Test Harness for Web Application Attacks

Kishan Chudasama[1] Mr. Girish Khilari[2] Mr. Suresh Sikka[3]
[1]GTU PG School, Gandhinagar [2,3]CDAC ACTS, Pune

Abstract— Web application security is major concern these days. There are various attacks listed by OWASP and provide Top 10 List over period of time. Preventing from those attacks is real challenge so identifying those attacks are challenge so proposing the test environment to identifying attacks like SQL injection, LDAP injection, OS command injection for the web application or OS. The result will be useful for identifying root cause of security incident and creating self-aware security infrastructure.

Key words: Real-time Testing, Test Harness, Web Application Attacks, Web Application Security

I. Introduction

Web application testing involves following testes: Functionality, Usability, Interface, Compatibility and Performance.

Security testing involves testing of unauthorized data theft and unauthorized access. It ensures security against following: Injection [1], Broken Authentication and Session Management [1], XSS [1], Insecure Direct Object References [1], Security Misconfiguration [1], Sensitive Data Exposure [1], CSRF [1], Using Components with Known Vulnerabilities and Invalidated Redirects and Forwards [1].

II. Test Harness

According to automation testing phenomenon, Test harness refers to the framework and the software systems that contain the test scripts, parameters necessary to run these scripts, gather test results, compare them and monitor the results [1].
It executes the tests using test library and at the end it generates a report.

III. Test harness framework

A. *Configurator [2]*

It provides more flexibility to harness. Configurator, for initial settings processes the XML file. To find out individual test level information it loads tests from the XML file and processes it. Generally, a XML file contains following: style of test, name of test, conditions for the input or the output and the number of threads assigned for the simulation of more than one users.

B. *Runtime Engine [2]*

Runtime engine is the heart of the test harness. It basically acts as a controller that administers the other subsystems, starts the harness, initializes the whole system, manages the threads being spawned, and co-ordinates interactions necessary among all the subsystems.

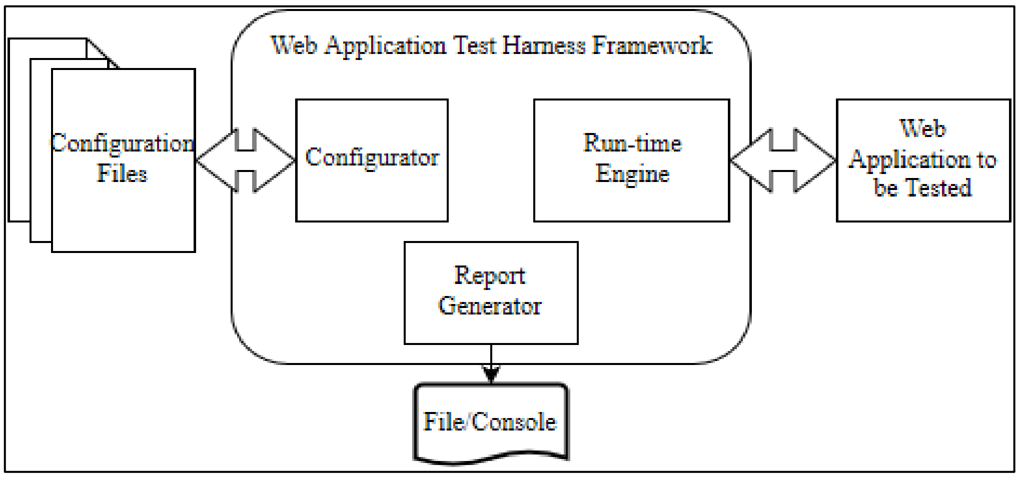


Fig. 1: Web Application Test Harness Framework

C. *Report Generator [2]*

Report generator, generates the test result, extracts the required information and reports it to a file or console, or both, as per requirement.

IV. Web Application Threat Modelling

Threat modelling involves analyses of the security of an application. It is a structured approach to first of all identifying the security risks, quantifying them and finally addressing them with an application.

A. Spoofing [3]

To spoof means to pretend fake identity of a user or a process in unauthorized manner. Spoofing can be just using some other person's credentials to do some actions. A malicious use could involve changing of a cookie to pretend identity of other user or it might be showing that the cookie comes from other server.

To prevent from spoofing it requires authenticating the users hardly or making the authentication system even harder to bypass.

B. Tampering [3]

Tampering involves changing or deleting of a resource without having proper permissions. To avoid this type of attack the application should be running in lowest privilege.

C. Repudiation [3]

A repudiation threat involves carrying out a transaction in such a way that there is no proof after the fact of the principals involved in the transaction. In a Web application, this can mean impersonating an innocent user's credentials. To prevent from this logging could be a possible solution for auditing.

D. Information Disclosure [3]

Information disclosure means revealing information that is private. Proper authentication system can reduce the information disclosure attempts.

E. Denial of Service [3]

It reduces the availability of the web application for the users. There are various ways to do so. To prevent from these attacks proper Firewall rules can be able to find the attack and to block certain ip addresses.

F. Elevation of Privilege [3]

It involves elevation of privileges that are not normally assigned. It generally involves getting administrator privileges for the web application to harm the information system or to gain access to certain level of information. To avoid it the application should run at it minimum required privilege.

V. Web Application Attacks and Countermeasures

The public interfaces exposed to the attacker by an application and if the network and host are fully secured than the application becomes the source of attack.

A. Buffer Overflow [4]

Buffer overflow vulnerabilities involve DoS attack or code injection attacks. It involves crashing of the system by crashing a process and injecting with the malicious code that can harm the entire system.
Consider following example

```
\\Buffer overflow example
void Some_Function( char *Input )
{
   char sizeBuffer[12];
   // Here without type checking input is directly copied to buffer
   strcpy(sizeBuffer, Input);
   . . .
}
```

1) Countermeasures to help prevent buffer overflows include

Best approach to avoid this type of attack is to validate the input. Applying certain constraints to the inputs like specifying type, length, format and range of the input.

By inspecting the managed code that is calling to the un-managed API that only appropriate values are being passed as parameters ensures prevention from this type of attack.

B. Cross-Site Scripting [4]

In this type of attack the victim's browser is used to perform the attack while it is connected to trusted site. Here the arbitrary code runs in browser.
Consider following link for understanding scenario.
Here is a legitimate link:
www.yourwebapp.com/x=bobthebuilder
Here is a malicious link:
www.yorwebapp.com/x=<script>alert(malicious code here')</script>

If the web application accept the above script and could not able to validate it and return it to the browser then in that case the script code will execute on the browser.

1) Countermeasures to prevent XSS include:

Input validation is an approach to minimize the risk from this type of attacks. To convert script input into harmless HTML use HTMLEncode() and URLEncode() functions.

C. SQL Injection [4]

Application code is vulnerable that constructs dynamic SQL statements with unfiltered user input.

SqlDataAdapter myCommand = new SqlDataAdapter("SELECT * FROM Usrs WHERE UsrNm ='" + txtuid.Text + "'", conn);

Consider the following character string entered into the txtuid field.
'; DROP TABLE Cust -

When the following statement is resolved the result is submitted to database for the execution.

SELECT * FROM Usrs WHERE UsrNm=''; DROP TABLE Cust --'

It will delete the specified table, considering that application's privilege are sufficient permission in database. The double dash (--) denotes a SQL comment and is used to comment out any other characters added by the programmer, such as the trailing quote.

' OR 1=1 -builds this command:
SELECT * FROM Usrs WHERE UsrNm='' OR 1=1 -

As 1=1 will remain True, the attacker will be able to retrieve all the table details for the specified table name.

1) Countermeasures to prevent SQL injection include:

To avoid SQL injection, implement strong validation of inputs. Use possible low privileged user account to connect to the database. So, the risk minimizes.

D. Canonicalization [4]

The term Canonicalization is referred when different input of the name resolve to the same standard name.
URL address, File location and paths are vulnerable to this type of attack because for each of them there are multiple ways to present the same.
The same file can be referenced as follows:

c:\temp\file.log
file.log
c:\temp\subdir\..\file.log
c:\ temp\ file.log
..\file.log

1) Countermeasures to address canonicalization issues include:

– Using the absolute file path that cannot be changeable by the end user will be a good practice.
– Use of character encoding could be helpful for the input representation.
– Ensure requestEncoding and responseEncodeing attributes of globalization is element are set in web.conf file of web application.

VI. PROPOSED MODEL

All the available system test the web application against fabricated test cases but real-time scenario could be different as there are multiple way to compromise the security of an application as the technology is emerging and new attacks are introduced periodically.

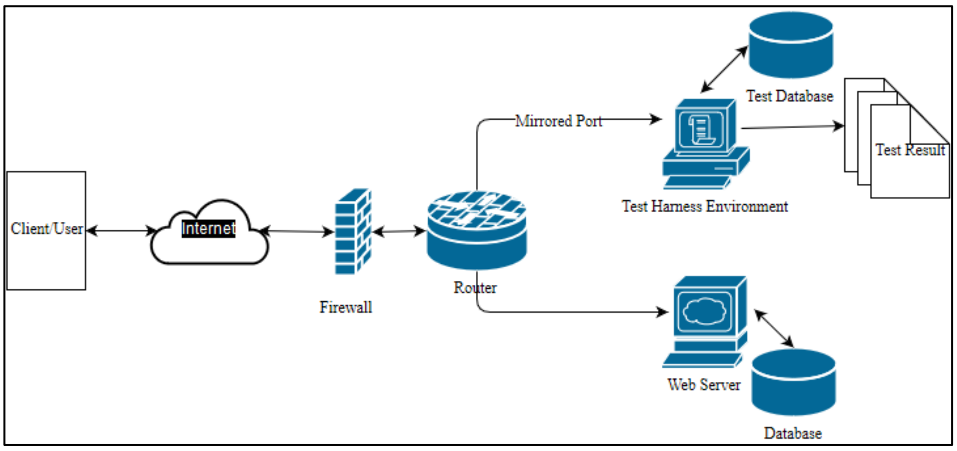


Fig. 2: Proposed Model

So here proposing a new methodology to test the web application against real time scenarios. The result could be useful for improving web application and to and the root cause of any security incident occurred. The proposed system is capable of inspecting the security breach and the impact of the incident to the infrastructure.

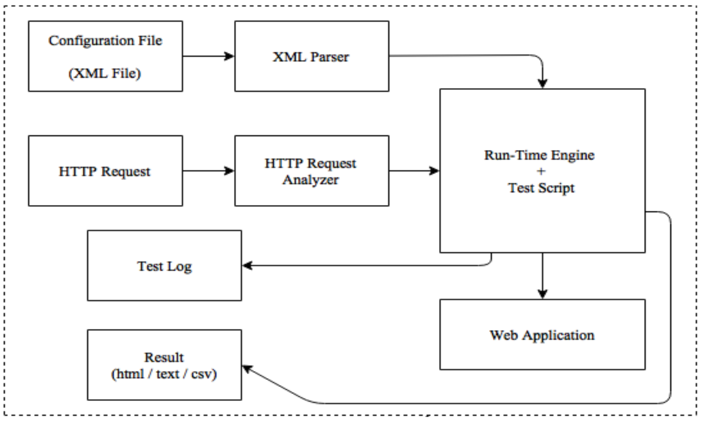
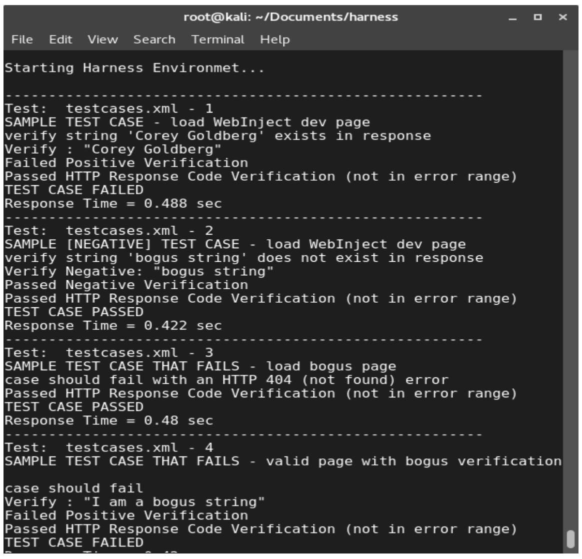


Fig. 3: Harness Configuration

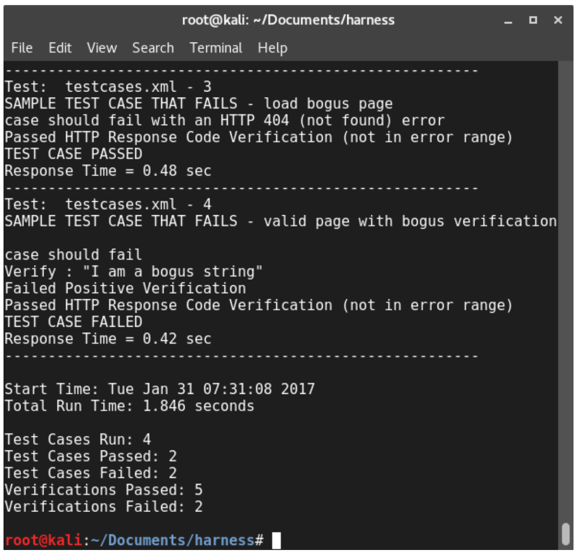
LAB SETUP:

Web Server – Ubuntu machine
Client – Win XP and Kali Machine for Automated Scripts
Virtual Router – MikroTik Router OS
The proposed test harness required following:
Config File, Test Script, Result file, log file


root@kali: ~/Documents/harness
File Edit View Search Terminal Help

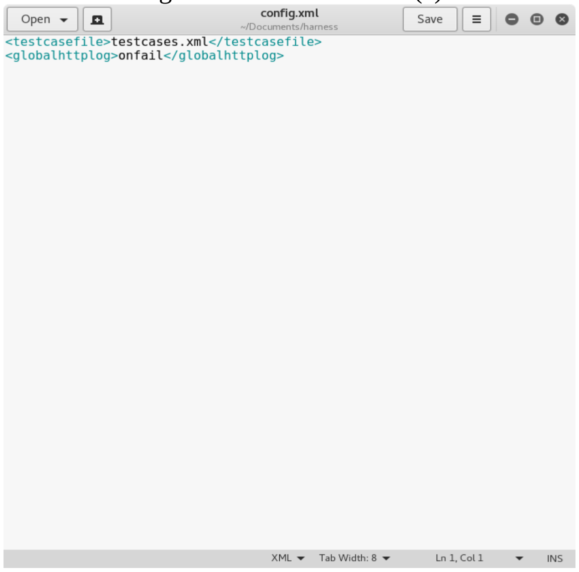
```
Starting Harness Environmet...

-------------------------------------------------------
Test:  testcases.xml - 1
SAMPLE TEST CASE - load WebInject dev page
verify string 'Corey Goldberg' exists in response
Verify : "Corey Goldberg"
Failed Positive Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE FAILED
Response Time = 0.488 sec
-------------------------------------------------------
Test:  testcases.xml - 2
SAMPLE [NEGATIVE] TEST CASE - load WebInject dev page
verify string 'bogus string' does not exist in response
Verify Negative: "bogus string"
Passed Negative Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.422 sec
-------------------------------------------------------
Test:  testcases.xml - 3
SAMPLE TEST CASE THAT FAILS - load bogus page
case should fail with an HTTP 404 (not found) error
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.48 sec
-------------------------------------------------------
Test:  testcases.xml - 4
SAMPLE TEST CASE THAT FAILS - valid page with bogus verification

case should fail
Verify : "I am a bogus string"
Failed Positive Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE FAILED
```

Fig. 4: Harness Execution(a)


root@kali: ~/Documents/harness
File Edit View Search Terminal Help

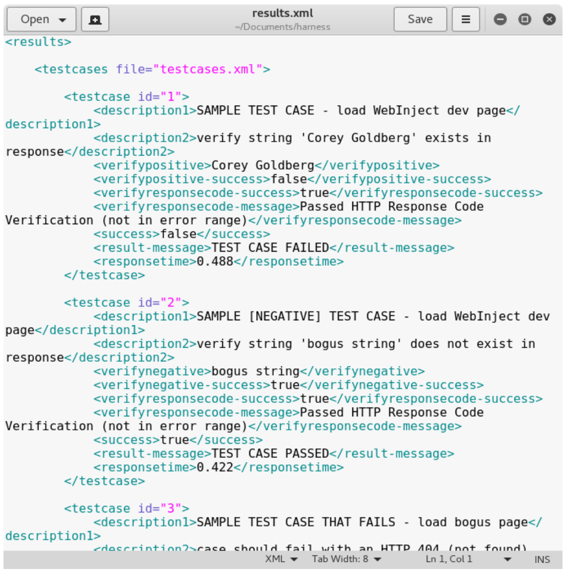
```
-------------------------------------------------------
Test:  testcases.xml - 3
SAMPLE TEST CASE THAT FAILS - load bogus page
case should fail with an HTTP 404 (not found) error
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.48 sec
-------------------------------------------------------
Test:  testcases.xml - 4
SAMPLE TEST CASE THAT FAILS - valid page with bogus verification

case should fail
Verify : "I am a bogus string"
Failed Positive Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE FAILED
Response Time = 0.42 sec
-------------------------------------------------------

Start Time: Tue Jan 31 07:31:08 2017
Total Run Time: 1.846 seconds

Test Cases Run: 4
Test Cases Passed: 2
Test Cases Failed: 2
Verifications Passed: 5
Verifications Failed: 2

root@kali:~/Documents/harness# 
```

Fig. 5: Harness Execution(b)



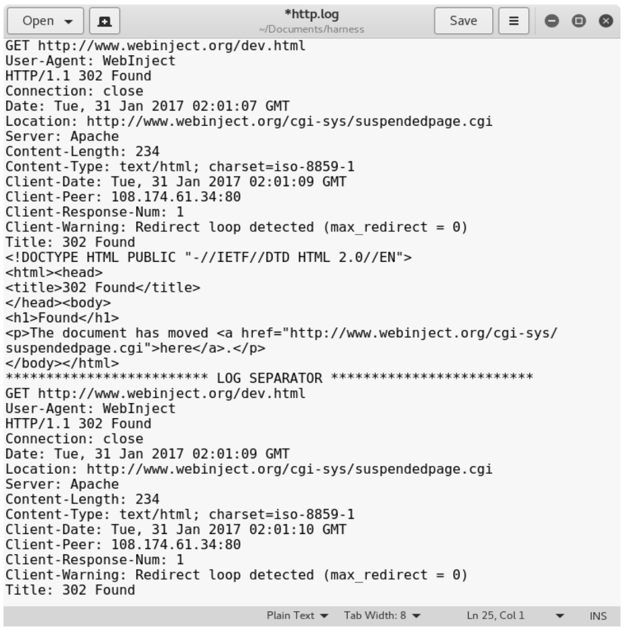
```
<testcasefile>testcases.xml</testcasefile>
<globalhttplog>onfail</globalhttplog>
```

Fig. 6: Configuration file



```
<results>

    <testcases file="testcases.xml">

        <testcase id="1">
            <description1>SAMPLE TEST CASE - load WebInject dev page</
description1>
            <description2>verify string 'Corey Goldberg' exists in
response</description2>
            <verifypositive>Corey Goldberg</verifypositive>
            <verifypositive-success>false</verifypositive-success>
            <verifyresponsecode-success>true</verifyresponsecode-success>
            <verifyresponsecode-message>Passed HTTP Response Code
Verification (not in error range)</verifyresponsecode-message>
            <success>false</success>
            <result-message>TEST CASE FAILED</result-message>
            <responsetime>0.488</responsetime>
        </testcase>

        <testcase id="2">
            <description1>SAMPLE [NEGATIVE] TEST CASE - load WebInject dev
page</description1>
            <description2>verify string 'bogus string' does not exist in
response</description2>
            <verifynegative>bogus string</verifynegative>
            <verifynegative-success>true</verifynegative-success>
            <verifyresponsecode-success>true</verifyresponsecode-success>
            <verifyresponsecode-message>Passed HTTP Response Code
Verification (not in error range)</verifyresponsecode-message>
            <success>true</success>
            <result-message>TEST CASE PASSED</result-message>
            <responsetime>0.422</responsetime>
        </testcase>

        <testcase id="3">
            <description1>SAMPLE TEST CASE THAT FAILS - load bogus page</
description1>
            <description2>case should fail with an HTTP 404 (not found)
```

Fig. 7: Result File <XML>



```
GET http://www.webinject.org/dev.html
User-Agent: WebInject
HTTP/1.1 302 Found
Connection: close
Date: Tue, 31 Jan 2017 02:01:07 GMT
Location: http://www.webinject.org/cgi-sys/suspendedpage.cgi
Server: Apache
Content-Length: 234
Content-Type: text/html; charset=iso-8859-1
Client-Date: Tue, 31 Jan 2017 02:01:09 GMT
Client-Peer: 108.174.61.34:80
Client-Response-Num: 1
Client-Warning: Redirect loop detected (max_redirect = 0)
Title: 302 Found
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.webinject.org/cgi-sys/
suspendedpage.cgi">here</a>.</p>
</body></html>
************************ LOG SEPARATOR ************************
GET http://www.webinject.org/dev.html
User-Agent: WebInject
HTTP/1.1 302 Found
Connection: close
Date: Tue, 31 Jan 2017 02:01:09 GMT
Location: http://www.webinject.org/cgi-sys/suspendedpage.cgi
Server: Apache
Content-Length: 234
Content-Type: text/html; charset=iso-8859-1
Client-Date: Tue, 31 Jan 2017 02:01:10 GMT
Client-Peer: 108.174.61.34:80
Client-Response-Num: 1
Client-Warning: Redirect loop detected (max_redirect = 0)
Title: 302 Found
```

Fig. 8: Log File

VII. Conclusion

Defending web application from real-time attack is a great challenge. The present methods are good enough for testing web application using test cases and they are ensuring security against the same. JIT Compiler Methods are expensive to test for all the traffic. So, there is requirement for a reliable system that is good enough to test web application against real-time traffic.

References

[1] S.t. m.S.S., "What is Test Harness and How is it Applicable to us Testers". Available at: http://www. Softwaretestinghelp .com /what-is-test-harness/>.[Accessed 1 January 2016].
[2] "A Functional, Load, and Performance Testing Framework for Web," 1 January 2016. [Online]. Available at: <http://www.developer.com/services/article.php/2229161/Web-Services-Test-Harness-A-Functional-Load-and-Performance-Testing-Framework-for-Web-Services.htm.>[Accessed 8 January 2016]
[3] "Application Threat Modelling Available," [Online]. Available at: <https://www.owasp.org/index.php/Application_Threat_Modeling#Threat_Model_Information>. [Accessed 10 January 2016].
[4] "Improving Web Application Security: Threats and Countermeasures," [Online]. Available at: <https://msdn.microsoft.com/en-us/library/ff648641.aspx>. [Accessed 15 January 2016].
[5] F. W. M.-H. W. a. J.-W. L. Chi-Yun Wu, "Automated testing of web applications with text input," IEEE International Conference in Progress in Informatics and Computing (PIC), pp. 343-347, 2015.
[6] J. B. a. M. Vella, "Wexpose: Towards on-line dynamic analysis of web attack payloads using just-in-time binary modification," In e-Business and Telecommunications (ICETE), pp. 5-15, 2015.
[7] C. E. a. Y. H. Jairo Pava, "A self-configuring test harness for web applications," ACM, p. 66, 2009.
[8] J. P. a. S. Kansomkeat, "Syntax-based test case generation for web application," IEEE, pp. 389-393, 2015.
[9] T. Nivas, "Test harness and script design principles for automated testing of non-gui or web based applications," ACM, pp. 30-37, 2011.
[10] N. A. a. M. Harman, "State aware test case regeneration for improving web application test suite coverage and fault detection," ACM, pp. 45-55, 2012.
[11] R. V. a. A. K. Bose, "Vulnerability assessment of web applications-a testing approach," IEEE, pp. 1-6, 2015.