



Monthly Research

# 固定アドレスのポインタによるASLRバイパスの理解

株式会社 F F R I  
<http://www.ffri.jp>

## MS13-063

- マイクロソフトが2013年8月に公開したセキュリティパッチ
- ASLRセキュリティ機能のバイパスの脆弱性(CVE-2013-2556)の対策を含む
- 今回は、この脆弱性の問題点と、パッチによる変更点を調査

## ASLRバイパスの脆弱性(CVE-2013-2556)概要

- CanSecWest2013で発表された脆弱性
  - この脆弱性だけで攻撃が成立するわけではない（他の脆弱性と組み合わせる）
  - この問題により、特定の脆弱性があった場合に、ASLRをバイパスして攻撃可能

## CanSecWest2013での発表内容

- この脆弱性は「DEP/ASLR bypass without ROP/JIT」というタイトルでCanSecWest2013にて、Yang Yu氏により発表された
- 大きく二つの問題について発表
  - 32bit版Windowsにて、KiFastSystemCallへのポインタが固定アドレスに存在する
  - 64bit版Windows上の32bitプロセスにて、LdrHotPatchRoutineへのポインタが固定アドレスに存在する

なぜ、これらの固定アドレスが問題なのか



Use-after-freeまたは、ヒープオーバーフローにより、C++オブジェクトのvtableポインタの書き換えが起きる場合にASLRをバイパスして攻撃可能

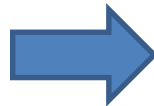
vtableポインタが書き換えられるとは？

## 前提知識：C++のオブジェクトレイアウト

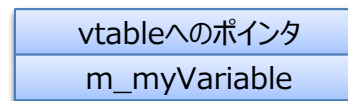
- 一般的なC++の実装による、オブジェクトのレイアウト

```
// メンバ関数が” virtual” であることに注意
class MyClass {
public:
    MyClass();
    virtual ~MyClass();
    virtual void doWork();
private:
    int m_myVariable;
};
```

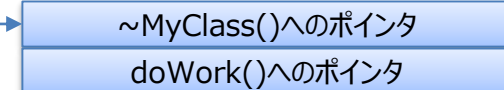
インスタンス化



MyClassオブジェクト



MyClassクラスのvtable



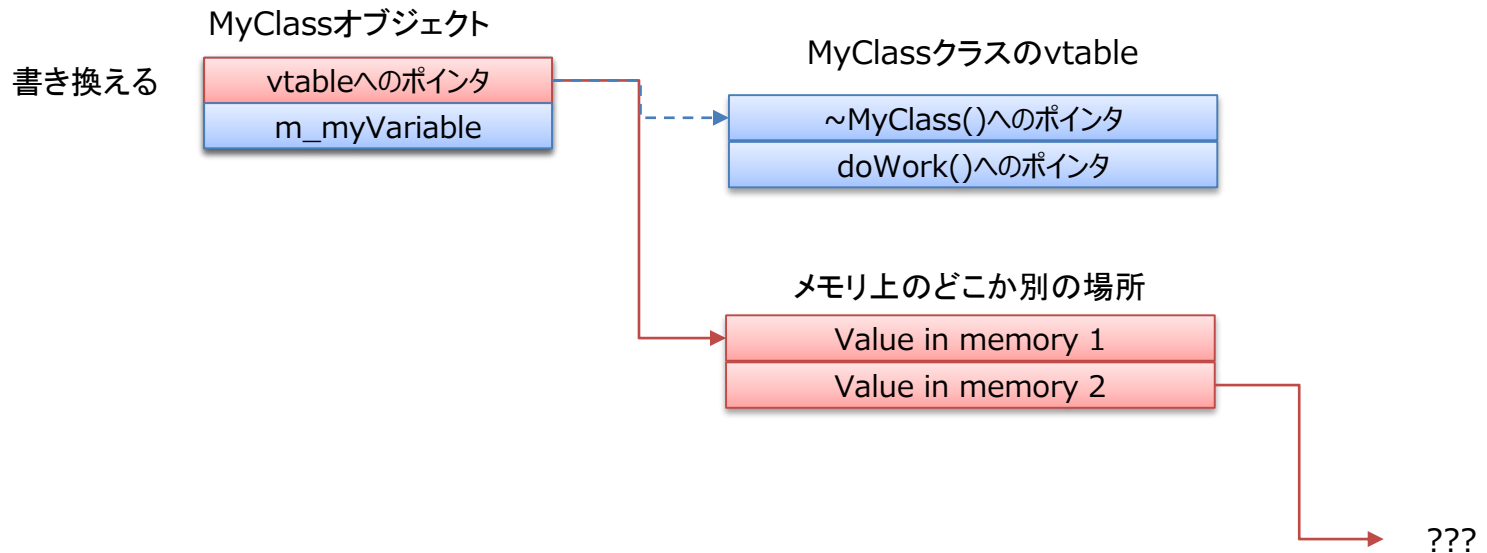
- メンバ関数の呼び出し方

```
// doWork() を呼び出す場合のコード
// ecxにMyClassオブジェクトのアドレスがセットされている
mov  eax, dword ptr [ecx] // eaxにvtableのアドレス
push ecx // 関数呼び出しの引数(*)
call dword ptr [eax+4] // doWork()呼び出し(vtableよりアドレス取得)
```

\* 呼び出し規約がcdeclの場合、第一引数にthisポインタを渡す

## vtableの書き換えの問題

- オブジェクトのvtableへのポインタが書き換わるとどうなるか？



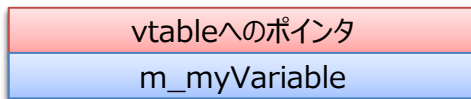
```
//前スライドと同じコード
//vtableへのポインタを書き換えると、実行される関数が変わる
mov  eax, dword ptr [ecx] // eaxにvtableのアドレス
push ecx                // 関数呼び出しの引数
call dword ptr [eax+4]  // Value in memory 2が指す場所を実行する
```

## 固定アドレスにKiFastSystemCallへのポインタがある場合

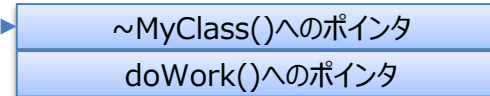
- オブジェクトのvtableの位置をKiFastSystemCallを呼び出すように上書き
- KiFastSystemCallはWindowsが利用するシステムコール呼び出しの共通コード
- ASLRは特に意味をなさない

脆弱性(use-after-free /  
heap overflow)  
を利用し固定値で書き換える

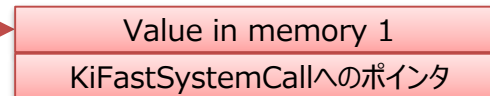
MyClassオブジェクト



MyClassクラスのvtable



Fixed address



KiFastSystemCall

```

mov  eax, dword ptr [ecx] // eaxにvtableのアドレス
push ecx                // KiFastSystemCallの引数
call dword ptr [eax+4]  // KiFastSystemCallを呼び出す
  
```



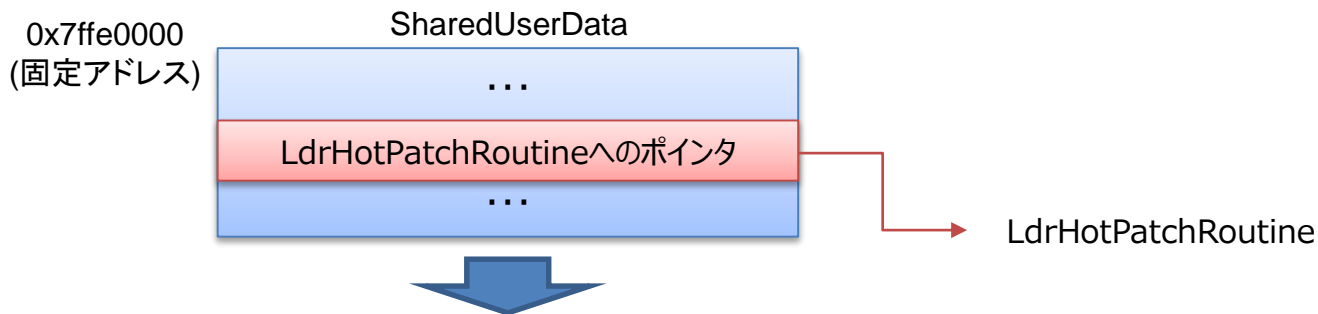
KiFastSystemCallが呼び出される  
ただし、引数を取るシステムコールを攻撃者が意図したとおりに呼び出すのは難しい

## LdrHotPatchRoutineの利用

- 64bit WindowsにはKiFastSystemCallへのポインタが固定位置に**存在しない**
- ただし、64bit Windows上の32bitプロセスには、LdrHotPatchRoutineへのポインタが固定位置に存在する
- LdrHotPatchRoutineは内部で引数に渡されたDLLをロードする

```
struct HotPatchBuffer {  
    ...  
    USHORT PatcherNameOffset; // ロードするDLL名へのオフセット  
    USHORT PatcherNameLen;   // ロードするDLL名の長さ  
    ...  
};  
void LdrHotPatchRoutine( struct *HotPatchBuffer);
```

- LdrHotPatchRoutineへのポインタはSharedUserData内のデータとしてすべての64bit Windows上の32bitプロセスプロセスに存在
- SharedUserDataは固定アドレス(0x7ffe0000)に存在

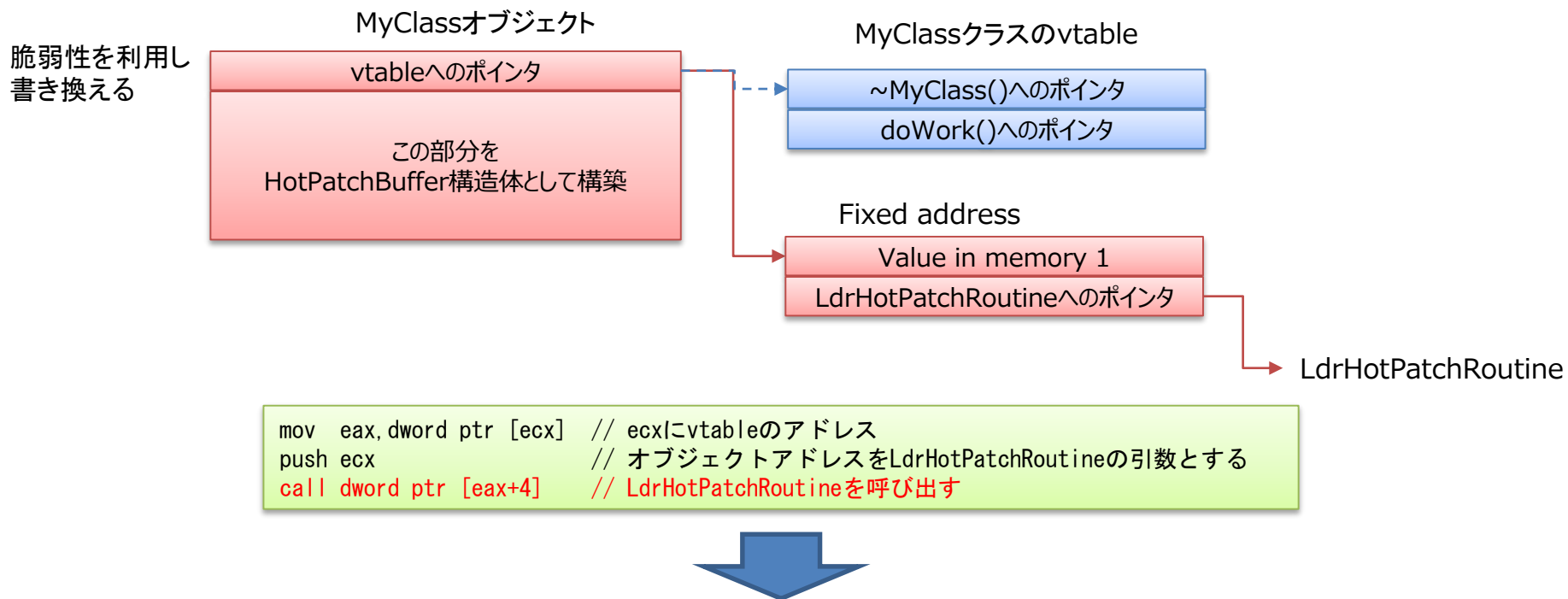


LdrHotPatchRoutineへのポインタをvtableが含むようにC++オブジェクトを上書き  
DLLのロードが可能



## 固定アドレスにLdrHotPatchRoutineへのポインタがある場合

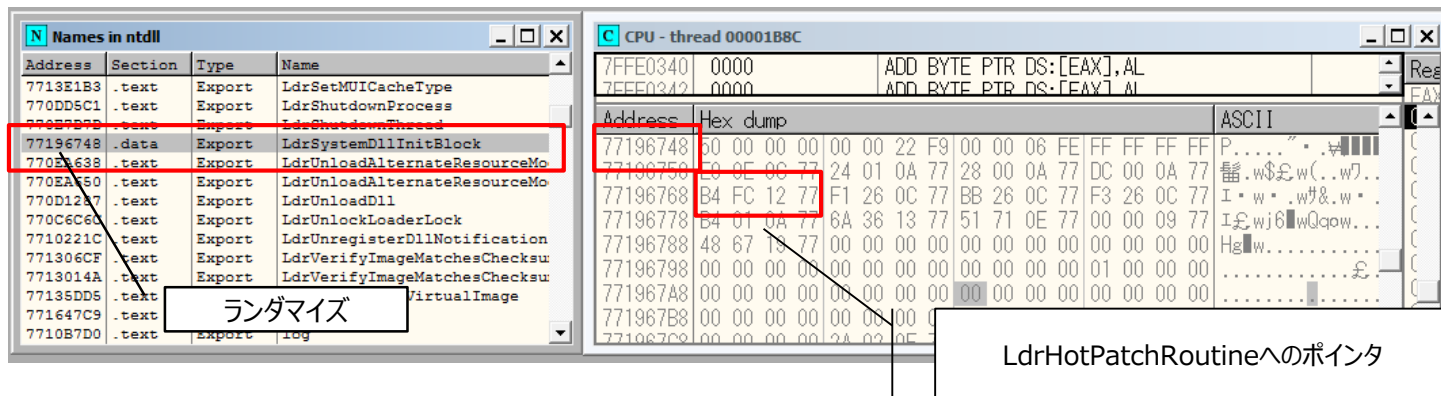
- オブジェクトのvtableの位置をLdrHotPatchRoutineを呼び出すように上書き



- LdrHotPatchRoutineへの引数(DLL名など)を¥¥192.168.1.100¥share¥hoge.dllなどとする事でサーバー上のDLLをロード可能
- オブジェクトはvtableへのポインタの書き換え時に任意の値に上書き可能であることに注意

## MS13-063による変更点

- MS13-063では、LdrHotPatchRoutineのアドレスが固定アドレスに保存されないように修正
  - SharedUserData内から、関数テーブルを削除
  - ntdll.dllのデータセクションに移し、LdrSystemDllInitBlockとしてエクスポート



The screenshot shows two windows from a debugger:

- Names in ntdll:** A table of exports with columns Address, Section, Type, and Name. The entry for `LdrSystemDllInitBlock` at address `77196748` is highlighted with a red box.
- CPU - thread 00001B8C:** A window showing assembly instructions and a hex dump. The hex dump shows a pointer value `00 00 00 00` at address `77196748`, which is highlighted with a blue box. An arrow points from a label `LdrHotPatchRoutineへのポインタ` to this value.

A label `ランダムサイズ` points to the `VirtualImage` entry in the exports list.

- ntdll.dllはASLRが有効なモジュールであるため、この関数テーブルもアドレスが固定されない



ASLRをバイパスし、LdrHotPatchRoutineを利用したDLLのロードはできない

## 参考資料

- <http://technet.microsoft.com/ja-jp/security/bulletin/ms13-063>
- <http://cansecwest.com/slides/2013/DEP-ASLR%20bypass%20without%20ROP-JIT.pdf>
- <http://blogs.technet.com/b/srd/archive/2013/08/12/mitigating-the-ldrhotpatchroutine-dep-aslr-bypass-with-ms13-063.aspx>
- <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2556>



## Contact Information

E-Mail : [research-feedback@ffri.jp](mailto:research-feedback@ffri.jp)

Twitter : [@FFRI\\_Research](https://twitter.com/FFRI_Research)