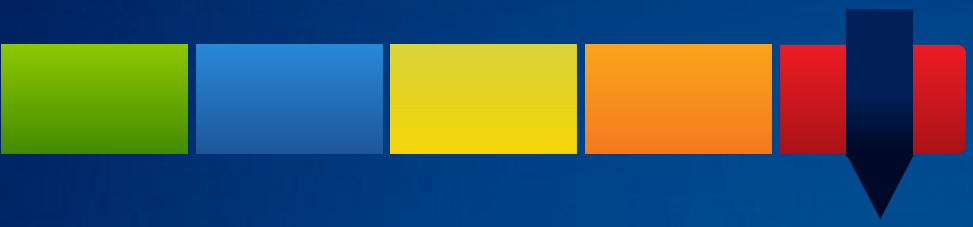
Contents

Key Services

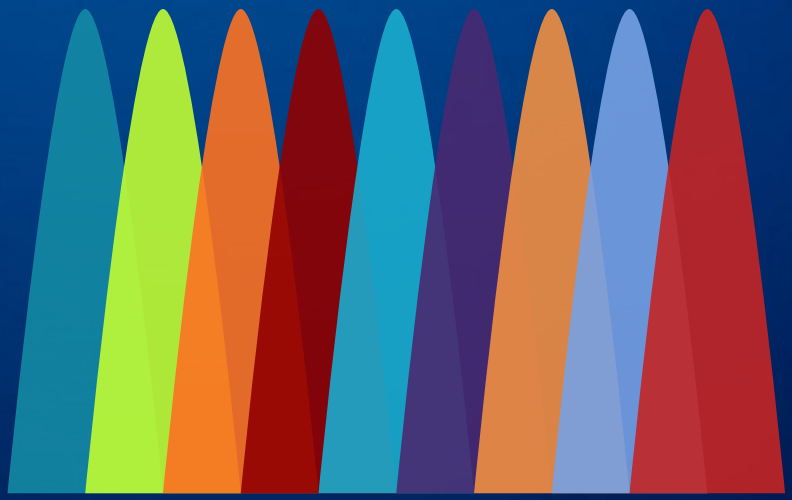
Breakdown of Key Metrics

High Risk	Medium Risk	Low Risk	Informational
1 Issues	5 Issues	2 Issues	4 Issues



Severe Risk

Insufficient Input Validation
Insecure Configuration
Patch Management
Insufficient Encryption
Insufficient Patch Management
Server Configuration
Application Misconfiguration
Insufficient Webroot Management
Insufficient Session Validation

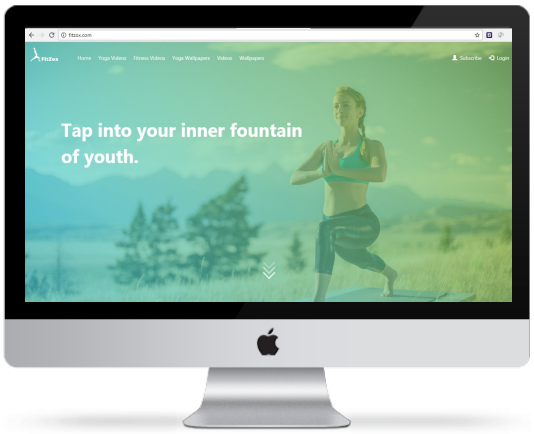


100% of all high-risk issues could be resolved by fixing the "Insufficient Input Validation" issue.

Overview

About Application

This Application provides content that are related to the fitness where a user can subscribe under different subscription plan and accordingly view the contents.



Scope of Work

The following hosts were considered to be part of the scope of this engagement.

No.	URL	Description
1	fitzox.com	WAP Based 1 Video Portal (2nd Application)

Timelines

The project Fitzox WAP Application was started on 2017-05-05 and completed on 2017-05-09.

Milestone	Start Date	End Date
Initial Analysis	2017-05-05	2017-05-05
Technical Audit	2017-05-05	2017-05-08
Approval Review	2017-05-06	2017-05-09
Reporting	2017-05-08	2017-05-09

Vulnerabilities

Title	Risk	Status
SQL Injection Vulnerabilities	High	Vulnerable
Directory Listing enabled on the Server	Medium	Vulnerable
PHP Outdated Version contains Several Vulnerabilities	Medium	Vulnerable
SSL not Implemented	Medium	Vulnerable
Cross-Site Scripting Vulnerabilities	Medium	Vulnerable
Apache Outdated Version Contains Several Security Vulnerabilities	Medium	Vulnerable
In-secure HTTP Methods are Enabled	Low	Vulnerable
Cookie without HTTPOnly flag set	Low	Vulnerable
Information Disclosure by Verbose Error Messages	Informational	Vulnerable
Information Disclosure by Test, Old and Backup Files	Informational	Vulnerable
Cross-Site Request Forgery Token in Header Missing	Informational	Vulnerable
Potential Clickjacking Vulnerability	Informational	Vulnerable

SQL Injection Vulnerabilities

High Risk

Vulnerable

Description

Applications often use databases at the backend to store and manage large amounts of information. The de-facto standard language for querying databases is SQL. Web applications often take user input (taken out of the HTTP request) and incorporate it in an SQL query, which is then sent to the backend database. The query results are then processed by the application and sometimes displayed to the user.

Database Compromise
Impact

Insufficient Input Validation
Cause

Developer
Responsibility

Medium
Difficulty

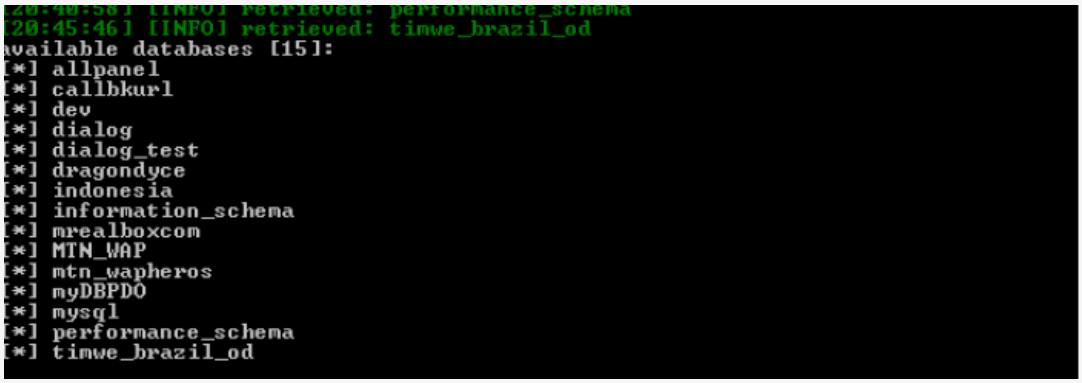
Critical / CVSS Base Score: **9.6**
(AV:N/AC:L/Au:N/C:C/I:C/A:P)

By exploiting this vulnerability, an attacker can directly pass malicious queries and inputs to the database and interpret the responses from the database. It allows an attacker to read, write, modify or delete information stored within the database along with sometimes gaining system level access to the underlying operating system.

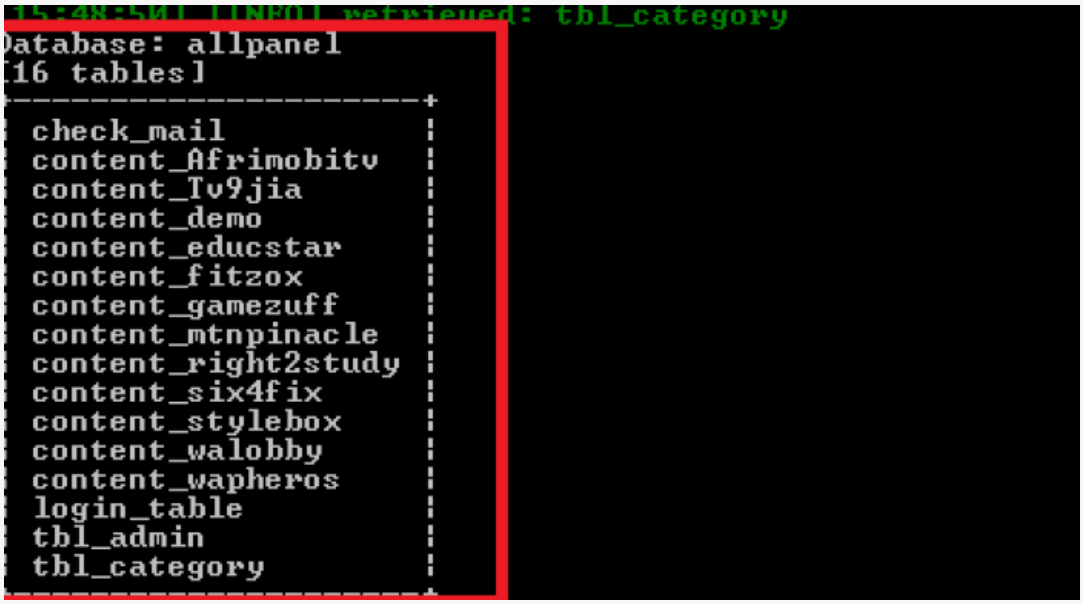
In the instances below, the affected parameters were passing user input directly to the back-end database without proper validation. Because of this, it is possible to insert malicious data into these fields to not only cause errors but also to gain complete access to the database. On successful exploitation of the vulnerability, an attacker would have access to Read, Write and Modify any data stored within the database.

Proof-Of-Concept

Step 1. While testing the application we were able to inject malacious query and able to enumarate the database name on the server.



Step 2. After retreving the database name we then enumerate the table present in the database.



Step 3. Similarly we were able to enumerate the columns present in the table content_fitzox.

Recommendations

SQL injection and blind SQL injection:

There are four possible ways to protect your web application against SQL injection attacks.

1. Store Procedures

Use a stored procedure rather than dynamically built SQL query string. The way parameters are passed to SQL Server stored procedures, prevents the use of apostrophes and hyphens. After the storing of the commands is done, the tasks can be performed or executed continuously, without being repeatedly sent to the server. This also helps in decreasing the traffic in the networks and also reduces the CPU load.

Why use a stored procedure?

Develop the functionality once and all the applications can call the same commands.

Network Traffic reduced to a greater extent.

Centralization of all commands made possible, which is helpful for various applications that repeatedly call the same set of complicated commands.

Runs on any kind of environment.

Create a Stored Procedure in the Database example for PHP

```
/**
 * @author  Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

DELIMITER $$
// Check if stored procedure exists, if yes delete it and create new one.
DROP PROCEDURE IF EXISTS `UName`.`get_user`$$
CREATE PROCEDURE  `Uname`.`get_user`
(

// Pass the value for userId to the stored procedure
IN userId INT,

// Return the value after the stored procedure has executed
OUT firstName VARCHAR(100),
OUT lastName VARCHAR(100)
)
BEGIN
SELECT first_name, last_name
INTO firstName, lastName
FROM users
WHERE users_id = userId;
END $$
DELIMITER ;
```

2. Parameterized Queries

A parameterized query is a query in which placeholders are used for parameters and the parameter

values are supplied at execution time. The placeholders are typically type-specific (for example, int for integer data and text for strings) which allows the database to interpret the data strictly. For instance, a text placeholder is always interpreted as a literal, avoiding exploits such as the query stacking SQL injection. A mismatch between a placeholder's type and its incoming datum causes execution errors, adding further validation to the query.

This is the absolute best method to prevent any SQL injection. Although complex nature of the code makes it bit difficult to implement

.

Parameterized Queries example for PHP

```
/**
 * @author   Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

<?php
// Basic query to be used, with ? representing the input data
$csquery = "SELECT `fname` FROM students WHERE (`sname` LIKE ?);

// Parse the input data through the function to escape any malicious data
$sname = mysqli_real_escape_string ( $dblink, $_GET['sname'] );

// Initialize transaction with database
$cs = mysqli_stmt_init ( $dblink );

// Send base query to database
mysqli_stmt_prepare ( $cs, $csquery ) or die ( "Error: Please contact the administrator" );

// Send parameter type and value separately to the database
mysqli_stmt_bind_param ( $cs, "s", $sname);

// Execute Query
mysqli_stmt_execute ( $cs );?>
```

3. Input Validation

You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation - for example, testing for valid dates or values within a range - plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

In the below example the value for age is validated using an int function. This would prevent user from passing any string character to the database. This is not a full-proof method to prevent SQL injection but it will make attacker's job a bit harder.

Input Validation example for PHP

```
/**
 * @author   Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

<?php
$name = htmlspecialchars($_POST['name'], ENT_QUOTES);
echo $name;
?>
```

Instances

fitzox.com

URL	Parameters	Status	POC
getpost.php	cat	Vulnerable	⊘

Reference Documents

GENERAL

- SQL Injection - WASC
- CAPEC-66: SQL Injection
- SQL Injection - OWASP
- Improper Neutralization of Special Elements used in an SQL Command

Directory Listing enabled on the Server

Medium Risk

Vulnerable

Description

A web directory is a listing of websites organized in a hierarchy or interconnected list of categories. If Directory Listing is enabled on the web-server, an attacker can view a list of all files from this directory, possibly exposing sensitive information.

An attacker can use this vulnerability to view and understand the functionality of the website and use it to conduct targeted attacks on the website.

Information Disclosure
Impact

Insecure Configuration
Cause

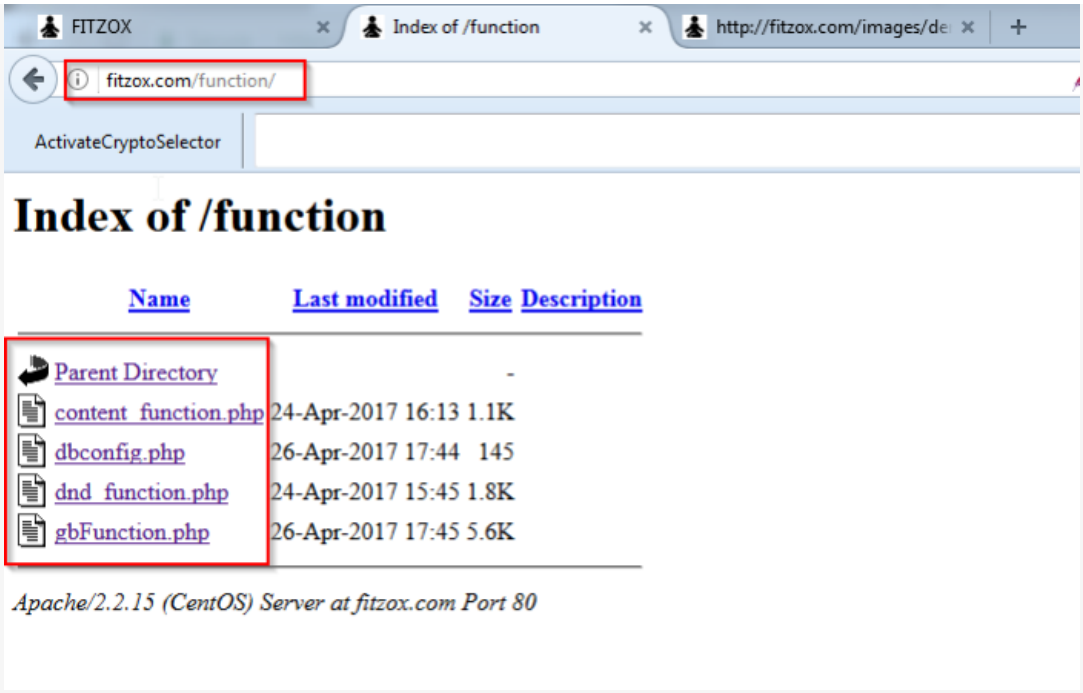
System Administrator
Responsibility

Easy
Difficulty

Critical / CVSS Base Score: **6.4**
(AV:N/AC:L/Au:N/C:P/I:P/A:N)

Proof-Of-Concept

As can be seen below directoy listing is enabled on the server.



Recommendations

It's advisable to disable Directory Listing on the webserver

Disable Directory List example for Apache

```
/**
 * @author  Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */
```

To prevent directory listings, create an .htaccess file following the main instructions and guidance which includes the following text:
IndexIgnore *

Instances

fitzox.com

URL	Parameters	Status	POC
includes/	–	Vulnerable	⊘
CG/	–	Vulnerable	⊘
integration	–	Vulnerable	⊘
GCG	–	Vulnerable	⊘
FCG/	–	Vulnerable	⊘
FGCG/	–	Vulnerable	⊘
logs/	–	Vulnerable	⊘
layout/	–	Vulnerable	⊘
function/	–	Vulnerable	⊘
images/	–	Vulnerable	⊘
content/	–	Vulnerable	⊘
icons/	–	Vulnerable	⊘
favicons/	–	Vulnerable	⊘

Reference Documents

APACHE

- Disable Directory Listing

PHP Outdated Version contains Several Vulnerabilities

Medium Risk

Vulnerable

Description

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. The version of PHP being used has been identified as being outdated and containing several security vulnerabilities.

The version of PHP being used has been identified as being outdated and containing several security vulnerabilities.

Here are some of the know vulnerabilities that is present in the current version 5.3.3 :

System Compromise
Impact

Patch Management
Cause

System Administrator
Responsibility

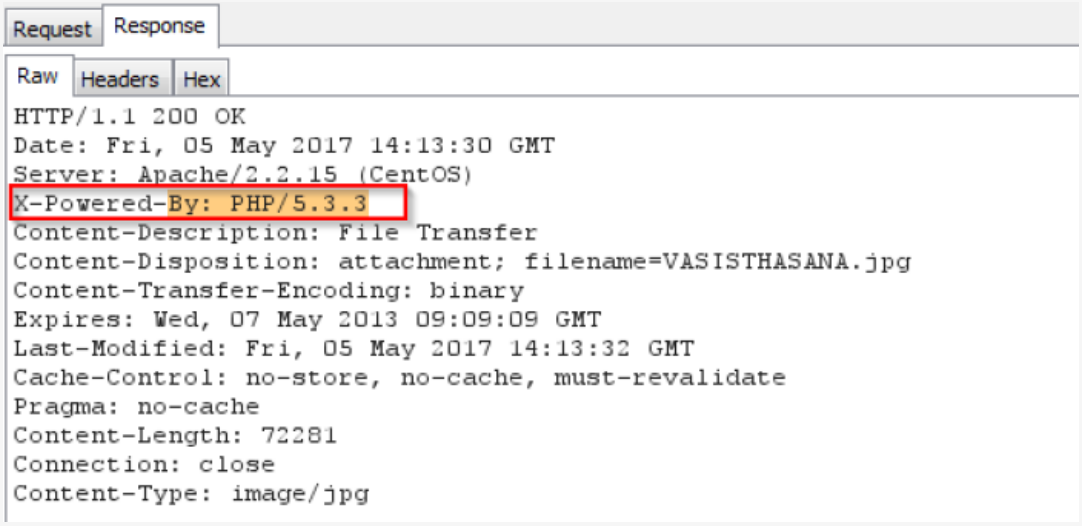
Medium
Difficulty

Critical / CVSS Base Score: **6.4**
(AV:N/AC:L/Au:N/C:P/I:P/A:N)

- CVE-2016-7478 - It allows remote attackers to cause a denial of service (infinite loop via a crafted Exception object in serialized data.
- CVE-2014-9427 119 - It allows remote attackers to obtain sensitive information from php-cgi process memory by leveraging the ability to upload a .php file or (2) trigger unexpected code execution if a valid PHP script is present in memory locations adjacent to the mapping.
- CVE-2013-6420 119 - It allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted certificate that is not properly handled by the openssl_x509_parse function.

Proof-Of-Concept

While scanning the Application we found that the PHP version being used is outdated.



Recommendations

Upgrade to the latest version of PHP from the following url:
http://php.net/downloads.php

Instances

fitzox.com

URL	Parameters	Status	POC
/	–	Vulnerable	⊘

Reference Documents

PHP

- PHP Security Vulnerabilities
- Download

SSL not Implemented

Medium Risk

Vulnerable

Description

SSL is the most common mechanism to ensure server authenticity and provide data privacy. Using a non-SSL page for transmitting sensitive parameter makes it vulnerable to network sniffing as data is sent in clear-text. An attacker can thus obtain users credentials and in-turn compromises his account. In this case we found that the side does not use SSL for sending the traffic, making it vunlerable to sniffing attack.

User Account Compromise
Impact

Insufficient Encryption
Cause

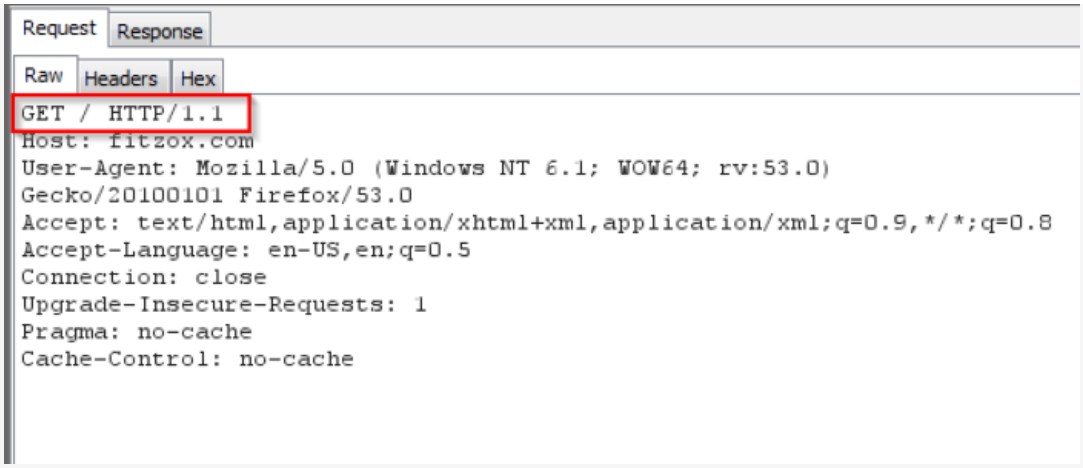
System Administrator
Responsibility

Easy
Difficulty

Critical / CVSS Base Score: **6.4**
(AV:N/AC:L/Au:N/C:P/I:P/A:N)

Proof-Of-Concept

While testing we found that the application was not using SSL for sending traffic over the network.



Recommendations

It is advisible to use and implement SSL throughout the website to prevent attacker from sniffing in to the networks and retrieving sensitive information.

Instances

fitzox.com

URL	Parameters	Status	POC
/	–	Vulnerable	⊘

Reference Documents

GENERAL

- Secure HTTP
- Doing https: (http over SSL)

Cross-Site Scripting Vulnerabilities

Medium Risk

Vulnerable

Description

Websites often accept user input for the application to display on the screen. If the application is not careful enough with its treatment of user (attacker) input, it is possible for an attacker to inject malicious data, which when displayed on the screen can execute HTML or JavaScript code in the user's browser.

This vulnerability allows an attacker to either permanently or temporarily inject client-side code into the target website. This code executes when the page is loaded by the victim and the client-side code may carry out activities such as: stealing cookies/sessions, modifying the page contents, logging key strokes, etc.

User Account Compromise
Impact

Insufficient Input Validation
Cause

Developer
Responsibility

Medium
Difficulty

Critical / CVSS Base Score: **5.7**
(AV:N/AC:M/Au:N/C:P/I:P/A:N)

Proof-Of-Concept

While testing we found that the application was vulnerable to XSS attack.

Recommendations

In-order to prevent Cross-Site Scripting issues, you can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation - for example, testing for valid dates or values within a range - plus ways to provide custom-written validation.

Input Validation example for PHP

```
/**
 * @author   Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

<?php
$name = htmlspecialchars($_POST['name'], ENT_QUOTES);
echo $name;
?>
```

In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

Instances

fitzox.com

URL	Parameters	Status	POC
getpost.php	cat	Vulnerable	⊘
fcg.php	–	Vulnerable	⊘

Reference Documents

GENERAL

- CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests
- The Cross-Site Scripting (XSS) FAQ
- Cross Site Scripting Info
- Character entity references in HTML 4
- Understanding Malicious Content Mitigation for Web Developers
- Cross Site Scripting Explained

Apache Outdated Version Contains Several Security Vulnerabilities

Medium Risk

Vulnerable

Description

Apache is an open-source web server platform, which guarantees the online availability of the majority of the websites active today. The server is aimed at serving a great deal of widely popular modern web platforms.
Latest Stable Version:2.4.25
Here are some of the know vulnerabilities that is present in the current version (Apache httpd 2.2.15) :

Network Compromise	**Impact**
Insufficient Patch Management	**Cause**
System Administrator	**Responsibility**
Easy	**Difficulty**
Critical / CVSS Base Score: **5.6**	**(AV:L/AC:L/Au:N/C:C/I:N/A:P)**

- CVE-2011-3192 - It allows remote attackers to cause a denial of service (memory and CPU consumption) via a Range header that expresses multiple overlapping ranges.
- CVE-2014-031 - It alows remote attackers to cause denial of service (process hang) via a request to a CGI script that does not read from its stdin file descriptor.
- CVE-2014-0098 - It allows remote attackers to cause a denial of service (segmentation fault and daemon crash) via a crafted cookie that is not properly handled during truncation.

Proof-Of-Concept

While scanning the application we found that the Apache version being used is outdated.

Recommendations

Upgrade the Apache httpd to the latest stable version .

Instances

fitzox.com

URL	Parameters	Status	POC
/	–	Vulnerable	⊘

Reference Documents

APACHE

- Download Latest Version

In-secure HTTP Methods are Enabled

Low Risk

Vulnerable

Description

HTTP Methods such as TRACK, TRACE, DEBUG, PUT, DELETE, OPTIONS are intended for debugging or testing purposes. Production environments that allow these HTTP methods can be vulnerable to a range of attacks that are facilitated by these HTTP methods.
The following HTTP method is enabled on the server:

- TRACE - TRACE allows the client to see what is being received at the other end of the request chain and use that data for testing or diagnostic information

Information Disclosure
Impact

Server Configuration
Cause

System Administrator
Responsibility

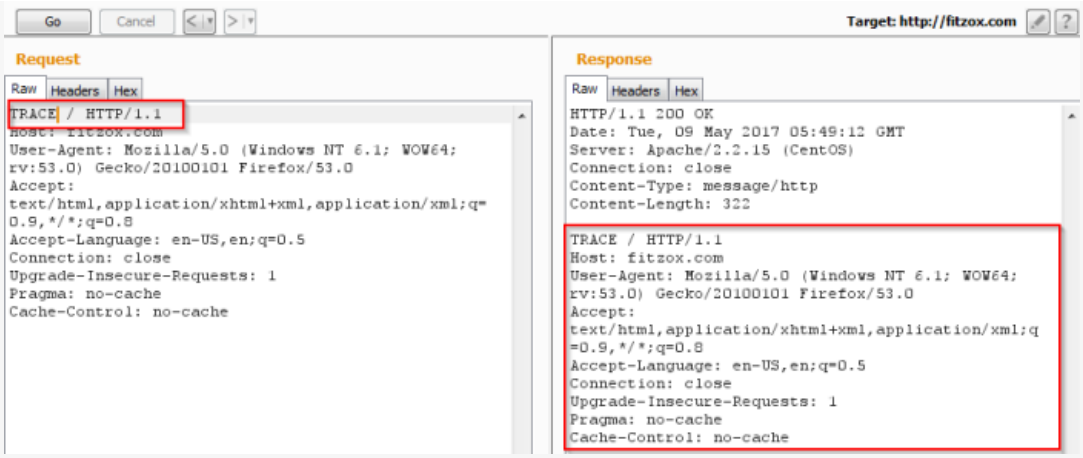
Easy
Difficulty

Critical / CVSS Base Score: **2.6**
(AV:N/AC:H/Au:N/C:P/I:N/A:N)

The above methods would allow an attacker to upload malicious scripts onto the server as well as delete files from the server.

Proof-Of-Concept

While testing the application we were able to find the HTTP TRACE method enabled on the server.



Recommendations

The changes can be made as follows:

Disable HTTP Methods example for APACHE

```
/**
 * @author  Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

IHS and Apache are configured to disable normal TRACE request processing so that the request fails with 403 (forbidden) and any private information sent in the TRACE request does not appear in the response. The way to disable normal TRACE request processing is to add several mod_rewrite directives to the web server configuration file, at main scope as well as in every <VirtualHost > container. Here is an example:
...
# disable TRACE in the main scope of httpd.conf
RewriteEngine On
RewriteCond \%{REQUEST_METHOD ^TRACE
RewriteRule .* - [F]
...
<VirtualHost www.example.com>
...
# disable TRACE in the www.example.com virtual host
RewriteEngine On
RewriteCond \%{REQUEST_METHOD ^TRACE
RewriteRule .* - [F]
</VirtualHost>

mod_rewrite must be active for these directives to be accepted. If mod_rewrite is not already active in your configuration
```

Instances

fitzox.com

URL	Parameters	Status	POC
/	–	Vulnerable	⊘

Reference Documents

APACHE

- DISABLE TRACE AND TRACK METHODS

Cookie without HTTPOnly flag set

Low Risk

Vulnerable

Description

HttpOnly is an additional flag included in a Set-Cookie HTTP response header. If the HttpOnly flag is included in the HTTP response header, the cookie cannot be accessed through client side script. As a result, even if a cross-site scripting (XSS) flaw exists, and a user accidentally accesses a link that exploits this flaw, the browser will not reveal the cookie to a third party.

The cookie value issued by the application does not have the HttpOnly flag set and hence an attacker may use it as leverage with Cross-Site Scripting to gain access sensitive parameter in the cookie value.

User Account Compromise
Impact

Application Misconfiguration
Cause

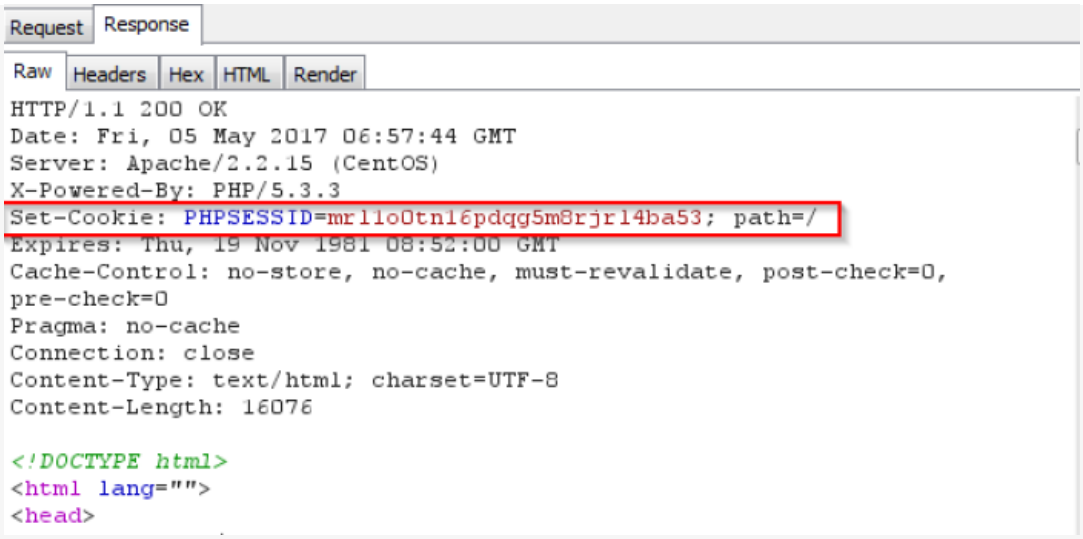
Developer
Responsibility

Medium
Difficulty

Critical / CVSS Base Score: **2.6**
(AV:N/AC:H/Au:N/C:P/I:N/A:N)

Proof-Of-Concept

We found that the cookie does not have HttpOnly flag enabled leaving the cookies vulnerable from scripting attacks.



Recommendations

For Apache use:

set HTTPOnly example for APACHE

```
/**
 * @author   Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

The [CO], or [cookie] flag, allows you to set a cookie when a particular RewriteRule matches. The argument consists of three required fields and four optional fields.
The full syntax for the flag, including all attributes, is as follows:
[CO=NAME:VALUE:DOMAIN:lifetime:path:secure:httponly]
You must declare a name, a value, and a domain for the cookie to be set.
```

For PHP use:

set in php.ini example for PHP

```
/**
 * @author   Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */

session.cookie_httponly = True
```

Instances

fitzox.com

URL	Parameters	Status	POC
/	PHPSESSID	Vulnerable	⊘
fcg.php	PHPSESSID	Vulnerable	⊘

Reference Documents

GENERAL

- HTTP cookie
- HttpOnly

Information Disclosure by Verbose Error Messages

Informational Risk

Vulnerable

Description

The website contains files and directories that result in verbose error messages that disclose information about the internal operation and functioning of the application. Information related to the website such as the framework, languages and other functions are revealed in a verbose error messages. This information provides an attacker a detailed insight about the website.

An attacker is able to trigger verbose error messages on various parts of the server and application. As a result an attacker may be able to view sensitive internal information that may be used to carry out additional attacks.

Information Disclosure
Impact

Server Configuration
Cause

System Administrator
Responsibility

Easy
Difficulty

Critical / CVSS Base Score: **0**
(AV:N/AC:L/Au:N/C:N/I:N/A:N)

Proof-Of-Concept

While testing the application we found that the application was displaying verbose error message.

Recommendations

There are two key aspects to tackling this vulnerability:
1. Writing code with managed errors
2. Disabling error messages in server configuration

Instances

fitzox.com

URL	Parameters	Status	POC
/	–	Vulnerable	⊘

Reference Documents

APACHE

- Replacing Verbose Errors with Custom Messages

Information Disclosure by Test, Old and Backup Files

Informational Risk

Vulnerable

Description

The webserver contains files and directories that have been created for testing or development purposes. These files may disclose information about the website and its functioning. The more information an attacker learns, the easier it becomes for him to compromise the system.

Information Disclosure
Impact

Insufficient Webroot Management
Cause

System Administrator
Responsibility

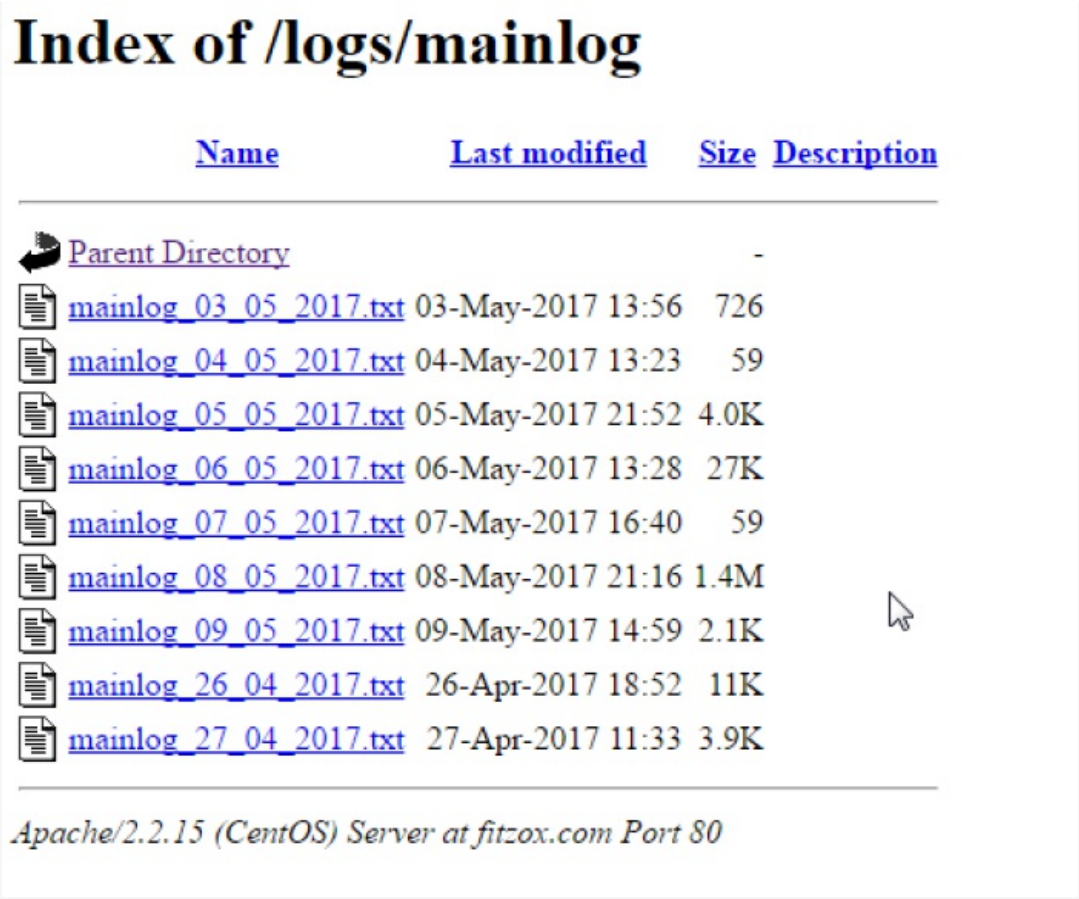
Easy
Difficulty

Critical / CVSS Base Score: **0**
(AV:N/AC:L/Au:N/C:N/I:N/A:N)

The webserver contains files and directories that have been identified as test or debug resources. These files generally contain internal information and have not gone through a thorough release cycle. As a result, they may disclose sensitive information or be vulnerable to attacks and compromise the security of the server.

Proof-Of-Concept

While testing the Application we found that the logs file were accessible pubilcly.



Index of /logs/mainlog

Name	Last modified	Size	Description
Parent Directory		-	
mainlog_03_05_2017.txt	03-May-2017 13:56	726	
mainlog_04_05_2017.txt	04-May-2017 13:23	59	
mainlog_05_05_2017.txt	05-May-2017 21:52	4.0K	
mainlog_06_05_2017.txt	06-May-2017 13:28	27K	
mainlog_07_05_2017.txt	07-May-2017 16:40	59	
mainlog_08_05_2017.txt	08-May-2017 21:16	1.4M	
mainlog_09_05_2017.txt	09-May-2017 14:59	2.1K	
mainlog_26_04_2017.txt	26-Apr-2017 18:52	11K	
mainlog_27_04_2017.txt	27-Apr-2017 11:33	3.9K	

Apache/2.2.15 (CentOS) Server at fitzox.com Port 80

Recommendations

There are several different steps that can be taken to ensure that these vulnerabilities are removed:

1. Implement strong production and development processes to prevent unapproved files from reaching a production environment.
2. Carry out regular audits of the webroot and remove any unnecessary files or directories.

Instances

fitzox.com

URL	Parameters	Status	POC
logs	–	Vulnerable	⊘
test.html	–	Vulnerable	⊘

Reference Documents

GENERAL

- SSL Cache

Cross-Site Request Forgery Token in Header Missing

Informational Risk

Vulnerable

Description

Cross-Site Request Forgery (CSRF) is an attack that allows a hacker to perform an action on the vulnerable site on behalf of the victim. The attack is possible when the vulnerable site does not properly validate the origin of the request.

User Account Compromise
Impact

Insufficient Session Validation
Cause

Developer
Responsibility

Specialized
Difficulty

Critical / CVSS Base Score: **0**
(AV:N/AC:L/Au:N/C:N/I:N/A:N)

The attack is performed by forcing the victim's browser to issue an HTTP request to the vulnerable site. If the user is currently logged-in to the victim site, the request will automatically use the user's credentials (like session cookies, user's IP address, and other browser authentication methods). Using this method, the attacker forges the victim's identity and submits actions on his or her behalf. In other words, the vulnerable site does not take the proper measures to validate that the user indeed wanted to perform the specific action.

This vulnerability allows a remote attacker to carry out malicious transactions and execute requests as the victim without their knowledge or permission.

Forcing the victim to send the unintended request can be done in numerous ways:

- Sending the victim a malicious link to the vulnerable application via email.
- Putting a hot-link (like an image or frame) to the vulnerable site on the hacker's web page.
- Posting a link to the vulnerable site in a public forum.
- Using Cross-Site Scripting or Link Injection vulnerabilities in the site (or another site) and automatically redirecting the browser to the vulnerable site.

Proof-Of-Concept

While testing the application we found that the CSRF token was missing from the fcg.php response header.

Recommendations

In-order to prevent CSRF attacks, it is necessary to implement a unique identifier in every request, which is a parameter that an attacker cannot guess.

One suggested option is to add the session id taken from the session cookie and adding it as a parameter. The server must check that this parameter matches the session cookie, and if not discard the request. The reason an attacker cannot guess this parameter is the "same origin policy" that applies to cookies, so the attacker cannot forge a fake request that will seem real to the server

Anti example for PHP

```
/**
 * @author  Security Brigade InfoSec Private Limited
 * @project  Automated Application Security Audit of Fitzox WAP Application for Coolbox Innovation Studio Private Limited
 */
```

There are many Anti-CSRF Frameworks that can be utilized to prevent this issue, such as: CSRF-Magic.

Instances

fitzox.com

URL	Parameters	Status	POC
/	–	Vulnerable	⊘

Reference Documents

GENERAL

- Server Side Request Forgery – SSRF
- Cross Site Port Attacks
- WASC CSRF

PHP

- Cross-Site Request Forgeries

Potential Clickjacking Vulnerability

Informational Risk

Vulnerable

Description

Click jacking is a client side vulnerability which mainly occur when the attacker is able to frame the website content mostly forms and put some transparent layer over the page in order to trick the user to click on buttons that are not intended by the victim.

For example:- If there is delete profile button on the page,the attacker could easily frame the page and put a transparent layer over the page saying win lottery,the victim will click on the button with the intention of wining the lottery but he/she will lose his account.

Source Code Compromise
Impact

Insecure Configuration
Cause

System Administrator
Responsibility

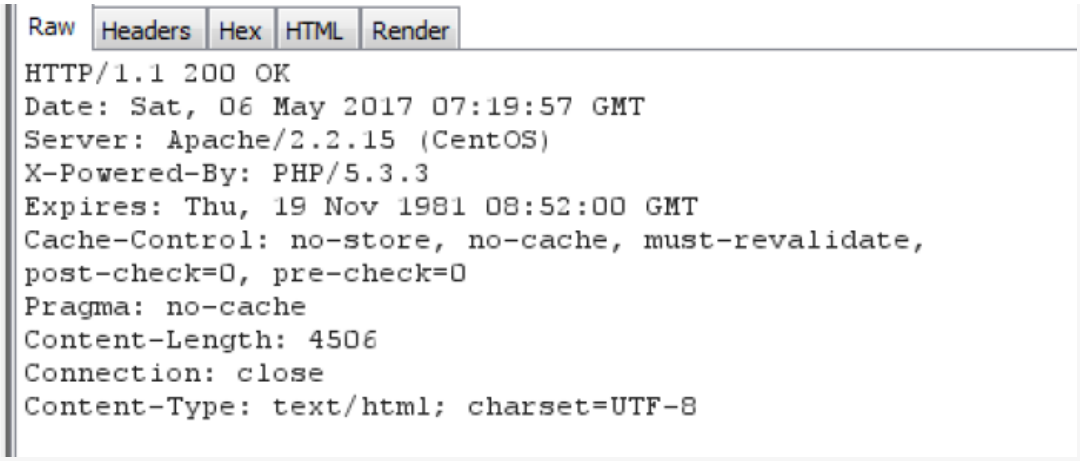
Easy
Difficulty

Critical / CVSS Base Score: **0**
(AV:L/AC:H/Au:M/C:N/I:N/A:N)

It was observed that the X-Frame Option header is not present in the response bodies which helps preventing from Click Jacking vulnerability.Also appropriate frame bursting javascript should be implemented in order
to overcome the cases where some browsers does not support X-Frame-Options header.

Proof-Of-Concept

While testing we found that the X-frame-Options was missing from the response header.



Recommendations

To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself.

Instances

fitzox.com

URL	Parameters	Status	POC
fcg.php	–	Vulnerable	⊘
index.php	–	Vulnerable	⊘
wallpaper.php	–	Vulnerable	⊘
video.php	–	Vulnerable	⊘
getpost.php	–	Vulnerable	⊘

Reference Documents

GENERAL

- Enable anti-clickjacking X-Frame-Options header

Recommendation

Recommendation Summary

1. Implement input validation
2. Disable Directory Listing on the webserver
3. Upgrade to the latest version of PHP
4. Implement and enforce SSL.
5. Implement Validation controls.
6. Upgrade the Apache httpd to the latest stable version.
7. Disable in-secure HTTP methods
8. set HTTPOnly flag
9. Writing code with managed errors
10. Implement strong production and development processes to prevent unapproved files from reaching a production environment
11. Implement Tokenization
12. It is advised to reconfigure the server and enable this option in the response header.

Long-Term Action Plan

Security Brigade Infosec Private Limited recommends the following Action Plan to enhance the long-term security posture at Coolbox Innovation Studio Private Limited.

Actionable Items	Priority
Comprehensive Web Application Penetration Testing	High
Application Malware Scan	High
Penetration Testing Service	Medium
Source Code Security Review	Low