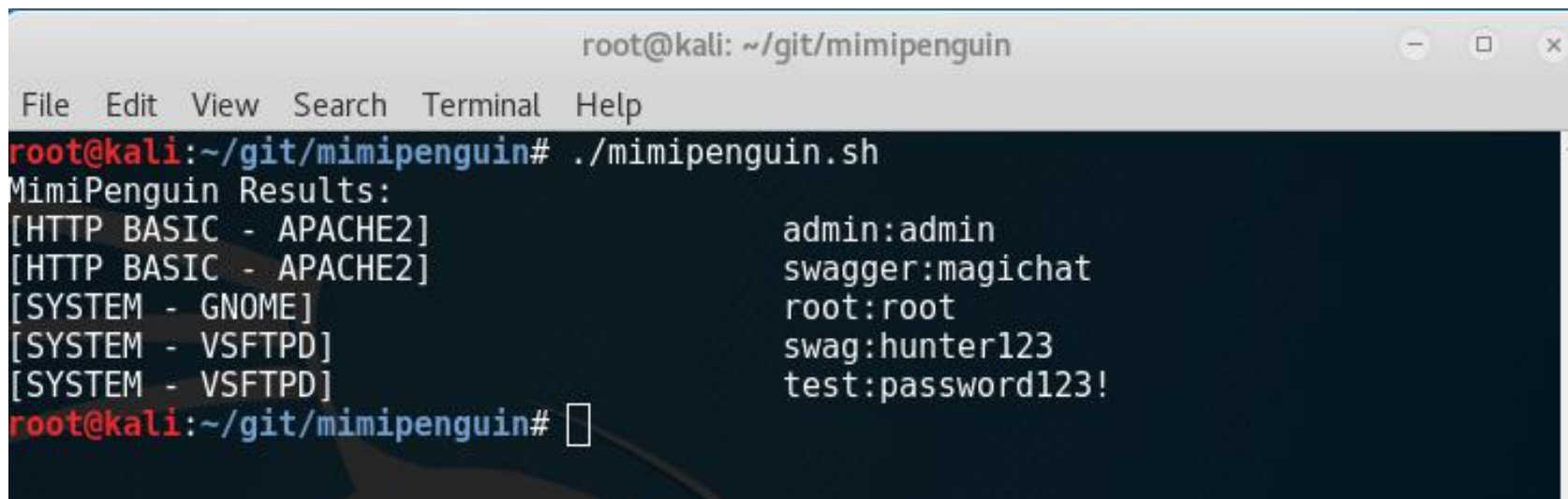


mimipenguin分析

by ming0

- mimipenguin是一款Linux下的密码抓取神器
- 项目地址：
<https://github.com/huntergregal/mimipenguin>
- 使用实例：



```
root@kali: ~/git/mimipenguin
File Edit View Search Terminal Help
root@kali:~/git/mimipenguin# ./mimipenguin.sh
MimiPenguin Results:
[HTTP BASIC - APACHE2]      admin:admin
[HTTP BASIC - APACHE2]      swagger:magichat
[SYSTEM - GNOME]           root:root
[SYSTEM - VSFTPD]           swag:hunter123
[SYSTEM - VSFTPD]           test:password123!
root@kali:~/git/mimipenguin#
```

- 使用要求: **root**权限
- 以下环境测试通过

Kali 4.3.0 (rolling) x64 (gdm3)

Ubuntu Desktop 12.04 LTS x64 (Gnome Keyring 3.18.3-0ubuntu2)

Ubuntu Desktop 16.04 LTS x64 (Gnome Keyring 3.18.3-0ubuntu2)

XUbuntu Desktop 16.04 x64 (Gnome Keyring 3.18.3-0ubuntu2)

VSFTPd 3.0.3-8+b1 (Active FTP client connections)

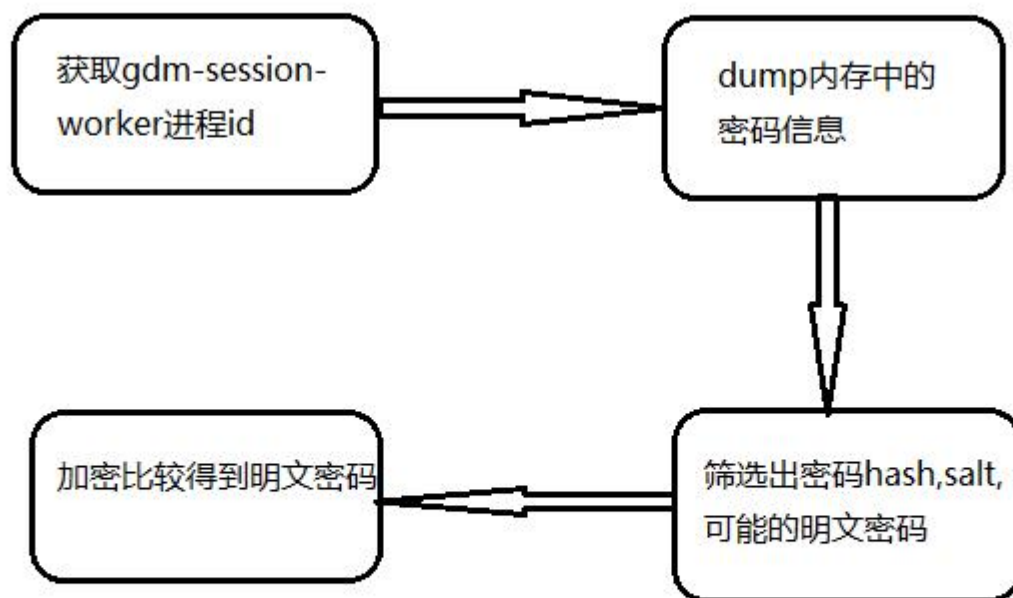
Apache2 2.4.25-3 (Active/Old HTTP BASIC AUTH Sessions)

openssh-server 1:7.3p1-1 (Active SSH connections sudo usage)

实现细节

- 通过转储进程并提取包含明文密码的可能性很高的行，利用内存中的明文凭证。将尝试通过检查/ **etc** / **shadow**中的哈希值，内存中的哈希值和正则表达式搜索来计算每个单词的概率。
- Takes advantage of cleartext credentials in memory by dumping the process and extracting lines that have a high probability of containing cleartext passwords. Will attempt to calculate each word's probability by checking hashes in /etc/shadow, hashes in memory, and regex searches.

具体分析--以抓取kali密码为例



1.判断是否是root权限

```
if [[ "$EUID" -ne 0 ]];
```

2.获取gdm-session-worker进程id

```
PID="$(ps -eo pid,command | sed -  
rn '/gdm-password\]/p' | awk 'BEGIN  
{FS = " " }; { print $1 }')"
```

awk是行文本分析工具

3.调用dump_pid(), 通过

/etc/[pid]/maps得到进程的内存信息,
再通过dd命令写入到临时文件

/tmp/dump.\${pid}

4.筛选出hash, salt和可能的明文密码

5.调用parse_pass(), 通过采用和密码相同的hash方式和已得到的hash进行对比, 确定明文密码

程序dump_pid()中/proc/[pid]/maps 文件解释

查看进程的虚拟地址空间是如何使用的。

该文件有**6**列，分别为：

地址：库在进程里地址范围

权限：虚拟内存的权限，**r**=读，**w**=写,**x**=, **s**=共享,**p**=私有；

偏移量：库在进程里地址范围

设备：映像文件的主设备号和次设备号；

节点：映像文件的节点号；

路径：映像文件的路径

每项都与一个**vm_area_struct**结构成员对应

程序dump_pid()中的> /dev/null 2>&1

/dev/null : 代表空设备文件

> : 代表重定向到哪里, 例如: `echo "123" > /home/123.txt`

1 : 表示stdout标准输出, 系统默认值是1, 所以"`>/dev/null`"等同于"`1>/dev/null`"

2 : 表示stderr标准错误

& : 表示等同于的意思, `2>&1`, 表示2的输出重定向等同于1

1 > /dev/null 2>&1 语句含义：

1 > /dev/null：首先表示标准输出重定向到空设备文件，也就是不输出任何信息到终端，说白了就是不显示任何信息。

2>&1：接着，标准错误输出重定向（等同于）标准输出，因为之前标准输出已经重定向到了空设备文件，所以标准错误输出也重定向到空设备文件。

将标准输出和错误输出全部重定向到 /dev/null中,也就是将产生的所有信息丢弃.

/etc/shadow详解

文件中7个字段含义

登录名:加密口令:最后一次修改时间:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志

1. “登录名”是与/etc/passwd文件中的登录名相一致的用户账号
2. “口令”字段存放的是加密后的用户口令字：
如果为空，则对应用户没有口令，登录时不需要口令；
星号代表帐号被锁定；
双叹号表示这个密码已经过期了；
\$6\$开头的，表明是用SHA-512加密；
\$1\$表明是用MD5加密；
\$2\$ 是用Blowfish加密；
\$5\$ 是用 SHA-256加密；

3. “最后一次修改时间”表示的是从某个时刻起，到用户最后一次修改口令时的天数。时间起点对不同的系统可能不一样。例如在**SCOLinux**中，这个时间起点是1970年1月1日。

4. “最小时间间隔”指的是两次修改口令之间所需的最小天数。

5. “最大时间间隔”指的是口令保持有效的最大天数。

6. “警告时间”字段表示的是从系统开始警告用户到用户密码正式失效之间的天数。

7. “不活动时间”表示的是用户没有登录活动但账号仍能保持有效的最大天数。

“失效时间”字段给出的是一个绝对的天数，如果使用了这个字段，那么就给出相应账号的生存期。期满后，该账号就不再是一个合法的账号，也就不能再用来登录了。