# NSD SECURITY DAY01

# 1：Linux基本防护措施

## 用户账号安全

### chage工具设置/查看账号有效期

[root@client-50 ~]# useradd tom

[root@client-50 ~]# chage -l tom

最近一次密码修改时间 ：3月 06, 2019

密码过期时间 ：从不

密码失效时间 ：从不

帐户过期时间 ：从不

两次改变密码之间相距的最小天数 ：0

两次改变密码之间相距的最大天数 ：99999

在密码过期之前警告的天数 ：7

[root@client-50 ~]# chage -l #查看用法

用法：chage [选项] 登录

选项：

-d, --lastday 最近日期 将最近一次密码设置时间设为“最近日期”

-E, --expiredate 过期日期 将帐户过期时间设为“过期日期”

-h, --help 显示此帮助信息并推出

-I, --inactive INACITVE 过期 INACTIVE 天数后，设定密码为失效状态

-l, --list 显示帐户年龄信息

-m, --mindays 最小天数 将两次改变密码之间相距的最小天数设为“最小天数”

-M, --maxdays 最大天数 将两次改变密码之间相距的最大天数设为“最大天数”

-R, --root CHROOT\_DIR chroot 到的目录

-W, --warndays 警告天数 将过期警告天数设为“警告天数”

[root@client-50 ~]# chage -E 2019-03-31 tom

### passwd命令对账号的锁定解锁

不让用户登录的两种办法:

1,修改shell为/sbin/nologin

2,锁定

passwd命令:

-l 锁定 -u 解锁 -s 查看状态

[root@client-50 ~]# grep tom /etc/shadow

tom:!!:17961:0:99999:7::17986:

[root@client-50 ~]# passwd -l tom

锁定用户 tom 的密码 。

passwd: 操作成功

[root@client-50 ~]# passwd -u tom

解锁用户 tom 的密码。

passwd: 警告：未锁定的密码将是空的。

passwd: 不安全的操作(使用 -f 参数强制进行该操作)

[root@client-50 ~]# passwd -s tom

passwd：错误的参数 -s：未知的选项

[root@client-50 ~]# passwd -S tom

tom LK 2019-03-06 0 99999 7 -1 (密码已被锁定。)

## 强制定期修改密码

### /etc/login.defs用户默认属性配置文件

新建用户的默认属性配置文件

PASS\_MAX\_DAYS 99999 最长天数

PASS\_MIN\_DAYS 0 最短天数

PASS\_MIN\_LEN 5 长度

PASS\_WARN\_AGE 7

### /etc/skel/ 用户初始环境目录

添加用户后家目录下都有隐藏文件,系统登录系统的配置文件

添加用户后自动从/etc/skel/ 目录下拷贝所有配置文件到用户家目录里面

利用此特性,在/etc/skel/预先放入文件,新建用户之后,每个用户家目录都会有此文件

[root@client-50 ~]# ls -a /etc/skel/

. .. .bash\_logout .bash\_profile .bashrc .mozilla

[root@client-50 ~]# touch /etc/skel/1.txt

[root@client-50 ~]# user add jerry

bash: user: 未找到命令...

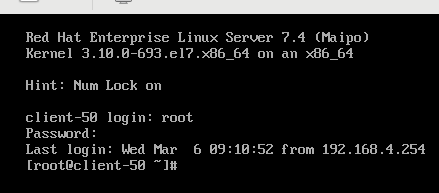
[root@client-50 ~]# useradd jerry

[root@client-50 ~]# ls -a /home/jerry/

. .. 1.txt .bash\_logout .bash\_profile .bashrc .mozilla

所有新建用户的目录下都会多1.txt文件了

## 伪装登录提示



配置文件:

/etc/issue 系统版本和内核版本信息隐藏

/etc/issue.net 系统网络连接隐藏

/etc/redhat-release 版本信息

修改文件内容即可

## 文件系统安全

### 程序和服务控制

禁用非必要的系统服务

RHEL6 开启自启 chkconfig httpd on

RHEL6 开启不自启 chkconfig httpd off

RHEL7 开启自启 systemctl enable httpd

RHEL7 开启不自启 systemctl disable httpd

### 锁定/解锁保护文件

重要的文件系统不允许被随便修改

lsattr 查看有没有特殊属性

chattr 添加(+)/删除(-)文件属性

chattr命令用于设置文件的隐藏权限，格式为：“chattr [参数] 文件”

-i 将无法对文件进行修改,

若对目录设置后则仅能修改子文件与复制子文件,而不能新建或删除。

-a 仅允许补充（追加）内容.无法覆盖/删除(Append Only)。

[root@client-50 ~]# mkdir test

[root@client-50 ~]# echo a >test/a.txt

[root@client-50 ~]# ls test/

a.txt

[root@client-50 ~]# cat test/a.txt

a

[root@client-50 ~]# chattr +i test

[root@client-50 ~]# lsattr -d test

----i----------- test

[root@client-50 ~]# rm test/a.txt

rm：是否删除普通文件 "test/a.txt"？y

rm: 无法删除"test/a.txt": 权限不够

[root@client-50 ~]# vim test/a.txt #vim可修改

[root@client-50 ~]# cat test/a.txt

asdfasdfa

[root@client-50 ~]# mv test/a.txt test/1.txt

mv: 无法将"test/a.txt" 移动至"test/1.txt": 权限不够

[root@client-50 ~]# touch test/b.txt

touch: 无法创建"test/b.txt": 权限不够

[root@client-50 ~]# cp test/a.txt b.txt

[root@client-50 ~]# ls test

a.txt

[root@client-50 ~]# lsattr test

---------------- test/a.txt

# 2：用户切换与提权

### su用户切换时带”-” 与不带”-”

[root@client-50 ~]# id tom

uid=1001(tom) gid=1001(tom) 组=1001(tom)

[root@client-50 ~]# passwd -S tom

tom PS 2019-03-06 0 99999 7 -1 (密码已设置，使用 SHA512 算法。)

[root@client-50 ~]# echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

[root@client-50 ~]# echo $PWD

/root

[root@client-50 ~]# whoami

root

[root@client-50 ~]# su tom #不带” - ”切换

[tom@client-50 root]$ echo $PATH #查看系统环境,还是root环境中

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

[tom@client-50 root]$ echo $PWD

/root

[tom@client-50 root]$ whoami

tom

[tom@client-50 root]$ su - tom

密码：

[tom@client-50 root]$ exit

exit

[root@client-50 ~]# su - tom

上一次登录：三 3月 6 10:34:55 CST 2019pts/0 上

[tom@client-50 ~]$ echo $PATH

/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/tom/.local/bin:/home/tom/bin

[tom@client-50 ~]$ echo $PWD

/home/tom

[tom@client-50 ~]$ whoami

tom

[tom@client-50 ~]$ su - #不输入用户名则切换root用户

密码：

上一次登录：三 3月 6 09:47:35 CST 2019tty1 上

[root@client-50 ~]# exit

### su - -c “命令” 用户

-c 变更为帐号为 root用户的使用者并执行指令（command）后再变回原来使用者

[tom@client-50 ~]$ su - -c "touch /root/tom.txt" root

密码：

[tom@client-50 ~]$

[tom@client-50 ~]$ ls /root

ls: 无法打开目录/root: 权限不够

[tom@client-50 ~]$ exit

登出

[root@client-50 ~]# ls

tom.txt

**/var/log/secure 记录用户执行过su验证的操作日志**

### sudo 用户提权

sudo是linux下常用的允许普通用户使用超级用户权限的工具，允许系统管理员让普通用户执行一些或者全部的root命令，如halt，reboot，su等等。这样不仅减少了root用户的登陆和管理时间，同样也提高了安全性。Sudo不是对shell的一个代替，它是面向每个命令的。

**它的特性主要有这样几点**：

1、sudo能够限制用户只在某台主机上运行某些命令。

2、sudo提供了丰富的日志，详细地记录了每个用户干了什么。它能够将日志传到中心主机或者日志服务器。

3、sudo使用时间戳文件--日志 来执行类似的“检票”系统。当用户调用sudo并且输入它的密码时，用户获得了一张存活期为5分钟的票（这个值可以在编译的时候改变）。

4、sudo的配置文件是/etc/sudoers，属性必须为0440，它允许系统管理员集中的管理用户的使用权限和使用的主机。

**编辑配置文件命令:visudo**

注意：编辑sudo的配置文件/etc/sudoers是一般不要直接使用vi（vi /etc/sudoers）去编辑，因为sudoers配置有一定的语法，直接用vi编辑保存系统不会检查语法，如有错也保存了可能导致无法使用sudo工具，最好使用visudo命令去配置。虽然visudo也是调用vi去编辑，但是保存时会进行语法检查，有错会有提示。

visudo命令去配置 保存是直接 wq

vim 编辑工具去配置 保存加”!” wq!

给用户提权操作

编辑/etc/sudoers 文件.添加权限

提权格式:

**sudo命令用法**

sudo -l #查看自己的sudo授权

sudo 特权命令

sudo -u \*用户名 “命令” #以\*用户的权限执行命令

参数说明：

-V 显示版本编号

-h 会显示版本编号及指令的使用方式说明

-l 显示出自己（执行 sudo 的使用者）的权限

-v 因为 sudo 在第一次执行时或是在 N 分钟内没有执行（N 预设为五）会问密码，这个 参数是重新做一次确认，如果超过 N 分钟，也会问密码

-k 将会强迫使用者在下一次执行 sudo 时问密码（不论有没有超过 N 分钟）

-b 将要执行的指令放在背景执行

-p prompt 可以更改问密码的提示语，其中 %u 会代换为使用者的帐号名称， %h 会显 示主机名称

-u username/#uid 不加此参数，代表要以 root 的身份执行指令，而加了此参数，可以 以 username 的身份执行指令（#uid 为该 username 的使用者号码）

-s 执行环境变数中的 SHELL 所指定的 shell ，或是 /etc/passwd 里所指定的 shell

-H 将环境变数中的 HOME （家目录）指定为要变更身份的使用者家目录（如不加 -u 参 数就是系统管理者 root ）

command 要以系统管理者身份（或以 -u 更改为其他人）执行的指令

[root@client-50 ~]# vim /etc/sudoers

91 ## Allow root to run any commands anywhere

92 root ALL=(ALL) ALL

98 ## Allows people in group wheel to run all commands

99 #%wheel ALL=(ALL) ALL #wheel组权限,过于危险,一般注释掉

101 ## Same thing without a password

102 # %wheel ALL=(ALL) NOPASSWD: ALL #

**解释:**

root ALL=(ALL:ALL) ALL

zdd ALL=(ALL) ALL

zdd:表示 zdd 用户。

第一个字段zdd指定的是用户：可以是用户名，也可以是别名。每个用户设置一行，多个用户设置多行，也可以将多个用户设置成一个别名后再进行设置。

ALL:表示从任何的主机上都可以执行，也可以这样 192.168.100.0/24。

第二个字段ALL指定的是用户所在的主机：可以是ip,也可以是主机名，表示这个sudo设置后的文件即/etc/sudoers只在这个主机上生效，ALL表示在所有主机上都生效！限制的一般都是本机，也就是限制使用这个文件的主机;如果指定为"192.168.1.88="表示这个文件只有在这台主机上生效，如果拷贝到别的机子上是不起作用的！一般都指定为"ALL"表示所有的主机，不管文件拷到那里都可以用。

(ALL):是以谁的身份来执行，ALL就代表 root 可以任何人的身份来执行命令。

第三个字段（ALL）括号里指定的也是用户：指定以什么用户身份执行sudo，即使用sudo后可以享有所有账号下的权限。如果要排除个别用户，可以在括号内设置，比如ALL=(ALL,!root,!ops)。也可以设置别名

ALL :表示任何命令。

第四个字段ALL指定的是执行的命令：即使用sudo后可以执行所有的命令。也可以设置别名。NOPASSWD: ALL表示使用sudo的不需要输入密码。

%wheel 表示wheel组提权权限

NOPASSWD: ALL 执行命令时不验证密码

mike localhost,client-50=/sbin/\* , !/sbin/ifconfig eth0

\*号通配所有 !取反设置

### sudo操作实例

[root@client-50 ~]# useradd dachui

[root@client-50 ~]# useradd tiechui

[root@client-50 ~]# useradd niuben

[root@client-50 ~]# echo 123456 | passwd --stdin dachui

更改用户 dachui 的密码 。

passwd：所有的身份验证令牌已经成功更新。

[root@client-50 ~]# echo 123456 | passwd --stdin tiechui

更改用户 tiechui 的密码 。

passwd：所有的身份验证令牌已经成功更新。

[root@client-50 ~]# echo 123456 | passwd --stdin niuben

更改用户 niuben 的密码 。

passwd：所有的身份验证令牌已经成功更新。

[root@client-50 ~]# which vim #查看命令的绝对路径

/usr/bin/vim

[root@client-50 ~]# which systemctl

/usr/bin/systemctl

**开始提权**

[root@client-50 ~]# vim /etc/sudoers

91 ## Allow root to run any commands anywhere

92 root ALL=(ALL) ALL

93 dachui localhost,client-50=/usr/bin/vim /etc/httpd/conf/httpd.conf , /usr/b

in/systemctl \* httpd #新加提权命令

[root@client-50 ~]# su - dachui

[dachui@client-50 ~]$ sudo -l

我们信任您已经从系统管理员那里了解了日常注意事项。

总结起来无外乎这三点：

#1) 尊重别人的隐私。

#2) 输入前要先考虑(后果和风险)。

#3) 权力越大，责任越大。

[sudo] dachui 的密码：

匹配 %2$s 上 %1$s 的默认条目：

....

....

用户 dachui 可以在 client-50 上运行以下命令：

(root) /usr/bin/vim /etc/httpd/conf/httpd.conf, /usr/bin/systemctl \* httpd

[dachui@client-50 ~]$

### sudo别名设置

提高可重用性 易读性 简化配置,使记录更有条理

别名名称必须全大写

此外sudo还支持别名的定义，我们通过引用定义好的别名可以提供工作效率：

who User\_Alias

which\_hosts Host\_Alias

runas Runas\_Alias

command Cmnd\_Alias

User\_Alias Host\_Alias Runas\_Alias Cmnd\_Alias

用户名 主机名 用户名 命令路径（全路径）

组名(%) IP 地址 组名（%） 目录

其他User\_Alias 网络地址 其他Runas\_Alias 其他Cmnd\_Alias

其他Host\_Alias

[root@client-50 ~]# vim /etc/sudoers

**#普通提权方式:**

用户名 ALL=(ALL) ALL

dachui localhost,client-50=/usr/bin/vim /etc/httpd/conf/httpd.conf , /usr/bin/systemctl \* httpd

**#定义别名并调用使用:**

Cmnd\_Alias MYCMND=/usr/bin/vim /etc/httpd/conf/httpd.conf , /usr/bin/systemctl \* httpd , /usr/bin/vim /etc/my.cnf , /usr/bin/systemctl \* mysqld

Cmnd\_Alias SOFTMGM=/bin/yum , /bin/rpm

Host\_Alias MYPC=localhost,client-50

#定义好之后使用别名给niuben用户提权

niuben MYPC=MYCMND,SOFTMGM,/sbin\*,!/sbin/ifconfig eth0

[root@client-50 ~]# su - niuben

上一次登录：三 3月 6 11:54:34 CST 2019从 192.168.4.50pts/2 上

[niuben@client-50 ~]$ sudo -l

我们信任您已经从系统管理员那里了解了日常注意事项。

....

[sudo] niuben 的密码：

....

用户 niuben 可以在 client-50 上运行以下命令：

(root) /usr/bin/vim /etc/httpd/conf/httpd.conf, /usr/bin/systemctl \* httpd, /usr/bin/vim /etc/my.cnf, /usr/bin/systemctl \* mysqld, /bin/yum, /bin/rpm,

/sbin\*, !/sbin/ifconfig eth0

**在/etc/sudoers 中定义了很多功能别名,去掉注释即可调用使用**

分析sudo提权使用日志

[root@client-50 ~]# vim /etc/sudoers

Defaults logfile="/var/log/sudo" #文本最后一行添加

[niuben@client-50 ~]$ sudo -l

[root@client-50 ~]# cat /var/log/sudo

Mar 6 14:41:03 : niuben : TTY=pts/1 ; PWD=/home/niuben ; USER=root ;

COMMAND=list

[root@client-50 ~]# cat /var/log/sudo

Mar 6 14:41:03 : niuben : TTY=pts/1 ; PWD=/home/niuben ; USER=root ;

COMMAND=list #操作内容

Mar 6 14:42:29 : niuben : TTY=pts/1 ; PWD=/home/niuben ; USER=root ;

COMMAND=/bin/vim /etc/httpd/conf/httpd.conf #操作内容

# 3：SSH访问控制

SSH(远程连接工具)连接原理：ssh服务是一个守护进程(demon)，系统后台监听客户端的连接，ssh服务端的进程名为sshd,负责实时监听客户端的请求(IP 22端口)，包括公共秘钥等交换等信息。

ssh服务端由2部分组成： openssh(提供ssh服务) openssl(提供加密的程序)

ssh的客户端可以用 XSHELL，Securecrt, Mobaxterm等工具进行连接

**SSH的工作机制**

服务器启动的时候自己产生一个密钥(768bit公钥)，本地的ssh客户端发送连接请求到ssh服务器，服务器检查连接点客户端发送的数据和IP地址，确认合法后发送密钥(768bits)给客户端，此时客户端将本地私钥(256bit)和服务器的公钥(768bit)结合成密钥对key(1024bit),发回给服务器端，建立连接通过key-pair数据传输。

**SSH的加密技术**

加密技术：传输过程，数据加密。

1.SSH1没有对客户端的秘钥进行校验，很容易被植入恶意代码

2.SSH2增加了一个确认联机正确性的Diffe\_Hellman机制，每次数据的传输，Server都会检查数据来源的正确性，避免黑客入侵。

SSH2支持RSA和DSA密钥

DSA:digital signature Algorithm 数字签名

RSA:既可以数字签名又可以加密

**SSH知识小结**

1.SSH是安全的加密协议，用于远程连接Linux服务器

2.SSH的默认端口是22，安全协议版本是SSH2

3.SSH服务器端主要包含2个服务功能SSH连接和SFTP服务器

4.SSH客户端包含ssh连接命令和远程拷贝scp命令等

**如何防止SSH登录入侵**

1.密钥登录,更改端口

2.牤牛阵法

3.监听本地内网IP(ListenAddress 192.168.25.\*)

**SSH功能大全**

1.登录

ssh -p22 omd@192.168.25.137

2.直接执行命令 -->最好全路径

ssh root@192.168.25.137 ls -ltr /backup/data

==>ssh root@192.168.25.137 /bin/ls -ltr /backup/data

3.查看已知主机

cat /root/.ssh/known\_hosts

4.ssh远程执行sudo命令

ssh -t omd@192.168.25.137 sudo rsync hosts /etc/

5.scp

1.功能 -->远程文件的安全(加密)拷贝

scp -P22 -r -p /home/omd/h.txt omd@192.168.25.137:/home/omd/

2.scp知识小结

scp是加密远程拷贝，cp为本地拷贝

可以推送过去，也可以拉过来

每次都是全量拷贝(效率不高，适合第一次)，增量拷贝用rsync

6.ssh自带的sftp功能

1.Window和Linux的传输工具

wincp filezip

sftp -->基于ssh的安全加密传输

samba

2.sftp客户端连接

sftp -oPort=22 root@192.168.25.137

put /etc/hosts /tmp

get /etc/hosts /home/omd

3.sftp小结：

1.linux下使用命令： sftp -oPort=22 root@x.x.x.x

2.put加客户端本地路径上传

3.get下载服务器端内容到本地

4.远程连接默认连接用户的家目录

**ssh免密设置**

1.根据DSA算法生成私钥和公钥【默认建立在当前用户的家目录】

]# ssh-keygen # 一路回车即可

id\_dsa -->私钥(钥匙)

id\_dsa.pub -->公钥(锁)

1. 拷贝公钥给目标服务器

]# ssh-copy-id 对方IP

## ssh基本防护

**存在安全隐患**

密码嗅探/键盘记录

暴力枚举账号,猜解密码

**常见的防护措施**

用户限制,黑白名单

更改验证方式(密码--秘钥对)

防火墙

**修改端口**

[root@client-50 ~]# vim /etc/ssh/sshd\_config

17 Port 2222

修改端口

[root@client-50 ~]# vim /etc/ssh/sshd\_config

17 Port 2222

[root@client-50 ~]# systemctl restart sshd

[root@client-50 ~]# exit

登出

Connection to 192.168.4.50 closed.

[root@room9pc01 ~]# ssh 192.168.4.50

ssh: connect to host 192.168.4.50 port 22: Connection refused #连接拒绝

[root@room9pc01 ~]# ssh -p2222 192.168.4.50

Last login: Wed Mar 6 14:57:47 2019 from 192.168.4.254

[root@client-50 ~]#

**修改配置**

37 #LoginGraceTime 2m #登录超时2分钟

38 #PermitRootLogin yes #禁止root登录

39 #StrictModes yes

40 #MaxAuthTries 6 #每次连接最多认证6次密码

41 #MaxSessions 10

115 #UseDNS no #不解析客户机地址

**ssh黑白名单配置**

[root@client-50 ~]# vim /etc/ssh/sshd\_config #在文本末尾添加

DenyUsers user1 user2 #黑名单

AllowUsers user1@host user2@ #白名单

DenyGroups group1 group2..

AllowGroups group1 group2 ..

**ssh白名单配置**

[root@client-50 ~]# vim /etc/ssh/sshd\_config #在文本末尾添加

AllowUsers root@192.168.4.254 niuben

[root@client-50 ~]# systemctl restart sshd

真机测试

[root@room9pc01 ~]# ssh -p 2222 niuben@192.168.4.50

niuben@192.168.4.50's password:

Last login: Wed Mar 6 14:32:03 2019

[niuben@client-50 ~]$ exit

登出

[root@room9pc01 ~]# ssh -p 2222 dachui@192.168.4.50

tom@192.168.4.50's password: #密码正确也无法登录.连续三次密码后退出

Permission denied, please try again.

tom@192.168.4.50's password:

Permission denied, please try again.

dachui@192.168.4.50's password:

Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).

重新添加白名单

AllowUsers root@192.168.4.254 niuben dachui

真机测试

[root@room9pc01 ~]# ssh -p 2222 dachui@192.168.4.50

dachui@192.168.4.50's password:

Last failed login: Wed Mar 6 15:38:46 CST 2019 from 192.168.4.254 on ssh:notty

There were 3 failed login attempts since the last successful login.

Last login: Wed Mar 6 11:43:03 2019

[dachui@client-50 ~]$

**ssh黑名单配置**

[root@client-50 ~]# vim /etc/ssh/sshd\_config

DenyUsers dachui

[root@client-50 ~]# systemctl restart sshd

## ss登录方式

### 默认用户密码登录

65 PasswordAuthentication yes #默认开着

### ssh秘钥对验证

[root@client-50 ~]# vim /etc/ssh/sshd\_config

43 #PubkeyAuthentication yes #默认启用

47 AuthorizedKeysFile .ssh/authorized\_keys #公钥文件名称和存储路径

**ssh免密设置**

**1.根据DSA算法生成私钥和公钥【默认建立在当前用户的家目录】**

]# ssh-keygen #一路回车即可,若指定算法(-t rsa 或 -t dsa),不指定默认RSA算法

id\_dsa -->私钥(钥匙)

id\_dsa.pub -->公钥(锁)

**2.拷贝公钥给目标服务器**

]# ssh-copy-id 用户名@需远程免密登录的IP

私钥文件：id\_rsa

公钥文件：id\_rsa.pub

[root@room9pc01 ~]# ls -Al /root/.ssh

总用量 20

-rw-r--r--. 1 root root 404 4月 27 2018 authorized\_keys

-rw-r--r--. 1 root root 1675 2月 14 08:30 id\_rsa

-rw-r--r--. 1 root root 404 2月 14 08:30 id\_rsa.pub

-rw-r--r-- 1 root root 6424 3月 5 17:01 known\_hosts

1. **拷贝id\_rsa私钥文件给别的主机免密登录**

[root@room9pc01 ~]# cd /root/.ssh

[root@room9pc01 .ssh]# ls

authorized\_keys id\_rsa id\_rsa.pub known\_hosts

[root@room9pc01 .ssh]# scp id\_rsa 192.168.4.51:/root/.ssh

[root@sql51 ~]# ls ./.ssh -l

总用量 8

-rw-r--r--. 1 root root 1675 3月 6 16:52 id\_rsa

[root@sql51 ~]# ssh root@192.168.4.50

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ WARNING: UNPROTECTED PRIVATE KEY FILE! @

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Permissions 0644 for '/root/.ssh/id\_rsa' are too open.

It is required that your private key files are NOT accessible by others.

This private key will be ignored.

Load key "/root/.ssh/id\_rsa": bad permissions

Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

解决方法：

系统提示说644对私钥文件来说太奔放了，不够私密.后面两个4代表的含义为：同组的小伙伴可以读，其他小伙伴也可以读。这怎么能行，果断收掉他们的权限，改为600。so执行以下命令就ok啦。

[root@sql51 ~]# chmod 0600 /root/.ssh/id\_rsa

[root@sql51 ~]# ssh root@192.168.4.50

Last login: Wed Mar 6 17:04:25 2019 from 192.168.4.254

[root@client-50 ~]# exit

登出

Connection to 192.168.4.50 closed.

[root@sql51 ~]# ll /root/.ssh/id\_rsa

-rw-------. 1 root root 1675 3月 6 16:52 /root/.ssh/id\_rsa

# 4：SELinux安全防护

SELinux(Security-Enhanced Linux) 是美国国家安全局（NSA）对于强制访问控制的实现，是 Linux历史上最杰出的新安全子系统。NSA是在Linux社区的帮助下开发了一种访问控制体系，在这种访问控制体系的限制下，进程只能访问那些在他的任务中所需要文件。

SELinux是一种基于 域-类型 模型（domain-type）的强制访问控制（MAC）安全系统，它由NSA编写并设计成内核模块包含到内核中，相应的某些安全相关的应用也被打了SELinux的补丁，最后还有一个相应的安全策略。任何程序对其资源享有完全的控制权。假设某个程序打算把含有潜在重要信息的文件扔到/tmp目录下，那么在DAC情况下没人能阻止他。SELinux提供了比传统的UNIX权限更好的访问控制。

## SELinux的运行模式

enforcing(强制):已强制执行SELinux安全策略。

permissive(宽松):SELinux打印警告而不是强制执行

disabled(彻底禁用) :未加载SELinux策略。

变成disabled状态,必须配置/etc/selinux/config文件并重起系统

• 切换运行模式

– 临时切换:setenforce 1|0 1为enforcing模式 0为permissive模式

– 查看当前模式:getenforce

– 固定配置:/etc/selinux/config 文件

## 查看SElinux状态

[root@sql51 ~]# sestatus

SELinux status: enabled

SELinuxfs mount: /sys/fs/selinux #相关文件

SELinux root directory: /etc/selinux #工作根目录

Loaded policy name: targeted

Current mode: permissive #当前模式permissive(宽松)

Mode from config file: permissive

Policy MLS status: enabled

Policy deny\_unknown status: allowed

Max kernel policy version: 28

## 当前类型

[root@client-50 ~]# vim /etc/sysconfig/selinux

SELINUX=permissive

SELINUXTYPE=targeted #当前类型

targeted:有针对性的保护当前机器的常用服务(目标流程受到保护).

minimum: 最低限度修改目标政策。仅保护选定的进程。

mls:全部保护(多级安全保护).

# 5：SELinux策略设置

## -Z查看安全上下文(标签)

加 -Z 查看

[root@client-50 ~]# ps aux -Z | grep httpd

system\_u:system\_r:httpd\_t:s0 root 962 0.0 0.8 232560 8844 ? Ss 09:10 0:01 php-fpm: master process (/etc/php-fpm.conf)

[root@client-50 ~]# ps aux | grep httpd #不加-Z区别

root 3781 0.0 1.2 312900 12212 ? Ss 11:54 0:01 /usr/sbin/httpd -DFOREGROUND

[root@client-50 ~]# ls -lZ /etc/passwd

-rw-r--r--. root root system\_u:object\_r:passwd\_file\_t:s0 /etc/passwd

[root@client-50 ~]# ll -dZ /var/www/html/

drwxr-xr-x. root root system\_u:object\_r:httpd\_sys\_content\_t:s0 /var/www/html/

[root@client-50 ~]# ll -dZ /var/lib/mysql

drwxr-x--x. mysql mysql unconfined\_u:object\_r:mysqld\_db\_t:s0 /var/lib/mysql

用户:角色:访问类型:选择

system\_u:system\_r:httpd\_t:s0

system\_u:object\_r:passwd\_file\_t:s0

system\_u:object\_r:httpd\_sys\_content\_t:s0

## chcon工具修改安全上下文(标签)

使用chcon工具

-t :指定访问类型

-R:递归修改

一般操作规律

移动的文件,会保持原有的上下文属性不变

复制的文件,会自动继承目标目录的访问类型

新建的文件,会自动继承父目录的访问类型

]# chcon -t 访问类型 文件

# chcon -R -t 访问类型 目录

]# chcon -t httpd\_sys\_content\_t a2.html

]# chcon -R -t httpd\_sys\_content\_t /var/www/html

恢复所在位置的默认上下文属性

]# restorecon /var/www/html/a2.html

]# restorecon -R /var/www/html

# 6调整SELinux布尔值

功能开关,

查看SELinux布尔值

设置SELinux布尔值

例如:修改布尔值,允许ftp匿名上传文件,50为ftp服务器,51为客户端

[root@client-50 ~]# yum -y install vsftpd

[root@client-50 ~]# vim /etc/vsftpd/vsftpd.conf

29 anon\_upload\_enable=YES

[root@client-50 ~]# mkdir /var/ftp/sharedir

[root@client-50 ~]# chmod o+w /var/ftp/sharedir

[root@client-50 ~]# cp /etc/passwd /var/ftp/sharedir

[root@client-50 ~]# systemctl restart vsftpd

[root@client-50 ~]# netstat -untlp | grep :21

tcp6 0 0 :::21 :::\* LISTEN 2179/vsftpd

51主机

[root@sql51 ~]# which ftp

/usr/bin/which: no ftp in (/root/perl5/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin)

[root@sql51 myload]# ftp 192.168.4.50

bash: ftp: 未找到命令...

[root@sql51 ~]# mkdir /myload

[root@sql51 ~]# cd /myload

[root@sql51 myload]# yum -y install ftp

[root@sql51 myload]# which ftp

/usr/bin/ftp

[root@sql51 myload]# ftp 192.168

ftp: connect: 网络不可达

ftp> exit

[root@sql51 myload]# ftp 192.168.4.50

Connected to 192.168.4.50 (192.168.4.50).

220 (vsFTPd 3.0.2)

Name (192.168.4.50:root): ftp

331 Please specify the password.

Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> ls

227 Entering Passive Mode (192,168,4,50,237,86).

150 Here comes the directory listing.

drwxr-xr-x 2 0 0 6 Mar 23 2017 pub

drwxr-xrwx 2 0 0 20 Mar 07 01:16 sharedir

226 Directory send OK.

[root@sql51 myload]# ftp 192.168.4.50

Connected to 192.168.4.50 (192.168.4.50).

220 (vsFTPd 3.0.2)

Name (192.168.4.50:root): ftp

331 Please specify the password.

Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> ls

227 Entering Passive Mode (192,168,4,50,20,134).

150 Here comes the directory listing.

drwxr-xr-x 2 0 0 6 Mar 23 2017 pub

drwxr-xrwx 2 0 0 20 Mar 07 01:16 sharedir

226 Directory send OK.

ftp> cd sharedir

250 Directory successfully changed.

ftp>

ftp>

ftp> ls

227 Entering Passive Mode (192,168,4,50,79,150).

150 Here comes the directory listing.

-rw-r--r-- 1 0 0 2374 Mar 07 01:16 passwd

226 Directory send OK.

ftp> get passwd

local: passwd remote: passwd

227 Entering Passive Mode (192,168,4,50,202,41).

150 Opening BINARY mode data connection for passwd (2374 bytes).

226 Transfer complete.

2374 bytes received in 6.6e-05 secs (35969.70 Kbytes/sec)

上传文件

ftp> lcd /etc

Local directory now /etc

ftp> put hosts

local: hosts remote: hosts

227 Entering Passive Mode (192,168,4,50,227,75).

553 Could not create file. #上传失败

修改布尔后再上传

[root@client-50 ~]# setsebool -P ftpd\_anon\_write on

[root@client-50 ~]# setsebool -P ftpd\_full\_access on

这两个修改是在ftp上的配置文件中指定的

51上

ftp> put hosts

local: hosts remote: hosts

227 Entering Passive Mode (192,168,4,50,156,12).

150 Ok to send data.

226 Transfer complete.

158 bytes sent in 3.9e-05 secs (4051.28 Kbytes/sec) #上传成功

# 7查看SELinux日志排错

[root@client-50 ~]# rpm -qa |grep -i setroubles

setroubleshoot-3.2.28-3.el7.x86\_64

setroubleshoot-server-3.2.28-3.el7.x86\_64

setroubleshoot-plugins-3.0.65-1.el7.noarch

[root@client-50 ~]# rpm -qa |grep -i setroubleshoot

setroubleshoot-3.2.28-3.el7.x86\_64

setroubleshoot-server-3.2.28-3.el7.x86\_64

setroubleshoot-plugins-3.0.65-1.el7.noarch

[root@client-50 ~]# ls /var/log/messages #没有单独日志的程序的所有日志记录

/var/log/messages

[root@client-50 ~]# grep setroubleshoot /var/log/messages |tail -1

Mar 7 09:36:42 client-50 setroubleshoot: SELinux is preventing vsftpd from write access on the directory sharedir. For complete SELinux messages run: sealert -l 96d20b8c-e48f-43d6-9e38-4598fac41003

翻译

3月7日09:36:42 client-50 setroubleshoot:selinux正在阻止vsftpd对目录sharedir进行写访问。对于完整的SELinux消息，请运行：sealert-l 96d20b8c-e48f-43d6-9e38-4598fac41003

例子:启动SELinux 后(selinux的端口策略为80),设置本机网站httpd服务端口8060,

[root@client-50 ~]# getenforce

Enforcing #强制执行模式

[root@client-50 ~]# vim /etc/httpd/conf/httpd.conf

Listen 8060

[root@client-50 ~]# systemctl start httpd #修改httpd配置文件断口,启动报错

Job for httpd.service failed because the control process exited with error code. See "systemctl status httpd.service" and "journalctl -xe" for details.

[root@client-50 ~]# grep setroubleshoot /var/log/messages | tail -1 #查看报错

Mar 7 09:55:37 client-50 setroubleshoot: SELinux is preventing /usr/sbin/httpd from name\_bind access on the tcp\_socket port 8060. For complete SELinux messages run: sealert -l 869458fb-03ac-4cdc-9b21-5ccd5f78752b

[root@client-50 ~]# sealert -l 869458fb-03ac-4cdc-9b21-5ccd5f78752b

....

# **semanage port -a -t PORT\_TYPE -p tcp 8060 #日志显示解决办法**

其中 PORT\_TYPE 是以下之一：http\_cache\_port\_t, http\_port\_t, jboss\_management\_port\_t, jboss\_messaging\_port\_t, ntop\_port\_t, puppet\_port\_t。

\*\*\*\*\* Plugin catchall\_boolean (7.83 confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If 要 allow nis to enabled

Then 必须启用 'nis\_enabled' 布尔值告知 SELinux 此情况。

可以阅读 'None' 手册页面来了解详情。

Do

setsebool -P nis\_enabled 1 #?

...

[root@client-50 ~]# **semanage port -a -t http\_port\_t -p tcp 8060**

[root@client-50 ~]# systemctl start httpd #重启成功

[root@client-50 ~]#

# 案例

案例1：Linux基本防护措施

案例2：使用sudo分配管理权限

案例3：提高SSH服务安全

案例4：SELinux安全防护

1 案例1：Linux基本防护措施

1.1 问题

本案例要求练习Linux系统的基本防护措施，完成以下任务：

修改用户zhangsan的账号属性，设置为2019-12-31日失效（禁止登录）

临时锁定用户lisi的账户，使其无法登录，验证效果后解除锁定

修改tty终端提示，使得登录前看到的第一行文本为“Windows Server 2012 Enterprise R2”，第二行文本为“NT 6.2 Hybrid”

锁定文件/etc/resolv.conf、/etc/hosts，以防止其内容被无意中修改

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：修改用户zhangsan的账户属性，设置为2019-12-31日失效（禁止登录）

1）正常情况下，未过期的账号可以正常登录，使用chage可以修改账户有效期。

chage命令的语法格式：

chage –l 账户名称 //查看账户信息

chage –E 时间 账户名称 //修改账户有效期

2）失效的用户将无法登录

使用chage命令将用户zhangsan的账户设为当前已失效（比如已经过去的某个时间）：

[root@proxy ~]# useradd zhangsan

[root@proxy ~]# chage -E 2015-05-15 zhangsan

尝试以用户zhangsan重新登录，输入正确的用户名、密码后直接闪退，返回登录页，说明此帐号已失效。

3）重设用户zhangsan的属性，将失效时间设为2019-12-31

[root@proxy ~]# chage -E 2019-12-31 zhangsan //修改失效日期

[root@proxy ~]# chage -l zhangsan //查看账户年龄信息

Last password change : May 15, 2017

Password expires : never

Password inactive : never

Account expires : Dec 31, 2019

Minimum number of days between password change : 0

Maximum number of days between password change : 99999

Number of days of warning before password expires : 7

4）定义默认有效期（扩展知识）

/etc/login.defs这个配置文件，决定了账户密码的默认有效期。

[root@proxy ~]# cat /etc/login.defs

PASS\_MAX\_DAYS 99999 //密码最长有效期

PASS\_MIN\_DAYS 0 //密码最短有效期

PASS\_MIN\_LEN 5 //密码最短长度

PASS\_WARN\_AGE 7 //密码过期前几天提示警告信息

UID\_MIN 1000 //UID最小值

UID\_MAX 60000 //UID最大值

步骤二：临时锁定用户zhangsan的账户，使其无法登录，验证效果后解除锁定

1）锁定用户账号

使用passwd或usermod命令将用户zhangsan的账户锁定。

[root@proxy ~]# passwd -l zhangsan //锁定用户账号lock

锁定用户 zhangsan 的密码。

passwd: 操作成功

[root@proxy ~]# passwd -S zhangsan //查看状态status

zhangsan LK 2018-02-22 0 99999 7 -1 (密码已被锁定。)

2）验证用户zhangsan已无法登录，说明锁定生效

输入正确的用户名、密码，始终提示“Login incorrect”，无法登录。

3）解除对用户zhangsan的锁定

[root@proxy ~]# passwd -u zhangsan //解锁用户账号

解锁用户 zhangsan 的密码 。

passwd: 操作成功

[root@proxy ~]# passwd -S zhangsan //查看状态

zhangsan PS 2018-08-14 0 99999 7 -1 (密码已设置，使用 SHA512 加密。)

步骤三：修改tty登录的提示信息，隐藏系统版本

1）账户在登录Linux系统时，默认会显示登陆信息（包括操作系统内核信息）

/etc/issue这个配置文件里保存的就是这些登陆信息，修改该文件防止内核信息泄露。

[root@proxy ~]# cat /etc/issue //确认原始文件

Red Hat Enterprise Linux Server release 6.5 (Santiago)

Kernel \r on an \m

[root@proxy ~]# cp /etc/issue /etc/issue.origin //备份文件

[root@proxy ~]# vim /etc/issue //修改文件内容

Windows Server 2012 Enterprise R2

NT 6.2 Hybrid

2）测试版本伪装效果

退出已登录的tty终端，或者重启Linux系统，刷新后的终端提示信息会变成自定义的文本内容，如图-1所示。

图-1

附加：对于操作系统来说，文件系统也可以通过添加额外属性来提高性能与安全性。

[root@proxy ~]# cat /etc/fstab

/dev/vda1 /boot xfs defaults,noexec 0 0

/dev/vda3 /home xfs defaults,noatime 0 0

备注：

noexec属性可以让分区下的所有程序都不可执行，包括病毒与木马

noatime让分区下的所有文件都不再更新atime时间，atime时间为文件的访问时间

步骤四：锁定文件/etc/resolv.conf、/etc/hosts

1）语法格式：

# chattr +i 文件名 //锁定文件（无法修改、删除等）

# chattr -i 文件名 //解锁文件

# chattr +a 文件名 //锁定后文件仅可追加

# chattr -a 文件名 //解锁文件

# lsattr 文件名 //查看文件特殊属性

2) 使用+i锁定文件，使用lsattr查看属性

[root@proxy ~]# chattr +i /etc/resolv.conf

[root@proxy ~]# lsattr /etc/resolv.conf

----i---------- /etc/resolv.conf

3）使用+a锁定文件(仅可追加)，使用lsattr查看属性

[root@proxy ~]# chattr +a /etc/hosts

[root@proxy ~]# lsattr /etc/hosts

-----a---------- /etc/hosts

4）测试文件锁定效果

[root@proxy ~]# rm -rf /etc/resolv.conf

rm: 无法删除"/etc/resolv.conf": 不允许的操作

[root@proxy ~]# echo xyz > /etc/resolv.conf

-bash: resolv.conf: 权限不够

[root@proxy ~]# rm -rf /etc/hosts //失败

[root@proxy ~]# echo "192.168.4.1 xyz" > /etc/hosts //失败

[root@proxy ~]# echo "192.168.4.1 xyz" >> /etc/hosts //成功

5）恢复这两个文件原有的属性（避免对后续实验造成影响）

[root@proxy ~]# chattr -i /etc/resolv.conf

[root@proxy ~]# chattr -i /etc/hosts

[root@proxy ~]# lsattr /etc/resolv.conf /etc/hosts

--------------- /etc/resolv.conf

--------------- /etc/hosts

2 案例2：使用sudo分配管理权限

2.1 问题

本案例要求利用sudo机制分配管理操作权限，主要完成以下任务：

使用su命令临时切换账户身份，并执行命令

允许softadm管理系统服务的权限

允许用户useradm通过sudo方式添加/删除/修改除root以外的用户账号

允许wheel组成员以特权执行所有命令

为sudo机制启用日志记录，以便跟踪sudo执行操作

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用su命令临时切换账户身份，并以root执行命令

su(Substitute User)命令可以快速切换账户身份，普通用户切换账户身份时需要输入密码，root使用su命令切换任何身份都不需要密码，如法格式如下：

# su - [账户名称]

# su - [账户名称] -c '命令'

1)从普通用户切换为root账户身份(如果没有普通账户则需要先创建)

[zhangsan@proxy ~]# whoami

zhangsan

[zhangsan@proxy ~]# su - //切换账户，默认切换为root账户

密码: //输入root的密码

[root@proxy ~]# whoami //确认结果

root

2)以普通身份创建文件(如果没有普通账户则需要先创建)，以root身份重启服务

[root@proxy ~]# su - zhangsan -c "touch /tmp/test.txt" //管理员切换普通用户

[root@proxy ~]# ll /tmp/test.txt

[zhangsan@proxy ~]# su - -c "systemctl restart sshd" //以管理员重启服务

密码：

● sshd.service - OpenSSH server daemon

Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)

active: active (running) since 五 2018-01-19 08:59:40 CST; 1 months 4 days ago

步骤二：允许softadm管理系统服务的权限

1）修改/etc/sudoers配置

修改/etc/sudoers可以直接使用vim编辑该文件，或使用visudo命令修改该文件。

为softadm授予相关脚本的执行权限，允许通过systemctl工具来管理系统服务。

如果没有softadm账户可以先创建该账户。

[root@proxy ~]# useradd softadm

[root@proxy ~]# vim /etc/sudoers //修改文件后，需要使用wq强制保存

.. ..

softadm ALL=(ALL) /usr/bin/systemctl

//授权softadm以root身份执行systemctl命令（ALL包括root）

2）切换为softadm用户，并验证sudo执行权限

[root@proxy ~]# su - softadm

[softadm@proxy ~]$ sudo -l

… …

[sudo] password for softadm: //输入softadm的口令

.. ..

用户 softadm 可以在该主机上运行以下命令：

(ALL) /usr/bin/systemctl

[softadm@proxy ~]$ systemctl start httpd //不用sudo时启动服务失败

Authentication is required

.. ..

[softadm@proxy ~]$ sudo systemctl restart httpd //通过sudo启动服务成功

步骤三：允许用户useradm通过sudo方式添加/删除/修改除root以外的用户账号

1）修改/etc/sudoers配置

为useradm授予用户管理相关命令的执行权限，例外程序以!符号取反，放在后面。在执行相关程序时，可以利用通配符\*。

[root@proxy ~]# useradd useradm

[root@proxy ~]# vim /etc/sudoers

.. ..

useradm ALL=(ALL) /usr/bin/passwd,!/usr/bin/passwd root,/usr/sbin/user\*,

!/usr/sbin/user\* \* root

2）切换为useradm用户，验证sudo权限

可以通过sudo方式来添加/删除/修改普通用户：

[useradm@proxy ~]$ sudo -l

.. ..

用户useradm可以在该主机上运行以下命令：

(root) /usr/bin/passwd, !/usr/bin/passwd root, /usr/sbin/user\*,

!/usr/sbin/user\* \* root

[useradm@proxy ~]$ sudo useradd newuser01 //可以添加用户

[useradm@proxy ~]$ sudo passwd newuser01 //可以修改普通用户的口令

更改用户 newuser01 的密码 。

新的 密码：

重新输入新的 密码：

passwd： 所有的身份验证令牌已经成功更新。

但是不能修改root用户的密码：

[useradm@proxy ~]$ sudo passwd root

对不起，用户 useradm 无权以 root 的身份在 localhost 上

执行 /usr/bin/passwd root。

步骤四：允许wheel组成员以特权执行所有命令

此案例用来展示sudo的便利性及设置不当带来的危险性，生产环境下慎用。

实现时参考下列操作(如果没有普通用户则先创建该账户)：

[root@proxy ~]# vim /etc/sudoers

.. ..

%wheel ALL=(ALL) ALL

[root@proxy ~]# usermod -a -G wheel zengye

[zengye@proxy ~]$ sudo -l

.. ..

用户 zengye 可以在该主机上运行以下命令：

(root) /bin/\*

步骤五：为sudo机制启用日志记录，以便跟踪sudo执行操作

1）修改/etc/sudoers配置，添加日志设置

[root@proxy ~]# visudo

Defaults logfile="/var/log/sudo"

.. ..

2）以root（默认有所有权限）执行sudo操作

[root@proxy ~]# sudo -l //查看授权的sudo操作

[softadm@proxy ~]# sudo systemctl status httpd //查看授权的sudo操作

3）确认日志记录已生效

[root@proxy ~]# tail /var/log/sudo

.. ..

May 16 22:14:49 : root : TTY=pts/1 ; PWD=/root ; USER=root ; COMMAND=list

Feb 22 22:35:43 : softadm : TTY=pts/11 ; PWD=/home/softadm ; USER=root ;

COMMAND=/bin/systemctl status httpd

3 案例3：提高SSH服务安全

3.1 问题

本案例要求提高Linux主机上SSH服务端的安全性，完成以下任务：

配置基本安全策略（禁止root、禁止空口令）

针对SSH访问采用仅允许的策略，未明确列出的用户一概拒绝登录

实现密钥验证登录（私钥口令）、免密码登入

确认密钥验证使用正常后，禁用口令验证

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置基本安全策略

1）调整sshd服务配置，并重载服务

[root@proxy ~]# vim /etc/ssh/sshd\_config

.. ..

Protocol 2 //SSH协议

PermitRootLogin no //禁止root用户登录

PermitEmptyPasswords no //禁止密码为空的用户登录

UseDNS no //不解析客户机地址

LoginGraceTime 1m //登录限时

MaxAuthTries 3 //每连接最多认证次数

.. ..

[root@proxy ~]# systemctl restart sshd

2）测试基本安全策略

尝试以root用户SSH登录，失败：

[root@proxy ~]# ssh root@192.168.4.5

root@192.168.4.5's password:

Permission denied, please try again.

将服务器上用户kate(如无该账户则先创建)的密码设为空，尝试SSH登录，也会失败：

[root@proxy ~]# passwd -d kate //清空用户口令

清除用户的密码 kate。

passwd: 操作成功

[root@proxy ~]# ssh kate@192.168.4.5

kate@192.168.4.5's password:

Permission denied, please try again.

步骤二：针对SSH访问采用仅允许的策略，未明确列出的用户一概拒绝登录

1）调整sshd服务配置，添加AllowUsers策略，仅允许用户zhangsan、tom、useradm，其中useradm只能从网段192.168.4.0/24登录。

注意：如果没有这些用户，需要提前创建用户并设置密码。

[root@proxy ~]# vim /etc/ssh/sshd\_config

.. ..

AllowUsers zhangsan tom useradm@192.168.4.0/24 //定义账户白名单

##DenyUsers USER1 USER2 //定义账户黑名单

##DenyGroups GROUP1 GROUP2 //定义组黑名单

##AllowGroups GROUP1 GROUP2 //定义组白名单

[root@proxy ~]# systemctl restart sshd

2）验证SSH访问控制，未授权的用户将拒绝登录。

[root@proxy ~]# ssh useradm@192.168.4.5 //已授权的用户允许登录

useradm@192.168.4.5's password:

[useradm@proxy ~]$ exit

[root@proxy ~]# ssh root@192.168.4.5 //未授权的用户被拒绝登录

root@192.168.4.5's password:

Permission denied, please try again.

步骤三：实现密钥对验证登录（私钥口令）、免密码登入

1）准备客户机测试环境

为客户机的用户root建立SSH密钥对

使用ssh-keygen创建密钥对，将私钥口令设为空（直接回车）：

[root@client ~]$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase): //直接回车将口令设为空

Enter same passphrase again: //再次回车确认

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

63:6e:cf:45:f0:56:e2:89:6f:62:64:5a:5e:fd:68:d2

The key's randomart image is:

+--[ RSA 2048]----+

| |

| |

| . . . |

| = = |

| S = B . |

| o B = . o |

| + + = E .|

| . + + o |

| o |

+-----------------+

[root@client ~]$ ls -lh ~/.ssh/id\_rsa\* //确认密钥对文件

-rw-------. 1 root root 1.8K 8月 15 10:35 /root/.ssh/id\_rsa

-rw-r--r--. 1 root root 403 8月 15 10:35 /root/.ssh/id\_rsa.pub

2）将客户机上用户root的公钥部署到SSH服务器

以用户root登入客户机，使用ssh-copy-id命令将自己的公钥部署到服务器：

[root@client ~]$ ssh-copy-id root@192.168.4.5

root@192.168.4.5's password:

Now try logging into the machine, with "ssh 'root@192.168.4.5'", and check in:

.ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting.

3）在服务器上确认客户机用户root上传的公钥信息

默认部署位置为目标用户的家目录下 ~/.ssh/authorized\_keys文件：

[root@proxy ~]# tail -2 ~/.ssh/authorized\_keys

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAzz+5AiFMGQ7LfuiV7eBnOcmRO9JRTcqRoynGO2y5

RyFL+LxR1IpEbkNrUyIZDk5uaX1Y8rwsf+pa7UZ2NyqmUEvNSUo0hQyDGsU9SPyAdzRCCvDgwpOFhaHi/OFnT+zqjAqXH2M9fFYEVUU4PIVL8HT19zCQRVZ/q3acQA34UsQUR0PpLJAobsf1BLe2EDM8BsSHckDGsNoDT9vk+u3e83RaehBMuy1cVEN5sLAaIrIeyM8Q0WxQNlqknL908HRkTlTeKrRoHbMnOBFj8StwlnscKHlkrsKkhUf8A9WWz/vL4GDwGND5jdca3I2hdITAySjMdfL1HMHnMYOgMjPM0Q== root@192.168.4.100

4）在客户机上测试SSH密钥对验证

在客户机用户root的环境中，以远程用户root登入192.168.4.5主机时，无需验证口令即可登入（因为私钥口令为空）：

[root@client ~]$ ssh root@192.168.4.5 //免交互直接登入

Last login: Thu Aug 15 10:48:09 2013 from 192.168.4.100

步骤四：确认密钥验证使用正常后，禁用口令验证

1）调整sshd服务配置，将PasswordAuthentication设为no

[root@proxy ~]# vim /etc/ssh/sshd\_config

.. ..

PasswordAuthentication no //将此行yes改成no

[root@proxy ~]# systemctl restart sshd

4 案例4：SELinux安全防护

4.1 问题

本案例要求熟悉SELinux防护机制的开关及策略配置，完成以下任务：

将Linux服务器的SELinux设为enforcing强制模式

从/root目录下移动一个包文件到FTP下载目录，调整策略使其能够被下载

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：将Linux服务器的SELinux设为enforcing强制模式

1）固定配置：修改/etc/selinux/config文件

确认或修改SELINUX为enforcing模式：

[root@proxy ~]# vim /etc/selinux/config

SELINUX=enforcing //设置SELinux为强制模式

SELINUXTYPE=targeted //保护策略为保护主要的网络服务安全

2）临时配置：使用setenforce命令

查看当前SELinux状态，如果是disabled则需要根据第1）步的配置重启系统；如果是permissive则使用setenforce命令修改为enforcing即可：

[root@proxy ~]# getenforce //查看当前状态为警告模式

Permissive

[root@proxy ~]# setenforce 1 //设置SELinux为强制模式

[root@proxy ~]# getenforce //查看当前模式为强制模式

Enforcing

[root@proxy ~]# setenforce 0 //设置SELinux为强制模式

[root@proxy ~]# getenforce //查看当前模式为警告模式

Permissive

步骤二：在SELinux启用状态下，调整策略打开vsftpd服务的匿名上传访问

1）配置一个允许匿名上传的vsftpd服务作为测试环境

[root@proxy ~]# setenforce 1

[root@proxy ~]# yum -y install vsftpd

.. ..

[root@proxy ~]# vim /etc/vsftpd/vsftpd.conf

anonymous\_enable=YES //开启匿名访问

anon\_upload\_enable=YES //允许上传文件

anon\_mkdir\_write\_enable=YES //允许上传目录

[root@proxy ~]# systemctl start vsftpd //启动服务

//默认Vsftpd共享目录为/var/ftp/

步骤三：从/root目录下移动2个包文件到FTP下载目录，调整文件的安全上下文

1）建立两个FTP下载用的测试文件

由root用户创建两个测试压缩包，一个直接建立到/var/ftp/目录下，另一个先在/root/下建立，然后移动至/var/ftp/目录。

//测试文件1，直接在ftp目录下创建文件

[root@proxy ~]# tar -czf /var/ftp/log1.tar /var/log

[root@proxy ~]# ls -lh /var/ftp/

-rw-r--r--. 1 root root 8M 8月 16 10:16 log1.tar

[root@proxy ~]# ls -Z /var/ftp/

-rw-r--r--. root root unconfined\_u:object\_r:public\_content\_t:s0 log1.tar

//测试文件2，在/root下建立，然后移动至/var/ftp目录

[root@proxy ~]# tar -czf log2.tar /var/log

[root@proxy ~]# mv log2.tar /var/ftp/

[root@proxy ~]# ls -lh /var/ftp/

-rw-r--r--. 1 root root 8M 8月 16 10:16 log2.tar

[root@proxy ~]# ls -Z /var/ftp/

-rw-r--r--. 1 root root unconfined\_u:object\_r:admin\_home\_t:s0 log2.tar

3）通过FTP方式测试下载

使用wget命令分别下载这两个包文件，第二个包将会下载失败（看不到文件）。

[root@proxy ~]# wget ftp://192.168.4.5/log1.tar //下载第一个文件，成功

[root@proxy ~]# wget ftp://192.168.4.5/log2.tar //下载第二个文件，失败

4）检查该测试包的安全上下文，正确调整后再次下载第二个包成功。

文件已经存放到共享目录下，但客户端无法访问下载，是因为被SELinux拦截了！

[root@proxy ~]# ls -Z /var/ftp/

-rw-r--r--. root root unconfined\_u:object\_r:public\_content\_t:s0 log1.tar

-rw-r--r--. 1 root root unconfined\_u:object\_r:admin\_home\_t:s0 log2.tar

[root@proxy ~]# chcon -t public\_content\_t /var/ftp/d2.tar.gz

[root@proxy ~]# ls -Z /var/ftp/log2.tar

-rw-r--r--. root root unconfined\_u:object\_r:public\_content\_t:s0 log2.tar

[root@proxy ~]# wget ftp://192.168.4.5/log2.tar //再次下载，成功

注意：上例中的chcon操作可替换为（效果相同）：

# restorecon /var/ftp/log2.tar.gz

或者

# chcon --reference=/var/ftp/log1.tar.gz /var/ftp/log2.tar.gz

# NSD SECURITY DAY02

# 1加密与解密

加解密概述

发送方加密

接收方解密

**对称加密:**

采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密，这种加密方法称为对称加密，也称为单密钥加密。

常用的算法有：

DES、3DES、TDEA、Blowfish、RC2、RC4、RC5、IDEA、SKIPJACK等。

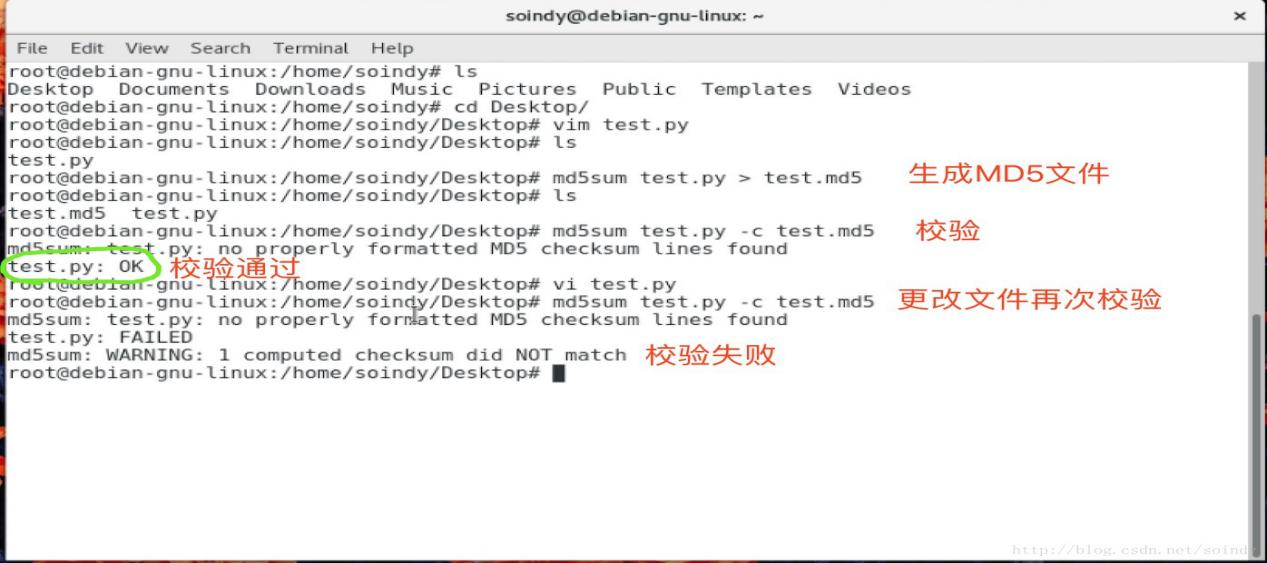
**非对称加密:**

非对称加密算法需要两个密钥：公开密钥（publickey）和私有密钥（privatekey）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。

常用算法:

RSA、Elgamal、背包算法、Rabin、D-H、ECC（椭圆曲线加密算法）。

# 2 MD5完整性检验



# 3 GPG加/解密工具

GnuPG 软件（简称 GPG），它是目前最流行、最好用的加密工具之一。

## GPG命令集:

]# gpg --version

]# gpg --help

**对称加密命令:**

]$ gpg -c a.txt

]$ gpg -d a.txt.gpg >a.txt

**非对称加密命令:**

]$ gpg --gen-key #生成秘钥对

]$ gpg -a --export >/tmp/userb.pub #导出公钥

]$ gpg --import /tmp/userb.pub #导入公钥

]$ gpg -e -r hksbdb test.txt #公钥加密 (hksbdb密码)

]$ gpg -d test.txt.gpg >b.txt #输入密码,进行解密

[root@room9pc01 ~]# **gpg --version**

gpg (GnuPG) 2.0.22

libgcrypt 1.5.3

Copyright (C) 2013 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

Home: ~/.gnupg

支持的算法：

公钥：RSA, ?, ?, ELG, DSA

对称加密：IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256,

TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256

散列：MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224

压缩：不压缩, ZIP, ZLIB, BZIP2

[root@room9pc01 ~]# gpg --help

## 对称加/解密方式：

1、实用gpg完成文件加密

使用方法：gpg -c file

在弹出页面输入两遍口令即可加密文件，加密后会生成原文件名加.gpg的加密文件，注意此时未加密文件仍然存在，可删除未加密原文件。

2、查出加密文件

使用方法：gpg -d 加密文件

在弹出如上图界面输入口令即可查看文件，

注：在加密本机或其他主机输入一次密码后将记录秘钥，解密不需要再次输入秘钥。加密后的文件为data文件类型，更改文件名后仍可解密，与后缀无关。

3、把加密的文件解密出来

使用方法：gpg -o newfilename -d file.gpg

例如:创建用户usera 密码:123456 用户userb 密码:123456

两个终端usera创建文件加密.userb解密查看文件

### A加密

[usera@room9pc01 ~]$ vim a.txt

[usera@room9pc01 ~]$ gpg -c a.txt

gpg: 已创建目录‘/home/usera/.gnupg’

gpg: 新的配置文件‘/home/usera/.gnupg/gpg.conf’已建立

gpg: 警告：在‘/home/usera/.gnupg/gpg.conf’里的选项于此次运行期间未被使用

gpg: 钥匙环‘/home/usera/.gnupg/pubring.gpg’已建立

gpg: 用户取消

gpg: 生成密码的时候发生错误：操作已取消

gpg: 对称加密‘a.txt’失败：操作已取消

[usera@room9pc01 ~]$ gpg -c a.txt

[usera@room9pc01 ~]$ cp a.txt /tmp/

### B解密

[userb@room9pc01 ~]$ ls /tmp/a.txt.gpg

/tmp/a.txt.gpg

[userb@room9pc01 ~]$ cp /tmp/a.txt.gpg ~/

[userb@room9pc01 ~]$ ls

a.txt.gpg

[userb@room9pc01 ~]$ gpg -d a.txt.gpg >a.txt

gpg: 已创建目录‘/home/userb/.gnupg’

gpg: 新的配置文件‘/home/userb/.gnupg/gpg.conf’已建立

gpg: 警告：在‘/home/userb/.gnupg/gpg.conf’里的选项于此次运行期间未被使用

gpg: 钥匙环‘/home/userb/.gnupg/secring.gpg’已建立

gpg: 钥匙环‘/home/userb/.gnupg/pubring.gpg’已建立

gpg: CAST5 加密过的数据

gpg: 以 1 个密码加密

gpg: 警告：报文未受到完整的保护

[userb@room9pc01 ~]$ cat a.txt #解密后即可看到内容

hk hk

## 非对称加/解密方式

B用户创建公钥与私钥,将公钥传送给A

A收到公钥,将数据用公钥加密,加密后的文件传送给B

B收到加密的文件,用私钥解开

### B秘钥对生成

[userb@room9pc01 ~]$ gpg --gen-key

gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

请选择您要使用的密钥种类：

(1) RSA and RSA (default)

(2) DSA and Elgamal

(3) DSA (仅用于签名)

(4) RSA (仅用于签名)

您的选择？ #直接回车默认第一种方式

RSA 密钥长度应在 1024 位与 4096 位之间。

您想要用多大的密钥尺寸？(2048)

您所要求的密钥尺寸是 2048 位

请设定这把密钥的有效期限。

0 = 密钥永不过期

<n> = 密钥在 n 天后过期

<n>w = 密钥在 n 周后过期

<n>m = 密钥在 n 月后过期

<n>y = 密钥在 n 年后过期

密钥的有效期限是？(0)

密钥永远不会过期

以上正确吗？(y/n)y

You need a user ID to identify your key; the software constructs the user ID

from the Real Name, Comment and Email Address in this form:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

真实姓名：hksb

姓名至少要有五个字符长

真实姓名：hksbdb #加密需要gpg -e -r hksbdb 文件名

电子邮件地址：hk@sb.cn

注释：userb-password #此处需要记住

您选定了这个用户标识：

“hksbdb (userb-password) <hk@sb.cn>”

更改姓名(N)、注释(C)、电子邮件地址(E)或确定(O)/退出(Q)？O

您需要一个密码来保护您的私钥。

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动

鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动

鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

gpg: /home/userb/.gnupg/trustdb.gpg：建立了信任度数据库

gpg: 密钥 B5417601 被标记为绝对信任

#如果一直卡在这是应为调用RANDOM生成随机数

公钥和私钥已经生成并经签名。

gpg: 正在检查信任度数据库

gpg: 需要 3 份勉强信任和 1 份完全信任，PGP 信任模型

gpg: 深度：0 有效性： 1 已签名： 0 信任度：0-，0q，0n，0m，0f，1u

pub 2048R/B5417601 2019-03-07

密钥指纹 = 8B47 2A93 2FF3 C2C9 97BD 812C EA54 445D B541 7601

uid hksbdb (userb-password) <hk@sb.cn>

sub 2048R/88ADEA9F 2019-03-07

[userb@room9pc01 ~]$ ls

a.txt

[userb@room9pc01 ~]$ ls .gnupg/ #如需重新创建,删除此隐藏文件

gpg.conf private-keys-v1.d pubring.gpg pubring.gpg~ random\_seed secring.gpg S.gpg-agent trustdb.gpg

pubring.gpg公钥文件

secring.gpg私钥文件

### B导出公钥

gpg -a --export 用户名(可省略) > 文件名.pub ## -a 为可读方式

[userb@room9pc01 ~]$ gpg -a --export >/tmp/userb.pub

[userb@room9pc01 ~]$ cat /tmp/userb.pub

### A导入公钥

gpg --import /文件名.pub

[usera@room9pc01 ~]$ gpg --import /tmp/userb.pub

gpg: 钥匙环‘/home/usera/.gnupg/secring.gpg’已建立

gpg: /home/usera/.gnupg/trustdb.gpg：建立了信任度数据库

gpg: 密钥 B5417601：公钥“hksbdb (userb-password) <hk@sb.cn>”已导入

gpg: 合计被处理的数量：1

gpg: 已导入：1 (RSA: 1)

[usera@room9pc01 ~]$ ls .gnupg/

gpg.conf private-keys-v1.d pubring.gpg pubring.gpg~ random\_seed secring.gpg S.gpg-agent trustdb.gpg

### A公钥加密文件

gpg -e -r hksbdb test.tx

[usera@room9pc01 ~]$ echo hksb >test.txt

[usera@room9pc01 ~]$ ls

a.txt b.txt b.txt.gpg test.txt

[usera@room9pc01 ~]$ gpg -e -r hksbdb test.txt

gpg: 88ADEA9F：没有证据表明这把密钥真的属于它所声称的持有者

pub 2048R/88ADEA9F 2019-03-07 hksbdb (userb-password) <hk@sb.cn>

主钥指纹： 8B47 2A93 2FF3 C2C9 97BD 812C EA54 445D B541 7601

子钥指纹： 20C5 2F32 1E22 35F1 830B 4B4E A70B E3BF 88AD EA9F

这把密钥并不一定属于用户标识声称的那个人。如果您真的知道自

己在做什么，您可以在下一个问题回答 yes。

无论如何还是使用这把密钥吗？(y/N)y

[usera@room9pc01 ~]$ ls

test.txt test.txt.gpg

[usera@room9pc01 ~]$ cp test.txt.gpg /tmp/

[usera@room9pc01 ~]$

### B用私钥解密

[userb@room9pc01 ~]$ cp /tmp/test.txt.gpg ~/

[userb@room9pc01 ~]$ ls

a.txt test.txt.gpg

[userb@room9pc01 ~]$ gpg -d test.txt.gpg >b.txt #输入密码

您需要输入密码，才能解开这个用户的私钥：“hksbdb (userb-password) <hk@sb.cn>”

2048 位的 RSA 密钥，钥匙号 88ADEA9F，建立于 2019-03-07 (主钥匙号 B5417601)

gpg: 由 2048 位的 RSA 密钥加密，钥匙号为 88ADEA9F、生成于 2019-03-07

“hksbdb (userb-password) <hk@sb.cn>”

[userb@room9pc01 ~]$ ls

a.txt b.txt test.txt.gpg

[userb@room9pc01 ~]$ cat b.txt

hksb

## GPG数字签名

软件签名与验证过程

软件官方以私钥对软件包执行数字签名

用户下载软件包,软件官方的公钥

以官方公钥验证软件包签名.确保数据来源正确

### B使用私钥做数字签名

[userb@room9pc01 tmp]$ echo FFFFFAAAADDDDD > a2.txt

[userb@room9pc01 tmp]$ gpg -b a2.txt

您需要输入密码，才能解开这个用户的私钥：“hkhkhk (hkhkhk-password) <hk@tedu.cn>”

2048 位的 RSA 密钥，钥匙号 EB3C37CA，建立于 2019-03-07

[userb@room9pc01 tmp]$ ls a2.\*

a2.txt a2.txt.sig #a2.txt.sig:签名文件

B共享签名文件和原文件给A

### A使用公钥验证签名

[usera@room9pc01 ~]$ gpg --verify a2.txt.sig

gpg: 于 2019年03月07日 星期四 14时26分13秒 CST 创建的签名，使用 RSA，钥匙号 EB3C37CA

gpg: 完好的签名，来自于“hkhkhk (hkhkhk-password) <hk@tedu.cn>”

gpg: 警告：这把密钥未经受信任的签名认证！

gpg: 没有证据表明这个签名属于它所声称的持有者。

主钥指纹： A01B D9A0 EC97 FE9E 0F2A D203 3778 6B0C EB3C 37CA

[usera@room9pc01 ~]$ echo CCC >> a2.txt

[usera@room9pc01 ~]$ gpg --verify a2.txt.sig

gpg: 于 2019年03月07日 星期四 14时26分13秒 CST 创建的签名，使用 RSA，钥匙号 EB3C37CA

gpg: 已损坏的签名，来自于“hkhkhk (hkhkhk-password) [<hk@tedu.cn>”](mailto:<hk@tedu.cn>\”)

# 4 AIDE入侵检测系统

AIDE(Advanced Intrusion Detection Environment,高级入侵检测环境)是个入侵检测工具，主要用途是检查文档的完整性。

## 安装

[root@client-50 ~]# yum -y install aide

## 配置文件解释:

[root@client-50 ~]# vim /etc/aide.conf

3 @@define DBDIR /var/lib/aide #aide数据库目录,存放检测信息

4 @@define LOGDIR /var/log/aide #日志

7 database=file:@@{DBDIR}/aide.db.gz

#检测时对比文件

12 database\_out=file:@@{DBDIR}/aide.db.new.gz

#当前检测时生成的检测信息存放文件

15 gzip\_dbout=yes #是否做压缩

54 FIPSR = p+i+n+u+g+s+m+c+acl+selinux+xattrs+sha256 #调用的检测集

89 CONTENT\_EX = sha256+ftype+p+u+g+n+acl+selinux+xattrs

99 /boot/ CONTENT\_EX

100 /bin/ CONTENT\_EX

101 /sbin/ CONTENT\_EX

112 !/usr/src/ #取反不检查/usr/src/

113 !/usr/tmp/

114 # Otherwise get all of /usr.

115 /usr/ CONTENT\_EX #检查这个目录里面的所有

## 配置入侵检测目录调用检测规则

[root@client-50 ~]# sed -i '99,312s/^/#/' /etc/aide.conf #注释掉默认检测项

[root@client-50 ~]# vim /etc/aide.conf

/root/ FIPSR #检测/root/目录,,调用FIPSR规则

[root@client-50 ~]# aide --help #查看命令帮助

## 生成入侵数据库文件

[root@client-50 ~]# aide --init

AIDE, version 0.15.1

### AIDE database at /var/lib/aide/aide.db.new.gz initialized.

#检测完的状态存放在aide.db.new.gz这个文件内

[root@client-50 aide]# cp aide.db.new.gz /tmp/ #备份检测信息

[root@client-50 aide]# mv aide.db.new.gz aide.db.gz

#将aide.db.new.gz文件修改成aide.db.gz ,用做下次检测对比

#下次检测时重新生成aide.db.new.gz文件,并与之前改名而来的aide.db.gz对比

[root@client-50 aide]# ls

aide.db.gz

[root@client-50 aide]# cd /root

[root@client-50 ~]# rm -rf anaconda-ks.cfg #删除一个文件

[root@client-50 ~]# vim test.txt #新建一个文件

## 开始入侵检测

[root@client-50 ~]# aide --check

AIDE 0.15.1 found differences between database and filesystem!!

Start timestamp: 2019-03-07 15:30:17

Summary:

Total number of files: 2101

Added files: 1 #新建一个文件

Removed files: 1 #删除一个文件

Changed files: 1 #改变一个文件

---------------------------------------------------

Added files:

---------------------------------------------------

added: /root/test.txt #新建的文件

---------------------------------------------------

Removed files:

---------------------------------------------------

removed: /root/anaconda-ks.cfg #删除的文件

---------------------------------------------------

Changed files:

---------------------------------------------------

changed: /root/.viminfo

--------------------------------------------------

Detailed information about changes:

---------------------------------------------------

File: /root/.viminfo #修改的文件,vim的操作记录,创建文件是vim创建的

Size : 6255 , 6240

Mtime : 2019-03-07 15:00:44 , 2019-03-07 15:29:42

Ctime : 2019-03-07 15:00:44 , 2019-03-07 15:29:42

Inode : 33574986 , 34198411

SHA256 : i/ZhxCSZY82D1QNrzZyJ4fD7RuCyCc61 , XztXHvArlOlJrhjQYU9saLyU28Wu5Az1

[root@client-50 ~]# ls /var/lib/aide/

aide.db.gz

# 5扫描nmap

**为扫描需要扫描:**

* 获取一些公开与非公开信息为目的
* 检测潜在的风险
* 查找可攻击目标
* 收集设备/主机/系统/软件信息
* 发现可利用的安全漏洞

**扫描方式**

* Scan 主动探测
* Sniff 被动监听/嗅探
* Capture 数据包捕获(抓包)

## nmap介绍

nmap，也就是Network Mapper，是Linux下的网络扫描和嗅探工具包。

nmap是一个网络连接端扫描软件，用来扫描网上电脑开放的网络连接端。确定哪些服务运行在哪些连接端，并且推断计算机运行哪个操作系统（这是亦称 fingerprinting）。它是网络管理员必用的软件之一，以及用以评估网络系统安全。

正如大多数被用于网络安全的工具，nmap 也是不少黑客及骇客（又称脚本小子）爱用的工具 。系统管理员可以利用nmap来探测工作环境中未经批准使用的服务器，但是黑客会利用nmap来搜集目标电脑的网络设定，从而计划攻击的方法。

Nmap 常被跟评估系统漏洞软件Nessus 混为一谈。Nmap 以隐秘的手法，避开闯入检测系统的监视，并尽可能不影响目标系统的日常操作。

**基本用法**

nmap [扫描类型] [选项] <扫描目标 ...>

**常用扫描类型**

-sS,TCP SYN扫描(半开) A三次握手B,A-B B-A A不在回应B

-sT,TCP 连接扫描(全开) A三次握手B,A-B B-A A-B

-sU,UDP扫描

-sP,ICMP扫描

-A,目标系统全面分析

**常用选项**

**-p 指定端口**

**-n 不做域名解析**

## 安装软件nmap

[root@client-50 ~]# which nmap

/usr/bin/nmap

[root@client-50 ~]# yum -y install nmap

## 命令用法:

nmap [扫描类型] [选项] <扫描目标 ...>

[root@client-50 ~]# nmap 192.168.4.51 #可不填写扫描类型与选项

Starting Nmap 6.40 ( http://nmap.org ) at 2019-03-07 16:08 CST

mass\_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers

Nmap scan report for 192.168.4.51

Host is up (0.00032s latency).

Not shown: 997 closed ports

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

111/tcp open rpcbind

MAC Address: 52:54:00:57:34:54 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds

**-sP,扫描类型(**ICMP扫描)

扫描哪些机器在线,

可扫单个主机,主机范围(51-55),网络段(192.168.4.0)

[root@client-50 ~]# nmap -sP 192.168.4.51-55

Starting Nmap 6.40 ( http://nmap.org ) at 2019-03-07 16:17 CST

mass\_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers

Nmap scan report for 192.168.4.51

Host is up (0.00036s latency).

MAC Address: 52:54:00:57:34:54 (QEMU Virtual NIC)

Nmap scan report for 192.168.4.52

Host is up (0.00044s latency).

MAC Address: 52:54:00:25:78:11 (QEMU Virtual NIC)

Nmap done: 5 IP addresses (2 hosts up) scanned in 0.23 seconds

**-P 指定扫描端口(只识别默认的软件服务端口)**

# 6 tcpdump命令抓包

TcpDump可以将网络中传送的数据包完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供and、or、not等逻辑语句来帮助你去掉无用的信息。

包必须经过主机或者到达这个主机,这个主机才能抓包

## 基本用法

tcpdump [选项] [过滤条件]

常见监控选项

-i 指定监控的网络接口(网卡) 默认是br0

-A 转换为ACSII码,以方便阅读

-w 将数据包信息保存到指定文件

-r 从指定文件读取数据包信息

-c 指定抓包数量

安装

[root@client-50 ~]# yum -y install tcpdump

例子

[root@client-50 ~]# tcpdump #抓取所有到达本机的包

[root@client-50 ~]# tcpdump -i eth0

[root@client-50 ~]# tcpdump -i eth0 -c 2

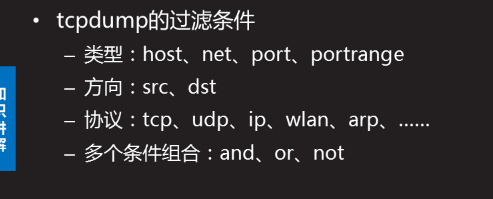
[root@client-50 ~]# tcpdump -i eth0 -c 2 -A

[root@client-50 ~]# tcpdump -i eth0 -c 2 -A -w plj.cap

[root@client-50 ~]# cat plj.cap

[root@client-50 ~]# tcpdump -A -r plj.cap

## 过滤条件



host 主机 net 网段 port 端口 portrange

]# tcpdump -i eth0 icmp #抓ping包

611 tcpdump -i eth0 icmp and host 176.52.8.208

612 tcpdump -i eth0 tcp port 80 and host 176.52.8.208

613 tcpdump -i eth0 tcp port 22

# 7WireShark图形抓包软件

安装

618 rpm -qa |grep -i wireshark

619 yum -y install wireshark wireshark-gnome

wireshark 主软件

wireshark-gnome 图形工具

打开转包软件

应用程序--互连网---软件名

WireShark可以把tcpdump命令的抓包文件打开查看抓包内容

1 使用tcpdump命令 抓包并保存到文件里

]# tcpdump -i br1 -A -c 2 -w /tmp/ssh.cap tcp port 22

2 使用WireShark打开抓包文件

应用程序--互连网---软件名--> file ---> open --/tmp/ssh.cap

# 案例

案例1：加密与解密应用

案例2：使用AIDE做入侵检测

案例3：扫描与抓包分析

1 案例1：加密与解密应用

1.1 问题

本案例要求采用gpg工具实现加/解密及软件签名等功能，分别完成以下任务：

检查文件的MD5校验和

使用GPG实现文件机密性保护，加密和解密操作

使用GPG的签名机制，验证数据的来源正确性

1.2 方案

加密算法主要有以下几种分类：

1.为确保数据机密性算法：

a) 对称加密算法(AES,DES)

b) 非对称加密算法（RSA，DSA）

2.为确保数据完整性算法：

a) 信息摘要（MD5，SHA256，SHA512）

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：检查文件的MD5校验和

1） 查看文件改动前的校验和，复制为新文件其校验和不变

[root@proxy ~]# vim file1.txt

abcdef

123456779

[root@proxy ~]# cp file1.txt file2.txt

[root@proxy ~]# cat file1.txt > file3.txt

[root@proxy ~]# md5sum file?.txt //文件内容一致，则校验和也不变

b92aa0f8aa5d5af5a47c6896283f3536 file1.txt

b92aa0f8aa5d5af5a47c6896283f3536 file2.txt

b92aa0f8aa5d5af5a47c6896283f3536 file3.txt

2） 对文件内容稍作改动，再次检查校验和，会发现校验和已大不相同

[root@proxy ~]# echo "x" >> file1.txt

[root@proxy ~]# md5sum file?.txt

6be3efe71d8b4b1ed34ac45f4edd2ba7 file1.txt

b92aa0f8aa5d5af5a47c6896283f3536 file2.txt

b92aa0f8aa5d5af5a47c6896283f3536 file3.txt

步骤二：使用GPG对称加密方式保护文件

GnuPG是非常流行的加密软件，支持所有常见加密算法，并且开源免费使用。

1）确保已经安装了相关软件（默认已经安装好了）

[root@proxy ~]# yum -y install gnupg2 //安装软件

[root@proxy ~]# gpg --version //查看版本

gpg (GnuPG) 2.0.22

2） gpg使用对称加密算法加密数据的操作

执行下列操作：

[root@proxy ~]# gpg -c file2.txt

.. ..

根据提示依次输入两次密码即可。如果是在GNOME桌面环境，设置密码的交互界面会是弹出的窗口程序，如图-1所示：

图－1

如果是在tty终端执行的上述加密操作，则提示界面也是文本方式的，如图-2所示。

图-2

根据提示输入两次口令，加密后的文件（自动添加后缀 .gpg）就生成了，传递过程中只要发送加密的文件（比如 file2.txt.gpg）就可以了。

[root@proxy ~]# cat file2.txt.gpg //查看加密数据为乱码

3）使用gpg对加密文件进行解密操作

收到加密的文件后，必须进行解密才能查看其内容。

[root@proxy ~]# gpg -d file2.txt.gpg > file2.txt //解密后保存

gpg: 3DES 加密过的数据

.. .. //根据提示输入正确密码

[root@proxy ~]# cat file2.txt //查看解密后的文件

abcdef

123456779

步骤三：使用GPG非对称加密方式保护文件

非对称加密/解密文件时，UserA（192.168.4.100）生成私钥与公钥，并把公钥发送给UserB（192.168.4.5），UserB使用公钥加密数据，并把加密后的数据传给UserA，UserA最后使用自己的私钥解密数据。

实现过程如下所述。

1）接收方UserA创建自己的公钥、私钥对(在192.168.4.100操作)

[root@client ~]# gpg --gen-key //创建密钥对

… …

请选择您要使用的密钥种类：

(1) RSA and RSA (default) //默认算法为RSA

(2) DSA and Elgamal

(3) DSA (仅用于签名)

(4) RSA (仅用于签名)

您的选择？ //直接回车默认(1)

RSA 密钥长度应在 1024 位与 4096 位之间。

您想要用多大的密钥尺寸？(2048) //接受默认2048位

您所要求的密钥尺寸是 2048 位

请设定这把密钥的有效期限。

0 = 密钥永不过期

<n> = 密钥在 n 天后过期

<n>w = 密钥在 n 周后过期

<n>m = 密钥在 n 月后过期

<n>y = 密钥在 n 年后过期

密钥的有效期限是？(0) //接受默认永不过期

密钥永远不会过期

以上正确吗？(y/n)y //输入y确认

You need a user ID to identify your key; the software constructs the user ID

from the Real Name, Comment and Email Address in this form:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

真实姓名：UserA

电子邮件地址：UserA@tarena.com

注释：UserA

您选定了这个用户标识：

“UserA (UserA) <UserA@tarena.com>”

更改姓名(N)、注释(C)、电子邮件地址(E)或确定(O)/退出(Q)？O //输入大写O确认

您需要一个密码来保护您的私钥。

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动

鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

gpg: 正在检查信任度数据库

gpg: 需要 3 份勉强信任和 1 份完全信任，PGP 信任模型

gpg: 深度：0 有效性： 1 已签名： 0 信任度：0-，0q，0n，0m，0f，1u

pub 2048R/421C9354 2017-08-16

密钥指纹 = 8A27 6FB5 1315 CEF8 D8A0 A65B F0C9 7DA6 421C 9354

uid UserA (UserA) <UserA@tarena.com>

sub 2048R/9FA3AD25 2017-08-16

注意：生产密钥后当前终端可能会变的无法使用，执行reset命令即可，或者关闭后再开一个终端。

2）UserA导出自己的公钥文件(在192.168.4.100操作)

用户的公钥、私钥信息分别保存在pubring.gpg和secring.gpg文件内：

[root@client ~]# gpg --list-keys //查看公钥环

/root/.gnupg/pubring.gpg

------------------------------

pub 2048R/421C9354 2017-08-16

uid UserA (User A) <UserA@tarena.com>

sub 2048R/9FA3AD25 2017-08-16

使用gpg命令结合--export选项将其中的公钥文本导出：

[root@client ~]# gpg -a --export UserA > UserA.pub

//--export的作用是导出密钥，-a的作用是导出的密钥存储为ASCII格式

[root@client ~]# scp UserA.pub 192.168.4.5:/tmp/

//将密钥传给Proxy

3）UserB导入接收的公钥信息（在192.168.4.5操作）

使用gpg命令结合--import选项导入发送方的公钥信息，以便在加密文件时指定对应的公钥。

[root@proxy ~]# gpg --import /tmp/UserA.pub

gpg: 密钥 421C9354：公钥“UserA (UserA) <UserA@tarena.com>”已导入

gpg: 合计被处理的数量：1

gpg: 已导入：1 (RSA: 1)

4) UserB使用公钥加密数据，并把加密后的数据传给UserA（在192.168.4.5操作）

[root@proxy ~]# echo "I love you ." > love.txt

[root@proxy ~]# gpg -e -r UserA love.txt

无论如何还是使用这把密钥吗？(y/N)y //确认使用此密钥加密文件

//-e选项是使用密钥加密数据

//-r选项后面跟的是密钥，说明使用哪个密钥对文件加密

[root@proxy ~]# scp love.txt.gpg 192.168.4.100:/root //加密的数据传给UserA

4）UserA以自己的私钥解密文件（在192.168.4.100操作）

[root@client ~]# gpg -d love.txt.gpg > love.txt

您需要输入密码，才能解开这个用户的私钥：“UserA (UserA) <UserA@tarena.com>”

2048 位的 RSA 密钥，钥匙号 9FA3AD25，建立于 2017-08-16 (主钥匙号 421C9354)

//验证私钥口令

gpg: 由 2048 位的 RSA 密钥加密，钥匙号为 9FA3AD25、生成于 2017-08-16

“UserA (UserA) <UserA@tarena.com>”

[root@client ~]# cat love.txt //获得解密后的文件内容

I love you.

步骤四：使用GPG的签名机制，检查数据来源的正确性

使用私钥签名的文件，是可以使用对应的公钥验证签名的，只要验证成功，则说明这个文件一定是出自对应的私钥签名，除非私钥被盗，否则一定能证明这个文件来自于某个人！

1）在client(192.168.4.100)上，UserA为软件包创建分离式签名

将软件包、签名文件、公钥文件一起发布给其他用户下载。

[root@client ~]# tar zcf log.tar /var/log //建立测试软件包

[root@client ~]# gpg -b log.tar //创建分离式数字签名

[root@client ~]# ls -lh log.tar\*

-rw-rw-r--. 1 root root 170 8月 17 21:18 log.tar

-rw-rw-r--. 1 root root 287 8月 17 21:22 log.tar.sig

[root@client ~]# scp log.tar\* 192.168.4.5:/root //将签名文件与签名传给UserB

2）在192.168.4.5上验证签名

[root@proxy ~]# gpg --verify log.tar.sig log.tar

gpg:于2028年06月07日 星期六 23时23分23秒 CST 创建的签名，使用 RSA，钥匙号 421C9354

gpg: 完好的签名，来自于“UserA (UserA) <UserA@tarena.com>”

.. ..

2 案例2：使用AIDE做入侵检测

2.1 问题

本案例要求熟悉Linux主机环境下的常用安全工具，完成以下任务操作：

安装aide软件

执行初始化校验操作，生成校验数据库文件

备份数据库文件到安全的地方

使用数据库执行入侵检测操作

2.2 方案

Aide通过检查数据文件的权限、时间、大小、哈希值等，校验数据的完整性。

使用Aide需要在数据没有被破坏前，对数据完成初始化校验，生成校验数据库文件，在被攻击后，可以使用数据库文件，快速定位被人篡改的文件。

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：部署AIDE入侵检测系统

1）安装软件包

[root@proxy ~]# yum -y install aide

2) 修改配置文件

确定对哪些数据进行校验，如何校验数据

[root@proxy ~]# vim /etc/aide.conf

@@define DBDIR /var/lib/aide //数据库目录

@@define LOGDIR /var/log/aide //日志目录

database\_out=file:@@{DBDIR}/aide.db.new.gz //数据库文件名

//一下内容为可以检查的项目（权限，用户，组，大小，哈希值等）

#p: permissions

#i: inode:

#n: number of links

#u: user

#g: group

#s: size

#md5: md5 checksum

#sha1: sha1 checksum

#sha256: sha256 checksum

DATAONLY = p+n+u+g+s+acl+selinux+xattrs+sha256

//以下内容设置需要对哪些数据进行入侵校验检查

//注意：为了校验的效率，这里将所有默认的校验目录与文件都注释

//仅保留/root目录，其他目录都注释掉

/root DATAONLY

#/boot NORMAL //对哪些目录进行什么校验

#/bin NORMAL

#/sbin NORMAL

#/lib NORMAL

#/lib64 NORMAL

#/opt NORMAL

#/usr NORMAL

#!/usr/src //使用[!]，设置不校验的目录

#!/usr/tmp

步骤二：初始化数据库，入侵后检测

1）入侵前对数据进行校验，生成初始化数据库

[root@proxy ~]# aide --init

AIDE, version 0.15.1

AIDE database at /var/lib/aide/aide.db.new.gz initialized.

//生成校验数据库，数据保存在/var/lib/aide/aide.db.new.gz

2）备份数据库，将数据库文件拷贝到U盘（非必须的操作）

[root@proxy ~]# cp /var/lib/aide/aide.db.new.gz /media/

3）入侵后检测

[root@proxy ~]# cd /var/lib/aide/

[root@proxy ~]# mv aide.db.new.gz aide.db.gz

[root@proxy ~]# aide --check //检查哪些数据发生了变化

3 案例3：扫描与抓包分析

3.1 问题

本案例要求熟悉Linux主机环境下的常用安全工具，完成以下任务操作：

使用NMAP扫描来获取指定主机/网段的相关信息

使用tcpdump分析FTP访问中的明文交换信息

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用NMAP扫描来获取指定主机/网段的相关信息

1）安装软件

[root@proxy ~]# yum -y install nmap

//基本用法：

# nmap [扫描类型] [选项] <扫描目标 ...>

//常用的扫描类型

// -sS，TCP SYN扫描（半开）

// -sT，TCP 连接扫描（全开）

// -sU，UDP扫描

// -sP，ICMP扫描

// -A，目标系统全面分析

2）检查192.168.4.100主机是否可以ping通

[root@proxy ~]# nmap -sP 192.168.4.100

Starting Nmap 6.40 ( http://nmap.org ) at 2018-06-06 21:59 CST

mass\_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers

Nmap scan report for host3 (192.168.4.100)

Host is up (0.00036s latency).

MAC Address: 52:54:00:71:07:76 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds

使用-n选项可以不执行DNS解析

[root@proxy ~]# nmap -n -sP 192.168.4.100

Starting Nmap 6.40 ( http://nmap.org ) at 2018-06-06 22:00 CST

Nmap scan report for 192.168.4.100

Host is up (0.00046s latency).

MAC Address: 52:54:00:71:07:76 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds

3）检查192.168.4.0/24网段内哪些主机可以ping通

[root@proxy ~]# nmap -n -sP 192.168.4.0/24

Starting Nmap 5.51 ( http://nmap.org ) at 2017-05-17 18:01 CST

Nmap scan report for 192.168.4.1

Host is up.

Nmap scan report for 192.168.4.7

Host is up.

Nmap scan report for 192.168.4.120

Host is up (0.00027s latency).

MAC Address: 00:0C:29:74:BE:21 (VMware)

Nmap scan report for 192.168.4.110

Host is up (0.00016s latency).

MAC Address: 00:50:56:C0:00:01 (VMware)

Nmap scan report for 192.168.4.120

Host is up (0.00046s latency).

MAC Address: 00:0C:29:DB:84:46 (VMware)

Nmap done: 256 IP addresses (5 hosts up) scanned in 3.57 seconds

4）检查目标主机所开启的TCP服务

[root@proxy ~]# nmap -sT 192.168.4.100

Starting Nmap 5.51 ( http://nmap.org ) at 2018-05-17 17:55 CST

Nmap scan report for 192.168.4.100

Host is up (0.00028s latency).

Not shown: 990 closed ports

PORT STATE SERVICE

21/tcp open ftp

22/tcp open ssh

25/tcp open smtp

80/tcp open http

110/tcp open pop3

111/tcp open rpcbind

143/tcp open imap

443/tcp open https

993/tcp open imaps

995/tcp open pop3s

MAC Address: 00:0C:29:74:BE:21 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.31 seconds

5）检查192.168.4.0/24网段内哪些主机开启了FTP、SSH服务

[root@proxy ~]# nmap -p 21-22 192.168.4.0/24

Starting Nmap 5.51 ( http://nmap.org ) at 2017-05-17 18:00 CST

Nmap scan report for 192.168.4.1

Host is up (0.000025s latency).

PORT STATE SERVICE

21/tcp open ftp

22/tcp open ssh

Nmap scan report for 192.168.4.7

Host is up.

PORT STATE SERVICE

21/tcp filtered ftp

22/tcp filtered ssh

Nmap scan report for 192.168.4.120

Host is up (0.00052s latency).

PORT STATE SERVICE

21/tcp open ftp

22/tcp open ssh

MAC Address: 00:0C:29:74:BE:21 (VMware)

Nmap scan report for pc110.tarena.com (192.168.4.110)

Host is up (0.00038s latency).

PORT STATE SERVICE

21/tcp closed ftp

22/tcp closed ssh

MAC Address: 00:50:56:C0:00:01 (VMware)

Nmap scan report for 192.168.4.120

Host is up (0.00051s latency).

PORT STATE SERVICE

21/tcp closed ftp

22/tcp closed ssh

MAC Address: 00:0C:29:DB:84:46 (VMware)

Nmap done: 256 IP addresses (5 hosts up) scanned in 4.88 seconds

6）检查目标主机所开启的UDP服务

[root@proxy ~]# nmap -sU 192.168.4.100 //指定-sU扫描UDP

53/udp open domain

111/udp open rpcbind

7）全面分析目标主机192.168.4.100和192.168.4.5的操作系统信息

[root@proxy ~]# nmap -A 192.168.4.100,5

Starting Nmap 5.51 ( http://nmap.org ) at 2017-05-17 18:03 CST

Nmap scan report for 192.168.4.100 //主机mail的扫描报告

Host is up (0.0016s latency).

Not shown: 990 closed ports

PORT STATE SERVICE VERSION

21/tcp open ftp vsftpd 2.2.2

| ftp-anon: Anonymous FTP login allowed (FTP code 230)

| -rw-r--r-- 1 0 0 1719 Aug 17 13:33 UserB.pub

| -rw-r--r-- 1 0 0 122 Aug 13 05:27 dl.txt

| drwxr-xr-x 2 14 0 4096 Aug 13 09:07 pub

| -rw-rw-r-- 1 505 505 170 Aug 17 13:18 tools-1.2.3.tar.gz

|\_-rw-rw-r-- 1 505 505 287 Aug 17 13:22 tools-1.2.3.tar.gz.sig

22/tcp open ssh OpenSSH 5.3 (protocol 2.0)

| ssh-hostkey: 1024 86:be:d6:89:c1:2d:d9:1f:57:2f:66:d1:af:a8:d3:c6 (DSA)

|\_2048 16:0a:15:01:fa:bb:91:1d:cc:ab:68:17:58:f9:49:4f (RSA)

25/tcp open smtp Postfix smtpd

80/tcp open http Apache httpd 2.2.15 ((Red Hat))

|\_http-methods: No Allow or Public header in OPTIONS response (status code 302)

| http-title: 302 Found

|\_Did not follow redirect to https://192.168.4.100//

110/tcp open pop3 Dovecot pop3d

|\_pop3-capabilities: USER CAPA UIDL TOP OK(K) RESP-CODES PIPELINING STLS SASL(PLAIN)

111/tcp open rpcbind

MAC Address: 00:0C:29:74:BE:21 (VMware)

No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).

TCP/IP fingerprint:

OS:SCAN(V=5.51%D=8/19%OT=21%CT=1%CU=34804%PV=Y%DS=1%DC=D%G=Y%M=000C29%TM=52

OS:11ED90%P=x86\_64-redhat-linux-gnu)SEQ(SP=106%GCD=1%ISR=10B%TI=Z%CI=Z%II=I

OS:%TS=A)OPS(O1=M5B4ST11NW6%O2=M5B4ST11NW6%O3=M5B4NNT11NW6%O4=M5B4ST11NW6%O

OS:5=M5B4ST11NW6%O6=M5B4ST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6

OS:=3890)ECN(R=Y%DF=Y%T=40%W=3908%O=M5B4NNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O

OS:%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=

OS:0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%

OS:S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(

OS:R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=

OS:N%T=40%CD=S)

Network Distance: 1 hop

Service Info: Host: mail.tarena.com; OS: Unix

TRACEROUTE

HOP RTT ADDRESS

1 1.55 ms 192.168.4.100

步骤二：使用tcpdump分析FTP访问中的明文交换信息

1）准备Vsftpd服务器（192.168.4.5操作）

[root@proxy ~]# yum -y install vsftpd

[root@proxy ~]# systemctl restart vsftpd

2）启用tcpdump命令行抓包

执行tcpdump命令行，添加适当的过滤条件，只抓取访问主机192.168.4.5的21端口的数据通信 ，并转换为ASCII码格式的易读文本。

这里假设，192.168.4.5主机有vsftpd服务，如果没有需要提前安装并启动服务！！！

[root@proxy ~]# tcpdump -A host 192.168.4.5 and tcp port 21

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes

.. .. //进入等待捕获数据包的状态

//监控选项如下：

// -i，指定监控的网络接口（默认监听第一个网卡）

// -A，转换为 ACSII 码，以方便阅读

// -w，将数据包信息保存到指定文件

// -r，从指定文件读取数据包信息

//tcpdump的过滤条件：

// 类型：host、net、port、portrange

// 方向：src、dst

// 协议：tcp、udp、ip、wlan、arp、……

// 多个条件组合：and、or、not

3）执行FTP访问，并观察tcpdump抓包结果

从192.168.4.100访问主机192.168.4.5的vsftpd服务。

[root@client ~]# yum -y install ftp

[root@client ~]# ftp 192.168.4.5

Connected to 192.168.4.200 (192.168.4.200).

220 (vsFTPd 3.0.2)

Name (192.168.4.200:root): tom //输入用户名

331 Please specify the password.

Password: //输入密码

530 Login incorrect.

Login failed.

ftp>quit //退出

观察抓包的结果（回到porxy主机观察tcpdump抓包的结果）：

[root@proxy ~]#

... …

18:47:27.960530 IP 192.168.4.100.novation > 192.168.4.5.ftp: Flags [P.], seq 1:14, ack 21, win 65515, length 13

E..5..@.@......x...d.\*..G.\c.1BvP.......USER tom

18:47:29.657364 IP 192.168.4.100.novation > 192.168.4.5.ftp: Flags [P.], seq 14:27, ack 55, win 65481, length 13

E..5..@.@......x...d.\*..G.\p.1B.P.......PASS 123

4)再次使用tcpdump抓包，使用-w选项可以将抓取的数据包另存为文件，方便后期慢慢分析。

[root@proxy ~]# tcpdump -A -w ftp.cap \

> host 192.168.4.5 and tcp port 21 //抓包并保存

tcpdump命令的-r选项，可以去读之前抓取的历史数据文件

[root@proxy ~]# tcpdump -A -r ftp.cap | egrep '(USER|PASS)' //分析数据包

.. ..

E..(..@.@.. ...x...d.\*..G.\c.1BbP.............

18:47:25.967592 IP 192.168.4.5.ftp > 192.168.4.100.novation: Flags [P.], seq 1:21, ack 1, win 229, length 20

E..<FJ@.@.jE...d...x...\*.1BbG.\cP...V...220 (vsFTPd 2.2.2)

… …

18:47:27.960530 IP 192.168.4.100.novation > 192.168.4.5.ftp: Flags [P.], seq 1:14, ack 21, win 65515, length 13

E..5..@.@......x...d.\*..G.\c.1BvP.......USER mickey

… …

18:47:27.960783 IP 192.168.4.5.ftp > 192.168.4.100.novation: Flags [P.], seq 21:55, ack 14, win 229, length 34

E..JFL@.@.j5...d...x...\*.1BvG.\pP...i~..331 Please specify the password.

… …

18:47:29.657364 IP 192.168.4.5.ftp > 192.168.4.100.novation: Flags [P.], seq 14:27, ack 55, win 65481, length 13

E..5..@.@......x...d.\*..G.\p.1B.P.......PASS pwd123

… …

18:47:29.702671 IP 192.168.4.100.novation > 192.168.4.5.ftp: Flags [P.], seq 55:78, ack 27, win 229, length 23

E..?FN@.@.j>...d...x...\*.1B.G.\}P.......230 Login successful.

步骤三：扩展知识，使用tcpdump分析Nginx的明文账户认证信息信息

1）在proxy主机(192.168.4.5)准备一台需要用户认证的Nginx服务器

[root@proxy ~]# cd /usr/local/nginx/conf/

[root@proxy ~]# cp nginx.conf.default nginx.conf //还原配置文件

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

server {

listen 80;

server\_name localhost;

auth\_basic "xx";

auth\_basic\_user\_file "/usr/local/nignx/pass";

… …

[root@proxy ~]# htpasswd -c /usr/local/nginx/pass jerry //创建账户文件

New password:123 //输入密码

Re-type new password:123 //确认密码

[root@proxy ~]# nginx -s reload

2）在proxy主机使用tcpdump命令抓包

[root@proxy ~]# tcpdump -A host 192.168.4.5 and tcp port 80

3)在真实机使用浏览器访问192.168.4.5

[root@pc001 ~]# firefox http://192.168.4.5 //根据提示输入用户名与密码

4）回到proxy查看抓包的数据结果

[root@proxy ~]# tcpdump -A host 192.168.4.5 and tcp port 80

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

… …

Authorization: Basic dG9tOjEyMzQ1Ng==

… …

5) 查看base64编码内容

[root@proxy ~]# echo "dG9tOjEyMzQ1Ng==" | base64 -d

tom:123456

[root@proxy ~]# echo "tom:123456" | base64

dG9tOjEyMzQ1Ngo=

# NSD SECURITY DAY03

# 系统审计

系统审计（又称信息系统审计）指对一个信息系统的运行状况进行检查与评价，以判断信息系统是否能够保证资产的安全、数据的完整以及有效率利用[组织](https://baike.baidu.com/item/%E7%BB%84%E7%BB%87" \t "/root/文档\\x/_blank)的资源并有效果地实现组织目标。

**系统审计三种基本类型**

信息系统真实性审计

息系统安全性审计

信息系统绩效审计

# 部署audit审计工具

服务名:auditd 系统无法停止该服务

yum -y install audit

24 rpm -qa | grep audit

25 cat /etc/audit/auditd.conf

26 ls /var/log/audit/audit.log

27 cat -b /var/log/audit/audit.log # -b 输出到屏幕显示行号(日志)

30 systemctl status auditd

34 systemctl enable auditd

## auditctl 命令配置审计规则

[root@client50 lnmp\_soft]# auditctl -s #查询规则状态

enabled 1

failure 1

pid 637

rate\_limit 0

backlog\_limit 8192

lost 0

backlog 0

loginuid\_immutable 0 unlocked

[root@client50 lnmp\_soft]# auditctl -l #查看已有规则

No rules

[root@client50 lnmp\_soft]# auditctl -D #删除所有规则

No rules

## auditctl**定义临时规则：**

auditctl  -w path(文件路径/命令路径) -p permission -k key\_name(自起的规则名)

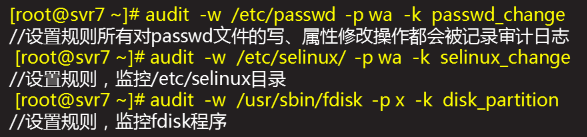
]# auditctl  -w /etc/passwd -p wxa -k "mon\_dead"

-w 监控文件路径 /etc/passwd  可以是命令

-p 监控文件筛选 r(读) w(写) x(执行) a(属性改变)

-k 筛选字符串，用于查询监控日志

**例子:**



[root@client50 ~]# auditctl -w /usr/sbin/fdisk -p x -k disk\_one

[root@client50 ~]# auditctl -l

-w /usr/sbin/fdisk -p x -k disk\_one

[root@client50 ~]# auditctl -w /etc/passwd -p rwa -k passwd\_change

[root@client50 ~]# auditctl -l

-w /usr/sbin/fdisk -p x -k disk\_one

-w /etc/passwd -p rwa -k passwd\_change

## audit.rules定义永久规则

配置文件:/etc/audit/rules.d/audit.rules

[root@client50 ~]# ls /etc/audit/rules.d/

audit.rules

[root@client50 ~]# cat /etc/audit/rules.d/

cat: /etc/audit/rules.d/: 是一个目录

[root@client50 ~]# auditctl -l

-w /usr/sbin/fdisk -p x -k disk\_one

-w /etc/passwd -p rwa -k passwd\_change

[root@client50 ~]# echo "-w /usr/sbin/fdisk -p x -k disk\_one" >> /etc/audit/rules.d/audit.rules

[root@client50 ~]# echo "-w /etc/passwd -p rwa -k passwd\_change" >> /etc/audit/rules.d/audit.rules

[root@client50 ~]# cat /etc/audit/rules.d/audit.rules

## First rule - delete all

-D

## Increase the buffers to survive stress events.

## Make this bigger for busy systems

-b 8192

## Set failure mode to syslog

-f 1

-w /usr/sbin/fdisk -p x -k disk\_one #末尾添加规则

-w /etc/passwd -p rwa -k passwd\_change #末尾添加规则

## ausearch查看审计日志信息

设置了监控后，会在/var/log/audit/audit.log里出现日志。

可以用此命令查看日志：

ausearch -f /etc/passwd -x vim

-k  利用auditctl指定的key\_name查询

-x  执行程序

-i 交互式操作

# ausearch -ts today -k passwd\_change

# ausearch -ts 3/12/07 -k passwd\_change

-ts 指定时间后的log (start time)

-te 指定时间前的log (end time)

56 ausearch -k passwd\_change

57 ausearch -k disk\_on

59 ausearch -f /etc/passwd

60 ausearch -f /etc/passwd -x vim

## 审计日志分析

[root@client50 ~]# useradd audit

[root@client50 ~]# ausearch -k passwd\_change

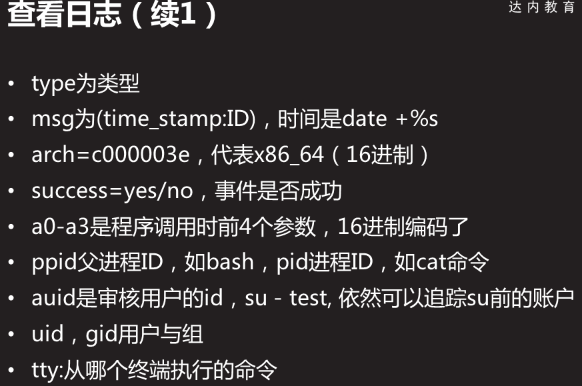
time->Fri Mar 8 10:16:08 2019

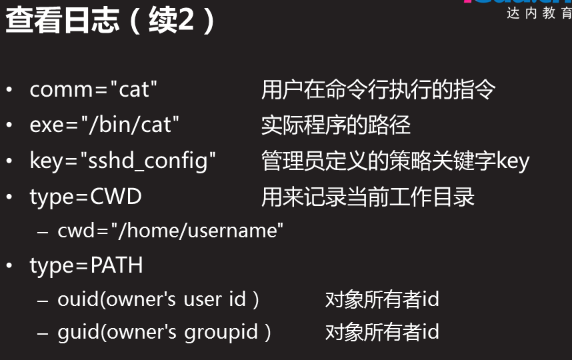
type=PROCTITLE msg=audit(1552011368.758:282): proctitle=75736572616464006175646974

type=PATH msg=audit(1552011368.758:282): item=0 name="/etc/passwd" inode=17357461 dev=fd:00 mode=0100644 ouid=0 ogid=0 rdev=00:00 obj=system\_u:object\_r:passwd\_file\_t:s0 objtype=NORMAL

type=CWD msg=audit(1552011368.758:282): cwd="/root"

type=SYSCALL msg=audit(1552011368.758:282): arch=c000003e syscall=2 success=yes exit=4 a0=7f27f80a1432 a1=80000 a2=1b6 a3=24 items=1 ppid=1631 pid=2556 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=2 comm="useradd" exe="/usr/sbin/useradd" subj=unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023 key="passwd\_change"





# 网站服务安全

## nginx服务安全

安装时候去掉不需要的模块

./configure --without(禁用模块) --with(加载模块)

## **nginx服务,自动索引模块**

**未取消自动索引模块的**

[root@client50 nginx-1.12.2]# ./configure --help #查看模块

[root@client50 nginx-1.12.2]# ./configure --help | grep -i autoindex

--without-http\_autoindex\_module disable ngx\_http\_autoindex\_module

68 tar -xf nginx-1.12.2.tar.gz

69 ls

70 cd nginx-1.12.2/

71 ls

72 yum -y install gcc pcre-devel openssl-devel

73 ./configure --help

74 ./configure --prefix=/usr/local/nginx

75 make && make install

76 ./configure --help | grep -i autoindex

77 vim /usr/local/nginx/conf/nginx.conf

.....

server {

autoindex on ;

listen 80;

server\_name localhost;

......

78 /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

83 mkdir /usr/local/nginx/html/bbs

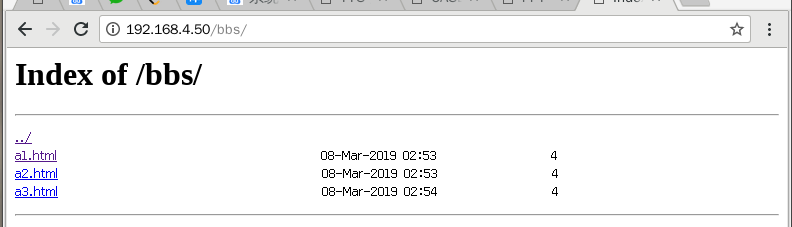
84 echo 123 >/usr/local/nginx/html/bbs/a1.html

85 echo 456 >/usr/local/nginx/html/bbs/a2.html

86 echo 789 >/usr/local/nginx/html/bbs/a3.html

87 /usr/local/nginx/sbin/nginx

**未禁用自动索引模块访问bbs目录暴露所有的网页,**



114 /usr/local/nginx/sbin/nginx -stop

115 ./configure --prefix=/usr/local/nginx --without-http\_autoindex\_module

116 make && make install

117 vim /usr/local/nginx/conf/nginx.conf

.....

server {

autoindex on ;

listen 80;

server\_name localhost;

......

118 /usr/local/nginx/sbin/nginx

nginx: [emerg] unknown directive "autoindex" in /usr/local/nginx/conf/nginx.conf:36

#因为去掉了without-http\_autoindex\_module 模块.不能在配置文件nginx.conf

中添加(索引开关)autoindex on ; 的配置,添加了则在启动时报错

重装nginx 并去自动掉索引模块 就报错

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx

nginx: [emerg] unknown directive "autoindex" in /usr/local/nginx/conf/nginx.conf:36

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -t

nginx: [emerg] unknown directive "autoindex" in /usr/local/nginx/conf/nginx.conf:36

nginx: configuration file /usr/local/nginx/conf/nginx.conf test failed

## nginx修改版本信息(修改源码)

修改版本信息(修改源码)

隐藏版本号信息(server\_token off)

[root@client50 src]# curl -I http://192.168.4.50

HTTP/1.1 200 OK

Server: nginx/1.12.2 #nginx软件名版本

Date: Fri, 08 Mar 2019 03:28:16 GMT

Content-Type: text/html

Content-Length: 612

Last-Modified: Fri, 08 Mar 2019 03:10:14 GMT

Connection: keep-alive

ETag: "5c81dd16-264"

Accept-Ranges: bytes

## nginx限制并发

[root@client50 src]# ./configure --help | grep -i limit

-bash: ./configure: 没有那个文件或目录

[root@client50 src]# cd ..

[root@client50 nginx-1.12.2]# ./configure --help | grep -i limit

--without-http\_limit\_conn\_module disable ngx\_http\_limit\_conn\_module

--without-http\_limit\_req\_module disable ngx\_http\_limit\_req\_module

--without-stream\_limit\_conn\_module disable ngx\_stream\_limit\_conn\_module

--without-http\_limit\_req\_module 默认启用模块,该模块可以降低DDos攻击风险

**修改配置文件,定义并发参数**

[root@client50 nginx-1.12.2]# vim /usr/local/nginx/conf/nginx.conf

....

http {

limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

#

...

server {

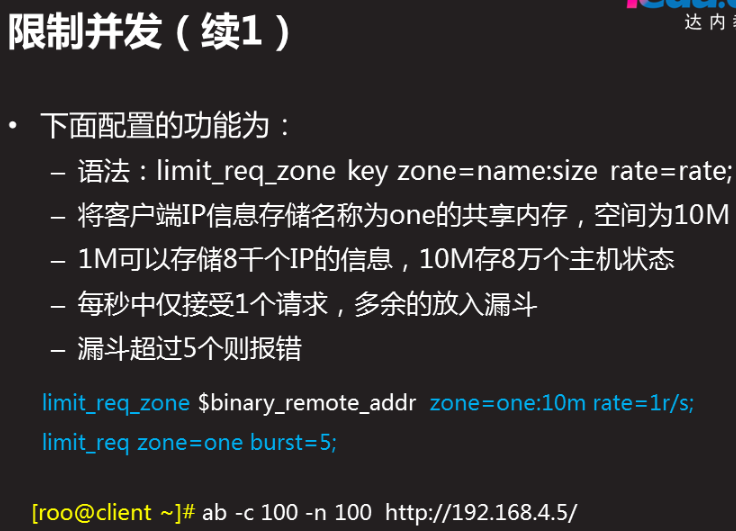
limit\_req zone=one burst=5; #

....

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -t

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx

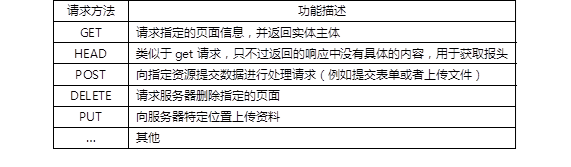
660 ab -n 10 -c 10 http://192.168.4.50/ #压力测试



## nginx拒绝非法请求

网站使用的是HTTP协议，该协议中定义了很多方法，可以让用户连接服务器，获得需要的资源。但实际应用中一般仅需要get和post。

具体HTTP请求方法的含义如表-1所示。



170 curl -i -X GET http://192.168.4.50

171 curl -i -X HEAD http://192.168.4.50

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

server {

listen 80;

#这里，!符号表示对正则取反，~符号是正则匹配符号

#如果用户使用非GET或POST方法访问网站，则retrun返回444的错误信息

if ($request\_method !~ ^(GET|POST)$ ) {

return 444;

}

}

}

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx

[root@client50 nginx-1.12.2]# curl -i -X HEAD http://192.168.4.50

curl: (52) Empty reply from server #使用 HEAD访问则报:来自服务器的空答复

## nginx防止buffer(缓冲区)溢出

当客户端连接服务器时，服务器会启用各种缓存，用来存放连接的状态信息。

如果攻击者发送大量的连接请求，而服务器不对缓存做限制的话，内存数据就有可能溢出（空间不足）。

修改Nginx配置文件，调整各种buffer参数，可以有效降低溢出风险。

有效降低机器Dos攻击风险

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

client\_body\_buffer\_size 1K;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 1k;

large\_client\_header\_buffers 2 1k;

… …

server {

.....

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

# mariadb数据库服务安全

192 rpm -q mariadb

193 yum -y install mariadb mariadb-server.x86\_64

194 systemctl start mariadb.service

195 netstat -unltp | grep 3306

196 ps -C mysqd

197 ps -C mysqld

## 初始化安全脚本

mysql\_secure\_installation

[root@client50 /]# mysql #连续tab

mysqlbug mysqldumpslow mysql\_secure\_installation mysql\_waitpid

mysqlcheck mysql\_find\_rows mysql\_setpermission

[root@client50 /]# mysql\_secure\_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB

SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current

password for the root user. If you've just installed MariaDB, and

you haven't set the root password yet, the password will be blank,

so you should just press enter here.

Enter current password for root (enter for none): #

OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB

root user without the proper authorisation.

Set root password? [Y/n] y #

New password:

Re-enter new password:

Password updated successfully!

Reloading privilege tables..

... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone

to log into MariaDB without having to have a user account created for

them. This is intended only for testing, and to make the installation

go a bit smoother. You should remove them before moving into a

production environment.

Remove anonymous users? [Y/n] y #

... Success!

Normally, root should only be allowed to connect from 'localhost'. This

ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y #

... Success!

By default, MariaDB comes with a database named 'test' that anyone can

access. This is also intended only for testing, and should be removed

before moving into a production environment.

Remove test database and access to it? [Y/n] y #

- Dropping test database...

... Success!

- Removing privileges on test database...

... Success!

Reloading the privilege tables will ensure that all changes made so far

will take effect immediately.

Reload privilege tables now? [Y/n] y #

... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB

installation should now be secure.

Thanks for using MariaDB!

## 密码安全

**手动修改MariaDB或MySQL数据库密码的方法：**

[root@proxy ~]# mysqladmin -uroot -predhat password 'mysql'

//修改密码，旧密码为redhat，新密码为mysql

[root@proxy ~]# mysql -uroot -pmysql

MariaDB [(none)]>set password for root@'localhost'=password('redhat');

//使用账户登录数据库，修改密码

MariaDB [(none)]> select user,host,password from mysql.user;

+--------+---------+---------------------------------------------+

| user | host | password |

+--------+---------+---------------------------------------------+

| root | localhost | \*84BB5DF4823DA319BBF86C99624479A198E6EEE9 |

| root | 127.0.0.1 | \*84BB5DF4823DA319BBF86C99624479A198E6EEE9 |

| root | ::1 | \*84BB5DF4823DA319BBF86C99624479A198E6EEE9 |

+--------+-----------+--------------------------------------------+

修改密码成功，而且密码在数据库中是加密的，有什么问题吗？问题是你的密码被明文记录了，下面来看看明文密码：

**.bash\_history**

[root@proxy ~]# cat .bash\_history

mysqladmin -uroot -pxxx password 'redhat'

//通过命令行修改的密码，bash会自动记录历史，历史记录中记录了明文密码

**.mysql\_history**

[root@proxy ~]# cat .mysql\_history

set password for root@'localhost'=password('redhat');

select user,host,password from mysql.user;

flush privileges;

//通过mysql命令修改的密码，mysql也会有所有操作指令的记录，这里也记录了明文密码

另外数据库还有一个binlog日志里也有明文密码（5.6版本后修复了）。

怎么解决？

管理好自己的历史，不使用明文登录，选择合适的版本5.6以后的版本，

日志，行为审计（找到行为人），使用防火墙从TCP层设置ACL（禁止外网接触数据库）。

**定期执行密码安全操作**

* **管理好历史记录**

rm .bash\_history

history -C

rm .mysql\_history

* 日志,行为审计
* 防火墙

## 数据备份与安全

## 数据安全

# tomcat服务安全

## 隐藏版本信息

245 cd /usr/local/tomcat/

246 ls

247 cd lib/

248 ls

249 jar -xf catalina.jar

[root@client50 ~]# ls /usr/local/tomcat/lib/org/apache/catalina/util/ServerInfo.properties

/usr/local/tomcat/lib/org/apache/catalina/util/ServerInfo.properties

[root@client50 ~]# vim /usr/local/tomcat/lib/org/apache/catalina/util/ServerInfo.properties

修改之前

server.info=Apache Tomcat/8.0.30

server.number=8.0.30.0

server.built=Dec 1 2015 22:30:46 UTC

修改之后

server.info=hksb

server.number=9.0.9.0

server.built=Dec 1 2015 22:30:46 UTC



错误页面最后显示”hksb”

[root@client50 ~]# vim /usr/local/tomcat/conf/server.xml

69 <Connector port="8080" protocol="HTTP/1.1"

70 connectionTimeout="20000"

71 redirectPort="8443" server="hkdb2" />

[root@client50 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@client50 ~]# /usr/local/tomcat/bin/startup.sh

## 降权启动

[root@client50 ~]# useradd tomcat

[root@client50 ~]# chown -R tomcat:tomcat /usr/local/tomcat

[root@client50 ~]# su -c /usr/local/tomcat/bin/startup.sh tomcat

Using CATALINA\_BASE: /usr/local/tomcat

Using CATALINA\_HOME: /usr/local/tomcat

Using CATALINA\_TMPDIR: /usr/local/tomcat/temp

Using JRE\_HOME: /usr

Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar

Tomcat started.

[root@client50 ~]# ps aux |grep java

## 设置源码服务开机自启

进系统瞬间调用/etc/rc.local文件.将启动的命令写入这个文件中

[root@client50 ~]# vim /etc/rc.local

touch /var/lock/subsys/local

su -c /usr/local/tomcat/bin/startup.sh tomcat #文件末尾添加

## 删除默认的测试页面

[root@client50 ~]# rm -rf /usr/local/tomcat/webapps/\*

# Linux安全之打补丁

## 1、 diff命令

diff的功能就是用来比较两个文件的不同，然后记录下来，也就是所谓的diff补丁。

语法格式：diff 【选项】 源文件(夹) 目的文件(夹)，就是要给源文件(夹)打个补丁，使之变成目的文件(夹)，术语也就是“升级”。

下面介绍三个最为常用选项：

-r 是一个递归选项，设置了这个选项，diff会将两个不同版本源代码目录中的所有对应文 件全部都进行一次比较，包括子目录文件。

-N 选项确保补丁文件将正确地处理已经创建或删除文件的情况。

-u 输出同一内容的头部信息(打补丁使用),生成补丁文件

-a 所有文件视为文本,包括二进制程序

[root@client50 ~]# sh v1.sh #版本

hello boy

[root@client50 ~]# sh v1-1.sh #版本第一次修改

hello boy

hello girl

[root@client50 ~]# diff -u v1.sh v1-1.sh

--- v1.sh 2019-03-08 16:49:11.830548575 +0800

+++ v1-1.sh 2019-03-08 16:49:42.118548575 +0800

@@ -1,2 +1,3 @@ #第一个文件减去1行,减第2行, 第二个文件加1行,加第3行

#! /bin/bash

echo "hello boy"

+echo "hello girl"

## 2、diff命令生成补丁(50主机)

[root@client50 ~]# diff -u v1.sh v1-1.sh >one.patch

[root@client50 ~]# scp one.patch 192.168.4.51:/root

## patch命令

查看安装

[root@sql51 ~]# which patch

/usr/bin/which: no patch in (/root/perl5/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin)

[root@sql51 ~]# yum -y install patch #安装

[root@sql51 ~]# which patch

/usr/bin/patch

patch就是利用diff制作的补丁来实现源文件(夹)和目的文件(夹)的转换。这样说就意味着你可以有源文件(夹)――>目的文件(夹)，也可以目的文件(夹)――>源文件(夹)。

下面介绍几个最常用选项：

-p0 选项要从当前目录查找目的文件(夹),补丁文件路径不变

p:path 0:为删除补丁文件中0层路径前缀

-p1 选项要忽略掉第一层目录，从当前目录开始查找(当前所在目录判断)。

在这里以实例说明：

--- old/modules/pcitable Mon Sep 27 11:03:56 1999

+++ new/modules/pcitable Tue Dec 19 20:05:41 2000

如果使用参数-p0，那就表示从当前目录找一个叫做old的文件夹，在它下面寻找modules下的pcitable文件来执行patch操作。

如果使用参数-p1，那就表示忽略第一层目录(即不管old)，从当前目录寻找modules的文件夹，在它下面找pcitable。这样的前提是 当前目录必须为modules所在的目录。而diff补丁文件则可以在任意位置，只要指明了diff补丁文件的路径就可以了。当然，可以用相对路径，也可 以用绝对路径。不过我一般习惯用相对路径。

-E 选项说明如果发现了空文件，那么就删除它

-R 选项说明在补丁文件中的“新”文件和“旧”文件现在要调换过来了(实际上就是给新版本打补丁，让它变成老版本)

## 4、patch命令打补丁(51主机)

第一次打补丁

[root@sql51 ~]# cat one.patch

--- v1.sh 2019-03-08 16:49:11.830548575 +0800

+++ v1-1.sh 2019-03-08 16:49:42.118548575 +0800

@@ -1,2 +1,3 @@

#! /bin/bash

echo "hello boy"

+echo "hello girl"

[root@sql51 ~]# ls one.patch v1.sh

one.patch v1.sh

[root@sql51 ~]# patch -p0 -i one.patch #-i和反向重定向(重定向输入) < 功能一样

patching file v1.sh

[root@sql51 ~]# cat v1.sh

#! /bin/bash

echo "hello boy"

echo "hello girl"

第二次打补丁

## 目录补丁

**环境准备(50)**

[root@client50 demo]# which find

/usr/bin/find #二进制文件find

[root@client50 demo]# cp /usr/bin/find source1/

[root@client50 demo]# cp /usr/bin/find source2/

[root@client50 demo]# echo 1 >>source2/find #造成不同的find文件

[root@client50 demo]# tree /demo

/demo

├── source1

│   ├── find

│   └── test.sh

└── source2

├── find #find文件中多了个1

├── test.sh

└── tmp.txt #source2多了tmp.txt文件

2 directories, 5 files

[root@client50 demo]# ls -l /demo

总用量 0

drwxr-xr-x. 2 root root 33 3月 8 17:36 source1

drwxr-xr-x. 2 root root 48 3月 8 17:36 source2

[root@client50 demo]# diff -uarN source1 source2 >/root/plj.patch

[root@client50 demo]# scp -r source1 192.168.4.51:/demo/

#将source1拷贝给客户端51的安装目录,模拟原始安装

[root@client50 demo]# scp /root/plj.patch 192.168.4.51:/root

#将补丁文件plj.patch 拷贝给客户端51

**客户端打补丁(51)**

[root@sql51 demo]# tree /demo

/demo

└── source1

├── find

└── test.sh

1 directory, 2 files

[root@sql51 demo]# cat /root/plj.patch #查看一下补丁文件的路径

diff -uarN source1/find source2/find

--- source1/find 2019-03-08 17:36:09.993548575 +0800

#要给source1目录打补丁,一定要进入source1目录打补丁

+++ source2/find 2019-03-08 17:36:36.295548575 +0800 #相对路径

[root@sql51 demo]# cd source1/

[root@sql51 demo]# pwd

/demo

#要给source1目录打补丁,一定要进入source1上级目录范围内打补丁

[root@sql51 demo]# patch -p0 < /root/plj.patch

patching file source1/find

patching file source1/test.sh

patching file source1/tmp.txt

绝对路径打补丁

[root@sql51 demo]# patch -p2 < /root/zdd.patch

patching file source1/find

patching file source1/test.sh

patching file source1/tmp.txt

[root@sql51 demo]# cat /root/zdd.patch

diff -uarN /demo/source1/find /demo/source2/find

--- /demo/source1/find 2019-03-08 17:36:09.993548575 +0800

+++ /demo/source2/find 2019-03-08 17:36:36.295548575 +0800

# tree工具

[root@client50 demo]# yum -y install tree

[root@client50 demo]# mkdir source1 source2

[root@client50 demo]# tree /demo

/demo

├── source1

└── source2

2 directories, 0 files

[root@client50 demo]# echo "hello world" > source1/test.sh

[root@client50 demo]# echo "hello the world" > source2/test.sh

[root@client50 demo]# tree /demo

/demo

├── source1

│   └── test.sh

└── source2

└── test.sh

# 案例

案例1：部署audit监控文件

案例2：加固常见服务的安全

案例3：使用diff和patch工具打补丁

1 案例1：部署audit监控文件

1.1 问题

本案例要求熟悉audit审计工具的基本使用，完成以下任务操作：

使用audit监控/etc/ssh/sshd\_config

当该文件发生任何变化即记录日志

通过手动和ausearch工具查看日志内容

1.2 方案

审计的目的是基于事先配置的规则生成日志，记录可能发生在系统上的事件（正常或非正常行为的事件），审计不会为系统提供额外的安全保护，但她会发现并记录违反安全策略的人及其对应的行为。

审计能够记录的日志内容：

a) 日期与事件以及事件的结果

b) 触发事件的用户

c) 所有认证机制的使用都可以被记录，如ssh等

d) 对关键数据文件的修改行为等都可以被记录

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置audit审计系统

1）安装软件包，查看配置文件（确定审计日志的位置）

[root@proxy ~]# yum -y install audit //安装软件包

[root@proxy ~]# cat /etc/audit/auditd.conf //查看配置文件，确定日志位置

log\_file = /var/log/audit/audit.log //日志文件路径

[root@proxy ~]# systemctl start auditd //启动服务

[root@proxy ~]# systemctl enable auditd //设置开机自启

2）配置审计规则

可以使用auditctl命令控制审计系统并设置规则决定哪些行为会被记录日志。

语法格式如下：

[root@proxy ~]# auditctl -s //查询状态

[root@proxy ~]# auditctl -l //查看规则

[root@proxy ~]# auditctl -D //删除所有规则

定义临时文件系统规则：

#语法格式：auditctl -w path -p permission -k key\_name

# path为需要审计的文件或目录

# 权限可以是r,w,x,a(文件或目录的属性发生变化)

# Key\_name为可选项，方便识别哪些规则生成特定的日志项

[root@proxy ~]# auditctl -w /etc/passwd -p wa -k passwd\_change

//设置规则所有对passwd文件的写、属性修改操作都会被记录审计日志

[root@proxy ~]# auditctl -w /etc/selinux/ -p wa -k selinux\_change

//设置规则，监控/etc/selinux目录

[root@proxy ~]# auditctl -w /usr/sbin/fdisk -p x -k disk\_partition

//设置规则，监控fdisk程序

[root@proxy ~]# auditclt -w /etc/ssh/sshd\_conf -p warx -k sshd\_config

//设置规则，监控sshd\_conf文件

如果需要创建永久审计规则，则需要修改规则配置文件：

[root@proxy ~]# vim /etc/audit/rules.d/audit.rules

-w /etc/passwd -p wa -k passwd\_changes

-w /usr/sbin/fdisk -p x -k partition\_disks

步骤二：查看并分析日志

1）手动查看日志

查看SSH的主配置文件/etc/ssh/sshd\_conf，查看audit日志信息：

[root@proxy ~]# tailf /var/log/audit/audit.log

type=SYSCALL msg=audit(1517557590.644:229228): arch=c000003e

syscall=2 success=yes exit=3

a0=7fff71721839 a1=0 a2=1fffffffffff0000 a3=7fff717204c0

items=1 ppid=7654 pid=7808 auid=0 uid=0 gid=0 euid=0 suid=0

fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=3 comm="cat"

exe="/usr/bin/cat"

subj=unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023 key="sshd\_config"

.. ..

#内容分析

# type为类型

# msg为(time\_stamp:ID)，时间是date +%s（1970-1-1至今的秒数）

# arch=c000003e，代表x86\_64（16进制）

# success=yes/no，事件是否成功

# a0-a3是程序调用时前4个参数，16进制编码了

# ppid父进程ID，如bash，pid进程ID，如cat命令

# auid是审核用户的id，su - test, 依然可以追踪su前的账户

# uid，gid用户与组

# tty:从哪个终端执行的命令

# comm="cat" 用户在命令行执行的指令

# exe="/bin/cat" 实际程序的路径

# key="sshd\_config" 管理员定义的策略关键字key

# type=CWD 用来记录当前工作目录

# cwd="/home/username"

# type=PATH

# ouid(owner's user id） 对象所有者id

# guid(owner's groupid） 对象所有者id

2）通过工具搜索日志

系统提供的ausearch命令可以方便的搜索特定日志，默认该程序会搜索/var/log/audit/audit.log，ausearch options -if file\_name可以指定文件名。

[root@proxy ~]# ausearch -k sshd\_config -i

//根据key搜索日志，-i选项表示以交互式方式操作

2 案例2：加固常见服务的安全

2.1 问题

本案例要求优化提升常见网络服务的安全性，主要完成以下任务操作：

优化Nginx服务的安全配置

优化MySQL数据库的安全配置

优化Tomcat的安全配置

2.2 方案

Nginx安全优化包括：删除不要的模块、修改版本信息、限制并发、拒绝非法请求、防止buffer溢出。

MySQL安全优化包括：初始化安全脚本、密码安全、备份与还原、数据安全。

Tomcat安全优化包括：隐藏版本信息、降权启动、删除默认测试页面.

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：优化Nginx服务的安全配置

1） 删除不需要的模块

Nignx是模块化设计的软件，需要什么功能与模块以及不需要哪些模块，都可以在编译安装软件时自定义，使用--with参数可以开启某些模块，使用--without可以禁用某些模块。最小化安装永远都是对的方案！

下面是禁用某些模块的案例：

[root@proxy ~]# tar -xf nginx-1.12.tar.gz

[root@proxy ~]# cd nginx-1.12

[root@proxy nginx-1.12]# ./configure \

>--without-http\_autoindex\_module \ //禁用自动索引文件目录模块

>--without-http\_ssi\_module

[root@proxy nginx-1.12]# make

[root@proxy nginx-1.12]# make install

2） 修改版本信息，并隐藏具体的版本号

默认Nginx会显示版本信息以及具体的版本号，这些信息给攻击者带来了便利性，便于他们找到具体版本的漏洞。

如果需要屏蔽版本号信息，执行如下操作，可以隐藏版本号。

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

server\_tokens off; //在http下面手动添加这么一行

… …

}

[root@proxy ~]# nginx -s reload

[root@proxy ~]# curl -I http://192.168.4.5 //查看服务器响应的头部信息

但服务器还是显示了使用的软件为nginx，通过如下方法可以修改该信息。

[root@proxy nginx-1.12]# vim +48 src/http/ngx\_http\_header\_filter\_module.c

//注意：vim这条命令必须在nginx-1.12源码包目录下执行！！！！！！

//该文件修改前效果如下：

static u\_char ngx\_http\_server\_string[] = "Server: nginx" CRLF;

static u\_char ngx\_http\_server\_full\_string[] = "Server: " NGINX\_VER CRLF;

static u\_char ngx\_http\_server\_build\_string[] = "Server: " NGINX\_VER\_BUILD CRLF;

//下面是我们修改后的效果：

static u\_char ngx\_http\_server\_string[] = "Server: Jacob" CRLF;

static u\_char ngx\_http\_server\_full\_string[] = "Server: Jacob" CRLF;

static u\_char ngx\_http\_server\_build\_string[] = "Server: Jacob" CRLF;

//修改完成后，再去编译安装Nignx，版本信息将不再显示为Nginx，而是Jacob

[root@proxy nginx-1.12]# ./configure

[root@proxy nginx-1.12]# make && make install

[root@proxy nginx-1.12]# killall nginx

[root@proxy nginx-1.12]# /usr/local/nginx/sbin/nginx //启动服务

[root@proxy nginx-1.12]# curl -I http://192.168.4.5 //查看版本信息验证

3） 限制并发量

DDOS攻击者会发送大量的并发连接，占用服务器资源（包括连接数、带宽等），这样会导致正常用户处于等待或无法访问服务器的状态。

Nginx提供了一个ngx\_http\_limit\_req\_module模块，可以有效降低DDOS攻击的风险，操作方法如下：

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

… …

limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

server {

listen 80;

server\_name localhost;

limit\_req zone=one burst=5;

}

}

//备注说明：

//limit\_req\_zone语法格式如下：

//limit\_req\_zone key zone=name:size rate=rate;

//上面案例中是将客户端IP信息存储名称为one的共享内存，内存空间为10M

//1M可以存储8千个IP信息，10M可以存储8万个主机连接的状态，容量可以根据需要任意调整

//每秒中仅接受1个请求，多余的放入漏斗

//漏斗超过5个则报错

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

客户端使用ab测试软件测试效果：

[root@client ~]# ab -c 100 -n 100 http://192.168.4.5/

4） 拒绝非法的请求

网站使用的是HTTP协议，该协议中定义了很多方法，可以让用户连接服务器，获得需要的资源。但实际应用中一般仅需要get和post。

具体HTTP请求方法的含义如表-1所示。

表-1 HTTP请求方法及含义

未修改服务器配置前，客户端使用不同请求方法测试：

[root@client ~]# curl -i -X GET http://192.168.4.5 //正常

[root@client ~]# curl -i -X HEAD http://192.168.4.5 //正常

//curl命令选项说明：

//-i选项：访问服务器页面时，显示HTTP的头部信息

//-X选项：指定请求服务器的方法

通过如下设置可以让Nginx拒绝非法的请求方法：

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

server {

listen 80;

#这里，!符号表示对正则取反，~符号是正则匹配符号

#如果用户使用非GET或POST方法访问网站，则retrun返回444的错误信息

if ($request\_method !~ ^(GET|POST)$ ) {

return 444;

}

}

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

修改服务器配置后，客户端使用不同请求方法测试：

[root@client ~]# curl -i -X GET http://192.168.4.5 //正常

[root@client ~]# curl -i -X HEAD http://192.168.4.5 //报错

4） 防止buffer溢出

当客户端连接服务器时，服务器会启用各种缓存，用来存放连接的状态信息。

如果攻击者发送大量的连接请求，而服务器不对缓存做限制的话，内存数据就有可能溢出（空间不足）。

修改Nginx配置文件，调整各种buffer参数，可以有效降低溢出风险。

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

client\_body\_buffer\_size 1K;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 1k;

large\_client\_header\_buffers 2 1k;

… …

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

步骤二：数据库安全

1） 初始化安全脚本

安装完MariaDB或MySQL后，默认root没有密码，并且提供了一个任何人都可以操作的test测试数据库。有一个名称为mysql\_secure\_installation的脚本，该脚本可以帮助我们为root设置密码，并禁止root从远程其他主机登陆数据库，并删除测试性数据库test。

[root@proxy ~]# systemctl status mariadb

//确保服务已启动

[root@proxy ~]# mysql\_secure\_installation

//执行初始化安全脚本

2）密码安全

手动修改MariaDB或MySQL数据库密码的方法：

[root@proxy ~]# mysqladmin -uroot -predhat password 'mysql'

//修改密码，旧密码为redhat，新密码为mysql

[root@proxy ~]# mysql -uroot -pmysql

MariaDB [(none)]>set password for root@'localhost'=password('redhat');

//使用账户登录数据库，修改密码

MariaDB [(none)]> select user,host,password from mysql.user;

+--------+---------+---------------------------------------------+

| user | host | password |

+--------+---------+---------------------------------------------+

| root | localhost | \*84BB5DF4823DA319BBF86C99624479A198E6EEE9 |

| root | 127.0.0.1 | \*84BB5DF4823DA319BBF86C99624479A198E6EEE9 |

| root | ::1 | \*84BB5DF4823DA319BBF86C99624479A198E6EEE9 |

+--------+-----------+--------------------------------------------+

修改密码成功，而且密码在数据库中是加密的，有什么问题吗？问题是你的密码被明文记录了，下面来看看明文密码：

[root@proxy ~]# cat .bash\_history

mysqladmin -uroot -pxxx password 'redhat'

//通过命令行修改的密码，bash会自动记录历史，历史记录中记录了明文密码

[root@proxy ~]# cat .mysql\_history

set password for root@'localhost'=password('redhat');

select user,host,password from mysql.user;

flush privileges;

//通过mysql命令修改的密码，mysql也会有所有操作指令的记录，这里也记录了明文密码

另外数据库还有一个binlog日志里也有明文密码（5.6版本后修复了）。

怎么解决？

管理好自己的历史，不使用明文登录，选择合适的版本5.6以后的版本，

日志，行为审计（找到行为人），使用防火墙从TCP层设置ACL（禁止外网接触数据库）。

3）数据备份与还原

首先，备份数据库（注意用户名为root，密码为redhat）：

[root@proxy ~]# mysqldump -uroot -predhat mydb table > table.sql

//备份数据库中的某个数据表

[root@proxy ~]# mysqldump -uroot -predhat mydb > mydb.sql

//备份某个数据库

[root@proxy ~]# mysqldump -uroot -predhat --all-databases > all.sql

//备份所有数据库

接下来，还原数据库（注意用户名为root，密码为redhat）：

[root@proxy ~]# mysql -uroot -predhat mydb < table.sql //还原数据表

[root@proxy ~]# mysql -uroot -predhat mydb < mydb.sql //还原数据库

[root@proxy ~]# mysql -uroot -predhat < all.sql //还原所有数据库

4）数据安全

在服务器上（192.168.4.5），创建一个数据库账户：

[root@proxy ~]# mysql -uroot -predhat

//使用管理员，登陆数据库

MariaDB [(none)]> grant all on \*.\* to tom@'%' identified by '123';

//创建一个新账户tom

使用tcpdump抓包（192.168.4.5）

[root@proxy ~]# tcpdump -w log -i any src or dst port 3306

//抓取源或目标端口是3306的数据包，保存到log文件中

客户端（192.168.4.100）从远程登陆数据库服务器（192.168.4.5）

[root@client ~]# mysql -utom -p123 -h 192.168.4.5

//在192.168.4.100这台主机使用mysql命令登陆远程数据库服务器（192.168.4.5）

//用户名为tom，密码为123

MariaDB [(none)]> select \* from mysql.user;

//登陆数据库后，任意执行一条查询语句

回到服务器查看抓取的数据包

[root@proxy ~]# tcpdump -A -r log

//使用tcpdump查看之前抓取的数据包，很多数据库的数据都明文显示出来

如何解决？

可以使用SSH远程连接服务器后，再从本地登陆数据库（避免在网络中传输数据，因为网络环境中不知道有没有抓包者）。

或者也可以使用SSL对MySQL服务器进行加密，类似与HTTP+SSL一样，MySQL也支持SSL加密（确保网络中传输的数据是被加密的）。

步骤三：Tomcat安全性

1） 隐藏版本信息、修改tomcat主配置文件（隐藏版本信息）

未修改版本信息前，使用命令查看服务器的版本信息

注意：proxy有192.168.2.5的IP地址，这里使用proxy作为客户端访问192.168.2.100服务器。

[root@proxy ~]# curl -I http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看头部信息

[root@proxy ~]# curl -I http://192.168.2.100:8080

//访问存在的页面文件，查看头部信息

[root@proxy ~]# curl http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看错误信息

修改tomcat配置文件，修改版本信息(在192.168.2.100操作)：

[root@web1 tomcat]# yum -y install java-1.8.0-openjdk-devel

[root@web1 tomcat]# cd /usr/local/tomcat/lib/

[root@web1 lib]# jar -xf catalina.jar

[root@web1 lib]# vim org/apache/catalina/util/ServerInfo.properties

//根据自己的需要，修改版本信息的内容

[root@web1 lib]# /usr/local/tomcat/bin/shutdown.sh //关闭服务

[root@web1 lib]# /usr/local/tomcat/bin/startup.sh //启动服务

修改后，客户端再次查看版本信息（在192.168.2.5操作）：

[root@proxy ~]# curl -I http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看头部信息

[root@proxy ~]# curl -I http://192.168.2.100:8080

//访问存在的页面文件，查看头部信息

[root@proxy ~]# curl http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看错误信息

再次修改tomcat服务器配置文件，修改版本信息，手动添加server参数（在192.168.2.100操作）：

[root@web1 lib]# vim /usr/local/tomcat/conf/server.xml

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000" redirectPort="8443" server="jacob" />

[root@web1 lib]# /usr/local/tomcat/bin/shutdown.sh //关闭服务

[root@web1 lib]# /usr/local/tomcat/bin/startup.sh //启动服务

修改后,客户端再次查看版本信息（在192.168.2.5操作）：

[root@proxy ~]# curl -I http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看头部信息

[root@proxy ~]# curl -I http://192.168.2.100:8080

//访问存在的页面文件，查看头部信息

[root@proxy ~]# curl http://192.168.2.100:8080/xx

//访问不存在的页面文件，查看错误信息

2）降级启动

默认tomcat使用系统高级管理员账户root启动服务，启动服务尽量使用普通用户。

[root@web1 ~]# useradd tomcat

[root@web1 ~]# chown -R tomcat:tomcat /usr/local/tomcat/

//修改tomcat目录的权限，让tomcat账户对该目录有操作权限

[root@web1 ~]# su -c /usr/local/tomcat/bin/startup.sh tomcat

//使用su命令切换为tomcat账户，以tomcat账户的身份启动tomcat服务

[root@web1 ~]# chmod +x /etc/rc.local //该文件为开机启动文件

[root@web1 ~]# vim /etc/rc.local //修改文件，添加如下内容

su -c /usr/local/tomcat/bin/startup.sh tomcat

3）删除默认的测试页面

[root@web1 ~]# rm -rf /usr/local/tomcat/webapps/\*

3 案例3：使用diff和patch工具打补丁

3.1 问题

本案例要求优化提升常见网络服务的安全性，主要完成以下任务操作：

使用diff对比文件差异

使用diff生成补丁文件

使用patch命令为旧版本打补丁

3.2 方案

程序是人设计出来的，总是会有这样那样的问题与漏洞，目前的主流解决方法就是为有问题的程序打补丁，升级新版本。

在Linux系统中diff命令可以为我们生成补丁文件，然后使用patch命令为有问题的程序代码打补丁。

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：对比单个文件差异

1） 编写两个版本的脚本，一个为v1版本，一个为v2版本。

[root@proxy ~]# cat test1.sh //v1版本脚本

#!/bin/bash

echo "hello wrld"

[root@proxy ~]# cat test2.sh //v2版本脚本

#!/bin/bash

echo "hello the world"

echo "test file"

2） 使用diff命令语法

使用diff命令查看不同版本文件的差异。

[root@proxy ~]# diff test1.sh test2.sh //查看文件差异

@@ -1,3 +1,3 @@

#!/bin/bash

-echo "hello world"

-echo "test"

+echo "hello the world"

+echo "test file"

[root@proxy ~]# diff -u test1.sh test2.sh //查看差异，包含头部信息

--- test1.sh 2018-02-07 22:20:02.723971251 +0800

+++ test2.sh 2018-02-07 22:20:13.358760687 +0800

@@ -1,3 +1,3 @@

#!/bin/bash

-echo "hello world"

-echo "test"

+echo "hello the world"

+echo "test file"

diff制作补丁文件的原理：告诉我们怎么修改第一个文件后能得到第二个文件。

这样如果第一个版本的脚本有漏洞，我们不需要将整个脚本都替换，仅需要修改有问题的一小部分代码即可，diff刚好可以满足这个需求！

像Linux内核这样的大块头，一旦发现有一个小漏洞，我们不可能把整个内核都重新下载，全部替换一遍，而仅需要更新有问题的那一小部分代码即可！

diff命令常用选项：

-u 输出统一内容的头部信息（打补丁使用），计算机知道是哪个文件需要修改

-r 递归对比目录中的所有资源（可以对比目录）

-a 所有文件视为文本（包括二进制程序）

-N 无文件视为空文件（空文件怎么变成第二个文件）

-N选项备注说明：

A目录下没有txt文件，B目录下有txt文件

diff比较两个目录时，默认会提示txt仅在B目录有（无法对比差异，修复文件）

diff比较时使用N选项，则diff会拿B下的txt与A下的空文件对比，补丁信息会明确说明如何从空文件修改后变成txt文件，打补丁即可成功！

步骤二：使用patch命令对单文件代码打补丁

1）准备实验环境

[root@proxy ~]# cd demo

[root@proxy demo]# vim test1.sh

#!/bin/bash

echo "hello world"

echo "test"

[root@proxy demo]# vim test2.sh

#!/bin/bash

echo "hello the world"

echo "test file"

2) 生成补丁文件

[root@proxy demo]# diff -u test1.sh test2.sh > test.patch

3）使用patch命令打补丁

在代码相同目录下为代码打补丁

[root@proxy demo]# yum -y install patch

[root@proxy demo]# patch -p0 < test.patch //打补丁

patching file test1.sh

//patch -pnum（其中num为数字，指定删除补丁文件中多少层路径前缀）

//如原始路径为/u/howard/src/blurfl/blurfl.c

//-p0则整个路径不变

//-p1则修改路径为u/howard/src/blurfl/blurfl.c

//-p4则修改路径为blurfl/blurfl.c

//-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件

[root@proxy demo]# patch -RE < test.patch //还原旧版本，反向修复

步骤三：对比目录中所有文件的差异

1） 准备实验环境

[root@proxy ~]# mkdir demo

[root@proxy ~]# cd demo

[root@proxy demo]# mkdir {source1,source2}

[root@proxy demo]# echo "hello world" > source1/test.sh

[root@proxy demo]# cp /bin/find source1/

[root@proxy demo]# tree source1/ //source1目录下2个文件

|-- find

`-- test.sh

[root@proxy demo]# echo "hello the world" > source2/test.sh

[root@proxy demo]# echo "test" > source2/tmp.txt

[root@proxy demo]# cp /bin/find source2/

[root@proxy demo]# echo "1" >> source2/find

[root@proxy demo]# tree source2/ //source1目录下3个文件

|-- find

|-- test.sh

`-- tmp.txt

//注意：两个目录下find和test.sh文件内容不同，source2有tmp.txt而source1没有该文件

2）制作补丁文件

[root@proxy demo]# diff -u source1/ source2/

//仅对比了文本文件test.sh；二进制文件、tmp都没有对比差异，仅提示，因为没有-a和-N选项

[root@proxy demo]# diff -Nu source1/ source2/

//对比了test.sh，并且使用source2目录的tmp.txt与source1的空文件对比差异。

[root@proxy demo]# diff -Nua source1/ source2/

//对比了test.sh、tmp.txt、find(程序)。

步骤四：使用patch命令对目录下的所有代码打补丁

1）使用前面创建的source1和source2目录下的代码为素材，生成补丁文件

[root@proxy ~]# cd demo

[root@proxy demo]# diff -Nuar source1/ source2/ > source.patch

2）使用patch命令为代码打补丁

[root@proxy demo]# ls

source1 source2 source.patch

[root@proxy demo]# cat source.patch //对比的文件有路径信息

--- source1/test.sh 2018-02-07 22:51:33.034879417 +0800

+++ source2/test.sh 2018-02-07 22:47:32.531754268 +0800

@@ -1 +1 @@

-hello world

+hello the world

[root@proxy demo]# cd source1

[root@proxy source1]# patch -p1 < ../source.patch

# NSD SECURITY DAY04

RHEL7的防火墙服务是firewalld,调用的还是iptables

RHEL6的防火墙服务是iptables

# 1：iptables防火墙

**一、防火墙的分类**

**(一)、包过滤防火墙iptables。**

数据包过滤(packet Filtering)技术是在网络层对数据包进行选择，选择的依据是系统内设置的过滤逻辑，称为访问控制表(access control lable,ACL)。通过检查数据流中每个数据包的源地址和目的地址，所用的端口号和协议状态等因素，或他们的组合来确定是否允许该数据包通过。

包过滤防火墙的优点是它对用户来说是透明的，处理速度快且易于维护。缺点是：非法访问一旦突破防火墙，即可对主机上的软件和配置漏洞进行攻击；数据包的源 地址、目的地址和IP的端口号都在数据包的头部，可以很轻易地伪造。“IP地址欺骗”是黑客针对该类型防火墙比较常用的攻击手段。

**(二)、代理服务型防火墙**

代理服务(proxy service)也称链路级网关或TCP通道。它是针对数据包过滤和应用网关技术存在的缺点而引入的防火墙技术，其特点是将所有跨跃防火墙的网络通信链路 分为两段。当代理服务器接收到用户对某个站点的访问请求后就会检查请求是否符合控制规则。如果规则允许用户访问该站点，代理服务器就会替用户去那个站点取 回所需的信息，再转发给用户，内外网用户的访问都是通过代理服务器上的“链接”来实现的，从而起到隔离防火墙内外计算机系统的作用。

**二、防火墙的工作原理**

(一)、包过滤防火墙的工作原理

包过滤是在IP层实现的，因此，它可以只用路由器来完成。包过滤根据包的源IP地址、目的IP地址、源端口、目的端口及包传递方向等报头信息来判断是否允 许包通过，过滤用户定义的内容，如IP地址。其工作原理是系统在网络层检查数据包，与应用层无关。包过滤器的应用非常广泛，因为CPU用来处理包过滤的时 间可以忽略不计。而且这种防护措施对用户透明，合法用户在进出网络时，根本感觉不到它的存在，使用起来很方便。这样系统就具有很好的传输性能，易扩展。

但是这种防火墙不太安全，因为系统对应用层信息无感知－－也就是说，它们不理解通信的内容，不能在用户级别上进行过滤，即不能识别不同的用户和防止地址的 盗用。如果攻击者把自己主机的IP地址设成一个合法主机的IP地址，就可以很轻易地通过包过滤器，这样更容易被黑客攻破。基于这种工作机制，包过滤防火墙 有以下缺陷：

1、通信信息：包过滤防火墙只能访问部分数据包的头信息。

2、通信和应用状态信息：包过滤防火墙是无状态的，所以它不可能保存来自于通信和应用的状态信息。

3、信息处理：包过滤防火墙处理信息的能力是有限的。

(二)、代理服务型防火墙工作原理

代理服务型防火墙在应用层上实现防火墙功能。它能提供部分与传输有关的状态，能外圈提供与应用相关的状态和部分传输的信息，它还能处理和管理信息。

## iptables安装

33 systemctl stop firewalld

34 rpm -q firewalld

35 rpm -q iptables

36 rpm -q iptables -services

37 rpm -q iptables-services

38 yum -y install iptables-services

39 systemctl start iptables

40 systemctl status iptables

# 2iptables防火墙的组成

## 四张表



## raw表:对数据包做状态跟踪

## mangle表对到达防火墙的数据包打标记

## nat表对到达防火墙的数据进行地址转换

## filter表对到达防火墙的数据进行按规则过滤(默认)

## 五条链(数据传输方向)

## INPUT链(进):

匹配进入防火墙本机的数据包

## OUTPUT链(出):

匹配从防火墙本机出去的数据包

## FORWARD链(经过):

匹配经过防火墙主机的数据包

## POSTROUTING链(路由后):

数据包路由后的处理

## PREROUTING链(路由前):

数据包路由前的处理

每条链内有多条规则的匹配顺序

顺序比对,[匹配即停止(LOG除外)

若无任何匹配,则按该链的默认策略处理



# iptables命令管理防火墙

**命令格式:**

iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

**注意事项:**

可以不指定表,默认为filter表

可以不指定链,默认为对应表的所有链

如果没有匹配的规则,则使用防火墙默认规则

选项/链名/目标操作用大写字母,其余都小写

**命令选项**

命令选项用于指定iptables的执行方式，包括插入规则、删除规则和添加规则等：

默认策略 -P 或--policy 定义默认策略

添加 -A 或--append 在规则列表的最后增加一条规则

添加 -I 或--insert 在指定的位置插入一条规则

删除 -D 或--delete 在规则列表中删除一条规则

删除 -F 或--flush 删除表中的所有规则

查看 -L 或--list 查看iptables规则列表

查看 -n 以数字形式显示地址\端口等信息

查看 --line-nimbers 查看规则时显示规则行号

-R 或--replace 替换规则列表中的某条规则

-Z 或--zero 将表中所有链的计数和流量计数器都清零

**基本的目标操作:**

ACCEPT: 允许通过/放行

DROP: 直接丢弃,不给出任何回应

REJECT: 拒绝通过,必要时会给出提示

LOG: 记录日志,然后传给下一条规则(匹配即停止规律的唯一例外)

## 添加新的规则

**命令格式:**

iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

添加 -A 或--append 在规则列表的最后增加一条规则

添加 -I 或--insert 在指定的位置插入一条规则

[root@client50 ~]# iptables -t filter -nL INPUT #查看为空

Chain INPUT (policy ACCEPT)

target prot opt source destination

[root@client50 ~]# iptables -t filter -A INPUT -p tcp -j ACCEPT #添加-p(指定协议)

[root@client50 ~]# iptables -t filter -nL INPUT

Chain INPUT (policy ACCEPT)

target prot opt source destination

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0

[root@client50 ~]# iptables -t filter -A INPUT -p udp -j ACCEPT #添加

[root@client50 ~]# iptables -t filter -nL INPUT

Chain INPUT (policy ACCEPT)

target prot opt source destination

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0

ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

[root@client50 ~]# iptables -I INPUT -p udp -j ACCEPT #不指定表默认filter表

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0

3 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

[root@client50 ~]# iptables -I INPUT 2 -p icmp -j ACCEPT #添加指定排第二

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

2 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0

3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0

4 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

[root@client50 ~]# iptables -D INPUT 1 #删除INPUT链中第一条

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 #第二条变第一条

2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0

3 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

[root@client50 ~]# iptables -F INPUT #清空INPUT链规则

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

## 查看:

**命令格式:**

iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

查看 -L 或--list 查看iptables规则列表

查看 -n 以数字形式显示地址\端口等信息

查看 --line-nimbers 查看规则时显示规则行号

47 iptables -L

48 iptables -nL

49 iptables -nL INPUT --line-numbers

50 iptables -t nat -nL --line-numbers

51 iptables -t nat -nL INPUT --line-numbers

[root@client50 ~]# iptables -L

Chain INPUT (policy ACCEPT)

#INPUT链

target prot opt source destination

ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED

ACCEPT icmp -- anywhere anywhere

ACCEPT all -- anywhere anywhere

ACCEPT tcp -- anywhere anywhere state NEW tcp dpt:ssh

REJECT all -- anywhere anywhere reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)

target prot opt source destination

REJECT all -- anywhere anywhere reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

## 清空表规则

命令格式:

iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

删除 -D 或--delete 在规则列表中删除一条规则

删除 -F 或--flush 删除表中的所有规则

iptables -F #清空所有表所有规则

53 iptables -t filter -F

55 iptables -t nat -F

56 iptables -t mangle -F

57 iptables -t raw -F

## 设置默认规则(DROP或ACCEPT)



[root@client50 ~]# iptables -t filter -P FORWARD DROP

[root@client50 ~]# iptables -t filter -nL

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy DROP)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

[root@client50 ~]# iptables -t filter -P FORWARD ACCEPT

[root@client50 ~]# iptables -t filter -nL

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

[root@client50 ~]#

# 4：filter表控制(主机型和网络型防火墙)

**主机型防火墙:(在服务器本机运行防火墙服务)**

**网络型防火墙:在两个网络之间运行的防火墙**

## 4.1匹配条件

**命令格式:**

iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

### 4.1.1基本匹配条件



104 iptables -t filter -A INPUT -p tcp --dport 8080 -j ACCEPT

105 iptables -t filter -A INPUT -p tcp --dport 3306 -j ACCEPT

114 iptables -t filter -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

**设置规则,在INPUT链路,禁用22端口**

[root@client50 ~]# iptables -A INPUT -s 192.168.4.254 -p tcp --dport 22 -j ACCEPT

[root@client50 ~]# iptables -nL

Chain INPUT (policy ACCEPT)

target prot opt source destination

ACCEPT tcp -- 192.168.4.254 0.0.0.0/0 tcp dpt:22

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

**#设置INPUT默认为DROP**

[root@client50 ~]# iptables -t filter -P INPUT DROP

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy DROP)

num target prot opt source destination

1 ACCEPT tcp -- 192.168.4.254 0.0.0.0/0 tcp dpt:22

**禁用80端口**

[root@client50 ~]# iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy DROP)

num target prot opt source destination

1 ACCEPT tcp -- 192.168.4.254 0.0.0.0/0 tcp dpt:22

2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80

107 iptables -t filter -A INPUT -p icmp -j ACCEPT #允许ping包

108 iptables -t filter -D INPUT -p icmp -j ACCEPT #去掉允许你ping包的规则

允许50ping别的主机,别的主机不能ping50主机

ping操作的请求包类型:echo-request

ping 操作的应答包类型:echo-reply

[root@client50 ~]# iptables -t filter -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy DROP)

num target prot opt source destination

1 ACCEPT tcp -- 192.168.4.254 0.0.0.0/0 tcp dpt:22

2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80

3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:8080

4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:3306

5 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 0

此时50主机能ping别的主机,,别的主机ping 不了51

在50上ping 192.168.4.51 在51上抓包可看到

### 4.1.2扩展匹配条件



[root@client50 ~]# iptables -t filter -I INPUT -P tcp -m multiport --dport 22,88,8080,3306 -j ACCEPT

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy DROP)

num target prot opt source destination

1 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 22,80,8080,3306

2 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 0

[root@client50 ~]# iptables -t filter -A INPUT -p icmp --icmp-type echo-request -m iprange --src-range 192.168.4.51-192.168.4.56 -j ACCEPT

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy DROP)

num target prot opt source destination

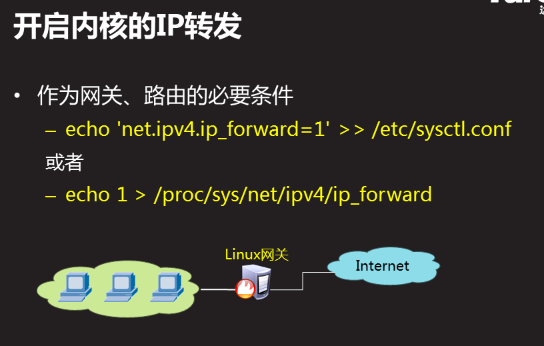
1 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 22,80,8080,3306

2 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 0

3 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 8 source IP range 192.168.4.51-192.168.4.56

## 网络型防火墙:

在两个网络之间运行的防火墙



查看内核的路由转发功能:

[root@client50 ~]# sysctl -a |grep ip\_forward

net.ipv4.ip\_forward = 1

net.ipv4.ip\_forward\_use\_pmtu = 0

## 本机配置路由

15 systemctl stop NetworkManager #必须先停止

16 route add default gw 192.168.4.50 #添加路由

17 route del default gw 192.168.4.50 #删除路由

18 route -n #查看路由

## 网络防火墙实验

### 环境准备:

51 eth0:192.168.4.51 #作为内网客户端访问外网http

50 eth0:192.168.4.50 eth1:192.168.2.50 #作为网络防火墙

53 eth1:192.168.2.53 #作为外网网页客户端

添加地址命令参考:

nmcli connection add ifname eth1 type ethernet con-name eh1 connection.autoconnect yes

### 清空主机型防火墙规则:

[root@client50 ~]# iptables -t filter -P INPUT ACCEPT

[root@client50 ~]# iptables -t filter -F INPUT

[root@client50 ~]# iptables -t filter -nL INPUT --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

[root@client50 ~]# iptables -t filter -nL --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

Chain FORWARD (policy ACCEPT)

num target prot opt source destination

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

### 50配置规则

[root@client50 ~]# iptables -t filter -I FORWARD -p tcp --dport 80 -j ACCEPT

[root@client50 ~]# iptables -t filter -I FORWARD -p tcp --sport 80 -j ACCEPT

[root@client50 ~]# iptables -t filter -A FORWARD -p tcp --dport 22 -j ACCEPT

[root@client50 ~]# iptables -t filter -A FORWARD -p tcp --sport 22 -j ACCEPT

[root@client50 ~]# iptables -nL

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy DROP)

target prot opt source destination

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:80

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:22

# 5：nat表典型应用

所有局域网内的主机共享同一个公网IP地址上网

## 53取消路由网关

取消上个实验

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 192.168.2.50 0.0.0.0 UG 0 0 0 eth1

192.168.2.0 0.0.0.0 255.255.255.0 U 100 0 0 eth1

192.168.122.0 0.0.0.0 255.255.255.0 U 0 0 0 virbr0

[root@sql53 ~]# route del default gw 192.168.2.50

[root@sql53 ~]# route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

192.168.2.0 0.0.0.0 255.255.255.0 U 100 0 0 eth1

192.168.122.0 0.0.0.0 255.255.255.0 U 0 0 0 virbr0

[root@sql53 ~]# tcpdump -i eth1 port 80 #进行抓包

## 50清空规则

取消上个实验

[root@client50 ~]# iptables -t filter -F FORWARD

[root@client50 ~]# iptables -t filter -P FORWARD ACCEPT

[root@client50 ~]# iptables -nL

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

配置nat表的POSTROUTING链路

[root@client50 ~]# iptables -t nat -F

[root@client50 ~]# iptables -t nat -nL

Chain PREROUTING (policy ACCEPT)

target prot opt source destination

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

Chain POSTROUTING (policy ACCEPT)

target prot opt source destination

[root@client50 ~]# iptables -t nat -A POSTROUTING -s 192.168.4.0/24 -p tcp --dport 80 -j SNAT --to-source 192.168.2.50

最终效果:

51是192.168.4.51 能访问53的192.168.2.53的web,

51的路由是

[root@sql51 ~]# route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 192.168.4.50 0.0.0.0 UG 0 0 0 eth0

# 案例

案例1：iptables基本管理

案例2：filter过滤和转发控制

案例3：防火墙扩展规则

案例4：配置SNAT实现共享上网

1 案例1：iptables基本管理

1.1 问题

本案例要求练习iptables命令的使用，按照要求完成以下任务：

关闭firewalld，开启iptables服务

查看防火墙规则

追加、插入防火墙规则

删除、清空防火墙规则

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：关闭firewalld，启动iptables服务

1）关闭firewalld服务器

[root@proxy ~]# systemctl stop firewalld.service

[root@proxy ~]# systemctl disable firewalld.service

2）安装iptables-services并启动服务

[root@proxy ~]# yum -y install iptables-services

[root@proxy ~]# systemctl start iptables.service

步骤二：熟悉iptables框架

1）iptables的4个表（区分大小写）：

iptables默认有4个表，nat表（地址转换表）、filter表（数据过滤表）、raw表（状态跟踪表）、mangle表（包标记表）。

2）iptables的5个链（区分大小写）：

INPUT链（入站规则）

OUTPUT链（出站规则）

FORWARD链（转发规则）

PREROUTING链（路由前规则）

POSTROUTING链（路由后规则）

步骤三：iptables命令的基本使用方法

1）iptabels语法格式

[root@proxy ~]# iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

[root@proxy ~]# iptables -t filter -I INPUT -p icmp -j REJECT

[root@proxy ~]# iptables -t filter -I INPUT -p icmp -j ACCEPT

[root@proxy ~]# iptables -I INPUT -p icmp -j REJECT

//注意事项与规律：

//可以不指定表，默认为filter表

//可以不指定链，默认为对应表的所有链

//如果没有找到匹配条件，则执行防火墙默认规则

//选项/链名/目标操作用大写字母，其余都小写

########################################################################

//目标操作：

// ACCEPT：允许通过/放行

// DROP：直接丢弃，不给出任何回应

// REJECT：拒绝通过，必要时会给出提示

// LOG：记录日志，然后传给下一条规则

iptables命令的常用选项如表-1所示。

表-1 iptables常用选项

2）iptables命令的使用案例

创建规则的案例：

[root@proxy ~]# iptables -t filter -A INPUT -p tcp -j ACCEPT

//追加规则至filter表中的INPUT链的末尾，允许任何人使用TCP协议访问本机

[root@proxy ~]# iptables -I INPUT -p udp -j ACCEPT

//插入规则至filter表中的INPUT链的开头，允许任何人使用UDP协议访问本机

[root@proxy ~]# iptables -I INPUT 2 -p icmp -j ACCEPT

//插入规则至filter表中的INPUT链的第2行，允许任何人使用ICMP协议访问本机

查看iptables防火墙规则

[root@proxy ~]# iptables -nL INPUT //仅查看INPUT链的规则

target prot opt source destination

ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0

ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0

ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0

[root@proxy ~]# iptables -L INPUT --line-numbers //查看规则，显示行号

num target prot opt source destination

1 ACCEPT udp -- anywhere anywhere

2 ACCEPT icmp -- anywhere anywhere

3 ACCEPT tcp -- anywhere anywhere

删除规则，清空所有规则

[root@proxy ~]# iptables -D INPUT 3

//删除filter表中INPUT链的第3条规则

[root@proxy ~]# iptables -nL INPUT //查看规则，确认是否删除

[root@proxy ~]# iptables -F

//清空filter表中所有链的防火墙规则

[root@proxy ~]# iptables -t nat -F

//清空nat表中所有链的防火墙规则

[root@proxy ~]# iptables -t mangle -F

//清空mangle表中所有链的防火墙规则

[root@proxy ~]# iptables -t raw -F

//清空raw表中所有链的防火墙规则

设置防火墙默认规则

[root@proxy ~]# iptables -t filter -P INPUT DROP

[root@proxy ~]# iptables -nL

Chain INPUT (policy DROP)

… …

2 案例2：filter过滤和转发控制

2.1 问题

本案例要求创建常用主机防火墙规则以及网络防火墙规则：

针对Linux主机进行出站、入站控制

利用ip\_forward机制实现Linux路由/网关功能

在Linux网关上实现数据包转发访问控制

2.2 方案

根据防火墙保护的对象不同，防火墙可以分为主机型防火墙与网络型防火墙，如图-1所示。

主机型防火墙，主要保护的是服务器本机（过滤威胁本机的数据包）。

网络防火墙，主要保护的是防火墙后面的其他服务器，如web服务器、FTP服务器等。

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：iptables防火墙规则的条件

iptables防火墙可以根据很多很灵活的规则进行过滤行为，具体常用的过滤条件如表-2所示。

表-2 iptables过滤条件

1）主机型防火墙案例

[root@proxy ~]# iptables -I INPUT -p tcp --dport 80 -j REJECT

[root@proxy ~]# iptables -I INPUT -s 192.168.2.100 -j REJECT

[root@proxy ~]# iptables -I INPUT -d 192.168.2.5 -p tcp --dport 80 -j REJECT

[root@proxy ~]# iptables -I INPUT -i eth0 -p tcp --dport 80 -j REJECT

[root@proxy ~]# iptables -A INPUT -s 192.168.4.100 -j DROP

//丢弃192.168.4.100发给本机的所有数据包

[root@proxy ~]# iptables -A INPUT -s 192.168.2.0/24 -j DROP

//丢弃192.168.2.0/24网络中所有主机发送给本机的所有数据包

[root@proxy ~]# iptables -A INPUT -s 114.212.33.12 -p tcp --dport 22 -j REJECT

//拒绝114.212.33.12使用tcp协议远程连接本机ssh（22端口）

步骤二：开启Linux的路由转发功能

1）Linux内核默认支持软路由功能，通过修改内核参数即可开启或关闭路由转发功能。

[root@proxy ~]# echo 0 > /proc/sys/net/ipv4/ip\_forward //关闭路由转发

[root@proxy ~]# echo 1 > /proc/sys/net/ipv4/ip\_forward //开启路由转发

//注意以上操作仅当前有效，计算机重启后无效

[root@proxy ~]# echo 'net.ipv4.ip\_forward=1' >> /etc/sysctl.conf

//修改/etc/sysctl.conf配置文件，可以实现永久有效规则

步骤四：网络型防火墙案例

1）网络型防火墙案例

部署如表-3所示的网络拓扑，一定要把proxy主机的路由转发功能打开。

表-3 实验拓扑

添加网关的命令

[root@client ~]# nmcli connection modify eth0 ipv4.gateway 192.168.4.5

[root@client ~]# nmcli connection up eth0

[root@web1 ~]# nmcli connection modify eth1 ipv4.gateway 192.168.2.5

[root@web1 ~]# nmcli connection up eth1

确认不同网络的联通性

[root@client ~]# ping 192.168.2.100

[root@web1 ~]# ping 192.168.4.100

在web1主机上启动http服务

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "test page" > /var/www/html/index.html

[root@web1 ~]# systemctl restart httpd

没有防火墙的情况下client访问web服务

[root@client ~]# curl http://192.168.2.100 //成功

设置proxy主机的防火墙规则，保护防火墙后面的Web服务器

[root@proxy ~]# iptables -I FORWARD -s 192.168.4.100 -p tcp --dport 80 -j DROP

设置完防火墙规则后，再次使用client客户端访问测试效果

[root@client ~]# curl http://192.168.2.100 //失败

步骤三：禁ping的相关策略

1）默认直接禁ping的问题？

[root@proxy ~]# iptables -I INPUT -p icmp -j DROP

//设置完上面的规则后，其他主机确实无法ping本机，但本机也无法ping其他主机

//当本机ping其他主机，其他主机回应也是使用icmp，对方的回应被丢弃

2）禁止其他主机ping本机，允许本机ping其他主机

[root@proxy ~]# iptables -A INPUT -p icmp \

> --icmp-type echo-request -j DROP

//仅禁止入站的ping请求，不拒绝入站的ping回应包

注意：关于ICMP的类型，可以参考help帮助，参考命令如下：

[root@proxy ~]# iptables -p icmp --help

3 案例3：防火墙扩展规则

3.1 问题

本案例要求熟悉使用iptables的扩展规则，实现更丰富的过滤功能，完成以下任务：

根据MAC地址封锁主机

在一条规则中开放多个TCP服务

根据IP范围设置封锁规则

3.2 方案

iptables在基本过滤条件的基础上还扩展了很多其他条件，在使用时需要使用-m参数来启动这些扩展功能，语法如下：

iptables 选项 链名称 -m 扩展模块 --具体扩展条件 -j 动作

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：根据MAC地址过滤

1）根据IP过滤的规则，当对方修改IP后，防火墙会失效

[root@proxy ~]# iptables -F

[root@proxy ~]# iptables -I INPUT -s 192.168.4.100 -p tcp --dport 22 -j DROP

//设置规则禁止192.168.4.100使用ssh远程本机

但是，当client主机修改IP地址后，该规则就会失效，注意因为修改了IP，对client主机的远程连接会断开，需要使用virt-manager开启虚拟机操作：

[root@client ~]# ifconfig eth0 192.168.4.101

[root@client ~]# ssh 192.168.4.5 //依然成功

根据MAC地址过滤，可以防止这种情况的发生

[root@client ~]# ip link show eth0 //查看client的MAC地址

eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP mode DEFAULT qlen 1000

link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff

[root@proxy ~]# iptables -A INPUT -p tcp --dport 22\

> -m mac --mac-source 52:54:00:00:00:0b -j DROP

//拒绝52:54:00:00:00:0b这台主机远程本机

步骤二：基于多端口设置过滤规则

1）一次需要过滤或放行很多端口时会比较方便

[root@proxy ~]# iptables -A INPUT -p tcp \

> -m multiport --dports 20:22,25,80,110,143,16501:16800 -j ACCEPT

//一次性开启20,21,22,25,80,110,143,16501到16800所有的端口

提示，多端口还可以限制多个源端口，但因为源端口不固定，一般不会使用，限制多个源端口的参数是--sports.

步骤三：根据IP地址范围设置规则

1）允许从 192.168.4.10-192.168.4.20 登录

[root@proxy ~]# iptables -A INPUT -p tcp --dport 22 \

> -m iprange --src-range 192.168.4.10-192.168.4.20 -j ACCEPT

注意，这里也可以限制多个目标IP的范围，参数是--dst-range,用法与--src-range一致。

2）禁止从 192.168.4.0/24 网段其他的主机登录

[root@proxy ~]# iptables -A INPUT -p tcp --dport 22 -s 192.168.4.0/24 -j DROP

4 案例4：配置SNAT实现共享上网

4.1 问题

本案例要求设置防火墙规则，允许位于局域网中的主机可以访问外网，主要包括下列服务：

搭建内外网案例环境

配置SNAT策略实现共享上网访问

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：搭建内外网案例环境

表-4 实验拓扑

这里，我们设定192.168.2.0/24网络为外部网络，192.168.4.0/24为内部网络。

现在，在外部网络中有一台web服务器192.168.2.100，因为设置了网关，client已经可以访问此web服务器了。但，如果查看web1的日志就会发现，日志里记录的是192.168.4.100在访问网页。

我们需要实现的效果是，client可以访问web服务器，但要伪装为192.168.2.5后再访问web服务器(模拟所有位于公司内部的电脑都使用的是私有IP，希望访问外网，就需要伪装为公司的外网IP后才可以)。

步骤二：设置防火墙规则，实现IP地址的伪装（SNAT源地址转换）

1）确保proxy主机开启了路由转发功能

[root@proxy ~]# echo 1 > /proc/sys/net/ipv4/ip\_forward //开启路由转发

2）设置防火墙规则，实现SNAT地址转换

[root@proxy ~]# iptables -t nat -A POSTROUTING \

> -s 192.168.4.0/24 -p tcp --dport 80 -j SNAT --to-source 192.168.2.5

3）登陆web主机查看日志

[root@proxy ~]# tail /var/log/httpd/access\_log

.. ..

192.168.2.5 - - [12/Aug/2018:17:57:10 +0800] "GET / HTTP/1.1" 200 27 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"

通过日志会发现，客户端是先伪装为了192.168.2.5之后再访问的web服务器！

4）扩展知识，对于proxy外网IP不固定的情况可以执行下面的地址伪装，动态伪装IP。

[root@proxy ~]# iptables -t nat -A POSTROUTING \

> -s 192.168.4.0/24 -p tcp --dport 80 -j MASQUERADE

最后，所有iptables规则都是临时规则，如果需要永久保留规则需要执行如下命令:

[root@proxy ~]# service iptables save

# NSD SECURITY DAY05

## 查看Linux系统状态的常用命令

查看内存信息:free free| awk ’/Mem/{print $4} 查看剩余内存容量’

查看交换分区信息:swapon -s

查看磁盘信息:df df | awk '/\/$/{print $5}' //查看根分区的利用率

查看磁盘状态信息:iostat

查看CPU信息:uptime //查看CPU负载（1，5，15分钟）

uptime |awk '{print $NF}' //仅查看CPU的15分钟平均负载

查看网卡信息

ifconfig eth0

ifconfig eth0 |awk '/inet /{print $2}' //查看IP地址信息

ifconfig eth0 |awk '/RX p/{print $5}' //网卡接受数据包流量

ifconfig eth0 |awk '/TX p/{print $5}' //网卡发送数据包流量

查看端口信息

ss -ntulp //查看本机监听的所有端口

查看网络连接信息

ss -antup //查看所有的网络连接信息

查看路由追踪: traceroute www.baidu.com

查看进程: ps

-A ：所有的进程均显示出来，与 -e 具有同样的效用；

-a ： 显示现行终端机下的所有进程，包括其他用户的进程；

-u ：以用户为主的进程状态 ；

x ：通常与 a 这个参数一起使用，可列出较完整信息。

查看系统运行时间:uptime

## awk基本操作方法

格式一: 前置命令 |　awk [选项] '[条件]{指令}' 文件

格式二：awk [选项] '[条件]{指令}' 文件

选项 -F 可指定分隔符,不指定，默认为空格（包括tab）

print：是最常用的编辑指令，若有多条编辑指令，可用分号分隔。

**awk 常用内置变量:**

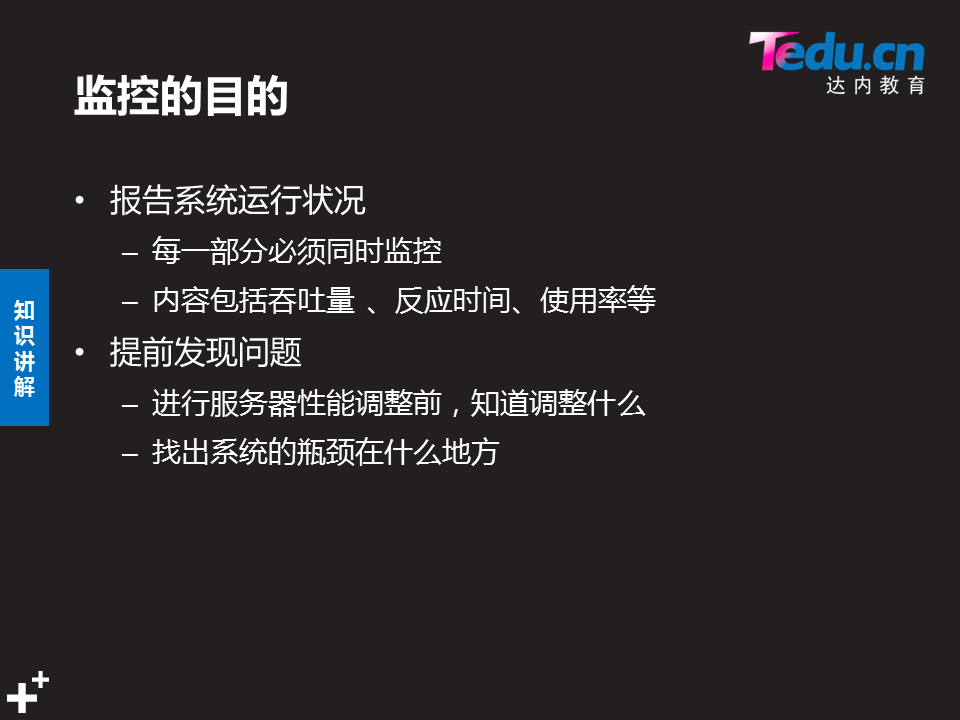
$0 文本当前行的全部内容 $1 文本的第1列

$2 文件的第2列 $3 文件的第3列，依此类推

NR 文件当前行的行号 NF 文件当前行的列数（有几列）

# 1 监控概述

## 监控的目的

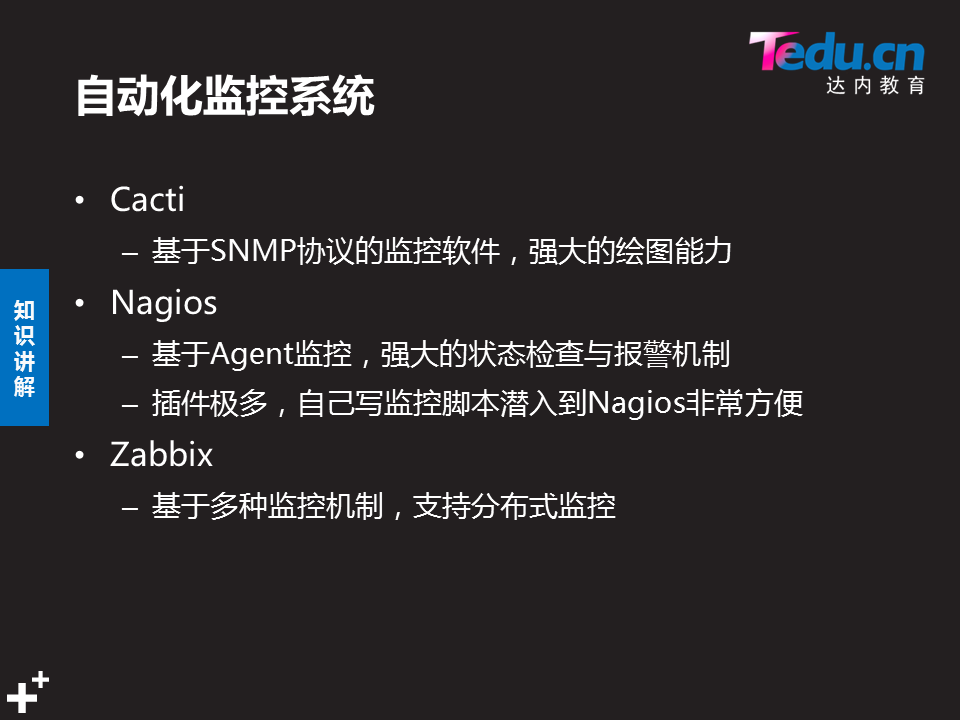


## 监控的资源类别

公开数据:web ftp ssh 数据库等应用服务 tcp或udp 端口

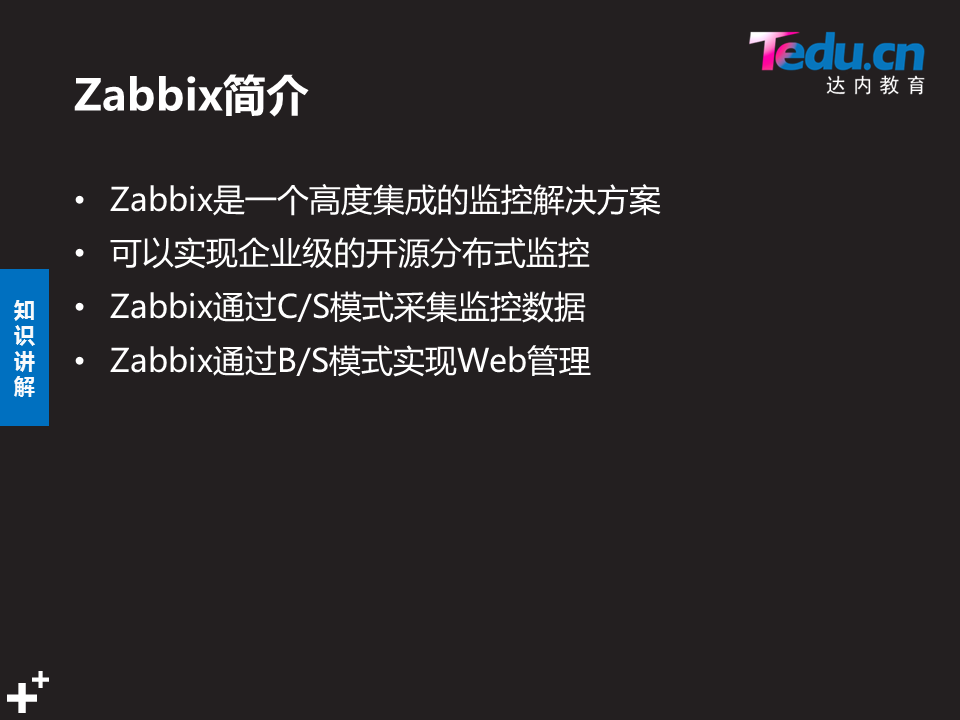
私有数据:cpu 内存 磁盘 网卡 用户 进程等运行信息

## 常见自动化监控系统



当前互联网监控软件大约有300多中,

# 2 Zabbix基础



## 1 安装LNMP环境

15 yum -y install gcc pcre-devel openssl-devel

18 tar -xf lnmp\_soft.tar.gz #需要单独传

24 tar -xf nginx-1.12.2.tar.gz #需要单独传

26 cd nginx-1.12.2/

28 ./configure --with-http\_ssl\_module

29 make && make install

30 ls /usr/local/nginx/

33 cd lnmp\_soft/

35 ls php-fpm-5.4.16-42.el7.x86\_64.rpm

36 yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

38 yum -y install php php-mysql

40 yum -y install mariadb mariadb-devel mariadb-server

## 2 配置启动nginx

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

include fastcgi.conf;

}

[root@zabbixserver ~]# /usr/local/nginx/sbin/nginx

[root@zabbixserver ~]# netstat -unltp | grep 80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 6449/nginx: master

## 3 启动php-ftp服务

[root@zabbixserver ~]# systemctl start php-fpm

[root@zabbixserver ~]# netstat -unltp | grep 9000

tcp 0 0 127.0.0.1:9000 0.0.0.0:\* LISTEN 6507/php-fpm: maste

## 4 启动mariadb

[root@zabbixserver ~]# systemctl start mariadb

[root@zabbixserver ~]# netstat -unltp | grep 3306

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 6764/mysqld

[root@zabbixserver ~]# vim /usr/local/nginx/html/test.php

[root@zabbixserver ~]# curl http://localhost/test.php

## 5 配置nginx,

满足Zbbix服务的运行需求

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

http{

… …

fastcgi\_buffers 8 16k; //缓存php生成的页面内容，8个16k

fastcgi\_buffer\_size 32k; //缓存php生产的头部信息

fastcgi\_connect\_timeout 300; //连接PHP的超时时间

fastcgi\_send\_timeout 300; //发送请求的超时时间

fastcgi\_read\_timeout 300; //读取请求的超时时间

[root@zabbixserver ~]# /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

[root@zabbixserver ~]# /usr/local/nginx/sbin/nginx -s reload

## 6 安装Zabbix依赖包

[root@zabbixserver lnmp\_soft]#yum -y install net-snmp-devel curl-devel

[root@zabbixserver lnmp\_soft]# yum -y install libevent-devel-2.0.21-4.el7.x86\_64.rpm

#注意libevent-devel这个软件包在lnmp\_soft目录下有提供

## 7安装Zabbix

[root@zabbixserver ~]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixserver ~]# cd zabbix-3.4.4/

[root@zabbixserver zabbix-3.4.4]# ls

[root@zabbixserver zabbix-3.4.4]# ./configure --prefix=/usr/local/zabbix --enable-server --enable-proxy \

> --enable-agent --with-mysql=/usr/bin/mysql\_config --with-net-snmp --with-libcurl

....

....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Now run 'make install' \* #提示直接 'make install'

\* \*

\* Thank you for using Zabbix! \*

\* <http://www.zabbix.com> \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//--prefix=/usr/local/zabbix #如果未指定路径,默认装在/user/local/etc/zabbix

// --enable-server安装部署zabbix服务器端软件

// --enable-agent安装部署zabbix被监控端软件

// --enable-proxy安装部署zabbix代理相关软件

// --with-mysql配置mysql\_config路径

// --with-net-snmp允许zabbix通过snmp协议监控其他设备

// --with-libcurl安装相关curl库文件，这样zabbix就可以通过curl连接http等服务，测试被监控主机服务的状态

[root@zabbixserver zabbix-3.4.4]# make install

[root@zabbixserver ~]# ls /usr/local/zabbix/ #安装成功后查看安装的目录里文件

bin etc lib sbin share

## 8初始化Zabbix

### 1创建数据库与授权用户

[root@zabbixserver ~]# systemctl start mariadb

[root@zabbixserver ~]# mysql

MariaDB [(none)]> create database zabbix character set utf8;

Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all on zabbix.\* to zabbix@"localhost" identified by "zabbix";

Query OK, 0 rows affected (0.00 sec)

### 2创建表

(软件提供了恢复文件,在源码目录里面)

[root@zabbixserver ~]# ls /root/zabbix-3.4.4/database/mysql/

data.sql images.sql schema.sql

[root@zabbixserver ~]# cd /root/zabbix-3.4.4/database/mysql/

[root@zabbixserver mysql]# ls

data.sql images.sql schema.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < schema.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < images.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < data.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

....

MariaDB [zabbix]>

### 3创建web页面

[root@zabbixserver mysql]# pwd

/root/zabbix-3.4.4/database/mysql

[root@zabbixserver mysql]# ls /root/zabbix-3.4.4/frontends/

php

[root@zabbixserver mysql]# cd /root/zabbix-3.4.4/frontends/

[root@zabbixserver frontends]# cp -r php /usr/local/nginx/html/zabbix

[root@zabbixserver frontends]# ls /usr/local/nginx/html

50x.html index.html test.php zabbix

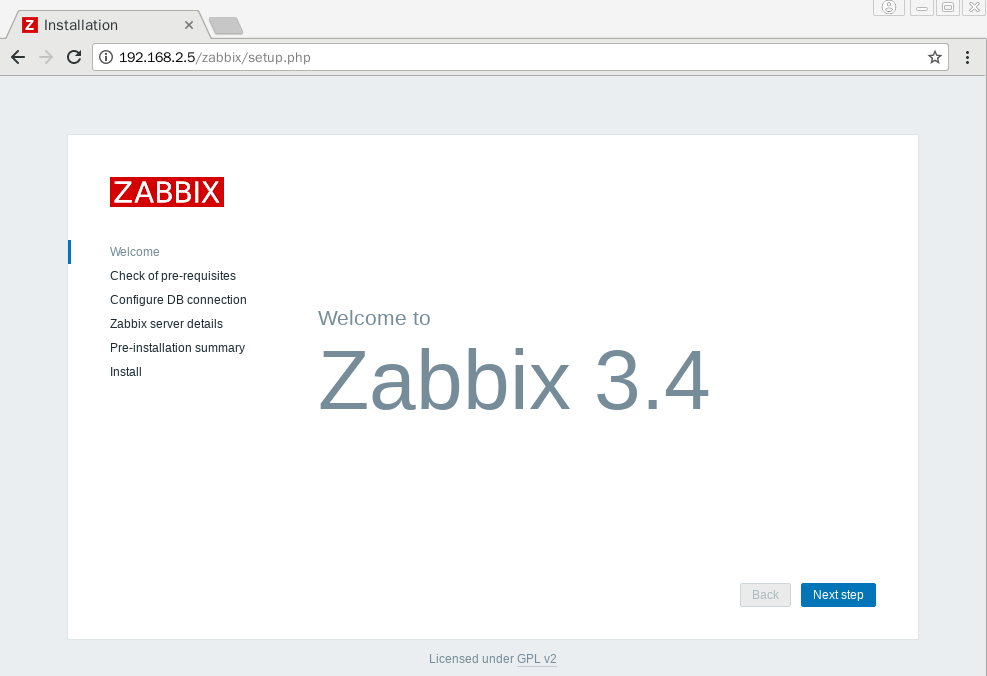
[root@zabbixserver frontends]# ls /usr/local/nginx/html/zabbix/index.php

/usr/local/nginx/html/zabbix/index.php

## 9访问web页面做初始化配置

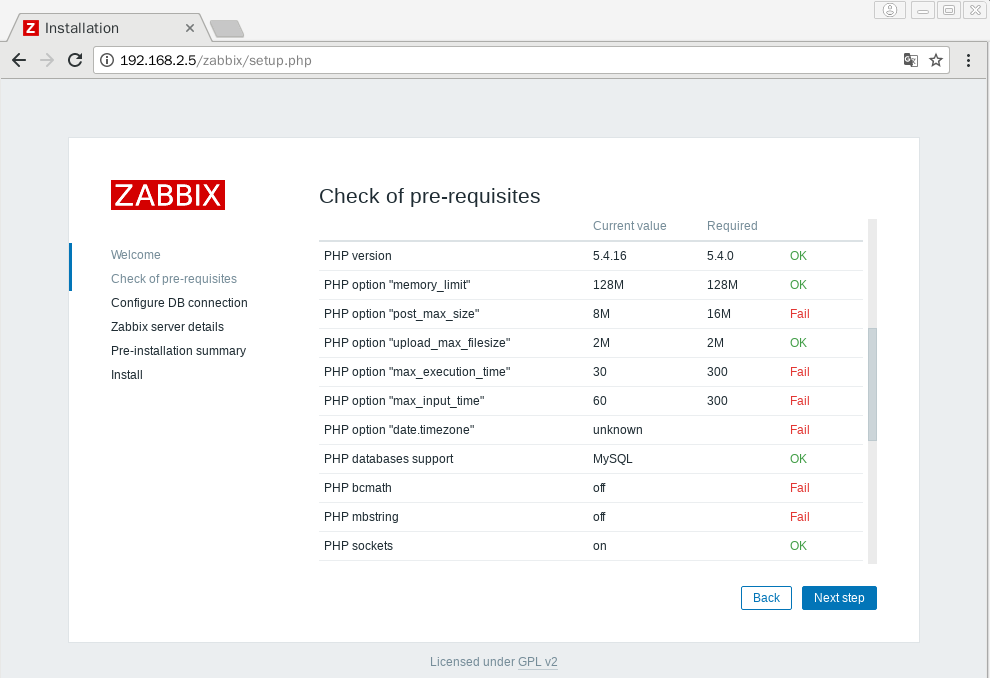
: http://192.168.2.5/zabbix/index.php 如下图

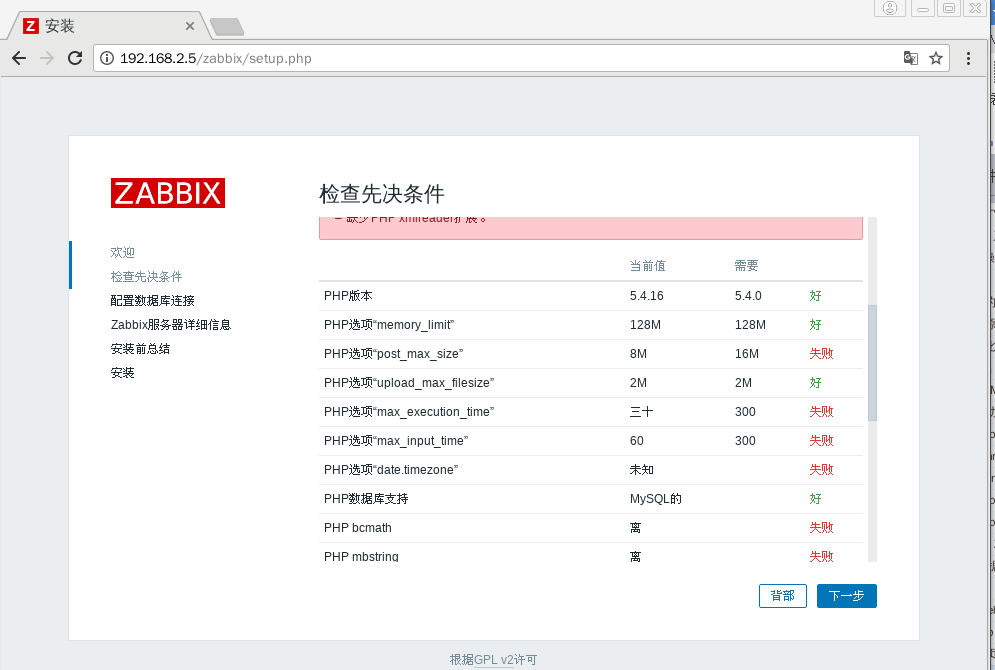
### 1欢迎页面



点击下一步进入下一步

### 2根据检查的报警提示,修改系统环境





101 yum -y install php-mbstring-5.4.16-42.el7.x86\_64.rpm

103 yum -y install php-bcmath-5.4.16-42.el7.x86\_64.rpm

105 yum -y install php-gd.x86\_64 php-xml php-ldap #支持图片的软件包

vim /etc/php.ini

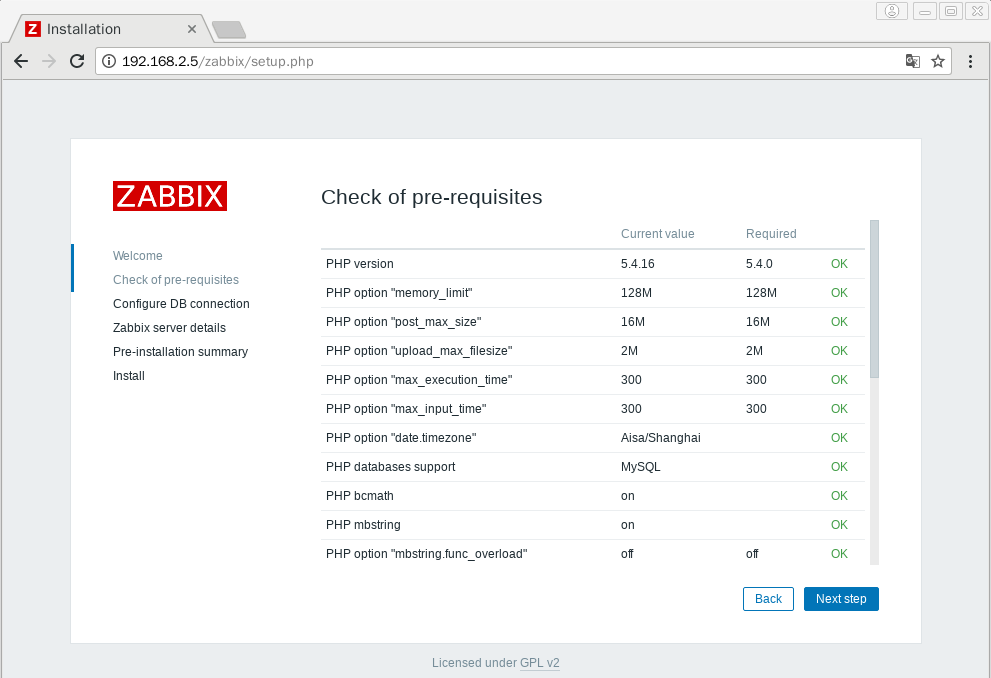
672 post\_max\_size = 16M

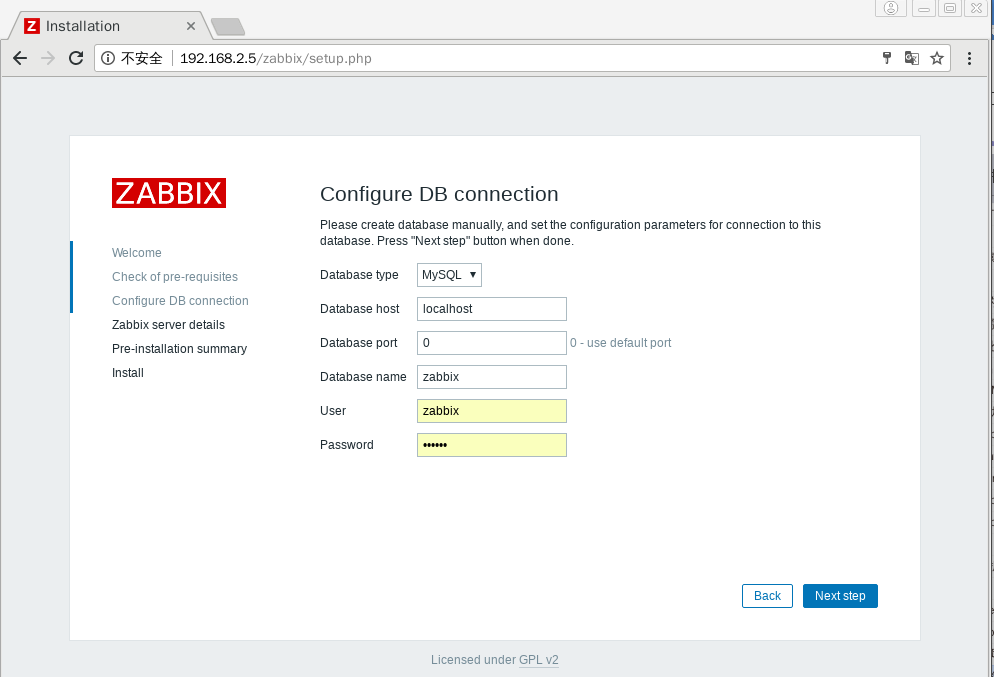
384 max\_execution\_time = 300

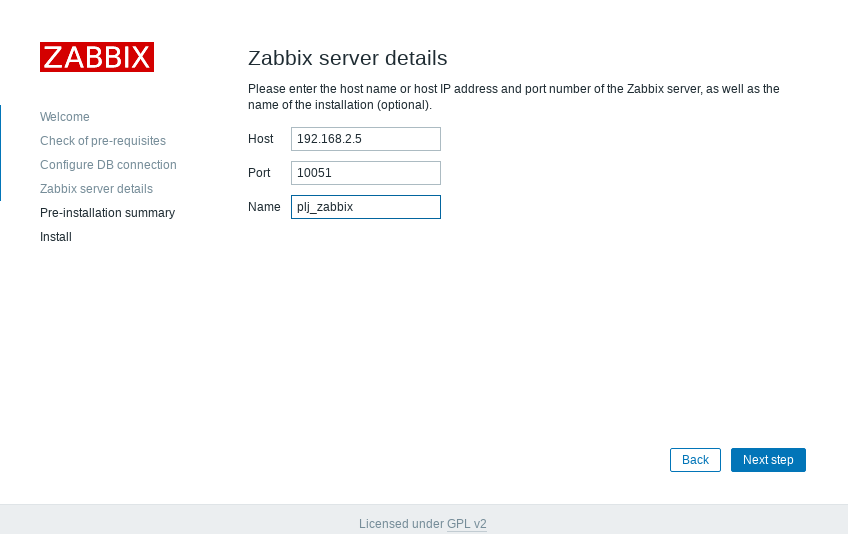
394 max\_input\_time = 300

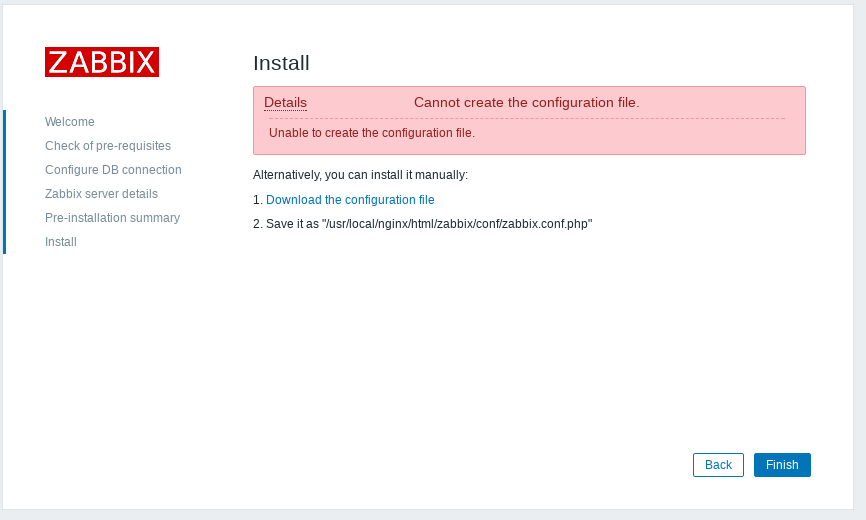
878 date.timezone = Asia/Shanghai

107 systemctl restart php-fpm





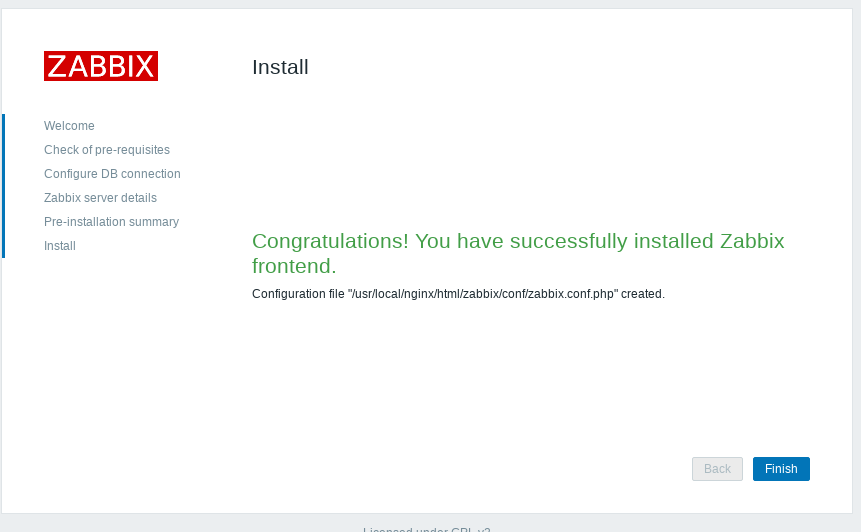


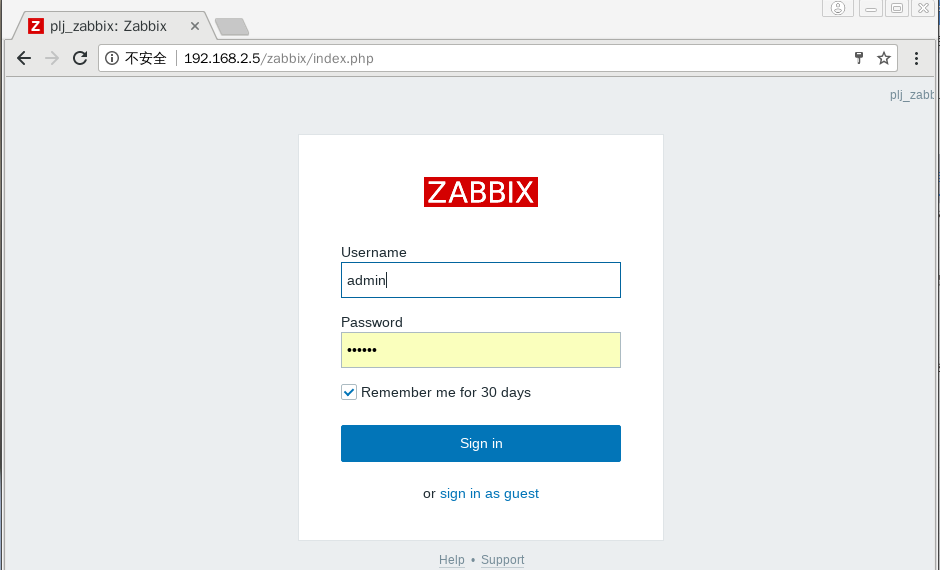


此处报错,是因为/usr/local/nginx/html/zabbix/conf/ 无权限

[root@zabbixserver lnmp\_soft]# chmod o+w /usr/local/nginx/html/zabbix/conf/

上一步back再finish





用户名改为:admin 密码:zabbix

10启动zabbix-server服务

1修改主配置文件

[root@zabbixserver ~]# vim /usr/local/zabbix/etc/zabbix\_server.conf

38 LogFile=/tmp/zabbix\_server.log

85 DBHost=localhost

95 DBName=zabbix

111 DBUser=zabbix

119 DBPassword=zabbix

2创建服务启动脚本

源码目录有模板

[root@zabbixserver ~]# ls /root/zabbix-3.4.4/misc/init.d/fedora/core/zabbix\_server

/root/zabbix-3.4.4/misc/init.d/fedora/core/zabbix\_server

[root@zabbixserver ~]# chmod +x /etc/init.d/zabbix\_server

[root@zabbixserver ~]# sed -i '22s;/usr/local;/usr/local/zabbix;' /etc/init.d/zabbix\_server

[root@zabbixserver ~]# sed -n '22p' /etc/init.d/zabbix\_server

### 3启动并查看服务

[root@zabbixserver ~]# useradd zabbix

[root@zabbixserver ~]# /etc/init.d/zabbix\_server start

Reloading systemd: [ 确定 ]

Starting zabbix\_server (via systemctl): [ 确定 ]

[root@zabbixserver ~]# netstat -utnlp |grep :10051

tcp 0 0 0.0.0.0:10051 0.0.0.0:\* LISTEN 20234/zabbix\_server

[root@zabbixserver ~]# /etc/init.d/zabbix\_server status

### 4排错

启动之后一定要查看得到10051端口

1 /etc/init.d/zabbix\_server status 查看状态

2 查看日志 38 LogFile=/tmp/zabbix\_server.log

3 最开始编译安装未指定路径的

//--prefix=/usr/local/zabbix #如果未指定路径,默认装在/user/local/etc/zabbix

这样/user/local/etc/下有个zabbix脚本运行的都是这个zabbix

## 10网页调试Zabbix

1修改语言为中文

2修改密码

3查看可以使用的模板



4查看监控了哪些主机

## 11监控本机

### 1运行zabbix\_agentd服务

在编译安装时就以及安装了zabbix\_agentd服务 --enable-agent

**修改zabbix\_agentd服务的主配置文件**

[root@zabbixserver ~]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf

93 Server=127.0.0.1 #谁可以监控本机(被动监控)

134 ServerActive=127.0.0.1:10051 #谁可以监控本机(主动监控)

145 Hostname=zabbixserver #被监控端自己的主机名

69 EnableRemoteCommands=1 #监控异常后,是否允许服务器远程过来执行命令

280 UnsafeUserParameters=1 #是否允许自定定义key监控

30 LogFile=/tmp/zabbix\_agentd.log #日志位置

**启动zabbix\_agentd服务**

[root@zabbixserver ~]# ls /root/zabbix-3.4.4/misc/init.d/fedora/core

zabbix\_agentd zabbix\_server #客户端启动脚本

[root@zabbixserver ~]# cp /root/zabbix-3.4.4/misc/init.d/fedora/core/zabbix\_agentd /etc/init.d/

[root@zabbixserver ~]# chmod o+w /etc/init.d/zabbix\_agentd

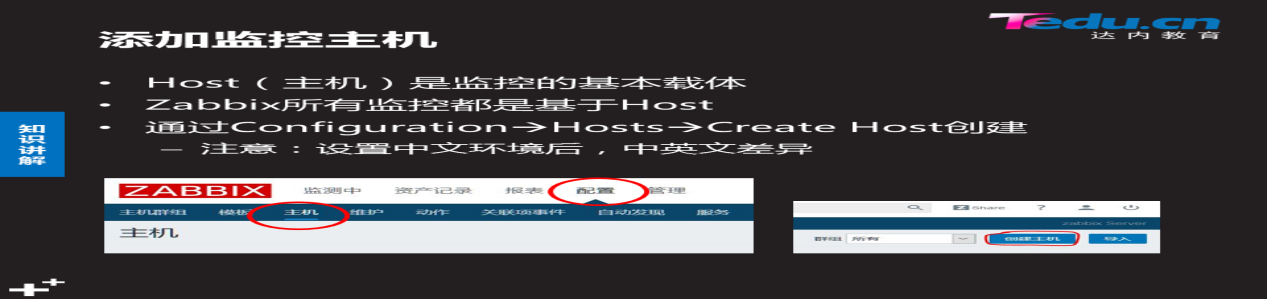
[root@zabbixserver ~]# sed -i '22s;/usr/local;/usr/local/zabbix;' /etc/init.d/zabbix\_agentd

[root@zabbixserver ~]# netstat -unltp |grep 10050

tcp 0 0 0.0.0.0:10050 0.0.0.0:\* LISTEN 21778/zabbix\_agentd

### 2登录管理页面做配置

**添加监控主机并选择监控目标**



**查看监控数据**

## **12监控远端主机**

### 1客户端与运行zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# yum -y install gcc pcre-devel

[root@zabbixweb1 zabbix-3.4.4]# ./configure --prefix=/usr/local/zabbix --enable-agent

[root@zabbixweb1 zabbix-3.4.4]# make install

[root@zabbixweb1 zabbix-3.4.4]# ls /usr/local/zabbix/

bin etc lib sbin share

### 2修改配置文件

[root@zabbixweb1 zabbix-3.4.4]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf

30 LogFile=/tmp/zabbix\_agentd.log

93 Server=127.0.0.1,192.168.2.5

134 ServerActive=192.168.2.5:10051

[root@zabbixweb1 zabbix-3.4.4]# cp misc/init.d/fedora/core/zabbix\_agentd /etc/init.d/

[root@zabbixweb1 zabbix-3.4.4]# chmod o+w /etc/init.d/zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# vim /etc/init.d/zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# useradd zabbix

[root@zabbixweb1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd start

Reloading systemd: [ 确定 ]

Starting zabbix\_agentd (via systemctl): [ 确定 ]

[root@web3 zabbix-3.4.4]# sed -i '22s;/usr/local;/usr/local/zabbix;' /etc/init.d/zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# netstat -untlp |grep :10050

tcp 0 0 0.0.0.0:10050 0.0.0.0:\* LISTEN 17842/zabbix\_agentd

### 3监控服务器配置

192.168.2.5登录页面,添加监控主机.选择监控目标

## 13自定义监控项

### 1客户端配置

启用自定义监控项功能

[root@zabbixweb1 ~]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf

265 Include=/usr/local/zabbix/etc/zabbix\_agentd.conf.d/\*.conf

#自定义的监控项明文件目录,以及命名

280 UnsafeUserParameters=1 #去掉注释改为1

### 2创建自定义规则项

[root@zabbixweb1 ~]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf.d/a1.conf

UserParameter=get\_users,wc -l /etc/passwd |awk '{print $1}'

#固定格式,统计有多少系统用户名

[root@zabbixweb1 ~]# /etc/init.d/zabbix\_agentd stop

Stopping zabbix\_agentd (via systemctl): [ 确定 ]

[root@zabbixweb1 ~]# /etc/init.d/zabbix\_agentd start

Starting zabbix\_agentd (via systemctl): [ 确定 ]

[root@zabbixweb1 zabbix-3.4.4]# netstat -untlp |grep :10050

tcp 0 0 0.0.0.0:10050 0.0.0.0:\* LISTEN 17842/zabbix\_agentd

#确保10050端口正常否者,服务不正常需要检查

### 3测试自定义规则项

[root@zabbixweb1 ~]# /usr/local/zabbix/bin/zabbix\_get --help

[root@zabbixweb1 ~]# /usr/local/zabbix/bin/zabbix\_get -s 127.0.0.1 -p 10050 -k get\_users

42

### 4网站上配置

1主机群组: 组名qunzu

2新建模板:

模板名称:muban

可见的名称:muban.name (其他各项中被选看得到的名称)

上一步创建了,群组:qunzu,这里直接在右边选择qunzu,

如一步没创建群组,也可新建群组



3创建应用集:名称yinyongji #(模板里面)

4创建监控项: #(模板里面)

名称jiankongxiang

键值 get\_users #自定义规则里面的UserParameter=get\_users,.....

应用集选择:选择上一步创建的yinyongji

5创建主机

主机名称:web1.name

可见的名称:web1

群组选择:选择上一步创建的组名为:qunzu

进入模板界面选择链接指示器:右上角点击上面定义的名为qunzu的群组,选择弹出新 建的名为muban.name的模板,点击”添加”,再点击更新,最后如下图



6查看数据,如下图则成功



# 案例

案例1：常用系统监控命令

案例2：部署Zabbix监控平台

案例3：配置及使用Zabbix监控系统

案例4：自定义Zabbix监控项目

1 案例1：常用系统监控命令

1.1 问题

本案例要求熟悉查看Linux系统状态的常用命令，为进一步执行具体的监控任务做准备：

查看内存信息

查看交换分区信息

查看磁盘信息

查看CPU信息

查看网卡信息

查看端口信息

查看网络连接信息

1.2 方案

一般企业做监控的目的：实时报告系统状态，提前发现系统的问题。

监控的资源可以分为：共有数据（HTTP、FTP等）和私有数据（CPU、内存、进程数等）。

监控软件可以使用：系统自带的命令、Cacti监控系统、Nagios监控系统、Zabbix监控系统。

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用命令查看计算机状态数据

1）查看内存与交换分区信息

[root@proxy ~]# free //查看内存信息

total used free shared buff/cache available

Mem: 16166888 8017696 720016 106504 7429176 7731740

Swap: 4194300 218268 3976032

[root@proxy ~]# free | awk '/Mem/{print $4}' //查看剩余内存容量

720928

[root@proxy ~]# swapon -s //查看交换分区信息

文件名 类型 大小 已用 权限

/dev/sda3 partition 4194300 218268 -1

步骤二：查看磁盘与CPU利用率

1）查看磁盘信息

[root@proxy ~]# df //查看所有磁盘的使用率

文件系统 1K-块 已用 可用 已用% 挂载点

/dev/sda2 476254208 116879624 335159084 26% /

/dev/sda1 198174 133897 49737 73% /boot

[root@proxy ~]# df | awk '/\/$/{print $5}' //查看根分区的利用率

2）查看CPU平均负载

[root@proxy ~]# uptime //查看CPU负载（1，5，15分钟）

23:54:12 up 38 days, 14:54, 9 users, load average: 0.00, 0.04, 0.05

[root@proxy ~]# uptime |awk '{print $NF}' //仅查看CPU的15分钟平均负载

0.05

步骤二：查看网卡信息、端口信息、网络连接信息

1）查看网卡信息

[root@proxy ~]# ifconfig eth0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.4.5 netmask 255.255.255.0 broadcast 172.25.0.255

inet6 fe80::5054:ff:fe00:b prefixlen 64 scopeid 0x20<link>

ether 52:54:00:00:00:0b txqueuelen 1000 (Ethernet)

RX packets 62429 bytes 10612049 (10.1 MiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 5674 bytes 4121143 (3.9 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@proxy ~]# ifconfig eth0 |awk '/inet /{print $2}' //查看IP地址信息

192.168.4.5

[root@proxy ~]# ifconfig eth0 |awk '/RX p/{print $5}' //网卡接受数据包流量

10625295

[root@proxy ~]# ifconfig eth0 |awk '/TX p/{print $5}' //网卡发送数据包流量

4130821

2）查看端口信息

[root@proxy ~]# ss -ntulp //查看本机监听的所有端口

//-n以数字显示端口号

//-t显示tcp连接

//-u显示udp连接

//-p显示监听端口对应的程序名称

3）查看网络连接信息

[root@proxy ~]# ss -antup //查看所有的网络连接信息

//-a查看所有连接状态信息

2 案例2：部署Zabbix监控平台

2.1 问题

本案例要求部署一台Zabbix监控服务器，一台被监控主机，为进一步执行具体的监控任务做准备：

安装LNMP环境

源码安装Zabbix

安装监控端主机，修改基本配置

初始化Zabbix监控Web页面

修改PHP配置文件，满足Zabbix需求

安装被监控端主机，修改基本配置

2.2 方案

使用1台RHEL7虚拟机，安装部署LNMP环境、Zabbix及相关的依赖包，配置数据库并对Zabbix监控平台进行初始化操作。使用2台被监控端，源码安装Zabbix Agent。完成Zabbix实验需要我们搭建一个实验环境，拓扑结构如表-1所示。

表-1 实验拓扑结构

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：部署监控服务器

1）安装LNMP环境

Zabbix监控管理控制台需要通过Web页面展示出来，并且还需要使用MySQL来存储数据，因此需要先为Zabbix准备基础LNMP环境。

[root@zabbixserver ~]# yum -y install gcc pcre-devel openssl-devel

[root@zabbixserver ~]# tar -xf nginx-1.12.2.tar.gz

[root@zabbixserver ~]# cd nginx-1.12.2

[root@zabbixserver nginx-1.12.2]# ./configure --with-http\_ssl\_module

[root@zabbixserver nginx-1.12.2]# make && make install

[root@zabbixserver ~]# yum -y install php php-mysql \

> mariadb mariadb-devel mariadb-server

[root@zabbixserver ~]# yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

//注意，php-fpm这个软件包在lnmp\_soft/目录下

2）修改Nginx配置文件

配置Nginx支持PHP动态网站，因为有大量PHP脚本需要执行，因此还需要开启Nginx的各种fastcgi缓存，加速PHP脚本的执行速度。

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

… …

fastcgi\_buffers 8 16k; //缓存php生成的页面内容，8个16k

fastcgi\_buffer\_size 32k; //缓存php生产的头部信息

fastcgi\_connect\_timeout 300; //连接PHP的超时时间

fastcgi\_send\_timeout 300; //发送请求的超时时间

fastcgi\_read\_timeout 300; //读取请求的超时时间

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

… …

3）启动服务

启动Nginx、PHP-FPM、MariaDB服务，关闭SELinux与防火墙。

[root@zabbixserver ~]# systemctl start mariadb

[root@zabbixserver ~]# systemctl start php-fpm

[root@zabbixserver ~]# ln -s /usr/local/nginx/sbin/nginx /sbin/nginx

[root@zabbixserver ~]# nginx

[root@zabbixserver ~]# firewall-cmd --set-default-zone=trusted

[root@zabbixserver ~]# setenforce 0

4）客户端测试LNMP环境

服务器创建PHP测试页面，浏览器访问页面测试网页连通性。

[root@zabbixserver ~]# cat /usr/local/nginx/html/test.php

<?php

$i=33;

echo $i;

?>

[root@zabbixserver ~]# curl http://192.168.2.5/test.php

步骤二：部署监控服务器Zabbix Server

1）源码安装Zabbix Server

多数源码包都是需要依赖包的，zabbix也一样，源码编译前需要先安装相关依赖包。

[root@zabbixserver lnmp\_soft]# yum -y install net-snmp-devel \

> curl-devel

//安装相关依赖包

[root@zabbixserver lnmp\_soft]# yum -y install \

> libevent-devel-2.0.21-4.el7.x86\_64.rpm

//注意libevent-devel这个软件包在lnmp\_soft目录下有提供

[root@zabbixserver lnmp\_soft]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixserver lnmp\_soft]# cd zabbix-3.4.4/

[root@zabbixserver zabbix-3.4.4]# ./configure --enable-server \

> --enable-proxy --enable-agent --with-mysql=/usr/bin/mysql\_config \

> --with-net-snmp --with-libcurl

// --enable-server安装部署zabbix服务器端软件

// --enable-agent安装部署zabbix被监控端软件

// --enable-proxy安装部署zabbix代理相关软件

// --with-mysql配置mysql\_config路径

// --with-net-snmp允许zabbix通过snmp协议监控其他设备

// --with-libcurl安装相关curl库文件，这样zabbix就可以通过curl连接http等服务，测试被监控主机服务的状态

[root@zabbixserver zabbix-3.4.4]# make && make install

2）初始化Zabbix

创建数据库，上线Zabbix的Web页面

[root@zabbixserver ~]# mysql

mysql> create database zabbix character set utf8;

//创建数据库，支持中文字符集

mysql> grant all on zabbix.\* to zabbix@'localhost' identified by 'zabbix';

//创建可以访问数据库的账户与密码

[root@zabbixserver ~]# cd lnmp\_soft/zabbix-3.4.4/database/mysql/

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < schema.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < images.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < data.sql

//刚刚创建是空数据库，zabbix源码包目录下，有提前准备好的数据

//使用mysql导入这些数据即可（注意导入顺序）

上线Zabbix的Web页面

[root@zabbixserver ~]# cd lnmp\_soft/zabbix-3.4.4/frontends/php/

[root@zabbixserver php]# cp -r \* /usr/local/nginx/html/

[root@zabbixserver php]# chmod -R 777 /usr/local/nginx/html/\*

修改Zabbix\_server配置文件，设置数据库相关参数，启动Zabbix\_server服务

[root@zabbixserver ~]# vim /usr/local/etc/zabbix\_server.conf

DBHost=localhost

//数据库主机，默认该行被注释

DBName=zabbix

//设置数据库名称

DBUser=zabbix

//设置数据库账户

DBPassword=zabbix

//设置数据库密码，默认该行被注释

LogFile=/tmp/zabbix\_server.log

//设置日志，仅查看以下即可

[root@zabbixserver ~]# useradd -s /sbin/nologin zabbix

//不创建用户无法启动服务

[root@zabbixserver ~]# zabbix\_server //启动服务

[root@zabbixserver ~]# ss -ntulp |grep zabbix\_server //确认连接状态，端口10051

tcp LISTEN 0 128 \*:10051 \*:\* users:(("zabbix\_server",pid=23275,fd=4),("zabbix\_server",pid=23274,fd=4)

提示：如果是因为配置文件不对，导致服务无法启动时，不要重复执行zabbix\_server，

一定要先使用killall zabbix\_server关闭服务后，再重新启动一次。

修改Zabbix\_agent配置文件，启动Zabbix\_agent服务

[root@zabbixserver ~]# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //允许哪些主机监控本机

ServerActive=127.0.0.1,192.168.2.5 //允许哪些主机通过主动模式监控本机

Hostname=zabbix\_server //设置本机主机名

LogFile=/tmp/zabbix\_server.log //设置日志文件

UnsafeUserParameters=1 //是否允许自定义key

[root@zabbixserver ~]# zabbix\_agentd //启动监控agent

[root@zabbixserver ~]# ss -ntulp |grep zabbix\_agentd //查看端口信息为10050

tcp LISTEN 0 128 \*:10050 \*:\* users:(("zabbix\_agentd",pid=23505,fd=4),("zabbix\_agentd",pid=23504,fd=4)

提示：如果是因为配置文件不对，导致服务无法启动时，不要重复执行zabbix\_agentd，

一定要先使用killall zabbix\_agentd关闭服务后，再重新启动一次。

浏览器访问Zabbix\_server服务器的Web页面

[root@zabbixserver ~]# firefox http://192.168.2.5/index.php

//第一次访问，初始化PHP页面会检查计算机环境是否满足要求，如果不满足会给出修改建议

//默认会提示PHP的配置不满足环境要求，需要修改PHP配置文件

根据错误提示，修改PHP配置文件，满足Zabbix\_server的Web环境要求

php-bcmath和php-mbstring都在lnmp\_soft目录下有提供软件包。

[root@zabbixserver ~]# yum -y install php-gd php-xml

[root@zabbixserver ~]# yum install php-bcmath-5.4.16-42.el7.x86\_64.rpm

[root@zabbixserver ~]# yum install php-mbstring-5.4.16-42.el7.x86\_64.rpm

[root@zabbixserver ~]# vim /etc/php.ini

date.timezone = Asia/Shanghai //设置时区

max\_execution\_time = 300 //最大执行时间，秒

post\_max\_size = 32M //POST数据最大容量

max\_input\_time = 300 //服务器接收数据的时间限制

memory\_limit = 128M //内存容量限制

[root@zabbixserver ~]# systemctl restart php-fpm

修改完PHP配置文件后，再次使用浏览器访问服务器，则会提示如图-1和图-2所示的提示信息。

图-1

图-2

注意：这里有一个PHP LDAP是warning状态是没有问题的！

在初始化数据库页面，填写数据库相关参数，如图-3所示。

图-3

在登陆页面，使用用户(admin)和密码(zabbix)登陆，登陆后设置语言环境为中文，如图-4和图-5所示。

图-4

图-5

步骤三：部署被监控主机Zabbix Agent

1）源码安装Zabbix agent软件

在2.100和2.200做相同操作（以zabbixclient\_web1为例）。

[root@zabbixclient\_web1 ~]# useradd -s /sbin/nologin zabbix

[root@zabbixclient\_web1 ~]# yum -y install gcc pcre-devel

[root@zabbixclient\_web1 ~]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixclient\_web1 ~]# cd zabbix-3.4.4/

[root@zabbixclient\_web1 zabbix-3.4.4]# ./configure --enable-agent

[root@zabbixclient\_web1 zabbix-3.4.4]# make && make install

2）修改agent配置文件，启动Agent

[root@zabbixclient\_web1 ~]# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //谁可以监控本机（被动监控模式）

ServerActive=127.0.0.1,192.168.2.5 //谁可以监控本机（主动监控模式）

Hostname=zabbixclient\_web1 //被监控端自己的主机名

EnableRemoteCommands=1

//监控异常后，是否允许服务器远程过来执行命令，如重启某个服务

UnsafeUserParameters=1 //是否允许自定义key监控

[root@zabbixclient\_web1 ~]# zabbix\_agentd //启动agent服务

3）拷贝启动脚本（非必须操作，可选做），有启动脚本可以方便管理服务，启动与关闭服务。启动脚本位于zabbix源码目录下。

[root@zabbixclient\_web1 zabbix-3.4.4]# cd misc/init.d/fedora/core

[root@zabbixclient\_web1 zabbix-3.4.4]# cp zabbix\_agentd /etc/init.d/

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd start

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd stop

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd status

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd restart

3 案例3：配置及使用Zabbix监控系统

3.1 问题

沿用练习一，使用Zabbix监控平台监控Linux服务器，实现以下目标：

监控CPU

监控内存

监控进程

监控网络流量

监控硬盘

3.2 方案

通过Zabbix监控平台，添加被监控zabbixclient\_web1主机（192.168.2.100）并链接监控模板即可，Zabbix默认模板就可以监控CPU、内存、进程、网络、磁盘等项目。

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：添加监控主机

主机是Zabbix监控的基础，Zabbix所有监控都是基于Host主机。

使用火狐浏览器登录http://192.168.2.5，通过Configuration（配置）-->Hosts（主机）-->Create Host（创建主机）添加被监控Linux主机，如图-7所示。

图-7

添加被监控主机时，需要根据提示输入被监控Linux主机的主机名称（最好与电脑的主机名一致，但也允许不一致）、主机组、IP地址等参数，具体参考图-8所示。

图-8

步骤二：为被监控主机添加监控模板

Zabbix通过监控模板来对监控对象实施具体的监控功能，根据模板来定义需要监控哪些数据，对于Linux服务器的监控，Zabbix已经内置了相关的模板（Template OS Linux），选择模板并链接到主机即可，如图-9所示。

图-9

步骤三：查看监控数据

查看监控数据，登录Zabbix Web控制台，点击Monitoring(监控中)—> Latest data(最新数据)，正过滤器中填写过滤条件，根据监控组和监控主机选择需要查看哪些监控数据，如图-10所示。

图-10

找到需要监控的数据后，可以点击后面的Graph查看监控图形，如图-11所示。

图-11

4 案例4：自定义Zabbix监控项目

4.1 问题

沿用练习二，使用Zabbix实现自定义监控，实现以下目标：监控Linux服务器系统账户的数量。

4.2 方案

需要使用Zabbix自定义key的方式实现自定义监控，参考如下操作步骤：

创建自定义key

创建监控项目

创建监控图形

将监控模板关联到主机

4.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：被监控主机创建自定义key（在192.168.2.100操作）

1）创建自定义key

自定义key语法格式为：UserParameter=自定义key名称,命令。

自定义的key文件一般存储在/usr/local/etc/zabbix\_agentd.conf.d/目录，这里还需要修改zabbix\_agentd.conf文件，允许自定义监控key，来读取该目录下的所有文件 。

[root@zabbixclient\_web1 ~]# vim /usr/local/etc/zabbix\_agentd.conf

Include=/usr/local/etc/zabbix\_agentd.conf.d/ //加载配置文件目录

[root@zabbixclient\_web1 ~]# cd /usr/local/etc/zabbix\_agentd.conf.d/

[root@zabbixclient\_web1 zabbix\_agentd.conf.d]# vim count.line.passwd

UserParameter=count.line.passwd,wc -l /etc/passwd | awk ' {print $1} '

////自定义key语法格式:

//UserParameter=自定义key名称,命令

2）测试自定义key是否正常工作

[root@zabbixclient\_web1 ~]# killall zabbix\_agentd

[root@zabbixclient\_web1 ~]# zabbix\_agentd //重启agent服务

[root@zabbixclient\_web1 ~]# zabbix\_get -s 127.0.0.1 -k count.line.passwd

21

注意：如zabbix\_get命令执行错误，提示Check access restrictions in Zabbix agent configuration，则需要检查agent配置文件是否正确：

[root@zabbixclient\_web1 ~]# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5

ServerActive=127.0.0.1,192.168.2.5

步骤二：创建监控模板

模板、应用集与监控项目的关系图，参考图-12所示

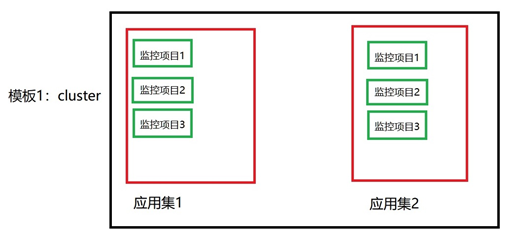


图-12

1）添加监控模板

登录Zabbix Web监控控制台，通过Configuration(配置)-->Template(模板)-->Create template(创建模板)，填写模板名称，新建模板群组，如图-13所示。



图-13

创建模板后，默认模板中没有任何应用、项目、触发器、图形等，如图-14所示。

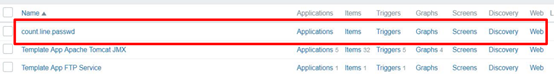


图-14

2）创建应用

创建完成模板后，默认模板中没有任何应用、项目、触发器、图形等资源。这里需要点击模板后面的Application（应用集）链接打开创建应用的页面，如图-15所示。

image014

图-15

点击Application（应用集）后，会刷新出图-16所示页面，在该页面中点击Create application（创建应用集）按钮。

image015

图-16

设置应用名称如图-17所示。



图-17

3）创建监控项目item（监控项）

与创建应用一样，在模板中还需要创建监控项目，如图-18所示，并在刷新出的新页面中选择Create items（创建监控项）创建项目，如图-19所示。

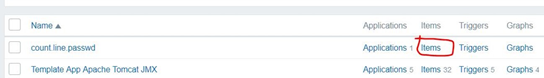


图-18

image018

图-19

接下来，还需要给项目设置名称及对应的自定义key，如图-20所示。

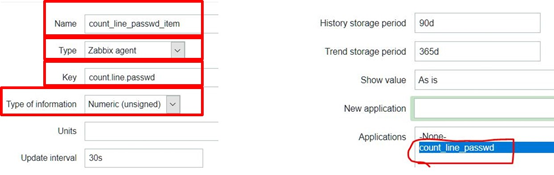


图-20

4）创建图形

为了后期可以通过图形的方式展示监控数据，还需要在模板中创建图形，设置方法与前面的步骤一致，在监控模板后面点击Graph（图形）即可创建图形，设置监控图形基于什么监控数据，如图-21所示。



图-21

5）将模板链接到被监控主机

将完整的监控模板制作完成后，就可以将模板链接到主机实现监控功能了。首先找到被监控主机Configuration（配置）-->Hosts（主机），如图-22所示。



图-22

点击需要的被监控主机链接，打开监控主机设置页面，在Template（模板）页面中选择需要链接到该主机的模板，在此选择刚刚创建的模板count\_line.passwd添加即可，如图-23所示。

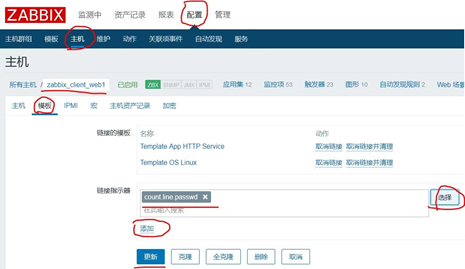


图-23

6）查看监控数据图形

点击Monitoring（监控中）-->Craphs（图形），根据需要选择条件，查看监控图形，如图-24和图-25所示。



图-24



图-25

# NSD SECURITY DAY05

## 查看Linux系统状态的常用命令

查看内存信息:free free| awk ’/Mem/{print $4} 查看剩余内存容量’

查看交换分区信息:swapon -s

查看磁盘信息:df df | awk '/\/$/{print $5}' //查看根分区的利用率

查看磁盘状态信息:iostat

查看CPU信息:uptime //查看CPU负载（1，5，15分钟）

uptime |awk '{print $NF}' //仅查看CPU的15分钟平均负载

查看网卡信息

ifconfig eth0

ifconfig eth0 |awk '/inet /{print $2}' //查看IP地址信息

ifconfig eth0 |awk '/RX p/{print $5}' //网卡接受数据包流量

ifconfig eth0 |awk '/TX p/{print $5}' //网卡发送数据包流量

查看端口信息

ss -ntulp //查看本机监听的所有端口

查看网络连接信息

ss -antup //查看所有的网络连接信息

查看路由追踪: traceroute www.baidu.com

查看进程: ps

-A ：所有的进程均显示出来，与 -e 具有同样的效用；

-a ： 显示现行终端机下的所有进程，包括其他用户的进程；

-u ：以用户为主的进程状态 ；

x ：通常与 a 这个参数一起使用，可列出较完整信息。

查看系统运行时间:uptime

## awk基本操作方法

格式一: 前置命令 |　awk [选项] '[条件]{指令}' 文件

格式二：awk [选项] '[条件]{指令}' 文件

选项 -F 可指定分隔符,不指定，默认为空格（包括tab）

print：是最常用的编辑指令，若有多条编辑指令，可用分号分隔。

**awk 常用内置变量:**

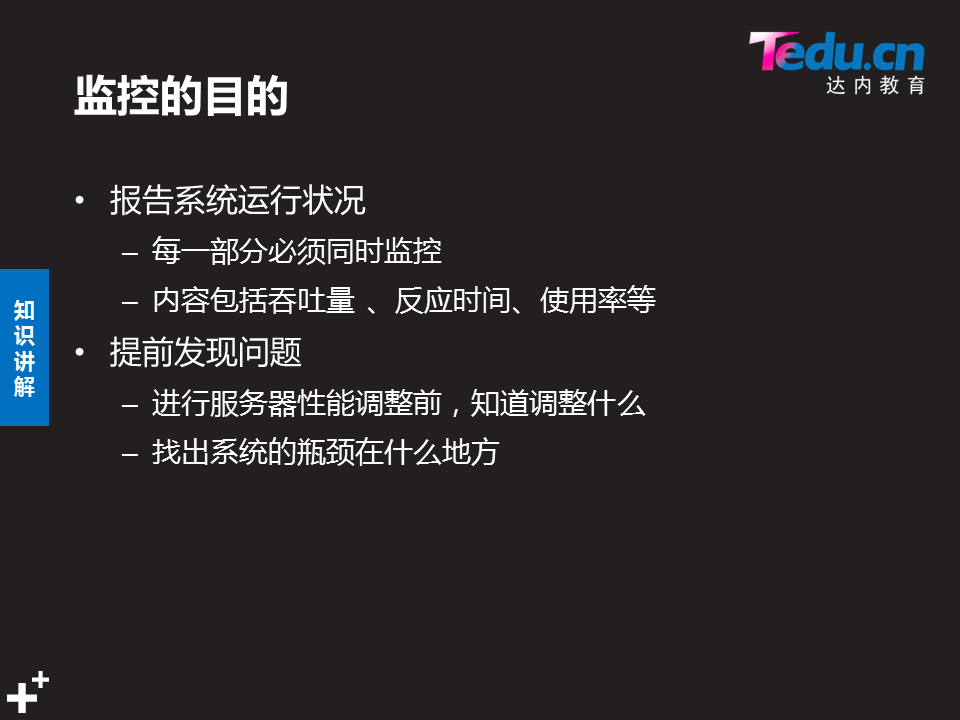
$0 文本当前行的全部内容 $1 文本的第1列

$2 文件的第2列 $3 文件的第3列，依此类推

NR 文件当前行的行号 NF 文件当前行的列数（有几列）

# 1 监控概述

## 监控的目的

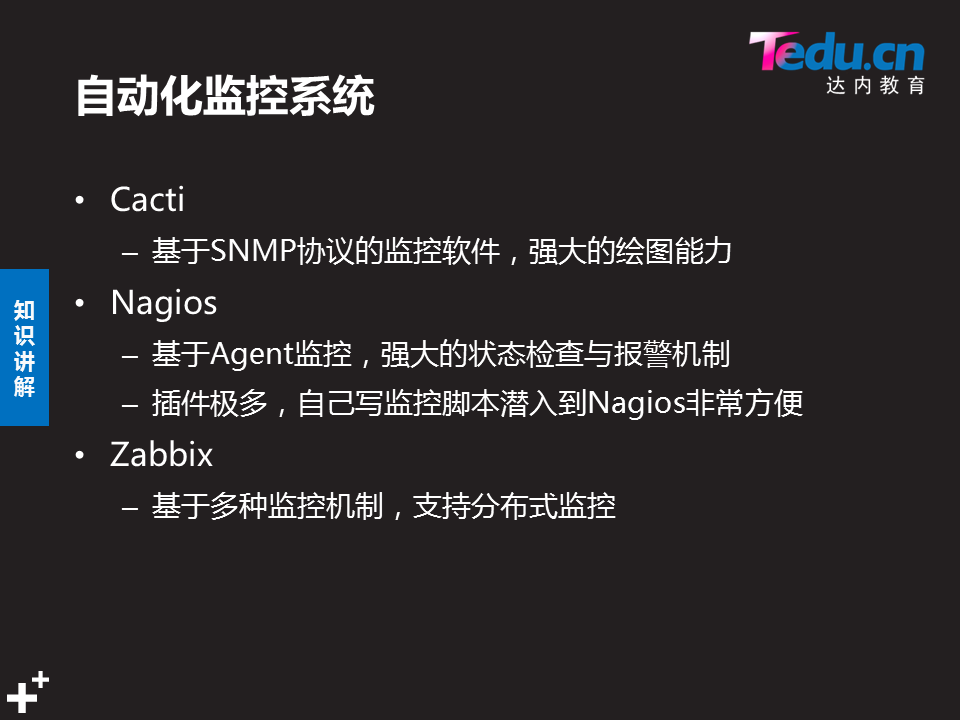


## 监控的资源类别

公开数据:web ftp ssh 数据库等应用服务 tcp或udp 端口

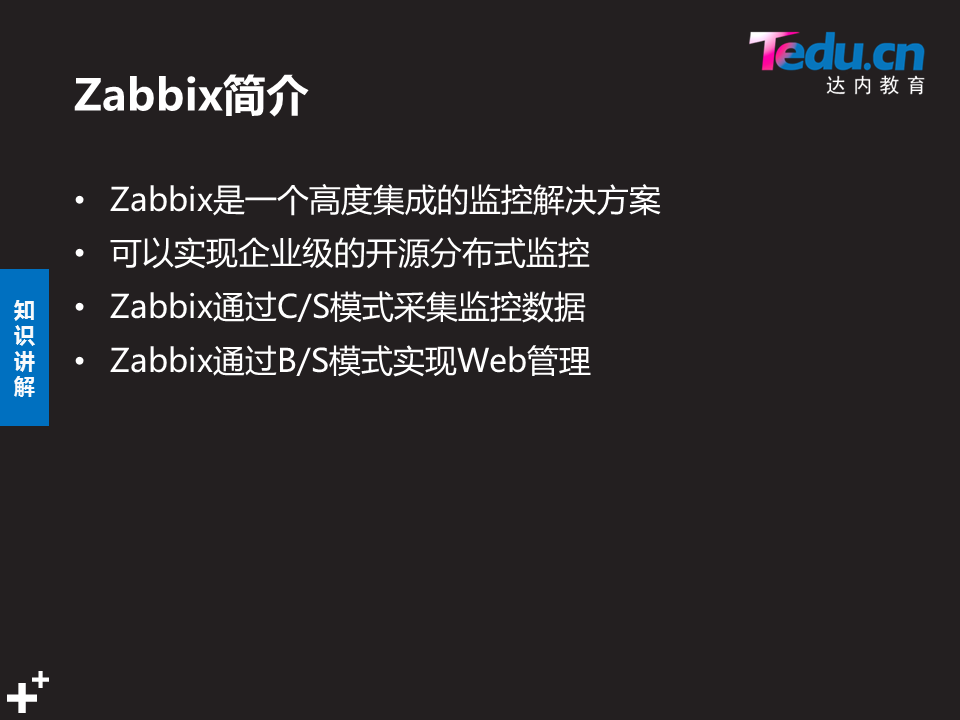
私有数据:cpu 内存 磁盘 网卡 用户 进程等运行信息

## 常见自动化监控系统



当前互联网监控软件大约有300多中,

# 2 Zabbix基础



## 1 安装LNMP环境

15 yum -y install gcc pcre-devel openssl-devel

18 tar -xf lnmp\_soft.tar.gz #需要单独传

24 tar -xf nginx-1.12.2.tar.gz #需要单独传

26 cd nginx-1.12.2/

28 ./configure --with-http\_ssl\_module

29 make && make install

30 ls /usr/local/nginx/

33 cd lnmp\_soft/

35 ls php-fpm-5.4.16-42.el7.x86\_64.rpm

36 yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

38 yum -y install php php-mysql

40 yum -y install mariadb mariadb-devel mariadb-server

## 2 配置启动nginx

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

include fastcgi.conf;

}

[root@zabbixserver ~]# /usr/local/nginx/sbin/nginx

[root@zabbixserver ~]# netstat -unltp | grep 80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 6449/nginx: master

## 3 启动php-fpm服务

[root@zabbixserver ~]# systemctl start php-fpm

[root@zabbixserver ~]# netstat -unltp | grep 9000

tcp 0 0 127.0.0.1:9000 0.0.0.0:\* LISTEN 6507/php-fpm: maste

## 4 启动mariadb

[root@zabbixserver ~]# systemctl start mariadb

[root@zabbixserver ~]# netstat -unltp | grep 3306

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 6764/mysqld

[root@zabbixserver ~]# vim /usr/local/nginx/html/test.php

[root@zabbixserver ~]# curl http://localhost/test.php

## 5 配置nginx,

满足Zbbix服务的运行需求

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

http{

… …

fastcgi\_buffers 8 16k; //缓存php生成的页面内容，8个16k

fastcgi\_buffer\_size 32k; //缓存php生产的头部信息

fastcgi\_connect\_timeout 300; //连接PHP的超时时间

fastcgi\_send\_timeout 300; //发送请求的超时时间

fastcgi\_read\_timeout 300; //读取请求的超时时间

[root@zabbixserver ~]# /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

[root@zabbixserver ~]# /usr/local/nginx/sbin/nginx -s reload

## 6 安装Zabbix依赖包

[root@zabbixserver lnmp\_soft]#yum -y install net-snmp-devel curl-devel

[root@zabbixserver lnmp\_soft]# yum -y install libevent-devel-2.0.21-4.el7.x86\_64.rpm

#注意libevent-devel这个软件包在lnmp\_soft目录下有提供

## 7安装Zabbix

[root@zabbixserver ~]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixserver ~]# cd zabbix-3.4.4/

[root@zabbixserver zabbix-3.4.4]# ls

[root@zabbixserver zabbix-3.4.4]# ./configure --prefix=/usr/local/zabbix --enable-server --enable-proxy \

> --enable-agent --with-mysql=/usr/bin/mysql\_config --with-net-snmp --with-libcurl

....

....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Now run 'make install' \* #提示直接 'make install'

\* \*

\* Thank you for using Zabbix! \*

\* <http://www.zabbix.com> \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//--prefix=/usr/local/zabbix #如果未指定路径,默认装在/user/local/etc/zabbix

// --enable-server安装部署zabbix服务器端软件

// --enable-agent安装部署zabbix被监控端软件

// --enable-proxy安装部署zabbix代理相关软件

// --with-mysql配置mysql\_config路径

// --with-net-snmp允许zabbix通过snmp协议监控其他设备

// --with-libcurl安装相关curl库文件，这样zabbix就可以通过curl连接http等服务，测试被监控主机服务的状态

[root@zabbixserver zabbix-3.4.4]# make install

[root@zabbixserver ~]# ls /usr/local/zabbix/ #安装成功后查看安装的目录里文件

bin etc lib sbin share

## 8初始化Zabbix

### 1创建数据库与授权用户

[root@zabbixserver ~]# systemctl start mariadb

[root@zabbixserver ~]# mysql

MariaDB [(none)]> create database zabbix character set utf8;

Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all on zabbix.\* to zabbix@"localhost" identified by "zabbix";

Query OK, 0 rows affected (0.00 sec)

### 2创建表

(软件提供了恢复文件,在源码目录里面)

[root@zabbixserver ~]# ls /root/zabbix-3.4.4/database/mysql/

data.sql images.sql schema.sql

[root@zabbixserver ~]# cd /root/zabbix-3.4.4/database/mysql/

[root@zabbixserver mysql]# ls

data.sql images.sql schema.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < schema.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < images.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < data.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

....

MariaDB [zabbix]>

### 3创建web页面

[root@zabbixserver mysql]# pwd

/root/zabbix-3.4.4/database/mysql

[root@zabbixserver mysql]# ls /root/zabbix-3.4.4/frontends/

php

[root@zabbixserver mysql]# cd /root/zabbix-3.4.4/frontends/

[root@zabbixserver frontends]# cp -r php /usr/local/nginx/html/zabbix

[root@zabbixserver frontends]# ls /usr/local/nginx/html

50x.html index.html test.php zabbix

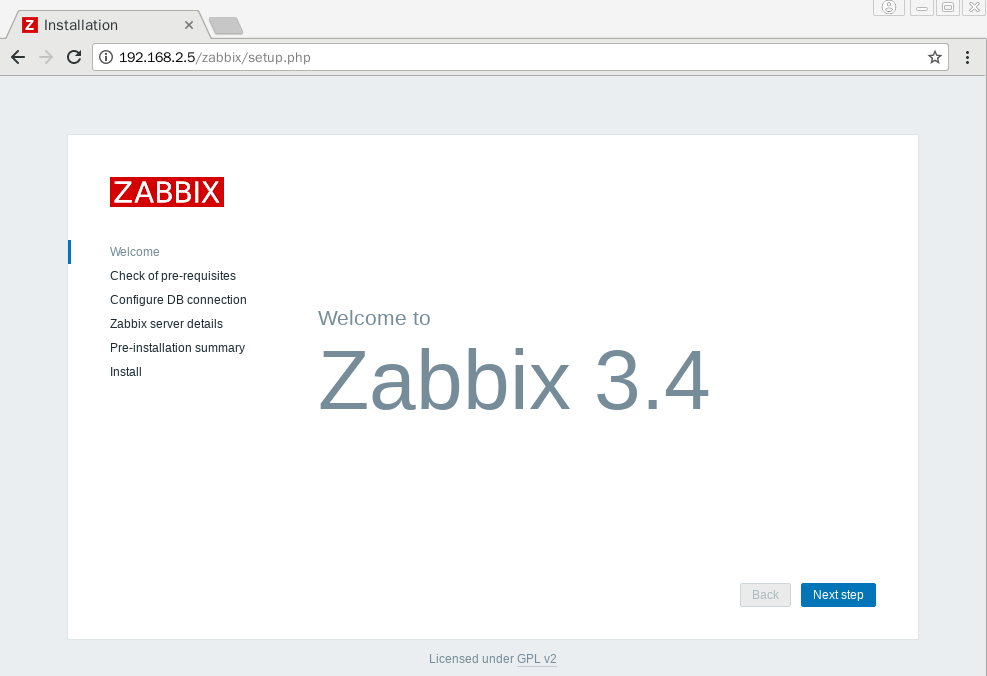
[root@zabbixserver frontends]# ls /usr/local/nginx/html/zabbix/index.php

/usr/local/nginx/html/zabbix/index.php

## 9访问web页面做初始化配置

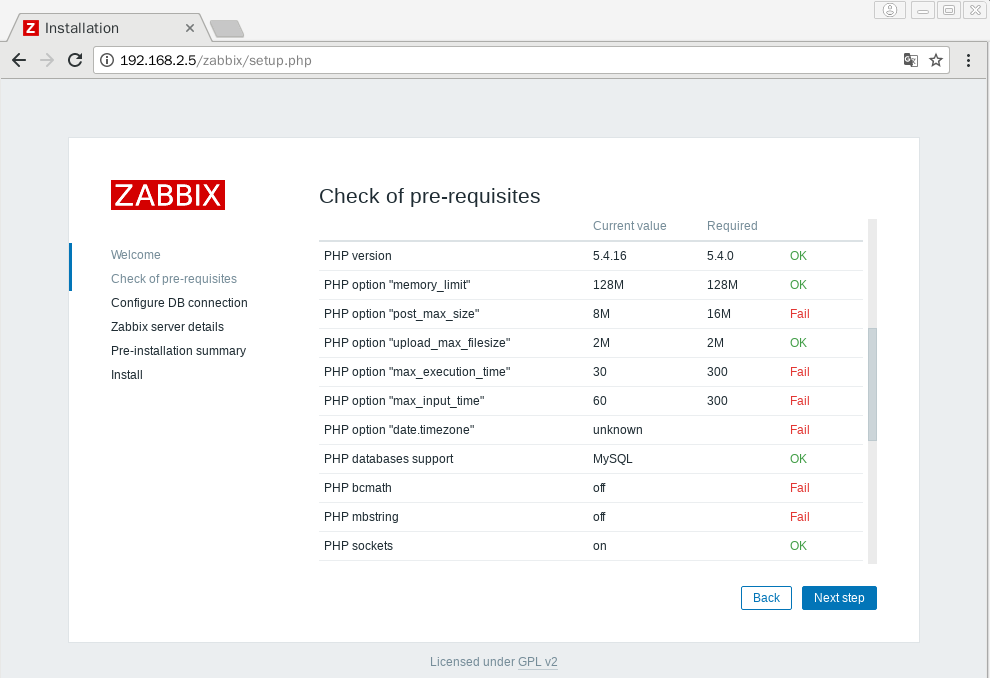
: http://192.168.2.5/zabbix/index.php 如下图

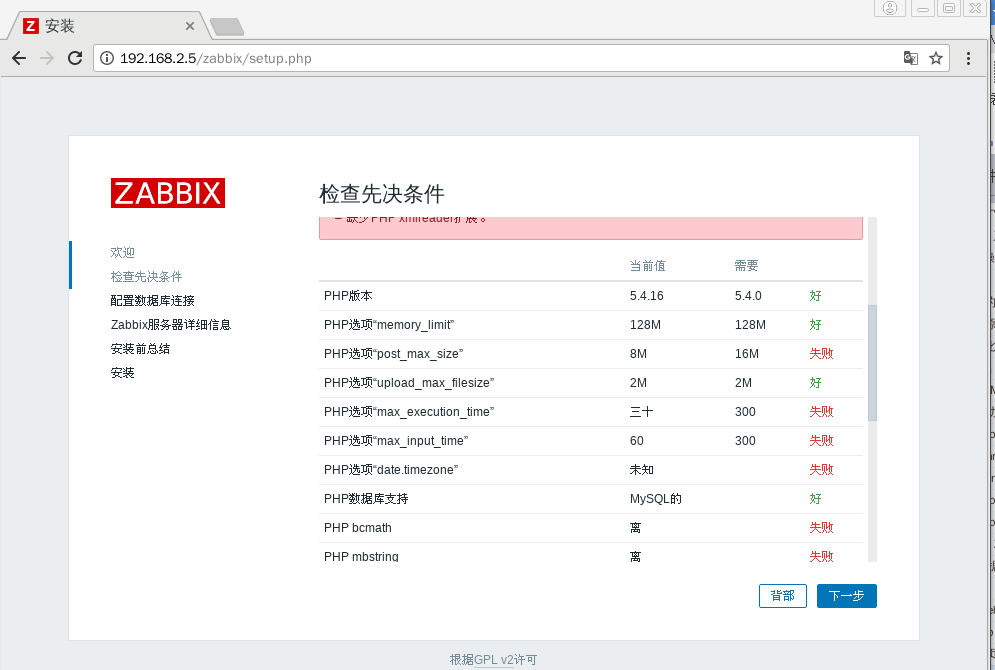
### 1欢迎页面



点击下一步进入下一步

### 2根据检查的报警提示,修改系统环境





**以下安装包在:lnmp\_soft目录下有提供**

101 yum -y install php-mbstring-5.4.16-42.el7.x86\_64.rpm

103 yum -y install php-bcmath-5.4.16-42.el7.x86\_64.rpm

**以下安装包用yum源安装:**

105 yum -y install php-gd.x86\_64 php-xml php-ldap #支持图片的软件包

vim /etc/php.ini

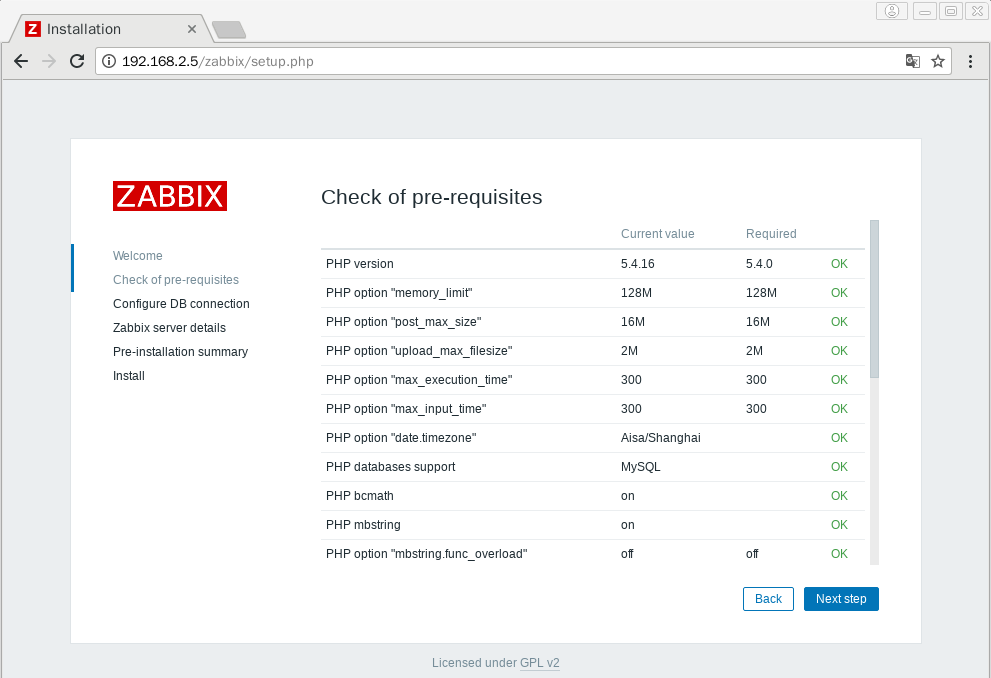
672 post\_max\_size = 16M

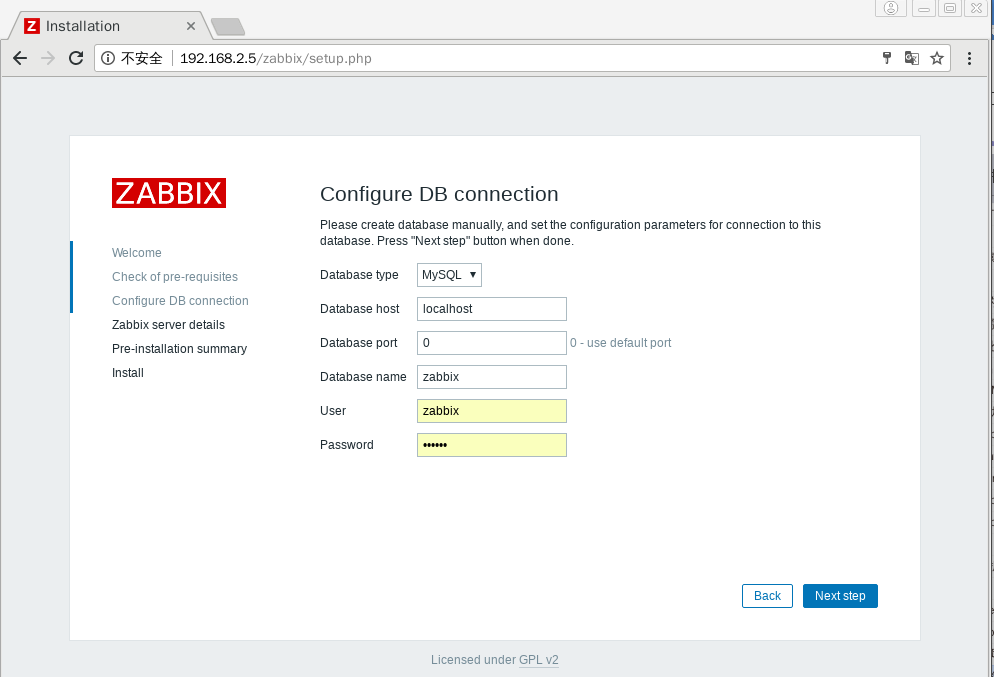
384 max\_execution\_time = 300

394 max\_input\_time = 300

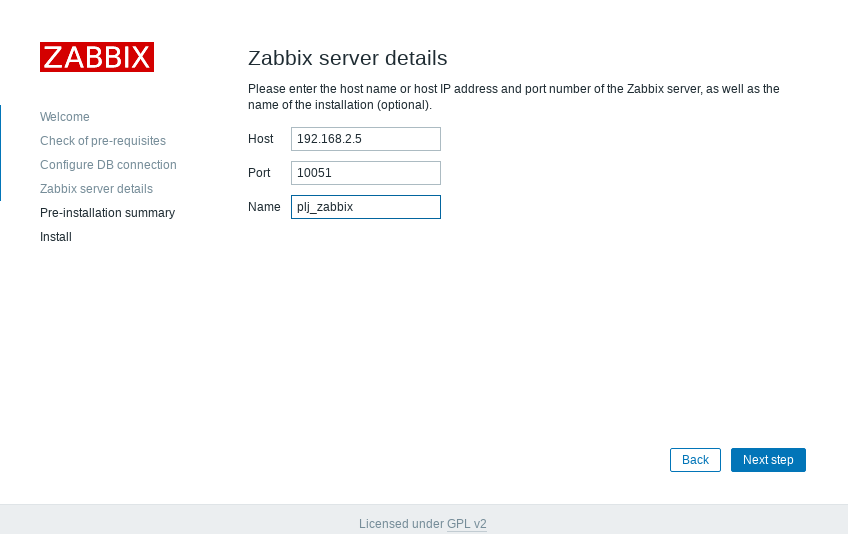
878 date.timezone = Asia/Shanghai

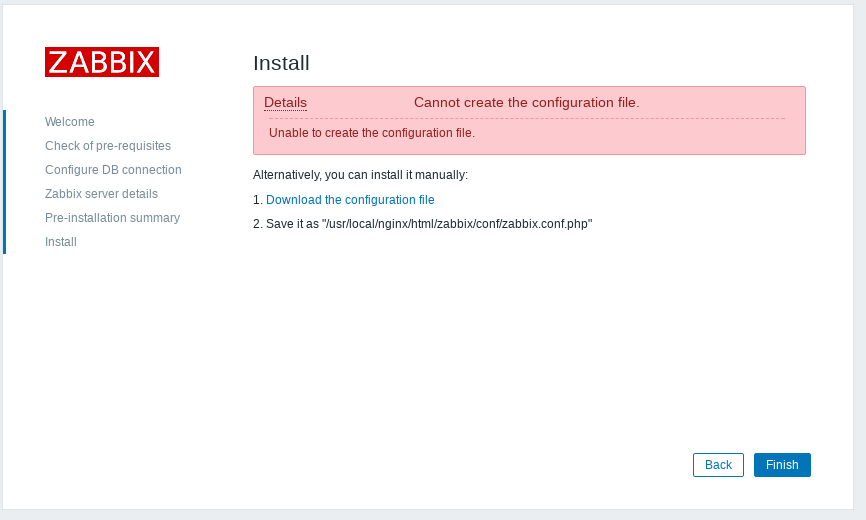
107 systemctl restart php-fpm





以上填写的是数据库相关信息,如果网页访问不在zabbix服务器上注意数据库zabbix用户的权限设置

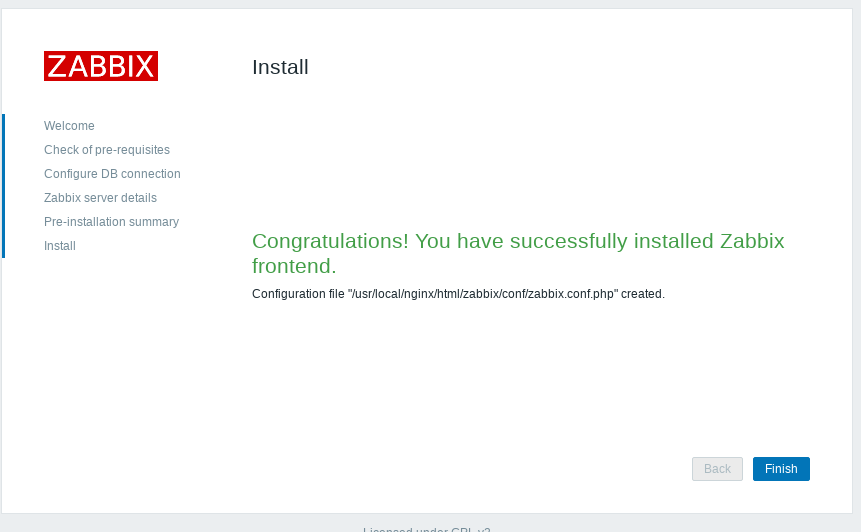


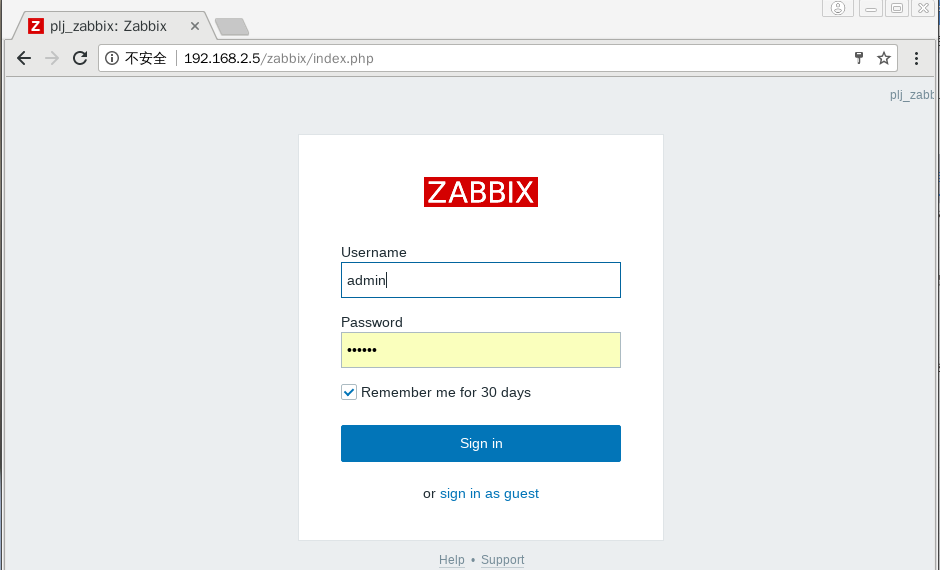


此处报错,是因为/usr/local/nginx/html/zabbix/conf/ 无权限

[root@zabbixserver lnmp\_soft]# chmod o+w /usr/local/nginx/html/zabbix/conf/

上一步back再finish





用户名改为:admin 密码:zabbix

10启动zabbix-server服务

1修改主配置文件

[root@zabbixserver ~]# vim /usr/local/zabbix/etc/zabbix\_server.conf

38 LogFile=/tmp/zabbix\_server.log

85 DBHost=localhost

95 DBName=zabbix

111 DBUser=zabbix

119 DBPassword=zabbix

2创建服务启动脚本

源码目录有模板,拷贝到 /etc/init.d/ 目录

[root@zabbixserver ~]# ls /root/zabbix-3.4.4/misc/init.d/fedora/core/zabbix\_server

/root/zabbix-3.4.4/misc/init.d/fedora/core/zabbix\_server

[root@zabbixserver ~]# chmod +x /etc/init.d/zabbix\_server

[root@zabbixserver ~]# sed -i '22s;/usr/local;/usr/local/zabbix;' /etc/init.d/zabbix\_server

[root@zabbixserver ~]# sed -n '22p' /etc/init.d/zabbix\_server

### 3启动并查看服务

[root@zabbixserver ~]# useradd zabbix

[root@zabbixserver ~]# /etc/init.d/zabbix\_server start

Reloading systemd: [ 确定 ]

Starting zabbix\_server (via systemctl): [ 确定 ]

[root@zabbixserver ~]# netstat -utnlp |grep :10051

tcp 0 0 0.0.0.0:10051 0.0.0.0:\* LISTEN 20234/zabbix\_server

[root@zabbixserver ~]# /etc/init.d/zabbix\_server status

### 4排错

启动之后一定要查看得到10051端口

1 /etc/init.d/zabbix\_server status 查看状态

2 查看日志 38 LogFile=/tmp/zabbix\_server.log

3 最开始编译安装未指定路径的

//--prefix=/usr/local/zabbix #如果未指定路径,默认装在/user/local/etc/zabbix

这样/user/local/etc/下有个zabbix脚本运行的都是这个zabbix

## 10网页调试Zabbix

1修改语言为中文

2修改密码

3查看可以使用的模板



4查看监控了哪些主机

## 11监控本机

### 1运行zabbix\_agentd服务

在编译安装时就以及安装了zabbix\_agentd服务 --enable-agent

**修改zabbix\_agentd服务的主配置文件**

[root@zabbixserver ~]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf

93 Server=127.0.0.1 #谁可以监控本机(被动监控)

134 ServerActive=127.0.0.1:10051 #谁可以监控本机(主动监控)

145 Hostname=zabbixserver #被监控端自己的主机名

69 EnableRemoteCommands=1 #监控异常后,是否允许服务器远程过来执行命令

280 UnsafeUserParameters=1 #是否允许自定定义key监控

30 LogFile=/tmp/zabbix\_agentd.log #日志位置

**启动zabbix\_agentd服务**

[root@zabbixserver ~]# ls /root/zabbix-3.4.4/misc/init.d/fedora/core

zabbix\_agentd zabbix\_server #客户端启动脚本

[root@zabbixserver ~]# cp /root/zabbix-3.4.4/misc/init.d/fedora/core/zabbix\_agentd /etc/init.d/

[root@zabbixserver ~]# chmod o+w /etc/init.d/zabbix\_agentd

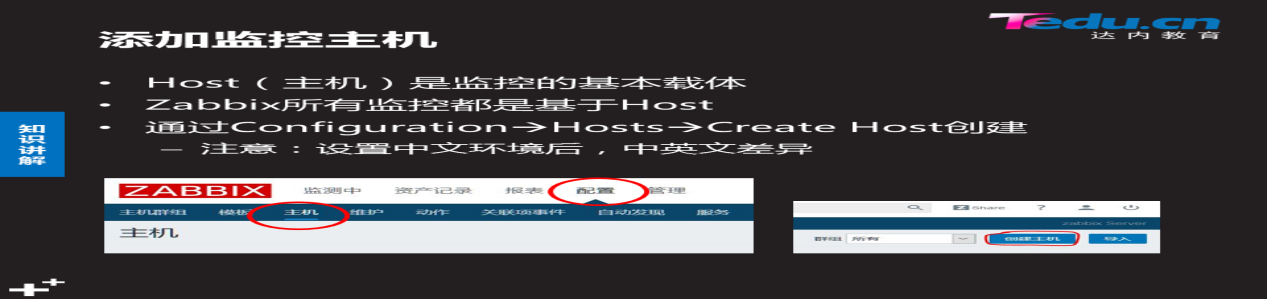
[root@zabbixserver ~]# sed -i '22s;/usr/local;/usr/local/zabbix;' /etc/init.d/zabbix\_agentd

[root@zabbixserver ~]# netstat -unltp |grep 10050

tcp 0 0 0.0.0.0:10050 0.0.0.0:\* LISTEN 21778/zabbix\_agentd

### 2登录管理页面做配置

**添加监控主机并选择监控目标**



**查看监控数据**

## **12监控远端主机**

### 1客户端与运行zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# yum -y install gcc pcre-devel

[root@zabbixweb1 zabbix-3.4.4]# ./configure --prefix=/usr/local/zabbix --enable-agent

[root@zabbixweb1 zabbix-3.4.4]# make install

[root@zabbixweb1 zabbix-3.4.4]# ls /usr/local/zabbix/

bin etc lib sbin share

### 2修改配置文件

[root@zabbixweb1 zabbix-3.4.4]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf

30 LogFile=/tmp/zabbix\_agentd.log

93 Server=127.0.0.1,192.168.2.5

134 ServerActive=192.168.2.5:10051

[root@zabbixweb1 zabbix-3.4.4]# cp misc/init.d/fedora/core/zabbix\_agentd /etc/init.d/

[root@zabbixweb1 zabbix-3.4.4]# chmod o+w /etc/init.d/zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# vim /etc/init.d/zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# useradd zabbix

[root@zabbixweb1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd start

Reloading systemd: [ 确定 ]

Starting zabbix\_agentd (via systemctl): [ 确定 ]

[root@web3 zabbix-3.4.4]# sed -i '22s;/usr/local;/usr/local/zabbix;' /etc/init.d/zabbix\_agentd

[root@zabbixweb1 zabbix-3.4.4]# netstat -untlp |grep :10050

tcp 0 0 0.0.0.0:10050 0.0.0.0:\* LISTEN 17842/zabbix\_agentd

### 3监控服务器配置

192.168.2.5登录页面,添加监控主机.选择监控目标

## 13自定义监控项

### 1客户端配置

启用自定义监控项功能

[root@zabbixweb1 ~]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf

265 Include=/usr/local/zabbix/etc/zabbix\_agentd.conf.d/\*.conf

#自定义的监控项明文件目录,以及命名

280 UnsafeUserParameters=1 #去掉注释改为1

### 2创建自定义规则项

[root@zabbixweb1 ~]# vim /usr/local/zabbix/etc/zabbix\_agentd.conf.d/a1.conf

UserParameter=get\_users,wc -l /etc/passwd |awk '{print $1}'

#固定格式,统计有多少系统用户名

[root@zabbixweb1 ~]# /etc/init.d/zabbix\_agentd stop

Stopping zabbix\_agentd (via systemctl): [ 确定 ]

[root@zabbixweb1 ~]# /etc/init.d/zabbix\_agentd start

Starting zabbix\_agentd (via systemctl): [ 确定 ]

[root@zabbixweb1 zabbix-3.4.4]# netstat -untlp |grep :10050

tcp 0 0 0.0.0.0:10050 0.0.0.0:\* LISTEN 17842/zabbix\_agentd

#确保10050端口正常否者,服务不正常需要检查

### 3测试自定义规则项

[root@zabbixweb1 ~]# /usr/local/zabbix/bin/zabbix\_get --help

[root@zabbixweb1 ~]# /usr/local/zabbix/bin/zabbix\_get -s 127.0.0.1 -p 10050 -k get\_users

42

### 4网站上配置

1主机群组: 组名qunzu

2新建模板:

模板名称:muban

可见的名称:muban.name (其他各项中被选看得到的名称)

上一步创建了,群组:qunzu,这里直接在右边选择qunzu,

如一步没创建群组,也可新建群组



3创建应用集:名称yinyongji #(模板里面)

4创建监控项: #(模板里面)

名称jiankongxiang

键值 get\_users #自定义规则里面的UserParameter=get\_users,.....

应用集选择:选择上一步创建的yinyongji

5创建主机

主机名称:web1.name

可见的名称:web1

群组选择:选择上一步创建的组名为:qunzu

进入模板界面选择链接指示器:右上角点击上面定义的名为qunzu的群组,选择弹出新 建的名为muban.name的模板,点击”添加”,再点击更新,最后如下图



6查看数据,如下图则成功



# 重启服务后检测所有的服务

[root@zabbix ~]# netstat -unltp | grep 3306

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 848/mysqld

[root@zabbix ~]# /usr/local/nginx/sbin/nginx

[root@zabbix ~]# netstat -unltp | grep 80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 918/nginx: master p

[root@zabbix ~]# netstat -unltp | grep 9000

tcp 0 0 127.0.0.1:9000 0.0.0.0:\* LISTEN

632/php-fpm: master

[root@zabbix ~]# /etc/init.d/zabbix\_server start

Starting zabbix\_server (via systemctl): [ 确定 ]

[root@zabbix ~]# netstat -unltp | grep 10051

tcp 0 0 0.0.0.0:10051 0.0.0.0:\* LISTEN

946/zabbix\_server

**访问:**

http://192.168.2.5/zabbix/index.php

# 案例

案例1：常用系统监控命令

案例2：部署Zabbix监控平台

案例3：配置及使用Zabbix监控系统

案例4：自定义Zabbix监控项目

1 案例1：常用系统监控命令

1.1 问题

本案例要求熟悉查看Linux系统状态的常用命令，为进一步执行具体的监控任务做准备：

查看内存信息

查看交换分区信息

查看磁盘信息

查看CPU信息

查看网卡信息

查看端口信息

查看网络连接信息

1.2 方案

一般企业做监控的目的：实时报告系统状态，提前发现系统的问题。

监控的资源可以分为：共有数据（HTTP、FTP等）和私有数据（CPU、内存、进程数等）。

监控软件可以使用：系统自带的命令、Cacti监控系统、Nagios监控系统、Zabbix监控系统。

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用命令查看计算机状态数据

1）查看内存与交换分区信息

[root@proxy ~]# free //查看内存信息

total used free shared buff/cache available

Mem: 16166888 8017696 720016 106504 7429176 7731740

Swap: 4194300 218268 3976032

[root@proxy ~]# free | awk '/Mem/{print $4}' //查看剩余内存容量

720928

[root@proxy ~]# swapon -s //查看交换分区信息

文件名 类型 大小 已用 权限

/dev/sda3 partition 4194300 218268 -1

步骤二：查看磁盘与CPU利用率

1）查看磁盘信息

[root@proxy ~]# df //查看所有磁盘的使用率

文件系统 1K-块 已用 可用 已用% 挂载点

/dev/sda2 476254208 116879624 335159084 26% /

/dev/sda1 198174 133897 49737 73% /boot

[root@proxy ~]# df | awk '/\/$/{print $5}' //查看根分区的利用率

2）查看CPU平均负载

[root@proxy ~]# uptime //查看CPU负载（1，5，15分钟）

23:54:12 up 38 days, 14:54, 9 users, load average: 0.00, 0.04, 0.05

[root@proxy ~]# uptime |awk '{print $NF}' //仅查看CPU的15分钟平均负载

0.05

步骤二：查看网卡信息、端口信息、网络连接信息

1）查看网卡信息

[root@proxy ~]# ifconfig eth0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.4.5 netmask 255.255.255.0 broadcast 172.25.0.255

inet6 fe80::5054:ff:fe00:b prefixlen 64 scopeid 0x20<link>

ether 52:54:00:00:00:0b txqueuelen 1000 (Ethernet)

RX packets 62429 bytes 10612049 (10.1 MiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 5674 bytes 4121143 (3.9 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@proxy ~]# ifconfig eth0 |awk '/inet /{print $2}' //查看IP地址信息

192.168.4.5

[root@proxy ~]# ifconfig eth0 |awk '/RX p/{print $5}' //网卡接受数据包流量

10625295

[root@proxy ~]# ifconfig eth0 |awk '/TX p/{print $5}' //网卡发送数据包流量

4130821

2）查看端口信息

[root@proxy ~]# ss -ntulp //查看本机监听的所有端口

//-n以数字显示端口号

//-t显示tcp连接

//-u显示udp连接

//-p显示监听端口对应的程序名称

3）查看网络连接信息

[root@proxy ~]# ss -antup //查看所有的网络连接信息

//-a查看所有连接状态信息

2 案例2：部署Zabbix监控平台

2.1 问题

本案例要求部署一台Zabbix监控服务器，一台被监控主机，为进一步执行具体的监控任务做准备：

安装LNMP环境

源码安装Zabbix

安装监控端主机，修改基本配置

初始化Zabbix监控Web页面

修改PHP配置文件，满足Zabbix需求

安装被监控端主机，修改基本配置

2.2 方案

使用1台RHEL7虚拟机，安装部署LNMP环境、Zabbix及相关的依赖包，配置数据库并对Zabbix监控平台进行初始化操作。使用2台被监控端，源码安装Zabbix Agent。完成Zabbix实验需要我们搭建一个实验环境，拓扑结构如表-1所示。

表-1 实验拓扑结构

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：部署监控服务器

1）安装LNMP环境

Zabbix监控管理控制台需要通过Web页面展示出来，并且还需要使用MySQL来存储数据，因此需要先为Zabbix准备基础LNMP环境。

[root@zabbixserver ~]# yum -y install gcc pcre-devel openssl-devel

[root@zabbixserver ~]# tar -xf nginx-1.12.2.tar.gz

[root@zabbixserver ~]# cd nginx-1.12.2

[root@zabbixserver nginx-1.12.2]# ./configure --with-http\_ssl\_module

[root@zabbixserver nginx-1.12.2]# make && make install

[root@zabbixserver ~]# yum -y install php php-mysql \

> mariadb mariadb-devel mariadb-server

[root@zabbixserver ~]# yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

//注意，php-fpm这个软件包在lnmp\_soft/目录下

2）修改Nginx配置文件

配置Nginx支持PHP动态网站，因为有大量PHP脚本需要执行，因此还需要开启Nginx的各种fastcgi缓存，加速PHP脚本的执行速度。

[root@zabbixserver ~]# vim /usr/local/nginx/conf/nginx.conf

… …

http{

… …

fastcgi\_buffers 8 16k; //缓存php生成的页面内容，8个16k

fastcgi\_buffer\_size 32k; //缓存php生产的头部信息

fastcgi\_connect\_timeout 300; //连接PHP的超时时间

fastcgi\_send\_timeout 300; //发送请求的超时时间

fastcgi\_read\_timeout 300; //读取请求的超时时间

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

… …

3）启动服务

启动Nginx、PHP-FPM、MariaDB服务，关闭SELinux与防火墙。

[root@zabbixserver ~]# systemctl start mariadb

[root@zabbixserver ~]# systemctl start php-fpm

[root@zabbixserver ~]# ln -s /usr/local/nginx/sbin/nginx /sbin/nginx

[root@zabbixserver ~]# nginx

[root@zabbixserver ~]# firewall-cmd --set-default-zone=trusted

[root@zabbixserver ~]# setenforce 0

4）客户端测试LNMP环境

服务器创建PHP测试页面，浏览器访问页面测试网页连通性。

[root@zabbixserver ~]# cat /usr/local/nginx/html/test.php

<?php

$i=33;

echo $i;

?>

[root@zabbixserver ~]# curl http://192.168.2.5/test.php

步骤二：部署监控服务器Zabbix Server

1）源码安装Zabbix Server

多数源码包都是需要依赖包的，zabbix也一样，源码编译前需要先安装相关依赖包。

[root@zabbixserver lnmp\_soft]# yum -y install net-snmp-devel \

> curl-devel

//安装相关依赖包

[root@zabbixserver lnmp\_soft]# yum -y install \

> libevent-devel-2.0.21-4.el7.x86\_64.rpm

//注意libevent-devel这个软件包在lnmp\_soft目录下有提供

[root@zabbixserver lnmp\_soft]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixserver lnmp\_soft]# cd zabbix-3.4.4/

[root@zabbixserver zabbix-3.4.4]# ./configure --enable-server \

> --enable-proxy --enable-agent --with-mysql=/usr/bin/mysql\_config \

> --with-net-snmp --with-libcurl

// --enable-server安装部署zabbix服务器端软件

// --enable-agent安装部署zabbix被监控端软件

// --enable-proxy安装部署zabbix代理相关软件

// --with-mysql配置mysql\_config路径

// --with-net-snmp允许zabbix通过snmp协议监控其他设备

// --with-libcurl安装相关curl库文件，这样zabbix就可以通过curl连接http等服务，测试被监控主机服务的状态

[root@zabbixserver zabbix-3.4.4]# make && make install

2）初始化Zabbix

创建数据库，上线Zabbix的Web页面

[root@zabbixserver ~]# mysql

mysql> create database zabbix character set utf8;

//创建数据库，支持中文字符集

mysql> grant all on zabbix.\* to zabbix@'localhost' identified by 'zabbix';

//创建可以访问数据库的账户与密码

[root@zabbixserver ~]# cd lnmp\_soft/zabbix-3.4.4/database/mysql/

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < schema.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < images.sql

[root@zabbixserver mysql]# mysql -uzabbix -pzabbix zabbix < data.sql

//刚刚创建是空数据库，zabbix源码包目录下，有提前准备好的数据

//使用mysql导入这些数据即可（注意导入顺序）

上线Zabbix的Web页面

[root@zabbixserver ~]# cd lnmp\_soft/zabbix-3.4.4/frontends/php/

[root@zabbixserver php]# cp -r \* /usr/local/nginx/html/

[root@zabbixserver php]# chmod -R 777 /usr/local/nginx/html/\*

修改Zabbix\_server配置文件，设置数据库相关参数，启动Zabbix\_server服务

[root@zabbixserver ~]# vim /usr/local/etc/zabbix\_server.conf

DBHost=localhost

//数据库主机，默认该行被注释

DBName=zabbix

//设置数据库名称

DBUser=zabbix

//设置数据库账户

DBPassword=zabbix

//设置数据库密码，默认该行被注释

LogFile=/tmp/zabbix\_server.log

//设置日志，仅查看以下即可

[root@zabbixserver ~]# useradd -s /sbin/nologin zabbix

//不创建用户无法启动服务

[root@zabbixserver ~]# zabbix\_server //启动服务

[root@zabbixserver ~]# ss -ntulp |grep zabbix\_server //确认连接状态，端口10051

tcp LISTEN 0 128 \*:10051 \*:\* users:(("zabbix\_server",pid=23275,fd=4),("zabbix\_server",pid=23274,fd=4)

提示：如果是因为配置文件不对，导致服务无法启动时，不要重复执行zabbix\_server，

一定要先使用killall zabbix\_server关闭服务后，再重新启动一次。

修改Zabbix\_agent配置文件，启动Zabbix\_agent服务

[root@zabbixserver ~]# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //允许哪些主机监控本机

ServerActive=127.0.0.1,192.168.2.5 //允许哪些主机通过主动模式监控本机

Hostname=zabbix\_server //设置本机主机名

LogFile=/tmp/zabbix\_server.log //设置日志文件

UnsafeUserParameters=1 //是否允许自定义key

[root@zabbixserver ~]# zabbix\_agentd //启动监控agent

[root@zabbixserver ~]# ss -ntulp |grep zabbix\_agentd //查看端口信息为10050

tcp LISTEN 0 128 \*:10050 \*:\* users:(("zabbix\_agentd",pid=23505,fd=4),("zabbix\_agentd",pid=23504,fd=4)

提示：如果是因为配置文件不对，导致服务无法启动时，不要重复执行zabbix\_agentd，

一定要先使用killall zabbix\_agentd关闭服务后，再重新启动一次。

浏览器访问Zabbix\_server服务器的Web页面

[root@zabbixserver ~]# firefox http://192.168.2.5/index.php

//第一次访问，初始化PHP页面会检查计算机环境是否满足要求，如果不满足会给出修改建议

//默认会提示PHP的配置不满足环境要求，需要修改PHP配置文件

根据错误提示，修改PHP配置文件，满足Zabbix\_server的Web环境要求

php-bcmath和php-mbstring都在lnmp\_soft目录下有提供软件包。

[root@zabbixserver ~]# yum -y install php-gd php-xml

[root@zabbixserver ~]# yum install php-bcmath-5.4.16-42.el7.x86\_64.rpm

[root@zabbixserver ~]# yum install php-mbstring-5.4.16-42.el7.x86\_64.rpm

[root@zabbixserver ~]# vim /etc/php.ini

date.timezone = Asia/Shanghai //设置时区

max\_execution\_time = 300 //最大执行时间，秒

post\_max\_size = 32M //POST数据最大容量

max\_input\_time = 300 //服务器接收数据的时间限制

memory\_limit = 128M //内存容量限制

[root@zabbixserver ~]# systemctl restart php-fpm

修改完PHP配置文件后，再次使用浏览器访问服务器，则会提示如图-1和图-2所示的提示信息。

图-1

图-2

注意：这里有一个PHP LDAP是warning状态是没有问题的！

在初始化数据库页面，填写数据库相关参数，如图-3所示。

图-3

在登陆页面，使用用户(admin)和密码(zabbix)登陆，登陆后设置语言环境为中文，如图-4和图-5所示。

图-4

图-5

步骤三：部署被监控主机Zabbix Agent

1）源码安装Zabbix agent软件

在2.100和2.200做相同操作（以zabbixclient\_web1为例）。

[root@zabbixclient\_web1 ~]# useradd -s /sbin/nologin zabbix

[root@zabbixclient\_web1 ~]# yum -y install gcc pcre-devel

[root@zabbixclient\_web1 ~]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixclient\_web1 ~]# cd zabbix-3.4.4/

[root@zabbixclient\_web1 zabbix-3.4.4]# ./configure --enable-agent

[root@zabbixclient\_web1 zabbix-3.4.4]# make && make install

2）修改agent配置文件，启动Agent

[root@zabbixclient\_web1 ~]# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //谁可以监控本机（被动监控模式）

ServerActive=127.0.0.1,192.168.2.5 //谁可以监控本机（主动监控模式）

Hostname=zabbixclient\_web1 //被监控端自己的主机名

EnableRemoteCommands=1

//监控异常后，是否允许服务器远程过来执行命令，如重启某个服务

UnsafeUserParameters=1 //是否允许自定义key监控

[root@zabbixclient\_web1 ~]# zabbix\_agentd //启动agent服务

3）拷贝启动脚本（非必须操作，可选做），有启动脚本可以方便管理服务，启动与关闭服务。启动脚本位于zabbix源码目录下。

[root@zabbixclient\_web1 zabbix-3.4.4]# cd misc/init.d/fedora/core

[root@zabbixclient\_web1 zabbix-3.4.4]# cp zabbix\_agentd /etc/init.d/

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd start

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd stop

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd status

[root@zabbixclient\_web1 zabbix-3.4.4]# /etc/init.d/zabbix\_agentd restart

3 案例3：配置及使用Zabbix监控系统

3.1 问题

沿用练习一，使用Zabbix监控平台监控Linux服务器，实现以下目标：

监控CPU

监控内存

监控进程

监控网络流量

监控硬盘

3.2 方案

通过Zabbix监控平台，添加被监控zabbixclient\_web1主机（192.168.2.100）并链接监控模板即可，Zabbix默认模板就可以监控CPU、内存、进程、网络、磁盘等项目。

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：添加监控主机

主机是Zabbix监控的基础，Zabbix所有监控都是基于Host主机。

使用火狐浏览器登录http://192.168.2.5，通过Configuration（配置）-->Hosts（主机）-->Create Host（创建主机）添加被监控Linux主机，如图-7所示。

图-7

添加被监控主机时，需要根据提示输入被监控Linux主机的主机名称（最好与电脑的主机名一致，但也允许不一致）、主机组、IP地址等参数，具体参考图-8所示。

图-8

步骤二：为被监控主机添加监控模板

Zabbix通过监控模板来对监控对象实施具体的监控功能，根据模板来定义需要监控哪些数据，对于Linux服务器的监控，Zabbix已经内置了相关的模板（Template OS Linux），选择模板并链接到主机即可，如图-9所示。

图-9

步骤三：查看监控数据

查看监控数据，登录Zabbix Web控制台，点击Monitoring(监控中)—> Latest data(最新数据)，正过滤器中填写过滤条件，根据监控组和监控主机选择需要查看哪些监控数据，如图-10所示。

图-10

找到需要监控的数据后，可以点击后面的Graph查看监控图形，如图-11所示。

图-11

4 案例4：自定义Zabbix监控项目

4.1 问题

沿用练习二，使用Zabbix实现自定义监控，实现以下目标：监控Linux服务器系统账户的数量。

4.2 方案

需要使用Zabbix自定义key的方式实现自定义监控，参考如下操作步骤：

创建自定义key

创建监控项目

创建监控图形

将监控模板关联到主机

4.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：被监控主机创建自定义key（在192.168.2.100操作）

1）创建自定义key

自定义key语法格式为：UserParameter=自定义key名称,命令。

自定义的key文件一般存储在/usr/local/etc/zabbix\_agentd.conf.d/目录，这里还需要修改zabbix\_agentd.conf文件，允许自定义监控key，来读取该目录下的所有文件 。

[root@zabbixclient\_web1 ~]# vim /usr/local/etc/zabbix\_agentd.conf

Include=/usr/local/etc/zabbix\_agentd.conf.d/ //加载配置文件目录

[root@zabbixclient\_web1 ~]# cd /usr/local/etc/zabbix\_agentd.conf.d/

[root@zabbixclient\_web1 zabbix\_agentd.conf.d]# vim count.line.passwd

UserParameter=count.line.passwd,wc -l /etc/passwd | awk ' {print $1} '

////自定义key语法格式:

//UserParameter=自定义key名称,命令

2）测试自定义key是否正常工作

[root@zabbixclient\_web1 ~]# killall zabbix\_agentd

[root@zabbixclient\_web1 ~]# zabbix\_agentd //重启agent服务

[root@zabbixclient\_web1 ~]# zabbix\_get -s 127.0.0.1 -k count.line.passwd

21

注意：如zabbix\_get命令执行错误，提示Check access restrictions in Zabbix agent configuration，则需要检查agent配置文件是否正确：

[root@zabbixclient\_web1 ~]# vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5

ServerActive=127.0.0.1,192.168.2.5

步骤二：创建监控模板

模板、应用集与监控项目的关系图，参考图-12所示

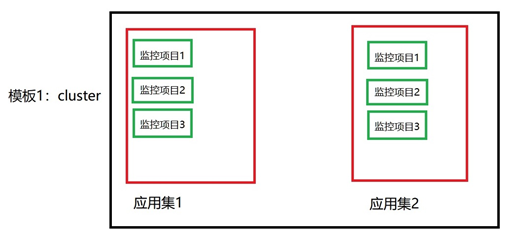


图-12

1）添加监控模板

登录Zabbix Web监控控制台，通过Configuration(配置)-->Template(模板)-->Create template(创建模板)，填写模板名称，新建模板群组，如图-13所示。



图-13

创建模板后，默认模板中没有任何应用、项目、触发器、图形等，如图-14所示。

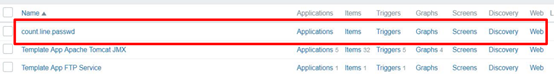


图-14

2）创建应用

创建完成模板后，默认模板中没有任何应用、项目、触发器、图形等资源。这里需要点击模板后面的Application（应用集）链接打开创建应用的页面，如图-15所示。

image014

图-15

点击Application（应用集）后，会刷新出图-16所示页面，在该页面中点击Create application（创建应用集）按钮。

image015

图-16

设置应用名称如图-17所示。



图-17

3）创建监控项目item（监控项）

与创建应用一样，在模板中还需要创建监控项目，如图-18所示，并在刷新出的新页面中选择Create items（创建监控项）创建项目，如图-19所示。

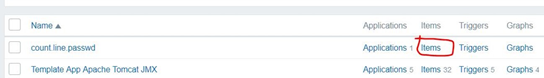


图-18

image018

图-19

接下来，还需要给项目设置名称及对应的自定义key，如图-20所示。

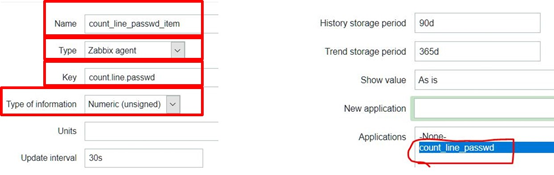


图-20

4）创建图形

为了后期可以通过图形的方式展示监控数据，还需要在模板中创建图形，设置方法与前面的步骤一致，在监控模板后面点击Graph（图形）即可创建图形，设置监控图形基于什么监控数据，如图-21所示。



图-21

5）将模板链接到被监控主机

将完整的监控模板制作完成后，就可以将模板链接到主机实现监控功能了。首先找到被监控主机Configuration（配置）-->Hosts（主机），如图-22所示。



图-22

点击需要的被监控主机链接，打开监控主机设置页面，在Template（模板）页面中选择需要链接到该主机的模板，在此选择刚刚创建的模板count\_line.passwd添加即可，如图-23所示。

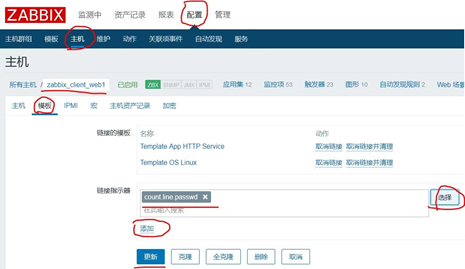


图-23

6）查看监控数据图形

点击Monitoring（监控中）-->Craphs（图形），根据需要选择条件，查看监控图形，如图-24和图-25所示。



图-24



图-25