Hack the Box Netmon

OS: Windows

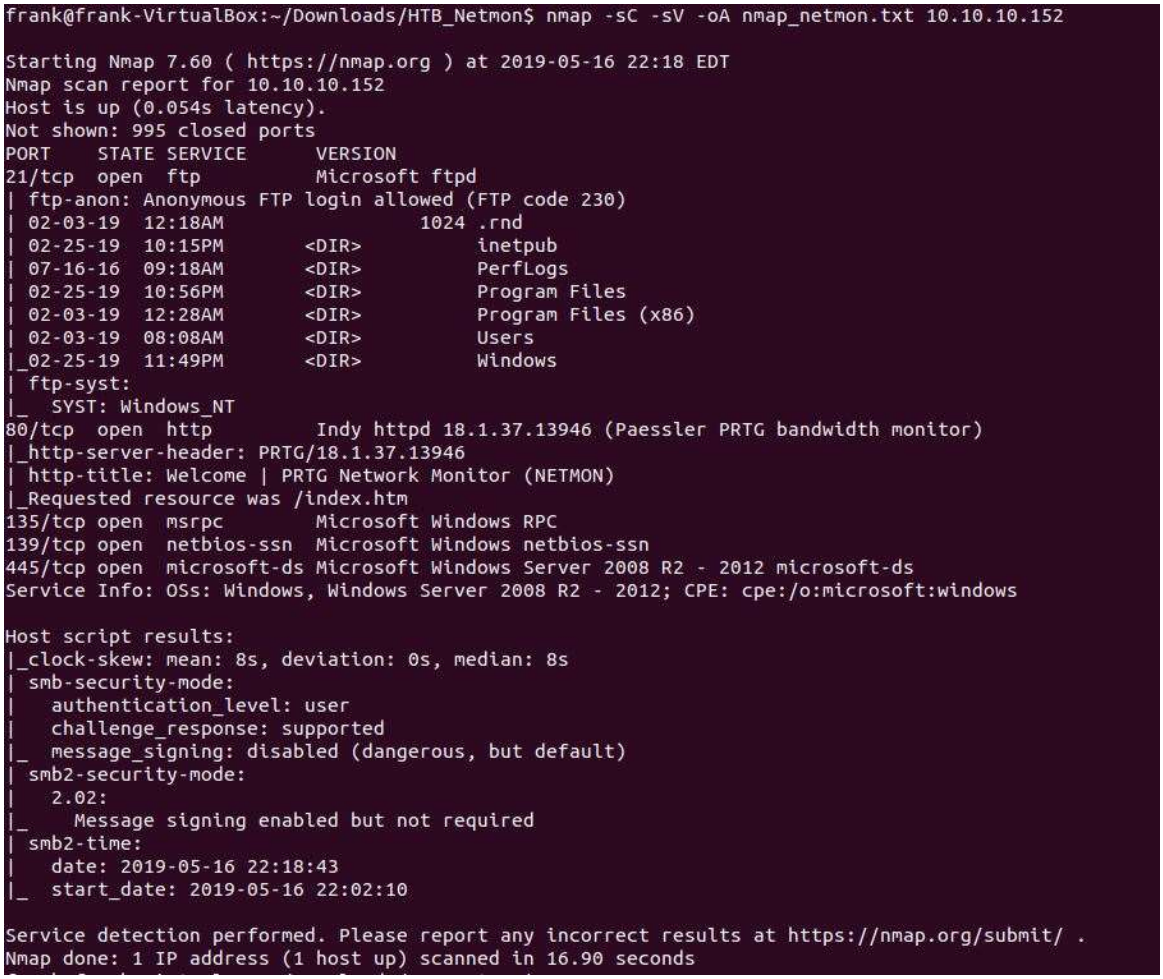
Difficulty: 3.1/10

Written by: n0tac0p

This was my first box I attempted/owned on Hack the Box. I required some subtle hints, and although this is arguably the easiest box on HTB, I had a lot of fun learning with it. As my first box and writeup I hope to make the solution path I chose as simple to follow as possible. With that, lets jump write in.

Recon & Enumeration of Content

Let's start with the common nmap (network map) command to map out what ports are in use on the server. **IP is 10.10.10.152**



```
frank@frank-VirtualBox:~/Downloads/HTB_Netmon$ nmap -sC -sV -oA nmap_netmon.txt 10.10.10.152

Starting Nmap 7.60 ( https://nmap.org ) at 2019-05-16 22:18 EDT
Nmap scan report for 10.10.10.152
Host is up (0.054s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE      VERSION
21/tcp  open  ftp          Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 02-03-19  12:18AM                1024 .rnd
| 02-25-19  10:15PM       <DIR>          inetpub
| 07-16-16  09:18AM       <DIR>          PerfLogs
| 02-25-19  10:56PM       <DIR>          Program Files
| 02-03-19  12:28AM       <DIR>          Program Files (x86)
| 02-03-19  08:08AM       <DIR>          Users
|_02-25-19  11:49PM       <DIR>          Windows
| ftp-syst:
|_  SYST: Windows_NT
80/tcp  open  http         Indy httpd 18.1.37.13946 (Paessler PRTG bandwidth monitor)
|_http-server-header: PRTG/18.1.37.13946
| http-title: Welcome | PRTG Network Monitor (NETMON)
|_Requested resource was /index.htm
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 8s, deviation: 0s, median: 8s
| smb-security-mode:
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-05-16 22:18:43
|_  start_date: 2019-05-16 22:02:10

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.90 seconds
```

Command used: nmap -sC -sV -oA nmap_netmon.txt 10.10.10.152

I found this command in IppSec's youtube videos (link)

-sC: Use default scripts

-sV: For open ports, find out service and version information

-oA: Output file base name

Output:

21: FTP Server of the windows filesystem, *with anonymous login available*

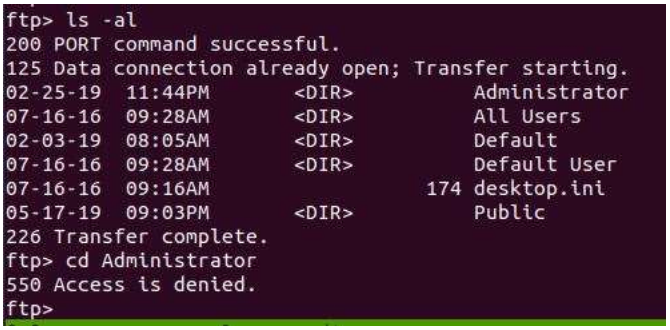
80: PRTG Network Monitor, some type of web application (version 18.1.37.13946)

135, 139, 445: Common Windows Services (did not investigate that much)

All based on Windows Server 2008 V2. Given there is a web application, I assumed the vulnerability most likely lied there.

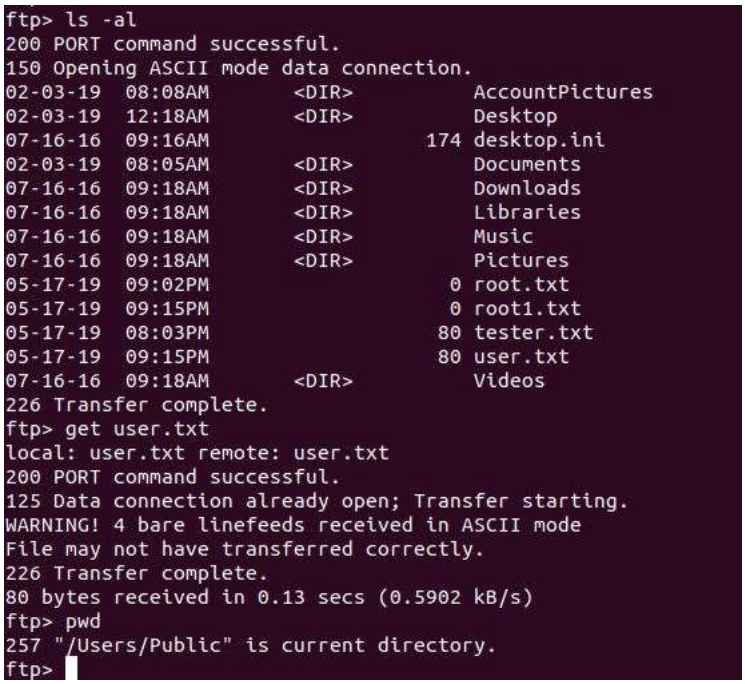
Knowing that the user hash can be found on the desktop of the user, and root hash can be found on the desktop of the admin, my first step was to investigate and enumerate what access I had on the ftp server. Since the server allows for anonymous logins, getting on the server was easy. (Since this is my first box, I am not sure if anonymous logins exist on all windows boxes, or what)

I tried the simplest things I could think of to start with. I checked what users existed, and which ones I could access. Sadly, we do not have access to the Administrator folder.

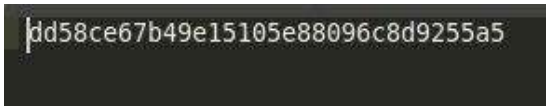


Very important for this box that we use ls -al and NOT ls by itself. This will come into play later.

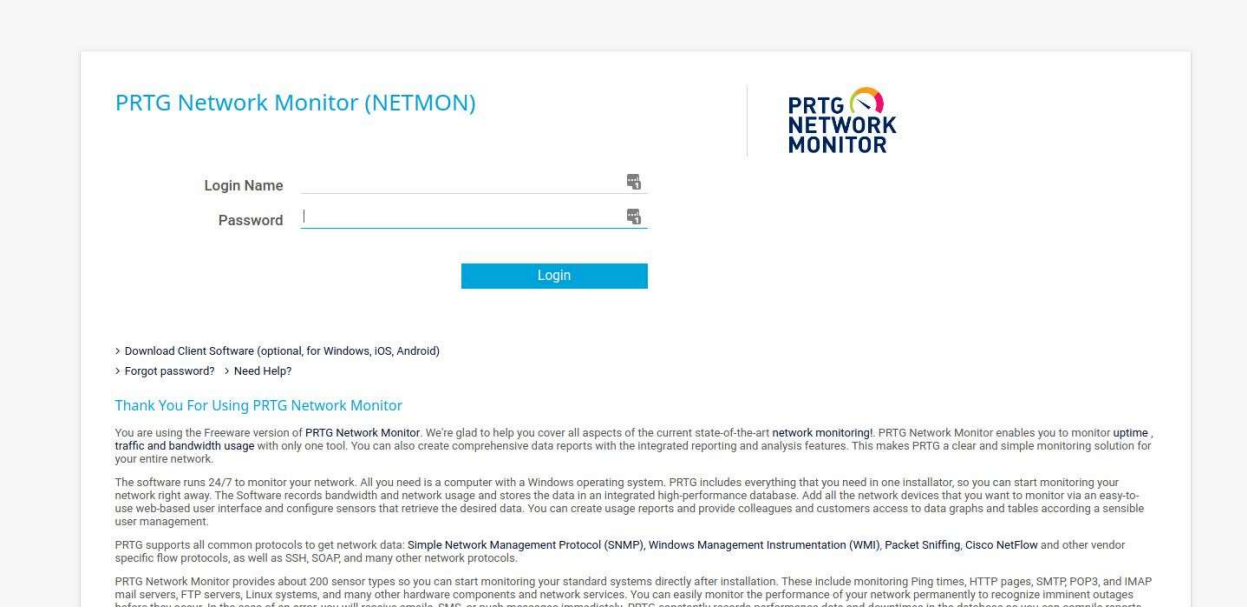
Finding nothing to note in Default User or Default, and with no access to the admin folder, the next obvious choice was Public. To my legitimate surprise and confusion, this folder contained the user.txt file with the user hash, and incidentally was the folder of the user we were meant to own. It took me a few minutes to realize we did not have to do anything to own a user on the system, except opening the folder and reading the file.



After grabbing the local user.txt file we find:



Alright, user hash out of the way, let's move on to getting the root hash. Since we cannot get into the administrator folder, lets head to the web application, since it is most likely vulnerable. Navigating to 10.10.10.152 in browser:

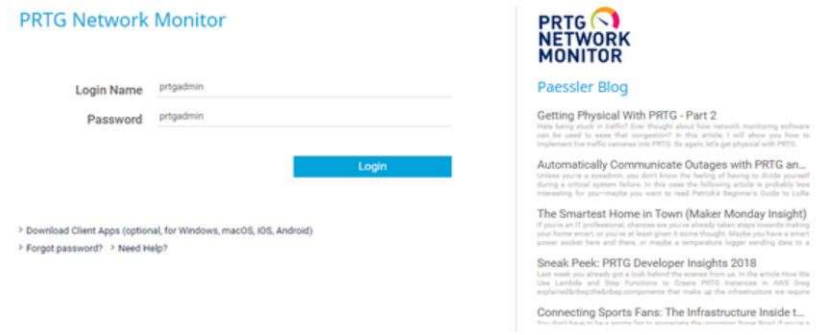


The web application is the PRTG Network Monitor, which does exactly as it sounds like. It is a complete network monitoring suite for any size network. The version we are running is 18.1.37.13946, yet the most up to date version of 18 is 18.4.47 (very interesting, but it makes sense if an older version is vulnerable). Let's head to the documentation (can be found here) and see what we can find. After perusing the quick start guide, we find that the default username/password of the admin account is prtgadmin/prtgadmin

This is straight from the documentation

Login

If everything works fine, the first thing you will see will not be the login screen, but the device tree. You only have to log in manually if you use a different browser.



PRTG Login Screen

- The default administrator credentials (login name **prtgadmin** and password **prtgadmin**) are automatically filled in. Select **Login** to proceed if you use PRTG on premises.

Of course, these credentials do no work. But, we have a username at least (prtgadmin). We can confirm this is a real username by using the forgot password tool on the login page. It returns an error message on invalid usernames, but tells us an email was sent to a recovery address when we enter “prtgadmin”.

Vulnerabilities/Exploitation

After some head scratching, I started to google around for "PRTG Network Monitor password location," or similar queries. At the top of the results I found a reddit post with some very interesting finds (located here). Essentially, the program accidently stored some domain and PRTG account information (passwords) in the Configuration File in plain text. This post was from a year ago, and reading about it on the Paessler website we find:

Important Notice: This issue affects PRTG version **17.4.35** (17.4.35.3326) through **18.1.37**. Previous versions are not affected.

What exactly is the issue?

An internal PRTG Network Monitor error caused some Active Directory integrated PRTG user account passwords and some other account passwords from the PRTG System Administration to be stored to the configuration file **PRTG Configuration.dat** in plain text, instead of being encrypted. We have fixed this issue as of PRTG version **18.1.38**.

Which passwords were affected?

Only the Active Directory integrated PRTG user account passwords of users that logged into PRTG for the **first** time after the affected version was installed were exposed.

Other possibly exposed passwords include the following passwords from the PRTG System Administration:

- Active Directory integration account password
- Proxy password
- SMTP relay password - primary server
- SMTP relay password - fallback server
- SMS delivery password
- Messenger passwords (from deprecated PRTG versions)

Not surprising that this issue was fixed in the version directly following the version we are working with 😉. Paessler suggests system admins to delete all the following files:

> We recommend that you delete all affected copies of the PRTG Configuration.dat file:
>
> - Automatically generated backups under:
> - C:\ProgramData\Paessler\PRTG Network Monitor\Configuration Auto-Backups\
> - Automatically generated temporary files that may exist:
> - C:\ProgramData\Paessler\PRTG Network Monitor\PRTG Configuration.old
> - C:\ProgramData\Paessler\PRTG Network Monitor\PRTG Configuration.nul
> - If you run PRTG Network Monitor in cluster mode, please also remember to remove the configuration backups in the PRTG data path on every failover node.
> - Also remember that you may have additional copies of the PRTG Configuration.dat file for backup purposes. We recommend deleting all affected copies of this file.

So, it makes sense for us to look at the current configuration file, as well as all the possible affected files listed above (if they exist). Finding the Program Data Folder on the ftp server took some time, but I found it eventually under:

Users/All Users/Paessler/PRTG Network Monitor (**All Users only shows up with ls -al**)

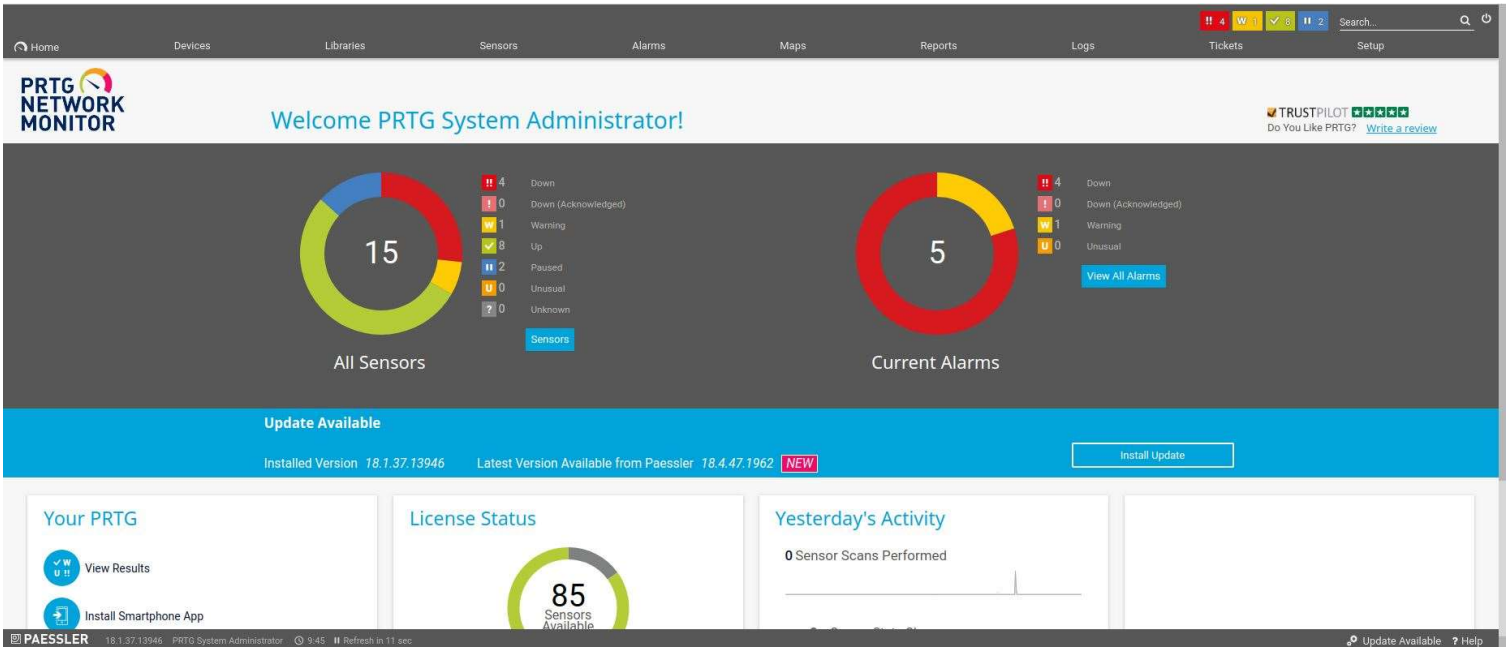
```
ftp> ls -al
200 PORT command successful.
125 Data connection already open; Transfer starting.
02-03-19  12:40AM       <DIR>          Configuration Auto-Backups
05-17-19  08:00PM       <DIR>          Log Database
02-03-19  12:18AM       <DIR>          Logs (Debug)
02-03-19  12:18AM       <DIR>          Logs (Sensors)
02-03-19  12:18AM       <DIR>          Logs (System)
05-17-19  07:52PM       <DIR>          Logs (Web Server)
05-17-19  08:02PM       <DIR>          Monitoring Database
02-25-19  10:54PM              1189697 PRTG Configuration.dat
05-17-19  09:07PM              1203154 PRTG Configuration.old
07-14-18  03:13AM              1153755 PRTG Configuration.old.bak
05-17-19  09:15PM              1697124 PRTG Graph Data Cache.dat
02-25-19  11:00PM       <DIR>          Report PDFs
02-03-19  12:18AM       <DIR>          System Information Database
02-03-19  12:40AM       <DIR>          Ticket Database
02-03-19  12:18AM       <DIR>          ToDo Database
226 Transfer complete.
ftp> pwd
257 "/Users/All Users/Paessler/PRTG Network Monitor" is current directory.
ftp>
```

My goal was to look through all the backups and search for “prtgadmin,” since we know that’s the admin username. I started with “PRTG Configuration.dat,” then .old, then .old.bak. The first two files contained the username, but all the passwords were hidden behind encrypted tags (thus they were of no use to us). But in .old.bak, I found exactly what I was looking for.

```
        </comments>
        <dbauth>
          0
        </dbauth>
        <dbcredentials>
          0
        </dbcredentials>
        <dbpassword>
      <!-- User: prtgadmin -->
      PrTg@dmin2018
        </dbpassword>
        <dbtimeout>
          60
        </dbtimeout>
        <depdelay>
          0
        </depdelay>
        <dependencytype>
```

Bingo, the admin password is PrTg@dmin2018. I should say I tried to use Hydra to brute force the password to no avail a few times, then I found a few posts saying brute force was not needed, and instead slowed the server. That was all the hint I needed to resort to google. I should have started with google, but again, first box here. So, we try the new password……. And still doesn’t work…. What?!?!?

This was supposed to be the ticket in, and instead we fail again. Not exactly. A simple change of the year to the current year (2019), makes the password PrTg@dmin2019, which gets us in to the admin panel.



A lot to be seen here. After looking through some parts of the panel I started to google around for any known vulnerabilities of the tool. Not surprising, we find this. Essentially, this blog post found a command execution vulnerability inside the PowerShell script that can be run for notifications in the system. The script in question:

```
# Demo 'Powershell' Notification for Paessler Network Monitor
# Writes current Date/Time into a File
#
# How to use it:
#
# Create a exe-notification on PRTG, select 'Demo Exe Notifcation - OutFile.ps1' as program,
# The Parametersection consists of one parameter:
#
# - Filename
#
# e.g.
#
#         "C:\temp\test.txt"
#
# Note that the directory specified must exist.
# Adapt Errorhandling to your needs.
# This script comes without warranty or support.


if ($Args.Count -eq 0) {

  #No Arguments. Filename must be specified.

  exit 1;
 }elseif ($Args.Count -eq 1){


  $Path = split-path $Args[0];

  if (Test-Path $Path)
  {
    $Text = Get-Date;
    # out-file is the output file cmdlet
    # Send (pipe) $Text to the file at $Args[0]
    $Text | out-File $Args[0];
    exit 0;

  }else
  {
    # Directory does not exist.
    exit 2;
  }
}
```

The Network Monitor allows us to create notifications, that fire when certain events occur. The actions taken when triggered vary, from sending us a text message, email, or even executing a program. The application gives us two demo programs, both of which do the same thing. These files are OutFile.ps and Outfile.bat.

OutFile.ps takes in a given file location and writes the date to the file (and creates the file if it does not exist). This application has direct access to the files in the ftp server, and better yet, we have admin privileges since we logged in as the admin! The vulnerability lies in the underlined portion above. When we create a notification, and execute a program as an action, we can specify the parameters to use (just as if we were executing the program from the command line). We can pass in a file location, but then append other commands that the system will execute. We can do this, since there is no type of input scrubbing or validation done on the parameters we pass in. In the blog, the writer uses the following example as a proof of concept of the harm that could be done:

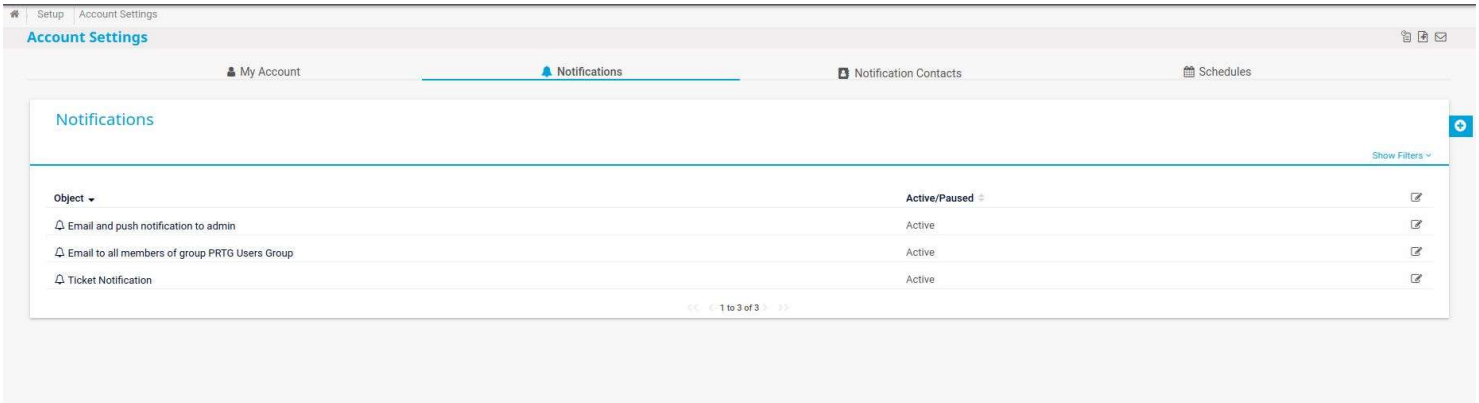
```
test.txt;net user pentest p3nT3st! /add
```

I do not know much about PowerShell, but in theory I came up with 2 types of payloads I could try.

1. Creating a new user and adding them to the administrator group
2. Reading the admin root.txt file to a file we can read without admin privileges

So, let's try the first type of payload. First, we must find where the notification settings are.

All notifications can be found under Account Settings→Notifications



If we click on the little blue plus on the right-hand side, we can create a new notification. We can give the notification any name we choose, then scroll down and select "Execute Program" as the action. No need to worry about triggering these notifications, we will get to that.

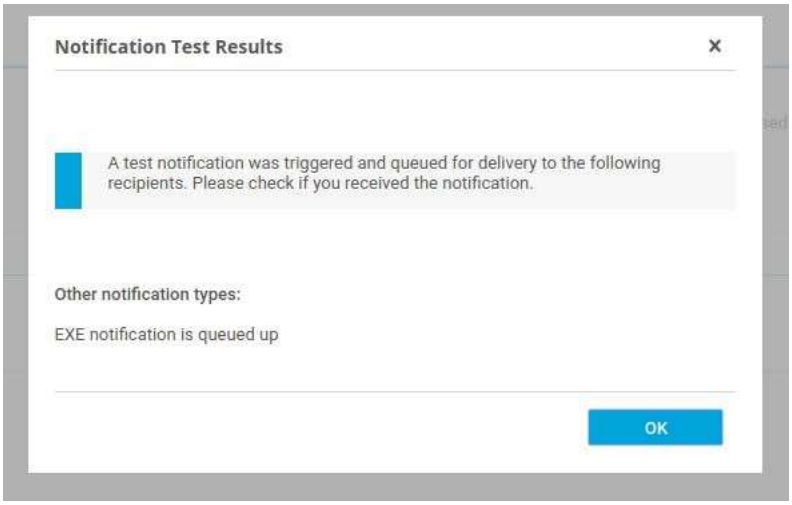
We have chosen the Power Shell script and to test to ensure the basic functionality works, have included a very simple parameter:

"C:\Users\Public\test.txt"

When this notification fires then, we should be able to find this file on the ftp server, at the given path.

Now let's test it out. To test fire a notification, click the little button on the far right of each notification (red circle), then click on test notification (circled in blue).

You should see this screen if you clicked the correct buttons:



Sure enough, if we check the ftp server, we find test1.txt in the Public User folder. Awesome. Oddly, the text inside the folder is not the date or time, but some non-printable characters. We will ignore this for now, as it has no bearing on the solution path. From here I tried everything to create a user, and add it to the admin group. With the hope that I could log into the ftp server with admin privledges and go home happy. Unfortunately, I could not figure out a way to do so. My payloads to try this included (parameters to the program in the notification settings):

To Create a user:

net user <username> <password> /<option>

To Add to group:

net localgroup <group> <username> /<option>

After each command, I waited some time and checked the logs in the admin panel. If any error occurred, the logs would output a very helpful message.

C:\Users\Public\not_a_drill.txt;NET USER pirate pirate /add;NET LOCALGROUP Administrators pirate /add

"C:\Users\Public\not_a_drill.txt;net user pirate pirate /add;net localgroup Administrators pirate /add"

In both above cases, the error explained the password was not matching requirements. So, I spruced it up a bit

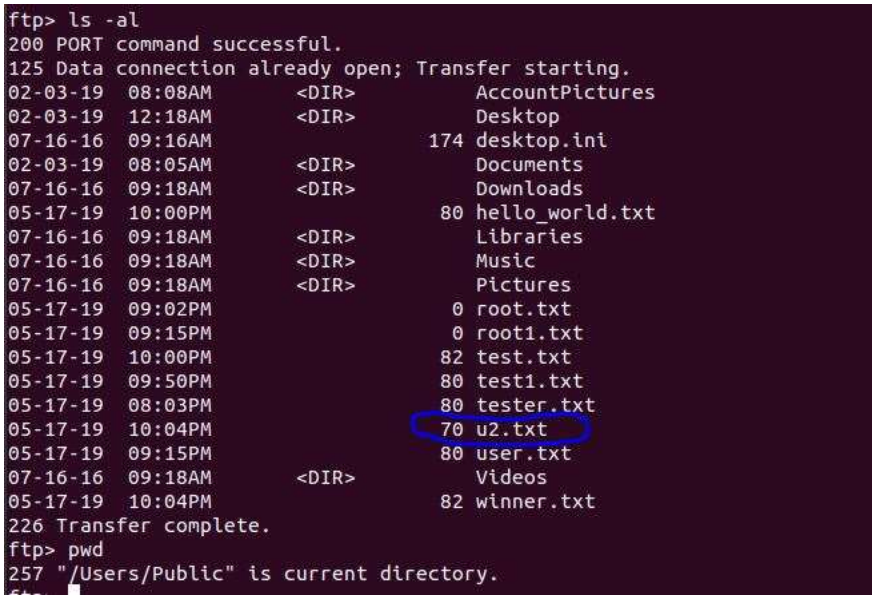
C:\Users\Public\not_a_drill.txt;net user pirate Pirate_@12345 /add;net localgroup Administrators pirate /add

Error sending "EXE": Error1. There is no such global user or group: pirate. More help is available by typing NET HELPMSG 3783.

After a few days of switching arguments, trying different usernames, and scratching my head, I decided to try plan B. The idea is to read the Admin root.txt file and output it to a file in the Public User directory, so even as an anonymous user, I could read the root hash. Now I understand I am not technically gaining root privileges here, but I was out of ideas.

Payload: 'C:\Users\Public\winner.txt';cat 'C:\Users\Administrator\Desktop\root.txt' | out-file 'C:\Users\Public\u2.txt'

Let’s check the filesystem for u2.txt in the Public User directory. Sure enough, it’s there.



All that is left to do is grab the file and read what we hope is the hash of root.

Forgetting the two weird characters at the end, there is the root hash.

Conclusion:

Whew, for a straightforward box, that was a lot of work! Being my first box, I learned a ton, and hope to use this experience for other, more challenging boxes. I am also excited to see how others ended up owning root, as my solution was not technically “owing” (although I am new to this, so who knows). This was rather long, but I wanted to be as in depth as possible, so if you made it this far kudos. If I missed anything, or you have any comments, just reach out and let me know! Until next time…