# 加密和安全

## 一.安全机制

- 安全攻击的几种典型方式： STRIDE

```
Spoofing                假冒
Tampering               篡改
Repudiation             否认
Information Disclosure  信息泄漏
Denial of Service       拒绝服务
Elevation of Privilege  提升权限
```

- 安全设计基本原则
    - 使用成熟的安全系统
    - 以小人之心度输入数据
    - 部系统是不安全的
    - 小授权
    - 少外部接口
    - 省使用安全模式
    - 全不是似是而非
    - STRIDE思考
    - 入口处检查
    - 管理上保护好你的系统
- 常见加密算法和协议
    - 对称加密
    - 公钥加密
    - 单向加密
    - 认证协议

## 二.对称和非对称加密

### 1.对称加密算法:加密和解密使用同一个密钥

```
DES：Data Encryption Standard，56bits
3DES：
AES：Advanced (128, 192, 256bits)
Blowfish,Twofish,IDEA,RC6,CAST5
```

- 特性： 1、加密、解密使用同一个密钥，效率高 2、将原始数据分割成固定大小的块，逐个进行加密

- 缺陷： 1、密钥过多
  2、密钥分发 3、数据来源无法确认

## 2.非对称加密算法

- 公钥加密：密钥成对出现
    - 公钥：公开给所有人；public key
    - 私钥：自己留存，必须保证其私密性；secret key
- 特点：用公钥加密数据，只能使用与之配对的私钥解密；反之亦然
- 功能：
    - 数字签名：主要在于让接收方确认发送方身份对称密钥交换：发送方用对方的公钥加密一个对称密钥后发送给对方
    - 数据加密：适合加密较小数据
- 缺点：密钥长，加密解密效率低下
- 算法： RSA（加密，数字签名） DSA（数字签名） ELGamal
- 基于一对公钥/密钥对 •用密钥对中的一个加密，另一个解密
- 实现加密： •接收者 生成公钥/密钥对：P和S 公开公钥P，保密密钥S •发送者 使用接收者的公钥来加密消息M 将P(M)发送给接收者 •接收者 使用密钥S来解密：M=S(P(M))
- 实现数字签名： •发送者 生成公钥/密钥对：P和S 公开公钥P，保密密钥S 使用密钥S来加密消息M 发送给接收者S(M) •接收者 使用发送者的公钥来解密M=P(S(M)) 结合签名和加密 分离签名

## 3.RSA和DSA

- RSA：公钥加密算法是1977年由Ron Rivest、Adi Shamirh和LenAdleman在 （美国麻省理工学院）开发的，RSA取名来自开发他们三者的名字，后成立RSA 数据安全有限公司。RSA是目前最有影响力的公钥加密算法，它能够抵抗到目 前为止已知的所有密码攻击，已被ISO推荐为公钥数据加密标准。RSA算法基于一个十分简单的数论事实：将两个大素数相乘十分容易，但想要对其乘积进行因式 分解却极其困难，因此可以将乘积公开作为加密密钥
- DSA (Digital Signature Algorithm)：1991年7月26日提交，并归属于David W. Kravitz前NSA员工，DSA是Schnorr和ElGamal签名算法的变种，被美国 NIST作为SS(DigitalSignature Standard)， DSA是基于整数有限域离散对数难 题的，其安全性与RSA相比差不多。DSA只是一种算法，和RSA不同之处在于 它不能用作加密和解密，也不能进行密钥交换，只用于签名,它比RSA要快很多

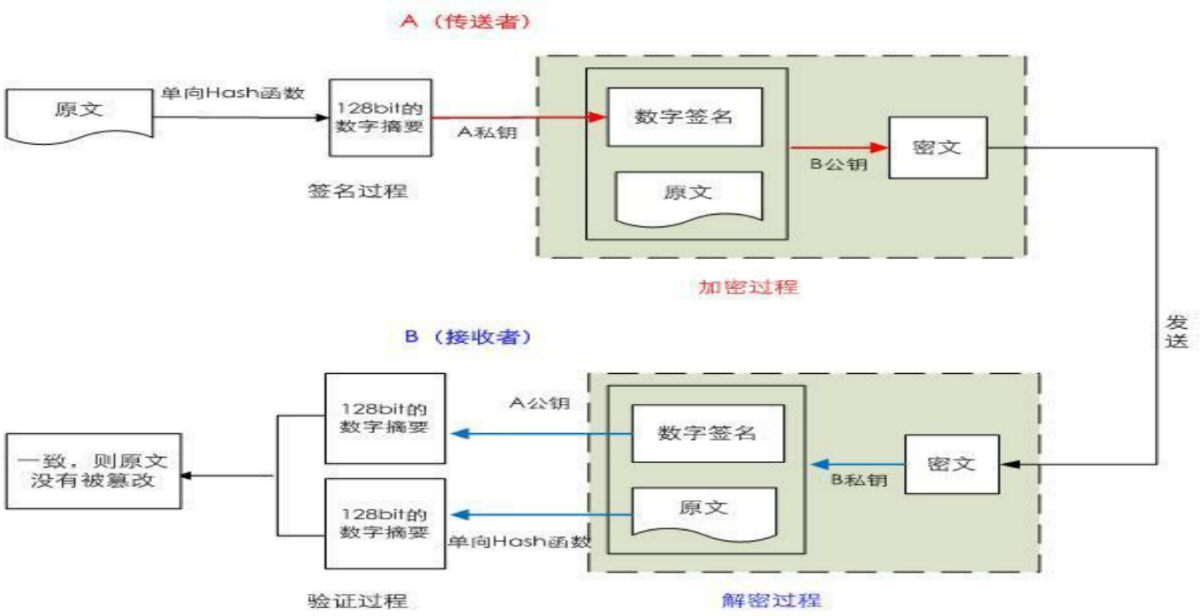
# 三.散列算法

## 1.单向散列:将任意数据缩小成固定大小的“指纹”

•任意长度输入 •固定长度输出 •若修改数据，指纹也会改变（“不会产生冲突”） •无法从指纹中重新生成数据（“单向”）

- 功能：保护数据完整性

- 常见算法:md5: 128bits、sha1: 160bits、sha224 、sha256、sha384、sha512
- 常用工具 •md5sum | sha1sum [ --check ] file •openssl、gpg •rpm -V
- 数字签名图解




## 2.密钥交换:IKE(Internet Key Exchange)

- 公钥加密DH
- DH参看

```
DH (Deffie-Hellman)：生成会话密钥，由惠特菲尔德·迪菲（Bailey Whitfield Diffie）和
马丁·赫尔曼（Martin Edward Hellman）在1976年发表.
DH:
A: g,p 协商生成公开的整数g，大素数p
B: g,p
A:生成隐私数据 :a (a<p )，计算得出 gâ%p，发送给B
B:生成隐私数据 :b,计算得出 g^b%p，发送给A
A:计算得出 [(g^b%p)â] %p = gâb%p，生成为密钥
B:计算得出 [(gâ%p)^b] %p = gâb%p，生成为密钥
```

# 四.gpg

## 1.使用gpg实现对称加密file文件

```
gpg -c file
ls file.gpg
```

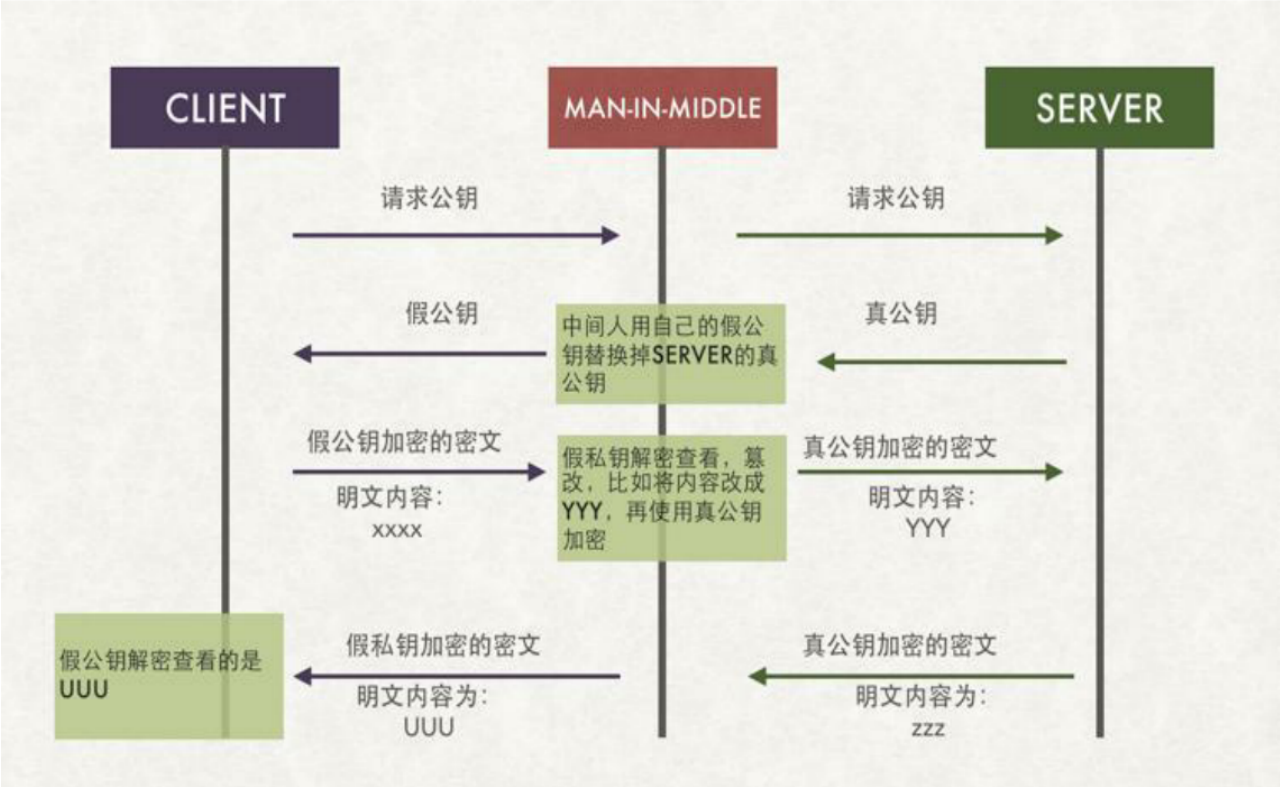
- 在另一台主机上解密file

```
gpg -o file -d file.gpg
```

## 2.使用gpg工具实现公钥加密

```
在hostB主机上用公钥加密，在hostA主机上解密
在hostA主机上生成公钥/私钥对
    gpg --gen-key
在hostA主机上查看公钥
    gpg --list-keys
在hostA主机上导出公钥到wang.pubkey
    gpg -a  --export -o wang.pubkey
从hostA主机上复制公钥文件到需加密的B主机上
    scp wang.pubkey  hostB:
在需加密数据的hostB主机上生成公钥/私钥对
    gpg --list-keys
    gpg --gen-key
在hostB主机上导入公钥
    gpg --import wang.pubkey
    gpg --list-keys
用从hostA主机导入的公钥，加密hostB主机的文件file,生成file.gpg
    gpg -e -r wangxiaochun file
    file file.gpg
复制加密文件到hostA主机
    scp fstab.gpg hostA:
在hostA主机解密文件
    gpg -d file.gpg
    gpg -o file  -d file.gpg
删除公钥和私钥
    gpg --delete-keys  wangxiaochun
    gpg --delete-secret-keys  wangxiaochun
```

- 中间人攻击




# 五.PKI和CA

- PKI：Public Key Infrastructure
- CA和证书

```
签证机构:CA(Certificate Authority)
注册机构:RA(registration authority )
证书吊销列表:CRL(Certificate Revoke List)
证书存取库:Certificate access library
```

- X.509：定义了证书的结构以及认证协议标准

```
版本号
序列号
签名算法
颁发者
有效期限
主体名称
主体公钥
CRL分发点
扩展信息
发行者签名
```

- 证书类型

```
证书授权机构的证书
服务器证书
用户证书
```

- 获取证书两种方法：
    - 使用证书授权机构授权的证书

```
生成证书请求(csr)(The certificate request)
将证书请求csr发送给CA
CA签名颁发证书
```

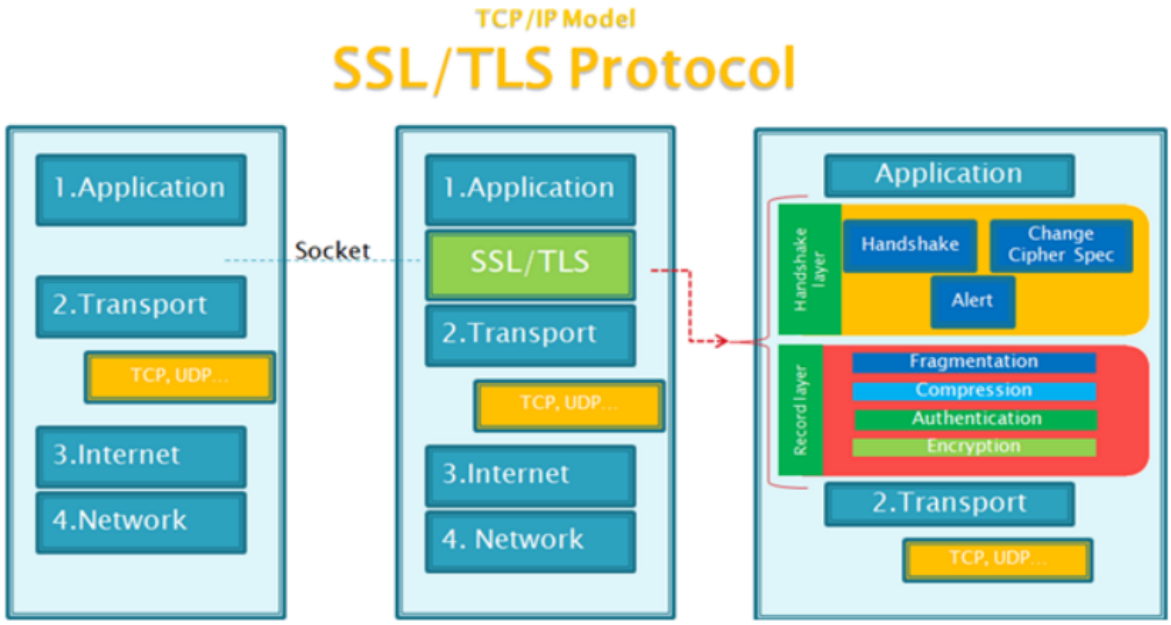
- 自签名的证书:自己签发自己的公钥

# 六.openssl

## 1.SSL：Secure Socket Layer，TLS: Transport Layer Security

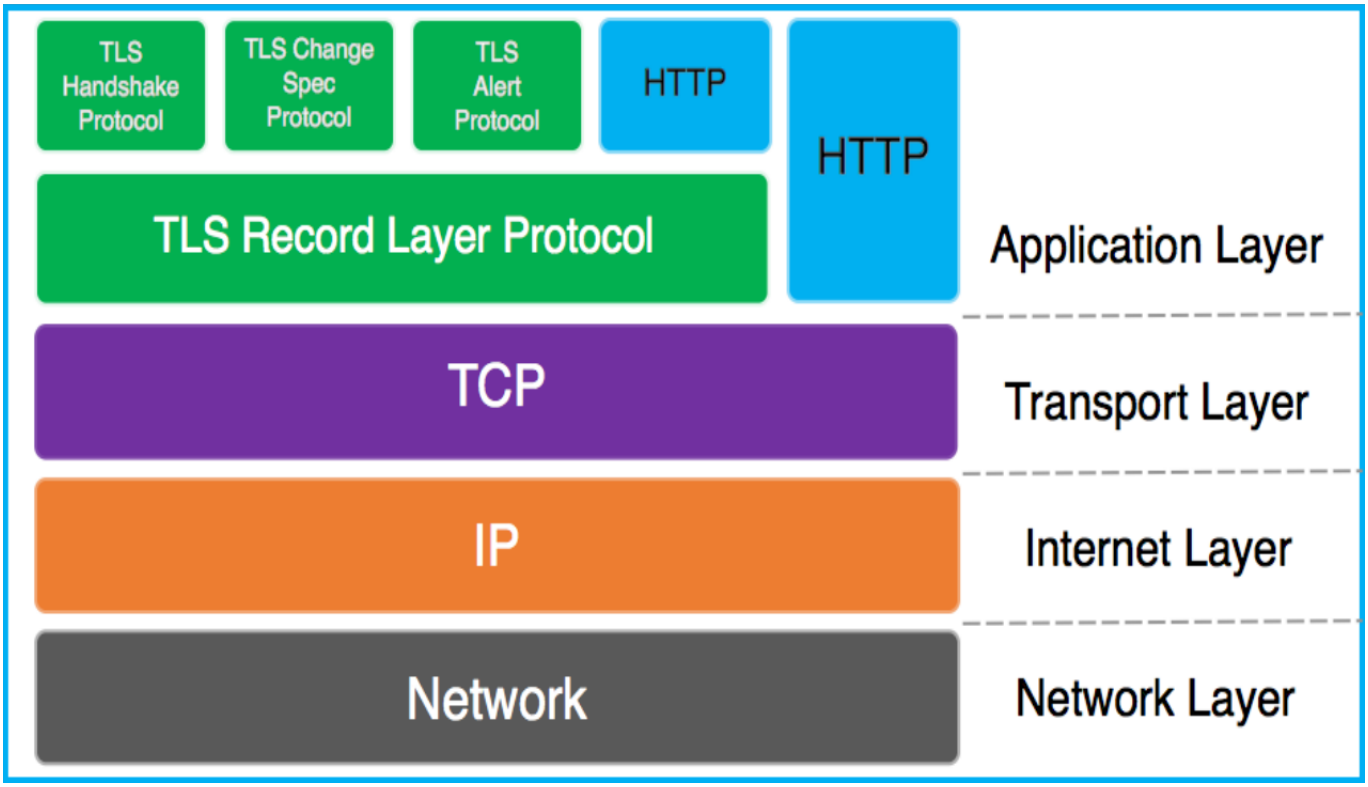
```
1995: SSL 2.0 Netscape
1996: SSL 3.0
1999: TLS 1.0
2006: TLS 1.1 IETF(Internet工程任务组) RFC 4346
2008: TLS 1.2 当前使用
2015: TLS 1.3
```

- 功能：机密性，认证，完整性，重放保护
- 两阶段协议，分为握手阶段和应用阶段
    - 握手阶段(协商阶段):客户端和服务器端认证对方身份（依赖于PKI体系，利用数字 证书进行身份认证），并协商通信中使用的安全参数、密码套件以及主密钥。后续通信使 用的所有密钥都是通过MasterSecret生成
    - 应用阶段:在握手阶段完成后进入，在应用阶段通信双方使用握手阶段协商好的密钥进行安全通信

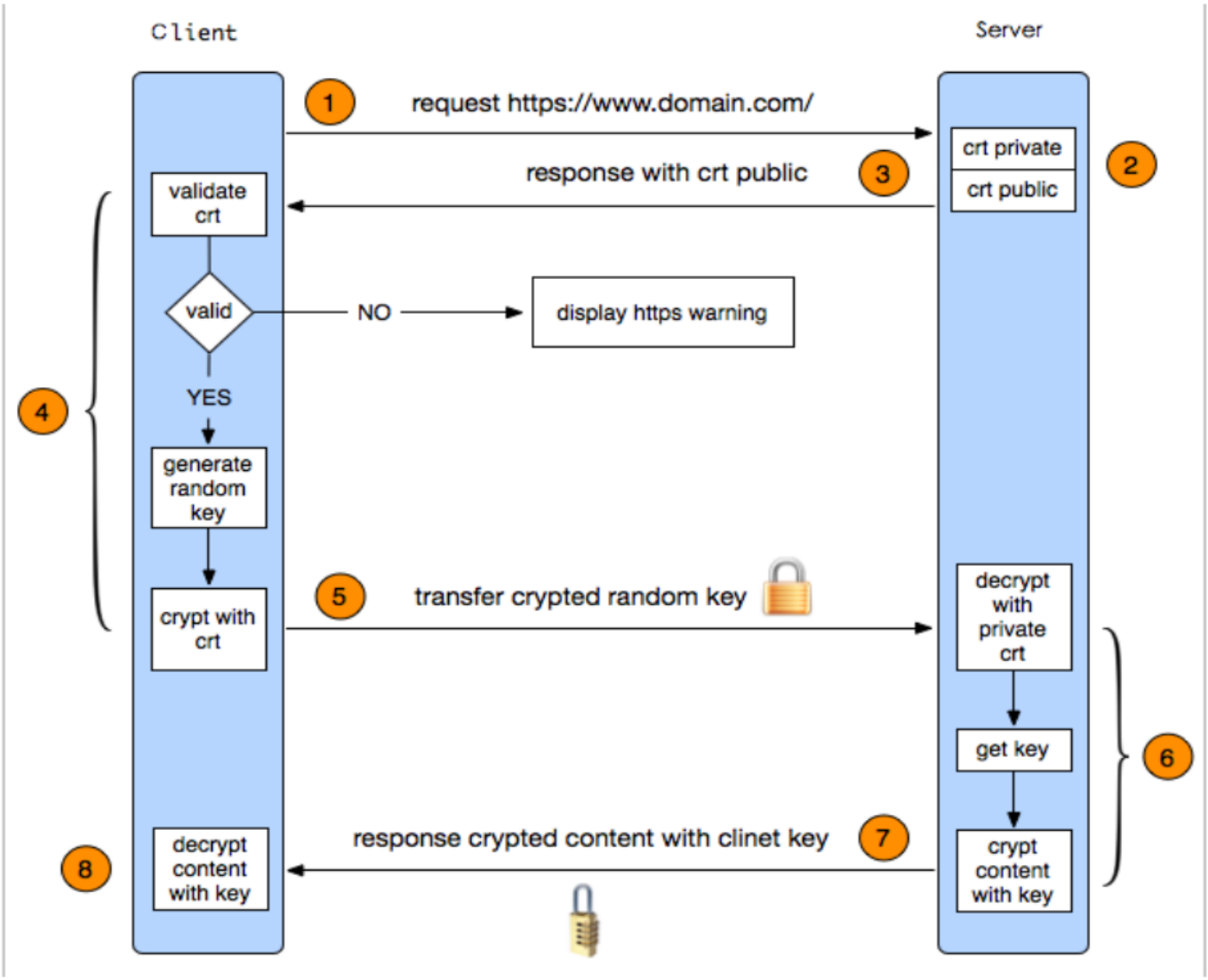



SSL/TLS协议在四层网络模型中的位置及模块

- Handshake协议：包括协商安全参数和密码套件、服务器身份认证（客户端身 份认证可选）、密钥交换
- ChangeCipherSpec 协议：一条消息表明握手协议已经完成
- Alert 协议：对握手协议中一些异常的错误提醒，分为fatal和warning两个级别，
- fatal类型错误会直接中断SSL链接，而warning级别的错误SSL链接仍可继续， 只是会给出错误警告
- Record 协议：包括对消息的分段、压缩、消息认证和完整性保护、加密等
- HTTPS协议：就是"HTTP协议"和"SSL/TLS协议"的组合。HTTP over SSL”或“HTTP over TLS”，对http协议的文本数据进行加密处理后，成为二 进制形式传输




HTTPS结构




HTTPS工作过程

## 2.openssl软件

- openssl命令包括三个组件：
  - openssl：多用途的命令行工具，包openssl
  - libcrypto：加密算法库，包openssl-libs
  - libssl：加密模块应用库，实现了ssl及tls，包nss
- openssl命令用法
- 两种运行模式：交互模式和批处理模式
- 标准命令、消息摘要命令、加密命令
  - 标准命令：enc, ca, req, ...
  - 对称加密:

```
工具：openssl enc, gpg
算法：3des, aes, blowfish, twofish
```

- openssl的enc子命令
  - 加密： `openssl enc -e -des3 -a -salt -in testfile -out testfile.cipher`
  - 解密： `openssl enc -d -des3 -a -salt -in testfile.cipher -out testfile`
- 单向加密：
  - 工具：md5sum, sha1sum, sha224sum,sha256sum... ,openssl dgst
- dgst命令 `openssl dgst -md5 [-hex默认] /PATH/SOMEFILE` `openssl dgst -md5 testfile` `md5sum /PATH/TO/SOMEFILE`
- MAC: Message Authentication Code，单向加密的一种延伸应用，用于实现网络通信 中保证所传输数据的完整性机制
- CBC-MAC
- HMAC：使用md5或sha1算法 生成用户密码：
- openssl的passwd命令 `openssl passwd -1 -salt SALT(最多8位)` `openssl passwd -1 -salt centos`
- 生成随机数： `openssl rand -base64|-hex NUM`
  - NUM: 表示字节数，使用-hex，每个字符为十六进制，相当于4位二进制,出现的字符数为NUM*2
- 公钥加密：

```
算法：RSA, ELGamal
工具：gpg, openssl rsautl（man rsautl）
```

- 数字签名：

```
算法：RSA, DSA, ELGamal
```

- 密钥交换：

```
算法：dh
DSA: Digital Signature Algorithm
DSS: Digital Signature Standard
```

- 生成密钥对儿：man genrsa
- 生成私钥 `openssl genrsa -out /PATH/TO/PRIVATEKEY.FILE NUM_BITS` (umask 077; openssl genrsa –out test.key –des 2048) `openssl rsa -in test.key -out test2.key` 将加密key解密
- 从私钥中提取出公钥 `openssl rsa -in PRIVATEKEYFILE –pubout –out PUBLICKEYFILE` `openssl rsa –in test.key –pubout –out test.key.pub`
- 随机数生成器：伪随机数字

- 键盘和鼠标，块设备中断 /dev/random：仅从熵池返回随机数；随机数用尽，阻塞
  /dev/urandom：从熵池返回随机数；随机数用尽，会利用软件生成伪随机数,非阻塞

# 七.CA创建和管理

## 1.PKI：Public Key Infrastructure

```
CA
RA
CRL
证书存取库
```

- 建立私有CA一般使用工具:

```
OpenCA
openssl
```

- 证书申请及签署步骤：

```
1、生成申请请求
2、RA核验
3、CA签署
4、获取证书
```

## 3.创建私有CA、证书申请和管理

### 创建私有CA

- openssl的配置文件：/etc/pki/tls/openssl.cnf
- 三种策略：match匹配、optional可选、supplied提供

```
match：要求申请填写的信息跟CA设置信息必须一致
optional：可有可无，跟CA设置信息可不一致
supplied：必须填写这项申请信息，
```

### 步骤

1、创建所需要的文件

```
touch /etc/pki/CA/index.txt 生成证书索引数据库文件
echo 01 > /etc/pki/CA/serial 指定第一个颁发证书的序列号
```

2、 CA自签证书

```
生成私钥
    cd /etc/pki/CA/
    (umask 066; openssl genrsa -out  private/cakey.pem 2048)
生成自签名证书
   openssl req -new -x509 -key   /etc/pki/CA/private/cakey.pem -days 3650 -out
/etc/pki/CA/cacert.pem
```

- 选项说明：

```
-new：生成新证书签署请求
-x509：专用于CA生成自签证书
-key：生成请求时用到的私钥文件
-days n：证书的有效期限
-out /PATH/TO/SOMECERTFILE: 证书的保存路径
```

3、颁发证书

```
3.1 在需要使用证书的主机生成证书请求
eg:给web服务器生成私钥
(umask 066; openssl genrsa -out   /data/test.key 2048)
3.2 生成证书申请文件
openssl req -new -key /data/test.key -out /data/test.csr
3.3 将证书请求文件传输给CA
3.4 CA签署证书，并将证书颁发给请求者
openssl ca -in /tmp/test.csr -out   /etc/pki/CA/certs/test.crt -days 100
注意：默认要求 国家，省，公司名称三项必须和CA一致
查看证书中的信息：
    openssl x509 -in /PATH/FROM/CERT_FILE -noout   -
text|issuer|subject|serial|dates
查看指定编号的证书状态
    openssl  ca -status SERIAL
```

4、吊销证书

```
4.1 在客户端获取要吊销的证书的serial
openssl x509 -in /PATH/FROM/CERT_FILE   -noout   -serial  -subject
4.2 在CA上，根据客户提交的serial与subject信息，对比检验是否与index.txt文件中的信息一致
吊销证书命令：
openssl ca -revoke /etc/pki/CA/newcerts/SERIAL.pem
4.3 指定第一个吊销证书的编号,注意：第一次更新证书吊销列表前，才需要执行
echo 01 > /etc/pki/CA/crlnumber
4.4 更新证书吊销列表
openssl ca -gencrl -out /etc/pki/CA/crl.pem
```

```
4.5 查看crl文件:
openssl crl -in /etc/pki/CA/crl.pem -noout -text
```

## openssl常用命令例子

| 命令 | 作用 |
| --- | --- |
| openssl enc -e -des3 -a -salt -in testfile -out testfile.cipher | 对称加密 |
| openssl enc -d -des3 -a -salt –in testfile.cipher -out testfile | 解密 |
| openssl dgst -md5 [-hex默认] /PATH/SOMEFILE | 为文件生成摘要(也叫指纹或者hash),同md5sum命令的计算结果 |
| openssl passwd -1 -salt SALT(最多8位) | 生成用户密码(-1:使用基于MD5的BSD密码算法;-salt:指定掺杂的字符) |
| openssl rand -base64 | -hex NUM |
| (umask 077; openssl genrsa –out test.key –des 2048) | 生成私钥 |
| openssl rsa -in test.key –out test2.key | 将加密keyjiemi |
| openssl rsa –in test.key –pubout –out test.key.pub | 从私钥中提取公钥 |
| (umask 066; openssl genrsa -out private/cakey.pem 2048) | 生成私钥 |
| openssl req -new -x509 -key /etc/pki/CA/private/cakey.pem -days 3650 -out /etc/pki/CA/cacert.pem | 生成自签名证书 |
| | -new：生成新证书签署请求 |
| | -x509：专用于CA生成自签证书 |
| | -key：生成请求时用到的私钥文件 |
| | -days n：证书的有效期限 |
| | -out /PATH/TO/SOMECERTFILE: 证书的保存路径 |
| openssl req -new -key /data/test.key -out /data/test.csr | 生成证书申请文件 |
| openssl ca -in /tmp/test.csr –out /etc/pki/CA/certs/test.crt -days 100 | CA签署证书 |
| | 注意：默认要求 国家，省，公司名称三项必须和CA一致 |
| openssl x509 -in /PATH/FROM/CERT_FILE -noout -text | issuer |
| openssl ca -status SERIAL | 查看指定编号的证书状态 |
| openssl ca -revoke /etc/pki/CA/newcerts/SERIAL.pem | 吊销证书 |
| openssl ca -gencrl -out /etc/pki/CA/crl.pem | 更新证书吊销列表 |

| 命令 | 作用 |
| --- | --- |
| openssl crl -in /etc/pki/CA/crl.pem -noout -text | 查看crl文件 |

# 八.ssh服务和dropbear

## 1.SSH

- ssh: secure shell, protocol, 22/tcp, 安全的远程登录
- 具体的软件实现：
  - **OpenSSH**:ssh协议的开源实现，CentOS默认安装
  - **dropbear**:另一个开源实现
- SSH协议版本
  - v1: 基于CRC-32做MAC，不安全；man-in-middle
  - v2：双方主机协议选择安全的MAC方式
- 基于DH算法做密钥交换，基于RSA或DSA实现身份认证
- OpenSSH介绍
  - 相关包：

```
openssh
openssh-clients
openssh-server
```

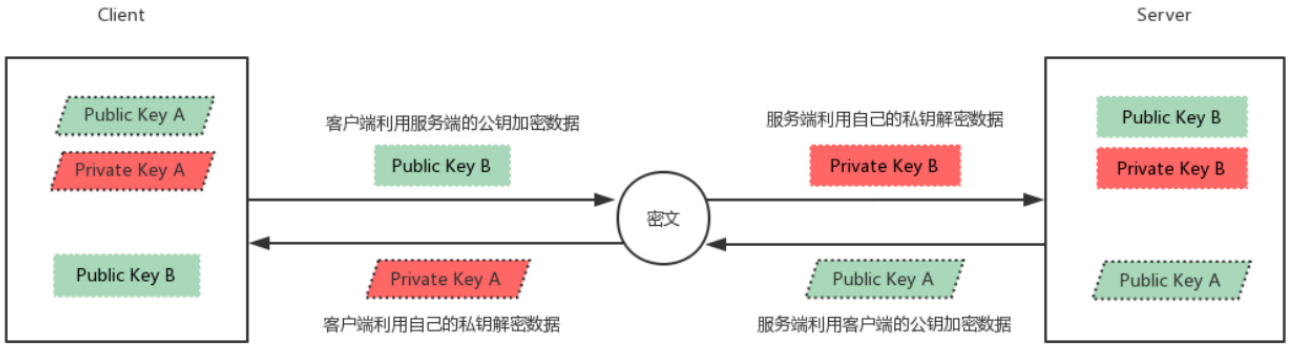
- 工具：基于C/S结构
  - Linux Client: ssh, scp, sftp，slogin
  - Windows Client：xshell, putty, securecrt, sshsecureshellclient
  - Server: sshd

- ssh建立连接时的公钥交换


公钥交换的过程

1.客户端发起链接请求
2.服务端返回自己的公钥，以及一个会话ID（这一步客户端得到服务端公钥）
3.客户端生成密钥对
4.客户端用自己的公钥异或会话ID，计算出一个值Res，并用服务端的公钥加密
5.客户端发送加密后的值到服务端，服务端用私钥解密,得到Res
6.服务端用解密后的值Res异或会话ID，计算出客户端的公钥（这一步服务端得到客户端公钥）
7.最终：双方各自持有三个秘钥，分别为自己的一对公、私钥，以及对方的公钥，之后的所有通讯都会被加密

- 建立连接后的数据交换过程




ssh客户端

- 客户端组件：

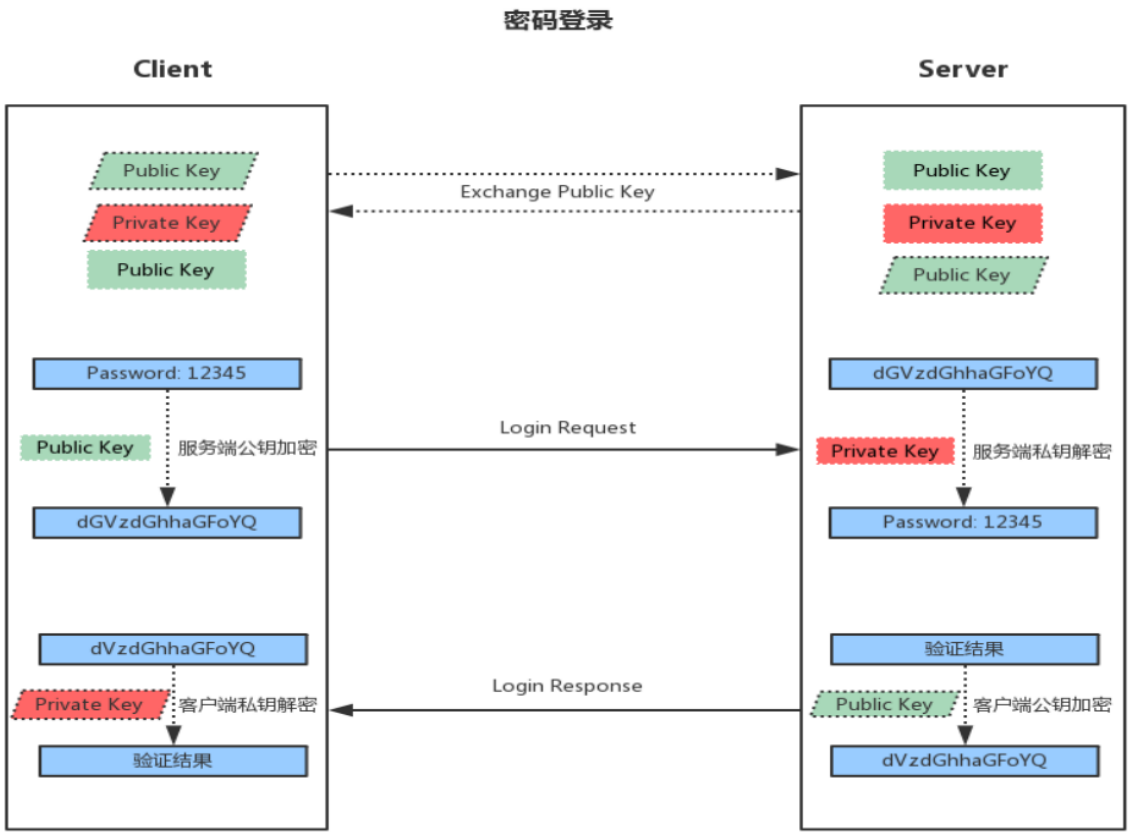
    - 配置文件：/etc/ssh/ssh_config
    - StrictHostKeyChecking no 首次登录不显示检查提示

```
格式:ssh [user@]host [COMMAND]
     ssh [-l user] host [COMMAND]
常见选项
-p port：远程服务器监听的端口
-b：指定连接的源IP
-v：调试模式
-C：压缩方式
-X：支持x11转发
-t：强制伪tty分配
ssh -t remoteserver1 ssh -t  remoteserver2   ssh   remoteserver3
```

- ssh允许实现对远程系统经验证地加密安全访问
- 当用户远程连接ssh服务器时，会复制ssh服务器/etc/ssh/ssh_host*key.pub （CentOS7默认是ssh_host_ecdsa_key.pub）文件中的公钥到客户机的 ~./ssh/know_hosts中。下次连接时，会自动匹配相应私钥，不能匹配，将拒绝连接

## ssh服务登录验证

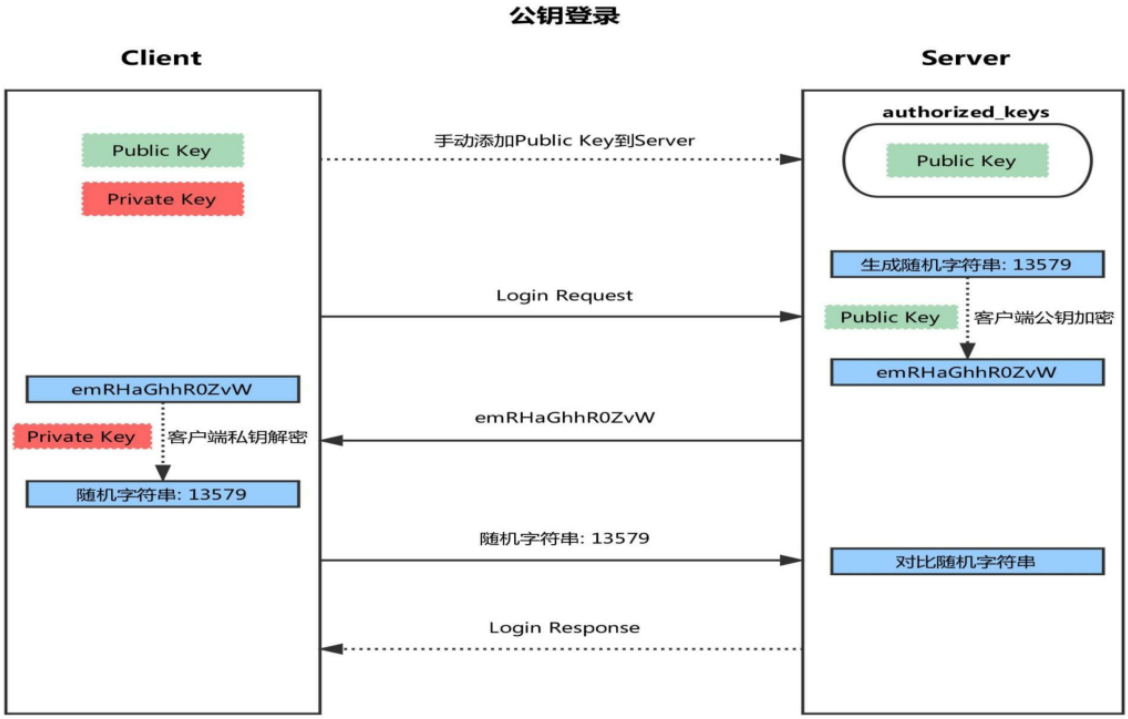
- ssh服务登录的验证方式两种
    - 用户/口令
    - 基于密钥
- 基于用户名和口令的登录验证过程



- 基于用户和口令登录验证过程描述

```
1 客户端发起ssh请求，服务器会把自己的公钥发送给用户
2 用户会根据服务器发来的公钥对密码进行加密
3 加密后的信息回传给服务器，服务器用自己的私钥解密，如果密码正确，则用户登录成功
```

- 基于密钥的登录验证




- 基于密钥的登录方式过程

```
1 首先在客户端生成一对密钥（ssh-keygen）
2 并将客户端的公钥ssh-copy-id 拷贝到服务端
3 当客户端再次发送一个连接请求，包括ip、用户名
4 服务端得到客户端的请求后，会到authorized_keys中查找，如果有响应的
IP和用户，就会随机生成一个字符串，例如：magedu
5 服务端将使用客户端拷贝过来的公钥进行加密，然后发送给客户端
6 得到服务端发来的消息后，客户端会使用私钥进行解密，然后将解密后的
字符串发送给服务端
7 服务端接受到客户端发来的字符串后，跟之前的字符串进行对比，如果一
致，就允许免密码登录
```

## 基于key认证的实现

- 基于密钥的认证实现步骤

```
(1)在客户端生成密钥对
    ssh-keygen -t rsa [-P ''] [-f “~/.ssh/id_rsa"]
```

```
(2)把公钥文件传输至远程服务器对应用户的家目录
    ssh-copy-id [-i [identity_file]] [user@]host
(3)测试
(4)在SecureCRT或Xshell实现基于key验证
    在SecureCRT工具—>创建公钥—>生成Identity.pub文件转化为openssh兼容格式
    （适合SecureCRT，Xshell不需要转化格式），并复制到需登录主机上相应文件
    authorized_keys中,注意权限必须为600，在需登录的ssh主机上执行：
    ssh-keygen  -i -f Identity.pub >> .ssh/authorized_keys
(5)重设私钥口令：
    ssh-keygen -p
(6)验证代理（authentication agent）保密解密后的密钥这样口令就只需要输入一次
    在GNOME中，代理被自动提供给root用户
    否则运行ssh-agent bash
(7)钥匙通过命令添加给代理
    ssh-add
```

## sshfs将远程主机的目录挂载到本地

- 用法
  - 挂载 `sshfs [user@]host:[dir] mountpoint [options]`
  - 卸载 `fusermount -u mountpoint umount mountpoint`

## scp命令

- 用法 `scp [options] SRC... DEST/`
- 两种方式 `scp [options] [user@]host:/sourcefile /destpath scp [options] /sourcefile [user@]host:/destpath`
- 常用选项

```
-C 压缩数据流
-r 递归复制
-p 保持原文件的属性信息
-q 静默模式
-P PORT
指明remote host的监听的端口
```

## rsync命令

- rsync基于ssh和rsh服务实现高效率的远程系统之间复制文件,其使用安全的shell连接做为传输方式 `rsync -av /etc server1:/tmp`
- 复制目录和目录下文件 `rsync -av /etc/ server1:/tmp`
- rsync只复制目录下有改动的文件，包括文件元数据的变化；scp更快.
- 常用选项

```
-n 模拟复制过程
-v 显示详细过程
-r 递归复制目录树
-p 保留权限
```

```
-t 保留时间戳
-g 保留组信息
-o 保留所有者信息
-l 将软链接文件本身进行复制（默认）
-L 将软链接文件指向的文件复制
-a 存档，相当于-rlptgoD，但不保留ACL（-A）和SELinux属性（-X）
```

### sftp命令

- 交互式文件传输工具用法和传统的ftp工具相似
- 其利用ssh服务实现安全的文件上传和下载
- 使用ls cd mkdir rmdir pwd get put等指令，可用？或help获取帮助信息

```
sftp [user@]host
sftp> help
```

## 2.轻量级自动化运维工具

### pssh

- 基于python编写，可在多台服务器上执行命令的工具，也可实现文件复制，提供了基于ssh和 scp的多个并行工具
- 用法

```
--version：查看版本
-h：主机文件列表，内容格式"[user@]host[:port]"
-H：主机字符串，内容格式"[user@]host[:port]"
-A：手动输入密码模式
-i：每个服务器内部处理信息输出
-l：登录使用的用户名
-p：并发的线程数[可选]
-o：输出的文件目录[可选] # 生成的文件名为主机名
-e：错误输出文件[可选]
-t：TIMEOUT 超时时间设置,0无限制[可选]
-O：SSH的选项
-P：打印出服务器返回信息
-v：详细模式
```

- 示例

```
1.通过pssh批量关闭seLinux
    pssh -H root@192.168.1.10  -i  ‘sed -i "s/^SELINUX=.*/SELINUX=disabled/"
/etc/selinux/config’
2.批量发送指令
    pssh -H root@192.168.1.10  -i  setenforce 0
```

```
    pssh -H wang@192.168.1.10  -i  hostname
3.当不支持ssh的key认证时，通过 -A选项，使用密码认证批量执行指令
    pssh -H wang@192.168.1.10  -A -i  hostname
4.将标准错误和标准正确重定向都保存至本地主机的/app目录下
    pssh -H 192.168.1.10 -o /app -e /app   -i “hostname”
5.使用host主机列表来发送指令到多台电脑
[root@centos7 /data]#cat hostlist
172.20.2.16
[root@centos7 /data]#pssh -h hostlist -i ls /data
[1] 09:04:55 [SUCCESS] 172.20.2.16
authorized_keys
bucket_man.gif
encrypt
giphy.gif
hand.gif
mice.gif
penguin.gif
rabbit.gif
```

> 项目

## pscp.pssh命令

- pscp.pssh功能是将本地文件批量复制到远程主机
- 用法 `pscp [-vAr] [-h hosts_file] [-H [user@]host[:port]] [-l user] [-p par] [-o outdir] [-e errdir] [-t timeout] [-O options] [-x args] [-X arg] local remote`
- pscp-pssh选项及用法

```
    -v 显示复制过程
    -r 递归复制目录
将本地curl.sh 复制到/app/目录
    pscp.pssh -H 192.168.1.10 /root/test/curl.sh    /app/
    pscp.pssh -h host.txt  /root/test/curl.sh    /app/
将本地多个文件批量复制到/app/目录
    pscp.pssh -H 192.168.1.10  /root/f1.sh  /root/f2.sh    /app/
将本地目录批量复制到/app/目录
    pscp.pssh -H 192.168.1.10  -r /root/test/    /app/
```

## pslurp命令

- pslurp功能是将远程主机的文件批量复制到本地 `pslurp [-vAr] [-h hosts_file] [-H [user@]host[:port]] [-l user] [-p par] [-o outdir] [-e errdir] [-t timeout] [-O options] [-x args] [-X arg] [-L localdir] remote local（本地名）`
- pslurp选项

```
-L 指定从远程主机下载到本机的存储的目录，local是下载到本地后的名称
-r 递归复制目录
```

- 批量下载目标服务器的passwd文件至/app下，并更名为user `pslurp -H 192.168.1.10 -L /app /etc/passwd user`

## pdsh

- Parallel remote shell program，是一个多线程远程shell客户端，可以 并行执行多个远程主机上的命令。pdsh可以使用几种不同的远程shell服务，包 括标准的“rsh”，Kerberos IV和ssh

> 项目地址

## mussh

- Multihost SSH wrapper，是一个shell脚本，允许您使用一个命令在 多个主机上通过ssh执行命令或脚本。 mussh可使用ssh-agent和RSA / DSA密 钥，以减少输入密码

> 项目

- pssh,pdsh,mussh工具都包含在EPEL源中

## SSH端口转发

- SSH会自动加密和解密所有SSH客户端与服务端之间的网络数据。但是，SSH还能够将其他TCP端 口的网络数据通过SSH链接来转发，并且自动提供 相应的加密及解密服务。这一过程也被叫做“隧道”（tunneling），这是因为SSH为其他TCP链接提供了一个安全的通道来进行传输而得名。例 如，Telnet，SMTP，LDAP这些TCP应用均能够从中得益，避免了用户名，密码以及隐私 息的明文 传输。而与此同时，如果工作环境中的防火墙限制了一些网络端口的使用，但是允许SSH的连接， 也能够通过将TCP端口转发来使用SSH进行通讯。

- SSH 端口转发能够提供两大功能：
  - 1.加密 SSH Client 端至 SSH Server 端之间的通讯数据
  - 2.突破防火墙的限制完成一些之前无法建立的 TCP 连接

- 本地转发实现 `ssh -L localport:remotehost:remotehostport sshserver`

- 选项

```
-f 后台启用
-N 不打开远程shell，处于等待状态
-g 启用网关功能
```

- 示例

```
ssh -L  9527:telnetsrv:23 -Nfg  sshsrv
ssh -L 9527:192.168.39.18:23 192.168.39.7 -fN
telnet 127.0.0.1 9527
```

- 当访问本机的9527的端口时，被加密后转发到sshsrv的ssh服务，再解密被转发到telnetsrv:23
- 数据传输过程示意
  `data <---> localhost:9527 <---> localhost:XXXXX <---> sshsrv:22 <---> sshsrv:YYYYY <---> telnetsrv:23`
- 远程转发 `ssh -R sshserverport:remotehost:remotehostport sshserver`
- 此处:sshserver表示出差人员的主机
- 此处:remotehost表示企业内部的跳板机 **注意：该命令一定是企业内部的主机执行**
- 示例 ssh –R 9527:telnetsrv:23 –Nf sshsrv
- 让sshsrv侦听9527端口的访问，如有访问，就加密后通过ssh服务转发请求 到本机ssh客户端，再由本机解密后转发到telnetsrv:23 `Data <---> sshsrv:9527 <---> sshsrv:22 <---> localhost:XXXXX <---> localhost:YYYYY <---> telnetsrv:23`
- 动态端口转发：
- 当用firefox访问internet时，本机的1080端口做为代理服务器，firefox的访问请求被转发到 sshserver上，由sshserver替之访问internet `ssh -D 1080 root@sshserver -fNg`
- 在本机firefox设置代理socket proxy:127.0.0.1:1080
- curl --socks5 127.0.0.1:1080 http://www.google.com
- X/windows 协议转发
  - 所有图形化应用程序都是X客户程序
  - 能够通过tcp/ip连接远程X服务器
  - 数据没有加密机，但是它通过ssh连接隧道安全进行
  - ssh -X user@remotehost gedit
  - remotehost主机上的gedit工具，将会显示在本机的X服务器上
  - 传输的数据将通过ssh连接加密

## 3.ssh服务器

- 服务器端sshd配置文件: /etc/ssh/sshd_config
- 常用参数

```
Port
ListenAddress ip
LoginGraceTime 2m
PermitRootLogin yes
StrictModes yes     检查.ssh/文件的所有者，权限等
MaxAuthTries   6
MaxSessions  10  同一个连接最大会话
PubkeyAuthentication yes
PermitEmptyPasswords no
PasswordAuthentication yes
GatewayPorts no
```

```
ClientAliveInterval    单位:秒
ClientAliveCountMax   默认3
UseDNS yes
GSSAPIAuthentication yes  提高速度可改为no
MaxStartups               未认证连接最大值，默认值10
Banner /path/file

限制可登录用户的办法:
    AllowUsers user1 user2 user3
    DenyUsers
    AllowGroups
    DenyGroups
```

- ssh服务的最佳实践

```
建议使用非默认端口
禁止使用protocol version 1
限制可登录用户
设定空闲会话超时时长
利用防火墙设置ssh访问策略
仅监听特定的IP地址
基于口令认证时，使用强密码策略 `tr -dc A-Za-z0-9_ < /dev/urandom | head -c 12| xargs`
使用基于密钥的认证
禁止使用空密码
禁止root用户直接登录
限制ssh的访问频度和并发在线数
经常分析日志
```

# 4.dropbear

- ssh协议的另一个实现dropbear,其占用空间少，功能丰富
- 源码编译安装

```
1、安装相关包:yum install gcc
2、下载dropbear-2019.78.tar.bz2
3、tar xf dropbear-2019.78.tar.bz2
4、less INSTALL README
5、./configure
6、make PROGRAMS="dropbear dbclient dropbearkey dropbearconvert scp"
7、make PROGRAMS="dropbear dbclient dropbearkey dropbearconvert scp" install
启动ssh服务:
    8、ls /usr/local/sbin/  /usr/local/bin/
    9、/usr/local/sbin/dropbear  -h
    10、mkdir /etc/dropbear
    11、dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key -s 2048
    12、dropbearkey -t dss -f /etc/dropbear/dropbear_dsa_host_key
    13、dropbear -p :2222 -F -E   #前台运行
```

```
    dropbear -p :2222 #后台运行
客户端访问：
    14、ssh -p 2222 root@127.0.0.1
    15、dbclient -p 2222 root@127.0.0.1
```

# 十.aide

## 1.简介

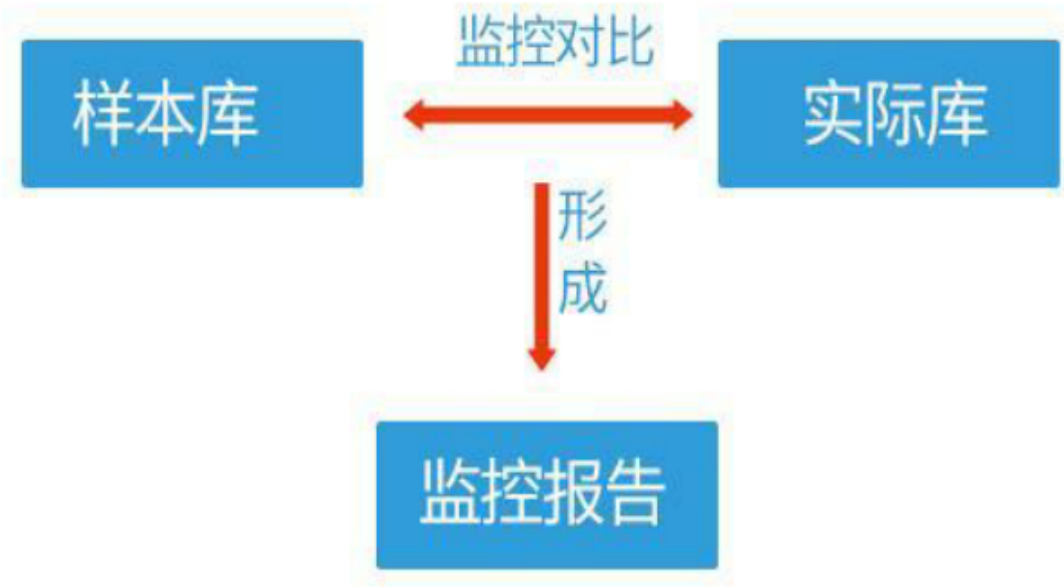
- 当一个入侵者进入了你的系统并且种植了木马，通常会想办法来隐蔽这个木马（除了木马自身的 一些隐蔽特性外，他会尽量给你检查系统的过程设置障碍），通常入侵者会修改一些文件，比如管 理员通常用ps aux来查看系统进程，那么入侵者很可能用自己经过修改的ps程序来替换掉你系统 上的ps程序，以使用ps命令查不到正在运行的木马程序。如果入侵者发现管理员正在运行crontab 作业，也有可能替换掉crontab程序等等。所以由此可以看出对于系统文件或是关键文件的检查是 很必要的。目前就系统完整性检查的工具用的比较多的有两款：Tripwire和AIDE，前者是一款商 业软件，后者是一款免费的但功能也很强大的工具。

- AIDE(Advanced Intrusion Detection Environment高级入侵检测环境)是一个入侵检测工 具，主要用途是检查文件的完整性，审计计算机上的那些文件被更改过了

- AIDE能够构造一个指定文件的数据库，它使用aide.conf作为其配置文件。AIDE数据库能够保 存文件的各种属性，包括：权限(permission)、索引节点序号(inode number)、所属用户 (user)、所属用户组(group)、文件大小、最后修改时间(mtime)、创建时间(ctime)、最后 访问时间(atime)、增加的大小以及连接数。AIDE还能够使用下列算法：sha1、md5、rmd160、 tiger，以密文形式建立每个文件的校验码或散列号

- 该数据库不应该保存那些经常变动的文件信息，例如：日志文件、邮件、/proc文件系统、用户起始目录以及临时目录

- AIDE使用样本库和实际库实现监控报告




## 2.安装配置

- 安装 `yum install aide`
- 修改配置文件

```
vim /etc/aide.conf (指定对哪些文件进行检测)
/test/chameleon R
/bin/ps R+a
/usr/bin/crontab R+a
/etc    PERMS
!/etc/mtab   #“!”表示忽略这个文件的检查
R=p+i+n+u+g+s+m+c+md5 权限+索引节点+链接数+用户+组+大小+最后一次修改时间+创建时间+md5校验值
NORMAL = R+rmd60+sha256
```

- 初始化默认的AIDE的库： `/usr/local/bin/aide --init`
- 生成检查数据库（建议初始数据库存放到安全的地方） `cd /var/lib/aide mv aide.db.new.gz aide.db.gz`
- 检测 `/usr/local/bin/aide --check`
- 更新数据库 `aide --update`

# 十一.Sudo

## 1.简介

- sudo能够授权指定用户在指定主机上运行某些命令。如果未授权用户尝试使用sudo，会提示联 系管理员
- sudo工具来自sudo包

```
[root@old_centos7 ~]#rpm -qf `which sudo`
sudo-1.8.19p2-13.el7.x86_64
```

- sudo也可以提供日志，记录每个用户使用sudo操作
- sudo为系统管理员提供配置文件，允许系统管理员集中地管理用户的使用权限和使用的主机
- sudo使用时间戳文件来完成类似“检票”的系统，默认存活期为5分钟的“入场券”

## 2.配置文件

- 通过visudo命令编辑配置文件，具有语法检查功能 `visudo -c`检查语法 `visudo -f /etc/sudoers.d/test export EDITOR=vim`

```
配置文件：/etc/sudoers, /etc/sudoers.d/
时间戳文件：/var/db/sudo
日志文件：/var/log/secure
```

- 配置文件支持使用通配符glob

```
?       任意单一字符
*       匹配任意长度字符
[wxc]   匹配其中一个字符
[!wxc]  除了这三个字符的其它字符
\x 转义
[[alpha]] 字母
示例：  /bin/ls [[alpha]]*
```

- 配置文件规则有两类
  - 1、别名定义：不是必须的
  - 2、授权规则：必须的

## 3.sudoers

- 授权规则格式：
  - 用户 登入主机=(代表用户) 命令 `user host=(runas) command`
- 示例 `root ALL=(ALL) ALL`
- 格式说明

```
user: 运行命令者的身份
host: 通过哪些主机
```

```
(runas)：以哪个用户的身份
command：运行哪些命令
```

## 4.别名

- User和runas:

```
username
#uid
%group_name
%#gid
user_alias|runas_alias
```

- host

```
ip或hostname
network(/netmask)
host_alias
```

- command

```
command name
directory
sudoedit
Cmnd_Alias
```

## 5.sudo别名和示例

- 别名有四种类型：User_Alias，Runas_Alias，Host_Alias，Cmnd_Alias
- 别名格式：`[A-Z]([A-Z][0-9]_)*`
- 别名定义： `Alias_Type NAME1 = item1,item2,item3 : NAME2 = item4, item5`
- 示例1 `Student ALL=(ALL) ALL %wheel ALL=(ALL) ALL`
- 示例2 `student ALL=(root) /sbin/pidof,/sbin/ifconfig %wheel ALL=(ALL) NOPASSWD: ALL`
- 示例3

```
User_Alias  NETADMIN= netuser1,netuser2
Cmnd_Alias  NETCMD = /usr/sbin/ip
NETADMIN ALL=（root） NETCMD
```

- 示例4

```

```

```
User_Alias SYSADER=wang,mage,%admins
User_Alias DISKADER=tom
Host_Alias SERS=www.magedu.com,172.16.0.0/24
Runas_Alias OP=root
Cmnd_Alias SYDCMD=/bin/chown,/bin/chmod
Cmnd_Alias DSKCMD=/sbin/parted,/sbin/fdisk
SYSADER SERS=   SYDCMD,DSKCMD
DISKADER ALL=(OP)  DSKCMD
```

- 示例5

```
User_Alias ADMINUSER = adminuser1,adminuser2
Cmnd_Alias ADMINCMD = /usr/sbin/useradd，/usr/sbin/usermod,
/usr/bin/passwd [a-zA-Z]*, !/usr/bin/passwd root
ADMINUSER ALL=(root) NOPASSWD:ADMINCMD，
PASSWD:/usr/sbin/userdel
```

- 示例6 `Defaults:wang runas_default=tom wang ALL=(tom,jerry) ALL`
- 示例7 `wang 192.168.1.6,192.168.1.8=(root) /usr/sbin/,!/usr/sbin/useradd`
- 示例8 `wang ALL=(ALL) /bin/cat /var/log/messages*`

## 6.sudo命令用法

- ls -l /usr/bin/sudo

```
[root@centos7 ~]#ll `which sudo`
---s--x--x 1 root root 147320 Oct 24 23:27 /usr/bin/sudo
```

- sudo –i –u wang 切换身份
- 用法 `sudo [-u user] COMMAND`

```
-V 显示版本信息等配置信息
-u user  默认为root
-l,ll 列出用户在主机上可用的和被禁止的命令
-v 再延长密码有效期限5分钟,更新时间戳
-k 清除时间戳（1970-01-01），下次需要重新输密码
-K 与-k类似，还要删除时间戳文件
-b 在后台执行指令
-p 改变询问密码的提示符号
```

- 示例：-p "password on %h for user %p:"

# 十二.TCP Wrappers

# 1.简介

- 作者：Wieste Venema，IBM，Google
- 工作在第四层（传输层）的TCP协议
- 对有状态连接的特定服务进行安全检测并实现访问控制
- 以库文件形式实现
- 某进程是否接受libwrap的控制取决于发起此进程的程序在编译时是否针对libwrap进行编译的
- 判断服务程序是否能够由tcp_wrapper进行访问控制的方法： `ldd /PATH/TO/PROGRAM|grep libwrap.so strings PATH/TO/PROGRAM|grep libwrap.so`

# 2.配置文件及格式

- 配置文件：/etc/hosts.allow, /etc/hosts.deny
- 帮助参考：man 5 hosts_access，man 5 hosts_options
- 检查顺序：hosts.allow，hosts.deny（默认允许） **注意：一旦前面规则匹配，直接生效，将不再继续**
- 基本语法 `daemon_list@host: client_list [ :options :option… ]`
- Daemon_list@host格式

```
单个应用程序的二进制文件名，而非服务名，例如vsftpd
以逗号或空格分隔的应用程序文件名列表，如:sshd,vsftpd
ALL表示所有接受tcp_wrapper控制的服务程序
主机有多个IP，可用@hostIP来实现控制
```

- 如：in.telnetd@192.168.0.254
- 客户端Client_list格式

```
以逗号或空格分隔的客户端列表
基于IP地址：192.168.10.1   192.168.1.
基于主机名：www.magedu.com  .magedu.com 较少用
基于网络/掩码：192.168.0.0/255.255.255.0
基于net/prefixlen: 192.168.1.0/24（CentOS7）
基于网络组（NIS 域）：@mynetwork
内置ACL：ALL，LOCAL，KNOWN，UNKNOWN，PARANOID
```

- EXCEPT用法：
  - 示例 `vsftpd: 172.16. EXCEPT 172.16.100.0/24 EXCEPT 172.16.100.1`
  - 示例：只允许192.168.1.0/24的主机访问sshd

```
/etc/hosts.allow
    sshd:192.168.1.
/etc/hosts.deny
    sshd:ALL
```

- 示例：只允许192.168.1.0/24的主机访问telnet和vsftpd服务

```
etc/hosts.allow
    vsftpd,in.telnetd: 192.168.1.
/etc/host.deny
    vsftpd,in.telnetd:  ALL
```

- [:options]选项：
- 帮助：man 5 hosts_options
- deny 主要用在/etc/hosts.allow定义“拒绝”规则
    - 如：vsftpd: 172.16. :deny
- allow 主要用在/etc/hosts.deny定义“允许”规则
    - 如：vsftpd:172.16. :allow
- spawn 启动一个外部程序完成执行的操作
- twist 实际动作是拒绝访问,使用指定操作替换当前服务,标准输出和ERROR发送到客户端,默认至/dev/null
- 测试工具 `tcpdmatch [-d] daemon[@host] client -d 测试当前目录下的hosts.allow和hosts.deny`
- 示例

```
sshd: ALL :spawn echo "$(date +%%F) login attempt from %c to %s,%d" >>/var/log/sshd.log
```

- 说明

```
在/etc/hosts.allow中添加，允许登录，并记录日志
在/etc/hosts.deny中添加，拒绝登录，并记录日志
%c 客户端信息
%s 服务器端信息
%d 服务名
%p 守护进程的PID
%% 表示%
```

- 例如：`vsftpd: 172.16. :twist /bin/echo "connection prohibited"`

## 3.练习

- 仅开放本机两个IP地址中的一个地址172.16.0.X上绑定的sshd和vsftpd服务给172.16.0.0/16 网络中除了172.16.0.0/24网络中的主机之外的所有主机，但允许172.16.0.200访问,每次的用 户访问都要记录于日志文件中，注：其中X为学号

```
vim /etc/hosts.allow
sshd,vsftpd@172.16.0.x : 172.16.0.0/16 EXPECT 172.16.0.0/24 EXPECT 172.16.0.200
```

```
:spawn echo
"`date` login attempt from %c to %s,%d" >> /var/log/sshd.log
```

- 编写脚本/root/bin/checkip.sh，每5分钟检查一次，如果发现通过ssh登录失次数超过10次，自动 将此远程IP放入Tcp Wrapper的黑名单中予以禁止防问

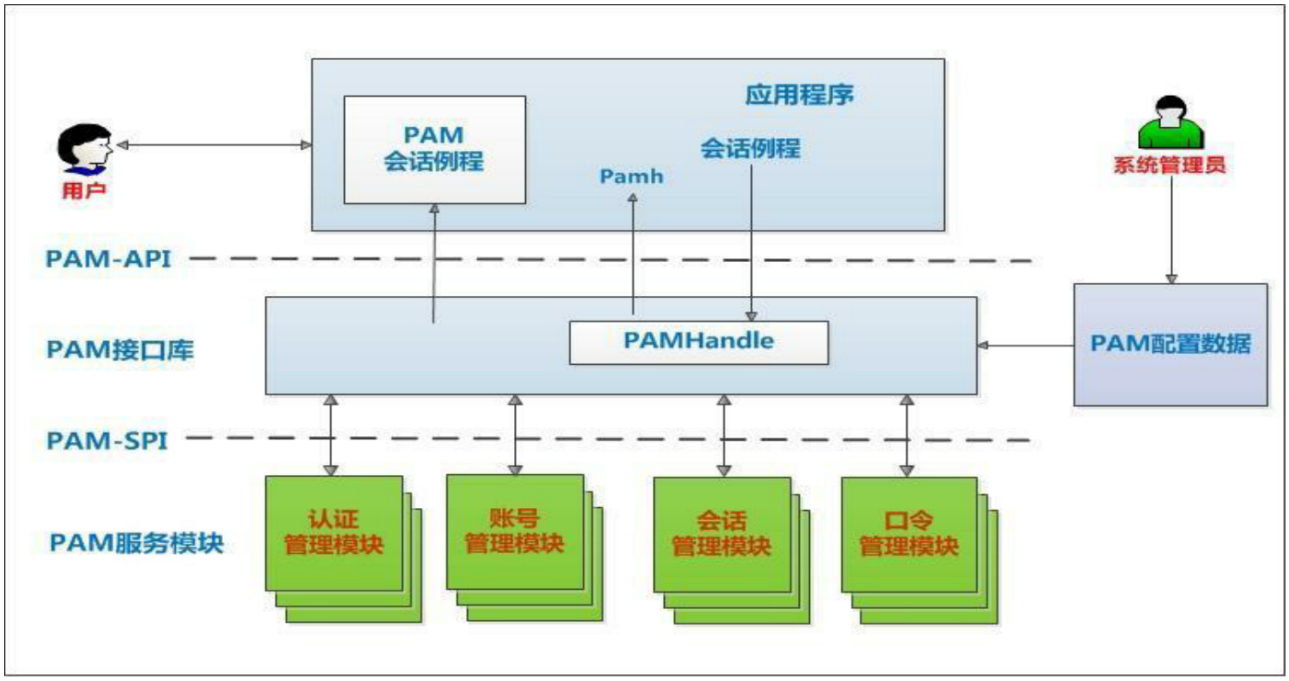
```
#!/bin/bash
while true;do
    cat /var/log/secure|awk '/Failed password/{ip[$(NF-3)]++}END{for(i in ip)
{if(ip[i]>=10)
{system("echo sshd:"i" >> /etc/hosts.deny" )}}}'
    sleep 5m
done
```

# 十三.PAM模块

## 1.PAM介绍

- PAM:Pluggable Authentication Modules
- 认证库包括：文本文件，MySQL，NIS，LDAP等
- Sun公司于1995 年开发的一种与认证相关的通用框架机制
- PAM 是关注如何为服务验证用户的 API，通过提供一些动态链接库和一套统一的API，将系统 提供的服务和该服务的认证方式分开
- 使得系统管理员可以灵活地根据需要给不同的服务配置不同的认证方式而无需更改服务程序
- 一种认证框架，自身不做认证
- 它提供了对所有服务进行认证的中央机制，适用于本地登录，远程登录，如：telnet,rlogin, fsh,ftp,点对点协议PPP，su等应用程序中，系统管理员通过PAM配置文件来制定不同应用程序的 不同认证策略；应用程序开发者通过在服务程序中使用PAM API(pam_xxxx( ))来实现对认证方 法的调用；而PAM服务模块的开发者则利用PAM SPI来编写模块（主要调用函数pam_sm_xxxx( ) 供PAM接口库调用，将不同的认证机制加入到系统中；PAM接口库（libpam）则读取配置 文件，将应用程序和相应的PAM服务模块联系起来。

- PAM框架

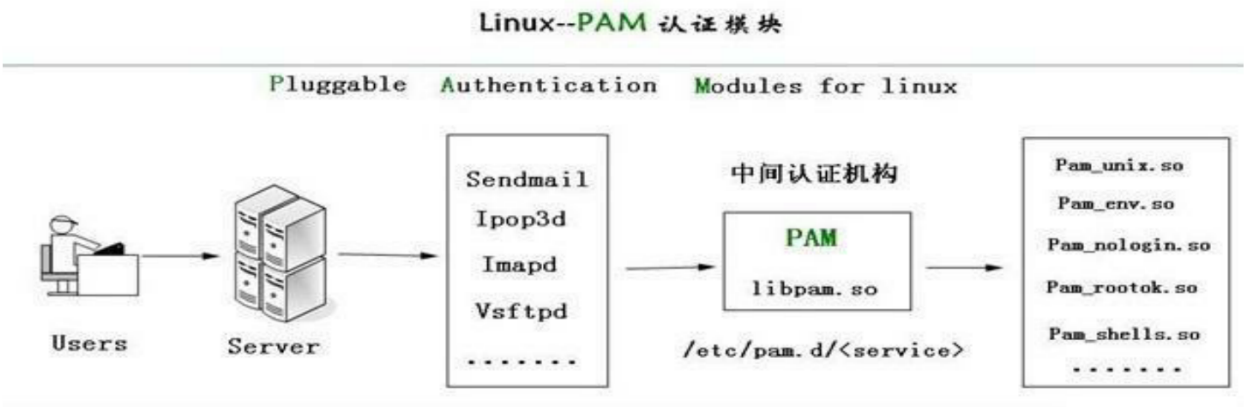



## 2.PAM相关文件

- 模块文件目录：/lib64/security/*.so
- 环境相关的设置：/etc/security/
- 主配置文件：/etc/pam.conf，默认不存在
- 为每种应用模块提供一个专用的配置文件：/etc/pam.d/APP_NAME
- 注意：如/etc/pam.d存在，/etc/pam.conf将失效

## 3.pam认证原理

- PAM认证一般遵循这样的顺序：Service(服务)→PAM(配置文件)→pam_*.so
- PAM认证首先要确定哪一项服务，然后加载相应的PAM的配置文件(位于/etc/pam.d下) ,最后调用认证文件(位于/lib64/security下)进行安全认证
- Linux-PAM认证模块




## 4.PAM认证过程：

```
1.使用者执行/usr/bin/passwd 程序，并输入密码
2.passwd开始调用PAM模块，PAM模块会搜寻passwd程序的PAM相关设置文件，这个设置文件一般是在/etc/pam.d/里边的与程序同名的文件，即PAM会搜寻/etc/pam.d/passwd此设置文件
3.经由/etc/pam.d/passwd设定文件的数据，取用PAM所提供的相关模块来进行验证
4.将验证结果回传给passwd这个程序，而passwd这个程序会根据PAM回传的结果决定下一个动作（重新输入密码或者通过验证）
```

# 5.配置文件及格式

- 通用配置文件/etc/pam.conf格式 `application type control module-path arguments`
- 专用配置文件/etc/pam.d/* 格式 `type control module-path arguments`
- 说明
    - 服务名（application） `telnet、login、ftp等，服务名字“OTHER”代表所有没有在该文件中明确配置的其它服务模块类型（module-type）`
    - control PAM库该如何处理与该服务相关的PAM模块的成功或失败情况
    - module-path 用来指明本模块对应的程序文件的路径名
    - Arguments 用来传递给该模块的参数
- 模块类型（module-type）

```
Auth    账号的认证和授权
Account 与账号管理相关的非认证类的功能，如：用来限制/允许用户对某个服务的访问时间，当前有效的系统资源(最多可以有多少个用户)，限制用户的位置(例如：root用户只能从控制台登录)
Password 用户修改密码时密码复杂度检查机制等功能
Session 用户获取到服务之前或使用服务完成之后需要进行一些附加的操作，如：记录打开/关闭数据的信息，监视目录等
-type 表示因为缺失而不能加载的模块将不记录到系统日志,对于那些不总是安装在系统上的模块有用
```

- Control
    - PAM库如何处理与该服务相关的PAM模块成功或失败情况
- 两种方式实现:简单和复杂
- 简单方式实现：一个关健词实现

```
required
一票否决，表示本模块必须返回成功才能通过认证，但是如果该模块返回失败，失败结果也不会立即通知用户，而是要等到同一type中的所有模块全部执行完毕再将失败结果返回给应用程序，即为必要条件
requisite
一票否决，该模块必须返回成功才能通过认证，但是一旦该模块返回失败，将不再执行同一type内的任何模块，而是直接将控制权返回给应用程序。是一个必要条件
sufficient
一票通过，表明本模块返回成功则通过身份认证的要求，不必再 行同一type内的其它模块，但如果本模块返回失败可忽略，即为充分条件
optional
```

```
表明本模块是可选的，它的成功与否不会对身份认证起关键作用，其返回值一般被忽略
include：  调用其他的配置文件中定义的配置信息
```

- 复杂详细实现：使用一个或多个“status=action” `[status1=action1 status2=action …]`
  - Status:检查结果的返回状态
  - Action:采取行为 ok，done，die，bad，ignore，reset

```
ok      模块通过，继续检查
done    模块通过，返回最后结果给应用
bad     结果失败，继续检查
die     结果失败，返回失败结果给应用
ignore 结果忽略，不影响最后结果
reset   忽略已经得到的结果
```

- module-path: 模块路径
  - 相对路径

```
/lib64/security目录下的模块可使用相对路径
如：pam_shells.so、pam_limits.so
```

- 绝对路径
- 模块通过读取配置文件完成用户对系统资源的使用控制 `/etc/security/*.conf`
- 注意：修改PAM配置文件将马上生效
- 建议：编辑pam规则时，保持至少打开一个root会话，以防止root身份验证错误
- Arguments 用来传递给该模块的参数

# 6.pam文档说明

```
/user/share/doc/pam-*
rpm -qd pam
man -k pam_
man 模块名 如man rootok
<The Linux-PAM System  Administrators' Guide>
```

# 7.PAM模块示例

例子1

```
模块：pam_shells
功能：检查有效shell
man pam_shells
示例：不允许使用/bin/csh的用户本地登录
vim /etc/pam.d/login
```

```
    auth required pam_shells.so
vim /etc/shells
   去掉 /bin/csh
useradd -s /bin/csh testuser
testuser将不可登录
tail /var/log/secure
```

例子2

```
模块：pam_securetty.so
功能：只允许root用户在/etc/securetty列出的安全终端上登陆
示例：允许root在telnet登陆
vi /etc/pam.d/remote
#auth required pam_securetty.so #将这一行加上注释
或者/etc/securetty文件中加入
pts/0,pts/1…pts/n
```

例子3

```
模块：pam_nologin.so
功能：
如果/etc/nologin文件存在,将导致非root用户不能登陆
如果用户shell是/sbin/nologin 时，当该用户登陆时，会显示/etc/nologin
文件内容，并拒绝登陆
```

例子3

```
模块：pam_limits.so
功能：在用户级别实现对其可使用的资源的限制，例如：可打开的文件数量，
可运行的进程数量，可用内存空间
修改限制的实现方式：
    (1) ulimit命令，立即生效，但无法保存
        -n    每个进程最多的打开的文件描述符个数
        -u    最大用户进程数
        -S    使用 soft（软）资源限制
        -H    使用 hard（硬）资源限制
    (2) 配置文件：/etc/security/limits.conf, /etc/security/limits.d/*.conf
配置文件：每行一个定义；
    <domain>        <type>  <item>  <value>
```

```
ulimit [-HSTabcdefilmnpqrstuvx [limit]]
              Provides control over the resources available to the shell and to
processes started by it,
              on  systems  that allow such control.  The -H and -S options specify
```

```
that the hard or soft
              limit is set for the given resource.  A hard limit cannot be
increased by a non-root  user
              once  it is set; a soft limit may be increased up to the value of
the hard limit.  If nei-
              ther -H nor -S is specified, both the soft and hard limits are set.
The  value  of  limit
              can  be a number in the unit specified for the resource or one of
the special values hard,
              soft, or unlimited, which stand for the current hard limit, the
current soft limit, and no
              limit,  respectively.   If  limit  is  omitted, the current value of
the soft limit of the
              resource is printed, unless the -H option is given.  When more than
one resource is speci-
              fied, the limit name and unit are printed before the value.  Other
options are interpreted
              as follows:
              -a     All current limits are reported
              -b     The maximum socket buffer size
              -c     The maximum size of core files created
              -d     The maximum size of a process's data segment
              -e     The maximum scheduling priority ("nice")
              -f     The maximum size of files written by the shell and its
children
              -i     The maximum number of pending signals
              -l     The maximum size that may be locked into memory
              -m     The maximum resident set size (many systems do not honor this
limit)
              -n     The maximum number of open file descriptors (most systems do
not allow  this  value
                     to be set)
              -p     The pipe size in 512-byte blocks (this may not be set)
              -q     The maximum number of bytes in POSIX message queues
              -r     The maximum real-time scheduling priority
              -s     The maximum stack size
              -t     The maximum amount of cpu time in seconds
              -u     The maximum number of processes available to a single user
              -v     The  maximum  amount of virtual memory available to the shell
and, on some systems,
                     to its children
              -x     The maximum number of file locks
              -T     The maximum number of threads

              If limit is given, it is the new value of the specified resource
(the -a option is display
              only).   If  no  option is given, then -f is assumed.  Values are in
1024-byte increments,
              except for -t, which is in seconds, -p, which is in units of 512-
byte blocks, and -T,  -b,
              -n, and -u, which are unscaled values.  The return status is 0
unless an invalid option or
              argument is supplied, or an error occurs  while  setting  a  new
```

```
limit.   In  POSIX  Mode
             512-byte blocks are used for the `-c' and `-f' options.
```

- 翻译如下

| | | |
|---|---|---|
| -H | 设置硬资源限制，一旦设置不能增加。 | ulimit – Hs 64；限制硬资源，线程栈大小为 64K。 |
| -S | 设置软资源限制，设置后可以增加，但是不能超过硬资源设置。 | ulimit – Sn 32；限制软资源，32 个文件描述符。 |
| -a | 显示当前所有的 limit 信息。 | ulimit – a；显示当前所有的 limit 信息。 |
| -c | 最大的 core 文件的大小， 以 blocks 为单位。 | ulimit – c unlimited； 对生成的 core 文件的大小不进行限制。 |
| -d | 进程最大的数据段的大小，以 Kbytes 为单位。 | ulimit -d unlimited；对进程的数据段大小不进行限制。 |
| -f | 进程可以创建文件的最大值，以 blocks 为单位。 | ulimit – f 2048；限制进程可以创建的最大文件大小为 2048 blocks。 |
| -l | 最大可加锁内存大小，以 Kbytes 为单位。 | ulimit – l 32；限制最大可加锁内存大小为 32 Kbytes。 |
| -m | 最大内存大小，以 Kbytes 为单位。 | ulimit – m unlimited；对最大内存不进行限制。 |
| -n | 可以打开最大文件描述符的数量。 | ulimit – n 128；限制最大可以使用 128 个文件描述符。 |
| -p | 管道缓冲区的大小，以 Kbytes 为单位。 | ulimit – p 512；限制管道缓冲区的大小为 512 Kbytes。 |
| -s | 线程栈大小，以 Kbytes 为单位。 | ulimit – s 512；限制线程栈的大小为 512 Kbytes。 |
| -t | 最大的 CPU 占用时间，以秒为单位。 | ulimit – t unlimited；对最大的 CPU 占用时间不进行限制。 |
| -u | 用户最大可用的进程数。 | ulimit – u 64；限制用户最多可以使用 64 个进程。 |
| -v | 进程最大可用的虚拟内存，以 Kbytes 为单位。 | ulimit – v 200000；限制最大可用的虚拟内存为 200000 Kbytes。 |

- pam_limits.so
- 应用于哪些对象

```
Username 单个用户
@group 组内所有用户
*  所有用户
```

- 限制的类型

```
Soft 软限制,普通用户自己可以修改
Hard 硬限制,由root用户设定，且通过kernel强制生效
-  二者同时限定
```

- 限制的资源

```
nofile 所能够同时打开的最大文件数量,默认为1024
nproc 所能够同时运行的进程的最大数量,默认为1024
```

- 指定具体值
- 限制用户最多打开的文件数和运行进程数 `/etc/pam.d/system-auth` session required pam_limits.so
  `vim /etc/security/limits.conf apache – nofile 10240` 用户apache可打开10240个文件
  `student hard nproc 20` 用户student不能运行超过20个进程

/etc/security/limits.ocnf生产案例

```
*               soft    core            unlimited
*               hard    core            unlimited
*               soft    nproc           1000000
*               hard    nproc           1000000
*               soft    nofile          1000000
*               hard    nofile          1000000
*               soft    memlock         32000
*               hard    memlock         32000
*               soft    msgqueue        8192000
*               hard    msgqueue        8192000
```