

CRLFsuite - CRLF injection scanner

1. What is CRLF?

The term CRLF refers to the combination of two special characters, the Carriage Return (CR) and the Line Feed (LF), that are used to mark the end of a line in text files.

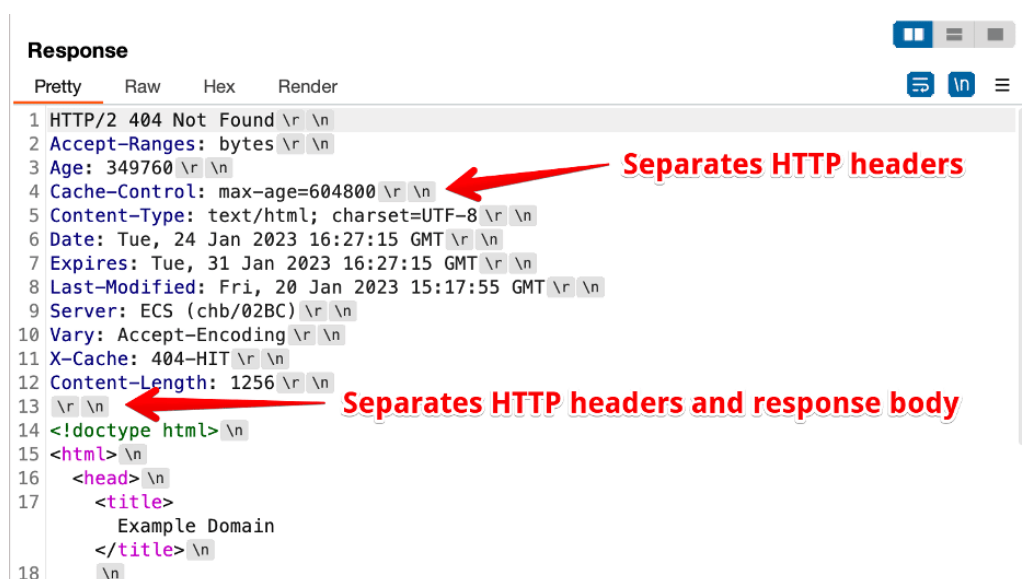
Carriage return is denoted by ASCII 13 (r), indicating the end of a line.

We place 'r' to represent the end of a line.

Line feed is denoted by ASCII 10 (n), indicating the place where a new line begins. Whenever we have to start a new line, we place 'n'.

Anything written after 'n' will come in a new line. The 'n' and 'r' (CR and LF) are called CRLF characters.

In web applications, these characters are used to separate different parts of an HTTP request or response, such as the headers, the body, and the status line.



```

Response
Pretty Raw Hex Render
1 HTTP/2 404 Not Found \r \n
2 Accept-Ranges: bytes \r \n
3 Age: 349760 \r \n
4 Cache-Control: max-age=604800 \r \n
5 Content-Type: text/html; charset=UTF-8 \r \n
6 Date: Tue, 24 Jan 2023 16:27:15 GMT \r \n
7 Expires: Tue, 31 Jan 2023 16:27:15 GMT \r \n
8 Last-Modified: Fri, 20 Jan 2023 15:17:55 GMT \r \n
9 Server: ECS (chb/02BC) \r \n
10 Vary: Accept-Encoding \r \n
11 X-Cache: 404-HIT \r \n
12 Content-Length: 1256 \r \n
13 \r \n
14 <!doctype html> \n
15 <html> \n
16 <head> \n
17 <title> \n
   Example Domain \n
18 </title> \n
   \n

```

Separates HTTP headers

Separates HTTP headers and response body

2. What is CRLF Injection?

CRLF injection is carried out by an attacker by just simply inserting the carriage return line feed in the user input area to deceive the server or a web application, thus making them to think that an object is terminated and another new object has been started. CRLF characters are not intended to be malicious because they are originally used to separate the HTTP headers and actual web content. However, they can still be used as payloads for malicious attacks.

Note: CRLF injection is rated as P3 severity in bug crowd. CRLF injection can further be escalated from information disclosure to Remote Code Execution.

3. Methods to find CRLF injection

To find CRLF vulnerabilities, you can use various tools such as web application scanners, penetration testing tools, and manual testing methods.

Web application scanners can automatically scan your web application for CRLF vulnerabilities, while penetration testing tools can be used to manually test for the presence of these vulnerabilities.

Manual testing methods include sending HTTP requests with special characters in the headers and observing the web application's response.

Example:



This is an example of HTTP Response Splitting using a manual testing method here an attacker injected malicious code on request for set cookies and then injected code was set a cookie on the response.

www.target.com/%0d%0aSet-Cookie:CRLFInjection=MaliciousCookieSet

Note: Here "%0d" represents the URL-encoded value for the carriage return character (CR), and "%0a" represents the URL-encoded value for the line feed character (LF).

4. Impact of CRLF Vulnerability

The impacts of CRLF injection varies and the risk depends upon the type of scenarios. CRLF Injection enables an attacker to deactivate and bypass certain security restrictions like XSS filters and Same Origin Policy (SOP) in the victim's browsers, making them susceptible to further malicious attacks as below:

- Malicious Script Injection
- Phishing Attacks
- HTTP Header Injection
- HTTP Response Splitting
- Client web browser cache poisoning
- Client-side session hijacking

CRLFsuite



1. Description

CRLFsuite is a Python-based tool specifically designed for CRLF injection detection and exploitation. With its efficient implementation, it offers fast and powerful capabilities in identifying and mitigating CRLF injection vulnerabilities. By focusing on this particular type of security flaw, CRLFsuite provides targeted functionality to detect and exploit CRLF injection issues. Its specialized nature allows security professionals to efficiently assess web applications for potential CRLF injection risks, enabling them to take appropriate remedial actions.

This tool automatically performs identification and exploitation of CRLF injection we only need to specify the URL and other options so it reduces our time and is very helpful for penetration testers.

2. Features and uses of CRLFsuite

CRLFsuite is a powerful tool that offers several useful features. It supports scanning a single URL or multiple URLs to detect CRLF injection vulnerabilities. It also has the ability to read input from the standard input (stdin). The tool includes a feature to detect Web Application Firewalls (WAFs).

CRLFsuite has a unique feature called "CRLF Injection to XSS Chaining" that enables chaining CRLF injection with Cross-Site Scripting (XSS) attacks.

The tool supports both GET and POST methods for scanning, providing flexibility in testing different types of requests. It also includes concurrency capabilities, allowing for efficient scanning of multiple URLs simultaneously.

It has a robust payload generator that can generate WAF evasion payloads for testing and exploiting CRLF injection issues.

One of the advantages of CRLFsuite is its fast and efficient scanning process, which minimizes false-positive results. To enhance its capabilities, it incorporates WAF detection methodologies and user agent lists from tools like xssstrike and paramspider. Additionally, it leverages WAF signatures from tools such as XSSStrike and wafwoof.

3. Installation

There are two methods to install CRLFsuite on your machine

Method 1: Using Command

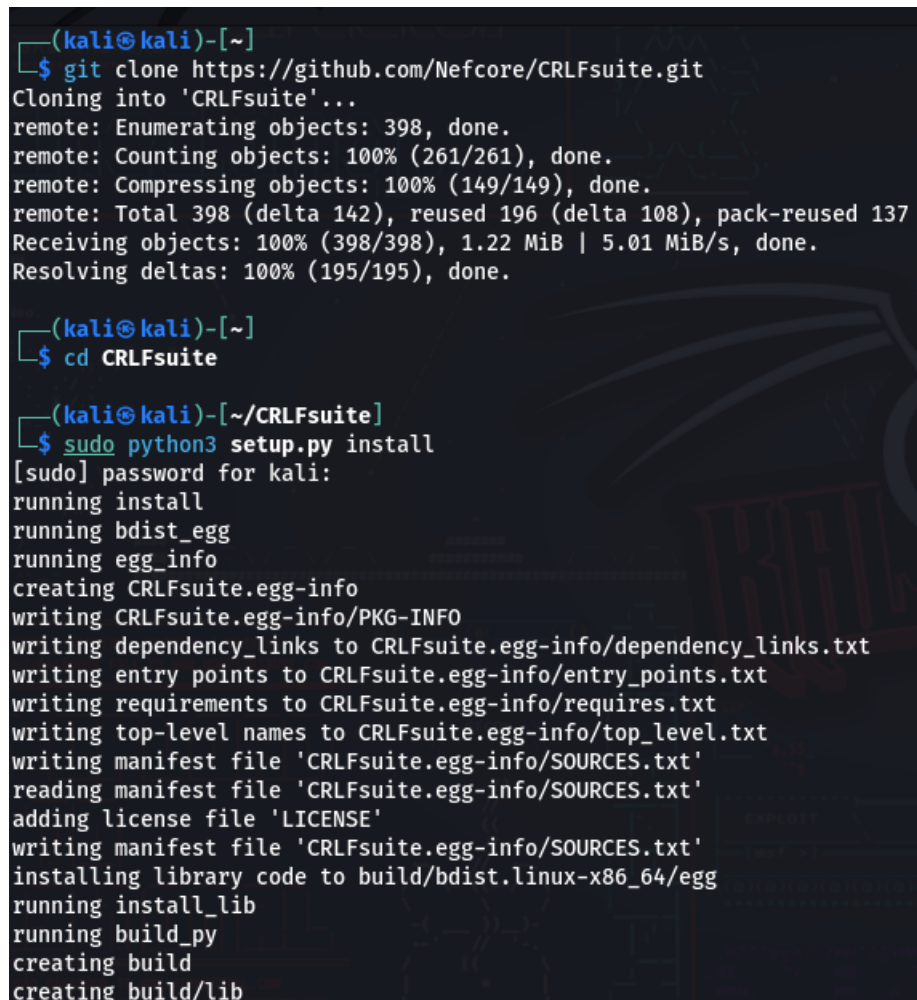
Step 1: Install using `$ pip3 install crlfsuite` command

```
(kali@kali)-[~]
$ pip3 install crlfsuite
Defaulting to user installation because normal site-packages is not writeable
Collecting crlfsuite
  Downloading CRLFsuite-2.5.2.tar.gz (19 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: colorama in /usr/lib/python3/dist-packages (from crlfsuite) (0.4.6)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from crlfsuite) (2.28.1)
Building wheels for collected packages: crlfsuite
  Building wheel for crlfsuite (setup.py) ... done
  Created wheel for crlfsuite: filename=CRLFsuite-2.5.2-py3-none-any.whl size=23835 sha256=223fe9868bc6ec542cf5efd4e05c6e4f23bc2123aaf31d3d9248cd91dea5326d
  Stored in directory: /home/kali/.cache/pip/wheels/93/ae/84/876087551642485f384676621e106e55974ecd1f0e3045e352
Successfully built crlfsuite
Installing collected packages: crlfsuite
Successfully installed crlfsuite-2.5.2
```

Method 2: Download the repository using git clone command

Step 1: Now run the following command to install CRLFsuite

```
$ git clone https://github.com/Nefcore/CRLFsuite.git
$ cd CRLFsuite
$ sudo python3 setup.py install
```

A terminal window with a dark background and light-colored text. The prompt is (kali㉿kali)-[~]. The user enters \$ git clone https://github.com/Nefcore/CRLFsuite.git. The output shows the cloning process: Cloning into 'CRLFsuite'..., remote: Enumerating objects: 398, done., remote: Counting objects: 100% (261/261), done., remote: Compressing objects: 100% (149/149), done., remote: Total 398 (delta 142), reused 196 (delta 108), pack-reused 137, Receiving objects: 100% (398/398), 1.22 MiB | 5.01 MiB/s, done., Resolving deltas: 100% (195/195), done. The user then enters \$ cd CRLFsuite. The prompt changes to (kali㉿kali)-[~/CRLFsuite]. The user enters \$ sudo python3 setup.py install. The output shows the installation process: [sudo] password for kali:, running install, running bdist_egg, running egg_info, creating CRLFsuite.egg-info, writing CRLFsuite.egg-info/PKG-INFO, writing dependency_links to CRLFsuite.egg-info/dependency_links.txt, writing entry points to CRLFsuite.egg-info/entry_points.txt, writing requirements to CRLFsuite.egg-info/requirements.txt, writing top-level names to CRLFsuite.egg-info/top_level.txt, writing manifest file 'CRLFsuite.egg-info/SOURCES.txt', reading manifest file 'CRLFsuite.egg-info/SOURCES.txt', adding license file 'LICENSE', writing manifest file 'CRLFsuite.egg-info/SOURCES.txt', installing library code to build/bdist.linux-x86_64/egg, running install_lib, running build_py, creating build, creating build/lib.

```
(kali㉿kali)-[~]
$ git clone https://github.com/Nefcore/CRLFsuite.git
Cloning into 'CRLFsuite'...
remote: Enumerating objects: 398, done.
remote: Counting objects: 100% (261/261), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 398 (delta 142), reused 196 (delta 108), pack-reused 137
Receiving objects: 100% (398/398), 1.22 MiB | 5.01 MiB/s, done.
Resolving deltas: 100% (195/195), done.

(kali㉿kali)-[~]
$ cd CRLFsuite

(kali㉿kali)-[~/CRLFsuite]
$ sudo python3 setup.py install
[sudo] password for kali:
running install
running bdist_egg
running egg_info
creating CRLFsuite.egg-info
writing CRLFsuite.egg-info/PKG-INFO
writing dependency_links to CRLFsuite.egg-info/dependency_links.txt
writing entry points to CRLFsuite.egg-info/entry_points.txt
writing requirements to CRLFsuite.egg-info/requirements.txt
writing top-level names to CRLFsuite.egg-info/top_level.txt
writing manifest file 'CRLFsuite.egg-info/SOURCES.txt'
reading manifest file 'CRLFsuite.egg-info/SOURCES.txt'
adding license file 'LICENSE'
writing manifest file 'CRLFsuite.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build
creating build/lib
```

Step 2: After Install check the tool work correctly by running the following command `$ crlfsuite`

4. How CRLFsuite works?

Takes target(s)

--> Generates payloads and injects them in the URL(s)

--> Sends HTTP request to all the URL(s)

--> Checks if the payload reflects in the response headers or text.

```
(kali㉿kali)-[~]
$ curlfsuite -t http://testphp.vulnweb.com/

      H
    ---
   [X]
  [X]
 [X]
[X]
[X]
[X]
[X]
V                                     v2.5.2
(By Nefcore Security)

:: TARGET       : http://testphp.vulnweb.com/
:: THREADS     : 10
:: METHOD       : GET
:: Delay       : 0
:: TIMEOUT     : 15
:: VERIFY      : False
:: Verbosity   : 1

[INF] CRLFSuite v2.5.2 (CRLF Injection Scanner)
[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[INF] Generating random headers
[INF] Initializing Heuristic scanner...
[INF] Heuristic scan completed.
[INF] Initializing WAF detector at 14-07-2023 07:17...
[INF] WAF status: offline
[INF] WAF detection completed.
[INF] Initializing Payload Generator...
[INF] Heuristic scanner found no injectable URL(s).
[INF] Creating resumable_data.crlfsuite.
[INF] Initializing CRLF Injection scanner at 14-07-2023 07:17...

(kali㉿kali)-[~]
$
```


Examples of using CRLFsuite Tool

1. Single URL scanning

```
$ crlfsuite -t http://testphp.vulnweb.com/
```

```
(kali㉿kali)-[~]
$ curlfsuite -t http://testphp.vulnweb.com/

      H
    ---
   [X]
  [X]
 [X]
[X]
V                                     v2.5.2
    (By Nefcore Security)

:: TARGET       : http://testphp.vulnweb.com/
:: THREADS     : 10
:: METHOD       : GET
:: Delay       : 0
:: TIMEOUT     : 15
:: VERIFY      : False
:: Verbosity   : 1

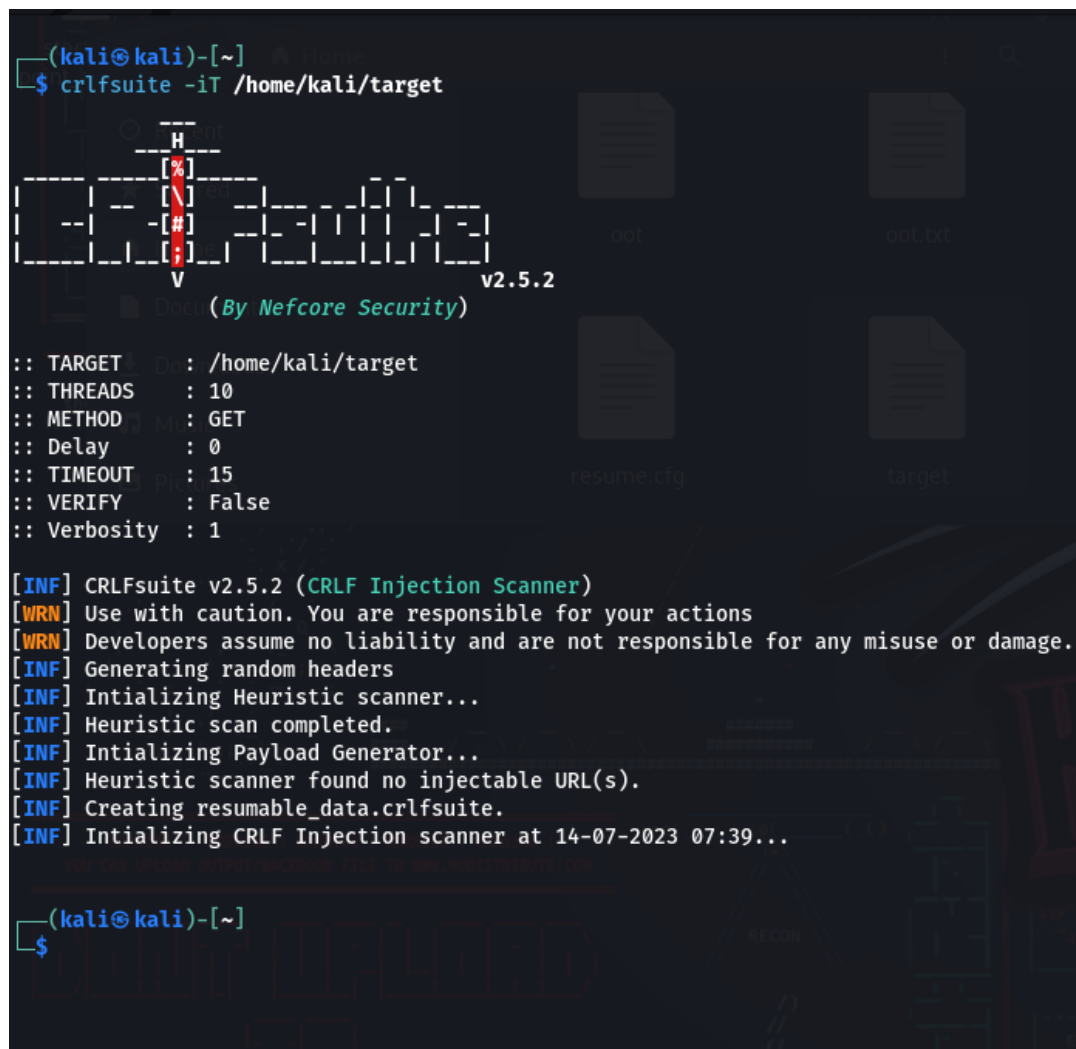
[INF] CRLFSuite v2.5.2 (CRLF Injection Scanner)
[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[INF] Generating random headers
[INF] Initializing Heuristic scanner...
[INF] Heuristic scan completed.
[INF] Initializing WAF detector at 14-07-2023 07:17...
[INF] WAF status: offline
[INF] WAF detection completed.
[INF] Initializing Payload Generator...
[INF] Heuristic scanner found no injectable URL(s).
[INF] Creating resumable_data.crlfsuite.
[INF] Initializing CRLF Injection scanner at 14-07-2023 07:17...

(kali㉿kali)-[~]
$
```

In this example, we will be performing a CRLF Vulnerability scan on our target domain. -t tag is used to specify the domain URL.

2. Multiple URLs scanning

```
$ crlfsuite -iT /home/kali/target
```



```
(kali@kali)-[~]  
$ crlfsuite -iT /home/kali/target  
  
v2.5.2  
(By Nefcore Security)  
  
:: TARGET : /home/kali/target  
:: THREADS : 10  
:: METHOD : GET  
:: Delay : 0  
:: TIMEOUT : 15  
:: VERIFY : False  
:: Verbosity : 1  
  
[INF] CRLFSuite v2.5.2 (CRLF Injection Scanner)  
[WRN] Use with caution. You are responsible for your actions  
[WRN] Developers assume no liability and are not responsible for any misuse or damage.  
[INF] Generating random headers  
[INF] Initializing Heuristic scanner...  
[INF] Heuristic scan completed.  
[INF] Initializing Payload Generator...  
[INF] Heuristic scanner found no injectable URL(s).  
[INF] Creating resumable_data.crlfsuite.  
[INF] Initializing CRLF Injection scanner at 14-07-2023 07:39...  
  
(kali@kali)-[~]  
$
```

In this example, we will be performing a CRLF Vulnerability scan on multiple target domain. -iT tag is used to specify the file contain multiple URLs for scan.

3. POST Method

```
$ crlfsuite -t http://testphp.vulnweb.com/ --method POST
```

```
(kali㉿kali)-[~]
$ crlfsuite -t http://testphp.vulnweb.com/ --method POST
```

v2.5.2
v2.5.2

(By Nefcore Security)

```
:: TARGET      : http://testphp.vulnweb.com/  
:: THREADS    : 10  
:: METHOD      : POST  
:: Delay      : 0  
:: TIMEOUT    : 15  
:: VERIFY     : False  
:: Verbosity  : 1
```

```
[INF] CRLFSuite v2.5.2 (CRLF Injection Scanner)  
[WRN] Use with caution. You are responsible for your actions  
[WRN] Developers assume no liability and are not responsible for any misuse or damage.  
[INF] Generating random headers  
[INF] Initializing heuristic scanner...  
[INF] Heuristic scan completed.  
[INF] Initializing WAF detector at 14-07-2023 07:25...  
[INF] WAF status: offline  
[INF] WAF detection completed.  
[INF] Initializing Payload Generator...  
[INF] Heuristic scanner found no injectable URL(s).  
[INF] Creating resumable_data.crlfsuite.  
[INF] Initializing CRLF Injection scanner at 14-07-2023 07:25...
```

```
(kali㉿kali)-[~]  
$
```

In this example, we will be changing the method of Scan from GET to POST method. --method tag is used to specify the method of the scan.

4. Silent

```
$ crlfsuite -t http://testphp.vulnweb.com/ -sL
```



```
(kali㉿kali)-[~]  
$ crlfsuite -t http://testphp.vulnweb.com/ -sL  
  
(kali㉿kali)-[~]  
$
```

In this example, we will be performing a silent scan. In Silent Scan only the vulnerable targets will be displayed. If target has no vulnerability, it does not return output.

5. Verbose

```
$ crlfsuite -t http://testphp.vulnweb.com/ -v2
```

```
kali@kali:~$ curl -i http://testphp.vulnweb.com/ -v2
```

```
      N  
    (S)  
   (SS)  
  (SSS)  
 (SSSS)  
V  
  
(By Necore Security) v2.5.2
```

```
== TARGET : http://testphp.vulnweb.com/  
== THREADS : 10  
== METHOD : GET  
== Delay : 0  
== TIMEOUT : 15  
== VERIFY : False  
== Verbosity : 2
```

```
[*] CRFSuite v2.5.2 (CRLF Injection Scanner)  
[W] Use with caution. You are responsible for your actions  
[W] Developers assume no liability and are not responsible for any misuse or damage.  
[I] Generating random headers  
[I] Initializing Heuristic scanner...  
[I] GET Request: http://testphp.vulnweb.com/%0d%0aSet-Cookie:necore=crfsuite  
[I] - Status Code: 404  
[I] Heuristic scan completed.  
[I] Initializing WAF detector at 14-07-2023 07:28...  
[I] WAF status: offline  
[I] WAF detection completed.  
[I] Initializing Payload Generator...  
[I] Total URL(s) generated: 203  
[I] Heuristic scanner found no injectable URL(s).  
[I] Creating reusable_data.crfsuite.  
[I] Initializing CRLF injection scanner at 14-07-2023 07:28...  
[I] GET Request: http://testphp.vulnweb.com/%e59b8daae59b8dd%0aSet-Cookie:necore=crfsuite;  
[I] - Status Code: 404  
[I] GET Request: http://testphp.vulnweb.com/%2fxxx:1x2P0AX-XSS-Protection:0%0aContent-Type:text/html%0aContent-Length:39%0a%0axscript%3cscript%3calert(document.cookie)%3c/script%3ex2F..%2F..%2F..%2F..  
[I] - Status Code: 404  
[I] GET Request: http://testphp.vulnweb.com/crfsuite%0a%0a%0aSet-Cookie:necore=crfsuite;  
[I] - Status Code: 404  
[I] GET Request: http://testphp.vulnweb.com/crfsuite%0a%0a%0aSet-Cookie:necore=crfsuite;  
[I] - Status Code: 404  
[I] GET Request: http://testphp.vulnweb.com/%0d%0a%0a%0aSet-Cookie:necore=crfsuite;  
[I] - Status Code: 200  
[I] GET Request: http://testphp.vulnweb.com/%rSet-Cookie:necore=crfsuite;  
[I] - Status Code: 404
```

In this example, we will be displaying the verbose or detailed output of our scan. -v tag is used to display output in verbose mode It has level from v1 to v3.

6. Specifying cookie

```
$ crlfsuite -t http://testphp.vulnweb.com -c "PHPSESSID=c91ef49b7069ca6da302c6798d504eb3"
```



```
(kali@kali)-[~]
$ crlfsuite -t http://testphp.vulnweb.com -c "PHPSESSID=c91ef49b7069ca6da302c6798d504eb3"

  --H--
  [X]
  --#--
  [X]
  --V--
  (By Nefcore Security) v2.5.2

:: TARGET      : http://testphp.vulnweb.com
:: THREADS     : 10
:: METHOD       : GET
:: Delay       : 0
:: COOKIES      : "PHPSESSID=c91ef49b7069ca6da302c6798d504eb3"
:: TIMEOUT     : 15
:: VERIFY      : False
:: Verbosity   : 1

[INF] CRLFsuite v2.5.2 (CRLF Injection Scanner)
[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[INF] Parsing cookies
[INF] Generating random headers
[INF] Initializing Heuristic scanner...
[INF] Heuristic scan completed.
[INF] Initializing WAF detector at 14-07-2023 07:29...
[INF] WAF status: offline
[INF] WAF detection completed.
[INF] Initializing Payload Generator...
[INF] Heuristic scanner found no injectable URL(s).
[INF] Creating resumable_data.crlfsuite.
[INF] Initializing CRLF Injection scanner at 14-07-2023 07:29...

(kali@kali)-[~]
$
```

In this example, we specify the cookie of the site to scan for CRLF Vulnerabilities. -c tag is used to specify the cookie

These are some examples of using this tool for more details refer help option in this tool using `$ crlfsuite -h` command

Note: This Tool can minimize the False Positive but cannot rectify it. So we have to check it manually sometimes

References

- <https://book.hacktricks.xyz/pentesting-web/crlf-0d-0a>
- <https://www.geeksforgeeks.org/crlf-injection-attack/amp/>
- <https://infosecwriteups.com/crlf-carriage-return-and-line-feed-in-short-2023-1647758900f0>
- <https://youtu.be/HgLElaQRc-Q>
- <https://www.briskinfosec.com/blogs/blogsdetail/CRLF-Injection-Attack>
- https://youtube.com/playlist?list=PLtiuVR3b_k4DlnU97uvXDyErEdFd4diDx
- <https://github.com/Raghavd3v/CRLFsuite>