

## EventPairHandle as Anti-Dbg Trick

**Author:** Giuseppe 'Evilcry' Bonfa'

**E-Mail:** evilcry {AT} gmail {DOT} com

**Website:** <http://evilcry.netsons.org> - <http://evilcodecave.wordpress.com>

### EventPairHandle

An **EventPair Object** is an Event constructed by two **\_KEVENT** structures which are conventionally named **High** and **Low**. EventPairs are used for synchronization in **Quick LPC**, they allow the called thread to continue the current quantum, reducing scheduling overhead and latency. Now by looking to the basic operations that a debugger need to accomplish, we can see that these tasks are conceptually simple, when the target is normally running, the debugger is sleeping, but when certain events occur Dbg Wakes Up. Became clear that there is a strict relation between generic **Event Objects** and Debuggers cause they have to create a custom Event called **DebugEvent** able to handle exceptions. Due to the presence of Events owned by the Debugger, every information relative to the Events of a normal process differs from a debugged process.

This is the struct that describes an EventPair:

```
typedef struct _KEVENT_PAIR {
    USHORT Type;
    USHORT Size;
    KEVENT Event1;
    KEVENT Event2;
} KEVENT_PAIR, *PKEVENT_PAIR;
```

Nothing more than a couple of **\_KEVENTS**. **NtCreateEventPair** is the responsible of EventPair creation, here the prototype:

**NTSYSAPI**

**NTSTATUS**

**NTAPI**

**NtCreateEventPair(**

```
    OUT PHANDLE                EventPairHandle,
    IN ACCESS_MASK              DesiredAccess,
    IN POBJECT_ATTRIBUTES       ObjectAttributes OPTIONAL );
```

**PARAMETERS**

**hEventPair:** Pointer to the variable that receives handle to the event-pair object.

**AccessMask:** Type of access requested to the event-pair object. This can be a combination of any of the following flags:

EVENT\_QUERY\_STATE, EVENT\_MODIFY\_STATE, and  
EVENT\_ALL\_ACCESS.

**ObjectAttributes:** Points to the OBJECTS\_ATTRIBUTES structure containing the information about the event-pair object to be created, such as name, parent directory, objectflags, and so on.

EventPairs are used by the Win32 subsystem to provide notification when the client thread has copied a message to the Win32 server, or vice versa. LPC messages are passed in the section object, and synchronization is performed by the event-pair object. The event-pair object eliminates the overhead of using the port object to pass messages containing pointers and lengths.

LPC was used into **(Kernel/User)-mode Debug Support** before Windows XP for various

notifications. The new Debugging Support makes use of **\_DEBUG\_OBJECT**. This struct is a wrapper around the event used by **WaitForDebugEvent**, consequently we need to see how **\_DEBUG\_EVENT** is structured, here the struct:

```
typedef struct _DEBUG_EVENT
{
    LIST_ENTRY EventList;
    KEVENT ContinueEvent;
    CLIENT_ID ClientId;
    PEPROCESS Process;
    PETHREAD Thread;
    NTSTATUS Status;
    ULONG Flags;
    PETHREAD BackoutThread;
    DBGKM_MSG ApiMsg;
} DEBUG_EVENT, *PDEBUG_EVENT;
```

As you can see between the members of this struct we have the last one that sounds really interesting **DBGKM\_MSG** that by the name we can understand is referred to Debug Messaging (Messaging means implications with LPC mechanism), so let's see **DBGKM\_MSG** struct:

```
typedef struct _DBGKM_MSG
{
    PORT_MESSAGE h;
    DBGKM_APINUMBER ApiNumber;
    ULONG ReturnedStatus;
    union
    {
        DBGKM_EXCEPTION Exception;
        DBGKM_CREATE_THREAD CreateThread;
        DBGKM_CREATE_PROCESS CreateProcess;
        DBGKM_EXIT_THREAD ExitThread;
        DBGKM_EXIT_PROCESS ExitProcess;
        DBGKM_LOAD_DLL LoadDll;
        DBGKM_UNLOAD_DLL UnloadDll;
    };
} DBGKM_MSG, *PDBGKM_MSG;
```

**PORT\_MESSAGE** defines the **LPC Message Header** that is used for every communication between client and server. Became clear that, despite to the fact that LPC is not used, the presence of **PORT\_MESSAGE** reveals that LPC is supported and consequently influences EventPair count, read as EventPair Handle. We have seen that LPC is used into Debugging System, so a debugged process will present an EventPair Handle different from a not debugged.

I've tested this fact into a Windows XP sp2 machine and this method works (on OllyDbg, other debuggers are untested).

Here the Code:

```

#define WIN32_LEAN_AND_MEAN
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "defs.h"

#pragma comment(lib,"ntdll.lib")
#pragma comment(lib,"psapi.lib")

void QueryProcessHeapMethod(void)
{
    PDEBUG_BUFFER buffer;

    buffer = RtlCreateQueryDebugBuffer(0,FALSE);

    RtlQueryProcessHeapInformation(buffer);

    if (buffer->RemoteSectionBase == (PVOID) 0x50000062)
        MessageBoxA(NULL,"Debugged","Warning",MB_OK);
    else
        MessageBoxA(NULL,"Not Debugged","Warning",MB_OK);

    if (buffer->EventPairHandle == (PVOID) 0x00002b98)
        MessageBoxA(NULL,"Debugged","Warning",MB_OK);
    else
        MessageBoxA(NULL,"Not Debugged","Warning",MB_OK);

    printf("EventPairHandle= %x", (int)buffer->EventPairHandle);
}

int main()
{
    QueryProcessHeapMethod();

    return (EXIT_SUCCESS);
}

```

References:

<http://www.alex-ionescu.com/dbgk-3.pdf>

Feel free to contact me, test this feature and make me know with a mail if you can.

Regards,  
Giuseppe 'Evilcry' Bonfa'

The rest of the world as IT suckers that tried to deceive me with their fake work proposal can die.