# Transforming Open Source to Open Access in Closed Applications

## FINDING VULNERABILITIES IN ADOBE READER'S XSLT ENGINE

# Introduction – Brian Gorenc

- MS in Software Engineering – Southern Methodist University
- BS in Computer Engineering – Texas A&M University
- Director of Vulnerability Research at Trend Micro
  - Leader of the Zero Day Initiative
  - Organize Pwn2Own Hacking Competition
  - Focus on Vulnerability Research and Discovery
- Past Experiences
  - Lead Software Engineer at Lockheed Martin
- Past research:
  - Microsoft Bounty Submission
  - Patents on Exploit Mitigation Technologies
  - Adobe Reader Research
- Twitter: @MaliciousInput

# Introduction - Abdul-Aziz Hariri

- BS in Computer Sciences – University of Balamand
- Currently a Senior Security Researcher at ZDI
  - Root Cause analysis / Vulnerability Research / Exploit development
  - Pwn2Own Preparation / Judging entries
- Past Experiences
  - Bits Arabia, Insight-Tech and Morgan Stanley
- Past research:
  - Pwn4Fun 2014 renderer exploit writer
  - Microsoft Bounty submission
  - Patents on Exploit Mitigation Technologies
  - Adobe Reader research
- Twitter: @abdhariri

# Introduction – Jasiel Spelman

- BA in Computer Sciences – University of Texas
- Currently a Senior Security Researcher at ZDI
  - Root Cause analysis / Vulnerability Research / Exploit development
  - Pwn2Own Preparation / Judging entries
- Past Experiences
  - AlterPoint
- Past research:
  - Pwn4Fun 2014 Broker exploit writer
  - Windows Kernel information leaks
  - Adobe Flash research
- Twitter: @WanderingGlitch

# Understanding the Attack Surface

# Open-source Components

- Integrating open-source projects into large closed-source applications is becoming common practice
  - Quick add functionality without costly engineering efforts
- Security side effects not well understood by vendors
  - Misunderstood assumptions
  - Poorly implemented components
  - Missing security mitigations

# Case Study: Adobe Reader

- Widely used document reader
  - Large number of features to aid document authors
- Highlights some of these interesting security side effects
- Integrates several open-source projects
  - Modified version of libtiff for Tagged Image File Formation (TIFF)
  - Modified version of Sablotron for Extensible Stylesheet Language Transformations (XSLT) handling
- Sablotron is now abandoned with no community to implement fixes

# Verifying Sablotron in Reader

- Reader uses Sablotron 1.0.2
- Compiled into AXSLE.dll and found in Reader's install directories

```
.text:10026F9E loc_10026F9E:                           ; CODE XREF: sub_100252EB+1C1F↑j
.text:10026F9E                 call    sub_1002A1F6
.text:10026FA3                 mov     ecx, esi
.text:10026FA5                 cmp     edi, [eax+1BCh]
.text:10026FAB                 jnz     short loc_10026FE0
.text:10026FAD                 push    dword ptr [ebp+var_48+4]
.text:10026FB0                 call    sub_1002A1F6
.text:10026FB5                 mov     ecx, eax
.text:10026FB7                 call    sub_1003399E
.text:10026FBC                 push    offset aVersion_0 ; "version"
.text:10026FC1                 mov     ecx, eax
.text:10026FC3                 mov     edx, [eax]
.text:10026FC5                 call    dword ptr [edx+4]
.text:10026FC8                 push    eax             ; char *
.text:10026FC9                 call    _strcmp
.text:10026FCE                 pop     ecx
.text:10026FCF                 pop     ecx
.text:10026FD0                 mov     ecx, ebx
.text:10026FD2                 test    eax, eax
.text:10026FD4                 jnz     short loc_10026F94
.text:10026FD6                 push    offset a1_0_2   ; "1.0.2"
.text:10026FDB                 jmp     loc_1002706A
```

# XSLT Overview

- Developed in 1998-1999
  - Part of W3C's Extensible Stylesheet Language (XSL)
- Used to transform XML documents into other formats
  - HTML
  - Plain Text
  - XSL Formatting Objects

# XSLT Logic Elements

- Used to describe the transformations to be applied
- `xsl:for-each`
  - Select every XML element of a specified node-set
- `xsl:attribute`
  - Creates an attribute node and attaches it to an output element
- `xsl:copy`
  - Copies current node from the source
- `xsl:value-of`
  - Extract the value of an XML element and add it to the output

# XPath Functions

- XPath is a query language used to for nodes selection and calculations
  - `/music/artist[1]` – Selects the first artist element that is child of music element
- XPath functions add flexibility and power to Xpath
  - *Node-Set*
  - *String*
  - *Boolean*
  - *Number*

# XSLT Transformation Example

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
</catalog>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
<html>
<body>
 <h2>My CDs</h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th style="text-align:left">Title</th>
   <th style="text-align:left">Artist</th>
  </tr>
  <xsl:for-each select="catalog/cd">
  <tr>
   <td><xsl:value-of select="title"/></td>
   <td><xsl:value-of select="artist"/></td>
  </tr>
  </xsl:for-each>
 </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

**My CDs**

| Title | Artist |
|---|---|
| Empire Burlesque | Bob Dylan |

# Adobe Reader's XSLT Engine

- Uses a modified version of Sablotron
- XML and XSL can be parsed a couple ways
  - XFA
  - JavaScript APIs

```
var xmlDoc = "<x/>";
var xslDoc = "<xsl:stylesheet xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\"
              version=\"1.0\"> <xsl:template match=\"/\">  <xsl:for-each select=\"a\">
              <xsl:sort select=\"(/)/*[a()]\"/>
              </xsl:for-each>   </xsl:template></xsl:stylesheet>";

var x = XMLData.parse(xmlDoc, false);
var oNode = x.nodes.item(0);
oNode.applyXSL(xslDoc);
```

# Sablotron's XSLT Logic Element Definition

```
enum XSL_OP
{
    XSL_APPLY_IMPORTS, XSL_APPLY_TEMPLATES,
    XSL_ATTRIBUTE, XSL_ATTRIBUTE_SET,
    XSL_CALL_TEMPLATE, XSL_CHOOSE,
    XSL_COMMENT, XSL_COPY, XSL_COPY_OF,
    XSL_DECIMAL_FORMAT, XSL_ELEMENT,
    XSL_FALLBACK, XSL_FOR_EACH,
    XSL_IF, XSL_IMPORT,
    XSL_INCLUDE, XSL_KEY,
    XSL_MESSAGE, XSL_NAMESPACE_ALIAS,
    XSL_NUMBER, XSL_OTHERWISE,
    XSL_OUTPUT, XSL_PARAM,
    XSL_PRESERVE_SPACE, XSL_PROCESSING_INSTR,
    XSL_SORT, XSL_STRIP_SPACE,
    XSL_STYLESHEET, XSL_TEMPLATE,
    XSL_TEXT, XSL_TRANSFORM,
    XSL_VALUE_OF, XSL_VARIABLE,
    XSL_WHEN, XSL_WITH_PARAM,
    XSL_NONE
};
```

# XSLElement Class

- XSLElement::execute method handles the elements inside a switch case

```
class XSLElement : public Element
{
public:
    XSLElement(Tree& owner_, QName&, XSL_OP);
    virtual eFlag execute(Sit S, Context *c, Bool resolvingGlobals);
    virtual eFlag newChild(Sit S, Vertex*);
    eFlag checkToplevel(Sit S);
    virtual eFlag checkChildren(Sit S);
    void checkExtraChildren(int& k);
    Expression* getAttExpr(XSL_ATT);
    virtual int strip();
    XSL_OP op;
    eFlag checkAtts(Sit S);
private:
    eFlag makeSortDefs(Sit S, SortDefList &sortDefs, Context *c);
    eFlag make1SortDef(Sit S, SortDef *&def, Context *c);
};
```

# XPath Functions

- Implementation of these functions reside inside expr.cpp, inside the *Expression::callFunc* method.

```
case EXFF_LAST:
{
  checkArgsCount(0);
  retxpr.setAtom( Number(c -> getSize()) );
}; break;
case EXFF_POSITION:
{
  checkArgsCount(0);
  retxpr.setAtom( Number(c -> getPosition() + 1) );
}; break;
case EXFF_COUNT:
{
  checkArgsCount(1);
  checkIsNodeset(0);
  retxpr.setAtom(
    Number(atoms[0] -> tonodesetRef().getSize()) );
}; break;
```

# Reader's XSLT Engine

- In 2012, Adobe fixed two vulnerabilities
  - Discussed at-length on the conference circuit
- Community not paid much attention since those disclosures
- In 2015, Zero Day Initiative started receiving new vulnerability submissions in Reader's XSLT Engine
  - Discovered using a combination of fuzzing and auditing Sablotron source code

# Vulnerability Focus

| Name | Description | CVE Involvement |
|---|---|---|
| arena | Implementation of an Arena memory manager. | |
| base | Contains various macro definitions. | CVE-2012-1530 |
| context | Implementation of the expression evaluation context | |
| datastr | Various List implementations | CVE-2016-4198 |
| Error | Contains various macro definitions that deal with error handling. | CVE-2016-6963 |
| Expr | Includes the implementation of various XPath functions. | CVE-2016-6977 |
| Guard | Macro definitions of guarded pointers. | CVE-2016-6963 |
| Situa | Implementation of the situation object. It reflects the current situation, which is the vertex that is currently being parsed or processed. | |
| Tree | Contains XSL elements implementation. | |
| utf8 | Handles string conversions. | CVE-2012-1525 |
| verts | Implements various objects including the Attribute object. | CVE-2012-1530 |

# Code Matching

# Code Matching – Strings

```
struct FuncInfoItem
{
    const char *name;
    ExFunctor func;
    ExType type;
}
funcInfoTable[] =
{
    // XPath functions - NODESET category
    {"last",EXFF_LAST,EX_NUMBER},
    {"position",EXFF_POSITION,EX_NUMBER},
    {"count",EXFF_COUNT,EX_NUMBER},
    {"id",EXFF_ID,EX_NODESET},
    {"local-name",EXFF_LOCAL_NAME,EX_STRING},
    {"namespace-uri",EXFF_NAMESPACE_URI,EX_STRING},
    {"name",EXFF_NAME,EX_STRING},
/* ... */
    // XSLT core
    {"document",EXFF_DOCUMENT,EX_NODESET},
    {"key",EXFF_KEY,EX_NODESET},
    {"format-number",EXFF_FORMAT_NUMBER, EX_STRING},
    {"current",EXFF_CURRENT, EX_NODESET},
    {"unparsed-entity-uri",EXFF_UNPARSED_ENTITY_URI, EX_STRING},
    {"generate-id",EXFF_GENERATE_ID,EX_STRING},
    {"system-property",EXFF_SYSTEM_PROPERTY, EX_STRING},
```

# Code Matching – Strings

```
.data:1008B834                  align 8
.data:1008B838 off_1008B838     dd offset aLast         ; DATA XREF: sub_100252EB:loc_10027B4E↑r
.data:1008B838                                          ; sub_100252EB+286A↑o ...
.data:1008B838                                          ; "last"
.data:1008B83C dword_1008B83C   dd 1Ah                  ; DATA XREF: sub_10029F28+43↑r
.data:1008B840 dword_1008B840   dd 0                    ; DATA XREF: sub_10029F28+4E↑r
.data:1008B844                  dd offset aPosition     ; "position"
.data:1008B848                  db  1Bh
.data:1008B849                  db    0
.data:1008B84A                  db    0
.data:1008B84B                  db    0
.data:1008B84C                  db    0
.data:1008B84D                  db    0
.data:1008B84E                  db    0
.data:1008B84F                  db    0
.data:1008B850                  dd offset aCount        ; "count"
.data:1008B854                  db  1Ch
.data:1008B855                  db    0
.data:1008B856                  db    0
.data:1008B857                  db    0
.data:1008B858                  db    0
.data:1008B859                  db    0
.data:1008B85A                  db    0
.data:1008B85B                  db    0
.data:1008B85C                  dd offset aId_0         ; "id"
.data:1008B860                  db  1Dh
.data:1008B861                  db    0
.data:1008B862                  db    0
.data:1008B863                  db    0
.data:1008B864                  db    3
.data:1008B865                  db    0
.data:1008B866                  db    0
.data:1008B867                  db    0
.data:1008B868                  dd offset aLocalName    ; "local-name"
.data:1008B86C                  db  1Eh
```

# Code Matching – Strings

```
.data:1008B9B6                  db    0
.data:1008B9B7                  db    0
.data:1008B9B8                  dd offset aGenerateId    ; "generate-id"
.data:1008B9BC                  db  3Ah ; :
.data:1008B9BD                  db    0
.data:1008B9BE                  db    0
.data:1008B9BF                  db    0
.data:1008B9C0                  db    1
.data:1008B9C1                  db    0
.data:1008B9C2                  db    0
.data:1008B9C3                  db    0
.data:1008B9C4                  dd offset aSystemProperty ; "system-property"
.data:1008B9C8                  db  3Bh ; ;
.data:1008B9C9                  db    0
.data:1008B9CA                  db    0
.data:1008B9CB                  db    0
.data:1008B9CC                  db    1
.data:1008B9CD                  db    0
.data:1008B9CE                  db    0
.data:1008B9CF                  db    0
.data:1008B9D0                  dd offset aFunctionAvaila ; "function-available"
```

# Code Matching – Strings

```
eFlag Expression::callFunc(Sit S, Expression &retxpr, ExprList &atoms, Context *c)
    /* ... */
            case EXFF_SYSTEM_PROPERTY:
                {
                    checkArgsCount(1);
                    checkIsString(0);
                    QName q;
                    Str a0Str;  // CT.PERF.OPP
                    E( atoms[0] -> tostring(S, a0Str) );
#if !defined(CT_CHANGE)  // CT.PERF  CT.FIX
                    E( getOwnerElement().setLogical(S, q, a0Str, FALSE) );
                    if (q.getUri() == getOwnerTree().stdPhrase(PHRASE_XSL_NAMESPACE))
                    {
                        const Str& localStr = getOwnerTree().expand(q.getLocal());
                        if (localStr == (const char*) "version")
                            retxpr.setAtom(Number(1.0));
                        else if (localStr == (const char*) "vendor")
                            retxpr.setAtom(Str("Ginger Alliance"));
                        else if (localStr == (const char*) "vendor-url")
                            retxpr.setAtom(Str("www.gingerall.com"));
                        else
                            retxpr.setAtom(Str(""));
                    }
```

# Code Matching – Strings

```
.text:10026F4E ; ---------------------------------------------------------------------------
.text:10026F4E
.text:10026F4E loc_10026F4E:                           ; CODE XREF: sub_100252EB+1C4C↑j
.text:10026F4E                 mov     eax, [esi]
.text:10026F50                 mov     ecx, esi
.text:10026F52                 push    offset aVendor  ; "vendor"
.text:10026F57                 call    dword ptr [eax+4]
.text:10026F5A                 push    eax             ; char *
.text:10026F5B                 call    _strcmp
.text:10026F60                 pop     ecx
.text:10026F61                 pop     ecx
.text:10026F62                 test    eax, eax
.text:10026F64                 jnz     short loc_10026F70
.text:10026F66                 push    offset aGingerAlliance ; "Ginger Alliance"
.text:10026F6B                 jmp     loc_10027068
.text:10026F70 ; ---------------------------------------------------------------------------
.text:10026F70
.text:10026F70 loc_10026F70:                           ; CODE XREF: sub_100252EB+1C79↑j
.text:10026F70                 mov     eax, [esi]
.text:10026F72                 mov     ecx, esi
.text:10026F74                 push    offset aVendorUrl ; "vendor-url"
.text:10026F79                 call    dword ptr [eax+4]
.text:10026F7C                 push    eax             ; char *
.text:10026F7D                 call    _strcmp
.text:10026F82                 pop     ecx
.text:10026F83                 pop     ecx
.text:10026F84                 mov     ecx, ebx
.text:10026F86                 test    eax, eax
.text:10026F88                 jnz     short loc_10026F94
.text:10026F8A                 push    offset aWww_gingerall_ ; "www.gingerall.com"
.text:10026F8F                 jmp     loc_1002706A
.text:10026F94 ; ---------------------------------------------------------------------------
.text:10026F94
.text:10026F94 loc_10026F94:                           ; CODE XREF: sub_100252EB+1C9D↑j
.text:10026F94                                         ; sub_100252EB+1CE9↓j ...
.text:10026F94                 push    offset byte_10069332
.text:10026F99                 jmp     loc_1002706A
.text:10026F9E ; ---------------------------------------------------------------------------
.text:10026F9E
.text:10026F9E loc_10026F9E:                           ; CODE XREF: sub_100252EB+1C1F↑j
.text:10026F9E                 call    sub_1002A1F6
.text:10026FA3                 mov     ecx, esi
.text:10026FA5                 cmp     edi, [eax+1BCh]
.text:10026FAB                 jnz     short loc_10026FE0
.text:10026FAD                 push    dword ptr [ebp+var_48+4]
.text:10026FB0                 call    sub_1002A1F6
.text:10026FB5                 mov     ecx, eax
.text:10026FB7                 call    sub_1003399E
.text:10026FBC                 push    offset aVersion_0 ; "version"
.text:10026FC1                 mov     ecx, eax
```

# Code Matching – Data structures

```
static ExToken namerTable[] = {
    TOK_ATSIGN, TOK_DCOLON, TOK_LPAREN, TOK_LBRACKET,
        // operators:
    TOK_OR, TOK_AND, TOK_EQ, TOK_NEQ, TOK_LT, TOK_GT, TOK_LE, TOK_GE,
    TOK_PLUS, TOK_MINUS, TOK_MINUS1, TOK_MULT, TOK_DIV, TOK_MOD, TOK_VERT,
        // slashes are operators too but not for us
    TOK_SLASH, TOK_DSLASH, TOK_COMMA,
        // TOK_NONE (terminator)
    TOK_NONE};
```

# Code Matching – Data structures

```
dword_1008C9F8  dd 0Bh

dword_1008C9FC  dd 0Dh
                dd 5
                dd 7
                db  13h
                db    0
                db    0
                db    0
                db  14h
                db    0
                db    0
                db    0
```

```
; DATA XREF: sub_1002A5F8:loc_1002A608↑r
; sub_1002A5F8+2C↑r
; DATA XREF: sub_1002A5F8+1E↑r
```

# Code Matching – Preprocessor defines

```
1150 #if !defined(CT_CHANGE) /* original version */
1151 eFlag LocStep::shift(Sit S, NodeHandle &v, NodeHandle baseV)
1152 +--199 lines: {-----------------------------------------
1351 #else /* CT_CHANGE */
1352 eFlag LocStep::shift(Sit S, NodeHandle &v, NodeHandle baseV)
1353 {
```

# Code Matching – Preprocessor defines

```
const char* xslOpNames[]=
{
    "apply-imports","apply-templates",
        "attribute","attribute-set",
        "call-template","choose",
        "comment", "copy","copy-of",
        "decimal-format","element",
        "fallback","for-each",
        "if","import",
        "include","key",
        "message","namespace-alias",
        "number","otherwise",
        "output","param",
        "preserve-space","processing-instruction",
        "sort","strip-space",
        "stylesheet","template",
        "text","transform",
        "value-of","variable",
        "when","with-param",
#if defined(CT_CHANGE)
        "&forward-compatible",  // leave this unmatchable string
#endif
        NULL
};
```

# Code Matching – Preprocessor defines

```
.data:1008B300                 dd offset aOtherwise    ; "otherwise"
.data:1008B304                 dd offset aOutput_0     ; "output"
.data:1008B308                 dd offset aParam        ; "param"
.data:1008B30C                 dd offset aPreserveSpace ; "preserve-space"
.data:1008B310                 dd offset aProcessingInst ; "processing-instruction"
.data:1008B314                 dd offset aSort         ; "sort"
.data:1008B318                 dd offset aStripSpace   ; "strip-space"
.data:1008B31C ; char *off_1008B31C
.data:1008B31C off_1008B31C    dd offset aStylesheet   ; DATA XREF: sub_10034185+54↑r
.data:1008B31C                                         ; sub_10034E5A+6C3↑r
.data:1008B31C                                         ; "stylesheet"
.data:1008B320                 dd offset aTemplate     ; "template"
.data:1008B324                 dd offset aText         ; "text"
.data:1008B328 ; char *off_1008B328
.data:1008B328 off_1008B328    dd offset aTransform    ; DATA XREF: sub_10034185+83↑r
.data:1008B328                                         ; "transform"
.data:1008B32C                 dd offset aValueOf      ; "value-of"
.data:1008B330                 dd offset aVariable     ; "variable"
.data:1008B334                 dd offset aWhen         ; "when"
.data:1008B338                 dd offset aWithParam    ; "with-param"
.data:1008B33C                 dd offset aForwardCompati ; "&forward-compatible"
```

# Vulnerability Analysis

CASE STUDIES

# CVE-2012-1525

## HEAP-BASED BUFFER OVERFLOW

# CVE-2012-1525 – Root Cause

- Sablotron parses XSL elements in *XSLElement::execute*
  - Function contains a switch case that determines the type of the XSL element
- XSL element specifically checks to see if there's a name attribute, if it exists:
  - Length of the name is calculated using utf8StrLength
  - Allocates spaces based on the length
  - Attribute value is converted to a UTF16 string using utf8ToUtf16

```
Bool isValidNCName(const char* name)
{
  int len = utf8StrLength(name);
  if (len == 0) return FALSE;

  wchar_t *buff = new wchar_t[len + 1];

  utf8ToUtf16(buff, name);
```

# CVE-2012-1525 – Root Cause

- utf8StrLength has broken logic:
  - Checks for ascii characters
  - If there's a non-ascii character it calculates the length of the character
  - Keeps incrementing by 1
- Things can go wrong with special Unicode characters like &#xE004D

```
int utf8StrLength (const char* text)
{
    int len;
    for (len = 0; *text; len++)
    {
        if (!(*text & 0x80))
            text++;
        else text += utf8SingleCharLength(text);
    }
    return len;
}
```

# CVE-2012-1525 – Root Cause

- Miscalculated size is later used in an allocation of a buffer
- utf8ToUtf16 is responsible for the string conversion
- Ends up overflowing the allocated buffer

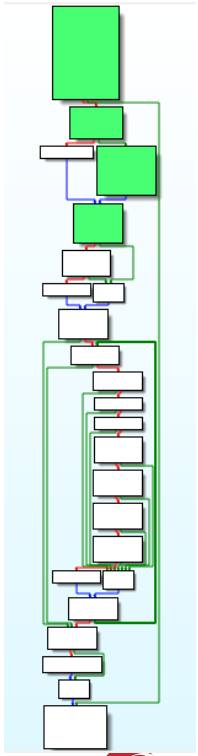
```
int utf8ToUtf16(wchar_t *dest, const char *src)
{
    unsigned long code;
    int len = 0,
        thislen;
    for (const char *p = src; *p; p += utf8SingleCharLength(p))
    {
        code = utf8CharCode(p);
        if (code < 0x10000UL)
        {
            *dest = (wchar_t)(code);
            thislen = 1;
        }
        else
        {
            dest[0] = 0xd7c0U + (code >> 10);
            dest[1] = 0xdc00U | code & 0x3ff;
            thislen = 2;
        };
        dest += thislen;
        len += thislen;
    }
```

# CVE-2012-1525 – Pinpointing in Adobe

- One of the first bugs patched in AXSLE.dll
- utf8ToUtf16 can be found easily by text searching for DC00h

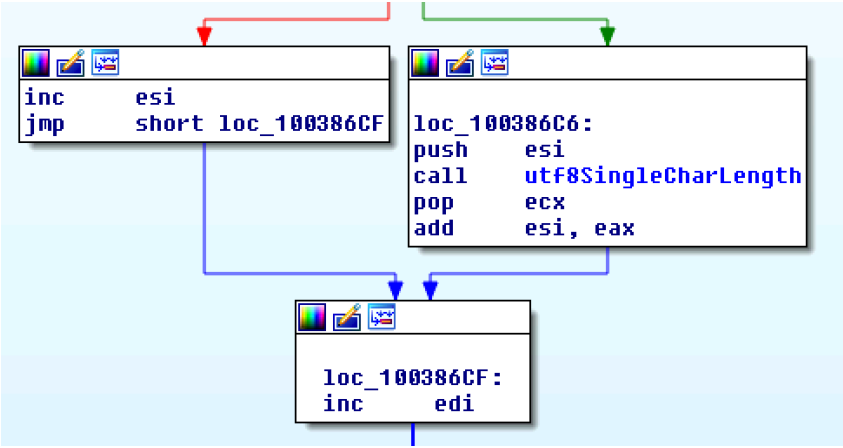
```
len = utf8StrLength((_BYTE *)a1);
len_ = len;
if ( len )
{
  v2 = len + 1;
  v9 = v2 <= 64;
  if ( v2 > 64 )
    v3 = (char *)operator new(2 * v2);
  else
    v3 = &v10;
  utf8ToUtf16(v3, (_BYTE *)a1);
```

```
dest_ = dest;
v3 = src;
for ( i = 0; *v3; v3 += utf8SingleCharLength(v3) )
{
  code = utf8CharCode(v3);
  if ( code >= 0x10000 )
  {
    dest_[1] = code & 0x3FF | 0xDC00;
    *dest_ = (code >> 10) - 0x2840;
    thislen = 2;
  }
  else
  {
    *dest_ = code;
    thislen = 1;
  }
  dest_ += thislen;
  i += thislen;
}
*dest_ = 0;
return i;
```



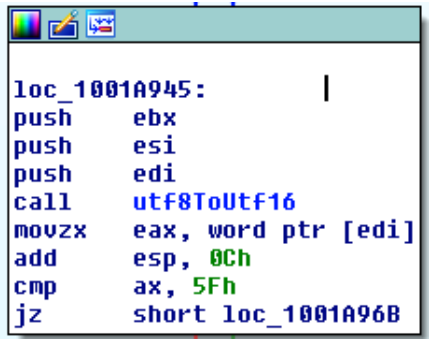
# CVE-2012-1525 – Pinpointing in Adobe

- There's only one call to utf8ToUtf16 which is from isValidNCName
- Trivial to find utf8StrLength from isValidNCName



# CVE-2012-1525 – Patch Analysis

- Modified the utf8ToUtf16 prototype and added an extra argument

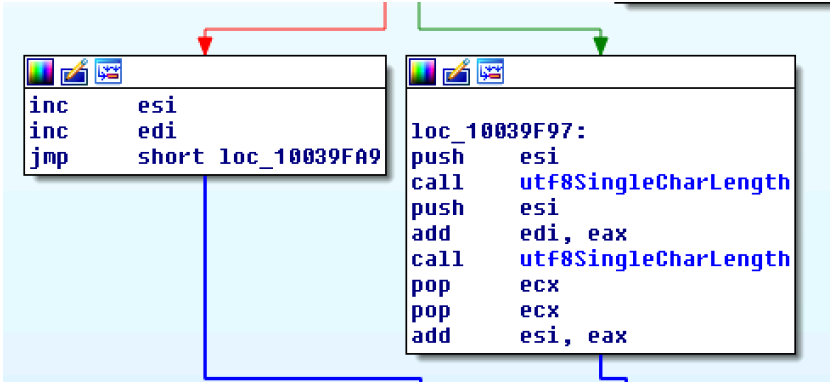


```
loc_1001A945:
push    ebx
push    esi
push    edi
call    utf8ToUtf16
movzx   eax, word ptr [edi]
add     esp, 0Ch
cmp     ax, 5Fh
jz      short loc_1001A96B
```

```
v3 = src_len;
l = 0;
v4 = dest;
for ( i = src; *i; i += sub_10021469(i) )
{
  v6 = utf8CharCode(i);
  if ( v6 >= 0x10000 )
  {
    if ( v3 >= 2 )
    {
      *v4 = (v6 >> 10) - 10304;
      v4[1] = v6 & 0x3FF | 0xDC00;
    }
    v7 = 2;
  }
  else
  {
    if ( v3 >= 1 )
      *v4 = v6;
    v7 = 1;
  }
  l += v7;
  v4 += v7;
  v3 -= v7;
}
*v4 = 0;
return l;
```

# Finally Addressed in 2017

- Really fixed in the January bulletin release
  - Assigned CVE-2017-2948
- Actual correct fix is incrementing the counter by the size of the character



# CVE-2012-1530

TYPE-CONFUSION

# CVE-2012-1530 – Root Cause

- XPath lang function is handled inside the EXFF_LANG switch case inside *Expression::callFunc*
- Checks if the argument count is exactly 1
- Checks if the first argument is a string

```
case EXFF_LANG:
{
checkArgsCount(1);
checkIsString(0);
// get the argument
Str langQuery;
E( atoms[0] -> tostring(S, langQuery) );
NodeHandle w, att = NULL;
int attCount, i;
const char* langValue = NULL;
for (w = c -> current(); w && !langValue; w = S.dom().getParent(w))
{
    // find whether w has an xml:lang attribute
  if (!S.domExternal(w)) {
    QName searchName;
    searchName.setUri(getOwnerTree().unexpand(theXMLNamespace));
    searchName.setLocal(getOwnerTree().unexpand("lang"));
    int idx = toE(w) -> atts.findNdx(searchName);
```

# CVE-2012-1530 – Root Cause

- *NodeHandle* element is casted to an *Element* object using the *toE* macro is defined as follows:

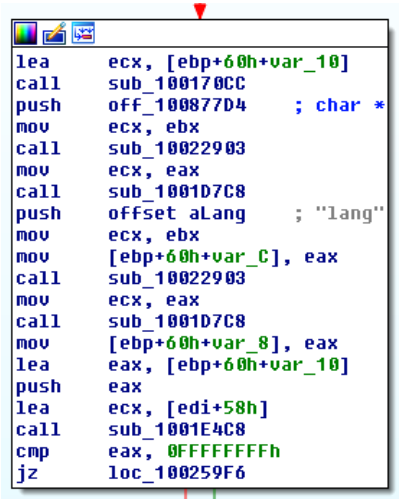
```
#define toV(v) ((Vertex*)v)
#define toNS(v) (cast(NmSpace*,toV(v)))
#define toE(v) (cast(Element*,toV(v)))
#define toX(v) (cast(XSLElement*,toV(v)))
#define toExtension(v) (cast(ExtensionElement*,toV(v)))
#define toA(v) (cast(Attribute*,toV(v)))
```

- cast() is an interesting macro

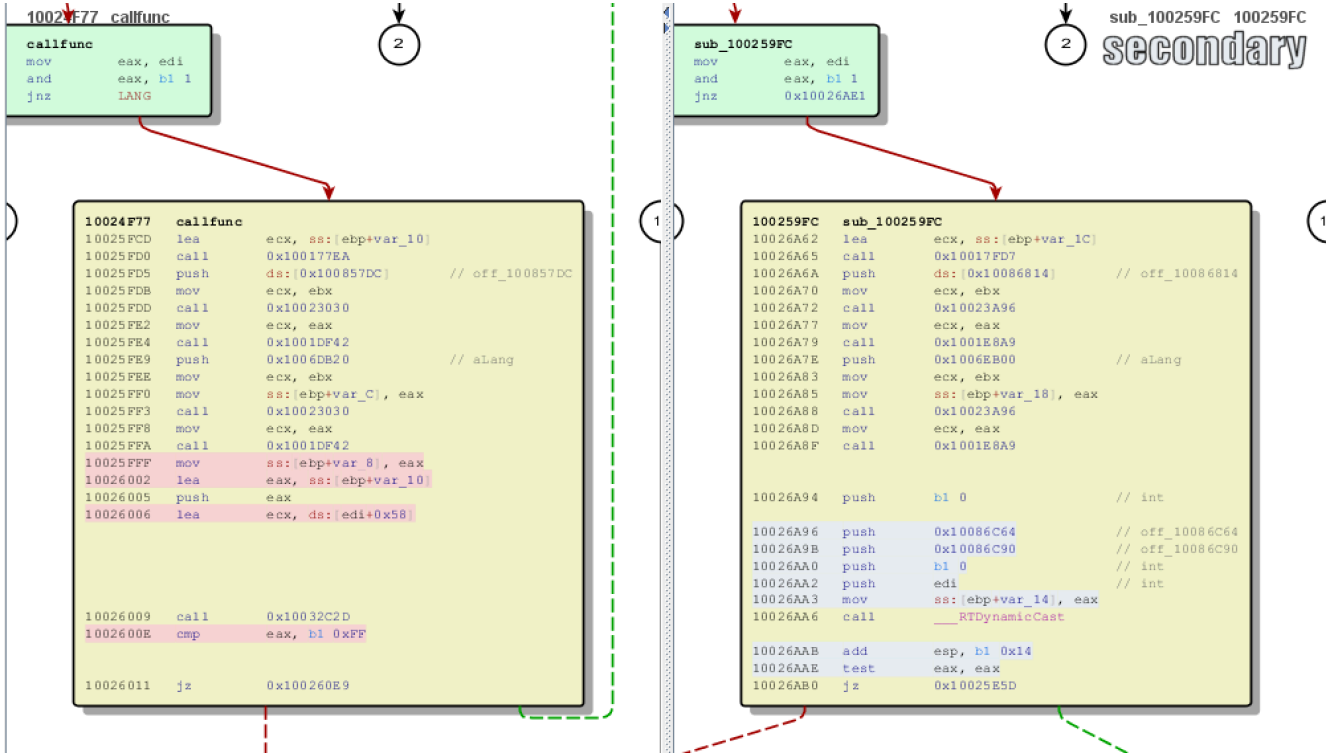
```
#ifdef _DEBUG
#define cast(TYPE,PTR) (NZ(dynamic_cast<TYPE>(PTR)))
#else
#define cast(TYPE,PTR) ((TYPE) PTR)
#endif
```

# CVE-2012-1530 – Pinpoint in Adobe

- String matching technique works well
- Looking for references to the *lang* string will yield three results
- Trivial to find the right one

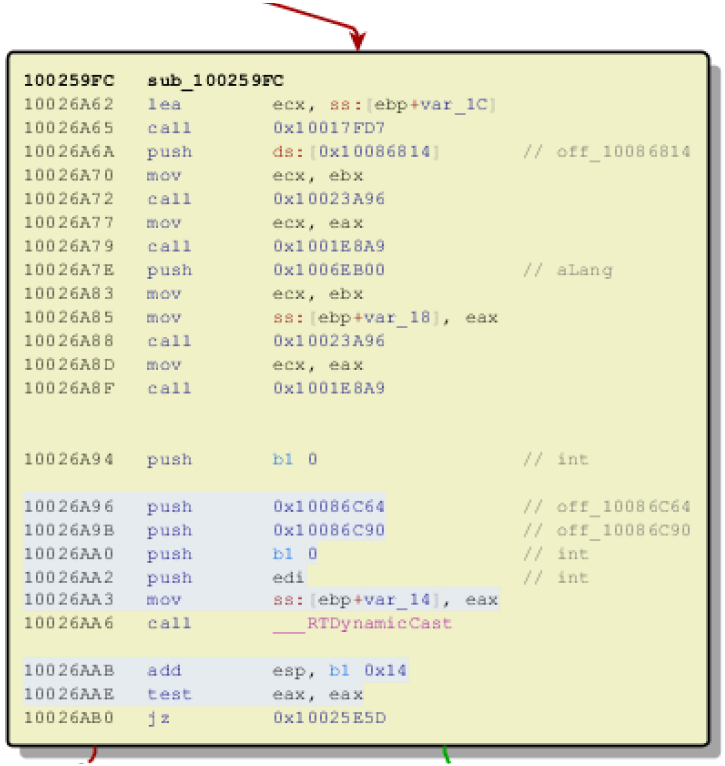


# CVE-2012-1530 – Patch Analysis



# Finally Addressed in 2017

- Patched in the January 2017 bulletin release
  - Assigned CVE-2017-2962
  - Was correctly patched by calling __RTDynamicCast



# CVE-2016-6977

OUT-OF-BOUNDS READ

# CVE-2016-6977 – OOBR Root Cause

- In Sablotron, arguments passed to XPath functions are kept inside an *ExprList*
  - *ExprList* is simply a *PList* of *Expressions*
  - *PList* is an object that inherits from the *List* object
- Sablotron uses macros to check if the number of arguments/types passed to XPath functions is valid:

```
#define checkArgsCount(x) if (atomsNumber != x)\
    Err(S, ET_BAD_ARGS_N);
#define checkArgsCountMax(x) if (atomsNumber > x)\
    Err(S, ET_BAD_ARGS_N);
#define checkArgsCountMin(x) if (atomsNumber < x)\
    Err(S, ET_BAD_ARGS_N);
#define checkArgsCountBetween(x,y) if ((atomsNumber < x) || \
    (atomsNumber > y)) Err(S, ET_BAD_ARGS_N);
```

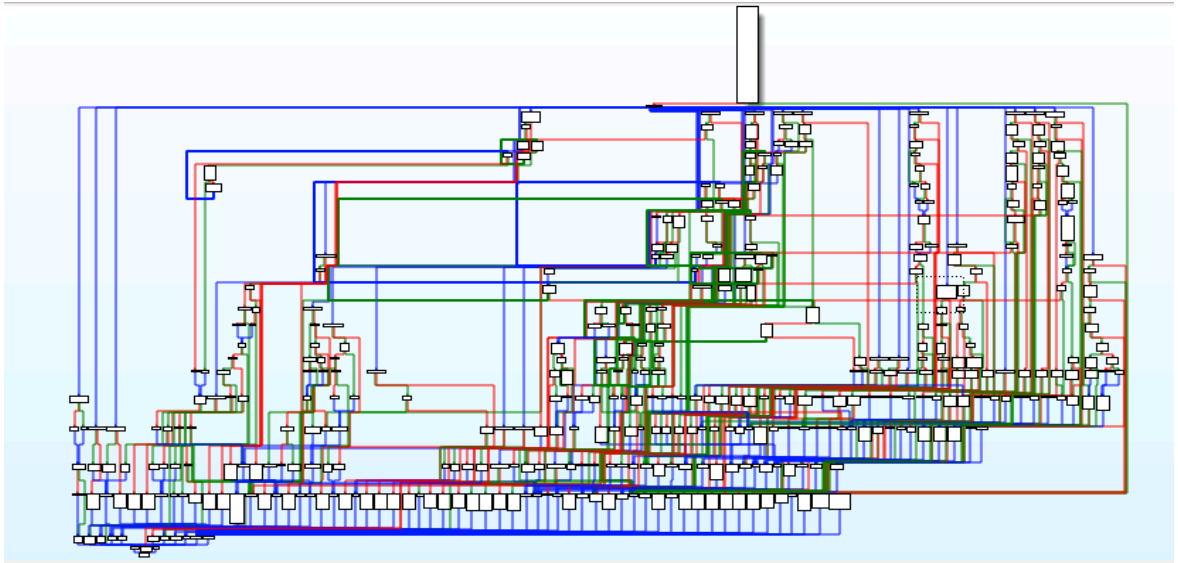
# CVE-2016-6977 – OOBR Root Cause

- If no arguments are passed to an XPath function and elements from atoms were referenced without checking for the arguments count, then an Out-Of-Bounds Read might occur
- That was exactly the case with the XPath functions *EXFF_SUBSTRING_BEFORE* and *EXFF_SUBSTRING_AFTER*:

```
case EXFF_SUBSTRING_BEFORE:
case EXFF_SUBSTRING_AFTER:
{
Str strg;
Str theBigger, theSmaller;
E( atoms[0] -> tostring(S, theBigger) );
E( atoms[1] -> tostring(S, theSmaller) );
checkArgsCount(2);
checkIsString2(0,1);
int where = firstOccurence(theBigger,theSmaller);
```

# CVE-2016-6977 – Pinpoint in Adobe

- Bug is rather easy to spot with proper matching and function renaming
- *Expression::callFunc* is a function that is composed of a big switch case



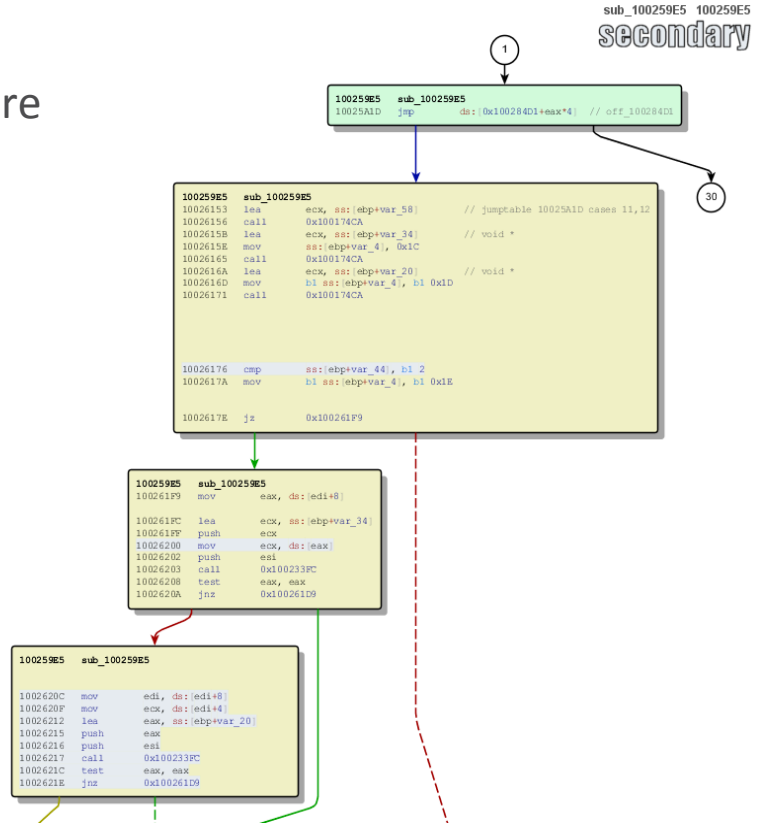
# CVE-2016-6977 – Pinpoint in Adobe

- This can be found in multiple ways.
- Searching for the “lang” string is one option.

```
.text:1002613A substring_before_after:                ; CODE XREF: Expression__callFunc_0+38↑j
.text:1002613A                                        ; DATA XREF: .text:off_100284BC↓o
.text:1002613A                lea     ecx, [ebp+var_58] ; jumptable 10025A04 cases 11,12
.text:1002613D                call    sub_100174E5
.text:10026142                lea     ecx, [ebp+var_34] ; void *
.text:10026145                mov     [ebp+var_4], 1Ch
.text:1002614C                call    sub_100174E5
.text:10026151                lea     ecx, [ebp+var_20] ; void *
.text:10026154                mov     byte ptr [ebp+var_4], 1Dh
.text:10026158                call    sub_100174E5
.text:1002615D                mov     eax, [edi+8]
.text:10026160                lea     ecx, [ebp+var_34]
.text:10026163                push    ecx
.text:10026164                mov     ecx, [eax]
.text:10026166                push    esi
.text:10026167                mov     byte ptr [ebp+var_4], 1Eh
.text:1002616B                call    Expression__tostring
.text:10026170                test    eax, eax
.text:10026172                jz      short loc_10026194
```

# CVE-2016-6977 – Patch Analysis

- As expected, Adobe made sure to check for the arguments count before referencing.



# CVE-2016-4198

## OUT-OF-BOUNDS WRITE

# CVE-2016-4198 – OOBW Root Cause

- List object is constructed by setting origBlockSize to pow(2,logBlocksize_)
- Controlling logBlocksize_ will lead to an integer wrap when calling append

```
template <class T>
List<T>::List(int logBlocksize_)
:
origBlocksize(TWO_TO(logBlocksize_))
{
    nItems = 0;
    blocksize = 0;
    block = NULL;
};
```

```
template <class T>
void List<T>::append(T what)
{
    if (nItems >= blocksize)
    {
        if (block)
            grow();
        else
        {
            blocksize = origBlocksize;
            block = (T*) claimMemory(blocksize * sizeof(T));
            // FIXME: asserting memory request ok
            sabassert(block);
        }
    }
    block[nItems++] = what;
};
```

# CVE-2016-4198 – OOBW Root Cause

- createContext defines the variable *reached* that is based on the number of predicates
- Number of predicates is user-controlled
- If dryRun is set to false the second loop calls reached.append(0)

```
// keep a stack of positions, one for each predicate IN THIS STEP
List<int> reached(predsCount),  // serves as position for next pred
    totalReached(predsCount);   // serves as size for next (bad) pred
```

```
// i ranges over predicates. Value i==predsCount is the special last run
for (i = 0; i <= predsCount; i++)
{
    if (i == predsCount)
        // the last run, not a dry-run
        dryRun = FALSE;
    // if this is the last run, or if the current pred uses last(), compute
    // the context size
    if (!dryRun || args[stepLevel] -> step -> preds[i] -> usesLast)
    {
        // initialize the size arrays:
        // append base values for preds past the last bad one,
        // up to this bad one (incl.)
        for (init = 0; init <= lastBad; init++)
            reached[init] = 0;
        for (init = lastBad + 1; init <= i; init++)
        {
            reached.append(0);
            totalReached.append(-1);     // -1 just for safety
        };
```

# CVE-2016-4198 – Pinpoint in Adobe

- Find Expression::createContext
  - Trivial to find the EXF_LOCPATH switch case
- Find createLPContext that calls createLPContextLevel
- Two calls to the Constructor list can be found

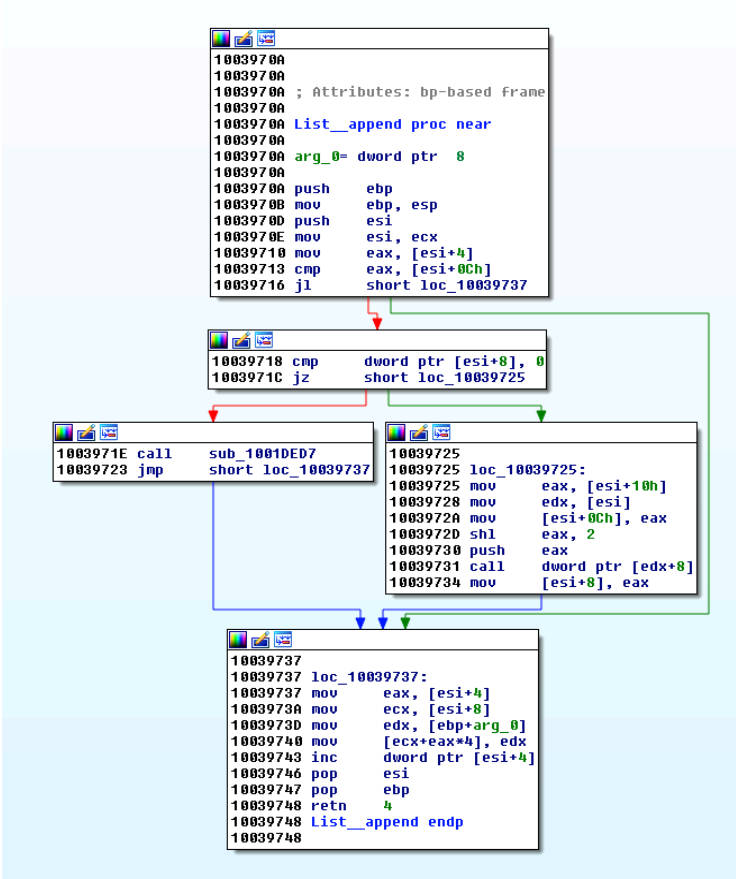
```
1001613A
1001613A
1001613A ; Attributes: bp-based frame
1001613A
1001613A List__Constructor proc near
1001613A
1001613A arg_0= dword ptr  8
1001613A
1001613A push    ebp
1001613B mov     ebp, esp
1001613D mov     eax, ecx
1001613F mov     ecx, [ebp+arg_0]
10016142 xor     edx, edx
10016144 inc     edx
10016145 shl     edx, cl
10016147 xor     ecx, ecx
10016149 mov     dword ptr [eax], offset off_1006D3BC
1001614F mov     [eax+4], ecx
10016152 mov     [eax+10h], edx
10016155 mov     [eax+0Ch], ecx
10016158 mov     [eax+8], ecx
1001615B pop     ebp
1001615C retn    4
1001615C List__Constructor endp
1001615C
```

```
result = this;
*this = &off_1006D3BC;
this[1] = 0;
this[4] = 1 << a2;
this[3] = 0;
this[2] = 0;
return result;
```

```
10024F1E lea     ecx, [ebp+var_44]
10024F21 mov     [ebp+predsCount], edi
10024F24 call    List__Constructor
10024F29 xor     esi, esi
10024F2B push    edi
10024F2C lea     ecx, [ebp+var_58]
10024F2F mov     [ebp+var_4], esi
10024F32 call    List__Constructor
10024F37 mov     byte ptr [ebp+var_4], 1
10024F3B mov     [ebp+dryRun], 1
10024F42 mov     [ebp+var_14], esi
10024F45 test    edi, edi
10024F47 js      loc_1002513F
```

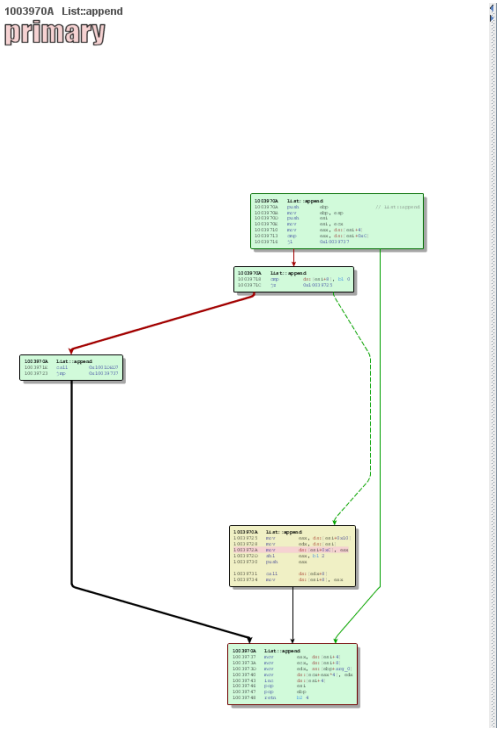
# CVE-2016-4198 – Pinpoint in Adobe

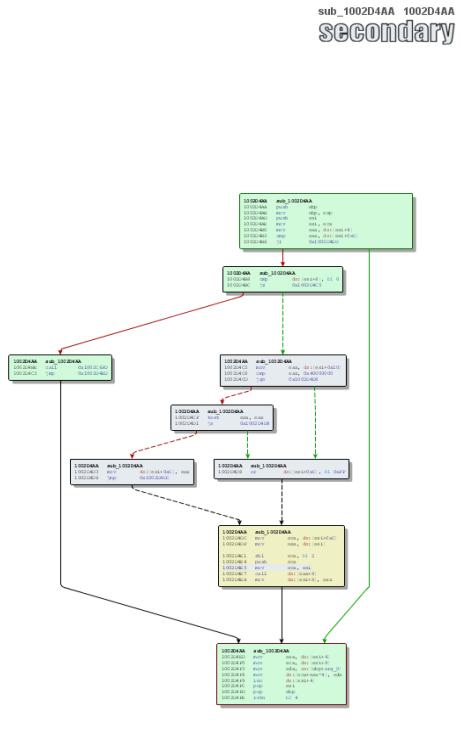
- List::append looks exactly the same and haven't been modified by Adobe
- Allocates block based on origBlockSize



# CVE-2016-4198 – Patch Analysis

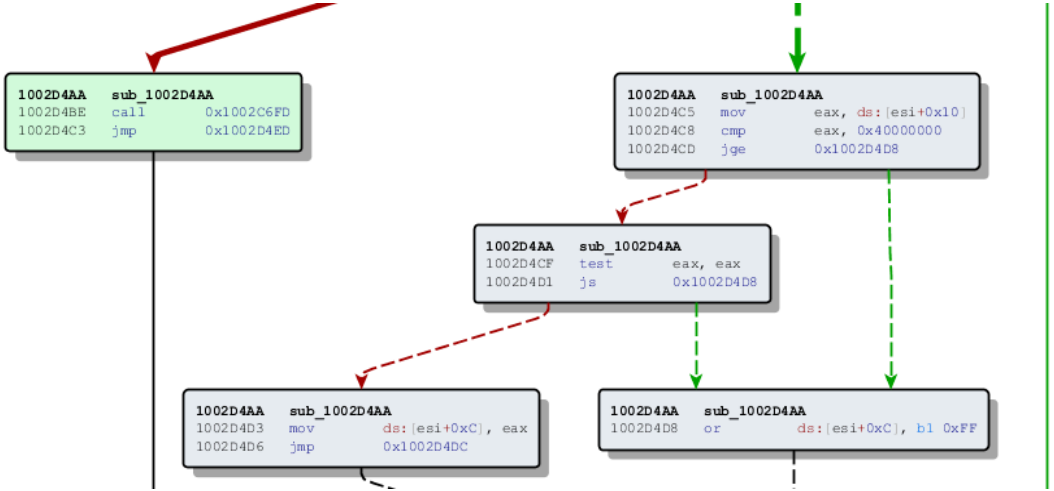
- Adobe patched this bug in an interesting way
- Patched the append function rather than anywhere else (specifically the way origBlockSize is being initialized)



# CVE-2016-4198 – Patch Analysis

- A size check was introduced inside the append function
- If the length >= 0x40000000 or < 0, length is set to -1 and allocation will fail



# CVE-2016-6963

DOUBLE-FREE

# CVE-2016-6963 – Root Cause

- Sablotron has the ability to declare “Guarded Pointers”
  - Defined in *guard.h*
- operator= is interesting, it returns a raw pointer
- Problem arises when the function defining these GP cannot guarantee the lifetime of these pointers.
  - Especially when having multiple pointers pointing to the same location

```
#define DeclGuard( TYPE ) \
class TYPE##_G \
{ \
public: \
    TYPE##_G(TYPE *ptr_ = NULL): ptr(ptr_), kill(TRUE) {} \
    TYPE* keep() {kill = FALSE; return ptr;} \
    TYPE* unkeep() {kill = TRUE; return ptr;} \
    void assign(TYPE* ptr_) {kill = FALSE; ptr = ptr_;} \
    TYPE* operator= (TYPE *ptr_) {kill = ptr_? TRUE : FALSE; return ptr = ptr_;} \
    operator TYPE*&() {return ptr;} \
    TYPE& operator*(void) {sabassert(ptr); return *ptr;} \
    TYPE* operator()(void) {return ptr;} \
    void del(void) {if (ptr) delete ptr; ptr = NULL; kill = FALSE;} \
    void delArray(void) {if (ptr) delete[] ptr; ptr = NULL; kill = FALSE;} \
    ~TYPE##_G() {if (kill) del();} \
protected: \
    TYPE *ptr; \
    Bool kill; \
};
```

# CVE-2016-6963 – Root Cause

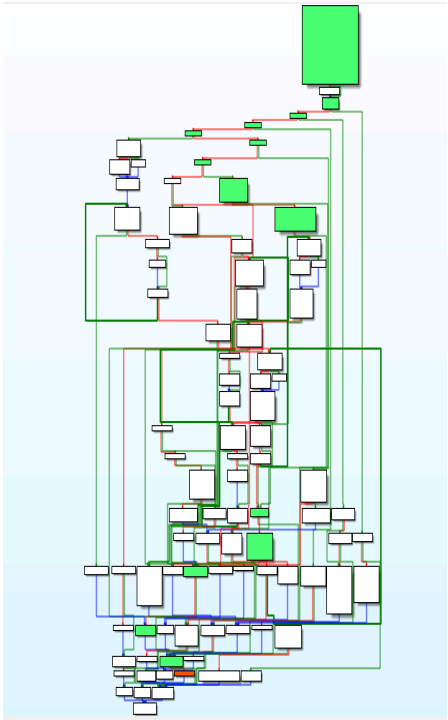
```
case EXF_FILTER:
    {
        sabassert(baseNdx != -1);  // meaningful only for a locpath
        NodeHandle wasCurrent = (*newc).getCurrentNode();
        E( args[0] -> createContext(S, newc, baseNdx) );
        newc.unkeep();
        (*newc).setCurrentNode(wasCurrent);

        GP( Context ) filteredc;
        for (i = 1; i < argsNumber - (int) hasPath; i++)
        {
            filteredc = new Context(c_orig -> getCurrentNode());
            (*newc).reset();
            Bool istrue;
            int newcNumber = (*newc).getSize();
            for (j = 0; j < newcNumber; j++)
            {
                E(args[i] -> trueFor(S, newc, istrue));
                if (istrue)
                    (*filteredc).append((*newc)[j]);
                (*newc).shift();
            };
            newc.del();
            newc = filteredc.keep();
            if (!(*newc).getSize()) break;
        };
        if (hasPath)
        {
            filteredc.assign(newc);
            filteredc = newc;  // a patch due to SGI MIPSpro compiler
            E( args[argsNumber-1] -> createLPContextSum(S, filteredc, (*newc).getCurrentNode()) );
            newc.del();
            newc = filteredc.keep();
        }
    }
    break;
```

# CVE-2016-6963 – Pinpoint in Adobe

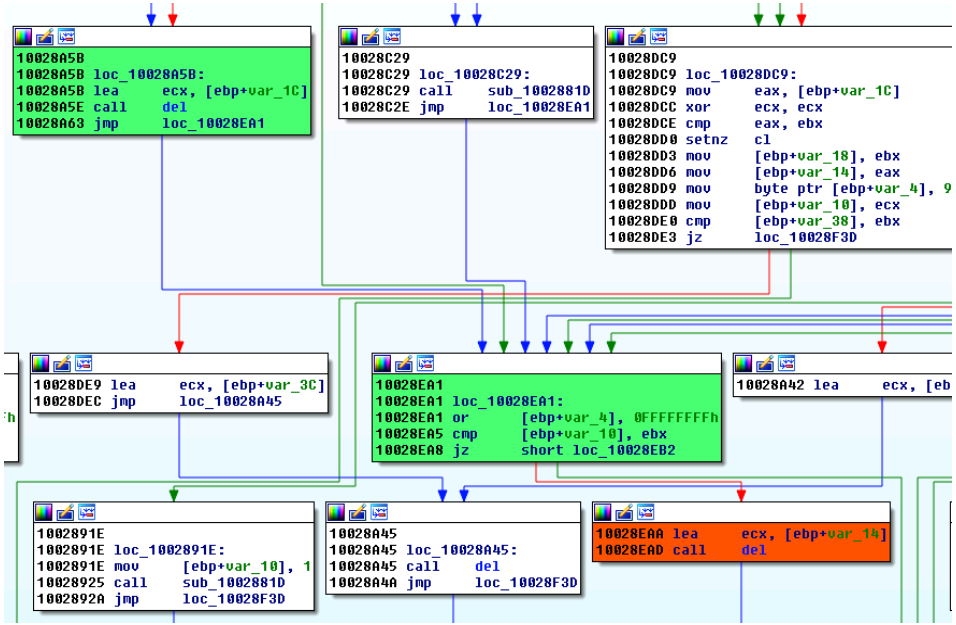
- In binary, a reference is retrieved and stored in a local variable

```
10028B9C mov     eax, [ebp+var_10]
10028B9F mov     ecx, [ebp+var_14] ; retrieve
10028BA2 mov     edx, [ebp+var_24]
10028BA5 mov     [ebp+var_18], eax
10028BA8 mov     eax, [esi+10h]
10028BAB mov     [ebp+var_1C], ecx ; store
10028BAE lea     esi, [eax+edx*4-4]
10028BB2 call    sub_10014580
10028BB7 mov     ecx, [esi]
10028BB9 push    eax             ; int
10028BBA lea     eax, [ebp+var_1C]
10028BBD push    eax             ; int
10028BBE push    [ebp+Situation_S] ; char *
10028BC1 call    sub_10028623
10028BC6 test    eax, eax
10028BC8 jnz     loc_10028A4F
```



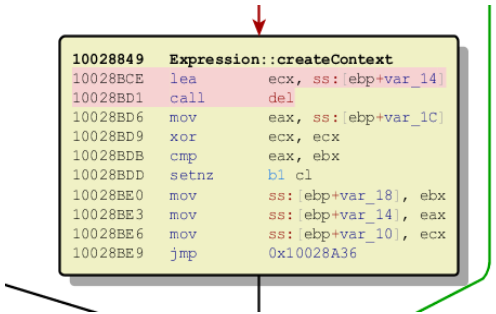
# CVE-2016-6963 – Pinpoint in Adobe

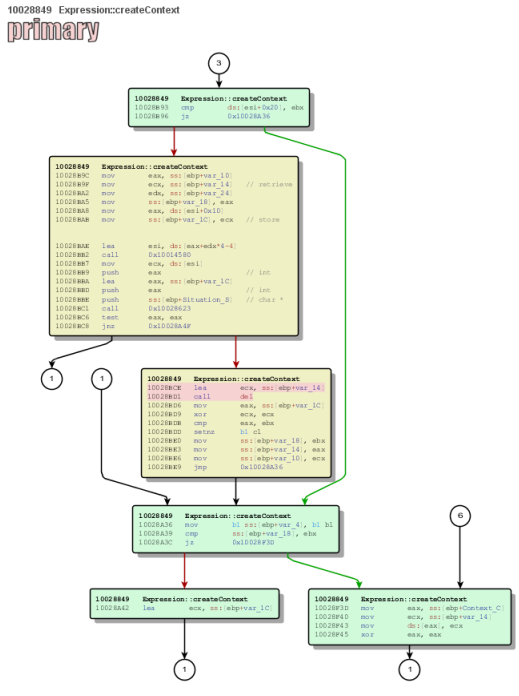
- First reference stored in var_1c is deleted
- Second reference stored in var_14 is also deleted triggering the double free

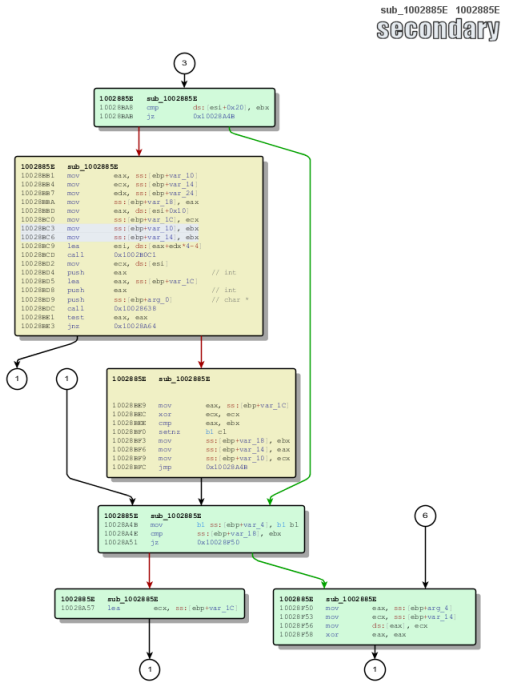


# CVE-2016-6963 – Patch Analysis

- Adobe removed the code responsible of deleting the second reference



# Conclusion

# Conclusion

- Use of open-source components is becoming popular
- Poorly implemented open-source allows attackers access to closed source systems
- New source-to-binary matching techniques can help you locate these weaknesses

# ZERO DAY INITIATIVE