CyberSecurity dan Forensik Digital e-ISSN: 2615-8442
Vol.3, No. 1, Mei 2020, hlm. 27-32

ANALISA DAN SIMULASI KEAMANAN JARINGAN *UBUNTU SERVER* DENGAN *PORT KNOCKING, HONEYPOT, IPTABLES, ICMP*

Muhammad Iqbal[1], Arini MT [2] , Hendra Bayu Suseno M.Kom[3]

[1]UIN Syarif Hidayatullah Jakarta
[2]UIN Syarif Hidayatullah Jakarta
[3]UIN Syarif Hidayatullah Jakarta
Email: [1]iqbalzlatan@rocketmail.com, [2]arini@uinjkt.ac.id, [3]hendra.bayu@uinjkt.ac.id

Abstrak

Keamanan jaringan komputer sangatlah penting untuk menjaga kerahasiaan data dan informasi yang terdapat pada *server*. Data dan informasi ini hanya ditujukan untuk *administrator* dan *user* yang berhak mengakses saja melalui *port* layanan. Membiarkan *port* penting terbuka adalah kesalahan fatal yang dapat mengakibatkan serangan terhadap *server*, umumnya teknik yang sering dilakukan adalah *scan port* dan *bruteforce*. Hal lain untuk melindungi *server* dengan *firewall* adalah salah satu metode yang dapat diterapkan pula, namun penggunaan *firewall* tidak efektif dikarenakan akan memblok semua layanan tanpa memperdulikan siapapun termasuk *administrator*, oleh karena itu penelitian ini dilakukan menggunakan *port knocking* yang digabungkan dengan *Honeypot, IPtables dan ICMP*. *IPtables* menggantikan peran dari *firewall*, untuk menentukan aturan *port* mana yang akan di *filter*, sehingga setiap paket yang masuk pada *filtered port* akan di-*refused*. *Port knocking* berfungsi untuk menentukan ketukan rahasia terhadap *port* layanan *server*. *Honeypot* untuk mengalihkan *server port* pada *port* tiruan dan sengaja terbuka untuk mengetahui apa saja upaya yang dilakukan untuk memasuki *server*, dan ICMP berfungsi sebagai *ping request* dari *client* terhadap *server*. Pengembangan selanjutnya dapat dengan menggunakan *static chiper knock.*

Kata kunci: *Keamanan Jaringan, Port Knocking, Honeypot, IPtables, ICMP.*

ANALYSIS AND SIMULATION OF UBUNTU SERVER NETWORK SECURITY USING PORT KNOCKING, HONEYPOT, IPTABLES, ICMP

Abstract

Computer network security is very important to maintain the confidentiality of data and information contained on the server. This data and information is only intended for administrators and users who have the right to access it through the service port. Leaving important ports open is a fatal error that can result in an attack on the server, generally the most commonly performed technique is port scan and bruteforce. protecting the server with a firewall is one method that can be applied, but the use of a firewall is not effective because it will block all services without caring about anyone, including administrators. For this purpose the port knocking method is developed with other methods, which are combining with Honeypot, IPtables and ICMP. IPtables itself replaces the role of the firewall, to determine which port rules will be filtered, so that every packet that enters the filtered port will be refused. Port knocking functions to determine the secret knock on the server service port. Honeypot to switch the server port on the artificial port and deliberately open so that we know what efforts are being made to enter the server, and ICMP functions as a ping request from the client to the server.For the next, it able use static chiper knock.

Keywords: *Network Security, Port Knocking, Honeypot, IPtables, ICMP.*

1. PENDAHULUAN

Seiring dengan pesatnya perkembangan teknologi tentu akan berdampak pada segala aspek kehidupan, walaupun memberikan dampak yang positif dalam menyediakan dan mendapatkan informasi, namun disisi negatifnya perlu adanya upaya untuk pencegahan. Ada faktor dimana kita tidak menyadari akan adanya ancaman, pencurian data dan perusakan data pada sebuah jaringan. Berbagai cara sudah diterapkan seperti menggunakan *firewall* sebagai dinding penghalang pembatasan akses. Penggunaan *firewall* sendiri masih kurang efektif dikarenakan menutup semua akses tanpa

memperdulikan siapapun yang sedang terkoneksi dalam jaringan. Suatu metode kemanan yang dapat menutup celah dan masalah pembatasan hak akses dari *firewall* yaitu dengan metode *port knocking*, metode ini digunakan dalam membantu mengamankan *server* (*Linux* dan *Unix)* dan monitoring jaringan melalui pembatasan akses *blocking* pada *port* yang terdapat dalam jaringan.

Penelitian pada (MUZAWI, 2016) melakukan pengamanan terhadap Port/Sniffer namun tidak pada DoS *Attack*, NAT *Attack*, dan *Port Scan*. Pada (ZAINAL, 2018) memfokuskan pada authentikasi dengan menggunakan *Knockd* dan *Python*, dan (KUSUMA, 2016) mengimplementasikan *Simple Port Knocking* Pada *Dynamc Routing* (OSFP) Menggunakan Simulasi GNS3. Pada penelitian ini fokus pada protokol TCP, SSH port, keamanan mencakup *DoS Attack*, *Port Scan* dan *port/sniffer* pada *Ubuntu* serta menambahkan protokol ICMP

2. LANDASAN TEORI

2.1. Port Knoking

Metode *Port Knocking* digunakan untuk dapat akses secara *remote* dengan tidak mengijinkan *port* dalam kondisi terbuka sehingga dapat melindungi *server* dari *port scanning* dan serangan *scripts kiddies.* Dengan *user* diberikan akses untuk mengakses *port* dan diakhiri dengan menutup *port* agar *firewall* menghapus *rule* yang ditulis sebelumnya untuk membuka *port* (FAJRI, 2013).

2.2. *Honeypot*

Security resource yang digunakan untuk diserang, diselidiki, atau dikompromikan namun terisolasi dan termonitor adalah *Honeypot*.

Honeypot memberi kontribusi terhadap keamanan namun tidak secara langsung mencegah serangan dan dapat mengurangi intensitas serangan penyusup ke *server*. *Honeypot* menurut tingkat interaksi (aktivitas penyerangan) dapat dikategorikan menjadi :

1. Low Interaction Honeypot

Didesain untuk mengemulasikan *service* (layanan) layaknya ke *server* yang asli, namun penyerang hanya mampu terkoneksi dan memeriksa satu atau beberapa *port*.

2. Medium Interaction Honeypot

Penyerang dapat menyisipkan *worm* akan digunakan untk melakukan data analissi yang tertera pada payload *worm* dari penyerang.

3. High Interaction Honeypot

Metode ini penyerang dapat berinteraksi langsung dan tidak ada batasan sehingga jika sudah dapat mengakses root maka akan dapat berinteraksi secara penuh.

2.3. *Iptables*

Iptables berfungsi sebagai firewall juga NAT. Tabel aturan seperti Filter (terdiri dari 3 chain : forward, input dan output), NAT (3 chain yaitu output, pre-rouitng dan post-routing), Mangle (5 chain : forward, output, input, postrouting dan presouting). Chain ini untuk TCP protocol Packet Quality of Service saat sebelum proses routing dilakukan.

3. METODOLOGI

3.1. Metode Pengumpulan Data

1. Studi Dokumen

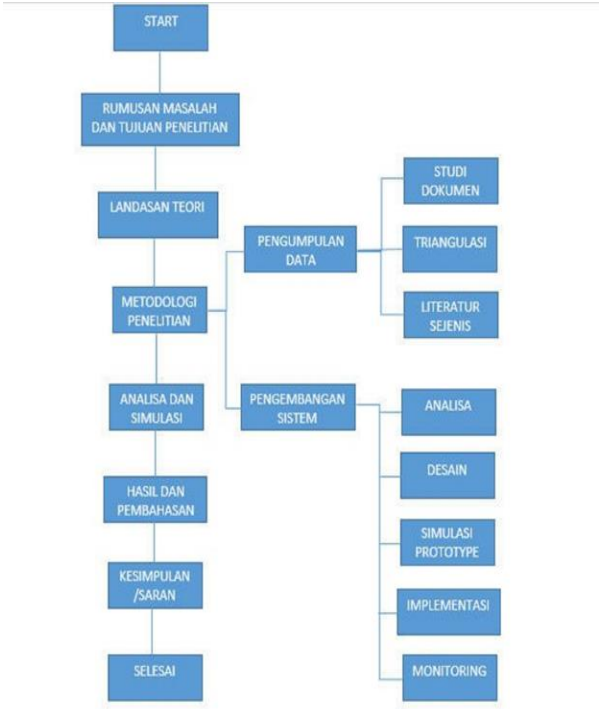
Penelitian ini menggunakan penulusuran data histori objek penelitian dan studi literatur.

2. Pengembangan Sistem

Kami menggunakan metode pengembangan NDLC (*Network Development Life Cycle*), dengan tahapan : analisa, desain, simulasi *prototype*, implementasi dan *monitoring*

3.2. Kerangka Berfikir

Berikut gambar 1 adalah kerangka berfikir pada penelitian ini.



Gambar 1. Kerangka Berfikir

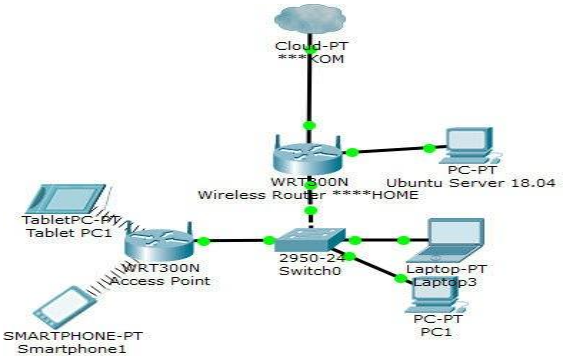
4. ANALISA DAN SIMULASI

4.1. Analisa Sistem

Pada tahap ini peneliti melakukan analisa sistem yang akan diterapkan untuk pengamanan *Ubuntu Server* dengan menggunakan metode *port knocking, honeypot, Iptables, dan ICMP*.

4.2. Topologi Jaringan

Gambar 2 adalah topologi jaringan yang digunakan, peneliti menggunakan jaringan LAN. Komputer *server* terhubung langsung dengan *Wireless Router*, untuk *client* terhubung melalui *switch*, dan AP.



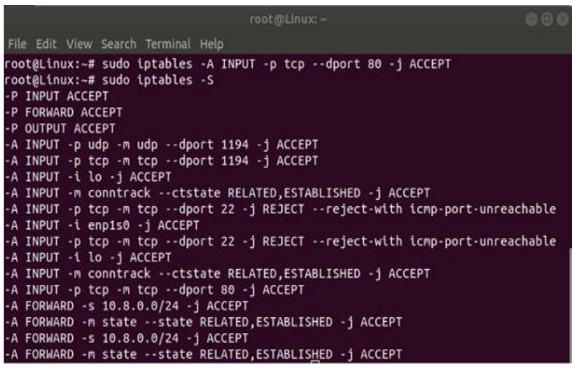
Gambar 2. Topologi Jaringan

4.3. Implementasi

Pada tahap ini penulis melakukan konfigurasi, implementasi dan pengujian, dimulai dari konfigurasi *IPtables*, menginstal *netfilter*, *knockd*, dan *honepot* (*cowrie honeypot*).

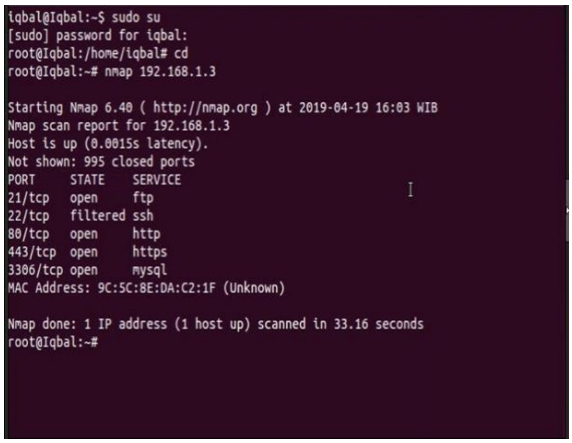
4.3.1. *IPtables*

Pada konfigurasi ini *server* masih menerima koneksi selain koneksi untuk SSH yang di-*block* ditunjukan pada gambar 3.



Gambar 3. *Rule Traffic*

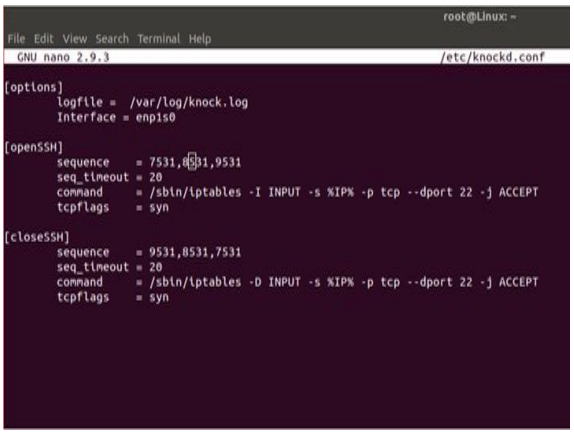
Gambar 4 adalah hasil *scan* hanya *port 22* saja yang di-*filter*, sementara port lainnya terbuka.



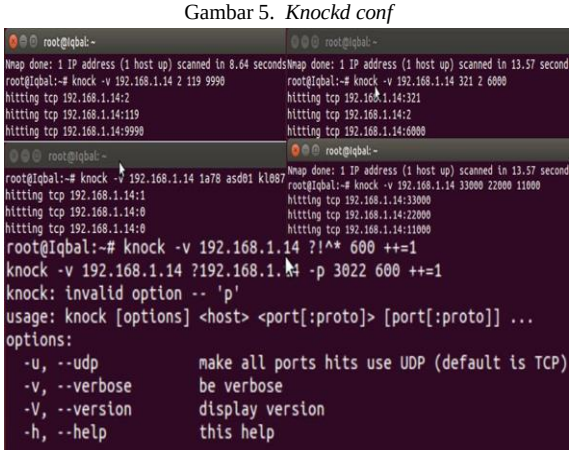
Gambar 4. *Port filtered*

4.3.2. *Port Knocking*

Setelah mengkonfigurasi *iptables*, selanjutnya konfigurasi *port knocking*. Pada fase inilah nantinya *user* yang berhak aksek atau tidak dapat mengakses *port* yang ditentukan, disini memfilter *port* 22 untuk layanan *service* SSH. Sehingga untuk mengakses *port* 22 tidak bisa sembarangan masuk untuk *client* yang memang tidak dikenal.



Gambar 5. *Knockd conf*



Gambar 6. Pengujian *Sequence*

Pada gambar 5 memperlihatkan konfigurasi *port knocking*, untuk menyimpan *log*, bisa rubah dari penyimpanan *default* dengan *"logfolder"* yang kita tentukan. Pada *open* dan *closeSSH*, dapat

mengkombinasikan huruf dengan angka, angka acak, untuk penggunaan *symbol*, ketukan *sequence* tidak akan bisa, terlihat pada gambar 6. Penulis menggunakan *Basic Static Sequence*.

4.3.3. ICMP

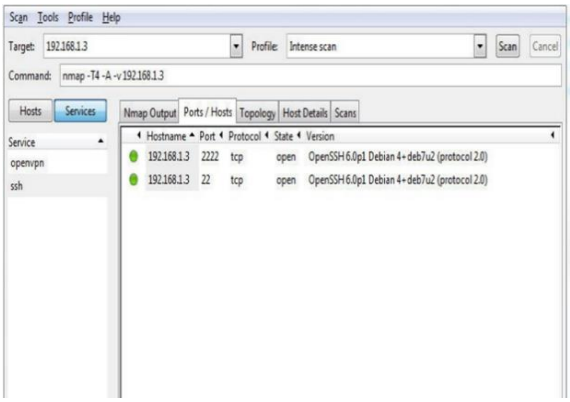
ICMP tidak ada hubungannya dengan metode keamanan *server* dengan *port knocking*, namun dalam penelitian ini digunakan untuk pengamanan *server* dengan memanfaatkan fitur dari ICMP ini, yaitu digunakan untuk *block ping request* dari pihak *client*, ditunjukkan gambar 7.

```
#net.ipv6.conf.all.forwarding=1
net.ipv4.icmp_echo_ignore_all=1
```

Gambar 7. ICMP *Script*

4.3.4. *Honeypot*

Pada penelitian ini, penulis menggunakan aplikasi *Honeypot* (C*owrie honeypot*) untuk keamanan tambahan di *port knocking*. C*owrie honeypot* berfungi sebagai *server* palsu yang sengaja disusupi oleh penyusup. Karena sebelumnya sudah dikonfigurasi paket *OpenSSH* pada Ubuntu menggunakan menggunakan *port* 22, sehingga harus merubah *port* 22 SSH pada *"sshd_config"* menggantinya menjadi 3022 dan berikutnya membuat akun pengguna baru dengan nama *Cowrie*, berikutnya membuat *virtual* baru, gambar 8 adalah *cowrie server honeypot*.

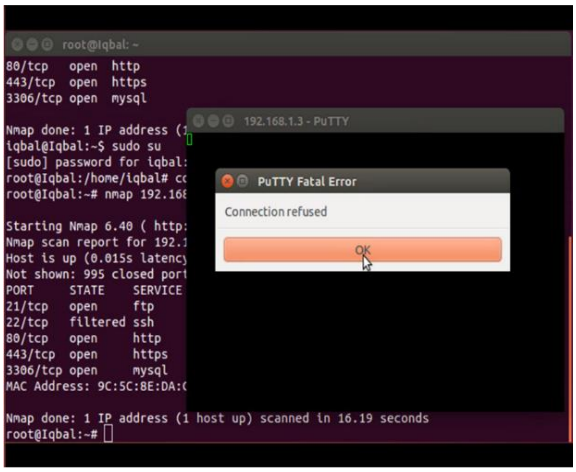


Gambar 8. *Cowrie Honeypot*

5. HASIL DAN PEMBAHASAN

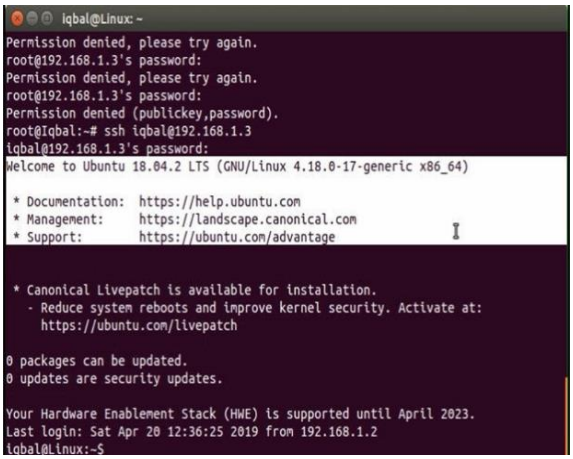
Penelitian ini memanfaatkan *honeypot* sebagai *server* tiruan untuk mencegah terjadinya serangan ke *server* utama dan dapat mengetahui apa saja upaya yang dilakukan untuk masuk kedalam serta mengganti peran dari *firewall* dengan *IPtables*, juga menggunakan ICMP protokol.

5.1. *Port Knocking* dan *Iptables*



Gambar 9. *Connection Refused*

Gambar 9 dari hasil *scan port* dengan menggunaka *nmap* pada *client* maupun *server* layanan SSH statusnya : *filtered* dan koneksi menuju layanan SSH akan di *refused*. Ini menjukkan bahwa pengamanan dari *IPtables* sudah aman dalam memproteksi layanan dengan mem-*filter port*. Gambar 10 adalah pengamanan ketukan *sequence* pada *IPtables*, dengan menginput *sequence* untuk membuka *filter port* dari *IPtables*.

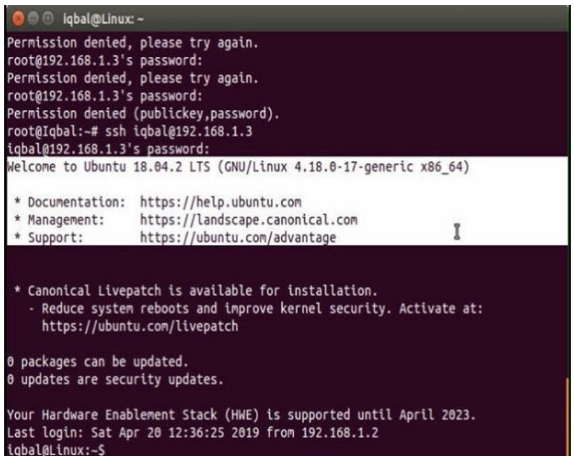


Gambar 10. *Log* Ketukan

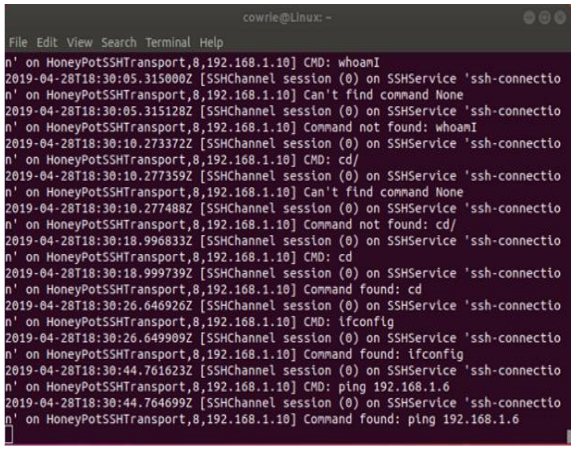
5.2. *Honeypot*

Penggabungan *port knocking* dan *honeypot* (c*owrie honeypot)* sebagai *server* palsu yang mudah di masuki penyusup, dan me-*monitoring* apa saja yang dilakukan penyusup. Pada informasi yang didapat dari *nmap*, *port* 22 dan *port* 2222 adalah *server* palsu dari *honeypot*, tertulis *"OpenSSH 6.0p1 Debian4"*, pada *server* SSH *port* 3022 adalah *"Ubuntu 18.04 LTS"*. Itu berarti *cowrie honeypot*

sudah berhasil men-*direct* dari *port* 22 menjadi 3022, dan *port* 3022 tidak akan muncul dalam pencarian *nmap*, ditunjukkan pada gambar 11.



Gambar 11. SSH 3022



Gambar 12. *Honeypot Log*

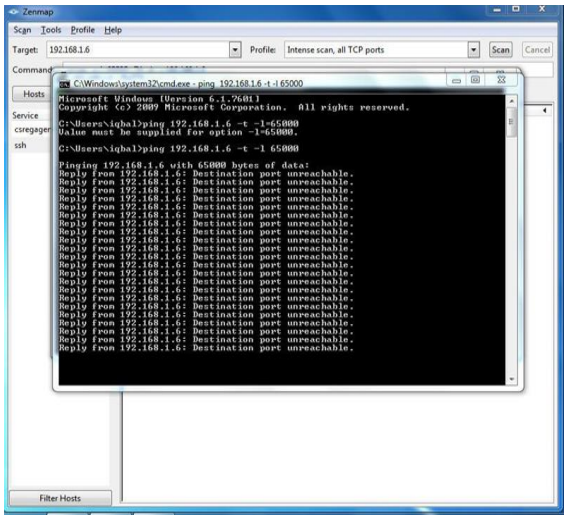
Aktifitas setelah masuk *honeypot* akan tersimpan pada *cowrie.log*, disni kita bisa melihat apa saja yang coba dilakukan oleh penyusup, kita dapat berinteraksi langsung, *honeypot* yang penulis terapkan termasuk *Medium Interaction Honeypot* dan layanan yang bisa digunakan terbatas. Gambar 12. Menunjukkan bahwa penyusup berhasil dialihkan pada *server honeypot.*

5.3. ICMP

Setelah pengujian dengan *honeypot* dan *IPtables*, penulis menambahkan ICMP untuk pengamanan *server*, ICMP ini berfungsi agar menolak *ping* dari berbagai *user* yang terkoneksi pada jaringan yang sama. Gunanya untuk menghindari serangan DoS yaitu *Ping of Death.* Sebuah *Ping* biasanya berukuran 56 *byte* atau 84 *byte*, ketika *header* IP lebih besar dari pada ukuran maksimal paket IP, yaitu 65.535 *byte*. Mengirim *ping* dalam ukuran ini (65.535 byte) bisa mengakibatkan kerusakan (*crash*) pada komputer target. Untuk menghindari itu penulis mem-*block ping* dengan menggunakan ICMP ditunjukan pada gambar 13.

Gambar 13. *Ping Client to Server*

Gambar 14 adalah hasil dari penggunaan ICMP, *client* tidak bisa mencoba untuk ping request terhadap *host server*.



Gambar 14. *Server to Client*

6. KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan dari hasil analisa dan simulasi kami menghasilkan :

1. Pengamanan *server Ubuntu* dengan *metode port knocking* berjalan dengan baik, dengan otentikasi sebagai syarat untuk menggunakan *port service*, hanya pengguna yang diperbolehkan saja untuk mengakses *port*.
2. Dengan menggunakan *Honeypot* sebagai *server* tiruan pada metode *port knocking*, dapat mengalihkan *port service* kepada *fake port* yang dibuat *honeypot*.
3. Dengan menggunakan *Honeypot* dapat mencegah dari upaya seseorang untuk *scanning port* dan *brute* force, maka yang akan tampil pada aplikasi *scan* adalah *fake port*,

4. Penggunaan *IPtbales* dapat memblok *incoming packet* sehingga *port service* tidak akan terbuka.
5. Penerapan ICMP pada *server* efektif sehingga dapat mengatasi serangan DoS *knocking* seperti PoD.

6.2. Saran

Penulis menyadari bahwa sistem keamanan *port knocking* masih dibutuhkan inovasi baru untuk mengembangkan sistem keamanan ini.

1. Untuk keamanan *server* kedepannya, *metode port knocking* menggunakan *VPN tunneling*.
2. Menggunakan *static chiper knock*.
3. Diharapakan untuk penerapan keamanan yang lebih tinggi menggunakan *high interaction honeypot*.

DAFTAR PUSTAKA

COBBAUT, P. Mastering Linux Fundamentals, Samurai Media Limited, 2016.

COMPARITECH. (2018). What is ICMP?. "https://www.comparitech.com/net-admin/what-is-icmp/" (diakses 15-04-2019)

HELP.UBUNTU. (2017). Iptables How To. https://help.ubuntu.com/community/IptablesHowTo (diakses 11- 04-2019).

FAJRI, M.S.H. SUHATMAN, R., & PUTRA, Y.E. 2013. Analisa Port Knocking Pada Sistem Operasi Linux Ubuntu Server 12.04 LTS. Vol 2 No 2 (2013)

KUSUMA, A.P.A. (2016). Implementasi Simple Port Knocking Pada Dynamc Routing (OSFP) Menggunakan Simulasi GNS3. Jurnal Manajemen Informatika. Volume 5 Nomor 2 Tahun 2016, 7 – 17

MEDIUM. (2018). How to Setup "Cowrie"—An SSH Honeypot. "https://medium.com/threatpunter/how-to-setup-cowrie-an-ssh- honeypot-535a68832e4c" (diakses 13-04-2019).

MUZAWI, R. (2016). Aplikasi Pengendalian Port dengan Utilitas Port Knocking untuk Optimalisasi Sistem Keamanan Jaringan. SATIN - Sains dan Teknologi Informasi, Vol. 2, No. 1.

SLIDESHARE. (2015). Honeypots for Active Defense. "https://www.slideshare.net/heinzarelli/honeypots-activedefense- gregfoss" (diakses 13-04-2019)

SUARTANA, I., INDRIYANI, T., & MARDIYANTO, B, (2016), Analisis Dan Implementasi Honeypot Dalam Mendeteksi Serangan Distributed Denial-Of-Services (DDOS) Pada Jaringan Wireless, INTEGER : Journal of Information Technology, Vol 1, No 2

UTDIRARTATMO, F. (2005), Teknik Kompilasi, Graha Ilmu, Yogyakarta.

WAFI, H (2016). Implementasi Sistem Keamanan Honeypot dengan Modern Honeynetwork Pada Jaringan Wireless. Skripsi Teknologi Informasi. Perpustakaan FST UIN Jakarta, 0356 TI 2016

WILMAN, FITRI, I., & NATHASIA, N.D. (2018). Port Knocking Dan Honeypot Sebagai Keamanan Jaringan Pada Server Ubuntu Virtual, 3(1), 27–33. http://ejurnal.unmerpas.ac.id/index.php/informatika/article/view/86.

ZAINAL, M. SYAIFUDDIN, & RISQIWATI, D. (2018). Implementasi Authentication Pada Port Knocking Ubuntu Server Menggunakan Knockd dan Python, Jurnal SISTEMASI, Volume 7, Nomor 2, Mei 2018 : 169 – 175.