CHAMELEON

FOR OUR SMARTER WORLD

Local/Remote File Inclusion – Attack

Hamish Burnett – (222282244)

Contents

Executive Summary

The attack that will be tested in this operation, will be a File Inclusion attack. A File Inclusion attack is an attack that allows a malicious actor to execute files on a target webserver (OWASP Foundation, n.d.-b). File Inclusion attacks are split into two categories, being Local File Inclusion attacks, where the webserver executes and runs a file stored on the webserver, and Remote File Inclusion attacks, where the webserver executes and runs a file hosted on a remote server. File Inclusion attacks can be used to launch Code Execution, Denial of Service, and Sensitive Information Disclosure on the targeted website.

The tools used for this test, include Kali Linux, as the operating system, *dotdotpwn* as a Path Traversal fuzzer (Path Traversal attacks can be used to execute File Inclusion attacks), Burp Suite, to analyse and modify packets, and execute the attack, and FoxyProxy, to direct web traffic to Burp Suite.

It was found that the web application may be vulnerable to Path Traversal attacks, which can be used to execute Local File Inclusion attacks. As a result of the testing, it was found that Local/Remote File Inclusion attacks were currently unsuccessful against the Chameleon webserver. Although the web application has been deemed resistant to File Inclusion attacks, further research needs to be completed, and further testing needs to be carried out, when more functionality is added to the application.

Introduction

A File Inclusion attack, is an attack that allows a malicious actor to execute files on a target webserver (OWASP Foundation, n.d.-b). File Inclusion attacks can be separated into two different sub-categories; being Local File Inclusion, and Remote File Inclusion. In a Local File Inclusion attack, the attacker executes and runs files stored on the server. These may include files containing sensitive information such as system information files, and malware, if the attacker has succeeded in uploading malware to the system. In contrast, a Remote File Inclusion attack occurs when a web server executes files stored outside of the server, on a remote server. This may occur if an attacker hosts their own webserver, uploads malware to it, and executes the malware on the targeted webserver.

File Inclusion attacks can cause Code Execution, Denial of Service, and Sensitive Information Disclosure on the targeted website.

The operation of this security test will determine whether the Chameleon website is vulnerable to File Inclusion attacks, and if so, present recommendations to prevent File Inclusion attacks in the future.

Tools Used

Three main tools were used, which are listed below:

- Kali Linux
- *dotdotpwn* (https://github.com/wireghoul/dotdotpwn)
- Burp Suite – Community Edition (Comes preinstalled on Kali Linux)
- FoxyProxy

The attacks were performed using Kali Linux as the Operating System (OS), through a virtual machine. *dotdotpwn* was used to perform fuzz testing of the website, to determine if there were any Path Traversal vulnerabilities which were present. A fuzz test is performed by an automated software application, that sends large amounts of requests to a website, with each request being slightly different, to determine whether there are any vulnerabilities present in the application (OWASP Foundation, n.d.-a). A Path Traversal attack allows an attacker to read files on the server. A Path Traversal attack can be used to launch a File Inclusion attack. The difference between a Path Traversal, and a File Inclusion attack, is that the Path Traversal only reads files on the server, while the File Inclusion attacks executes and runs files on the server. Burp Suite is used to perform penetration testing of websites, through manual operation. FoxyProxy is used to intercept and direct traffic to Burp Suite.

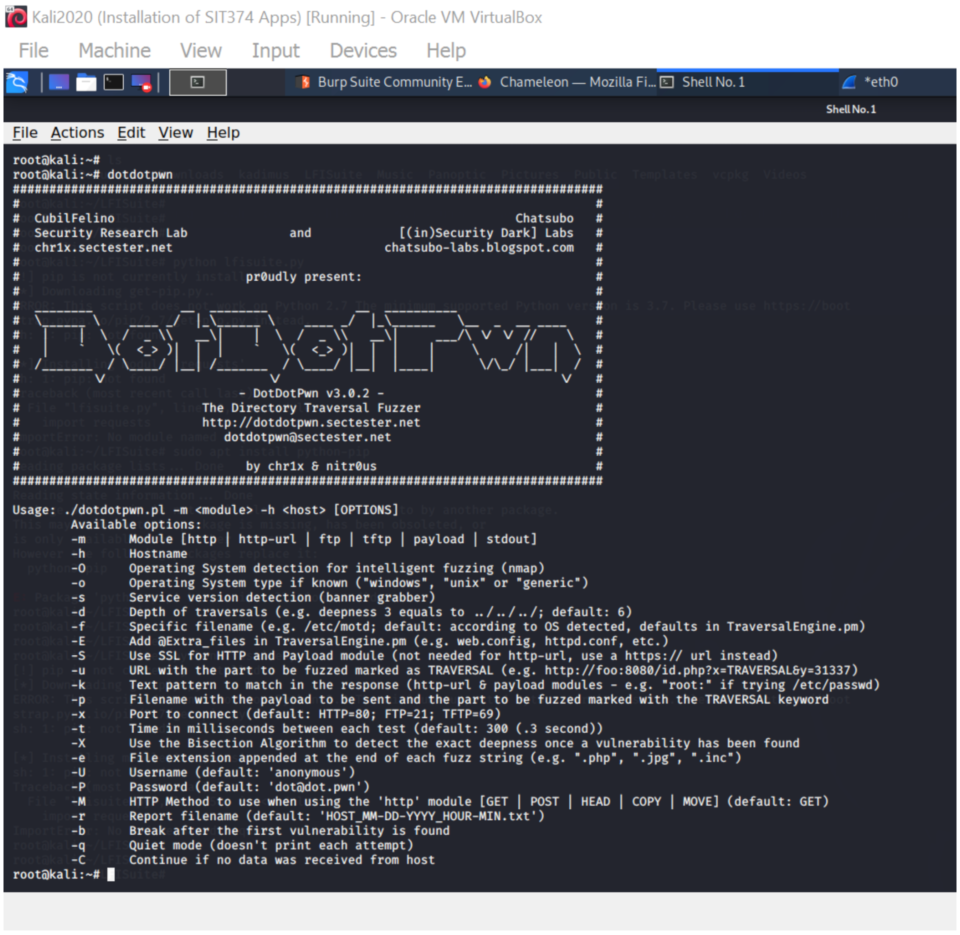
Scope of Testing

The scope of this penetration test, was limited to the Chameleon website. Most testing was carried out on the home page, but some testing was also carried out on the other pages.

Methodology

To execute a Local File Inclusion attack, a Path Traversal attack was used. (Path Traversal attacks read files on the server, while Local File Inclusion attacks execute files on the server). If the Path Traversal attack was successful, then it would indicate that a Local File Inclusion vulnerability is present, with a high level of certainty.

A Kali Linux Virtual Machine (VM) was first set up. Burp Suite came pre-installed on Kali Linux, and as such, did not require any installation. *dotdotpwn* was installed on the VM from GitHub (https://github.com/wireghoul/dotdotpwn). The command *sudo apt-get install dotdotpwn* was used to install *dotdotpwn*. *dotdotpwn* was then run, using the command *dotdotpwn*. The result of this command, is shown below, in Figure 1:


Kali2020 (Installation of SIT374 Apps) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Burp Suite Community E...
Chameleon — Mozilla Fi...
Shell No. 1
*eth0
Shell No. 1
File Actions Edit View Help

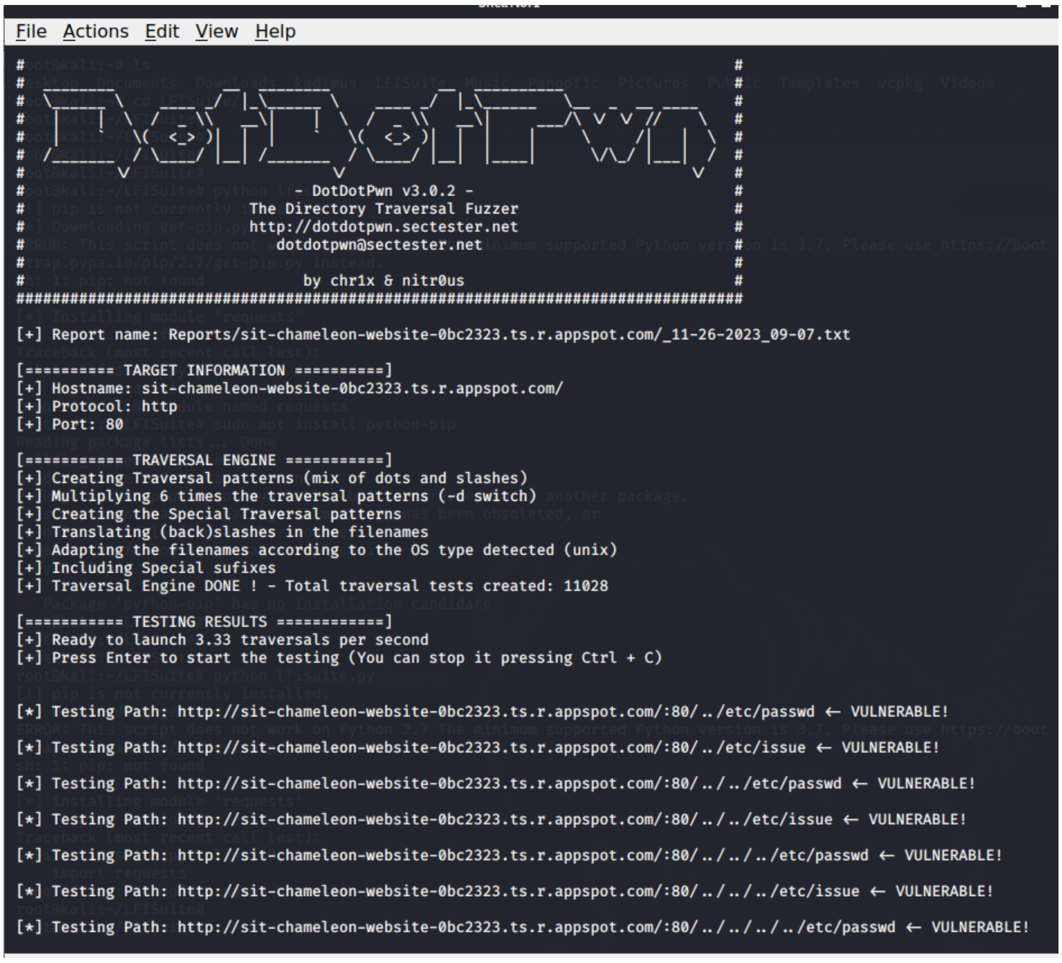
```
root@kali:~#
root@kali:~# dotdotpwn
#################################################################################
#                                                                               #
#  CubilFelino                                                       Chatsubo   #
#  Security Research Lab              and               [(in)Security Dark] Labs #
#  chr1x.sectester.net                             chatsubo-labs.blogspot.com   #
#                                                                               #
#                               pr0udly present:                                #
#                                                                               #
#  ________            __  ________            __  __________                   #
#  \______ \    ____ _/  |_\______ \    ____ _/  |_\______   \__  _  __ ____    #
#   |    |  \  /  _ \\   __\|    |  \  /  _ \\   __\|     ___/\ \/ \/ //    \   #
#   |    `   \(  <_> )|  |  |    `   \(  <_> )|  |  |    |     \     /|   |  \  #
#  /_______  / \____/ |__| /_______  / \____/ |__|  |____|      \/\_/ |___|  /  #
#          \/                      \/                                       \/  #
#                              - DotDotPwn v3.0.2 -                             #
#                         The Directory Traversal Fuzzer                        #
#                         http://dotdotpwn.sectester.net                        #
#                            dotdotpwn@sectester.net                            #
#                                                                               #
#                               by chr1x & nitr0us                              #
#################################################################################

Usage: ./dotdotpwn.pl -m <module> -h <host> [OPTIONS]
        Available options:
        -m      Module [http | http-url | ftp | tftp | payload | stdout]
        -h      Hostname
        -O      Operating System detection for intelligent fuzzing (nmap)
        -o      Operating System type if known ("windows", "unix" or "generic")
        -s      Service version detection (banner grabber)
        -d      Depth of traversals (e.g. deepness 3 equals to ../../../; default: 6)
        -f      Specific filename (e.g. /etc/motd; default: according to OS detected, defaults in TraversalEngine.pm)
        -E      Add @Extra_files in TraversalEngine.pm (e.g. web.config, httpd.conf, etc.)
        -S      Use SSL for HTTP and Payload module (not needed for http-url, use a https:// url instead)
        -u      URL with the part to be fuzzed marked as TRAVERSAL (e.g. http://foo:8080/id.php?x=TRAVERSAL&y=31337)
        -k      Text pattern to match in the response (http-url & payload modules - e.g. "root:" if trying /etc/passwd)
        -p      Filename with the payload to be sent and the part to be fuzzed marked with the TRAVERSAL keyword
        -x      Port to connect (default: HTTP=80; FTP=21; TFTP=69)
        -t      Time in milliseconds between each test (default: 300 (.3 second))
        -X      Use the Bisection Algorithm to detect the exact deepness once a vulnerability has been found
        -e      File extension appended at the end of each fuzz string (e.g. ".php", ".jpg", ".inc")
        -U      Username (default: 'anonymous')
        -P      Password (default: 'dot@dot.pwn')
        -M      HTTP Method to use when using the 'http' module [GET | POST | HEAD | COPY | MOVE] (default: GET)
        -r      Report filename (default: 'HOST_MM-DD-YYYY_HOUR-MIN.txt')
        -b      Break after the first vulnerability is found
        -q      Quiet mode (doesn't print each attempt)
        -C      Continue if no data was received from host
root@kali:~#
```

Figure 1: Screenshot showing dotdotpwn installed, and operating.

The *dotdotpwn* application was then run, using the Chameleon website URL. The command used, was: *sudo dotdotpwn -m http -h "sit-chameleon-website-0bc2323.ts.r.appspot.com/"* . The switches of the command are shown below:

> -m http: Indicates to use the http module. Other modules available include http-url, ftp, tftp, payload, and stdout
>
> -h "URL": Indicates the host URL to perform the fuzz test on.

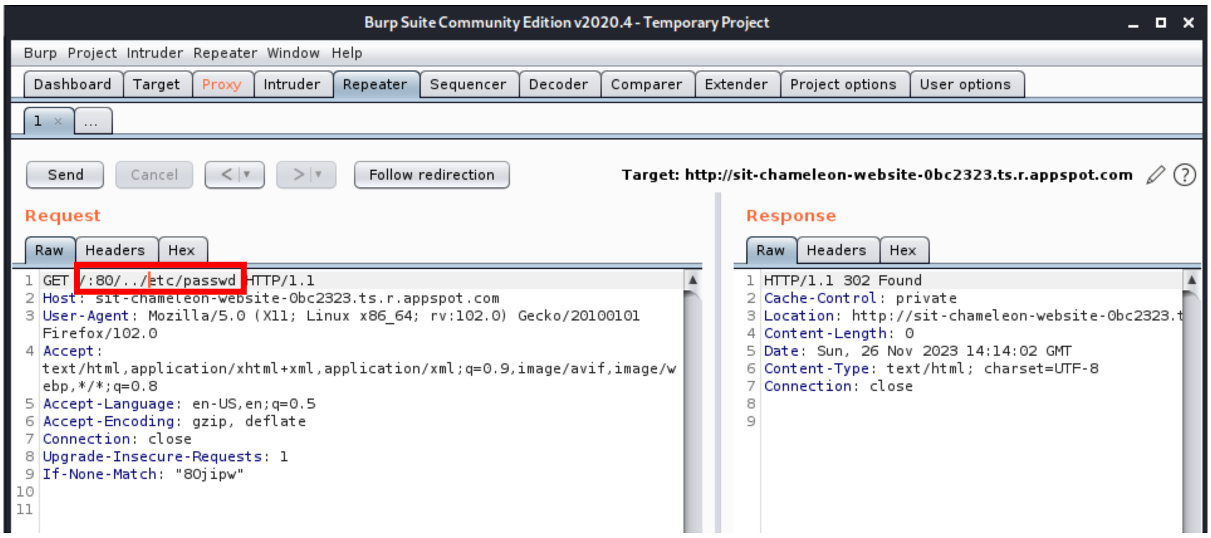
Initial results of performing this command, are shown below, in Figure 2. It can be seen that a number of the URLs tested, are vulnerable to Path Traversal attacks.



```
#                                                                            #
#                                                                            #
#                                                                            #
#                                                                            #
#                                                                            #
#                                                                            #
#                                                                            #
#                         - DotDotPwn v3.0.2 -                               #
#                      The Directory Traversal Fuzzer                        #
#                      http://dotdotpwn.sectester.net                        #
#                        dotdotpwn@sectester.net                             #
#                                                                            #
#                            by chr1x & nitr0us                              #
##############################################################################

[+] Report name: Reports/sit-chameleon-website-0bc2323.ts.r.appspot.com/_11-26-2023_09-07.txt

[========== TARGET INFORMATION ==========]
[+] Hostname: sit-chameleon-website-0bc2323.ts.r.appspot.com/
[+] Protocol: http
[+] Port: 80

[=========== TRAVERSAL ENGINE ===========]
[+] Creating Traversal patterns (mix of dots and slashes)
[+] Multiplying 6 times the traversal patterns (-d switch)
[+] Creating the Special Traversal patterns
[+] Translating (back)slashes in the filenames
[+] Adapting the filenames according to the OS type detected (unix)
[+] Including Special sufixes
[+] Traversal Engine DONE ! - Total traversal tests created: 11028

[=========== TESTING RESULTS ============]
[+] Ready to launch 3.33 traversals per second
[+] Press Enter to start the testing (You can stop it pressing Ctrl + C)

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../etc/passwd <- VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../etc/issue <- VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../etc/passwd <- VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../etc/issue <- VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../etc/passwd <- VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../etc/issue <- VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../etc/passwd <- VULNERABLE!
```

Figure 2: Screenshot showing dotdotpwn running, with several website URLs being identified as vulnerable.

From there, several of the vulnerable URLs were tested in Burp Suite, to confirm that they were vulnerable. FoxyProxy was used to direct traffic to Burp Suite. The first URL that was identified as vulnerable, was tested in Burp Suite. In Burp Suite, the traffic was modified, to perform the request of ":80/../etc/passwd", shown in the red box, in Figure 3.


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel Follow redirection
Target: http://sit-chameleon-website-0bc2323.ts.r.appspot.com
Request
Raw Headers Hex
Response
Raw Headers Hex

Request:

```
GET /:80/../etc/passwd HTTP/1.1
Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
```

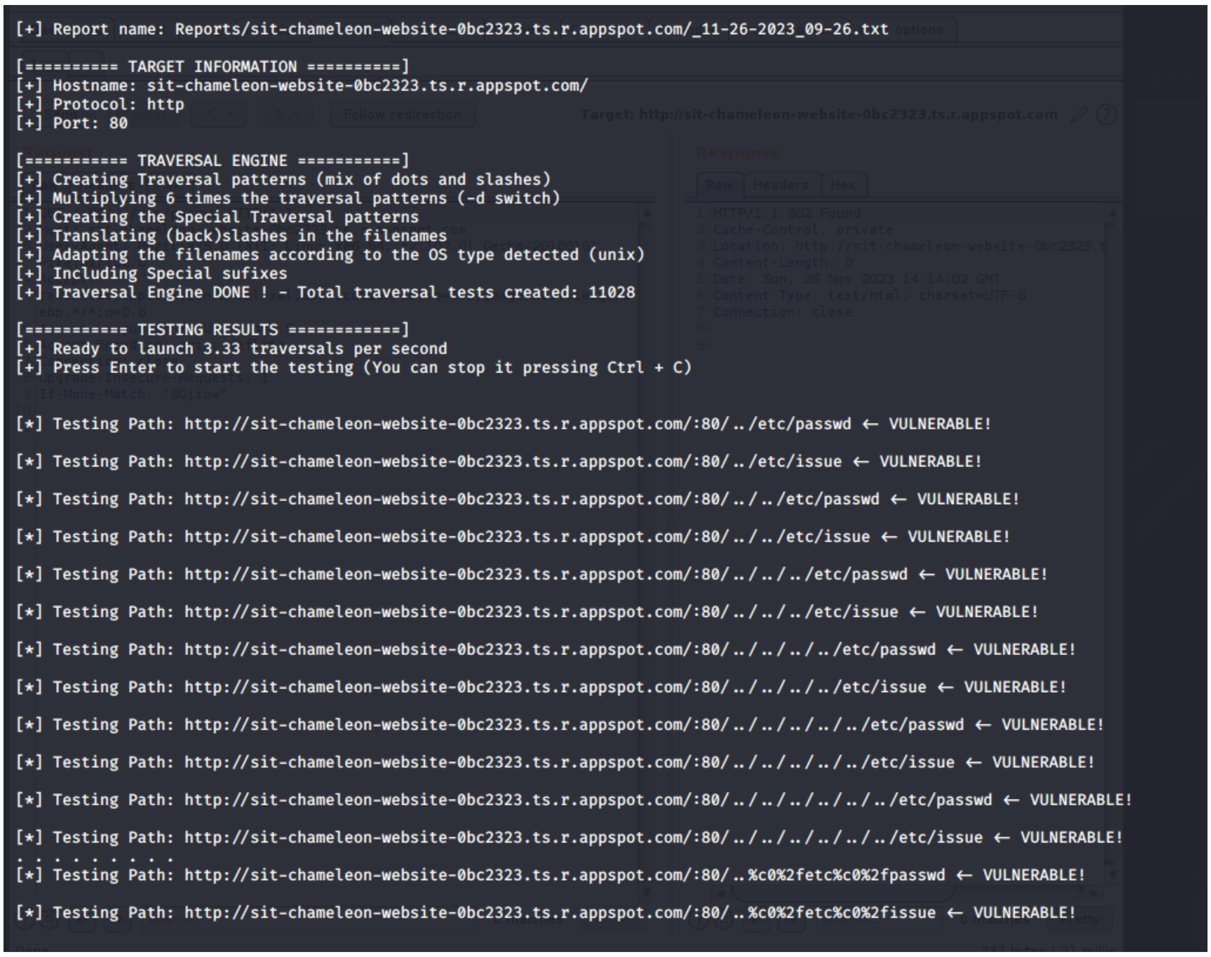
Response:

```
HTTP/1.1 302 Found
Cache-Control: private
Location: http://sit-chameleon-website-0bc2323.t
Content-Length: 0
Date: Sun, 26 Nov 2023 14:14:02 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
```

Figure 3: Screenshot showing Burp Suite performing testing of the vulnerable URL. The red box shows the vulnerable URL.

Results

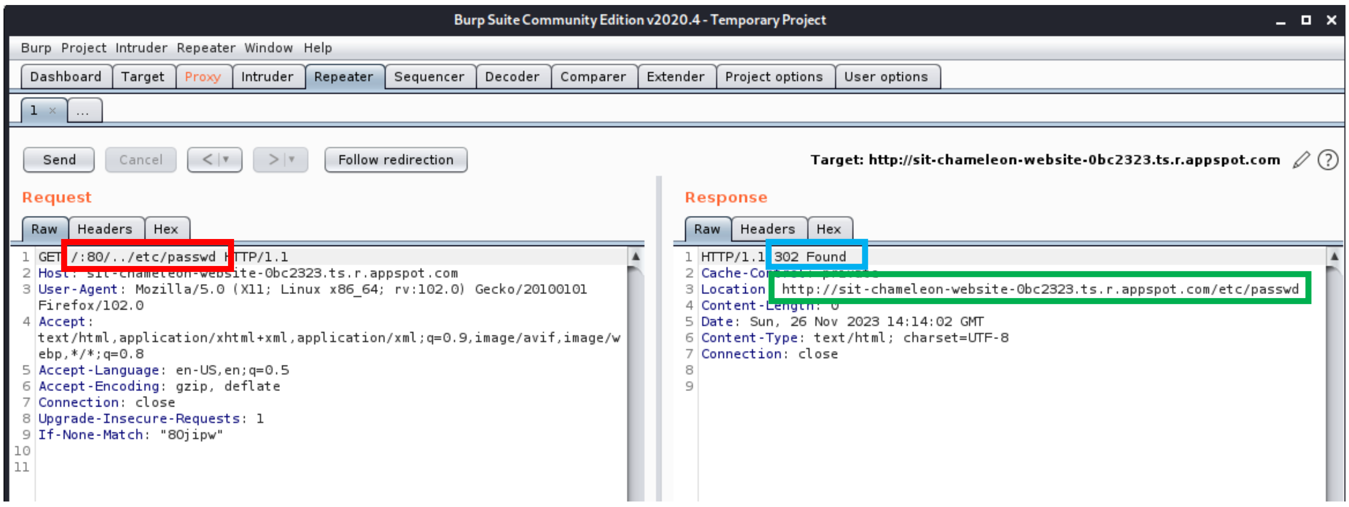
From an initial analysis, the *dotdotpwn* application identified a high number of URLs that were deemed to be vulnerable. The command was modified to include the -q switch, which resulted in the scan displaying only vulnerable URLs. The updated command was: *sudo dotdotpwn -m http -h "sit-chameleon-website-0bc2323.ts.r.appspot.com/" -q* . The results of this scan are shown below, in Figure 4.



```
[+] Report name: Reports/sit-chameleon-website-0bc2323.ts.r.appspot.com/_11-26-2023_09-26.txt

[========== TARGET INFORMATION ==========]
[+] Hostname: sit-chameleon-website-0bc2323.ts.r.appspot.com/
[+] Protocol: http
[+] Port: 80

[=========== TRAVERSAL ENGINE ===========]
[+] Creating Traversal patterns (mix of dots and slashes)
[+] Multiplying 6 times the traversal patterns (-d switch)
[+] Creating the Special Traversal patterns
[+] Translating (back)slashes in the filenames
[+] Adapting the filenames according to the OS type detected (unix)
[+] Including Special sufixes
[+] Traversal Engine DONE ! - Total traversal tests created: 11028

[=========== TESTING RESULTS ============]
[+] Ready to launch 3.33 traversals per second
[+] Press Enter to start the testing (You can stop it pressing Ctrl + C)

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../etc/passwd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../etc/issue ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../etc/passwd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../etc/issue ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../etc/passwd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../etc/issue ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../etc/passwd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../etc/issue ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../../etc/passwd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../../etc/issue ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../../../etc/passwd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/../../../../../../etc/issue ← VULNERABLE!
. . . . . . . . .
[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/..%c0%2fetc%c0%2fpasswd ← VULNERABLE!

[*] Testing Path: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/:80/..%c0%2fetc%c0%2fissue ← VULNERABLE!
```

Figure 4: Screenshot of the dotdotpwn application identifying that certain URLs are vulnerable.

Several of these URLs were then tested using Burp Suite. A normal request (consisting of the Chameleon website URL) was sent to Burp Suite via FoxyProxy. The request was forwarded to the Repeater, to be able to perform the tests in a fast manner. The URL was modified, to include the *:80/../etc/passwd* command (shown in a red box in Figure 5). The response to this request was *302 Found* (shown in a blue box). The *302 Found* response, indicates that the URL has temporarily changed to to the URL shown in the green box (mdn web docs_, 2023). The /etc/passwd component of the URL, is meant to retrieve the *passwd* file from the server, which contains information about the user accounts on the file system (nishant0073, 2021). However, the file cannot be seen in the response, indicating that this attack configuration was not successful.


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel Follow redirection
Target: http://sit-chameleon-website-0bc2323.ts.r.appspot.com
Request
Raw Headers Hex
Response
Raw Headers Hex

```
GET /:80/../etc/passwd HTTP/1.1
Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
```

```
HTTP/1.1 302 Found
Cache-Control: private
Location: http://sit-chameleon-website-0bc2323.ts.r.appspot.com/etc/passwd
Content-Length: 0
Date: Sun, 26 Nov 2023 14:14:02 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
```

Figure 5: Screenshot showing testing of vulnerable URLs in Burp Suite. The red box indicates the modified URL, the blue box indicates the response code, and the green box indicates the redirected URL.

However, when this webpage was loaded in the web browser, the website displayed the menu bar at the top of the page, but did not show any content, as shown below, in Figure 6.

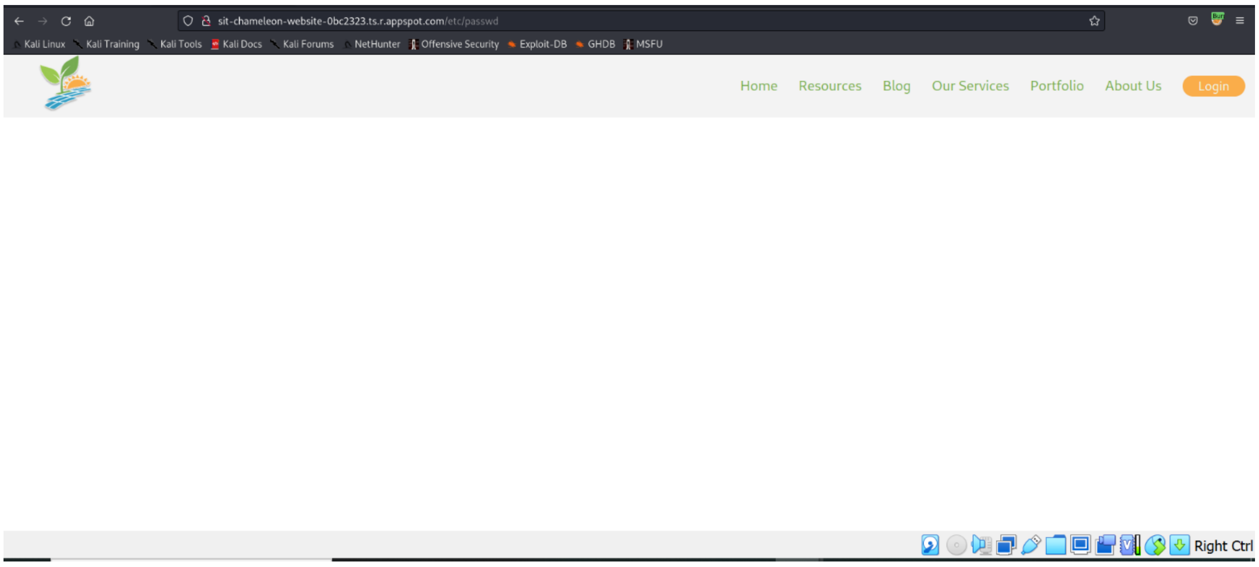


Figure 6: Screenshot of the Chameleon website, with the malicious URL. Note that the menu bar is shown, but the content of the website is not shown.

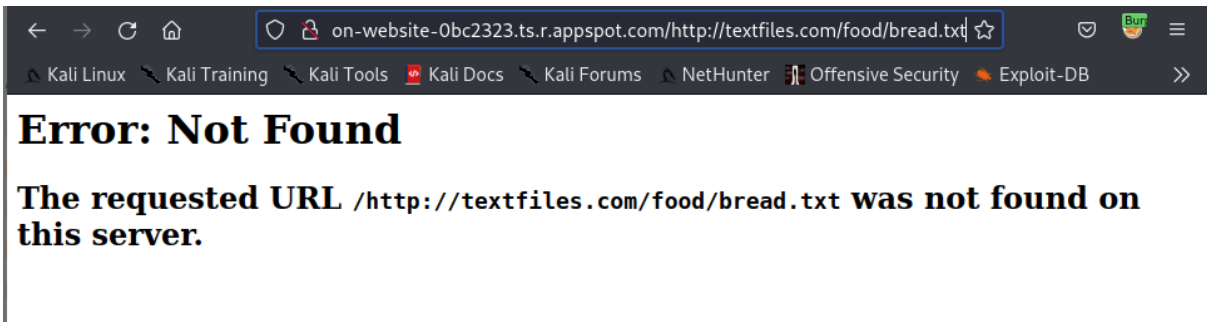
This process was repeated a number of times, where a URL that was flagged as vulnerable, was tested in Burp Suite. The result was the same for each of the selected URLs that were tested. The URLs were also tested in the browser, but this resulted in the same blank screen, as shown above, in Figure 6. Occasionally, the response code was *304 Not Modified*, which indicates that the response had not changed, and that the cached version can be used.

Therefore, the Path Traversal attack was partially successful. It was unsuccessful, due to being unable to read the files, but it partially worked, as the website loaded a blank page, rather than the normal content, or an error. As a result, it may be possible that the correct combination of a URL, may successfully trigger a Path Traversal Attack.

As the Path Traversal attack was not successful, it was found that the Local File Inclusion attack was unsuccessful as well.

Remote File Inclusion attacks were then tested. Remote File Inclusion is when a website executes content that is not located on the web server. The content could be from another webserver, or a webserver created by a malicious actor.

To test for Remote File Inclusion, I used the web browser, to enter the chameleon website, followed by the link to another website. The second website that I attached in the URL, was a website that contained text files on a range of topics. The URL that I used, was: https://sit-chameleon-website-0bc2323.ts.r.appspot.com/http://textfiles.com/food/bread.txt . This resulted in the Chameleon webserver showing an error, which stated that the second URL could not be found. If this attack was successful, the Chameleon website would have displayed the contents of the bread.txt file. This is shown below, in Figure



Error: Not Found

The requested URL /http://textfiles.com/food/bread.txt was not found on this server.

Figure 7: Screenshot of the result from the Remote File Inclusion attack. The attack resulted in the page returning an error.

This attack was repeated a number of times, each with a different combination of letters. However, this resulted in the same error, and as a result, it was found that the Chameleon webserver is not vulnerable to Remote File Inclusion attacks.

There are several conclusions that may be drawn from the failure of the Local File Inclusion attack. The first, is that there may be systems in place to prevent the unauthorised files being displayed. The second, is that the specific files that were being searched (i.e. etc/passwd, and etc/issue), were either not located on the webserver, or the webserver was not running Linux. The third, is that Local File Inclusion may be possible, due to the lack of errors that were generated from malicious URLs, and the blank screen, which only contained the menu bar, and no content.

At this stage, it is believed that the Chameleon website is not vulnerable to Remote File Inclusion attacks.

Recommendations

It is recommended that further testing be conducted on whether Local File Inclusion is possible or not, on the Chameleon website. This is suggested, as the Path Traversal attacks were deemed to be inconclusive, as to whether Path Traversal vulnerabilities existed. As a result, it can be said that the Local File Inclusion attack was not successful. However, due to the lack of error messages, and the website displaying a blank page, it can be considered that certain input, may result in a Local File Inclusion vulnerability.

Through research, it has been found that Local File Inclusion vulnerabilities are more common on websites that use PHP (OWASP Foundation, n.d.-b). The Chameleon website uses JavaScript, rather than PHP. It was also found that File Inclusion vulnerabilities are more likely to occur when user input is received from the user, and stored in the URL (i.e. The ?*language=english* section), in the URL: https://examplewebsite.com/input?language=english

It is recommended that testing of File Inclusion vulnerabilities occurs regularly, especially if/when functionality is added to enable the user to interact with forms.

Mitigations to prevent File Inclusion attacks:

- Perform whitelisting of files that are allowed to be requested by the user (OWASP Foundation, n.d.-b).
- Sanitise all user input, to reduce the likelihood that the user input contains malicious commands (SiteLock, 2021).

Conclusion

Through the security test, it was found that the Chameleon website may be vulnerable to Path Traversal attacks, which enable an attacker to read files located on the webserver. The results were inconclusive, due to the website not displaying files that were requested by the user. However, the website returned a blank page when malicious input was entered into the URL. It could be conceivable that it is possible to launch a Path Traversal attack against the Chameleon website.

File Inclusion attacks are used to execute files on the target webserver. They can be used to run malware, which can lead to code execution, Denial of Service, and Sensitive Information Disclosure (OWASP Foundation, n.d.-b).

Local File Inclusion attacks occur when an attacker is able to execute and run files stored on the webserver. Local File Inclusion attacks are built on the Path Traversal vulnerability, listed above. Therefore, the website may be vulnerable to Local File Inclusion attacks. From the testing that was performed throughout this attack, it was found that the webserver is currently safe from Local File Inclusion attacks. However, if the application is designed to collect more data from users, then the website may become vulnerable to this attack.

Remote File Inclusion attacks occur when an attacker is able to run files on the webserver that are not located on the target webserver. These files may be hosted on the attacker’s webserver. The current testing indicates that the Chameleon website is not vulnerable to this attack. However, if the application collects more data from users, then the website may become vulnerable to this attack.

In summary, it was found that the web application is not currently vulnerable to Path Traversal, Local File Inclusion, and Remote File Inclusion vulnerabilities.

References

mdn web docs_ (2023) *302 Found*: Mozilla, accessed 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/302

nishant0073 (2021) *Understanding the /etc/passwd File,* accessed 2023. https://www.geeksforgeeks.org/understanding-the-etc-passwd-file/

OWASP Foundation (n.d.-a) *Fuzzing*, accessed 2023. https://owasp.org/www-community/Fuzzing

OWASP Foundation (n.d.-b) *Testing for File Inclusion*, accessed 2023. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_File_Inclusion

SiteLock (2021) *Remote File Inclusion: What It Is, How It Works, and How To Prevent It*, accessed 2023. https://www.sitelock.com/blog/remote-file-inclusion-what-it-is-how-it-works-and-how-to-prevent-it/