# Introduction

This document uncovers critical bugs and vulnerabilities found in the Malwarebytes antivirus application and website. By sharing these discoveries, the aim is to empower the Malwarebytes company to enhance its antivirus software and website. This document has already been sent to their support team. Upon installing Malwarebytes antivirus version 4.5.0.152, I noticed programmer comments in certain files, unintended for release. Additionally, I found poorly written code fragments in DLL files, warranting optimization. To protect sensitive data, plain text, and code segments should be encoded or obfuscated in compiled files. Within this document, four vital vulnerabilities affecting the antivirus application are outlined, presenting opportunities for hackers to gain unauthorized access or execute malicious actions. Furthermore, I've identified information and vulnerabilities related to two Malwarebytes website subdomains. By addressing these issues, Malwarebytes can bolster their defenses, ensuring a safer digital environment for their users. Together, we strengthen the shield against cyber threats. **This document was created in January 2022.**

Contents

[Introduction 1](#_Toc93354208)

[arwlib.dll 3](#_Toc93354209)

[BrowserSDKDLL.dll 4](#_Toc93354210)

[CloudControllerImpl.dll 4](#_Toc93354211)

[mbam.exe 5](#_Toc93354212)

[mbam.manifest.json 6](#_Toc93354213)

[mbamelam.inf 6](#_Toc93354214)

[User Interface Bug 7](#_Toc93354215)

[UpdateControllerImpl.dll 7](#_Toc93354216)

[UIPlugin.dll 8](#_Toc93354217)

[TrayPlugin.dll 8](#_Toc93354218)

[sample.dll 8](#_Toc93354219)

[Web Site User Interface Bug 9](#_Toc93354220)

[Malwarebytes Antivirus Application Vulnerabilities 11](#_Toc93354221)

[1-Simple Uninstallation 11](#_Toc93354222)

[2-Custom UAC Dialog 11](#_Toc93354223)

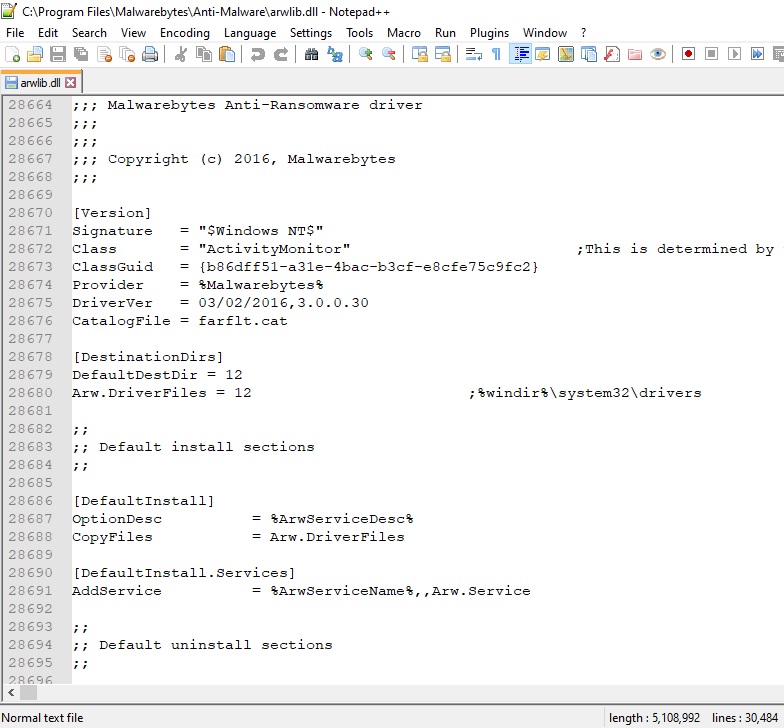
[3-Validating Mechanism for Installation Files 11](#_Toc93354224)

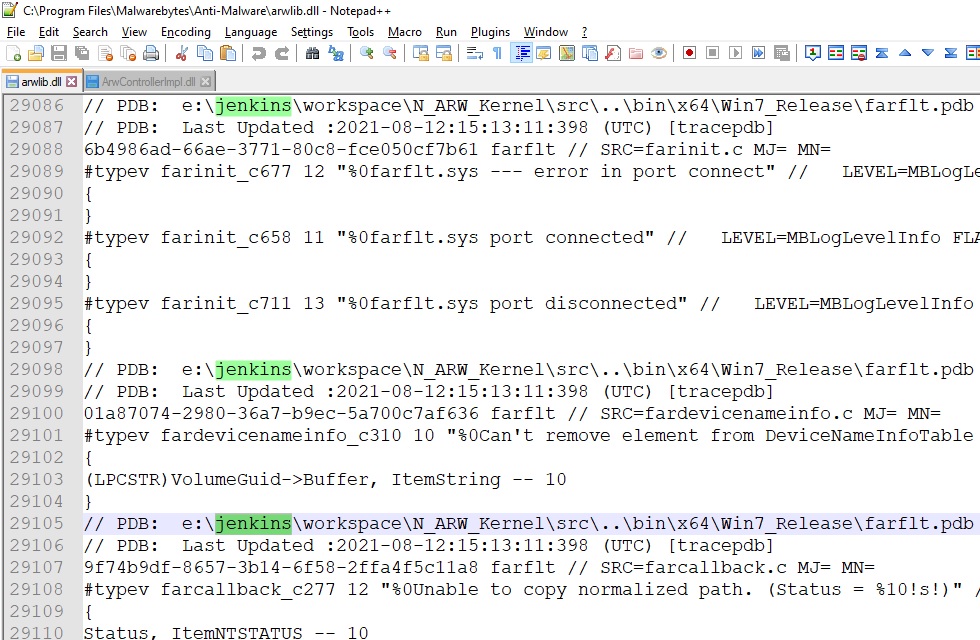
[4-Links and information Vulnerability 12](#_Toc93354225)

[Vulnerabilities in Press.malwarebytes.com and Blog.malwarebytes.com 13](#_Toc93354226)

# arwlib.dll

If you open the arwlib.dll file with Notepad++, from line 28664 to 29892, you can see some code and plain text. Perhaps, they were added by the compiler or other tools during the compile time, but I believe they shouldn't be present in the released file. This text contains valuable information, such as its association with the Malwarebytes Anti-Ransomware driver, usage in the Jenkins pipeline, etc. Visual references can be observed in the pictures below:





# BrowserSDKDLL.dll

If you open the BrowserSDKDLL.dll file with Notepad++, from line 23063 to 24994, you can observe a combination of mixed binary and plain text. This information may provide valuable insights, including the utilization of the Golang descriptor library, integration of chromium, involvement in the Jenkins pipeline, and more.

# CloudControllerImpl.dll

If you open the CloudControllerImpl.dll file with Notepad++, you can observe a combination of mixed binary and plain text, containing valuable data. This data includes the usage of the Jenkins pipeline and references to 35 CPP files and several header files. These files are associated with the implementation of sending and receiving HTTP requests and other functionalities.

For example:

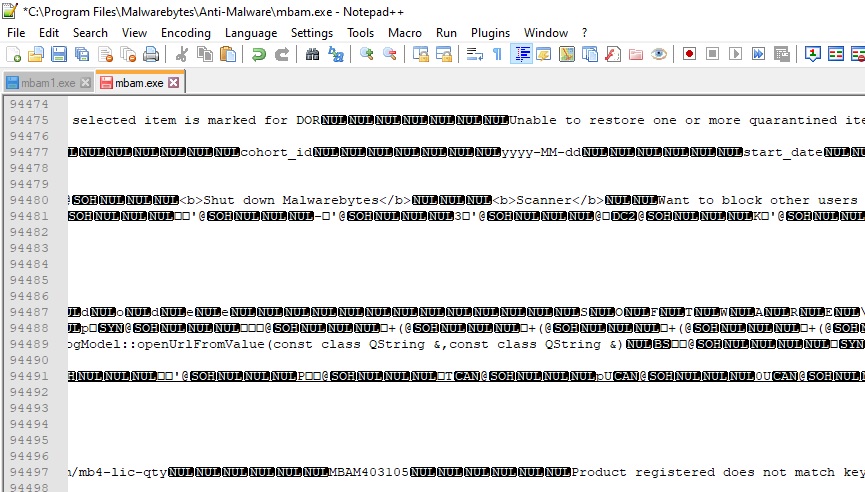
* d:\jenkins\workspace\n\_poco-vc120\poco-1.9.0\net\src\httpchunkedstream.cpp
* d:\jenkins\workspace\n\_poco-vc120\poco-1.9.0\foundation\include\poco\scopedlock.h

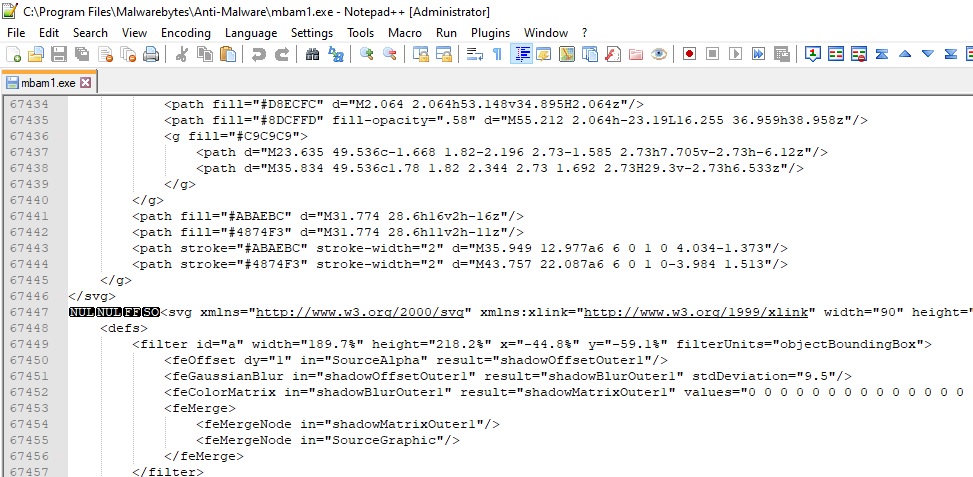
It is worth noting that similar data can be found in almost all Malwarebytes installed files. Malwarebytes utilizes third-party projects, as mentioned in the document at https://www.malwarebytes.com/pdf/guides/thirdpartylicensesupplement.pdf. Therefore, some of the data may be related to these third-party projects.

# mbam.exe

If you open the mbam.exe file with Notepad++, you will come across an abundance of plain text and code, primarily associated with application settings and user interface components. It is perplexing why this information remains in plain text instead of being in binary or encoded formats.

Within this file, you can also find some Malwarebytes messages presented in plain text or HTML format. As an illustration, please refer to the image below:

****



# mbam.manifest.json

This JSON file includes the below text:

{

"name": "mbambgnativemsg.exe",

"description": "MBAM Native host",

"path": "C:\\Program Files\\Malwarebytes\\Anti-Malware\\mbambgnativemsg.exe",

"type": "stdio",

"allowed\_origins": ["chrome-extension://ihcjicgdanjaechkgeegckofjjedodee/"]

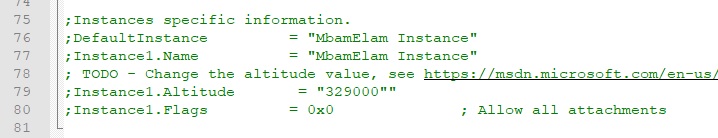
}

You can see the chrome-extension://ihcjicgdanjaechkgeegckofjjedodee/ parameter. This value is related to the Malwarebytes browser guard extension in the below link : <https://chrome.google.com/webstore/detail/malwarebytes-browser-guar/ihcjicgdanjaechkgeegckofjjedodee>

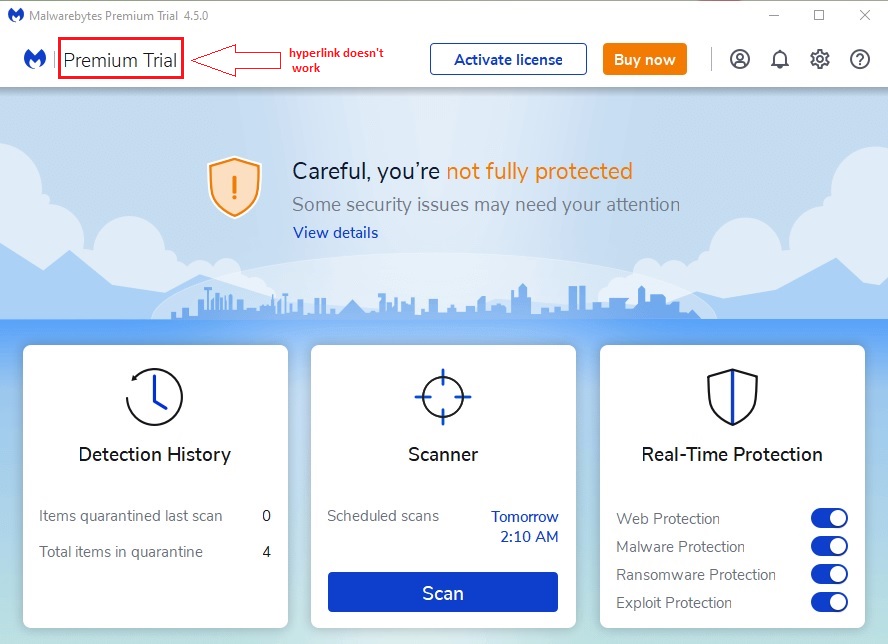
If this value is important, you shouldn’t save it on a JSON file, it can be changed very easily.

# mbamelam.inf

In the mbamelam.inf file, you will find specific data about the Malwarebytes service (driver). However, at the end of this file, there is a concerning TODO comment left by a programmer for future tasks, which includes a reference to a Microsoft link. It is crucial that such TODO comments are not present in the release project. Please refer to the image below for a visual representation:



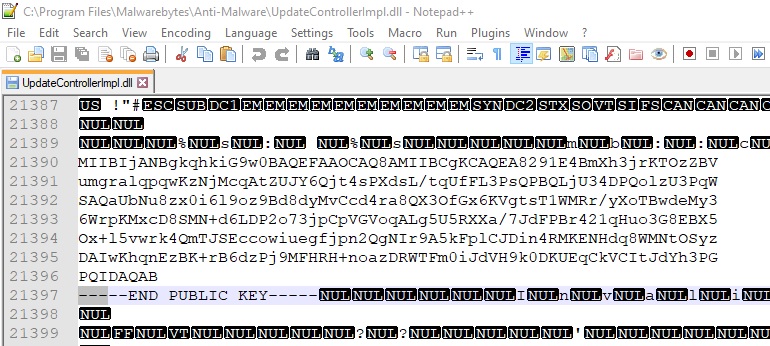
# User Interface Bug



As indicated by the red rectangle in the image above, the "Premium Trial" component serves as a hyperlink, designed to redirect users to the main form from any section within the application. However, it appears that this hyperlink is not functional, as hovering the mouse pointer over it does not display the hand pointer. It is highly likely that another transparent UI component is overlaying this hyperlink, causing the issue. Further investigation is needed to resolve this matter and restore the functionality of the "Premium Trial" hyperlink.

# UpdateControllerImpl.dll

In the UpdateControllerImpl.dll file, you can see an RSA public key.



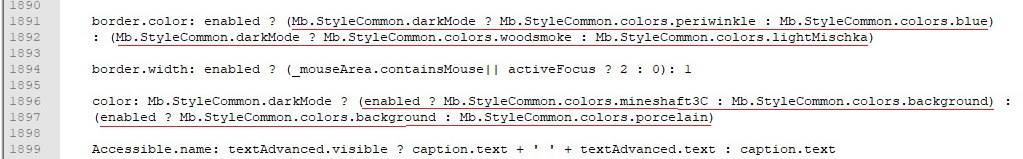
# UIPlugin.dll

Within the UIPlugin.dll file, a significant amount of code is dedicated to the application's user interface and themes. However, some of this code appears to be poorly written. For instance, numerous hard-coded image paths are present, which would benefit from a more structured approach. A recommended solution involves creating a struct or a class to centralize these paths, providing better maintainability and organization for each theme.

Furthermore, the file contains numerous conditions written using the "?: " operator and some of them are quite complex. This approach might lead to difficulties in the future, especially when adding new themes. With each new theme addition, additional conditions will be required, leading to code growth and increasing the chances of introducing bugs.

To address these concerns, consider implementing design patterns such as the "Chain of Responsibility." By adopting appropriate design patterns, you can effectively manage theme-related conditions and streamline the codebase. Embracing such practices will enhance the maintainability and scalability of the application, especially as it continues to grow and evolve.

Indeed, it is worth noting that the code may have taken on this particular form after the compilation process.



# TrayPlugin.dll

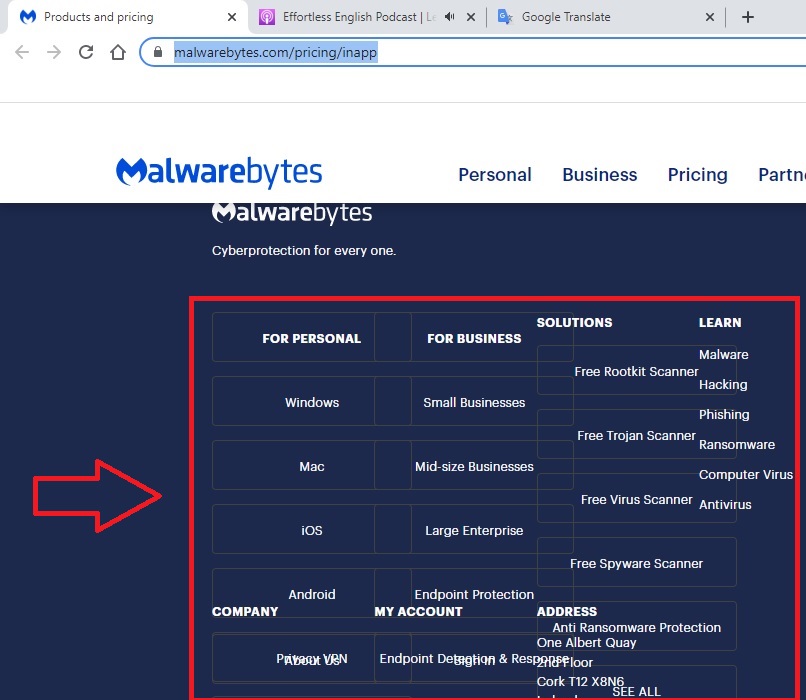
Within the TrayPlugin.dll file, an extensive collection of code can be observed, intricately intertwined with the application's notification and system tray functionalities.

# sample.dll

I'm unfamiliar with this file as well. During one of my experiments, I deleted it and ran Malwarebytes antivirus, and surprisingly, the application worked well without it. However, it's possible that the file serves another purpose. Additionally, I noticed the presence of the phrase "hello world" within the file, leading me to speculate that it might be for testing or some other reason. Further investigation is required to determine its actual function.

# Web Site User Interface Bug

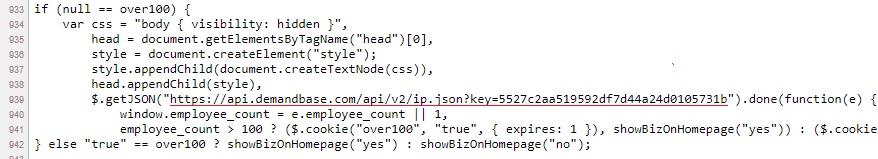
1. In one of my experiments, I developed an application to extract all links from Malwarebytes antivirus installed files. I discovered more than 160 links during this process. Among them, I examined <https://www.malwarebytes.com/pricing/inapp>, and I noticed that the footer of this page lacks a good style and appears messy, especially when viewed on non-mobile devices. It seems to be primarily designed for the mobile application.

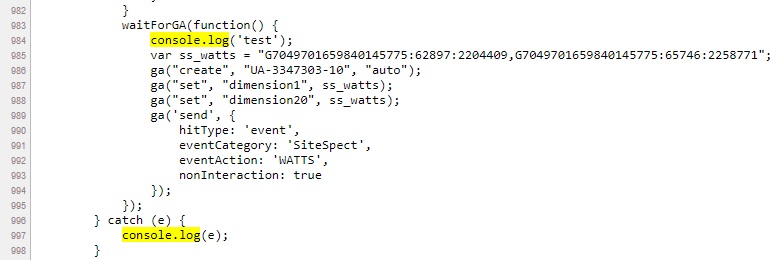


1. Upon checking <https://blog.malwarebytes.com/detections>, I observed a vast amount of data condensed onto a single page, requiring extensive scrolling to view it all. Although not a bug, I believe implementing a grid view with pagination and filtering would greatly enhance user experience.
2. Notably, there are JavaScript code comments present on the support page of the Malwarebytes website: [https://support.malwarebytes.com](https://support.malwarebytes.com/). In my view, comments should not be visible on the released website. Please refer to the image below for a visual representation:



1. On the home page of the Malwarebytes website (https://www.malwarebytes.com), I noticed the presence of console.log () JavaScript codes intended for displaying data in the browser console. These code snippets appear to be associated with development and testing purposes. In my view, for the release website, it is best to exclude such test code, ensuring a cleaner and more secure end-user experience.
2. Additionally, one of the JavaScript functions contains an API with a valid token, which might pose a security risk if exposed on the release website. Please refer to the images below for visual context:





# Malwarebytes Antivirus Application Vulnerabilities

## Simple Uninstallation

## The Malwarebytes antivirus application offers a straightforward uninstallation process. By executing C:\Program Files\Malwarebytes\Anti-Malware\mbuns.exe and selecting the "U" button, users can easily uninstall it. However, this simplicity exposes a potential vulnerability. Malicious actors could develop applications to remove this antivirus before executing harmful software.

## For example, ransomware can be encoded and concealed as a resource within an application. This application first disables the mouse and keyboard, restricting user control, and then proceeds to uninstall Malwarebytes antivirus. Subsequently, the ransomware is executed. In situations where Windows UAC protection is enabled, this malicious software would also need to bypass UAC.

## In contrast, other antivirus applications employ more complex uninstallation methods to mitigate such vulnerabilities. Some use Captcha or other security measures before allowing uninstallation, enhancing protection against malicious attempts.

## Custom UAC Dialog

## When running the Malwarebytes antivirus as an administrator, the UAC dialog does not appear before making any changes. However, if we examine Kaspersky antivirus, it incorporates a custom dialog similar to the Windows UAC dialog. For instance, when attempting to disable ransomware protection, approval is required through this dialog. This feature serves as an excellent defense against malicious software attempting automation activities or unauthorized changes to the antivirus settings.

## Validating Mechanism for Installation Files

## During one of my experiments, I conducted a search within the content of the mbam.exe file for the word "Ransomware" and replaced it with "ransomware". However, upon running the modified file, I encountered an error message stating "Unable to connect service." Upon clicking "OK," the application proceeded to run normally.

## The issue here is apparent. The Malwarebytes antivirus application lacks a mechanism for verifying its own files after execution. In my view, every antivirus application should incorporate such a mechanism. Implementing this functionality is not overly complex but is crucial.

## For instance, consider a scenario where the operating system has Malwarebytes antivirus installed, and someone logs in from a different operating system, safe mode, or live Windows environment and makes changes to some files. Upon reloading the primary operating system with Malwarebytes, certain functionalities may not function correctly.

## One of the solutions to address this vulnerability involves developing a separate Windows service to verify the files. This service can create a hash code database of the genuine files and check them before execution. If any file is found to be altered, the service would report this to the user and the Malwarebytes server. If the user has an internet connection, the service can download the correct file and replace the modified one, ensuring the integrity and security of the application.

## Links and information Vulnerability

## During my development process, I created an application that successfully extracts all links, IPs, and plain text from Malwarebytes antivirus installed files. This endeavor yielded a significant number of links and plain text data. While some of this information may not be a direct threat, it is possible that it could potentially aid a hacker in targeting the Malwarebytes website or circumventing the antivirus protection.

## To maintain a record of the discovered links and relevant data, I saved them in separate files: "links.txt" and "info.txt". These files serve as documentation of the collected information and could assist in further analysis or investigation. It is essential to treat this data with caution and ensure it remains secure to prevent any unintended exploitation.

you can see some of the links in the below:

https://idms-holocron-stage.mwbsys.com/graphql/

<https://telemetry.malwarebytes.com/api/v2/streams/applog/record>

<https://my-device.malwarebytes.com/>

<https://keystone-staging.mwbsys.com/>

<https://block.malwarebytes.com>

<https://myaccount-device-stg.malwarebytes.com/>

<https://lic-iris-content-stage.mwbsys.com/>

<https://iris.mwbsys.com/api/v1/control_statements>

https://75.2.42.184/reputation

https://99.83.150.43/reputation

https://75.2.38.178/reputation

<https://99.83.240.23/reputation>

<https://holocron.mwbsys.com/graphql>

<https://ark.mwbsys.com/mbst/latest>

<https://staging-www.malwarebytes.com/js/json/global_threat.json>

# Vulnerabilities in [Press.malwarebytes.com](https://press.malwarebytes.com/) and [Blog.malwarebytes.com](https://press.malwarebytes.com/)

The Malwarebytes company maintains two subdomains for publishing news and events, both developed using WordPress CMS. During my examination, I discovered certain vulnerabilities that, if left unpatched, could be exploited by skilled hackers. The vulnerabilities include SQL injection, XSS, and file inclusion, some of which are associated with installed plugins.

For instance, the plugin "responsive-lightbox" is utilized, and information about plugins and other data can potentially aid a hacker in exploiting these vulnerabilities. To obtain more details about these security issues, I performed a scan using the WPSCAN tool and saved the results in the "scanResult.txt" file. I urge you to carefully review this file.

Below, I've provided some potentially vulnerable links that warrant immediate attention:

<https://blog.malwarebytes.com/robots.txt>

<https://blog.malwarebytes.com/wp-admin/admin-ajax.php>

<https://press.malwarebytes.com/wp-json/>

<https://press.malwarebytes.com/wp-json/wp/v2/pages/5>

https://press.malwarebytes.com/wp-json/oembed/1.0/embed?url=press.malwarebytes.com

<https://press.malwarebytes.com/wp-includes/wlwmanifest.xml>

https://press.blog.malwarebytes.com/wp-content/plugins/responsive-lightbox/assets/infinitescroll/infinite-scroll.pkgd.min.js?ver=%27

https://press.malwarebytes.com/wp-content/plugins/responsive-lightbox/js/front.js?ver=2.3.3

https://blog.malwarebytes.com/wp-includes/js/wp-embed.min.js?ver=5.8.2

https://blog.malwarebytes.com/wp-includes/js/wp-emoji-release.min.js

https://blog.malwarebytes.com/wp-content/themes/mb-labs-theme/style.css?05-11-2021

https://blog.malwarebytes.com/wp-content/plugins/disqus-comment-system/README.txt

https://blog.malwarebytes.com/wp-content/plugins/jetpack/readme.txt

https://blog.malwarebytes.com/wp-content/plugins/responsive-lightbox/js/front.js?ver=2.3.3

https://blog.malwarebytes.com/wp-content/plugins/responsive-lightbox/readme.txt

https://blog.malwarebytes.com/wp-content/plugins/wordpress-seo/readme.txt