Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.336 – 343

RESEARCH ARTICLE

GENERATION OF SECURITY TEST TO FIND INJECTION ATTACKS BY CODE REVIEW

N. Parthiban[1], R. Ravi[2], Dr. Beulah Shekhar[3]

[1]PG Scholar, Department of Network Engineering, Francis Xavier Engineering College, Tirunelveli, Tamil Nadu State, India

[2]Professor & Head, Department of Computer Science and Engineering, Francis Xavier Engineering College, Tirunelveli, Tamil Nadu State, India

[3]Associate Professor, Department of Criminology and Criminal Justice, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu State, India

[1] parthi88@hotmail.com, [2] csehod@francisxavier.ac.in, [3] fxhodcse@gmail.com

Abstract- Security Testing is much important in Software Development to find the vulnerabilities while developing the product. In this paper I presented a method to find the injection vulnerabilities. Vulnerability is a security weakness or it is a hole in the product which is not fixed while developing the software. Normally Hackers hack the software by finding these vulnerabilities and cause data loss or severe damage to the entire product. Injection is one of the most dangerous attacks among much vulnerability which is placed in number one in the top 10 web application vulnerabilities by OWASP. This paper presents how to find these vulnerabilities using Code Review Testing technique. These implementations are used for only educational purpose. It is a crime if you apply these to any website without getting any permission from the owner. If crime is proved, he will be punished under Information Technology (Amendmen) Act, 2008, Section 43(a) read with section 66 is applicable.

Keywords: Injection; Security Testing; SQL Injection; HTML Script Injection; Dynamic Evaluation Injection

I. INTRODUCTION

1.1 Injection

Injection is a security flaw caused by the untrusted input to the interpreter. In other words, the command which works well would behave strange when we give the different input. Normally, the injection attacks found in SQL, HTML and OS commands.

The injection attacks classified into the following types, they are,

- SQL Injection
- HTML Script Injection
- Dynamic Evaluation Injection
- Remote File Injection

These are the attacks which happen when the attacker sends malicious input to the application. The vulnerability in the code has been identified by supplying the malicious input to the code by tester. The behavior of the code will be monitored based on the output of the application. Normally the code would execute when the tester sends the actual input. But the code also executes the malicious input too. Based on the malicious input, the execution of the application causes damage to the entire application.

- *SQL Injection* - SQL Injection happens when the code generates the dynamic SQL Query. Dynamic SQL queries are generated when the user enters the input at run time. Instead of correct input, when the user enter the malicious input the dynamic query executes the malicious operation. It causes data loss or severe damage to the application. This vulnerability is easily identified by monitoring the dynamic generation of SQL code.
- *HTML Script Injection* - Web browsers parse the HTML Script and provide the output as document page from the web server. The web server provides the HTML page as requested from the user. The user can click the HTML Link to go to the other page. The hackers put the HTML code which cause malicious operation into the application. When the victim clicks the link, their Session ID might be stolen.
- *Dynamic Evaluation Injection* - Many operations works based on the input given by the user at run time. But what happens when the attacker gives the unauthorized malicious input to the operation which makes the evaluation dynamically?
- *Remote File Injection* - Remote File Injection is implemented by the attacker by modifying the original file with the malicious file in the Web application.

II. METHODS

Many Techniques used to find the Vulnerabilities in the Web Applications. They are,

i. Manual Process
ii. Thread Model
iii. Code Review
iv. Penetration Testing

Manual Process is applied at the initial stage of the Software Development Life Cycle (SDLC). It contains the Documentation Model of the whole product. The vulnerability has identified from the expected operation of the application. Thread Model is also applied at the initial stage of the SDLC. It represents some predefined attacks to find the weakness. Code Review is a testing technique which is applied to the source code of the application. It check the behavior of the code when executes. This is the one of the best method in security testing. By using Code Review we can find much vulnerability possible. Normally Code Review is difficult to apply in the Software Development Life Cycle (SDLC) while developing the product because testing applied at the end of the SDLC process. Testers are not allowed to access the source code in some project. This makes very hard for the Security Testers. Penetration Testing is the art of 'ethical hacking'. It has been applied at the implementation of the product.

III. IMPLEMENTING INJECTION ATTACKS

First of all, we need a web application to implement those attacks. So I created the E-Banking Web Application. Fig.1 shows the operations of the E-banking Web Application.

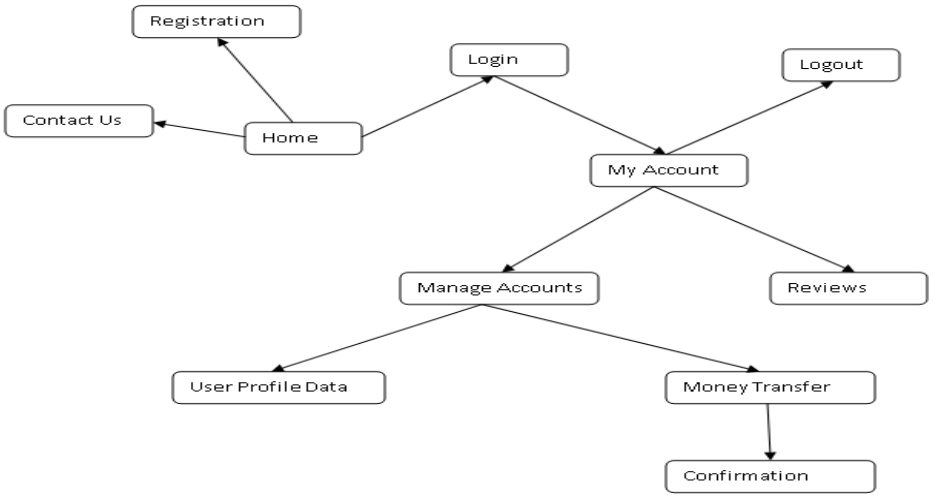

Fig1 E-Banking Web Application

By using Code Review Testing Technique, we have to analyze the source code for the vulnerabilities. Code Review technique is applied while developing the software product.

3.1 FINDING SQL INJECTION VULNERABILITY

Mostly many web applications have the authentication to the users for accessing those services. Our application also has the authentication to allow only the registered users. For example, In Fig 2, our code for the authentication is as follows,

```
ResultSet r=st.executeQuery(
        "select * from registration where email='"+
        request.getParameter("uname")+"'"+"AND password='"
        +request.getParameter("psw")+"'");
```

Fig 2 Dynamic Generation of SQL Query

In our code, the email and password has been checked for the registered table 'registration'. If the password and email matches from the table, the application allows the user to access the services. The flaw in our application is, it generates the SQL Query dynamically. The email and password is checked by the SQL Query 'AND' at run time. The Dynamic Generation of the SQL Query allows the attacker to implement the malicious code like "abc' OR 1=1--". Figure 3 shows how the SQL Injection attack has been done. If the attacker knows only the username, he can put the above malicious query to the password field to make the SQL Injection. For example, the username is 'parthi88@hotmail.com' and the password is 'abc' OR 1=1--'. The Code executes the password field as true.

Because 1=1 is always true. The operation is done by the 'OR' SQL Command. It is the logical operation which makes the SQL command true.

Fig 3 SQL Injection Attack

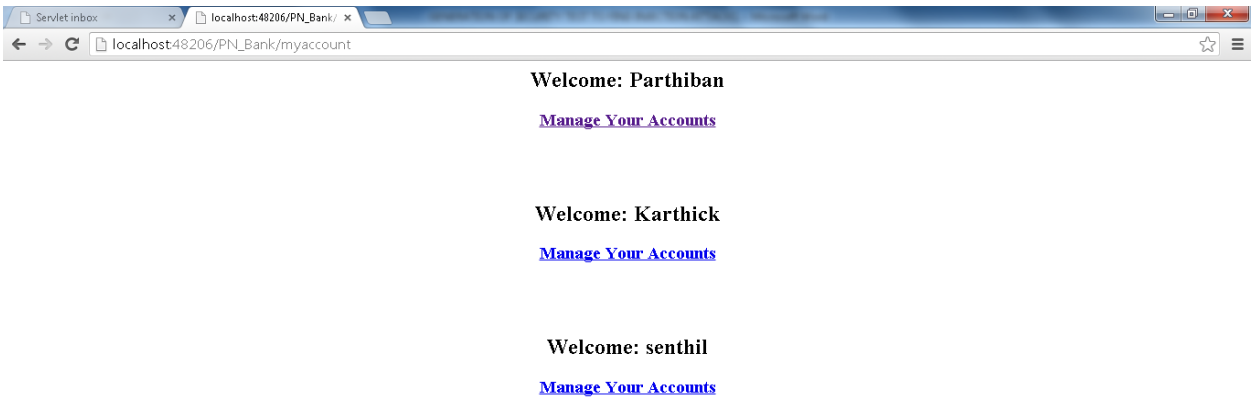

Fig 4 Result of SQL Injection

In Fig 4, While executing the username 'parthi88@hotmail.com' with the password 'abc' OR 1=1--', the code displays all the users from the database.

3.2 Finding Html Script Injection

Review is the operation in our E-banking Websites. Like a Twitter, any users can post a message in the review page to other users. This is the logical flaw in our E-banking Application because the attacker could be a user who can put any malicious post to the review page. If the victim clicks the link the malicious post can send the user to any harmful sites. Fig 5 shows the review page.

Fig 5 Review Page

From this, the attacker sends the malicious link with the fascinating title like 'Offer' and Hurry. If the victim clicks the link, the malicious page could steal the session ID or do any other harmful operations to the victim.

Title: Hurry!!!!! Offer.....

comments This offer is only valid for today... Hurryyyyyyyyyyyyy....

More Details...

Fig 6 HTML Script Injection

Fig 6 shows one message to the victim from the review page. When the victim clicks the 'More Details....' he dragged to the malicious website to steal the session id.

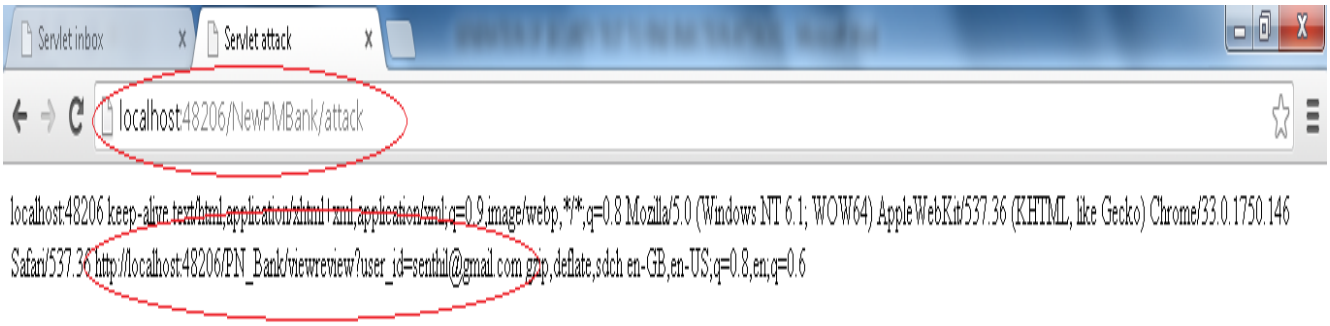

Fig 7 Result of HTML Script Injection

After the Victim clicks the malicious link, his session id has been store to the new malicious site. Fig 7 shows the result of HTML Script Injection

3.3 Dynamic Evaluation Injection

When the application calculates the output at the runtime, the attacker can send their own input to make the malicious operations. For example, when our E-Banking Web Application sends the money from one user to another user, the attacker can modify which user can send and which user can receive. Fig 8 shows the Dynamic Evaluation while transferring money.

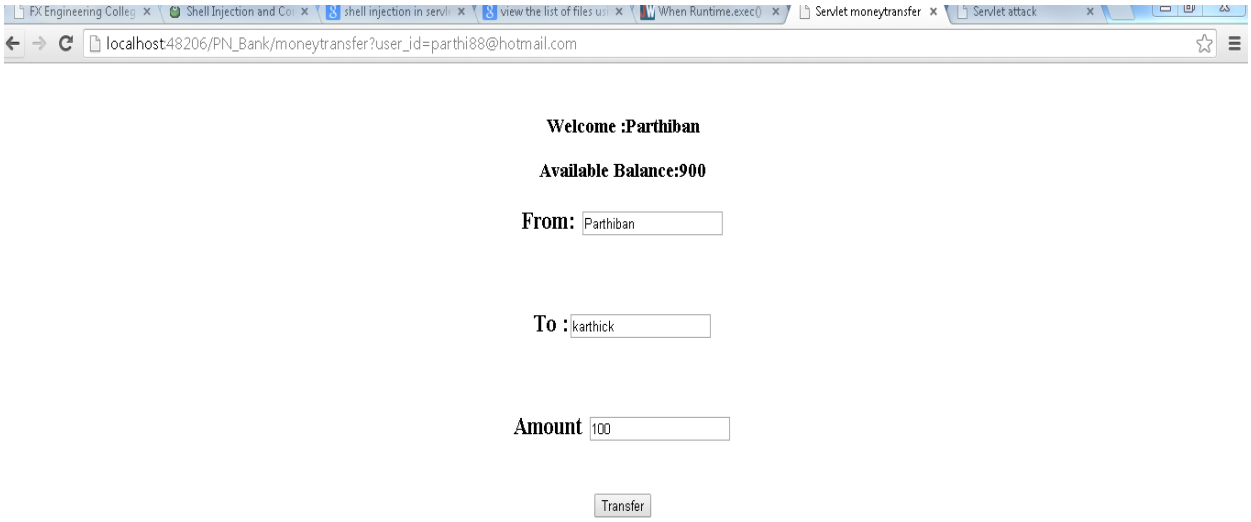

Fig 8 Money Transfer

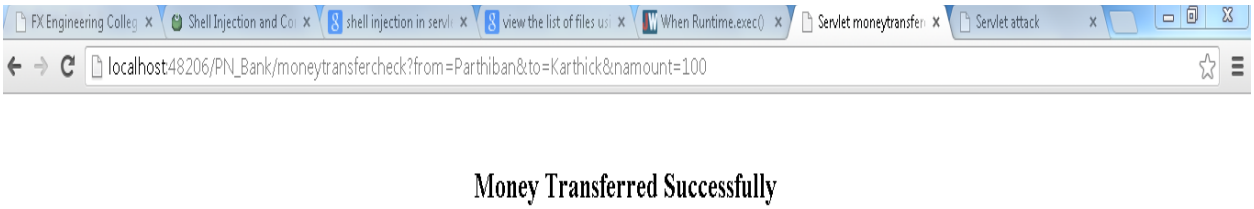

Fig 9 HTTP Request shows the operation.

In Fig 9, The address bar from the browser shows the internal operation and the variables used for operation. This is flaw from bad coding. Because when we send the sensitive information from the browser to server, we should not use the HTTP GET method.

Now the attacker can get the link from the browser History, and modifies the sender or receiver from the browser address bar. If the attacker modifies the sender from 'Parthiban' to 'Senthil' this application still works and sends the money 100 from Senthil to Karthick. We don't need to know the Senthil Account Password, but we must know the account name of the sender. Fig 10 shows the result of Dynamic Evaluation Injection

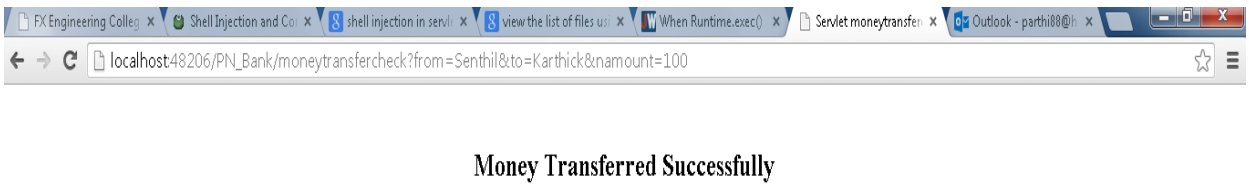

Fig 10 Result of Dynamic Evaluation Injection

3.4 Finding Remote File Injection Vulnerability

When the server sends the file to the client through HTTP Protocol, Remote File Injection is Possible. In our E-banking Example, our Branch addresses are stored in file. When the client tries to access the branch address, the html file which contains the address has sent to the client.

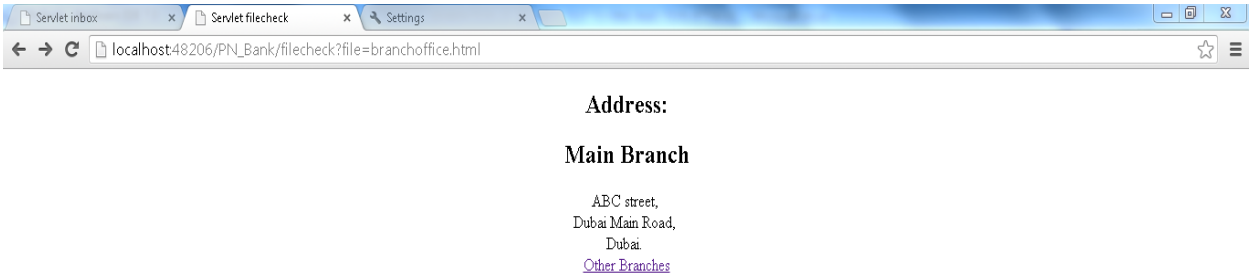

Fig 11 Reason for Remote File Injection

In Fig 11 , the browser address bar contains the file name 'branchoffice.html' which is going to execute when the user clicks the 'Other Branches' link. The attacker uses this link and uses his own file which is going to execute in the server. Fig 9 shows the result of File Injection Vulnerability.

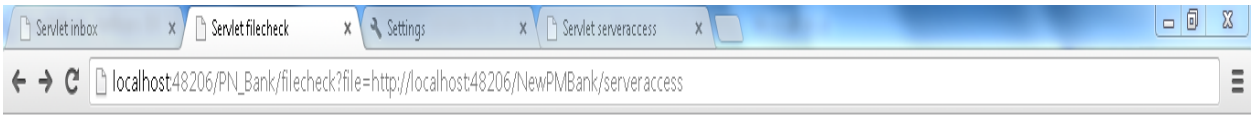

Fig 12 Result of Remote File Injection

The attacker uses his own file 'http://localhost:48206/NewPMBank/serveraccess' to execute into the E-Banking Server. Fig 12 shows the result of Remote File Injection.

IV. CONCLUSION

The attacks defined in this paper are only considered for the Educational purpose. We can use these methods to find the web application vulnerabilities. The Source Code Review Testing method is only applied while developing the product. By analyzing the Code We can find much vulnerability than any other testing methods. The Injection attack is placed in top 10 web application security weakness by OWASP. The SQL Injection breaks the database of the whole product. HTML Script Injection drags the victim to the malicious web sites by sending the malicious link from the attacker. Dynamic evaluation Injection method calculates the normal input from the user and also calculates the malicious input from the attacker. It makes severe damage to the whole application. Remote File Injection executes the malicious file in the own server. It may cause severe server damage. All Attacks are done through the HTTP Protocol. The attacker will be punished for imprisonment, which may extend to three years or fine with five lakh rupees under section 43(a).

REFERENCES

[1] Dianxiang Xu, Michael Sanford, Lijo Thomas, Daniel Woodraska and Weifeng Xu " Automated Security Test Generation with Formal Threat Models" IEEE Trans. On Dependable and Secure Computing, Vol.9, No.4, July/August 2012

[2] Elizabeth Fong and Vadim Okun " Web Application Scanners: Definitions and Functions" Proc. Of 40th Inter. Conf. 2007

[3] Chia-Mei Chen, Wan-Yi Tsai and Hsaio-Chung Lin "Anomaly Behavior Analysis for Web Page Inspection" Intr. Conf on Networks & Communications 2009

[4] Avinash Kumar Singh and Sangita Roy " A Network Based Vulnerability Scanner for Detecting SQLI Attacks in Web Applications Int'l Conf On Recent Advances in Information Technology RAIT 2012

[5] https://www.owasp.org/index.php/Main_Page

[6] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[7] http://www.csi-india.org/

ABOUT THE AUTHORS

N. Parthiban was born in 1988. He is currently a student of Francis Xavier College of engineering and technology, Tirunelveli. He is doing his masters in Network engineering. He received his bachelors in Computer Science and Engineering from Shri Andal Alagar College of engineering and technology in 2010. His area of interest is network security and his subject of interest is Security Testing.

R. Ravi is an Editor in International Journal of Security and its Applications (South Korea). He is presently working as a Professor & Head and Research Centre Head, Department of Computer Science and Engineering, Francis Xavier Engineering College, Tirunelveli. He completed his B.E in Computer Science and Engineering from Thiagarajar College of engineering, Madurai in the year 1994 and M.E in Computer Science and Engineering from Jadavpur Government research University, Kolkatta. He has 18 years of experience in teaching as Professor and Head of department in various colleges. He published 12 International Journals, 1 National Journal. His areas of interest are Virtual Private networks, Networks, Natural Language Processing and Cyber security.

Dr. Beulah Shekhar is a Coordinator for Victimology & Victim Assistance, in the Department of Criminology and Criminal Justice Sciences; she is presently working as a Associate Professor in the Department of Criminology and Criminal Justice Sciences. And her areas of interest are Crimes against Women Empowerment, Human Rights, and Police Training.

