Detection And Prevention of A Buffer Overflow Attacks Based on Hybrid Approach

V.Nithya[1], R.Regan [2]
[1,2] Department of Computer Science and Engineering
[1] University College of Engineering Panruti, India, [2]University College of Engineering Villupuram, India.

Abstract- *Buffer overflows vulnerabilities to compromise critical data structures. We present a black-box testing approach to detecting buffer overflow vulnerabilities. Our approach is motivated by a reflection on how buffer overflow vulnerabilities are exploited in practice. In most cases the attacker can influence the behavior of a target system only by controlling its external parameters. Therefore, launching a successful attack often amounts to a clever way of tweaking the values of external parameters. We simulate the process performed by the attacker, but in a more systematic manner. In particular, our approach exploits the fact that combinatorial testing often achieves a high level of code coverage. We have implemented our approach in a prototype tool called Trance. The results of applying Trance to five open-source programs show that our approach can be very effective in detecting buffer overflow vulnerabilities.*

Keywords- Black box testing, Buffer over flow Attack, Hybrid approach

I. INTRODUCTION

In computing, a web application or web app is a client-server software application in which the client runs in a web browser. A Web application is an application program that is stored on a remote server and delivered over the Internet through a browser interface. Examples of browser applications are simple office software (word processors, online spreadsheets, and presentation tools), but can also include more advanced applications such as project management, computer-aided design, video editing and point-of-sale. At a high level, web application security draws on the principles of application security but applies them specifically to Internet and Web systems [1]. One of the most serious input hacks is a buffer overflow that specifically targets input fields in web applications. If someone managed to exploit a buffer overflow in a Web application, it would result in a critical situation.

A. Buffer over flow Attack

By this attack, we are trying to get past Client-Side Validations which come in effect due to the usage of Web-Browser. Since, we are not working with Browser but directly manipulating the HTML Source file we are able to bypass the Client Side Validations such as Java script sand “MAXLENGTH" field present in “input" tag fields. The “input" tags fetched from the HTML source file are embedded with an arbitrary long String in the„ value" attribute and resubmission occurs. The random

String is generated by a „randomizer" function. If the Server does not support proper Server-Side Validation, then a possible crash can take place at the Server end. There is noted by observing the HTTP Status codes received as a response whether the attack was a success or not.

II. LITERATURE SURVEY

A. Loop-extended symbolic execution and generalization

Shahriar, H., Haddad, H.M., Vaidya describes Mixed concrete and symbolic execution is an important technique for finding and understanding Software bugs, including security relevant ones. However, existing symbolic execution techniques are limited to examining one execution path at a time, in which symbolic variables reflect only direct data dependencies. We introduce loop-extended symbolic execution, a generalization that broadens the coverage of symbolic results in programs with loops. It introduces symbolic variables for the number of times each loop executes, and links these with features of a known input grammar such as variable-length or repeating fields. Our tool finds vulnerabilities in both a standard benchmark suite and 3 real-world applications, after generating only a handful of candidate inputs, and also diagnoses general vulnerability conditions [2].

B. Buffer over flow vulnerabilities at run time

Charier, H., Zulkernine, M states Buffer over flow program defects that can cause a buffer to overflow at runtime. Many security attacks exploit buffer overflow vulnerabilities to compromise critical data structures. In this paper, we present a black-box testing approach to detecting buffer overflow vulnerabilities. Our approach is motivated by a reflection on how buffer overflow vulnerabilities are exploited in practice. In most cases the attacker can influence the behavior of a target system only by controlling its external parameters. Therefore, launching a successful attack often amounts to a clever way of tweaking the values of external parameters [3].

C. Fizzing a black box and white box testing

Padmana bruin, B.M., Tan Many describes that security attacks exploit buffer overflow vulnerabilities to compromise critical data structures, so that they can influence or even take control over the behavior of a target system. Our approach is a specification-based or black-box testing approach. That is, we generate test data based on a specification of the subject program, without analyzing the source code of the program. The specification required by our approach is lightweight and does not have to be formal. In contrast, white-box testing approaches derive test inputs by analyzing the data and/or control structure of the source code. Black-box testing has the advantage of requiring no access to the source code, and is widely used in practice[4].

D. Buffer over flow attack parameters for using security testing

Wagner, D., Foster, J.S., Brewer, E.A.,et al. states Security testing essentially needs to do the same thing, but in a more systematic manner and with a good in tents as an effort to validate this hypothesis, we inspected buffer overflow vulnerability reports in three public databases[5].

E. Single path loop using buffer overflow attack

Saxena, P.Poosankam, P., McCamant, S describes, when single-path symbolic execution is applied to test case generation to increase coverage, it will be unable (in one iteration)to generate an input that forces execution down a different branch than in the original execution, if taking that branch is only feasible with a different number of loop iterations. In other words, in single-path symbolic execution, the values of a symbolic variable reflect only the data dependencies on the symbolic inputs control dependencies, including loop dependencies, are ignored [6].

III. PROPOSED SYSTEM

There are more than million websites on the Internet. With such at remand ours growth, the issue of security has achieved a wide angle and is very important due to the following reasons:

Most of the transactions are Online

Usage of Legacy

User Trust factor

Shift in focus of Attacker towards monetary benefits

poorly written code

There are many more reasons besides these. The security can be maintained by having Secure Coding practice. But, that is not always the case. Hence, there is a need for an application tool which would be able to uncover vulnerabilities besides those which are already well-known. If such vulnerabilities are uncovered, then the security of the product can been hence and guaranteed to a large extent. Software Testers would benefit tremendously from such a utility. We look into the following categories of errors:

Failure to handle exceptions

Failure to validate input on server

A. BOF Detector BOF Detector is a Web Site vulnerability detector which rigorously injects malformed data and SQL injections within the input tag fan HTML source file fetched from the Web Server. It then analyzes the HTTP status codes received as a response from the Web Server as a possible indication for hidden vulnerabilities. There are many BOF already present in the market written in Pearland Python.BOF Detector, written exclusively in Java is going to be very popular and can be used by Web Portal Testing teams before putting their Site into Production.

BOF Detector is a context-aware type of BOF whereby it is aware of the HTML source file to be BOF Our projects cope is limited to dealing with ”input “ tag expressions. Steps involved in the case of BOF Detector are as follows:

BOF Detector fetches a HTML source file from a URL using HTTP Connection class.

BOF Detector traverses through the HTML Source file and populates a List containing all the different types of input commands corresponding to a particular form on a Web Page file.

Random strings are generated and embedded into the„ input‟ tags in the ”value" field Attribute.

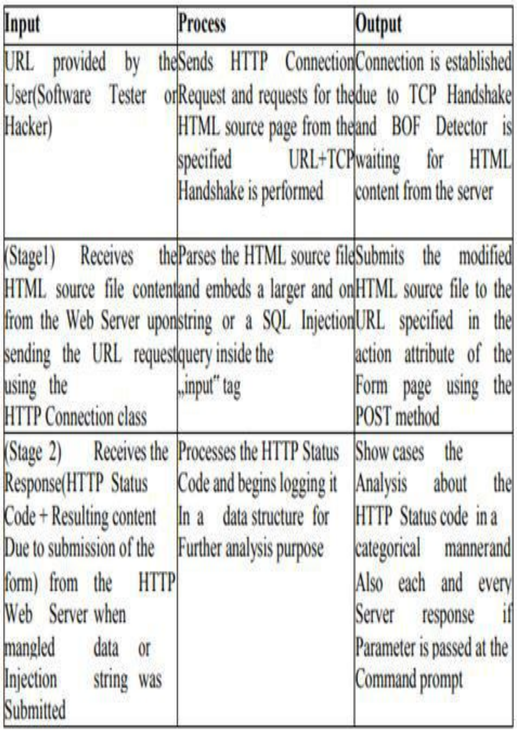
Page is resubmitted to the„ action ‟field specified in the form tag.

There sponge received from the Server is in the form of HTTP Status Codes and is used for analysis regarding the vulnerability status of a Web Page B. Functional Matrix

It basically consists of 3 columns comprising of Input, process and output. The input at each entity undergoes certain processing which is displayed at the output. Also, it helped using a the ring the complete know-how about the features and concepts pertaining to each entity and their requirements.

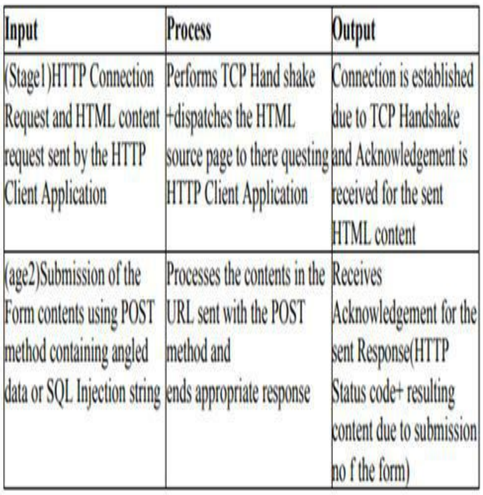
<Tabulation done on basis of Input to Actors involved>

Table 1 Input to BOF Detector



Input	Process	Output
URL provided by the User(Software Tester or Hacker)	Sends HTTP Connection Request and requests for the HTML source page from the specified URL+TCP Handshake is performed	Connection is established due to TCP Handshake and BOF Detector is waiting for HTML content from the server
(Stage1) Receives the HTML source file content from the Web Server upon sending the URL request using the HTTP Connection class	Parses the HTML source file and embeds a larger and on string or a SQL Injection query inside the „input" tag	Submits the modified HTML source file to the URL specified in the action attribute of the Form page using the POST method
(Stage 2) Receives the Response(HTTP Status Code+Resulting content Due to submission of the form) from the HTTP Web Server when mangled data or Injection string was Submitted	Processes the HTTP Status Code and begins logging it In a data structure for Further analysis purpose	Show cases the Analysis about the HTTP Status code in a categorical mannerand Also each and every Server response if Parameter is passed at the Command prompt

Table 2 Input to HTTP Web Server



Input	Process	Output
(Stage1)HTTP Connection Request and HTML content request sent by the HTTP Client Application	Performs TCP Hand shake +dispatches the HTML source page to there questing HTTP Client Application	Connection is established due to TCP Handshake and Acknowledgement is received for the sent HTML content
(age2)Submission of the Form contents using POST method containing angled data or SQL Injection string	Processes the contents in the URL sent with the POST method and ends appropriate response	Receives Acknowledgement for the sent Response(HTTP Status code+ resulting content due to submission of the form)

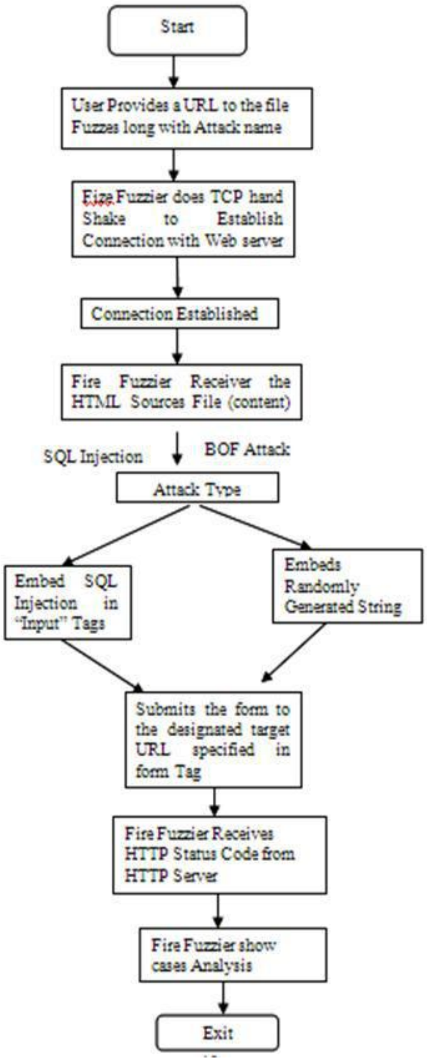

Start
User Provides a URL to the file Fuzzes long with Attack name
Fire Fuzzier does TCP hand Shake to Establish Connection with Web server
Connection Established
Fire Fuzzier Receiver the HTML Sources File (content)
SQL Injection
BOF Attack
Attack Type
Embed SQL Injection in "Input" Tags
Embeds Randomly Generated String
Submits the form to the designated target URL specified in form Tag
Fire Fuzzier Receives HTTP Status Code from HTTP Server
Fire Fuzzier show cases Analysis
Exit

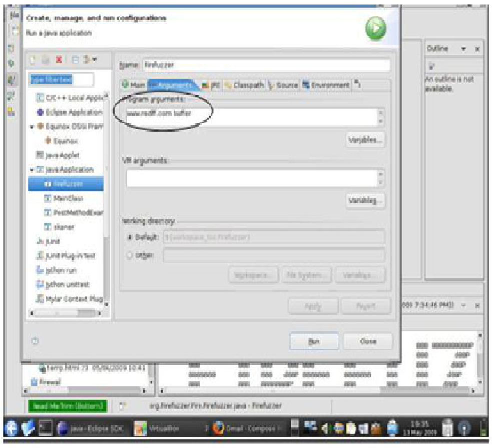
Fig: 1 Flow diagram for Buffer overflows Attack Detection

IV. EXPERIMENT RESULT

Step1: We wish to perform buffer overflow

http://www.rediff.com/
Gmail - Compose Mail - gau...
Welcome to rediff.com
THE FINEST NEW HOMES ON THE RIVER
Click HERE to learn more
HENLEY ON HUDSON
rediff INDIA ABROAD
Web Images Air Tickets Jobs in India Stocks LocalAds More..
Search the web:
Search
Search today - Swine Flu, IPL, Daisy Bopanna, Angels & Demons, Astrology
News Stocks Live! Election 09 IPL 2 Tata Nano Videos
Connect Videos Blogs Messenger Get Answers Tool bar
LocalAds Education Travel Jobs more.. Post Ad
Shop Books Flowers Gift Center Jewellery Movies
India Abroad Classifieds/Submit Events Person of the Year Subscribe/Gift This week's issue
» view all services
Rediffmail Free unlimited storage
Username Password
Go
New User? Sign Up
Intersections Matchmaking
Know more ▸
Catch all the action on Indian Elections
Sponsored Links
Featured Users
Get Answers
Can Yoga be an effective cure for insomnia?
Answer this question
Headlines News Business Movies Sports Videos
May 13, 2009 1915 EDT
Exit polls predict slender Cong advantage | More
Delhi beat Hyderabad | Report | More
Aravind Adiga: A head start for India's next PM
HOT ONLINE DEAL
2 MEDIUM CHEESE PIZZAS
ORDER NOW
$9.99
GIFT CENTER
Hotel Search

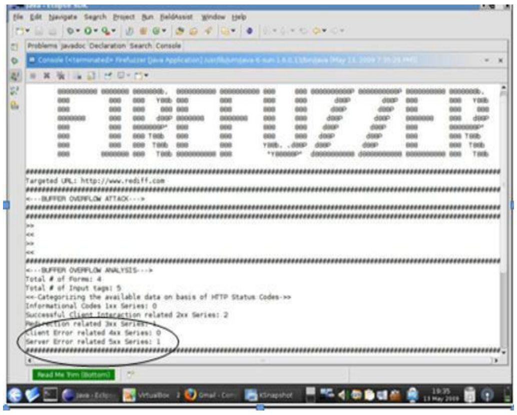
Step: 2 start the run configuration in Eclipse



Create, manage, and run configurations
Run a Java application

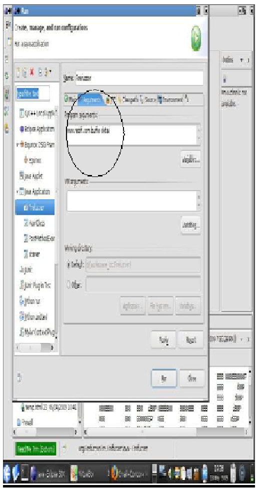
Name: Firefuzzer
Program arguments: www.rediff.com buffer

Step3: We choose to perform Buffer Overflow by using the parameter ‘buffer’



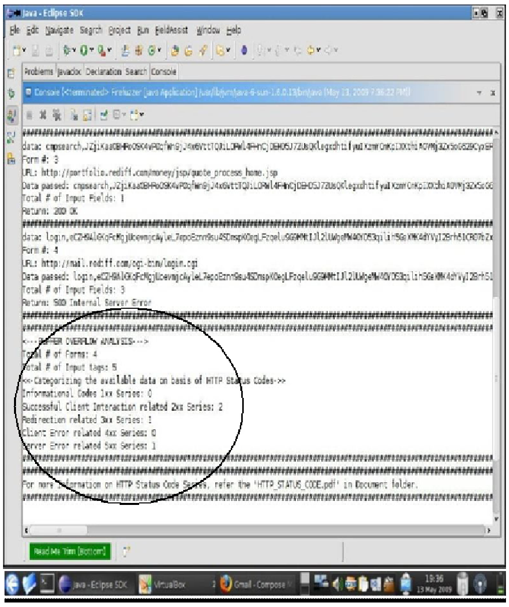
```
#############################################################################
Targeted URL: http://www.rediff.com
#############################################################################
<---BUFFER OVERFLOW ATTACK--->
#############################################################################
#############################################################################
>>
<<
>>
<<
#############################################################################
<---BUFFER OVERFLOW ANALYSIS--->
Total # of Forms: 4
Total # of Input tags: 5
<<-Categorizing the available data on basis of HTTP Status Codes->>
Informational Codes 1xx Series: 0
Successful Client Interaction related 2xx Series: 2
Redirection related 3xx Series: 1
Client Error related 4xx Series: 0
Server Error related 5xx Series: 1
#############################################################################
```

Step 4: The detail view in Buffer Overflow by using ‘detail’ parameter



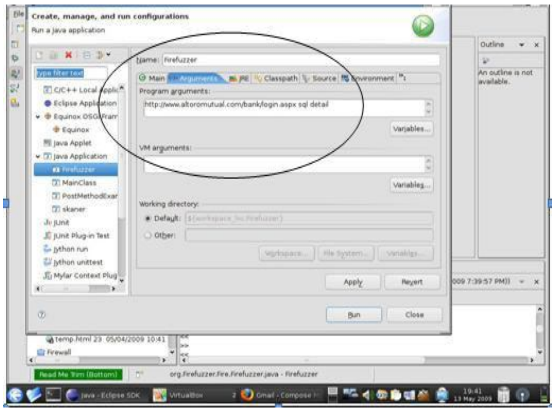
Name: Firefuzzer
Program arguments: www.rediff.com buffer detail

Step 5: Observe the Server responses shown in detail view



```
#############################################################################
data: cmpsearch,JZjiKaaOBHRoO9K4vPOqfWn9jJ4x6VttTQJiLOPWl4FHnCjDEHO5J72UsQKlegxdhtifyuIXzmYOnKpIXXthiAOVMj3ZxSoG6Z9CypEP
Form #: 3
URL: http://portfolio.rediff.com/money/jsp/quote_process_home.jsp
Data passed: cmpsearch,JZjiKaaOBHRoO9K4vPOqfWn9jJ4x6VttTQJiLOPWl4FHnCjDEHO5J72UsQKlegxdhtifyuIXzmYOnKpIXXthiAOVMj3ZxSoG6
Total # of Input Fields: 1
Return: 200 OK
#############################################################################
data: login,eCZH9AlGKqFcMgjUcevmjcAyleL7epoEznn9su4SDmspKOegLFzqeluSGGMMtIJl2lUWgeMW4OYO53qilih5GaXMK4dYVyI2Brh51CRO7b2x
Form #: 4
URL: http://mail.rediff.com/cgi-bin/login.cgi
Data passed: login,eCZH9AlGKqFcMgjUcevmjcAyleL7epoEznn9su4SDmspKOegLFzqeluSGGMMtIJl2lUWgeMW4OYO53qilih5GaXMK4dYVyI2Brh51
Total # of Input Fields: 3
Return: 500 Internal Server Error
#############################################################################
#############################################################################
<---BUFFER OVERFLOW ANALYSIS--->
Total # of Forms: 4
Total # of Input tags: 5
<<-Categorizing the available data on basis of HTTP Status Codes->>
Informational Codes 1xx Series: 0
Successful Client Interaction related 2xx Series: 2
Redirection related 3xx Series: 1
Client Error related 4xx Series: 0
Server Error related 5xx Series: 1
#############################################################################
#############################################################################
For more Information on HTTP Status Code Series, refer the 'HTTP_STATUS_CODE.pdf' in Document folder.
#############################################################################
```

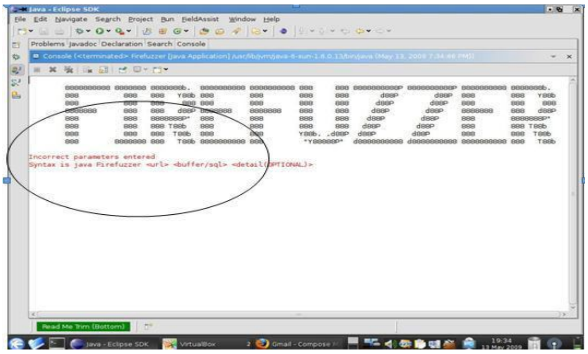
Step 6: Final Analysis in Buffer Overflow done using detailed view



Create, manage, and run configurations
Run a Java application

Name: Firefuzzer
Program arguments: http://www.altoromutual.com/bank/login.aspx sql detail

Step 7: Validation we wish to pass ‘buff’ which is a wrong parameter to the Program (correct should be ‘buffer’)



```
Incorrect parameters entered
Syntax is java Firefuzzer <url> <buffer/sql> <detail(OPTIONAL)>
```

V. CONCLUSION

BOF Detector show cases the vulnerabilities in typical Websites. As per our tests, we have proven that if thus vulnerabilities come in the knowledge of Attackers then Exploitation of vulnerabilities will not take much time it also show cases then need for much improved and secure coding standard. Even though as secure coding mode exist Security Development Life Cycle, it is still being implemented in a phased manner. That might be because corporation shave not realized it s importance yet.

But, it is indeed time to use penetration testing tools such as BOF Detector and the other likes which do exist to become aware of the vulnerabilities be for putting the system in to production environment.

REFERENCES

[1] V. Nithya, R. Regan, and J. Vijayaraghavan, "A survey of SQL injection attacks, their Detection and Prevention techniques," International Journal of Engineering and Computer Science(IJECS), vol. 2, no. 4, April 2013.

[2] Shahriar, H., Haddad, H.M., Vaidya, I.:"Buffer overflow patching for C and C++ programs: rule-based approach", SIGAPP Appl. Compute. Rev., 2013, 13

[3] Charier, H., Zulkernine, M.: 'Mutation-based testing of buffer overflow vulnerabilities'. Proc. IEEE Int. Computer Software and Applications Conf.2008, pp. 979–984

[4] Padmana bruin, B.M., Tan, H.B.K.: "Auditing buffer overflow vulnerabilities using hybrid static–dynamic analysis". Proc. IEEE Int. Computer Software and Applications Conf., 2014, pp. 394–399.

[5] Wagner, D., Foster, J.S., Brewer, E.A.,et al.: "A first step towards automated detection of buffer overrun vulnerabilities". Proc. Network and Distributed System Security Symp., 2000, pp. 3–17

[6] Saxena, P.Poosankam, P., McCamant, S., et al.: 'Loop-extended symbolic execution on binary programs'. Proc. Int. Symp. On Software testing an analysis, 2009, pp. 225–325

[7] V Nithya, SL Pandian, R Regan ,The SQL Injection Attack Detection and Prevention by Classification and Analysis Asian Journal of Information Technology 12 (4), 131-139,2013

[8] Cowan, C., Pu, C., Maier, D.,et al.:'StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks'. Proc. USENIX Security Symp., vol. 7, 1998

[9] Brumley, D., Newsome, J., Somg, D.,et al.: 'Towards automatic generation of vulnerability-based signatures'. Proc. IEEE Symp. Security and Privacy, 2006.

[10] Zitser, M., Lippmann, R., Leek, T.:'Testing static analysis tools using exploitable buffer overflows from open source code'. Proc. Int. Symp. on Foundations of Software Engineering, 2004, pp. 97–106.