我是广宏伟,很高兴在高效运维群和大家分享开发jumpserver时的一些心得体会,感谢萧总,黄总邀约。

如果朋友们还没看过Jumpserver的介绍,可以访问 http://t.cn/R4N9xDH 了解 官网: http://jumpserver.org

github: https://github.com/ibuler/jumpserver 今天分享的主要内容有:

现了一个跳板机模型,以后就可以在这个模型上进行扩展,开枝散叶。

权限管理

1. 从开始到现在 2. 开发构思和结构

一. Jumpserver的开发历程

3. 开发心得分享 4. 未来规划

14年8月Jumpserver发布了 1.0版本,当时python刚刚入门,还不懂1.0意味着什么,现在看来是无知者无

畏,下面是当时的界面 运维管理系统 主页 用户管理 主机管理

!!! 用户管理

▲查看权限

▲查看用户 试试吧。» ▲添加用户 !!! 主机管理 ★主机列表 ▲添加主机 ||| 主机权限

跳板机系统管理系统,负责添加用户,添加主机和用户权限,不要笑它简陋哦

我们知道跳板机的核心是3A, 认证(Authentication) 授权(Authorization) 审计(Audit), 实现它们的核心就是去实 现一个跳转的网关,透明的转发用户的ssh连接。实现了这个网关就可以来认证用户,授权可以通过数据库建

立用户和主机的关系来完成。审计我觉得比较难一些,要记录用户的输入输出。如果实现了这个网关就基本实

admin▼

▲添加权限 ₩ VPN管理 ✓ OpenVPN v1.0版本使用python pexpect模块来构建了这个网关。认证和授权查询数据库,查询后使用pexpect模块连接 后端服务器,利用模块自带的方法记录log,通过web完成用户、主机和授权关系的增删该查。 再做了一个简单的查看日志的页面,来完成审计的功能。

在多次使用中发现我封装的pexpect方法不稳定,后来使用了pexpect自带的方法pxssh来完成,稳定了很多, 于是发布了v1.1版。

v1.x版本很简陋,但实现了基本的3A,虽然当时还不知道3A的概念。其中v1.x版本当时采用LDAP认证的方

用python的ldap库可很轻松的完成用户的增删改查。 下面是v1.x的结构图

pexpect

结构很简单,隐去了web的结构。web页面使用bootstrap基本也写的。

输入搜索...

用户总数

504

活跃用户TOP5

雷军

2 柳传志

3 马宏

4 李开复

5 丁磊

Jumpserver实时监控 登录用户名: qiaofeng 登录主机: 127.0.0.1

clear;echo -e '\033[32mLogin 127.0.0.1 done. Enjoy it.\033[0m'

Last login: Sun Apr 19 18:26:07 2015 from localhost

Starttime is 2015-04-19 21:57:18.479036

PS1='[\u@127.0.0.1 \W]\\$ '

[root@localhost ~]# clean[Kr [H[J[root@localhost ~]# [root@localhost ~]# ifconfig

[root@localhost ~]#

[root@localhost ~]# [root@localhost ~]#

过去一周共有10位用户登录870次服务

超级管理员。

22 仪表盘

山 用户管理

查看部门 添加部门

查看小组

添加小组

查看用户

添加用户

€ 资产管理

② 授权管理

② 日志审计

畫 访问官员

改成我们需要的格式。

仪表盘 / 查看日志 / 在线用户

监控页面截图:

用

504/504

案,当时考虑了很多最终选择了LDAP,主要原因在于开发起来比较容易。客户端和服务器端配置完成后,使

端交给用户 主机 成功 返回服务器账号密码 DB 查询认证、授权→ logfile方法记录日志 ssh

V1.x架构图

过了没多长时间,我的大学同学 王墉(halcyon)觉得功能还不错,正好他也在学习Django,于是他也加入进

连接

后端

成功

interact, 终

web审计时读取

欢迎使用Jumpserver开源跳板机系统 👺 🕩 Log out

196

已连接服务器 Connected

196 3

Version 2.0.0 GPI

登录

2015-04-1

实时在线用户

10

活跃用户数

100% J

Online user

一周数据总览

Source: JumpServer

04-16

→ 登录次数 → 活跃主机数

• 登录次数: 135次

来,一起开发新的版本。 王墉 是个比较有艺术气息的屌丝(虽然他当时python水平也刚入门),他嫌弃界面难看,于是我们打算做一个漂 板,网上找了很多没发现合适的,最后发现了一个出售模板的网站:wrapbootstrap,花了100多块买了一 个。 🗸 这时我们什么还没做。 后来了解paramiko是python里的标准ssh库,打算使用paramiko来完成pexpect的工作,毕竟一个是支持原生 使用它完成交互式,完成日志记录。网上搜这种资料,查看各种文档,知道了paramiko的channel,总算是完 成了基本的一个雏形。 其中还是存在很多问题,比如 窗口大小改变,比如方向键等bug,都没有一蹴而就 解决,我说这个是想告诉大家,当时我们真的很菜,都是边找资料,边测试案例代码完成的。每次一个bug的 改进都会欣喜若狂,正是这个一直鼓励我们一直向前。 ✓ 当时我们的工作也比较轻松,上班时有很多时间来 思考,每周六我们都会带着刚买不久的Mac去咖啡馆谈论一下需求和bug。Mac对于满足屌丝的虚荣心还是很 受用的,刚开始没想到咖啡馆,是去的国家图书馆,那里有电源,有免费的wifi可用,后来觉得太远,才去的 咖啡馆。 🗸 有朋友可能问,当时是怎么协作的呢? 当时用的是gitcafe,他一个分支,我一个分支,周末 merge一次,只会简单的pull, push, merge。每次出现冲突都整半天。4个月后,也就是15年4月发布了v2.0 版本。下面是v2.0的首页截图

主机总数

2366

All host

150

100

界面漂亮了很多,看来团队里有爱装逼的队友是挺不错的。 v2.0版本相比v1.x版本进步了很多,支持了批

量命令,批量授权,web端实时查看日志,统计命令等。我们也在分析模板时学习了bootstrap,js,jquery,

没错,基本都是现学的(虽然现在也一般)。布局的确定,按钮的位置基本都是从模板上找的,然后复制进去修

100% 5

101次/周

97次/周

92次/周

91次/周

88次/期

Copyright Jumpserver.org Organization © 2014-2015

Link encap:Ethernet HWaddr 00:0C:29:86:C1:F3 inet addr:192.168.20.209 Bcast:192.168.20.255 Mask:255.255.255.0 inet6 addr: fe80::20c:29ff:fe86:c1f3/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:14321382 errors:0 dropped:0 overruns:0 frame:0 TX packets:1542011 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:2382767464 (2.2 GiB) TX bytes:1178853425 (1.0 GiB) Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope:Host UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:1242111 errors:0 dropped:0 overruns:0 frame:0 TX packets:1242111 errors:0 dropped:0 overruns:0 carrier:0 Copyr collisions:0 txqueuelen:0 (其它图略,不浪费大家流量) 这里需要说一下这个实时监控是用 websocket做的,当时还不知道什么是websocket,就和公司开发的同事-宿朝阳 说了我的需求,他说websocket可以完成服务器主动向浏览器push,看了半天资料,试了半天案例, 发现django来实现很是吃力。于是让他帮助用nodejs实现了websocket。 <a> 监控的理论是tailf日志文件,发现 改动就push到浏览器上,感觉好像是实时监控的样子,对于这个新功能我很满意,在审计方面又向前走了一 大步 连接后端服务器还是用的LDAP,授权用的是组对组,就是将主机组授权给用户组,不支持给用户授 权单个主机 当时这么考虑就是在方便用户授权和开发难易做的折中,后来发现组对组也是有不方便的地 方,当然这是后话。 结构图就不画了,差别不大,只是将pexpect换做了paramiko。 v2.0版本发布以来,有很多运维朋友喜欢上了这个项目,可能是被漂亮的外表吸引,也可能是想学习

devops 随着大家的反馈,也是发现了各种问题,其中对LDAP的抱怨最多,可能是LDAP在国内不太流行的

原因吧,安装使用的大部分问题都集中在了这里,我们对LDAP的了解也止步于增删改查。于是我们做了一个

艰难的决定,下个版本放弃LDAP(最近 郭大勇 老师写了本 OpenLDAP实战指南,已经开卖,决定学习并接

经过v2.0版本后,jumpserver有了点小名气,吸引来了几位新的队友:陈尚委,喻茂峻,刘正,柯连春。海贼

新的队友的加入,带来了新的活力,新的思想。每个人的思维和眼界都是局限的,他们的到来无疑让我们有了 更多的选择,从Web Terminal到录像,从gatecafe到coding,从nodejs到tornado,从LDAP到ansible。当时

新的团队协作靠的是tower和qq群组语音,tower上下发任务,指定bug、需求到成员,tower产品设计的确实

iumpserver设想图

Jumpserver

拆分了模块,每个人负责一个模块,商定好api对接,没完成前可以使用假数据测试,最后整合。小红旗代表

有了团队后,新需求的完成速度变快,群里相互探讨可以快速搞定问题,每个人都可以高度关注自己的模块。

团队有团队的好处,也有它的不足,因为是个开源项目,大家都在用业余时间来写码,每个人的时间精力投入

但是我们还是在圣诞节那天发布了版本,主要考虑是 1. 制定了计划 2. 项目进展不快,希望发版后大家会将精

再次感谢队友 王墉,陈尚委,喻茂峻,刘正,柯连春的付出。尤其感谢 陈尚委的认真负责,在论坛、qq群运

connect.py

Django

增加授权

ansible

批量命令

批量命令

ansible

1. 用户管理 这里和常见的用户管理的区别是,不仅在web上管理,还需要在服务器上useradd用户,让用户可

2. 资产管理 这里我们想做成一个基本的CMDB,我们也一直朝这个方面努力。我们在添加、修改资产作出了

<u>parami</u>ko

监控

监控

获取资产和授

权系统用户

查询

增加用户,主

机,组关系

推送系统用户

web shell

web shell

以登陆到跳板机,触发脚本。这里使用了django内置的auth,只是做了一点点扩展

添加资产力求添加更少的内容,收集信息使用ansible来完成硬件的收集,能自动的不手动。

努力,支持批量更改,execl上传、导出。支持了资产信息的收集。

2). 用户、资产变更组后不需要考虑用户的推送和回收

中间层,不过这也给大家理解带来了困惑。

关联sudo,赋予它一些特殊的权限

时间文件,使用js渲染成动画

3). 可以直接添加硬件设备上的用户到系统用户, 然后关联给用户即可。

成功,运行

新的版本界面变化不大,功能改进了不少,详见开始 发的文档地址。 下面进入下一主题 新版本的架构。

各个模块

用户用户组主机主机细颗粒授权

授权类型,映射

sudo授权

授权申请

提供api

提供api

▶ 日志审计 ④

文件上次下载 web设置

批量执行命令

首页展示绘图

队列异步支持 双主支持

自动化授权

命令授权或过滤

▶ 授权管理

▶ 其它 ∈

▶ 授权后动作、添加用户分发key

监控 😑

命令统计

命令搜索

→ 录像 报表

强行结束

登录

服务器

记录日志

新建用户 -

数据库

王的队伍基本集齐,再不起航又更待何时,于是我们合计了一下协同开发模式,开始了新的征程。

心想,这次说不定我们会成功,做一个稳定的产品(虽然后来证明3.0版本也未能完成使命)。

不错。 qq群组里每周一次语音聊天碰需求和任务完成情况。下面是刚开始画的需求图

(++) 跳板机 =

登录管理

(▶)资产管理

了当时认为的难点,每个模块细分了功能。就是这样我们一步一步的进行着开发,

不一致,项目进展速度不一致,模块之间比较有相互依赖,导致后来进展并不是很顺利。

努力,于是低下头准备下一个版本的迭代,这距离上个版本发布过了小半年时间。

我们给自己指定了计划,打算多长时间完成一个需求,打算多长时间修复一个bug。 为了营造紧迫感,我们指定了发布计划,当时打算圣诞节发一版。

力集中到bug修复上。

营做出的努力,致敬。

先来一张结构图

登陆

二、Jumpserver 3.0架构分析

sshd

web shell, 监控,批量命令执行

下面来分一下各组件

web

connect.py

server认证

DB认证

token 输入

查询api

增删改查用户

提供api

websocket

输入和输出

筛选搜索统计 excel导入 基于默认账号扫描

网页批量

硬件配置

服务器状态

服务器性能

筛选搜索统计

ssh 工具

web terminal

登录认证

▶ 记录日志

跳转授权

增删改查资产

批量增删改查

基本cmdb

提供api

(▶) 用户管理

日志 返回让用户请求 输入,输入输 tornado | websocket 用户 监控日志变化

7. connect.py 用户登陆跳板机(非web)后会运行这个文件,因为写了脚本在 /etc/profile.d/下,登陆运行,运 行结束退出系统,这是跳板机登陆的入口,用户可以查看自己有权限的主机,输入主机ID后调用api查询是否 授权,选择授权的系统用户,以系统用户身份登陆主机,记录期间来回的数据,把输入命令记录到数据库,把 输入输出记录成日志 录像截图:

6. 录像回放 这里采用了TermRecord的代码,改动后揉了进入,核心理念是借鉴 script的机制,生成log文件和

7. Mac是装逼利器 8. 取舍 做开源项目和公司项目不同,很多时候都不得不在功能上做出取舍,满足大多数,放弃小部分,在通 用性上做出一些牺牲,使用ansible和saltstack就是例子 9. 语言是工具 产品更多关注的是功能和设计,语言只是实现它的工具。产品好比文章,语言是文字,除了写 字外,还需要更生动和唯美的修辞

4. 命令的黑白名单过滤 这也是一个跳板机的高级功能,可以对用户输入的命令进行过滤 5. 重构整合 Django class base view, 提供 Restful API

系统用户推送就是利用ansible在后面主机建立这些用户,设置key,设置它的sudo,仅此而已。 添加授权 就是将 用户、资产和资产上已有的系统用户关联起来,将来登录时查询这个关系即可。用户组,资 产组同理。 4. 日志审计 日志审计里面集合了 实时监控,命令统计,上传下载,批量命令等记录。 实时监控 就是调用

为系统用户关联上sudo,这样 系统用户就成为了拥有某些特殊权限用户的集合。 这个系统用户概念算是一个

系统用户新建时可以指定账号密码,这个场景适用于使用了其他工具 如saltstack推送了这些系统用户,或硬

件设备,总之就是系统用户已经在后面资产上建立了的情况,否则会自动为用户生成随机密码和密钥,为用户

Jumpserver的结构就分享到这,想必大家也看烦了。 咱们进入下一个话题: 开发中的感悟 三、开发Jumpserver中的感悟

CMDB作为一个基础的服务,是有需求的,而jumpserver也恰好有这个条件,所以将会细化资产管理,以 CMDB的形式呈现 2. 稳定性、易用性需要继续努力 不得不承认,现在这个版本还不是一个成熟的版本,很多设计存在纰漏,我们将会在稳定和易用方面继续努 力,这个是最基本的需求

1. 资产作为CMDB继续深入

3. 对windows、网络设备的支持

Django CBV 更加抽象,方便大家二次开发定制,并提供API易于和已有系统整合 6. 提供详细的文档、注释和测试用例,供大家学习和二次开发

现在的代码、文档、注释做的远远不够,将来一定在这里下足功夫,做一个合格的开源项目

很多朋友有这方面的需求,我们也会努力去做,有相关经验的朋友,可以指点一下

(为路上朋友节省流量,其他截图去 http://t.cn/R4N9xDH 查看) 起协作,累的时候可以看看他人,然后撸一发继续低头向前 么协作开发,不会去思考如何实现,不会去查麻烦的英文文档,不会去学git,不会在凌晨2、3点还在写代 码,总之痛苦并成长,享受SM的快感 3. 开始并坚持下去 Jumpserver刚开始做的也挺烂,也有很多人批评,这需要不懈的坚持,我终相信毛毛虫总 会变成蝴蝶。 因为毛毛虫如果有个蝴蝶心,这相当于冰女有了蝴蝶和龙心,这能不赢? 4. 时间! 时间! 开发一个项目会占用很多时间,如果你想做好的话, 这需要整个团队的付出,所以一定要分 配好时间。 5. 装逼无罪 这是老罗说的,装逼也是我们动力的春药 6. 很多以为过不去的坎,也许只是当时而已 遇到很多困难,总是多次尝试无法逾越,也许是水平不够,也许 是时间未到,总之不要放弃,心怀信念,一点点的拱,总会拨开乌云,我草,还得拨开雾霾才能见太阳的 想必大家已经不愿看我装逼了,咱们开始下个话题吧,未来的计划 四. 未来规划

3. 授权管理 这个模块算是重要的模块,对应的是3A中的授权,也是比较难做的,放弃了LDAP,就需要推送 用户,但是没有选择推送和用户相同名称的用户,而是使用了系统用户,系统用户间后面的解释 这里考虑有以下几点: 1). 推送同名账户 需要每次授权更改都要推送,出于对推送api稳定性考虑 2). 由于做了细颗粒化授权,用户添加到一个用户组后,理应继承这个组的授权,这是也需要在用户组授权的 主机上推送这个同名用户, 资产添加到一个资产组同理 3). 删除用户,用户组,修改用户和用户组授权,需要后端回收用户 4). 对硬件设备的兼容 开发起来难度重重,于是做出了一个折中的选择,添加了一个系统用户概念。系统用户是一系列常见角色的集 合或者抽象,比如 dev, op, dba 这三个系统用户,运维童鞋登录,就映射到op这个用户登录主机,开发就 dev, 数据库管理员就 dba, 这样可以解决上述遇到的难题 1). 系统用户只需要推送好,以后授权直接关联到资产上的某个系统用户即可

websocket服务,监控日志的变化,如果发生变化,就在web上展示出来。命令统计这个版本做了很大的改 善,基本可以精确统计用户执行的命令,之前的版本是通过日志处理截取的,不够准确。至于 上传下载,批 量命令就很简单了,查询数据库完成 5. Web Terminal 使用了tornado来完成了websocket的任务,放弃了之前的nodejs,原因是tornado是python 的框架,并且对websocket支持很好。前端js使用的是term.js,找个案例看一下很容易实现,这里实现的核心 是将用户的输入中转传给paramiko的channel,将channel返回的数据push给用户浏览器。感叹一下,用 tornado实现web socket真的很简单

1. 团队 有个好的团队,让项目充满多样性,一起做点事情远比自己玩有动力,责任感,比较心促使着我们一 2. 做个项目让你成长更快 这个真的真的是切身感悟,如果你不做项目,不会去想那么多事情,不要去了解怎

到此分享结束,感谢朋友们的耐心观看,再次感谢萧总,黄总邀约,感谢团队。