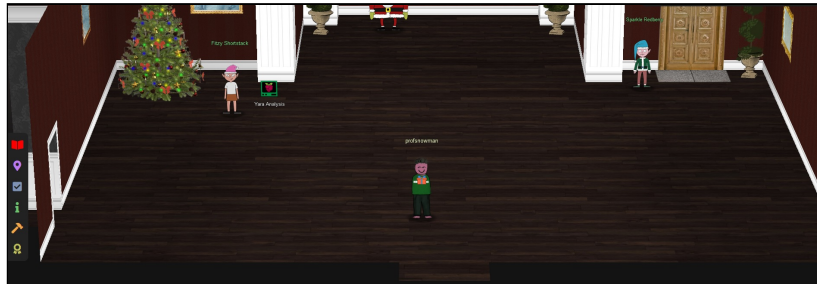


KringleCon 4: Calling Birds!

Entry



1. Click to talk to **Fity Shortstack**

Hiya, I'm Fitzy Shortstack!

I was just trying to learn a bit more about YARA with this here Cranberry Pi terminal.

I mean, I'm not saying I'm worried about attack threats from that other con next door, but...

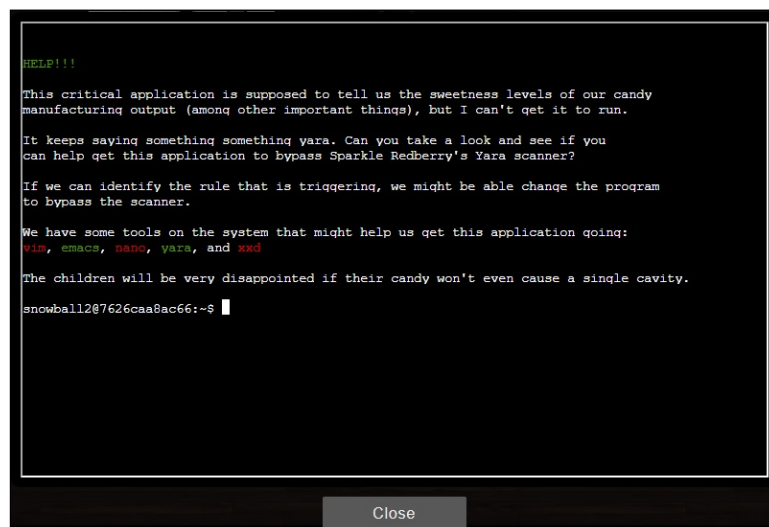
OK. I AM worried. I've been thinking a bit about how malware might bypass YARA rules.

If you can help me solve the issue in this terminal, I'll understand YARA so much better! Would you please check it out so I can learn?

And, I'll tell you what - if you help me with YARA, I'll give you some tips for Splunk!

I think if you make small, innocuous changes to the executable, you can get it to run in spite of the YARA rules.

2. Click **Yara Analysis** Cranberry Pi terminal



3. Type `ls`

```
the_critical_elf_app yara_rules
```

4. Type `./the_critical_elf_app`

```
yara_rule_135 ./the_critical_elf_app
```

5. Type `ls yara_rules/`

```
rules.yar
```

6. Type `vi yara_rules/rules.yar`7. Type `/135 {`

```
$x5 = "c:\\windows\\system32\\netstat.exe" fullword wide
$x6 = "c:\\windows\\system32\\netsh.exe" fullword wide
$x7 = "@*\\Ac\\Users\\tmc\\Documents\\TorUnzip\\Project1.vbp" fullword wide
$x8 = "svchost.TorUnzip.exe" fullword wide
$x9 = "svchost.taskhost.exe" fullword wide
$x10 = "TorUnzip.exe" fullword wide
$x11 = "Executes the Microsoft Task Host" fullword wide
$x12 = "\\taskhost.exe" fullword wide
$x13 = "c:\\Program Files\\Microsoft Updates\\temp\\tor.zip" fullword wide
$x14 = "\\svchost.exe \\Microsoft Update Service\\\" ENABLE" fullword wide
$x15 = "\\corunzip.exe" fullword wide
$x16 = "c:\\Program Files\\Microsoft Updates\\temp" fullword wide
$s17 = "SetTargetComputer" fullword ascii
$s18 = "set TargetComputer" fullword ascii
$s19 = "GetHostPortBytes" fullword ascii
$s20 = "GetTargetComputer" fullword ascii
condition:
  uint16(0) == 0x5a4d and filesize < 400KB and
  1 of ($x*) and 4 of them
}

rule yara rule 135 {
  meta:
    description = "binaries - file Sugar in the machinery"
    author = "Sparkle Redberry"
    reference = "North Pole Malware Research Lab"
    date = "1955-04-21"
    hash = "19ecaadb2159b566c39c999b0f860b4d8fc2824eb648e275f57a6dbceaf9b488"
  strings:
    $s = "candycane"
  condition:
    $s
}

rule yara rule 136 {
  meta:
    description = "binaries - file slide.exe"
    author = "Sparkle Redberry"
    reference = "North Pole Malware Research Lab"
    date = "2021-04-22"
    hash1 = "8e2bdcaee8dfefcfe42740a43a0079eb1babfc530200bcfb57b1b1a548852af1"
  strings:
    $s1 = "JSCRIPT.ENCODE" fullword ascii
/135 {
```

Close

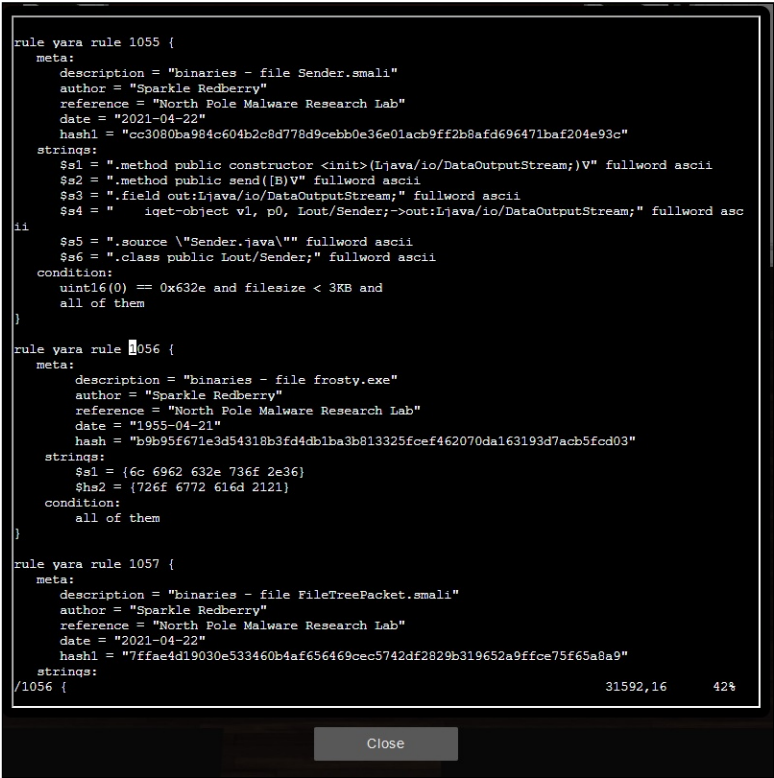
8. Close `vi`9. Type `vi the_critical_elf_app`10. Type `/candy`11. Change candycane to `Candycane`12. Save the modified file and close `vi`

13. Type `./the_critical_elf_app`

```
yara_rule_1056 ./the_critical_elf_app
```

14. Type `vi yara_rules/rules.yar`

15. Type `/1056 {`



\$s1	ASCII
6c 6962 632e 736f 2e36	libc.so.6
\$hs2	ASCII
726f 6772 616d 2121	rogram!!

16. Close `vi`

17. Type `vi the_critical_elf_app`

18. Type `/rogram!!`

19. Change rogram!! to `rogram!!`

20. Save the modified file and close `vi`

21. Type `./the_critical_elf_app`

```
yara_rule_1732 ./the_critical_elf_app
```

22. Type `vi yara_rules/rules.yar`

23. Type `/1732 {`

```

    $s20 = "0;h<q;h++){if(m[h][1]<0){this.side=\"left\"}}if(this. yaxis.name===\"yMidAxis\"&&
this.side===\"right\"){this. xnudge=this. xaxis" ascii
condition:
  uint16(0) == 0x2a2f and filesize < 20KB and
  1 of ($x*) and 4 of them
}

rule yara rule 1732 {
  meta:
    description = "binaries - always winter.exe"
    author = "Santa"
    reference = "North Pole Malware Research Lab"
    date = "1955-04-22"
    hash = "c1e31a539898aab18f483d9e7b3c698ea45799e78bddc919a7dbebb1b40193a8"
  strings:
    $s1 = "This is critical for the execution of this program!!" fullword ascii
    $s2 = " frame dummy init array entry" fullword ascii
    $s3 = ".note.gnu.property" fullword ascii
    $s4 = ".eh frame hdr" fullword ascii
    $s5 = " FRAME END " fullword ascii
    $s6 = " GNU EH FRAME HDR" fullword ascii
    $s7 = "frame dummy" fullword ascii
    $s8 = ".note.gnu.build-id" fullword ascii
    $s9 = "completed.8060" fullword ascii
    $s10 = " IO stdin used" fullword ascii
    $s11 = ".note.ABI-tag" fullword ascii
    $s12 = "naughty string" fullword ascii
    $s13 = "dastardly string" fullword ascii
    $s14 = " do global dtors aux fini array entry" fullword ascii
    $s15 = " libc start main@GLIBC 2.2.5" fullword ascii
    $s16 = "GLIBC 2.2.5" fullword ascii
    $s17 = "its a holly jolly variable" fullword ascii
    $s18 = " cxa finalize" fullword ascii
    $s19 = "HolidayBackChallenge{NotReallyAFlag}" fullword ascii
    $s20 = " libc csu init" fullword ascii
  condition:
    uint32(1) == 0x02464c45 and filesize < 50KB and
    10 of them
}

rule yara rule 1733 {
  meta:
    description = "binaries - file jqplot.barRenderer.min.js"

```

24. Type `vi the_critical_elf_app`

25. Type `__frame`

26. Change the following

String	Modified String
<code>__frame_dummy_init_array_entry</code>	<code>__frame_dummy_init_Array_entry</code>
<code>.note.gnu.property</code>	<code>.note.gnu.Property</code>
<code>.eh_frame_hdr</code>	<code>.eh_frame_Hdr</code>
<code>__FRAME_END__</code>	<code>__FRAME_eND__</code>
<code>__GNU_EH_FRAME_HDR</code>	<code>__GNU_EH_FRAME_hdr</code>
<code>frame_dummy</code>	<code>frame_Dummy</code>
<code>.note.gnu.build-id</code>	<code>.note.gnu.build-Id</code>
<code>.note.ABI-tag</code>	<code>.note.ABI-Tag</code>
<code>naughty string</code>	<code>naughty String</code>
<code>dastardly string</code>	<code>dastardly String</code>
<code>__do_global_dtors_aux_fini_array_entry</code>	<code>__do_global_dtors_aux_fini_array_Entry</code>
<code>its_a_holly_jolly_variable</code>	<code>its_a_Holly_jolly_variable</code>

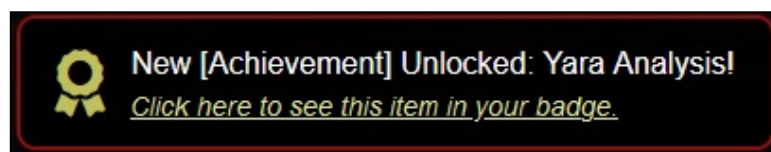
String**Modified String**

HolidayHackChallenge{NotReallyAFlag} **HolidayHackChallenge{NotReallyaFlag}**

27. Save the modified file and close **vi**

28. Type **./the_critical_elf_app**

```
Machine Running..  
Toy Levels: Very Merry, Terry  
Naughty/Nice Blockchain Assessment: Untampered  
Candy Sweetness Gauge: Exceedingly Sugarlicious  
Elf Jolliness Quotient: 4a6f6c6c7920456e6f7567682c204f76657274696d6520417070726f7666564
```



29. Click the **Close** button

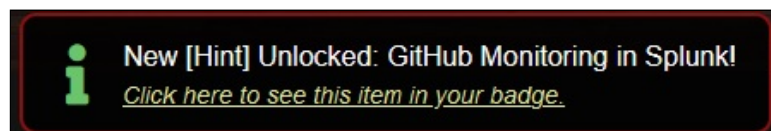
30. Click to talk to **Fity Shortstack**

Thanks - you figured it out!

Let me tell you what I know about Splunk.

Did you know Splunk recently added support for new data sources including Sysmon for Linux and GitHub Audit Log data?

Between GitHub audit log and webhook event recording, you can monitor all activity in a repository, including common git commands such as git add, git status, and git commit.



31. Click to talk to **Fity Shortstack**

You can also see cloned GitHub projects. There's a lot of interesting stuff out there. Did you know there are repositories of code that are Darn Vulnerable?

Sysmon provides a lot of valuable data, but sometimes correlation across data types is still necessary.

Sysmon network events don't reveal the process parent ID for example. Fortunately, we can pivot with a query to investigate process creation events once you get a process ID.



New [Hint] Unlocked: Sysmon Monitoring in Splunk!

[Click here to see this item in your badge.](#)

32. Click to talk to **Fity Shortstack**

Sometimes Sysmon data collection is awkward. Pipelining multiple commands generates multiple Sysmon events, for example.

Did you know there are multiple versions of the Netcat command that can be used maliciously? nc.openbsd, for example.



New [Hint] Unlocked: Malicious NetCat??!

[Click here to see this item in your badge.](#)

33. Click the **i** (Hints) icon

34. Click **Malicious NetCat??**

Malicious NetCat??

From: Fity Shortstack
Objective: 9) Splunk!

Did you know there are multiple versions of the Netcat command that can be used maliciously?
`nc.openbsd`, for example.

35. Click **Sysmon Monitoring in Splunk**

Sysmon Monitoring in Splunk

From: Fity Shortstack
Objective: 9) Splunk!

Sysmon network events don't reveal the process parent ID for example. Fortunately, we can pivot with a query to investigate process creation events once you get a process ID.

36. Click **GitHub Monitoring in Splunk**

GitHub Monitoring in Splunk

From: Fity Shortstack
Objective: 9) Splunk!

Between GitHub audit log and webhook event recording, you can monitor all activity in a repository, including common `git` commands such as `git add`, `git status`, and `git commit`.

37. Click **[Exit]**

38. Click to talk to **Sparkle Redberry**

Hey there! I'm Sparkle Redberry.

This year, the Santavator is in top working shape! We ironed out all of the issues from last year with it.

As for that tower next door, I hear they have an elevator of some sort too.

I just don't know if it would take me anywhere I'd really want to go.

39. Click the **Santavator**



41. Click **2** to move to the Talks Lobby

42. Click the **Santavator**

43. Click **3** to move to Santa's Office

44. Click the **Santavator**

45. Click **R** to move to NetWars

46. Click the **Santavator**

47. Click ***1** to move to the Lobby

48. Move **Left** and enter the **Dining Room**