// Note: For this lab, we will be working with QRP Corporation's CQC-11 FPGA.

// The CQC-11 operates with a 125MHz clock.

// Your design for a tone generator must support the following

// inputs/outputs:

// (NOTE: DO NOT CHANGE THE NAMES. OUR AUTOMATED GRADING TOOL

// REQUIRES THE USE OF THESE NAMES!)

// input clk - this will be connected to the 125MHz system clock

// input rst - this will be connected to the system board's reset bus

// input freq - a 32 bit integer indicating the required frequency

// (0 - 9999.99Hz) formatted as follows:

// 32'hf1206 or 32'd987654 = 9876.54Hz

// output wave\_out - a square wave output of the desired frequency

// you can create whatever other variables you need, but remember

// to initialize them to something!

`timescale 1ns/1ns

module tone\_generator (

input clk,

input rst,

input [31:0] freq,

output wave\_out

);

// ---- DO NOT CHANGE THE CODE ABOVE THIS LINE ----

// ---- IT IS NECESSARY FOR AUTOMATED ANALYSIS ----

// TODO: Add your code below.

// Remove the following line and add your own implementation.

// Note: It's silly, but it compiles...

reg [31:0] CLOCK\_FREQ = 125000000;

reg [31:0] counter =0;

reg [31:0] orig\_counter=0;

reg sq\_wave\_reg = 0;

reg [31:0] freq\_calc=0;

assign wave\_out = sq\_wave\_reg;

always @(posedge clk or posedge rst)

begin

if (rst==1)

begin

if (($rtoi(freq \* 10) - ($rtoi(freq) \* 10)) > 4)

freq\_calc <= $rtoi(freq) +1;

else

freq\_calc <= $rtoi(freq);

//freq\_calc <=440487;

counter <= CLOCK\_FREQ/(freq\_calc/50);

// counter <= CLOCK\_FREQ/freq\*50;

if ($rtoi(counter\*10) - ($rtoi(counter)\*10)>4)

orig\_counter <= ($rtoi(counter))+1;

else

orig\_counter <= $rtoi(counter);

sq\_wave\_reg <= 1;

//counter <= $rtoi(counter);;

end

else

begin

if (counter == 0)

begin

sq\_wave\_reg <= ~ sq\_wave\_reg;

counter <= orig\_counter - 1;

end

else

counter <=counter - 1;

end

end

endmodule

SPLUNK

Git status

git@github.com:elfnp3/partnerapi.git

Task 3

The 'partnerapi' project that Eddie worked on uses Docker. Gather the full docker command line that Eddie used to start the 'partnerapi' project on his workstation.

docker compose up

Task 4

Eddie had been testing automated static application security testing (SAST) in GitHub. Vulnerability reports have been coming into Splunk in JSON format via GitHub webhooks. Search all the events in the main index in Splunk and use the sourcetype field to locate these reports. Determine the URL of the vulnerable GitHub repository that the elves cloned for testing and document it here. You will need to search outside of Splunk (try GitHub) for the original name of the repository.

<https://github.com/snoopysecurity/dvws-node>

Task 5

Santa asked Eddie to add a JavaScript library from NPM to the 'partnerapi' project. Determine the name of the library and record it here for our workshop documentation.

holiday-utils-js

Task 6

Another elf started gathering a baseline of the network activity that Eddie generated. Start with their search and capture the full process\_name field of anything that looks suspicious.

/usr/bin/nc.openbsd

Task 7

Uh oh. This documentation exercise just turned into an investigation. Starting with the process identified in the previous task, look for additional suspicious commands launched by the same parent process. One thing to know about these Sysmon events is that Network connection events don't indicate the parent process ID, but Process creation events do! Determine the number of files that were accessed by a related process and record it here.

\* ParentProcessId=6788

6

Task 8

Use Splunk and Sysmon Process creation data to identify the name of the Bash script that accessed sensitive files and (likely) transmitted them to a remote IP address.

Preinstall.sh

**Thank you for helping Santa complete his investigation! Santa says you're a whiz!**

Answer all the questions in the quizme executable:

- What port does 34.76.1.22 have open?

- What port does 34.77.207.226 have open?

- How many hosts appear "Up" in the scan?

- How many hosts have a web port open? (Let's just use TCP ports 80, 443, and 8080)

- How many hosts with status Up have no (detected) open TCP ports?

- What's the greatest number of TCP ports any one host has open?

Grep 34.76.1.22 bigscan.gnmap

62078

grep 34.77.207.226 bigscan.gnmap

8080

How many appear UP in scan

grep Up bigscan.gnmap | wc -l

26054

grep -E "(80|443|8080)/open" bigscan.gnmap | wc -l

14372

Ipv6

PieceonEarth

Elf challenge

import elf, munchkins, levers, lollipops, yeeters, pits

all\_lollipops = lollipops.get()

munch=munchkins.get()

q=munch[0].ask()

for lollipop in all\_lollipops:

elf.moveTo(lollipop.position)

elf.moveTo(munch[0].position)

munch[0].answer(list(q.keys())[list(q.values()).index("lollipop")])

elf.moveRight(1)

elf.moveUp(4)

shell code pimer

https://tracer.kringlecastle.com/?**cheat**#7a16e8eb-9ab0-4b45-9921-2d077f138c4d

; TODO: Get a reference to this

call Mess

db '/var/northpolesecrets.txt',0

Mess:

pop rbp

; TODO: Call sys\_open

mov rax,2

mov rdi,rbp

mov rsi,0

mov rdx,0

syscall

; TODO: Call sys\_read on the file handle and read it into rsp

mov rbp,rax

mov rax,0

mov rdi,rbp

mov rsi,rsp

mov rdx, 10000

syscall

; TODO: Call sys\_write to write the contents from rsp to stdout (1)

mov rax,1

mov rdi,1

mov rsi, rsp

mov rdx,10000

syscall

; TODO: Call sys\_exit

mov rax,60

mov rdi,0

syscall

$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 = "HolidayHackChallenge{NotReallyAFlag}" fullword ascii

$s20 = "\_\_libc\_csu\_init" fullword ascii

HINTS

For this exercise, we're going to read a specific file… let's say, /var/northpolesecrets.txt… and write it to stdout. No reason for the name, but since this is Jack Frost's troll-trainer, it might be related to a top-secret mission!