文章编号:1001 - 9081(2008)03 - 0640 - 03

基于漏洞的蠕虫特征自动提取技术研究

李晓冬,李毅超
(电子科技大学 计算机科学与工程学院,成都 610054)
(lixiaodong@uestc.edu.cn)

摘 要:提出一种新的基于漏洞的蠕虫特征,其区别于传统的基于语法或语义分析的技术,对蠕虫攻击的漏洞特征进行分析,将该算法应用于检测系统中。通过实验证明,该检测系统能有效地检测出各种多态变形蠕虫。

关键词:计算机网络;蠕虫;漏洞,特征提取,检测系统
中图分类号: TP309.5 **文献标志码**:A

Research of automatic worm signature generation based on vulnerability

LI Xiao-dong, LI Yi-chao
(*College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*)

Abstract: A new worm signature based on vulnerability was proposed, which was different from traditional methods based on syntactic or semantic. It focused on vulnerability signature that did not change in worm attack. Finally, we applied this method to a detection system and it proved this system can efficiently detect various kinds of worms.

Key words: computer network; worm; vulnerability; signature generation; detection system;

0 引言

蠕虫作为恶意代码的一种,对计算机系统安全和网络安全的威胁日益增加。特别是在开放的计算机网络环境和复杂的应用环境下,多样化的传播途径和复杂的应用环境使网络蠕虫的发生频率增高,潜伏性变强,覆盖面更广,造成的损失也更大。

针对蠕虫的检测也呈现出不断发展的趋势,出现了动态特征码扫描、虚拟机查毒、行为特征识别等方法,但是这些方式方法中最成熟,利用也最广泛的还是传统的基于特征码的检测技术[1],一旦蠕虫被检测程序锁定,那么它其中的某些代码片段就将作为检测程序识别它的特征。

EarlyBird[2]1是最早的一类蠕虫特征检测技术,它对网络数据报进行分类,标记为良性报文和可疑报文,从可疑报文中提取特征,降低检测误报率。HoneyComb[2]234检测技术对EarlyBird进行了改进,使用honeynet来避免分类步骤,收集可疑的数据,搜索最短的相同子串作为特征,它基于主机而非网络流。Autograph[2]检测技术能够更广泛地部署,通过一个预过滤步骤识别可疑扫描行为,它不考虑未过滤的包,因此存在漏报,如E-mail蠕虫、使用列表扫描的蠕虫等。这几种技术所提取的特征都是基于报文的语法,特征由一个单独的连续字节序列组成,无法检测后期出现的多态蠕虫(一种采用不断对自身的执行代码进行变换,完成同一功能但采用不同的方法以此躲避检测程序的蠕虫)。后来,学者们又提出了Polygraph[4]技术,它生成多字节更短的序列作为特征。针对动态蠕虫中的不变量,但目前一些防火墙并不支持联合特征和贝叶斯特征,该算法存在一定的局限性。Hamsa[1]2则是在Polygraph基础上对速度和攻击的恢复力进行了改进。Nemean[5]提出了基于语义分析的特征提取技术,结合协议信息和二进制执行代码信息共同生成蠕虫特征,它使用协议语义信息来分类不同的蠕虫,然后自动学习重建连接会话级的特征。其不足在于对每个应用协议需要详细的协议规范说明。当可疑流中存在干扰时不能生成有效的特征。Christopher提出了基于控制流程图(CFG)的结构化相似性特征提取技术,其不足在于使用SED技术可绕过其检测而且特征的匹配过程计算量大。TaintCheck[1]345和DACODA检测技术采用动态跟踪、关联网络输入与控制流改变之间的关系,来发现可疑的输入,并推断蠕虫的特性,提取蠕虫行为特征。COVERS基于地址空间随机化(ASR)检测关联网络输入,生成特征。最后Mihai模拟了恶意程序的行为,检测并抽象模型相似的代码片段形成蠕虫特征,但其计算量达到了指数级别。

综上所述,目前已经存在多种蠕虫特征提取技术,对蠕虫特征的提取已经成为主动实时防御响应系统的一大热点和难点。但不论是基于语法、基于语义还是基于蠕虫行为的特征提取技术,其核心都是针对各种不同的蠕虫进行分析,最终特征的提取仍来自于蠕虫的攻击报文,即Exploit代码片断。因此,日益复杂的多态、变形等迷惑攻击技术便可轻而易举地躲避以上基于蠕虫特征的检测技术,产生较高的漏报率和误报率。为了解决以上问题,本文提出了一种新的蠕虫特征提取技术,它不再对蠕虫程序进行分析,而是对蠕虫所攻击的漏洞进行分析。因为蠕虫程序可以经过多态、变形技术不断地发生变化,但特定类型蠕虫所攻击的特定漏洞却是不变的。因此基于漏洞的蠕虫特征提取技术能有效地检测各种变形、多态蠕虫,降低漏报率和误报率。最后,本文将该特征提取算法应用于蠕虫检测系统中,并通过实验评估证明该算法的有效性。

1 基于漏洞的蠕虫特征提取算法

1.1 基本定义

1)漏洞。程序中的一个漏洞是一条特定的指令,它被恶

收稿日期:2007 - 09 - 26。

作者简介:李晓冬(1982 -),女,四川成都人,硕士研究生,主要研究方向:计算机网络安全; 李毅超(1968 -),男,四川德阳人,副教授,硕士,主要研究方向:网络安全、嵌入式系统。

意或异常的操作执行将导致系统处于不安全状态,如重写返回地址,死锁,类型安全违反等。指令号又称为漏洞点,用于区分程序中潜在的大量漏洞。我们将程序的不安全执行状态认为是利用(Exploit)状态。

2)漏洞条件。一个漏洞点不能充分、唯一地描述一个单一漏洞。如 strcpy 函数在拷贝源缓冲区到目标缓冲区时,只有当源缓冲区长度小于目标缓冲区时 strcpy 才是安全的。因此,需要提供一个附加的漏洞条件来与漏洞点一起唯一识别一个漏洞。strcpy 的漏洞条件则描述为目标缓冲区至少应该大于等于源缓冲区。处于状态 M_p 的程序 P(在我们的算法中 M_p 在路径跟踪中指定),漏洞条件 c 返回指令 i 的执行是否导致 Explot 状态。将漏洞条件 c 建模为以下函数:

$$c: P \times M_p \times i \rightarrow \{\text{EXPLOIT, BENIGN}\}$$

3)漏洞语言。一个漏洞 V 是一个三元式(P, l, c),P 是程序,l 是漏洞点,c 是漏洞条件。定义一个输入 x 利用一个漏洞 V,当在程序 P 中执行指令 l 后,c 返回 Exploit 时,记为 $P_l(x) \models V$。一个漏洞语言 $L_{p,l,c}$ 由所有利用(攻击)字段集组成:

$$L_{p,l,c} = \{x \in \sum^* \mid P_l(x) \mid = V\}$$

即:当执行语句 l 行时,引发漏洞条件返回 Exploit 的所有输入构成漏洞语言。

4)漏洞特征。漏洞特征是一个匹配函数 MATCH,它不需要运行程序 P,对于输入要么返回 EXPLOIT,要么返回 BENIGN。一个完整的漏洞特征满足以下属性:

$$MATCH(x) = \begin{cases} \text{EXPLOIT when } x \in L_{p,l,c} \\ \text{BENIGN when } x \notin L_{p,l,c} \end{cases}$$

5)特征的健壮性和完整性。定义漏洞特征的完整性为 MATCH:任意 x:$x \in L_{P,l,c} \Rightarrow MATCH(x) = EXPLOIT$。所有 $L_{p,l,c}$ 满足的 x,MATCH 都接受。非完全方案将存在漏报率。定义健壮性为任意 x:$x \notin L_{p,l,c} \Rightarrow MATCH(x) = $ BENIGN,MATCH 不接受任何不属于 $L_{p,l,c}$ 的输入。非健壮的方案将导致误报率。

1.2　特征表示方法

本文选择三种准确的语言类别来表示特征:图灵机特征、符号限制特征和规则表示特征,它们强调准确性和匹配效率之间的一种基本的折中。下面以代码片断为例来分别介绍基于漏洞的特征表示方法。

```
char * GetURL(char input[10]){
  char *url = malloc(4);
  int num=0;
  if(input[num]!='G'&& input[num]!='g')
    return NULL;
  input[num] ='G';
  num++;
  while(input[num] ==")
    num++;
  while(input[num]!=") {
    *url = input[num]; num++; url++;
  }
  printf("%s", url);
  return url;
}
```

图灵机特征　图灵机特征(TM)就是一个程序,它识别漏洞语言。一个图灵机特征 S 由漏洞条件计算出执行程序漏洞点的指令创建。不会到达漏洞点的路径(程序中的一条路径是程序控制流程图中的一条路径)将返回 BENIGN。到达漏洞点并满足漏洞条件的路径将返回 EXPLOIT。TM 特征可以很精确,一个没有误报率的最简单的 TM 特征即是整个程序,如下所示。

```
Char *url = malloc(4);
int num=0;
if(input[num]!='g'&&input[num]!='G')
  return BENIGN;
num++;
while(input[num] ==") num++;
  while(input[num]!=") {
    if(num >=4) return EXPLOIT;
    *url = input[num]; num++; url++;
  }
Return BENIGN;
```

符号限制特征　符号限制特征是一个类似于图灵机特征的 BOOLEAN 规则集。它没有循环,但可能有全局量词和存在量词。与图灵特征不同,匹配(评估)基于输入 x 的符号限制特征最终是会终止的。符号限制特征只近似地静态创建循环,存储区更新等。这样,符号限制特征可能不如图灵机特征精确。如图 3 所示,如果 10 字节的输入以"G"或"g"开头,然后是 0-4 个空格字符,最后是至少 5 个非空字符,符号限制特征值为真,将匹配特征。

符号限制特征表示法如下:

```
(input[0] ='g'V input[0] ='G') Λ
 [(input[1:5] !=") V(input[1] =" Λ input[2:6]!=") V(input[1:
 2] =" Λ input[3:7]!=") V(input[1:3] =" Λ Input[4:8]!=") V
 (input[1:4] =" Λ Input[5:9]!=")]
```

规则表示特征　规则表示特征是三种特征中最弱的一种,在一定环境下可能存在错误率。实际中,因为匹配的高效率性规则表示被广泛地采用。针对给定示例的规则表示特征是[g|G][ ]*[?_]{5,},它可以匹配任何以"g"或"G"开头,然后是 0 或多个空格,最后包含至少 5 个(用{5,}表示)非空字符的输入。

其他特征类型　特征可以被其他语言类型表示。例如,程序的 call/return 语义可以由上下文无关语言准确表示。本文的主要一个贡献在于任何语言类型都可能被使用到特征表示中。使用者可根据具体情况随意地选择适当的表示方法。

1.3　算法描述

计算一个程序 P,漏洞条件 c,利用样本 x,及对应指令跟踪 T 的漏洞特征的算法如下。

算法 1　漏洞特征生成算法

输入:P,二进制程序;T,将输入 x 在 P 上运行的指令跟踪;c,对程序 P 的一个漏洞条件

输出:S,程序 P 针对漏洞条件 c 的漏洞特征

1)在接收到任何利用(攻击)之前预处理程序 P;反汇编程序 P;将汇编转化为中间表示语言 IR。

2)计算关于指令跟踪 T 的一个分片。分片包括到达攻击样本所利用的漏洞点的所有路径。

3)计算特征:计算图灵机特征。当所有路径运行完成时停止;从图灵机特征计算符号限制特征。当所有路径运行完成时停止;从符号限制特征计算规则表示特征。

首先,我们反汇编二进制并识别函数边界,并将反汇编指令转化为一种中间表示(IR),IR 通过将含蓄的硬件副作用清晰化来消除指令歧义。然后,在 IR 上计算利用样本 x 在程序中执行的路径分片,分片基于程序调用图来提取,其结果是一个更小的程序 P',其中每条路径从跟踪 T 中的读入语句开始,

以漏洞点结束。在漏洞特征生成步骤，我们便可以选择 P' 中的任何路径集来计算特征。特征的计算是采用叠加法，先考虑单路径 MEP 特征再以次叠加产生一个多路径 PEP 特征。

2 基于漏洞的蠕虫特征提取检测系统

2.1 系统设计

我们实现了一个原型系统来评估自动特征生成技术。该系统主要包括反汇编引擎、IR 转换、程序分片和特征生成四大部分。重点在于创建规则表示特征，因为它包括了图灵机和符号限制特征的生成。图 4 描述了漏洞特征生成系统的总体架构。

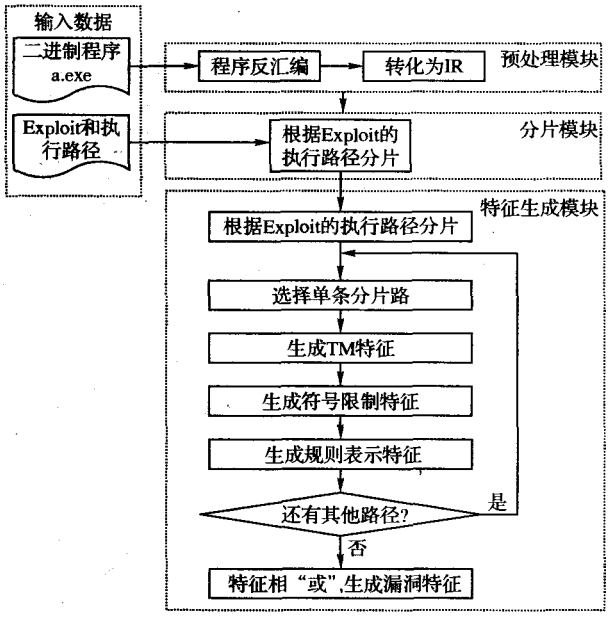

图 1 基于漏洞的特征自动提取系统架构

2.2 系统实现

我们使用 C++ 代码来实现原型系统。使用 CBMB，一个边界模型检查器来帮助构建和解决符号限制来产生规则表示特征。

1）反汇编程序，转为 IR 并获得指令跟踪。二进制程序反汇编是基于 Kruegel 等的方法，然后通过我们自己的转化语言将每条指令转换为近似的 IR 语句。

指令跟踪可以由多数目前的架构包括 x86 通过硬件或软件有效的生成。一个指令跟踪包括指令地址和每条指令执行的任意的操作值。尽管执行的指令数量可能很多，相应的跟踪仍可以被有效地表示。目前，我们使用 Pin 来创建跟踪。

2）特征提取。特征共分三类表示方法：图灵机、符号限制和规则表示。首先图灵机特征即程序本身已由指令跟踪和路径分片进行了分析。然后，使用模型检测器来提取符号限制特征。使用 CBMC 将限制符转换为 C 变量上的限制符。初始化模型检测器为漏洞条件不满足，继续在模型检测器中运行分片。运行结果有 2 种：(1) 漏洞条件确实不满足；(2) 漏洞条件满足，输出一个不满足该漏洞条件的输入反例（即对当前输入的一个修正）。该过程可以递归地应用于所有可能满足条件的输入中。最后，规则表示特征是所有满足条件的输入的"或"。由此获取漏洞特征。

3）实现限制。目前的实现是用于研究自动特征生成的原型。尽管原型在我们的研究环境下正常工作，这里仍有很多的限制。如不支持别名分析，假设没有两个内存地址会是别名关系。除了在符号执行中可能存在的不准确，这个限制不能计算真正的分片。我们采用的基于调用图的分片算法也比真正分片算法的精确度低，最后，IR 转换不能处理浮动指针操作。

3 实验评估

我们对系统进行了实验测试，以 ATPhttpd 为例来描述测试结果。ATPhttpd 是用 C 写的一个 Web 服务器。当一个 HTTP 请求过长时，ATPhttpd 版本 0.4b 存在一个常见的 sprintf 缓冲区漏洞。一个 ATPhttpd 漏洞的利用必须满足以下条件：1）HTTP 请求方法是大小写无关的，必须是"get"或"head"；2）被请求文件名的第一个字节必须是"/"，后面不能再跟一个"/"；3）被请求的文件名不能包"/../"或以此为结尾的子串。4）被请求的文件名必须超过 677 个字符长度。我们使用的利用样本，它由 GET /，后接一个 shellcode，然后是 HTTP 协议字段 HTTP/1.1 组成。在此次实验中，设置给 ATPhttpd 的漏洞条件是：任何指针都不能更改返回地址。

生成的规则表示特征为 [g|G][e|E][t|T][ ]/.{432}//.{3}/.{386}，耗时为 0.1216 s。这个特征接近完美，它识别不区分大小的 get 关键字和任何其他的字节。所有限制的符号（特征中的"/"，"//"）都包含在利用中。将本文的特征与基于蠕虫的特征生成方法进行了比较，后者只是特征的一小部分，不能匹配不同的样本变量，如利用破坏服务而非代码注入的样本变量。

4 结语

本文提出了一种基于漏洞特征自动提取技术的系统基本框架，只需给定一个蠕虫样本，便可以自动提取出比以前更好的特征，可以有效地检测各种多态、变形蠕虫样本。同时，提出了一个广阔的多样化的特征表示法，讨论了三种不同类型的漏洞特征表示法：图灵机、符号限制和规则表示特征。最后，将基于漏洞的特征提取技术应用于一个原型系统中，经实验评估证明，基于漏洞的特征提取技术是区别于以蠕虫样本为中心的特征提取技术的另一个有效方法。

参考文献：

[1] NEWSOME J, SONG D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software[C]// Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS). 2005.

[2] KREIBICH C, CROWCROFT J. Honeycomb-creating intrusion detection signatures using honeypots[J]. Computer Communication Review, 2004, 34(1): 51-56.

[3] KIM H-A, KARP B. Autograph: toward automated, distributed worm signature detection[C]// Proceedings of the 13th Usenix Security Symposium. 2004.

[4] KRUEGEL C, KIRDA E, MUTZ D, *et al*. Polymophic worm detection using structural information of executables[C]// Proceedings of Rapid Advances in Intrusion Detection (RAID), LNCS 3858. Berlin: Springer-Verlag, 2005: 207-226.

[5] BRUMLEY D, LIU L-H, POOSANK P, *et al*. Design space and analysis of worm defense systems[C]// Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security. New York: ACM Press, 2006: 125-137.