

# Insecure Design Assessment

Author: Akpovona Agbaire

Institution: University of Derby

Submitted in partial fulfilment of the requirements for the MSc in Cyber Security

## Contents

|                                                             |    |
|-------------------------------------------------------------|----|
| Insecure Design .....                                       | 3  |
| Introduction .....                                          | 3  |
| Critical Review and Illustration of Suitable Exploits ..... | 4  |
| I.    Identifying Opened Ports:.....                        | 4  |
| II.   Identify Other Services:.....                         | 4  |
| III.  Understanding Our Findings:.....                      | 5  |
| IV.  Confirming Exploit: .....                              | 5  |
| Business Impact .....                                       | 7  |
| Protection Solution .....                                   | 8  |
| (a)  Threat Modelling:.....                                 | 8  |
| (c)  Encryption: .....                                      | 8  |
| (d)  Untrusted By Default: .....                            | 8  |
| (e)  Fail Application Securely: .....                       | 8  |
| (f)  Event Monitoring and Logging:.....                     | 9  |
| (g)  Learning from Mistakes:.....                           | 9  |
| (h)  Code Review: .....                                     | 9  |
| (i)  Penetration Testing: .....                             | 9  |
| (j)  Segregation: .....                                     | 9  |
| Recommendation .....                                        | 10 |
| References .....                                            | 11 |

# Insecure Design

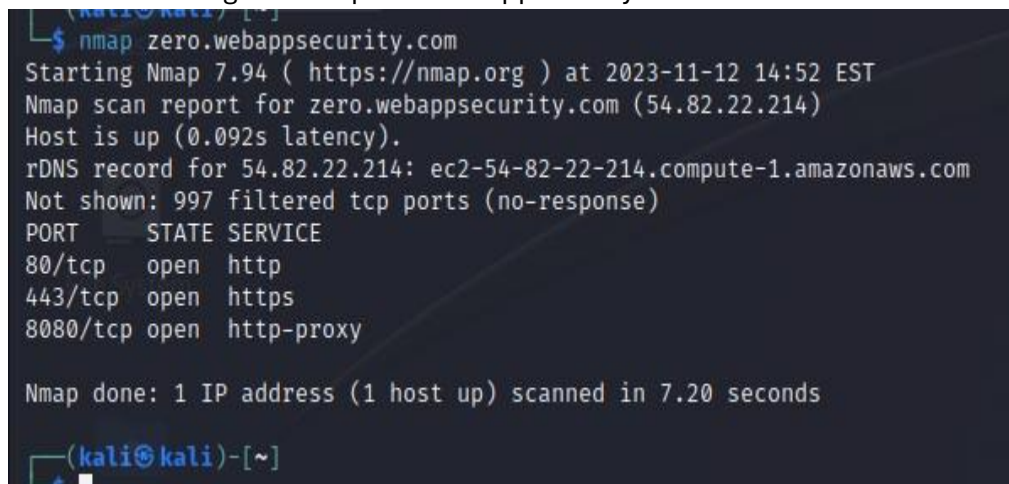
## Introduction

Insecure design refers to a type of design that lacks sufficient security measures (Cunha, 2022). These design flaws could be software, hardware, or infrastructure designs (Cunha, 2022). Hence, it is important to have a lot of planning and careful thought when designing a web application before actual implementation starts (Pant, 2022). Insecure design is a new category introduced in 2021 that focuses on risks associated with design flaws (OWASP, 2021). It is a wide category displaying different weaknesses and expressed as “missing or ineffective control design” (OWASP, 2021). Insecure design should not be likened to insecure implementation because they have different root causes and solutions (OWASP, 2021). A secure design might have some implementation defects. But when a web application was developed without putting security into consideration, a perfect implementation would not solve this issue because by default it was never designed with needed security controls in place (OWASP, 2021). This can occur when there is a lack of business risk profiling inherent in the application being developed and as such failure to ascertain the level of security design that is required (OWASP, 2021). OWASP believes that if we genuinely intend to “move left as an industry, we need more threat modelling, secure design patterns and principles, and reference architecture (OWASP, 2021).

## Critical Review and Illustration of Suitable Exploits

There is no unique way to exploit vulnerabilities since every web application has different flaws (Petranović & Žarić, 2023). However, irrespective of the web application, vulnerabilities, or penetration testing tool in use, information gathering is always the first step (Petranović & Žarić, 2023). In this research, we will be making use of NMAP (Network Mapper) as our penetration tool while the test will be done on the zero.webappsecurity.com web application. This test is done on a Kali Linux virtual machine.

- I. **Identifying Opened Ports:** We start by scanning the first 1000 ports to check for any active ports. The opened port would be our focus for the penetration testing. This can be done using the Nmap zero.webappsecurity.com command as shown below.



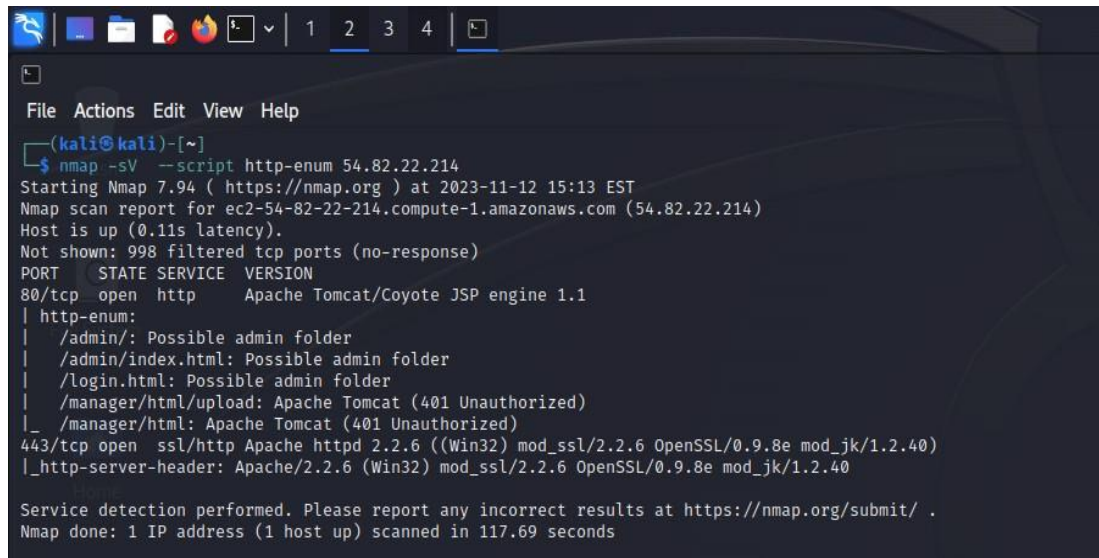
```
(kali@kali) [~]
$ nmap zero.webappsecurity.com
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-12 14:52 EST
Nmap scan report for zero.webappsecurity.com (54.82.22.214)
Host is up (0.092s latency).
rDNS record for 54.82.22.214: ec2-54-82-22-214.compute-1.amazonaws.com
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 7.20 seconds

(kali@kali) [~]
```

Figure 1: Identifying Opened Ports

- II. **Identify Other Services:** Using the `--script http-enum` built-in command, we can attempt to identify web servers and versions as well as virtual hosts or directories on the server. This can be seen below.



```
(kali@kali)-[~]
$ nmap -sV -script http-enum 54.82.22.214
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-12 15:13 EST
Nmap scan report for ec2-54-82-22-214.compute-1.amazonaws.com (54.82.22.214)
Host is up (0.11s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache Tomcat/Coyote JSP engine 1.1
|_ http-enum:
|   /admin/: Possible admin folder
|   /admin/index.html: Possible admin folder
|   /login.html: Possible admin folder
|   /manager/html/upload: Apache Tomcat (401 Unauthorized)
|_  /manager/html: Apache Tomcat (401 Unauthorized)
443/tcp   open  ssl/http Apache httpd 2.2.6 ((Win32) mod_ssl/2.2.6 OpenSSL/0.9.8e mod_jk/1.2.40)
|_ http-server-header: Apache/2.2.6 (Win32) mod_ssl/2.2.6 OpenSSL/0.9.8e mod_jk/1.2.40

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 117.69 seconds
```

Figure 2: Identifying Other Services

- III. **Understanding Our Findings:** With the scans, we have been able to identify key vulnerabilities to exploits. We have been able to identify the type of server in use as well as the version in use. This is key because a simple search with the phrase “Apache Tomcat vulnerabilities” would reveal its vulnerabilities. Also, we can find a list of directories we had no idea about. This can be typed in directly into our browsers to find a suitable attack vector as shown below.

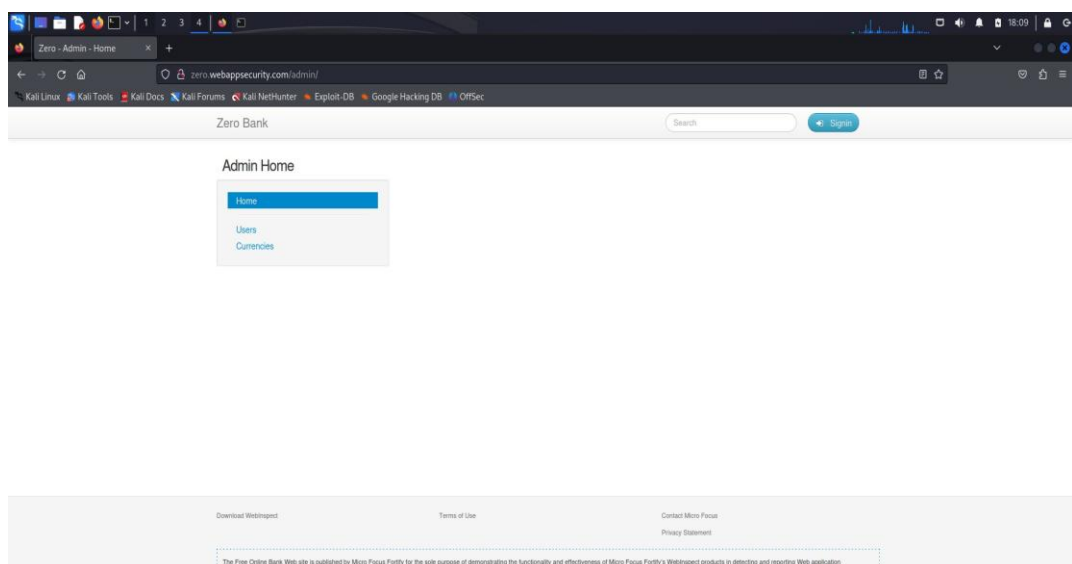


Figure 3: Broken Access via Unprotected Admin Directory

- IV. **Confirming Exploit:** Gaining access by adding /admin to the initial URL would give us access to the admin user account which is a broken access control vulnerability. But what is more worrisome is the fact that the password and PII (personally identifiable information) are all stored in plaintext as seen below. Now this

is an insecure design as listed as a common weakness enumeration (CWE). CWE-257 and CWE-312 are the specific weaknesses or vulnerabilities found in this case.

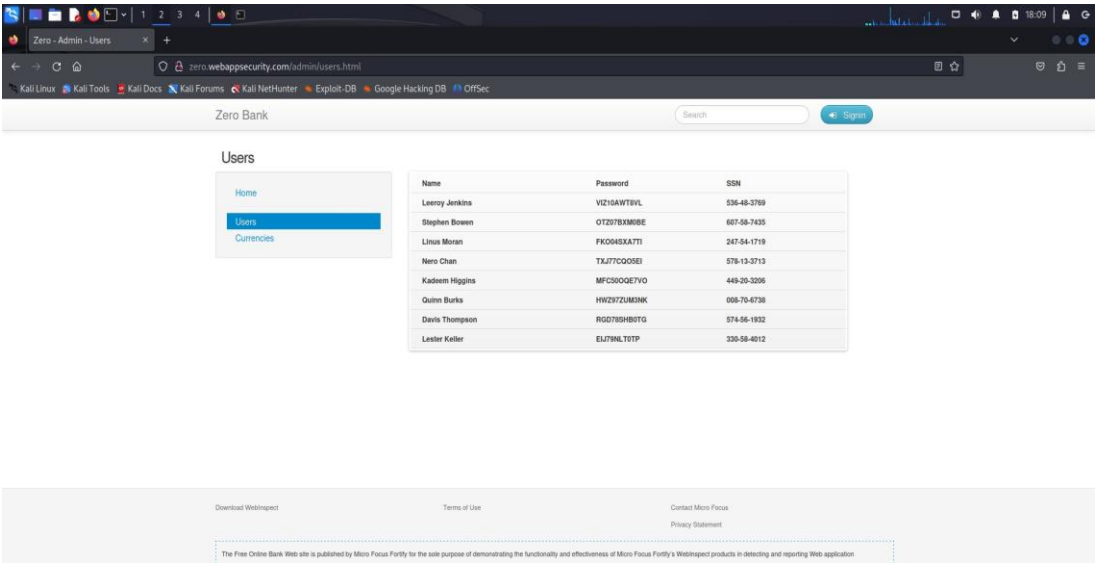


Figure 4: Insecure Design on password and PII

## Business Impact

Based on the attack's extent, the data it exposes, and how long it lasts, an insecure design vulnerability's business impact may vary (Sengupta, 2022). A successful attack could have several effects, including as total takeover, system and data breach, and user and system enumeration (Sengupta, 2022). Other possible outcomes include privilege escalation, denial of service (DOS), and the execution of additional attacks (Sengupta, 2022). Additional consequences of this could include monetary loss, data loss or destruction, fines and penalties, a decline in trust, and legal difficulties.

# Protection Solution

Some security techniques to mitigate insecure design are as follows.

- (a) **Threat Modelling:** This has to do with considering all the possible threats the web application would face, what the attack surface looks like, and how it can be prevented (Pant, 2022). This helps to identify possible threats and understand how and where attack vectors might exist. And as such anticipate circumstances where the web application might fail and thus plan to fail in a managed manner (Pant, 2022). A threat modelling is done using a flow diagram as shown below highlighting how data is stored, processed, and flows as well as its interaction and trust boundaries (Pant, 2022).

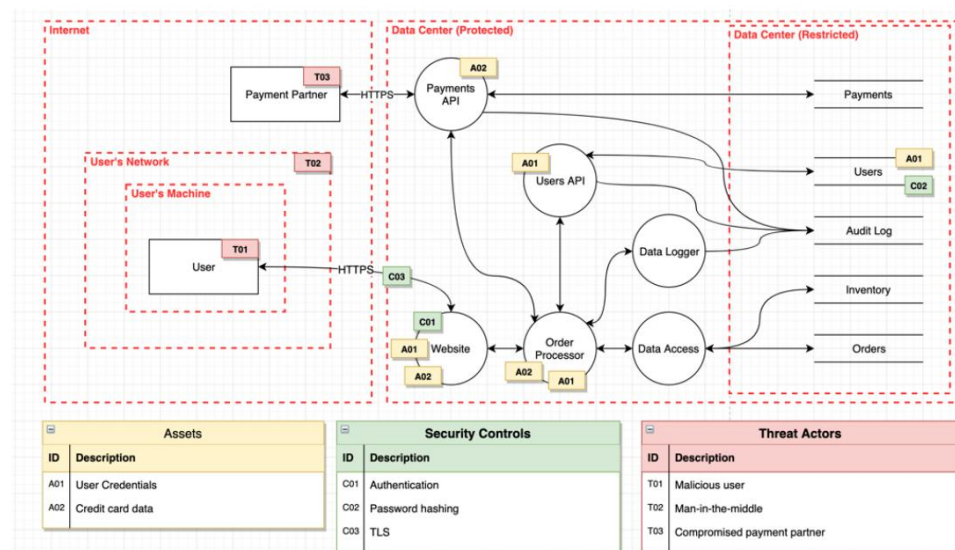


Figure 5: Threat Modelling Data Flow Diagram (Pant, 2022)

- (b) **Principle Of Least Privilege:** The principle of least privilege should be applied where each process, program, and user should only operate with the bare minimum of privilege required to perform their task. This would help to limit the amount of damage done when an attacker compromises the application (Pant, 2022).
- (c) **Encryption:** Data should be encrypted with a strong cryptographic function both in transit and at rest (Pant, 2022). This would help mitigate against man-in-the-middle attacks while providing an extra layer of protection (Pant, 2022).
- (d) **Untrusted By Default:** All user input should be untrusted by default and must be validated to ensure it meets the expected parameters (Pant, 2022). If it does not meet the expected parameters, then it should be rejected immediately (Pant, 2022).
- (e) **Fail Application Securely:** This means that when an application eventually fails, error messages need to be consistent and not disclose any internal architectural details (Pant, 2022). This is a common tactic used by attackers to try and create an error as many error message outputs disclose



internal details and mechanism of an application when an error occurs (Pant, 2022).

- (f) **Event Monitoring and Logging:** It is imperative to implement real-time monitoring and event logging into your application. This would help get an overview of the type, volume, and traffic reaching the application server and help distinguish between real and automated traffic generated by likely attackers (Pant, 2022). Thus, this would provide a prompt to protect your application whenever such automated traffic is detected (Pant, 2022).
- (g) **Learning from Mistakes:** We must learn from mistakes that eventually happen. When a vulnerability is detected in the production environment, it needs to be carefully analysed to find out what safeguards and processes were overlooked to ensure it does not happen again (Pant, 2022).
- (h) **Code Review:** Use secure design patterns and reference architecture style to conduct a thorough code review to stop faulty code from being released into production (Djeki, et al., 2022).
- (i) **Penetration Testing:** Prior to deployment, establish a red team to carry out threat modelling and penetration testing. To address found security vulnerabilities, audits and penetration tests must be conducted on a regular basis (Djeki, et al., 2022).
- (j) **Segregation:** Depending on their exposure and protection needs, layers on the system and network levels should be divided. Additionally, tenants must be firmly separated by design (OWASP, 2021).

## Recommendation

Considering all the precautions mentioned above, UnivBooks must incorporate pre-code tasks that are essential to the security by design tenets. To prevent known attack methods, the team must adopt a culture and methodology that continuously assesses threats and makes sure the code is thoroughly tested and designed. When modifying data flow, access control, or any other security measure, threat modelling must be incorporated into the pre-code procedure. The team should also determine the appropriate flow and failure states and making sure everyone is aware of and in agreement with them are equally crucial. To make sure they are correct and desired, assumptions and conditions for expected and failure flows should also be examined. Regretfully, secure design is neither a software tool nor an add-on. Nonetheless, a significant step towards reducing the risks connected with insecure design would be for all team members to internalise the culture and practice of secure by design.

## References

Aljabri, M. et al., 2022. *Testing and Exploiting Tools to Improve OWASP Top Ten Security Vulnerabilities Detection*. Saudi Arabia, IEEE.

Alvarenga, G., 2022. *SHIFT LEFT SECURITY EXPLAINED*. [Online]  
Available at: <https://www.crowdstrike.com/cybersecurity-101/shift-left-security/>

Cunha, J. P., 2022. Cybersecurity Threats for a Web Development. *Advanced Research on Information Systems Security*.

Djeki, E., Degila, J., Bondiombouy, C. & Alhassan, M. H., 2022. *Preventive Measures for Digital Learning Spaces' Security Issues*. Turkey, IEEE.

OWASP, 2021. *A04:2021 – Insecure Design*. [Online]  
Available at: [https://owasp.org/Top10/A04\\_2021-Insecure\\_Design/](https://owasp.org/Top10/A04_2021-Insecure_Design/)

OWASP, 2021. *OWASP Top 10:2021*. [Online]  
Available at: <https://owasp.org/Top10/>

Pant, P., 2022. *Secure web development*. s.l.:s.n.

Petranović, T. & Žarić, N., 2023. *Effectiveness of Using OWASP TOP 10 as AppSec Standard*. Montenegro, IEEE.

Sengupta, S., 2022. *Insecure Design*. [Online]  
Available at: <https://crashtest-security.com/insecure-design-vulnerability/>