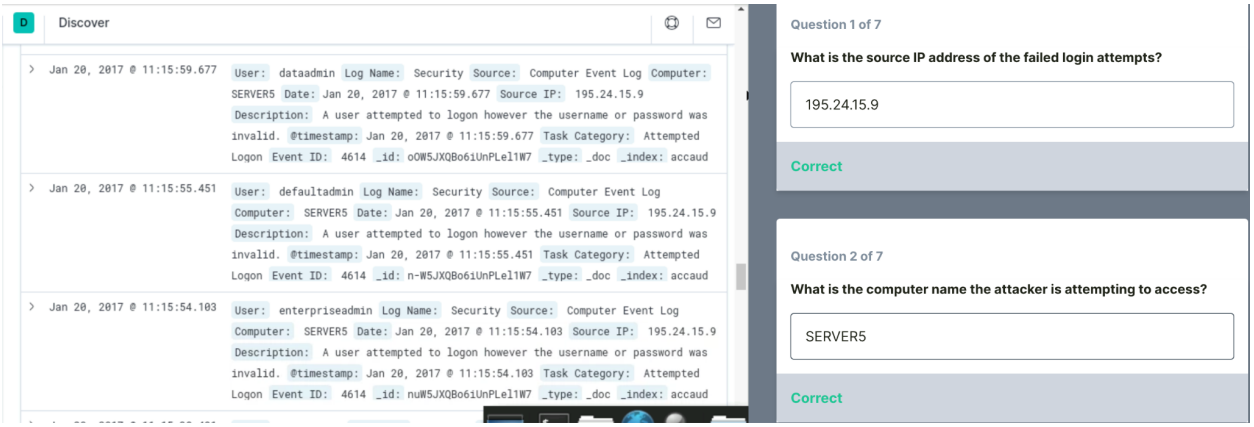
Amanda Gonzalez

Lab 3

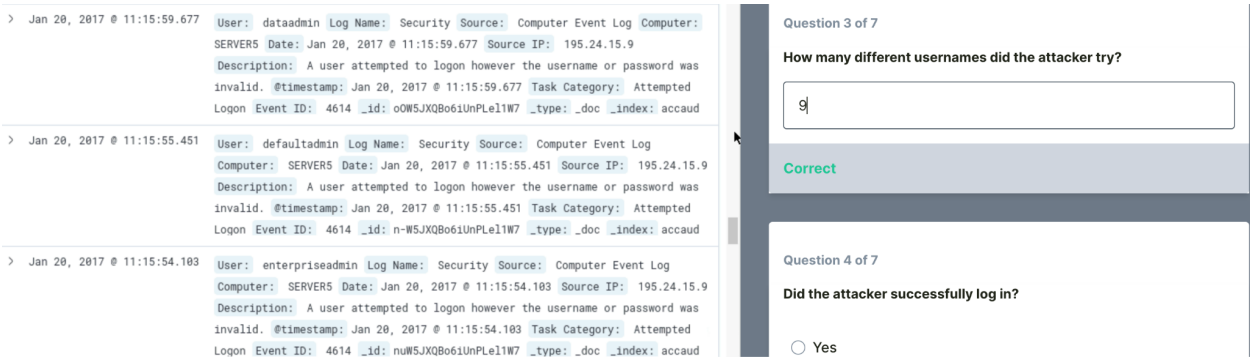
Accounting and Audit

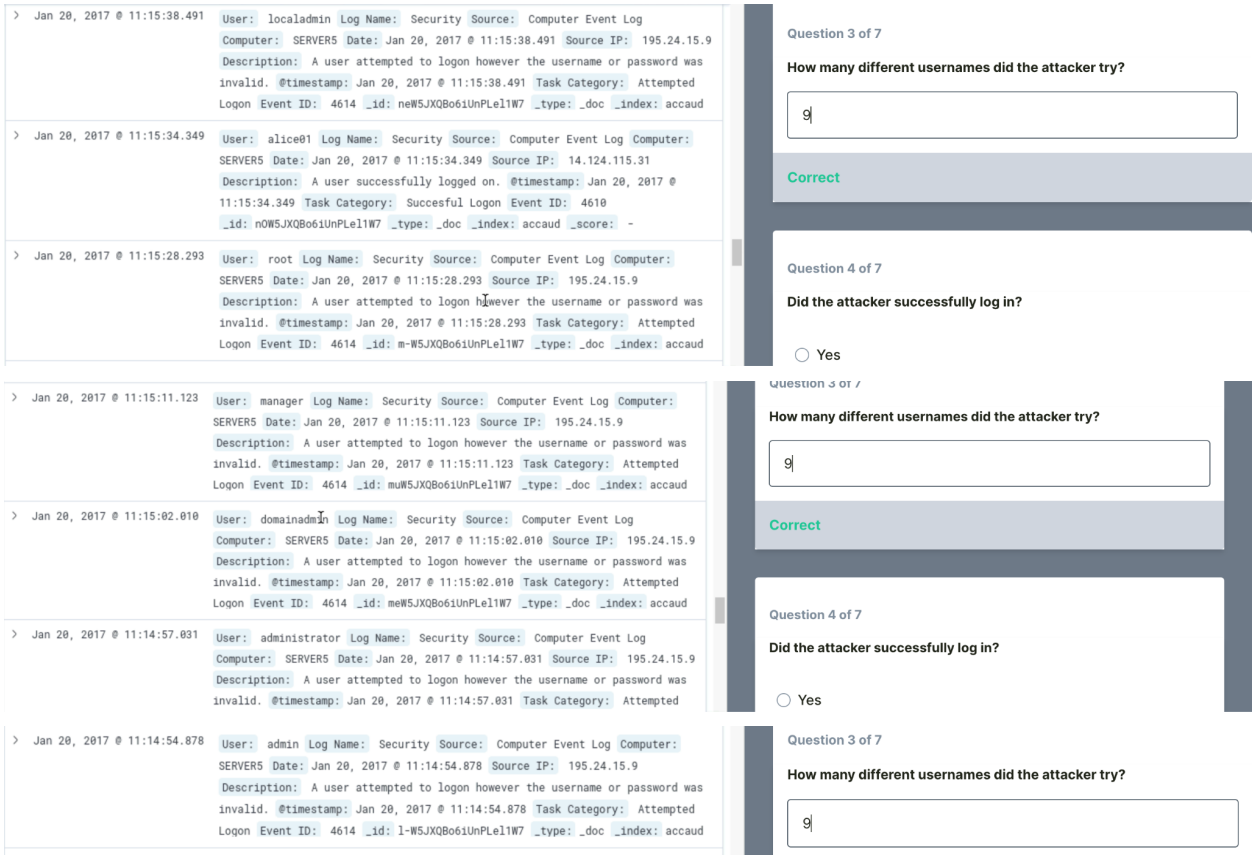
Question 1: As shown in the screenshot below, the source IP of the attacker is 195.21.15.9. You can tell they failed to log in because the description message states “A use attempted to longon however the username or password was invalid”.



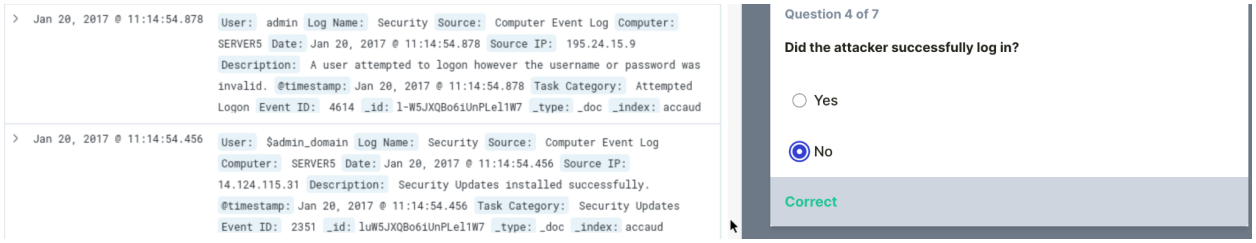
Question 2: As from the q1 screenshot above, the attacker is attempting to access SERVER5.

Question 3: As shown from the screenshots below, the attacker tried 9 usernames: dataadmin, defaultadmin, enterpriseadmin, localadmin, root, manager, domainadmin, administrator, and admin.

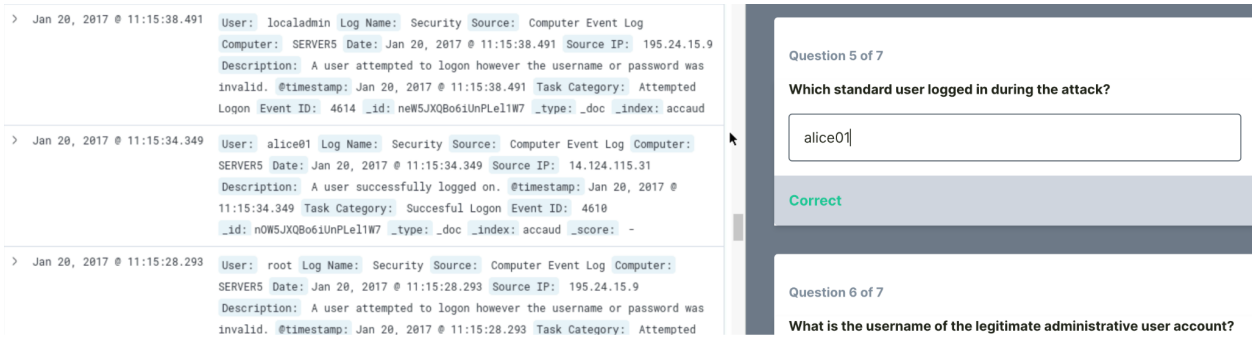




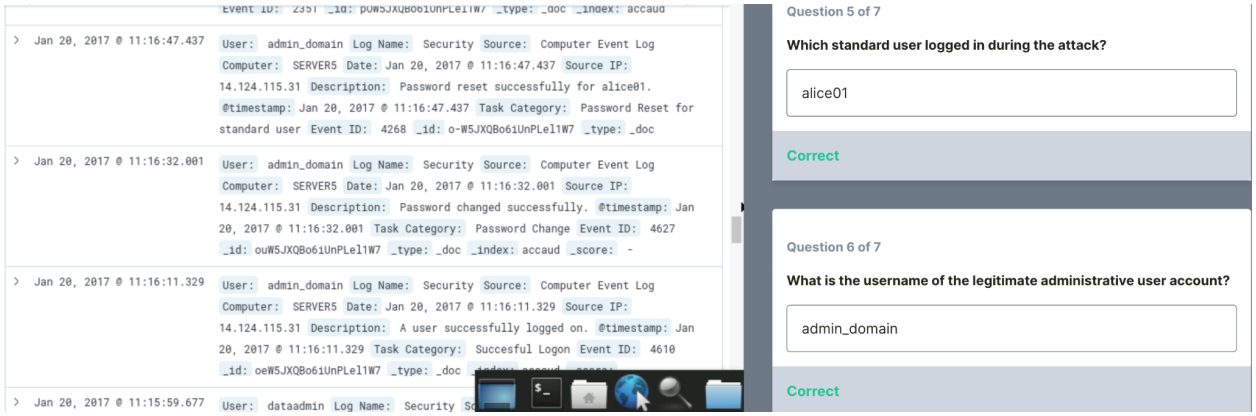
Question 4: From the attempts above, all prompted a “A use attempted to longon however the username or password was invalid” message because they did not match the username(s) with access, such as admin_domain, alice01, or $admin_domain, etc.



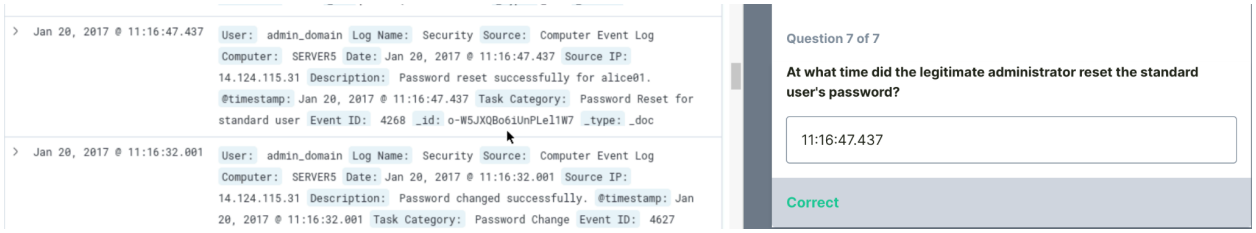
Question 5: As shown in the screenshot below, between the attacks from q3, user alice01 sucessfully logged on.



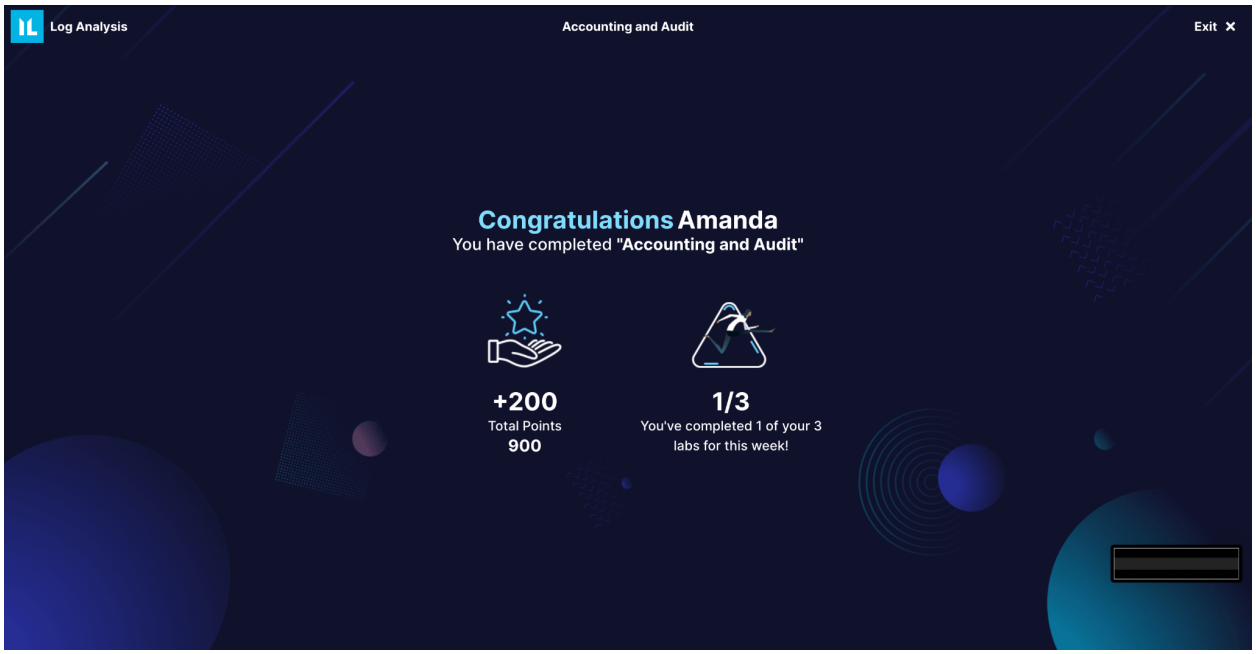
Question 6: As shown in the screenshot below, admin_domain is the administrative account, they can change passwords, such as shown below.



Question 7: As shown in screenshot below, the description states “Password reset successfully for alice01” from the user admin_domain, at 11:16:47.437.

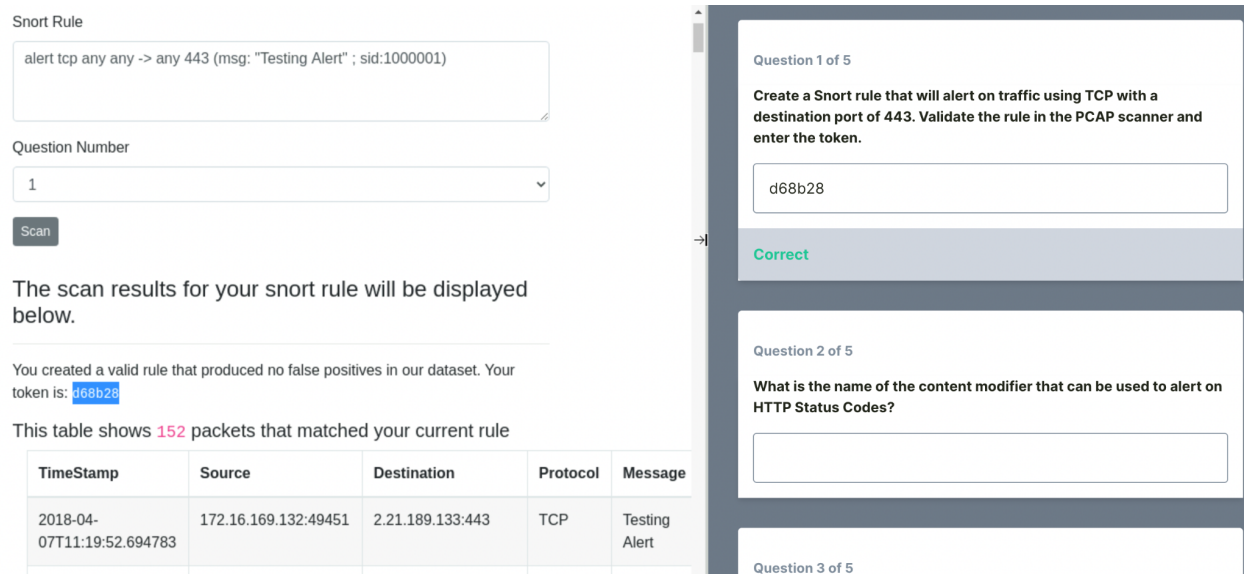


Results:

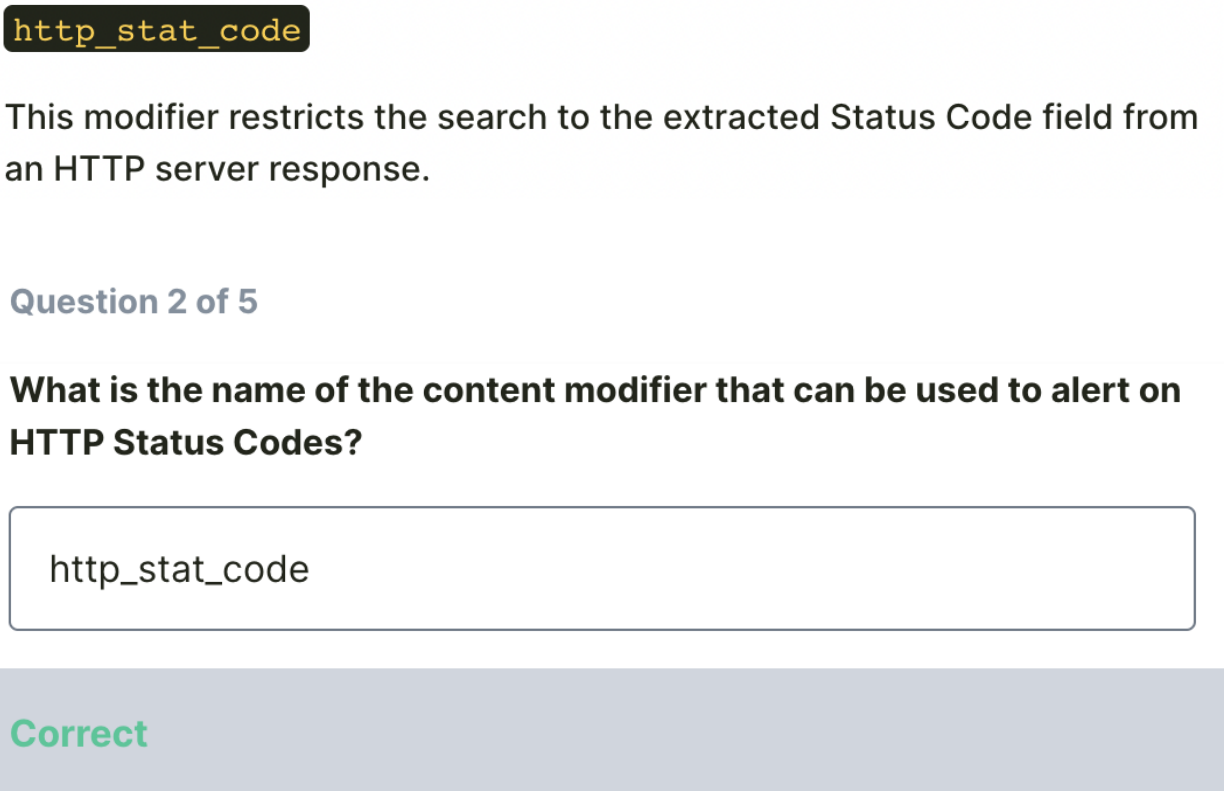


Snort Rules Ep.1

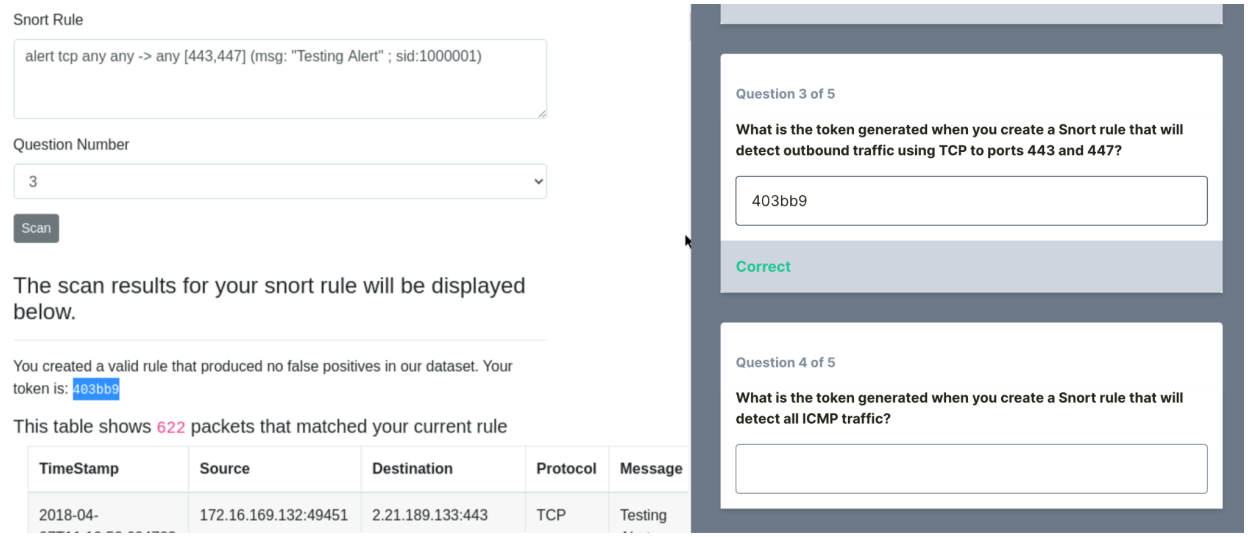
Question 1: As shown in screenshot below, the snort rule used to alert on traffic to port 443 was *alert tcp any any -> any 443 (msg: "Testing Alert": sid: 1000001)* this means that we are accepting from all sources and ports to all destinations that are on port 443. We get the token d68b28 as a result.



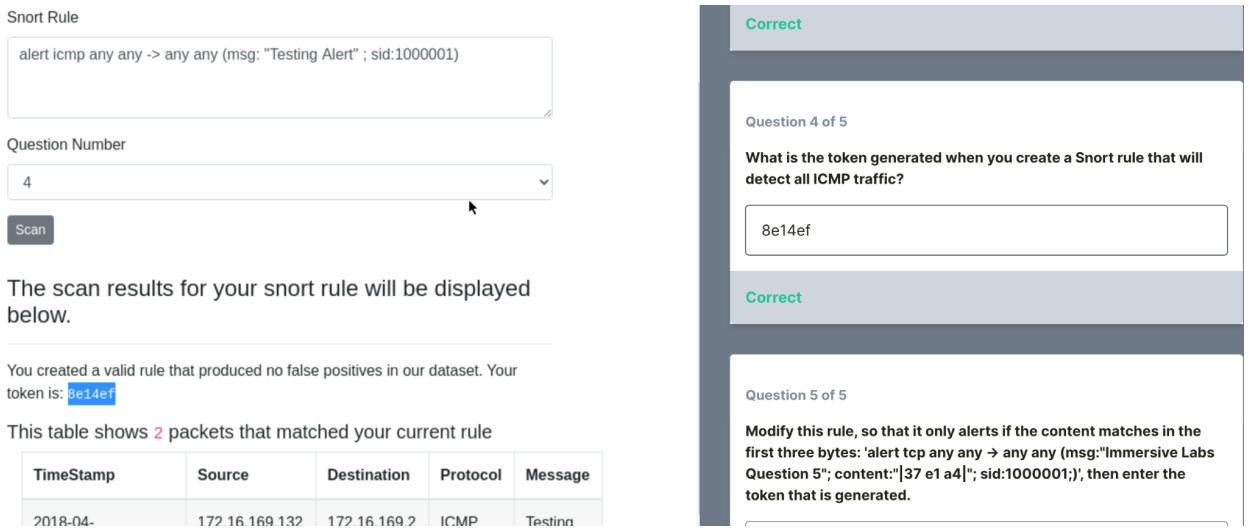
Question 2: As shown in the screenshots below, the modifier *http_stat_code* is used to alert on http.



Question 3: As shown in the screenshot below, the snort rule, *alert tcp any any -> any [443,447] (msg: "Testing Alert": sid: 1000001)* alerts traffic from any sources to any destinations on port 443 or 447 only. THe resulting token is 403bb9.



Question 4: As shown below, to detect all ICMP traffic, the snort rule, *alert icmp any any -> any any (msg: "Testing Alert": sid: 1000001).* All ICMP packets from all sources to all destinations are alerted. The result token is 8e14ef.



Question 5: As shown below, I modified the original message, *alert tcp any any -> any any (msg: "Immersive Labs Question 5"; content: "|37 e1 a4|"; sid: 1000001;)* to include a depth of 3 to specify matches in the first 3 bytes only as: *alert tcp any any -> any any (msg: "Immersive Labs Question 5"; content: "|37 e1 a4|"; depth: 3; sid: 1000001;).* The generated token was 31e9ba.

Snort Rule

alert tcp any any -> any any (msg: "Immersive Labs Question 5" ; content:"|37 e1 a4|"; depth:3; sid:1000001;)

Question Number

5

Scan

The scan results for your snort rule will be displayed below.

You created a valid rule that produced no false positives in our dataset. Your token is: 31e9ba

This table shows 1 packets that matched your current rule

TimeStamp	Source	Destination	Protocol	Message
2018-04-	202.169.44.149:80	172.16.169.132:49448	TCP	Immersive

Question 4 of 5

What is the token generated when you create a Snort rule that will detect all ICMP traffic?

8e14ef

Correct

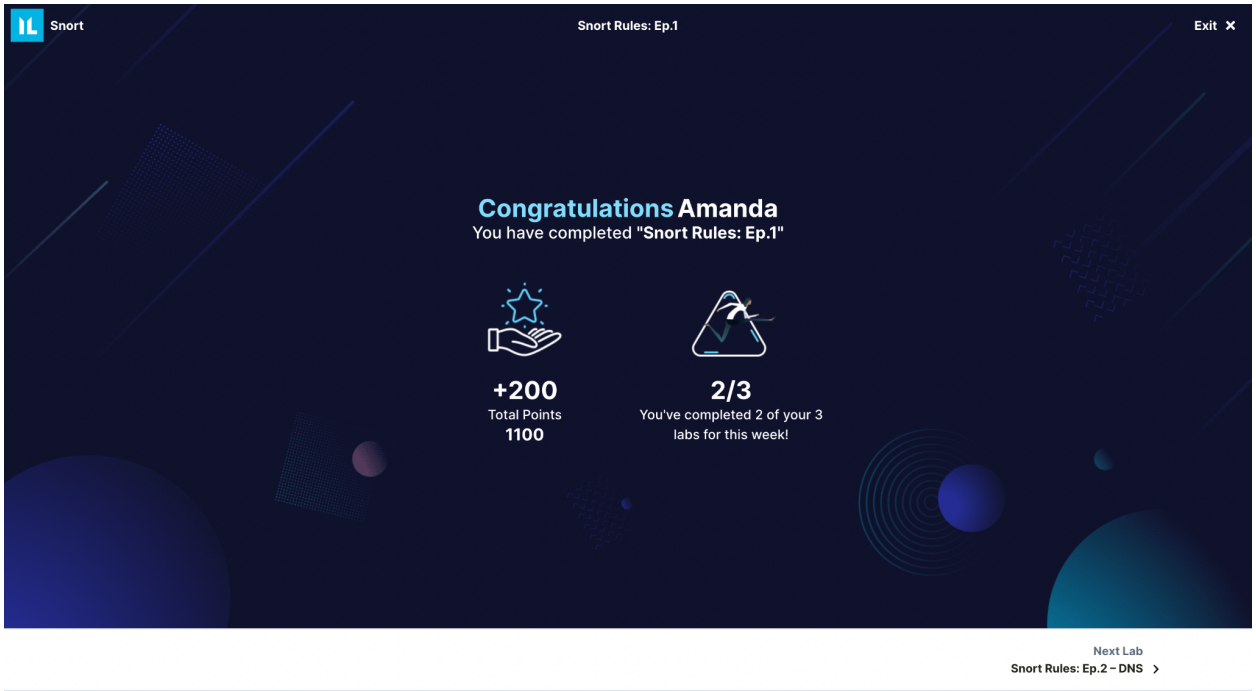
Question 5 of 5

Modify this rule, so that it only alerts if the content matches in the first three bytes: 'alert tcp any any → any any (msg:"Immersive Labs Question 5"; content:"|37 e1 a4|"; sid:1000001;)', then enter the token that is generated.

31e9ba

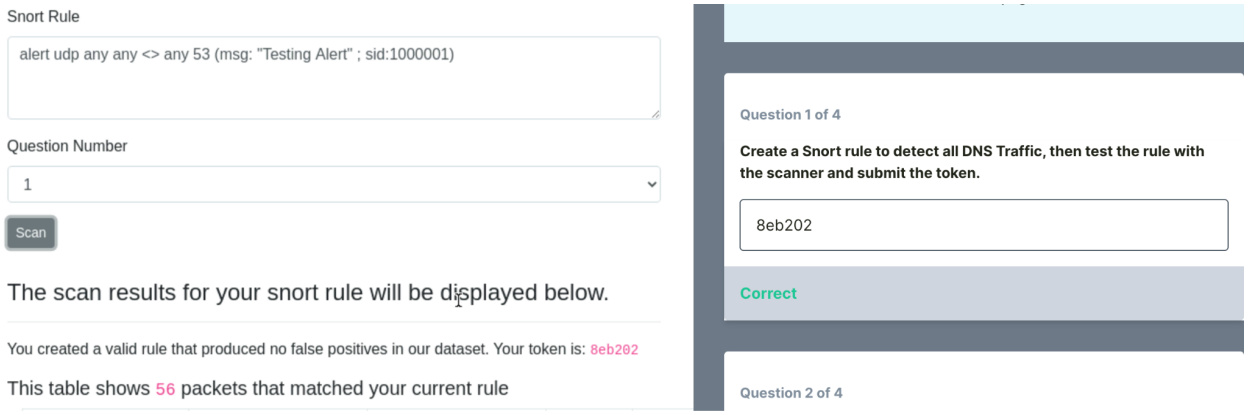
Correct

Result:

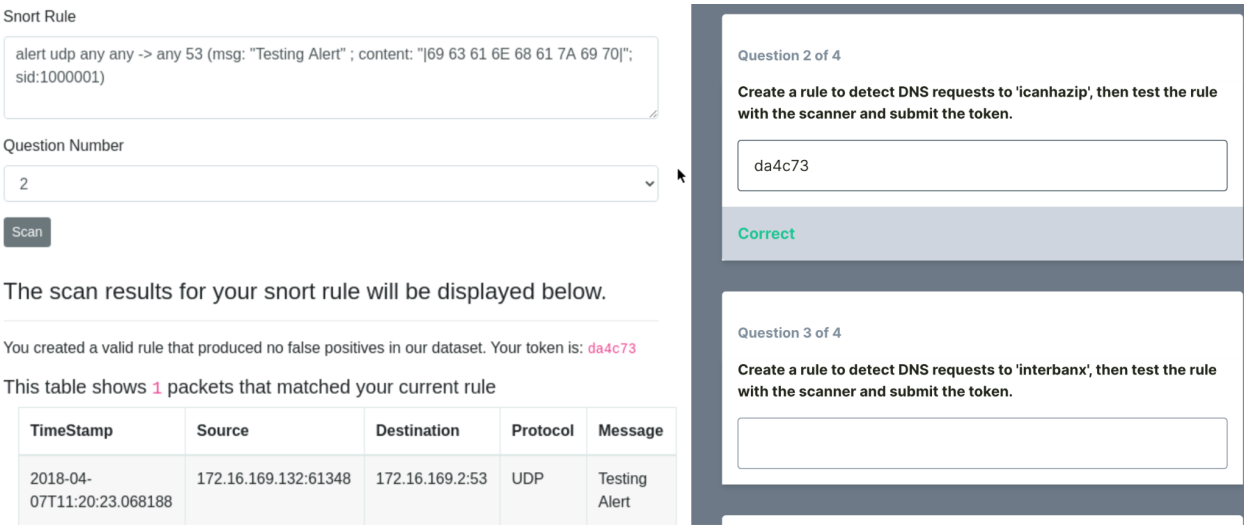


Snort Rules Ep.2 DNS

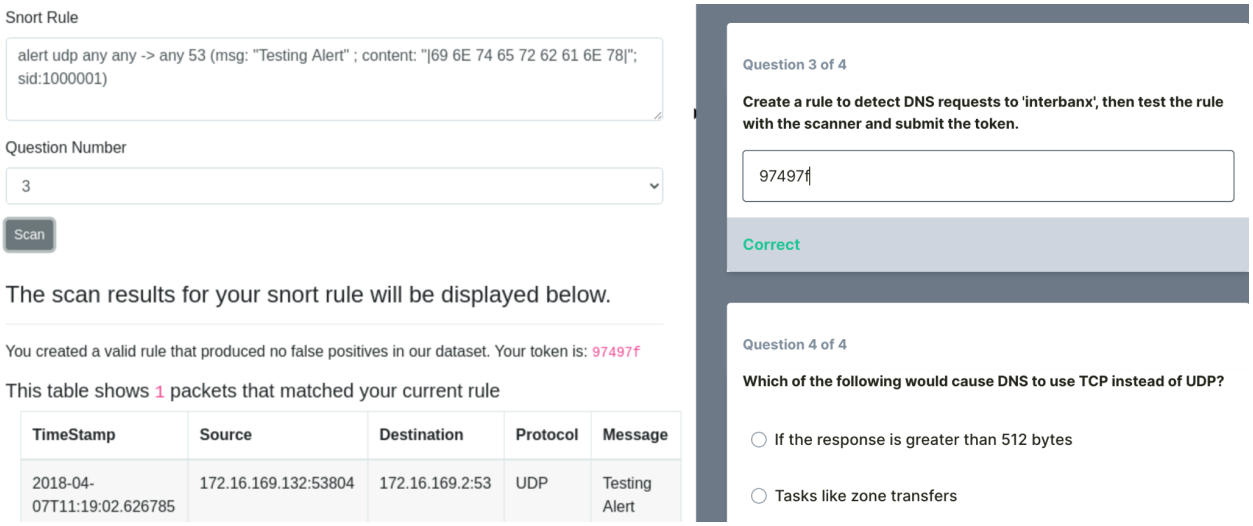
Question 1: As shown in the screenshot, the snort rule, *alert udp any any <> any 53*, captures bidirectional traffic on port 53 for udp which is DNS traffic. The resulting token is 8eb202.



Question 2: As shown below, the snort rule, *alert udp any any -> any 53 (msg: "Testing Alert"; content: "|69 63 61 6E 68 61 7A 69 70|"; sid: 1000001),* captures udp traffic from any source to port 53 (dns) to the domain 'icanhazip', which matches the content's hex value '69 63 61 6E 68 61 7A 69 70'. The resulting token is da4c73.



Question 3: As shown below, the snort rule, *alert udp any any -> any 53 (msg: "Testing Alert"; content: "|69 6E 74 65 72 62 61 6E 78|"; sid: 1000001),* captures udp traffic from any source to port 53 (dns) to the domain 'icanhazip', which matches the content's hex value '*69 6E 74 65 72 62 61 6E 78*'. The resulting token is 97497f.



Question 4: As shown below, the exceptions to DNS using UDP are when the response is bigger than 512 bytes, zone transfers, or DNS settings set by the operator. These circumstances will use TCP port 53 instead.

There are a few exceptions when the DNS will instead use TCP port 53:

- If the response is greater than 512 bytes
- Tasks like zone transfers
- Explicitly set by the DNS operator

Question 4 of 4

Which of the following would cause DNS to use TCP instead of UDP?

- ○ If the response is greater than 512 bytes
- ○ Tasks like zone transfers
- ○ Explicitly set by the DNS operator
- ◉ All of them

Correct

Result:

Snort

Snort Rules: Ep.2 – DNS

Exit

Congratulations Amanda

You have completed "**Snort Rules: Ep.2 – DNS**"

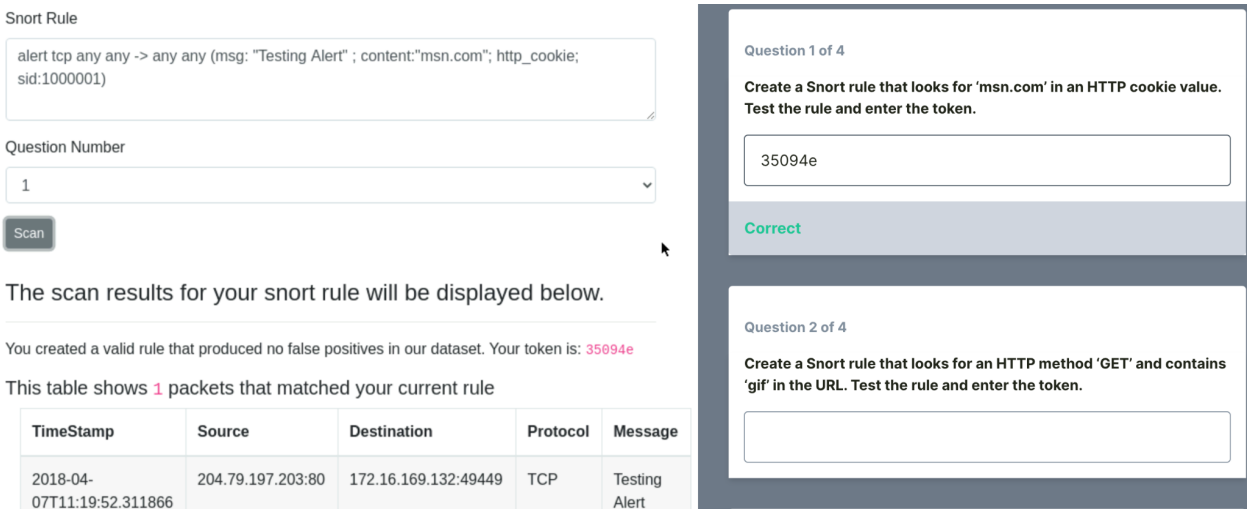
+300
Total Points
1400

3/3
You've completed 3 of your 3 labs for this week!

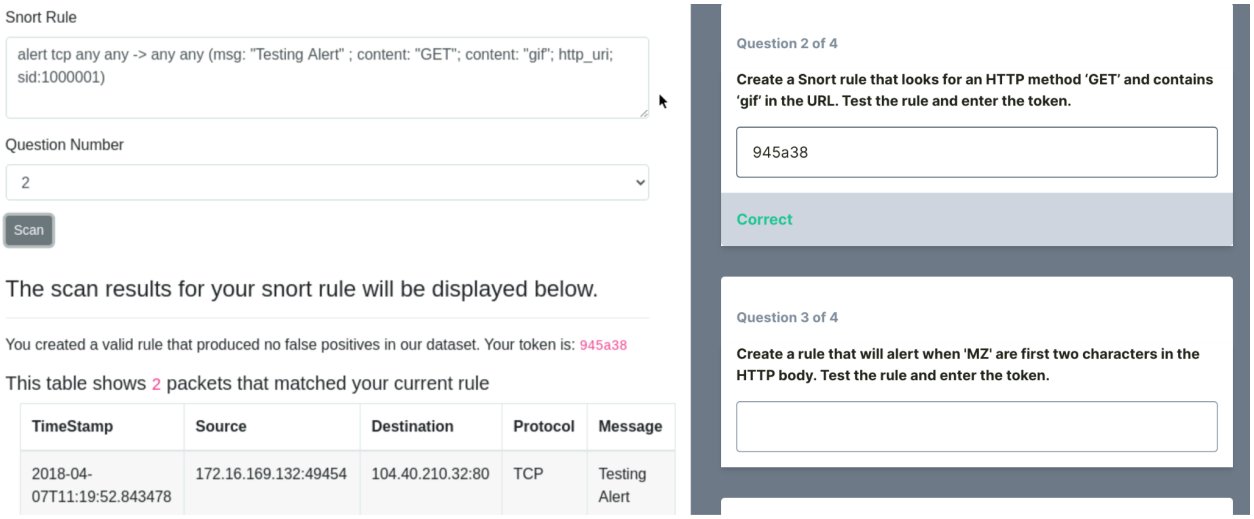
Next Lab
Snort Rules: Ep.3 – HTTP ›

Snort Rules Ep.3 HTTP

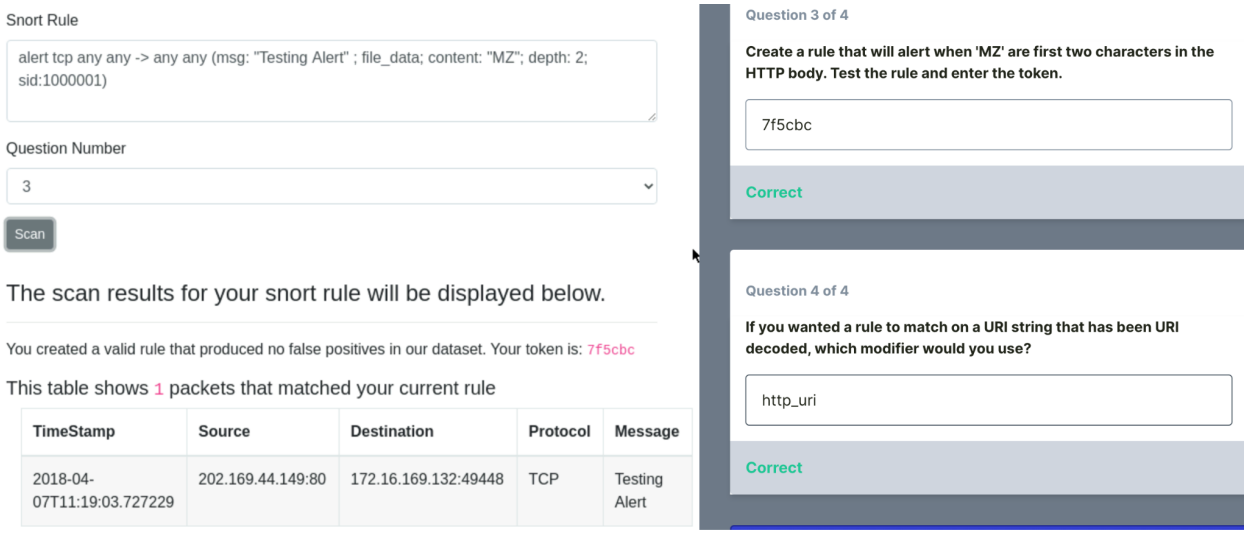
Question 1: As shown in the screenshot below, the snort rule, *alert tcp any any -> any any (msg: "Testing Alert"; content: "msn.com"; http_cookie; sid: 1000001),* looks for the content 'msn.com' in the cookie value from all tcp traffic. The resulting token is 35094e.



Question 2: As shown below, to look for content 'GET' and 'gif' in the url, i used the snort rule, *alert tcp any any -> any any (msg: "Testing Alert"; content: "GET"; content: "gif"; http_uri; sid: 1000001).* The resulting token was 945a38.

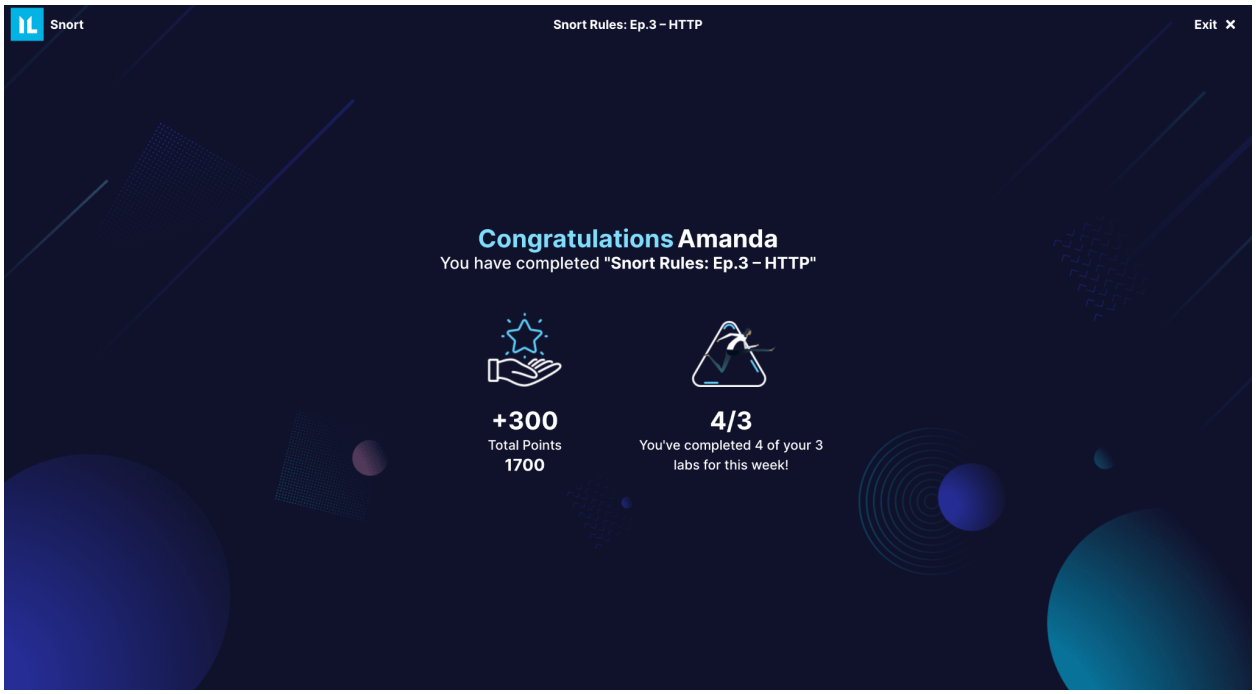


Question 3: As shown below, the snort rule, *alert tcp any any -> any any (msg: "Testing Alert"; file_data; content: "MZ"; depth: 2; sid: 1000001),* specifies the content of 'MZ' in file_data, which must be before the content. The depth of 2 is for the first two characters only. The resulting token is 7f5cbc.



Question 4: As shown in the screenshot above from q3, the modifier http_uri is used to match a decoded URI to a URI string, as opposed to http_raw_uri.

Result:



Write-Up

The first part of the lab was auditing data from log files; we did this with Kibana, which allowed us to see features like connections and access attempts. In this, we were able to see attempts from an attacker to log in using incorrect credentials. The UI allowed us to see messages alerting us of the invalid username logon, and the details associated with the person doing so, such as their IP, and which computer they were targeting. Further, we saw examples of a valid user log on and valid administrator that had the privilege to change passwords. This tool could be extremely useful to monitor networks and firewalls to prevent unauthorized access.

The latter part of the lab focused on snort rules to create rules for specific packet information. Alerts can be generated for specific packets on protocols like tcp, udp, and icmp, which we focused on. Some specific information we alerted on was content on specific ports. For http, snort rules had more modifiers and components to hone in on uri, cookie, or body information. This tool can be used to build very specific protections against known attacker techniques.