1. Snort

入侵检测系统

1. PAM

PMA(Pluggable Authentication Module)是一个可插入式认证模块,在Linux系统中,各种不同的应用程序都需要完成认证功能,为了实现统一调配,把所有需要认证的功能做成一个模块(认证机制特别复杂的除外,如:https),当特定的程序需要完成认证功能的时候,就去调用PMA的认证模块,这些模块都位于系统中的/lib64/security(/lib/security:32位操作系统)目录下,但并不是所有的模块都是用来完成认证的,有些模块是为了实现PAM的某些高级功能而存在的,其中PMA的认证库是由glibc提供的,应用程序最终使用哪个PMA模块,取决于/etc/pma.d/\*这个目录下的定义

例如：只有sshusers组的用户能使用ssh远程登录

[root@lidefu etc]# groupadd sshusers #添加一个组

[root@lidefu etc]# vim /etc/pam.d/sshd #编辑远程登录的pam

auth required pam\_listfile.so item=group sense=allow file=/etc/security/allow-groups #添加这一句

[root@lidefu etc]# vim /etc/security/allow-groups #编辑组文件,就是上面那个

sshusers #添加组

例如：检查login程序是否支持PAM

ldd `which login` | grep libpam

1. YAML

YAML 是专门用来写配置文件的语言，非常简洁和强大，远比 JSON 格式方便。

1. svn

yum -y install subversion

svn checkout https://10.95.24.33/svn/qwhale

(svn co)

1. git相关命令

连接前需要生成ssh公私钥，然后添加到github，随后关联（git remote add \*\*\*），然后git clone便可以克隆

1. 生成ssh密钥

ssh-keygen -t rsa -C "email@\*\*\*"

1. 关联本地仓和远程仓，同步代码

git init --初始化本地git仓

远程仓相关操作：

git remote add origin git@github.com:zxl8121/learngit.git //本地仓关联远程仓账户

git pull --rebase origin master //合并分支master的远程仓和本地仓代码

git pull origin master //更新远程master分支的仓代码到本地

文件更新

git push -u origin master //将本地仓同步到远程 -u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令

git push origin master //提交本地仓到远端master分支

1. 操作分支

git branch //查看当前分支

git checkout [file name]//若文件有修改，可以还原到最初状态 ; 若文件需要更新到服务器上，应该先merge到服务器，再更新到本地

分支切换

git checkout [branch](切换分支)

git checkout -b zxl\_libta //创建分支(等同于 git branch zxl\_libta)

git branch -d master//删除master分支

git remote -v//查看远程库信息

git clone git@git.skyeye.world:zhuoliang/libta.git

git status //查看本地分支文件信息，确保更新时不产生冲突

git tag　列出所有版本号

git checkout　+某版本号

1. 新增修改文件提交

git add readme.txt

git commit -m "branch test"

1. 合并分支

git merge dev //命令用于合并指定分支到当前分支

1. 删除/还原仓文件

git rm test.txt

git rm –r [目录]

git commit -m "remove test.txt"

git checkout [文件/目录] //用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”

1. 日志文件查看 比较

git log --pretty=oneline

git diff HEAD -- readme.txt

1. Git fetch和git pull的区别:

都可以从远程获取最新版本到本地

1.Git fetch:只是从远程获取最新版本到本地,不会merge(合并)

$:git fetch origin master //从远程的origin的master主分支上获取最新版本到origin/master分支上

$:git log -p master..origin/master //比较本地的master分支和origin/master分支的区别

$:git merge origin/master //合并

2.Git fetch:从远程获取最新版本并merge(合并)到本地

$:git pull origin master //相当于进行了 git fetch 和 git merge两步操作

实际工作中,可能git fetch更好一些, 因为在merge前,可以根据实际情况决定是否merge

再说导致报错:error: You have not concluded your merge (MERGE\_HEAD exists).的原因可能是在以前pull下来的代码自动合并失败

解决办法一:保留本地的更改,中止合并->重新合并->重新拉取

$:git merge --abort

$:git reset --merge

$:git pull

解决办法二:舍弃本地代码,远端版本覆盖本地版本(慎重)

$:git fetch --all

$:git reset --hard origin/master

$:git fetch

1. vim常用命令记录

Ctrl + O 上次光标

Ctrl + I 后续光标

Ctrl+】 装了Ctag的插件可以跳转函数定义

纵向调整

res(ize) num 例如：:res 5，显示行数调整为5行

横向调整

:vertical res(ize) num 指定当前窗口为num列

:nohl 取消高亮

TAB替换为空格：

:set ts=4

:set expandtab

:%retab!

空格替换为TAB：

:set ts=4

:set noexpandtab

:%retab!

树形插件

:NERDTree

ctag相关：j

ctags -R --sort=yes --c++-kinds=+p --fields=+iaS --extra=+q --language-force=C++

目录背景色：

cp /etc/DIR\_COLORs ~/.dir\_colors

python相关：

补全和条状到函数定义

jedi-vim

\d 调转到定义出命令

1. docker相关记录

docker run -ti -v /tmp:/mnt -p<host\_port:contain\_port> 3d7ac13b921a /bin/bash ----P :是容器内部端口随机映射到主机的高端口。

docker run -ti -v /tmp:/mnt ubuntu:15.10 /bin/bash ----p : 是容器内部端口绑定到指定的主机端口。

docker run -d -v /etc/localtime:/etc/localtime -v /tmp/:/tmp/ --ulimit core=0:0 -p 2230:22 -p 8020:81 --name app\_attack app\_attack /usr/local/sbin/run.sh

docker run -ti --name centos\_6.6\_v4 --security-opt seccomp=unconfined -ti -v /data:/mnt -v /root/zxl\_workdir/docker\_code/centos\_pcap\_2:/opt/ -v /root/zxl\_workdir:/root/zxl\_workdir --privileged=true -p 22089:22 centos\_6.6\_v4:latest /bin/bash

docker run -ti --name centos\_6.6\_v4 --security-opt seccomp=unconfined -ti -v /mnt:/mnt -v /root/code\_dir/docker\_code:/opt/ --privileged=true -p 22089:22 centos\_6.6\_v4:latest /bin/bash

搜索可用的centos的docker镜像

docker search <image>：在docker index中搜索image

docker search centos

下载centos镜像（拉取镜像）

docker pull centos:latest

查看镜像

docker images：列出images

docker images -a：列出所有的images（包含历史）

docker images --tree ：显示镜像的所有层(layer)

docker rmi <image ID>：删除一个或多个image

查看容器

docker ps ：列出当前所有正在运行的container

docker ps -l：列出最近一次启动的container

docker ps -a ：列出所有的container（包含历史，即运行过的container）

docker ps -q ：列出最近一次运行的container ID

再次启动容器

docker start/stop/restart <container> ：开启/停止/重启container

docker start [container\_id]：再次运行某个container（包括历史container）

docker run -i -t <image> /bin/bash ：使用image创建container并进入交互模式, login shell是/bin/bash

docker run -i -t -p <host\_port:contain\_port> ：映射 HOST端口到容器，方便外部访问容器内服务，host\_port可以省略，省略表示把 container\_port映射到一个动态端口。

注：使用start是启动已经创建过得container，使用run则通过image开启一个新的container。

删除容器

docker rm <container...> ：删除一个或多个container

docker rm -f <>:删除一个正在运行的容器

docker rm `docker ps -a -q` ：删除所有的container

docker ps -a -q | xargs docker rm：同上,删除所有的container

删除镜像

docker rmi <镜像名或ID>

进入容器

docker attach \*\*\*\*(容器ID或名)

docker exec -it <name/ID> bin/bash

将容器封装为一个镜像

docker commit -m="commit update info" -a="image author" 6c4b67800f97(为容器名/ID) node:node1(为新的镜像名)

基于新镜像运行容器

docker run -it -d -p 88:80 --name test1 node:node1 /bin/bash

允许用户对容器内的系统调用

–security-opt seccomp=unconfined 由于Seccomp filtering的限制，在默认的配置下，会导致我们在用GDB的时候run失败，所以在执行docker run的时候加入–security-opt seccomp=unconfined这个参数，可以关闭seccomp profile的功能；

退出容器但不关闭容器

ctrl+P+Q

exit 退出但会关闭容器

--privileged=true

在运行容器的时候，给容器加特权

docker 导入导出镜像

docker save 9045 > tomcat8-apr.tar

docker load < tomcat8-apr.tar

docker tag 9045 tomcat8-apr:3.0

docker export cbe3cb7799ed > update.tar

docker import - update < update.tar

1. valgrind内存检测工具
2. ftp

yum install vsftpd

ftp客户端：

yum install ftp

service vsftpd status

service vsftpd start

netstat -anp|grep 21

开启root访问权限(注释掉下列文件中的root)：

/etc/vsftpd/ftpusers

/etc/vsftpd/user\_list

1. gdb调试

x/600xb 0x321922f0-128

ptype

p/s

p/x

info locals --显示当前函数的局部变量值

disass

frame 0 --进入栈帧