

2021

Notes Magazine #05



by Cody Sixteen

1/11/2021

Hello World

Hi ;]

„New Year – New Me” – isn’t it? ;) So below let’s jump directly to 2021 with the quick summary of the ‘noted-articles’ I prepared for you (and for a „future me” of course ;))

Are you ready...?



First part is related to my adventures related to routers, IoT and other „network appliances” we can find online during our pentest activities. Few similar you can find already published on the blog.

Second part is related to the introduction I made for my self when I was learning Rust for a very first time.

Third part is dedicated to the Rosie. We’ll try to talk in the same language.

I hope you’ll find it useful. ;)

Here we go...

Contents

Hello World	1
Automated Route(r) Learning	3
Intro	4
Recreating the Lab for MikroTik	4
Similarity	8
Going down?	10
Lazy Monitoring	13
References	18
Rust In Pieces	19
Intro	20
Environment	20
Read the Manual	21
Example 01	21
Example 02	22
References	23
Rosie Da Stoned	24
Intro	25
Watching the Tower of Babel	26
References	30
Outro	31

Automated Route(r) Learning

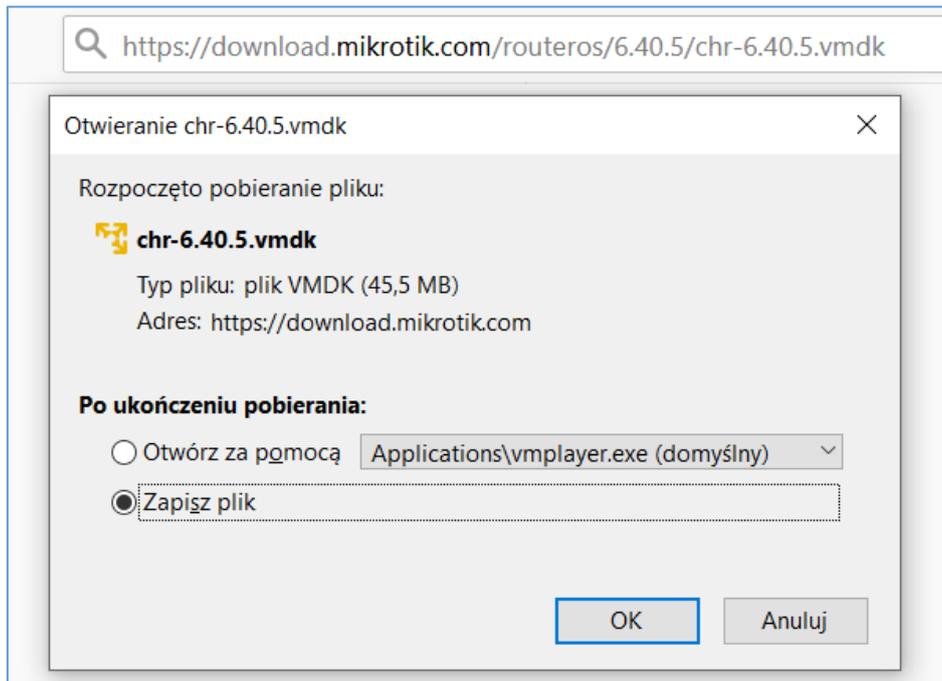


Intro

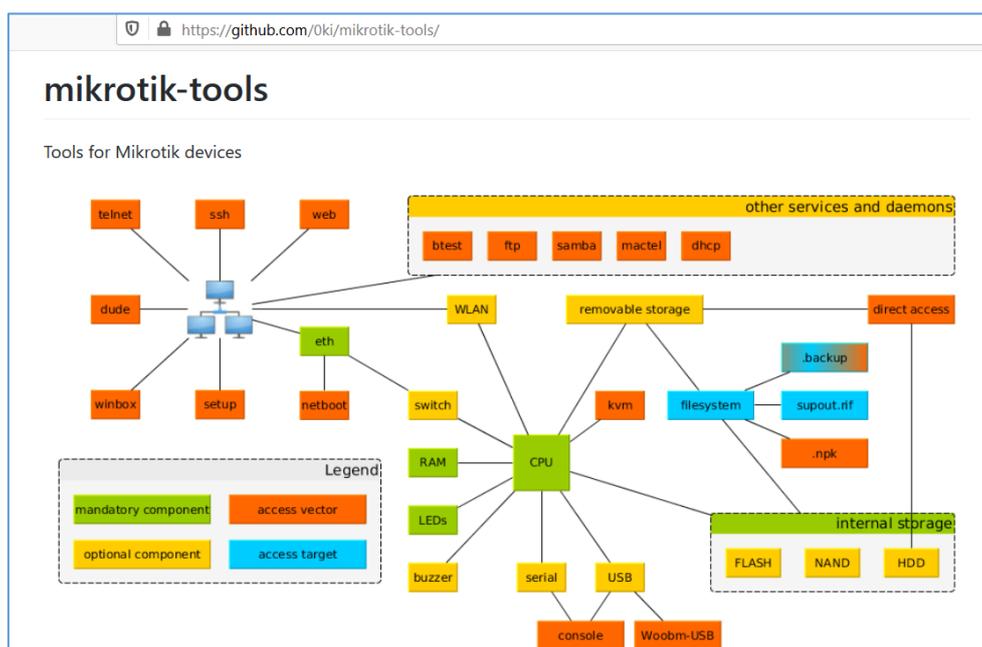
Some time ago I started learning about the routers[1, 2] and other network appliances[3, 4]. Last week I found a very interesting article[5] about „routers security” so I decided to check it. Below we'll try to recreate the „lab” described in the article. It should help us to start doing our own research „related to the routers”. Let's go...

Recreating the Lab for MikroTik

According to the article[5] – we should start here[6]:



Next „few minutes” I spent on preparing my Ubuntu 18 VM as well as the Kali VM (I used 2019 version). When all is ready we can continue here[7] (or here[8]):



(I used the first tool mentioned above.) Your results should be pretty similar to the one presented below – achieving root shell on our MikroTik VM:

```
c@kali: ~/tools/mikrotik-tools/exploit-backup

We got admin@192.168.1.124 with password ''.
Is this correct? (y/N) y

Let's begin.
Verifying version...
Downloading current configuration...
Patching...
Uploading exploit...

* * * * *
Congrats! Jailbreak was (likely) successfull. Device will now reboot.
* * * * *
Linux mode can be accessed via telnet using user 'devel' with admin's password.

Device is now rebooting...
You may opt to install additional utilities to make using the shell easier.
Please note that this will enable telnet service on port 23/tcp,
send YOUR PASSWORD AND USERNAME UNENCRYPTED over the network, and
may REMOVE YOUR ABILITY TO UPDATE software on smaller devices.

Would you like some additional utilities with your jailbreak? (y/N) y
Waiting for device to reboot...
```

Ok... Ok?



;*


```
EMORY:77522f62 db 0CDh ; =
EMORY:77522f63 db 80h ;  
EMORY:77522f64 ; -----
EMORY:77522f64 pop ebx
EMORY:77522f65 mov esi, eax
EMORY:77522f67 cmp eax, 0FFFFFF00h
EMORY:77522 root@ubuntu18:/home/c/tools/mutiny-fuzzer
EMORY:77522
EMORY:77522** Sleeping for 0.500 seconds **
EMORY:77522
EMORY:77522
EMORY:77522Fuzzing with seed 262
EMORY:77522 Sent 350 byte packet
EMORY:77522Logging run number 262
EMORY:77522
UNKNOWN 77522** Sleeping for 0.500 seconds **
New-1
01A E8 41 8F Fuzzing with seed 263
02A 55 89 8F Fuzzing with seed 263
03A 38 00 8F Sent 416 byte packet
04A 01 00 8F Logging run number 263
05A 8D 75 8F
4A 0805494A: ** Sleeping for 0.500 seconds **
it window
-----
d to add st
nitial auto
ng Imports
ng Imports
60: using g
D4: using g
F2: using guessed type int __fastcall sub_80533F2(_DWORD, _DWORD);

c@kali: ~/tools/mikrotik-tools/exploit-backup
Usage: gdbserver [OPTIONS] COMM PROG [ARGS ...]
gdbserver [OPTIONS] --attach COMM PID
gdbserver [OPTIONS] --multi COMM

COMM may either be a tty device (for serial debugging),
HOST:PORT to listen for a TCP connection.

Options:
--debug Enable general debugging output.
--debug-format=opt1[,opt2,...] Specify extra content in debugg
Options:
all
none
timestamp
--remote-debug Enable remote protocol debugging
--version Display version information and
--wrapper WRAPPER -- Run WRAPPER to start new program
--once Exit after the first connection
# ./gdbserver.i686 --attach 192.168.1.124:12345 &(pgrep
Attached; pid = 287
Listening on port 12345
Remote debugging from host 192.168.1.10
```

My next step was to upload the tools (lik gdb, etc) to the target VM. To do that I used ftp just like before. At this stage when all is prepared properly – let’s take a snapshot (just to save some time in the future ;)). Now we can continue...

Similarity

For now we should be here:

```
netstat: /proc/net/tcp6: No such file or directory
tcp        0      0 0.0.0.0:80          0.0.0.0:*           LISTEN     203/www
tcp        0      0 0.0.0.0:2000       0.0.0.0:*           LISTEN     187/bttest
tcp        0      0 0.0.0.0:21        0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:22        0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:23        0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:8728      0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:8729      0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:8291      0.0.0.0:*           LISTEN     180/mproxy
# netstat -antp | grep LIST
tcp        0      0 0.0.0.0:139       0.0.0.0:*           LISTEN     269/smb
tcp        0      0 0.0.0.0:80        0.0.0.0:*           LISTEN     203/www
tcp        0      0 0.0.0.0:2000     0.0.0.0:*           LISTEN     187/bttest
tcp        0      0 0.0.0.0:21        0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:22        0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:23        0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:8728     0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:8729     0.0.0.0:*           LISTEN     192/sermgr
tcp        0      0 0.0.0.0:445      0.0.0.0:*           LISTEN     269/smb
tcp        0      0 0.0.0.0:8291     0.0.0.0:*           LISTEN     180/mproxy
netstat: /proc/net/tcp6: No such file or directory
#
```

Cool. As you can see we have a multiple ‘targets’ ready to „fuzz(me if you can” scenario described in *Notes Magazine 03, page 79*[9]). We can continue below, checking all the binary we’d like to in Ida:

```
C:\Users\c\Downloads\www.elf
: ELF for Intel 386 (Executable)
: 8048000
:/lib/ld-uClibc.so.0
ry `libumsg.so'
ry `libuxml++.so'
ry `libdl.so.0'
ry `libpthread.so.0'
ry `libssl.so.1.0.0'
ry `libcrypto.so.1.0.0'
ry `libc++.so'
ry `libgcc_s.so.1'
ry `libc.so.0'

=====
(mynchronized with EIP)
FF FF 00 0F 85 34 F5 FF 68 2E ..
6A 00 FF B5 B0 EF FF FF 0 8 j j A
```

Next I was recreating the steps from the article[5]:

```
Applications ▾ Places ▾ wireshark ▾ Tue 12:16 ●
any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
smb
No. Time Source Destination Protocol Length Info
54 21.231062550 192.168.1.170 192.168.1.10 SMB 272 Negotiate Protocol Request
c@kali: ~/tools/mikrowww
c@kali:~/tools/mikrowww$ smbclient -L \\192.168.1.10
mkdir failed on directory /var/run/samba/msg.lock: Permission denied
Unable to initialize messaging context
Enter WORKGROUP\c's password:
```

Next (according to the fuzzing notes mentioned above[9]) we can continue with *Mutiny* fuzzer[10] – so we should be somewhere here:

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

Written by James Spadaro (jaspadar@cisco.com) and Lilith Wyatt (liwyatt@cisco.com)

root@kali: /home/c/tools/mutiny-fuzzer
root@kali:/home/c/tools/mutiny-fuzzer# ^C
root@kali:/home/c/tools/mutiny-fuzzer# cp ../smb01.pcap ./
root@kali:/home/c/tools/mutiny-fuzzer# ./mutiny_prep.py smb01.pcap
Processing smb01.pcap...
Which port is the server listening on? (445/36412)
Default 445: 445

Message #0 - Processed 216 bytes outbound
Processed input file smb01.pcap

How many times should a test case causing a crash or error be repeated?
Default 3: 2

When the test case is repeated above, how many seconds should it wait between tests?
Default 5: 3

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

What port should the fuzzer connect to?
Default 445: 445

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

Wrote .fuzzer file: smb01-0.fuzzer

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

Next:

```
c@kali: ~/tools/mikrowww
# netstat -antp | grep LIST
tcp        0      0 0.0.0.0:139          0.0.0.0:*           LISTEN    269/smb
tcp        0      0 0.0.0.0:80          0.0.0.0:*           LISTEN    203/www
tcp        0      0 0.0.0.0:2000        0.0.0.0:*           LISTEN    187/btest
tcp        0      0 0.0.0.0:21          0.0.0.0:*           LISTEN    192/sermgr
tcp        0      0 0.0.0.0:22          0.0.0.0:*           LISTEN    192/sermgr
tcp        0      0 0.0.0.0:23          0.0.0.0:*           LISTEN    192/sermgr
tcp        0      0 0.0.0.0:8728        0.0.0.0:*           LISTEN    192/sermgr
tcp        0      0 0.0.0.0:8729        0.0.0.0:*           LISTEN    192/sermgr
tcp        0      0 0.0.0.0:445         0.0.0.0:*           LISTEN    269/smb
tcp        0      0 0.0.0.0:8291        0.0.0.0:*           LISTEN    180/mproxy

netstat: /proc/net/tcp6: No such file or directory
# pkill smb
# /nova/bin/smb
Main start
Socket: created socket: 15, on :139
Socket: created socket: 16, on :445
Socket: created socket: 17, on :137
Socket: created socket: 18, on :138
NBServ: Reg name on ip: 10.0.2.15

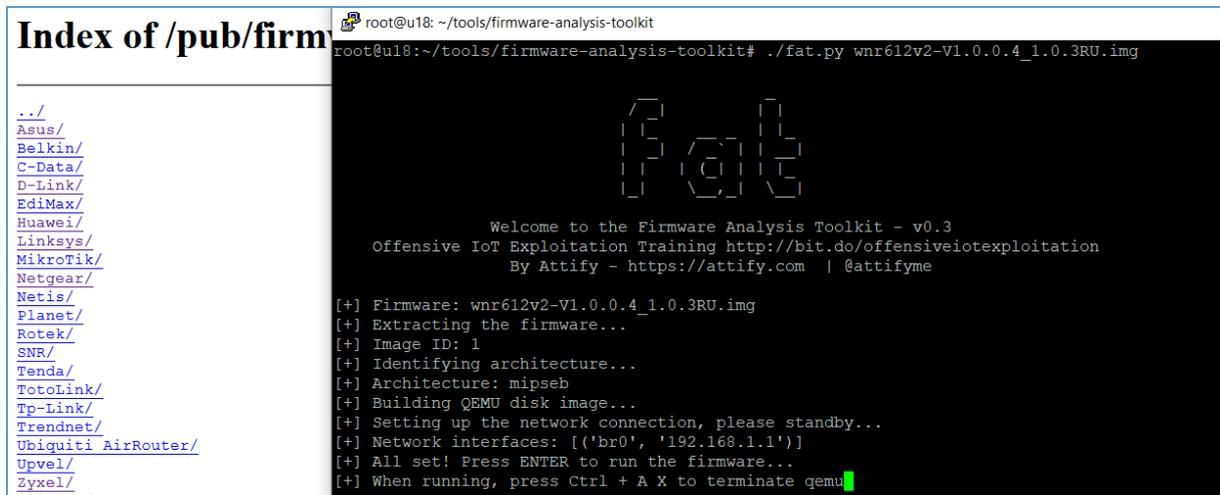
Fuzzing with seed 66
Sent 218 byte packet
Logging run number 66
** Sleeping for 0.500 seconds **
Fuzzing with seed 67
Sent 333 byte packet
Logging run number 67
** Sleeping for 0.500 seconds **
Fuzzing with seed 68
Sent 224 byte packet
Logging run number 68
** Sleeping for 0.500 seconds **
Fuzzing with seed 69
Sent 228 byte packet
```

So far so good. Now I believe the *time* is our 'enemy' so feel free to run the fuzzing process on your own lab to get some crashes and have fun. ;)

Going down?



After a while the idea evolved to something similar I described for Wordpress plugins[9]. When our 'lab' is prepared, everything works fine – what else should we do now to „try to find more bugs“? Well – my first guess was to create some kind of an *automated* environment now. We'll use the same *base* we did before – so you can easily stay with your Ubuntu 18 VM. Let's start here:



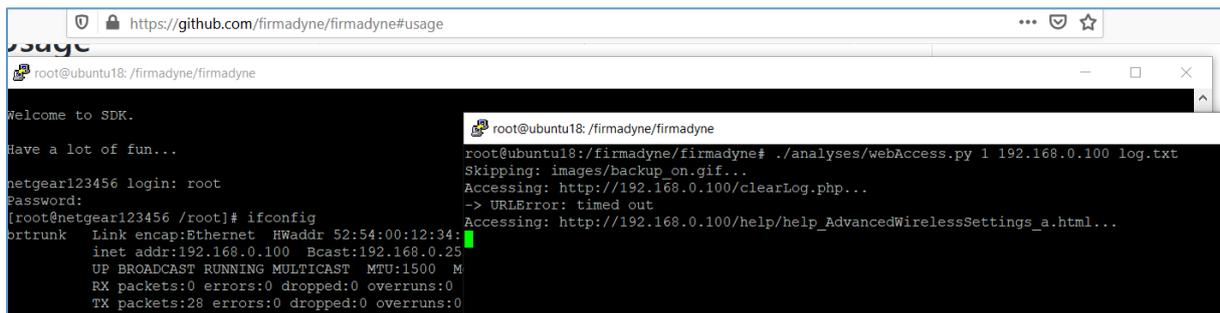
```
root@u18: ~/tools/firmware-analysis-toolkit
root@u18:~/tools/firmware-analysis-toolkit# ./fat.py wnr612v2-V1.0.0.4_1.0.3RU.img

Index of /pub/firm
../
Asus/
Belkin/
C-Data/
D-Link/
EdiMax/
Huawei/
Linksys/
MikroTik/
Netgear/
Netis/
Planet/
RoteK/
SNR/
Tenda/
TotoLink/
Tp-Link/
Trendnet/
Ubiquiti AirRouter/
Upvel/
Zyxel/

Welcome to the Firmware Analysis Toolkit - v0.3
Offensive IoT Exploitation Training http://bit.do/offensiveiotexploitation
By Attify - https://attify.com | @attifyme

[+] Firmware: wnr612v2-V1.0.0.4_1.0.3RU.img
[+] Extracting the firmware...
[+] Image ID: 1
[+] Identifying architecture...
[+] Architecture: mipseb
[+] Building QEMU disk image...
[+] Setting up the network connection, please standby...
[+] Network interfaces: [('br0', '192.168.1.1')]
[+] All set! Press ENTER to run the firmware..
[+] When running, press Ctrl + A X to terminate qemu
```

Finding an 'example firmware of the router' should be the easy part here ;) let's move forward then. As we can see – for this router firmware we can get an IP (it will be the most often error message you'll see during the research: somehow 'firmadyne'-related scripts can not obtain an IP sometime for some appliances – well, „nobody's perfect". ;) Let's jump here:



```
root@ubuntu18: ~/firmadyne/firmadyne
Welcome to SDK.
Have a lot of fun...

netgear123456 login: root
Password:
[root@netgear123456 /root]# ifconfig
brtrunk  Link encap:Ethernet  HWaddr 52:54:00:12:34:
          inet addr:192.168.0.100  Bcast:192.168.0.25
          UP BROADCAST RUNNING MULTICAST  MTU:1500  M
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:28 errors:0 dropped:0 overruns:0
          collisions:0 frame:0

root@ubuntu18: ~/firmadyne/firmadyne
root@ubuntu18:~/firmadyne/firmadyne# ./analyses/webAccess.py 1 192.168.0.100 log.txt
Skipping: images/backup_on.gif...
Accessing: http://192.168.0.100/clearLog.php...
-> URLError: timed out
Accessing: http://192.168.0.100/help/help_AdvancedWirelessSettings_a.html...
```

As you can see script collection[11] is prepared mostly for the same scenario that we're using during our adventures[13]. In the content of the directory we can find some 'example steps' (coded in small scripts) that we'll also try to take when we'll find an applicane „like the one presented on the screen" (ex. during our pentest or CTF), so: nmap, nikto, gobuster/dirb and so on should be your friend at this stage. We can also use *ssllscan* or Metasploit Framework but it depends on you what you'll now like to add to 'your' (or simply: modified) scripts from the *firmadyne* catalog[11]. In my case we are here:

```
if tail and ('.' not in tail or any(tail.endswith(ext) \
for ext in [".htm", ".html", ".cgi", ".asp", ".php",
".bin", ".xml", ".rg"])):
try:
url = urllib.parse.urlunsplit(
("http", cmd.ip, tail, None, None))
print("Accessing: %s..." % url)
req = urllib.request.urlopen(url, timeout=5)
data = req.read()
if b"location.href" in data or b"window.location" i
print("> Redirect")
accessible.append(tail + " (REDIR)")
else:
accessible.append(tail)
except socket.timeout as exc:
print("> Socket Timeout: %s" % exc)
except http.client.IncompleteRead as exc:
data = exc.partial
except urllib.error.HTTPError as exc:
print("> HTTPError: %d" % exc.code)
except urllib.error.URLError as exc:
print("> URLError: %s" % exc.reason)
elif tail:
print("Skipping: %s..." % tail)
with open(cmd.log, "w") as file:
for url in accessible:
```

As you can see I tried to understand the script by reading it ;] So let's continue below:

```
root@u18:~/tools#
root@u18:~/tools# cd ./firmadyne; ./download.sh
Downloading binaries...
Downloading kernel 2.6 (MIPS)...
--2021-01-20 17:26:47-- https://github.com/firmadyne/kernel-v2.6/releases/download/v1.
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/51779291/b67e
KIAIWNJYAX4CSVEH53A%2F20210120%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20210120T1626
22d67157ee3ca64c2f&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=51779291&respon
```

I restarted the *installation* process multiple times (after restoring the snapshot – also multiple times ;)). I was wondering what I do not understand (read as: why this-or-that firmware is not getting an IP). So I landed here:

```
Loading Ethernet module. [GENMAC]
BusyBox v1.11.0 (2011-06-23 15:54:48 IST) multi-call binary
Usage: ifconfig [-a] interface [address]
Configure a network interface
Options:
[[-]broadcast [ADDRESS]] [[-]pointopoint [ADDRESS]]
[netmask ADDRESS] [dstaddr ADDRESS]
[outfill NN] [keepalive NN]
[hw ether|infiniband ADDRESS] [metric NN] [mtu NN]
[[-]trailers] [[-]arp] [[-]allmulti]
[multicast] [[-]promisc] [txqueuelen NN] [[-]dynamic]
[mem_start NN] [io_addr NN] [irq NN]
[up|down] ...
[DONE]
Checking database. [DONE]
```

We'll not fix it here – for more hints what's going on – try this page[\[14\]](#). For now we'll jump to the next section. Here we go...

Lazy Monitoring

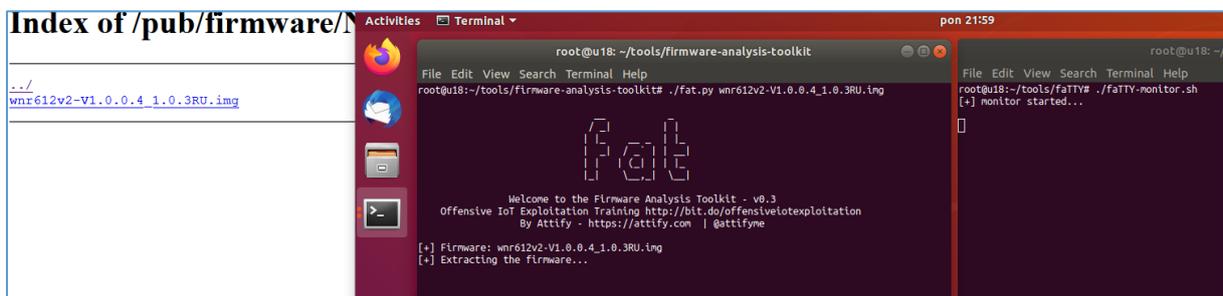
My initial goal was to create a new additional script to:

- loop to check if qemu (so 'our target firmware') is started
- check if firmware started in qemu has a valid IP address (ping it or something like that ;))
- run 'our scan scenarios' (so mentioned nmap, dirbuster(s) and so on... to grab some details about the target host).

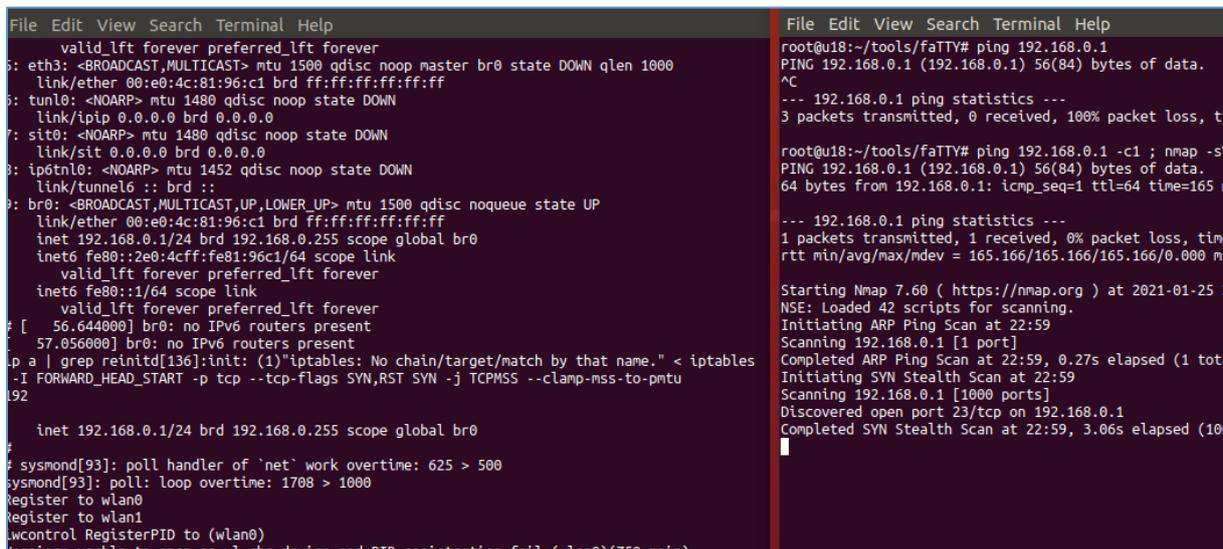
In other window we should run a 'loop' to:

- download 'example firmwares' we're looking for to check/scan/test
- try to *automatically* run it. ;]

So far we should be somewhere here:



Next the firmware I tried started properly so I was able to ping/access it:



Next – sure thing, portscan (`nmap -sV -F -n host -oN logfile`):

```

sys/class/gpio/unexport: nonexistent directory
Completed NSE at 23:00, 0.03s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 23:00
Completed NSE at 23:00, 0.00s elapsed
Nmap scan report for 192.168.0.1
Host is up, received arp-response (0.058s latency).
Scanned at 2021-01-25 22:59:52 CET for 40s
Not shown: 999 closed ports
Reason: 999 resets
PORT STATE SERVICE REASON VERSION
23/tcp open landesk-rc syn-ack ttl 64 LANDesk remote management
MAC Address: 00:E0:4C:81:96:C1 (Realtek Semiconductor)

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at
/.
Nmap done: 1 IP address (1 host up) scanned in 45.18 seconds

rootfs_offset=0x260000, back_linux_offset=0x840000,
0000! [ffffffff '++++']
s to backup linux/rootfs
backup
port 80
rt_ipv4_wan WAN1 start

```

In the meantime I tried to log in to the appliance (via telnet as you can see):

```

0 SW [mtdblock9]
0 SW< [kpsmoused]
0 SW [kworker/0:2]
928 S /bin/sh
932 S -/bin/sh
920 S klogd
0 SW [flush-8:0]
708 S /bin/superd
1032 S /bin/sysmond -f
680 S < watchdog 1000
868 S /bin/ubusd
0 SW [flush-31:0]
2620 S /bin/reinitd
920 S syslogd -L -s 32 -b 2
1040 S udhcpd /var/lan/udhcpd.conf
928 S telnetd
808 S lld2d br0
680 S fwd
884 S reload -k /var/wlscch.conf
708 S iwcontrol wlan0 wlan1
1708 S /bin/login
1784 D timelycheck
1708 S /bin/login
1708 S /bin/login
0 SW [flush-31:8]
0 SW [flush-31:10]
1704 S wps_checker
2440 S boa -d
1708 S /bin/login
848 S dnsmasq -C /var/lan/dnsmasq.conf --clear-on-reload -x
1016 S udhcpd -i eth1 -W 1 -p /var/wan/1/run/con-app.pid -a
1708 S /bin/login
1712 S /bin/login
924 R ps

Reason: 999 closed ports
PORT STATE SERVICE REASON VERSION
23/tcp open landesk-rc syn-ack ttl 64 LANDesk remote ma
MAC Address: 00:E0:4C:81:96:C1 (Realtek Semiconductor)

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect
/.
Nmap done: 1 IP address (1 host up) scanned in 45.18 sec
Raw packets sent: 1114 (48.984KB) | Rcvd: 11
root@u18:~/tools/faTTY# telnet 192.168.0.1
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '^]'.

Username : root
Password :
cli[2554]: Invalid username or password. Login failed.
Username : admin
Password :
cli[2554]: Invalid username or password. Login failed.
Username : admin
Password :
cli[2554]: admin; TELNET; LAN; Login

# id
/bin/sh: id: not found
# uname -a
/bin/sh: uname: not found
# ls
bin          firmadyne  jffs2      mnt          sys
dev          home       lib        proc         tmp
etc          init      lost+found root          usr

```

(At this stage you can try to create some quick script to bruteforce telnet service or simply check few 'default passwords' like root, admin, and so on... Next:

```

root@u18: ~/tools/firmware-analysis-toolkit
root@u18: ~/tools/faTTY

0 SW [mtdblock9]
0 SW [mtdblock10]
0 SW< [kpsmoused]
0 SW [kworker/0:2]
928 S /bin/sh
932 S -/bin/sh
920 S klogd
0 SW [flush-8:0]
708 S /bin/superd
1032 S /bin/sysmond -f
680 S < watchdog 1000
868 S /bin/ubusd
0 SW [flush-31:0]
2620 S /bin/reinitd
920 S syslogd -L -s 32 -b 2
1040 S udhcpd /var/lan/udhcpd.conf
928 S telnetd
808 S lld2d br0
680 S fwd
884 S reload -k /var/wlscch.conf
708 S iwcontrol wlan0 wlan1
1708 S /bin/login
1784 D timelycheck
1708 S /bin/login

[+] scanner module - started
23/tcp open telnet syn-ack ttl 64
53/tcp open domain syn-ack ttl 64 dnsmasq 2.76-14-g466d8f
80/tcp open http syn-ack ttl 64 Boa httpd

-----
DIRB v2.22
By The Dark Raver
-----

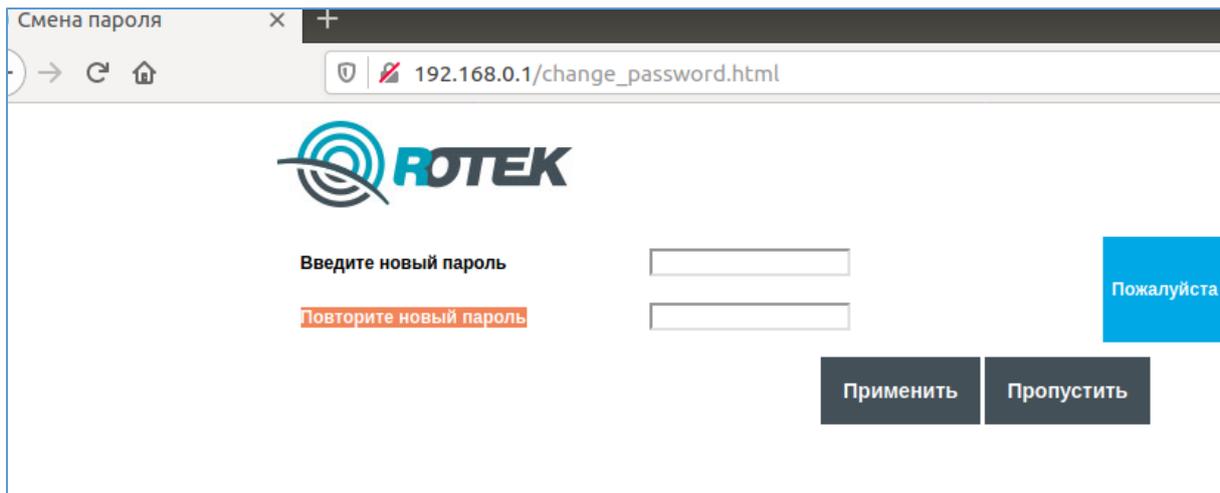
OUTPUT_FILE: dirb.tmp
START_TIME: Mon Jan 25 23:21:00 2021
URL_BASE: http://192.168.0.1/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

*** Generating Wordlist...

```

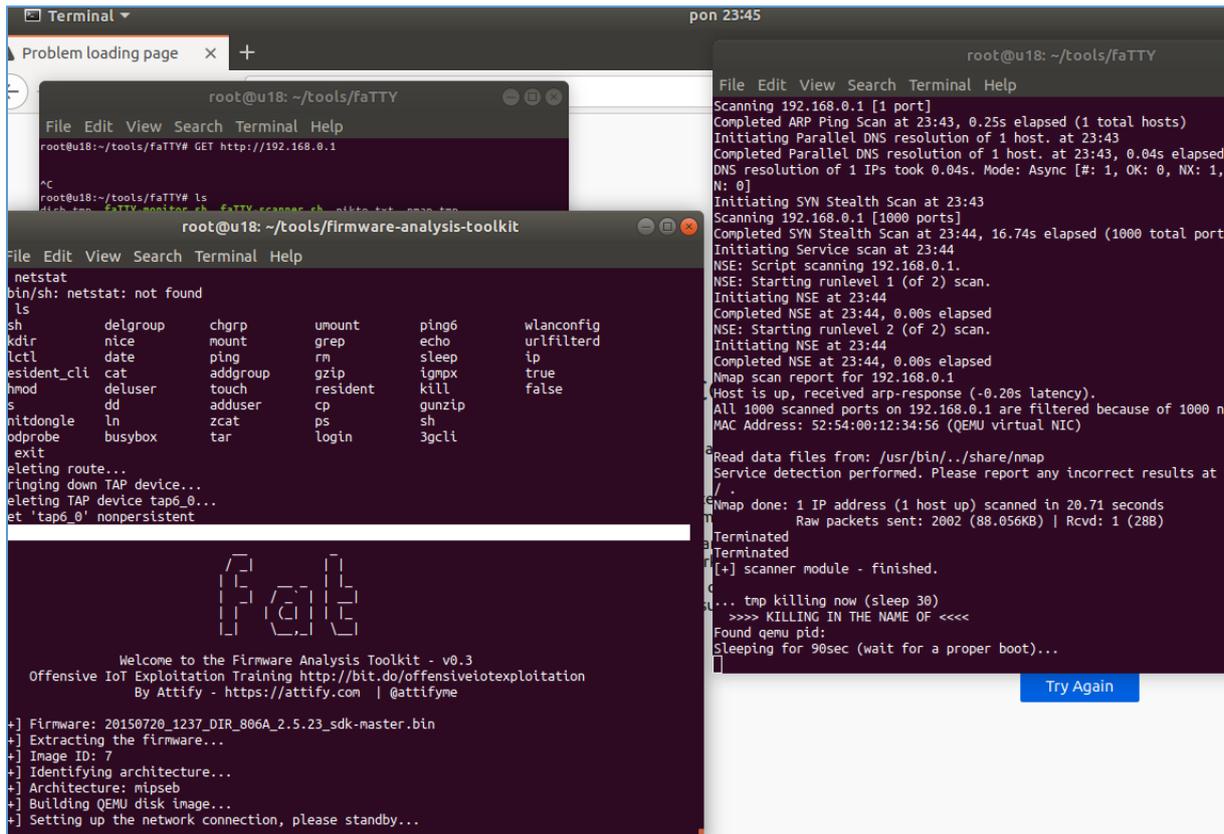
Feel free to use any other tool to enumerate WWW. I used *dirb*. Next:



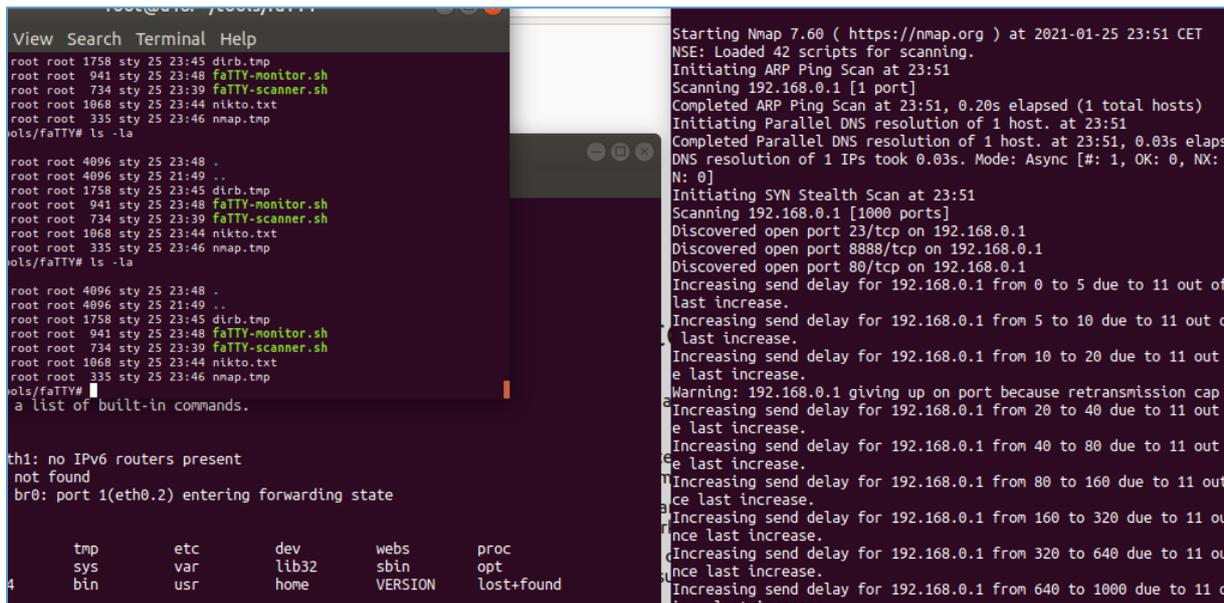
So far, so good. As you can see below I downloaded „few more” ;) firmwares. Next stage was to run *fat.py* script in a loop (remember to add *./reset.py* to the chain):

```
DIR880A1_FW108WwB03.bin
DIR890LA1_FW110b07.bin
fat.config
fat.py
firmadyne
FW_WDRt-731U_V2.0.1.5_EN_PLA01.bin
'FW-WNRT617v2_v3.16.5(130508).bin'
'Keenetic-Ultra-V2.06(AAGJ.1)C2.bin'
LICENSE
N300-1.1.0.32_101.img
'netis(MW5230_EU)-V2.2.39550.bin'
'netis(WF2409D)-V1.4.37280.trx'
'netis(WF2409E_RU)-V2.6.40535.bin'
'netis(WF2419E)-V2.2.36411.bin'
OLDFW
qemu-builds
README.md
reset.py
ROTEK-RX22101-V1.23.87.4-UFN.bin
setup.sh
TEW651BRV2_FW204B01.bin
US_FH456V2.0BR_V11.13.01.20_multi_TDEGYH01.bin
US_N301V1.0V1.0BR_V5.07.64.4_ru_RU01.bin
V5.07.46_en.bin
'WNAP320 Firmware Version 2.0.3.zip'
wnr2200-100-V1.0.2.24.img
wnr612v2-V1.0.0.4_1.0.3RU.img
root@u18:~/tools/firmware-analysis-toolkit# for i in `ls *.bin` ; do ./fat.py $i ; done
```

After a while:



Checking another and another firmware and collecting logs:



...when suddenly in the console I saw:

```
49.112000] Call Trace:
49.116000]
49.120000]
49.120000] Code: 00001821 10000028 00001021
49.124000] 90a30000 10400022 24e7ffff 14430024 00431023
49.584000] do_page_fault() #2: sending SIGSEGV to index.cgi for invalid read access from
49.584000] 00000000 (epc == 2adea5e8, ra == 00409d7c)
49.600000] Cpu 0
49.604000] $ 0 : 00000000 1000a401 00000000 2adcb000
49.604000] $ 4 : 00000000 0040bf18 00000021 00000008
49.608000] $ 8 : 2adcfb20 2adcc720 00000001 00000000
49.616000] $12 : 00000007 00000800 00000400 00409d7c
49.620000] $16 : 00410000 000001b9 0040b798 7ffb445c
49.620000] $20 : 00401c50 2ade9770 7ffb42b8 0000003d
49.624000] $24 : 00000060 2adea5d0
49.624000] $28 : 2ad6e010 7ffb3500 7f1f88c8 00409d7c
49.632000] Hi : 000001d2
49.632000] Lo : 000544ee
49.632000] epc : 2adea5e8 0x2adea5e8
49.640000] Not tainted
49.640000] ra : 00409d7c 0x409d7c
49.648000] Status: 0000a413 USER EXL IE
49.648000] Cause : 10000008
49.652000] BadVA : 00000000
49.652000] PrId : 00019300 (MIPS 24Kc)
49.656000] Modules linked in:
49.656000] Process index.cgi (pid: 978, threadinfo=8f27e000, task=8f116590, tls=00000000)
49.664000] Stack : 7ffb3d40 2ad539fc 2f6c6962 2f6c6962 632e736f 2e3000fc 00424050 2f6c6962
49.668000] 6a616e73 736f6e2e 736f2e34 006c6962 632e736f 2e3000fc 2f6c6962 2f6c6962
49.680000] 6763635f 732e736f 2e310062 2f6c6962 6763635f 732e736f 2e310062 2f6c6962
49.688000] 632e736f 2e3000fc 2f6c6962 2f6c6962 6d2e736f 2e3000fc 2f6c6962 2f6c6962
49.696000] 6763635f 732e736f 2e310062 2f6c6962 632e736f 2e3000fc 2f6c6962 2f6c6962
49.700000]
```

Well ;> I believe now you're ready to extend the *fat.py* script(s) and create your own 'small fuzzer' to find some new bugs in your routers/IoT firmwares. All described links and resources you'll find below.

In case of any questions – you'll know how to find me.

Have fun ! ;)

References

Below you'll find the list of links/resources I found interesting:

- 1- <https://code610.blogspot.com/2017/04/learning-routers.html>
- 2- <https://code610.blogspot.com/2018/11/learning-routers-part-2.html>
- 3- <https://code610.blogspot.com/2018/12/reading-firmware-fortigate-vm.html>
- 4- <https://code610.blogspot.com/2018/12/reading-firmware-foscam.html>
- 5- <https://medium.com/@maxi./finding-and-exploiting-cve-2018-7445-f3103f163cc1>
- 6- <https://download.mikrotik.com/routeros/6.40.5/chr-6.40.5.vmdk>
- 7- <https://github.com/0ki/mikrotik-tools/>
- 8- <https://github.com/tenable/routeros/tree/master/poc/bytheway>
- 9- <https://code610.blogspot.com/p/notes-magazine.html>
- 10- <https://github.com/Cisco-Talos/mutiny-fuzzer>
- 11- <https://github.com/firmadyne/firmadyne>
- 12- <https://github.com/attify/firmware-analysis-toolkit>
- 13- <https://code610.blogspot.com/p/mini-arts.html>
- 14- <https://github.com/firmadyne/firmadyne/issues>

Rust In Pieces

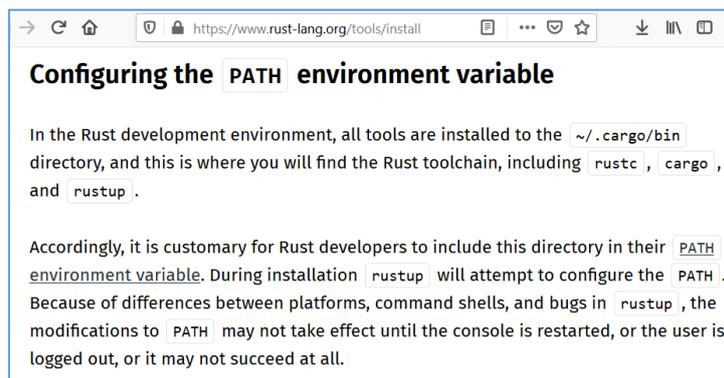


Intro

Thanks to one of the topics mentioned in the video series called ‘Random Topics’[\[1\]](#) once upon a time I decided to learn about the new language – Rust. Below we’ll try to understand few basic rules of the language as well as the syntax (but keep in mind that similar to the [\[2\]](#) I will try to stick strictly with the documentation[\[3, 4\]](#). As there is no point to rewrite the manual here – feel free to study it in your free time ;)). For now – we’ll start here...

Environment

According to the docs[\[3, 4\]](#) and few other resources available online I decided to prepare an environment on Ubuntu 18 VM. Next we’ll need to install few additional packages, see below:



You know I like to ‘try harder’[\[5\]](#) ;) so I decided to install all of the mentioned apps manually (but feel free to use the *curl* command mentioned in the documentation[\[3\]](#)). In my case ‘the basic environment’ looks like this:

```
root@ubuntu18:~# uname -a ; lsb_release -a
Linux ubuntu18 5.3.0-28-generic #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
No LSB modules are available.
Distributor ID: Ubuntu
Description:   Ubuntu 18.04.4 LTS
Release:      18.04
Codename:     bionic
root@ubuntu18:~# apt-get install cargo rustc rustup -y
Reading package lists... Done
Building dependency tree
Reading state information... Done

No apt package "rustup", but there is a snap with that name.
Try "snap install rustup"

E: Unable to locate package rustup
root@ubuntu18:~# snap install rustup
snap "rustup" is already installed, see 'snap help refresh'
root@ubuntu18:~#
```

If everything is ok so far and installation finished successfully we can move forward to the manual section. Ready...? ;]

Read the Manual

In my case, learning „new language” is done (ok... some about 50% of it ;) by finding and/or understanding the analogy to the other language(s) I „already know” (or at least *understand*).

The „other 50%” is done by reading the documentation(s), resources available online and by searching (through mentioned above) for some (maybe) „similar cases” that I would like to do/solve/try (using that ‘new language’).

So that’s why most of this section is related to few „examples” I found in the docs[3, 4]. Here we go...

Example 01

First of all I decided to create a ‘default „simple program”’ – so, yep, we’ll start from „HelloWorld”(.rs for the Rust’s extension). We are here in the Ubuntu (console):

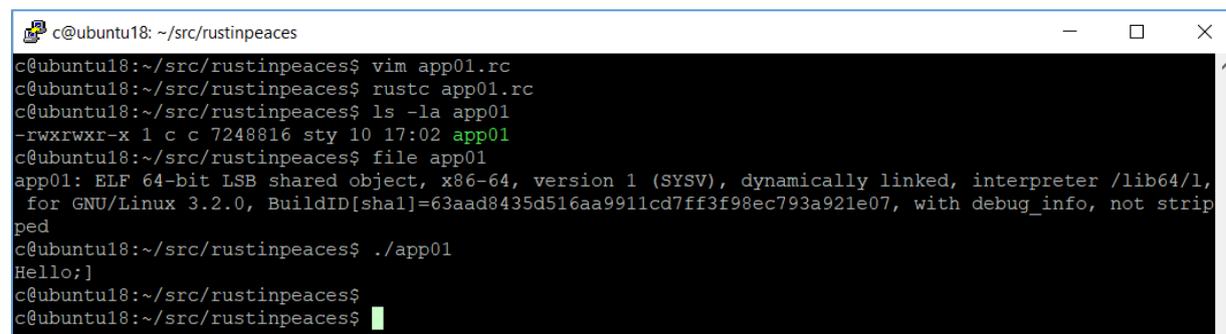


```
c@ubuntu18: ~/src/rustinpeaces
fn main() {
    println!("Hello;]");
}
```

That’s it! Save your file in the favourite editor and go back to the console. According to the docs[6] now we need to compile our source code using (*man* for) *rustc* compiler:

NAME
rustc - The Rust compiler
SYNOPSIS
rustc [OPTIONS] INPUT
DESCRIPTION
This program is a compiler for the Rust language, available at https://www.rust-lang.org .
OPTIONS
-h, --help
Display the help message.

So far, we should be here:



```
c@ubuntu18: ~/src/rustinpeaces
c@ubuntu18:~/src/rustinpeaces$ vim app01.rc
c@ubuntu18:~/src/rustinpeaces$ rustc app01.rc
c@ubuntu18:~/src/rustinpeaces$ ls -la app01
-rwxrwxr-x 1 c c 7248816 sty 10 17:02 app01
c@ubuntu18:~/src/rustinpeaces$ file app01
app01: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=63aad8435d516aa9911cd7ff3f98ec793a921e07, with debug_info, not stripped
c@ubuntu18:~/src/rustinpeaces$ ./app01
Hello;]
c@ubuntu18:~/src/rustinpeaces$
c@ubuntu18:~/src/rustinpeaces$
```

Great! Let’s move forward.

Example 02

I like to read the documentation prepared „by the vendor”. Reason(s?): beside learning „how things works” we can (also) try to *read between the lines* and (for example) find some bugs in the protocol design... but to not to go so far to (the past with DNS „by design bugs”;) *future* – let’s continue with our „simple example 02”. Here we go...

This time my goal was read the documentation to understand how can I create a simple app that will:

- say hello
- get user’s name
- say hello <to-the-user’s-name>.

Simple, isn’t it? ;) Yes it is. It’s even pretty similar to the ‘scenarios’ for multiple *crackmes* available here[7]. (Un)fortunately I wasn’t able to find any example ‘crackme’[7] prepared for the (or „ created in the”) Rust language – so I decided to create a simple one for you. Quick results - below. ;)

(Full „source” of the challenge presented below you’ll find on my github, somewhere between the other ‘notes’ files[8];)). Let’s say we should be here:

```
File Edit View Search Terminal Help
root@ubuntu18:/home/c/src/rustinpeaces/ex03# vim ex04.rs
root@ubuntu18:/home/c/src/rustinpeaces/ex03# head -n 4 ex04.rs
use std::io;

fn main() {
    let mut input = String::new();
root@ubuntu18:/home/c/src/rustinpeaces/ex03# ./ex04
What is the title?
asdfgh
I'm not so sure ;Z
root@ubuntu18:/home/c/src/rustinpeaces/ex03#
```

More details[9]:

```
root@ubuntu18:/home/c/src/rustinpeaces/ex03# base64 ex04 > crackme.ex04.txt
root@ubuntu18:/home/c/src/rustinpeaces/ex03# du -h crackme.ex04.txt
9,4M    crackme.ex04.txt
root@ubuntu18:/home/c/src/rustinpeaces/ex03# md5sum crackme.ex04.txt
36d658d40b6c7d76fa56912d88354208  crackme.ex04.txt
root@ubuntu18:/home/c/src/rustinpeaces/ex03#
```

Good luck! ;)

References

Below you'll find the list of links/resources I found interesting:

- 1 - https://www.youtube.com/watch?v=WaASAO3_WsY&ab_channel=GynvaelColdwind
- 2 - <https://www.rust-lang.org/learn>
- 3 - <https://www.rust-lang.org/tools/install>
- 4 - <https://doc.rust-lang.org/rust-by-example/meta/doc.html>
- 5 - <https://code610.blogspot.com/2020/02/trying-harder.html>
- 6 - <https://doc.rust-lang.org/book/ch01-02-hello-world.html>
- 7 - <https://crackmes.one>
- 8 - <https://github.com/c610/free>
- 9 - <https://github.com/c610/free/blob/master/crackme.ex04.txt>

Rosie Da Stoned



Intro

Maybe you know, maybe you don't – from time to time I like to take a break from the computer and (for example) read a book about some mythology, archeology or something else like the wor(l)ds created by Stephen King[1] or Neil Gaiman[2]. During one of those breaks (ok, to be honest: it was „many years ago“ and I didn't even have a computer yet ;)) I found some story in one book in my house about so called Rosetta Stone[3].

After few years (during one of my 'breaks' ;)) I was wondering „maybe it'll be a good idea to create some software to *automatically* read *languageA* and translate it to *languageB*“... In the meantime - days goes by, I'm doing another pentest project for another company when suddenly I found an article[4, 5] that „someone already did it“[6]. ;>

I decided that there is no time for the break and that's how I started preparing „my own small Rosetta Stone“. ;]

Here we go...

Watching the Tower of Babel

To continue with this *translator* idea we'll prepare a small environment. All examples presented in this text were prepared on Ubuntu 18 and Python language. If there are any other libs/resources needed to proceed – I'll mention it in the text below.

Let's start from the stage where we already have a picture of „some word” we'd like to translate. To proceed I found an example picture with the word. Then I asked my self: where should I start? What's next?

```
# apt install libopencv-dev python3-opencv tesseract-ocr -y; pip3 install Image pytesseract
```

In my case there was a 500MB update – so after a while we should be somewhere here:

```
root@ubuntu18:/home/c/src/rosie# wget https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/Wikipedia-word.svg/1200px-Wikipedia-word.svg.png
--2021-02-06 13:02:13-- https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/Wikipedia-word.svg/1200px-Wikipedia-word.svg.png
Resolving upload.wikimedia.org (upload.wikimedia.org)... 91.198.174.208, 2620:0:862:edla::2:b
Connecting to upload.wikimedia.org (upload.wikimedia.org)|91.198.174.208|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18006 (18K) [image/png]
Saving to: '1200px-Wikipedia-word.svg.png'

1200px-Wikipedia-word.svg.png 100%[=====>] 17,58K --.-KB/s in 0,02s

2021-02-06 13:02:13 (716 KB/s) - '1200px-Wikipedia-word.svg.png' saved [18006/18006]

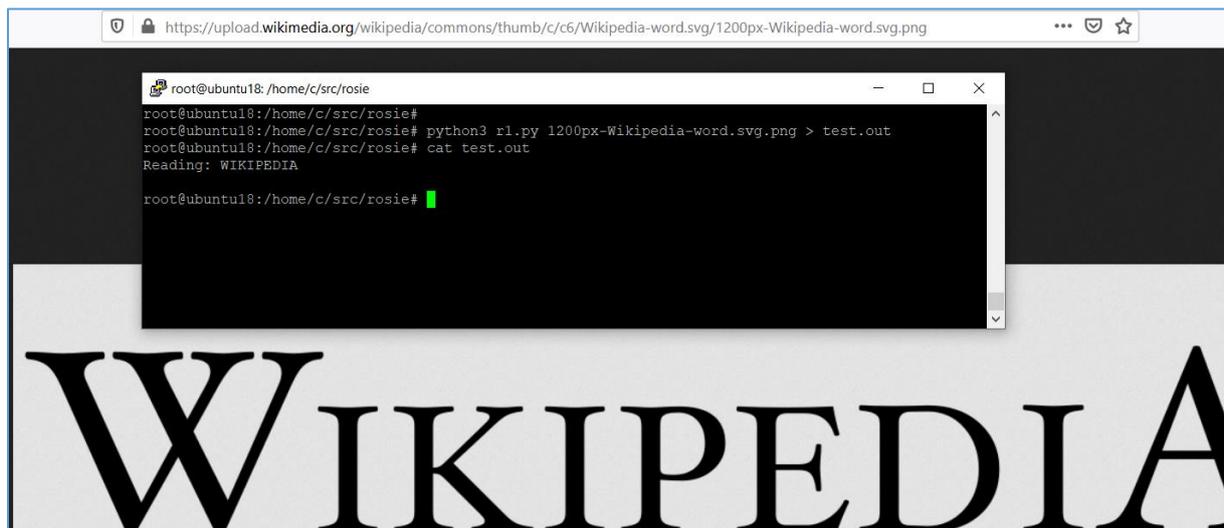
root@ubuntu18:/home/c/src/rosie# cat r1.py
from PIL import Image
import pytesseract
import sys

read_me = sys.argv[1]

out_txt = pytesseract.image_to_string(Image.open(read_me))

print ("Reading: %s" % (out_txt))
```

With this very basic example we should be able to read a 'sample image' (we grabbed from Wikipedia). Let's try:

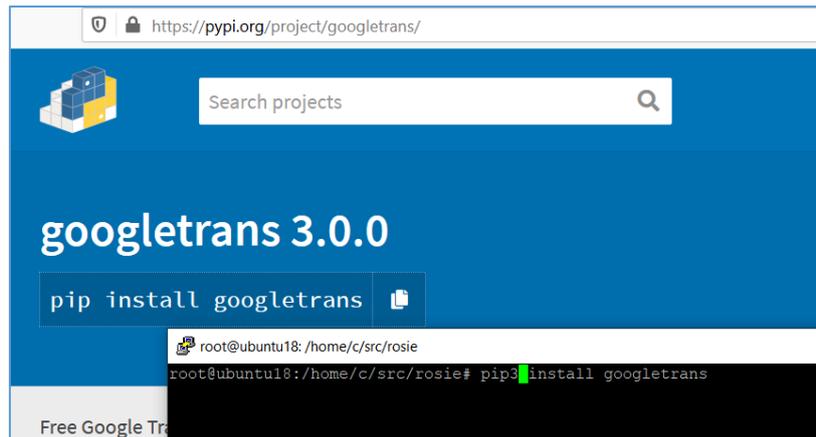


So far – looks good ;) Let's continue then.

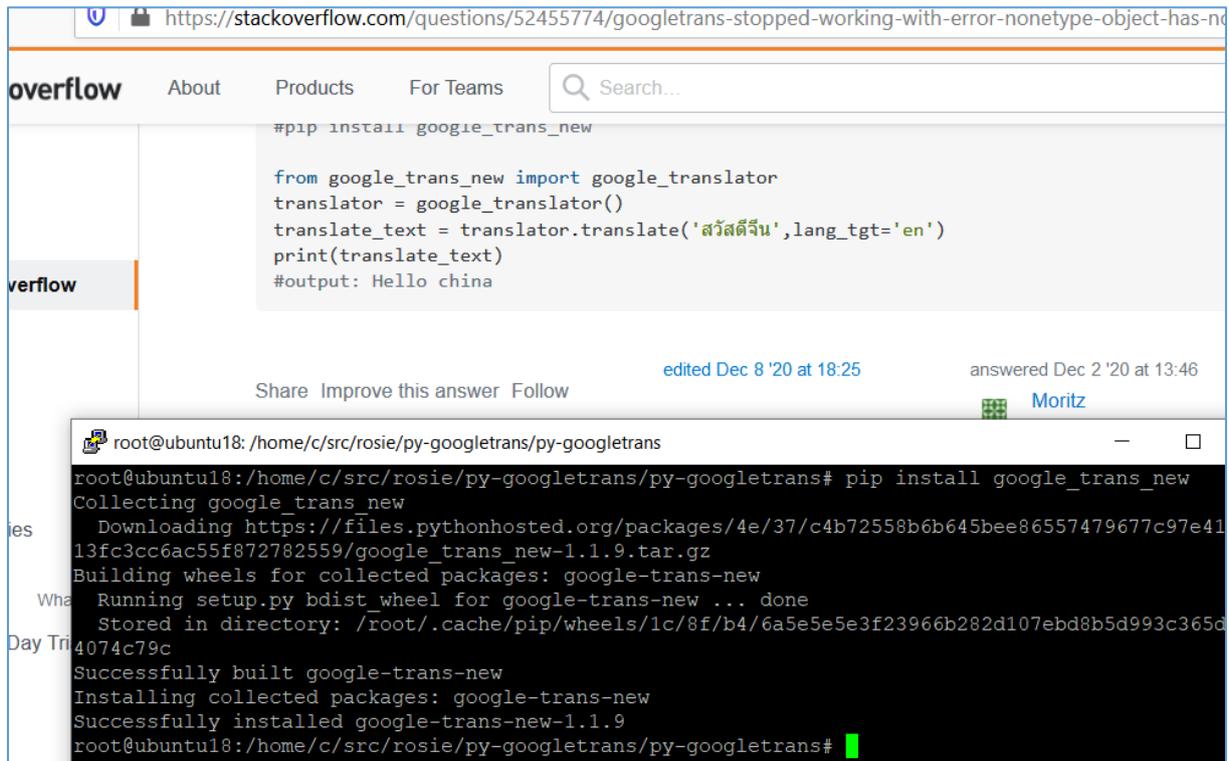
For now our new goal looks like this:

- open and read image
- save output to txt file
- connect to translator to check found word saved in txt file
- save translated word in new txt output file.

Let's try here:



When all is ready we can check initial example – but after we'll fix that nasty error from the console:



And in case you're wondering why it's not working with python3... ;)

I was trying `googletrans` and it was working quite well. Since this morning I started getting below error. I went through multiple posts from stackoverflow and other sites and found probably my ip is

```
root@ubuntu18: /home/c/src/rosie/py-googletrans/py-googletrans
root@ubuntu18:/home/c/src/rosie/py-googletrans/py-googletrans# python3 r3.py
Traceback (most recent call last):
  File "r3.py", line 1, in <module>
    from google_trans_new import google_translator
ModuleNotFoundError: No module named 'google_trans_new'
What root@ubuntu18:/home/c/src/rosie/py-googletrans/py-googletrans# pip3 install google_trans_new
Collecting google_trans_new
  Downloading https://files.pythonhosted.org/packages/f9/7b/9f136106dc5824dc98185c97991d3cd9b53e70a197154dd49f7b899128f6/google_trans_new-1.1.9-py3-none-any.whl
Installing collected packages: google-trans-new
Successfully installed google-trans-new-1.1.9
data = self._translate(text, dest, src)
File "/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/goog
token = self.token_acquirer.do(text)
File "/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/goog
```

Yep ;D So one more time:

```
root@ubuntu18: /home/c/src/rosie/py-googletrans/py-googletrans
root@ubuntu18:/home/c/src/rosie/py-googletrans/py-googletrans# python3 r3.py
Traceback (most recent call last):
  File "r3.py", line 1, in <module>
    from google_trans_new import google_translator
ModuleNotFoundError: No module named 'google_trans_new'
root@ubuntu18:/home/c/src/rosie/py-googletrans/py-googletrans# pip3 install googl
Collecting google_trans_new
  Downloading https://files.pythonhosted.org/packages/f9/7b/9f136106dc5824dc98185c97991d3cd9b53e70a197154dd49f7b899128f6/google_trans_new-1.1.9-py3-none-any.whl
Installing collected packages: google-trans-new
Successfully installed google-trans-new-1.1.9
root@ubuntu18:/home/c/src/rosie/py-googletrans/py-googletrans# python3 r3.py
Hello china
root@ubuntu18:/home/c/src/rosie/py-googletrans/py-googletrans#
```

That's great!

It looks that now we are:

- able to read text from the image
- translate it to some other language (in our case let's stay with english).

So far, so good – continuing then. Now it's time to fuze both small scripts: reader and translator. We should be somewhere here:

```
#!/usr/bin/env python
# rosie.py - small script to understand images in the stones ;)
#
# 06.02.2021 @ 13:31
#
from PIL import Image
import pytesseract, sys
from google_trans_new import google_translator

translator = google_translator()

# STEPS:
# 1] read image file
# 2] save text to file1
# 3] each line in file1 parse with translator()
# 4] save output to file2
#
# 1]
read_me = sys.argv[1]
out_txt = pytesseract.image_to_string(Image.open(read_me))

# 2]
fp1 = open('file1.txt', 'w')
fp1.write(out_txt)
fp1.close()
print("Image parsed, output text in file1.txt.")

# 3]
fp1 = open('file1.txt', 'r')
lines = fp1.read()

translate_text = translator.translate(lines, lang_tgt='pl')
print("Translation:\n----")
print(translate_text)
print("----\nEND.")
```

Quick results for the *History Channel* logo (found on Wikipedia):

```
HTTP request sent, awaiting response... 200 OK
Length: 467054 (456K) [image/png]
Saving to: '1200px-History_Logo.svg.png'

1200px-History_Logo.svg 100%[=====>] 456,11K  322KB/s  i
2021-02-06 13:45:13 (322 KB/s) - '1200px-History_Logo.svg.png' saved [467054/46

root@ubuntu18:/home/c/src/rosie# python3 rosie.py 1200px-History_Logo.svg.png
Image parsed, output text in file1.txt.
Translation:
----
HISTORIA
----
END.
root@ubuntu18:/home/c/src/rosie# █
```

Cool! What's next? I don't know... maybe it's time to buy a nice hat and join the "Raiders of The Lost Ark"...? but maybe you'd like to check the „Voynic Manuscript” first...? ;)

„Have fun & Good luck!” ;]

Cheers

References

Below you'll find the list of links/resources I found interesting:

- 1 - https://en.wikipedia.org/wiki/Stephen_King
- 2 - https://en.wikipedia.org/wiki/Neil_Gaiman
- 3 - https://en.wikipedia.org/wiki/Rosetta_Stone
- 4 - <https://www.bbc.com/news/technology-53420320>
- 5 - <https://mashable.com/article/google-ai-ancient-translations-fabircius/>
- 6 - <https://wired.me/technology/artificial-intelligence/google-translate-for-egyptian-hieroglyphics/>
- 7 - <https://code610.blogspot.com/p/notes-magazine.html>

Outro



Comments/questions – you'll know [how to find me](#).

Thank you. I appreciate it.

[Cheers](#)