开发记录

卷一

Dolphin

EDITED BY XIAOQIANG JIANG

HTTP://JIANGXIAOQIANG.GITHUB.COM/

Version 23:44, June 12, 2018

Contents

这里记录的是一些比较杂乱的笔记，绝大多数文字皆来源于网络，不是自己的原创，这里没有高深的算法，没有宏伟的技术及系统架构，只是一些平时工作中遇到的一些问题，和解决问题的思路以及所采用的方案。由于平时工作时还没有遇到前人没有遇到过的问题需要自己发明方去解决 (其实真的遇到估计也是没辙)，所以绝大部分内容是为了避免再遇到同样的问题时，又需要到处去搜寻，索性将之记录下来，以便于下次可以快刀斩乱麻，迅速解决问题。

Tool

1. Widgets

1.1 ssh(Secure Shell)

关闭不活动的 ssh 会话, 使用 w 命令来识别出不活动或者是空闲的 ssh 会话，使用 pstree 命令来获取空闲会话的 PID，就是使用 kill 命令来关闭会话了。

1.1.1 安全

修改默认端口 (Change Default Port)

修改默认端口无法阻止专业的攻击, 因为修改 ssh 默认端口后, 使用 nmap 工具一样可以识别出来 OpenSSH 服务. 但是修改默认端口为随机端口后有一点好处就是可以避免被脚本自动扫描到来尝试登录 (爆破).

免密登陆 (Login Without Password)

SSH 服务如果在公网上，非常容易受到攻击，特别是弱口令扫描，字典扫描。保护服务器免受攻击，可以使用 SSH 密钥，禁用口令认证，如果不能做到这一点，务必使用强壮的密码。还可以设置登陆 IP 白名单。更改服务器 ssh 端口 (基本上没有效果，调整了之后可以使用 nmap 轻松扫描到)。使用 snort、ossec 等开源的入侵检测设备保护服务器。免密登录需要注意的是，.ssh 文件夹下的 authorize_key 文件的权限需要是 600。而.ssh 文件夹的权限需要是 700。调整/etc/ssh/sshd_config 配置文件:

```
RSAAuthentication yes
PubkeyAuthentication yes
```

```
AuthorizedKeysFile      .ssh/authorized_keys
```

拷贝公钥到服务器：

```
ssh -p 22 pi@192.168.31.25 'mkdir -p .ssh && cat >> .
        ssh/authorized_keys' < ~/.ssh/id_rsa.pub
# 如果本地没有生成过 ssh key
# 使用如下命令生成 ssh key
ssh-keygen -t rsa -C "a@gmail.com"
```

调整文件夹权限：

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

在 Mac OS X 中，有时自动登陆需要反复输入密码，解决问题的方法是可以配置 Serria 记住密码：

```
usekeychain yes
```

有时在 Ubuntu 下使用 ssh 也需要输入密码 (Enter passphrase for key id_rsa), 此时也可以使用 keychain:

```
#安装 keychain
sudo apt install keychain
/usr/bin/keychain ~/.ssh/id_rsa
```

keychain 会起一个 ssh-agent，后来登录的人通过设置环境变量使用同一个 ssh-agent. 添加的只是在当前 shell 下生效, 如果需要每次都生效.

1.1.2 ssh 连接慢 (Connect Slow)

在使用 SSH 时，每次连接建立都相当的慢啊，要 20 秒以上才能够登录上去，严重影响工作效率。在服务端的/etc/ssh/sshd_config 文件中，修改配置，将 GSSAPI-Authentication 默认设置为关闭即可，配置文件修改后需要重新启动 sshd 守护进程配置才能够生效：

```
GSSAPIAuthentication  no
# 重启 sshd 守护进程
sudo service sshd restart
```

GSSAPI (Generic Security Services Application Programming Interface) 是一套类似 Kerberos 5 的通用网络安全系统接口。该接口是对各种不同的客户端服务器安全机制的封装，以消除安全接口的不同，降低编程难度。但该接口在目标机器无域名解析时会有问题。使用 strace 查看后发现，ssh 在验证完 key 之后，进行 authentication gssapi-with-mic，此时先去连接 DNS 服务器，在这之后会进行其他操作。连接慢也可以关闭 DNS 反向解析，在 linux 中，默认就是开启了 SSH 的反向 DNS 解析, 这个会消耗大量时间，因此需要关闭 [**book-wiresharkanalysis**]。

```
UseDNS=no
```

1.1.3 Permission denied (publickey)

在 ssh 登陆机器时，提示 Permission denied (publickey)。可以尝试的方法，在连接命令中加上-vvv 参数，观察详细的调试输出:

```
ssh -p 2222 -vvv hldev@10.0.0.22
```

设置文件夹对应的权限：

```
sudo chmod 700 .ssh
sudo chmod 600 .ssh/authorized_keys
```

登陆服务器观察相应的日志输出：

```
tail -f /var/log/auth.log
```

是不是服务器关闭了密码登陆？只能使用公钥认证登陆。后面检查确实如此，服务器端为了安全考虑，关闭了基于密码登录的方式，但是需要登录的主机并没有将自己的公钥拷贝到服务器上。所以服务器直接提示了 Permission denied (publickey)，解决的办法就是在服务器端暂时开启密码登录，在/etc/ssh/sshd_config 中调整配置：

```
#允许使用基于密钥认证的方式登陆
PubkeyAuthentication yes
```

1.1.4 Session 时间

在使用 ssh 的过程中，经常会遇到一会儿没有操作就自动断开了，不是非常方便。

ClientAliveInterval

修改/etc/ssh/sshd_config 配置文件 ClientAliveInterval 300（默认为 0），参数的是意思是每 5 分钟，服务器向客户端发一个消息，用于保持连接，使用 service sshd reload 让其修改后生效。如果发现还是有问题，可以试着把 300 设置小一点，例如 60。

ClientAliveCountMax

另外, 至于 ClientAliveCountMax, 使用默认值 3 即可.ClientAliveCountMax 表示服务器发出请求后客户端没有响应的次数达到一定值, 就自动断开。

ControlPersist 4h

在./ssh/config 中添加一行：

```
1 ControlPersist 4h
```

When used in conjunction with ControlMaster, specifies that the master connection should remain open in the background (waiting for future client connections) after the initial client connection has been closed. If set to no, then the master connection will not be placed into the background, and will close as soon as the initial client connection is closed. If set to yes or 0, then the master connection will remain in the background indefinitely (until killed or closed via a mechanism such as the “ssh -O exit”). If set to a time in seconds, or a time in any of the formats documented in sshd_config, then the backgrounded master connection will automatically terminate after it has remained idle (with no client connections) for the specified time[1]. 现在你每次通过 SSH 与服务器建立连接之后，这条连接将被保持 4 个小时，即使在你退出服务器之后，这条连接依然可以重用，因此，在你下一次（4 小时之内）登录服务器时，你会发现连接以闪电般的速度建立完成，这个选项对于通过 scp 拷贝多个文件提速尤其明显，因为你不在需要为每个文件做单独的认证了。

1.1.5 代理转发

本地转发 Local Forward

将本地机 (客户机) 的某个端口转发到远端指定机器的指定端口. 工作原理是这样的, 本地机器上分配了一个 socket 侦听 port 端口, 一旦这个端口上有了连接, 该连接就经过安全通道转发出去, 同时远程主机和 host 的 hostport 端口建立连接. 可以在配置文件中指定端口的转发. 只有 root 才能转发特权端口。应用实例可以参看**??**。

[1] http://man.openbsd.org/ssh_config.5

```
# Mac OS X 端口转发
/usr/bin/ssh -g -L 2222:10.10.30.1:22222 127.0.0.1
# 接口服务器设置代理转发
ssh -g -L 7805:192.168.250.100:7805 10.10.1.32
```

有时在本地转发会遇到一些问题，比如 Connection Refused。首先要确定本地要运行有 ssh 服务端，使用如下命令启动 sshd：

```
# Mac OS X 启动 sshd
sudo /usr/bin/sshd
# Ubuntu 启动 sshd
sudo /etc/init.d/ssh
```

启动 SSHD 的时候系统提示:Could not load host key: /etc/ssh/ssh_ed25519_key。新版的 opensshd 中添加了 Ed25519 做签名验证，而之前系统里没这个算法的证书，所以办法也很简单新生成下证书即可。

```
sudo ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key
sudo ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key
sudo ssh-keygen -t ecdsa -f /etc/ssh/
        ssh_host_ecdsa_key
ssh-keygen -t ed25519 -f /etc/ssh/
        ssh_host_ED25519_key
```

1.1.6 ssh 查看日志

如果需要查看 ssh 日志，可在/etc/ssh/sshd_config 配置日志输出：

```
SysLogFacility LOCAL7
```

Facility: 设施，是 rsyslog 引入的概念，从功能或者程序上对日志进行分类，并由专门的工作负责记录相对应的信息，分类有：

auth(授权),authpriv,cron(定时任务),daemon(守护进程),kern(内核),lpr,mail(邮件相关),mark,news,security(安全),syslog,user,uucp,local0,local7。

多线程，多协议（udp/tcp/ssl/tls/relp）；mysql,pgsql,oracle 等多种关系数据中，强大的过滤器，可实现过滤系统信息中的任意部分。自定义输出格式。适用于企业级别日志记录需求。ssh 配置文件配置后，在/etc/rsyslog.conf 文件中作如下配置：

```
LOCAL7.*    /var/log/sshd.log
```

1.2 VisualVM

使用 nmap 扫描 1099 端口，看 jstatd 是否生效。

1.3 Tool Set

1.3.1 ECS(Elastic Compute Service)

配置内网 ECS 端口映射规则, 在云服务器 ECS–> 网络和安全–> 安全组–> 配置规则–> 添加安全组规则.

1.3.2 shadowsocks

启动 ss 服务器:

```
#可以查看启动日志
ssserver -c /home/ec2-user/shadowsocks.json
#后台启动
ssserver -c /home/ec2-user/shadowsocks.json -d start
```

Shadowsocks 客户端操作:

```
sudo apt-get install python-pip
sudo apt-get install python-setuptools m2crypto
#安装 Shadowsocks(Ubuntu/Fedora)
pip install shadowsocks
#前台启动
#可以看到实时的日志输出
#关闭终端后代理断开
sslocal -c /etc/shadowsocks/shadowsocks.json
#后台启动
sslocal -c /etc/shadowsocks/shadowsocks.json -d start
```

1.3.3 youtube-dl

YouTube 视频一般是不能下载的, 但是是国内访问 YouTube 比较慢, 经常卡顿, 所以可以使用 youtube-dl 工具下载 YouTube 视频:

```
#下载默认的视频格式
youtube-dl https://www.youtube.com/watch?v=
        SnHxKQiXrFU
#查看所有视频格式
youtube-dl -F https://www.youtube.com/watch?v=
        SnHxKQiXrFU
#下载指定清晰度的视频
#137 为指定视频格式的编码 format code
youtube-dl -f 137 https://www.youtube.com/watch?v=
        SnHxKQiXrFU
```

查看 YouTube 所有格式视频输出效果如图所示:

```
[youtube] eq3KiAH4IBI: Downloading webpage
[youtube] eq3KiAH4IBI: Downloading video info webpage
[youtube] eq3KiAH4IBI: Extracting video information
[info] Available formats for eq3KiAH4IBI:
format code  extension  resolution note
249          webm       audio only DASH audio   55k , opus @ 50k, 21.82MiB
171          webm       audio only DASH audio   80k , vorbis@128k, 31.52MiB
250          webm       audio only DASH audio   90k , opus @ 70k, 30.13MiB
140          m4a        audio only DASH audio   99k , m4a_dash container, mp4a.40.2@128k, 40.92MiB
251          webm       audio only DASH audio  126k , opus @160k, 44.80MiB
278          webm       256x144    144p   64k , webm container, vp9, 13fps, video only, 16.01MiB
242          webm       426x240    240p  114k , vp9, 25fps, video only, 18.26MiB
160          mp4        256x144    144p  126k , avc1.42c00c, 13fps, video only, 47.61MiB
243          webm       640x360    360p  218k , vp9, 25fps, video only, 36.78MiB
134          mp4        640x360    360p  248k , avc1.4d401e, 25fps, video only, 49.41MiB
```

Figure 1.1: 查看所有格式视频

1.3.4 Jenkins

启动 Jenkins 提示 Job for jenkins.service failed because the control process exited with error code. See "systemctl status jenkins.service" and "journalctl -xe" for details。多半是由于 Java 未安装或者安装后未指定 Java 路径。打开文件：

```
#配置文件中指定 Java 路径
vim /etc/rc.d/init.d/jenkins
#启动 Jenkins
#在 RedHat 中通过 rpm 包安装时有效
service jenkins start
```

配置文件中指定 Java 路径如图所示。

Figure 1.2: 指定 Jenkins 的 Java 路径

1.3.5 OpenVPN

OpenVPN 从 2001 年开始开发，使用的是 C 语言。此处使用的 OpenVPN 版本是 2.4.1。如果使用 Mac 下的 brew 工具安装，则 OpenVPN 目录在：/usr/local/Cellar/openvpn/2.4.1，OpenVPN 的配置文件在：/usr/local/etc/openvpn。目前 OpenVPN 能在 Solaris、Linux、OpenBSD、FreeBSD、NetBSD、Mac OS X 与 Microsoft Windows 以及 Android 和 iOS 上运行，并包含了许多安全性的功能。此处的服务器使用的是 CentOS 7.3，客户端包含 Fedora 24、Ubuntu 14.04、Ubuntu 16.04、Window 7、Windows 10。在 OpenVPN 网络中查看存活的主机：

```
nmap -A -T4 10.0.0.*
```

安装 (Install)

安装基础包：

```
sudo yum -y install openssl openssl-devel lzo openvpn
         easy-rsa --allowerasing
#手动安装
wget -c https://swupdate.openvpn.org/community/
        releases/openvpn-2.4.1.tar.gz
tar -zxvf openvpn-2.4.1.tar.gz
./configure
make && make install
```

LZO 是致力于解压速度的一种数据压缩算法，LZO 是 Lempel-Ziv-Oberhumer 的缩写。这个算法是无损算法，参考实现程序是线程安全的。实现它的一个自由软件工具是 lzop。最初的库是用 ANSI C 编写、并且遵从 GNU 通用公共许可证发布的。现在 LZO 有用于 Perl、Python 以及 Java 的各种版本。代码版权的所有者是

Markus F. X. J. Oberhumer。如果没有安装 easyrsa 工具, 使用如下命令安装:

```
#下载 easy-rsa 源码
wget -c https://github.com/OpenVPN/easy-rsa/archive/
        master.zip
#目录/etc/openvpn/easy-rsa/easyrsa3 下拷贝默认配置
cp var.example vars
```

调整配置

修改证书生成的配置文件:

```
set_var EASYRSA_REQ_COUNTRY     "CN"
set_var EASYRSA_REQ_PROVINCE    "Chongqing"
set_var EASYRSA_REQ_CITY        "Chongqing"
set_var EASYRSA_REQ_ORG "Three people"
set_var EASYRSA_REQ_EMAIL       "jiangtingqiang@gmail
        .com"
set_var EASYRSA_REQ_OU          "My Organizational"

set_var EASYRSA_KEY_SIZE        2048
```

生成根证书

生成 OpenVPN 的根证书:

```
#清除之前生成的证书, 重新生成;
./easyrsa clean-all
#生成 root 根证书
#证书路径:/etc/openvpn/easy-rsa/easyrsa3/pki/ca.crt
./easyrsa build-ca
```

生成服务器端证书

生成服务器端证书命令如下:

```
# 指定服务端证书的文件名为 server, 可以任意改动
./easyrsa build-server-full server
```

生成客户端证书

生成客户端证书端步骤如下：

```
./easyrsa build-client-full jiangxiaoqiang
```

PKI：Public Key Infrastructure 公钥基础设施。生成请求：

```
./easyrsa gen-req dolphinfedora
```

输入 PEM 验证码。PEM - Privacy Enhanced Mail, 打开看文本格式, 以"——BEGIN..." 开头, "——END..." 结尾, 内容是 BASE64 编码。Apache 和 *NIX 服务器偏向于使用这种编码格式. 签约：

```
#切换到服务端生成 rsa 的目录
#导入 req
./easyrsa import-req ~/client/easyrsa/easy-rsa-master
        /easyrsa3/pki/reqs/dolphinfedora.req
        dolphinfedora
#用户签约，根据提示输入服务端的 ca 密码
./easyrsa sign client dolphinfedora
```

PKI：Public Key Infrastructure 公钥基础设施。输入 PEM 验证码。PEM - Privacy Enhanced Mail, 打开看文本格式, 以"——BEGIN..." 开头, "——END..." 结尾, 内容是 BASE64 编码. 查看 PEM 格式证书的信息:openssl x509 -in certificate.pem -text -noout。Apache 和 *NIX 服务器偏向于使用这种编码格式. 服务端生成的文件有：

文件名称	说明 (Purpose)	位置
ca.crt	根证书 (Root CA certificate) 件	Server+All Clients
reqs/server.req		
reqs/dolphin.req		
private/ca.key	根证书私钥 (Root CA key)	key signing machine only
private/server.key		
issued/server.crt	服务器证书 Server Certificate	server only
issued/dolphin.crt		
dh.pem	Diffie Hellman parameters	server only

客户端生成的文件有：

序号	名称
private/dolphinclient.key	
reqs/sdolphinclient.req	

拷贝出客户端证书文件：

```
cp easyrsa/easy-rsa-master/easyrsa3/pki/ca.crt ~/
        dolphinfedora/
cp easyrsa/easy-rsa-master/easyrsa3/pki/issued/
        dolphinfedora.crt ~/dolphinfedora/
cp ~/client/easyrsa/easy-rsa-master/easyrsa3/pki/
        private/dolphinfedora.key ~/dolphinfedora/
```

启动 OpenVPN：

```
sudo openvpn server.conf
# Mac 下启动 OpenVPN
sudo /usr/local/Cellar/openvpn/2.4.1/sbin/openvpn /
        usr/local/etc/openvpn/client.conf
# 需要以后台交互方式启动时
screen sudo openvpn client.conf
```

客户端端配置如下：

```
client          #指定当前 VPN 是客户端
dev tun         #必须与服务器端的保持一致
proto udp       #必须与服务器端的保持一致
#指定连接的远程服务器的实际 IP 地址和端口号
remote 192.168.1.106 1194
#断线自动重新连接
#在网络不稳定的情况下（例如：笔记本电脑无线网络）非常
        有用
resolv-retry infinite
nobind          #不绑定特定的本地端口号
persist-key
persist-tun
ca ca.crt       #指定 CA 证书的文件路径
cert client1.crt        #指定当前客户端的证书文件路径
key client1.key     #指定当前客户端的私钥文件路径
```

```
ns-cert-type server       #指定采用服务器校验方式
#如果服务器设置了防御 DoS 等攻击的 ta.key
#则必须每个客户端开启；如果未设置，则注释掉这一行；
tls-auth ta.key 1
comp-lzo                  #与服务器保持一致
#指定日志文件的记录详细级别，可选 0-9，等级越高日志内容越详细
verb 3
```

配置 ns-cert-type(Netscape Cert Type) 指定为 server 主要是防止中间人攻击(Man-in-the-Middle Attack)。在服务端做如下配置：

```
nsCertType server
```

生成 Diffle Hellman 参数

生成 Diffle Hellman 参数命令如下:

```
./easyrsa gen-dh
```

初始化配置文件

初始化配置文件如下:

```
#拷贝示例配置到配置目录
cp /usr/share/doc/openvpn-2.4.4/sample/sample-config-files/server.conf /etc/openvpn
```

拷贝 server.crt/ca.cert 等文件到配置目录.

生成 ta.key

使用如下命令生成 ta.key:

```
openvpn --genkey --secret ta.key
```

启动服务端:

```
openvpn --config /etc/openvpn/server.conf
```

1.3.6 fastDFS

启动 tracker server:

```
/usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
```

启动 storaged server:

```
/usr/bin/fdfs_storaged /etc/fdfs/storage.conf
```

II

DB

2. MariaDB

2.1 常用操作

在 CentOS 下启动 MariaDB:

```
systemctl start mariadb.service
```

2.1.1 导入导出

导出整个库结构和数据:

```
mysqldump -h localhost -uroot -p123456  database >
        dump.sql
```

启动 MariaDB 时注意是否带了–skip-networking 参数，否则会出现数据库启动了，但是端口没有监听的情况。

允许远程登录

首先以 root 用户登陆 MariaDB 服务器：

```
--允许用户名为‘dolphin‘的用户从任意ip以密码为123456访
        问所有数据库
grant all PRIVILEGES on *.* to dolphin@’%’ identified
         by ’123456’;
```

```
--使其生效
flush privileges;
```

修改数据库密码

修改数据库密码如下：

```
#登录 SQL
mysql -uroot -p
use mysql;
UPDATE user SET password=password('newpassword')
        WHERE user='root';
flush privileges;
exit;
```

2.2 常见问题

2.2.1 mariadb 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

在使用命令登录时, 出现错误. 依次执行如下语句即可:

```
#停止 MariaDB
systemctl stop mariadb.service
#登录
mysqld_safe --user=mysql --skip-grant-tables --skip-
        networking &
mysql -u root mysql
#设置新密码
UPDATE user SET Password=PASSWORD('123123') where
        USER='root';
FLUSH PRIVILEGES;
quit
#启动 MariaDB
systemctl start mariadb.service
```

修改后，登录数据库：

```
mysql -u root -p123123
```

Bibliography

Books

Articles