Directory Traversal Attack

Asadov Akpar

Abstract. The aim of the research is to investigate protocol bypass attacks. Web servers are in high demand worldwide. In recent years, as the use of web servers has increased, so has the number of hacking incidents and data theft from them. In this case, hacking is a method employed by malicious actors to steal data. Web servers are typically well-protected; however, there are still numerous methods of attacking them that persist to this day.

Keywords: Web server, web server security, protocol bypass.

In this article, we will delve into directory traversal attacks, also known as path traversal attacks[1]. This type of attack can be used to access confidential files on a web server and is typically feasible when the web server or web application lacks user input validation[7]. To serve a specific page, the web server provides files from the file system. These files can be located in the root directory of the website or in one of its subdirectories. In Linux systems, the directory /var/www/html/ is commonly used as the root web directory. When a web application displays a page, for example, http://example.com/file.html, it attempts to access /var/www/html/file.html. The HTTP link contains no path components apart from the file name because the root path to the front-end component of the site also serves as the web server's base directory. If a web application is vulnerable to directory traversal, a user can gain access to files beyond the root folder of the website using relative paths, thereby gaining access to confidential files such as SSH private keys or configuration files[5].

As one of the exploitation methods for this vulnerability, the "../" pattern can be cited, which was used in Microsoft Windows[2][3]. The existence of this vulnerability is due to the fact that each disk has a separate root directory, identified by a single letter. For example, "C:/," where C can be any disk, and there is no common root directory above it. This means that for most directory traversal vulnerabilities in Windows, attacks are limited to one method, where we specify the disk. While it's important to understand how to exploit directory traversal vulnerabilities, it's equally vital to be able to identify them. We should always check for vulnerabilities by hovering over all buttons, inspecting all links, exploring all accessible pages, and (if possible) examining the page's source code. Links can be a particularly valuable source of information, providing parameters or other data about the application.

For example, consider the following link:

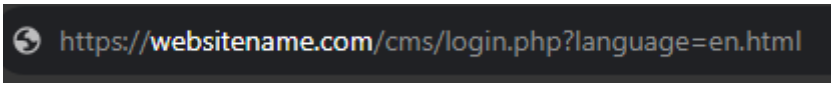

Figure 1 - Example of a website

Primarily, it is worth paying attention to `login.php`, which informs us that the web application uses PHP. We can utilize this to understand how the web application functions, which is valuable during the exploitation phase. Secondly, the URL contains a language parameter with an HTML page as its value. In such a situation, we should attempt to navigate directly to the file (https://websitename.com/cms/en.html). If we can successfully access it, we can confirm that `en.html` is a file on the server [6]. In other words, we can use this parameter to try inserting other file names into the search bar. It is essential to always meticulously examine parameters when they are used as values for files. Thirdly, the URL includes a directory named `cms`. This is important information indicating that the web application operates in a subdirectory of the root website.

Next, we will examine a more illustrative and practical example: we will run a website downloaded from

the internet on our local machine. After adding the host, we will be able to access it by following the following link:

Figure 2 - Local hosted website

And we will see our website:

Figure 3 - Website page

Let us consider Figure 3. It depicts a page after we opened it in the browser. The navigation panel displays a file named index.php, which suggests that the web application uses PHP. To gather more information about the page's structure, it is necessary to hover over all buttons and links, collecting information about parameters and various pages we will encounter when clicking different buttons. Scrolling down and hovering over all buttons and links, we will notice that most of them only reference the same page.

Figure 4 - Hovering the cursor over the button

Going even lower, we will find a link at the bottom of the page with the inscription "admin".

Figure 5 - Pointing the cursor at the "admin" link.

We know that the web application uses PHP and the "page" parameter, so let us assume that this parameter is used to display different pages. PHP uses $_GET [4] to manage variables via a GET request. When we click on the link, we get an error message stating that the page is currently under maintenance:

Figure 6 - Error message for the admin link

For us, this detail is important because it shows that the information is displayed on one page. In this case, we will make a few assumptions about how you can develop a web application that will behave in this way. For example, when we open www.websiteexample.com/meteor/admin .php in our browser, we notice the same message that was displayed on the page index.php after clicking the "admin" link:

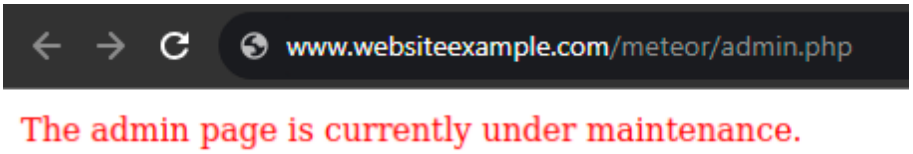

Figure 7- Admin page maintenance

This message means that the web application includes the content of this page via the page parameter and displays it via the "admin" link. Now we can try using "../" to crawl directories in a potentially vulnerable parameter. We will specify the relative path to /etc/passwd to check the page parameter for directory traversal.

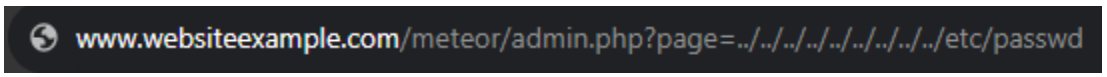

Figure 8 -The URL of our directory traversal attack

By inserting this link into our browser, we will get the following result:

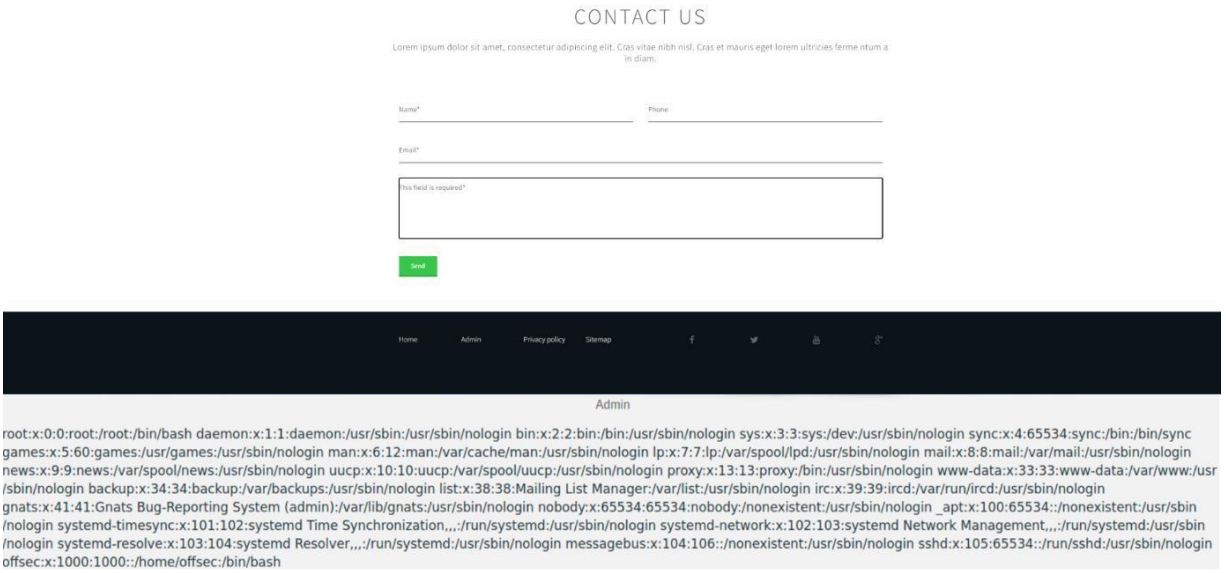

Figure 9 - The web application shows the contents of the Passwd file.

As can be seen in the figure above, the contents of the /etc/passwd directory file are on our screen. We successfully exploited a directory crawl vulnerability using a relative path.

Conclusion

Often, developers in web applications or web servers fail to validate user input, and in addition to this, applications do not employ filters to block suspicious activities. Administrators should maintain up-to-date software, including web server software and the underlying operating system, and apply all security patches. The practice of regularly updating software can significantly reduce security risks and decrease the likelihood of breaches.

References

[1] https://en.wikipedia.org/wiki/Directory_traversal_attack
[2] https://www.cvedetails.com/vulnerability-list/vendor_id-26/opdirt-1/Microsoft.html
[3] https://cve.mitre.org/cgi-bin/cvename.cgi?Name=CVE-2001-0333
[4] https://www.php.net/manual/en/reserved.variables.get.php
[5] https://security.snyk.io/vuln/npm:enserver:20170516
[6] https://docs.nginx.com/nginx/admin-guide/web-server/serving-static-content/
[7] https://brightsec.com/blog/directory-traversal/#testing-techniques