Web安全技术

Web Security

2.4 浏览器与扩展安全

刘奇旭

liuqixu@iie.ac.cn

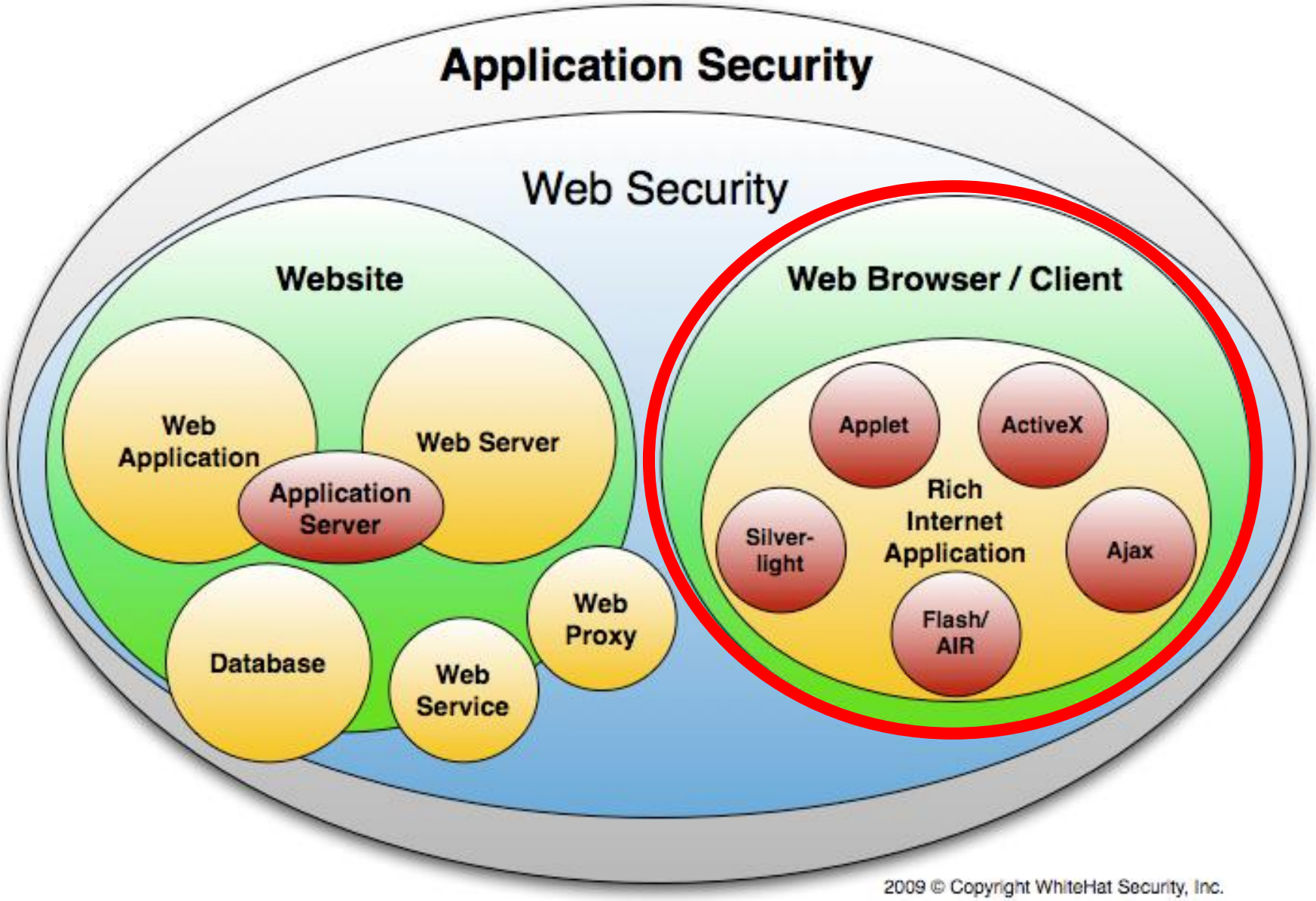
中科院信工所 第六研究室

一章一问

- 浏览器的安全问题及解决方法。

内容回顾

WEB SECURITY



极光行动(AURORA)

2010年1月

2010年1月12日，Google称Gmail服务器遭到来自中国的攻击。

搜集Google员工在Facebook、Twitter等社交网站上发布的信息；

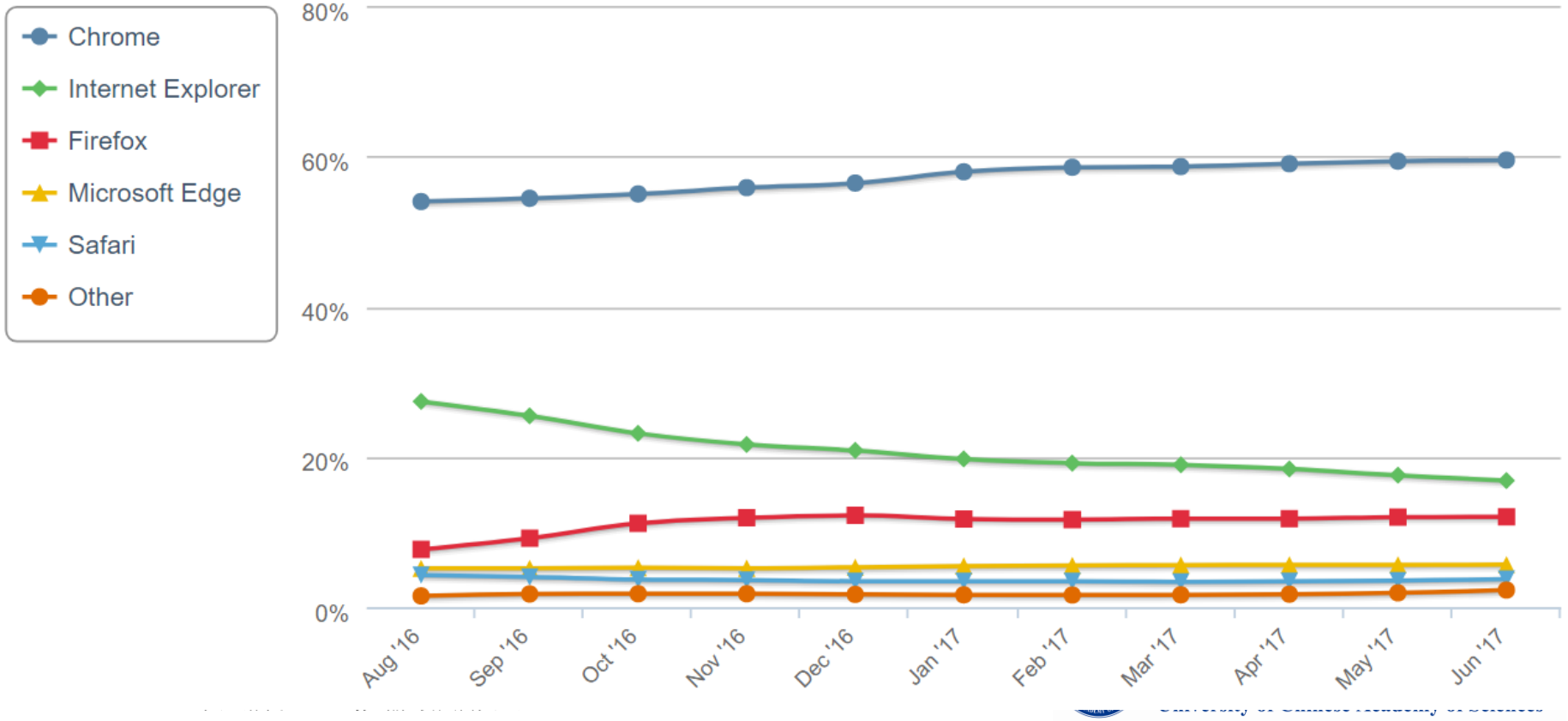
利用动态DNS供应商建立托管伪造照片网站的Web服务器，Google员工收到来自信任的人发来的网络链接并且点击，含有shellcode的JavaScript造成IE浏览器溢出，远程下载并运行程序；

通过SSL安全隧道与受害人机器建立连接，持续监听并最终获得该雇员访问Google服务器的帐号密码等信息；

使用该雇员的凭证成功渗透进入Google邮件服务器，进而不断获取特定Gmail账户的邮件内容信息。

浏览器市场份额

- 根据NetMarketShare的报告显示，截至2017年6月，谷歌Chrome市场份额占比59.49%，相比去年同期增长了10.84%。Internet Explorer拥有16.84%的份额，相比去年同期下跌了14.81%，几乎腰斩。



BROWSER SECURITY

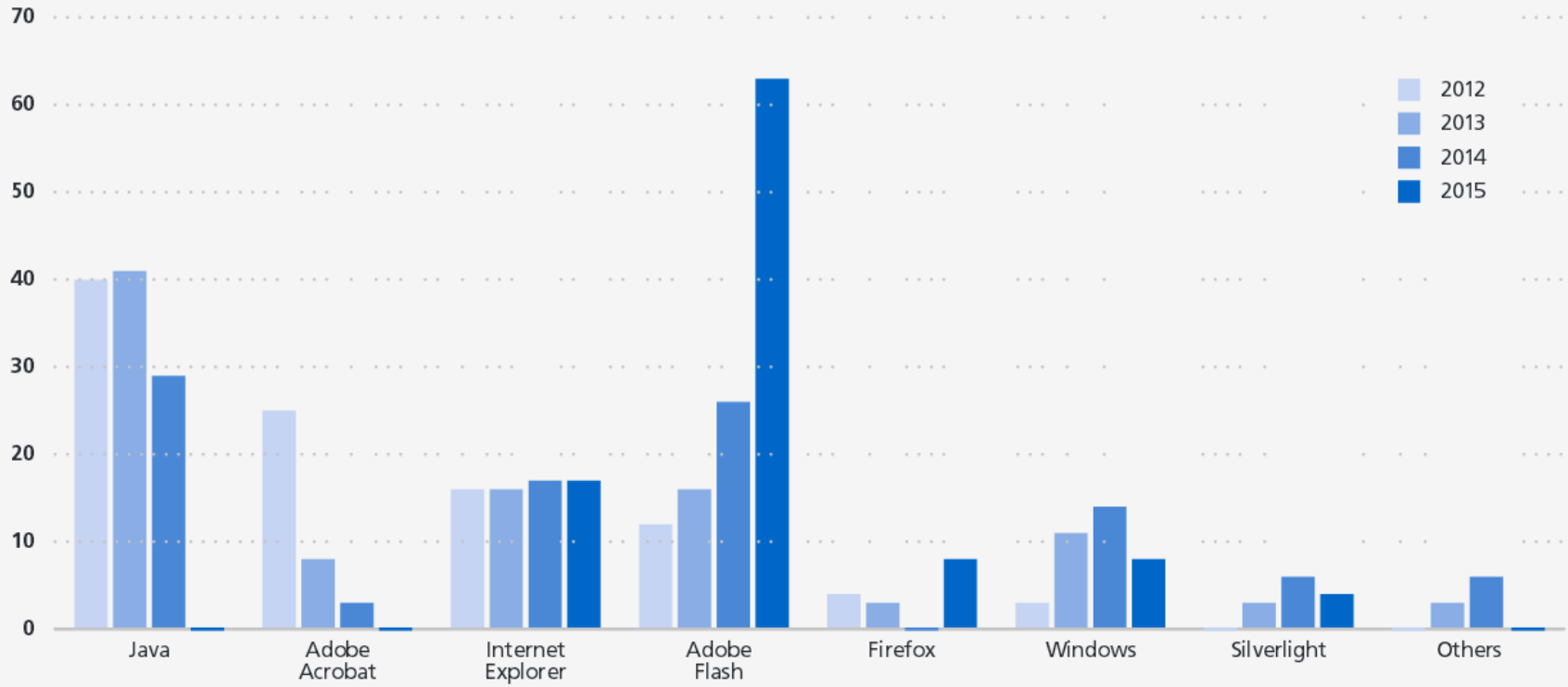
VULNERABILITY REVIEW
2017
Key figures and facts on vulnerabilities from a global information security perspective
Published **March 13, 2017**

Vulnerabilities indicate the number of new vulnerabilities in the last 12 months.
Market share is percentage of Personal Software Inspector users with the product installed on their PC.

Program	Unpatched	Market Share	Vulnerabilities
Google Chrome	39%	68%	335
Mozilla Firefox	41%	66%	258
Internet Explorer	6%	99%	114
Opera	27%	7%	6

2015年十大最危险漏洞

- 2012-2014年，Java的漏洞“利用率”一直都是最高的，2015年则是Flash第一次“夺魁”，Java则几乎销声匿迹，位列第二第三的分别是IE浏览器、Windows操作系统。



HACKING TEAM

- 2015年，意大利专门从事hacking活动的Hacking Team被黑，400G资料在网上泄露传播。
- 本次泄漏包括了Adobe Flash Player 0day（影响IE、Chrome等）。

Index of /

Name	Last Modified	Size	Type
Parent Directory/		-	Directory

Adobe Flash 的安全性一直饱受争议，被誉为“黑产军团的军火库”。APT28 和 Pawn Strom 都利用了 Adobe Flash 的 0day 漏洞进行 APT 攻击，2015 年全年上报的 Flash 漏洞更是多达 300 余条。Hacking -Team 的数据泄露事件，将 Flash 漏洞的实际危害性和影响力推到当年顶点，暴露出的三个漏洞几乎能够影响所有平台、所有版本的 Flash。其中被发现的第二个漏洞（CVE-2015-5122）甚至被黑客团队戏称为“过去四年里最漂亮的 Flash 漏洞”。

Exploit_Delivery_Network_android.tar.gz	2015-Jul-06 13:31:32	797.1M	application/gzip
Exploit_Delivery_Network_windows.tar.gz	2015-Jul-06 13:43:50	716.5M	application/gzip
support.hackingteam.com.tar.gz	2015-Jul-06 21:22:48	15.1G	application/gzip

lighttpd/1.4.35

PWN2OWN

Pwn2Own是全球黑客顶级赛事，以黑客“世界杯”著称，由美国国防部安全服务商、惠普旗下TippingPoint的ZDI项目组主办，微软、苹果、谷歌、Adobe、Intel等领导厂商提供赞助，旨在支持和鼓励帮助他们发现操作系统、浏览器、游戏引擎等软件平台安全漏洞的优秀人才。

表2 我国网络安全研究团队/个人成功参加国际性安全破解大赛的历史成绩

年份	安全破解赛	队伍／个人	比赛项目
2016	Pwn2Own	腾讯安全Sniper战队（KEEN和电脑管家）	Adobe Flash
2016	Pwn2Own	腾讯安全Sniper战队（KEEN和电脑管家）	苹果Safari浏览器
2016	Pwn2Own	腾讯安全Sniper战队（KEEN和电脑管家）	微软Edge浏览器
2016	Pwn2Own	奇虎360-Vulcan	Adobe Flash
2016	Pwn2Own	奇虎360-Vulcan	Google Chrome浏览器
2016	Pwn2Own	腾讯安全Shield战队（电脑管家和KEEN）	苹果Safari浏览器

浏览器漏洞攻击包

browsersploit

BrowserExploit is an advanced browser exploit pack for doing internal and external pentesting, helping gaining access to internal computers.

I started this project years ago, when still exploiting IE 6, 7 and 8. The exploits in kit are old so it keep scripts kiddies from running it in the wild and achieve malicious task.

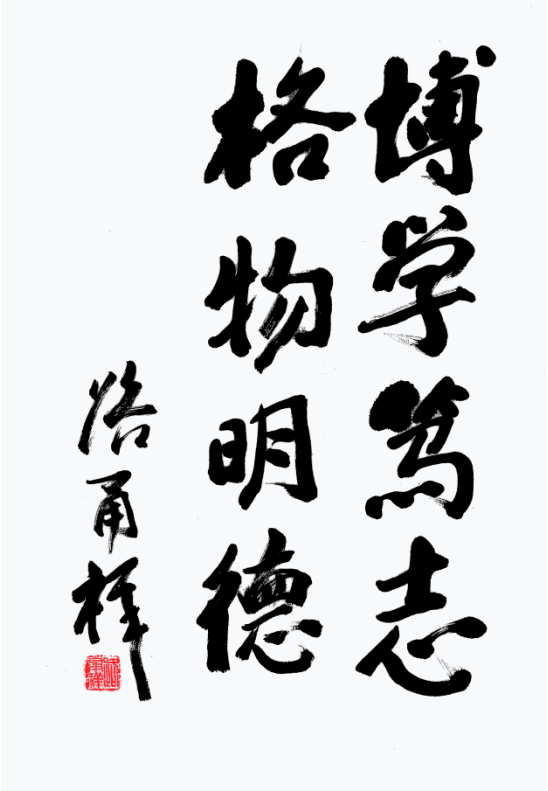
BrowserSploit use a lot of techniques to bypass anti-virus and is full of featured.

- Javascript obfuscation (XOR, JS Iframe Head, Cookie Encrypted, Split Encrypted Iframe, Base64 random space).
- Advanced exploitation techniques.
- Artificial Intelligence based on traffic learning.
- Multi-Users ready platform
- Filter Antivirus connections
- Evade AV domain filters
- Reverse Honeypot features to trick non legitimate users and sec users
- Bypass Windows DEP / ASLR / UAC
- Advanced polymorphic shellcoding

What it mean for the non-technical people: If you surf the web on your browser and you visit a page infected by an browser exploit pack, then you will likely be infected by malicious software without even notify it.

本章大纲

- 网页木马
- ActiveX
- Adobe Flash Player
- 浏览器安全机制



网页木马

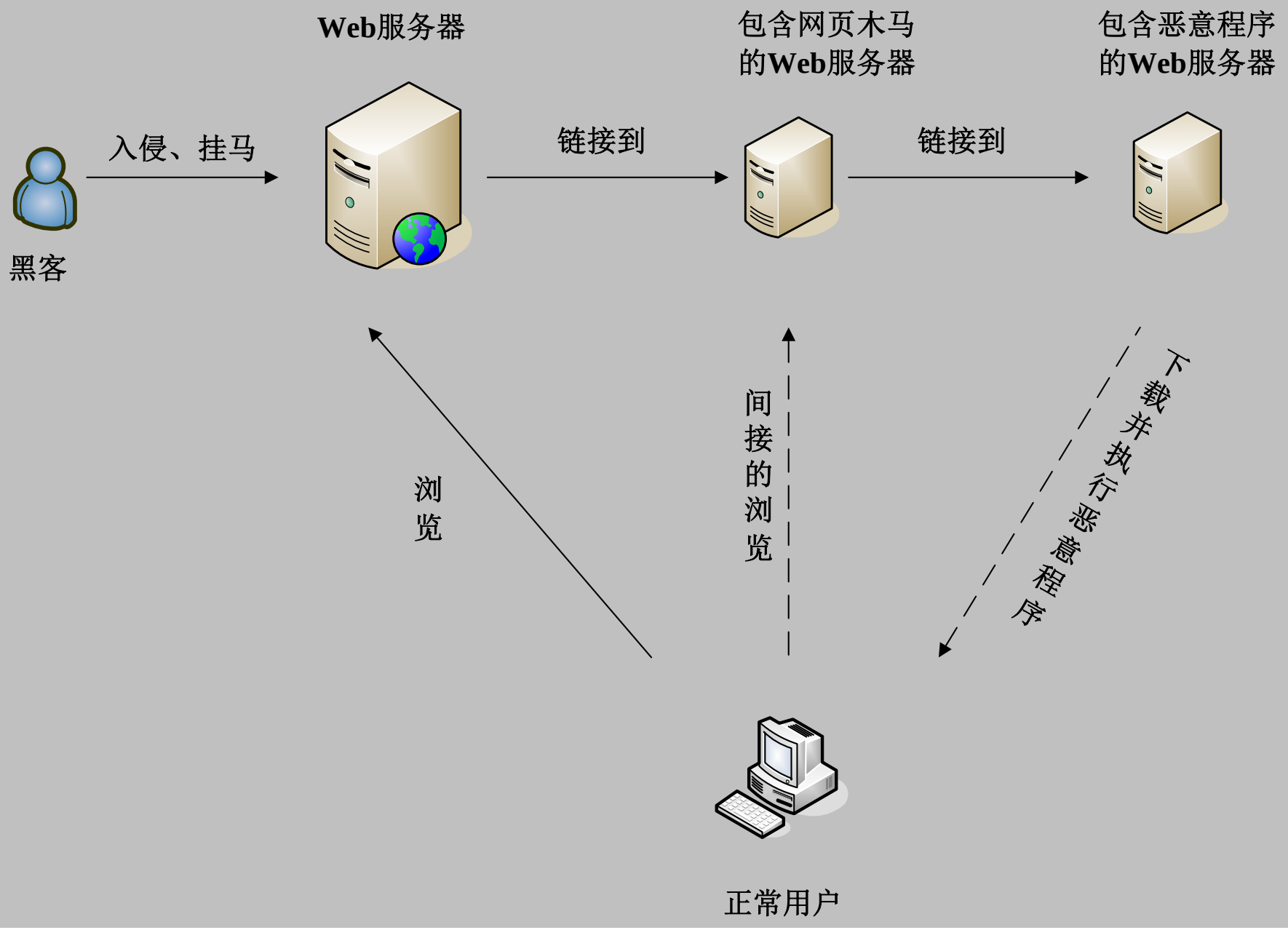
- 网页木马就是一个Web页面，可以是一个静态的HTML页面，也可是ASP、PHP、JSP等动态页面。

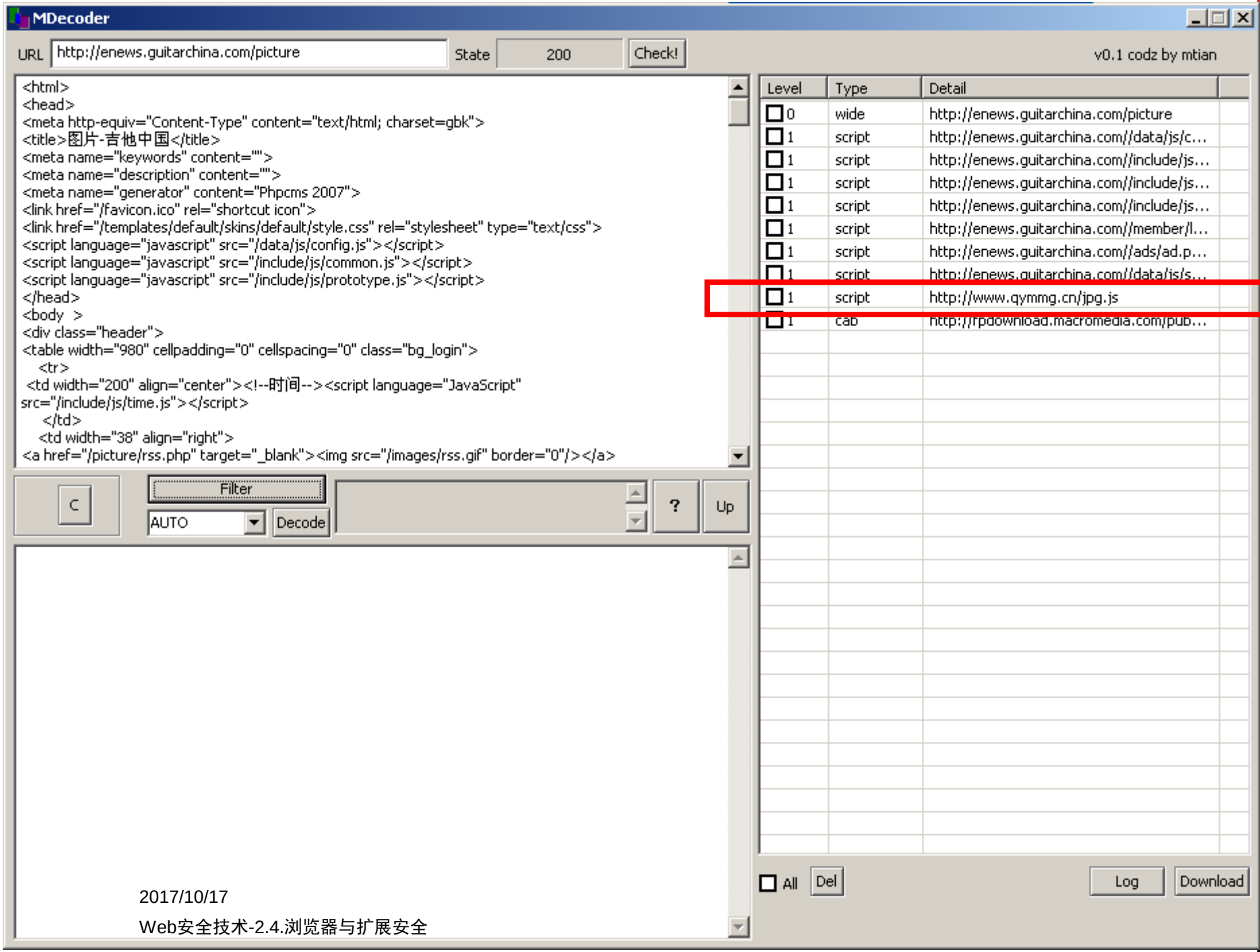
- 从表面上看，它和一个普通的页面并没有太大的区别，但是包含在HTML源代码中的恶意脚本可以使浏览器在后台、在用户不知情的情况下下载，并执行恶意的木马。

网页木马

- 网页木马出现的历史要比木马和病毒出现短得多；
- 早期大多数以恶意网页为主，比如劫持浏览器的首页，修改注册表，死循环弹出窗口等；
- 在2001年的时候出现过几个利用MIME头漏洞、BMP网页木马等。
- 真正的流行是在2004年，网页木马开始大量出现，并且传播手段从仅仅的Web传播到发展出了CHM网页木马、隐蔽在媒体文件中的RM/RMVB网页木马、WMV网页木马、Flash网页木马等几种新的形式；

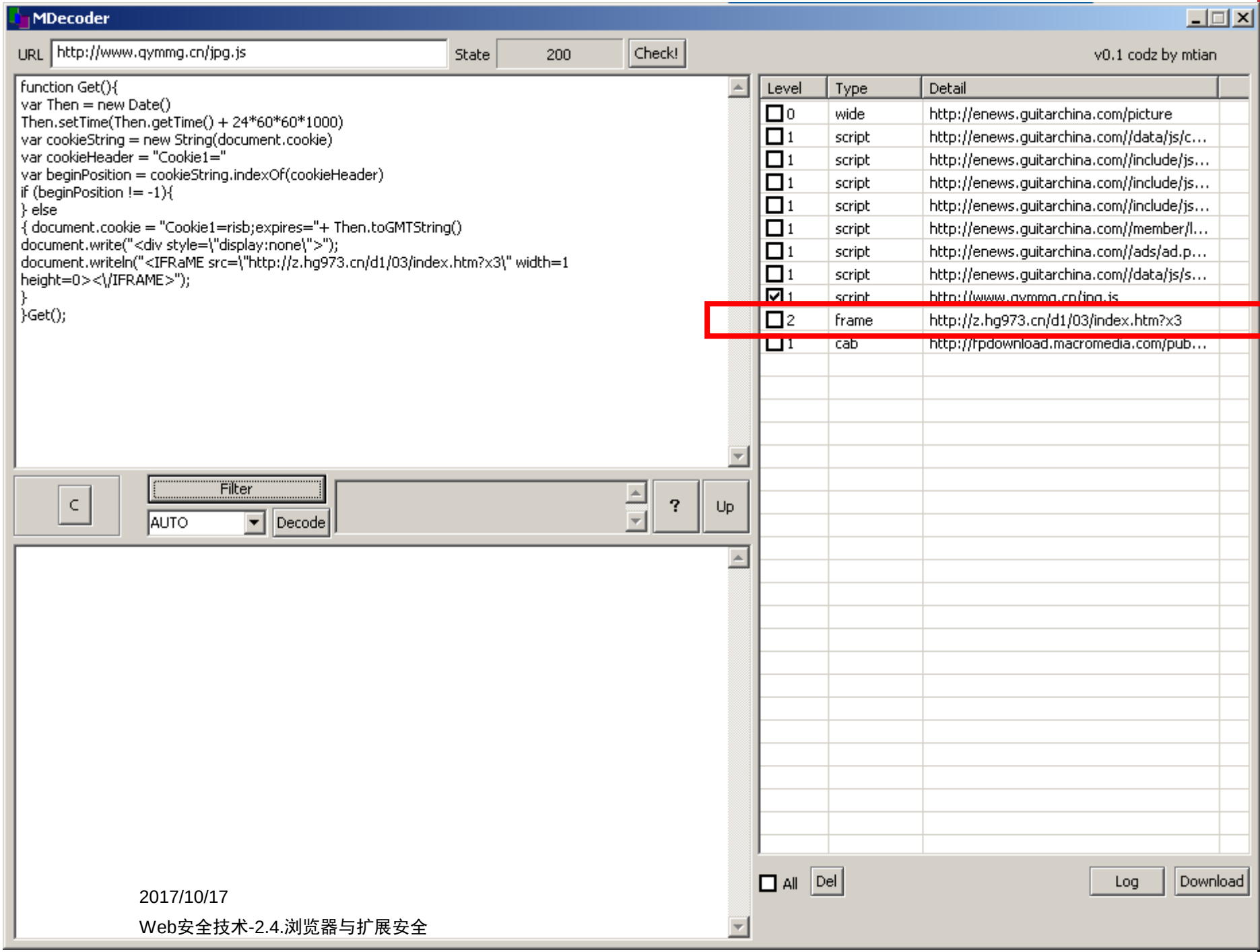
漏洞编号	漏洞名称
MS04-023	Microsoft HTML Help 任意代码执行漏洞
MS05-001	Microsoft IE Help ActiveX 控件本地安全域绕过漏洞
MS06-014	Microsoft MDAC RDS.Dataspace ActiveX 控件远程代码执行漏洞
MS06-057	Microsoft IE WebViewFolderIcon 远程整数溢出漏洞
MS06-071	Microsoft XML 核心服务 XMLHTTP 控件内存破坏漏洞
MS06-004	Microsoft IE WMF 图形解析内存破坏漏洞
MS06-005	Microsoft Windows Media Player 畸形位图文件处理堆溢出漏洞
MS06-013	Microsoft IE CreateTextRange 远程代码执行漏洞
MS06-055	Microsoft IE 畸形 VML 文档处理缓冲区溢出漏洞
MS06-067	Microsoft IE daxctle.ocx KeyFrame 方法堆溢出漏洞
MS06-068	Microsoft Agent Active 控件远程堆溢出漏洞
MS06-073	Microsoft Visual Studio “WMI Object Broker” 控件代码执行漏洞
MS07-009	Microsoft IE ADODB.Connection 对象 Execute 函数内存破坏漏洞
MS07-017	Microsoft Windows 动画光标畸形 ANI 头结构远程栈溢出漏洞
MS07-004	Microsoft Windows 矢量标记语言缓冲区溢出漏洞
MS07-027	Microsoft Windows 媒体服务器 mdsauth.dll 控件远程代码执行漏洞
MS08-052	Microsoft Windows GDI+中的漏洞可能允许远程执行代码
MS08-053	Microsoft Windows Media Encoder 9 中的漏洞可能允许远程执行代码
MS08-067	Microsoft Windows 服务器服务中的漏洞可能允许远程执行代码
MS08-078	Microsoft IE XML 解析引擎中的漏洞可能允许远程执行代码
MS09-002	Microsoft IE CFunctionPointer 函数内存破坏漏洞
MS09-014	Microsoft IE 多个内存破坏漏洞


Web服务器
包含网页木马
的Web服务器
包含恶意程序
的Web服务器
入侵、挂马
链接到
链接到
黑客
浏览
间接的浏览
下载并执行恶意程序
正常用户


MDecoder
URL
http://enews.guitarchina.com/picture
State
200
Check!
v0.1 codz by mtian
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gbk">
<title>图片-吉他中国</title>
<meta name="keywords" content="">
<meta name="description" content="">
<meta name="generator" content="Phpcms 2007">
<link href="/favicon.ico" rel="shortcut icon">
<link href="/templates/default/skins/default/style.css" rel="stylesheet" type="text/css">
<script language="javascript" src="/data/js/config.js"></script>
<script language="javascript" src="/include/js/common.js"></script>
<script language="javascript" src="/include/js/prototype.js"></script>
</head>
<body >

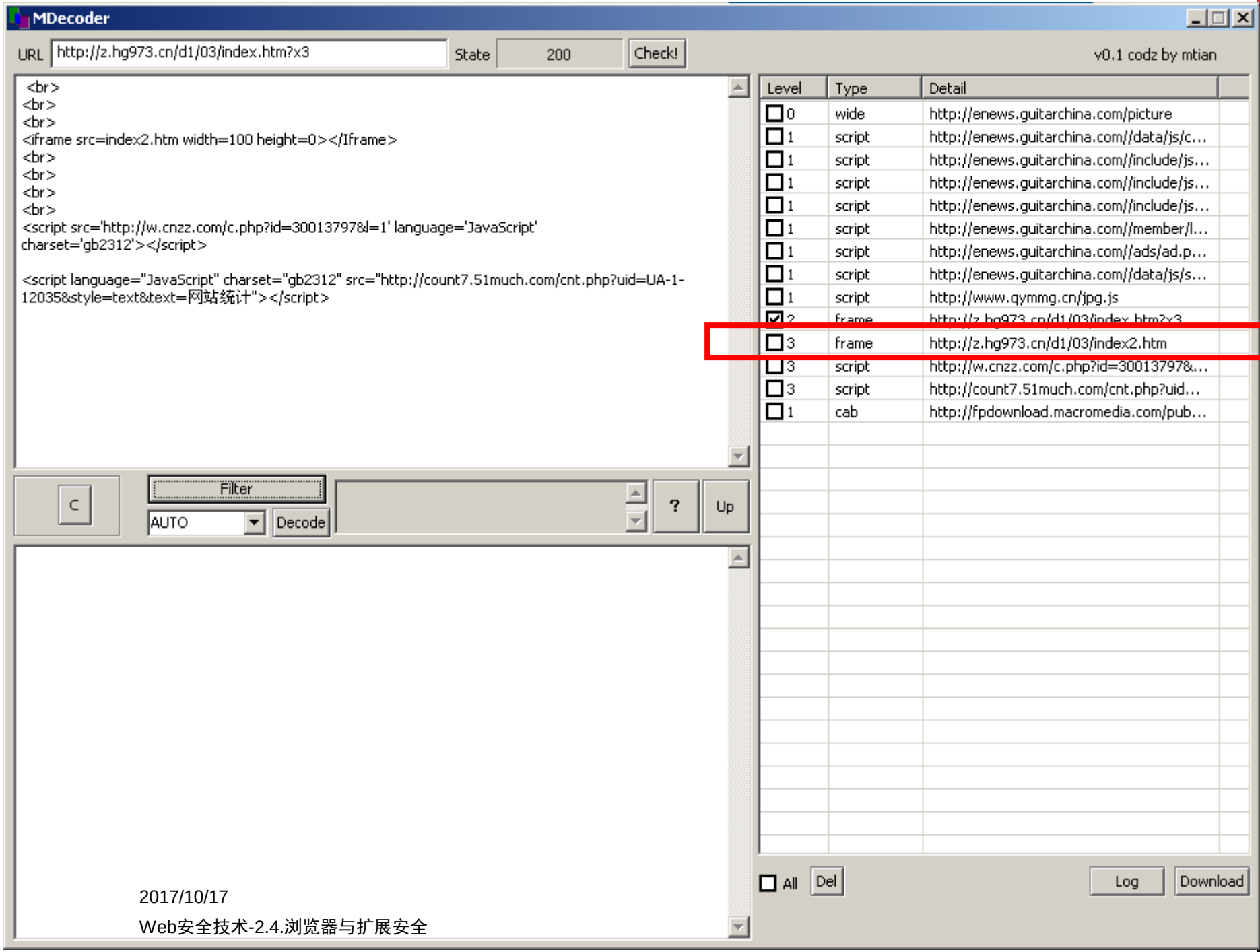
<table width="980" cellpadding="0" cellspacing="0" class="bg_login">
<tr>
<td width="200" align="center"><!--时间--><script language="JavaScript"
src="/include/js/time.js"></script>
</td>
<td width="38" align="right">

C
Filter
AUTO
Decode
?
Up
Level
Type
Detail
0
wide
http://enews.guitarchina.com/picture
1
script
http://enews.guitarchina.com//data/js/c...
1
script
http://enews.guitarchina.com//include/js...
1
script
http://enews.guitarchina.com//include/js...
1
script
http://enews.guitarchina.com//include/js...
1
script
http://enews.guitarchina.com//member/l...
1
script
http://enews.guitarchina.com//ads/ad.p...
1
script
http://enews.guitarchina.com//data/js/s...
1
script
http://www.qymmg.cn/jpg.js
1
cab
http://rpdownload.macromedia.com/pub...
All
Del
Log
Download
2017/10/17
Web安全技术-2.4.浏览器与扩展安全



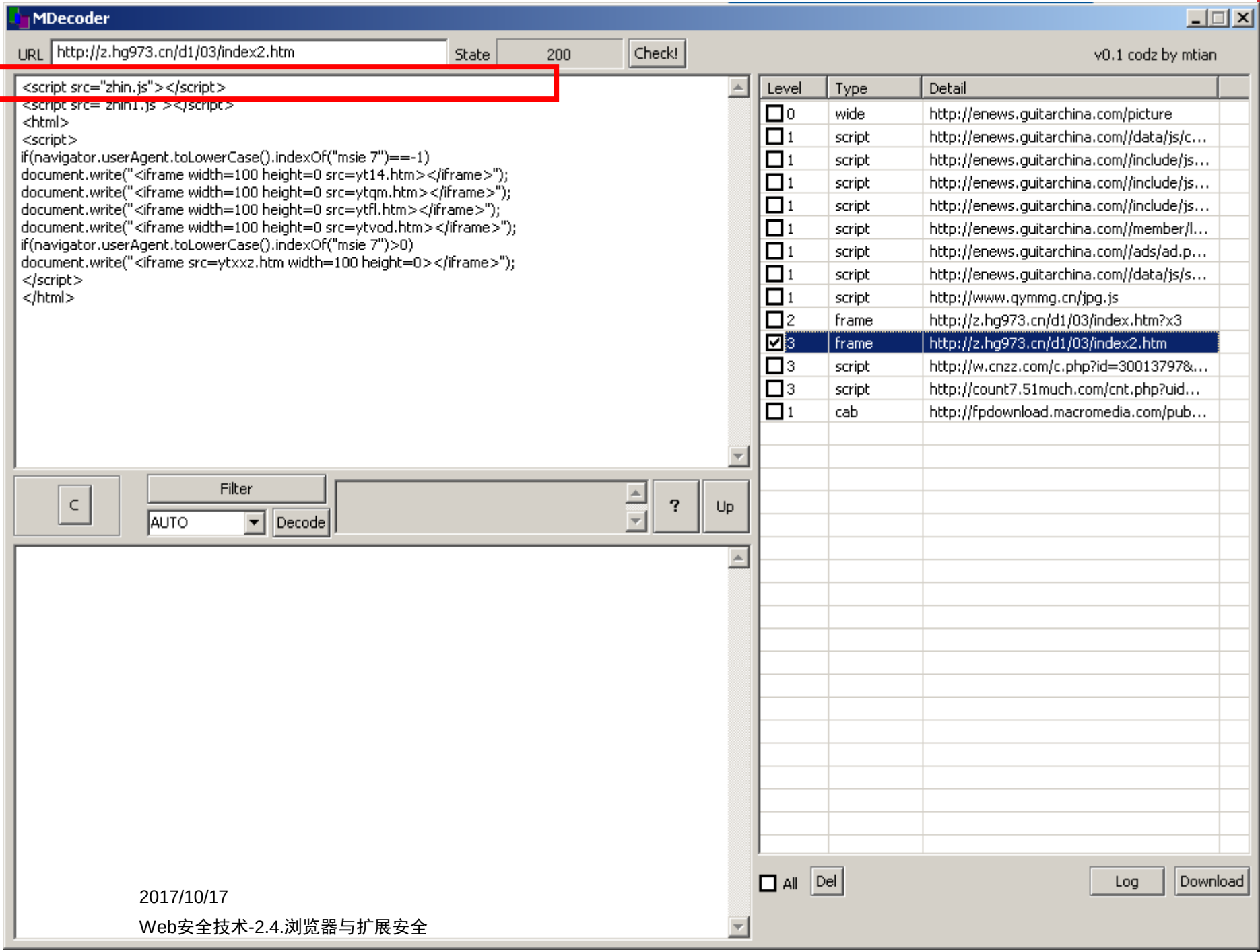
2017/10/17

Web安全技术-2.4.浏览器与扩展安全



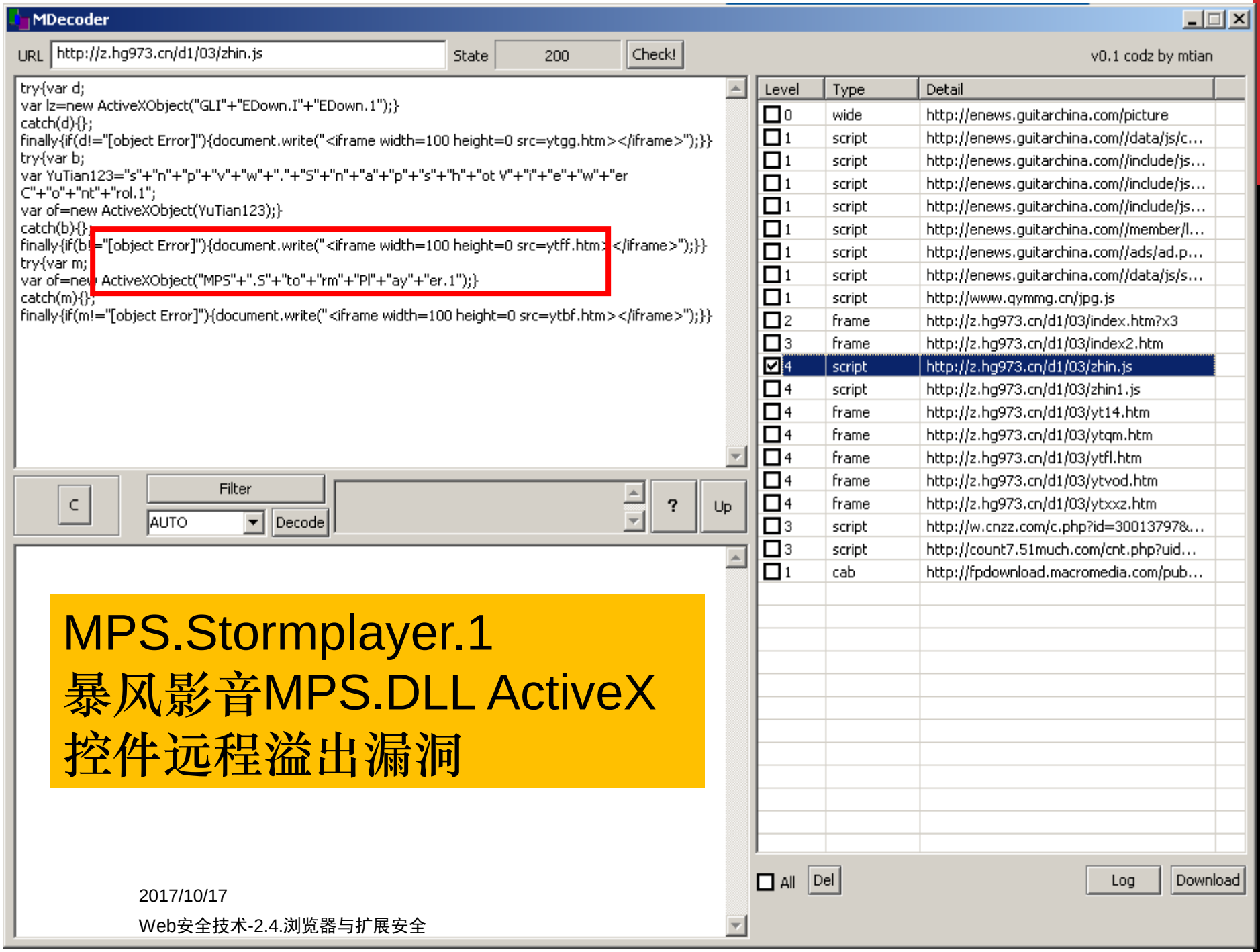
2017/10/17

Web安全技术-2.4.浏览器与扩展安全



2017/10/17

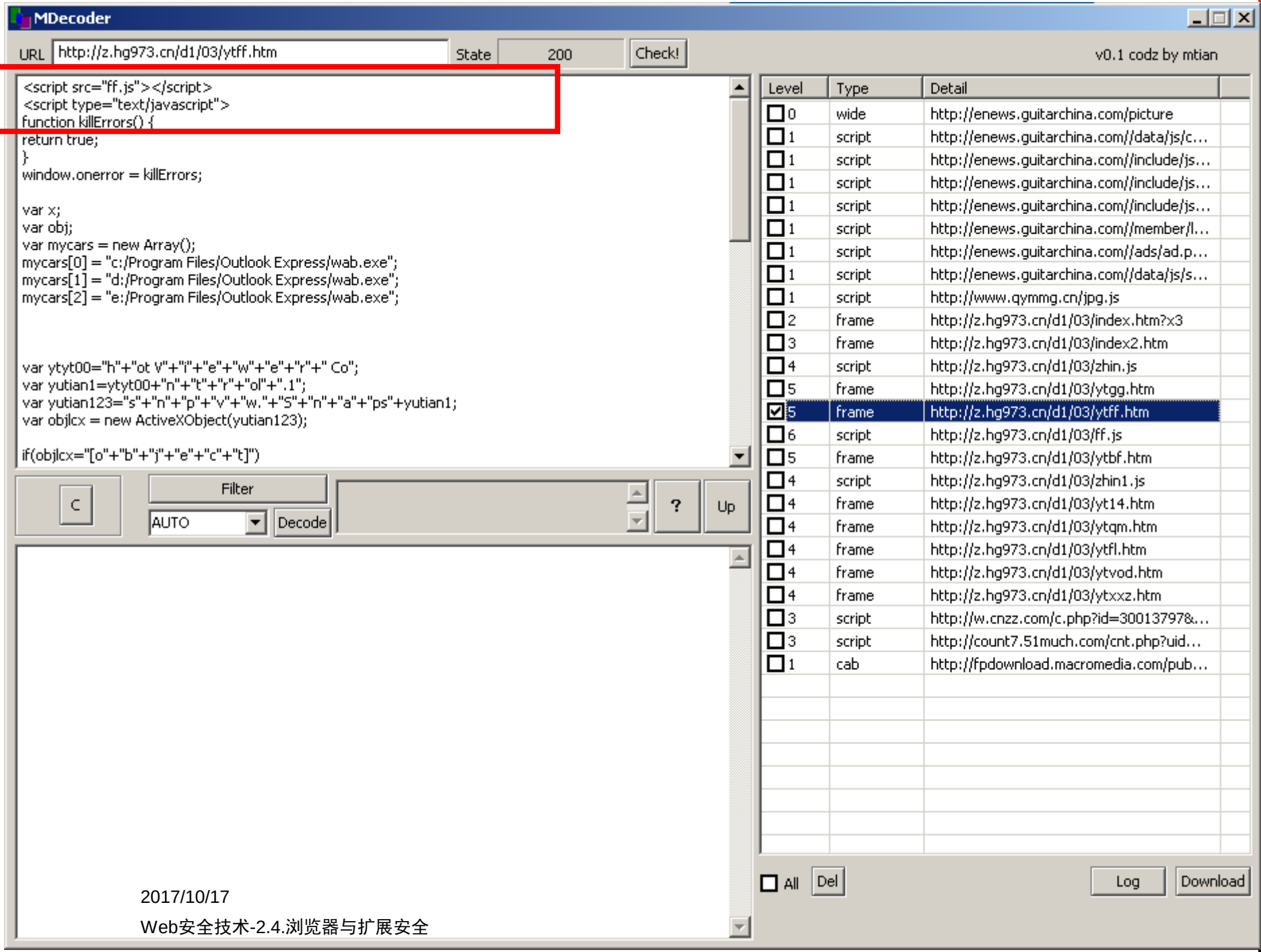
Web安全技术-2.4.浏览器与扩展安全



MPS.Stormplayer.1
暴风影音MPS.DLL ActiveX 控件远程溢出漏洞

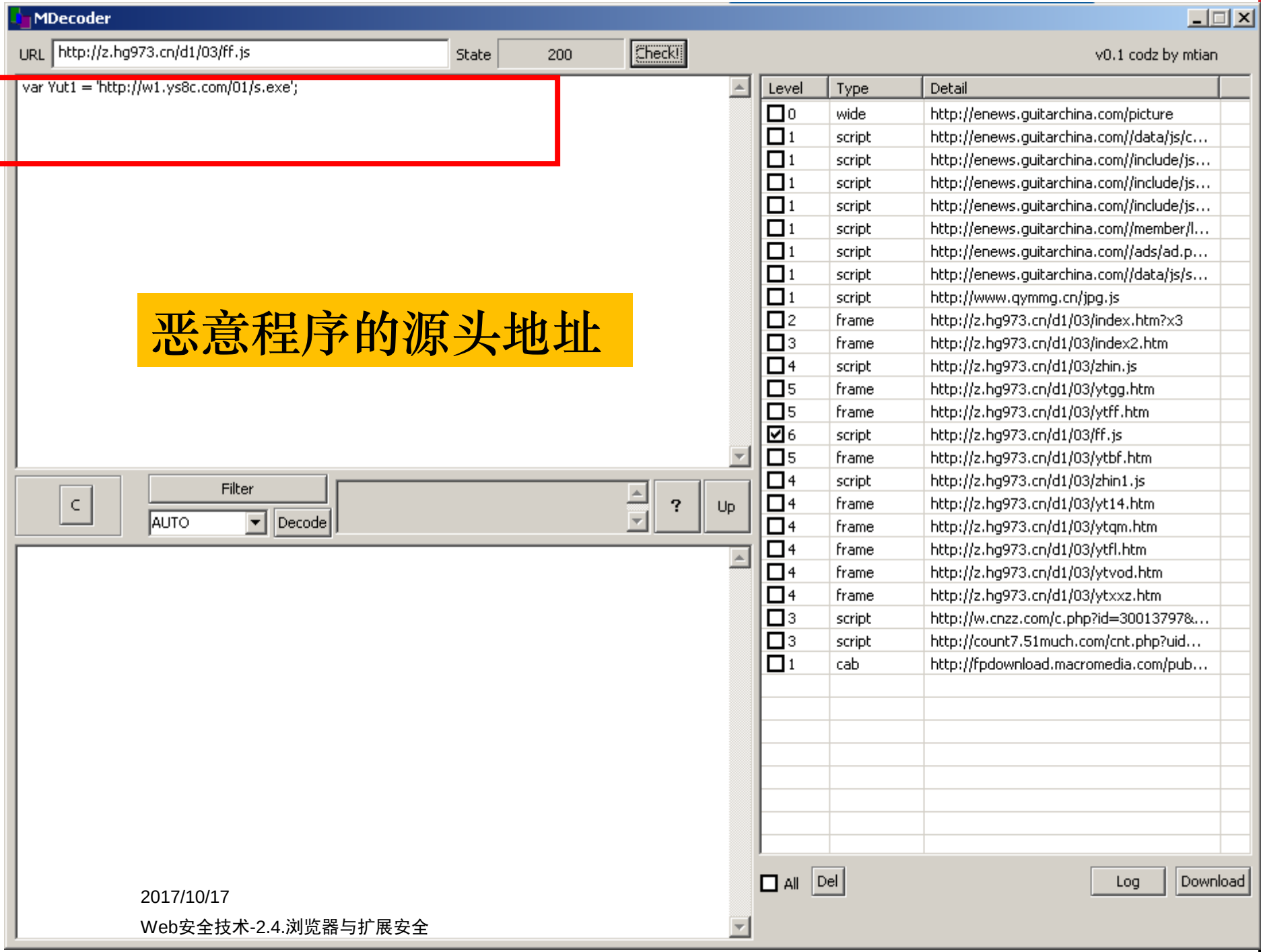
2017/10/17

Web安全技术-2.4.浏览器与扩展安全



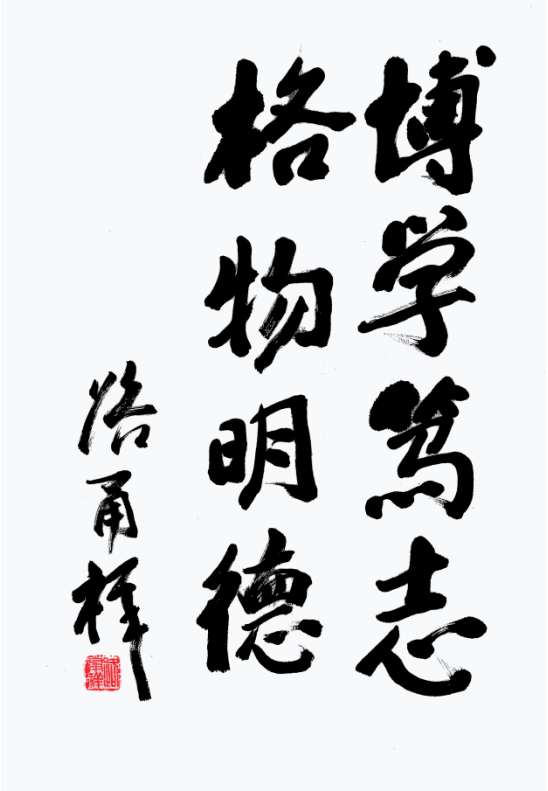
2017/10/17

Web安全技术-2.4.浏览器与扩展安全


MDecoder
URL
http://z.hg973.cn/d1/03/ff.js
State
200
Check!
v0.1 codz by mtian
var Yut1 = 'http://w1.ys8c.com/01/s.exe';
恶意程序的源头地址
C
Filter
AUTO
Decode
?
Up
Level
Type
Detail
0 wide http://enews.guitarchina.com/picture
1 script http://enews.guitarchina.com//data/js/c...
1 script http://enews.guitarchina.com//include/js...
1 script http://enews.guitarchina.com//include/js...
1 script http://enews.guitarchina.com//include/js...
1 script http://enews.guitarchina.com//member/l...
1 script http://enews.guitarchina.com//ads/ad.p...
1 script http://enews.guitarchina.com//data/js/s...
1 script http://www.qymmg.cn/jpg.js
2 frame http://z.hg973.cn/d1/03/index.htm?x3
3 frame http://z.hg973.cn/d1/03/index2.htm
4 script http://z.hg973.cn/d1/03/zhin.js
5 frame http://z.hg973.cn/d1/03/ytgg.htm
5 frame http://z.hg973.cn/d1/03/ytff.htm
6 script http://z.hg973.cn/d1/03/ff.js
5 frame http://z.hg973.cn/d1/03/ytbf.htm
4 script http://z.hg973.cn/d1/03/zhin1.js
4 frame http://z.hg973.cn/d1/03/yt14.htm
4 frame http://z.hg973.cn/d1/03/ytqm.htm
4 frame http://z.hg973.cn/d1/03/ytfl.htm
4 frame http://z.hg973.cn/d1/03/ytvod.htm
4 frame http://z.hg973.cn/d1/03/ytxxz.htm
3 script http://w.cnzz.com/c.php?id=30013797&...
3 script http://count7.51much.com/cnt.php?uid...
1 cab http://fpdownload.macromedia.com/pub...
All
Del
Log
Download

本章大纲

- 网页木马
- ActiveX
- Adobe Flash Player
- 浏览器安全机制

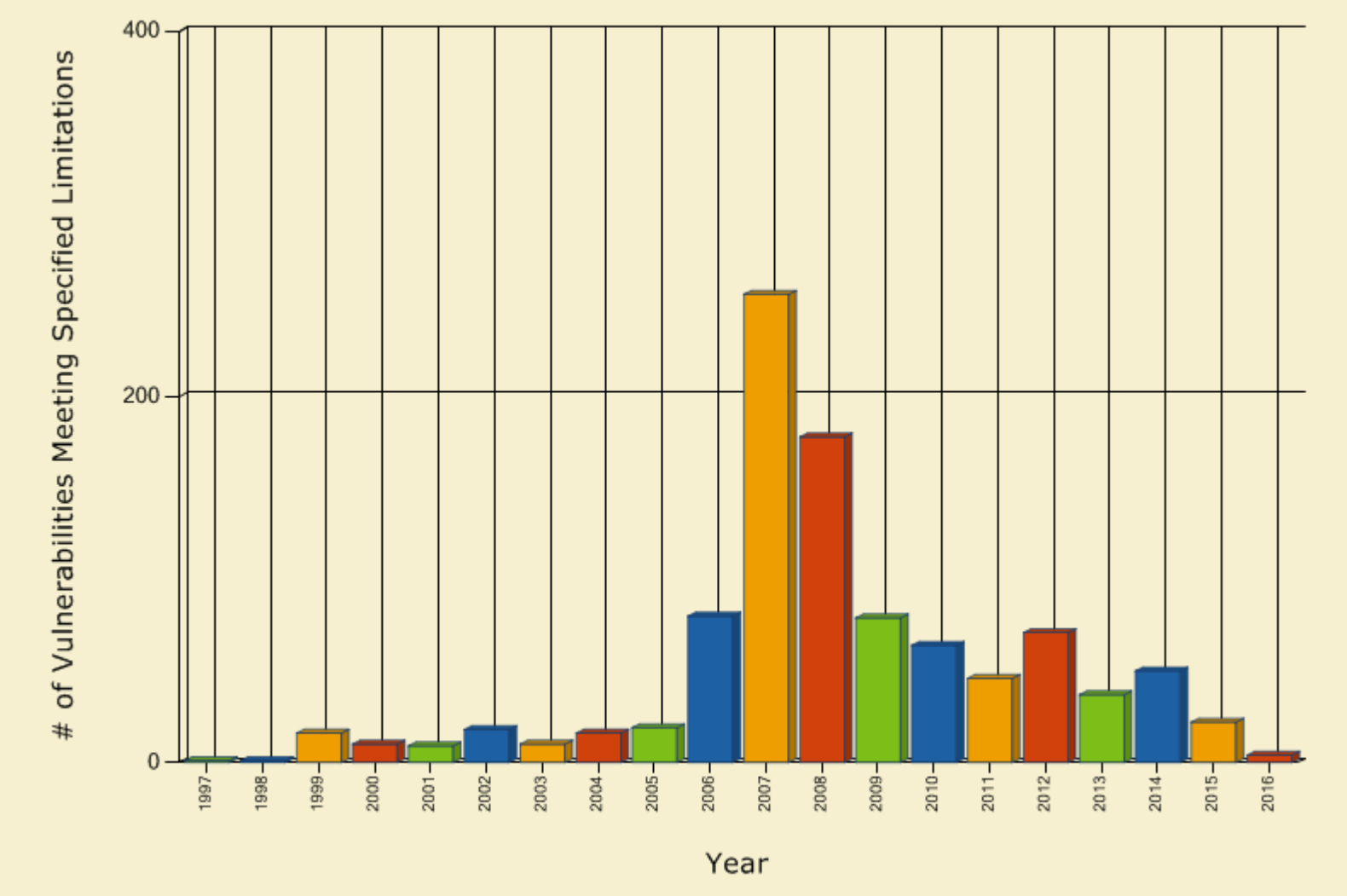


ACTIVEX控件

- ActiveX控件是1996年微软基于组件对象模型(COM)和对象连接与嵌入(OLE)技术提出的。COM技术是用来创建可重用的对象，其他的COM组件和程序可以通过COM对象提供的接口直接进行调用。
- ACTIVEX是微软WINDOWS IE特定支持的技术，很多复杂的B/S架构应用都采用了该技术，比如银行网银，证券公司客户端等。

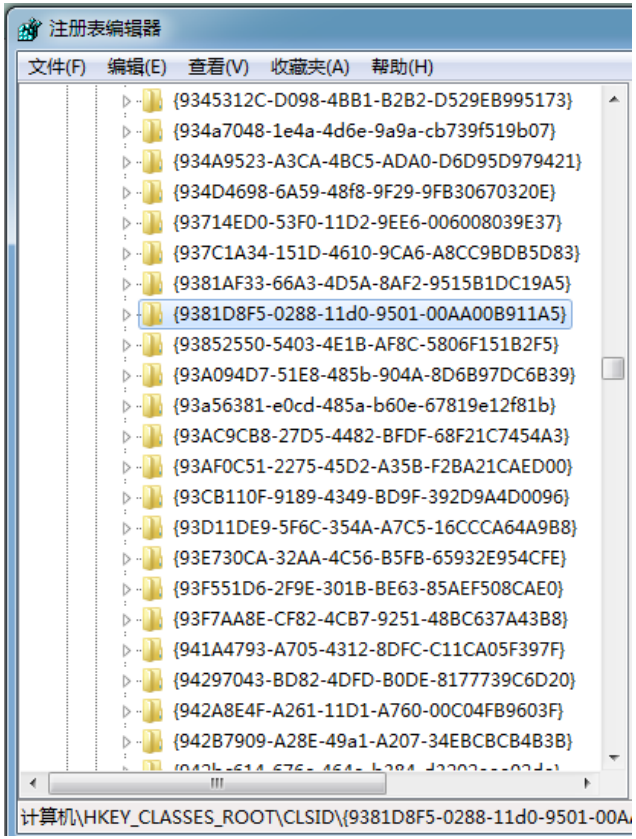
ACTIVEX控件

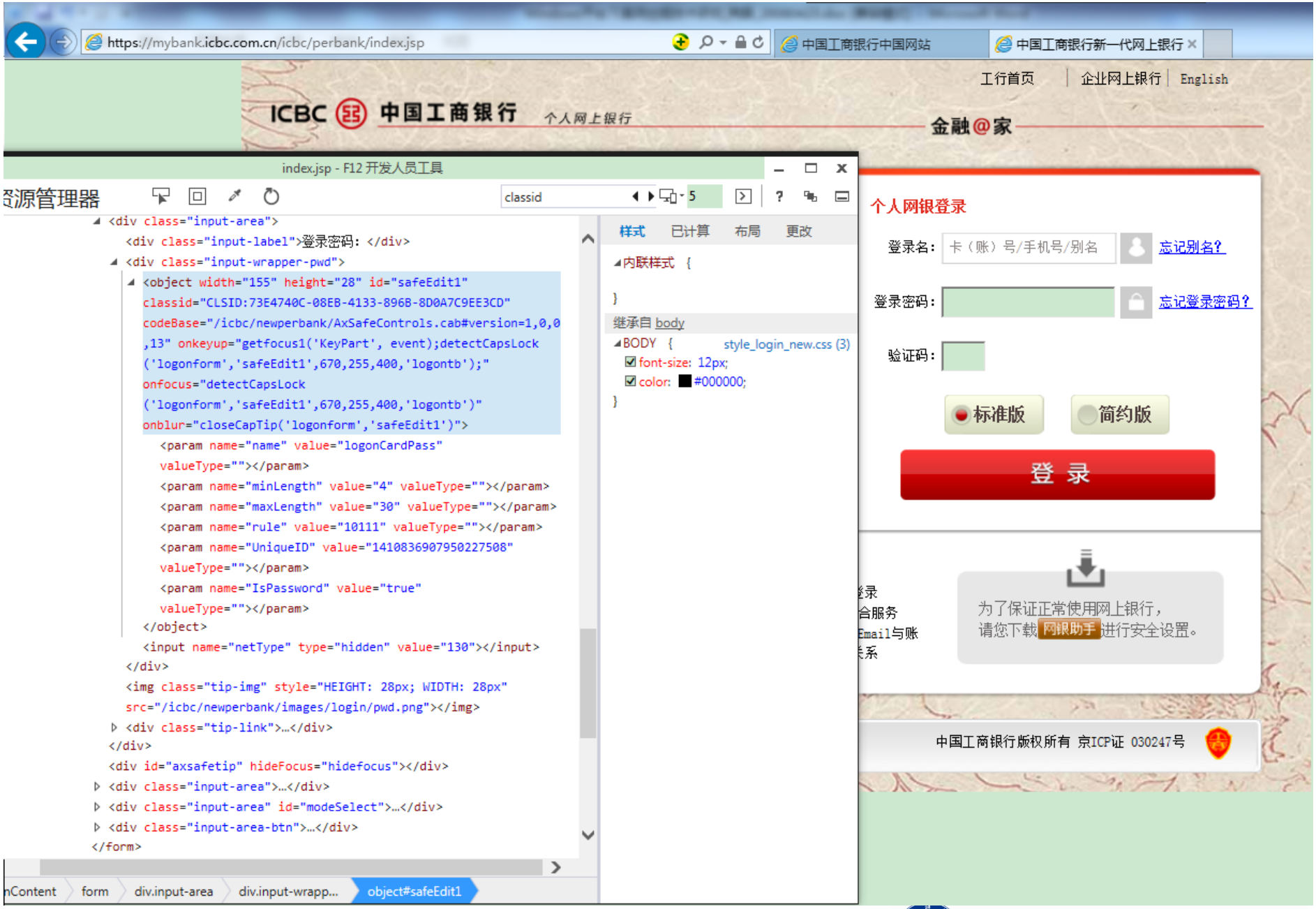
历史上**ACITVEX**出现的安全问题层出不穷，**ACTIVEX**的安全问题也是针对**IE**浏览器攻击最主要的模式之一。



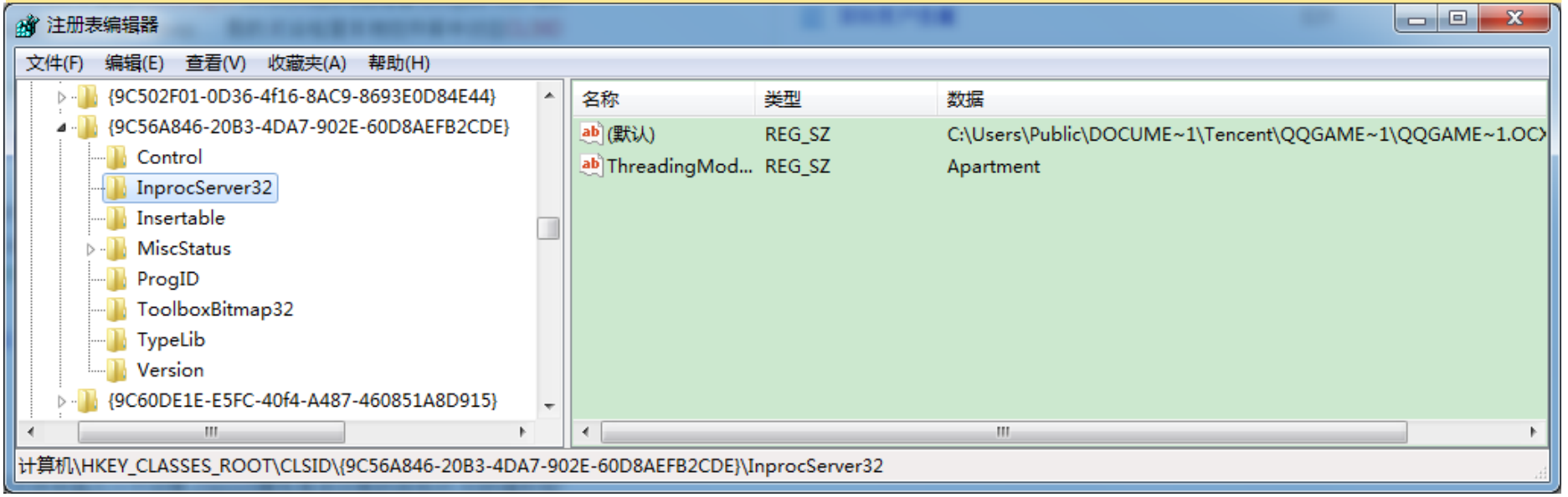
ACTIVEX控件

- 每个ActiveX控件都由16个字节（128位）的组件标识符CLSID来唯一对应，当需要调用该ActiveX控件时，可以通过该ActiveX控件的CLSID进行访问。
- 当ActiveX控件在系统中注册后，将在注册表的HKEY_CLASSES_ROOT\CLSID\下产生一个唯一的键值与其对应，在HKEY_CLASSES_ROOT\CLSID\下可以找到系统中所有注册的ActiveX控件。




https://mybank.icbc.com.cn/icbc/perbank/index.jsp
中国工商银行中国网站
中国工商银行新一代网上银行
工行首页
企业网上银行
English
ICBC 中国工商银行 个人网上银行
金融@家
index.jsp - F12 开发人员工具
classid
样式
已计算
布局
更改
个人网银登录
登录名：
卡（账）号/手机号/别名
忘记别名?
登录密码：
忘记登录密码?
验证码：
标准版
简约版
登 录
为了保证正常使用网上银行，
请您下载网银助手进行安全设置。
中国工商银行版权所有 京ICP证 030247号

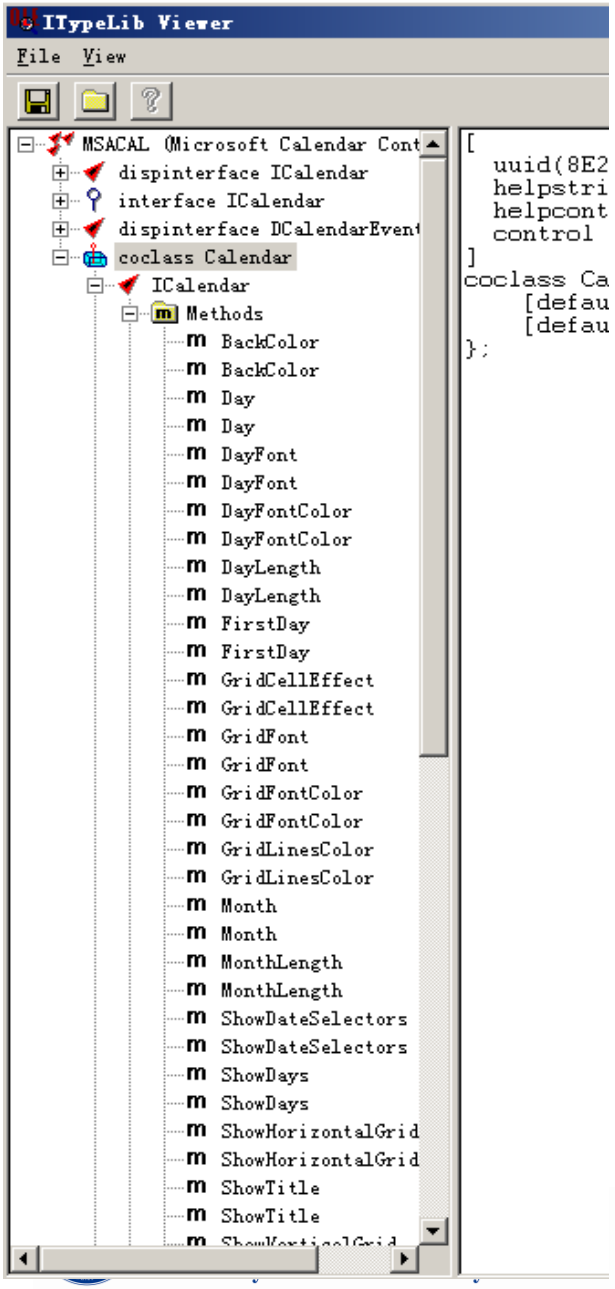
键值ImprocServer32的缺省值是组件所在的DLL或者OCX文件名称，当该ActiveX控件被调用时，该DLL或者OCX文件被加载



ProgID的命名通常采用如下格式：<Program>.<Component>.<Version>

ACTIVEX控件

- 每个ActiveX控件都实现了不同的接口，并提供方法供外部组件或程序调用。
- 可以通过VC 6.0提供的OleView工具查看ActiveX控件接口、方法和属性。



基于组件的软件工程

基于组件的软件工程

- 以组件作为构成软件的基本单元
- 组件 = 外部接口 + 内部实现

常见的组件模型

- 微软公司的 COM/ActiveX
- Oracle 公司的 EJB
- OMG 组织的 CORBA

ACTIVEX

- 基于 COM 组件模型
- 每个 ActiveX 控件都是一个相对独立的代码单元
  - 通过一组接口对外服务
- 应用程序通过“嵌入”控件来调用控件提供的服务
  - 该应用程序被称为“宿主”或“容器”

最常见的 ActiveX 容器：Web 浏览器

- 通过控件来实现浏览器功能的扩展

ACTIVEX

1. 存在广泛
   - 大量软件使用 ActiveX 来构建功能模块
   - 开发者往往忽视代码安全性
2. 便于利用
   - 可以通过任意一个站点的 Web 脚本来触发
   - 存在成熟的利用模式即 Heap-Spraying
3. 危害严重
   - 通常使用不安全的编程语言来编写
   - 控件运行于浏览器进程的上下文中

如何发现ActiveX的安全问题？

浏览器

如何有效对浏览器进行模糊测试？

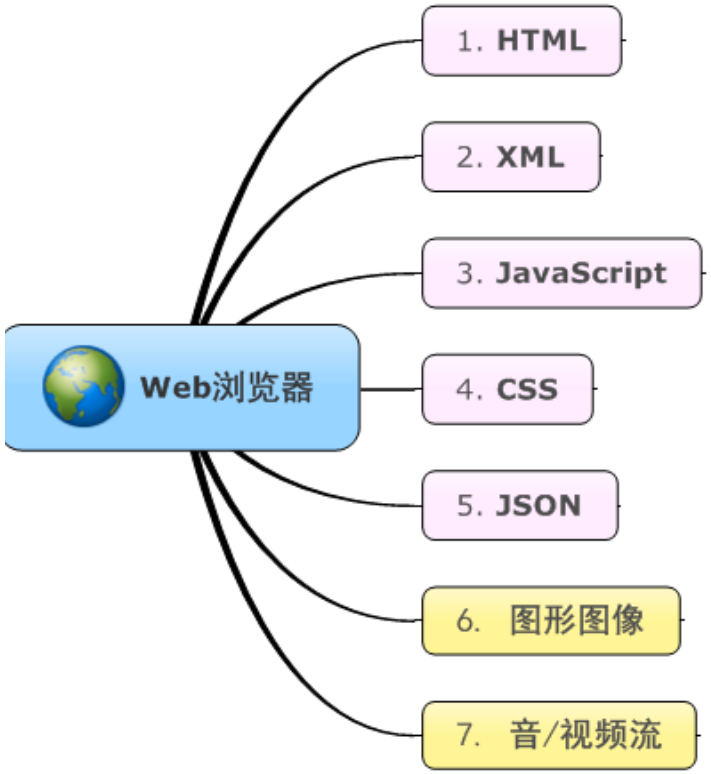
核心和关键：

- 分析浏览器的输入数据格式
- 根据输入格式特点生成半有效测试数据

问题分析

浏览器输入格式分析

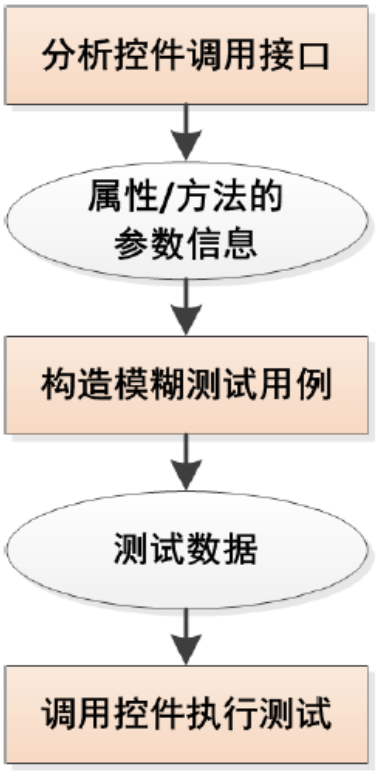
- 主体：高度结构化输入
  - 直接可读的文本
  - 需满足特定形式文法的约束
- 例子：JavaScript
  - 可读的文本代码
  - 代码格式规范：ECMA-262 标准



ACTIVEX 控件漏洞挖掘方法

分析现有的漏洞挖掘技术，总结出一种基于模糊测试的 ActiveX 漏洞挖掘思路：

1. 分析控件调用接口
   - 得到接口中属性和方法的参数信息
2. 构造模糊测试用例
   - 根据参数类型构造半有效测试数据
3. 调用控件执行测试
   - 使用调试器监控并记录异常



ACTIVEX：接口分析

ActiveX 使用 MIDL 语言定义接口

- 两种交互方式：属性和方法

发布控件时，MIDL 接口描述被编译为二进制类型库

- 解析类型库即可获得属性和方法的参数信息

```
dispinterface _DSecClient {
    properties:
        [id(0x00000001)] BSTR EncryptedData;
        [id(0x00000002)] BSTR PemSignature;
        [id(0x00000003)] BSTR PemCert;
        [id(0x00000004)] BSTR RandomString;
        [id(0x00000005)] BSTR PemHMAC;
    methods:
        [id(0x00000006)]
        long EncDataWithCert(BSTR PlainText,
            BSTR PemCert);
        [id(0x00000007)]
        long GenSignature(BSTR PlainText,
            BSTR PrivateKeyPath);
        [id(0x00000008)]
        long GenRandomString();
        [id(0x00000009)]
        long DisplayCertInfo(BSTR PemCert);
        [id(0x0000000a)]
        long GenHMAC(BSTR PlainText, BSTR Key);
        [id(0xfffffdd8)]
        void AboutBox();
};
```

ACTIVEX：测试用例构造

模糊测试用例的形式

- 畸形函数参数

模糊测试用例的构造策略

- 依据：参数的数据类型及漏洞类型

- 字符串型参数
  - 堆/栈缓冲区溢出漏洞
  - 格式化字符串漏洞
- 整形参数
  - 整数溢出漏洞

ACTIVEX：测试用例构造

堆/栈缓冲区溢出漏洞

成因：

- 程序未对输入字符串进行边界检查
- 数据溢出，覆盖关键的内存控制结构

数据构造方式：

- 使用不同长度的超长字符串作为输入参数
- 非线性的字符串长度选择策略
  - $(0, 8192]$：从长度为 32 的串开始，以 32 为增量依次增加
  - $(8192, \infty)$：只选择 2^{14}、2^{15}、2^{16}、2^{17}、2^{18} 这几种情况

ACTIVEX：测试用例构造

整数溢出漏洞

成因：

- 算术运算溢出： $0xFFFFFFFF + 1 = 0x00000000$
- 截断溢出： $0x000AAAAA \rightarrow 0xFFFFAAAA$

数据构造方式：

- 以各种位宽的有符号/无符号整数的边界为基数
  - 32 位整数：0x100000000、0x80000000、0x40000000、0x20000000 及 0x10000000
- 在这些值的基础上进行加 1、减 1 和乘以 -1 等变异操作

ACTIVEX：测试用例构造

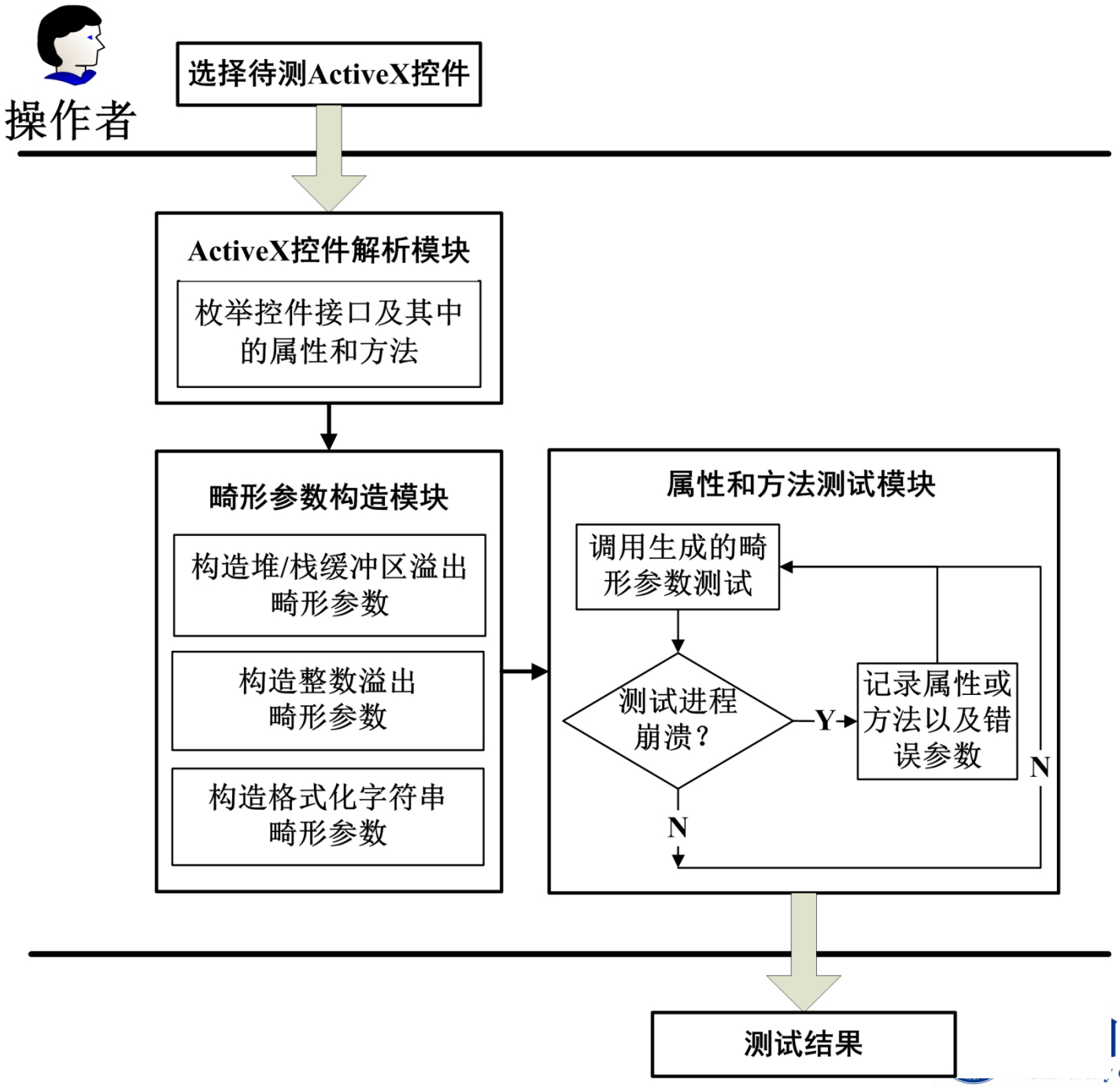
格式化字符串漏洞

成因：

- 格式化串引发的缓冲区溢出
  - 格式控制符 + 宽度定义
- 格式化串引发的内存数据改写
  - 格式控制符%d

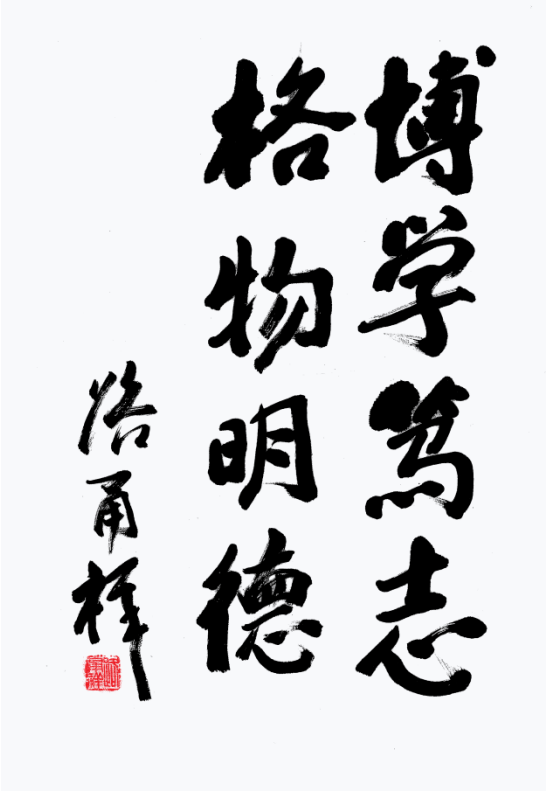
数据构造方式：

- 含有大量%99s 的长字符串
- 含有大量%d 的长字符串



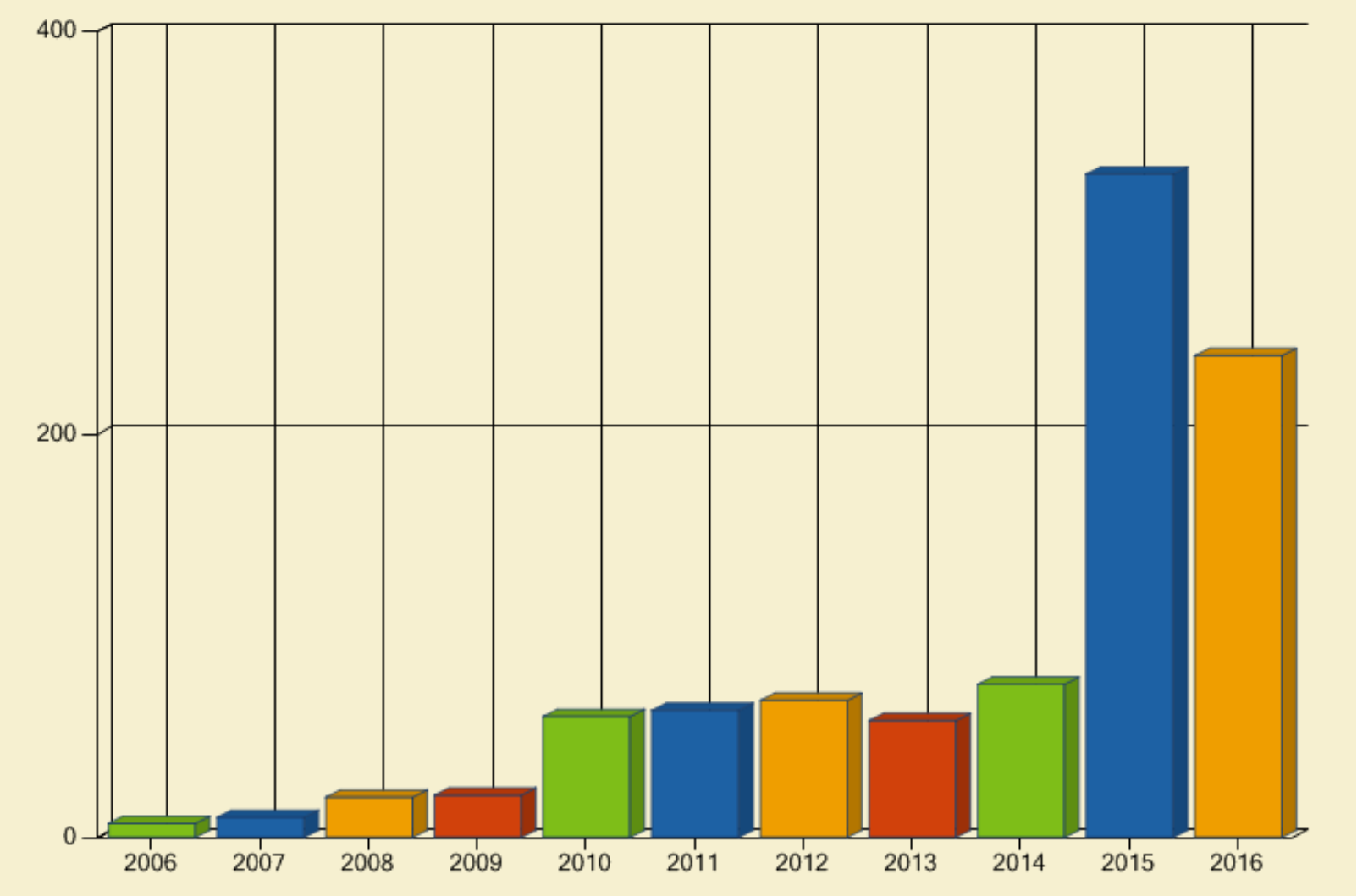
本章大纲

- 网页木马
- ActiveX
- Adobe Flash Player
- 浏览器安全机制




富文本文档
SWF
浏览器
SWF
SWF
富文本文档

ADOBE FLASH PLAYER漏洞数目



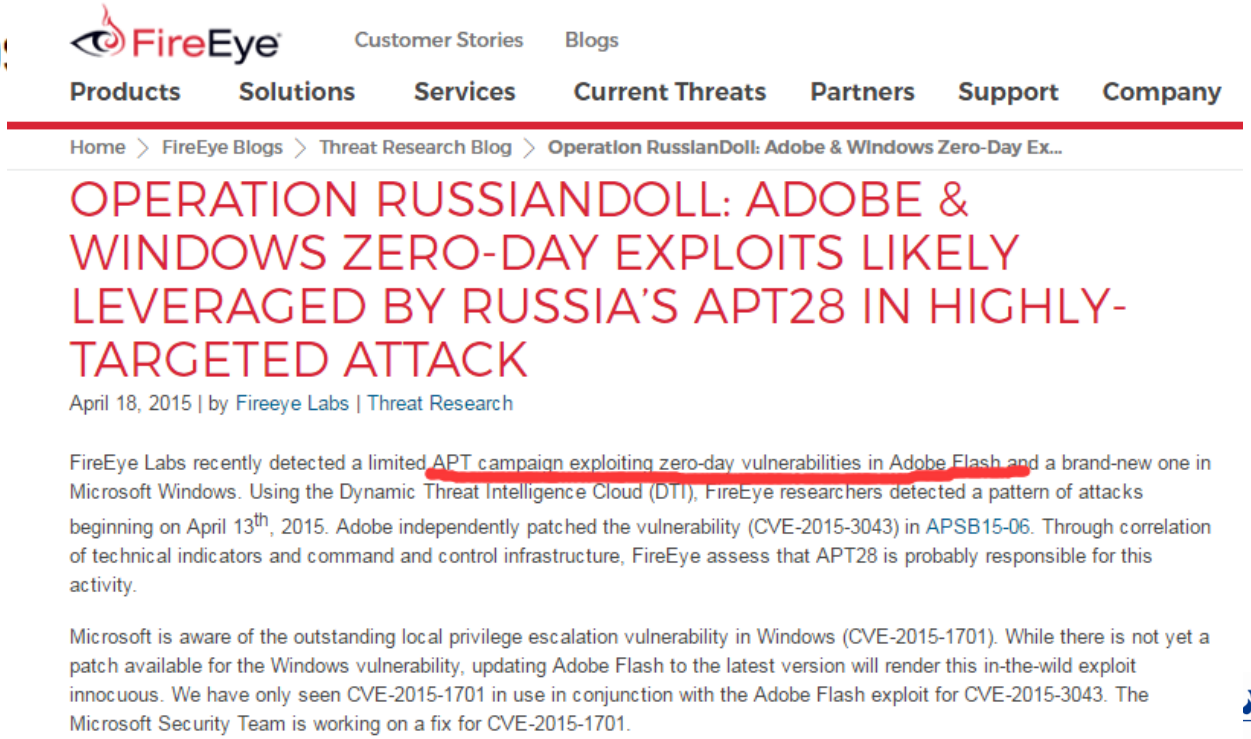
Pawn Storm used a new Flash Zero-Day in attacks on the NATO & the While House

October 15, 2015 By Pierluigi Paganini

G+1 10

My Page

Researchers at Trend Micro discovered a new Adobe Flash Zero-Day used in Pawn Storm Campaign Targeting Foreign Affairs Ministries acro



FireEye Customer Stories Blogs

Products Solutions Services Current Threats Partners Support Company

Home > FireEye Blogs > Threat Research Blog > Operation RussianDoll: Adobe & Windows Zero-Day Ex...

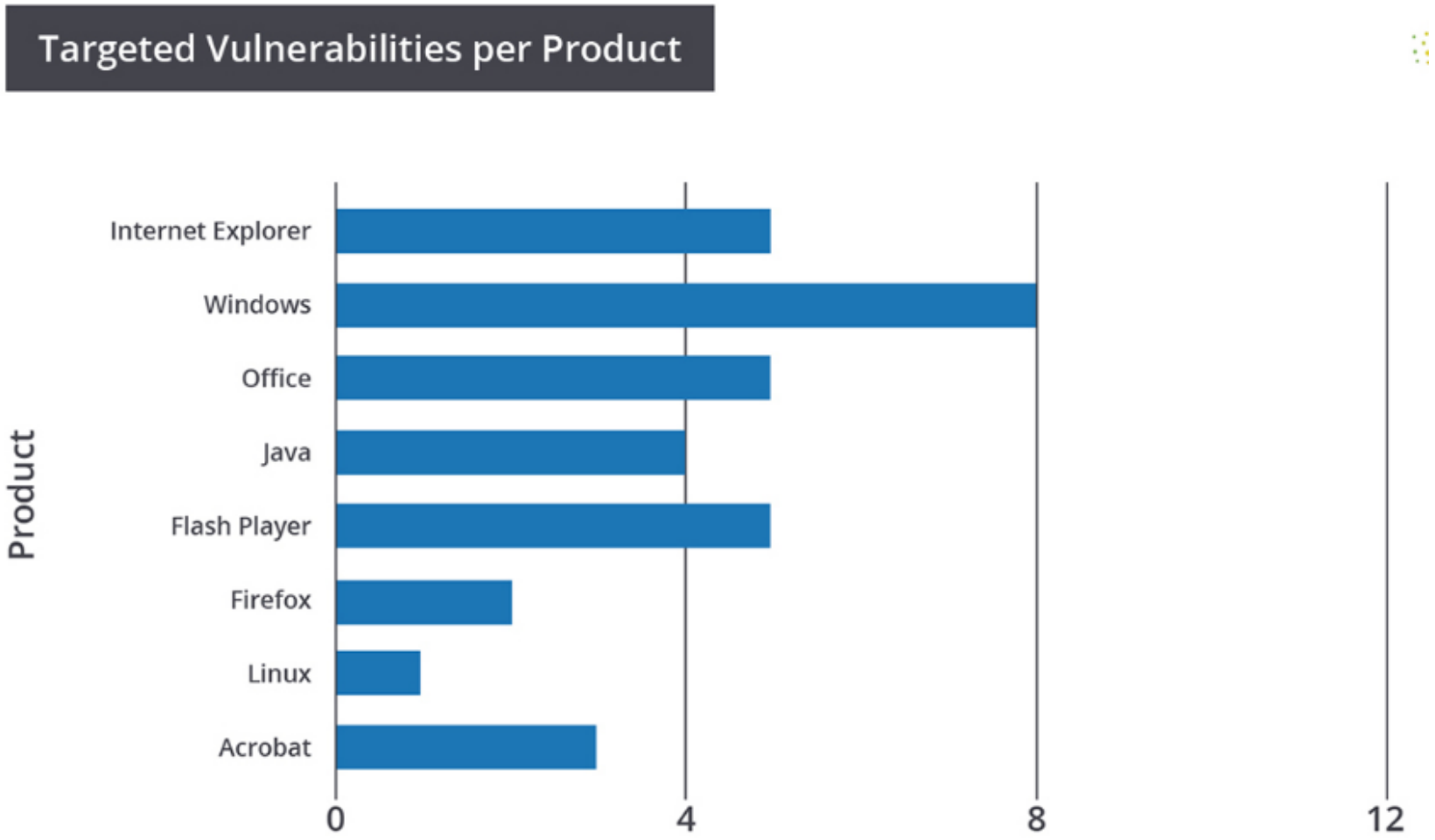
OPERATION RUSSIANDOLL: ADOBE & WINDOWS ZERO-DAY EXPLOITS LIKELY LEVERAGED BY RUSSIA'S APT28 IN HIGHLY-TARGETED ATTACK

April 18, 2015 | by Fireeye Labs | Threat Research

FireEye Labs recently detected a limited APT campaign exploiting zero-day vulnerabilities in Adobe Flash and a brand-new one in Microsoft Windows. Using the Dynamic Threat Intelligence Cloud (DTI), FireEye researchers detected a pattern of attacks beginning on April 13th, 2015. Adobe independently patched the vulnerability (CVE-2015-3043) in APSB15-06. Through correlation of technical indicators and command and control infrastructure, FireEye assess that APT28 is probably responsible for this activity.

Microsoft is aware of the outstanding local privilege escalation vulnerability in Windows (CVE-2015-1701). While there is not yet a patch available for the Windows vulnerability, updating Adobe Flash to the latest version will render this in-the-wild exploit innocuous. We have only seen CVE-2015-1701 in use in conjunction with the Adobe Flash exploit for CVE-2015-3043. The Microsoft Security Team is working on a fix for CVE-2015-1701.

THE FOLLOWING PRODUCTS WERE REGULARLY TARGETED BY THE FOUR RUSSIAN GROUPS



Exploited Vulnerabilities Linked to Russian APTs

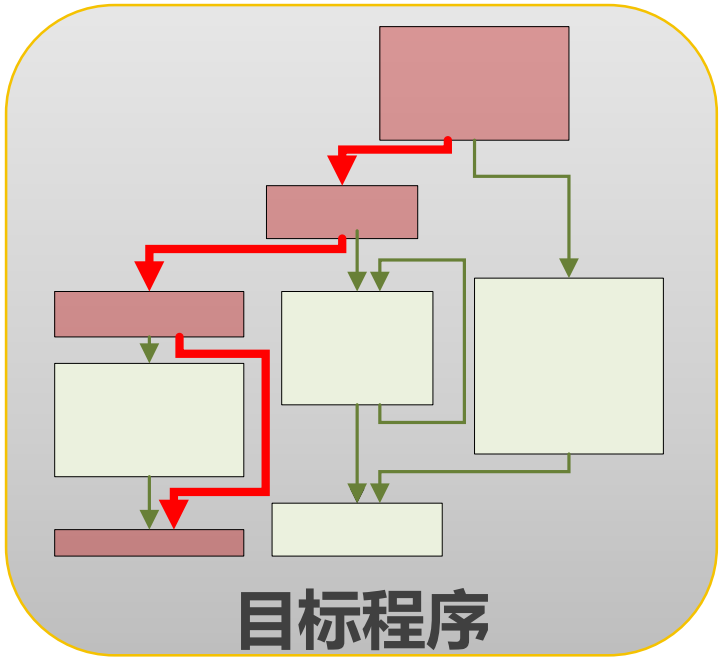
Recorded Future

	Vulnerability	Company	Product	Specific APT	Exploit Kit Presence	Exploit Available
1	CVE-2016-4117	Adobe	Flash Player	APT28	Yes	No
2	CVE-2016-0728	Community	Linux	APT28	No	Yes
3	CVE-2015-7645	Adobe	Flash Player	APT28	Yes	Yes
4	CVE-2015-5119	Adobe	Flash Player	APT28	Yes	Yes
5	CVE-2015-4902	Oracle	Java	APT28	No	No
6	CVE-2015-3043	Adobe	Flash Player	APT28	No	Yes
7	CVE-2015-2590	Oracle	Java	APT28	No	No
8	CVE-2015-2424	Microsoft	Office	APT28	No	No
9	CVE-2015-2387	Microsoft	Windows	APT28	No	No
10	CVE-2015-1701	Microsoft	Windows	APT28	No	Yes
11	CVE-2015-1641	Microsoft	Office	APT28	Yes	Yes
12	CVE-2014-6332	Microsoft	Internet Explorer	APT28	Yes	Yes
13	CVE-2014-4114	Microsoft	Windows	APT28	Yes	Yes
14	CVE-2014-4076	Microsoft	Windows	APT28	No	Yes
15	CVE-2014-3897	Microsoft	Internet Explorer	APT28	Yes	No
16	CVE-2014-1776	Microsoft	Internet Explorer	APT28	Yes	No
17	CVE-2014-1761	Microsoft	Office	APT29	Yes	Yes
18	CVE-2014-1511	Mozilla	Firefox	APT28	No	Yes
19	CVE-2014-1510	Mozilla	Firefox	APT28	No	Yes
20	CVE-2013-5065	Microsoft	Windows	Turla	No	Yes
21	CVE-2013-3346	Adobe	Acrobat	Turla	No	Yes
22	CVE-2013-2465	Oracle	Java	Energetic Bear	Yes	Yes
23	CVE-2013-1347	Microsoft	Internet Explorer	APT28, Energetic Bear	Yes	Yes
24	CVE-2013-0641	Adobe	Acrobat	APT29	No	No
25	CVE-2013-0640	Adobe	Acrobat	APT29	No	Yes
26	CVE-2012-1723	Oracle	Java	Turla, Energetic Bear	Yes	Yes
27	CVE-2012-0158	Microsoft	Office	APT28	Yes	Yes
28	CVE-2011-0611	Adobe	Flash Player	Energetic Bear	Yes	Yes
29	CVE-2010-4398	Microsoft	Windows	APT29	No	Yes
30	CVE-2010-3333	Microsoft	Office	APT28	No	Yes
31	CVE-2010-0232	Microsoft	Windows	APT29, Turla	No	Yes
32	CVE-2009-1123	Microsoft	Windows	Turla	No	No

如何发现Adobe Flash Player的安全问题？

FUZZING模糊测试

- **产生随机输入，送入目标程序**
  - 覆盖尽可能多的执行路径
  - 发现程序缺陷

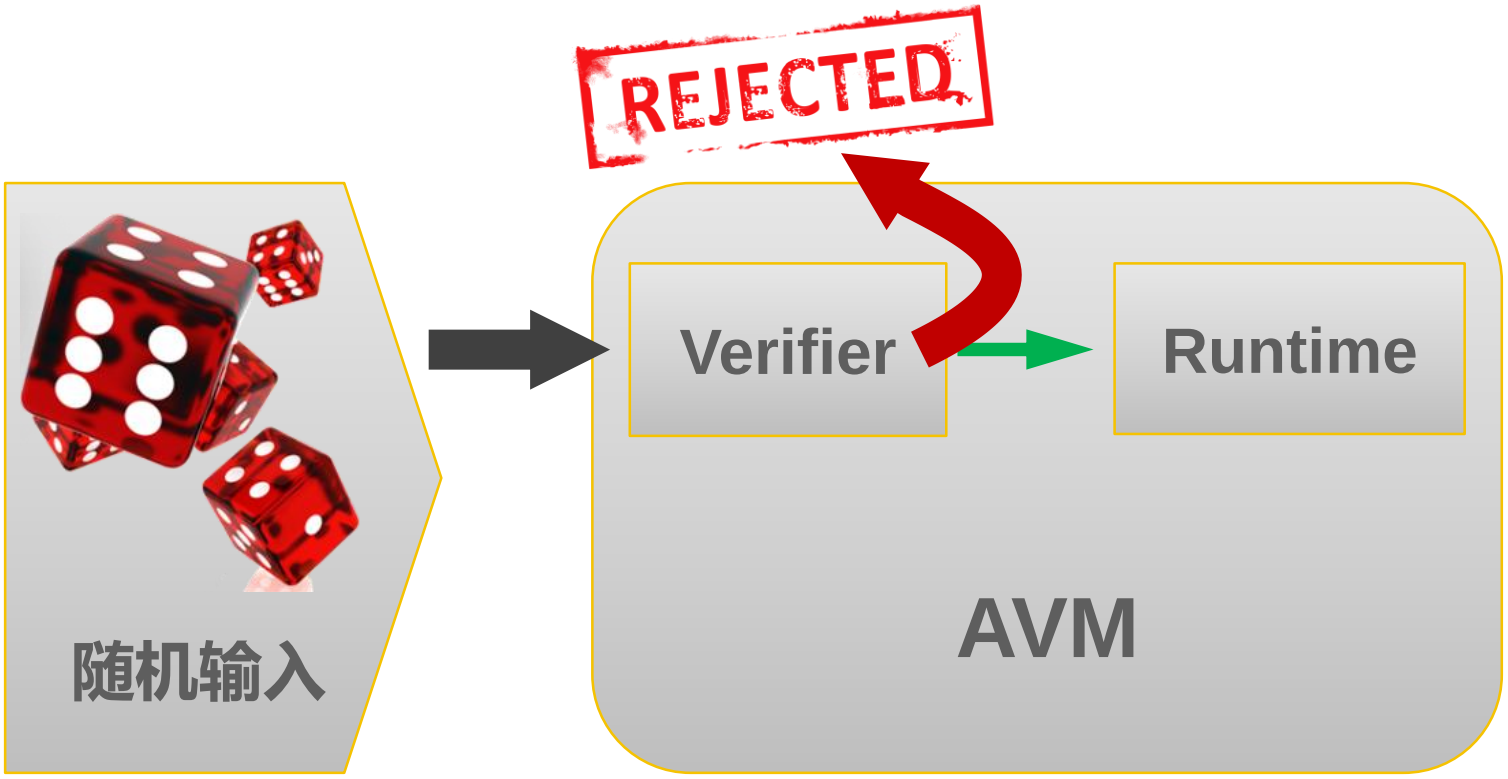


FUZZING模糊测试

ActionScript VM（AVM）

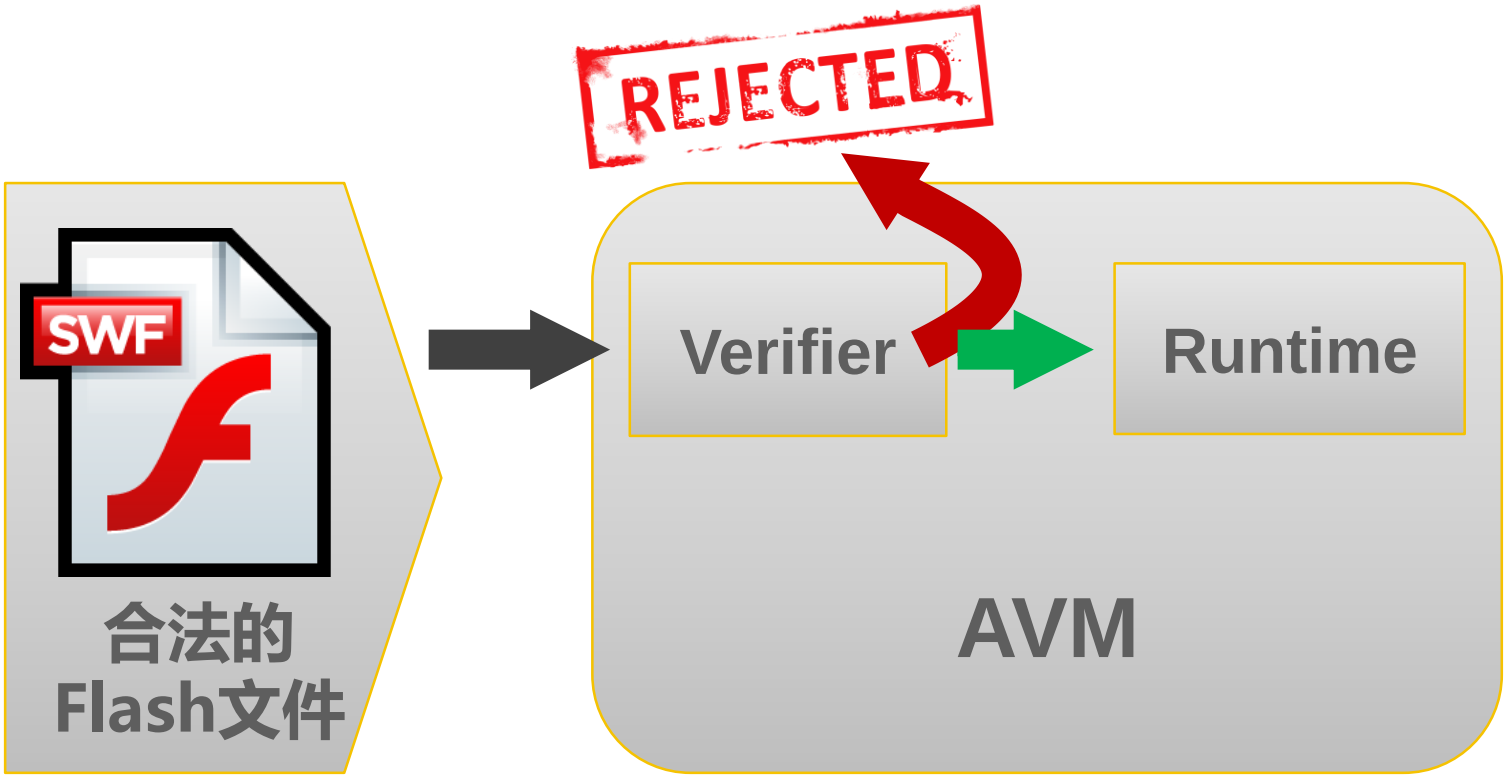
- Java Applet、JavaScript、Flash ActionScript广泛存在于时下流行的浏览器当中。浏览器会调用相应的脚本引擎/虚拟机（Virtual Machine，VM）解释执行当中的代码。执行脚本代码时需要创建一个VM实例，一旦脚本代码执行完毕，VM实例也会随之销毁。从执行代码的方式上可以将VM分为两种：
- 第一种VM的输入是脚本语言源代码，逐行逐句的解释执行，比如JavaScript；第二种VM的输入是字节码，开发人员需要使用编译器将源代码转化为包含有字节码二进制文件，才能送入到VM中执行，比如ActionScript。
- 站在测试者的角度来看，第二种VM执行代码的过程较第一种复杂。

测试AVM



很难测试到AVM代码解析模块

测试AVM

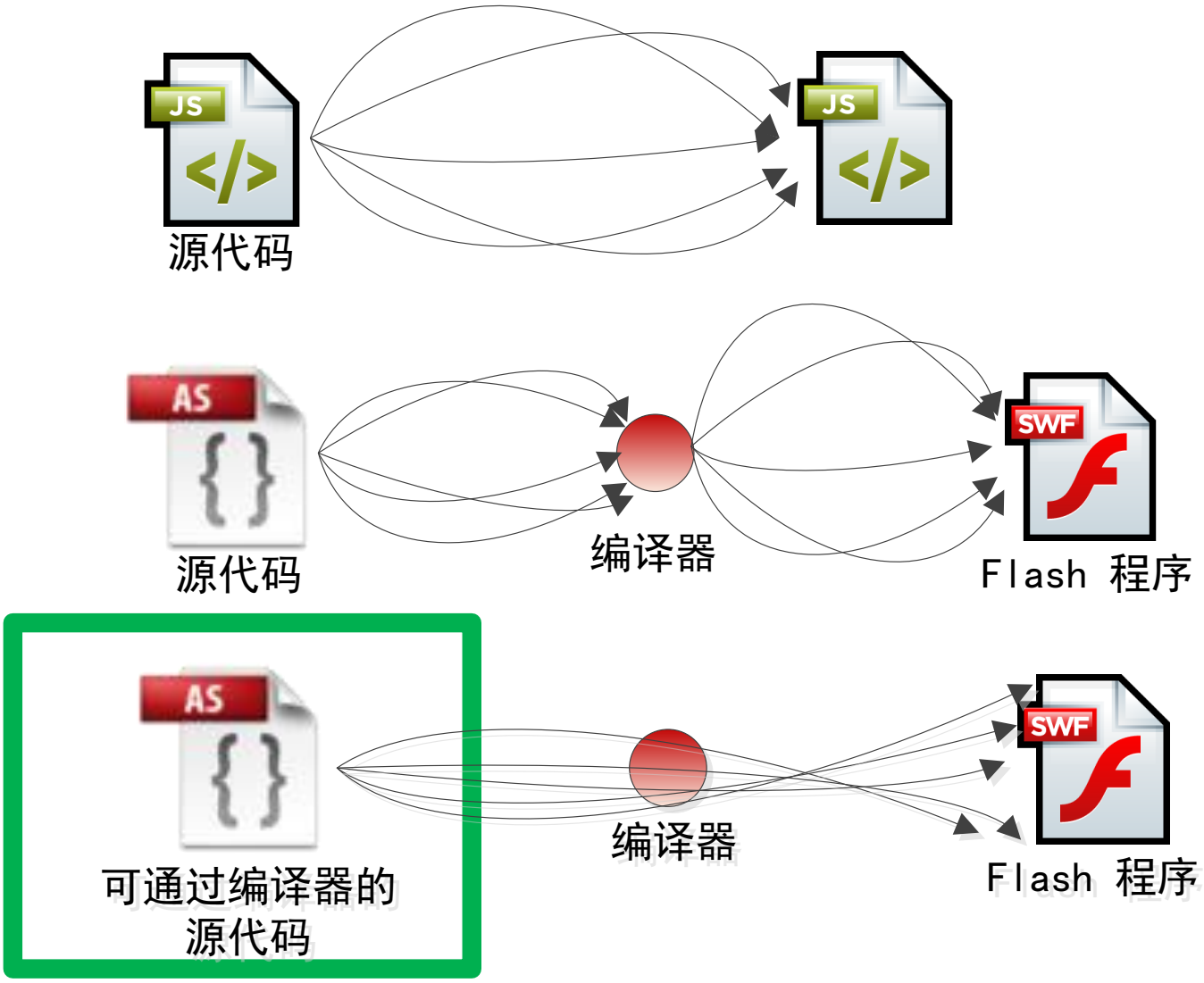


可以测试到AVM代码解析模块

产生合法FLASH


变异已有的Flash中的字节码
Dumb Fuzz，高度依赖于种子文件质量
直接生成字节码填充到Flash
复杂语法的语段难以建构
产生ActionScript源代码，编译得到合法Flash

产生合法FLASH


JS
JS
源代码
AS
源代码
编译器
SWF
Flash 程序
AS
可通过编译器的
源代码
编译器
SWF
Flash 程序

问题描述

- 在模糊测试中，测试数据的生成是非常关键的一环，因为测试数据的类型和结构直接决定了可测试的程序的范围，同时数据的质量也对测试的有效性有较大的影响。
- 早期的模糊测试研究通常使用随机字符发生器来产生测试数据。
- 虽然这种方式发现了大量的程序缺陷，但也存在明显的局限性：许多程序在真正处理输入数据之前都会对数据进行一定的验证，而随机生成的字节序列很难通过校验，从而无法对核心的数据处理代码进行有效测试。

FUZZ

- Fuzz是世界上第一款Fuzzer，产生于威斯康星大学(wisc)，Fuzzing也因此工具而得名。
- 该工具利用Windows的消息机制向窗口随机的发送数据，从而试图测试应用程序的稳定性。

```
Usage: Fuzz [-ws] [-wp] [-v] [-a application command line] [-i pid] [-n NumMessa
ges] [-c] [-l] [-e seed]

Requirements:
        One of -ws, -wp, -v
        One of -a, -i

Options:
        -ws     Random Win32 message test using SendMessage API function
        -wp     Random Win32 message test using PostMessage API function
        -v      Random mouse/keyboard event test
        -a      Launch and test given application
        -i      Test running application with given pid
        -n      Number of messages to send (infinite if not specified)
        -c      Send random WM_COMMAND messages (for pure random test only)
        -l      Send null parameters in messages (for pure random test only)
        -e      Set random seed to value
```

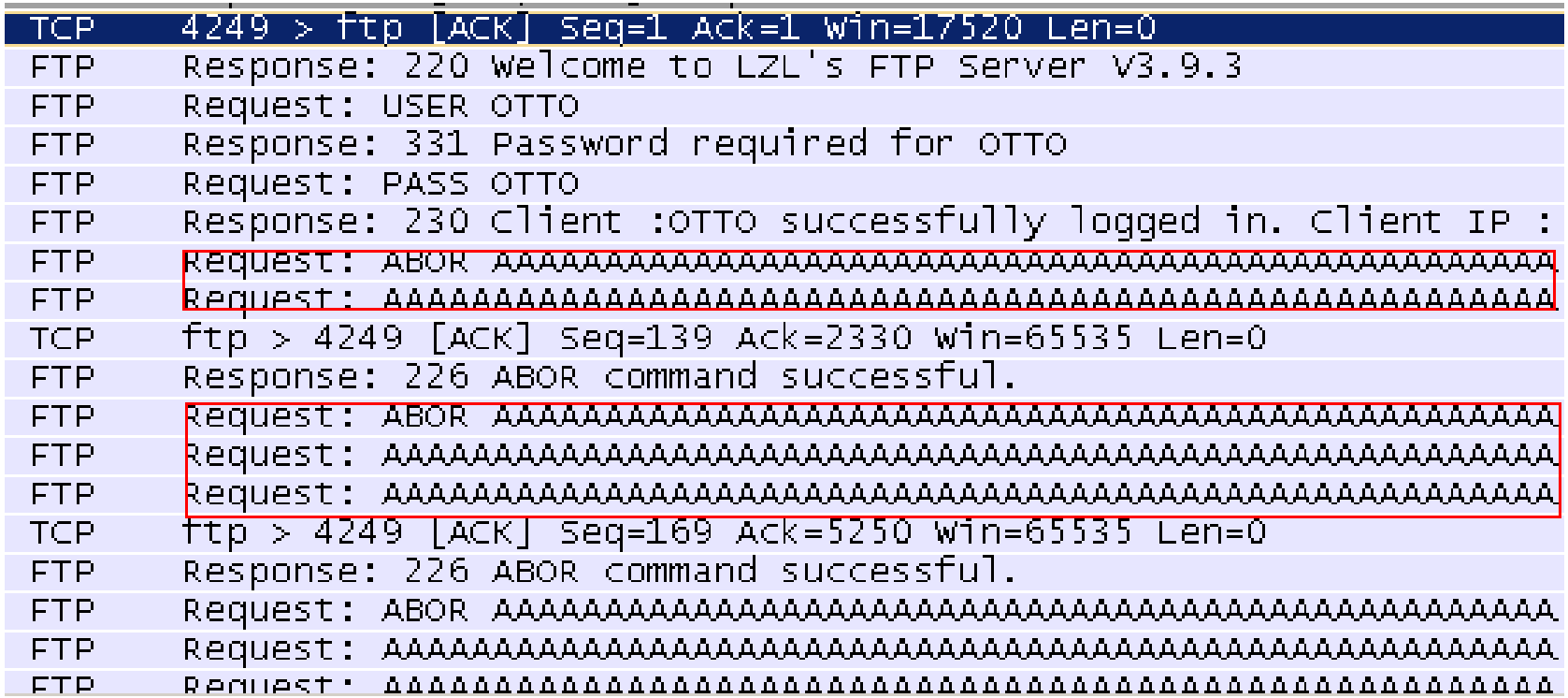
基于块的数据模型

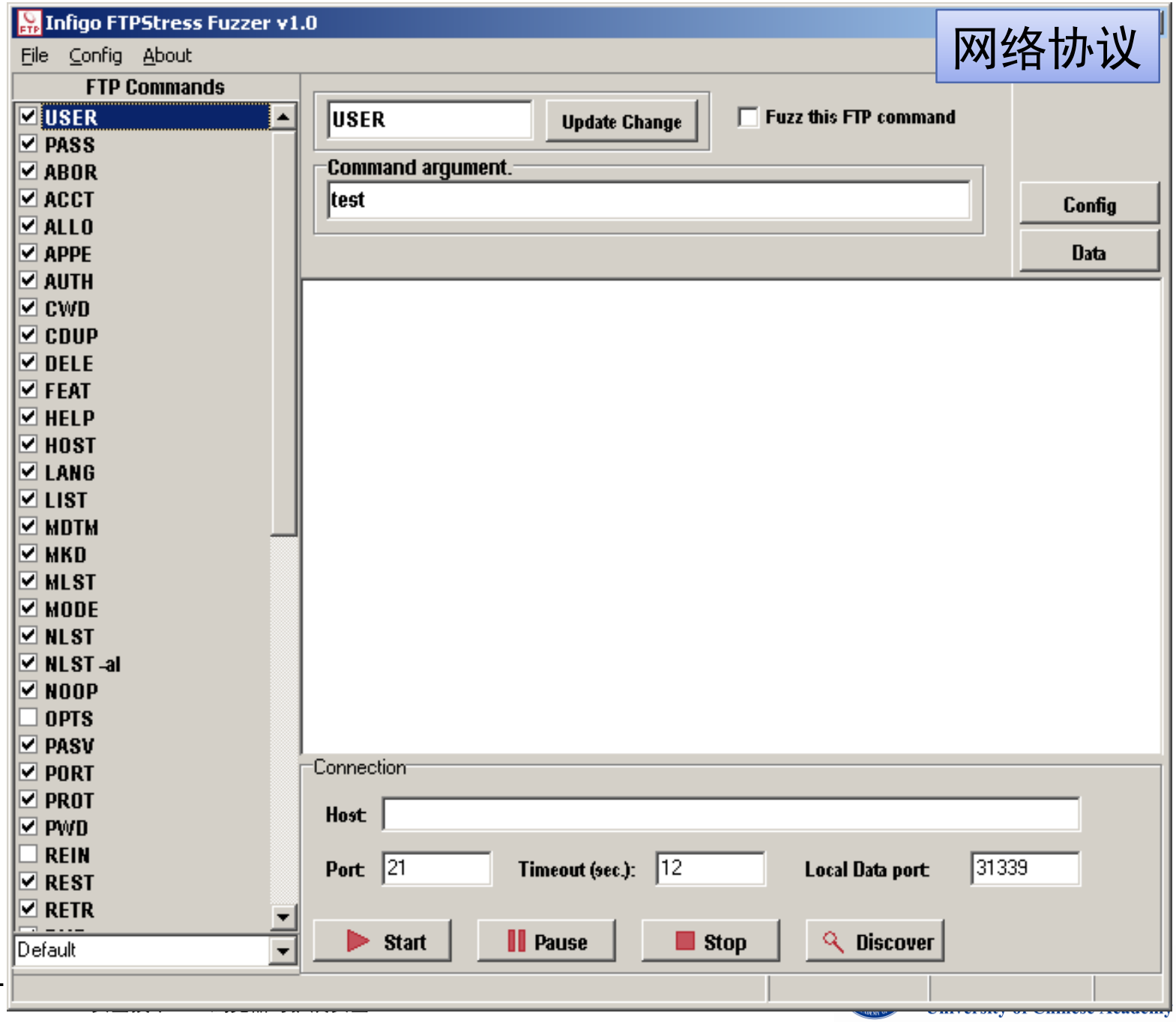
- 现有的模糊测试工具和框架通常采用一种基于块的方式来对输入数据进行建模。
- 在这种建模方式中，程序输入被划分成一系列数据块，同时块与块之间的约束关系通过元数据来表达。
- 这种表示方式非常适合于描述二进制输入，例如网络报文以及二进制文件。这类输入数据的内部结构较为简单，可以直接映射到数据模型中的数据块及其约束关系。

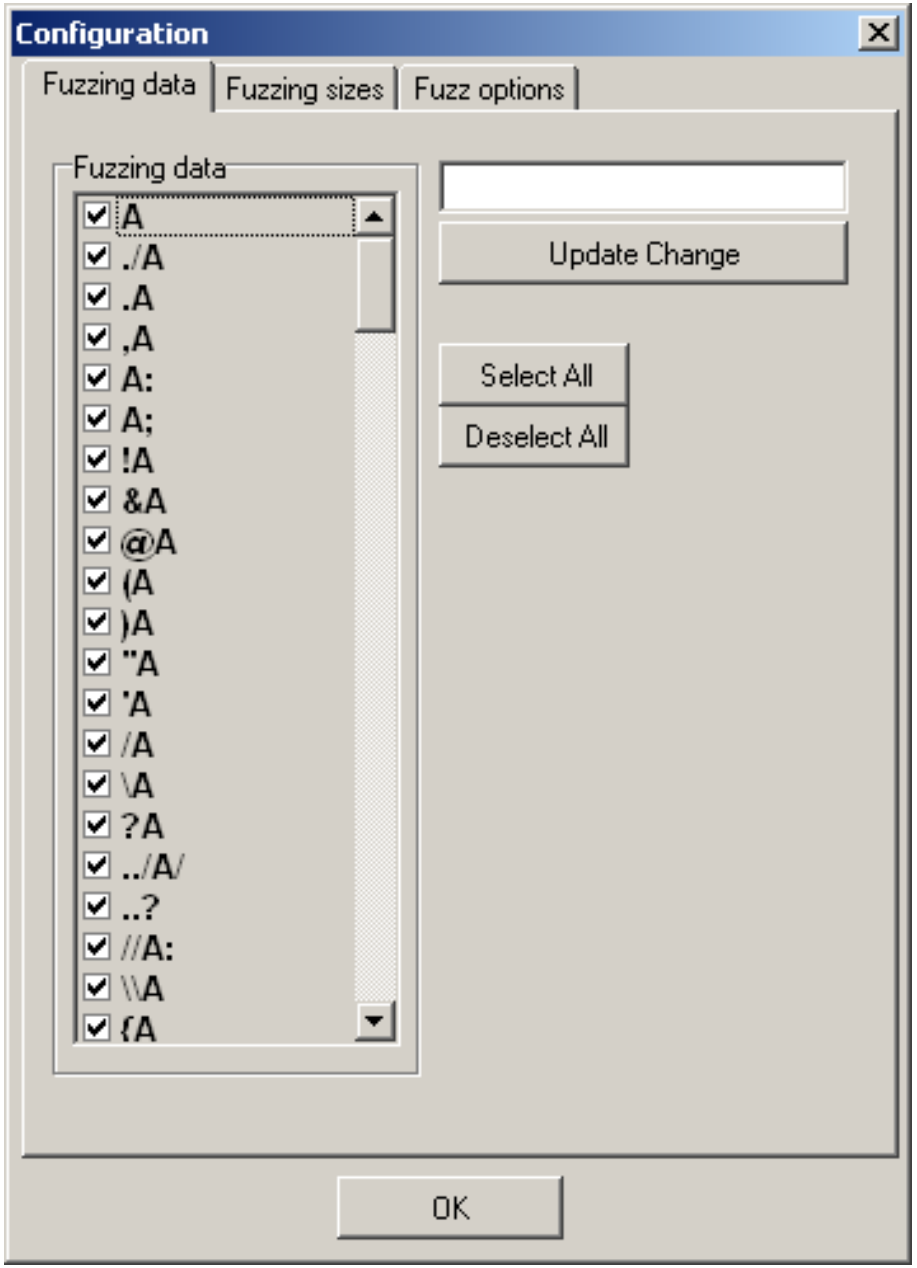
基于块的数据模型

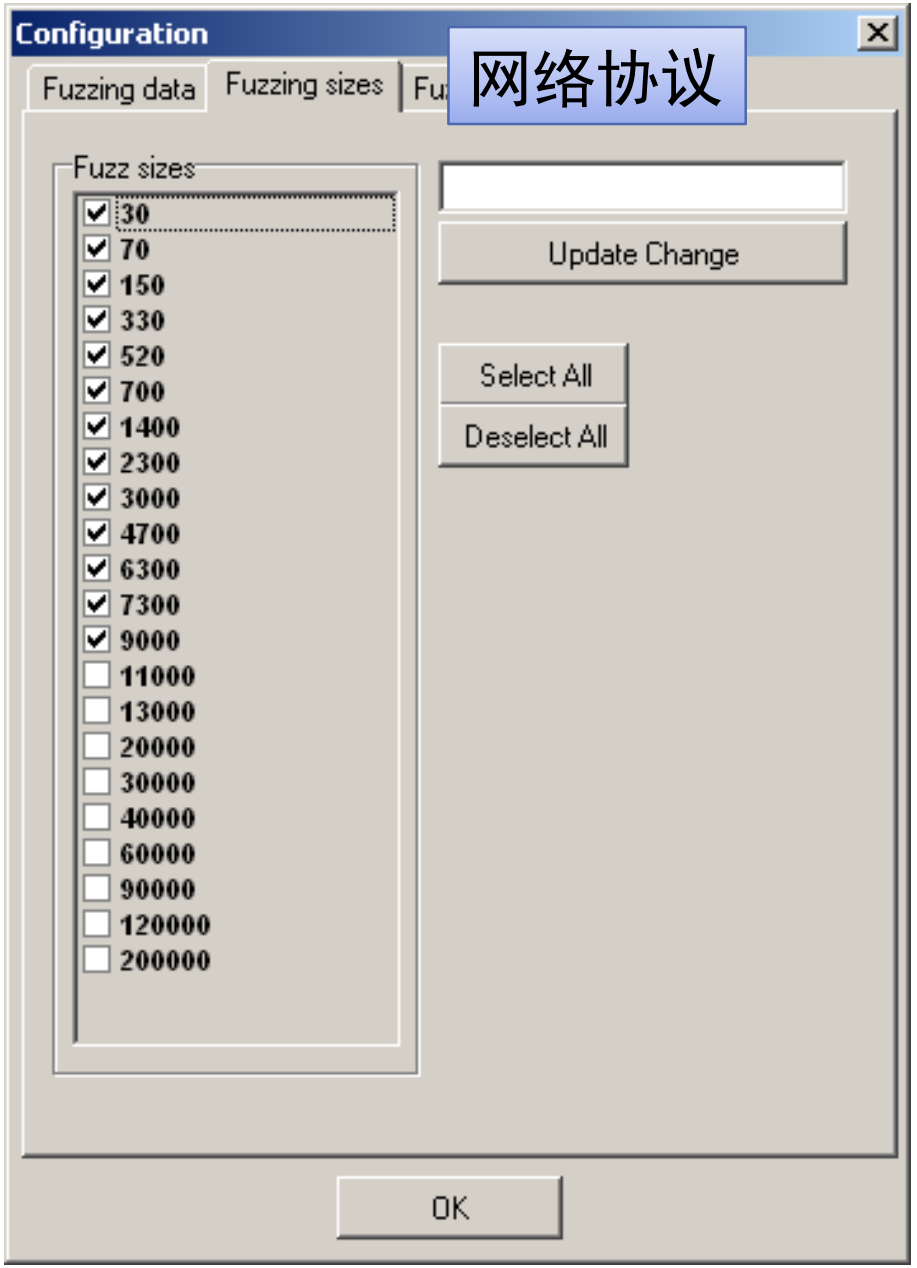
网络协议

- FTP命令+特殊数据
- ABOR+AAAAAAAAAAA…





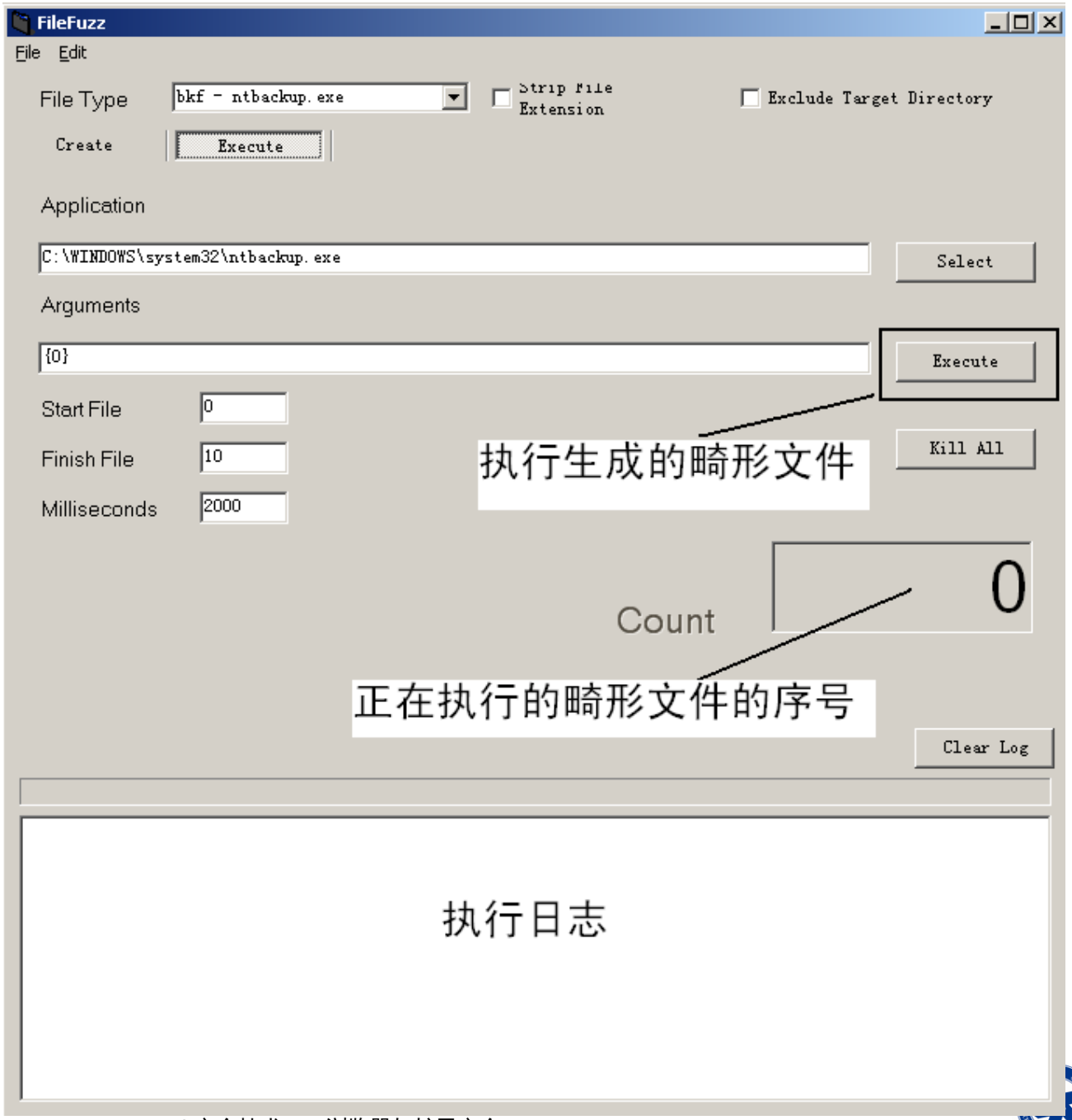

Configuration
Fuzzing data
Fuzzing sizes
Fuzz options
Fuzzing data
A
./A
.A
,A
A:
A;
!A
&A
@A
(A
)A
"A
'A
/A
\A
?A
../A/
..?
//A:
\\A
{A
Update Change
Select All
Deselect All
OK


Configuration
Fuzzing data
Fuzzing sizes
网络协议
Fuzz sizes
30
70
150
330
520
700
1400
2300
3000
4700
6300
7300
9000
11000
13000
20000
30000
40000
60000
90000
120000
200000
Update Change
Select All
Deselect All
OK


FileFuzz
File Edit
File Type
bkf - ntbackup.exe
Strip File Extension...
Exclude Target Directory
Create
Execute
选择要Fuzzing的文件格式
Source File
C:\Program Files\FileFuzz\Attack\test.bkf
源文件
Select
Target Directory
c:\fuzz\bkf\
畸形文件的存放目录
Select
Create
Target
Byte(s) to Overwrite
00
FF
X
4
怎样修改源文件
Scope
All Bytes
Range
Start
0
Finish
10
Depth
Location
0
Match
Find
Replace
A
X
100
X
10
在源文件的哪个部分修改
Clear Log
日志信息显示窗口

文件格式

文件格式



文件格式


C:\fuzz\jpg\gradient.jpg
原始文件
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: FF D8 FF E0 00 10 4A 46 49 46 00 01 02 00 00 64
C:\fuzz\jpg\0.jpg
FileFuzz产生的第一个jpg文件
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: 00 00 00 00 00 10 4A 46 49 46 00 01 02 00 00 64
C:\fuzz\jpg\1.jpg
FileFuzz产生的第二个jpg文件
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: FF 00 00 00 00 10 4A 46 49 46 00 01 02 00 00 64
C:\fuzz\jpg\2.jpg
FileFuzz产生的第三个jpg文件
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: FF D8 00 00 00 00 4A 46 49 46 00 01 02 00 00 64
C:\fuzz\jpg\3.jpg
FileFuzz产生的第四个jpg文件
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: FF D8 FF 00 00 00 00 46 49 46 00 01 02 00 00 64

基于块的数据模型

存在的问题

- 以JavaScript 代码为例来，这段输入数据没有明显的数据块边界。
  - 基于块的建模方式无法精确刻画记号之间的语法约束关系。
  - 基于块的约束关系只能描述块与块之间的数值约束和字符串长度约束等，无法表达这类语法结构的要求。

```
var Dom={};
Object.extend(Dom,{getViewPort:function(){
var _1;
var _2;
if(typeof window.innerWidth!="undefined"){
_1=window.innerWidth;
_2=window.innerHeight;
}else{
if(typeof document.documentElement!="undefined"&&typeof document.documentElement.clientWidth!="undefined"&&document.documentElement.clientWidth!=0){
_1=document.documentElement.clientWidth;
_2=document.documentElement.clientHeight;
}else{
_1=document.getElementsByTagName("body")[0].clientWidth;
_2=document.getElementsByTagName("body")[0].clientHeight;
}
}
return {width:_1,height:_2};
}});
```

基于文法的数据模型

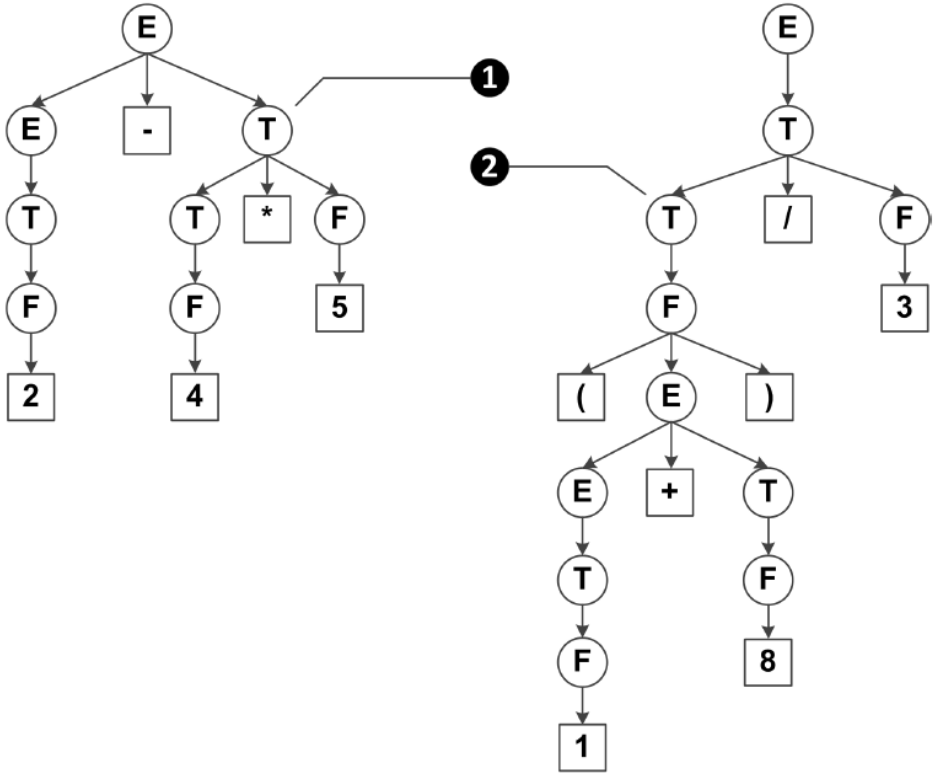
- 根据目标文法来产生符合文法要求的测试数据。这就使得生成的数据能够通过目标程序的词法分析和语法分析阶段，直接对更深层次的代码进行测试。
- 包括两个阶段的处理：

模型推断

```
expr → expr + term | expr - term | term
term → term * factor | term / factor | factor
factor → DIGIT | (expr)
```

四则计算器BNF描述

- 手工编写一个测试用例的集合（即种子集）。把种子集中的每个测试用例转化为一棵解析树，然后根据每棵树的结构来推断目标文法中的语法规则。



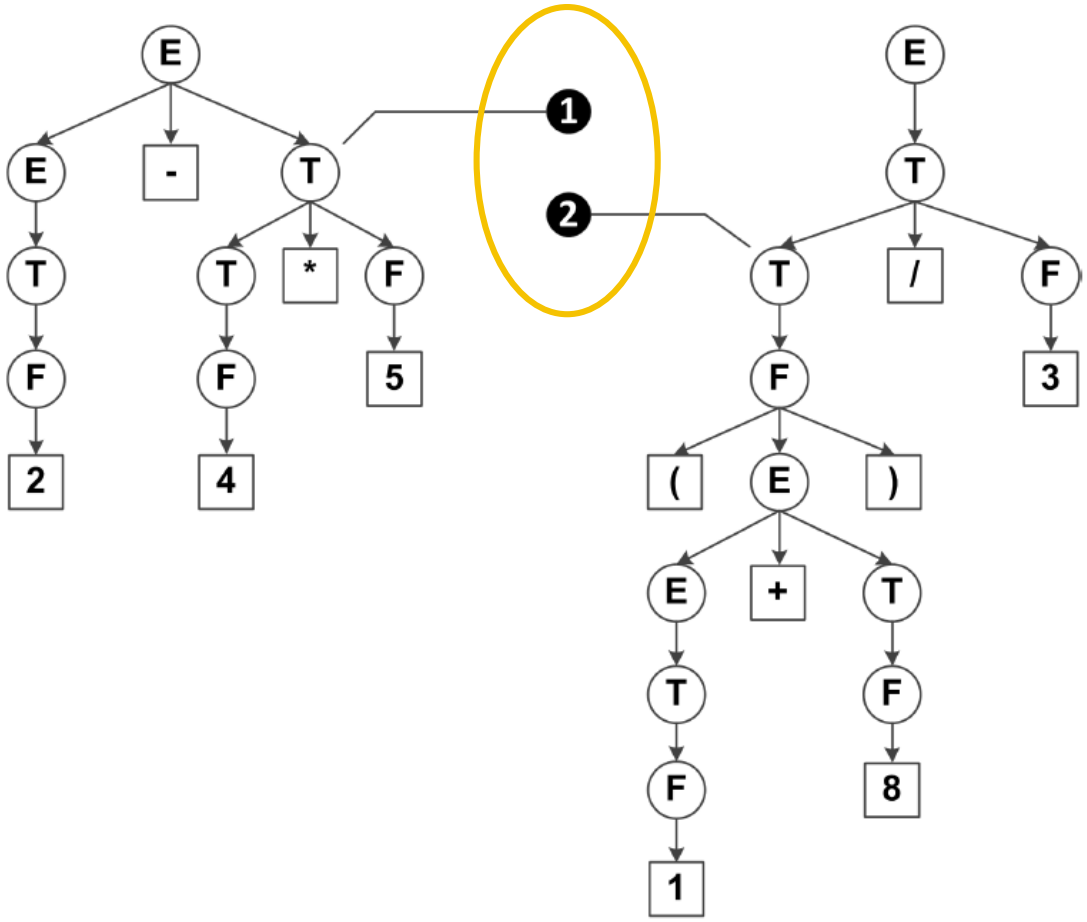
2-4×5的解析树

(1+8)/3的解析树

目标文法的BNF 描述。巴科斯范式(BNF: Backus-Naur Form 的缩写)是由 John Backus 和 Peter Naur 首次引入一种形式化符号来描述给定语言的语法

基于文法的变异

- 通过对已有测试用例进行变异来生成新的测试数据。
- 从一个现有的测试用例中选择一个语法片段，然后将其替换为另一个表示相同语法成分，但结构有所不同的语法片段。
- 通过系统地重复这个操作，就能得到一个规模更大、语法结构的组合也更丰富的新的测试用例的集合。



(a) “2 – 4 * 5”的解析树　　(b) “(1 + 8) / 3”的解析树

这样就可以把2 – 4 * 5 中的4 * 5 替换为(1 + 8)，从而得到一个新的测试用例2 – (1 + 8)。

总结

问题描述：针对以高度结构化数据为输入的程序进行模糊测试

给定：

- 一个上下文无关文法 $G = (V, T, P, S)$ -> 目标文法
- 一个程序 *PROGRAM* -> 目标程序
  - 其合法输入 $INPUT \subseteq L(G)$
- 一个手工编写的测试用例的集合 *SEED* -> 种子集
  - 容易获得
  - 规模太小，且缺乏足够的结构复杂度

目标：

- 生成一个测试用例的集合 *TESTCASES*
  - $TESTCASES \subseteq INPUT$
  - 应尽可能多地包含 $L(G)$ 中各种语法结构的复杂组合

对已有测试用例进行变异来生成新的测试数据

- 替换已有测试用例中的语法片段
- 要求：类型相同，结构不同
- 系统地重复进行替换

seed.txt

```
abcd\t\n\r\f\a\e\071\x3b\$\\\?caxyz
a*abc?xyz+pqr{3}ab{2,}xy{4,5}pq{0,6}AB{0,}z
^(abc){1,2}zz
^(b+?|a){1,2}?c
^(b+|a){1,2}c
^(b+|a){1,2}?bc
^(b*|ba){1,2}?bc
^(ba|b*){1,2}?bc
^\ca\cA\c[\c{\c:
^[ab\]cde]
^[]cde]
^[^ab\]cde]
^[^]cde]
^[0-9]+$
^.*nter
^xxx[0-9]+$
^.+[0-9][0-9][0-9]$
^.+?[0-9][0-9][0-9]$
^([^!]+)!(.+)=apquxz\.ixr\.zzz\.ac\.uk$
:
([\da-f:]+)$
^.*\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})$
^(\d+)\s+IN\s+SOA\s+(\S+)\s+(\S+)\s*\(\s*$
^[a-zA-Z\d][a-zA-Z\d\-]*(\.[a-zA-Z\d][a-zA-
^\*\.[a-z]([a-z\-\d]*[a-z\d]+)?(\.[a-z]([a-
^(?=ab(de))(abd)(e)
^(?!(ab)de|x)(abd)(f)
^(?=(ab(cd)))(ab)
^[\da-f](\.[\da-f])*$
^\".*\"\s*(;.*)?$
^$
^   a\ b[c ]d       $
^(a(b(c)))(d(e(f)))(h(i(j)))(k(l(m)))$
^(?:a(b(c)))(?:d(e(f)))(?:h(i(j)))(?:k(l(m)
^[\w][\W][\s][\S][\d][\D][\b][\n][\c]][\022
^[.^$|()*+?{,}]+
^a*\w
^a*?\w
^a+\w
^a+?\w
```

Seed

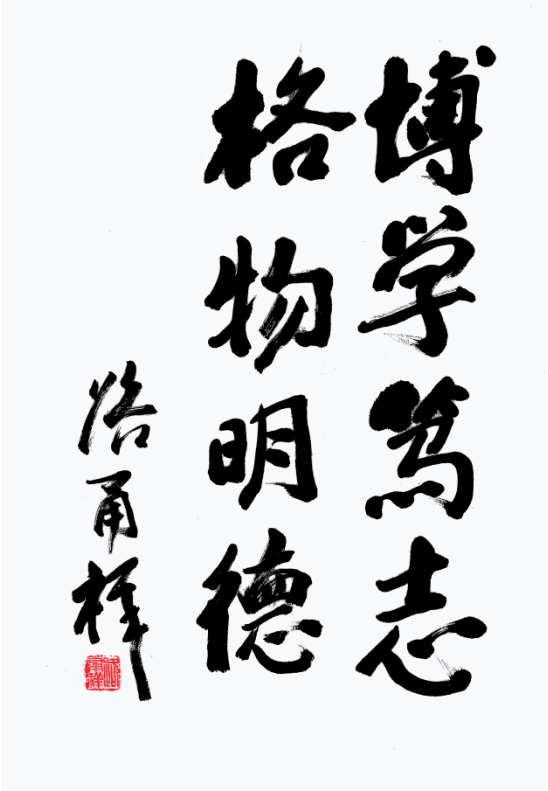
seed.txt | testcase_0.txt

```
!((?:a(?:(?:))*)*(ract|tonic)?)+$
!((?:a(?:(?:))*)*( (?>[^()<>]+) | ((?>[^()]+)) | (?R) )?)+$
!((?:a(?:(?:))*)*()?)+$
!((?:a(?:(?:))*)*(a+|b+|c+)?)+$
!((?:a(?:(?:))*)*(^b|(?i)^d)?)+$
!((?:a(?:(?:))*)*(a b(?x)c d (?-x)e f)?)+$
!((?:a(?:(?:))*)*(a|abcd|african)?)+$
!((?:a(?:(?:))*)*(a|b|c|d|e)?)+$
!((?:a(?:(?:))*)*(bc+d$|ef*g.|h?i(j|k))?)+$
!((?:a(?:(?:))*)*(\()?)+$
!((?:a(?:(?:))*)*(b?)?)+$
!((?:a(?:(?:))*)*(\z)?)+$
!((?:a(?:(?:))*)*((abc)*)?)+$
!((?:a(?:(?:))*)*(|)?)+$
!((?:a(?:(?:))*)*([ab\]cde])?)+$
!((?:a(?:(?:))*)*([abcd]{2,3})?)+$
!((?:a(?:(?:))*)*(.{0,5}?)?)+$
!((?:a(?:(?:))*)*((a|bbbb\1))?)+$
!((?:a(?:(?:))*)*(\d{8,})?)+$
!((?:a(?:(?:))*)*((?<=a))?)+$
!((?:a(?:(?:))*)*((?:(?:(?:(?:(?:(?:(?:(a)))))))))?)+$
!((?:a(?:(?:))*)*(\S)?)+$
!((?:a(?:(?:))*)*(\1)?)+$
!((?:a(?:(?:))*)*(.)?)+$
!((?:a(?:(?:))*)*(\1{2,4}?)?)+$
!((?:a(?:(?:))*)*((?#xxx))?)+$
!((?:a(?:(?:))*)*((?>\d+))?)+$
!((?:a(?:(?:))*)*((?>)+)?)+$
!((?:a(?:(?:))*)*((?:(f)(o)(o)|(b)(a)(r))*)?)+$
!((?:a(?:(?:))*)*(\(*)?)+$
!((?:a(?:(?:))*)*((?#xxx){3})?)+$
!((?:a(?:(?:))*)*(\Qabc\E)?)+$
!((?:a(?:(?:))*)*(^*)?)+$
!((?:a(?:(?:))*)*((?!)+)?)+$
!((?:a(?:(?:))*)*(\000)?)+$
!((?:a(?:(?:))*)*(\x00)?)+$
!((?:a(?:(?:))*)*(!)?)+$
!((?:a(?:(?:))*)*(=)?)+$
!((?:a(?:(?:))*)*(:)?)+$
!((?:a(?:(?:))*)*(3)?)+$
```

Testcase

本章大纲

- 网页木马
- ActiveX
- Adobe Flash Player
- 浏览器安全机制



浏览器安全机制总览

Sandbox

沙箱：是一种隔离对象/线程/进程的机制，目的是控制其访问系统资源的权限。最初的沙箱是基于Hook实现的，后来的Chrome沙箱是利用操作系统提供的一些安全机制实现的。

JIT Hardening

防止对JIT引擎本身的滥用。常用技术包括：代码库队列随机化、指令库队列随机化、常量合并、内存页面保护、资源限制等。

ASLR

地址空间布局随机化(Address Space Layout Randomization)：是一项缓解缓冲区溢出问题的安全技术。其原理是将进程运行所需的系统核心组件和对象在内存中的分布随机化。

浏览器安全机制简介

DEP

数据执行保护(Data Execution Prevention)：阻止数据页执行代码。将数据所在内存页标识为不可执行，当程序尝试在数据页面上执行指令时会抛出异常，而不是去执行恶意指令。

/GS

缓冲区安全检查：是一种不强制缓冲区大小限制的代码常用技术。通过将安全检查插入到已编译代码中完成，检测某些改写返回地址的缓冲区溢出。

CFG

执行流保护(Control Flow Guard)：是一种编译器和操作系统相结合的防护手段，目的在于防止不可信的间接调用。对基于虚表进行攻击的利用手段可以有效防御。

浏览器安全机制总览

Edge浏览器和IE浏览器对沙箱机制的实现最大的差异是：Edge浏览器将框架进程也包含在了整个安全体系里面，权限更低，大大提高了安全性。

	Chrome	IE	Edge	Safari	Firefox
Sandbox	✓	✓	✓	✓	✓
JIT Hardening	✓	✓	✓	✓	✓
ASLR	✓	✓	✓	✓	✓
DEP	✓	✓	✓	✓	✓
/GS	✓	✓	✓	✗	✓
CFG	✓	✓	✓	CFI*	✓

*CFI(Control-Flow Integrity)能对全部执行流进行保护，但在执行过程中检测的频率极高，难免对程序执行效率带来影响。

虽然对某一安全机制有多个浏览器支持，但各个浏览器的实现方式及实现程度不尽相同。

浏览器安全机制总览

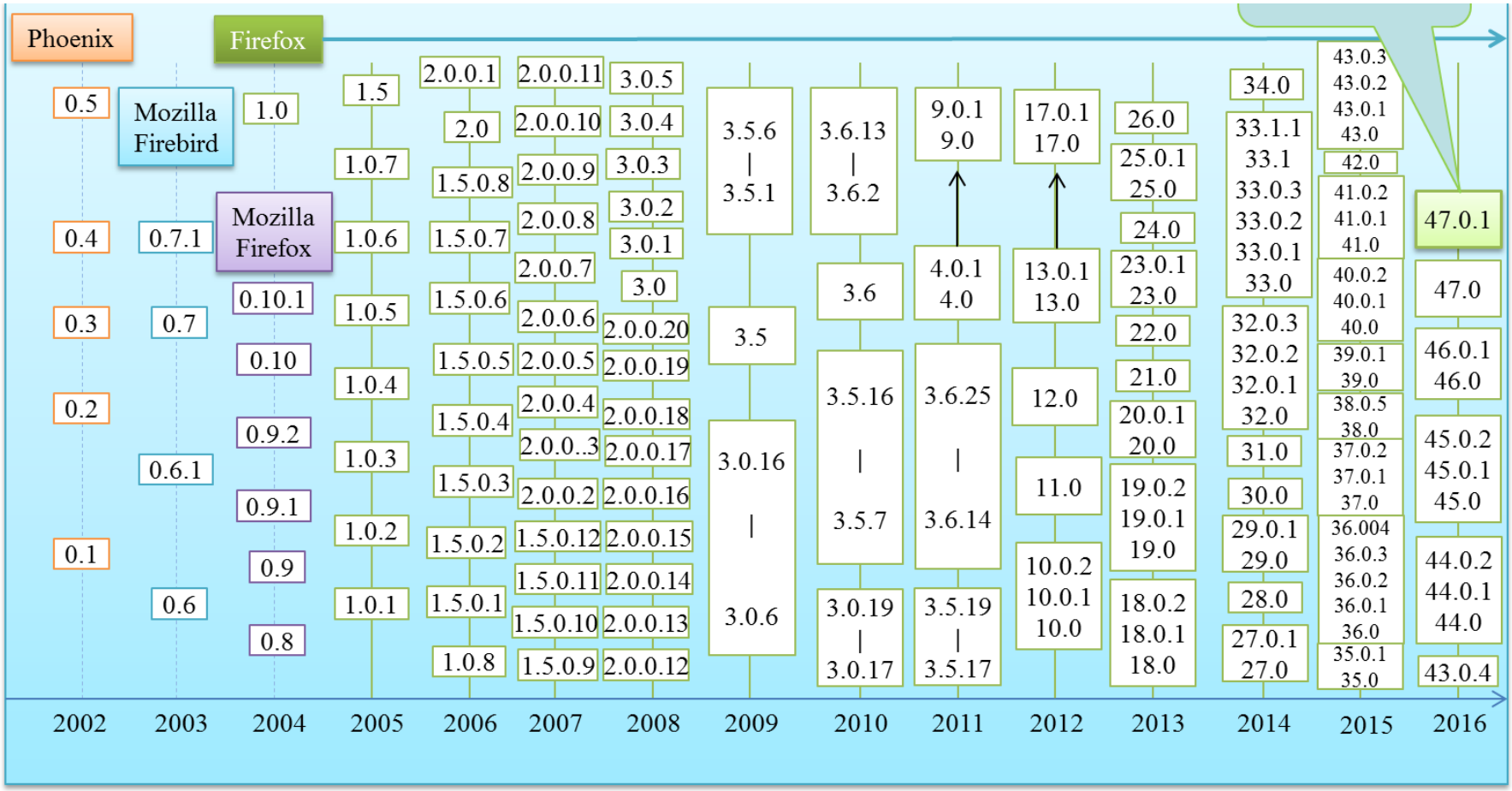
	Chrome	IE	Edge	Safari	Firefox
Sandbox	✓	✓	✓	✓	✓
JIT Hardening	✓	✓	✓	✓	✓
ASLR	✓	✓	✓	✓	✓
DEP	✓	✓	✓	✓	✓
/GS	✓	✓	✓	✗	✓
CFG	✓	✓	✓	CFI	✓
扩展签名	✗	✗	✓	✗	✓
W^X JIT	✗	✗	✗	✗	✓
MemGC	✗	✗	✓	✗	✗

新增

FIREFOX

附加组件签名机制

W^X JIT-code



附加组件签名机制

附加组件

附加组件是一种通过增添额外的功能或样式让用户实现个性化 Firefox 的应用程序，包括扩展、外观、插件、服务等类型。

早期机制—黑名单系统

已知会造成 Firefox 稳定性或安全性问题的附加组件（扩展、主题和插件）会放入“阻挡列表”（Blocklist）。

存在问题：

1、新增的附件组件的安全性？

2、第三方安装的附加组件的安全性？

已被封锁的附加组件

已知在 Firefox 上，以下软件会引发严重的安全性、稳定性或者性能问题。

附加组件签名机制

Mozilla 根据一套安全准则对附加组件进行验证并为其“签名”，需要签名的类型包括**扩展**。

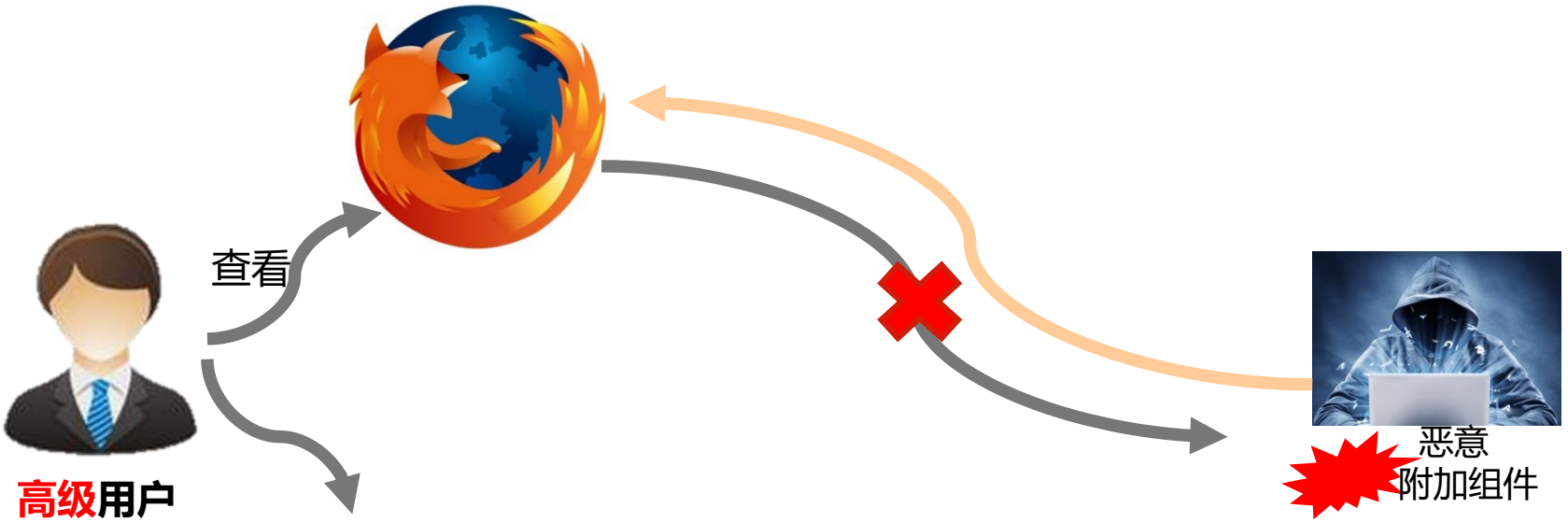
Firefox 40版本起，未被签名的扩展将被标记

附加组件签名机制

Android系统中的Firefox浏览器从46.0版本开始也增加了附件组件签名机制。

Mozilla 根据一套安全准则对附加组件进行验证并为其“签名”。

Firefox 43版本起，未被签名的扩展将默认被禁用



W^X JIT-CODE

Firefox 46版本起，开始支持 W^X机制

W^X”是“写异或执行”（Write XOR Execute）的缩写，是OpenBSD中具有代表性的安全特性之一。W^R内存保护机制能够让网页使用内存写入代码或执行代码，但不能够同时进行这两种操作。

W^X JIT-CODE

Firefox 46版本起，开始支持 W^X机制

存在缓冲区溢出的隐患

在加入W^X支持之前，Firefox给予了网页完整的RWX（读、写、执行）许可

- **Mozilla Firefox 缓冲区溢出及拒绝服务漏洞(CVE-2015-7179)**

  Mozilla Firefox 41.0之前版本、Firefox ESR 38.3之前版本，ANGLE内libGLES 的函数VertexBufferInterface::reserveVertexSpace，错误分配了渲染器属性数组的内存，存在安全漏洞，远程攻击者利用此漏洞可造成拒绝服务（内存破坏及应用崩溃）。

  http://www.linuxidc.com/Linux/2015-10/123726.htm 日期：2015/10/2 9:47:54

- **Mozilla Firefox/Thunderbird堆缓冲区溢出漏洞(CVE-2014-1549)**

  Firefox 31、Thunderbird 31在回放的Web Audio缓冲区内存在内存分配安全问题，这可导致某些音频内容导致的崩溃。

  http://www.linuxidc.com/Linux/2014-07/104590.htm 日期：2014/7/23 19:43:50

- **Mozilla Firefox/Thunderbird/SeaMonkey JS引擎远程缓冲区溢出漏洞(CVE-2013-5595)**

  Mozilla Firefox 25.0之前版本, Firefox ESR 24.1之前版本, Thunderbird 24.1之前版本, SeaMonkey 2.22之前版本的JavaScript引擎没有正确分配某些函数的内存，远程攻击者通过特制的网页，利用此漏洞可执行缓冲区溢出攻击。

  http://www.linuxidc.com/Linux/2013-11/92271.htm 日期：2013/11/3 7:42:36

- **Firefox Foxit Reader插件npFoxitReaderPlugin.dll栈缓冲区溢出漏洞**

  Foxit Reader npFoxitReaderPlugin.dll是Firefox的插件，npFoxitReaderPlugin.dll在copy字符串时没有正确检查缓冲区边界，导致存在栈溢出漏洞 。攻击者通过诱使用户打开包含超长查询串的特制文件，可达到使应用程序崩溃，甚至执行任意代码的目的。

  http://www.linuxidc.com/Linux/2013-01/77487.htm 日期：2013/1/9 8:12:29

- **Mozilla Firefox SeaMonkey和Thunderbird堆缓冲区溢出漏洞（CVE-2012-1941）**

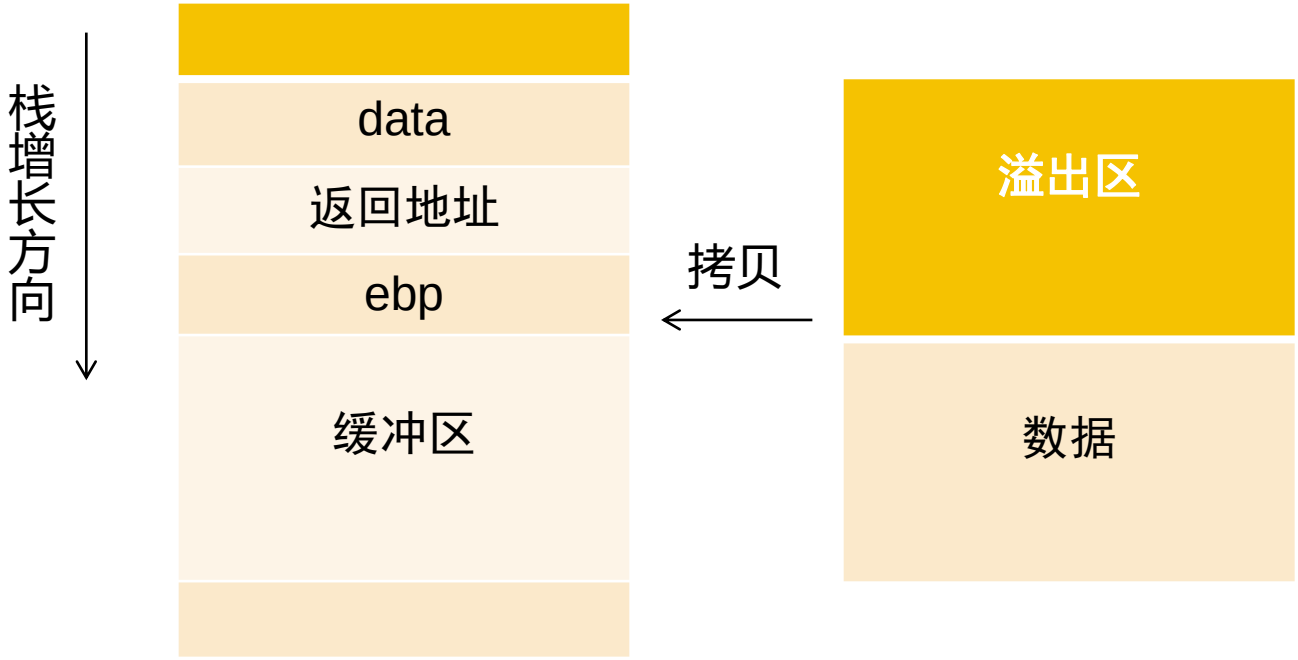
  Firefox 13.0、Thunderbird 13.0、SeaMonkey 2.10在具有嵌套的多列、相对位置、绝对位置的nsHTMLReflowState::CalculateHypotheticalBox的实现上存在堆缓冲区溢出漏洞，攻击者可利用此漏洞执行任意代码。

W^X JIT-CODE

Firefox 46版本起，开始支持 W^X机制

存在缓冲区溢出的隐患

在加入W^X支持之前，Firefox给予了网页完整的RWX（读、写、执行）许可

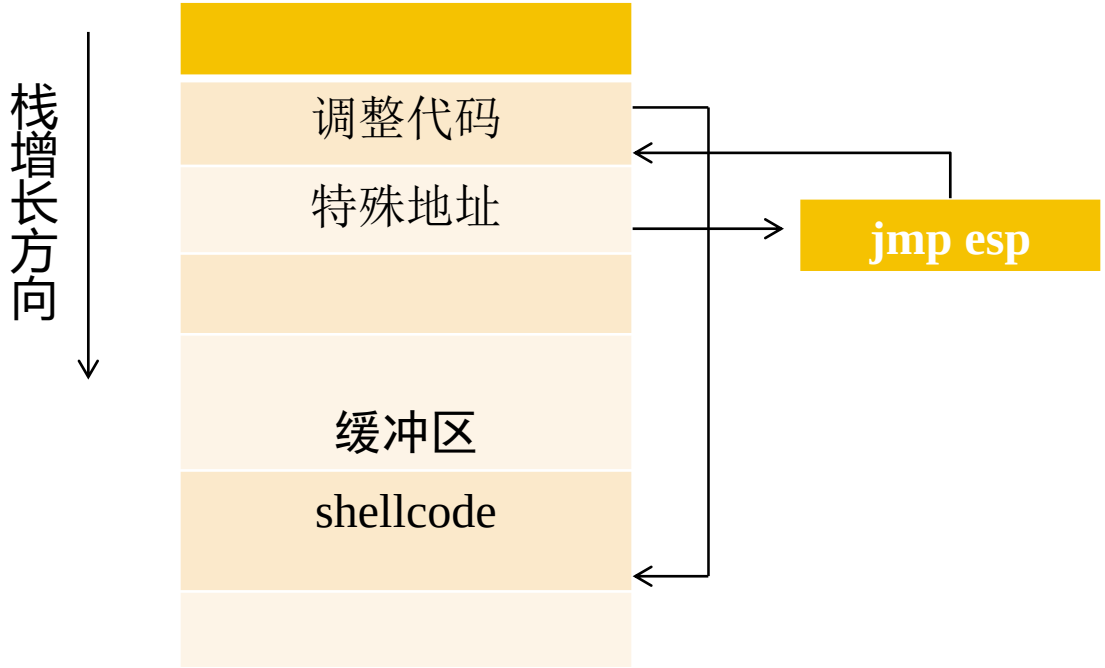


W^X JIT-CODE

Firefox 46版本起，开始支持 W^X机制

存在缓冲区溢出的隐患

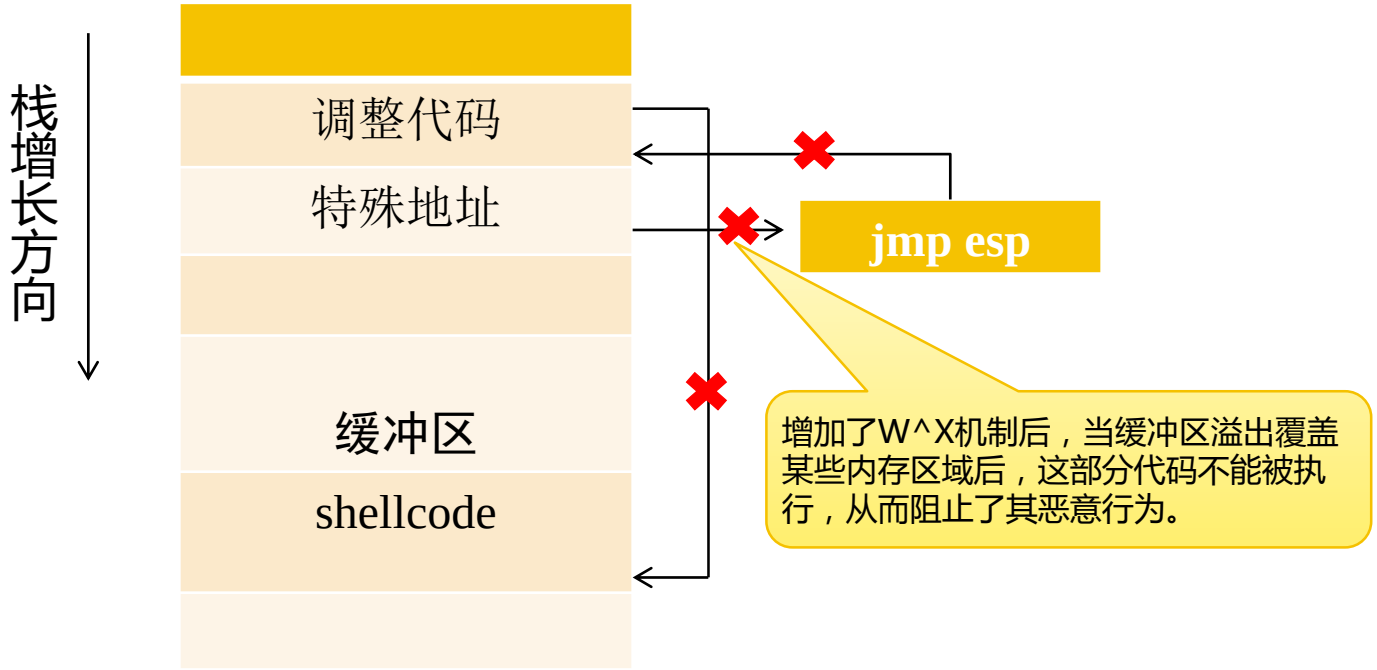
在加入W^X支持之前，Firefox给予了网页完整的RWX许可



W^X JIT-CODE

Firefox 46版本起，开始支持 W^X机制

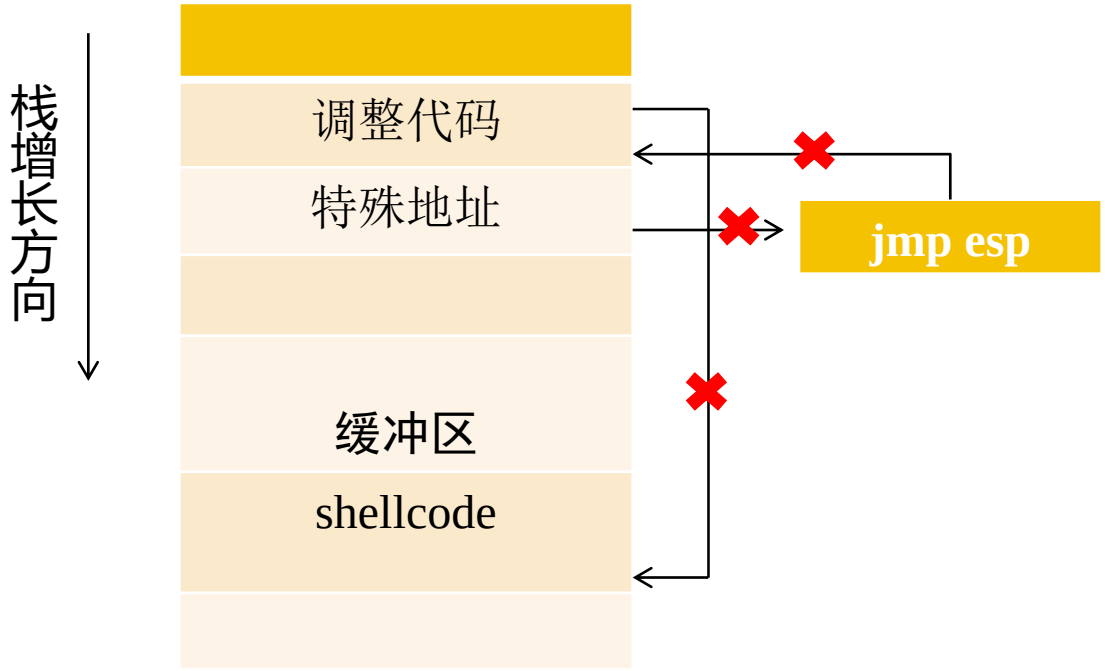
W^R内存保护机制能够让网页使用内存写入代码或执行代码，但不能够同时进行这两种操作。



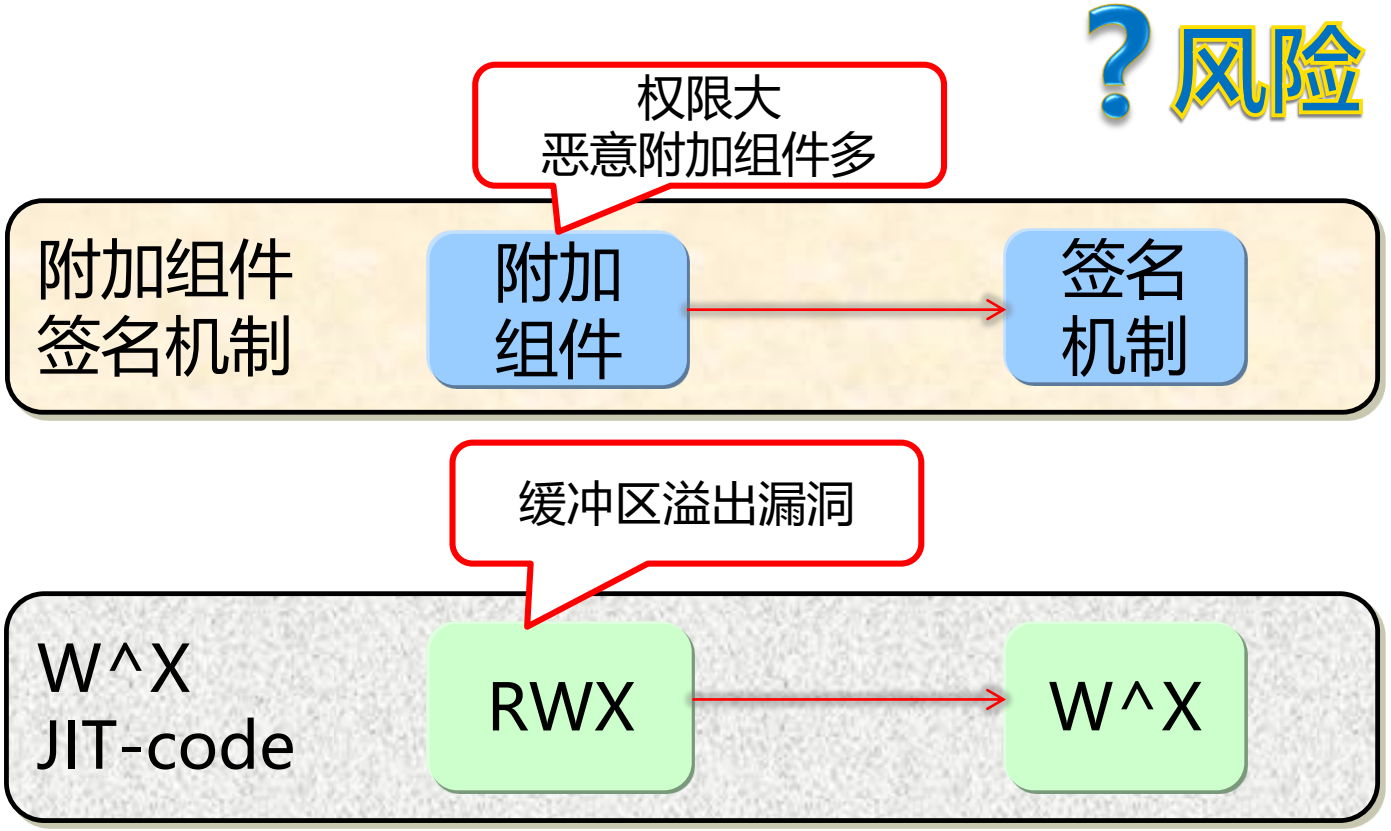
W^X JIT-CODE

Firefox 46版本起，开始支持 W^X机制

可能会出现缓冲区溢出导致崩溃，但阻止了某些缓冲区溢出的攻击

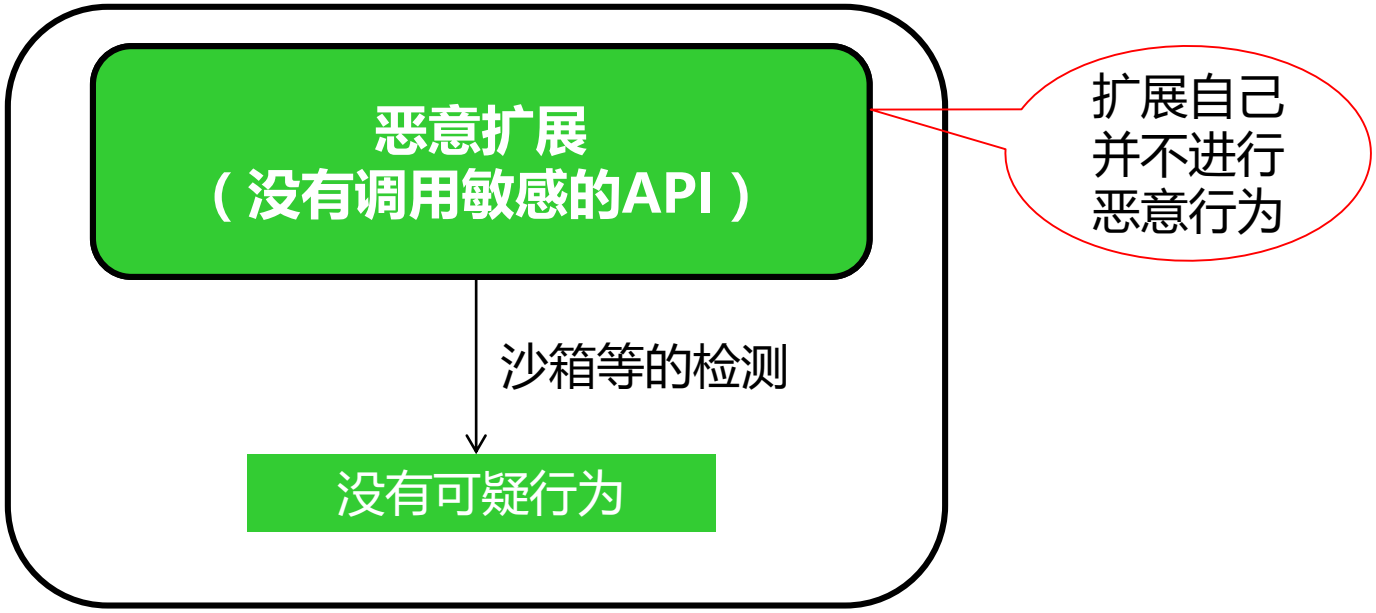


小结



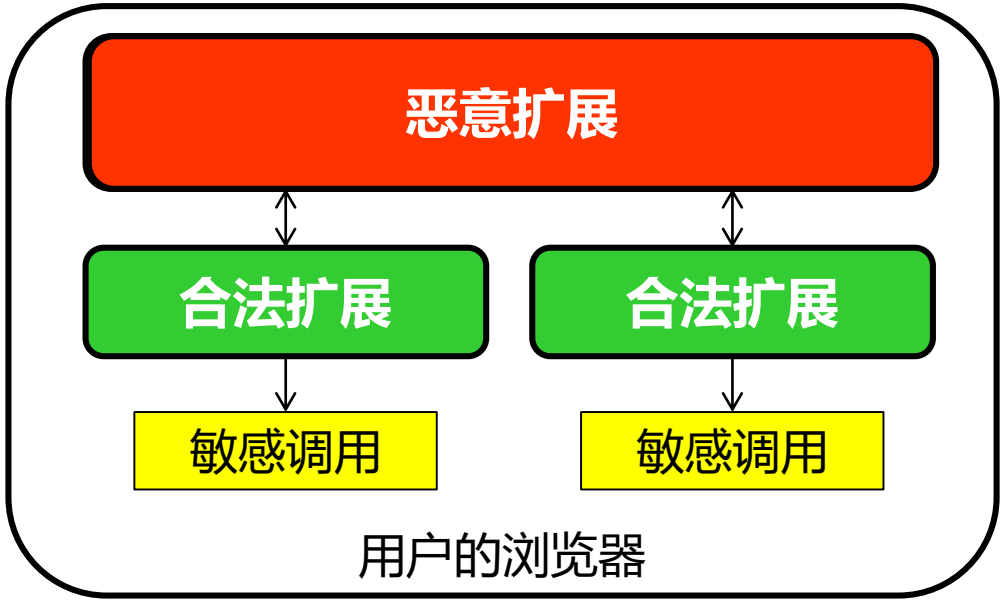
风险一

2016年的Blackhat大会上，来自美国波士顿大学的 Ahmet Buyukkayhan 博士和东北大学的 William Robertson 教授提出了“**extension reuse**”机制，称可以创建一个本身不调用敏感API的恶意扩展，但可以利用其它扩展，完成恶意行为，并最终感染系统。



风险一

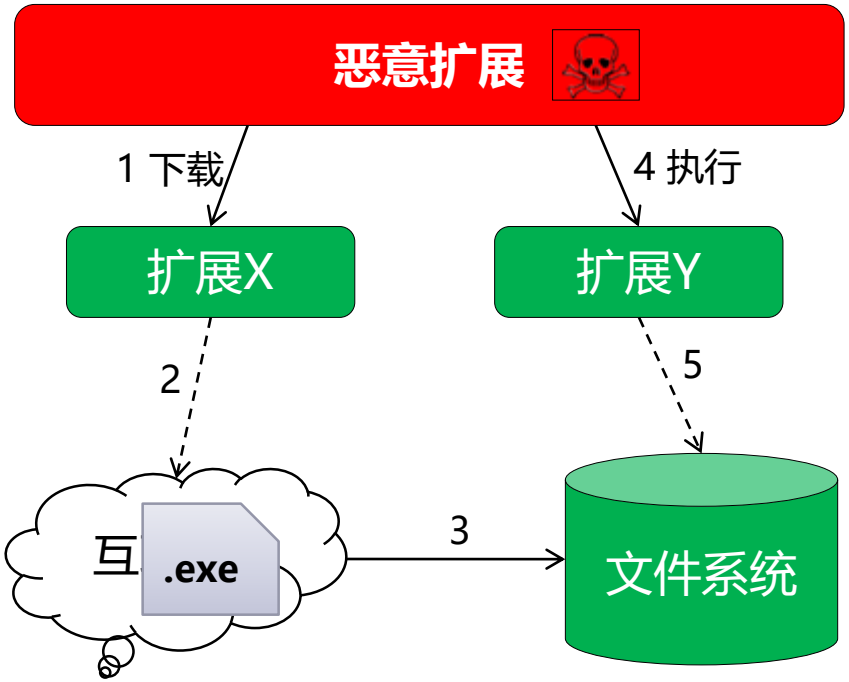
2016年的Blackhat大会上，来自美国波士顿大学的 Ahmet Buyukkayhan 博士和东北大学的 William Robertson 教授提出了“**extension reuse**”机制，称可以创建一个本身不调用敏感API的恶意扩展，但可以利用其它扩展，完成恶意行为，并最终感染系统。



由于Firefox的扩展共享Javascript命名空间，恶意扩展可利用其他扩展，完成恶意行为

风险一

2016年的Blackhat大会上，来自美国波士顿大学的 Ahmet Buyukkayhan 博士和东北大学的 William Robertson 教授提出了“**extension reuse**”机制，称可以创建一个本身不调用敏感API的恶意扩展，但可以利用其它扩展，完成恶意行为，并最终感染系统。



风险二

附加组件签名机制可以被强制关闭

手动关闭

①打开Firefox 配置 编辑器

②更改首选项 xpinstall.signatures.required 为 false

可以通过修改Firefox配置文件强制关闭！

风险二

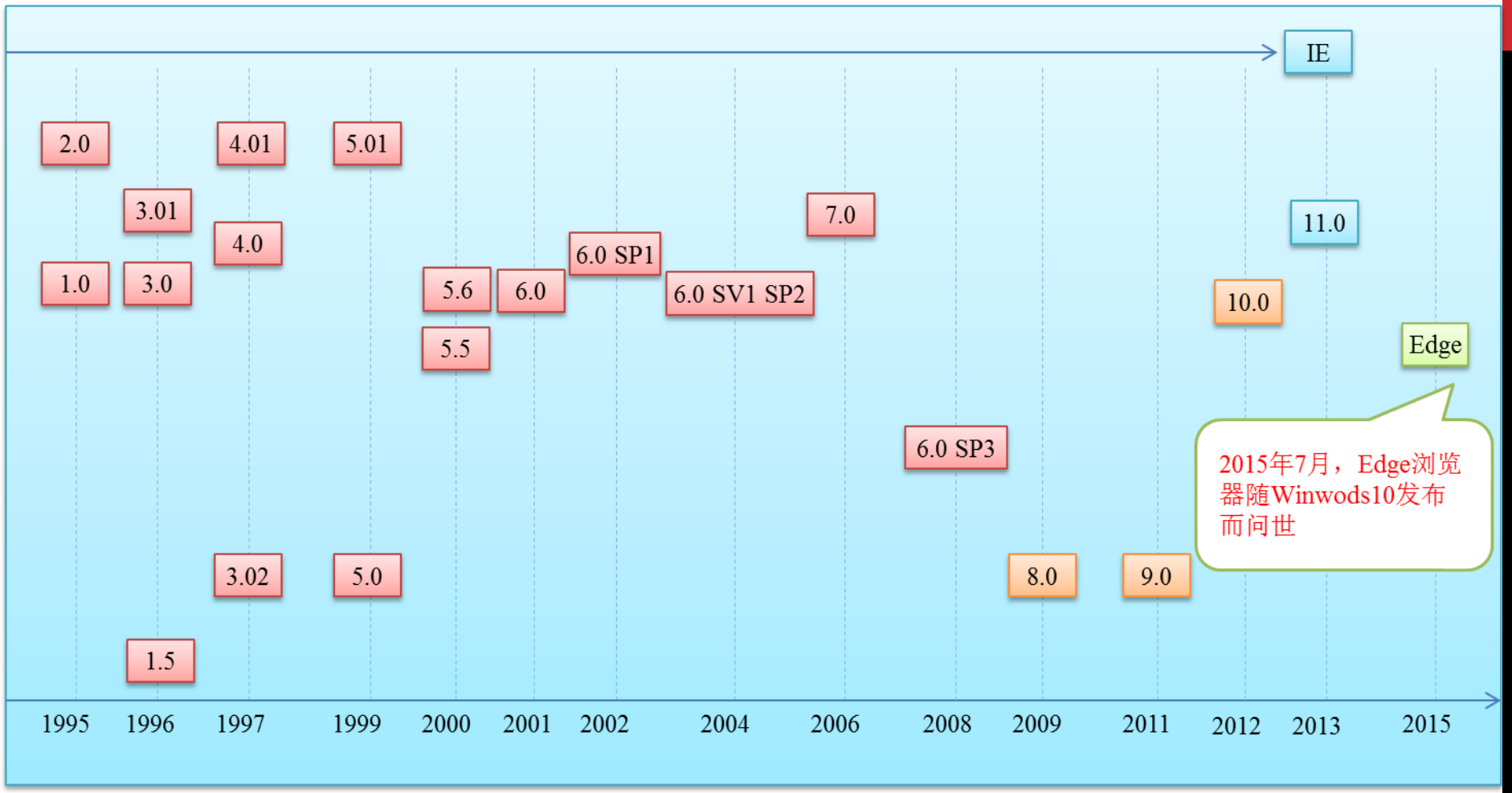
附加组件签名机制可以被强制关闭

关闭之后 → 查看

扩展功能正常

对于一般Firefox用户，查看并禁用扩展的可能性比较小，因此如果恶意扩展能够通过修改配置文件强行关闭扩展签名机制，那么恶意扩展仍有可能危害大部分用户。

EDGE



MEMGC内存垃圾收集器

MemGC(Memory Garbage Collector)是内存管理机制,由IE11的Memory Protector改进而来，首次在EdgeHTML和MSHTML中使用。

Edge使用MemGC来管理DOM和DOM支持的对象，MemGC采用标记清除(Mark-Sweep)算法对垃圾进行回收。

MemGC机制能够阻止部分UAF(Use After Free)漏洞。

MEMGC内存垃圾收集器

MemGC机制能够阻止部分UAF漏洞。

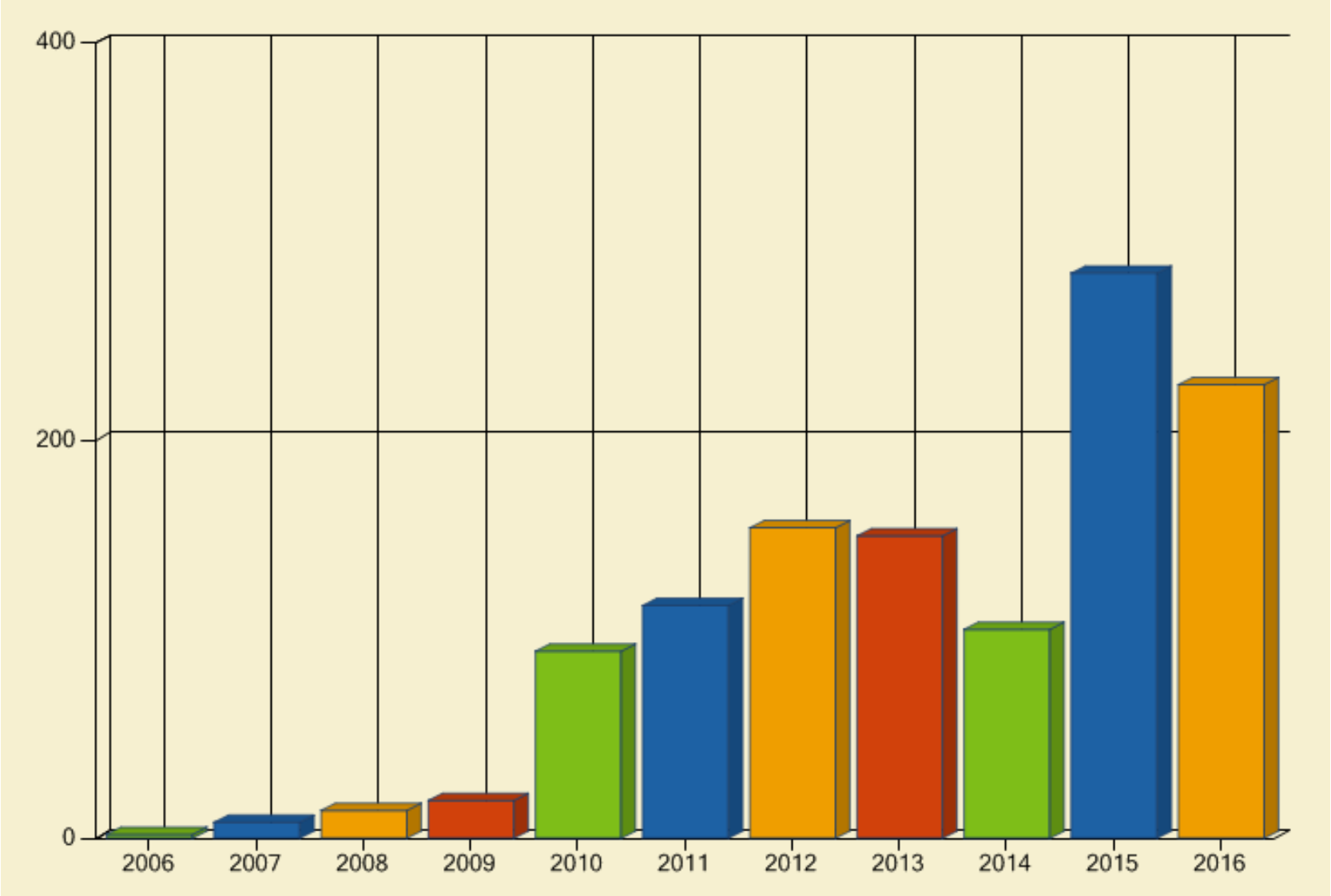
什么是UAF漏洞

UAF(Use After Free)：即释放后使用。CWE (Common Weakness Enumeration)给出的定义是：引用一段被释放的内存可导致程序崩溃，或处理非预期数值，或执行若干指令。

CVE-2016-1097

```
function poc()
{
    var ps:PSDK = PSDK.pSDK;
    ps.release();
    ps.createdispatcher();
}
```

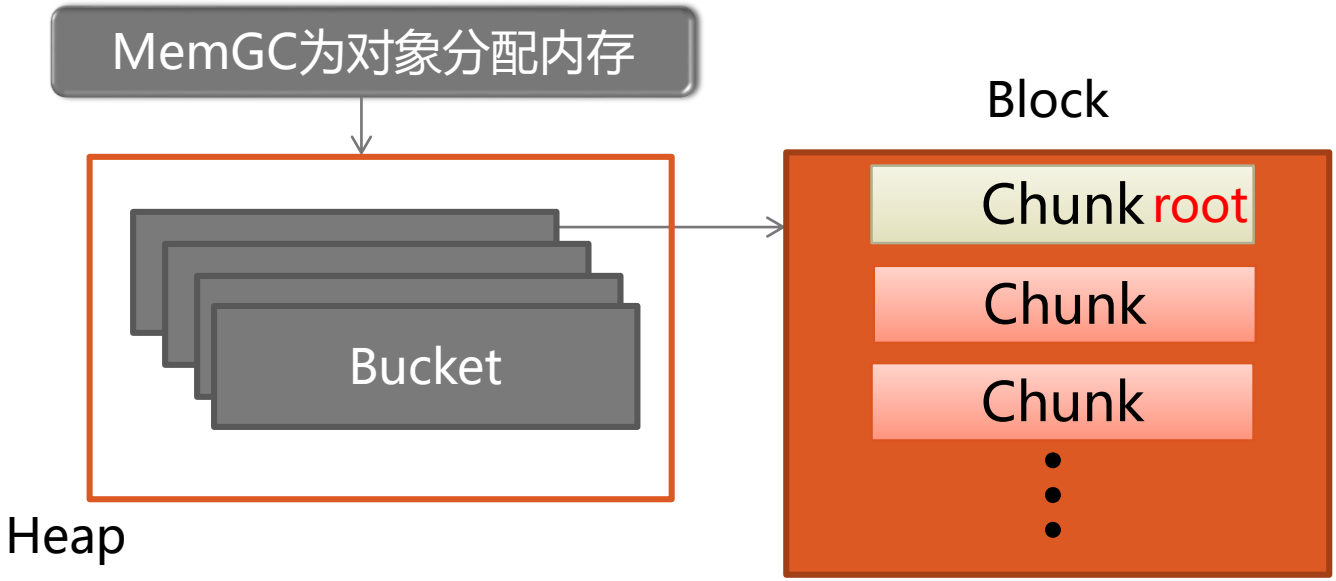
UAF(USE AFTER FREE)



MEMGC内存垃圾收集器

Allocate

MemGC为其管理的对象分配内存：MemGC首先找到合适的bucket，然后在bucket所指向的内存块block中分配一个小的数据块chunk给它，并将这个分配的chunk标记为root。

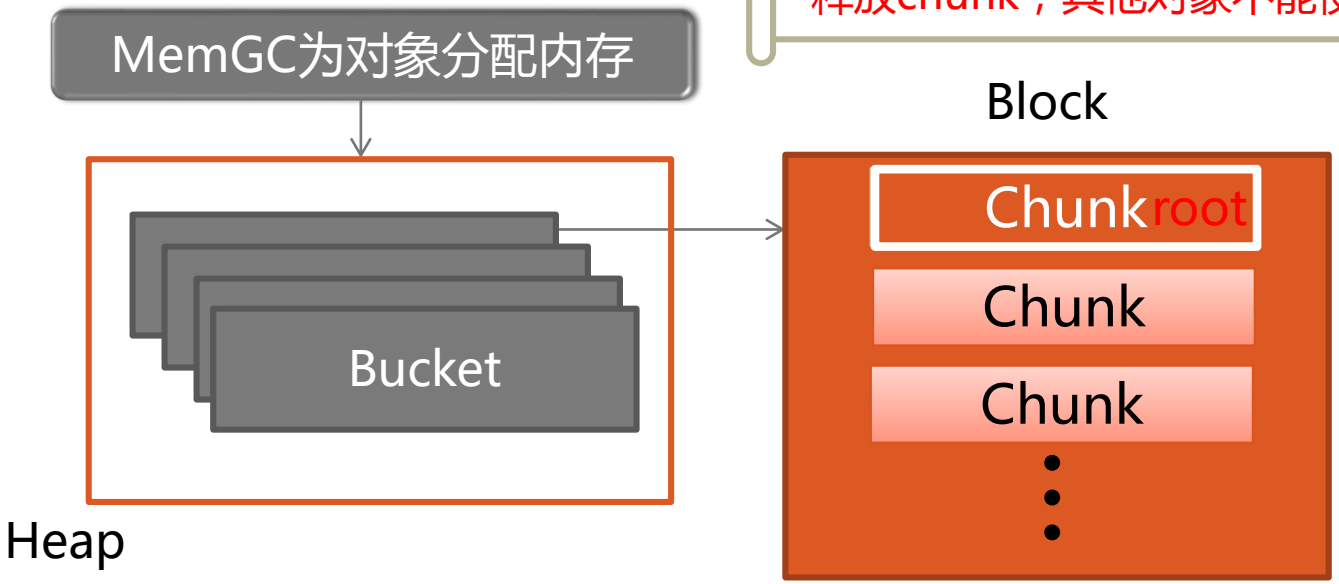


MEMGC内存垃圾收集器

Free

当对象被释放时，为其分配的chunk将清除root标记，成为垃圾回收的候选者。

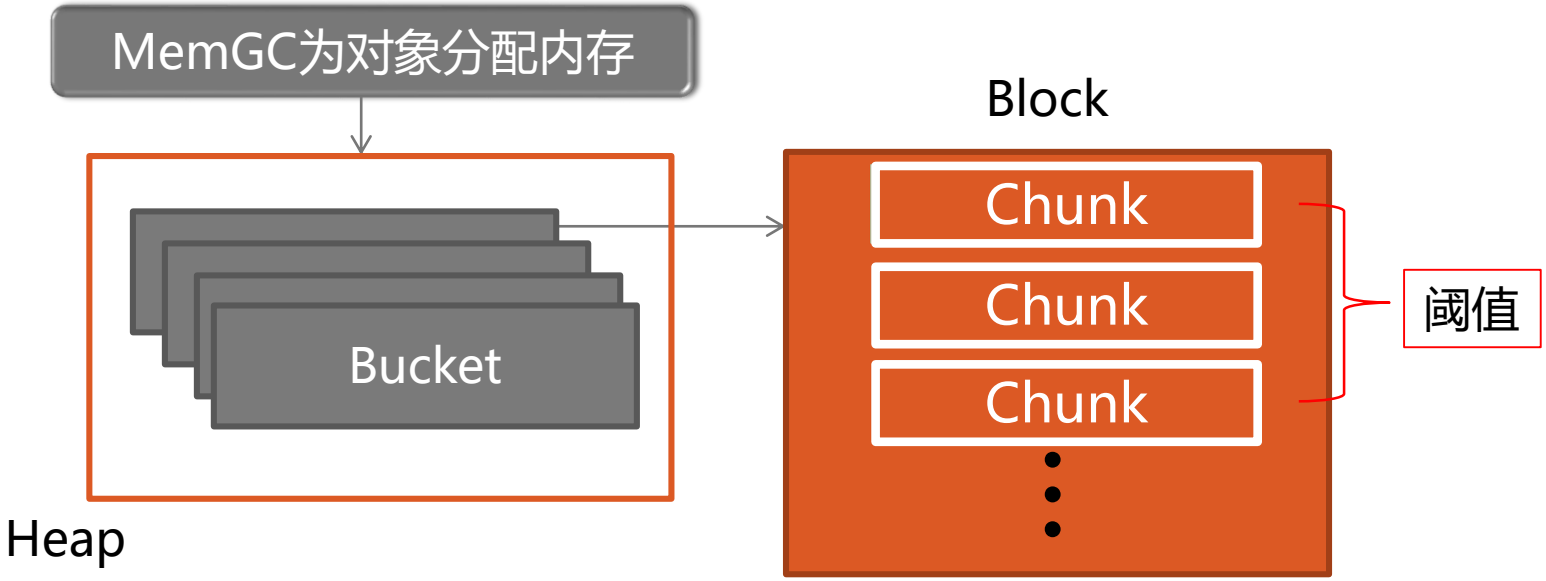
此处仅仅清除了chunk的root标记，并未真正释放chunk，其他对象不能使用这个chunk。



MEMGC内存垃圾收集器

Free

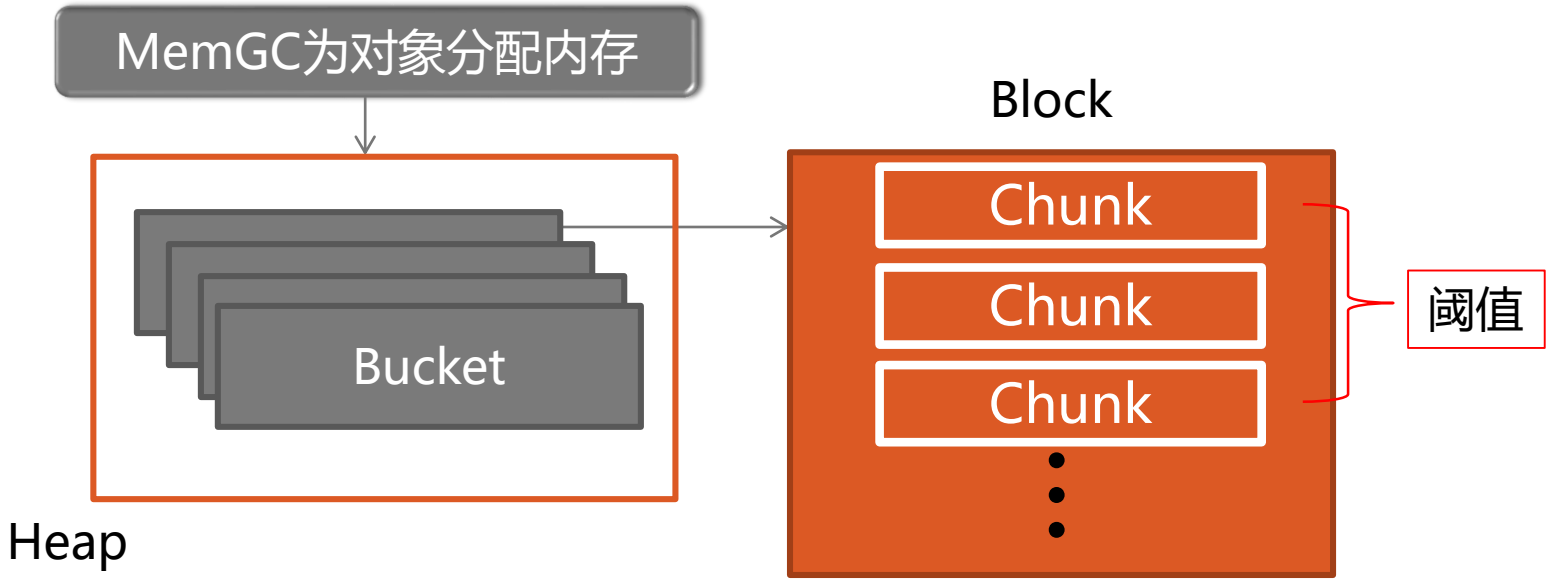
当被释放的chunk逐渐增多，达到特定的阈值，将触发垃圾回收机制。



MEMGC内存垃圾收集器

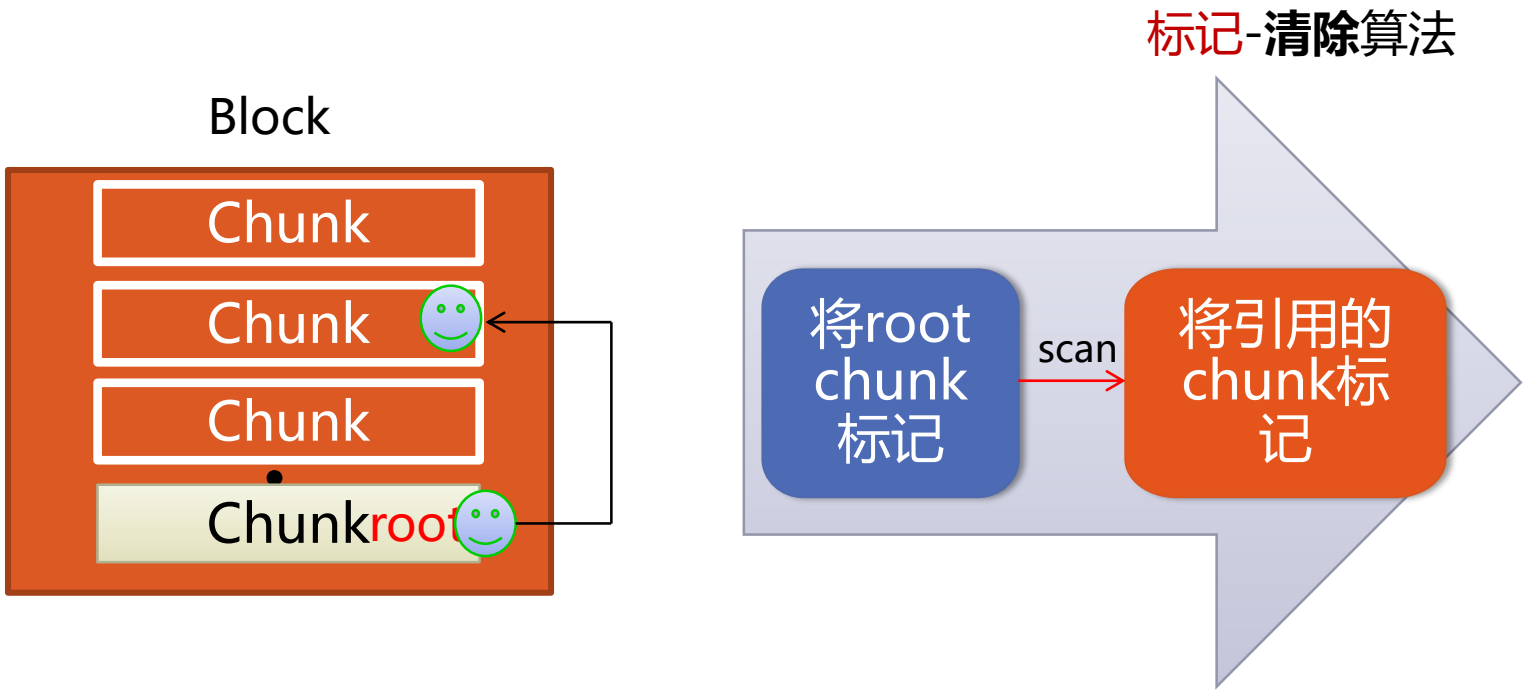
Mark-Sweep

标记-清除算法



MEMGC内存垃圾收集器

Mark-Sweep

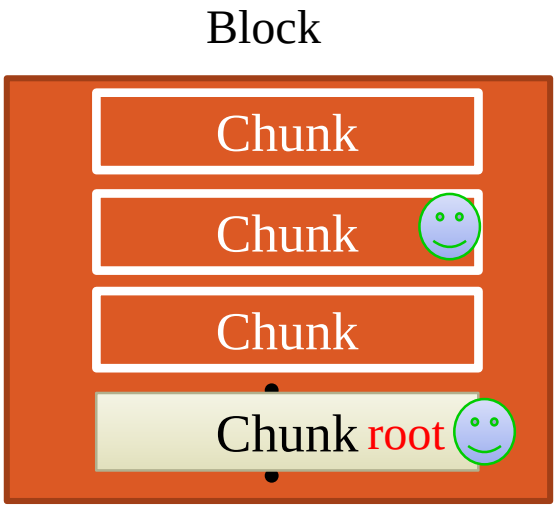


此时，除了待回收的chunk，还有未被释放的对象所占用的chunk，MemGC将全部进行扫描标记

MEMGC内存垃圾收集器

Mark-Sweep

标记-**清除**算法

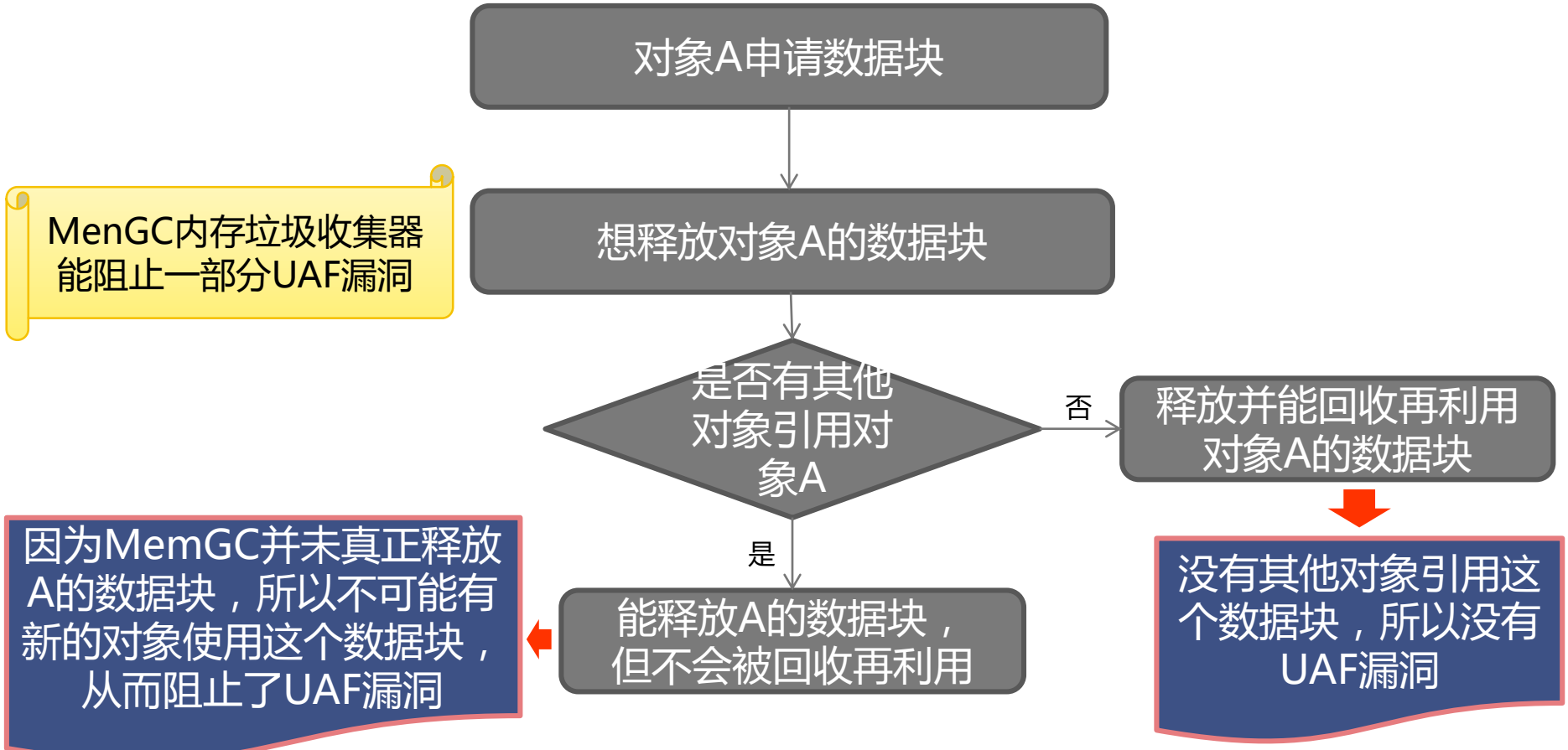


将**未标记**的chunk回收，可再次利用

此时，所有被申请对象释放且没有被其他未free的对象所引用的chunk才真正被回收利用

MEMGC内存垃圾收集器

如何阻止UAF漏洞



MenGC内存垃圾收集器能阻止一部分UAF漏洞

参考文献

TaintScope: A Checksum-Aware Directed Fuzzing Tool for Automatic Software Vulnerability Detection

引用 预览

作　者 Tielei Wang, Tao Wei, Guofei Gu, Wei Zou

摘　要 Fuzz testing has proven successful in finding security vulnerabilities in large programs. However, traditional fuzz testing tools have a well-known common drawback: they are ineffective if most generated malformed inputs are rejected in the early stage of program running, especially when target programs employ checksum mechanisms to verify the integrity of inputs. In this paper, we present TaintScope, an automatic fuzzing system using dynamic taint analysis and symbolic execution techniques, to tackle the above problem. TaintScope has several novel contributions: 1) TaintScope is the first checksum-aware fuzzing tool to the best of our knowledge. It can identify checksum fields in input instances, accurately locate checksum-based integrity checks by using branch profiling techniques, and bypass such checks via control flow alteration. 2) TaintScope is a directed fuzzing tool working at X86 binary level (on both Linux and Window). Based on fine-grained dynamic taint tracing, TaintScope identifies which bytes in a well-formed input are used in security-sensitive operations (e.g., invoking system/library calls) and then focuses on modifying such bytes. Thus, generated inputs are more likely to trigger potential vulnerabilities. 3) TaintScope is fully automatic, from detecting checksum, directed fuzzing, to repairing crashed samples. It can fix checksum values in generated inputs using combined concrete and symbolic execution techniques. We evaluate TaintScope on a number of large real-world applications. Experimental results show that TaintScope can accurately locate the checksum checks in programs and dramatically improve the effectiveness of fuzz testing. TaintScope has already found 27 previously unknown vulnerabilities in several widely used applications, including Adobe Acrobat, Google Picasa, Microsoft Paint, and ImageMagick. Most of these severe vulnerabilities have been confirmed by Secunia and oCERT, and assigned CVE identifiers (such as CVE-2009-1882, CVE-2009-2688). Corresponding patches from vendors are released or in progress based on our reports.

研究领域 concrete | assembly | control flow | digital signal processing | chromium

发表日期 2010

BlendFuzz: A Model-Based Framework for Fuzz Testing Programs with Grammatical Inputs

作　者 Dingning Yang, Yuqing Zhang, Qixu Liu

摘　要 Fuzz testing has been widely used in practice to detect software vulnerabilities. Traditional fuzzing tools typically use blocks to model program input. Despite the demonstrated success of this approach, its effectiveness is inherently limited when applied to test programs that process grammatical inputs, where the input data are mainly human-readable text with complex structures that are specified by a formal grammar. In this paper we present BlendFuzz, a fuzz testing framework that is grammar-aware. It works by breaking a set of existing test cases into units of grammar components, then using these units as variants to restructure existent test data, resulting in a wider range of test cases that have the potential to explore previously uncovered corner cases when used in testing. We've implemented this framework along with two language fuzzers on top of it. Experiments with these fuzzers have shown improved code coverage, and field testing has revealed over two dozens of previously unreported bugs in real-world applications, with seven of them being medium or high risk zero-day vulnerabilities.

研究领域 theoretical computer science | programming language | algorithm | computer science

发表日期 2012

被引量 5

Fuzzing the ActionScript virtual machine

引用

作　者 Guanxing Wen, Yuqing Zhang, Qixu Liu, Dingning Yang

摘　要 Fuzz testing is an automated testing technique where random data is used as an input to software systems in order to reveal security bugs/vulnerabilities. Fuzzed inputs must be binaries embedded with compiled bytecodes when testing against ActionScript virtual machines (AVMs). The current fuzzing method for JavaScript-like virtual machines is very limited when applied to compiler-involved AVMs. The complete source code should be both grammatically and semantically valid to allow execution by first passing through the compiler. In this paper, we present ScriptGene , an algorithmic approach to overcome the additional complexity of generating valid ActionScript programs. First, nearly-valid code snippets are randomly generated, with some controls on instruction flow. Second, we present a novel mutation method where the former code snippets are lexically analyzed and mutated with runtime information of the AVM, which helps us to build context for undefined behaviours against compiler-check and produce a high code coverage. Accordingly, we have implemented and evaluated ScriptGene on three different versions of Adobe AVMs. Results demonstrate that ScriptGene not only covers almost all the blocks of the official test suite (Tamarin), but also is capable of nearly twice the code coverage. The discovery of six bugs missed by the official test suite demonstrates the effectiveness, validity and novelty of ScriptGene .

研究领域 actionscript | fuzz testing | theoretical computer science | world wide web | computer security

发表日期 2013

会　议 Computer and Communications Security

《计算机研究与发展》 2012年07期　　加入收藏 投稿

基于Fuzzing的ActiveX控件漏洞挖掘技术研究

杨丁宁　肖晖　张玉清

【摘要】：ActiveX控件漏洞存在广泛且往往具有较高的威胁等级,有必要对此类漏洞的挖掘技术展开研究,发现并修复漏洞,从而杜绝安全隐患.在对ActiveX控件特性进行分析的基础上,设计并实现了ActiveX控件漏洞挖掘工具——ActiveX-Fuzzer.它基于黑盒Fuzzing测试技术,能够自动地构造半有效数据对控件接口展开测试,尝试发现潜在的缓冲区溢出、整数溢出及格式化字符串错误等安全问题.通过使用该工具对常用ActiveX控件进行广泛的测试,发现多个未公布的高危漏洞,受影响的软件包括腾讯QQ、WinZip、微软Office等国内外重要软件,以及部分知名银行的网上服务中使用的控件.该测试结果表明了ActiveX-Fuzzer的有效性和先进性.

【作者单位】：中国科学院研究生院国家计算机网络入侵防范中心;

【关键词】：软件脆弱性 漏洞挖掘 安全性测试 Fuzzing技术 ActiveX控件

【基金】：国家自然科学基金项目(60773135,90718007,60970140)

【分类号】：TP311.52

【正文快照】：

中国科学院大学 University of Chinese Academy of Sciences

后续课程内容

- **第二部分：Web客户端安全**
- **详细讲解XSS跨站、跨站点请求伪造、点击劫持等前端安全。**
- 2.1 OWASP Top Ten
- 2.2 XSS与CSRF
- 2.3 ClickJacking
- 2.4 浏览器与扩展安全
- 2.5 案例分析

[2017秋]Web Security

扫一扫二维码，加入该群。

谢谢大家

刘奇旭

liuqixu@iie.ac.cn

中科院信工所 第六研究室

中国科学院大学
University of Chinese Academy of Sciences