IMPLEMENTASI AUTHENTICATION SYSTEM PADA PORT KNOCKING UBUNTU SERVER MENGGUNAKAN KNOCKD DAN PYTHON

[1]Mukhammad Zainal Amir Mahmud, [2]Syaifuddin, [3]Diah Risqiwati
[1,2,3]Fakultas Teknik Informatika, Universitas Muhammadiyah Malang
Jalan Raya Tlogomas No. 246, Tlogomas, Malang
Email: Zainal0909@gmail.com , udinsaif@gmail.com , diahrisqiwati@gmail.com

ABSTRAK

Keamanan yang baik untuk sebuah *server* adalah menonaktifkan layanan yang sedang tidak digunakan untuk mencegah serangan pada *server*. Namun masih banyak ditemukan kasus yang menyerang *server*, umumnya serangan dilakukan oleh penyerang dengan menggali informasi detail dari calon korbannya. Serangan yang dilakukan pada *server* kebanyakan dari layanan *port* yang terbuka dari layanan *server*, teknik yang sering dilakukan oleh penyerang yaitu *Scanning* dan *bruteforce*. Untuk tujuan ini, digunakan sebuah metode *port knocking* yang dikembangkan. *Port knocking* berfungsi untuk menyembunyikan layanan *server* sampai urutan ketukan dilakukan. Perangkat *port knocking* pada sistem operasi linux adalah knockd. Knockd berperan mendengarkan ketukan semua lalu lintas pada *Ethernet*, mendengarkan urutan ketukan khusus sampai sesuai. Akan tetapi metode *port knocking* masih memiliki beberapa kelemahan seperti TCP replay. *port scanning* dan lain-lain. Penelitian ini mengusulkan metode menggunakan *authentication* untuk meningkatkan keamanan, *client* harus mempunyai file yang digunakan *authentication* pada *server* sehingga memperoleh hak untuk mengakses layanan yang diperlukan dari *server*.

Kata Kunci: port knocking; authentication; Knockd

1 PENDAHULUAN

Selama bertahun-tahun keamanan jaringan menjadi salah satu perhatian peneliti untuk mengembangkan sistem keamanan jaringan yang aman. Internet merupakan sebuah kumpulan *node* pada suatu jaringan yang lain dan terhubung untuk saling menukar informasi atau memberikan layanan yang dibutuhkan. Masalah yang terjadi pada keamanan jaringan adalah bagaimana melindungi sebuah sistem yang saling terhubung agar aman pada saat berkomunikasi, dan pada saat sistem berjalan dapat diakses oleh seorang administrator secara *online* untuk memantau sistem tersebut[1]. Salah satu cara untuk membatasi akses sebuah sistem dalam suatu jaringan adalah dengan mengharuskan pengguna sistem tersebut mengautentikasikan dirinya pada saat sistem sebelum memberikan sebuah layanan, akan tetapi masih banyak kasus serangan pada suatu jaringan[2].

Hal pertama yang dilakukan penyerang adalah mencari informasi lengkap tentang korbanya atau *vulnerability* pada sistem tersebut, seperti *service* yang berjalan, *port* penting yang terbuka dan versi dari software untuk menemukan kelemahan yang belum di-*Patch* atau bug dari software tersebut[11]. *Port Knocking* adalah sebuah metode yang menyembunyikan *service* dari *attacker* dengan cara mentransmisikan data melalui port yang tertutup dan memberikan *listen por*t yang sudah ditentukan. Pada dasarnya *Port Knocking* merupakan sebuah mekanisme keamanan jaringan dalam *Firewall*[3]. Pada referensi lain mengatakan bahwa *Port Knocking* adalah sebuah metode otorisasi user berdasarkan *firewall* untuk melakukan komunikasi melalui port yang tertutup[4] [10]. Serangan yang umum terjadi antara lain adalah *DDoS* dan *NAT* knocking, *DDoS* pada *port knocking* menyebabkan server terlalu sibuk memproses paket dari *client* yang melakukan serangan sehingga server menjadi sangat terbebani dan akhirnya layanan pada server tersebut menjadi terganggu, *DDoS* ini menekankan pada pengiriman paket *SYN* secara terus menerus dari *client* dengan pemalsuan *IP Address* dari *client* tersebut sehingga server akan sibuk mengirimkan paket *SYN,ACK* dan pada *client* yang menyerang server tersebut paket *ACK* yang seharusnya dikirim kembali ke server tidak dikirimkan sebagaimana mestinya[5]. *NAT knocking* adalah serangan yang memungkinkan *client* yang tidak valid dapat memiliki akses ke suatu sistem, konsep serangan ini adalah saat ada *client* yang berada dibalik *NAT* menggunakan *port knocking* untuk masuk ke suatu sistem maka si penyerang akan menunggu sampai *client* tersebut menyelesaikan

*sequence port knocking*nya sampai port yang menjadi tujuan *client* tersebut terbuka, dari sini penyerang akan mencoba masuk ke port tersebut tanpa melakukan knocking karena client yang sebelumnya melakukan knocking telah membuka akses ke port, bila si penyerang dapat masuk ke port tersebut maka pada server yang menjalankan *port knocking* dipastikan tidak memvalidasi *client* yang terhubung dengannya[9].

Pada penelitian yang dilakukan oleh Leleh Boroumand yang berjudul *"Virtualization Technique for Port Knockingin Mobile Cloud Computing"* memberikan saran agar *timeout sequence* yang digunakan memiliki banyak variasi dikarenakan celah ini memberikan kesempatan bagi seorang penyerang untuk menemukan *sequence* yang tepat[6]. Pada penelitian yang dilakukan oleh Mehran Pourvahab yang berjudul*"Secure Port Knocking-Tunneling"* menyatakan bahwa pemanfaatan *tunneling* pada penelitiannya memberikan pengamanan dalam komunikasi dan mencegah *NAT-knocking* dan untuk pengembangannya kedepan diperlukan *sequence* dan model enkripsi yang lebih baik guna mengurangi kinerja berlebih pada server[7]. pada dasarnya metode *port knocking* ini masih rentang terhadap serangan *sniffing*, namun yang membedakan dari penelitian sebelumnya yaitu tingkat kesulitan untuk memecahkan *payload* yang didapat dari hasil *sniffing*.

Pada penelitian ini peneliti akan mencoba mengimplementasikan autentikasi *client port knocking* pada ubuntu server versi 16. *Authentication* ini dilakukan *client* pada *server* yang bertujuan untuk memvalidasi *client* yang melakukan *knocking* ke server, guna menghindari serangan DOS *knocking*. Dan bertujuan meningkatkan keamanan *port knocking*.

2 TINJAUAN PUSTAKA

2.1 Port knocking

Port knocking memungkinkan terjadinya komunikasi antar *host* melalui *port* yang tertutup secara umum terdapat berbagai metode yang bisa diterapakan dalam *port knocking*, informasi yang dikirmkan dapat berupa *sequence* ataupun *payload* yang berisi data, dalam prakteknya data yang dikirm dalam bentuk *payload* akan diterima oleh daemon yang bertugas menunggu dan mengobservasi *sequnce* yang dikirm oleh client ataupun isi *payload* yang dikirm oleh *client*. Dalam *port knocking*, server yang bertindak sebagai penyedia layanan *port knocking* tidak membuka *port* yang akan digunakan untuk komunikasi, melainkan menyediakan listener terhadap paket SYN pada *port* tertentu sehingga ketika *port* yang diberi *listener* tersebut menerima paket SYN dari client dapat digunakan sebagai acuan suatu autentikasi berdasarkan *sequnce*. Sebagai contoh disediakan *port* (1145,1087,1172) yang bertugas menerima SYN dari *client*, ketika *client* berhasil mengirim SYN ke *port-port* tersebut dalam interval waktu tertentu maka dapat diasumsikan *client* tersebut valid karena informasi *knock sequence* ini bersifat sangat rahasia dan hanya diketahui oleh admin dan *client* saja[8].

2.2 Analisa Kebutuhan

port knocking merupakan suatu metode autentikasi pada *port* tertutup yang memungkinkan program pada suatu *port* dapat tersembunyi dan dapat dimunculkan ketika *client* yang terautentikasi memberikan *sequence* tertentu pada port-port yang telah ditentukan sehingga client tersebut dianggap valid dan diberi izin mengakses *port* dan layanan yang sebelumnya disembunyikan.

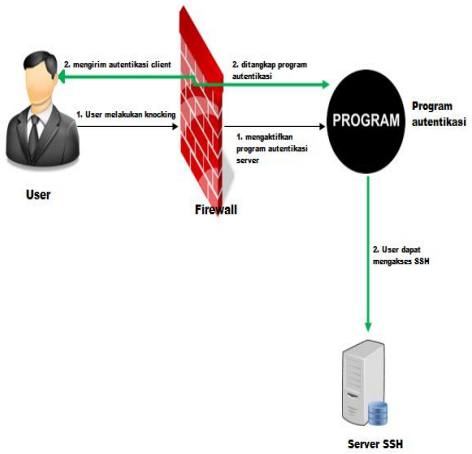
3 METODE PENELITIAN

Port knocking memungkinkan terjadinya komunikasi antar *host* melalui *port* yang tertutup secara umum terdapat berbagai metode yang bisa diterapakan dalam *port knocking*, informasi yang dikirmkan dapat berupa *sequence* ataupun *payload* yang berisi data, dalam prakteknya data yang dikirm dalam bentuk *payload* akan diterima oleh *daemon* yang bertugas menunggu dan mengobservasi *sequence* yang dikirm oleh *client* ataupun isi *payload* yang dikirm oleh *client*. Dalam *port knocking*, server yang bertindak sebagai penyedia layanan *port knocking* tidak membuka *port* yang akan digunakan untuk komunikasi, melainkan menyediakan *listener* terhadap paket SYN pada *port* tertentu sehingga ketika *port* yang diberi *listener* tersebut menerima paket SYN dari client dapat digunakan sebagai acuan suatu autentikasi berdasarkan *sequnce*. Sebagai contoh disediakan *port* (1145,1087,1172) yang bertugas menerima SYN dari *client*, ketika *client* berhasil mengirim SYN ke *port-port* tersebut dalam interval

waktu tertentu maka dapat diasumsikan *client* tersebut valid karena informasi *knock sequence* ini bersifat sangat rahasia dan hanya diketahui oleh admin dan *client* saja[8].

3.1 Skenario Program

Skenario program yang dirancang pada penelitian ini. Bagaimana program berjalan pada client dan server.



Gambar 3. 1 *Skenario program*.

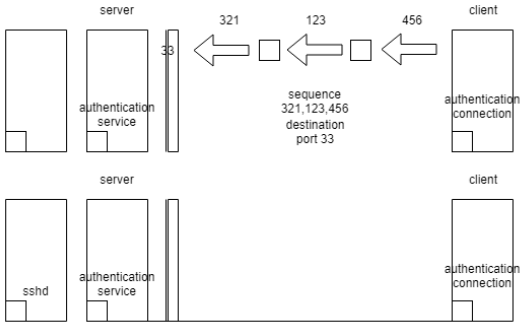
Skenario program yang dirancang dalam penelitian ini:

1. User melakukan *knocking* pada *server* dengan menggunakan *sequence* yang telah ditentukan.
2. Setelah user melakukan *knocking*, *firewall* akan meneruskan ke *server* dan mengaktifkan program autentikasi *server*.
3. *User* mengirimkan program autentikasi yang ada pada client.
4. Autentikasi berhasil *user* dapat memperoleh akses layanan ke *server* untuk mengakses layanan yang sebelunnya tertutup.

Metode *port knocking* pada penelitian ini terdapat dua proses yang digunakan untuk mengamankan oterasi *client*, yang pertama menentukan jumlah *sequence* yang digunakan dalam *knocking* dan yang kedua *authentication client* pada *server*.

Proses pertama adalah menentukan jumlah *sequence* dan *destination port* yang digunakan dalam proses *knocking*, jika proses *knocking* berhasil maka *destination port* terbuka dan *firewall* akan melakukan *listen port* yang sudah ditentukan tersebut sampai ada *authentication* dari klien.

Proses kedua adalah *authentication* dari *client*, *client* harus dapat mengautentikasikan diri sehingga proses dapat berlanjut, setelah *client* dapat mengautentikasikan dirinya, *client* dapat menggunakan layanan yang dibutuhkan, proses ini digunakan untuk mencegah serangan *replay attack* dan *Dos-Knocking*. Untuk *authentication* ini menggunakan program yang diletakan di *server* dan *client*, *client* harus mengaktifkan program sehingga dapat melakukan *authentication* pada *server*, sebaliknya pada *server* program *automatic* ketika *knocking* yang dilakukan *client* berhasil.



Gambar3. 2 *Port knocking authentication*

Dari gambar 3.2 diatas terlihat *client* melakukan *knocking* dengan *sequence* 321, 123, 356 dengan *destination port* 33. Bila berhasil melakukan *knocking* maka *authentication service* akan berjalan

otomatis ketika *destination port* 33 telah terbuka, dan server menunggu *client* mengautentikasikan dirinya pada *server* sehingga *client* dapat diizinkan mengakses sshd pada *server*, bila tidak paket akan di drop sehingga dapat mencegah serangan ke *server*. *Client* yang sudah tervalidasi dapat menggunakan layanan sshd yang ada *server*, dengan cara mengautentikasikan dirinya pada *server*. Autentikasi ini menggunakan program *socket client-server*. Metode pengambilan data pada penelitian ini dengan cara membandingakan jumlah *sequence* dan waktu yang dibutuhkan untuk melakukan *knocking*. Metode pengambilan data selanjutnya melakukan *bruteforce* pada *fake random port.*

4 HASIL DAN PEMBAHASAN

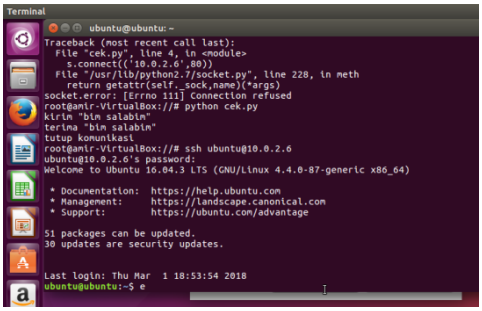
Pada penelitian ini metode *port kncoking authentication* ini di rancang pada ubuntu *server* 16.00, dan ubuntu *desktop* 16.00 yang dijadikan sebagai *client*. Pada *server* terinstall knockd dan melakukan konfigurasi knockd, knockd merupakan suatu fitur yang harus di *install* terlebih dahulu pada linux, knockd merupakan fitur pada linux yang berguna sebagai metode *port knocking*. Program autentikasi *server* di buat menggunakan bahasa pemrograman *python*. *Python* juga harus di *install* lebih dulu.

Pada *client* hanya membutuhkan program autentikasi *client*, yang berguna untuk melakukan autentikasi pada *server* pada saat selesai melakukan *kncoking*. *Client* ini juga bertugas meminta layanan yang telah ditutup *port*nya oleh *server* untuk dibuka. Namun hanya *client* yang valid saja yang dapat menggunakan layanan tersbut.

Pada program yang dipakai untuk melakukan autentikasi *client* dan *server* ini menggunakan komunikasi socket *client* dan *server*, yang diterapkan ke dalam bahasa pemrograman *python*.

Gambar 4.1 *Perintah Knocking*

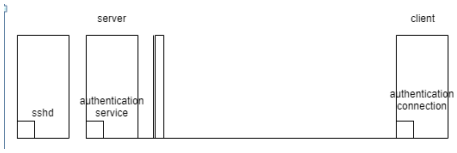
Pada gambar 4.1 merupakan cara mengetuk *client* pada *server* dengan menggunakan bahasa pemrograman *shell*.



Gambar 4.2 *Akses SSH server*

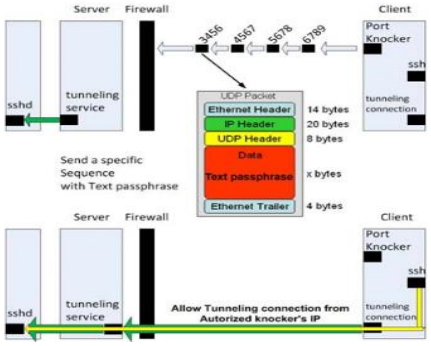
Pada gambar 4.2 menjelaskan client dapat mengakses SSH *server* setelah melakukan *knocking* dan autentikasi dengan benar.

Metode *port knocking authentication* ini dapat mengatasi masalah *bruteforce* yang terjadi dengan melakukan otentikasi *client* pada *server*, ketika proses *knocking* benar. Penyerang tidak dapat mengakses layanan ssh jika mereka tidak dapat mengautentikasikan diri mereka sebagai *client* yang sah.



Gambar 4.3 *Port knocking authentication*

Bila *client* telah selesai melakukan *knocking* dengan benar, *client* tersebut diwajibkan mengautentikasikan dirinya agar dapat mengakses ssh pada server, seperti gambar diatas. Sehingga jika penyerang berhasil menemukan kombinasi *sequence* yang benar, penyerang diwajibkan melakukan autentikasi dirinya. Namun jika tidak bisa mengautentikasikan dirinya, maka penyerangan hanya dapat menemukan kombinasi *sequence* pada *port knocking* saja, namun tidak bisa mengakses ssh pada *server*. Selain meningkatkan sisi keamanan, penelitian ini juga diharapkan dapat membuat proses *port knocking* ini menjadi lebih sederhana, dan membutuhkan waktu yang lebih sedikit dalam proses otorisasinya.



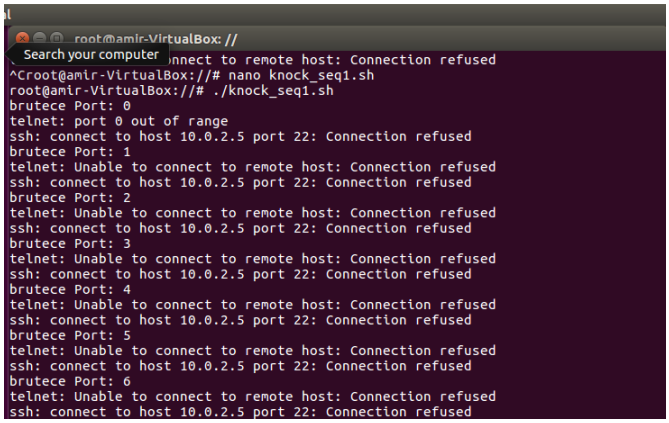
Gambar 4.4 *port knocking dengan source port*
sumber : SPKT: Secure Port Knock-Tunneling, an Enhanced Port Security Authentication Mechanism.

Terlihat pada gambar 4.4 diatas, membutuhkan *tunneling* setelah *client* dapat menemukan kombinasi *sequence*, dimana untuk membuat *tunneling* membutuhkan *resource* yang besar dan aplikasi penunjang untuk membangun *tunneling*. sehingga membutuhkan waktu yang lebih lama dalam proses *tunneling* tersebut, Untuk dapat mengakses layanan ssh. Berbeda hal nya dengan menggunakan metode autentikasi, hanya membutuhkan program autentikasi pada client, sehingga proses otorisasinya lebih sederhana dan waktu yang dibutuhkan lebih sedikit.

Tabel 1 tingkat kesulitan.

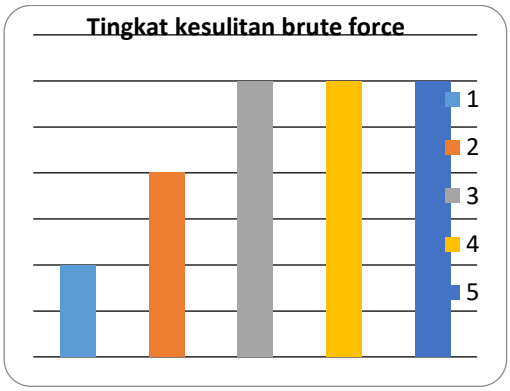
No	jumlah sequence	waktu	Tingkat kesulitan brute force
1	1	3 detik	1
2	2	5 detik	2
3	3	7 detik	3
4	4	10 detik	3
5	5	13 detik	3

Dalam penelitian ini, jika semakin banyak kunci atau *seqeunce* pada *port knocking* akan membutuhkan waktu yang cukup lama namun dapat menghindari serangan *bruteforce* yang dilakukan oleh seseorang yang tidak bertanggung jawab terhadah sebuah *server*.



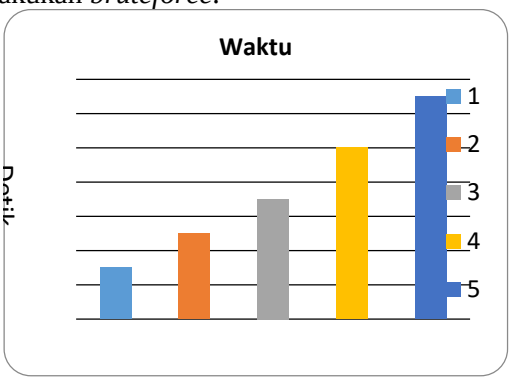
Gambar 4.5 Serangan brute force

Gambar 4.5 diatas menunjukan serangan *bruteforce* yang dilakukan oleh penyerang, penyerang mencoba menyerangan secara acak dengan menggunakan satu *sequence*. Jika metode *port knocking* menggunakan satu *sequence*, maka penyerang hanya membutuhkan waktu untuk menemukan *sequence* yang benar.



Gambar 4.6 Diagram kesulitan brute force

Pada gambar 4.6 diatas menjelaskan tingkat kesulitan melakukan serangan *brutefoce*, semakin sedikit kunci maka akan mudah untuk melakukan *bruteforce* namun jika semakin banyak kunci maka akan semakin sulit untuk melakukan *bruteforce*.



Gambar 4.7. Diagram waktu yang dibutuhkan saat knocking.

Pada Gambar 4.7 diatas menjelaskan jika kunci yang digunakan sedikit maka waktu yang dibutuhkan untuk melakukan knocking lebih singkat, namun jika kunci yang digunakan banyak maka akan dibutuhkan waktu yang lebih lama untuk melakukan *knocking*.

Pada penelitian ini untuk pengembangan metode *port knocking* ini yaitu dengan menambahkan *tunneling*, sehingga rancangan sistem pada penelitian ini semakin sempurna dalam mengatasi serangan yang sering terjadi pada metode ini, guna menutup celah pada rancang sistem penelitian ini.

5 KESIMPULAN

Port knocking memiliki potensi pengembangan yang cukup luas terutama dibagian pengamanan *sequence* dan *timeout* serta otentikasi yang terjadi setelah *sequence* terpenuhi, dalam penelitian ini metode *Authentication port knocking* digunakan guna otentikasi dari server ke *client* dan sebaliknya, Hasil yang diharapkan dari penelitian ini adalah tercapainya implementasi *port knocking* yang mampu mengatasi permasalahan otentikasi dan serangan *bruteforce*. Dan dapat menyederhanakan metode *port knocking* itu sendiri, namun juga menambah keamanan dengan waktu yang lebih sedikit.

REFERENSI

[1] Ramaki, Ali Ahmadian, and Reza Ebrahimi Atani. "LANGUAGE-BASED SECURITY TO APPLY COMPUTER SECURITY." *International Journal on Cryptography and Information Security (IJCIS)* 2.3 (2012): 37-47.

[2] Srivastava, Shiv Shakti, and Nitin Gupta. "A Novel Approach to Security using Extended Playfair Cipher." *International Journal of Computer Applications (0975–8887) Volume* (2011).

[3] Z. A. Khan, N. Javaid, M. H. Arshad, A. Bibi, and B. Qasim, "Performance evaluation of widely used portknoking algorithms," *Proc. 14th IEEE Int. Conf. High Perform. Comput. Commun. HPCC-2012 - 9th IEEE Int. Conf. Embed. Softw. Syst. ICESS-2012*, pp. 903–907, 2012..

[4] Ali, Fakariah Hani Mohd, Rozita Yunos, and Mohd Azuan Mohamad Alias. "Simple port knocking method: Against TCP replay attack and port scanning." *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on*. IEEE, 2012.

[5] Andreatos, A. S. Hiding the SSH port via smart Port Knocking. International Journal Of Computer .vol11 .2017.

[6] Boroumand, Laleh, et al. "Virtualization Technique for port-knocking in mobile cloud computing." *Int. J. Advanced Soft Comput. Appl.* 6.1 (2014).

[7] Mehran, Pourvahab, Ebrahimi Atani Reza, and Boroumand Laleh. "SPKT: Secure Port Knock-Tunneling, an enhanced port security authentication mechanism." *Computers & Informatics (ISCI), 2012 IEEE Symposium on*. IEEE, 2012.

[8] Datir, Mr Harshal N. "A Modified Hybrid Port Knocking Technique for Host Authentication: A Review. 2012"

[9] Rash, Michael. *Linux Firewalls: Attack Detection and Response with iptables,psad, and fwsnort*. No Starch Press, 2007.

[10] Awan, Awan. "Memberikan Akses Legal Terhadap Port Tertentu Yang Telah Ditutup oleh Firewall dengan Metode Port Knocking." *Jurnal Ilmiah CORE IT* 2.1 (2017).

[11] M. Krzywinski, "Port Knocking Implementations," Port Knocking, [Online]. Available:http://www.portknocking.org/view/implementations