Application Security
Vulnerabilities

A-Web Server info.php / phpinfo.php Detection

The remote web server contains a PHP script that is prone to an information disclosure attack.

Description

Many PHP installation tutorials instruct the user to create a PHP file that calls the PHP function 'phpinfo()' for
debugging purposes. Various PHP applications may also include such a file. By accessing such a file, a remote
attacker can discover a large amount of information about the remote web server, including :

- The username of the user who installed PHP and if they are a SUDO user.
- The IP address of the host.
- The version of the operating system.
- The web server version.
- The root directory of the web server.
- Configuration information about the remote PHP installation.

Solution:-

Protect or remove any phpinfo.php or other sensitive files from your server. As those files will have important information.

The best solution is to protect your phpinfo and other sensitive files with htaccess.
This is a powerful technique that combines the power of Apache with the flexibility of PHP. Enabling you to do things like log all unwanted traffic, send email reports for blocked requests, create a UI to display logged data, and just about anything else you can imagine.

```
# protect phpinfo
<Files php-info.php>
        Order Deny,Allow
        Deny from all
        Allow from <Your IP>
</Files>
```

B-Web Application Potentially Vulnerable to Clickjacking

The remote web server may fail to mitigate a class of web application vulnerabilities.

Description:-

The remote web server does not set an X-Frame-Options response header or a Content-Security-Policy 'frame-
ancestors' response header in all content responses. This could potentially expose the site to a clickjacking or UI

redress attack, in which an attacker can trick a user into clicking an area of the vulnerable page that is different than
what the user perceives the page to be. This can result in a user performing fraudulent or malicious transactions.
Content-Security-Policy (CSP) has been proposed by the W3C Web Application Security Working Group, with
increasing support among all major browser vendors, as a way to mitigate clickjacking and other attacks.

Solution:-
Use **X-FRAME-OPTIONS** to avoid your website being hacked from Clickjacking.
The **X-Frame-Options** in HTTP response header can be used to indicate whether or not a browser should be allowed to open a page in frame or iframe.

There are three possible directives for X-Frame-Options:
X-Frame-Options: DENY
X-Frame-Options: SAMEORIGIN
X-Frame-Options: ALLOW-FROM https://example.com/

DENY
The page cannot be displayed in a frame, regardless of the site attempting to do so.
SAMEORIGIN
The page can only be displayed in a frame on the same origin as the page itself. The spec leaves it up to browser vendors to decide whether this option applies to the top level, the parent, or the whole chain.
ALLOW-FROM *uri*
The page can only be displayed in a frame on the specified origin.

-Add following line in Apache Web Server's httpd.conf file
Header always append X-Frame-Options SAMEORIGIN

-Implement in shared web hosting

If your website is hosted on shared web hosting, then you won't have permission to modify httpd.conf. However, you can achieve this by adding following line in **.htaccess** file.
Header append X-FRAME-OPTIONS "SAMEORIGIN"

You can also use an online tool –Header Checker to verify.