RazorHack 2024 Writeup

“Someone’s Always Got to Fall for the Email Scam” & “Ghost In The Machine”

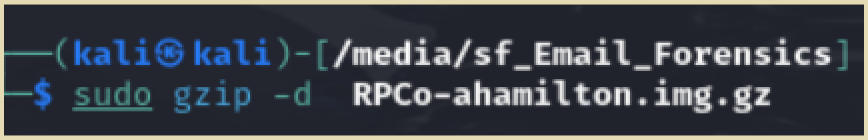
RPCO-M-41x01-10 a.k.a. Email Forensics

Well... looks like I need to make a blacklist for common passwords in hash crack databases. With Ayden's password the intruder could have done any number of things, but luckily, he doesn't hold too many privileged positions. Don't tell him I said that.

I pulled a disk image of Ayden's computer when I heard your report. See if you can find what the attacker did next. It doesn't look like anything critical was on his computer fortunately. Maybe they tried to impersonate him?

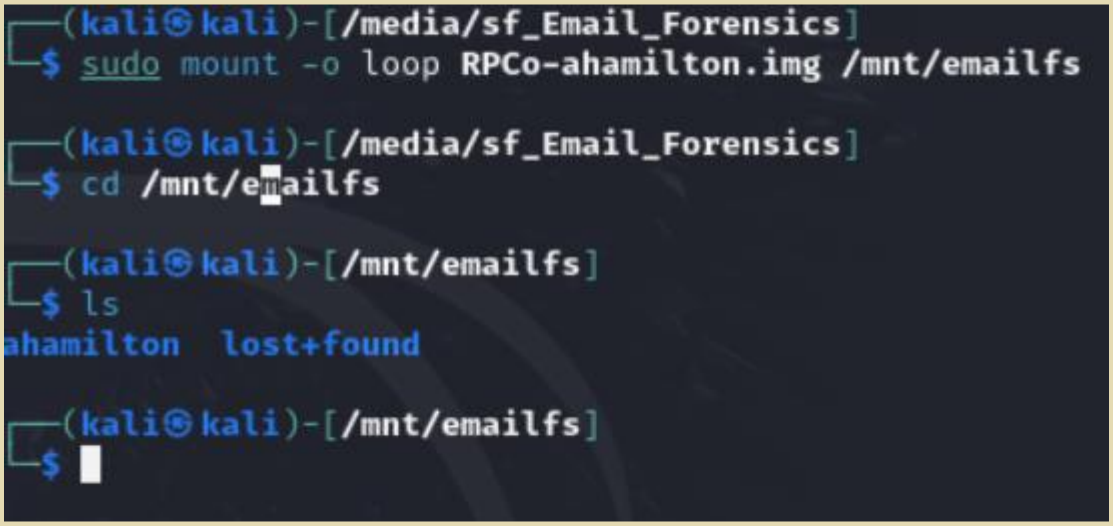
Author: Henry

Shane suspects the attacker tried to impersonate Ayden in an email or other form of communication after his password was compromised by the intruder. The first objective of this challenge is to find out what tracks the attacker left in Ayden's account. You're given a disk image file. What's your first task? To figure out how to mount the file and search its contents or use a tool to search it instead. First we unzip the RPCo-ahamilton.img.gz file using *gzip*.



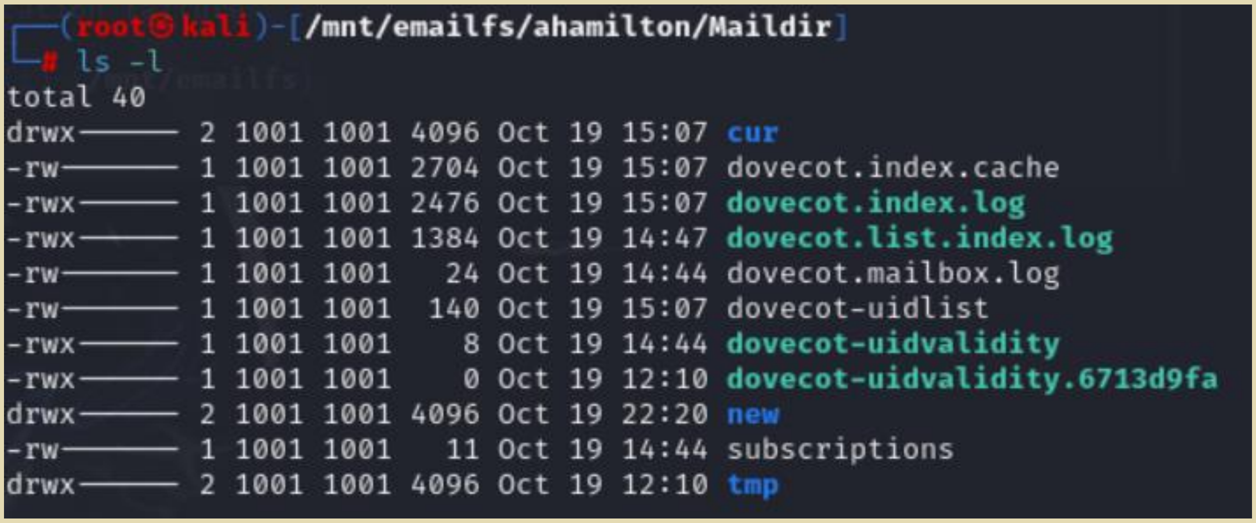
```
──(kali㉿kali)-[/media/sf_Email_Forensics]
─$ sudo gzip -d  RPCo-ahamilton.img.gz
```

We can then mount the image file using the mount command to our local file system.



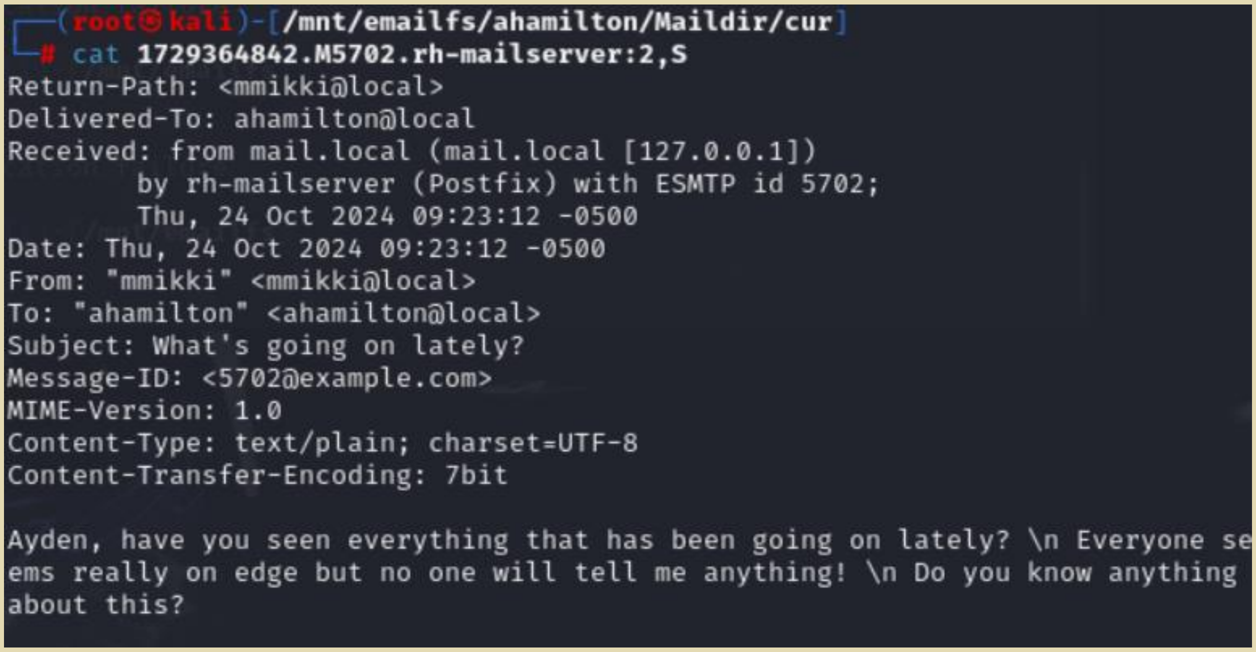
```
┌──(kali㉿kali)-[/media/sf_Email_Forensics]
└─$ sudo mount -o loop RPCo-ahamilton.img /mnt/emailfs

┌──(kali㉿kali)-[/media/sf_Email_Forensics]
└─$ cd /mnt/emailfs

┌──(kali㉿kali)-[/mnt/emailfs]
└─$ ls
ahamilton  lost+found

┌──(kali㉿kali)-[/mnt/emailfs]
└─$ 
```

The loop option just sets the loop device which is used in the mounting of the image. You can see in the image above that we can *cd* into the mounted image which is sign that it mounted correctly. Here you can see that the image has the user account for Ayden Hamilton (ahamilton). Now we must think about what we are looking for. Going into *ahamilton* we can see the *Maildir* directory. A quick Google search tells us that this is a typical email server or user account directory. Listing out the contents of the Maildir directory we find the following information.



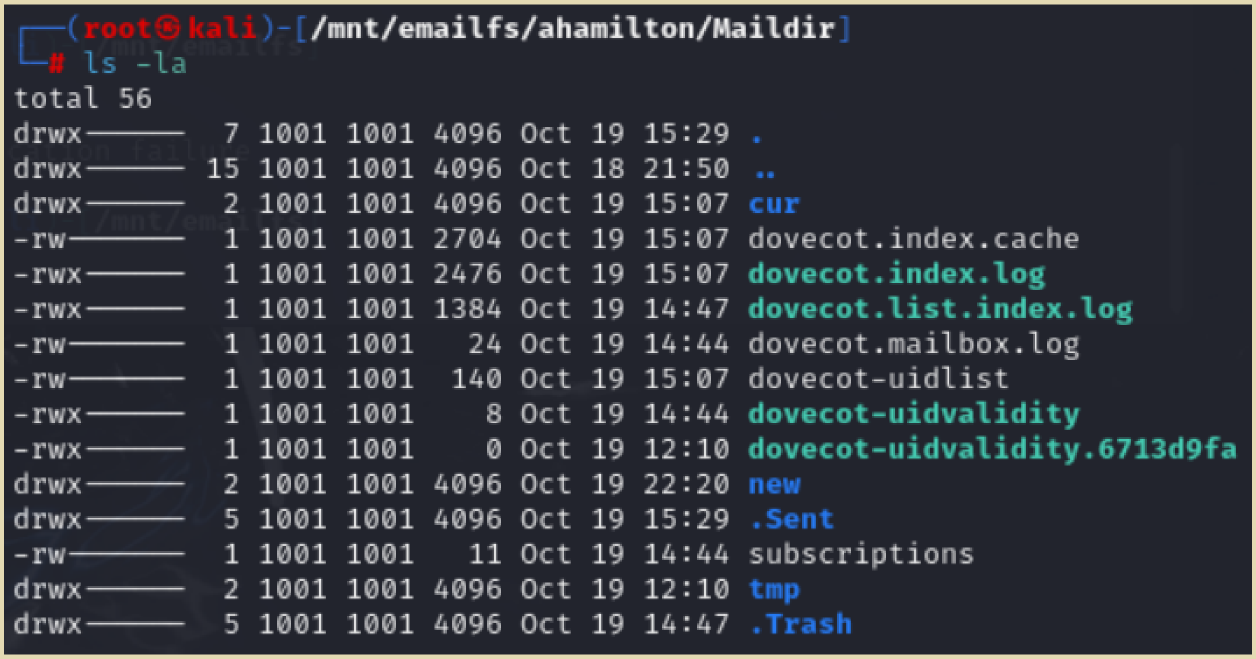
```
┌──(root㉿kali)-[/mnt/emailfs/ahamilton/Maildir]
└─# ls -l
total 40
drwx------ 2 1001 1001 4096 Oct 19 15:07 cur
-rw------- 1 1001 1001 2704 Oct 19 15:07 dovecot.index.cache
-rwx------ 1 1001 1001 2476 Oct 19 15:07 dovecot.index.log
-rwx------ 1 1001 1001 1384 Oct 19 14:47 dovecot.list.index.log
-rw------- 1 1001 1001   24 Oct 19 14:44 dovecot.mailbox.log
-rw------- 1 1001 1001  140 Oct 19 15:07 dovecot-uidlist
-rwx------ 1 1001 1001    8 Oct 19 14:44 dovecot-uidvalidity
-rwx------ 1 1001 1001    0 Oct 19 12:10 dovecot-uidvalidity.6713d9fa
drwx------ 2 1001 1001 4096 Oct 19 22:20 new
-rw------- 1 1001 1001   11 Oct 19 14:44 subscriptions
drwx------ 2 1001 1001 4096 Oct 19 12:10 tmp
```

Here we find various files and three directories: cur, new, and tmp. Looking in cur we can find two emails, but a quick print out shows us this isn't what we are looking for.



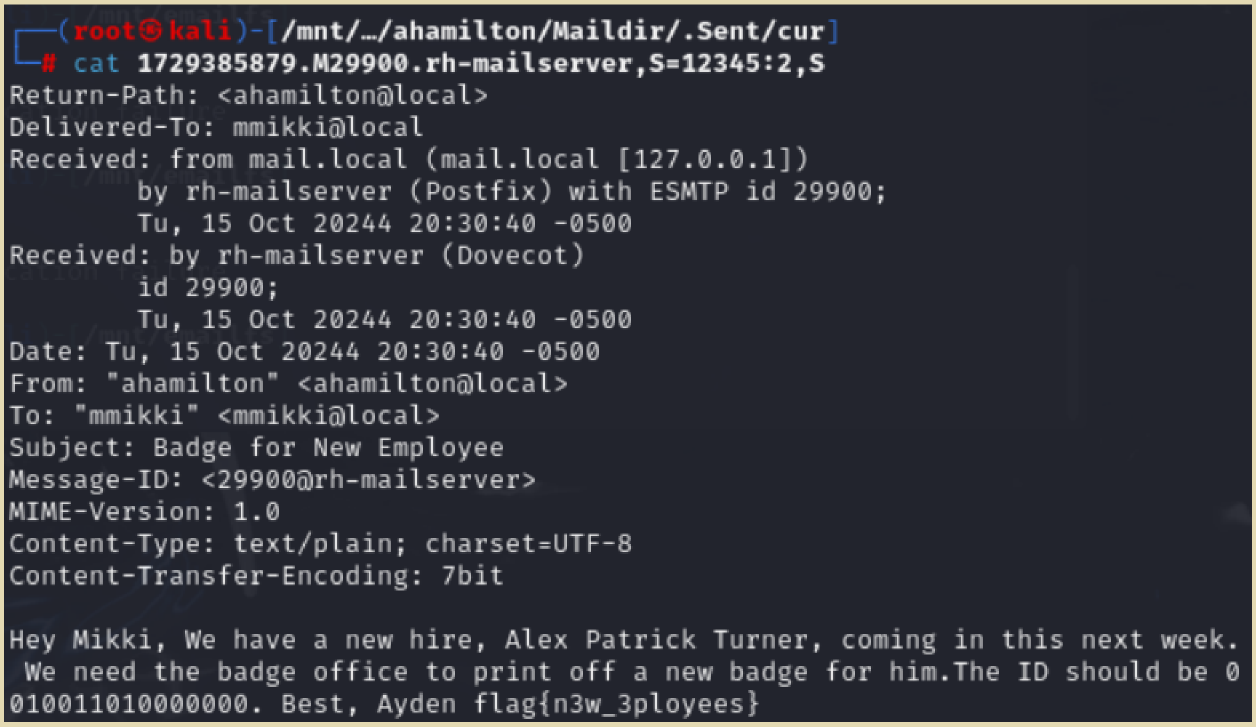
```
┌──(root㉿kali)-[/mnt/emailfs/ahamilton/Maildir/cur]
└─# cat 1729364842.M5702.rh-mailserver:2,S
Return-Path: <mmikki@local>
Delivered-To: ahamilton@local
Received: from mail.local (mail.local [127.0.0.1])
        by rh-mailserver (Postfix) with ESMTP id 5702;
        Thu, 24 Oct 2024 09:23:12 -0500
Date: Thu, 24 Oct 2024 09:23:12 -0500
From: "mmikki" <mmikki@local>
To: "ahamilton" <ahamilton@local>
Subject: What's going on lately?
Message-ID: <5702@example.com>
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

Ayden, have you seen everything that has been going on lately? \n Everyone se
ems really on edge but no one will tell me anything! \n Do you know anything
about this?
```

We must keep looking instead. Let's go back to Maildir. Last time we just did a standard list of all the files and directories, however, often these types of systems will use hidden directories for items that aren't essential to the user. Let's do another list but this time showing all the contents of the directory.



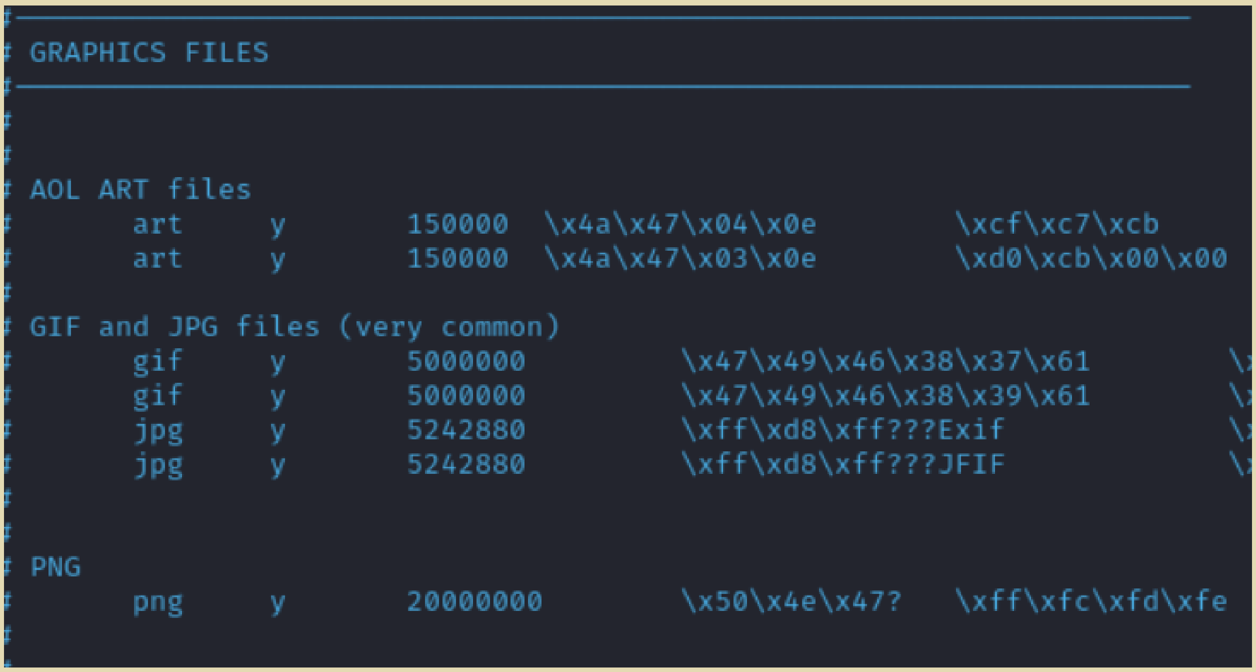
```
┌──(root㉿kali)-[/mnt/emailfs/ahamilton/Maildir]
└─# ls -la
total 56
drwx------  7 1001 1001 4096 Oct 19 15:29 .
drwx------ 15 1001 1001 4096 Oct 18 21:50 ..
drwx------  2 1001 1001 4096 Oct 19 15:07 cur
-rw-------  1 1001 1001 2704 Oct 19 15:07 dovecot.index.cache
-rwx------  1 1001 1001 2476 Oct 19 15:07 dovecot.index.log
-rwx------  1 1001 1001 1384 Oct 19 14:47 dovecot.list.index.log
-rw-------  1 1001 1001   24 Oct 19 14:44 dovecot.mailbox.log
-rw-------  1 1001 1001  140 Oct 19 15:07 dovecot-uidlist
-rwx------  1 1001 1001    8 Oct 19 14:44 dovecot-uidvalidity
-rwx------  1 1001 1001    0 Oct 19 12:10 dovecot-uidvalidity.6713d9fa
drwx------  2 1001 1001 4096 Oct 19 22:20 new
drwx------  5 1001 1001 4096 Oct 19 15:29 .Sent
-rw-------  1 1001 1001   11 Oct 19 14:44 subscriptions
drwx------  2 1001 1001 4096 Oct 19 12:10 tmp
drwx------  5 1001 1001 4096 Oct 19 14:47 .Trash
```

Now we see two hidden directories, *.Sent* and *.Trash*. We think the intruder sent an email under the guise of Ayden so let's check the sent folder. Checking the current emails, we see two files, which if we print out we find the flag.



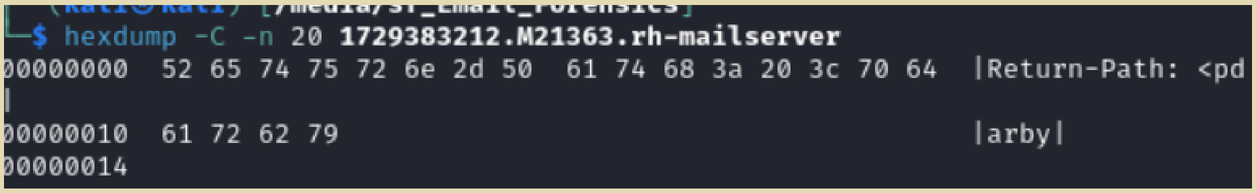
```
┌──(root㉿kali)-[/mnt/…/ahamilton/Maildir/.Sent/cur]
└─# cat 1729385879.M29900.rh-mailserver,S=12345:2,S
Return-Path: <ahamilton@local>
Delivered-To: mmikki@local
Received: from mail.local (mail.local [127.0.0.1])
        by rh-mailserver (Postfix) with ESMTP id 29900;
        Tu, 15 Oct 20244 20:30:40 -0500
Received: by rh-mailserver (Dovecot)
        id 29900;
        Tu, 15 Oct 20244 20:30:40 -0500
Date: Tu, 15 Oct 20244 20:30:40 -0500
From: "ahamilton" <ahamilton@local>
To: "mmikki" <mmikki@local>
Subject: Badge for New Employee
Message-ID: <29900@rh-mailserver>
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

Hey Mikki, We have a new hire, Alex Patrick Turner, coming in this next week.
 We need the badge office to print off a new badge for him.The ID should be 0
010011010000000. Best, Ayden flag{n3w_3ployees}
```

Now the sent folder is hidden by default so Ayden might not notice if something gets added to it. However, a reply from Marzel Mikki would show up in Ayden's inbox, the attacker would want to monitor for that and delete that file permanently when it came in. This is the second challenge. Using the same image, we must find the deleted email file with Marzell's reply to find out whether the badge was made for pseudonym Alex Patrick Turner. There are many different forensics tools out there that you can use, such as foremost, autopsy, scalpel, extundelete, and others. However, I found that scalpel solves this next problem the quickest as it does not require you to mount the image again. An important thing to note about forensics tools is that they search and recover files based on headers and footers on the provided image. These are the first and last bytes of hex of a file. When you delete a file, you are only deleting the reference to that file, the actual memory where it resided stays intact until it is rewritten with a different value. You'll notice in this challenge that I gave you a copy of one of the email files. This should have sparked your interest when you read you would have to recover the deleted email file. Let's look at how we specify to scalpel what files we want to search for in *scalpel.conf.*



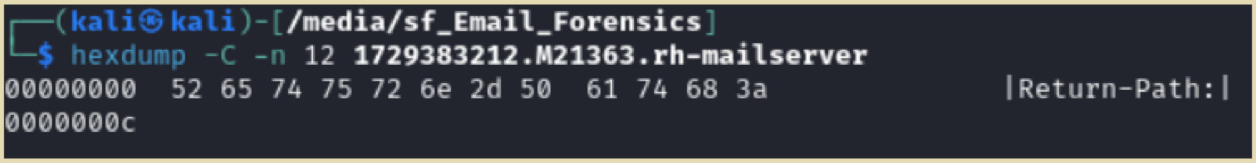
```
#-------------------------------------------------------------------------
# GRAPHICS FILES
#-------------------------------------------------------------------------
#
#
# AOL ART files
#       art     y       150000  \x4a\x47\x04\x0e        \xcf\xc7\xcb
#       art     y       150000  \x4a\x47\x03\x0e        \xd0\xcb\x00\x00
#
# GIF and JPG files (very common)
#       gif     y       5000000         \x47\x49\x46\x38\x37\x61        \
#       gif     y       5000000         \x47\x49\x46\x38\x39\x61        \
#       jpg     y       5242880         \xff\xd8\xff???Exif             \
#       jpg     y       5242880         \xff\xd8\xff???JFIF             \
#
#
# PNG
#       png     y       20000000        \x50\x4e\x47?   \xff\xfc\xfd\xfe
#
```

Here you will find common headers and footers for many different file types that are commented out. To run scalpel and look for these files you would simply uncomment that line, save the file, and run scalpel. However, we are looking to make a custom header to look for. First let's find the header for the file using *hexdump*. We want whatever we designate as the header to be consistent between email files in order for scalpel to pull the information effectively. Using *hexdump -C -n 20 {file}* we can print the first 20 bytes of the file.



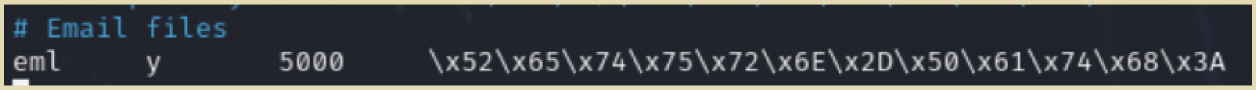
```
└─$ hexdump -C -n 20 1729383212.M21363.rh-mailserver
00000000  52 65 74 75 72 6e 2d 50  61 74 68 3a 20 3c 70 64  |Return-Path: <pd
|
00000010  61 72 62 79                                       |arby|
00000014
```

We can see here that gives a little more information than we need. We want our header to be the "Return-Path:" string since that's consistent between files. Instead let's use *hexdump -C -n 12 {file}* to get the first 12 bytes.



```
┌──(kali㉿kali)-[/media/sf_Email_Forensics]
└─$ hexdump -C -n 12 1729383212.M21363.rh-mailserver
00000000  52 65 74 75 72 6e 2d 50  61 74 68 3a              |Return-Path:|
0000000c
```

This is what we will use for our header. Let's go back to our scalpel.conf file and add a new header definition to the file.



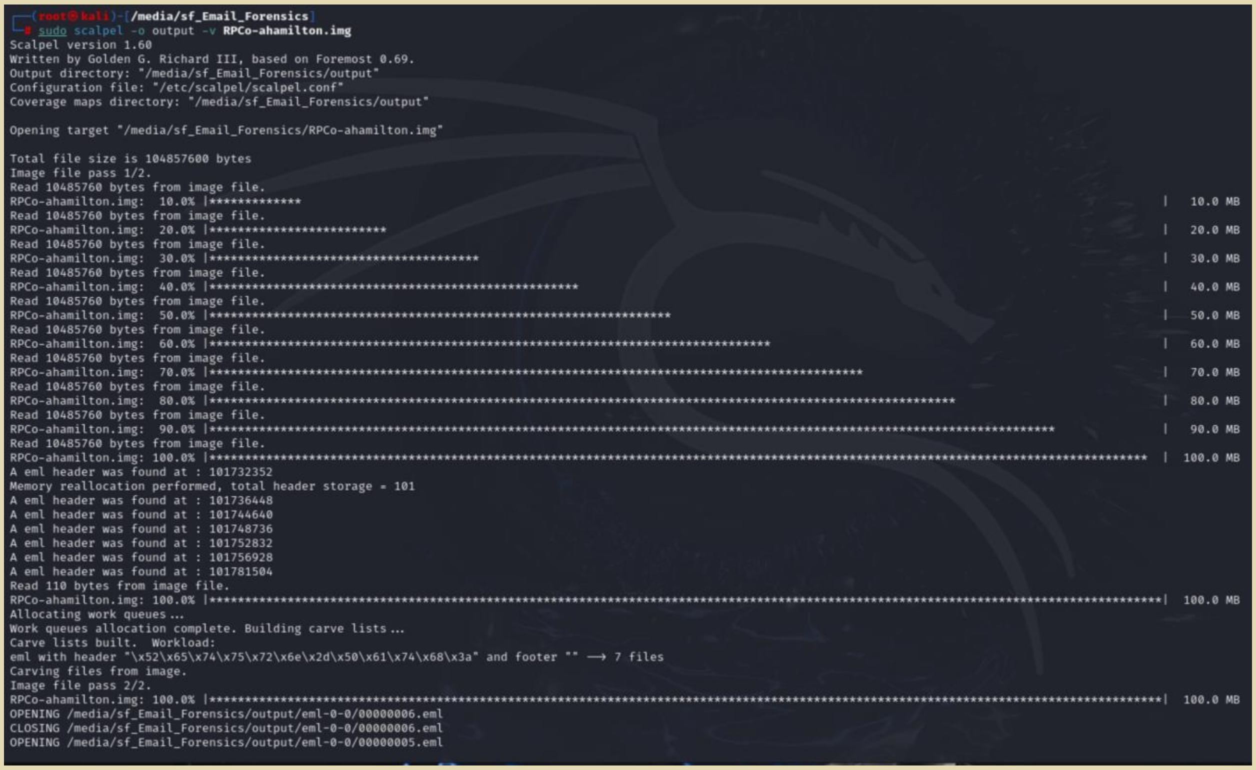
```
# Email files
eml     y       5000     \x52\x65\x74\x75\x72\x6E\x2D\x50\x61\x74\x68\x3A
```

The first attribute for the record is a placeholder for the file extension type. The second is whether or not the header or footer is a case sensitive. The third is how many bytes you want to scrape from the image when a header is found (if no footer is defined this is how many bytes will be scraped, otherwise the footer is used as the end). The fourth and fifth item the definition of the header and footer of the file that scalpel is going to be looking for. Now that we have the header defined lets save the conf file and run scalpel on the image.

This blog gives good instructions on the basics of scalpel: https://www.redhat.com/en/blog/find-lost-files-scalpel

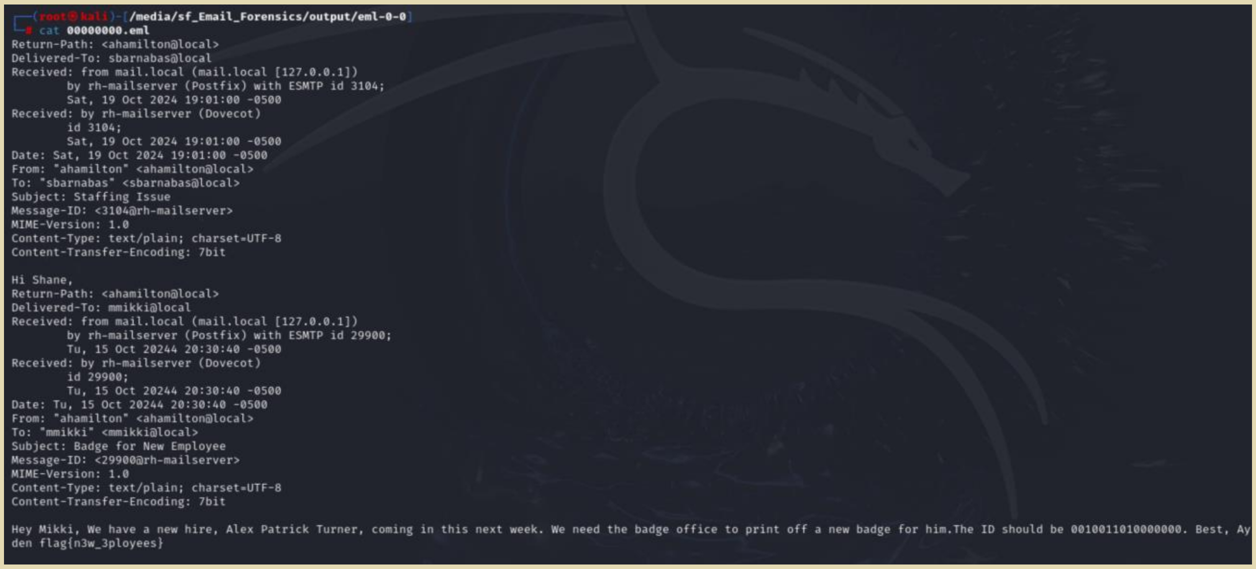
You can run scalpel on an image file or on a mounted image. We'll use the command *sudo scalpel -o output -v RPCo-ahamilton.img* to run it. This specifies a directory for results called */output* and runs scalpel in verbose mode on the provide .img file.

You should see an output similar to the one below when you run scalpel.



```
┌──(root㉿kali)-[/media/sf_Email_Forensics]
└─# sudo scalpel -o output -v RPCo-ahamilton.img
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.69.
Output directory: "/media/sf_Email_Forensics/output"
Configuration file: "/etc/scalpel/scalpel.conf"
Coverage maps directory: "/media/sf_Email_Forensics/output"

Opening target "/media/sf_Email_Forensics/RPCo-ahamilton.img"

Total file size is 104857600 bytes
Image file pass 1/2.
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  10.0% |*************                                                                                                                           |  10.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  20.0% |*************************                                                                                                               |  20.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  30.0% |**************************************                                                                                                  |  30.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  40.0% |*****************************************************                                                                                   |  40.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  50.0% |******************************************************************                                                                      |  50.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  60.0% |********************************************************************************                                                        |  60.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  70.0% |**********************************************************************************************                                          |  70.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  80.0% |***********************************************************************************************************                             |  80.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img:  90.0% |*************************************************************************************************************************               |  90.0 MB
Read 10485760 bytes from image file.
RPCo-ahamilton.img: 100.0% |*******************************************************************************************************************************************  | 100.0 MB
A eml header was found at : 101732352
Memory reallocation performed, total header storage = 101
A eml header was found at : 101736448
A eml header was found at : 101744640
A eml header was found at : 101748736
A eml header was found at : 101752832
A eml header was found at : 101756928
A eml header was found at : 101781504
Read 110 bytes from image file.
RPCo-ahamilton.img: 100.0% |*********************************************************************************************************************************************| 100.0 MB
Allocating work queues...
Work queues allocation complete. Building carve lists...
Carve lists built.  Workload:
eml with header "\x52\x65\x74\x75\x72\x6e\x2d\x50\x61\x74\x68\x3a" and footer "" --> 7 files
Carving files from image.
Image file pass 2/2.
RPCo-ahamilton.img: 100.0% |*********************************************************************************************************************************************| 100.0 MB
OPENING /media/sf_Email_Forensics/output/eml-0-0/00000006.eml
CLOSING /media/sf_Email_Forensics/output/eml-0-0/00000006.eml
OPENING /media/sf_Email_Forensics/output/eml-0-0/00000005.eml
```

Now we can just traverse to the output files and look through the result to see if scalpel found the email we were looking for.



```
┌──(root㉿kali)-[/media/sf_Email_Forensics/output/eml-0-0]
└─# cat 00000000.eml
Return-Path: <ahamilton@local>
Delivered-To: sbarnabas@local
Received: from mail.local (mail.local [127.0.0.1])
        by rh-mailserver (Postfix) with ESMTP id 3104;
        Sat, 19 Oct 2024 19:01:00 -0500
Received: by rh-mailserver (Dovecot)
        id 3104;
        Sat, 19 Oct 2024 19:01:00 -0500
Date: Sat, 19 Oct 2024 19:01:00 -0500
From: "ahamilton" <ahamilton@local>
To: "sbarnabas" <sbarnabas@local>
Subject: Staffing Issue
Message-ID: <3104@rh-mailserver>
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

Hi Shane,
Return-Path: <ahamilton@local>
Delivered-To: mmikki@local
Received: from mail.local (mail.local [127.0.0.1])
        by rh-mailserver (Postfix) with ESMTP id 29900;
        Tu, 15 Oct 20244 20:30:40 -0500
Received: by rh-mailserver (Dovecot)
        id 29900;
        Tu, 15 Oct 20244 20:30:40 -0500
Date: Tu, 15 Oct 20244 20:30:40 -0500
From: "ahamilton" <ahamilton@local>
To: "mmikki" <mmikki@local>
Subject: Badge for New Employee
Message-ID: <29900@rh-mailserver>
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

Hey Mikki, We have a new hire, Alex Patrick Turner, coming in this next week. We need the badge office to print off a new badge for him.The ID should be 001001101000000. Best, Ay
den flag{n3w_3ployees}
```

In the first file it scraped we find the flag we wanted!