Incident Response & the Windows PEB

Taz Wake | Halkyn Consulting Ltd | Originally published 2023-11-27

Introduction to Windows Memory Structure and PEB

The Windows operating system's architecture is renowned for its intricate and efficient management of applications and processes. Central to this architecture is the Process Environment Block (PEB), a crucial component introduced with Windows NT. This marked a significant evolution in process management compared to simpler structures in earlier versions such as Windows 95 and Windows 3.1. The PEB's introduction heralded a new era of more robust and complex process interactions within the system.

Residing in the user space of each process, the PEB epitomises the sophistication of Windows' memory management. It serves as a central repository for process-specific information, crucial for the operating system's efficient task management. This information includes process identifiers, version data, system DLL initialisation, and process heap details. Its design facilitates seamless and dynamic process management, essential for the stability and performance of the Windows operating system.

The PEB has evolved with each Windows release, adapting to the demands of modern computing. Its enduring presence and continual development reflect the changing landscape of software architecture and security, making the PEB a cornerstone for understanding and interacting with Windows processes. This is particularly true for application developers and those involved in security and forensic analysis. The PEB's comprehensive nature and information depth make it indispensable in the professional world of Windows environments.

The Role and Composition of the PEB

The PEB is integral to process management in Windows. More than just a data store, it is a complex framework containing key components that contribute to the smooth operation and oversight of processes. The Process ID (PID), a vital element in the PEB, is fundamental to the system's management of multiple processes. It's not merely a label, but a critical identifier for resource allocation and maintaining order in the processing environment.

The PEB also stores heap pointers, leading to memory spaces crucial for dynamic memory management. These pointers are instrumental in managing dynamic memory, ensuring efficient operation within designated memory confines. Additionally, the PEB houses environment variables: dynamic values pivotal to a process's execution. They encompass system and user configurations, influencing process interactions with the wider system environment, and are key to customising operational context.

Moreover, the PEB is a repository for information about loaded modules, such as Dynamic Link Libraries (DLLs) and executables. This information is crucial for understanding

process behaviour, revealing external dependencies, interfaced libraries, and leveraged functionalities. For security and forensic professionals, this data offers insights into the process's runtime environment and potential indicators of anomalous or malicious activities. The PEB's role in storing this information makes it a critical tool for system analysis, security monitoring, and forensic investigations, integral to both the functional and security aspects of process management.

Security Implications of PEB Accessibility

The accessibility of the Process Environment Block in user space, while essential for legitimate application functionality, also presents significant security risks. This vulnerability arises from the PEB's comprehensive repository of process-specific information, susceptible to malicious manipulation, leading to substantial system vulnerabilities. For example, malware can exploit the PEB to alter a process's runtime environment by injecting malicious code or modifying data structures within the PEB. This can effectively camouflage the malware's presence, evade detection, or escalate system privileges.

Malware often employs tactics like DLL injection to manipulate the DLL loading process. By modifying the PEB, malware can change the DLL search order or insert paths to compromised libraries, leading to the loading of malicious instead of legitimate libraries. This technique is facilitated by API calls like **SetWindowsHookEx** or **CreateRemoteThread**, allowing code execution within another process's context. Hijacking the PEB's module list enables malware to disguise itself as a legitimate process and access sensitive information and resources.

Additionally, sophisticated malware can exploit the PEB to monitor or alter environment variables. Using system calls like **NtSetInformationProcess**, malware can change these variables in the PEB, leading to altered process behaviours ranging from benign misconfigurations to severe security breaches. For instance, malware might modify system path variables to redirect execution to malicious executables, leading to data exfiltration, system corruption, or backdoor creation for persistent access.

Investigating the PEB during Incident Response

The Process Environment Block (PEB) is a veritable treasure trove of information for incident responders delving into the depths of a security investigation. It serves as a window into the process's inner workings, providing a detailed overview of its current state and activities. By examining the PEB, responders can unearth critical information, such as the list of modules loaded by the process, which is especially significant in the detection of malicious activities. Malicious DLLs, often indicators of a security breach, can be identified through their presence in the module list. This insight is crucial in discerning whether a process has been compromised or is behaving anomalously.

Moreover, the PEB holds valuable data regarding the process's environment variables and heap state, which can reveal subtle, yet telling, signs of tampering or exploitation. For instance, unusual changes in environment variables or unexpected heap allocations

might suggest that the process has been manipulated by an external entity, potentially for nefarious purposes. The ability to access and analyse this level of detail allows incident responders to create a comprehensive picture of a process's behaviour and interactions within the system, facilitating a more targeted and effective response to potential threats.

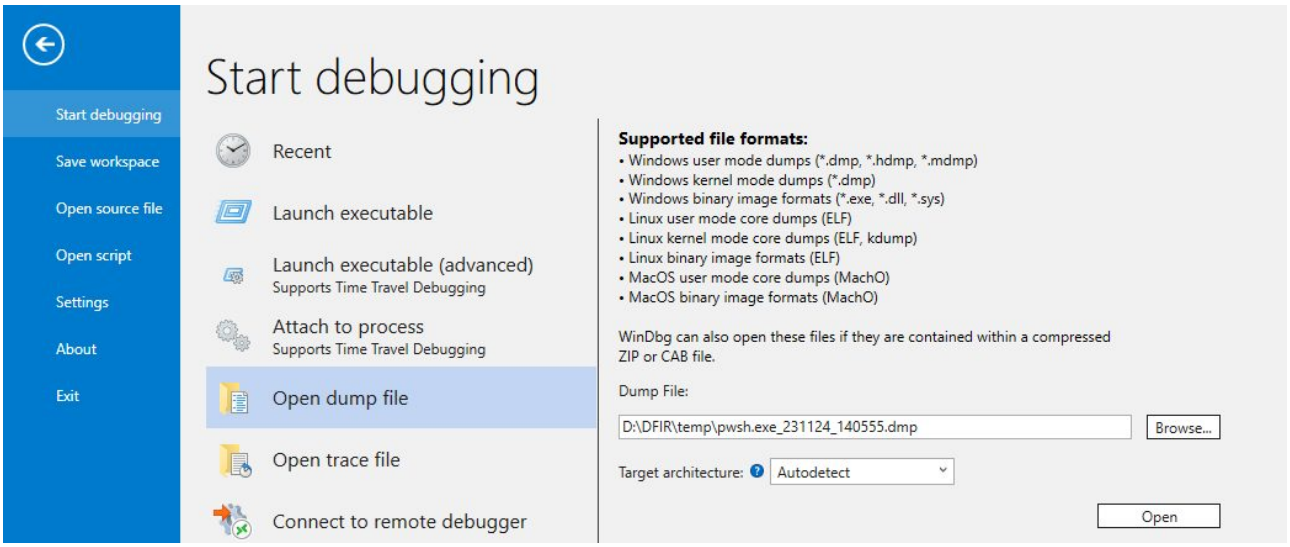
Extracting Process Command Line Arguments Using WinDBG

WinDBG (Windows Debugger) is an indispensable tool in the arsenal of any professional dealing with Windows environments, particularly for those in the realms of incident response and system debugging. This powerful debugger enables an in-depth examination of the PEB, providing access to a wealth of process-specific information. A key feature of WinDBG is its ability to inspect the contents of the PEB, including the command line arguments used to initiate the process. These arguments can offer crucial context about the process's purpose and origin, which is vital in understanding the nature of the process and assessing its legitimacy.

To utilise WinDBG effectively, responders attach the debugger to the target process, either through a direct connection or by analysing a memory dump. Once attached, specific commands like **!peb** can be employed to display the contents of the PEB. The command outputs various details, including the command line arguments, which are often pivotal in tracing the source of a process, especially in cases of process injection or exploitation. By dissecting these arguments, responders can ascertain if a process was launched with unusual or suspicious parameters, potentially indicative of malicious intent. WinDBG's prowess in extracting such nuanced information from the PEB makes it an invaluable tool in the intricate process of security analysis and incident investigation.

High-level guidance for analysing a process' PEB with WinDBG

- Ensure you have sufficient privileges to attach WinDBG to the target process. This can require Admin/SYSTEM privileges for a running process.
- Attach to the process, or to a dump of the processes. Generally, attaching to a process dump file is done by going to the **file** menu, selecting **start debugging** and then navigating to the target object.



Attaching to a process dump file in WinDBG

- Locate the PEB – enter the **!peb** command to see the contents of the process environment block for that process. This will identify the location in memory and provide a wealth of information regarding the process itself.

```
0:000> !peb
PEB at 00000088c754c000
    InheritedAddressSpace:    No
    ReadImageFileExecOptions: No
    BeingDebugged:            No
    ImageBaseAddress:         00007ff7c6fc0000
    NtGlobalFlag:             0
    NtGlobalFlag2:            0
    Ldr                       00007ff8bdc163e0
    Ldr.Initialized:          Yes
    Ldr.InInitializationOrderModuleList: 0000025e4ca046d0 . 0000025e7284f7e0
    Ldr.InLoadOrderModuleList:           0000025e4ca04860 . 0000025e7284f7c0
    Ldr.InMemoryOrderModuleList:         0000025e4ca04870 . 0000025e7284f7d0
                    Base TimeStamp                     Module
            7ff7c6fc0000 6331eb0e Sep 26 19:10:22 2022 C:\Program Files\WindowsApps\Microsoft.PowerShell_7.2.
            7ff8bda90000 bced4b82 Jun 11 00:19:30 2070 C:\windows\SYSTEM32\ntdll.dll
            7ff8bbfb0000 2e35230e Jul 26 15:51:58 1994 C:\windows\System32\KERNEL32.DLL
            7ff8bae00000 10f6a783 Jan 07 23:36:35 1979 C:\windows\System32\KERNELBASE.dll
            7ff8bd040000 9514b7b9 Apr 04 16:28:57 2049 C:\windows\System32\USER32.dll
            7ff8bb5a0000 11899c97 Apr 29 11:53:11 1979 C:\windows\System32\win32u.dll
            7ff8bd000000 3c42a789 Jan 14 09:40:25 2002 C:\windows\System32\GDI32.dll
            7ff8bb1b0000 bd8a3915 Oct 08 01:06:45 2070 C:\windows\System32\gdi32full.dll
```

Example of viewing the PEB start data using WinDBG

Begin your analysis! Some examples of things to review, include:

- The PEB contains a list of loaded DLLs. This should be reviewed to see if any standout with unusual names, paths or providing unexpected functionality.
- Check the command line arguments.
- Review the environment variables – this can often be used to hide malicious activity.

Remember to keep good notes!

Conclusion: The Value of PEB Analysis in Security

Understanding and analysing the Process Environment Block is crucial for any security professional dealing with incident response or forensic investigations in Windows

environments. By leveraging tools like WinDBG, responders can gain valuable insights into the workings of a process, aiding in the identification and mitigation of security threats.

Further reading/study

The following blog posts give some incredible insights into the PEB (and how it can be attacked), and are definitely worth reading:

https://medium.com/@boutnaru/the-windows-concept-journey-peb-process-environment-block-e8078a146612

https://mohamed-fakroud.gitbook.io/red-teamings-dojo/windows-internals/peb

https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/exploring-process-environment-block

You can also learn more about Windows Incident Response (including a broader look at Memory forensics) on the SANS Institute courses:

https://sans.org/for508 - Advanced Incident Response

https://sans.org/for532 - Enterprise Memory Forensics In-Depth

---

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858