Sri Lanka Institute of Information Technology

IT21299902 (ZAKEY M.S.M.A)

Bug bounty Report 06.

Domain: Inmobi.com

Web security – IE2062

B.Sc. (Hons) in Information Technology Specialization in cyber security.

Declaration:

- I hereby certify that no part of this assignment has been copied from any other work or any other source. No part of this assignment has been written/produced for me by another person.

- I hold a copy of this assignment that I can produce if the original is lost or damaged.

1. Sensitive data exposure.

- **PHP source code disclosure.**

I have identified many vulnerabilities in under domain www.inmboi.com/. I figured out it has vulnerabilities that listed by OWSAP Top 10. And the overall website risk level is **high**. I used net sparker to identify vulnerabilities in the following domain.

I hope this message finds you well. I am writing to report a vulnerability I discovered under the domain of **https://www.inmobi.com/company/press** during my participation in the bug bounty program. Please find below the details of the vulnerability for your review and further action.

Vulnerability title: PHP Source code disclosure

Severity: Medium

OWASP classification 2013: A3

CVSS 3.0 score: - 5.3

CVSS 3.1 score: - 5.3

CVSS string: - CVSS: 3.1/AV: N/AC: L/PR: N/UI: N/S: U/C: L/I: N/A: N

Effected components: Database.

Configuration files.

Source code files.

All the user and system accounts.

Date of Discovery: 2023/05/10

Date of Report: 2023/05/12

2. Vulnerability Description.

I have identified a possibility of PHP source code disclosure within the affected system. If this PHP code belong to any of importance sector of the web application intruders can get a massive advantage of it. They can do data manipulation, data theft, system hijack and so many attacks.

3. Impact assessment.

Under this OWSAP category I have identified many vulnerabilities like sensitive data exposure that can occur due to bad implementation of security implementations under domain of **https://www.inmobi.com/company/press**

From this scan I was able to find out a leakage of PHP code. The effect depends in the type of the source code that leaked it can be database connection string, username and passwords, the internal workings and business logic of the application might be revealed. In which information leaked attacker can perform these kinds of attacks.

1. the database or other data sources can be accessed. It may be able to read, update, or delete arbitrary data from the database depending on the privileges of the account determined from the source code.

2. To fully operate the application, acquire access to password-protected administrative tools including dashboards, management consoles, and admin panels.

3. Investigate the source code for input validation issues and logic flaws to develop additional attacks.

4. The exposed source code can be altered by attackers to add backdoors, logic bombs, or other malicious code. Because of this, they may be able to obtain unauthorized access, alter data, take unlawful action, or even launch an attack on users or other systems using your web application.

5. Leaked source code can reveal sensitive data, including API keys, database logins, encryption keys, and other private information. Attackers may use this information to launch additional assaults, obtain illegal access to databases, or jeopardize the security of other systems linked to your web application.

Because of this vulnerability marked as medium It’s a must to address it with a good solution.

4. Steps to Reproduce.

1. Find the specific PHP file or exposed PHP code by using code analysis tools, code reviews, or log analysis.

2. To find the main reason for the source code leak, employ security scanning tools or a manual inquiry. Misconfigured server settings, unsafe file permissions, an application vulnerability, or other elements could be to blame. OWASP ZAP, Burp Suite, and Nikto are a few examples of tools that might assist in spotting potential vulnerabilities.

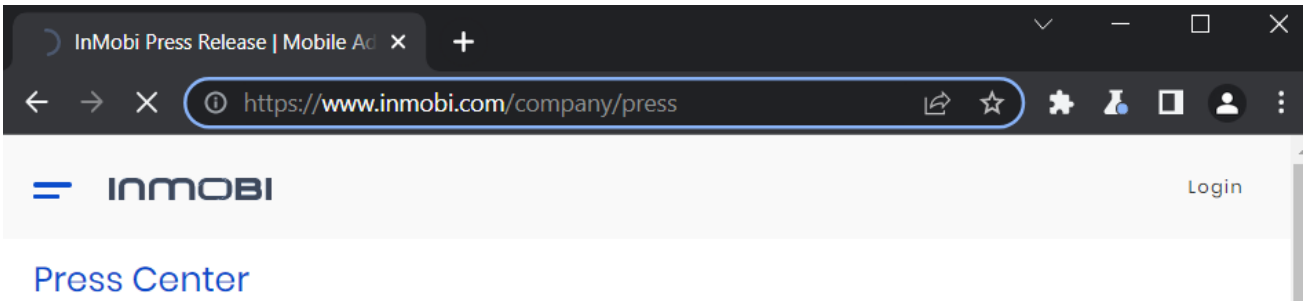
3. Utilize applications like VirtualBox, VMware, or Docker to create a local or isolated testing environment. You can use this to replicate and examine the vulnerability without having an adverse effect on the production system.

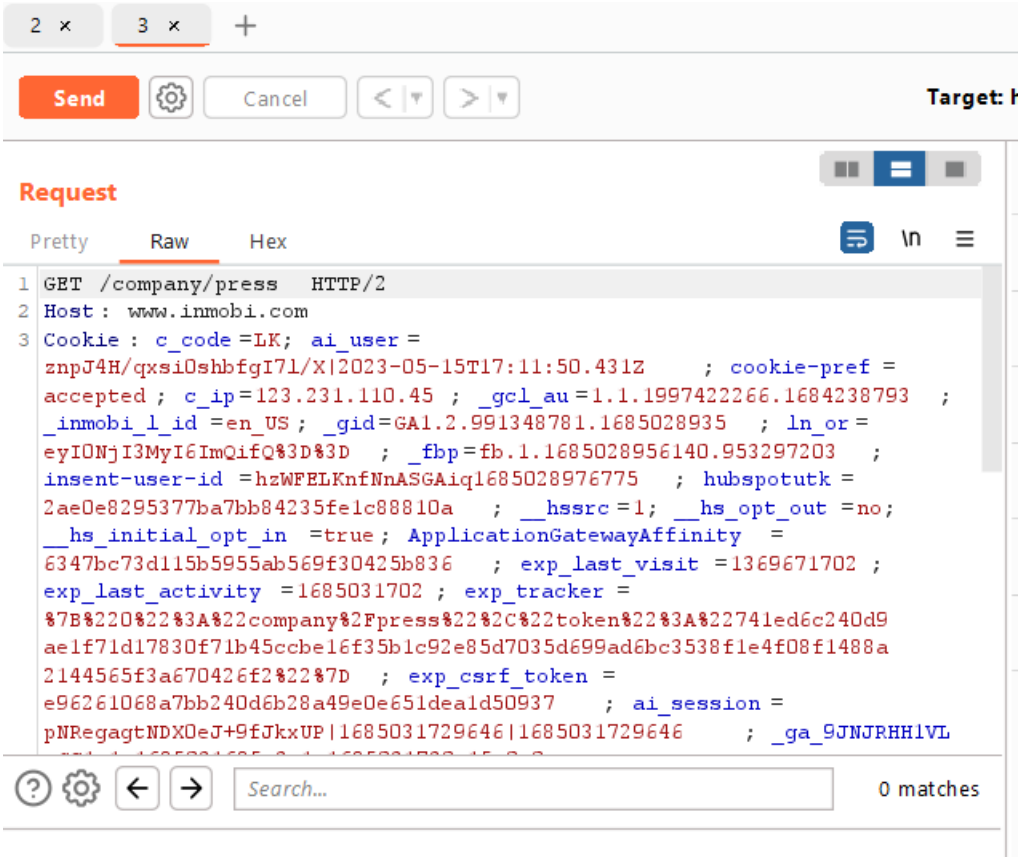
4. To duplicate the situation in which the source code fragment is exposed, modify the server settings, or take advantage of the vulnerability. To exploit vulnerabilities or replicate misconfigurations, you can utilize programs like Metasploit, OWASP ZAP, or your own scripts.

5. Access the weak PHP file or exposed fragment using a web browser or an HTTP client program like cURL or Postman. In the address bar or as part of the request, type the URL or path to the PHP file.

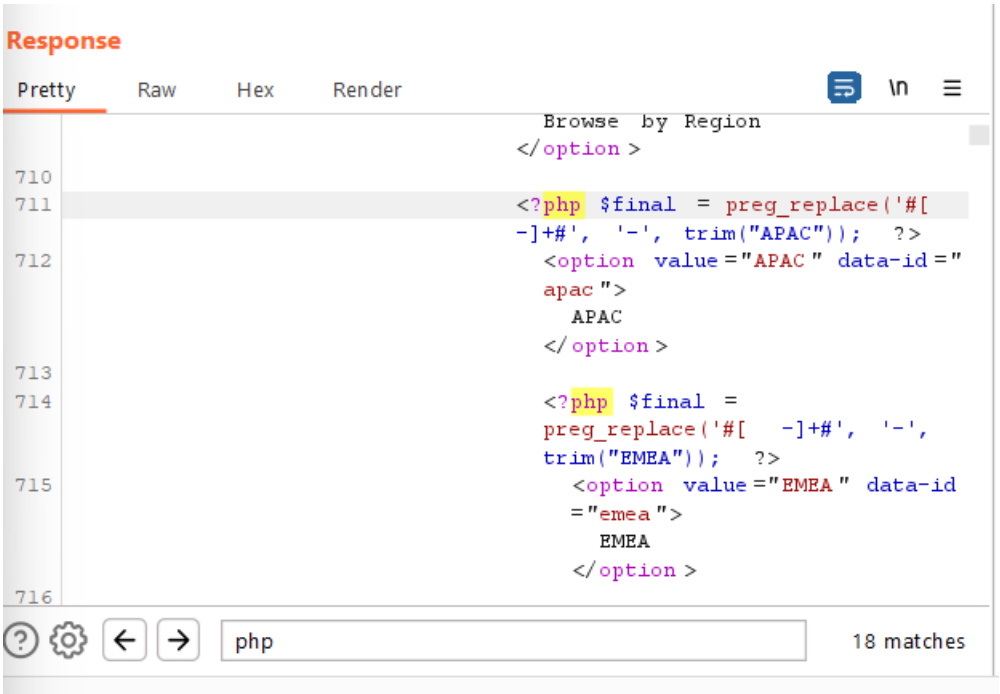
6. Verify that the PHP source code snippet is available and leaked. Examine the response in the browser or use programs like Burp Suite, ZAP, or Wireshark to analyze the server response. Verify that the code is being exposed as expected by looking at the content the server has returned.

5. Proof of concept.





Leaky PHP code



Identified Source Code

```
<?php $final = preg_replace('#[ -]+#', '-', trim("APAC")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("EMEA")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("Europe")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("Global")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("India")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("LATAM")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("MENA")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("North America")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("SEA")); ?>
…
<?php $final = preg_replace('#[ -]+#', '-', trim("APAC")); ?>
…
<?p...
```

Request

```
GET /company/press HTTP/1.1
Host: www.inmobi.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Cookie: c_code=LK; c_ip=116.206.245.109; ai_user=XJhwp5xxtTsAdHzSWDbpg5|2023-05-05T14:53:28.137Z; ai_se
ssion=R3UgM74Rq7bY17UmmXo+Ak|1683298408159|1683298484523; ApplicationGatewayAffinity=a2208e72bc92267e73
2a00a46c4d421c; exp_csrf_token=8e0a53e214129f87818cafdac2a6c1e3b0057cfd; exp_last_activity=1683298518;
 exp_last_visit=1367938472; exp_tracker=%7B%220%22%3A%22rss%2Finsights%22%2C%221%22%3A%22rss%2Fpress%2
2%2C%222%22%3A%22rss%2Fpress%2FN3TSP4RKE2%22%2C%223%22%3A%22rss%2Fpress%2Fnxtspxrkex%22%2C%22token%22%3
A%22ee7a1a0b00535968ed6c9fb381e573c2e12402ac5f83c7b0771d8f96644dd5c6ddb6fb470accc9f458fa8e22784b0f59%2
2%7D
Referer: https://www.inmobi.com/
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.
77 Safari/537.36
X-Scanner: Netsparker
```

Response

Response Time (ms) : 6494.2685 Total Bytes Received : 560662 Body Length : 559432 Is Compressed : No

```
HTTP/1.1 200 OK
Set-Cookie: ApplicationGatewayAffinity=c4e775a69ade163c9ae3e4cb423ea79b; Path=/
Set-Cookie: exp_last_visit=1367938472; expires=Sat, 04-May-2024 14:55:19 GMT; Max-Age=31536000; path=/;
 HttpOnly; SameSite=Lax
Set-Cookie: exp_last_activity=1683298519; expires=Sat, 04-May-2024 14:55:19 GMT; Max-Age=31536000; path
=/; HttpOnly; SameSite=Lax
Set-Cookie: exp_tracker=%7B%220%22%3A%22company%2Fpress%22%2C%221%22%3A%22rss%2Finsights%22%2C%222%22%3
A%22rss%2Fpress%22%2C%223%22%3A%22rss%2Fpress%2FN3TSP4RKE2%22%2C%224%22%3A%22rss%2Fpress%2Fnxtspxrkex%2
2%2C%22token%22%3A%22fe4eb58f929b17238d28614bb9ef2117542855655c705adf001ec8bed45641c7de6bd4f7ddf00903ab
f7cbdeb62c757e%22%7D; path=/; HttpOnly; SameSite=Lax
Set-Cookie: exp_csrf_token=8e0a53e214129f87818cafdac2a6c1e3b0057cfd; expires=Fri, 05-May-2023 16:55:19
 GMT; Max-Age=7200; path=/; HttpOnly; SameSite=Lax
Server: nginx
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Last-Modified: Fri, 05 May 2023 14:55:22 GMT
Pragma: no-cache
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Content-Encoding:
X-Forwarded-For: 116.206.
…
```

```
<select class="select-dropdown">
<option value="Browse by Region" data-id="browse-by-regions">Browse by Region</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("APAC")); ?>
<option value="APAC" data-id="apac">APAC</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("EMEA")); ?>
<option value="EMEA" data-id="emea">EMEA</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("Europe")); ?>
<option value="Europe" data-id="europe">Europe</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("Global")); ?>
<option value="Global" data-id="global">Global</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("India")); ?>
<option value="India" data-id="india">India</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("LATAM")); ?>
<option value="LATAM" data-id="latam">LATAM</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("MENA")); ?>
<option value="MENA" data-id="mena">MENA</option>
```

```
<?php $final = preg_replace('#[ -]+#', '-', trim("North America")); ?>
<option value="North America" data-id="north-america">North America</option>

<?php $final = preg_replace('#[ -]+#', '-', trim("SEA")); ?>
<option value="SEA" data-id="sea">SEA</option>

</select>
</div>
</div>

<!-- Mobile Filter -->
<div cl
…
</div>
<div class="filter-content">
<h4>Region</h4>
<ul class="filter-list" id="reg-filter">

<?php $final = preg_replace('#[ -]+#', '-', trim("APAC")); ?>
<li data-id="apac"><span>APAC</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("EMEA")); ?>
<li data-id="emea"><span>EMEA</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("Europe")); ?>
<li data-id="europe"><span>Europe</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("Global")); ?>
<li data-id="global"><span>Global</span></li>
```

```
<?php $final = preg_replace('#[ -]+#', '-', trim("India")); ?>
<li data-id="india"><span>India</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("LATAM")); ?>
<li data-id="latam"><span>LATAM</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("MENA")); ?>
<li data-id="mena"><span>MENA</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("North America")); ?>
<li data-id="north-america"><span>North America</span></li>

<?php $final = preg_replace('#[ -]+#', '-', trim("SEA")); ?>
<li data-id="sea"><span>SEA</span></li>

</ul>
</div>
</div>
<div class="filter-submit"><i
…
```

6. Proposed mitigation/fix.

As solutions there are lots of ways that can achieve the expected security about source code disclosure. Among them there are some methods that can be mentioned as good remedy methods.

1. Verify precisely which portions of the source code are really leaked; in certain cases, this may not be feasible because of the restrictions of this sort of vulnerability. Verify that this feature was not intended.

2. Remove any sensitive data from the source code files, including API keys, database logins, and other private information. Use the necessary techniques (e.g., environment variables) to incorporate sensitive information in the PHP files by storing it in a secure configuration file that isn't in the web root directory.

3. If it's a file that the application needs, modify its permissions to bar anyone from accessing it. Remove it from the web server if it isn't.

4. Verify that all recent security fixes have been installed on the server.

5. Delete all backup and temporary files from the web server.

6. Use secure code deployment techniques to reduce the possibility of source code leakage. Using secure file transfer protocols (such as SFTP and SCP) and adhering to secure coding standards, such as input validation, output encoding, and defense against code injection attacks, are examples of how to do this.