# Objective 9 Splunk!

Blue team professionals working in Security Operations Centers (SOCs) or incident response need software to collect log files from many different systems and display them in a useable, searchable format. Splunk is one product that fulfills this Security Information and Event Monitoring (SIEM) function. First, we to visit Fitzy Shortstack in the lobby of Santa’s castle.

|  |  |
| --- | --- |
| Graphical user interface, text  Description automatically generated | A picture containing text, floor, indoor  Description automatically generated |

## Terminal Yara Analysis

YARA is a system of rules for recording information about malware so it can be distributed to InfoSec people.

|  |  |
| --- | --- |
| Graphical user interface  Description automatically generated | Text  Description automatically generated |

In this terminal you will alter an executable binary file so that it passes YARA rules and measures the sweetness of North Pole candy. The terminal does not have a hex editor available, so you must edit the file with vim and xxd.

Text

Description automatically generated

### Step 1 question: Does it run?

Will ./the\_critical\_elf\_app run?

### Step 1 answer

No. When you run the application, it fails yara\_rule\_135.  
A screenshot of a computer

Description automatically generated with medium confidence

### Step 2 question: Find the problem rule

The applications more and less both allow you to search for a term. Just enter ‘/’ followed by the searchterm. What is the rule triggering on?

### Step 2 answer

The find feature of less (/searchterm) helps us locate rule 135.  
A screenshot of a computer

Description automatically generated with medium confidence

We must remove or alter the string “candycane” to pass this rule.  
Text

Description automatically generated

### Step 3 question: Edit a binary with vim? Really?

After much searching, I found these links which explain how to use vim and xxd together to edit binary files.  
<https://askubuntu.com/questions/786658/how-do-i-edit-the-binary-or-hexadecimal-data-of-a-file-in-ubuntu#:~:text=xxd%20is%20a%20linux%20command,b%20and%20then%20press%20ENTER%20.&text=Press%20ESC%20and%20then%20i,which%20allows%20you%20to%20edit.>  
<https://transang.me/edit-binary-file-with-vim-and-the-xxd-command/>  
<https://vi.stackexchange.com/questions/343/how-to-edit-binary-files-with-vim>

### Step 3 answer

Open the\_critical\_app in vim. Text

Description automatically generated

Then enter the vim command :%! xxd. This tells vim to run the xxd command against the text so we can read and edit it.  
Background pattern

Description automatically generated

We can edit this.  
A screenshot of a computer

Description automatically generated with medium confidence

Pressing “i” puts us in insert mode.  
Graphical user interface

Description automatically generated

Then change the string candycane. I chose to change 6e (last n in candycane) to 6d. Notice that the hex changed, but the ASCII translation in the right column did not change.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Now, change the text back to binary. Hit escape to exit insert mode, then scroll to the very top of the text.  
Text

Description automatically generated with medium confidence

Use :%! xxd -r to tell vi/xxd to put it back in binary.  
Background pattern

Description automatically generated

Enter :wq to write the changes and quit vm.

### Step 4 question: Next?

Try to run the binary again. If it fails, fix it (again.)

### Step 4 answer

Run the binary and fail the next rule.  
Text

Description automatically generated

This time the rule is looking for hex strings, either 6c 6962 632e 736f 2e36 or 726f 6772 616d 2121.  
Text

Description automatically generated

Our friend CyberChef from GCHQ can help.

|  |  |
| --- | --- |
| Graphical user interface  Description automatically generated with medium confidence | Graphical user interface  Description automatically generated with low confidence |

We had best change the second string, as the first may cause the libc.so.6 not to load. Use the same procedure as before. I like to pick on “m”, so I changed 6d to 6e.  
Graphical user interface, text, chat or text message

Description automatically generated

### Step 5 question: Again?

This time the binary hits a more complicated rule. There’s an easy way to fix it, and a hard way.

### Step 5 answer

When we run the app again, we hit yet another YARA rule.  
Text

Description automatically generated

This one is more complicated.  
Text

Description automatically generated

The condition is uint32(1) == 0x02464c45 AND filesize < 50KB AND 10 of them

Trying to find 10 strings that will not break the app by changing them seems like a lot of work. The uint32 test looks for the 32 bits, starting with the second byte in the file. It is little-endian, so will be backwards.  
 Graphical user interface, application, chat or text message

Description automatically generated

It is looking for the magic bytes of an ELF executable; we cannot change that. So, make the file larger than 50 kB.

### Step 6 question: Make it bigger?

The not quite correctly named Linux app, truncate, can also add padding to files.  
<https://linuxhint.com/truncate-command-in-linux/>  
<https://superuser.com/questions/1535572/can-i-create-an-image-with-a-specific-size-in-bytes/1536157#1536157>

### Step 6 answer

Use truncate to increase the file size to just over 50 kB.  
Text

Description automatically generated

Solved.  
Text

Description automatically generated

## Hints after solving Yara Analysis

Fitzy has assistance for us.

|  |  |
| --- | --- |
| Text  Description automatically generated  A green screen with white text  Description automatically generated with medium confidence  Text  Description automatically generated | Text  Description automatically generated |

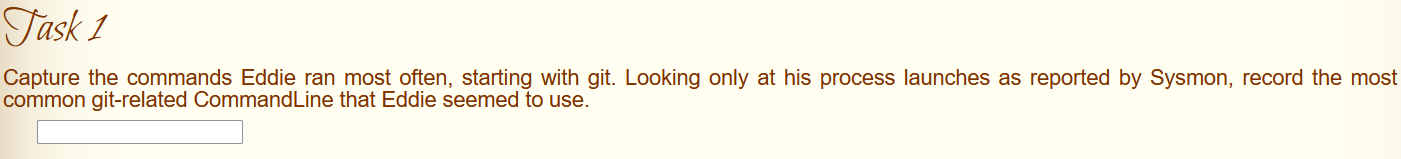
## Objective 9 Splunk!

Clicking the terminal next to Angel Candysalt in the castle great room takes you to a Splunk interface. The sample Splunk searches were very helpful.

Graphical user interface, text, application

Description automatically generated

### Task 1 question



### Task 1 answer

I used sample search 4 and found the most used git command was git status.

### Task 2 question

### Task 2 answer

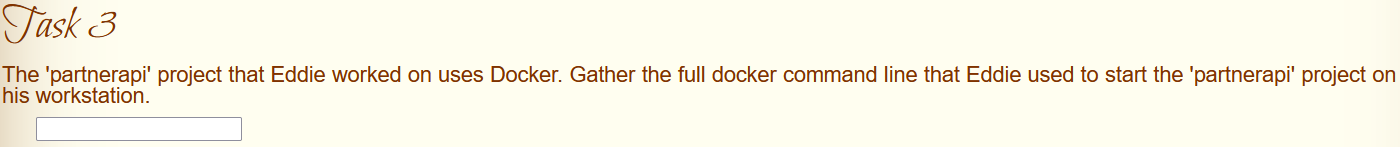
Use search 4 again, and add a regex to search for “origin” .

Graphical user interface, text, application, email

Description automatically generated

The answer was [git@github.com:elfnp3/partnerapi.git](mailto:git@github.com:elfnp3/partnerapi.git)

### Task 3 question



### Task 3 answer

Again, sample search 4 helps.

Graphical user interface, text, application, email

Description automatically generated

docker compose up

### Task 4 question

Note. Task 4 was modified in early January because the elfnp3 github entry was deleted.

A picture containing qr code

Description automatically generated

### Task 4 answer

Sample search 6 is set up for webhook events.  
Graphical user interface, text, application, email

Description automatically generated

Because of the change, they accept either <https://github.com/snoopysecurity/dvws-node> or dvws-node.

### Task 5 question



### Task 5 answer

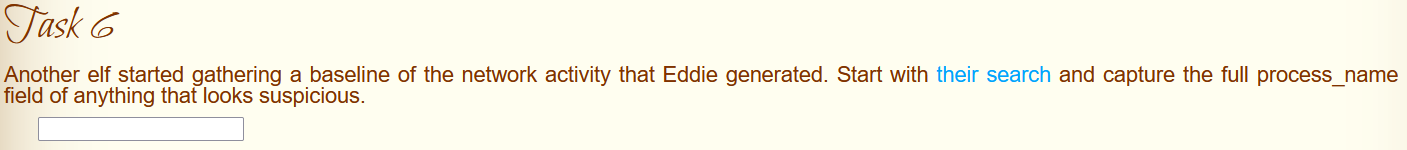
I modified search 4 again to add a regex for “npm”.

Graphical user interface, text, application, email

Description automatically generated

holiday-utils-js is the answer they are looking for.

### Task 6 question



### Task 6 answer

That search leads you to two IP addresses. The second is the one that had the alerts.

Graphical user interface, text, application

Description automatically generated

Drilling into one of the events took me here.

Text, application

Description automatically generated  
<snip>  
Graphical user interface, text, application

Description automatically generated

The problem process was /usr/bin/nc.openbsd.

### Task 7 question

Arrow

Description automatically generated with low confidence

### Task 7 answer

Starting with the query from Task 6, find the process\_id, 6791. Go back to sample search 2 and modify it to look for process\_id=6971.

Graphical user interface, text, application, email

Description automatically generated

Next, look for the ParentProcessID so we can look for other processes that it launched.  
Graphical user interface, text, application, email

Description automatically generated

The parent was 6788, so look for other processes that have the same parent.  
index=main sourcetype=journald source=Journald:Microsoft-Windows-Sysmon/Operational EventCode=1 ParentProcessId=6788

Graphical user interface, text, email

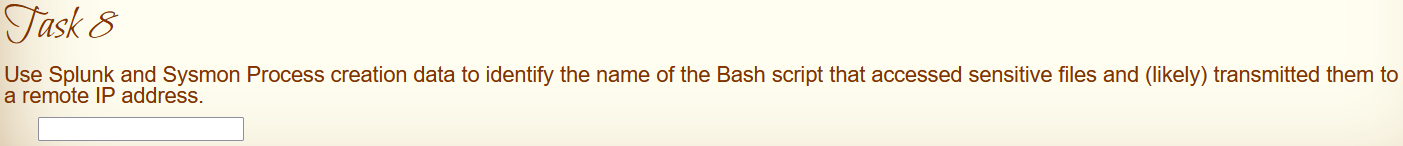
Description automatically generated

One of the children of 6788 launched a cat command to steal Eddie’s credentials.  
Graphical user interface, text, application

Description automatically generated

There were 6 files.

### Task 8 question



### Task 8 answer

Keep walking up the parent tree, starting with process\_id 6788.

Graphical user interface, text, application, email

Description automatically generated

The command line for 6788 was /bin/bash. No help there, find the parent for 6788.  
Graphical user interface, text, application, email

Description automatically generated

The parent of 6788 was 6784, what did it do?  
Graphical user interface, text, application, email

Description automatically generated

The command line was /bin/bash, so we keep looking. The parent of 6784 was 6783.  
Graphical user interface, text, application, email

Description automatically generated

The new parent is 6783, so search for it.  
Graphical user interface, text, application, email

Description automatically generated

Aha! It was /bin/bash/preinstall.sh, which is part of the package Santa asked Eddy to load. Oops.

Graphical user interface, text, application

Description automatically generated

The answer is preinstall.sh