文章编号:1001 - 9081(2005)07 - 1562 - 03

模拟攻击测试方式的漏洞检测系统的设计与实现

杨阔朝,蒋 凡
(中国科技大学 计算机科学技术系,安徽 合肥 230027)
(fjiang@ustc.edu.cn)

摘 要:扫描方式的漏洞检测工具往往因为无法得到目标系统的准确信息而无法准确判断目标系统的安全状况,而模拟攻击测试方法可以准确判断目标系统是否存在测试的漏洞。大部分新漏洞发布的同时也会发布相应的测试程序,但是测试程序参数的复杂多样造成了集成的困难,把参数分为DR(运行时决定的类型)、DL(运行时查表决定的类型)和DV(默认值参数)三种类型,利用XML在数据结构描述方面的灵活性解决了这个问题。介绍了一个利用XML描述测试程序接口参数的模拟攻击测试方式的漏洞检测系统。

关键词:漏洞测试;模拟攻击;XML
中图分类号:TP393.08 **文献标识码**:A

Design and realization of vulnerability testing system by imitating attack

YANG Kuo-zhao, JIANG Fan
(*Department of Computer Science and Technology, University of Science and Technology of China, Hefei Anhui* 230027, *China*)

Abstract: Traditional network-based vulnerability scanners can't get very exact information of the target system, they can't identify all of the vulnerabilities in the target system. The way of imitating attack can test the vulnerability exactly. When most of the new vulnerabilities were put forward, the test programs of the vulnerabilities were given together, but the diversity of the parameters of the test program made it difficult to integrate all of the programs. The parameters were classified as DR, DL and DV, and then XML was used to describe the parameters, and a vulnerabilities testing system by imitating attack with XML describing parameters was implemented.

Key words: vulnerability testing; imitating attack; XML

0 引言

随着计算机技术的发展,计算机系统的安全问题逐渐成为关注的焦点。利用已知安全漏洞进行的网络攻击和蠕虫越来越多,网络管理员必须了解网络内各个系统的安全状况,针对存在的安全隐患采取预防措施。

大部分管理员采用安全漏洞扫描工具对整个系统进行扫描来了解系统的安全状况。但这些工具的漏洞识别成功率还有待提高,并且扫描活动会对无线局域网接入点设备和一些打印服务器产生不良影响,还会吞噬大量目标主机的CPU处理能力和带宽[1]。

扫描方式的漏洞检测工具往往因为无法得到目标系统的准确信息而无法准确判断目标系统的安全状况,模拟攻击测试方法可以准确判断目标系统是否存在测试的漏洞。文献[2]提出了利用模拟攻击测试来发现漏洞的思想并开发了相应的测试工具SATAN[3];文献[4]描述并定义了模拟攻击测试方法,并叙述了一个大的公司或组织怎么利用这种方法来保证网络的安全。但是由于漏洞的多样性和复杂性,现有的模拟攻击测试系统发展缓慢[5,6]。

随着计算机技术的发展,安全漏洞涉及的软件和硬件广泛且复杂,针对每一新漏洞建立相应环境并开发相应的测试程序需要相关软硬件的支持,并需要耗费一定的时间。新漏洞公布时其发现者往往会提供相应的测试程序来验证这个漏洞,利用这一程序来测试漏洞不仅减少了开发投入,还减少了新漏洞检测工具的推出时间。

Perl是一种跨平台的语言,可以在Windows,Linux和Unix等许多平台上运行。Perl可以通过exec,system和反引号字符来启动其他的进程,可以通过管道接收和控制其他应用程序的输入输出,所以Perl可以集成不同平台下不同开发者开发的漏洞扫描程序来扫描系统的漏洞信息,扩展漏洞扫描手段。

本文叙述了一个利用Perl语言开发的、利用XML描述测试程序接口参数的模拟攻击测试方式的漏洞检测系统。

1 XML格式接口参数描述

由于存在安全漏洞的系统环境的复杂性和多样性,造成了漏洞测试程序参数的多样性,而且网上公布漏洞验证程序因为开发者不同,参数设置随意性很大。例如:Samba服务器call_trans2open远程缓冲区溢出漏洞测试程序参数包括目标IP、端口号、偏移地址、返回地址、执行模式等多个参数[7],而QPopper 4.0.x Qvsnprintf远程缓冲区溢出漏洞的测试程序只有目标IP和目标系统的用户名、密码三个参数[8],Sadmind漏洞测试程序有目标IP[9],想要执行的命令两个参数。所以,很难设计一个能够包容所有类型参数的数据结构,XML具有数据结构描述灵活的特点,可以用来描述复杂多样的测试程序参数。

收稿日期:2004 - 12 - 22

作者简介:杨阔朝(1976 -),男,河北赵县人,硕士研究生,主要研究方向:网络安全; 蒋凡(1956 -),男,江苏涟水人,教授,博士生导师,主要研究方向:计算机网络、协议与软件测试、信息安全.

通过对网上公布的漏洞测试程序分析和测试,我们对测试程序参数类型进行了统计分析,按照系统需要把参数主要分为三种类型:DR(运行时决定的类型)、DL(运行时查表决定的类型)和 DV(默认值参数)。运行时决定的类型表示此类参数需要在具体的运行环境中决定,例如:被测试系统的目标 IP 必须在运行时决定;运行时查表决定的类型表示运行时要根据具体的系统类型查表得到相应参数,例如:可以进行多系统测试的攻击程序,其系统代号参数需运行时根据系统具体类型决定;默认值参数表示进行漏洞测试时采用默认值即可进行测试,例如:对于可以远程执行 root 命令的漏洞测试程序,统一采用生成一个目录的命令来测试程序执行结果。

在 XML 描述中,对于运行时决定的参数信息采用标准参数名称来替换,最常见的需运行时决定的参数名称见表 1。对于运行时查表决定的参数,需要在 XML 内描述其参数具体内容及其相应的编号,程序调用时要根据参数具体内容查询决定参数编号值。对于默认值参数,描述其参数格式及相应的默认值,运行时程序直接读入默认值即可。

表 1　需运行时决定的参数列表

参数名称	含义	参数名称	含义
HostStartIP	起始 IP	Command	命令
HostEndIP	结束 IP	Offset	偏移地址
Port	端口号	Mode	执行模式
User	用户名	SysType	系统类型
Password	密码		

对测试程序的运行结果判断需要结合测试程序的运行效果来进行确认,根据测试程序的执行效果,不同的返回结果需要执行不同的指令进行判断。根据测试程序返回结果类型,我们把漏洞测试程序分为测试程序内直接返回结果和需要到目标系统察看结果两类。直接返回结果为可以远程得到 Shell,得到文件内容或信息类型的漏洞测试程序,这类漏洞测试程序可以在测试程序返回的结果内直接判断漏洞攻击是否成功。对于需要到目标系统察看结果的类型需要执行描述中相应的指令,根据指令的返回结果来决定程序是否成功。

XML 的漏洞测试程序描述首先要描述此测试程序的漏洞相关信息、漏洞 ID、攻击程序存放路径等漏洞的一些基本信息,再描述漏洞测试程序的接口参数,只要测试程序的参数 XML 描述遵循一定的规范,我们的执行模块和判断模块就可以根据 XML 描述自动执行测试程序并判断执行结果。

Solaris8 下 Sadmind 漏洞攻击程序的 XML 描述如下:

```
…
  <vuldetect name = "Sadmind" >
    <targetos > Soalris8 x86 </targetos >
    <vulid > </vulid >
    <path > </path >
    <para type = "remotecommand" >
      <hostip type = "-h" > HostStartIP </hostip >
      <offset type = "-s" > </offset >
      <command type = "-c" > /bin/mkdir /testvul
      </command >
    </para >
    <result type = "remotecommand" >
      <command > /bin/ls /testvul </command >
      <returnvalue > /testvul </returnvalue >
      <clearact > /rm /testvul < clearact >
    </result >
  </vuldetect >
…
```

2　漏洞检测系统结构

漏洞检测系统由安全漏洞信息服务器和扫描服务器组成。安全漏洞信息服务器负责安全漏洞数据的更新,通过指定的网站更新本地的安全漏洞信息和测试程序,保持最新的安全漏洞信息。管理员可以根据自己网络内的具体系统和应用情况决定需要更新的安全漏洞信息,可以根据测试模块 XML 描述规则增加漏洞测试程序。信息服务器提供查询接口,扫描服务器可以通过接口查询特定系统、特定应用和特定环境的漏洞信息及其检测方法。

扫描服务器可以根据网络划分设置在不同的网段内,负责本网段的扫描工作。扫描服务器负责测试程序的执行与结果判断,根据扫描网段内的系统类型和应用类型,扫描服务器向信息服务器查询相应的漏洞信息,接收其发送的相关的漏洞及其测试信息,并从安全漏洞信息服务器查询相应的测试模块下载到本地,根据测试信息描述进行漏洞测试,测试完成后把返回信息传送回服务器端。服务器对传回的信息进行整理,把漏洞测试结果以报告方式显示给管理员。

除了漏洞测试程序,本系统可以集成其他的安全检查工具,例如:查询未知 root 账户的工具,查询无密码用户的工具等。扫描服务器可以利用配置文件中设置的用户名和密码自动进入目标系统执行相应的测试动作并返回测试结果。

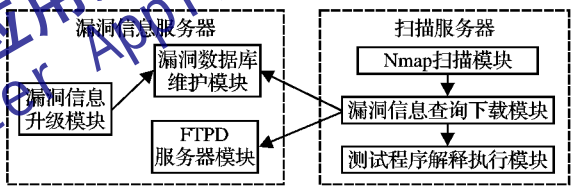

图 1　系统结构

2.1　安全漏洞信息服务器实现

漏洞信息服务器对漏洞信息进行维护,每个漏洞以单独 XML 文件描述其详细信息,按照漏洞存在的系统类型及漏洞本身类型存放于不同目录下。漏洞 XML 文件描述的测试模块信息与 FTP 服务器的测试模块相结合,便于测试模块的下载。同时利用关系数据库存放需要快速查询的数据项,例如:漏洞名称,CVE[10] 编号,BugtraqID[11] 等。通过关系数据库提高漏洞定位速度,可以先查询到详细 XML 描述文档,再查询漏洞具体细节。

对于第三方开发的,可以对特定漏洞进行测试或攻击的模块,存放于相应漏洞目录下,根据漏洞 XML 描述的数据定位到具体的程序,利用 FTP 接口下载。为了保证模块信息的唯一性,由多个文件组成的模块需压缩成一个单独的文件存放,Windows 系统采用 RAR 格式压缩成自解压的 EXE 文件,Linux 和 Unix 系统压缩成 tar 文件包。漏洞 XML 描述节包含模块位置、名称、展开命令、执行步骤及参数描述。

管理第三方模块的用 Perl 实现的 FTPD 服务端程序如下:

```
use strict; use IO: : Socket; my $ port = shift || 7780;
my $ sock = IO: : Socket: : INET - > new( LocalPort => $ port,
Listen => 1) or die;
while( 1)
{
  next unless my $ session = $ sock -> accept;
  next unless < $ session > = ~ /ftpbyustc/;
  my $ action = < $ session >;
  my $ filename = < $ session >;
```

```
    chomp( $ filename);
    if ( $ action = ~ /send/)
    {
      open( FH, " >". $ filename);
      my $ buf;
      while( $ session -> read( $ buf, 8192) >0)
      { print FH $ buf; };
      close ( FH);
    }
    elsif ( $ action = ~ /receive/)
    {
      open( FH, $ filename);
      my $ buf;
      while( <FH >)
      { print $ session $ _; };
      close( FH);
    };
};
```

2.2 扫描服务器实现

扫描服务器完成对网段环境的扫描、漏洞信息查询和测试工作。它直接调用 Nmap 对目标系统进行扫描,得到目标系统的详细信息,根据此详细信息向信息服务器查询相应的漏洞信息。为了减少网络流量,防止不必要的信息查询,扫描服务器并不是每次都向信息服务器查询所有的漏洞信息。因为扫描服务器是运行在固定的系统环境内,当扫描服务器安装时,首先收集当前网络环境的软件和配置信息,根据配置信息向服务器查询相关的漏洞信息并存储,根据漏洞细节查询系统信息判断是否存在安全隐患,每次扫描服务器只向服务器查询比本地信息新的安全漏洞信息并检查本地网络内各个系统的安全状况。

漏洞模拟攻击测试工作由 XML 解释模块和测试执行模块完成。XML 解释模块利用 Perl 中的 XML 模块实现对接收到的 XML 格式漏洞信息的分析和测试数据的提取。服务器端返回的数据为漏洞的 XML 格式描述,每个漏洞描述包含漏洞的基本信息及测试信息。利用 XML::DOM 分析出 XML 格式漏洞数据中的测试信息部分,传递给测试执行模块进行处理。

Perl 对 XML 数据解析程序如下:

```
...
foreach my $ vul ( $ vuls -> getElementsByTagName( 'vulid')){
  $ vulname = $ vul -> getElementsByTagName( 'vulname')
      -> item(0) -> getFirstChild -> getNodeValue;
  $ cveid = $ vul -> getAttribute( 'cve');
  $ bugtraqid = $ vul -> getAttribute( 'bugtraqid');
...
```

测试执行模块根据测试程序参数的 XML 描述,并根据目标系统具体情况生成相应的参数字符串,然后根据测试程序类型通过相应的调用方式实现测试程序的参数传递和自动执行,根据参数类型调整相应的参数值进行重复测试,最后根据测试模块的结果描述检查结果的执行情况。

各个漏洞的模拟攻击测试模块利用 Net::FTP 模块开发的 FTP 客户端实现模块自动下载,Perl 实现的 FTP 客户端如下:

```
...
my $ argnum = $#ARGV;
use strict;
use IO::Socket;
my $ host = shift;
my $ port = shift;
my $ filename = shift;
my $ name = $ filename;
if ( $ argnum == 3 )
{
  if ( $ filename = ~ /^(\S+)\/(\S+) $ /)
  { $ name = $ 2;};
}
$ name = shift || $ name;
my $ action = shift || "send";
my $ s = IO::Socket::INET -> new( " $ host: $ port") || die
$ !;
$ s -> print( "startbyustc\n");
$ s -> print( $ action. "\n"); $ s -> print( $ name. "\n");
if ( $ action = ~ /receive/)
{ open( OUTPUT, " >". $ filename) ||
  die $ !; while ( < $ s >){ print OUTPUT $ _;}
  close( OUTPUT);}
elsif ( $ action = ~ /send/)
{ open( INPUT, $ filename) || die $ !;
  while ( <INPUT >){ print $ s $ _;} close( INPUT);}
print file transfer complete;
...
```

3 系统应用

利用 XML 描述的测试模块接口,可以兼容大部分漏洞的测试程序,实现漏洞攻击测试的自动进行。它可以方便地集成最新的漏洞攻击测试程序模块,实现对最新漏洞的检测。

基于 Perl 的系统可以让管理员针对自己的网络环境进行系统的定制,或把它移植到自己的系统上。可扩充的模块接口可以让管理员把自己系统上已有的模块或自己开发的模块集成到现在的客户端,利用现有资源来增强系统的安全。

结合具体硬件和软件系统情况,我们已经对 Windows, Linux, FreeBSD, SCO Unix 和 Solaris 的 x86 与 SPARC 结构下从 1999 年至今的危害较大的漏洞进行了集成。本系统虽然可以实现对最新漏洞的准确检测,但是因为漏洞测试的复杂性,有的第三方模块的执行及结果判断需要管理员的参与,没能实现漏洞模拟攻击测试的完全自动化。

4 结语

安全漏洞检测需要多方的参与才有可能跟踪各种系统下最新漏洞的情况,单凭任何一方都难于实现对所有环境漏洞的跟踪和测试,应当按照开源软件的模式由相应的组织维护,有广泛的志愿者参与,充分利用不同志愿者拥有的不同测试环境来实现对所有漏洞的测试方法研究。管理员只对与自己网络内系统有关的漏洞测试程序感兴趣,他们可以利用自己的软硬件环境对网上公布的漏洞测试程序进行测试和修改,形成更准确的漏洞检测模块。

参考文献:

[1] ANDRESS M. Network vulnerability assessment management[EB/OL]. http://www.nwfusion.com/reviews/2004/110804rev.html. Network World, 2004-08-11.

[2] FRRMER D, VENEMA W. Improving the Security of Your Site by Breaking Into it[EB/OL]. http://www.porcupine.org/satan/admin-guide-to-cracking.html, 1993-12.

[3] SATAN[EB/OL]. http://www.porcupine.org/satan/, 2004-06.

[4] LAYTON SR TP. Penetration Studies - A Technical Overview[EB/OL]. http://rr.sans.org/, 2002-05.

[5] SARA[EB/OL]. http://www-arc.com/sara/, 2004-09.

[6] Nmap[EB/OL]. http://www.insecure.org, 2004-06.

[7] Samba 服务器 call_trans2open 远程缓冲区溢出漏洞[EB/OL].

(下转第 1567 页)

携带的权限是和时间相关联的。具体的含义为:在时间段 I 内,票据 T 中有效的权限。令牌的权限为:tp(t,i),它是令牌所携有的权限。若有 t ⊢T,则有:tp(t,i) ∈ TP(T,I)。

定义 5　定义已经发布的有效票据的集合为 E(t),其中 t 为一个时间点,则可知:E(t) ⊂ S^+, E(t)的具体描述如下:

$$E(t)=\{\ grant(i,\ h,\ t1,\ I,\ ts1,\ id)\in S^+ \mid t\in[I]\ \&(revoke(s,ts2,id)\in S^-\rightarrow ts2>t)\}$$

其含义为:一个票据在给定的时间点 t 是否有效的验证条件是:t 是否在票据的有效期 I 内,并且如果有撤销票据存在的话,则撤销票据的时间点 t2 应该大于 t。由符合这些条件的所有票据构成了与时间点 t 关联的票据有效集 E(t)。

定义 6　如果有 s1,s2 ∈ S^+,即 s1 = grant(i1, h1, t1, I1, ts1, id1) ∈ S^+,s2 = grant(i2, h2, t2, I2, ts2, id2) ∈ S^+,定义票据之间的支持关系为:如果有 s1 ∈ E(t2),且有 TP(s1) ⊂ TP(s2),则有 s1∏s2,其含义为票据 s1 所携带的权限可以被票据 s2 中的权限涵盖。

定义 7　定义票据库 D 中的票据链 chain 为关系∏的闭集。

2.2　权限的计算

票据权限计算的形式化描述如下,具体的实现可以结合采用的编程语言再加以处理。

Holds(S,H)用于计算在给定的时间点 H,票据集 S 内有效票据的集合。

1) Holds(S, H)←E(H),not[revoke(s,ts,id),ts≤H], C∈E(H),check(C, H);

2) check(C, H)←C = grant(i, h, t, I, ts, id), I = [Istart, Iend], Istart≤H≤Iend, SourceOfAuthority(C)。

SourceOfAuthority(C)用于验证票据 C 中的授权链是否有效。Verify(S,Q,P,H)用于判定在给定的票据集 S 和时间点 H,票据的持有者 Q 是否具有权限 P。

1) Verify(S,Q,P,H)←C = grant(i, h, t, I, ts, id), C∈Holds(S,H), effective(C,H)。

2) effective(C,H)←grant(i, h, t, I, ts, id),rooted(C), ts≤H, not[revoke(s,ts1,id),ts1≤H]

3) rooted(C)←chain(C1,C), C1 = grant(i1, h, t1, I1, ts1, id1)

4) chain (C,C)

5) chain (C1,C2)←support (C1,C3), chain (C3,C2)

6) supports(C1,C2)← C1 = grant(i1, h, t1, I1, ts1, id1),C2 = grant(i2, h, t2, I2, ts2, id2),not[revoke(s,t3, id3),t3≤t2]

Tickets(S,Q,H)用于计算在给定的票据集 S 和时间点 H,主体 Q 所持有的票据的集合。

1) Ticket(S,Q,H)←C = grant(i, Q, TOKENS, I, ts, id), C∈Holds(S,H),examine(C,TOKENS)

2) examime (C, TOKENS)← TOKENS not null, t∈TOKENS, t ⊢C

Privilege(S,Q,H)用于计算在给定的票据集 S 和时间点 H,主体 Q 持有的权限的集合。

1) Privilege(S,Q, H)←C = grant(i, Q, t, I, ts, id),C∈Tickets(S,H,Q),P∈TP(C,H),not[revoke(s,ts1,id),ts1≤H], validate(Q,P,H,I)

2) validate(Q,P,H,I)←Verify(S,Q,P,H),checktime(H,I)

3) checktime(T,I)←Istart≤H≤Iend

3　结语

在异构信息系统下,各个系统会采用不同的访问控制策略,为了在异构的系统之间实现安全策略的协同,本文提出了一个基于票据的安全策略协同模型,并重点用形式化的语言对其进行了描述,最后完成了对票据的权限计算。扩展开来,在票据中也可以携带授权代理的权限,这样获得此权限的节点,就可以代理授权锚发放票据,形成树状的授权模型。

参考文献:

[1] ROSENBERRY W, KENNEY D, FISHER K. Understanding DCE [M]. O'Reilly and Associates Inc, 1993.

[2] Final Proposed Draft Amendment on Certificate Extensions(v6)[S]. ITUI, SO/IEC, 1999.

[3] ROSENTHAL A, WILLIAMS J, HERNDON W, *et al*. A fine grained access control model for object-oriented DBMSs, in Database Security[A], VIII: Status and Prospects [C]. Elsevier, Amsterdam, 1994.319 - 334.

[4] HAYTON RJ, BACON JM, MOODY K. Access Control in an Open Distributed Enviroment[A]. Proceeding of IEEE Symposium on Security and Privacy[C]. Oakland, CA, 1998.3 - 14.

[5] HAMILTON G, CATTELL R, FISHER M. JDBC Database Access With Java: A Tutorial and Annotated Reference[M]. Addison-Wesley, 1997.

[6] JAJODIA S, KOGAN B. Integrating an object-oriented data model with multilevel security[A]. Proceedings of the IEEE Symposium on Research in Security and Privacy[C], 1990.76 - 85.

[7] JONSCHER D, DITTRICH KR. Argos - A configurable access control system for interoperable environments, in Database Security [A]. IX: Status and Prospects[C]. Chapman and Hall, London, 1995.43 - 60.

(上接第 1564 页)

http://www.nsfocus.net/index.php?act=sec_bug&do=view&bug_id=4652&keyword=Samba, 2004 - 06.

[8] QPopper4.0.xQvsnprintf 远程缓冲区溢出漏洞[EB/OL]. http://www.nsfocus.net/index.php?act=sec_bug&do=view&bug_id=4530&keyword=qpopper, 2004 - 06.

[9] Sadmind 漏洞[EB/OL]. http://www.nsfocus.net/index.php?act=sec_bug&do=view&bug_id=5404&keyword=sadmind, 2004 - 06.

[10] CVE Homepage[EB/OL]. http://cve.mitre.org/, 2004 - 09.

[11] Security Focus Homepage[EB/OL]. http://www.SecurityFocus.com, 2004 - 09.

[12] LOWERY J. Penetration Testing: The Third Party Hacker[EB/OL]. http://rr.sans.org/, 2002 - 02.

[13] SHARM A, MARTIN JR, ANANAD N, *et al*. Ferret: A Host Vulnerability Checking Tool[A]. Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04)[C], 2004.

[14] KOTENKO I. Active Vulnerability Assessment of Computer Networks by Simulation of Complex Remote Attacks [A]. Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing (ICCNMC03)[C], 2003.