Frozen Yoghurt Ltd

Penetration Testing Report 1

UP2009045

Table of Contents

[Executive Summary 3](#_Toc101788135)

[Summary of Results 3](#_Toc101788136)

[Attack Narrative (Nmap 7.91) 3](#_Toc101788137)

[WordPress Enumeration (wpscan v.) 5](#_Toc101788138)

[Directory Scanning (Gobuster v3.0.1) 6](#_Toc101788139)

[Google Maps Plugin Exploit 8](#_Toc101788140)

[WordPress Shell Upload 9](#_Toc101788141)

[Accessing admin account 9](#_Toc101788142)

[Privilege Escalation 10](#_Toc101788143)

[Conclusion 11](#_Toc101788144)

[Recommendations 12](#_Toc101788145)

[Appendices 13](#_Toc101788146)

[Code Listings 13](#_Toc101788147)

[Vulnerabilities 13](#_Toc101788148)

[Figures 13](#_Toc101788149)

[References 15](#_Toc101788150)

# Executive Summary

I have been contracted by Frozen Yoghurt Ltd to conduct penetration testing on two web servers with the purpose of revealing any vulnerabilities. Actions conducted to collect these results were done so with the aim to replicate a malicious attack on Frozen Yoghurt Ltd. The goals of this investigation include:

* Figuring out if and how the attacker can penetrate Frozen Yoghurt Ltd’s network.
* Impact of the security breach, i.e., confidentiality and availability of company data

With the focus of this being identification of vulnerabilities and exploitable weaknesses, all attacks were made as if an adversary, i.e., someone who does not have access to the network, were to try to attack the client network. Also find that investigation of Frozen Yoghurt Ltd’s web servers were performed within a controlled environment, thus, to limit risk.

## Summary of Results

Reconnaissance data revealed multiple open ports on the system, one being an Apache web server on port 60080. Further investigation found this web server was running WordPress and the enumeration stage, found several usernames and an out of data Google Maps plugin. Directory scanning revealed /security\_wp and /backup containing data used to exploit the server.

The database prefix found in /security\_wp was used in the “wp\_google\_maps\_sqli” exploit to retrieve password hashes from the WordPress server and resulted in two passwords being extracted. These passwords were then cracked, and the tom account was used to upload a shell to the web server using the “wp\_admin\_shell\_upload” exploit.

After uploading the shell to the server, the administrator password was found in /home/administrator/.automate. Navigating to ~/etc and using the “sudo su” command, it was possible to gain root access.

# Attack Narrative (Nmap 7.91)

Frozen Yoghurt Ltd provided a single IP address with no further information, this was to ensure the actions conducted during the investigation strictly imitated those of an adversary with no other information of the internal network or its systems.

Graphical user interface, text

Description automatically generated

Figure 1 – Open Ports on System

To get a scope of the network I ran a port scan on the network, Figure 1, this revealed open ports including 60080 with an Apache server running on it. Further investigation of this port revealed apache was running WordPress.

## WordPress Enumeration (wpscan v.)

Text

Description automatically generated

Figure 2 - Extracted Users

Text

Description automatically generated

Figure 3 - Vulnerable Google-Maps Plugin

During this enumeration process, three usernames were found, Figure 2: admin, webmaster, and tom. The plugin “wp-google-maps” was also discovered, Figure 3, and was found to be out of date leaving it vulnerable to exploit. (*NVD - CVE-2019-10692*, n.d.) Details how versions of this plugin prior to 7.11.18 are susceptible to remote SQL injection.

## Directory Scanning (Gobuster v3.0.1)

Text

Description automatically generated

Figure 4 - Webserver Directories

Scanning the directories of the server, /security\_wp and /backup was found, Figure 4.

Graphical user interface

Description automatically generated with low confidence

Figure 5 - Screenshot displaying database prefix

Within /security\_wp an uploaded screenshot taken within phpMyAdmin, Figure 5, confirms the client’s webserver is running MySQL and has multiple databases setup with the prefix changed from wp\_ to vt\_.

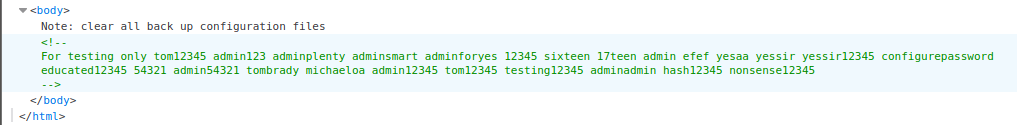


Figure 6 – List of potential passwords

Figure 6 displays what may be potential user passwords for the system.

## Google Maps Plugin Exploit

Text

Description automatically generated

Figure 7 - Settings used for exploit

Figure 7 displays the settings used for the remote SQL injection exploit using the out of date google maps plugin. Figure 5 supplied DB\_PREFIX for this exploit.

Text

Description automatically generated

Figure 8 - Extracted password hashes

Password hashes for the users of WordPress were found, Figure 8. These passwords were then decrypted producing the following:

* Username: tom Password: 1+1=windowtomy!
* Username: admin Password: qwerty1235

## WordPress Shell Upload

Text

Description automatically generated

Figure 9 - Settings used for shell upload

As tom is admin, “wp\_admin\_shell\_upload” was used to remotely upload a plugin containing a shell payload to the WordPress website, Figure 9.

Text

Description automatically generated

Figure 10 – Uploaded Shell

The payload shown in Figure 9 supplied access to the server, Figure 10, allowing access to system files.

## Accessing admin account

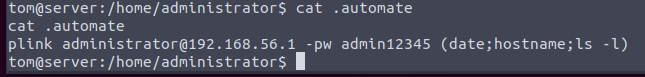


Figure 11 - .automate script displaying administrator password

The tom account allowed for navigation to the administrator home folder where the “. automate” file is found. This file, when read, holds the administrator password, shown in Figure 11 to be admin12345.



Figure 12 – Successful login to admin account

This password allowed for login to the administrator account, Figure 12, successfully escalating the privileges within the system.

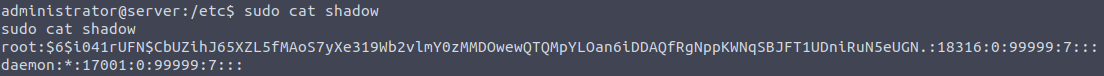


Figure 13 – Root hash for login

Using administrator account privileges to read the ~/etc/shadow file which holds all usernames and password hashes of the system. Shown in Figure 13, the root hash for login is found here.

## Privilege Escalation

Text

Description automatically generated with medium confidence

Figure 14 – Logging into root environment

Text

Description automatically generated

Figure 15 - Credit Card File

Sudo su command gives access to the root environment, Figure 14, through the administrator user allowing for folders such as /root to be accessed, Figure 15.

# Conclusion

Frozen Yoghurt Ltd experienced multiple instances where their system failed, and sensitive information was accessed. This led to complete root access of the system. The out-of-date led to the overall takeover of the system, had this plugin been updated, this may not have been the case. Also found during the exploitation process were insecure passwords for all users of the system including the administrative password which was accessible by a standard user on the system.

* Figuring out if and how the attacker can penetrate Frozen Yoghurt Ltd’s network.
* Impact of the security breach, i.e., confidentiality and availability of company data

Original tasks set out for this investigation, repeated above, were completed with Frozen Yoghurt Ltd’s network being fully exploited discovering the potential impact of a potential security breach on the system. In the event of a security breach, an adversary would have access to usernames, passwords, and potentially full root access to the system which could lead to credit card details being breached.

## Recommendations

The following recommendations are to be made due to the potential impact found from this investigation:

1. Update Scheduling

Yoghurt Ltd’s webserver was exploited via the out of date google maps plugin. This plugin should be updated to its latest version and an update schedule should be established to ensure the webserver is not vulnerable.

1. Password Policy Implantation

Found passwords were easily cracked due to insufficient complexity. NIST recommended password guidelines, (Poza, 2021), should be followed and a new company password policy should be created.

1. User Group Policies

A proper user group policy should be implemented to prevent users who should not have access to certain files such as the creditcard.zip file.

1. Limit Accessible Website Directories and/or removal of sensitive data

/security\_wp and /backup directories should be moderated, limiting their visibility with removal of sensitive data such as passwords and logs within an administrator account.

# Appendices

## Code Listings



Figure 16 - Code used to crack WordPress hashes

## Vulnerabilities

|  |  |  |
| --- | --- | --- |
| Vulnerability | Explanation | CVE or CWE |
| Google SQLI | The out of date google maps plugin allows for SQL Injection to retrieve password hashes from the webserver. These can later be cracked to give adversaries access to the server. | (*NVD - CVE-2019-10692*, n.d.) |
| Admin Shell Upload | This upload vulnerability allows a plugin to be uploaded with a shell payload using the WordPress admin user credentials. | (*NVD - CVE-2018-14028*, n.d.) |
| Privilege Escalation | The webserver is vulnerable to privilege escalation where an administrative user can use their own password to gain super user privileges. | (*CWE - CWE-264: Permissions, Privileges, and Access Controls (4.6)*, n.d.) |

## Figures

[Figure 1 – Open Ports on System 4](#_Toc101788119)

[Figure 2 - Extracted Users 5](#_Toc101788120)

[Figure 3 - Vulnerable Google-Maps Plugin 5](#_Toc101788121)

[Figure 4 - Webserver Directories 6](#_Toc101788122)

[Figure 5 - Screenshot displaying database prefix 6](#_Toc101788123)

[Figure 6 – List of potential passwords 7](#_Toc101788124)

[Figure 7 - Settings used for exploit 8](#_Toc101788125)

[Figure 8 - Extracted password hashes 8](#_Toc101788126)

[Figure 9 - Settings used for shell upload 9](#_Toc101788127)

[Figure 10 – Uploaded Shell 9](#_Toc101788128)

[Figure 11 - .automate script displaying administrator password 9](#_Toc101788129)

[Figure 12 – Successful login to admin account 10](#_Toc101788130)

[Figure 13 – Root hash for login 10](#_Toc101788131)

[Figure 14 – Logging into root environment 10](#_Toc101788132)

[Figure 15 - Credit Card File 10](#_Toc101788133)

[Figure 16 - Code used to crack WordPress hashes 13](#_Toc101788134)

# References

*CWE - CWE-264: Permissions, Privileges, and Access Controls (4.6)*. (n.d.). Retrieved April 25, 2022, from https://cwe.mitre.org/data/definitions/264.html

*NVD - CVE-2018-14028*. (n.d.). Retrieved April 25, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2018-14028

*NVD - CVE-2019-10692*. (n.d.). Retrieved April 21, 2022, from https://nvd.nist.gov/vuln/detail/CVE-2019-10692

Poza, D. (2021). *NIST Password Guidelines and Best Practices for 2020*. Januari. https://auth0.com/blog/dont-pass-on-the-new-nist-password-guidelines/