Technologies¹, Venkatramulu Sunkari² and Dr. C.V Guru Rao³

4 ¹ kits warangal

5 *Received: 12 December 2013 Accepted: 2 January 2014 Published: 15 January 2014*

6

7 **Abstract**

8 SQL Injection attacks are the most common attacks on the web applications. Statistical
9 analysis says that so many web sites which interact with the database are prone to SQL
10 Injection/XSS attacks. Different kinds of vulnerability detection system and attack detection
11 systems exist, there is no efficient system for detecting these kinds of attacks. SQL Injection
12 attacks are possible due to the design drawbacks of the websites which interact with back-end
13 databases. Successful attacks may damage more. The state-of-art web application input
14 validation techniques fails to identify the proper SQL/XSS Vulnerabilities accurately because
15 of the systems correctness of sanity checking capability, proper placement of valiators on the
16 applications. The systems fail while processing HTTP Parameter pollution attacks. An
17 extensive survey on the SQL Injection attacks is conducted to present various detection and
18 prevention mechanisms.

19

20 *Index terms—*

21 **1 Introduction**

22 SQL Injection attack is a web application vulnerability that occurs because of improper validations at the server
23 side. National Vulnerability Database (NVD) is an International security organization and is organized by the
24 U.S Government. In this, most of the security threats and the vulnerability (flaws) will be published. Each
25 Vulnerability (Software Flaws) is identified with CVE-ID. When we see the vulnerabilities (CVE-IDs) published
26 to till date there are total of 60598. Among all these vulnerabilities 5922 are sql injection flaws and 8074 are cross
27 site scripting flaws. Exploit-db is a security community. The site publishes vulnerability details possibly with
28 Proof Of Concept(POC). Vulnerability research or response teams and most of the hackers or crackers participate
29 for their fame and name. This site provides a separate category called web apps. In this category we can see
30 the website hacked details. Currently this site is publishing 100 to 200 POC for every month. Famous and Open
31 Source Intrusion Detection System SNORT is providing detection logics not more than twenty. By these logics
32 we can detect upto 20-40 sql injection attacks. So many commercial IDS/IPS Systems are also providing very few
33 logics. By this analysis we can conclude that, SQL Injection attacks are more and there is no efficient detection
34 system for detecting and for protecting web applications from SQL Injection attacks. In the most of the website
35 home pages we see as the Fig. ?? text and password boxes to enter into the website. For example if we have login
36 and password to use the web services, and login as admin and password as admin0123. We enter login, password
37 and then we click on submit. Our browser sends the http GET request and these values(login, password)
38 will be submitted to the appropriate program file, in the above example validate.jsp as an input parameters.
39 In the middle of the transmission we can observe this request as "GET http://www.example.com/validate.jsp?
40 username=admin&password=admin0123 HTTP/1.1".

41 Here the validation process on the server is validate.jsp and it accepts the parameters username and the
42 password. If the above request is received by the www.example.com webserver, then that server sends the
43 requested values to the validate.jsp with the argument values. Validate.jsp validates the username and password

4 III.

44 with its back-end database (Say ORACLE Server). Before interacting with the database validate. Jsp script
45 creates a dynamic SQL Query for validating the user inputs. Let us assume that the code for the validate.jsp is
46 designed as Fig. ??effunction. If this validate.jsp takes admin as username and admin0123 as password, then the
47 dynamic query will be created at the runtime is var sql = "select * from users where username = '" + username +
48 "' and password = '" + password + "'". Dynamic query will be sql=select * from users where username=admin
49 and password=admin0123. If the user or attacker enters the values for username, and password as "Username :
50 or 1=1 -" and "Password : xyz" In the scenario, the dynamic query will be created below sql=select * from users
51 where username= or 1=1-and password=xyz. In the sql statement username= will become one condition which
52 returns false and the condition 1=1 which is tautology condition and returns always true. These two conditions
53 here are joined with or. so that total result will be true for always. And the Statement (-) is used as comment
54 statement in the most of the sql supported database management systems. If this comment statement statement
55 appears in the middle of the SQL Query, then the rest of the query will be ignored. So that when we execute
56 the above SQL Query, The result of execute query(sql) will be non-zero and returns all the records of the users
57 table. And then attacker may gain the admin access,(Because of the The result of the SQL Injection will be
58 very severe. Like this we can find more number of attack or hacked details in the security websites.

59 2 II.

60 3 Sql Injection Attacks

61 In the most of the website home pages we see the text and password boxes as shown in Fig. ?? to enter into the
62 website. In general this page is used to allow the authorized persons from the remote to use the web application
63 services. For this kind of pages, most of the developers develop the code as Fig. 2. For example if we have login
64 and password to use the web services, and login as admin and password as admin0123. We enter login, password
65 and then we click on submit. Our browser sends the http GET request and these values(login, password) will be
66 submitted to the appropriate program file, in the above example validate.jsp as an input parameters. In the middle
67 of the transmission we can observe this request as GET/validate.jsp?username = admin&password = admin0123
68 HTTP/1.1 Here the validation process on the server is validate.jsp and it accepts the parameters username and
69 the password. If the above request is received by the www.example.com webserver, then that server sends the
70 requested values to the validate.jsp with the argument values. Validate.jsp validates the username and password
71 with its back-end database (Say ORACLE Server). Before interacting with the database validate. Jsp script
72 creates a dynamic SQL Query for validating the user inputs. Let us assume that the code for the validate.jsp
73 is designed as If this validate.jsp takes admin as username and admin0123 as password, then the dynamic query
74 will be created at the runtime is varsq = "select _fromuserswhereusername =0 " + username+ "0andpassword
75 =0 "+password+"0"; Dynamic query will be sql = select _fromuserswhereusername = adminandpassword =
76 admin0123; If the user or attacker enters the values for username, and password as below Username : or 1=1
77 -Password : xyz

78 In the above scenario, the dynamic query will be created below sql = select _fromuserswhereusername = or1
79 = 1 ? ?andpassword = xyz; in the above sql statement username= will become one condition which returns
80 false and the condition 1=1 which is tautology condition and returns always true. These two conditions here are
81 joined with or. so that total result will be true for always. And the Statement (-) is used as comment statement
82 in the most of the sql supported database management systems. If this comment statement statement appears
83 in the middle of the SQL Query, then the rest of the query will be ignored. So that when we execute the above
84 SQL Query, The result of execute query(sql)will be non-zero and returns all the records of the users table. And
85 then attacker may gain the admin access,(Because of the entered user will be treated as the result of the first
86 record and most of the SQL users Tables first record may be the admin). Because of no validations are done
87 at the server-side for the user inputs, an attacker execute his own queries, instead of the developer expected
88 query. And it is possible to insert another SQL Queries by combining with UNION Statement. Example: If
89 the attacker enters below values username: ' or 1=1 union insert into users values(venkat,venkat0123,admin)
90 -password: xyz Like this if any vulnerability found on the website parameter, an attacker can inject his own
91 queries for insert,update,etc.,.

92 The result of the SQL Injection will be very severe. Like this we can find more number of attack or hacked
93 details in the security websites.

94 4 III.

95 Sql Injection Defence Schemes a) PaulE et al Scheme

96 In [1] the authors developed a white-box tool to verify software security. In general software requirement
97 specifications, source code, designs and executable code to be analysed by tools. In this work, the authors
98 developed a security scanner tool. It can analyse the functional behavoir. Due to the widespread use of
99 the World Wide Web and proliferation of web application vulnerabilities, application level web security and
100 assurance requires major attention. This specification defines a minimum capability to help software professionals
101 understand how a tool will meet their software assurance needs. The tool can be used as software assurance tool
102 and it can scan the software for security vulnerability for some extent. b) Muthuprasanna et.al Method

103 In [2] the authors developed a model of hybrid approach, which combines static code verification and runtime
104 analysis. Webservice protection became necessary because of the use of webapplications is increasing in the
105 internet. Deployments of Webapplication firewalls, next generation firewalls, application detection systems and
106 intrusion detection and prevention systems are increasing to protect web servers.

107 **5 c) Hossain et.at Method**

108 In [3] the authors developed a mutation based testing tool to verify the web application resistance against
109 SQL Injection vulnerabilities. The authors stated that the current scenario of testing web application cannot
110 eliminate web application vulnerabilities. The proposed that injecting attack pattern into the source code of the
111 web application, by that mutations based test cases can be generated. The generated test cases can potentially
112 find the SQL Injection vulnerabilities. The authors named the tool as MUSIC. The tool is evaluated on open
113 source web applications written in JSP. The tool is further impleted for PHP and other known languages.

114 **6 d) Russell et.at Method**

115 In [4] the authors developed a low-level approach to find the runtime applicability of sql statement which are
116 prepared at runtime. The authors achieved this using call level interface (CLI) by interacting ODBC or JDBC.
117 Using this approach, the authors evaluated the runtime SQL statement with SQL DOM approach. CLI can be
118 used for to verify the correctness, but SQL DOM is can be used to identify the SQL statement applicability such
119 as user permissions. SQL DOM can be prepared automatically by interacting with the database schema. The
120 authors evaoluted the system for performance. The approach is a offline approach. The posed SQL statements
121 should be given as an input the tool. So, it cannot be directly applied for dynamic query evalution. The authors
122 are extending the work with XPATH query language for dynamic queries verification. e) Tania et.at Method
123 Software systems are complex for verification and validation. Software faults causes security vulnerabilities and
124 causes for security breaches. Several methods such as SQL attack tree models and fault injection models are
125 best comparable to this work [5]. The tool injects the critical attack patterns onto the system and verifies the
126 result for vulnerability existence. The validation methods provided with this tool avoids false positives. The tool
127 reports accurate report, each vulnerability reported by the tool will be based on the behavior of the application
128 at the time of attack injection. As a future work, the authors targeted to generate injection methods based
129 on attack tree models. f) Huajun et.al Method Phishing attack is an identity theft attack, mostly on banking,
130 online-transactions etc,. The attackers uses socail sites to steal the user's sensitive information such as credit-card
131 details, account details etc,. Phishing also includes social engineering schemes. Social engineering schemes can
132 be using emails, phone calls claiming that the callers are from valid authorities. Phishing attacks are typically
133 cross-site scripting attacks. The authors [6] proposed few strategies to avoid phishing attacks.

134 Anti-phishing are classified into three categories by the authors. Server-side anti-phishing strategies, browser-
135 side antiphishing strategies, and online training anti-phishing strategies. 1. Server-side anti-phishing strategies:
136 This approach will be applied the server side. It works similar to anti-spam systems. It verifies the content
137 delivered to the server. If anything which is very closely related to phishing, the detection system prevents it at
138 the server and not to reached to the victim. 2. Browser-side anti-phishing strategies: This approach is brower
139 based approach with plug-in. The plug-in monitors the application behavior at the user-side. If it behaves as
140 cheating or phishing it avoids the attack. The browser based approaches can be categorised as Blacklist approach,
141 visualclue-based approaches or capta based approach, webpage-feature-based approaches and information flow
142 approaches.

143 **7 Online training anti-phishing strategies:**

144 The last strategy suggests that the internet users should have proper training on phishing attacks, how to avoid
145 them. This approaches clears the antiphishing philosophies. The strategy is to create awareness on phishing
146 attacks. The authors suggest to have a technology called webpage watermarking to fight against phishing attacks.

147 **8 g) Anderson Morais et al Method**

148 Attack Injection model [7] for security protocol testing suggests to have attack injection to find the web application
149 vulnerabilities. The approaches includes attack tree model to generate testcases. The attack tree model prepares
150 all possible cases. fault injector injects attack patterns onto the server system. The fault injects prepares executes
151 scripts that are collected from the internet. The approach can be used as blackbox model. The authors created a
152 framework which executes the given scripts. The authors are focusing on UML based representation to generate
153 attack scenarios in future.

154 **9 h) Sushila Madan et al Method**

155 Web applications are most vulnerable to popular attacks and risks. SQL Injections and cross-site scripting
156 attacks are more popular attacks on web applications. Threat modeling provides a complete assessment on the
157 web application. With techniques such as attack possible entry point, attack trees, privilise escalation chances
158 the tester or security assessment team can indentify the threats on the system. In [8] the authors aimed to create

15 CONCLUSION

159 attack risk model called ADMIRE. The system is concise, structured. The approach is step wise approach. The
160 steps includes : (i)Analyze the security objectives (ii) Divide the application (iii) Mark the vulnerabilities (iv)
161 Identify the threats (v) Rank the threat (vi) Eliminate the threat. The model is white-box model. It verifies the
162 application code. It is specific to a programming language.

163 10 i) Parvaiz et al Method

164 In [9], the authors suggests that the attack tree model is not possible in all cases and is difficult to build the
165 security model. Applications operates in different modes, capturing every aspect becomes difficult to design the
166 security model with attack tree models. Hence the authors proposed a new approach which provides syntax and
167 graphical security models. The new model includes nodes such as PAND node, k/n node, SEQ node, CSUB node,
168 and Housing node. The system provides syntax and graphical representation for every node. The model allows
169 the developer to understand the system affectively. The system is fault resistant and avoid vulnerabilities during
170 development phase. In most of the cases the tree includes AND/OR models to represent the system structure.
171 As a future plan, the authors are working on to define calculation rules for the new nodes to distinguish the node
172 values for different security attributes up to the root node of the tree. j) Nenad et al Method [10] Along with
173 web applications even vulnerabilities have grown. Since reviews of manual code are costly, timeconsuming and
174 even error-prone, the need for solutions has become evident. This addresses the problem of web applications
175 which is vulnerable by means of static source code analysis. Many analysis like flow-sensitive, interprocedural,
176 context-sensitive data flow and even literal analysis are used to discover vulnerable points in a program and also
177 to improve the correctness and precision of the results. Pixy, the open source prototype implementation of our
178 concepts, is targeted at detecting cross-site scripting vulnerabilities in PHP scripts.

179 The system is capable to cover huge number of vulnerabilities. The system can scan the application code
180 dynamically.

181 11 k) Kaarina Karppinen et al Method

182 These days the big problem is Hidden functionality whereas we cannot be sure that the software does not contain
183 malicious code. Due to architecture violations many security vulnerabilities arises and architecture analysis tools
184 will assist in detecting these vulnerabilities. Such visual images can be used to detect vulnerabilities and ultimately
185 help to design software architectures that meet their security requirements. SAVE [11] is one approach used to
186 detect the violation and what effects the violation had on the system. This kind of analysis with SAVE is new
187 and proving to be advantageous as it adds more details to the evaluation. The SAVE downside is that it is more
188 complex compared to static analysis. The future plans will include developing the SAVE tool further by adding
189 more features, such as automatic comparison of dynamic views and encoding of correct visual images that visual
190 images that together could be used to identify malicious behaviour.

191 12 IV.

192 13 Cross-Site Scripting Attacks

193 This attack can be done on the vulnerable web application to inject the attacker code. Using this attack, an
194 attacker can inject his own code such as javascript into the web application. Some of the results of the Cross
195 Site Scripting attacks are website hacking or web site defacement,. Whenever user requests the hacked website,
196 then the attacker page will be returned. For example NOKIA website is hacked using cross site scripting, In the
197 Hacked time if any user access the NOKIA website, users will get the hackers page,. By this attack an attacker
198 can gain the sensitive information of the website,. And he can disrupt the webservices.

199 14 Example 1

200 In most of the websites, we can see the login and password information. If there are no validations for the user
201 inputs, then the attacker can inject his HTML or SCRIPT code as inputs to the vulnerable pages. By this
202 attacker executes his own script on the server side or the client side. The above script tag is executed in the web
203 server and the result will be submitted to the validate.jsp. if the above script is written for attackers purpose
204 then that will be very dangerous.

205 V.

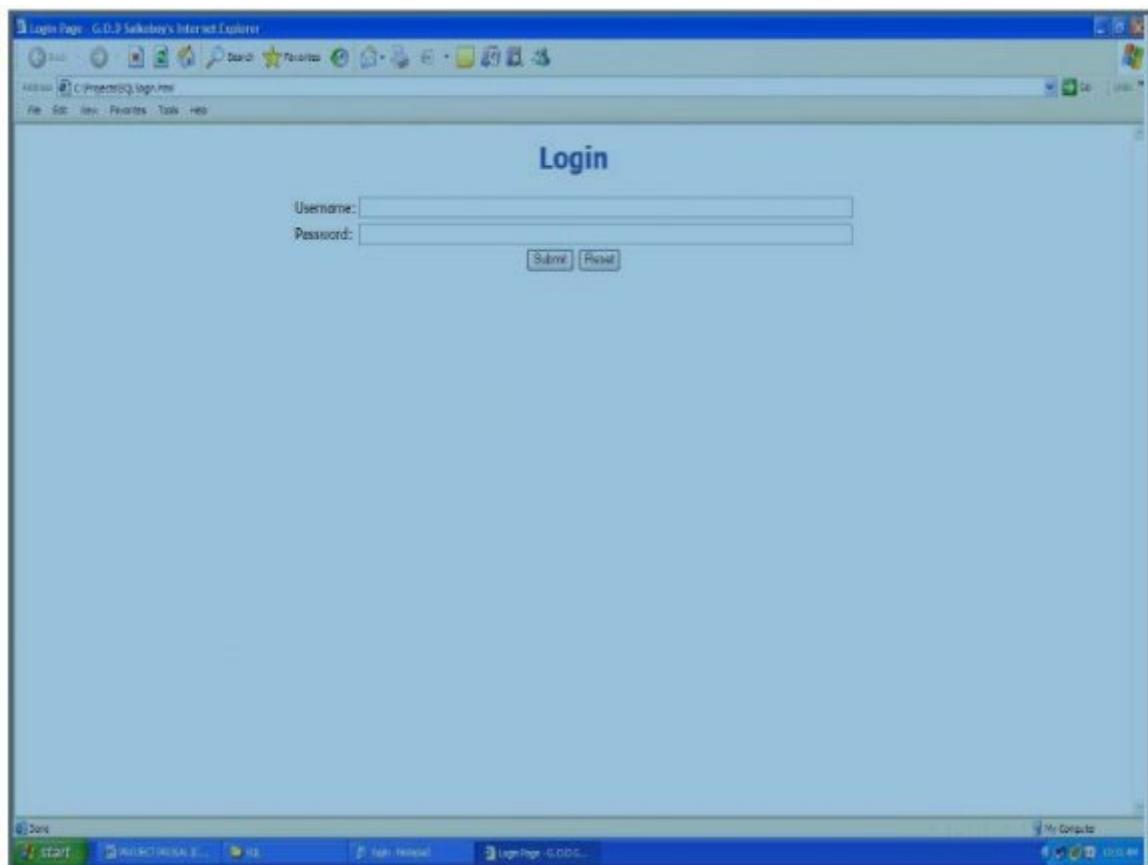
206 15 Conclusion

207 The state-of-art web application input validation techniques fails to identify the proper SQL/XSS Vulnerabilities
208 accurately because of the systems correctness of sanity checking capability, proper placement of valiators on the
209 applications. The systems fail while processing HTTP Parameter pollution attacks. Hence the paper proposes a
210 novel technique called Input Parameter Analysis System (IPAAS). The proposed system works in three phases
211 as Input Parameter Extraction, Parameter Type Learning, and Runtime detection with the learned Parameter



12

Figure 1: SFigure 1 :Figure 2 :



3

Figure 2: Figure 3 :

```
<HTML>
<HEAD>
<TITLE>Login Page</TITLE>
</HEAD>
<BODY>
<FONT Face='tahoma' color='cccccc'>
<CENTER><H1>Login</H1>
<FORM action='validate.jsp' method=post>
<TABLE>
<TR><TD>Username:</TD><TD><INPUT type=text name=username size=100% width=100></INPUT></TD></TR>
<TR><TD>Password:</TD><TD><INPUT type=password name=password size=100% width=100></INPUT></TD></TR>
</TABLE>
<INPUT type=submit value='Submit'> <INPUT type=reset value='Reset'>
</FORM>
</FONT>
</BODY>
</HTML>
```

Figure 3:

```
function Login( cn )
{
var username;
var password;
username = Request.form("username");
password = Request.form("password");
var rso = Server.CreateObject("ADODB.Recordset");
var sql = "select * from users where username = '" + username + "' and password = '" + password + "'";
if( execute_query(sql) !=0 )
{
Return("Welcome Page"); /* Here allows the user as authenticated and returns welcome page */
}
Else
{
Return(" Error Message");
}
```

Figure 4: Defensive

212 Types. Because the system operates on self learning approach, and applies on the HTTP traffic, it reduces the
213 developers or security analysts efforts and increases the chances of attack detection accuracy. ¹ ²

¹© 2014 Global Journals Inc. (US)

²© 2014 Global Journals Inc. (US)World Wide Web and proliferation of web application

214 [Basso et al. ()] 'Analysis of the effect of Java software faults on security vulnerabilities and their detection by
215 commercial web vulnerability scanner tool'. T Basso , P C S Fernandes , M Jino , R Moraes . *International
216 Conference on Dependable Systems and Networks Workshops*, 2010.

217 [Huang et al. ()] 'Countermeasure Techniques for Deceptive Phishing Attack'. H Huang , J Tan , L Liu .
218 *International Conference on New Trends in Information and Service Science*, 2009.

219 [Karppinen et al. ()] 'Detecting Security Vulnerabilities with Software Architecture Analysis Tools'. K Karppinen
220 , M Lindvall , L Yonkwa . *International Conference on Software Testing, Verification, and Validation*, 2008.

221 [Muthuprasanna et al. ()] 'Eliminating SQL Injection Attacks -A Transparent Defense Mechanism'. M
222 Muthuprasanna , K Wei , S Kothari . *International Workshop on Web Site Evolution*, 2006. p. .

223 [PE ()] *Feb) Software assurance tools: Web application security scanner. Functional Specification Version 1.0*,
224 PE . 2008.

225 [Shahriar and Zulkernine ()] 'MUSIC: Mutationbased SQL Injection Vulnerability Checking'. H Shahriar , M
226 Zulkernine . *International Conference on Quality Software*, 2008. p. .

227 [Jovanovic et al. ()] 'Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper)'.
228 N Jovanovic , C Kruegel , E Kirda . *IEEE Symposium on Security and Privacy*, 2006. p. .

229 [Morais et al. ()] 'Security Protocol Testing Using Attack Trees'. A N P Morais , E Martins , A R Cavalli , W
230 Jimenez . *IEEE International Conference on Computational Science and Engineering*, 2009. p. .

231 [Madan and Madan ()] 'Shielding against SQL Injection Attacks Using ADMIRE Model'. S Madan , S Madan .
232 *International Conference on Computational Intelligence, Communication Systems and Networks*, 2009.

233 [Mcclure and Krger ()] 'SQL DOM: compile time checking of dynamic SQL statements'. R A Mcclure , I H Krger
234 . *International Conference on Software Engineering*, 2005. p. .

235 [Khand ()] 'System level security modeling using attack trees'. P A Khand . *International Conference on
236 Computer*, 2009.