Keamanan Basis Data

Dosen Pengampu: Bapak Antoni Haikal, S.T., M.T

SSH TUNNELING ON POSTGRESQL

Submitted by:
Nisrina Amelia Putri
4332101006

Politeknik Negeri Batam
Teknik Informatika
Rekayasa Keamanan Siber

DAFTAR ISI

PENDAHULUAN

A. LATAR BELAKANG

Penggunaan enkripsi untuk melindungi privasi lalu lintas di Internet cukup umum digunakan pada saat ini. Salah satu teknik yang biasanya digunakan untuk mengamankan arus lalu lintas dikenal sebagai *tunneling* yang dilindungi secara kriptografis. Dalam hal ini, aliran paket tingkat IP atau tingkat TCP digunakan untuk menyalurkan segmen lapisan aplikasi yang berasal dari berbagai program perangkat lunak, sementara algoritme kriptografi melindungi privasi mereka. Salah satu mekanisme yang paling umum diimplementasikan oleh protokol *Secure Shell* (SSH), yang dapat diatur untuk melindungi secara kriptografis dengan terowongan aplikasi klien-server apa pun yang bekerja pada TCP.

Melindungi aliran data secara kriptografis dengan SSH memiliki dua tujuan, yaitu untuk menjaga privasi data pengguna, seperti kata sandi dan detail transaksi moneter, yang jika tidak akan diekspos oleh protokol teks biasa seperti POP3, SMTP, HTTP, dll. Di sisi lain, biasanya ada tujuan sekunder, tetapi sama pentingnya seperti perlindungan privasi dari perilaku pengguna. Dengan kata lain, aplikasi tunneling di SSH juga harus membuat perilaku pengguna menjadi pribadi bagi manajer jaringan snoopy mana pun di jalur tunnel SSH, dalam hal protokol yang ditunnelkan pengguna, dan situs yang mereka kunjungi. Alasan untuk jenis perlindungan kedua ini adalah dengan mengamati protokol yang digunakan seseorang, penyerang dapat memperoleh akses ke informasi sensitif yang seharusnya tetap bersifat pribadi, dan dapat membantu orang jahat menjawab beberapa pertanyaan penting tentang pengguna itu sendiri,

Pada laporan ini, akan mengimplementasikan SSH Tunneling serta membandingkan peformanya dengan SSH biasa berdasarkan kecepatan yang dihasilkan dari keduanya dengan beberapa cara yang telah dipelajari pada bab sebelumnya.

B. TUJUAN

Adapun tujuan pembuatan laporan ini ialah untuk:

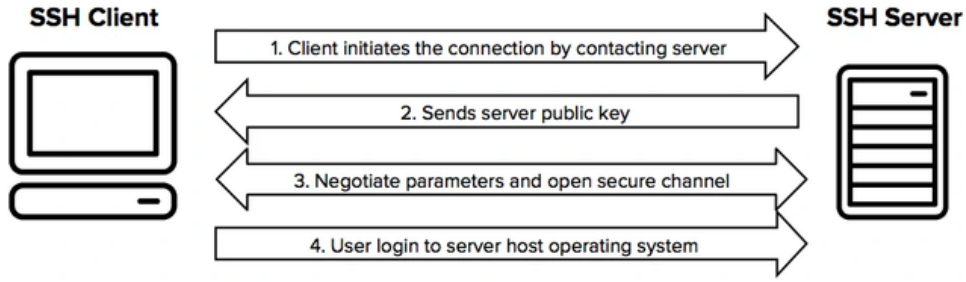
1. Mengetahui Dasar Teori pada SSH dan SSH TUNNEL.
2. Dapat Mengimplementasikan penggunaan SSH dan SSH TUNNEL.
3. Mengetahui Perbandingan Performa yang dihasilkan oleh SSH dan SSH TUNNEL.
4. Dapat membuktikan implementasi dan performa yang ada pada SSH TUNNEL.
5. Dapat melakukan testing SSH TUNNEL secara langsung.
6. Sebagai sumber untuk memenuhi nilai tugas Mata Kuliah Keamanan Basis Data.

C. LANDASAN TEORI

Apa itu SSH?

SSH adalah kepanjangan dari secure shell/socket shell yang merupakan sebuah protokol jaringan yang membantu pengguna terutama administrator sistem untuk dapat mengakses perangkat secara aman meski melalui jaringan yang tidak aman. SSH dapat digunakan untuk mengakses perangkat dari jarak yang sangat jauh. Sejauh ini, dengan adanya SSH adalah sebagai pengganti dari telnet yang dinilai masih kurang aman.

Dengan menggunakan SSH, maka setiap koneksi dan komunikasi antara client dengan server akan dienkripsi. Client SSH adalah sebuah aplikasi yang digunakan untuk menghubungkan sistem operasi pada perangkat menuju server. SSH akan membuat koneksi antara client dan server menjadi aman menggunakan port 22, setelah mengautentikasi keduanya, pertukaran pesan/data dapat dilakukan dengan aman. Hal ini sangat mirip dengan ketika Anda mengakses website yang sudah memasang sertifikat SSL sehingga protokol sudah berganti menjadi HTTPS. Sedangkan menggunakan telnet atau rlogin bisa dianalogikan seperti ketika Anda mengunjungi website yang masih HTTP.



Cara Kerja SSH

Apa itu SSH TUNNEL?

SSH Port Forwarding atau SSH Tunneling merupakan sebuah teknik atau cara untuk melakukan tunneling protocol yang memungkinkan data dapat ditransfer dari satu jaringan ke jaringan lain menggunakan SSH. Dengan memanfaatkan SSH koneksi akan menjadi lebih aman apabila dibandingkan dengan mengakses secara langsung. Secara singkat, SSH Port Forwarding dapat digunakan untuk mengakses protokol yang kurang secure atau rentan karena tidak dienkripsi menjadi lebih secure (aman) dengan menggunakan SSH karena SSH menggunakan enkripsi.

Port forwarding atau port mapping pengalihan (redirection) koneksi dari suatu IP:Port ke IP:Port yang lain. Ini artinya adalah semua koneksi yang ditujukan ke IP:Port asal akan dialihkan ke IP:Port tujuan seolah-olah client sedang menghubungi IP:Port tujuan secara langsung. Contoh: bila kita definisikan port forwarding 127.0.0.1:8080 dipetakan ke 192.168.10.10:80, artinya bila browser di arahkan ke url http://127.0.0.1:8080, maka request HTTP tersebut akan diteruskan ke 192.168.10.10:80. Jadi walaupun pada localhost (127.0.0.1) port 8080 tidak ada web server, namun web browser bisa membuka web pada url http://localhost:8080.

PEMBAHASAN

A. IMPLEMENTASI SSH TUNNEL

Untuk mengimplementasikan SSH TUNNEL pada postgresql, dapat menggunakan fitur openssh. SSH Tunnel dapat dilakukan melalui terminal windows dengan perintah ssh seperti biasa. Pertama-tama dapat melakukan SSH ke Web Server dengan perintah ***ssh ws.name@ip.ws***

1. SSH ke Web Server

```
C:\Users\HP>ssh nisrina@192.168.56.102
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be established.
ECDSA key fingerprint is SHA256:tWFsWYEF46DfEXgq0MUdhA+RKN40w29YXQtuAWw0BYY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.102' (ECDSA) to the list of known hosts.
nisrina@192.168.56.102's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-52-generic x86_64)

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

  System information as of Tue Dec 13 10:19:20 AM UTC 2022

  System load:  0.646484375       Processes:               108
  Usage of /:   66.9% of 8.02GB   Users logged in:         1
  Memory usage: 22%               IPv4 address for enp0s3: 192.168.56.102
  Swap usage:   0%                IPv4 address for enp0s8: 10.0.3.15

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

21 updates can be applied immediately.
7 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
```

2. SSH Tunnel ke Database Server

Untuk melakukan SSH Tunnel ke database server melalui web server dapat menggunakan perintah:
ssh -L portsamaran:ipdb:portdb uname@ipdb

```
nisrina@nisrinadb:~$ ssh -L 2112:192.168.56.106:5432 nisr@192.168.56.106
nisr@192.168.56.106's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-53-generic x86_64)

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

  System information as of Wed Dec 14 08:27:11 AM UTC 2022

  System load:  0.09375           Processes:               118
  Usage of /:   79.3% of 8.02GB   Users logged in:         1
  Memory usage: 19%               IPv4 address for enp0s3: 192.168.56.106
  Swap usage:   30%               IPv4 address for enp0s8: 10.0.3.15

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

34 updates can be applied immediately.
26 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Wed Dec 14 03:49:55 2022 from 192.168.56.102
nisr@nisr:~$ S
```

3. Memeriksa Port yang Digunakan

Pemeriksaan port ini berkaitan dengan berhasil atau tidaknya port fowarding yang sedang dilakukan. Dapat terlihat bahwa port 2112 telah terbuka, ini menandakan bahwa ssh tunneling berhasil dilakukan.

```
nisrina@nisrinadb:~$ sudo netstat -tunpln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      730/sshd: /usr/sbin
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      643/systemd-resolve
tcp        0      0 127.0.0.1:2112          0.0.0.0:*               LISTEN      10503/ssh
tcp6       0      0 :::22                   :::*                    LISTEN      730/sshd: /usr/sbin
tcp6       0      0 ::1:2112                :::*                    LISTEN      10503/ssh
udp        0      0 127.0.0.53:53           0.0.0.0:*                           643/systemd-resolve
udp        0      0 10.0.3.15:68            0.0.0.0:*                           641/systemd-network
nisrina@nisrinadb:~$ _
```

Dapat dilakukan pengetesan untuk masuk ke database. Terlihat disini bahwa tidak ada masalah pada database yang sedang diakses dan perintah dapat dilakukan dengan normal.

```
nisr@nisr:~$ sudo -i -u postgres
[sudo] password for nisr:
postgres@nisr:~$ psql
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# \q
postgres@nisr:~$
postgres@nisr:~$ psql
psql (14.5 (Ubuntu 14.5-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# create database kambasdat;
CREATE DATABASE
postgres=# \c kambasdat
You are now connected to database "kambasdat" as user "postgres".
kambasdat=# \q
```

B. PERFORMA SSH TUNNELING

Untuk melihat performa, disini membandingkan antara tiga paremeter yaitu secara ssh tunneling, ssh biasa, dan local.

1. Secara SSH Tunnel

```
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.73 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 4.72 s (drop tables 0.00 s, create tables 0.01 s, client-side generate 2.79 s, vacuum 0.82 s, primary keys 1.09 s).
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.77 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 4.43 s (drop tables 0.05 s, create tables 0.01 s, client-side generate 2.81 s, vacuum 0.53 s, primary keys 1.04 s).
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.72 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 4.44 s (drop tables 0.03 s, create tables 0.01 s, client-side generate 2.90 s, vacuum 0.47 s, primary keys 1.04 s).
```

2. Secara SSH Biasa

```
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.31 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 3.80 s (drop tables 0.03 s, create tables 0.02 s, client-side generate 2.37 s, vacuum 0.47 s, primary keys 0.90 s).
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.16 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 3.72 s (drop tables 0.03 s, create tables 0.01 s, client-side generate 2.25 s, vacuum 0.47 s, primary keys 0.96 s).
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.24 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 3.73 s (drop tables 0.03 s, create tables 0.01 s, client-side generate 2.27 s, vacuum 0.49 s, primary keys 0.93 s).
postgres@nisr:~$
```

3. Secara Local

```
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.17 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 3.71 s (drop tables 0.03 s, create tables 0.01 s, client-side generate 2.27 s, vacuum 0.47 s
, primary keys 0.93 s).
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.18 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 3.68 s (drop tables 0.03 s, create tables 0.01 s, client-side generate 2.22 s, vacuum 0.50 s
, primary keys 0.92 s).
postgres@nisr:~$ pgbench -i -s 10 kambasdat
dropping old tables...
creating tables...
generating data (client-side)...
1000000 of 1000000 tuples (100%) done (elapsed 2.17 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 3.69 s (drop tables 0.04 s, create tables 0.01 s, client-side generate 2.21 s, vacuum 0.49 s
, primary keys 0.94 s).
postgres@nisr:~$
```

ANALISIS SINGKAT

Data ditampilkan dalam tabel analisa:

METODE	Percobaan I	Percobaan II	Percobaan III	Rata-rata Percobaan
SSH TUNNELING	4,72 s	4,43 s	4,44 s	4,53 s
SSH	3,80 s	3,72 s	3,73 s	3,75 s
LOCAL	3,71 s	3,68 s	3,69 s	3,69 s

Dapat dilihat, akses dari local akan lebih cepat dibandingkan dengan ssh dan port forwarding. Hal ini disebabkan karena ssh dan port forwarding harus melewati server lain terlebih dahulu.

PENUTUP

A. KESIMPULAN

Dengan menggunakan SSH, maka setiap koneksi dan komunikasi antara client dengan server akan dienkripsi. Client SSH adalah sebuah aplikasi yang digunakan untuk menghubungkan sistem operasi pada perangkat menuju server. SSH akan membuat koneksi antara client dan server menjadi aman menggunakan port 22, setelah mengautentikasi keduanya, pertukaran pesan/data dapat dilakukan dengan aman.

SSH Port Forwarding atau SSH Tunneling merupakan sebuah teknik atau cara untuk melakukan tunneling protocol yang memungkinkan data dapat ditransfer dari satu jaringan ke jaringan lain menggunakan SSH. Dengan memanfaatkan SSH koneksi akan menjadi lebih aman apabila dibandingkan dengan mengakses secara langsung. Secara singkat, SSH Port Forwarding dapat digunakan untuk mengakses protokol yang kurang secure atau rentan karena tidak dienkripsi menjadi lebih secure (aman) dengan menggunakan SSH karena SSH menggunakan enkripsi.

Dari perbandingan performa dari ketiga perimeter, dihasilkan bahwa akses lokal memiliki kecepatan yang paling unggul, akses ssh biasa mendapatkan kecepatan unggu kedua, dan yang paling lambat adalah ssh tunneling atau port forwarding. Hal ini disebabkan karena ketika melakukan ssh, diperlukan akses ke server lokal terlebih dahulu untuk dapat masuk ke server tersebut sehingga aksesnya akan lebih lama.

DAFTAR PUSTAKA

Cao, F., Huang, H.K., Zhou, X.Q.: Medical Image security in a HIPAA mandated PACS environment. Computerized Medical Imaging and Graphics 27, 185–196 (2003)

Stein, S.: A cyberspace treaty - a United Nations convention or protocol on cyber security and cybercrime. In: Twelfth United Nations on Crime Prevention and Criminal Justice Salvador. United Nations, Brazil (2010)

Shimizu, S., Nakashima, N., Okamura, K., Tanaka, M.: One Hundred Case Studies of AsiaPacific Telemedicine Using a Digital Video Transport System over a Research and Education Network. Telemedicine Journal and E-Health 15(1), 112–117 (2009)

Hahm, J.S., Lee, H.L., Kim, S.I.: A remote educational system in medicine using digital video. Hepato Gastroenterology 54(74), 373–376 (2007)

Liu, B.J., Zhou, Z., Gutierrez, M.A., et al.: International Internet2 connectivity and performance in medical imaging applications: Bridging the America to Asia. Journal of High Speed Networks 16(1), 5–20 (2007)

Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. arXiv preprint arXiv:1801.01207, 2018.