Vulnerabilities Detected:

1. Web Server info.php / phpinfo.php Detection

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

Phpinfo Outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

The remote web server contains a PHP script that is prone to an information disclosure attack.By accessing such a file, a remote attacker can discover a large amount of information about the remote web server, including :
1.The username of the user who installed PHP and if they are a SUDO user.
2.The IP address of the host.
3.The version of the operating system.
4.The web server version.
5.The root directory of the web server.
6.Configuration information about the remote PHP installation.

This problem can be tackled by:
1.Remove phpinfo files that are on the server that are prone to attack.
2.Following HTAccess code can be added to ensure that the files can be accessed from specified IP only.

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

B.Web Application Potentially Vulnerable to Clickjacking

Synopsis: 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.X-Frame-Options has been proposed by Microsoft as a way to mitigate clickjacking attacks and is currently supported by all major browser vendors. 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. The 'frame-ancestors' policy directive restricts which sources can embed the protected resource.
Note that while the X-Frame-Options and Content-Security-Policy response headers are not the only mitigations for clickjacking, they are currently the most reliable methods that can be detected through automation. Therefore, this plugin may produce false positives if other mitigation strategies (e.g., frame-busting JavaScript) are deployed or if the page does not perform any security-sensitive transactions.

Solution:

Clickjacking (also known as user-interface or UI redressing) is an exploit in which malicious coding is hidden beneath apparently legitimate buttons or other clickable content on a website.

One of the best ways to go about preventing clickjacking on your site it to include an x-frame-options HTTP header that prevents your site’s content from being loaded in a frame (<frame> tag) or iframe (<iframe> tag). Because these are often used as attack vectors — not just for clickjacking, but for other threats as well — this is an effective way of mitigating the threat.
There are three possible values for X-frame options:
1.DENY: The page cannot be displayed in a frame, despite of the site attemting to do so.
2.SAMEORIGIN: The page can be displayed in the frame on the same origin as itself.

3.ALLOW-FROM uri: The page can only be displayed in the frame of specified origin.

Cross-Site scripting can also help in reducing clickjacking. Web application firewalls can also prevent someone from interjecting the website and inputting code. Implementing strong email spam filter can also help in preventing clickjacking.