文章编号:1001 - 9081(2008)09 - 2252 - 03

基于 Fuzzing 的 ActiveX 控件漏洞发掘技术

吴毓书,周安民,吴少华,何永强,徐 威
(四川大学 信息安全研究所, 成都 610064)
(cuit_xiao@163.com)

摘 要:Fuzzing 是一种有效的自动化的漏洞发掘技术,基于 Fuzzing 漏洞发掘思想,结合对 ActiveX 控件的研究,设计并实现了一个 Windows 系统下的 ActiveX 控件漏洞发掘平台,并改进了 Fuzzing 数据产生方案。通过对某些第三方软件安装的控件进行测试,发现了两个已知和一个未知的漏洞,提高了漏洞发掘效率。

关键词:ActicvX 控件;漏洞;漏洞挖掘;Fuzzing 技术
中图分类号: TP393.08 **文献标志码**:A

ActiveX vulnerability exploiting technique based on Fuzzing

WU Yu-shu, ZHOU An-min, WU Shao-hua, HE Yong-qiang, XU Wei
(*Institute of Information Security, Sichuan University, Chengdu Sichuan* 610064, *China*)

Abstract: Fuzzing is an automated vulnerability exploiting technique. A vulnerability exploiting approach based on Fuzzing and the technical details of ActiveX was proposed. A fuzzer was designed, and effective implementation of data generation was advanced. By testing some third-part software's ActiveX controls, one unreleased and two known vulnerabilities were discovered and the efficiency of the ActiveX fuzz was improved.

Key words: ActiveX controls; vulnerability; vulnerability exploiting; Fuzzing technique

0 引言

漏洞是在硬件、软件、协议的具体实现或系统安全策略上存在的缺陷,从而可以使攻击者能够在未授权的情况下访问或破坏系统。系统漏洞不仅是黑客攻防的焦点,也是病毒、木马传播的重要途径,这给用户带来巨大的危害。所以及时发现并修补系统漏洞对维护系统安全具有重要的意义。

漏洞挖掘是一个多种漏洞挖掘技术相互结合、共同使用、优势互补的过程。目前漏洞挖掘有效的方法有静态分析技术、补丁比较技术、动态调试技术和 Fuzzing 技术等[1]。Fuzzing 是一项自动化的漏洞发掘技术,它可以自动完成生成测试数据、构造样本、执行样本、捕获程序异常等一系列工作。高度自动化的 Fuzzer 工具可以使黑客轻松地找到危害严重的漏洞,例如 Microsoft Office 等办公软件的漏洞。然而 Fuzzing 技术也存在着局限和不足,传统的 Fuzzing 方法仅依靠生成随机数据来发现软件的缺陷,测试效率很低。

本文将着重研究如何利用 Fuzzing 技术挖掘基于 Windows 平台下各种第三方软件的 ActiveX 控件漏洞,并改进了 Fuzzing 数据生成的方案。

1 ActiveX 控件安全研究现状

一个系统的安全强度等于它最薄弱环节的安全强度。目前,ActiveX 控件在 Windows 操作平台上广泛使用,其安全性直接影响整个操作系统的安全。因此,任何 ActiveX 控件安全问题都将导致整个系统安全体系的崩溃。

然而,ActiveX 控件漏洞数量在最近两年里激增,根据 Symantec 公司在 2007 年一份关于 ActiveX 控件安全漏洞的报告,在 2001 年只有一份关于 ActiveX 控件漏洞的报告,而仅在 2006 年一年当中,就有 50 份关于 ActiveX 控件的安全报告,漏洞报告数量是前五年发现漏洞数量的总和。在 2007 年里,Microsoft Agent 、Yahoo! Messenger、MySpace 等广泛使用的软件中也频频曝出控件漏洞。

目前,ActiveX 控件技术被第三方软件厂商广泛使用,ActiveX 控件在网上发布的软件安装包中随处可见,例如在线病毒扫描、网络电话、网络游戏等。由于 ActiveX 控件接口对外部环境是开放的,一些控件中的漏洞很快就会被攻击者发现并加以利用,造成了 ActiveX 控件漏洞具有巨大的危害性。所以系统地研究 ActiveX 控件的安全问题,如何更有效、及时地发现隐藏的 ActiveX 控件漏洞,成为漏洞挖掘领域重要的研究方向。

2 ActiveX 控件漏洞自动发掘平台

2.1 ActiveX 控件分析

ActiveX 控件是 ActiveX 技术之一,ActiveX 技术是在 1996 年由微软提出来的一项基于 Component Object Model (COM) 和 Object Linking and Embedding (OLE) 的开发技术,是微软为了适应 Internet 和 Web 的发展对 OLE 进行的扩展。随着 ActiveX 技术的发展,ActiveX 很快成为了微软一项全新的技术。ActiveX 控件的出现标志着 COM 组件在微软 IE 浏览器中得到实际的应用,这种应用也标志着第三方软件和 IE 可以快速交互。ActiveX 控件广泛地应用在 Windows 操作系统中,并且大大简化了软件开发者对 IE 外部功能的开发。

现在的 ActiveX 控件等价于以前的 OLE 控件,一个典型的控件包括设计时和运行时的用户界面,唯一的 IDispatch 接

收稿日期:2008 - 04 - 03;**修回日期**:2008 - 06 - 16。

作者简介:吴毓书(1983 -),男,河北张家口人,硕士研究生,主要研究方向:网络与信息系统安全; 周安民(1963 -),男,四川成都人,研究员,主要研究方向:网络与信息系统安全; 吴少华(1977 -),男,福建福安人,讲师,博士研究生,主要研究方向:网络与信息系统安全; 何永强(1982 -),男,四川彭州人,主要研究方向:网络安全; 徐威(1983 -),男,江西高安人,硕士研究生,主要研究方向:网络与信息系统安全。

口定义了控件的属性和方法,唯一的 IConnectionPoint 接口定义控件可引发的事件。一个控件可以在容器中运行,所以从运行的角度看它类似于一个 DLL。由于在 IE 中添加了对控件的支持,所以用户可以在 Web 页面中对控件进行操纵。

ActiveX 控件可快速实现小型的组件重用、代码共享,但是 ActiveX 控件对于最终用户并不能直接使用,因为 ActiveX 控件必须先在 Windows 中注册。ActiveX 控件可以通过很多方式安装在系统中,除了 IE 和系统的一部分控件,开发人员也可以安装和注册自己编写的对 IE 功能扩展的 ActiveX 控件[2]。

ActiveX 控件通常是通过函数 CoCreateInstance 初始化,该函数需要指定一个类标识符(ClassID,CLSID),当控件对象创建成功,这个 128 b 的 CLSID 就成为该控件的唯一标识。ActiveX 控件的功能性通过接口定义。用户可以通过访问对象获得接口和函数。

除了 CLSID,控件对象还提供了编程标识符(ProgramID,ProgID),为了方便用户使用,它用的是易于读写的字符串形式。CLSID 和 ProgID 都在注册表中定义,并且可以通用。

ActiveX 控件与其他的 COM 对象一样使用了相似的 COM 接口, ActiveX 控件的属性和接口可以通过函数 QueryInterface 动态获取[3]。ActiveX 控件的这些性质为用户操纵控件提供了极大的便利。

2.2　漏洞自动发掘平台设计与实现

根据上述对 ActiveX 控件的研究,笔者利用 Python 语言设计一个可扩展的针对 ActiveX 控件的 Fuzzing 发掘平台。设计思路大体如下:1)枚举当前所有的可加载的 Active 控件,如果该控件不可加载则丢弃;2)对每一个可加载的控件枚举其方法和属性;3)解析控件的属性并确定控件方法的参数及其类型;4)根据控件导出函数的参数类型,产生测试数据,生成测试样本;5)执行样本并捕获异常,保存异常时的数据。Fuzzing 发掘平台流程如图 1 所示。

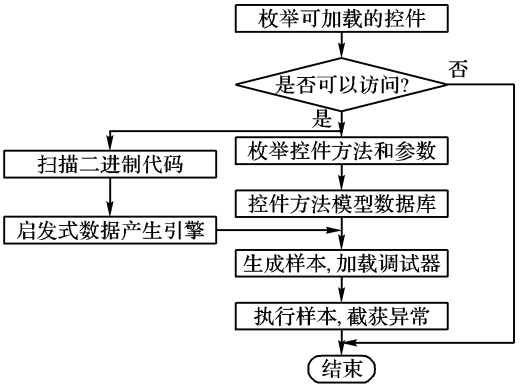

图 1　Fuzzing 发掘平台流程

1)判断控件是否可以被访问模块。当 IE 运行一个 ActiveX 控件时,需要确认该控件是否是安全的,首先查询 ActiveX 组件是否实现了 IObjectSafety 安全接口,并且返回脚本安全;其次查询 ActiveX 组件是否在注册表里表明自己实现了控件属性是脚本安全或初始化安全,这些安全属性在注册表 Component Categories 子键下定义,如果返回对于任何不安全接口,IE 将阻止该控件的运行,所以在枚举控件时,需要将这些不能运行的控件排除在外。

```
##CATID_SafeForScript
If Skey = "{7DD95801-9882-11CF-9FA9-00AA006C42C4}":
Return False
## CATID_SafeForInitialization
If Skey = "{7DD95802-9882-11CF-9FA9-00AA006C42C4}":
Return False
```

除此之外,IE 可以允许用户通过设置 ActiveX 控件 CLSID 的禁用标识位(Kill Bit)禁止 ActiveX 控件在 IE 中运行。因此对于每一个被枚举出来的控件需要再次判断是否被禁用。如果禁用标识位设置为 0x00000400,则表示该控件已被禁用,不进行加载。

2)枚举控件方法及方法参数模块。Python 语言提供了 COM 开发的接口[4-5],例如 win32api、win32com、Pythoncom 等,这些接口都可以方便操作控件,下列代码展示了如何利用 Python 语言枚举 ActiveX 控件的属性、方法以及其参数。

```
Adobe = r"D:\Program Files\...\ActiveX\AcroIEhelper.dll"
###加载 Adobe 控件
Target = Pythoncom.LoadTypeLib(Adobe)
###循环获取类型信息和属性
Print "properties:"
Info = Target.GetTypeInfo()
Attr = Info.GetTypeAttr()
For I in xrang(attr.cvars):
  Id = info.GetVarDesc(i)[0]
  Names = info.GetNames(Id)
```

在 ActiveX 控件中,数据是以 VARIANT 数据类型组织。VARIANT 数据结构支持整数、浮点型数、字符串、布尔型数等。这里需要将其表示的数据类型转换为 Python 的数据类型,以方便输入测试数据。表 1 展示了 Python 中的数据类型和与其等价的 VARIANT 数据类型[6-7]。

表 1　Python\VARIANT 数据类型对比

Python 数据类型	VARIANT 数据类型
Integer	VT_I4
String	VT_BSTR
Float	VT_R8
None	VT_NLL
True/False	VT_BOOL

```
###循环获取方法名和参数
Print "methods:"
For I in xrang(attr.cFuncs)
  Desc = info.GetFuncDesc(I)
If Desc.wFuncFlags:
Continue
  Id = Desc.memid
  Names = info.GetNames(Id)
```

3)启发式 Fuzzing 数据引擎模块。Fuzzing 技术的优势是自动化,一个好的 Fuzzing 平台核心在于其能够在不和用户交互的情况下生成高效的输入数据。传统的 Fuzzing 是通过预定模式,产生随机数据作为 Fuzzing 数据,很显然,在这种情况下,Fuzzing 效率低,因此一个良好的 Fuzzing 平台需要构造一个高效的数据引擎。下面对如何构造一个高效的数据引擎做进一步介绍。

对于字符串数据而言,可以看作由不同字符组合而成。根据字符在实际中的使用,将字符分为四种类型,分别是特殊字符、阿拉伯数字、字母和分隔字符,如表 2 所示。

引擎产生的字符串数由这四类字符组合而成,见式(1):

$$\begin{cases} String = Asc(t,l) \leftarrow Select(t) \times L \\ Select(x) = \{x \mid \exists x(x \in T)\} \\ T = \{Type1 \mid Type2 \mid Type3 \mid Type4\} \\ L = length \end{cases} \tag{1}$$

Select 函数表示从 *T* 类型中选取某种类型，*L* 表示字符串的长度。字符串由 *Asc* 函数生成，根据不同的字符类型组合和长度生成字符串。

对于整型数据，0～0xFFFFFFFF 是 32 位整数的边界，将此范围内的所有数据都进行测试是不明智的，也是不现实的，需要选取一个合理的数据子集。将生成的整数子集合表示为式(2)：

$$\begin{cases} Num = \{B_1 \mid B_2 \mid B_3 \mid \cdots\} \\ B_i = \{PB_i \mid NB_i\} = 2^{32-i} \pm \Delta\} \end{cases} \tag{2}$$

当 $i = 0,1,2$ 时，测试的整形数据范围为：

$MAX32 - \Delta \leqslant MAX32 \leqslant MAX32 + \Delta$

$MAX32/2 - \Delta \leqslant MAX32/2 \leqslant MAX32/2 + \Delta$

$MAX32/4 - \Delta \leqslant MAX32/4 \leqslant MAX32/4 + \Delta$

在这里 *MAX*32 表示最大的 32 位整数(0xFFFFFFFF)，Δ 是选择的整数测试范围，也可以随机加大测试范围。当 $16 < i < 24$ 时，该集合表示 16 b 的整数，同样该集合同样也可以表示 8 b 整数的测试。

表2 字符分类

T	十六进制数	内容类型
Type1	0x20～0x2E；0x3A～0x40	特殊字符
Type2	0x30～0x39；	数字
Type3	0x41～0x5A；0x61～0x7A	字母
Type4	0x5C(‘\’)0x2F(‘/’)	分隔字符

此外，本平台使用了扫描二进制代码获取特征字符方法，该模块可以动态地扫描控件中特殊的字符，把扫描得到的敏感数据或者字符动态地添加到数据生成引擎中，以增强 Fuzzing 的有效性。参见如下的代码：

```
Char cpy[16]; char * cursor, index;
for( cursor = inbuf; * cursor; cursor ++ )
{if ( * cursor = = '\') index = cursor;
 else
 length ++;
}
if( length < sizeof( cpy) -2)
{ strcpy( cpy, inbuf); }
```

根据源代码可知，代码作者试图以“\”作为字符串的分界符，也在拷贝前判断了输入字符串的长度，如果随机地输入字符串进行测试，Fuzz 的效率很低。

将代码反汇编后，部分二进制代码如下：

```
_text: 00401036   cmp   al, 5Ch
_text: 00401038   jz    short loc_0_40103B
_text: 0040103A   inc   ebx
```

进行二进制代码分析后，提取特征字符“\”(十六进制 ASCII 码 0x5C)，把“\”字符传送给数据产生引擎，由此产生特殊测试字符串。在输入字符串”1234\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\1234”后程序会出现异常，输入的字符串绕过了边界检查，拷贝长度超过了预定大小，最终导致缓冲区溢出。因此，经过二进制代码分析，提取敏感的数据和特征字符，有利于提高 Fuzzing 的效率。

3 测试结果

使用自主开发的 Fuzzer 工具 AxFuzzer 对一台安装有 RealPlayer 播放软件的虚拟主机进行测试，并同时使用目前流行的 ActiveX 控件 Fuzzer 工具 Axman1.0 做比较测试，测试目标为该播放软件的相关 ActiveX 控件。

Fuzzer 得到的漏洞信息描述如下。

1) ierpplug_dll ActiveX 控件中 OpenURLInPlayerBrowser 函数处理超长的 URL 出错。漏洞触发的原因是当传递超常的 URL 串给函数时，在为该串分配栈时，导致内存耗竭而出错。

2) ierpplug_dll ActiveX 控件中 Import 函数处理超长播放列表名(PlayList)导致栈溢出，在触发该漏洞时要保证其余参数正确，当测试的字符串长度大于 0x7FFF 时，触发漏洞。

3) ierpplug_dll ActiveX 控件中 Import 函数在处理媒体信息(metedata)时出错。

下面就生成样本数量、Fuzz 消耗时间、Fuzz 结果三个方面进行比较，如表 3 所示。

表3 结果比较

工具	生成样本数	消耗时间	Fuzz 结果
AxFuzzer	17 830	5 小时 30 分	(1)(2)(3)
Axman1.0	35 690	6 小时 40 分	(1)(3)

从表 3 可以看出，两种工具在对同一控件进行测试时，加入动态扫描模块的 AxFuzzer 在生成的样本数量上要远远小于 Axman1.0，但是由于 AxFuzzer 需要对控件进行动态扫描，所以，AxFuzzer 花费的时间并没有随样本数量成比例的减少。同时，AxFuzzer 在生成较少样本的情况下，成功找出 3 个漏洞，说明 AxFuzzer 在生成样本的有效性上有一定提高。

综上所述，本文开发的 AxFuzzer 具有实用性，较目前流行的 ActiveX 控件 Fuzz 软件，Axman 在效率上有所提高。

4 结语

本文对 ActiveX 控件展开研究，重点研究了如何构建一个在 Windows 平台下可以对 ActiveX 控件进行有效 Fuzzing 的工具。并针对 Windows 平台下的 RealPlayer 播放软件进行了安全测试，发现 ActiveX 控件中三个漏洞，包括两个已经公布的和一个未被公布的漏洞。测试结果不仅体现了该平台的适用性，更重要的是反映了目前 Windows 平台下 ActiveX 控件存在着相当大的安全问题。在下一步研究中，可以从很多方面扩展此 Fuzzing 平台，比如考虑更先进的数据产生方式，进一步完善该平台的模型。

参考文献：

[1] FX of Phenoelit. Bug hunting[EB/OL]. [2008-01-01]. http://www.phenoelit.de/stuff/Bugs.pdf.

[2] BOX D. Essential COM [M]. Reading, MA: Addison-Wesley, 1997.

[3] Warlord. ActiveX-Active Exploitation [EB/OL]. [2008-01-01]. packetstormsecurity.org/papers/attack/activex.pdf.

[4] SUTTON M, GREENE A, AMINI P. FUZZING brute force vulnerability discovery [M]. Reading, MA: Addison-Wesley, 2007.

[5] HAMMOND M, ROBINSON A. Python programming on Win 3 2 [EB/OL]. [2008-01-05]. O'Reilly, 2000. http://download.csdn.net/source/203224.

[6] DOWD M, MCDONALD J, SCHUH J. The art of software security assessment: Identifying and preventing software vulnerabilities[M]. Reading, MA: Addison-Wesley, 2006.

[7] SPARKS S, EMBLETON S, CUNNINGHAM R, *et al*. Automated vulnerability analysis [C]// Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual. [S.l.]: IEEE Press, 2007: 477-486.