JVMをクラッシュさせてみる

菅原泰樹

自己紹介

- アリエルという会社で働いてます
- Javaプログラマです
- Javaコミュニティへの貢献はほとんどないです
- pdfboxで日本語表示できるパッチを作ってみたことならあります
- Emacsコミュニティには少し貢献してます
- TwitterIDはbuzztaikiです

それではじめます

ある日会社の偉い人に
いわれました

JVMを落としてみろ

僕は今までJVMを落とした
ことはありません

Oracleなら何回か落としたことがあります

でもJVMはありません

JVMは簡単に落ちるのか？

実は結構簡単に落ちる
みたいです

これからいくつか落とし方を紹介します

JVMの落とし方その１

JNIを使って落とす

JNI使えば簡単に落ちます

卑怯だなんて言わないでください

JVM落としはJNIにつながるのがほとんどです

基本は大事です

それでは

その１

NULLにアクセスして落とす

JNIでNULLにアクセスすれば一発です

ぬるぽなんておきません

サンプル

CrashJni1.c

```
#include <jni.h>
#include <stdio.h>
#include "CrashJni1.h"

JNIEXPORT void JNICALL Java_CrashJni1_test(JNIEnv *env, jobject obj ) {
    jclass integer_cls = (*env)->FindClass(env, "Ljava/lang/Integer;");
    (*env)->CallObjectMethod(env, (jobject) NULL,
        (*env)->GetMethodID(env, integer_cls, "toString", "()V"));
}
```

CrashJni1.java

```
public class CrashJni1 {
    static {
        System.loadLibrary("CrashJni1");
    }

    public static void main(String[] args) {
        CrashJni1 o = new CrashJni1();
        o.test();
    }

    public native void test();
}
```

ふつうにアクセスバイオレーションです

簡単ですね

その２

違うクラスのオブジェクトを返す

JNIだと宣言されてない
型をかえせます

型チェックなんてされません

これを使って落とします

サンプル

CrashJni2.c

```
#include <jni.h>
#include <stdio.h>
#include "CrashJni1.h"

JNIEXPORT jobject JNICALL Java_CrashJni2_test (JNIEnv* env, jobject obj)
{
  jclass integer_cls = (*env)->FindClass(env, "Ljava/lang/Integer;");

  jobject illegal_date = (*env)->NewObject(env, integer_cls,
      (*env)->GetMethodID(env, integer_cls, "<init>", "(I)V"),
      10);
  return illegal_date;
}
```

CrashJni2.java

```
public class CrashJni2 {
public static void main(String[] args) throws Exception {
        CrashJni2 o = new CrashJni2();
        Date d = o.test();
        System.out.println(d);

        System.out.println("get class");
        System.out.println(d.getClass());

        System.out.println("call same declaration method");
        System.out.println(d.toString());

        System.out.println("call method as integer");
        Integer i = (Integer) ((Object) d);
        System.out.println(i.intValue());

        System.out.println("call method");
        System.out.println(d.getTime());
    }

    public native Date test();
}
```

やっぱりアクセスバイオレーションです

おもしろいことに、Integerとして扱う分にはエラーになりません

Integerを返してるんだから当然といえば当然ですね。

共通のメソッドを読んだときに落ちないのもおもしろいです

不思議ですね

JVMの落とし方その２

ふつうのJavaのコードで落とす

今度はふつうにJavaだけで落とします

一生けんめいバグデータベースを探しました

ここです

http://bugs.sun.com/

どうやらAWT関係で落ちることが多そうです

けどなかなか再現してくれません

これでは上司に怒られてしまいます

そんな中で再現できた
バグを一つ

Swingで落とす

BugID: 6699846

Crash rendering some combinations of Devanagari text

Devanagariとはインドの方の文字らしいです

それでは

サンプル

CrashUseSwing.java

```
import javax.swing.*;
import java.io.*;

public class CrashUseSwing {
    public static void main(String[] args) throws Exception {
        String text = "¥u0903¥u0951¥u0954";
        OutputStream out = new FileOutputStream("CrashUseSwing.txt");
        out.write(text.getBytes("UTF-8"));
        out.close();

        JOptionPane.showMessageDialog(null, text);
    }
}
```

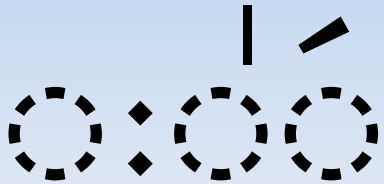
たったこれだけです

フォントのレンダリング関係らしいですが

理由はさっぱりわかりません

たぶんSwingは使うなってことでしょう

ちなみに表示しようと
おもっている文字は



でした。

その２

UnSafeを使って落とす

com.sunパッケージにUnsafeというクラスがあります

Javaから直接ヒープを触れる面白いクラスです

面白いですが当然危険です

これを使って落としてみます

サンプル

CrashUseUnsafe.java

```
import sun.misc.Unsafe;
import java.lang.reflect.*;

public class CrashUseUnsafe {
    public static Unsafe getUnsafe() throws Exception {
        Field f = Unsafe.class.getDeclaredField("theUnsafe");
        f.setAccessible(true);
        return (Unsafe) f.get(null);
    }

    public static void main(String[] args) throws Exception {
        Unsafe unsafe = getUnsafe();
        unsafe.putAddress(0, 0);
    }
}
```

putAddressで直接アドレスを指定して書き込んで一発です

怖いですね

ついでにUnsafeの正しい使い方

Unsafeを使うとアロケートされた直後のオブジェクトを作れます

これを利用するとfinalな変数をいじれたりします

java.io.ObjectStreamClassがデシリアライズのために使ってるそうです

サンプル

UseUnsafe.java

```
import sun.misc.Unsafe;
import java.lang.reflect.*;

public class UseUnsafe {
    public final String s;
    public UseUnsafe() {
        s = "hoge";
    }

    public static Unsafe getUnsafe() throws Exception {
        Field f = Unsafe.class.getDeclaredField("theUnsafe");
        f.setAccessible(true);
        return (Unsafe) f.get(null);
    }

    public static void main(String[] args) throws Exception {
        System.out.println(new UseUnsafe().s);
        Unsafe unsafe = getUnsafe();
        UseUnsafe o2 = (UseUnsafe) unsafe.allocateInstance(UseUnsafe.class);
        unsafe.putObject(o2, unsafe.objectFieldOffset(UseUnsafe.class.getField("s")), "fuga");
        System.out.println(o2.s);
    }
}
```

こんな風にfinalなフィールドを書き換えられます

面白いですね

その３

ヒープを使い切って落とす

ふつうJavaがヒープを使い切ると

OutOfMemoryErrorがでます

しかし

出さずに落とす方法も
あるようです

ほとんど魔法です

サンプル

CrashWithHeap.java

```
public class CrashWithHeap {
    public static void main(String[] args) {
        Object[] o = null;
        for (;;) {
            o = new Object[] {o};
        }
    }
}
```

まったく意味がわかりません

配列の代わりにListを使ったらふつうにエラーでした

配列は特別扱いされてるんでしょうか

だれか詳しい人がいたら懇親会で教えてください

以上でJVMの落とし方をおわります

ちなみに

自分で見つけたバグは
0個でした

おしまい

ご清聴ありがとうございました。