© 2017 IJSRST | Volume 3 | Issue 1 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

Discover Broken Authentication and Session Management Vulnerabilities in ASP.NET Web Application

Rupal R Sharma[1], Ravi K Sheth[2]
[1]M.Tech, Cyber Security, Student, Department of Information Technology, Raksha Shakti University, Ahmedabad, Gujarat, India
[2]Assistant Prof., Department of Information Technology, Raksha Shakti University, Ahmedabad, Gujarat, India

ABSTRACT

Today, web application security is most significant battlefield between victim, attacker and resource of web service. The websites which are written in ASP.NET might contain security vulnerabilities which are not seen to the owner of the website. This paper describes an algorithm that aims in the detection of security vulnerabilities of broken authentication and session management. The suggested algorithm of this paper performs a scanning process for website and web application files. Our scanner tool relies on studying the source code of the application depending on ASP.NET files and the code behind files (C sharp C#). A program written for this purpose is to generate a report that describes most leaks and vulnerabilities types by mentioning the file name, leak description and its location. The aim of the paper is to discover the broken authentication and session management vulnerabilities. The suggested algorithm will help organization and developer to fix the vulnerabilities and improve the overall security.

Keywords: Web security, session management, session hijack, Broken Authentication, ASP.NET

I. INTRODUCTION

World Wide Web has evolved from a system that delivers static pages to a platform that supports distributed applications, known as web applications and become one of the most prevalent technologies for information and service delivery over Internet. The increasing popularity of web application can be attributed to several factors, including remote accessibility, cross-platform compatibility, fast development, etc. Web application security is a branch of Information Security that deals specifically with security of websites, web applications and web services. [1]

II. METHODS AND MATERIAL

1. OWASP Top Ten vulnerabilities

A1	Injection.
A2	Broken Authentication and Session Management
A3	Cross-Site Scripting (XSS)
A4	Insecure Direct Object References.
A5	Security Misconfiguration.
A6	Sensitive Data Exposure.
A7	Missing Function Level Access Control.
A8	Cross-Site Request Forgery (CSRF)
A9	Using Components with Known Vulnerabilities
A10	Unvalidated Redirects and Forwards

OWASP TOP 10 Vulnerabilities [2]

The OWASP define that web application related functions related to authentication and session management are not implement correctly which is allowing attackers to compromise password, keys or session tokens or to exploit other implementation flaws to assume other user identities. [3]

Web application security statistics report also shows average vulnerability age by risk which display below. [4]. Following chart show that how many days need to fix or recover any web application which is affect by different attacks? By analysis we concluded that broken authentication and session management vulnerabilities are very harmful for web application. It is take more days to recover the web application.

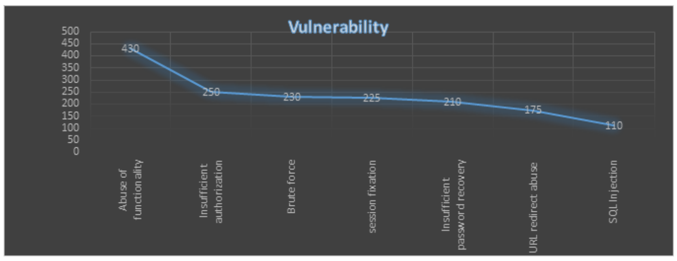


Figure 1. Average time to fix vulnerability

The organization of this document is as follows. In Section 2 (**Methods and Material**), I give detail of most harmful attacks which cause by broken authentication and session management vulnerabilities in asp.net web application. In Section 3(**Implementation of Suggested Algorithm**), I present our propose algorithm and its steps. In Section 4 (**Result and Discussion**), present our research findings and our analysis of those findings. Discussed in Section 5(**Conclusion**) a conclusion is the last part of paper, its end or result.

2. Vulnerability Types

A. Brute Force Attack:

A brute force attack is a trial-and-error method used to obtain information such as a user password or personal identification number (PIN). [6]

B. Hijack Session

Session hijacking is the exploitation of a valid computer session. It is also to gain unauthorized access to information or services in a computer system. It's nothing but hijacking a session.

The attacker needs the cookie form the victim to hijack the session. This is can be implemented by creating one form and make it submit to the attacker site.

```
<form name = 'x' action = 'attackersiteadd' method = 'post'>

<input type = hidden value = '<script> + document.cookie +
</script>'>
</form>
<script> x.submit () </script> [7]
```

C. Replay Attack

A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack by IP packet substitution such as stream cipher attack.

D. Session Fixation Attack

Session fixation attacks attempt to exploit the vulnerability of a system which allows one person to set another person's session identifier (SID). Most session fixation attacks are web based, and most rely on session identifiers being accepted from URLs (query string) or POST data.

E. Session timeout

The Timeout property specifies the time-out period assigned to the Session object for the application, in minutes. If the user does not refresh or request a page within the time-out period, the session ends.

Environments Affected from 'Broken Authentication and Session Management'. [6]

3. Suggested Algorithm

Following algorithm describe the leaks. We developed algorithm and it will generate in python [8]. This algorithm [9] consists of 12 steps where each steps handle a specific kind of leaks.

Check for Broken Authentication and session management:

These algorithms scan the following types of files which are 'aspx', 'aspx.cs and 'web.config'. [10]

Step 1 : Check if "ValidateRequest" attribute in web.config file exits and has value is false then report there is vulnerability. We should prevent displaying of the detailed errors information to users which may serve hackers.

Step 2 : Check if "debug" attribute in web.config file exists and has value is true then report that there is vulnerability.

Step 3 : Check if any TextboxID name in .aspx file is not validated using RegularExpressionValidator or RangeValidator then report there is vulnerability.

Step 4 : Check if "SessionState" mode is off in web.config file then report there is vulnerability.

Step 5 : Check "Timeout" of the session is define or not in session state of web.config. If timeout of the session is not exists in the web.config then report there is vulnerability.

Step 6 : Check authentication is exists or not in the web.config. If exists then check that it's deny all5 anonymous user or not. If "deny users" are not exists in web.config then report there is vulnerability

Step 7 : Check if "Request.Form, Request.QueryString, Request.Cookies" commands in .aspx, .aspx.cs files does not have any kind of AntiXss methods then report there is vulnerability.

Step 8 : check if after logout, cookie remove code exists or not in web.config. If it's not exists then report there is vulnerability.

Step 9 : Check at the logout function, session destroy function exists or not, if it does not exist then report there is vulnerability.

Step 10 : Check "autocomplete" attribute in the form is on or off, if it is "on" then report there is vulnerability. [5]

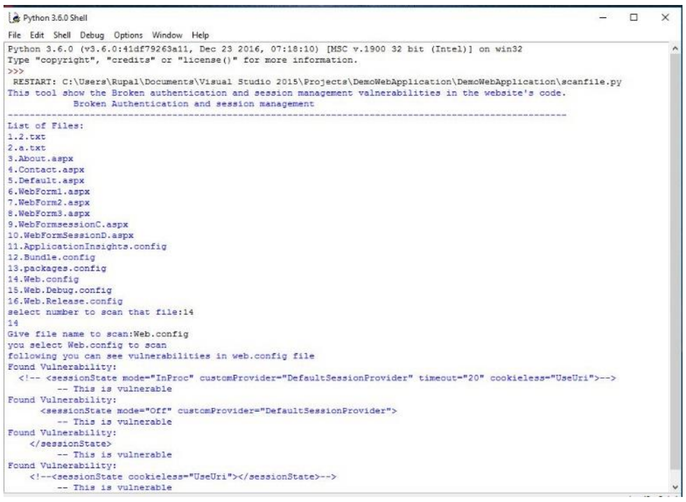
Step 11 : Check "cookieless" value exists in session state of web.config file. If it does not exists then report there is vulnerability.

III. RESULTS AND DISCUSSION

We tried to test our algorithm on some online websites. However, we are not able to get any live website. Therefore, we are forced to check it offline. For this purpose we created web application and put it into IIS server. We have also downloaded some code from following website:

https://www.codeproject.com
https://www.github.com

Sample of vulnerabilities scanning in asp.net web application is shown in the following snap shot:



```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Rupal\Documents\Visual Studio 2015\Projects\DemoWebApplication\DemoWebApplication\scanfile.py
This tool show the Broken authentication and session management valnerabilities in the website's code.
            Broken Authentication and session management
----------------------------------------------------------------------------------------------------------
List of Files:
1.2.txt
2.a.txt
3.About.aspx
4.Contact.aspx
5.Default.aspx
6.WebForm1.aspx
7.WebForm2.aspx
8.WebForm3.aspx
9.WebFormsessionC.aspx
10.WebFormSessionD.aspx
11.ApplicationInsights.config
12.Bundle.config
13.packages.config
14.Web.config
15.Web.Debug.config
16.Web.Release.config
select number to scan that file:14
14
Give file name to scan:Web.config
you select Web.config to scan
following you can see vulnerabilities in web.config file
Found Vulnerability:
  <!-- <sessionState mode="InProc" customProvider="DefaultSessionProvider" timeout="20" cookieless="UseUri">-->
         -- This is vulnerable
Found Vulnerability:
      <sessionState mode="Off" customProvider="DefaultSessionProvider">
         -- This is vulnerable
Found Vulnerability:
    </sessionState>
         -- This is vulnerable
Found Vulnerability:
    <!--<sessionState cookieless="UseUri"></sessionState>-->
         -- This is vulnerable
```

Figure 2. Snapshot of scanning vulnerabilities

We scan more than twenty website/web application in the various tools but we don't find root cause of the attacks and broken authentication and session management vulnerabilities. For that, we purpose this algorithm which shows the exact location in the source code which vulnerable and causes the attack.

The detection process for security vulnerabilities in ASP.NET web application is a complex process, where most of code is written by somebody else and there is no documentation to determine the purpose of some code. Other factor due to the fact that ASP.NET which is part of .NET framework that separate the HTML code from the programming code in two files, one for aspx file another for the programming code depending on the compiled language like Visual Basic, C# and java script. Since the C# is the most common language in use around the world with ASP.NET files, we have use that compiled language in the construction of our proposed algorithm to addition in aspx files. Therefore, the scanning processes at least those three types of files which are aspx, configuration and C#. The developed program tried to scan different forms of writing in code, where there are no standard form followed in writing some code and the presence of some alternatives to some of the commands.

IV. CONCLUSION

Reduce the broken authentication and session management vulnerability in any web application or website requires two things; The first thing is, a conscious developer who is aware to the responsibility which should be accompanied by instilling security into the application from the beginning of programming [11], and the website/ application owner role in inspecting his site/ application for vulnerabilities before making the website public. This paper describes our development tool that is designed to discover vulnerabilities in the source code of websites which help to developer to reduce the vulnerabilities in web application. After scanning process, it will generate a report list all the discovered leaks and vulnerabilities by displaying the name of the infected file, the description and its location.This paper has limitations that it’s only work for asp.net and their supported language. In future, we can make this type of algorithm for PHP or java base language.

V. REFERENCES

[1] Xiaowei Li and Yuan Xue, “A survey on Web Application Security” 2012 Institute of Electrical and Electronics Engineers(IEEE)

[2] OWASP Vulnerability Top ten, Retrieved on February,2017 from https://www.owasp.org/index.php/Category:Vulnerability

[3] The Open Web Application Security Project Book, b OWASP Foundation, https://www.owasp.org/images/f/f8/OWASP-Top-10-2013

[4] “VULNERABILITY LIKELIHOOD BY CLASS” , web security statistics report 2016[online] Retrieved on February,2017 from https://info.whitehatsec.com/rs/675-YBI-674/images/WH-2016-Stats-Report-FINAL.pdf

[5] Tony Hunt, “OWASP Top ten for .net developers”, by plural sight publication.

[6] Rajyalakshmi A.G, “broken authentication and session management” Retrieved on March 2017,from http://www.triadsquare.com/broken-authentication-and-session-management

[7] Huyam AL-Amro and Eyas El-Qawasmeh, “Security Vulnerabilities and Leaks in ASP.NET Websites*”,* 2012 International Conference on E-Learning and E-Technologies in Education (ICEEE).

[8] Paul Gries and Jennifer Campbell, Design Algorithm, Practical programming 2nd edition- A Introduction to computer science using python 3, 2013 The Pragmatic Programmers, LLC.

[9] Paul Gries and Jennifer Campbell, Reading and writing files, Practical programming 2nd edition- A Introduction to computer science using python 3, 2013 The Pragmatic Programmers, LLC.

[10] ASP.NET Web Forms page code model, https://msdn.microsoft.com/en-us/library/015103yb.aspx

[11] B. Sullivan, "Top 10 security vulnerabilities in .NET configuration files", Retrieved on February, 2017 from [Online] http://www.devx.com/dotnet/Article/32493/1954.