The room invites you a challenge to investigate a series of traffic data and stop malicious activity under different scenarios. Let's start working with Zeek to analyse the captured traffic.

We recommend completing the [**Zeek**](https://tryhackme.com/room/zeekbro)room first, which will teach you how to use the tool in depth.

**An alert triggered:**"Anomalous DNS Activity".

The case was assigned to you. Inspect the PCAP and retrieve the artefacts to confirm this alert is a true positive.

Answer the questions below

Qus: Investigate the **dns-tunneling.pcap** file. Investigate the **dns.log** file. What is the number of DNS records linked to the IPv6 address?

Sol:

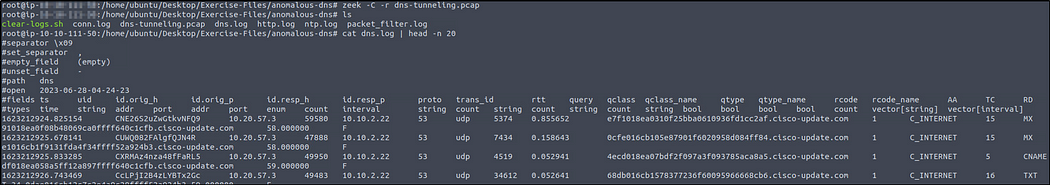
DNS “AAAA” records store IPV6 addresses. Note that there are other DNS record types that handle different purposes, such as the “A” record for IPv4 addresses, “CNAME” for canonical names, “MX” for mail exchange servers, PTR record for reverse DNS lookups, and the TXT record allows for storing textual information associated with a domain.

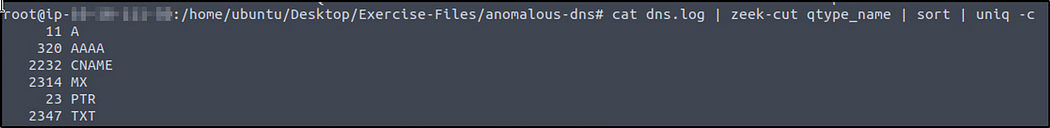
zeek -C -r dns-tunneling.pcap  
cat dns.log | zeek-cut qtype\_name | sort | uniq -c

1. **zeek -C -r dns-tunneling.pcap**:
   * **zeek**: This is the command to invoke Zeek.
   * **-C**: This option tells Zeek to load its configuration files from the current directory.
   * **-r dns-tunneling.pcap**: This specifies the input file for Zeek to analyze. In this case, it's a PCAP (Packet Capture) file named **dns-tunneling.pcap**, which likely contains network traffic data.

So, this command runs Zeek on the **dns-tunneling.pcap** file, analyzing the network traffic captured in the PCAP file.

1. **cat dns.log | zeek-cut qtype\_name | sort | uniq -c**:
   * **cat dns.log**: This command reads the contents of the file **dns.log**.
   * **|**: This symbol is a pipe, which takes the output of the command on its left and uses it as the input for the command on its right.
   * **zeek-cut qtype\_name**: This command is likely using the Zeek tool **zeek-cut** to extract the **qtype\_name** field from the input.
   * **sort**: This command sorts the output alphabetically.
   * **uniq -c**: This command filters out adjacent matching lines and counts the number of occurrences of each unique line.





**Qus: Investigate the conn.log file. What is the longest connection duration?**

**Answer: 9.420791**

cat conn.log | zeek-cut duration | sort -r | head -n 1

1. **cat conn.log**: This command reads the contents of the file **conn.log**. This file typically contains information about connections observed by Zeek.
2. **|**: The pipe symbol takes the output of the command on its left and passes it as input to the command on its right.
3. **zeek-cut duration**: This command uses the Zeek tool **zeek-cut** to extract the **duration** field from the input. The **duration** field in a connection log represents the duration of the connection.
4. **sort -r**: This command sorts the extracted **duration** values in reverse order (**-r** flag), putting the longest durations first.
5. **head -n 1**: This command selects only the first line from the sorted output. Since the output is sorted in descending order of duration, this will give us the connection with the longest duration.



**Investigate the dns.log file. Filter all unique DNS queries. What is the number of unique domain queries?**

**Answer: 6**

**Question Hint**

You need to use the DNS query values for summarising and counting the number of unique domains.

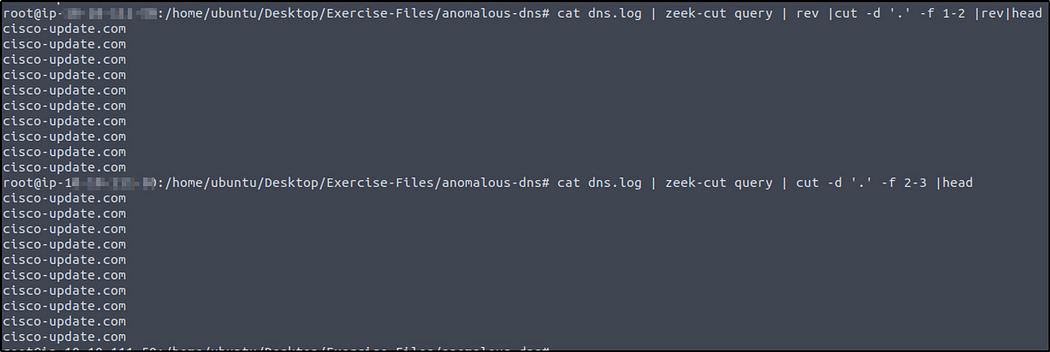
There are lots of “***.cisco-update.com” DNS queries, you need to filter the main address and find out the rest of the queries that don’t contain the “***.cisco-update.com” pattern.

You can filter the main “\*\*\*.cisco-update.com” DNS pattern as “[cisco-update.com](http://cisco-update.com/)” with the following command; “cat dns.log | zeek-cut query |rev | cut -d ‘.’ -f 1–2 | rev | head

The hint can be quite confusing. Basically, what we want is to extract the last two fields of the unique lines/domain queries like for example, “example.com”.

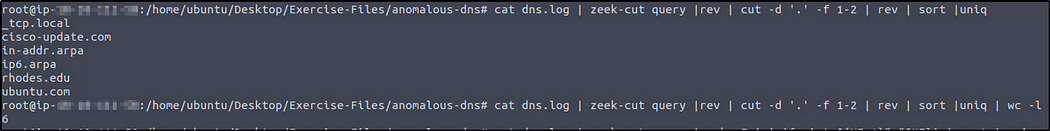
cat dns.log | zeek-cut query |rev | cut -d '.' -f 1-2 | rev | head

The reverse “rev” is used to reverse the line characters and then use the “cut” command to display the first and second field value.



We will add the “sort” and “uniq” command to avoid the duplication of values, and then “wc -l” to print the newlines count.

cat dns.log | zeek-cut query |rev | cut -d '.' -f 1-2 | rev | sort |uniq | wc -l

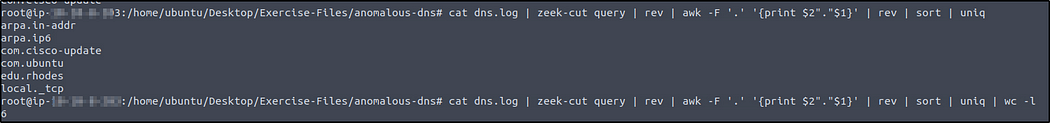


Are there other ways to get the same output”? Yes there are.

First set of commands to try.

cat dns.log | zeek-cut query | rev | awk -F '.' '{print $2"."$1}' | rev | sort |uniq  
cat dns.log | zeek-cut query | rev | awk -F '.' '{print $2"."$1}' | rev | sort |uniq | wc -l

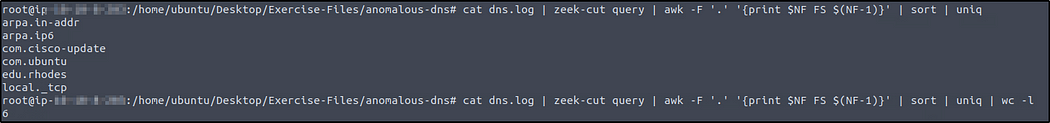
The commands now use awk with the -F option to specify the delimiter as a dot (.). Then, they print the desired fields in the required order. The rest of the pipeline remains the same, including the cat command to read the contents of dns.log, zeek-cut to extract the "query" field.



The second set commands to try is without the “rev” command.

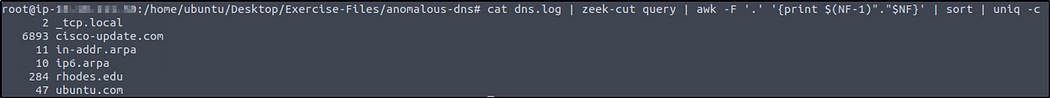
cat dns.log | zeek-cut query | awk -F '.' '{print $NF FS $(NF-1)}' | sort | uniq  
cat dns.log | zeek-cut query | awk -F '.' '{print $NF FS $(NF-1)}' | sort | uniq | wc -l

In the commands, awk is used with the -F option to set the field separator as a dot (.). The desired fields are printed in the required order by referencing the last field ($NF) and the second-to-last field ($(NF-1)). The FS variable represents the field separator and is used to reassemble the fields in the desired format.



If we want to know how many queries being made, we can modify one of the commands above by just adding “-c” to “uniq” command.

cat dns.log | zeek-cut query | awk -F '.' '{print $(NF-1)"."$NF}' | sort | uniq -c



**There are a massive amount of DNS queries sent to the same domain. This is abnormal. Let’s find out which hosts are involved in this activity. Investigate the conn.log file. What is the IP address of the source host?**

**Answer: 10.20.57.3**

cat conn.log | zeek-cut id.orig\_h | sort | uniq -c



**An alert triggered:**"Phishing Attempt".

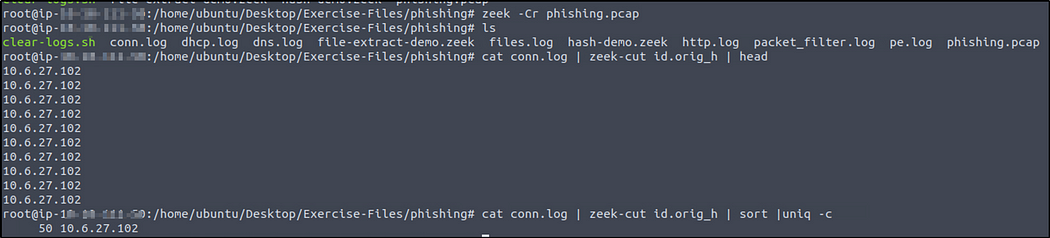
The case was assigned to you. Inspect the PCAP and retrieve the artefacts to confirm this alert is a true positive.

Answer the questions below

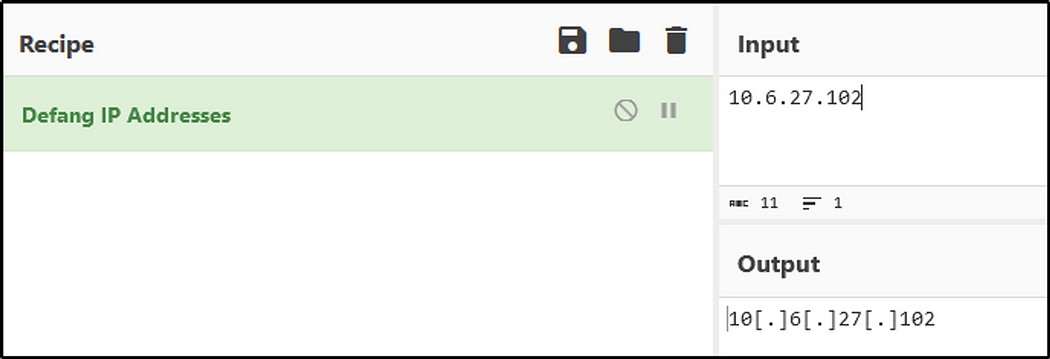
**Investigate the logs. What is the suspicious source address? Enter your answer in defanged format.**

**Answer: 10[.]6[.]27[.]102**

zeek -Cr phishing.pcap  
cat conn.log | zeek-cut id.orig\_h | sort | uniq -c



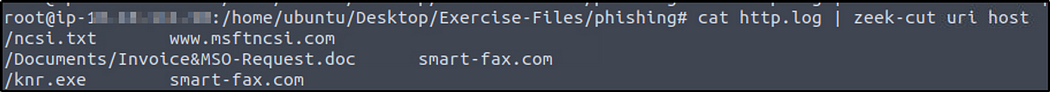
We see there’s only one source IP address. Use CyberChef to defang the IP address.



**Investigate the http.log file. Which domain address were the malicious files downloaded from? Enter your answer in defanged format.**

**Answer: smart-fax[.]com**

cat http.log | zeek-cut uri host



**Investigate the malicious document in VirusTotal. What kind of file is associated with the malicious document?**

**Answer: VBA**

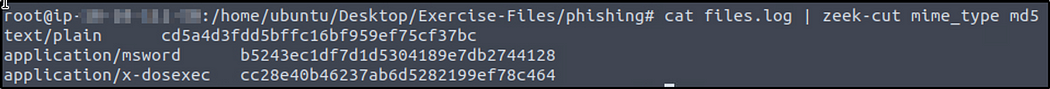
First, we must get the files’ md5 hash value. We will use the script provided.

zeek -Cr phishing.pcap hash-demo.zeek

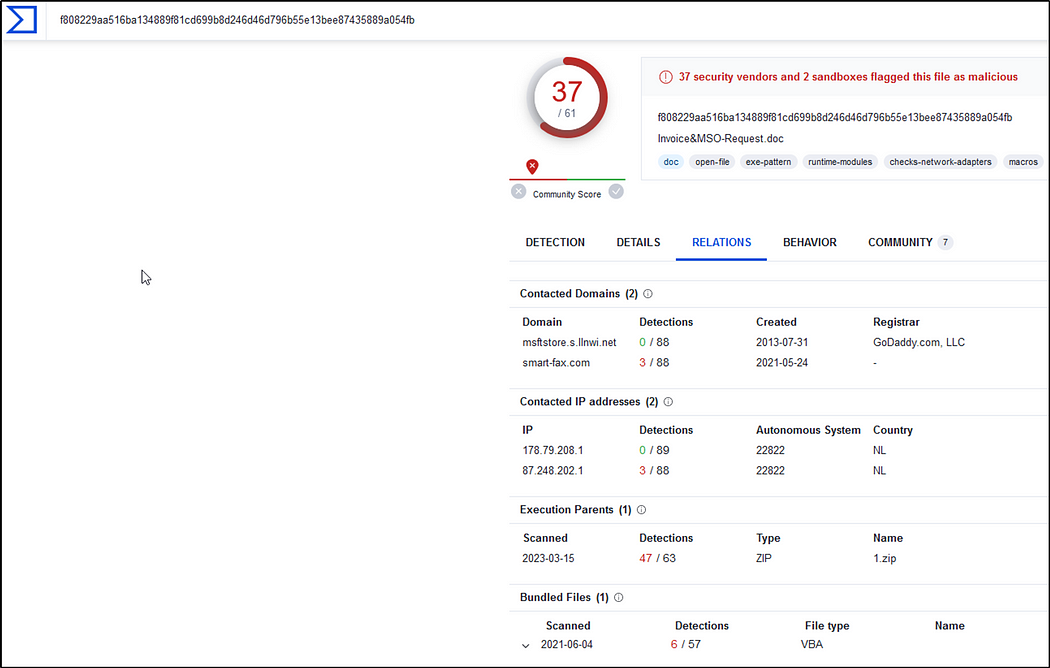
The task is easier because there’s only three files, but it wouldn’t be the case if there are hundred or thousand of files.

We will just select two field names.

cat files.log | zeek-cut mime\_type md5



We will select the second md5 value then go to VirusTotal and paste it in there.



Under the “Relations” tab is the file type for the malicious document.

**Investigate the extracted malicious .exe file. What is the given file name in Virustotal?**

**Answer: PleaseWaitWindow.exe**

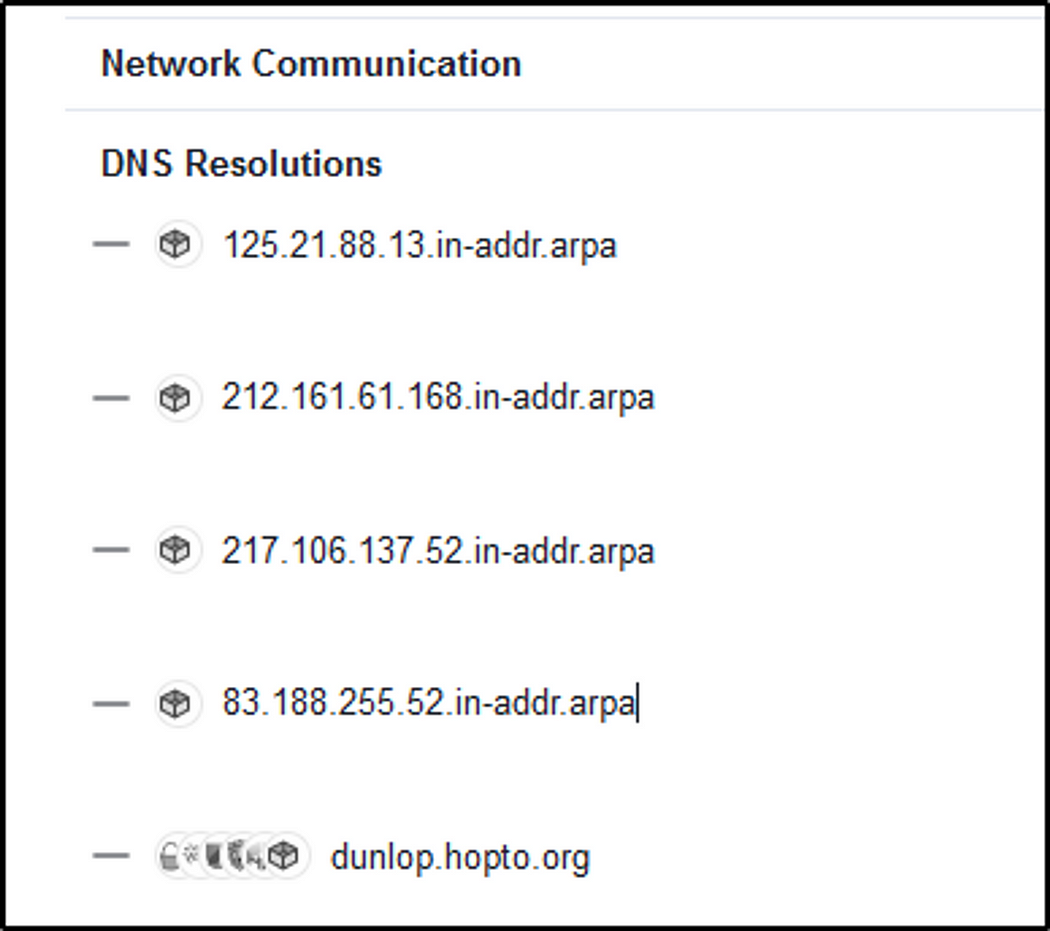
We will select the third md5 value then go to VirusTotal.

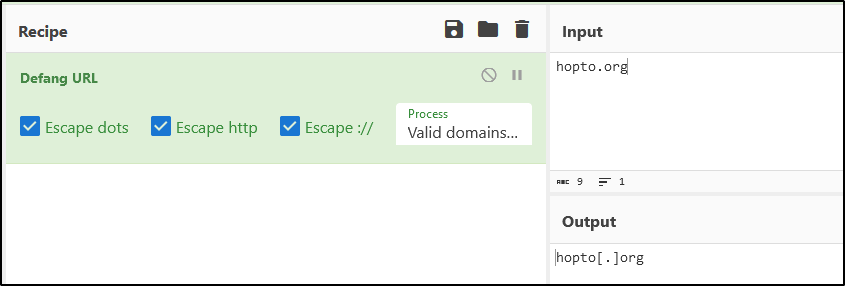


**Investigate the malicious .exe file in VirusTotal. What is the contacted domain name? Enter your answer in defanged format.**

**Answer: hopto[.]org**

Go to “Behavior” tab.





**Investigate the http.log file. What is the request name of the downloaded malicious .exe file?**

**Answer: knr.exe**

We found the answer when doing the first question.

**An alert triggered:**"Log4J Exploitation Attempt".

The case was assigned to you. Inspect the PCAP and retrieve the artefacts to confirm this alert is a true positive.

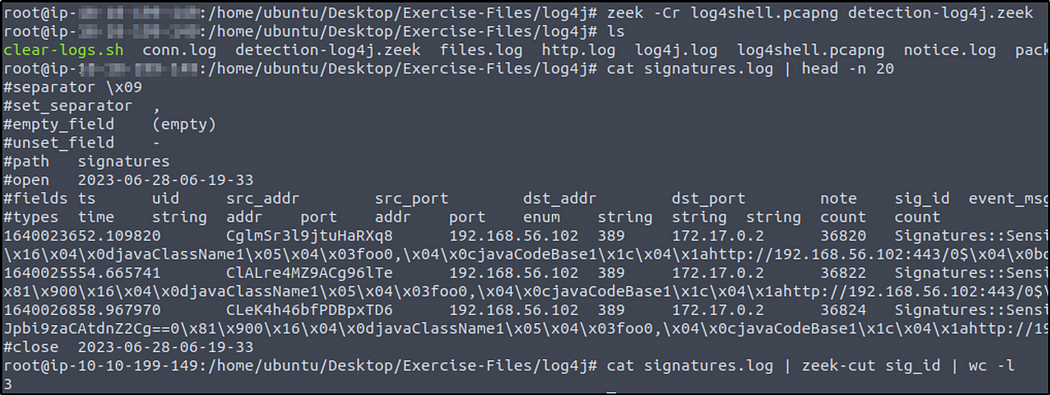
Answer the questions below

**Investigate the log4shell.pcapng file with detection-log4j.zeek script. Investigate the signature.log file. What is the number of signature hits?**

**Answer: 3**

zeek -Cr log4shell.pcapng detection-log4j.zeek  
cat signatures.log | zeek-cut sig\_id | wc -l

We will select the “sig\_id” field name.

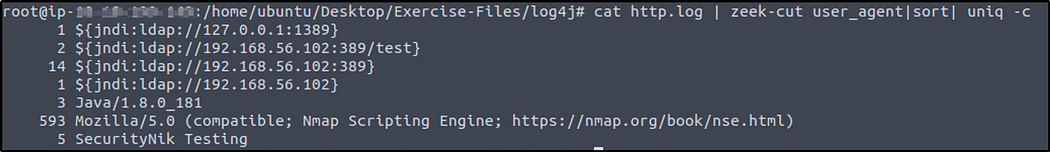


**Investigate the http.log file. Which tool is used for scanning?**

**Answer: Nmap**

cat http.log | zeek-cut user\_agent|sort| uniq -c

The information can be found in the field “user\_agent”.

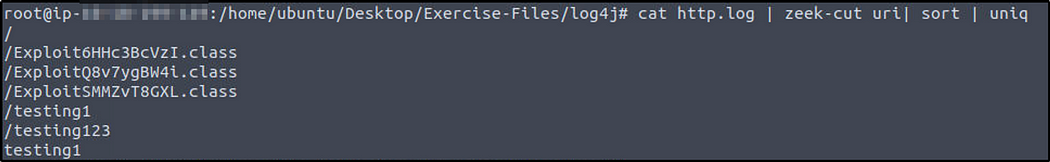


**Investigate the http.log file. What is the extension of the exploit file?**

**Answer: .class**

cat http.log | zeek-cut uri| sort | uniq

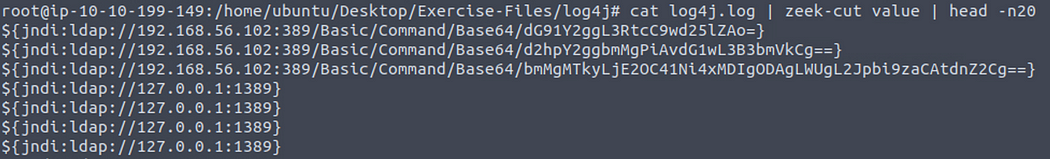
“uri” field contains the names of files downloaded with their extensions.



**Investigate the log4j.log file. Decode the base64 commands. What is the name of the created file?**

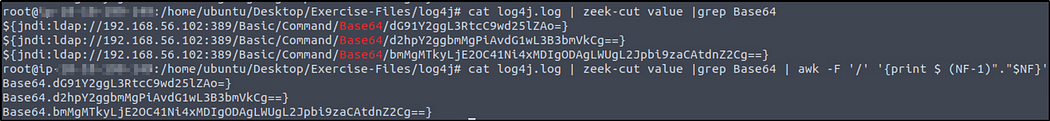
**Answer: pwned**

cat log4j.log | zeek-cut value | head -n20



We see that after the the path “/Basic/Command/Base64/”are base64 encoded values. What if there are other base64 encoded values? Let’s try to find all base64 encoded values.

cat log4j.log | zeek-cut value |grep Base64  
cat log4j.log | zeek-cut value |grep Base64 | awk -F '/' '{print $ (NF-1)"."$NF}'



Let’s copy the base64 strings and decode them. From the decoded output, we know the name of the created file.

