Countering EDRs With The Backing Of Protected Process Light (PPL)

zerosalarium.com/2025/08/countering-edrs-with-backing-of-ppl-protection.html

Zero Salarium August 23, 2025

I. INTRO

Important or sensitive processes of modern Windows operating systems are now protected by the **Protected Process Light (PPL)** feature. You might be familiar with this function if you've ever scrutinized the LSASS process.

Antivirus programs or Endpoint Detection and Response (EDR) solutions often protect their service programs and client agents by registering for protection using PPL.

In this article, I will present how to exploit a process protected by PPL to interact with other protected system components, which cannot be accessed by a normal process without the support of Windows drivers (even if you have **SYSTEM** or **TrustedInstaller** privileges). After that, I will experiment with using an existing PPL program on Windows to overwrite the service file of Windows Defender.

I will also introduce the "**CreateProcessAsPPL**" tool, which helps run a process with PPL functionality enabled, provided that the executable file of this process supports it.

Follow me on X to get the latest pentest and red team tricks that I've been researching: Two Seven One Three (@TwoSevenOneT) / X.

II. CENTRAL PART

1. Some basic information about Protected Process Light (PPL)

Protected Process Light (PPL) is a Windows security feature introduced in Windows 8.1 that prevents other processes from tampering with, terminating, or accessing the memory of critically important system services and anti-malware applications.

PPL assigns a security signature to processes, and only processes with an appropriate signature or "level" can interact with others, creating a security hierarchy. This mechanism helps to protect against malware that tries to compromise system integrity by attacking trusted processes.

Signatures and Trust: PPL relies on digital signatures to verify the integrity of code. Only processes with the correct PPL signature are allowed to perform certain operations on protected processes.

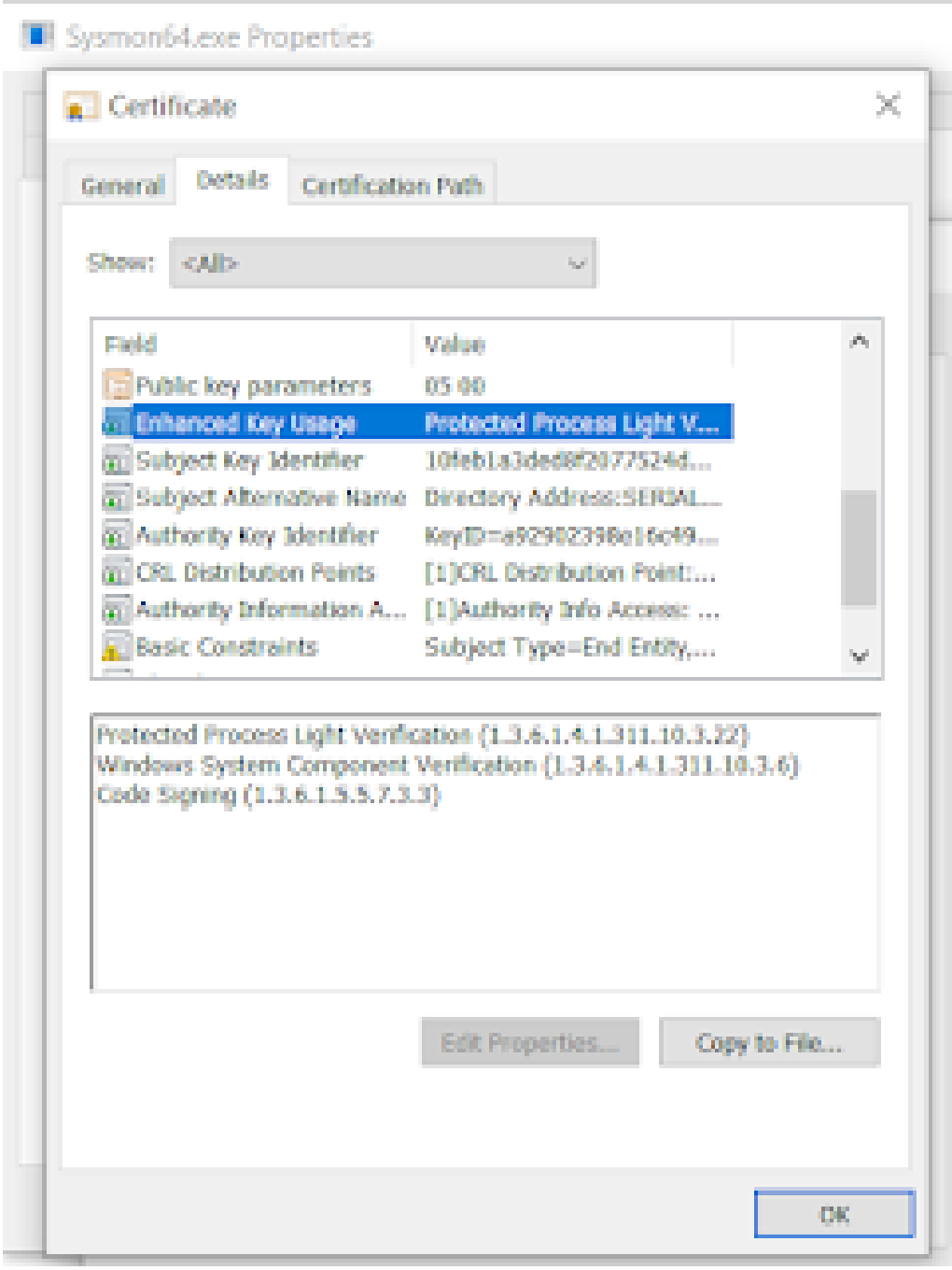
Hierarchy: A hierarchy of PPL "levels" is established, with higher-privilege processes being immune from tampering by lower-privilege ones.

Immunity: Processes running with the PPL flag are immune to termination and modification by other processes unless those other processes also possess a higher PPL level.

2. How to create a "PPL process"

To run an EXE with PPL (Protected Process Light) protection, the executable and any loaded DLLs must be signed with a specific certificate and have their digital signatures verified by the operating system, as detailed in the Anti-malware service signing requirements.

The signature used for PPL is called Enhanced Key Usage (EKU). This is a certificate extension that specifies the intended purposes of a public key, defining what the certificate can be used for.

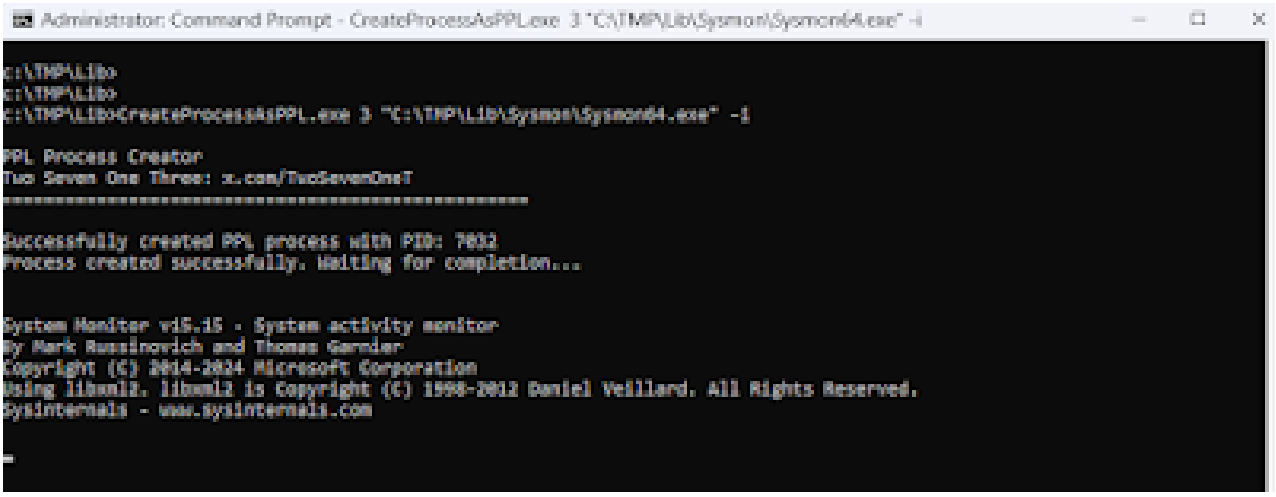


Even if your EXE has a valid EKU signature, it must be **called in a special way** to enable PPL functionality. If you run it like a regular EXE, PPL will not be activated.
To create a process with PPL functionality, the CreateProcess function must use the flags "**EXTENDED_STARTUPINFO_PRESENT | CREATE_PROTECTED_PROCESS**" to activate it.
In short, to have a protected process with PPL, the following conditions must be met:

- The executable file must have a valid **EKU** digital signature.
- Use the flags **EXTENDED_STARTUPINFO_PRESENT | CREATE_PROTECTED_PROCESS** in the **CreateProcess** function.

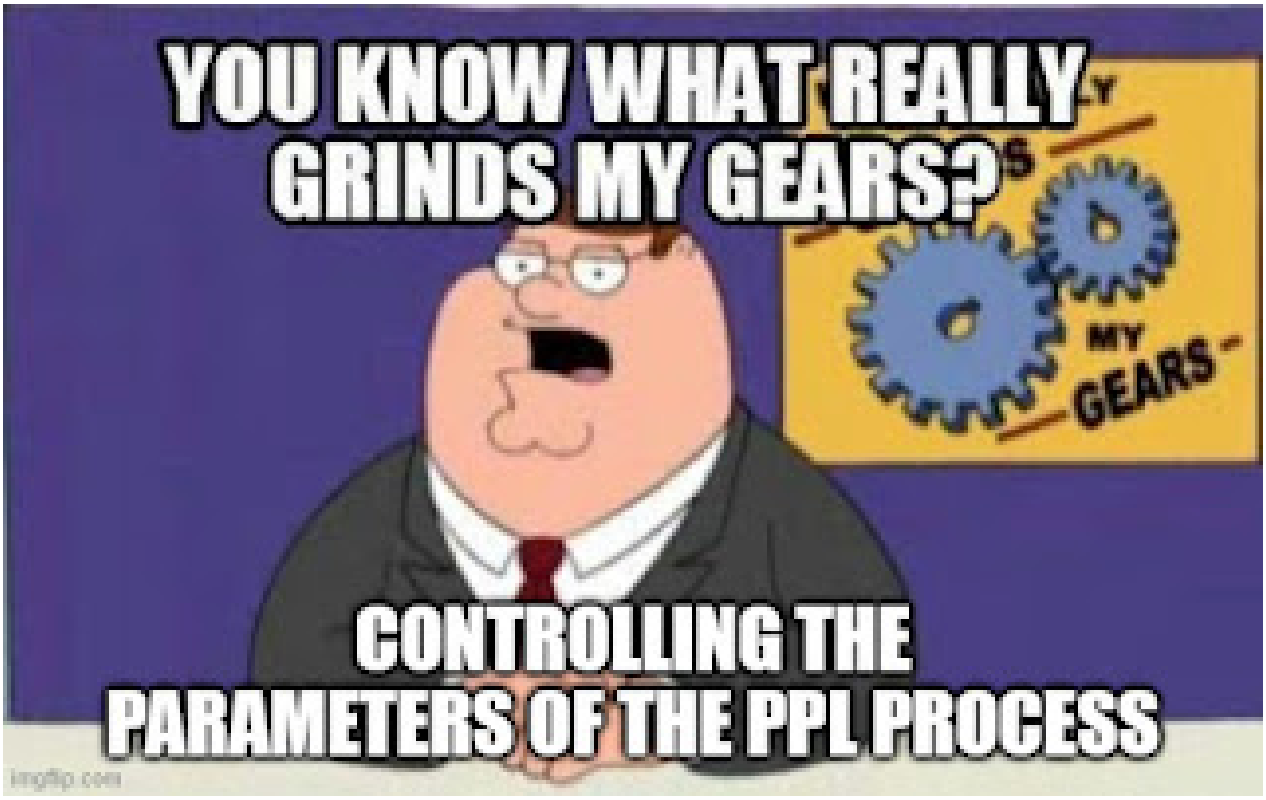
I have created a simple tool to run a program with PPL enabled. You can download this tool via the link.
https://github.com/TwoSevenOneT/CreateProcessAsPPL

The program accepts parameters such as the protection level from 0 to 4, the path to the executable file, and additional parameters for running this executable.
The image below shows me running the Sysmon program with PPL set to **PROTECTION_LEVEL_ANTIMALWARE_LIGHT** = 3.

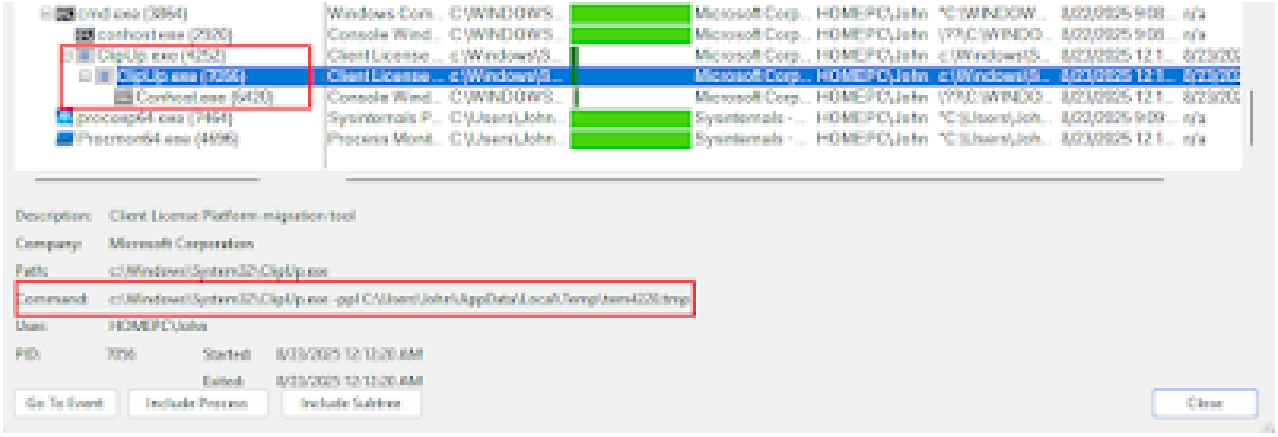


3. Abusing PPL processes

At this point, we have actively created a PPL process as desired. And when I say "**actively**" I mean that we can freely pass parameters to these PPL processes.



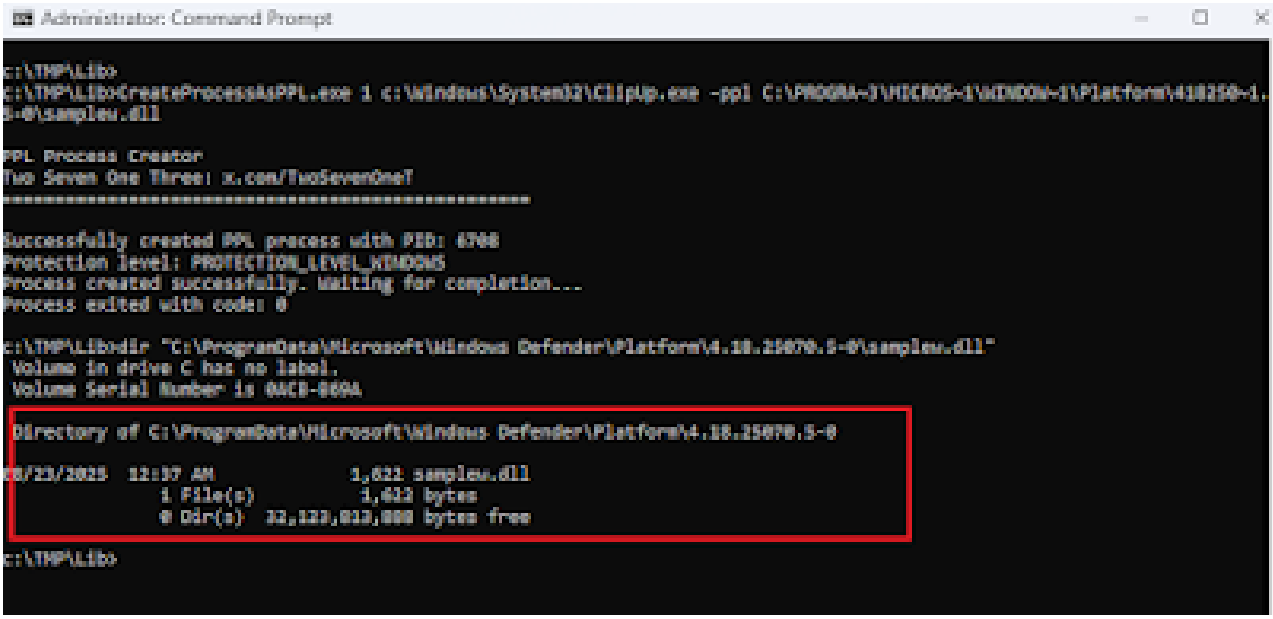
So the question arises: **What can we achieve if we can control the parameters of a valid PPL program**?
While searching for files with signatures that support PPL, I found a rather interesting file: `"C:\Windows\System32\ClipUp.exe"`
When running "**ClipUp.exe**", the program will automatically spawn a new process of itself, and this new process will have parameters to create a log file at any specified location.



So if I run **Clipup.exe** with PPL protection using the **CreateProcessAsPPL** tool, I will have the ability to control file writing restricted to a PPL process.
Next, I will check whether **ClipUp.exe** with PPL protection has the permission to write to locations that normal processes cannot access. Specifically, this is the executable folder of Windows Defender.

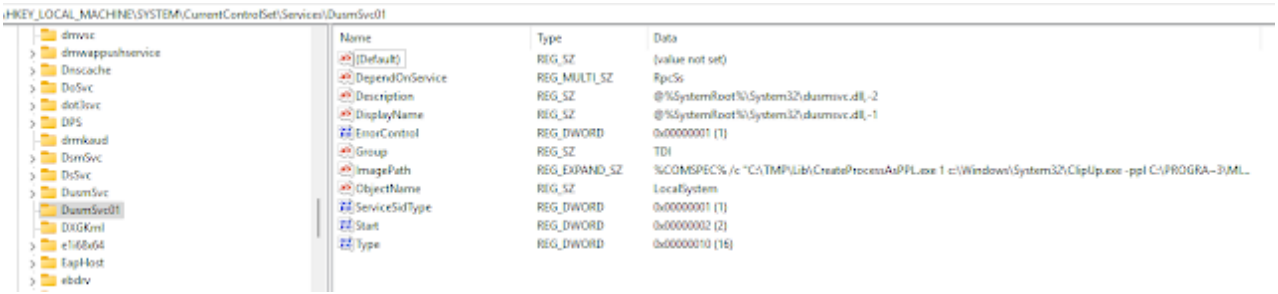
```
CreateProcessAsPPL.exe 1 c:\Windows\System32\ClipUp.exe -ppl
C:\PROGRA~3\MICROS~1\WINDOW~1\Platform\418250~1.5-0\samplew.dll
```

Since **Clipup.exe** supports a PPL protection level of **PROTECTION_LEVEL_WINDOWS**, I run it in mode number 01.
The **-ppl** parameter informs Clipup.exe where the log file will be written.
C:\PROGRA~3\MICROS~1\WINDOW~1\Platform\418250~1.5-0 is the short name that leads to the Defender folder, as **Clipup.exe** cannot parse paths with spaces.

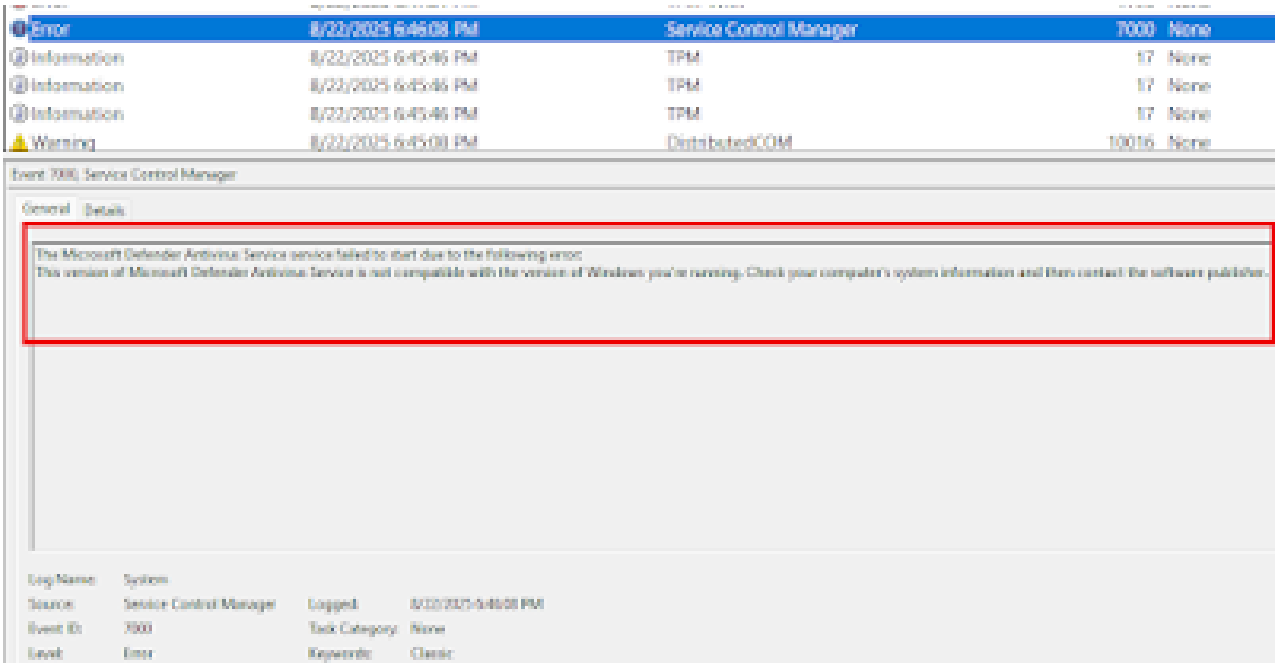


As you can see above, **Clipup.exe** was able to write a file in the Defender folder. However, at this point, we cannot control the content of the written file. Therefore, the most feasible way to exploit **Clipup** is to find a way to overwrite the service file of Defender: **MsMpEng.exe**.
The issue arises here because Defender is active, so the **MsMpEng.exe** file is open and cannot be overwritten. So, we need to find a time when the Defender service is not running. This is during the Windows startup; we just need to create a service that runs

before Defender, and this service will run the **CreateProcessAsPPL** tool to spawn **Clipup** with PPL protection.
After experimenting with **Process Monitor**, I created a service that will always run before Defender as follows:



After that, Defender will not be able to run because the executable file has been corrupted.



At this point, we have successfully exploited a program with PPL protection to bypass the Antivirus. You can experiment with the AVs or EDRs that you are interested in.

III. FINALE

PPL (Protected Process Light) is a Windows security feature introduced in Windows 8.1 that protects critical system processes and anti-malware services from unauthorized access, tampering, and termination.
With a PPL-protected process, you can fully interact with other PPL processes at the same level or lower.
If you have an executable file with a valid digital signature, CreateProcessAsPPL.exe will help you run that file with PPL protection.
Controlling the parameters of programs with PPL protection will open the door for exploitation of these processes for pentesters and red teamers.

In the previous article, I experimented with exploiting the control of parameters in the **Clipup.exe** program to destroy the executable file of Windows Defender.
Author of the article: Two Seven One Three