**Sandra Simonov**

September, 2024

Color Box Penetration Testing

This legal disclaimer regulates the usage of this Penetration Testing report and the content of this report. The contents of this document have been prepared from the information collected by the "ECOM" research team. It is worth noting that the information offered in this report constitutes a general framework for security screening and is not intended to ensure protection or compliance with any requirements and guidelines of any authority and the like. Nor can it be concluded that compliance with the recommendations of this report will ensure that the subject of the test is protected and that it does not contain any additional vulnerabilities. Under no circumstances will the Penetration Testing team and the " ECOM " team be responsible for any damage directly or indirectly caused by this test of any kind. The contents of this entire document are confidential and completely confidential and are the intellectual property of the " ECOM " Research Team and may be used only internally for educational purposes only. In addition, it is worth noting that the structure of this document is confidential and is the intellectual property of the " ECOM " research team and may not be used if you are not a member of this team.

**Table of Content**

[REPORT STRUCTURE 3](#_heading=h.gjdgxs)

[ABOUT THE EDITOR 3](#_heading=h.30j0zll)

[EXECUTIVE SUMMARY 4](#_heading=h.1fob9te)

[BACKGROUND 4](#_heading=h.3znysh7)

[PROJECT DESCRIPTION 4](#_heading=h.2et92p0)

[SCOPE & TARGETS 4](#_heading=h.tyjcwt)

[TEST LIMITATIONS 5](#_heading=h.3dy6vkm)

[SUMMARY & ASSESMENT 5](#_heading=h.1t3h5sf)

[CONCLUSIONS 5](#_heading=h.4d34og8)

[ATTACK TREE FOR COMPLEX SCENARIOS 6](#_heading=h.2s8eyo1)

[SETTING GOALS AND OBJECTIVES 7](#_heading=h.17dp8vu)

[IDENTIFIED VULNERABILITIES 8](#_heading=h.3rdcrjn)

[(/LOW/MEDIUM/HIGH/) 10-9](#_heading=h.26in1rg)

[FINDING DETAILS 11](#_heading=h.lnxbz9)

[information disclosure 13-11](#_heading=h.35nkun2)

Broken Authentication  [14-15](#_heading=h.1ksv4uv)

[Broken Access Control 16-17](#_heading=h.44sinio)

[LFI + RCE 18-20](#_heading=h.2jxsxqh)

[Path traversal and Web shell 21-23](#_heading=h.z337ya)

[privilege escalation 24-25](#_heading=h.3j2qqm3)

[Reflected XSS 26-27](#_heading=h.1y810tw)

[stored XSS 28](#_heading=h.4i7ojhp)

Appendices methodology [29-32](#_heading=h.2xcytpi)

[Findings classifications 33-34](#_heading=h.1ci93xb)

# REPORT STRUCTURE

This report contains three different sections:

1. **Executive Summary** - This section includes a brief description of the content of the work as well as a list of the main findings that constitute potential for damage and, as a result, require the organization to take corrective steps in our view.
2. **Details of the tests** - This section details all the tests performed by division into the various areas as well as a description of the information collected in the survey. This section also lists all the findings of the exam, the description of the risks as a result of the findings, and the recommendations for implementation based on the accumulated experience of ECOM.
3. **Appendices** - Brief of the methods used during the penetration test with additional explanation about our rating system fix effort.

## ABOUT THE EDITOR

Sandra Shimonov is Security & Network Administrator at Ness and PT student.

Sandra Shimonov, who holds a CCNA certification, works diligently at Ness and is nearing the completion of her studies at Ecom College.

https://www.linkedin.com/in/sandra-shimonov/

# EXECUTIVE SUMMARY

## BACKGROUND

The "ECOM" Cyber Security Team was asked to perform an applicative penetration test for the Ness on September 2024.

The test scenarios performed included attempts to infiltrate the customer's services, taking the advantage of the built-in weaknesses, taking into account the type of applications/operating systems and the type of components with which the customer works.

The test was performed to detect vulnerabilities that could put Ness at risk and to simulate a situation where an attack occurs while making maximum use of the resources available to the attacker.

This report includes a description of all the vulnerabilities found, a general explanation of them, Proof Of Concept and other findings for the customer to be able to harden his services and increase his level of security.

This test was performed from Ecom College Raul Wallenberg 22, Ramat Hay'il, Tel Aviv., by the Penetration Testing team of "ECOM".

This test was performed using Black Box Penetration Test methodology, and the test content was determined as part of the delineation, both in terms of the topics and components to be tested and the scope of resources that will be allocated to the test. Thus, the test may not detect all the infrastructural and applicative exposures of the client network.

The findings set forth in this document are correct as of the date of the test. Any applicative or infrastructural change made after the end of the test may affect the security level of the client.

It is worth noting that the official contact person on behalf of the company is Dan Cohen and all the tests were matched with him.

## PROJECT DESCRIPTION

### SCOPE & TARGETS

In advance with the client, the test team was given the following goals:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Target Address | | Extra Details |
| 1 | https://techie-world.xyz/ | File name in the server : “flag.txt” | |

This test contains a number of infrastructural / applicative test methodologies in order to examine the level of risk of the information that is output in the identified systems. As part of this examination, the following were examined:

**applicative examination**

* A number of code injection techniques at both the client and server level that can significantly compromise the information stored in this system.
* OWASP TOP 10 includes a variety of vulnerabilities and advanced attack techniques.
* Check for system bugs that can lead to malicious actions at the user level.

**In case of infrastructure inspection**

* Attempt to perform privilege escalation within the server using multiple combined techniques.

## **SUMMARY & ASSESMENT**

During the test, it was found that an attacker could perform brute force using Burp & Intruder along with other misconfiguration defects. The penetration test lasted 14 days and led to the identification of several vulnerabilities such as Broken Access Control, Privilege Escalation, XSS Reflected, and more. An attacker exploiting these flaws could gain control over the server. This may damage the organization's reputation and put the organization and its clients at risk.

## **CONCLUSIONS**

From my professional perspective, the current security level in the client's systems is between low and medium. The rating was given, as previously mentioned, due to the existence of several vulnerabilities such as **Broken Access Control**. The impact of this vulnerability is that an attacker can gain permissions and access to sensitive areas of the site and perform unauthorized actions. Exploiting most of the vulnerabilities mentioned above requires medium technical knowledge. There is room for improvement and the need to address critical security flaws.

## **ATTACK TREE FOR COMPLEX SCENARIOS**

**The following diagram describes each complex attack scenarios that can be applied in the client's system.**

Broken access control

Privilege escalation

Path traversal  
Web shell

Broken authentication

INFORMATION DISCLOSURE

LFI + RCE

## **SETTING GOALS AND OBJECTIVES**

**The following objectives were defined for intrusion testing operations as objectives of paramount importance.**

* **Search for *low hanging fruits* – ACHIEVED**
* **Finding a *number of vulnerabilities* that could endanger the target – ACHIEVED**

**In case of applicable external work:**

* **Performs a vulnerability combination *perform a complex attack* to maximize the attacker's abilities - ACHIEVED**
* **Exposing the target to the ability to *run code remotely* - ACHIEVED**

**In the case of infrastructure internal work:**

* **Obtaining *Domain Admin privileges* in the target's environment - ACHIEVED**
* **Performing *Lateral Movement*, exposing and exploiting additional positions - NOT ACHIEVED**
* ***Utilization of protocols* in favor of the attacker - NOT ACHIEVED**

**IDENTIFIED VULNERABILITIES**

## **(LOW/MEDIUM/HIGH)**

**Privilege Escalation - HIGH**  
This vulnerability is considered highly critical because it allows an attacker to gain elevated privileges in the system, usually root privileges. An attacker with root access can perform any action on the server, including data theft, file deletion, configuration changes, or even shutting down the entire system. In other words, the attacker gains full control of the server, making this vulnerability one that requires immediate attention.

**Path Traversal to Web Shell - HIGH**  
This attack allows an attacker to access sensitive files on the server by exploiting file path manipulation. In this case, the ability to obtain a web shell enables the attacker to execute commands on the server freely, just like in a Privilege Escalation attack. Having a web shell gives the attacker full control of the system, making this vulnerability highly critical.

**LFI + RCE (Local File Inclusion + Remote Code Execution) - HIGH**  
This vulnerability allows an attacker to include local files from the server and execute remote code. This powerful combination can lead to the theft of sensitive information, access to configuration files, command execution on the server, and even privilege escalation. The ability to execute remote code (RCE) is one of the most dangerous vulnerabilities because it grants full control over the server.

**Broken Access Control - HIGH**  
This attack occurs when an attacker bypasses the system's access controls and gains access to protected areas or performs unauthorized actions. The issue with this vulnerability is that it allows direct access to administrative areas or sensitive information, potentially causing significant damage. This vulnerability enables the attacker to act as an authorized user or administrator, making it highly critical.

**Broken Authentication – HIGH**This vulnerability occurs when the authentication process of a system is flawed, allowing attackers to bypass security controls and gain unauthorized access to user accounts. Once inside, attackers can steal sensitive data, manipulate account settings, and even gain access to other parts of the system.

**XSS Stored - MEDIUM**  
Stored XSS occurs when a malicious script is injected into the system and saved there, executing each time a user accesses the affected page. The primary risk here is data theft, session hijacking, or malware distribution among users. However, it requires user interaction (users need to visit the affected page), which is why it is considered a medium-level vulnerability.

**XSS Reflected - MEDIUM**  
Reflected XSS is similar to Stored XSS, but the malicious script is not saved in the system. Instead, it is executed immediately when the user clicks a link containing the malicious code. The impact is similar to Stored XSS, but it requires more user interaction, making it a medium-level vulnerability.

**Information Disclosure - LOW**  
This vulnerability provides the attacker with internal information about the system, such as server versions, database structures, or configuration files. This information is not dangerous by itself but can serve as a foundation for further attacks. For example, an attacker can use this information to identify additional vulnerabilities or perform more targeted attacks. Therefore, this vulnerability is rated as low severity.

### **CVSS**

Calculated by <https://www.first.org/cvss/calculator/3.1>>

**information disclosure**

**CVSS**:4.3/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N

**RISK**: 4.3(Medium)

**DESCRIPTION** :

This vulnerability provides the attacker with internal information about the system, such as server versions, database structures, or configuration files. This information is not dangerous by itself but can serve as a foundation for further attacks. For example, an attacker can use this information to identify additional vulnerabilities or perform more targeted attacks. Therefore, this vulnerability is rated as low severity.

**Broken Authentication**

**CVSS**:6.5/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

**RISK**: 6.5(Medium)

**DESCRIPTION**:

This vulnerability occurs when the authentication process of a system is flawed, allowing attackers to bypass security controls and gain unauthorized access to user accounts. Once inside, attackers can steal sensitive data, manipulate account settings, and even gain access to other parts of the system.

**Broken Access Control**

**CVSS**:6.0/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:L/A:L

**RISK**: 6.0(Medium)

**DESCRIPTION**:

This vulnerability occurs when the authentication process of a system is flawed, allowing attackers to bypass security controls and gain unauthorized access to user accounts. Once inside, attackers can steal sensitive data, manipulate account settings, and even gain access to other parts of the system.

**LFI + RCE (Local File Inclusion + Remote Code Execution)**

**CVSS**:8.0/AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H

**RISK**: 8.0(High)

**DESCRIPTION**:

This vulnerability allows an attacker to include local files from the server and execute remote code. This powerful combination can lead to the theft of sensitive information, access to configuration files, command execution on the server, and even privilege escalation. The ability to execute remote code (RCE) is one of the most dangerous vulnerabilities because it grants full control over the server.

**Path Traversal to Web Shell**

**CVSS**:8.0/AV:N/AC:H/PR:N/UI:L/S:C/C:H/I:H/A:H

**RISK**: 8.5(High)

**DESCRIPTION**:

This attack allows an attacker to access sensitive files on the server by exploiting file path manipulation. In this case, the ability to obtain a web shell enables the attacker to execute commands on the server freely, just like in a Privilege Escalation attack. Having a web shell gives the attacker full control of the system, making this vulnerability highly critical.

**Privilege Escalation**

**CVSS**:8.0/AV:N/AC:H/PR:H/UI:L/S:C/C:H/I:H/A:H

**RISK**: 8.5(High)

**DESCRIPTION**:

This vulnerability is considered highly critical because it allows an attacker to gain elevated privileges in the system, usually root privileges. An attacker with root access can perform any action on the server, including data theft, file deletion, configuration changes, or even shutting down the entire system. In other words, the attacker gains full control of the server, making this vulnerability one that requires immediate attention.

**XSS Reflected**

**CVSS**:8.0/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:N

**RISK**: 4.6(Medium)

**DESCRIPTION**:

Reflected XSS is similar to Stored XSS, but the malicious script is not saved in the system. Instead, it is executed immediately when the user clicks a link containing the malicious code. The impact is similar to Stored XSS, but it requires more user interaction, making it a medium-level vulnerability.

**XSS STORED**

**CVSS**:8.0/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:N

**RISK**: 4.6(Medium)

**DESCRIPTION**:

Stored XSS occurs when a malicious script is injected into the system and saved there, executing each time a user accesses the affected page. The primary risk here is data theft, session hijacking, or malware distribution among users. However, it requires user interaction (users need to visit the affected page), which is why it is considered a medium-level vulnerability.

**FINDING DETAILS**

**1. information disclosure**

**Risk - LOW**

**1.1 Vulnerability Explained**

**Information Disclosure is a security vulnerability where a system exposes information that should not be visible to users or attackers. This information can be direct, such as usernames, passwords, API keys, or sensitive files, or indirect, such as error messages that reveal hints about technologies in use, the database structure, or system code. This vulnerability can occur in several ways, such as:**

* **Detailed error messages: Error messages that contain information about the server or system, such as table names, field names, or paths to sensitive files.**
* **Exposed logs: Logs that are publicly accessible and include sensitive information.**
* **Source code or configuration file exposure: Access to source code or files containing passwords or other access details.**
* **Insecurely displayed information: Displaying sensitive information on the page or within the HTML code itself, such as forms containing unencrypted sensitive data.**

**1.2 PoC for the information disclosure in the ECOM web application:**

* After attempting a code injection to retrieve information through the admin panel, I received an error that revealed the server version and port of the website. Such information can help me identify which vulnerabilities to look for based on the version. If it is an outdated version, it becomes easier to attack and find weaknesses.

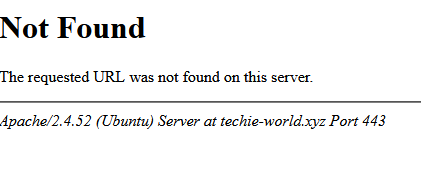
**The code injection:**

C:\Users\sandra\Desktop\עבודת הגשה\information disclosure\מה שגרם לקירוב מידע לאחר שגיאה.png

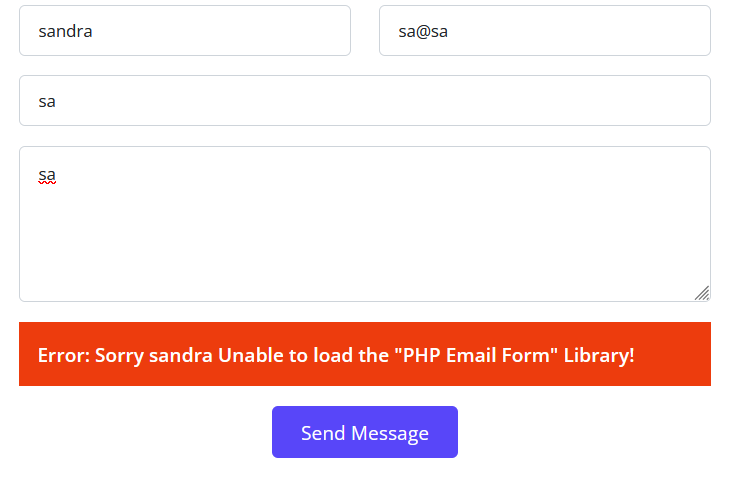
**The error:**



Even after attempting an SQLI injection without success, I received the same error, which appeared as follows:

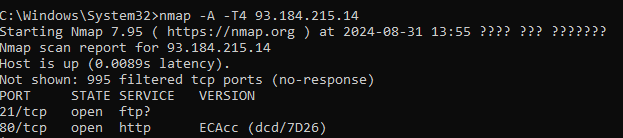


* Another thing that led to information disclosure is the error received after attempting to submit the contact form on the website. After submitting the form, the following error was received:

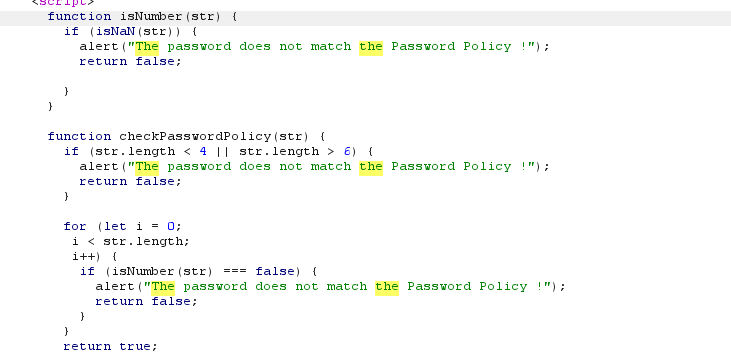


The error provides information about the libraries the form is using (PHP), allowing us to adjust the injections based on the information received**.**

* An attempt was made to run **Nmap** on the website, which resulted in a list of open ports on the server. This vulnerability could lead to an attacker knowing which ports they can target for further attacks.



* When I attempted to log in to the website, I intercepted the packet sent to me using Burp and identified a JavaScript pattern for the password. It specified how many characters the password should contain and noted that only numbers should be used. This information could help an attacker focus on finding the password and allow unauthorized access to the site.



**1.3 Organizational Impact**

**Disclosure of Sensitive Information:** This vulnerability could expose important details such as usernames, passwords, database structure, software versions, or system configurations, which could assist attackers in executing further attacks.

**Increased Risk of More Complex Attacks:** When technical information, such as software versions and open ports, is exposed, attackers can focus their attacks on known vulnerabilities, increasing the likelihood of success.

**Damage to Reputation:** The exposure of sensitive information can lead to a loss of trust from customers and partners, indicating potential issues with the organization's information security.

**1.4 Optional mitigation**

**Protection Against Port Scanning (Nmap):**

**Firewall Hardening** – A firewall should be configured to filter unwanted requests and prevent port scanning. Access to unused ports can be blocked, or ports can be hidden by restricting access only to authorized users.

**IDS/IPS Configuration** – Use intrusion detection and prevention systems to identify and monitor port scans.

**Server Version Protection:**

**Hide Software Versions** – Avoid exposing server version information in error messages or HTTP responses. Configure the server to ensure that version information is not revealed in error messages.

**Regular Updates –** Ensure the server and software are always updated to the latest versions to prevent the exploitation of known vulnerabilities.

Protection Against Error Disclosure in Forms:

**Display General Error Messages –** Instead of showing error messages with technical details, such as the use of PHP libraries, display general error messages that end users can understand without revealing technical information.

**Log Errors Internally –** Store full error details only in internal logs for IT staff.

Protection of Password Patterns in Login Error Messages:

**Strict Validation and Filtering –** Avoid exposing password patterns in error messages. Present a general authentication error without revealing the password length or required characters.

**1.5 Summary**

In the vulnerabilities I found, sensitive details such as the server version, open ports, password patterns, and the use of PHP libraries were exposed. This information can be used by attackers to target more effective attacks, such as Brute Force or attacks on vulnerable ports. The importance lies in the fact that such information disclosure serves as an entry point for further attacks, which could severely impact the organization.

**2. Broken Authentication**

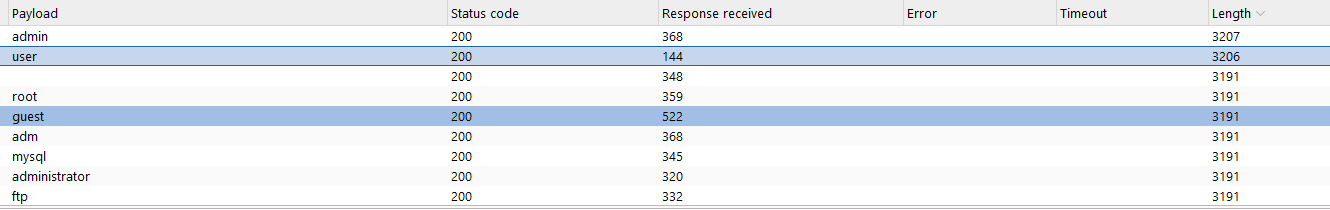
**Risk - High**

**21. Vulnerability Explained**

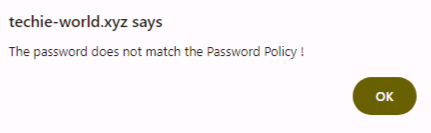
Broken Authentication is a security vulnerability that occurs when an application's authentication mechanisms are incorrectly implemented, allowing attackers to compromise user accounts. This can happen due to weak password policies, improper session management, or vulnerabilities like brute force attacks. Once exploited, attackers can gain unauthorized access to accounts and potentially escalate their privileges within the system. To prevent broken authentication, it's essential to use strong password requirements, implement multi-factor authentication (MFA), and secure session management practices.

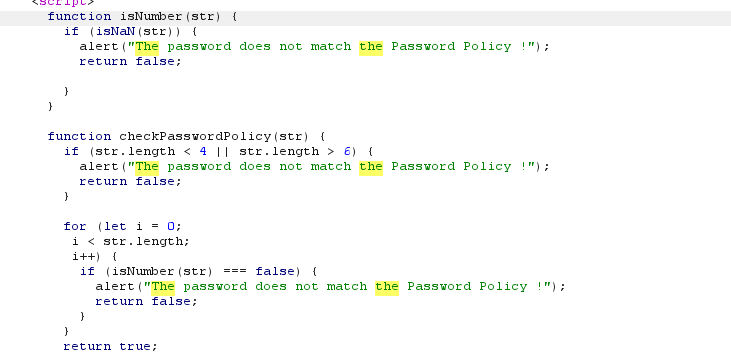
Brute force attack - A Brute Force Attack is a hacking method used to gain unauthorized access by systematically trying all possible combinations of usernames and passwords until the correct one is found. This attack relies on the sheer volume of attempts to eventually "force" a way into the system.

**2.2 PoC for the Broken Authentication in the ECOM web application:**

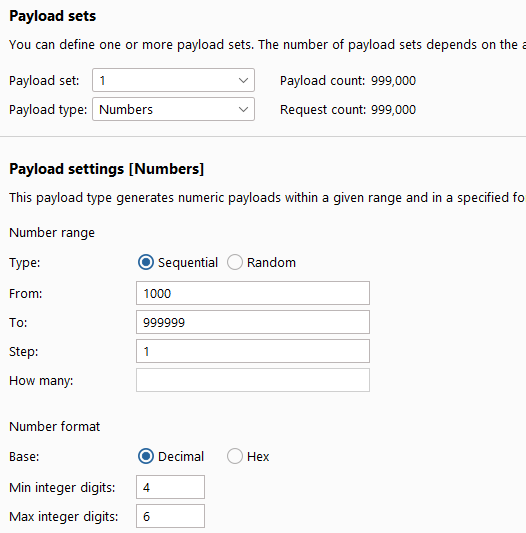
The first thing I did was access the website's login page, where I attempted an SQL injection (SQLI) without success. Afterward, I performed a brute force attack on the username using the Burp tool and its built-in Intruder feature. After receiving the results, I identified two lines with a different length, leading me to conclude that these were the usernames: admin, user.

After discovering the usernames, I proceeded to perform a brute force attack on the password. Here's how I did it: First, when the password was incorrect, the following message was displayed: "The password does not match the Password Policy!" Then, I inspected one of the packets and searched for the password pattern. I found a password pattern in the JavaScript file, which specified that the password must contain only numbers and be between 4 to 6 characters in length. This information gave me a closer approximation, which helped me crack the password using Burp and the Intruder tool.

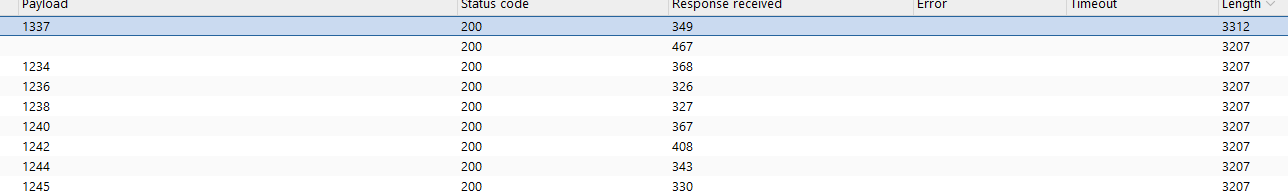




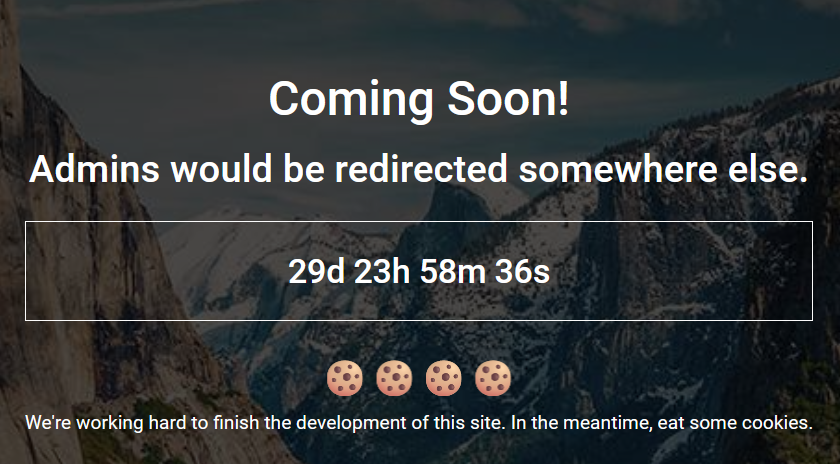
In Intruder, there is an option to set a numeric pattern in the Payload type and configure the character length and number range. Using the information, I discovered about the password pattern, I set this up to match the criteria.



I performed a brute force attack on the password and found one line with a different length, from which I concluded that the correct password for the admin user had been found.



After successfully logging in, the following screen appeared, and I captured its packet in Burp for further inspection.



*Figure - ALERT MADE BY THE WEB APPLICATION MENTIONING THAT THE ATTACK SUCCESSFULLY PERFORMED*

**2.3 Organizational Impact**

This vulnerability allows an attacker to bypass the system's authentication mechanism and access user accounts without knowing the correct password. An attacker could exploit this access to gain administrative privileges, access sensitive information such as user data, modify or delete important data, and perform critical actions within the system. This could lead to severe damage to the organization's reputation, data loss, and potential legal actions.

**2.4 Optional mitigation:**

 Enforce a strong password policy with complex password requirements.

 Limit the number of login attempts (rate limiting) to prevent brute force attacks.

 Implement CAPTCHA on the login form to ensure that the user is human and not an automated tool.

 Use PBKDF2, bcrypt, or scrypt for password storage to make it more difficult to crack them.

**2.5 Summary**

This vulnerability allows an attacker unauthorized access to critical accounts within the organization, making it essential to protect the system from brute force attacks and improper authentication.

**3. Broken Access Control**

**Risk - High**

**3.1 Vulnerability Explained**

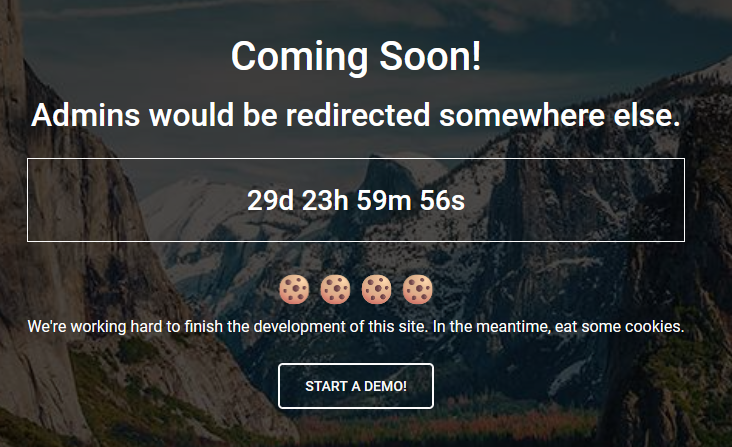
Broken Access Control is a vulnerability that occurs when an application fails to enforce proper permissions, allowing users to access resources or perform actions they shouldn't be authorized to. This can lead to unauthorized data exposure, modification, or escalation of privileges. Common examples include accessing restricted URLs, modifying user roles, or bypassing access controls through insecure APIs. To prevent this, developers should ensure strict access controls, validate user permissions at every layer, and avoid hardcoding permissions in the client-side code.

**3.2 PoC for the Broken Access Control in the ECOM web application:**

After inspecting the packet sent to Burp following my unauthorized login, I noticed cookies in the packet associated with the user named "user." I changed the cookies to the username "admin" and checked what had changed.



The page displayed to me after testing the vulnerability in the cookies was:



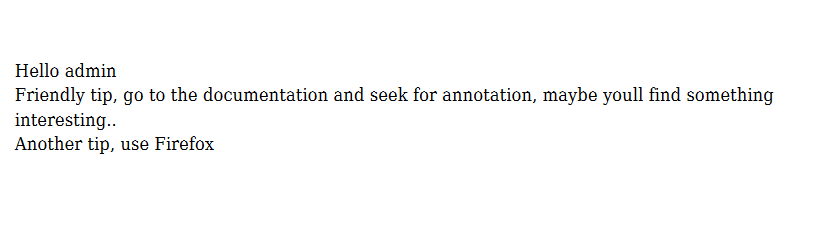
On the new page that was displayed, a new button appeared, leading me to conclude that there was a vulnerability in the cookies. After clicking the button, a page was shown with the message: "Only admins are allowed!"



I continued to exploit the vulnerability in the cookies. On the packet sent to me for the page that indicated I didn’t have access, I again changed the user in the cookies to "admin."



The result that appeared after further exploiting the vulnerability was the admin panel on the PDF page.



**3.3 Organizational Impact**

**Unauthorized access to sensitive data** – An attacker can bypass access control mechanisms and gain access to information that should not be available to them, such as user details, financial information, or sensitive commercial data. This could lead to the exposure of critical information and cause significant business damage.

**Data modification or deletion** – The vulnerability allows an attacker to perform actions they are not authorized to, such as modifying or deleting vital organizational data. This could disrupt essential business processes and even result in the loss of critical information.

**Privilege escalation** – An attacker can exploit the vulnerability to gain higher-level permissions, such as administrative access. This could allow them to control the organization's systems, make changes, or even disable them, potentially severely disrupting business operations.

**3.4 Optional mitigation**

**Cookie Signing & Encryption –** A digital signature should be added to the cookies to ensure they have not been altered by an attacker. Any change to the cookie will invalidate the signature and render the cookie useless. Additionally, the cookie content should be encrypted so that even if an attacker obtains the cookie, they cannot understand or modify its information.

**Security Flags –** Cookies should be configured with the HttpOnly flag to prevent access to cookies through JavaScript, and the Secure flag to ensure the cookie is only sent over a secure HTTPS connection. This will make it harder for an attacker to manipulate the cookies.

**Server-Side Access Control –** The server must verify that every request comes from a user with the appropriate permissions. Do not rely solely on the information in the cookies to validate permissions. Each time a user requests access to a protected area (such as an admin page), the user's permissions should be checked again against a database or authentication server.

**Secure Session Management –** Ensure that session cookies are set to expire after a short period of inactivity or after the user logs out. Additionally, any change in the user’s permissions (such as moving from "user" to "admin") should result in a new cookie being issued with updated details.

**Preventing Manual Cookie Modifications –** Cookies should be validated on the server to ensure they have not been manually modified. If such a modification is detected, the cookie should be considered invalid, and security alerts should be triggered.

**3.5 Summary**

The Broken Access Control vulnerability allows an attacker to modify user cookies and gain unauthorized access to admin areas or sensitive information within the system, bypassing access control mechanisms. An attacker can exploit this to perform critical actions without the necessary permissions. To protect against this vulnerability, encryption and digital signatures should be added to cookies, security flags such as HttpOnly and Secure should be used, and access control mechanisms should be properly and securely implemented.

**4.Local File Inclusion (LFI) and Remote Code Execution (RCE)**

**Risk - High**

**4.1** **Vulnerability Explained**

The **Local File Inclusion (LFI)** vulnerability allows an attacker to include files from the local server within the application. This vulnerability typically occurs when an application dynamically includes files based on user input without proper validation. An attacker can exploit this by manipulating file paths to access sensitive files such as configuration files, source code, or even /etc/passwd on Linux systems. LFI can lead to information disclosure, Remote Code Execution (RCE), or privilege escalation.

**Remote Code Execution (RCE)** occurs when an attacker is able to run arbitrary commands or code on a target machine remotely. In the context of this attack, an attacker could manipulate the input to execute commands on the server, leading to full control over the system, the ability to steal sensitive data, or escalate privileges. RCE is particularly dangerous because it allows the attacker to completely take over the server without any prior access. In this attack, the vulnerability was exploited through the Annotation tag in MPDF. Here is a brief explanation of the tag and its vulnerabilities:

The Annotation tag in PDF documents is used to add comments, footnotes, or other metadata, and is often found in tools like MPDF. When this tag is not properly secured, it can become vulnerable to attacks, particularly code injection or Cross-Site Scripting (XSS). An attacker could inject malicious code through the annotation fields, potentially leading to unauthorized file access, sensitive data exposure, or even Remote Code Execution (RCE) on the server.

**4.2 PoC for the Local File Inclusion (LFI) in the ECOM web application**

After accessing the admin panel, I attempted to retrieve sensitive information from the server. I noticed that the page was using MPDF, and through the Annotation tag, I tried to perform an injection via the packet in Burp to gain access to sensitive files. The injection was successful.

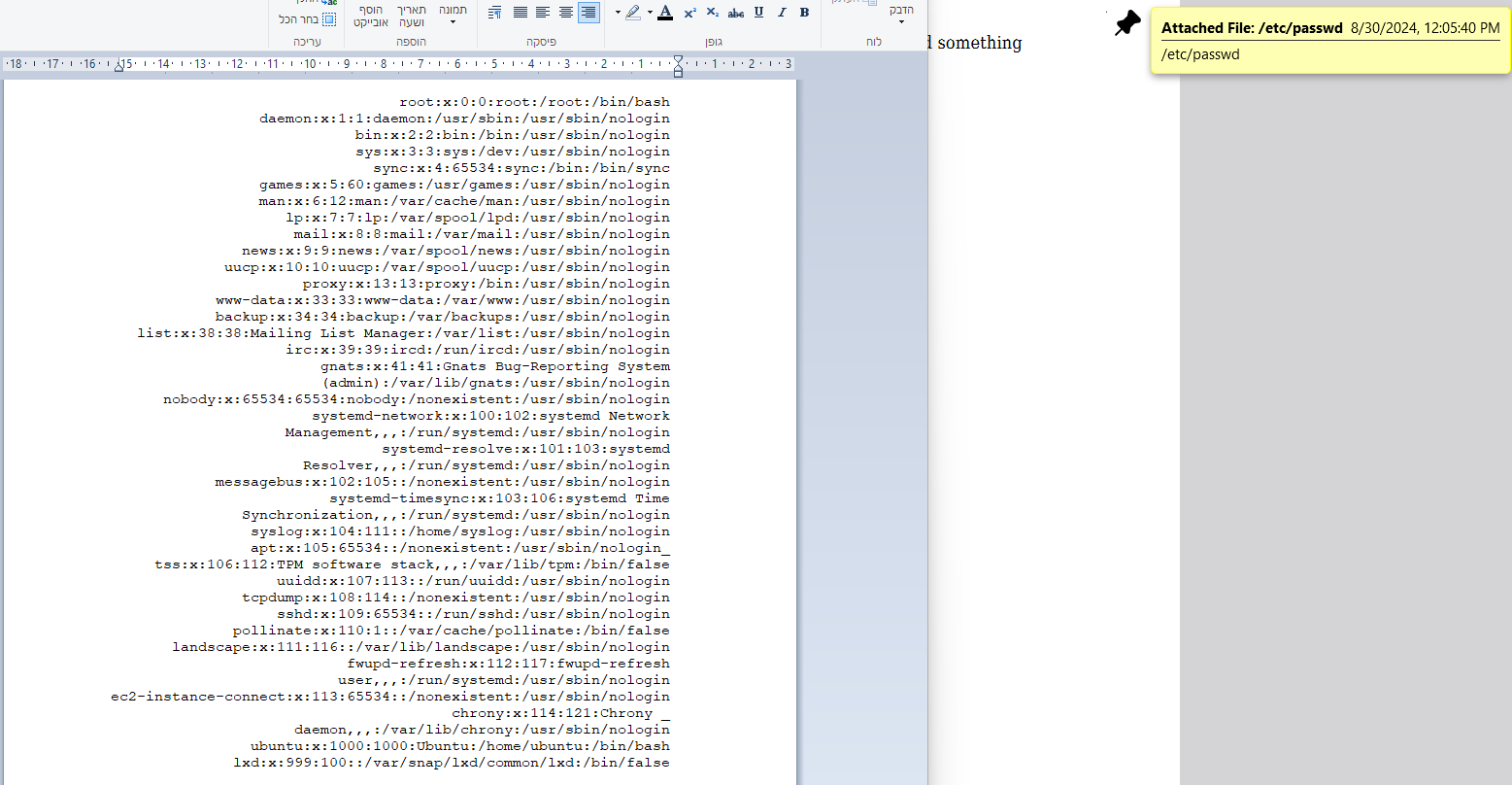
**Here is the injection:**



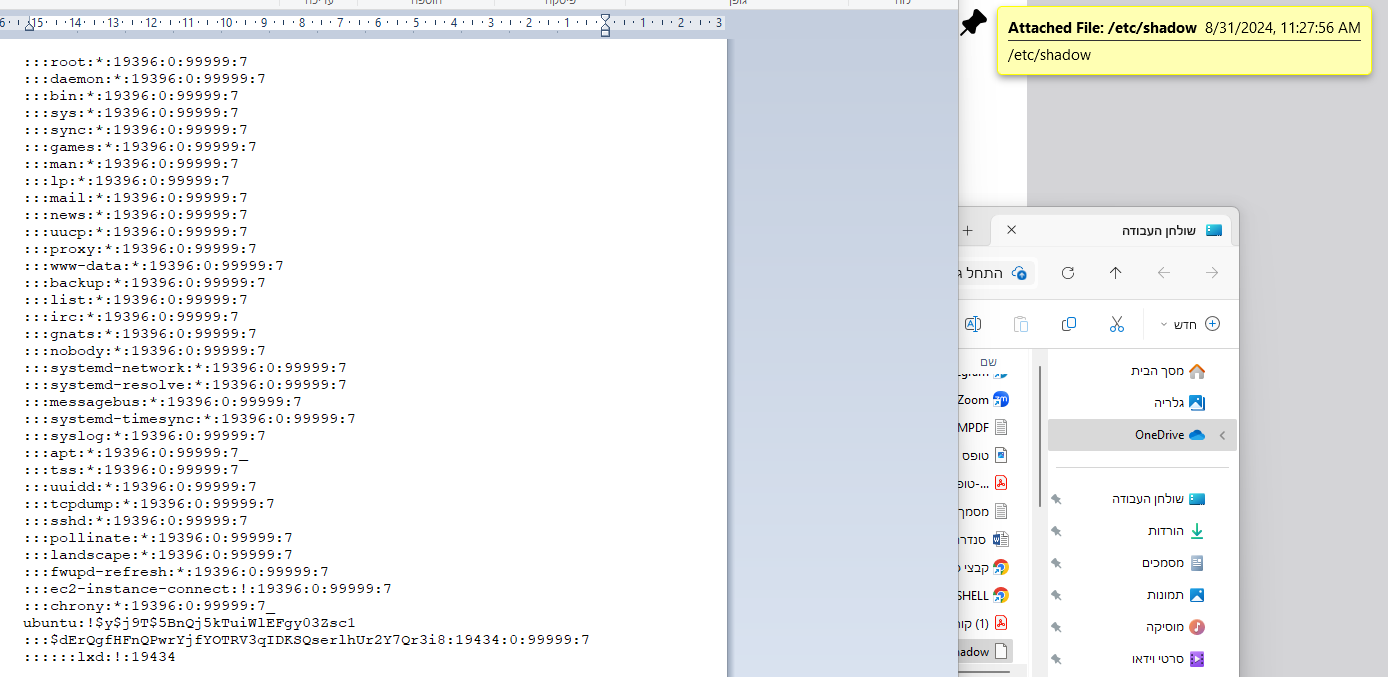
**Explanation of the injection**: Within the Annotation tag, there is an attempt to include a file from the local system, /etc/passwd, which is a sensitive file containing user information. Additionally, there is a PHP code injection within the tag: <?php system('ls'); ?>. This command runs the ls system command, which is a Linux command used to display the contents of the directory where it is executed. The code in the request manipulates the Annotation tag in the MPDF library to inject both local file inclusion (LFI) and system command execution, granting unauthorized access to sensitive files and allowing remote code execution (RCE) on the server. In summary, this is an attempt to inject code that combines access to sensitive files on the server (such as /etc/passwd) and the execution of system commands on the server via the MPDF mechanism present on the site.

After successfully injecting, I gained access through the annotation on the PDF to sensitive files such as /etc/passwd, /etc/shadow, and /etc/crontab.

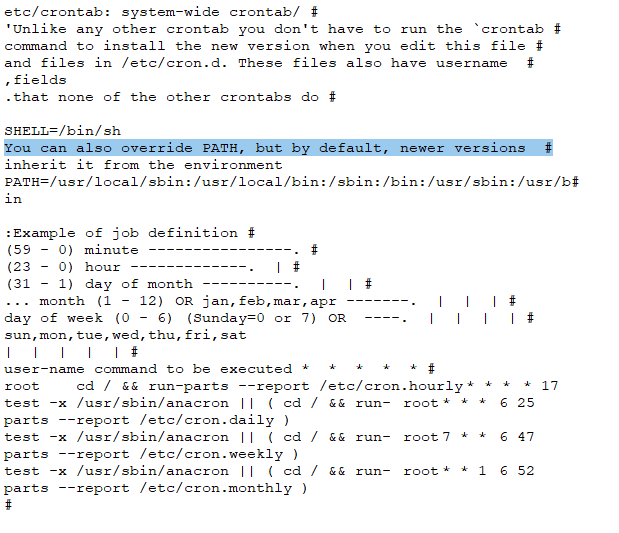
Access to /etc/passwd through the annotation:



Access to /etc/shadow through the annotation:



Access to /etc/crontab through the annotation:



**4.3 Organizational Impact**

Access to Sensitive Information: Access to files like /etc/passwd, /etc/shadow, and /etc/crontab allows an attacker to obtain highly sensitive information about the system's users, including usernames and encrypted passwords. These passwords can be cracked using tools, leading to identity theft or unauthorized access to important system accounts. Risk of Privilege Escalation: Through files like /etc/shadow and /etc/crontab, an attacker can exploit the information to escalate privileges and gain access to higher-level accounts, such as root, thereby gaining full control over the servers and the organizational system. Reputation Damage and Loss of Trust: The exposure of sensitive information and privilege escalation can severely damage the organization’s reputation. When clients and partners discover that their personal information has been exposed or that the system is not properly secured, it can lead to a loss of trust and serious financial consequences, including the loss of customers and legal actions.

**4.4 Optional mitigation**

**Input Validation and Sanitization –** The server must ensure that all input received from users, especially input entered through tags like Annotation, goes through a process of validation and sanitization before being processed. This ensures that malicious code cannot be injected through these inputs.

**Access Control for Sensitive Files –** Sensitive files such as /etc/passwd, /etc/shadow, and /etc/crontab should be protected with strong access control, ensuring that only authorized users can access them, and not through input mechanisms in the application.

**File Restriction –** Sensitive files like /etc/passwd should only be accessible internally and not exposed via the web server.

**Secure Templates –** Ensure that MPDF and related systems do not support code injection through user input fields. Use secure templates to prevent code execution within the annotation tags.

**Server Hardening –** Harden the server configuration to ensure that an attacker cannot run system commands through injected PHP code. This includes disabling dangerous PHP functions like system() or exec() where possible.

**Regular Updates –** Ensure that all components of MPDF, including PHP and the servers, are regularly updated to the latest versions to avoid exploitation of known security vulnerabilities.

By implementing these protections, the risk of code injection can be significantly reduced, and unauthorized access to sensitive files in the system can be prevented.

**4.5 Summary**

You exploited a vulnerability in an MPDF-based web application by injecting malicious code through the **Annotation** tag. This allowed access to sensitive files like /etc/passwd, /etc/shadow, and /etc/crontab, and enabled the execution of system commands through **Remote Code Execution (RCE)**. The attack combined **Local File Inclusion (LFI)** and RCE to gain unauthorized access to critical system data and execute commands on the server.

**5. Path traversal and Web shell**

**Risk - High**

5.1 **Vulnerability Explained**

Path Traversal is a security vulnerability where an attacker gains access to sensitive files on the server by manipulating input used to access files. In this vulnerability, the attacker can use special characters like ../ to navigate outside the application directory and reach sensitive directories or files on the system, such as configuration files, passwords, or personal user data.

Web Shell is a security vulnerability where an attacker successfully uploads or injects a malicious script into the website, allowing them to execute commands on the server remotely. The script enables control over the server through a command-line interface and grants the attacker unauthorized access to files, command execution, database access, and privilege escalation. In our case, the web shell was already present on the server, unrelated to our actions.

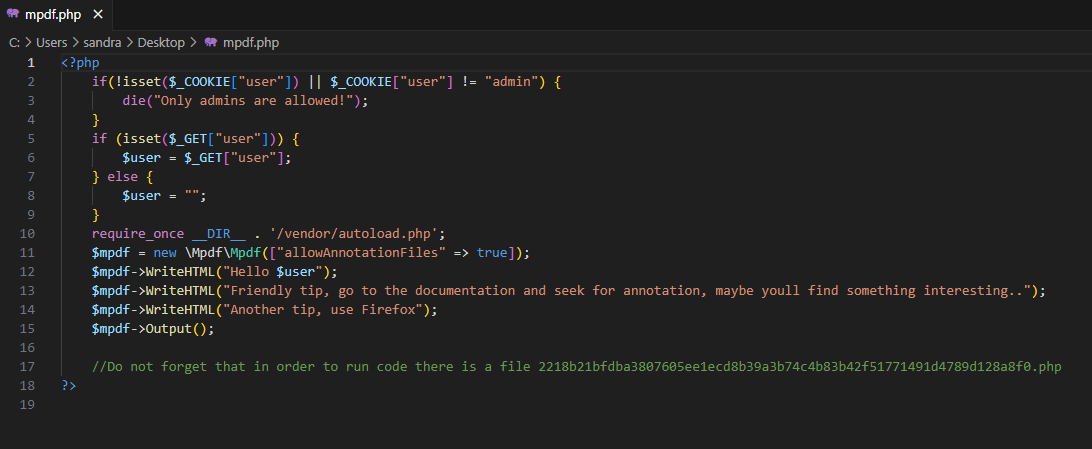
**5.2 PoC for the Path traversal&Web shell in the ECOM web application**

After discovering that the previous injection worked and granted me access to sensitive files, I attempted to inject various commands through the same injection template I found, but without success. I then tried to search for the source code of the PDF page to gather more information that could lead me to another vulnerability. Once I realized that I had the file name I was looking for (Mpdf.php), which appeared in the page's URL, and I also had the path shown in the URL, I attempted an injection to retrieve the file, but it also failed. I understood that I needed to try navigating one directory back, and that’s what I did. I injected code and simultaneously performed a Path Traversal.



After the injection was successful, I accessed the source code.



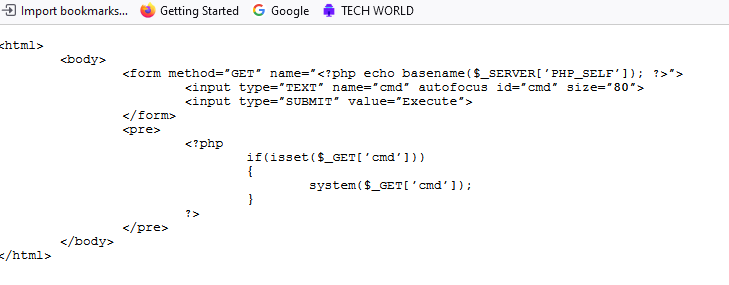


After finding the source code of the website, I noticed a file mentioned in the last line. I decided to import it into the annotation as well to see what it contains.

//Do not forget that in order to run code there is a file 2218b21bfdba3807605ee1ecd8b39a3b74c4b83b42f51771491d4789d128a8f0.php

The injection to see what the file contains:

The content of the file I imported:



After reading the contents of the file, I discovered that it is a web shell for the website's server. I then injected code in the URL to execute this file and gain a web shell.

C:\Users\sandra\Desktop\עבודת הגשה\Annotation+mpdf.php\WEB SHELL\הגעה ךWEB SHELL דרך הנתיב של הקובץ RCE.png

Afterward, a web shell opened on the server, and I began searching for the flag.txt file. I used the find command, which gave me the path to a file with the same name, but it wasn’t the file I was looking for. I quickly realized that the path to the file could only be accessed with higher privileges.

**5.3 Organizational Impact**

**Unauthorized Access to Sensitive Files:** The successful code injection and access to sensitive files via Path Traversal allows an attacker to gain access to critical information such as source code files, databases, or private user data. This could lead to data leaks, which might be used for further exploitation or disruption of the organization's activities.

**Full Control Over the Server**: The discovery and execution of the Web Shell give the attacker direct access to the server. An attacker can execute commands, modify files, steal data, or escalate privileges. This enables full control over the server and could serve as an entry point for more advanced attacks.

**Disruption of System Operations**: Control over the server via the Web Shell allows the attacker to disrupt system operations or halt critical services, which could affect users and result in the shutdown of important services.

**Legal and Regulatory Consequences**: Exposure of sensitive information could lead to lawsuits or fines from regulatory bodies, especially if protected data under privacy laws like GDPR is exposed.

**Reputation Damage:** The discovery of vulnerabilities like Path Traversal and Web Shell indicates serious issues with the organization's information security. If an attacker exploits these vulnerabilities, the organization's reputation could suffer greatly, and customers or partners may lose trust in the systems.

**5.4 Optional mitigation**

**Input Validation and Sanitization**:

Ensure that all user inputs go through strict validation and sanitization processes so that dangerous inputs like ../ (used for Path Traversal) cannot pass through.

Filter any input that might contain malicious scripts or code injection attempts.

Make sure to properly filter encoded inputs (e.g., URL encoding or base64) to prevent bypassing input validation.

**Restrict Access to Sensitive Files:**

Implement strong access control for all sensitive files on the server, ensuring that only authorized users can access them.

Prevent direct access to files via URLs and ensure a secure architecture that blocks direct access to code files and sensitive information.

**Server Hardening:**

Limit the privileges granted to users and applications, ensuring they operate with the least privilege necessary.

Ensure that critical files, such as source code, are located in protected directories and are not accessible via URLs.

Restrict the ability to upload files directly to the server or execute files without proper validation.

**Regular Vulnerability Scanning:**

Perform regular vulnerability scans to detect issues like Web Shells or Path Traversal early and fix them immediately.

Use tools like a WAF (Web Application Firewall) to detect and block injection attempts in real time.

**Regular Updates and Patching:**

Ensure all systems, servers, and third-party software are updated to the latest versions to prevent the exploitation of known vulnerabilities.

**Content Security Policy (CSP):**

Implement CSP to prevent the execution of scripts from unauthorized sources and minimize the risk of executing malicious code through vulnerabilities like Web Shell.

**5.5 Summary**

I identified several critical vulnerabilities in the ECOM application, including **Path Traversal** and **Web Shell**. I was able to inject code and gain access to sensitive files, including source code, and ultimately opened a **web shell** on the server.

**6. privilege escalation**

**Risk - High**

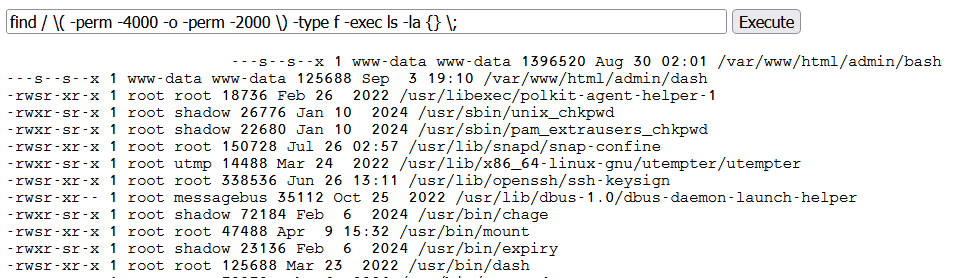
**6.1 Vulnerability Explained**

Privilege Escalation is a vulnerability or technique where an attacker manages to gain higher privileges than those assigned to them in the system. Typically, regular users are limited in their permissions, but if an attacker exploits a vulnerability in the system, they can escalate their privileges and gain access to sensitive operations that require administrator (root) permissions or access to sensitive information in the system.

**6.2 PoC for the privilege escalation in the ECOM web application**

So, I began the process of privilege escalation. I tried checking for vulnerabilities in crontab (including environment vulnerabilities) but without success. I also tried to crack the hash of the Ubuntu user from the /etc/shadow file using John the Ripper, but without success. I attempted to run the sudo -l command to see which commands I could run with sudo and exploit their weaknesses, but I had no such command available. I avoided a kernel attack to prevent damage to the server (and after checking the kernel version, there were no vulnerabilities in this version). Additionally, I searched for exploits in Metasploit for the current server version but found nothing.

This led me to focus on checking for vulnerabilities in SUID and SGID files. I began by scanning all files with the sticky bit enabled, and to view them, I executed the following command:

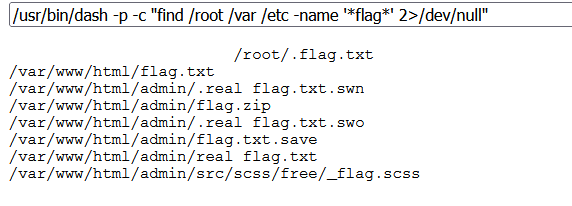


After the list opened, I noticed several files through which I could exploit their vulnerabilities. Among them were bash, dash, and python3. After unsuccessful attempts with some of them, I began to focus on dash.

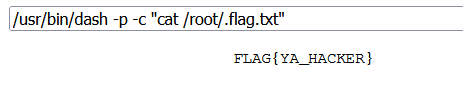
C:\Users\sandra\Desktop\עבודת הגשה\Annotation+mpdf.php\privilege escalation\מציאת ה DASH.png

When there is a vulnerability with SUID on a file like dash, it allows the file to be executed with root privileges, even if the user is not root. This means that with these elevated privileges, the file can be exploited to run commands on the system with higher privileges and perform Privilege Escalation.

So I ran the following command, which exploits dash to execute the find command with root privileges, and then I found the path to the desired file.



After finding the path, I used dash again to run a command that reads the contents of the file.



Based on the content of the file, I understood that I successfully reached the target.

**5.3 Organizational Impact**

**Unauthorized Access to Sensitive Information**: Successful privilege escalation allows an attacker to operate with root or administrative privileges, granting full access to sensitive files, user data, passwords, and other critical information stored in the system.

**Full Control Over the Server**: The attacker can exploit their access to execute commands on the server, modify configurations, delete or steal data, and perform actions that endanger system functionality, including disabling critical services or running malicious code.

**Damage to the Organization's Reputation**: Discovery of such a vulnerability could harm the organization's reputation, as it highlights a serious failure in information security. Customers and partners may lose trust in the organization if they find out that the system was not properly secured and was exposed to an attack.

**Legal and Regulatory Consequences**: If the exposed information includes personal user data or sensitive business information, the organization could face lawsuits or regulatory fines, especially if it fails to comply with standards like GDPR.

**Risk of Further Attacks**: After privilege escalation, the attacker can create additional access points within the system, such as installing backdoors, enabling further attacks in the future or maintaining ongoing access to the system.

Due to these risks, Privilege Escalation vulnerabilities are considered highly critical and require immediate attention to prevent widespread damage to the organization.

**5.4 Optional mitigation**

**Removing Unnecessary SUID and SGID Permissions**: Review all files marked with SUID and SGID and remove these permissions from files that do not need to run with elevated privileges. In most cases, these permissions are not necessary for most files in the system, especially for files like dash, bash, and python3.

**Minimalist Privilege Management**: Apply the Principle of Least Privilege to ensure that only essential files and processes run with SUID or SGID permissions. It is important to review the files being used and update their permissions to fit the actual needs only.

**Monitoring and Tracking SUID and SGID Activity**: Implement monitoring mechanisms to detect unusual access or use of files with SUID or SGID permissions. This helps identify exploitation attempts in real-time and respond accordingly.

**Hardening Critical Files**: Critical files with SUID and SGID permissions should be protected using strong access control policies. Additionally, ensure that regular users cannot easily modify them.

**Regular Updates and Security Patches**: Ensure that the system and all sensitive files are kept up to date with the latest versions to prevent the exploitation of known vulnerabilities in SUID and SGID.

**5.5 Summary**

I identified a SUID vulnerability in files like dash, which allowed me to execute commands with root privileges and perform privilege escalation. I used these commands to find and read a sensitive file on the server.

6. **Reflected XSS**

**Risk - Medium**

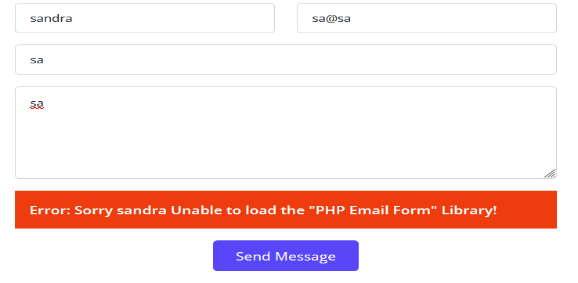
**6.1 Vulnerability Explained**

Reflected XSS (Cross-Site Scripting) is a security vulnerability where an attacker injects malicious script into an application, and the injected code is "reflected" back to the user as part of the server's immediate response. The vulnerability occurs when user input is incorporated into the page's content without proper validation or sanitization, such as in the URL or a form.

In this attack, the malicious code runs in the victim's browser, which can lead to data theft, session hijacking, or manipulation of the content displayed on the page. To prevent this vulnerability, strict validation and sanitization of user inputs should be performed, ensuring no unfiltered content is returned to the user.

**6.2 PoC for the Reflected XSS in the ECOM web application**

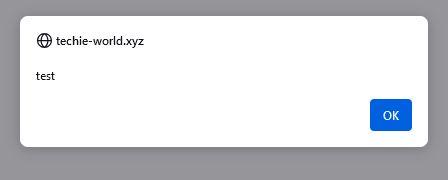
Initially, I identified that the website has a form that can be filled out for contact purposes. When I submitted the form, I received the following error, which led to an information disclosure confirming that PHP is being used.



Based on the fact that my name was reflected in the error itself, I realized that there was information being reflected back on the website. I assumed it was a template and attempted an SSTI injection, but it was unsuccessful. I then intercepted the packet in Burp and changed the cookies from user to admin to gain additional privileges. I accessed the part where the form submitter's name is entered, and there I injected XSS code in PHP (based on the information I gathered from the error, confirming that PHP code was in use).



After submitting the request, I received a 200 OK response, and then I checked whether the code had executed. Indeed, the result was successful as the alert I injected was displayed on the screen.



**6.3 Organizational Impact**

**Theft of User Information:** An attacker can use Reflected XSS to inject malicious scripts into the browsers of other users. These scripts can steal important information, such as cookies containing active sessions, login data, or personal information, enabling identity theft or unauthorized access to user accounts.

**Damage to the Organization's Reputation**: If attackers manage to inject malicious code that affects users, it can severely harm the trust of customers and partners in the organization. Users may lose confidence in the system if they discover it is insecure and allows for harmful code injections.

**Distribution of Malicious Content:** An attacker can use Reflected XSS to inject malicious content such as malware or phishing links into the website, targeting other users. This can lead to malware distribution, personal information theft, or ongoing attacks on users.

**Impact on Website Functionality:** Injected malicious code can alter the behavior of the website for users, leading to operational disruptions, user complaints, and loss of active users.

**Legal Consequences:** Depending on the type of information stolen or compromised, this vulnerability could lead to lawsuits or regulatory fines, especially if sensitive user information is involved. Additionally, non-compliance with security standards (such as GDPR) could result in penalties.

**6.4 Optional mitigation**

**Input Validation and Sanitization** – Ensure that all input received from users is filtered and validated before it is processed on the server to prevent malicious code injections like XSS.

**Cookie Security** – Make sure that cookies are protected by security flags like HttpOnly and Secure to prevent cookie manipulation and unauthorized changes to user privileges.

**Escape Output** – Apply output escaping for all content reflected back on the page, especially user inputs, to prevent injected scripts from executing.

**Content Security Policy (CSP)** – Implement a Content Security Policy (CSP) to restrict the execution of scripts on pages and prevent malicious code injection.

**Access Control** – Ensure that users cannot modify cookies or gain additional privileges without proper authorization, such as the scenario where you changed the cookie from "user" to "admin."

By implementing these measures, the risk of successful attacks like this can be significantly reduced.

**6.5 Summary**

Reflected XSS is a security vulnerability where an attacker injects malicious script into a web page, and the injected code is directly reflected in the response to the user. Typically, the malicious code comes through user input (such as a URL or form) and is executed in the victim's browser. In my attack, I succeeded by leveraging information disclosure, a cookie vulnerability, and a form vulnerability.

**7. Stored XSS**

**Risk – Medium**

**7.1 Vulnerability Explained**

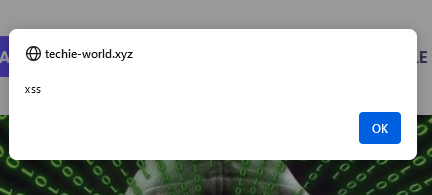
Stored XSS (Cross-Site Scripting Stored) is a security vulnerability where an attacker injects malicious script into a website, and the script is permanently stored on the server (for example, in a database or a file). Unlike Reflected XSS, where the script is executed immediately with the request, in Stored XSS, the malicious script is executed every time a user visits the page where the code is stored. The danger of Stored XSS is higher because the code is saved on the server, and any user who visits the affected page can be exposed to the injection. This can lead to data theft, session hijacking, or the distribution of malicious content to other users. To prevent this vulnerability, it is essential to filter and validate all input entered into the website and ensure that malicious code cannot be inserted into forms or comment fields.

**7.2 PoC for the Stored XSS in the ECOM web application**

When I accessed the website, I noticed images displayed in the Portfolio section. I inspected the page and searched for the image section within the HTML tags. In the IMG tag, I found an opportunity to try injecting XSS. So, I performed an injection that looked like this:

C:\Users\sandra\Desktop\עבודת הגשה\XXS (FORM)\xss stored\portfolio (שינוי בתגית IMG).png

After the injection, I saw that it was successful. The proof was the **alert** message that appeared on the screen afterward.



**7.3 Organizational Impact**

**Data Theft:** An attacker can inject malicious scripts into the browsers of other users visiting the site, which could lead to the theft of data such as session cookies, login credentials, or other sensitive information.

**Damage to Reputation:** If users are affected by the vulnerability, it could damage the trust of customers and partners in the organization, leading to a negative impact on the organization's reputation.

**Distribution of Malicious Content**: An attacker can use this vulnerability to spread malware, malicious links, or any other harmful content to site visitors, causing widespread damage to users.

**7.4 Optional mitigation**

**Escaping User Inputs**: Ensure that all user inputs displayed on the page, especially in text fields, comments, and form fields, are escaped. This will prevent malicious code from executing when the information is shown on the page.

**Use of Content Security Policy (CSP)**: Implement a Content Security Policy (CSP) that restricts the ability to run malicious scripts. CSP limits script execution in the browser to only pre-approved sources.

**Encryption of Sensitive Data**: Ensure that all sensitive data on the site is encrypted, so that even if this information is exposed through an XSS vulnerability, it remains protected.

**Strict Access Control**: Implement strict access control so that users without the proper permissions cannot add or edit content in sensitive areas of the system.

**7.5 Summary**

The vulnerability I found is Stored XSS, where an attacker can inject malicious scripts that are stored in the system and executed in the browsers of other users. This means the attacker can steal sensitive information, spread malware, and harm both users and the organization's reputation.

# **APPENDICES**

## **METHODOLOGY**

The work methodology of our penetration testing team includes some of the following potential inspected information according to the client's needs:

### APPLICATIVE PENETRATION TESTS

**The test was conducted identify the following:**

* Vulnerable functions used in the code.
* Un-sanitized Input provided by the user.
* Well known vulnerabilities exists in the system.
* Insecure error handling.
* Cross-user manipulations.
* Unhandled manipulation that can be used by an attacker.
* Sensitive information leakage.

**Performed general inspection of the code if requested by the client. In addition to the usage of automated tools to identify vulnerabilities and potential issues in the target application.**

**Understanding the system logic –** Before performing the test, the testers watched and examined the system in order to understand its purpose and mode of operation. During this exam the examiners try to understand the following:

* **Client Requests**:
  + Examined hidden parameters.
  + Examine important parameters that are in outgoing requests
  + Notice all the request titles heading towards the server
  + Examine paths and form of loading of data on the site
* **Server Answers:**
  + Check when a cookie is created or when the content of the cookie changes.
  + Examine the number of errors that recur from the site.
  + Examine when the server returns redirection in order to find *Open Redirect*.
* **Understanding the customer side of the system:**
  + The testers examined what could be done on the customer side of the system. Also, in what language is the system written and are there any comments in the client-side code.
  + The testers examined which JavaScript functions are called in the code.
  + Examined whether HTML code can be injected next to a client.
* **Data collection and scanning:**
  + Find additional servers and get information about those servers.
  + Scans were also performed by dedicated tools in order to find known vulnerabilities on the site.
* **Checking the user's identity management and authorization:**
  + The examiners examined the permission level in the system, what permission level they are at and whether it is possible to switch to another permission level.
  + In addition, we examined whether there are different APIs that allow manipulation of the authorization level in the system or do not check the authorization level at all.
* **Checking the user authentication process:**
  + The testers examined the mechanism of connection to the system, whether there is Anti-Automation protection such as CAPTCHA.
  + Attempts have also been made to locate and exploit JWT and SSO systems in order to detect security flaws in these protocols.
* **Authentication of the resulting input:**
  + The testers examined the user's call management in addition to verifying the inputs sent from the client alongside the server. Attempts were also made and exploits of systems to upload documents to the system, file reading systems and even injecting malicious code into the system.
* **Error management in the system:**
  + During the test, errors that were repeated by a customer were identified and conclusions were drawn according to the same errors that helped the testers during this test.
* **Logical Bypasses:**
  + During the test, the testers questioned the system logic in order to check the transition between forms, switching between one user and another, making a registration in the system and more. In order to test whether non-programmed operations can be performed by default.
* **Testing of potential attack vectors, and provideing a working POC for examination.**
* **The test result is a detailed report contains all the findings details about the vulnerabilities found:**
  + CVSS
  + RISK
  + DESCRIPTION.
  + POC
  + DETAILS
  + RECOMMENDED MITIGATIONS
* **Additionally, the following elements may be performed due to the client's request:**
  + Conducting a re-test to the system in order to verify the security again.
  + Providing the development team from "ECOM" to support the client during the mitigation process.
  + Providing the penetration testing team from "ECOM" explain in more depth about the report.

### 

### **INFRASTRUCTURE PENETRATION TEST**

The test was performed in a format that would allow the company to identify the main risk points that exist in the systems and infrastructures of the company under test and treat them in a way that will allow it to reduce the chance of realizing exposure to harm and leak information into the company's and businesses.

The computer systems test was performed in five main stages:

* **First stage** – an overview of the existing computer system and mapping of all the components in the computer system and the information processing processes.
* **Second stage** – planning the test stages as a result of the mapping.
* **Third stage** – comprehensive technological tests and processes of the various components.
* **Fourth stage** – sorting and analyzing the risk outline.
* **Fifth stage** – risk assessment and corrective recommendations.
* **For the purposes of documenting the existing situation:**
  + The security survey was conducted while studying the computer system in the demarcation of the test and how it operates on the basis of conducting questioning and examining various relevant factors and operational processes. In addition, various components and technological means related to the information systems and relevant to the various survey topics were examined.
* **The questioning and documentation**:
  + was carried out in a way that enabled learning and understanding of all the existing and implemented administrative and operational processes in practice in everything related to the security of the information systems in the organization.
* **The examination of the technological issues**:
  + was carried out in a way that enabled us to become familiar with the protection circuits in the system and their practical implementation, through the use of logical and physical security measures as well as the configuration of the hardening components of the technological system.
* **Resilience tests (optional)**:
  + simulate a potential intruder into the information systems, for the purpose of examining the quality of the application of the parameters and the logical security measures of the various technological components.

## FINDINGS CLASSIFICATIONS

The purpose of the presentation in the manner illustrated above is on several levels:

1. **The vulnerability name -** A main vulnerability of which an examination is performed.
2. **Description of the test -** Main description about the vulnerability.
3. **Findings of the test -** Findings that clearly and concisely describe an existing situation. The purpose of the section is to document the existing situation as found during the examination. The test results can be normal or in a status that endangers the entire array tested, at the level of exposure to damage to activity continuity, leaked sensitive information or damage to property and people.
4. **The risks as a result of the existing situation -** A rating that clarifies what is the risk arising to the customer from the findings.
5. **Severity of the damage** **-** The method of determining the level of damage is performed according to the following details:

**Critical** – For the following risks:

* + The realization of the risk will lead to a horizontal impairment in the information availability of the organization's systems and / or infrastructure.
  + The realization of the risk will lead to the disclosure of information that may threaten the stability of the organization or endanger human lives.
  + Unauthorized disruption / alteration of information that may threaten the stability of the organization or endanger human life.

**High** - For the following risks:

* + The realization of the risk will impair the information availability of a sensitive system.
  + Exposure of sensitive information.
  + Unauthorized disruption / change of sensitive information in the system.

**Medium** - For the following risks:

* + The realization of the risk will lead to the immediate and direct shutdown of an insensitive system.
  + The realization of the risk may, in an uncertain manner, lead to the shutdown of a sensitive system.
  + Exposure of non-public inside information.
  + Unauthorized disruption / change of information that is not sensitive in the system in a way that will require a lot of effort in data recovery.

**Low** - For other serious risks.

**Informative** - For information provided.

1. Probability of realization - how to define the reasonableness of the risk:

**Critical** - A critical likelihood will be defined in a situation where it is found that the exposure has already been actually exercised (by a non-examining entity) or is available for immediate exploitation without the need for any preparation.

**High** - High probability will be defined in the following situations:

* + The risk can be realized by Social Engineering simply.
  + No technological knowledge is required or the required technological knowledge is not extensive.
  + Well-documented behavior.
  + The time required to realize the risk is small.
  + Ability to use mechanized tools.

**Medium** - Moderate likelihood will be defined in the following situations:

* + Information is available online.
  + Well-documented behavior.
  + The period of time required to realize the risk is long.

**Low** - Lower than moderate probability or in situations only theoretically there is a chance of exploiting the weakness.