Swf decrypt

By 云柠  &  入侵 of 阿里基础安全威胁情报中心

攻击在持续，攻击的技术在演进。防御者需要持续的跟进研究和投入。下面是我们小伙伴的技术分享。

最近Flash0day频繁出现，让我们更多的目标转移到flash上。Flash作为脚本语言，可以编译，不少的恶意flash文件通过各种加密、混淆、动态解密来对抗人工分析，引擎检测。通过动态执行，常见的动态解密的swf都可以解决。不过遇到很多条件判断、环境依赖的限制，往往又无法稳定dump出解密的swf文件和关键的数据(shellcode)。如果能在静态的方法对混淆、加密的swf文件直接进行反混淆、反编译，则将能有效提升检测效果。

市面上可以对swf进行加密的软件很多，如swfencrypt、doswf、secureswf、dcomsoft等，我们这次仅对doswf以及secureswf进行简单的分析。

恶意的SWF常常通过doswf加密和secure swf混淆，真正利用的部分被加密、混淆，通过研究doswf的加密方式以及secureswf的混淆方式，可以直接通过静态的方法进行相应的解密和反混淆，直接检测最核心的恶意代码部分，有效提升引擎检测率。（用户可使用文件B超系统 http://b-chao.com 和云盾-星云APT检测产品享受到我们的技术成果）

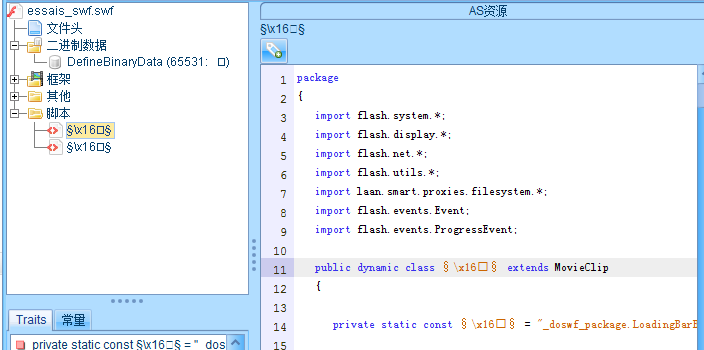
## doswf

[doswf](http://www.doswf.cn/)是国人开发的一款flash加密混淆软件，从官网可以看出该软件已经停止了开发，当前最新版本为5.4.3。

我们对比了版本4.9.7和最新版本之间加密效果的区别，发现两者相差不大，在介绍完整体的加密流程之后，再将它们之间的差异展示出来。

### Crypt Flow

经过doswf加密的swf文件，都存在DefineBinaryData的tag，因为原始swf文件将会存放在这个二进制数据中，二进制数据经过解密后通过loadbytes进行加载。



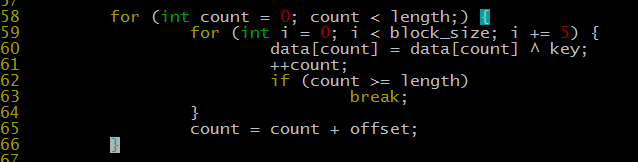
这个二进制数据的头部结构如下(B标示字节)

|  |  |  |  |
| --- | --- | --- | --- |
| 1B | 1B | 4B | 4B |
| block\_sz | key | offset | length |

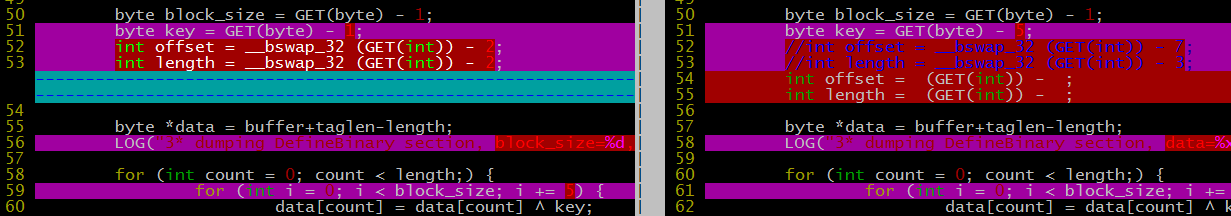
原始swf文件从起始以offset为步长，每次对block\_sz的块进行处理，块中的数据并不是逐一字节处理的，而是依据一个特定值跳跃处理，在整个文件处理完成后进行压缩，压缩后的数据再加上上面的头部信息就形成了完整的二进制数据了。

上面的描述简化了一些不重要的因素，如doswf加密时还会嵌入两个额外的swf文件，所以当解密二进制数据后，会发现解密后的数据中包含三个swf，这时只要根据硬编码特征”FWS”、”CWS”就可以将所有包含的swf文件完整dump出来。

关键解密代码：

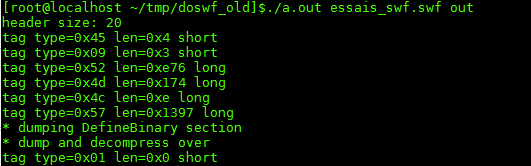


版本4.9.7和最新版本解密代码的差异：

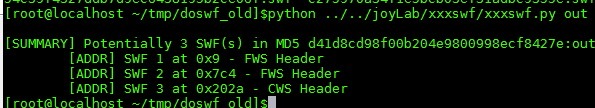


可以看到，差异在于头字节结构的几个值得获取、大小端以及跳跃值的不同。

我们拿CVE-2012-0779样本（md5：2b98d285c8b581855d59ac368956ee78）进行测试，这是一个doswf4.9.7版本加壳的样本：



检测dump出的数据，可以发现里面包含有三个swf文件。



### Decrypt Code

目前放出针对旧版的解密代码(新版自行解决，根据前面的提示很容易更改)，仅支持解压后的doswf解密

/\*

\* Decoder for DoSWF.

\* This extracts the DefineBinary tag from argv[1] to argv[2]. To un-zlib that using python:

\*

\* python -u -c \

\* "import sys, zlib; sys.stdout.write(zlib.decompress(sys.stdin.read()))" \

\* < definebinary.z > definebinary.txt

\*

\* Then trim off the first 6 bytes to get to the SWF header.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <byteswap.h>

#include <zlib.h>

#pragma warning(disable:4996)

#ifdef \_DEBUG

#define DBGBREAK() \_\_debugbreak()

#else

#define DBGBREAK()

#endif

#define LOG(...) fprintf(stderr, \_\_VA\_ARGS\_\_)

//#define DIE(...) (LOG(\_\_VA\_ARGS\_\_), DBGBREAK(), exit(1), 0)

#define DIE(...) (LOG(\_\_VA\_ARGS\_\_), exit(1),1)

typedef unsigned char byte;

void decompress\_string(byte\* data, int length, byte \*out, int \*outlength, int avaout);

void handle\_DefineBinary(FILE \*fi, FILE \*fo, int taglen, byte \*buffer)

{

LOG("\* dumping DefineBinary section\n");

byte \*end = buffer + taglen;

byte \*ptr = buffer, \*old;

#define GET(type) ( \

((ptr+sizeof(type) >= end) ? DIE("eof\n") : 0), \

old = ptr, \

ptr += sizeof(type), \

\*(type \*)old)

short tag = GET(short);

int reserved = GET(int);

LOG("2\* dumping DefineBinary section,tag=%u,reserved=%d\n", tag, reserved);

byte block\_size = GET(byte) - 1;

byte key = GET(byte) - 1;

int offset = \_\_bswap\_32 (GET(int)) - 2;

int length = \_\_bswap\_32 (GET(int)) - 2;

byte \*data = buffer+taglen-length;

LOG("3\* dumping DefineBinary section, block\_size=%d,offset=%d,data[0]=%x\n", block\_size, offset,data[0]);

for (int count = 0; count < length;) {

for (int i = 0; i < block\_size; i += 5) {

data[count] = data[count] ^ key;

++count;

if (count >= length)

break;

}

count = count + offset;

}

LOG("4\* dumping DefineBinary section\n");

//fwrite(data, 1, length, fo);

LOG("5\* dumping DefineBinary section\n");

int outl = length\*5;

byte \*outd = new byte[outl];

decompress\_string(data, length, outd, &outl, outl);

fwrite(outd, 1, outl, fo);

}

void decompress\_string(byte\* data, int length, byte \*out, int \*outlength, int avaout)

{

z\_stream zs; // z\_stream is zlib's control structure

memset(&zs, 0, sizeof(zs));

if (inflateInit(&zs) != Z\_OK)

LOG("init error!\n");

zs.next\_in = data;

zs.avail\_in = length;

int ret;

zs.next\_out = out;

zs.avail\_out = avaout;

ret = inflate(&zs, 0);

inflateEnd(&zs);

if (ret != Z\_STREAM\_END) { // an error occurred that was not EOF

LOG("inflate end error,ret=%d!\n", ret);

}

\*outlength = zs.total\_out;

return;

}

int main(int argc, char \*argv[])

{

if (argc != 3)

DIE("syntax: UndoSWF <input.swf> <output.as>\n");

FILE \*fi = fopen(argv[1], "rb");

if (!fi)

DIE("can't open %s\n", argv[1]);

FILE \*fo = fopen(argv[2], "wb");

if (!fo)

DIE("can't create %s\n", argv[2]);

char header[9];

if (fread(header, 1, 9, fi) != 9)

DIE("can't read header\n");

if (memcmp(header, "FWS", 3))

DIE("invalid header\n");

int rectbits = header[8] >> 3;

int hdrbits = 8\*8 + 5+rectbits\*4;

int hdrbytes = (hdrbits+7)/8 + 4;

LOG("header size: %d\n", hdrbytes);

if (fseek(fi, hdrbytes-9, SEEK\_CUR) < 0)

DIE("can't skip %d bytes\n", hdrbytes-9);

for (;;) {

unsigned short hdr;

if (fread(&hdr, 1, 2, fi) != 2)

DIE("can't read tag hdr\n");

if (hdr == 0) break;

int taglen = hdr & 0x3f;

int tagtype = hdr >> 6;

bool longtag = taglen == 0x3f;

if (longtag) {

if (fread(&taglen, 1, 4, fi) != 4)

DIE("can't read tag len\n");

}

LOG("tag type=0x%02x len=0x%x %s\n", tagtype, taglen, longtag ? "long" : "short");

byte \*buffer = new byte[taglen];

if (fread(buffer, 1, taglen, fi) != taglen)

DIE("can't read tag data\n");

if (tagtype == 0x57) handle\_DefineBinary(fi, fo, taglen, buffer);

delete buffer;

}

fclose(fi);

fclose(fo);

return 0;

}

测试的样本由MD5: 2b98d285c8b581855d59ac368956ee78 解压之后的样本

## SecureSwf

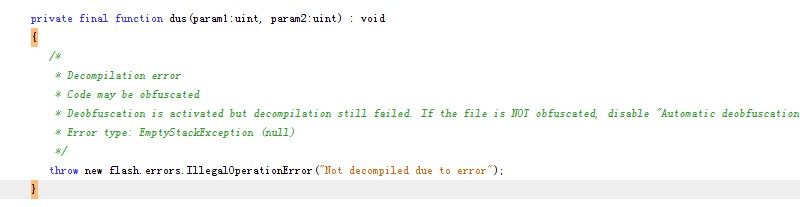
Secureswf加密软件，号称break掉所有的flash反编译软件，支持对资源加密、 支持对as code级别做混淆，通过添加各种jump 、反复跳转指令、垃圾指令填充 阻止反编译软件反编译 增加分析人员分析出具体的actionscript原理难度。

之前拿到CVE-2015-3105 样本就是用secure swf混淆过的，刚好可以分析下

MD5: 58d1022923950ad1452c72f46b1ee3d0

### DUS function

以这个dus函数为例做反混淆



对应的abc code

code

getlocal\_0

pushscope

pushbyte 0

newfunction 30

pop

jump ofs0017

convert\_i

declocal 4

urshift

rshift

declocal\_i 2

declocal 2

astypelate

newactivation

increment

ofs0017:getlocal 4

iffalse ofs0025

decrement\_i

increment\_i

increment\_i

increment\_i

pushbyte 7

multiply\_i

decrement\_i

ofs0025:setlocal\_3

getlocal 5

iftrue ofs0065

getlocal\_1

getlocal 5

iftrue ofs00ac

pushbyte 4

getlocal 5

iffalse ofs005a

increment\_i

pushbyte 52

subtract\_i

pushbyte 114

add\_i

pushbyte 49

jump ofs0055

nextvalue

setlocal\_2

setlocal\_3

kill 3

declocal 4

setlocal 4

declocal 4

getlocal\_3

istypelate

ofs0055:add\_i

decrement\_i

pushbyte 15

subtract\_i

ofs005a:modulo

getlocal 4

iftrue ofs00ab

iffalse ofs0065

ofs0065:getlocal\_1

getlocal 4

iftrue ofs00ac

getlex Qname(PrivateNamespace("dphgjxukp"),"var\_5")

pushbyte 8

getlocal 4

iffalse ofs0092

pushbyte 39

multiply\_i

jump ofs008b

inclocal 2

inclocal\_i 2

declocal 3

getlocal\_3

declocal\_i 2

convert\_i

multiply\_i

inclocal\_i 4

increment\_i

ofs008b:negate\_i

negate\_i

pushbyte 115

subtract\_i

increment\_i

decrement\_i

ofs0092:add

subtract

getlocal 5

not

iffalse ofs00ab

pushbyte 4

getlocal 4

iffalse ofs00aa

increment\_i

pushbyte 35

multiply\_i

pushbyte 29

subtract\_i

ofs00aa:divide

ofs00ab:convert\_u

ofs00ac:setlocal\_3

getlocal 4

iftrue ofs00b9

getlex Qname(PrivateNamespace("dphgjxukp"),"var\_1")

getlocal\_3

getlocal\_2

setproperty MultinameL([PrivateNamespace("dphgjxukp"),ProtectedNamespace("dphgjxukp"),StaticProtectedNs("dphgjxukp"),StaticProtectedNs("flash.display:Sprite"),StaticProtectedNs("flash.display:DisplayObjectContainer"),StaticProtectedNs("flash.display:InteractiveObject"),StaticProtectedNs("flash.display:DisplayObject"),StaticProtectedNs("flash.events:EventDispatcher"),StaticProtectedNs("Object"),PackageNamespace("flash.display"),PackageNamespace("flash.text"),PackageNamespace("flash.utils"),PackageNamespace("flash.net"),PackageNamespace("flash.system"),PackageNamespace(""),PackageInternalNs(""),PrivateNamespace("FilePrivateNS:dphgjxukp"),Namespace("http://adobe.com/AS3/2006/builtin")])

ofs00b9:returnvoid

returnvoid

#### Type1 jump

getlocal\_0

pushscope

pushbyte 0

newfunction 30

pop

jump ofs0017

convert\_i

declocal 4

urshift

rshift

declocal\_i 2

declocal 2

astypelate

newactivation

increment

ofs0017:getlocal 4

jump之间都是垃圾指令 可以直接改成

getlocal\_0

pushscope

pushbyte 0

newfunction 30

pop

getlocal 4

把这段abccode中所有jump之间的直接去掉

#### Type2 getlocal n

混淆指令里面充斥着getlocal n指令 获取寄存器的value 并PUSH STACK，紧接着跟着判断

如

getlocal 4 // reg4.value push stack

iffalse ofs0013 // pop reg4.value judge this value if is false

pushbyte 0

newfunction 30

pop

getlocal 4

iffalse ofs0013

increment\_i

increment\_i

pushbyte 7

multiply\_i

decrement\_i

ofs0013:setlocal\_3

local4 初始化值应该为0 ，则可以直接简化成

pushbyte 0

newfunction 30

pop

setlocal\_3

后面以此类似

getlocal 5

iftrue ofs0053

local5 为0，则不会跳转

直接简化这两条指令

所以没有经过setlocal n的n对应的getlocal n获取的值应该都是0，带着这样的设定来精简指令就很容易了

getlocal 13

not

iffalse ofs00d9 //条件不成立 直接去掉这样的三条指令

getlocal 13

iftrue ofs00ed //条件不成立 直接去掉这样的2条指令

getlocal 12

iffalse ofs0159//条件成立 直接去掉当前2条指令到ofs0159 直接的所有指令(中间代码没有其他地方跳转过来)

#### Type3 garbage ins

getlocal\_1

pushbyte 4

modulo

iffalse ofs0053

ofs0053:

local1 为0，modulo之后 0/4 为0 ，push stack，iffalse pop stack，条件成立，无意义片段。

还有一些

decrement\_i

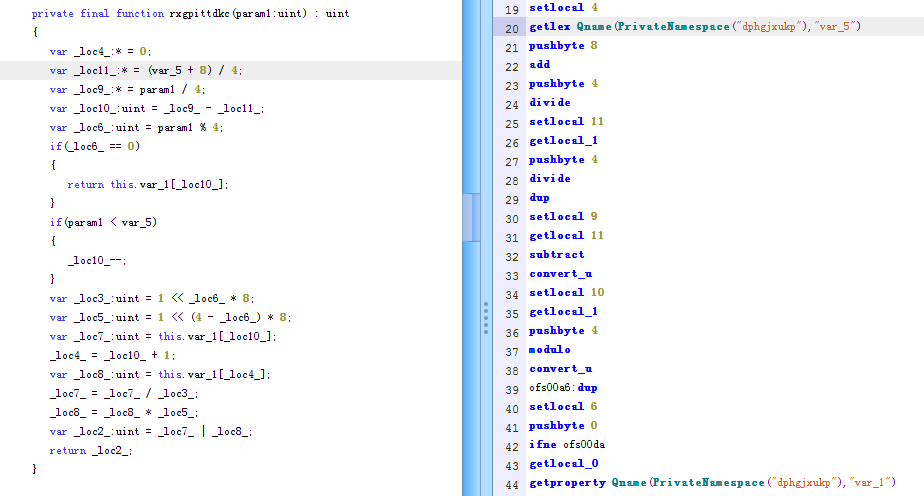
increment\_i

#### decrypt dus function



### Rxgpittdkc function

在以这个函数为例，按照上面的步骤，先去除jump 垃圾指令，在去除getloca n之类的垃圾指令



### Dphgjxukp function

#### Type4 kill ins

getlocal\_1 // reg1.value push stack

getlocal\_0 // reg0.value push stack

getlocal\_2 // reg2.value push stack

kill 1 //kill reg1.value

kill 0 //kill reg0.value

kill 2 //kill reg2.value

setlocal\_2 //pop stack,set value to reg2.value

setlocal\_0 //pop stack,set value to reg0.value

setlocal\_1 //pop stack,set value to reg1.value

这样的指令集也是垃圾指令

#### Type5 as function flow change

Secure Swf会对简单的as 函数调用从abc 层面流程进行混乱

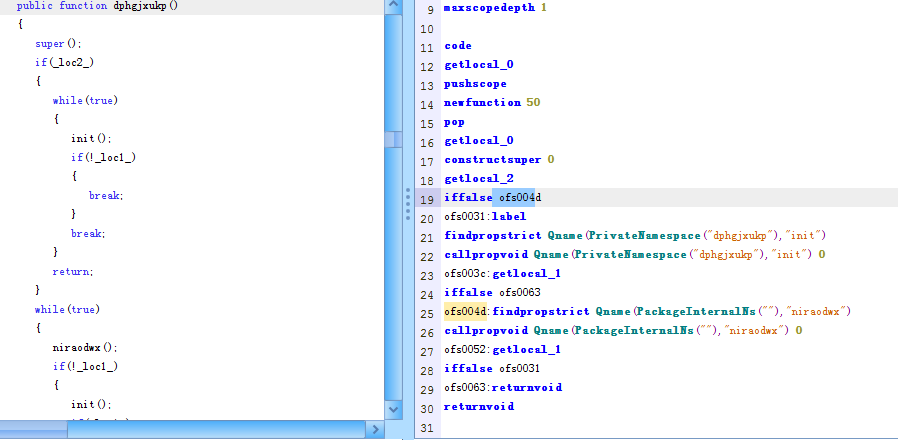
比如这个dphgjxukp 函数 还原之后真正的代码如下

niraodwx();

init();

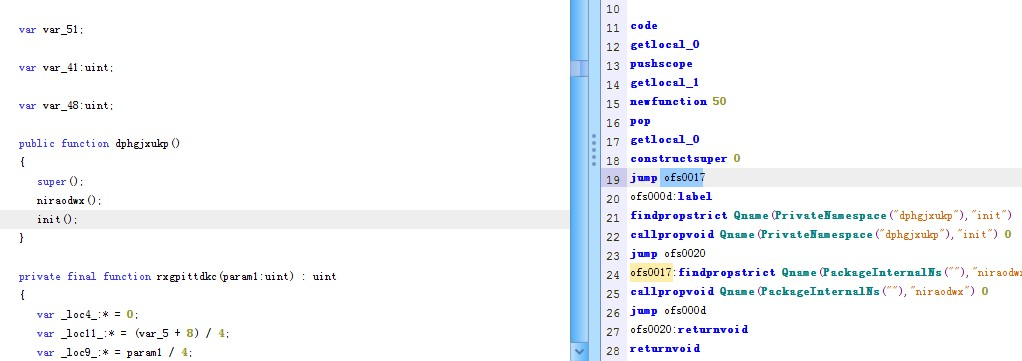
return;

解混淆之后



可以看到jpexs对这样的反编译是混乱的，可以把相应的跳转换成jump

比如getlocal\_2 iffalse 换成jump ofs0044 ,以此类似，最终得到的反编译结果



可以看到这个swf混淆的还是很容易解密出来的，更多的人工体力活可以通过编写自动化的脚本来实现。(有钱的主可以购买avs actionscript view的149USD的插件 直接反编译secureswf)

## 关于阿里基础安全威胁情报中心

阿里基础安全威胁情报中心是2015年6月新成立的部门，以沉淀阿里的安全数据、安全能力和建设阿里安全的威胁情报体系为使命。欢迎志同道合的小伙伴们加盟。