Sample Web Application

Black Box Security Assessment
Technical Report

Client A Inc.
1st Jan 2015
Scan ID: LSP –E19010

Electidea™
SECURING DIGITAL

CONFIDENTIALITY & PROPRIETARY

This document contains information that is confidential and proprietary, which shall not be disclosed outside Client A, transmitted, or duplicated, used in whole or in part for any purpose other than its intended purpose. Any use or disclosure in whole or in part of this information without explicit written permission of Client A is prohibited. Electidea makes no warranty that the information contained in this document is complete or error free.

This report is solely for the information of Client A and Client A management and should not be used, circulated, quoted or otherwise referred to for any other purpose, nor included or referred to in whole or in part in any document without our prior written consent

The specific IP addresses / Domain were identified by Client A. Our subsequent test work, study of issues in detail and developing action plans are directed towards the issues identified. Consequently this report may not necessarily comment on all the weaknesses perceived as important by the Client A and / or Client A management.

REPORT ANALYSIS

The issues identified and proposed action plans in this report are based on our testing. We made specific efforts to verify the accuracy and authenticity of the information gathered only in those cases where it was felt necessary.

The identification of the issues in the report is mainly based on the tests carried out during the limited time for conducting such an exercise. As the basis of selecting the most appropriate weaknesses / vulnerabilities is purely judgmental in view of the time available, the outcome of the analysis may not be exhaustive and representing all possibilities, though we have taken reasonable care to cover the major eventualities.

The vulnerabilities reported in this reported are valid as of Jan 1, 2015. Any vulnerability, which may have been discovered after this or any exploit been made available after May 9, 2014, does not come under the purview of this report.

Any configuration changes or software/hardware updates made on hosts/machines on the application covered in this test after the date mentioned herein may impact the security posture either positively or negatively and hence invalidates the claims & observations in this report. Whenever there is an update on the application, we recommend that you conduct penetration test to ensure that your security posture is compliant with your security policies.

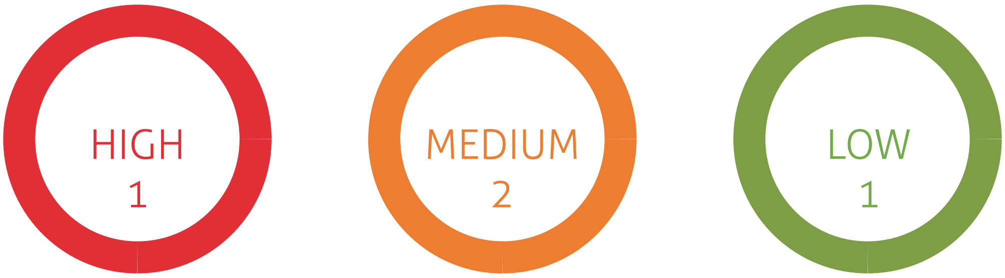
TABLE OF CONTENT

EXECUTIVE
SUMMARY

BACKGROUND

On Dec 1 2014, Client A engaged Electidea to perform a security assessment of their Damn Vulnerable Web Application (DVWA) in an effort to ensure the security of their customer's personal information, which is processed and stored by the DVWA application.

APPLICATION HEALTH



Vulnearbilties in DVWA Web Application

OBSERVATIONS

During the course of this engagement we observed several areas of concern that we believe could pose a significant risk to the security of the application, and should be addressed in a timely manner. Exploiting these vulnerabilities an attacker can retrieve any data from the database which includes sensitive customer data or take over other user's account.

IMMEDIATE ACTIONABLE RECOMMENDATIONS

- Validate all user inputs based on a whitelisting approach.
- Perform output encoding of all user supplied inputs which are reflected back in HTML response.
- Add CAPTCHA to the login page after three failed login attempts.

ELECTIDEA SCANNING PROTOCOL

ALGORITHM FOLLOWED TO CONDUCT THIS SCAN

After our understanding of the test subject from client's technical and business perspective, we executed 3 layers of Electidea Scanning Protocol.

1 Broad Sweep Scan

The test subject is subjected to our automated array of close source and open source tools which are relevant to the subject. Some of the tools include Burp Suite, Nmap, SQLmap, Nikto, etc. Through this scan the client gets a generic blanket of security using the industrially approved tools.

2 Elect Lense Scan

The test subject is subjected to our in house developed automated scanning tools and scripts to not only find vulnerabilities in the subject but also fingerprint any existing malicious signature in the DNA of the test subject i.e. we fingerprint backdoors in the subject.

3 WISE Scan

The above two layers of automated scanning generates multiple reports with multiple vulnerabilities at various threat levels. The first action taken by Electidea WISE (Web Intelligence and Security Experts) team is to thoroughly analyse and validate each test results generated by the above mentioned tools. This gives the client the guarantee of a ZERO false positive result at the end. The second action taken by the WISE team is to manually test every single variable, parameter, service, open ports etc. on the test subject to ensure that they are secure from publically known exploits and payloads in the community.

SCAN DETAILS

Start Date	5th May, 2014
Finish Date	9th May, 2014
Scan Time	5 Days
Server Technology	PHP
URL	http://s28280-101047-qho.sipontum.hack.me/login.php
Credentials	User: admin Role: Administrator
Scope	Black-Box

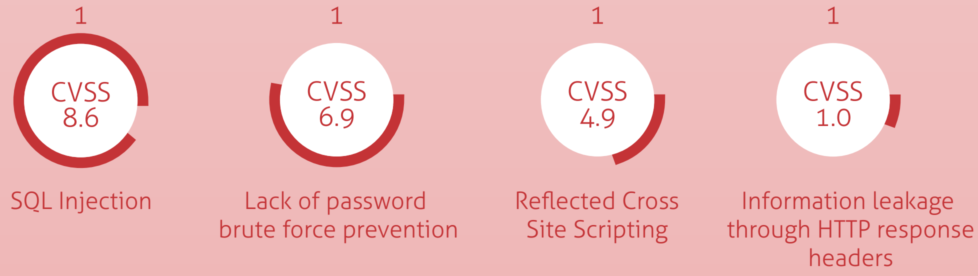
THREAT DISTRIBUTION

SEVERITY LEVEL	COLOR INDICATOR	CVSS CATEGORY
HIGH	RED	7.00-10.00
MEDIUM	ORANGE	4.00-6.69
LOW	GREEN	0.01-3.99

THREAT ANALYSIS

BY INSTANCE COUNT AND CVSS

INSTANCE COUNT



THREAT ANALYSIS

BY SEVERITY LEVEL

HIGH	MEDIUM	LOW
SQL Injection	Lack of password brute force prevention	Fingerprint Web Server
	Reflected Cross Site Scripting	

VULNERABILITIES & RECOMMENDATIONS

1 SQL INJECTION

Relative Risk	High
Vulnerability Class	User Input Handling ➡ Whitelisting User Inputs
CVSS	8.6 (AV:N/AC:L/Au:S/C:P/I:P/A:C/E:H/RL:W/RC:C/CDP:MH/TD:H/CR:M/IR:M/AR:M)
URL	http:// s28280-101047-qho.sipontum.hack.me /vulnerabilities/sqli/index.php
Parameter	id

OBSERVATION

DVWA web application does not validate a user input which is then consumed inside SQL queries. This allows an attacker to provide an input containing SQL statements to modify the output in a way to retrieve desired data from the database. This vulnerability in the application is termed as SQL injection. With this vulnerability, an attacker can dump entire data from the database which the current database user has privileges to access to.

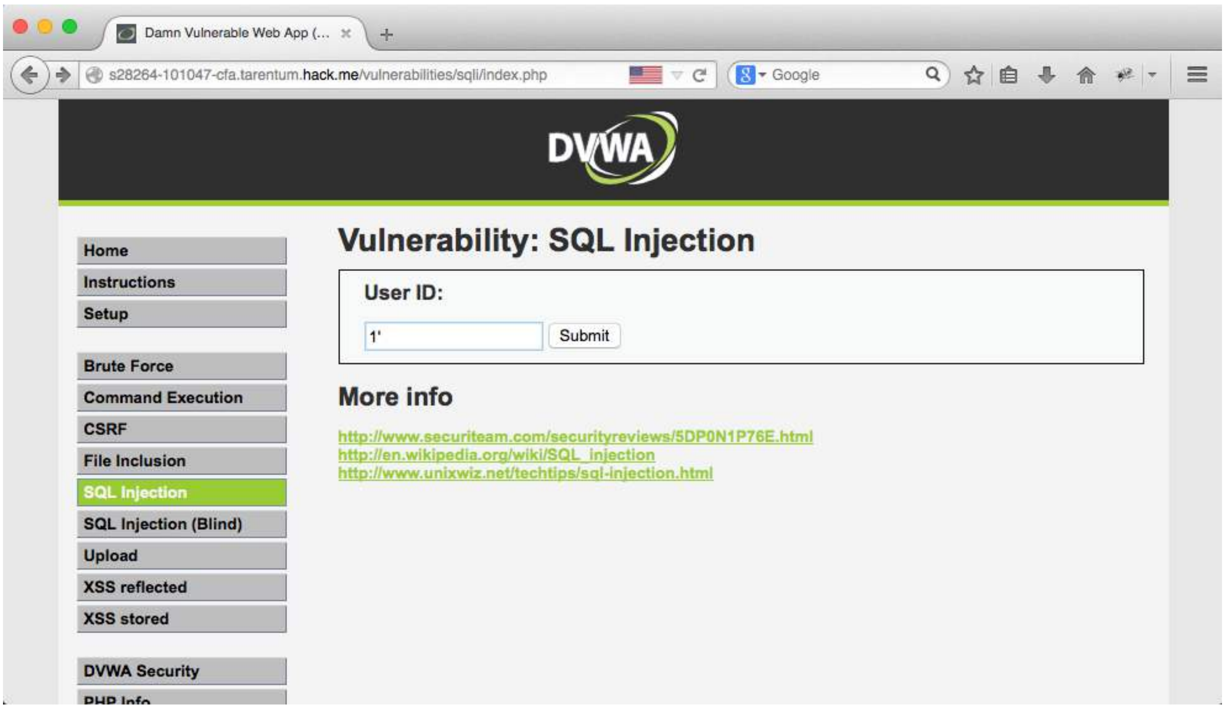
EXHIBITS

Step 1

Login to the web application with admin user account.

Step 2

Navigate to http:// s28280-101047-qho.sipontum.hack.me /vulnerabilities/ sqli/index.php and search for user id 1' as shown in the below screenshot.



s28264-101047-cfa.tarentum.hack.me/vulnerabilities/sqli/index.php

Vulnerability: SQL Injection

User ID: 1'

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

Step 3

You will notice following SQL error message,

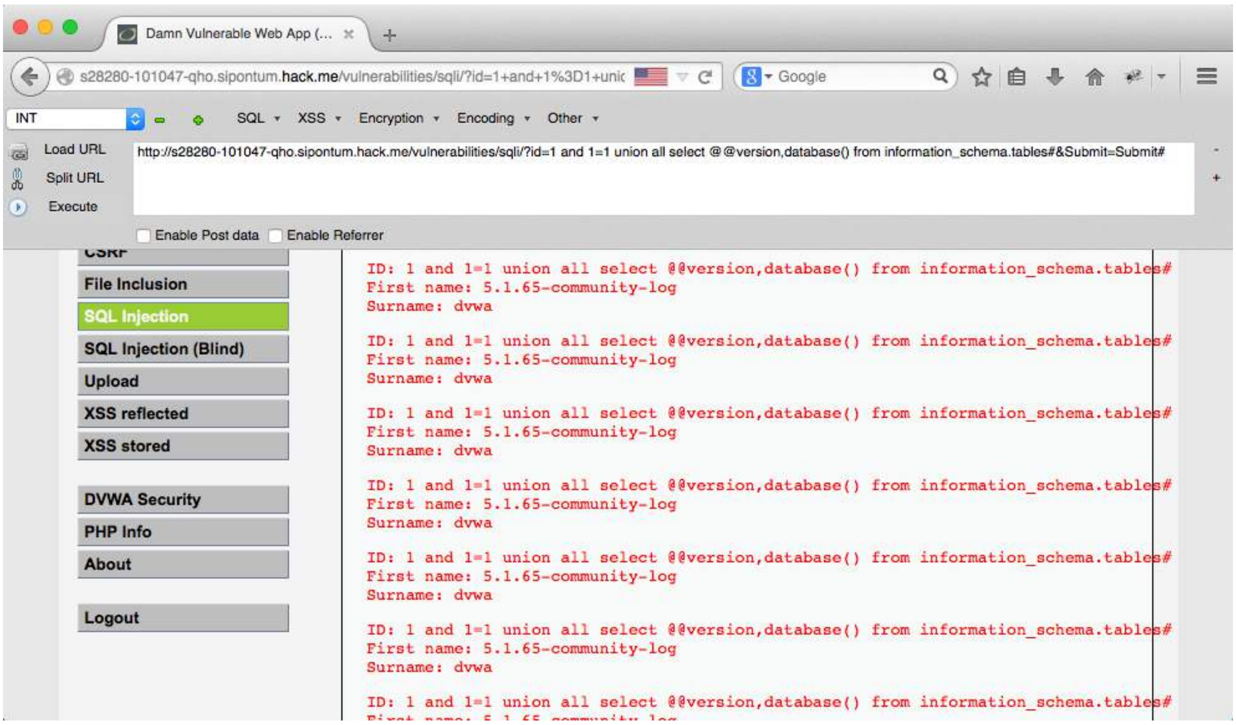
```
You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax
to use near '\'' at line 1
```

This error indicates that the user input is consumed to form dynamic SQL strings. With this knowledge, Electidea analysts were able to retrieve arbitrary data from the database

Step 4

Following payload will extract database version and the database name

```
1 and 1=1 union all select @@version,database() from information_schema.tables# ,
```



```
http://s28280-101047-qho.sipontum.hack.me/vulnerabilities/sqli/?id=1 and 1=1 union all select @@version,database() from information_schema.tables#&Submit=Submit#
```

```
ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
Surname: dvwa

ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
Surname: dvwa

ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
Surname: dvwa

ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
Surname: dvwa

ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
Surname: dvwa

ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
Surname: dvwa

ID: 1 and 1=1 union all select @@version,database() from information_schema.tables#
First name: 5.1.65-community-log
```

Step 5

Following sample data is retrieved from the database using the above mentioned approach

DATABASE INFORMATIONS:

Database Name	dvwa
Version of database	5.1.65-community-log
Current user	dvwaUser@localhost

USER INFORMATION

Username	Password
admin	5f4dcc3b5***65d61d8***deb882cf99
gordonb	e99a18c42***38d5f260***678922e03
1337	8d3533d75a***3966d7e0***cc69216b
pablo	0d107d09f5***40cade3de5***e9e9b7
smithy	5f4dcc3b5aa7***61d8327deb8***f99

Note: Password hashes are partially masked for the security reason.

IMPACT

An attacker can dump entire data from the database that is available to the privilege of current database user. User credentials dumped can further be misused to gain unauthorized access to other user's account. A user only privilege account can be used to conduct this attack in order to gain admin privilege access.

RECOMMENDATION

Following care must be taken in order to prevent application from the SQL injection vulnerability,

- Whitelist user inputs: Validate all user inputs based on allowed data types and data length i.e. for a user input for date parameter (e.g. 01/01/1980) allow only numbers and a forward slash character with the length limitation of 10 characters.

- Prepared Statements: Prepared statements ensure that an attacker is not able to change the intent of a query, even if an attacker inserts SQL commands. If an attacker were to enter the username as admin' or '1'='1, the parameterized query would not be vulnerable and would instead look for a username which literally matched the entire string admin' or '1'='1.

- Input encoding: For free form text inputs such as comment box, address field which may contain any character, application should convert special characters to its HTML entities i.e. convert less than (<) to &lt;, greater than (>) to &gt; etc.

Reference: https://www.owasp.org/index.php/SQL_Injection

2 REFLECTED CROSS SITE SCRIPTING

Relative Risk	Medium
Vulnerability Class	User Input Handling → Output Encoding
CVSS	6.9 (AV:N/AC:L/Au:N/C:N/I:P/A:N/E:H/RL:W/RC:C/CDP:MH/TD:H/CR:M/IR:M/AR:M)
URL	http://s28280-101047-qho.sipontum.hack.me/vulnerabilities/xss_r
Parameter	name

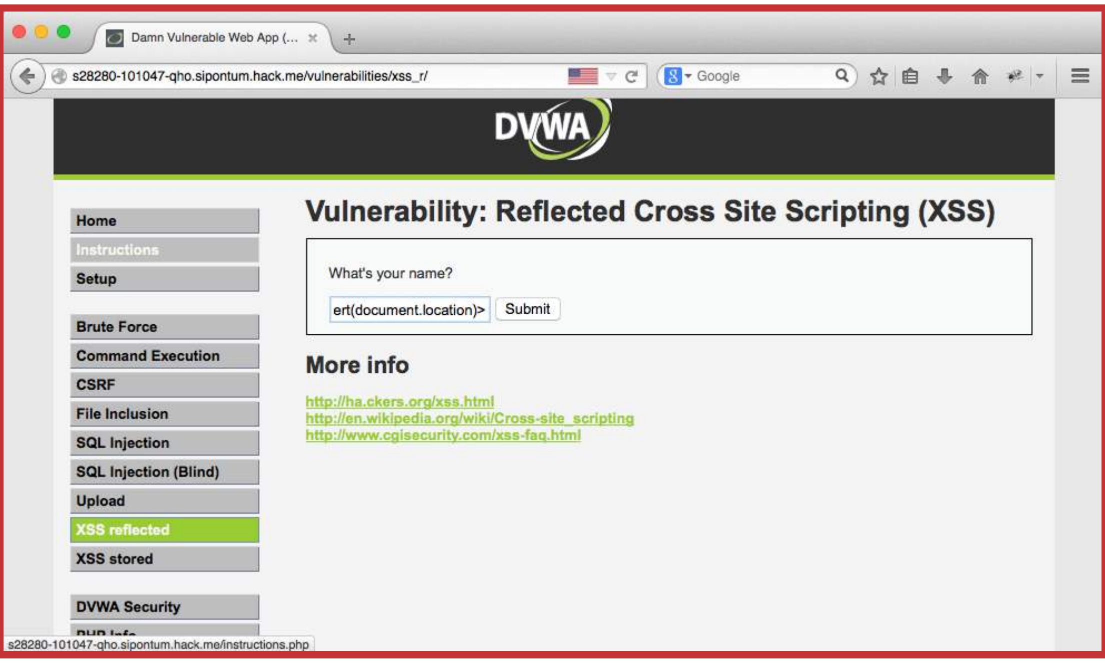
OBSERVATION

The DVWA web application for mobile does not perform output encoding of special characters to prevent Cross Site Scripting vulnerabilities. In one instance user supplied input containing special characters such as <, >, ', /, etc. is echoed back in HTML response without any output encoding performed. This allows an attacker to input malicious JavaScript which can steal victim's cookie, redirect them to other malicious website, etc.

EXHIBITS

Step 1

Navigate to http://s28280-101047-qho.sipontum.hack.me/vulnerabilities/xss_r.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

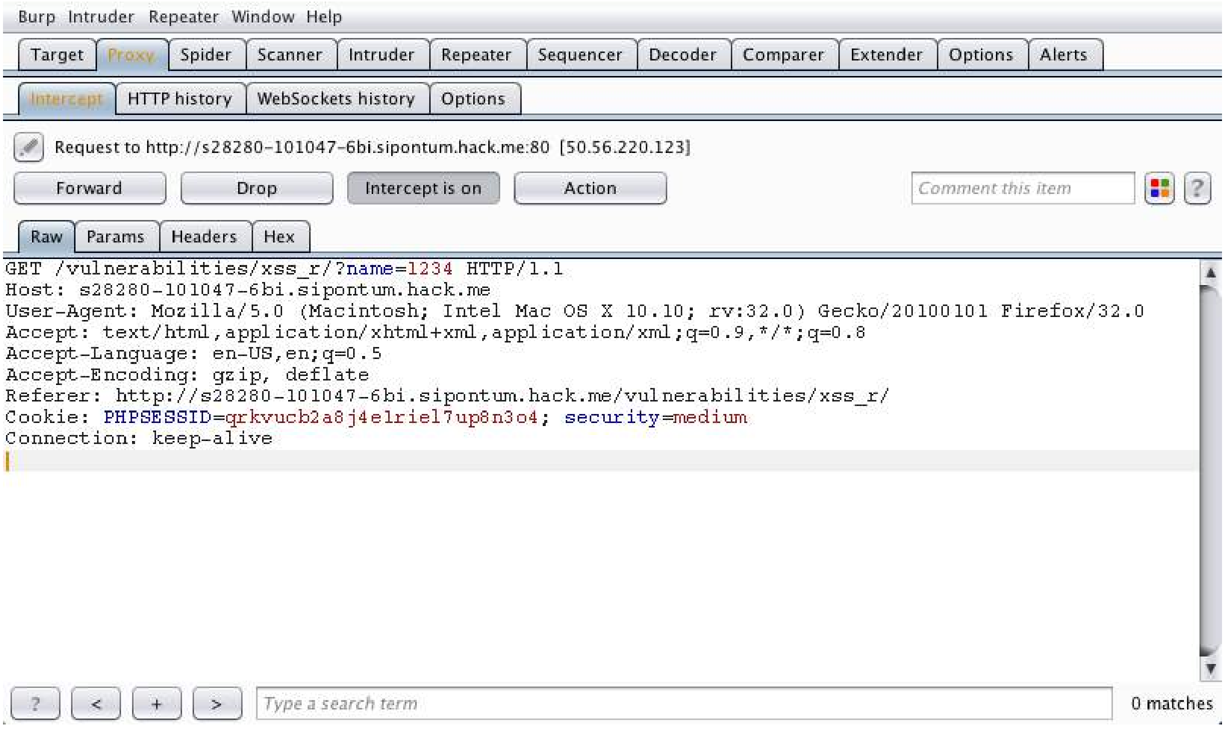
ert(document.location)>

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Step 2

Navigate to http://s28280-101047-qho.sipontum.hack.me/vulnerabilities/xss_r

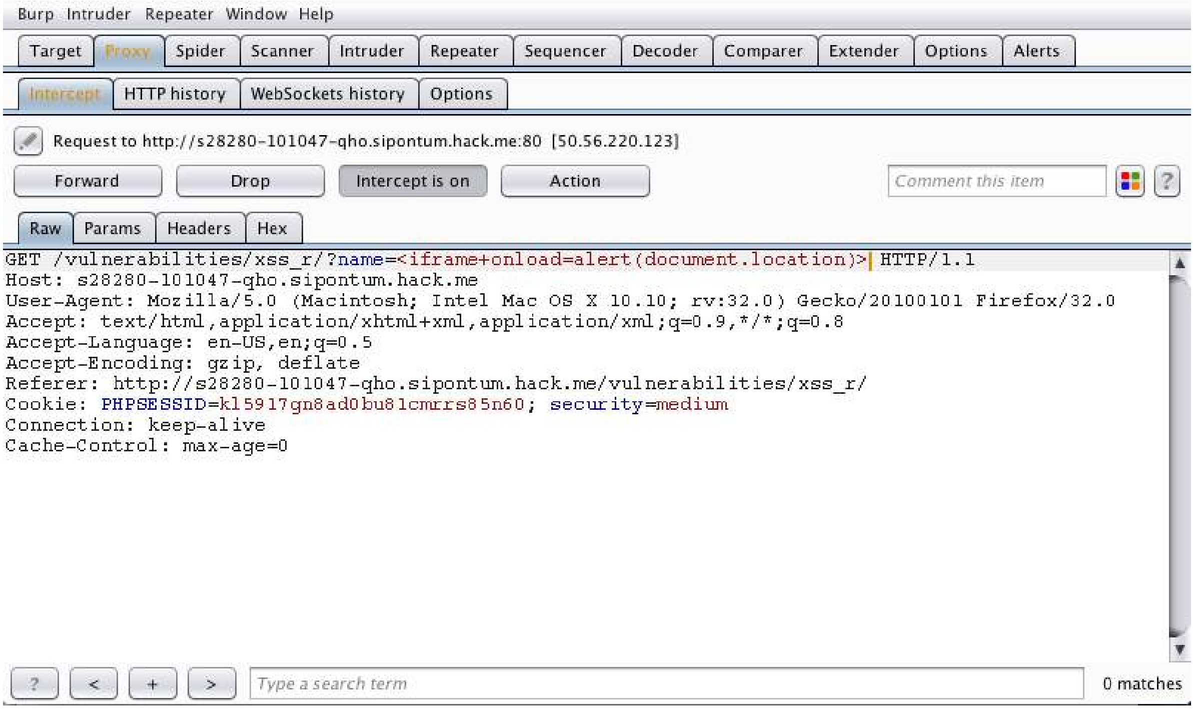

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Request to http://s28280-101047-6bi.sipontum.hack.me:80 [50.56.220.123]
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
? < + >
Type a search term
0 matches

```
GET /vulnerabilities/xss_r/?name=1234 HTTP/1.1
Host: s28280-101047-6bi.sipontum.hack.me
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://s28280-101047-6bi.sipontum.hack.me/vulnerabilities/xss_r/
Cookie: PHPSESSID=qrkvucb2a8j4e1riel7up8n3o4; security=medium
Connection: keep-alive
```

Step 3

You will notice following SQL error message,

```
<iframe onload="alert(document.location)">
```



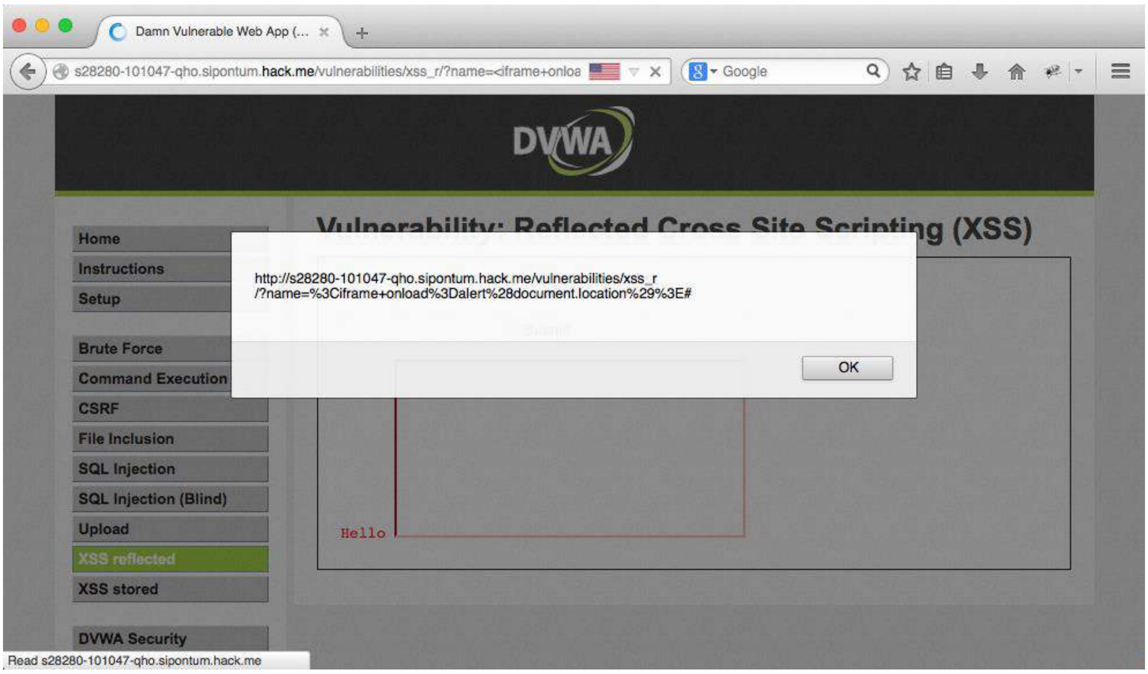
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
? < + >
Type a search term
0 matches

Request to http://s28280-101047-qho.sipontum.hack.me:80 [50.56.220.123]

```
GET /vulnerabilities/xss_r/?name=<iframe+onload=alert(document.location)> HTTP/1.1
Host: s28280-101047-qho.sipontum.hack.me
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://s28280-101047-qho.sipontum.hack.me/vulnerabilities/xss_r/
Cookie: PHPSESSID=kl5917gn8ad0bu8lcmrrs85n60; security=medium
Connection: keep-alive
Cache-Control: max-age=0
```

Step 4

A dialog box with current URL value will be shown as a result of our payload



Damn Vulnerable Web App (...
s28280-101047-qho.sipontum.hack.me/vulnerabilities/xss_r/?name=<iframe+onloa
Google
DVWA
Vulnerability: Reflected Cross Site Scripting (XSS)
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
Hello
OK
Read s28280-101047-qho.sipontum.hack.me

http://s28280-101047-qho.sipontum.hack.me/vulnerabilities/xss_r
/?name=%3Ciframe+onload%3Dalert%28document.location%29%3E#

IMPACT

Reflected Cross Site Scripting is relatively complex to exploit as the malicious payload has to be send as a part of URL and user should be tricked to visit that URL. However, it has the same impact as that of a persistent XSS. In DVWA application, XSS can be used to hijack victim's session and thereby gaining complete access to his/her user account. Additionally, it can be used to redirect victim to a malicious website which may contain browser exploits or a phishing page.

RECOMMENDATION

Following care must be taken to prevent application from XSS vulnerabilities,

- Whitelist user inputs: Validate all user inputs based on allowed data types and data length i.e. for a user input for date parameter (e.g. 01/01/1980) allow only numbers and a forward slash character with the length limitation of 10 characters.

- Output encoding: Encode user input into its equivalent HTML/URL encoding when a user input is reflected back in the HTML response.

References: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

3 LACK OF PASSWORD BRUTE FORCE PREVENTION

Relative Risk	Medium
Vulnerability Class	Authentication ➡ Password brute force
CVSS	4.9 (AV:N/AC:M/Au:S/C:P/I:P/A:N/E:POC/RL:W/RC:C/CDP:LM/TD:M/CR:M/IR:M/AR:M)
URL	http://s28280-101047-qho.sipontum.hack.me/login.php
Parameter	password

OBSERVATION

The DVWA application does not lockout a user account or provides CAPTCHA when 'n' failed login attempts is made. Electidea analysts tried with a threshold of 15 failed login attempts during which account neither locked out or a CAPTCHA was provided.

EXHIBITS

Step 1

Navigate to the login page of http://s28280-101047-qho.sipontum.hack.me and provide an invald username and password.

Step 2

Repeat step 1 multiple times. You will notice that application will neither provide any CAPTCHA to the user or will block victim user's account.

Step 3

Use a valid password and the application will redirect you to the account details rather than displaying an error message indicating that the account is locked out

IMPACT

An attacker can use brute force attack to guess valid password for an account. In a brute force attack, automated software is used to generate a large number of consecutive guesses as to the value of the desired data. Another form of brute force attack known as a dictionary attack might try all the words in a dictionary to guess the user password. Moreover, due to the failure of strong password policy control, this vulnerability is relatively easy to exploit.

RECOMMENDATION

Password brute force attacks can be prevented by providing user with a strong CAPTCHA value upon 3 failed attempts. Additionally, blocking IP address or temporary account lockout can be implemented after 15 failed attempts. The later method can also be misused by an attacker to lock multiple user accounts and thereby creating a denial of service like situation.

References: https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

4 INFORMATION LEAKAGE THROUGH **HTTP** RESPONSE HEADERS

Relative Risk	Low
Vulnerability Class	HTTP Security ➡ X-Powered-By header
CVSS	1.0 (AV:N/AC:L/Au:N/C:P/I:N/A:N/E:U/RL:W/RC:C/CDP:N/TD:L/CR:M/IR:M/AR:M)
URL	http://s28280-101047-qho.sipontum.hack.me/
Parameter	Not Applicable

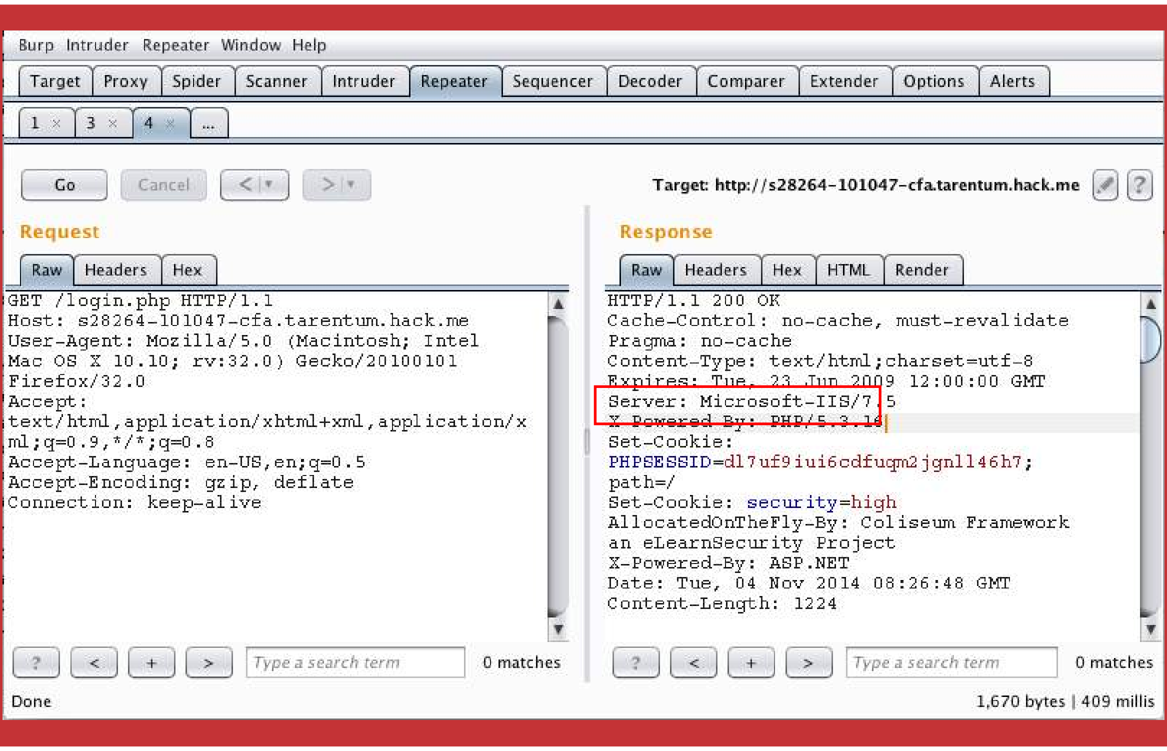
OBSERVATION

The web server hosting DVWA application is misconfigured due to which application server version is exposed to end users.

EXHIBITS

Step 1

Request any web page of the application and observe the response headers through BURP suite as shown below,.



Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
1 × 3 × 4 × ...
Go Cancel < >
Raw Headers Hex
Raw Headers Hex HTML Render
? < + > Type a search term 0 matches
Done
1,670 bytes | 409 millis

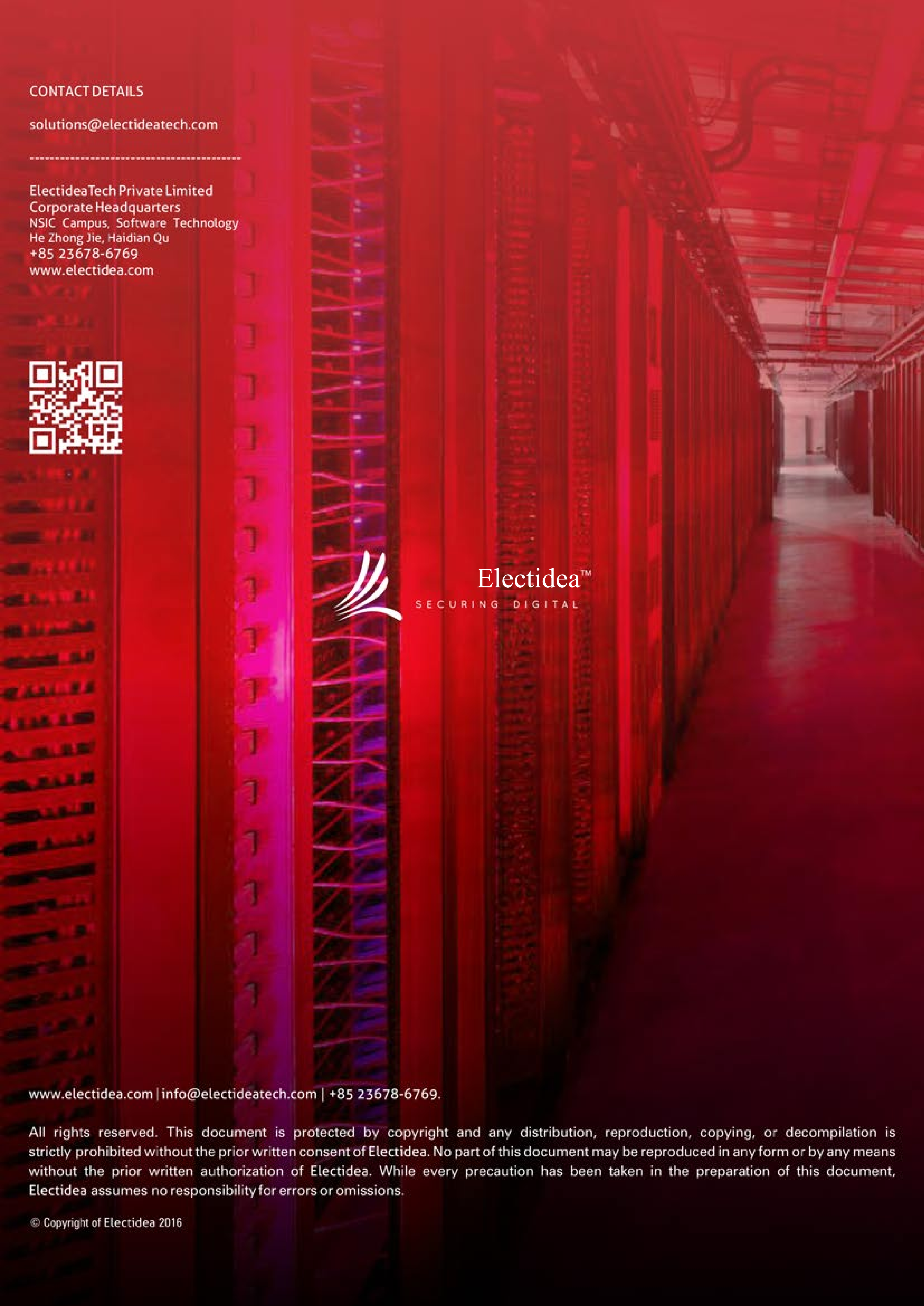
Target: http://s28264-101047-cfa.tarentum.hack.me

Request

```
GET /login.php HTTP/1.1
Host: s28264-101047-cfa.tarentum.hack.me
User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10.10; rv:32.0) Gecko/20100101
Firefox/32.0
Accept:
text/html,application/xhtml+xml,application/x
ml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Content-Type: text/html;charset=utf-8
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Server: Microsoft-IIS/7.5
X-Powered-By: PHP/5.3.10
Set-Cookie:
PHPSESSID=dl7uf9iui6cdfuqm2jgnll46h7;
path=/
Set-Cookie: security=high
AllocatedOnTheFly-By: Coliseum Framework
an eLearnSecurity Project
X-Powered-By: ASP.NET
Date: Tue, 04 Nov 2014 08:26:48 GMT
Content-Length: 1224
```



CONTACT DETAILS

solutions@electideatech.com

ElectideaTech Private Limited
Corporate Headquarters
NSIC Campus, Software Technology
He Zhong Jie, Haidian Qu
+85 23678-6769
www.electidea.com

Electidea™
SECURING DIGITAL

www.electidea.com | info@electideatech.com | +85 23678-6769.

IMPACT

Attacker can know the version of the PHP running on the web server by the header and can search for the known vulnerabilities of PHP 5.4.23 for further exploitation.
There is no direct impact to business with this vulnerability but falls under security best practices.

RECOMMENDATION

By default expose_php option is set to On. In php.ini file, locate the line containing "expose_php On"_ and set it to Off

- expose_php = Off

References: https://www.owasp.org/index.php/Fingerprint_Web_Application_Framework_(OTG- INFO-008)