SSH Server

Kebutuhan untuk mengendalikan server dari jarah jauh adalah hal yang terpenting dari seorang Administrator, dengan kemampuan remote dari jarah jauh seorang Administrator mempermudah beberapa pekerjaan yang bersangkutan dengan server, sebab tidak harus berada di lokasi dimana server berada, terutama kerusakan dalam segi software maka pengendalian hak akses jarak jauh sangat dibutuhkan. SSH merupakan service yang paling banyak digunakan untuk kebutuhan remote sebuah server. SSH merupakan alternative penganti dari telnet, telnet adalah software yang digunakan untuk aktifitas remote server, tapi telnet sendiri memeliki beberapa kekurangan diantaranya dari segi keamanan yang dengan mudahnya diambil alih dari pihak lain, sebab telnet dalam pengiriman berupa data tidak disertakan encripsi, sehingga kemanan data kurang terjamin.

Walaupun SSH memiliki keamanan yang bisa diandalkan, tetapi SSH juga perlu ditingkatkan dalam segi keamana sehingga tidak mudah untuk diambil alih oleh orang yang tidak di ijinkan, berikut ini adalah beberapa tahap yang perlu dilakukan dalam mengamankan SSH Server.

3.1. Mengunakan protokol 2 pada SSH

Protokol 2 adalah protokol baru yang dimiliki SSH dengan beberapa kemanan yang sudah di uji dan lebih baik dari protokol 1

```
[root@serv1 ~]# nano /etc/ssh/sshd_config
Protocol 2
```

3.2. Mengganti Port Default

Secara default SSH menggunakan port 22, sehingga secara tidak langsung akan memudahkan pada attacker dalam mengambil alih SSH jika port tersebut tidak diganti. Agar lebih aman maka ganti port tersebut sesuai dengan keinginan anda

```
[root@serv1 ~]# nano /etc/ssh/sshd_config
# Port 22 ganti standart port pada SSH Server dengan 1290
Port 1290
```

3.3. Membatasi Akses SSH pada User

Sebuah server bisa diakses dengan mudah melalui service SSH, hal tersebut memberikan kesempatan beberapa user untuk mengaksesnya secara penuh sehingga perlu adanya batasan akses sehingga tidak semua user memiliki hak akses secara penuh. Misalnya administrator menambahkan nama user yaitu user1 dan user2, setelah mendaftarkan user kedalam sistem selanjutnya memberikan batasan hak akses.

```
[root@serv1 ~]# useradd user1
[root@serv1 ~]# useradd user2
[root@serv1 ~]# passwd user1
Changing password for user user1.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@serv1 ~]# passwd user2
Changing password for user user2.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Setelah anda menambahkan user pada server langkah selanjutnya adalah menambahkan konfigurasi pada file /etc/ssh/sshd_config yaitu mendaftarkan user2 untuk diijinkan akses ke server, secara default script tersebut tidak terdapat di file sshd_config, sehingga anda harus menambahkannya sendiri di paling bawah script

```
[root@serv1 ~]# nano /etc/ssh/sshd_config
AllowUsers user2
```

Sekarang kita lakukan uji coba apakah script tersebut sudah berjalan dengan baik yaitu dengan restart terlebih dahulu service sshd dengan perintah dibawah ini

```
[root@serv1 ~]# service sshd restart
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
[root@serv1 ~]#
```

Uji coba pertama dengan user yang tidak didefinisikan pada sshd_config, Port yang digunakan tidak 22 melainkan 1290 sehingga cara login pada ssh berbeda dengan sebelumnya, melainkan disertai port yang sudah didaftarkan, yaitu

```
mafatahna@mafatahna:~$ ssh user1@129.93.88.11 -p 1290
maya@129.93.88.11's password:
Permission denied, please try again.
maya@129.93.88.11's password:
```

Gagal! Berarti script yang anda masukkan tadi sudah berjalan dengan sukses, selanjutnya login dengan user yang sudah didaftarkan pada sshd_config

```
mafatahna@mafatahna:~$ ssh user2@129.93.88.11 -p 1290
mafatahna@129.93.88.11's password:
[mafatahna@serv1 ~]$
```

Login berhasil !!
Silahkan anda menambahkan user yang anda inginkan dan jangan lupa untuk limit user yang di ijinkan untuk akses ke server.

3.4. Konfigurasi idle timeout

Ketika seorang user sedang login dan tidak ada aktifitas apapun (*idle*) maka kita bisa memberikan batasan dalam bentuk timer pada user tersebut, misalnya jika user tidak melakukan aktifitas selama 1 menit maka secara otomatis sistem akan menutup koneksi SSH user tersebut

```
[root@serv1 ~]# nano /etc/ssh/sshd_config
#ClientAliveInterval 0
#ClientAliveCountMax 3
```

Hilangkan tanda pagar (#) pada script tersebut, angka yang digunakan menggunakan satuan detik sehingga jika anda menginginkan timeout 1 menit berarti 1 x 60 = 60s. Edit script tersebut seperti dengan dibawah ini

```
ClientAliveInterval 60
```

```
ClientAliveCountMax 0
```

Selanjutnya restart service SSH anda dan lakukan uji coba dengan login pada SSH dan membiarkan user tersebut tanpa aktifitas selama lebih dari 1 menit

```
mafatahna@mafatahna:~$ ssh mafatahna@129.93.88.11 -p 1290
mafatahna@129.93.88.11's password:
Last login: Tue May  1 09:31:29 2012 from 192.168.10.1
[mafatahna@serv1 ~]$ Connection to 129.93.88.11 closed by remote host.
Connection to 129.93.88.11 closed.
```

Setelah lebih dari 1 menit maka server secara otomatis akan menutup koneksi service SSH server, sehingga anda diharuskan login lagi jika ingin mengakses SSH.

3.5. Disable login root via SSH

Hak akses root jika dibiarkan terbuka akan sangat berbahaya sebab hal tersebut akan dimanfaatkan para attacker untuk melakukan brute force untuk mendapatkan hak akses root, sehingga untuk mengatasinya anda bisa menutup akses yang menggunakan user root, untuk menutup user root agar tidak bisa login via ssh anda hanya merubah konfigurasi *Permitrootlogin* menjadi *no* seperti pada konfigurasi dibawah ini

```
[root@serv1 ~]# nano /etc/ssh/sshd_config
#PermitRootLogin yes
```

Hilangkan tanda pagar (#) ganti parameter *yes* menjadi *no* seperti pada perintah dibawah ini

```
[root@serv1 ~]# nano /etc/ssh/sshd_config
#PermitRootLogin yes
PermitRootLogin no
```

Langkah selanjutnya adalah menguji apakah perintah yang sudah ditambahkan pada server SSH bisa berjalan dengan baik, login mengunakan user root

```
mafatahna@mafatahna:~$ ssh root@129.93.88.11 -p 1290
root@129.93.88.11's password:
Permission denied, please try again.
root@129.93.88.11's password:
```

Login gagal! Berarti konfigurasi yang anda lakukan sudah berhasil, selanjutnya login mengunakan user yang di ijinkan mengaksesnya SSH Server.

```
mafatahna@mafatahna:~$ ssh mafatahna@129.93.88.11 -p 1290
mafatahna@129.93.88.11's password:
Last login: Tue May  1 09:56:11 2012 from 192.168.10.1
[mafatahna@serv1 ~]$ su
Password:
[root@serv1 mafatahna]#
```