# Nessus Scan of endpoint system:

Whilst not a realistic possibility outside of an enormous firewall misconfiguration, I started by installing Nessus essentials and running a scan against the host system to get a baseline for possible vulnerabilities that were evident to Nessus.

This bought back the following:

**CRITICAL:**

**Apache 2.4.x < 2.4.56 Multiple Vulnerabilities**

* CVE-2023-25690
* CVE-2023-27522

Resolution Provided:

Upgrade Apache to version 2.4.56 or later

(installed Version 2.4.53)

**PHP 8.1.x < 8.1.22 Multiple Vulnerabilities**

The Nessus output for this risk references the following CVE’s

* CVE-2023-3824
* CVE-2023-3823
* CVE-2022-31627
* CVE-2022-37454
* CVE-2022-31630

**HIGH:**

**Apache 2.4.x < 2.4.58 Multiple Vulnerabilities**

The Nessus output for this risk references the following CVE’s

* CVE-2023-31122
* CVE-2023-43622
* CVE-2023-45802

Resolution Provided:

Upgrade Apache server to 2.4.58 or later

**SSL Certificate Signed Using Weak Hashing Algorithm**

An attacker can exploit this to generate another certificate with the same digital signature, allowing an attacker to masquerade as the affected service.

Resolution Provided:

Contact Certificate Authority to have a new SSL certificate reissued signed with a more secure hashing algorithm.

**OpenSSL 1.1.1 < 1.1.1t Multiple Vulnerabilities**

Nessus Output for this risk references the following CVE’s

* CVE-2023-0286
* CVE-2023-0215
* CVE-2022-4450
* CVE-2022-4304

Solution Provided:

Upgrade to OpenSSL version 1.1.1t or later

**PHP 8.1x < 8.1.7**

The version of PHP installed on the machine is version 8.1.6 affected by at least 2 CVE’s

* CVE-2022-31625
* CVE-2022-31626

Solution Provided:

Upgrade to PHP version 8.1.7 or later

## Nessus Scan Conclusion

There are several misconfigurations with this system, primaryily out of date software that present some security risks that a motivated enough attacker could use to gain further access and control of the system in question.

# WPScan

Utilising the unix command line tool wpscan I was able to get some information back from the site that from experience should not be available.

* PHP Information page
* Administrative Username
* Directory locations
* Plugin information

## NiktoScan

Results of NiktoScan:

+ Server: Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6

+ Retrieved x-powered-by header: PHP/8.1.6

+ The anti-clickjacking X-Frame-Options header is not present.

+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS

+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

+ Root page / redirects to: http://10.0.0.3/dashboard/

+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST

+ OSVDB-3268: /img/: Directory indexing found.

+ OSVDB-3092: /img/: This might be interesting...

+ OSVDB-3268: /icons/: Directory indexing found.

+ OSVDB-3233: /icons/README: Apache default file found.

## Site Access

**Administrator Brute force:**

The administrative username sbsiteadmin was identified by wpscan.

Due to the configuration of the login page, the site tells you when a username is incorrect next to the password. The default Administrator user has been removed, however the site will inform the user that ‘Username is not found’ If the correct username is entered the user only gets a message about the password not being correct. This gives confidence that a username has been correctly identified and only the password needs to be collected.

The password field does not have an attempt timeout, with a little bit of experience a brute force would be easy to attempt and complete on this site with the administrative username.

This is concerning as the administrative user is expected to have full access to the site and the site itself is expected to have confidential information in it’s directories (more on that later)

Potential Resolutions:

Investigate hiding the login attempts of users on this site so that tools such as wpscan are not able to identify usernames

Add a password attempt limit to the login page

**PHP info page publicly accessible:**

Wpscan found that the phpinfo page is still available:

<http://10.0.0.3/dashboard/phpinfo.php>

This provides a substantial amount of information to an attacker giving them a significant advantage in finding and leveraging exploits.

Potential Resolution:

Remove / hide / move this page from the webserver

**XML-RPC enabled:**

While XML-RPC offers simplicity and ease of use, it is important to know the security risks associated with its implementation. Vulnerability in XML-RPC allows an attacker to make a system call, which can be dangerous for the application and servers.

**Uploads has listing enabled**

Site directories are navigable from the browser, wpscan provided launch to /wp-content/uploads which contains multiple files including temp files, one titled as CV which if it was legible would be a large data breach without consent.

I was not able to get substantial data from these files, due to not being a pen tester however I believe that there is useful data for an attacker within the files as both files hexdump consists of “Administrator” although possible conjecture, a motivated and skilled attacker would certainly be able to get more from this data than me.

The resolution is to make sure this directory is not listed publicly and encrypt sensitive data.

**WPCron Enabled**

**https://hackerone.com/reports/1888723**

The WordPress application is vulnerable to a Denial of Service (DoS) attack via the wp-cron.php script. This script is used by WordPress to perform scheduled tasks, such as publishing scheduled posts, checking for updates, and running plugins.

An attacker can exploit this vulnerability by sending a large number of requests to the wp-cron.php script, causing it to consume excessive resources and overload the server. This can lead to the application becoming unresponsive or crashing, potentially causing data loss and downtime.

# Closing Thoughts

* A motivated attacker would easily be able to access this system due to the found vulnerabilities and access misconfigurations presented.
* The ease of finding the administrative username and the lack of protections around password usage is the single largest risk to the wordpress installation presented.
* Other vulnerabilities and available exploits while available are unlikely to be used with this door so open
* Other exploits available such as the potential to DDoS and access the site through xmlrpc and WPCron
* thorough review of required plugins and their configuration / update status would be required.
* The machine itself gives away too much information freely with the phpinfo page and several of the major software packages in use are out of date and have known vulnerabilities all of these need updating.
* The machine should not be able to be scanned, both the machines in use here are on the same network so it’s not possible to say whether the wordpress installation is secure from wpscan / niktoscan / Nessus however if this was a public site I would suggest that it is made not possible through proxying and firewalls