GitHub - GhostPack/KeeThief: Methods for attacking KeePass 2.X databases, including extracting of encryption key material from memory.

By HarmJ0y

Archived: 2026-04-05 19:10:11 UTC

Allows for the extraction of KeePass 2.X key material from memory, as well as the backdooring and enumeration of the KeePass trigger system.

Author: Lee Christensen (@tifkin_), Will Schroeder (@harmj0y)

License: BSD 3-Clause
Required Dependencies: None
Optional Dependencies: None

This project includes a number of components:

DecryptionShellcode

A modified version of Matt Graeber (@mattifestation)'s PIC_Bindshell position-independent shellcode project, licensed under the 3-clause BSD license.

Modifications were made to build shellcode to inject into a KeePass.exe process that decrypts DPAPI blobs using RtlDecryptMemory.

KeePass-2.34-Source-Patched

Patched source code of the KeePass project (version 2.34), licensed under the GNU GENERAL PUBLIC LICENSE with the specific KeePass license located here. Modifications were made to allow the manual specification of key data (in the form of base64 strings) when decrypting a database. Changes are to KeePromptForm.cs, KeePromptForm.Designer.cs, KcpKeyFile.cs, and KcpUserAccount.cs.

KeeTheft

The main KeeThief code, "where the magic happens".

KeeThief's GetKeePassMasterKeys() will attach to the target KeePass process using CLR MD and enumerate all CLR heap objects, searching for a KeePassLib.PwDatabase object. If one is found, the path is extracted from the m_strUrl field, and all referenced objects are enumerated, searching for a KeePassLib.Keys.CompositeKey.

If a composite master key is found, information for each key type (KcpPassword, KcpKeyFile, KcpUserAccount) is extracted, including the RtlEncryptMemory() encrypted data blobs of key data. For any encrypted blobs found, shellcode is injected into the KeePass process that calls MyRtlDecryptMemory() to decrypt the memory blobs, returning the plaintext/unprotected key data.

This is a different approach than denandz' excellent KeeFarce project, which injects code to load a bootstrap DLL into the KeePass process, which then loads an C# assembly along with CLR MD, and executes the 'Export' method on a KeePass.DataExchange.Formats.KeePassCsv1x object in order to export all existing passwords to disk. KeeTheft walks the heap for composite key information and injects shellcode to decrypt each encryption material component as appropriate.

Included in the project is a .NET 2.0 backport of the CLR MD project (necessary for PowerShell v2 compatibility). The CLR MD project is licensed by Microsoft under the MIT license.

On building the project, a merged .\KeeTheft\bin\ReleaseKeeTheft.exe binary containing KeeTheft and the CLR MD will be produced.

PowerShell

The **KeeThief.ps1** PowerShell file contains **Get-KeePassDatabaseKey**, which loads/executes the KeeTheft assembly in memory to extract KeePass material from an KeePass.exe process with an open database.

The **KeePassConfig.ps1** file contains method to enumerate KeePass config files on a system (**Find-KeePassconfig**), retrieve the set triggers for a KeePass.config.xml file (**Get-KeePassConfigTrigger**), add malicious KeePass triggers (**Add-KeePassConfigTrigger**), and remove KeePass triggers (**Remove-KeePassConfigTrigger**).

KeeThief License
