p-ISSN: 2252-3006
e-ISSN: 2685-2411

Implementasi Algoritma PRNG pada Aplikasi Port Knocking Sebagai Perlindungan Server

Made Andika Verdiana, I Made Agus Dwi Suarjaya, Anak Agung Ketut Agung Cahyawan Wiranatha
Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana
Bukit Jimbaran, Bali, Indonesia, Telp. (0361) 701806
e-mail: andikaverdiana1@gmail.com, agussuarjaya@it.unud.ac.id,
agung.cahyawan@unud.ac.id

Abstrak

Keamanan jaringan harus dipertimbangkan dan dilindungi dengan baik. Masalah yang terjadi jika keamanan jaringan tidak terlindungi akan menyebabkan kerusakan sistem server dan dapat terjadinya akses tidak sah pada sistem server. Penelitian ini bertujuan untuk melakukan proses autentikasi jaringan antara komputer klien dengan komputer server dalam memvalidasi klien yang akan terhubung dengan sistem server. Penelitian dilakukan dengan metode autentikasi port knocking yang telah diimplementasikan algoritma PRNG (pseudo random number generator) dengan acuan seed waktu sistem. Hasil dari penelitian ini yaitu autentikasi port knocking dilakukan menggunakan hasil random sequence port yang berbeda sesuai dengan waktu sistem saat random number dibangkitkan guna untuk memperkuat autentikasi dalam memvalidasi klien yang akan terhubung dengan sistem server.

Kata kunci: *Keamanan jaringan, Autentikasi, Port Knocking, PRNG.*

Abstract

Network security must be considered and protected properly. Problems occur if unprotected network security causes damage to the server system and unauthorized access to the server system can occur. This study aims to carry out a network authentication process between client computers and server computers in validating clients that will be connected to the server system. The study was conducted with the port knocking authentication method which has been implemented by the PRNG algorithm (pseudo-random number generator) with a system time seed reference. The results of this study are that port knocking authentication will be performed using different random port sequence results according to the system time when the random number is generated, to strengthen authentication in validating clients who will be connected to the server system.

Keywords : *Network Security, Authentication, Port Knocking, PRNG.*

1. Pendahuluan

Keamanan jaringan harus dipertimbangkan dan dilindungi dengan baik. Keamanan jaringan merupakan aktivitas dalam pengamanan perangkat *server*, klien dan media transmisi data. Jika keamanan jaringan tidak dipertimbangkan dan dilindungi dengan baik, maka peretas mungkin dapat menargetkan saluran komunikasi untuk mendapatkan data yang di-enkripsi, kemudian men-dekripsi dan memasukkan kembali pesan palsu [1],[2]. *Man-in-the-middle-attack* merupakan serangan jaringan dimana penyerang menyampaikan dan mengubah komunikasi antara dua pihak yang terpercaya bahwa mereka sedang berkomunikasi secara langsung antara satu sama lain [3]. Autentikasi diperlukan sebagai pembentukan identitas antara dua belah pihak yang berkomunikasi untuk membangun layanan komunikasi yang aman saat dilakukan pertukaran data [4]. *Port knocking* merupakan sebuah metode autentikasi yang digunakan untuk menyembunyikan *service port*, beserta untuk membuka akses pada *service port* tertutup harus menggunakan ketukan *port sequence* [5]. *Pseudo random number generator* (PRNG) merupakan algoritma yang dapat menyiratkan rumus matematika, tabel perhitungan sederhana atau bahkan keduanya pada waktu tertentu dan menghasilkan angka acak berurutan [6],[7]. Melalui autentikasi *port knocking*, perangkat klien dapat mengakses *service*

port tertutup seperti SSH, FTP dan MySQL pada perangkat *server*. Autentikasi yang digunakan pada penelitian ini yaitu menggunakan aplikasi Knockd dan Python [8].

Gagasan dalam implementasi algoritma PRNG pada aplikasi *port knocking* dilakukan dengan tujuan untuk memperkuat proses autentikasi dalam memvalidasi komputer klien yang akan terhubung dengan sistem *server*. Dari beberapa penelitian yang telah dilakukan proses autentikasi masih menggunakan *static sequence port* [9],[10] yang menyebabkan proses autentikasi masih cukup rentan dari serangan jaringan. Maka dari itu dilakukan pengembangan sebuah aplikasi yang dapat melakukan proses autentikasi menggunakan *random sequence port* untuk memperkuat proses autentikasi. *Random sequence port* diperoleh dari pembangkitan nilai *random* menggunakan acuan *seed* waktu sistem. Selain itu, aplikasi dibangun menggunakan *framework* PyQt4 (GUI) dengan tujuan untuk mempermudah pengguna dalam menggunakan aplikasi yang dirancang pada penelitian.

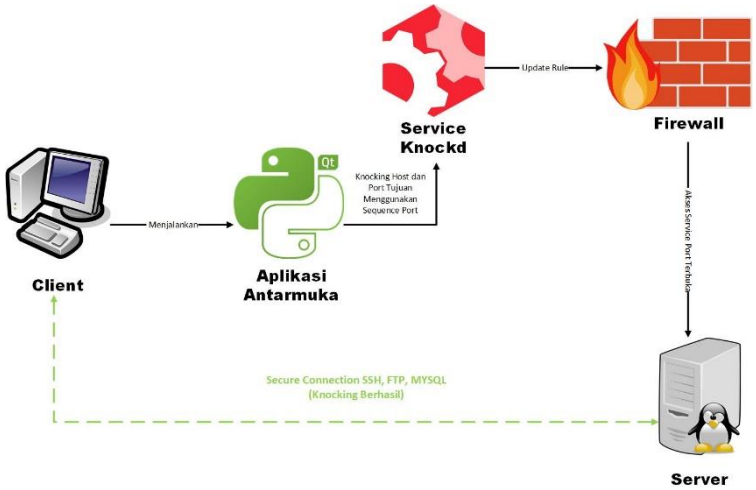
2. Metodologi Penelitian

Metode yang digunakan dalam perancangan aplikasi menggunakan metode *waterfall*, dimana tahapannya dilakukan sebanyak empat tahapan yaitu tahapan analisis, desain, implementasi dan pengujian.

Dalam tahap analisis dilakukan pengumpulan data penelitian, beserta menganalisis kebutuhan fungsional dan non fungsional guna untuk menghasilkan proses bisnis atau tujuan penelitian yang sesuai. Pengumpulan data penelitian didapatkan dari jurnal ilmiah, buku, e-*book*, website resmi dan paper publikasi yang diunduh melalui internet. Analisis kebutuhan fungsional merupakan sebuah analisis yang digunakan dalam menentukan metode yang digunakan untuk perancangan aplikasi pada penelitian. Metode yang digunakan pada penelitian yaitu metode *pseudo random number generator* (PRNG). Metode PRNG digunakan dalam menghasilkan *random number* yang hasilnya digunakan sebagai *sequence port* pada aplikasi. Metode PRNG dikombinasikan dengan nilai *epoctime* (waktu sistem) sebagai acuan *seed* agar hasil *random number* yang digunakan sebagai *sequence port* berbeda setiap harinya. Analisis kebutuhan non fungsional merupakan sebuah analisis yang dibutuhkan dalam melakukan pengujian aplikasi pada penelitian, dimana terdapat dua aspek penting yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak. Kebutuhan perangkat keras yang akan digunakan pada komputer server yaitu : komputer dengan processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz (2 CPUs) ~2.2GHz; RAM 2 GB; serta penyimpanan harddisk sebesar 100GB. Kebutuhan perangkat lunak yang akan digunakan pada komputer *server* yaitu : sistem operasi linux Ubuntu *server* 16.04; Python minimal (v.2.7.12); utility Knockd dan Cron. Kebutuhan perangkat keras yang akan digunakan pada komputer klien yaitu : komputer dengan processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz (12 CPUs) ~2.2GHz; RAM 8 GB; serta penyimpanan harddisk sebesar 1TB. Kebutuhan perangkat lunak yang akan digunakan pada kompuer klien yaitu : sistem operasi Windows 10; Python minimal (v.2.7.12); dan *framework* PyQt4.

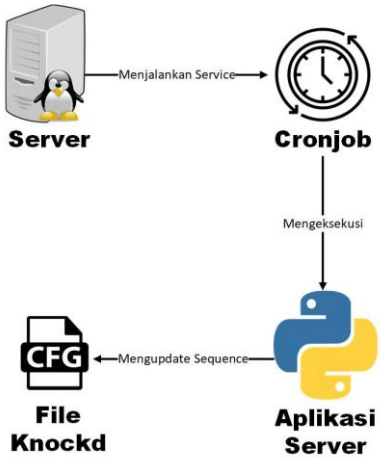
Dalam tahapan desain dilakukan perancangan desain yang menggambarkan gambaran umum aplikasi dan *flowchart* proses (alur) pada aplikasi. Gambaran umum aplikasi *port knocking* yang telah diimplementasikan algoritma PRNG sebagai perlindungan *server* dapat dilihat pada Gambar 1.

p-ISSN: 2252-3006
e-ISSN: 2685-2411



Gambar 1. Gambaran umum aplikasi

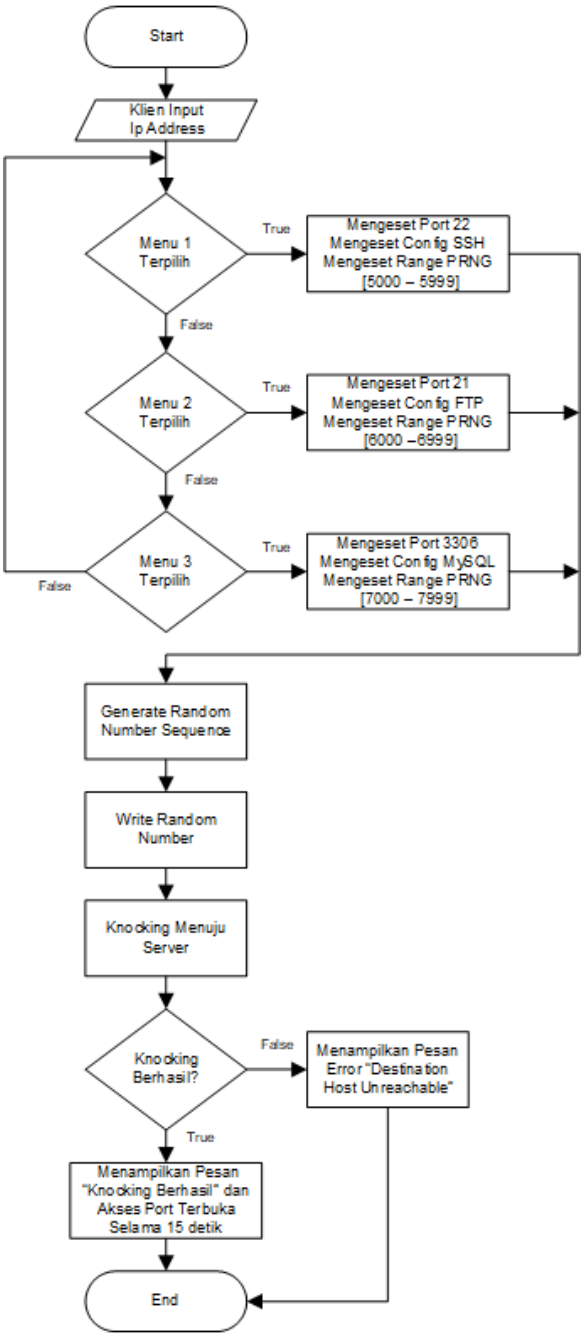
Klien saat ingin melakukan autentikasi dapat menjalankan aplikasi antarmuka. Saat aplikasi antarmuka dijalankan, klien dapat meng-*input host* dan memilih *service port* yang diinginkan untuk dibuka aksesnya. Setelah *host* di-*input* dan akses *service port* dipilih, klien dapat menjalankan proses autentikasi. Aplikasi antarmuka akan melakukan autentikasi menuju *host* dan akses *service port* dengan metode *port knocking* menggunakan *random sequence port* yang dihasilkan menggunakan algoritma PRNG. *Random sequence port* tersebut dibaca oleh *service* Knockd yang telah terpasang di komputer *server*. Jika autentikasi berhasil *menggunakan random sequence port* sesuai, maka *service* Knockd akan menjalankan perintah untuk mengupdate *rule firewall*. *Firewall* akan membuka akses *service port* pada klien yang melakukan autentikasi *port knocking*. Saat akses *service port* terbuka pada komputer *server*, aplikasi antarmuka menampilkan pesan bahwa *service port* yang ingin dibuka aksesnya telah dapat diakses. Klien dapat mengakses *service port* yang telah terbuka aksesnya menggunakan aplikasi *external*. *Secure connection* akan terbentuk antara komputer klien yang telah berhasil melakukan autentikasi dengan akses *service port* pada komputer *server*.



Gambar 2. Gambaran umum Cronjob

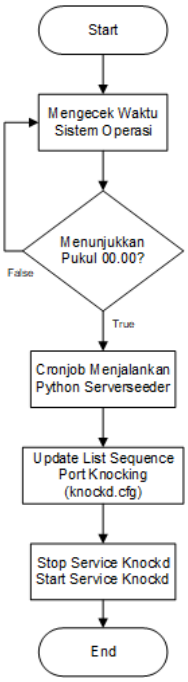
Komputer *server* akan menjalankan *service* cronjob. *Service* cron akan menjalankan *job* sesuai dengan konfigurasi yang telah dilakukan yaitu mengeksekusi aplikasi serverseeder pada pukul 00.00. Saat aplikasi serverseeder dijalankan, hasil *random sequence port* akan diupdate pada file ‘knockd.conf’.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

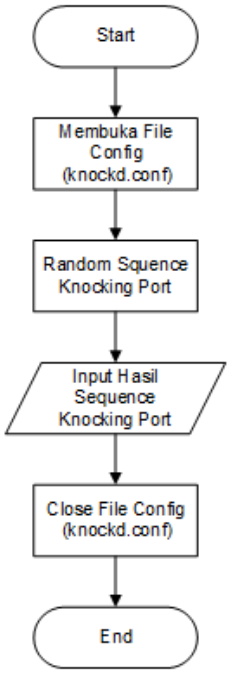


Gambar 3. *Flowchart* aplikasi antarmuka

Alur proses *flowchart* pada Gambar 3 diawali dengan mengambil *inputan ip host* beserta memilih menu *service port* yang akan dibuka aksesnya. Jika menu pertama terpilih, seluruh konfigurasi akan dikhususkan untuk membuka akses *service port* SSH. Jika menu kedua terpilih, seluruh konfigurasi akan dikhususkan untuk membuka akses *service port* FTP. Jika menu ketiga terpilih, seluruh konfigurasi akan dikhususkan untuk membuka akses *service port* MySQL. Proses berikutnya yaitu dilakukan *generate random number sequence*. Hasil *random number sequence* akan disimpan pada file '.cfg'. Proses berikutnya dilakukan autentikasi *port knocking* menuju komputer *server* menggunakan hasil *random number sequence*. Jika autentikasi berhasil, maka *service port* akan terbuka dan pesan *knocking* berhasil akan ditampilkan pada aplikasi antarmuka. Sedangkan jika autentikasi gagal, *service port* tidak akan terbuka dan pesan error akan ditampilkan pada aplikasi antarmuka.



Gambar 4. *Flowchart* Cronjob



Gambar 5. *Flowchart* aplikasi *server*

Alur proses *flowchart* pada Gambar 4 diawali dengan *service* cron akan mengecek waktu sistem operasi pada komputer *server*. Jika waktu menunjukkan pukul 00.00, terdapat proses yang dijalankan yaitu menjalankan aplikasi serverseeder. Sedangkan jika waktu tidak menunjukkan pukul 00.00 maka *service* cron akan menunggu hingga waktu 00.00 tiba. Proses berikutnya hasil *random number sequence* yang dihasilkan saat aplikasi serverseeder dijalankan akan diupdate kedalam file *config* 'knockd.conf'. Proses akhir yaitu *service* Knockd akan direstart. Alur proses *flowchart* pada Gambar 5 diawali dengan aplikasi serverseed akan membuka file *config* 'knockd.conf'. Proses berikutnya dilakukan proses *generate random number sequence*. Hasil *random number sequence* akan di-*input* kedalam file *config* 'knockd.conf'. Proses akhir file *config* 'knockd.conf' akan ditutup.

Dalam tahap *coding* dilakukan pembuatan aplikasi dengan menggunakan bahasa pemrograman sesuai dengan tujuan awal yaitu berbasis Python minimal (v.2.7.12) disertai dengan *framework* PyQt4 (GUI) dalam pembuatan aplikasi klien dan berbasis Python minimal (v.2.7.12) dalam pembuatan aplikasi *server*.

Dalam tahap pengujian dilakukan pengujian autentikasi menggunakan aplikasi yang telah dirancang dalam membuka akses *service port* tertutup. Pengujian selanjutnya dilakukan *generate sequence port* dalam kurun waktu satu tahun dengan tujuan agar dapat mengetahui kemunculan *random number* yang dihasilkan melalui implementasi algoritma PRNG telah sesuai dengan perancangan aplikasi.

3. Kajian Pustaka

Kajian Pustaka merupakan konsep atau teori sebagai dasar studi dalam penelitian yang berkaitan dengan topik yang diteliti. Aplikasi yang dirancang pada penelitian implementasi algoritma PRNG pada aplikasi *port knocking* sebagai perlindungan server dibangun menggunakan komponen sebagai berikut.

3.1. Python

Python merupakan bahasa pemrograman yang memungkinkan pengguna bekerja dengan cepat dan mengintegrasikan sistem dengan lebih efektif. Bahasa pemrograman Python sering digunakan sebagai bahasa pendukung untuk pengembang perangkat lunak, kontrol dan manajemen *build*, pengujian, komputasi ilmiah dan numerik, dan pengembangan web [11].

3.2. PyQt

PyQt merupakan *framework* aplikasi Qt untuk Python yang digunakan dalam membangun aplikasi berbasis antarmuka dan dapat berjalan di semua platform yang didukung oleh Qt yaitu Windows, macOS, Linux, iOS, dan Android [12].

3.3. Knockd

Knockd merupakan aplikasi yang digunakan sebagai *server port-knock*. Knockd mendengarkan semua lalu lintas pada antarmuka ethernet untuk mencari urutan "ketukan" khusus dari *port-hit*. Klien membuat *port-hit* dengan mengirimkan paket TCP (atau UDP) menuju layanan *port* pada *server*. *Port* pada *server* tidak perlu terbuka dikarenakan Knockd mendengarkan di tingkat lapisan *link-layer*. Knockd melihat semua lalu lintas meskipun ditujukan untuk *port* tertutup. Ketika *server* mendeteksi urutan tertentu dari *port-hit*, *server* akan menjalankan perintah yang ditentukan dalam file konfigurasi [13].

3.4. Cron

Cron merupakan utilitas perangkat lunak *daemon* yang digunakan dalam penjadwal pekerjaan berbasis waktu dalam sistem operasi komputer mirip Unix untuk menjalankan aplikasi atau *script* [14].

3.5. PRNG

Bilangan acak (*random number*) merupakan elemen penting dalam ilmu komputer. Pada dasarnya istilah bilangan acak berasal dari *Pseudo-random Number Generator* (PRNG) yang merupakan algoritma perangkat lunak deterministik untuk menciptakan bilangan acak. Bilangan acak digunakan dalam berbagai bagian *cryptography*, *random sampling*, *Monte Carlo method*, *statistics*, *simulation & modeling*, *lottery*, dan *random distribution* [15].

4. Hasil dan Pembahasan

Membahas mengenai hasil implementasi algoritma pada aplikasi, hasil implementasi cronjob dalam menjalankan aplikasi *server* dan hasil *generate sequence port* menggunakan aplikasi selama kurun waktu satu tahun yang digambarkan melalui histogram. Pembahasan aplikasi berupa pengujian autentikasi *port knocking* menggunakan aplikasi klien untuk membuka akses *port* tertutup dan pembahasan pengujian cronjob dalam mengeksekusi aplikasi *server*.

4.1. Implementasi Algoritma Pada Aplikasi dan Cronjob

Implementasi algoritma PRNG akan diimplementasikan pada aplikasi *server* dan aplikasi klien. Terdapat perbedaan implementasi pada aplikasi *server* dan aplikasi klien, dimana perbedaan tersebut yaitu aplikasi klien melakukan *generate sequence port* sesuai dengan pilihan yang ingin dibuka aksesnya pada aplikasi. Sedangkan aplikasi server melakukan *generate sequence port* pada seluruh *service port* yaitu SSH, FTP dan MySQL dalam satu eksekusi. Mekanisme *generate random number* pada kedua aplikasi menggunakan acuan *seed* nilai *epochtime* (waktu sistem) ditambah dengan nilai *random* kunci privat. Hasil *seed* digunakan dalam pembangkitan *random number* sebagai *sequence port*. Hasil *sequence port* pada aplikasi klien akan disimpan pada file *config* sesuai dengan pilihan yang akan dibuka aksesnya, sedangkan pada aplikasi server *sequence port* akan disimpan pada file *config* ‘knockd.conf’.

Tabel 1. Algoritma aplikasi *Port Knocking* klien

Algoritma Aplikasi Klien

Menyiapkan library sys, logging, time, threading, thread, random, knock.py;
Input : host ip address**,** index selectbox, current_time
Output : sequence1, sequence2, sequence3, sequence4
Deklarasi : epoch_time = (current_time, "%d %m %Y")
key1 = random.randint(1000, 1999)
key2 = random.randint(2000, 2999)
key3 = random.randint(3000, 3999)
key4 = random.randint(4000, 4999)
Algoritma :
1. **if** selectbox index == 0 **do**
set random seed (22);
set variabel port = 22;
set file config = 'knockSSH.cfg';
set array range = [5000, 5999]
2. **if** selectbox index == 1 **do**
set random seed (21);
set variabel port = 21;
set file config = 'knockFTP.cfg';
set array range = [6000, 6999]
3. **if** selectbox index == 2 **do**
set random seed (3306);
set variabel port = 3306;
set file config = 'knockMySQL.cfg';
set array range = [7000, 7999]
4. **open file** sesuai dengan index selectbox
random.seed(time.mktime(epoch_time) + key1)
sequence1 = str(random.randint(range[0], range[1]))
random.seed(time.mktime(epoch_time) + key2)
sequence2 = str(random.randint(range[0], range[1]))
random.seed(time.mktime(epoch_time) + key3)
sequence3 = str(random.randint(range[0], range[1]))
random.seed (time.mktime(epoch_time) + key4)
sequence4 = str(random.randint(range[0], range[1]))
5. **write** config
6. **close file**

Tabel 2. Algoritma aplikasi *Port Knocking* server

Algoritma Aplikasi Server
Menyiapkan library random, time, os **Input :** current_time, port[], keyssh[], keyftp[], keysql[] **Output :** ssh['', '', ''], ftp['', '', ''], sql['', '', ''] final['', '', ''] **Deklarasi :** epoch_time = (current_time, "%d %m %Y") **Algoritma :** 1. random.seed(22) 2. **for** x in range(4) **do** 3. **if** x!=3 **do** random.seed(time.mktime(epoch_time) + keyssh[x]) ssh[x] = str(random.randint(portssh[0], portssh[1])) 4. **else** : **do** random.seed (time.mktime(epoch_time) + keyssh[x]) final[0] = str(random.randint(portssh[0], portssh[1])) 5. **end for** 6. random.seed(21) 7. **for** x in range(4) **do** 8. **if** x!=3 **do** random.seed(time.mktime(epoch_time) + keyftp[x]) ftp[x] = str(random.randint(portftp[0], portftp[1])) 9. **else** : **do** random.seed (time.mktime(epoch_time) + keyftp[x]) final[1] = str(random.randint(portftp[0], portftp[1])) 10. **end for** 11. random.seed(3306) 12. **for** x in range(4) **do** 13. **if** x!=3 **do** random.seed(time.mktime(epoch_time) + keysql[x]) sql[x] = str(random.randint(portsql[0], portsql[1])) 14. **else** : **do** random.seed (time.mktime(epoch_time) + keysql[x]) final[2] = str(random.randint(portsql[0], portsql[1])) 15. **end for** 16. **write** config 17. **restart** service **knockd** 18. **start** service **knockd** (verbose)

Implementasi cronjob sebagai pendukung dalam melakukan update *sequence port knocking* jika hari telah berganti. Implementasi cronjob digunakan dalam mengeksekusi aplikasi pada waktu yang telah ditentukan. Pada penelitian ini, cronjob diberikan tugas untuk mengeksekusi aplikasi 'serverseeder.py' pada pukul 00.00 dan hasil *log* eksekusi aplikasi disimpan pada lokasi '/home/server/' dengan nama file 'cron.log'.

```
 GNU nano 2.5.3              File: /tmp/crontab.LGHH3O/crontab

# Menjalankan python setiap jam 00.00 (midnight)
0 0 * * * cd /home/server && /usr/bin/python2.7 serverseed.py >> /home/server/cron.log 2>&1
```

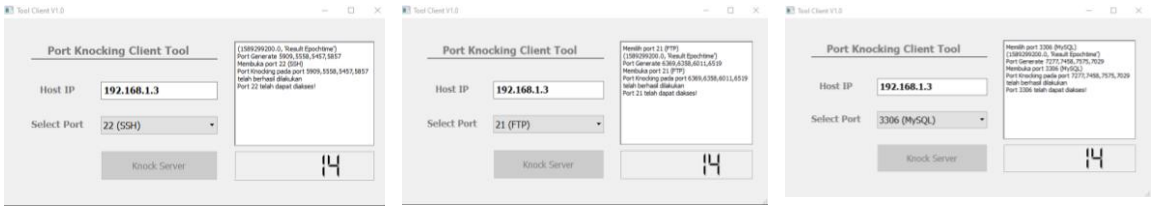
Gambar 6. Konfigurasi Cronjob

4.2. Pengujian Aplikasi dan Pengujian Cronjob

Pengujian aplikasi dilakukan pada seluruh akses *service port* yang telah diimplementasikan dengan metode autentikasi *port knocking*. Pengujian pertama pada Gambar 7 (a) dilakukan untuk autentikasi dalam membuka akses *service port* SSH. Pengujian kedua pada Gambar 7 (b) dilakukan untuk autentikasi dalam membuka akses *service port* FTP. Pengujian ketiga pada Gambar 7 (c) dilakukan untuk autentikasi dalam membuka akses *service port* MySQL. Hasil *sequence port* dihasilkan oleh aplikasi klien akan sesuai dengan *sequence port* yang dihasilkan oleh aplikasi *server*, selain itu *sequence port* akan berubah jika hari telah

p-ISSN: 2252-3006
e-ISSN: 2685-2411

berganti. Masing – masing akses *service port* menggunakan *sequence port* yang berbeda yaitu *service port* SSH menggunakan *sequence port* dengan *range port* 5000-5999, *service port* FTP menggunakan *sequence port* dengan *range port* 6000-6999, dan *service port* MySQL menggunakan *sequence port* dengan *range port* 7000-7999.



(a) (b) (c)

Gambar 7. (a) Pengujian autentikasi SSH, (b) Pengujian autentikasi FTP, (c) Pengujian autentikasi MySQL

Pengujian cronjob dilakukan dengan cara mengecek status cron saat *job* akan dieksekusi sesuai dengan waktu yang telah ditentukan. Status cronjob dicek pada komputer *server* menggunakan perintah 'service cron status'. Saat hari berganti, dapat dilihat pada Gambar 8 *service* cron mengeksekusi aplikasi 'serverseeder.py' pada pukul 00.00.01. Status ini menandakan bahwa implementasi cronjob telah sesuai dalam menjalankan tugas untuk melakukan update pada *sequence port knocking* komputer *server*.

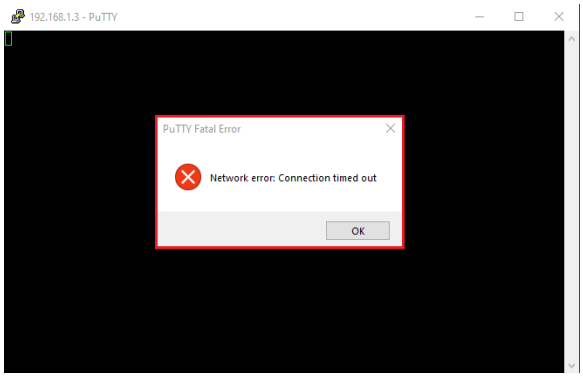
```
root@Server:/home/server# service cron status
• cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-09-02 00:00:00 +00; 12s ago
     Docs: man:cron(8)
 Main PID: 1506 (cron)
    Tasks: 4
   Memory: 7.4M
      CPU: 41ms
   CGroup: /system.slice/cron.service
           ├─1506 /usr/sbin/cron -f
           ├─1557 /usr/sbin/CRON -f
           ├─1558 /bin/sh -c cd /home/server && /usr/bin/python2.7 serverseed.py >> /home/server/cr
           └─1559 /usr/bin/python2.7 serverseed.py

Sep 02 00:00:00 Server systemd[1]: Started Regular background program processing daemon.
Sep 02 00:00:00 Server cron[1506]: (CRON) INFO (pidfile fd = 3)
Sep 02 00:00:00 Server cron[1506]: (CRON) INFO (Skipping @reboot jobs -- not system startup)
Sep 02 00:00:01 Server CRON[1557]: pam_unix(cron:session): session opened for user root by (uid=0)
Sep 02 00:00:01 Server CRON[1558]: (root) CMD (cd /home/server && /usr/bin/python2.7 serverseed.py
```

Gambar 8. Status Cronjob

4.3. Hasil Pengujian Autentikasi Port Knocking

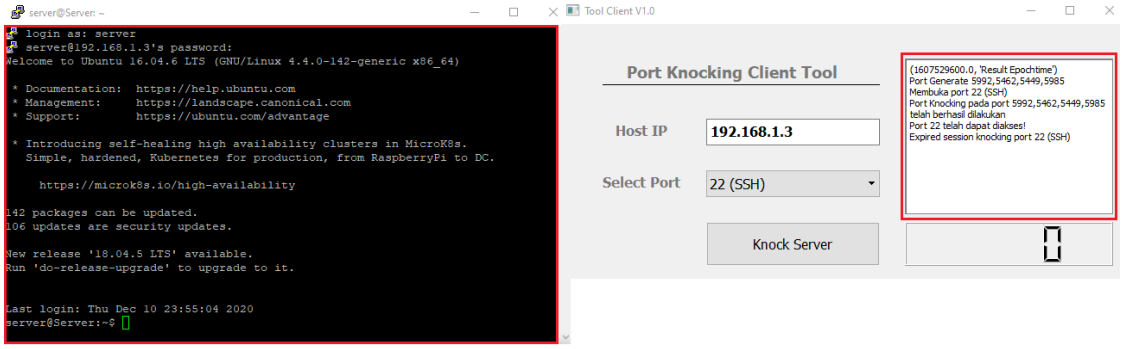
Hasil pengujian autentikasi didapatkan melalui skenario pengujian yang dilakukan dengan dua tahapan yaitu tahap pengujian mengakses *service port* tertutup tanpa proses autentikasi dimana tampilannya dapat dilihat pada Gambar 9 dan tahap pengujian mengakses *service port* tertutup dengan proses autentikasi dimana tampilannya dapat dilihat pada Gambar 10.



Gambar 9. Tanpa proses autentikasi

p-ISSN: 2252-3006
e-ISSN: 2685-2411

Hasil tahap pengujian mengakses *service port* tertutup tanpa proses autentikasi pada Gambar 9 yaitu komputer klien tidak dapat masuk kedalam *service port* tertutup secara langsung (*direct access*) dikarenakan aksesnya ditolak oleh *firewall*.



Gambar 10. Dengan proses autentikasi

Hasil tahap pengujian mengakses *service port* tertutup dengan proses autentikasi pada Gambar 10 yaitu komputer klien dapat masuk kedalam *service port* tertutup jika proses autentikasi menggunakan aplikasi telah sukses dikarenakan aksesnya dibuka oleh *firewall* saat proses autentikasi sesuai.

4.4. Hasil Sequence Aplikasi dan Cronjob

Hasil *sequence* didapatkan dari perhitungan nilai *seed* yang digunakan dalam membangkitkan *sequence port* pada aplikasi. Seluruh *service port* yang telah diimplementasikan dengan metode *port knocking* menggunakan perhitungan nilai *epochtime* (waktu sistem) ditambah dengan nilai kunci privat untuk memperoleh *final seed*, dimana nilai *final seed* digunakan dalam pembangkitan *sequence port* menggunakan implementasi algoritma PRNG. Empat urutan *sequence* menggunakan kunci privat yang berbeda dengan tujuan agar urutan *sequence port* yang dihasilkan tidak sama.

Tabel 3. Perhitungan nilai *Seed* SSH

No	Epochtime		Key		Final seed
1	1598976000.0	+	1958	=	1598977958.0
2	1598976000.0	+	2140	=	1598978140.0
3	1598976000.0	+	3023	=	1598979023.0
4	1598976000.0	+	4998	=	1598980998.0

Tabel 4. Perhitungan nilai *Seed* FTP

No	Epochtime		Key		Final seed
1	1598976000.0	+	1164	=	1598977164.0
2	1598976000.0	+	2689	=	1598978689.0
3	1598976000.0	+	3634	=	1598979634.0
4	1598976000.0	+	4479	=	1598980479.0

Tabel 5. Perhitungan nilai *Seed* MySQL

No	Epochtime		Key		Final seed
1	1598976000.0	+	1676	=	1598977676.0
2	1598976000.0	+	2344	=	1598978344.0
3	1598976000.0	+	3113	=	1598979113.0
4	1598976000.0	+	4907	=	1598980907.0

Hasil perhitungan nilai *final seed* SSH pada Tabel 6 digunakan dalam pembangkitan nilai *sequence port* untuk membuka akses *service port* SSH. Hasil *sequence port* tersebut yaitu *seed* pertama menghasilkan *sequence port* 5483, *seed* kedua menghasilkan *sequence port* 5073, *seed* ketiga menghasilkan *sequence port* 5046, dan seed keempat menghasilkan *sequence port* 5841. Empat urutan *sequence port* 5483,5073,5046,5841 digunakan dalam membuka akses *service port* SSH.

Tabel 6. Hasil *Sequence* SSH

No	Final seed		Hasil Sequence
1	random.seed (1598977958.0)	=	5483

p-ISSN: 2252-3006
e-ISSN: 2685-2411

2	random.seed (1598978140.0)	=	5073
3	random.seed (1598979023.0)	=	5046
4	random.seed (1598980998.0)	=	5841
Hasil Sequence :	5483,5073,5046,5841		

Hasil perhitungan nilai *final seed* FTP pada Tabel 7 digunakan dalam pembangkitan nilai *sequence port* untuk membuka akses *service port* FTP. Hasil *sequence port* tersebut yaitu *seed* pertama menghasilkan *sequence port* 6150, *seed* kedua menghasilkan *sequence port* 6153, *seed* ketiga menghasilkan *sequence port* 6480, dan *seed* keempat menghasilkan *sequence port* 6659. Empat urutan *sequence port* 6150,6153,6480,6659 digunakan dalam membuka akses *service port* FTP.

Tabel 7. Hasil *Sequence* FTP

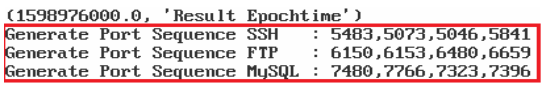
No	Final seed	Hasil Sequence	
1	random.seed (1598977164.0)	=	6150
2	random.seed (1598978689.0)	=	6153
3	random.seed (1598979634.0)	=	6480
4	random.seed (1598980479.0)	=	6659
Hasil Sequence :	6150,6153,6480,6659		

Hasil perhitungan nilai *final seed* MySQL pada Tabel 8 digunakan dalam pembangkitan nilai *sequence port* untuk membuka akses *service port* MySQL. Hasil *sequence port* tersebut yaitu *seed* pertama menghasilkan *sequence port* 7480, *seed* kedua menghasilkan *sequence port* 7766, *seed* ketiga menghasilkan *sequence port* 7323, dan *seed* keempat menghasilkan *sequence port* 7396. Empat urutan *sequence port* 7480,7766,7323,7396 digunakan dalam membuka akses service port MySQL.

Tabel 8. Hasil *Sequence* MySQL

No	Final seed	Hasil Sequence	
1	random.seed (1598977676.0)	=	7480
2	random.seed (1598978344.0)	=	7766
3	random.seed (1598979113.0)	=	7323
4	random.seed (1598980907.0)	=	7396
Hasil Sequence :	7480,7766,7323,7396		

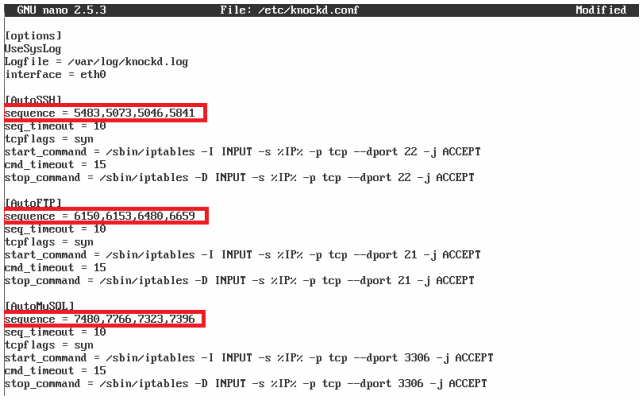
Hasil implementasi cronjob diperoleh dari *log service* cron dalam melakukan eksekusi aplikasi *server* yang tampilannya dapat dilihat pada Gambar 11. Saat file *log* cronjob dibuka, terdapat hasil eksekusi aplikasi *server* dalam melakukan *generate random number*. Proses yang tercatat saat *service* cron mengeksekusi aplikasi *server* yaitu aplikasi akan melakukan *generate sequence port knocking* pada seluruh akses *service port* yang diimplementasikan dengam metode *port knocking* yaitu pada *service port* SSH, FTP dan MySQL.



Gambar 11. Log eksekusi Cronjob

Hasil *generate sequence port knocking* diupdate kedalam file *config* 'knockd.conf' dimana tampilannya pada Gambar 12. Saat *sequence port* telah diupdate, aplikasi *server* akan melakukan *restart service* Knockd agar *sequence port* yang diupdate dapat digunakan dalam proses autentikasi membuka akses *service port*.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

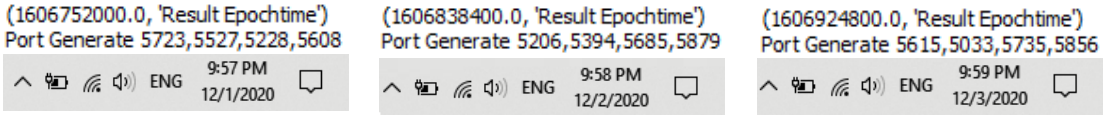


Gambar 12. Hasil eksekusi Cronjob

Hasil *sequence* SSH akan diupdate pada bagian [AutoSSH], Hasil *sequence* FTP akan diupdate pada bagian [AutoFTP], dan Hasil *sequence* MySQL akan diupdate pada bagian [AutoMySQL].

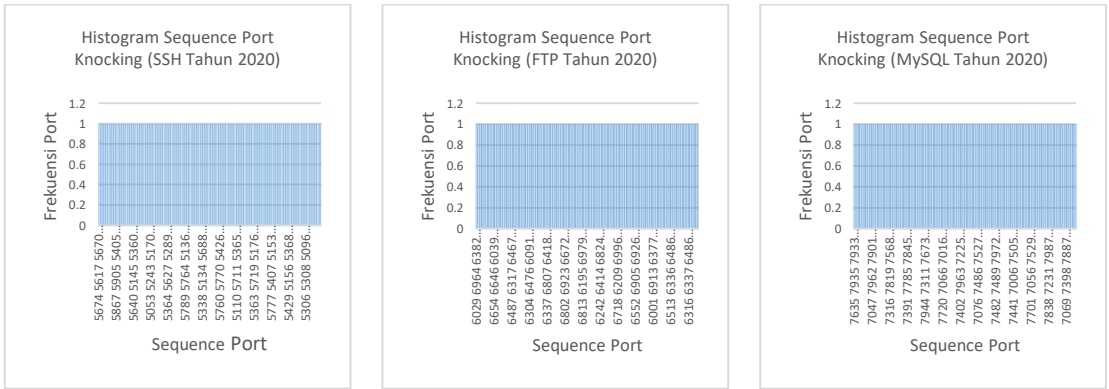
4.5. Hasil Generate Sequence Port Tahun 2020

Hasil *generate sequence port* didapatkan melalui pengumpulan data *generate* dalam kurun waktu selama satu tahun agar dapat digambarkan melalui histogram. Frekuensi kemunculan *random number* yang dihasilkan berbeda setiap harinya dikarenakan nilai *epochtime* yang digunakan untuk membangkitkan *random number* selalu berubah, beserta nilai kunci privat yang digunakan pada setiap akses *service port* berbeda.



(a) (b) (c)

Gambar 13. (a) Hasil *sequence* SSH hari pertama, (b) Hasil *sequence* SSH hari kedua, (c) Hasil *sequence* SSH hari ketiga.



(a) (b) (c)

Gambar 14. (a) Histogram *sequence* SSH, (b) Histogram *sequence* FTP, (c) Histogram *sequence* MySQL

Histogram pada Gambar 14 (a), (b), dan (c) menyatakan bahwa selama kurun waktu satu tahun, *random number sequence* yang digunakan dalam proses autentikasi tidak adanya duplikasi. Implementasi algoritma PRNG pada aplikasi yang dirancang telah sesuai dengan

tujuan penelitian agar proses autentikasi menggunakan *random number sequence* yang berbeda setiap harinya.

5. Kesimpulan

Berdasarkan kegiatan penelitian yang telah penulis lakukan, aplikasi yang dirancang pada penelitian dapat memperkuat proses autentikasi dalam memvalidasi komputer klien yang akan terhubung dengan sistem server menggunakan *random number sequence port*. *Random number sequence port* diperoleh dari implementasi algoritma PRNG yang mana pembangkitan nilai *random* menggunakan acuan *seed* nilai *epochtime* (waktu sistem) ditambah dengan kunci privat. Selain itu, aplikasi pada klien dibangun berbasis GUI menggunakan *framework* PyQT4 dengan tujuan untuk mempermudah pengguna dalam melakukan autentikasi. Dalam pengumpulan data *generate* selama kurun waktu satu tahun, *random number sequence port* yang digunakan untuk autentikasi pada masing - masing akses *service port* tidak terdapat duplikasi.

Daftar Pustaka

[1] S. B. Sinaga, “Pengamanan Pesan Komunikasi Menggunakan Algoritma Rsa, Rabbin Miller Dan Fungsi Sha-1 Serta Penanganan Man In The Middle Attack Dengan Interlock Protocol”, *Jurnal Teknik Informatika Unika St. Thomas (JTIUST)*, vol. 3 no. 1, pp. 64-71, 2018.

[2] I. G. A. S. Sanjaya, G. M. A. Sasmita, and D. M. S. Arsa, "Evaluasi Keamanan Website Lembaga X Melalui Penetration Testing Menggunakan Framework ISSAF", *Jurnal Ilmiah Merpati,* vol. 8, no. 2, pp. 113-124, 2020.

[3] T. Shubh, and S. Sharma, “Man-In-The-Middle-Attack Prevention Using HTTPS and SSL”, *International Journal of Computer Science and Mobile Computing,* vol. 5 no. 6, pp. 569-579, 2016.

[4] E. A. Darmadi, “Perancangan Sistem Otentikasi Radius pada Pengguna Jaringan Wireless untuk Meningkatkan Keamanan Jaringan Komputer”, *Jurnal IKRA-ITH Informatika*, vol. 2 no. 3, pp. 9-16, 2018.

[5] A. S. Andreatos, “Hiding the SSH port via smart Port Knocking”, *International Journal of Computers,* vol. 11, pp. 28-31, 2017.

[6] K. M. U. Maheswari, R. Kundu, and H. Saxena, “Pseudo Random Number Generators Algorithms and Applications”, *International Journal of Pure and Applied Mathematics,* vol. 118, no. 22, pp. 331-336, 2018.

[7] P. B. Stark and K. Ottoboni, “Random Sampling: Practice Makes Imperfect”, *Journal arXiv*, pp. 1-10, 2018.

[8] M. Z. A. Mahmud, Syaifuddin, and D. Risqiwati, “Implementasi Authentication System Pada Port Knocking Ubuntu Server Menggunakan Knockd Dan Python”, *Jurnal SISTEMASI,* vol. 7, no. 3, pp. 169-175, 2018.

[9] I. Marzuki, “Perancangan dan Implementasi Sistem Keamanan Jaringan Komputer Menggunakan Metode Port Knocking Pada Sistem Operasi Linux”, *Jurnal Teknologi Informasi Indonesia,* vol. 2, no. 2, pp. 18-24, 2017.

[10] S. Khadafi, S. Nurmuslimah, and F. K. Anggakusuma, “Implementasi Firewall Dan Port Knocking Sebagai Keamanan Data Transfer Pada Ftp Server Berbasiskan Linux Ubuntu Server”, *Jurnal Ilmiah Nero*, vol. 4, no. 3, pp. 181-188, 2019.

[11] https://www.python.org/about/apps, diakses tanggal 04 Oktober 2020.

[12] https://riverbankcomputing.com/software/pyqt, diakses tanggal 04 Oktober 2020.

[13] https://linux.die.net/man/1/knockd, diakses tanggal 04 Oktober 2020.

[14] https://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html, diakses tanggal 04 Oktober 2020.

[15] T. Ahmed, and M. M. Rahman, "The Hybrid Pseudo Random Number Generator", *International Journal of Hybrid Information Technology*, vol. 9, no. 7, pp. 299-312, 2016.