Seminar Nasional Sistem Informasi dan Teknologi Informasi 2018
SENSITEK 2018
STMIK Pontianak, 12 Juli 2018

Bot Alert Snort dengan Telegram Bot API pada Intrusion Detection System: Studi Kasus IDS pada Server Web

Yohanes Priyo Atmojo [1)]

Sekolah Tinggi Manajemen Informatika dan Teknik Komputer STIKOM Bali
Jalan Raya Puputan No.86, Denpasar, Bali. Telp (0361) 244445
e-mail: yohanes@stikom-bali.ac.id

Abstrak

Laporan serangan yang diterima oleh Intrustion Detection System merupakan salah satu indikasi adanya upaya untuk melakukan penyerangan terhadap sistem. Sebagai bagian dari pencegahan terhadap adanya serangan, system administrator harus mengawasi sistem tersebut setiap waktunya. Bot atau robot digunakan untuk melakukan otomatisasi terhadap suatu kegiatan yang berulang, dan dapat dimanfaatkan untuk menggantikan fungsi pengawasan secara terus menerus yang dilakukan oleh system administrator. Telegram menyediakan fitur untuk menggunakan layanan instant messenger secara otomatis yang salah satu fiturnya adealah layanan bot. Dengan adanya penggabungan antara fitur bot Telegram dan Intrusion Detection System, dapat menyederhanakan proses monitoring dengan cara mengirimkan informasi secara otomatis setiap kali sistem mengalami serangan dari pihak luar. Penelitian ini menguji kehandalan dari system IDS yang dibangun menggunkaan Raspberry Pi dan digabungkan dengan Telegram Bot API sebagai media alert serangan yang terjadi. Dari hasil pengujian didapat bahwa dalam intensitas serangan yang kecil dan menengah, sistem masih mampu melakukan pengiriman alert ke administrator, namun untuk intensitas serangan yang besar, sistem tidak dapat mengirim alert karena resource sistem telah habis untuk mendeteksi serangan tersebut.

Kata kunci: *Telegram Bot API, Intrution Detection System, Web Server*

1. Pendahuluan

Server web adalah salah satu *server* yang paling sering mengalami serangan baik dari sisi sistem *web* nya sendiri maupun aplikasi yang berjalan di dalamnya, terlebih serangan terhadap ketersedian informasi pada suatu *website* [1]. Data tahun 2017 dari GOV-CSIRT (*Goverment Computer Security Incident Response Team*), khusus di Indonesia kasus mengenai serangan terhadap *website*, tipe *flood attack* [2] merupakan serangan keempat terbanyak setelah *web defacement* dan *SQL Injection* [3]. *Flood-attack* dapat melumpuhkan koneksi dengan membanjiri server dengan paket data dengan intensitas besar yang dapat mengakibatkan client tidak dapat me-*request* data dari server dan sebalikanya server tidak dapat me-*response* request dari client [2]. Hal ini dapat menghambat pengguna website dalam mengakses informasi dari website tersebut.

Salah satu cara untuk mendeteksi adanya anomali dari jaringan, seperti *flood-attack* adalah dengan menggunakan *IDS*. *IDS* (*Intrution Detecting System*) adalah sebuah perangkat lunak atau perangkat keras yang bekerja secara otomatis untuk memonitor kejadian pada jaringan komputer dan menganalisis masalah keamanan jaringan [4]. Salah satu aplikasi *IDS* yang digunakan adalah *Snort* [5]. Informasi yang diperlukan dari sebuah *IDS* adalah informasi mengenai adanya serangan yang terjadi saat itu sehingga system administrator dapat melakukan pencegahan terhadap dampak dari serangan tersebut. Di satu sisi, peringatan ini bermanfaat, tapi di sisi lain diperlukan petugas yang melakukan pengawasan terhadap *IDS* ini secara terus menerus.

Bot atau yang dikenal dengan *robot*, merupakan solusi untuk menggantikan suatu aktifitas yang berulang dan dapat diandalkan untuk melakukan otomatisasi sebuah kegiatan. *Telegram* menyediakan fitur *bot* yang dapat memudahkan user untuk melakukan proses otomatisasi terhadap sebuah sistem, sehingga mengurangi campur tangan *user* di dalamnya [6][7][8][9][10][11][12]. Oleh karena itu, dibuat sebuah sistem otomatisasi peringatan dari *IDS* yang ditujukan kepada sistem *web*.

2. Pembahasan

2.1. *IDS*

IDS (*Intrution Detecting System*) adalah sebuah perangkat lunak atau perangkat keras yang bekerja secara otomatis untuk memonitor kejadian pada jaringan komputer dan menganalisis masalah keamanan jaringan. Terdapat dua jenis *IDS* berdasarkan penempatannya [4], yaitu:

a. *Network Based IDS* (*NIDS*). *NIDS* akan melakukan pemantauan terhadap seluruh bagian pada jaringan dengan mengumpulkan paket-paket data yang terdapat pada jaringan tersebut serta melakukan analisa dan menentukan apakah paket paket tersebut merupakan paket normal atau paket serangan.

b. *Host Based IDS* (*HIDS*). *HIDS* hanya melakukan pemantauan tertentu dalam jaringan. *HIDS* biasanya pada perangkat komputer akan memantau kejadian seperti kesalahan *login* berkali-kali dan melakukan pengecekan pada *file*. Hal yang perlu diperhatikan pada implementasi *IDS* adalah perihal *false positive* dan *false negative*. *False positive* adalah peringatan serangan yang dihasilkan oleh *IDS* akan sebuah paket normal pada sistem yang dimonitor. *False negative* adalah sebuah serangan yang benar – benar terjadi namun terlewatkan oleh *IDS* sehingga *IDS* tidak akan menghasilkan peringatan apapun atas serangan tersebut. *IDS* dapat melewatkan serangan karena serangan tersebut tidak dikenali oleh *IDS* atau karena penyerang berhasil menggunakan sebuah metode serangan yang dapat menghindari *IDS*.

2.2. *Snort*

Snort adalah suatu perangkat lunak untuk mendeteksi penyusup dan mampu menganalisis paket yang melintasi jaringan secara real time dan melakukan logging ke dalam *database* serta mampu mendeteksi berbagai serangan yang berasal dari luar jaringan. *Snort* merupakan sebuah produk *open source* yang dikembangkan oleh Marty Roesch dan tersedia gratis. *Snort* bisa digunakan pada sistem operasi *Linux*, *Windows*, BSD, *Solaris*, dan sistem operasi lainnya. *Snort* merupakan *network based IDS* yang menggunakan metode *Signature Based Detection*, menganalisis paket data apakah sesuai dengan jenis serangan yang sudah diketahui olehnya.

Terdapat banyak produk *IDS* lainnya seperti *Cisco IDS* dan *ISS Real Secure* serta produk-produk lain yang juga *open source*, *Snort* dipilih karena memilki beberapa kelebihan sebagai berikut [13]:

a. Snort mudah dalam konfigurasi. Semua konfigurasi pada *Snort* mulai dari *file* konfigurasi sampai pada rules-nya sudah tersedia dan mudah untuk dilakukan. Bahkan dapat ditambahkan *rule* sendiri untuk jenis-jenis serangan yang baru.

b. Gratis. Diluncurkan dengan lisensi *GNU GPL* yang berarti *Snort* bisa digunakan secara bebas tanpa biaya apapun.

c. Dapat berjalan pada berbagai macam sistem operasi. Awalnya *Snort* dikembangkan dalam lingkungan sistem operasi *UNIX*, tetapi *Snort* dapat digunakan pada sistem operasi yang lainnya.

2.3. Alur Penelitian

Dalam melakukan penelitian, tahapan-tahapan yang dilakukan seperti pada Gambar 1. Adapun rincian tahapannya adalah sebagai berikut:

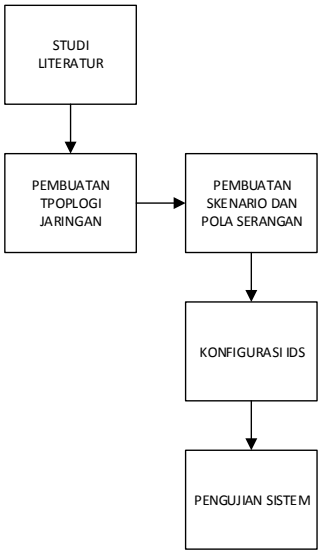
1. Studi literatur, yaitu mempelajari karakteristik *single-board computer*, dalam hal ini menggunakan *Raspberry Pi* [14][15]. Literatur yang digunakan lebih banyak menggunakan referensi dari web resmi dari *Raspberry Pi* dan *eLinux*. Studi literatur juga mempelajari aplikasi *IDS*, yaitu *Snort*.

2. Perancangan topologi jaringan pengujian, yaitu mempelajari dan merancang topologi jaringan yang digunakan sebagai bahan pengujian *IDS*.

3. Pembuatan skenario dan pola serangan terhadap sistem, yaitu pembuatan skenario yang dijalankan pada pengujian serta membuat pola serangan yang disimulasikan pada sistem yang dibangun.

4. Konfigurasi *rule* pada *IDS* dan pembuatan *bot*, yaitu melakukan konfigurasi agar *IDS* mengenali pola serangan yang dijalankan serta membuat optimasi pada program *IDS*. Pada tahap ini juga dilakukan pembuatan aplikasi *bot* yang mengirimkan pesan dari *IDS* ke aplikasi *Telegram* [16] apabila terdapat *alert* dari *IDS* saat menemukan pola serangan yang sesuai dengan *rules IDS*.

5. Pengujian sistem, yaitu melakukan pengujian dari seluruh skenario yang dibuat.

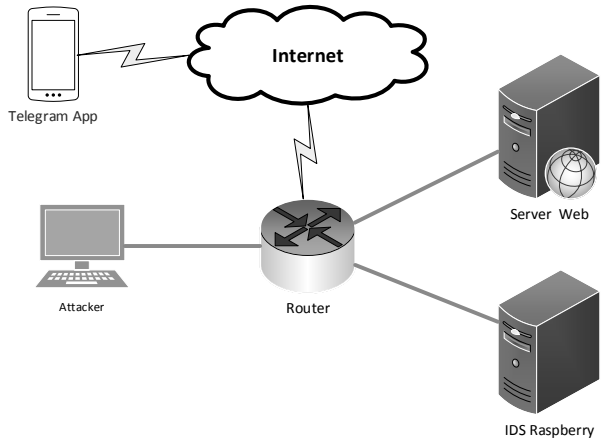


Gambar 1. *Alur Penelitian*

2.5. Topologi Jaringan Pengujian Sistem

Gambar 2 merupakan gambaran topologi jaringan yang digunakan dalam penelitian ini. Topogi ini adalah topologi yang mensimulasikan bahwa pengujian serangan dilakukan melalui komputer *Attacker* dan router dikonfigurasikan untuk mengaktifkan *port mirroring* [17], sehingga *traffic* data yang mengarah ke *Server Web* akan diterima juga oleh mesin *Raspberry Pi* yang difungsikan sebagai *IDS*. Topologi ini

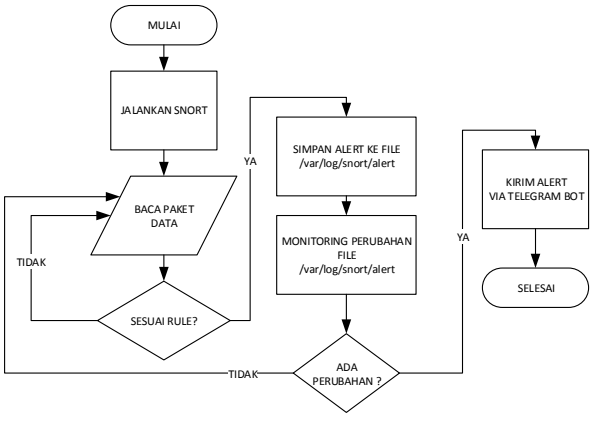
menggunakan *range IP local*, sehingga memudahkan untuk melakukan monitoring terhadap paket yang masuk serta mengurang adanya paket data yang terkirim dalam jaringan ini.



Gambar 2. *Topologi Jaringan Pengujian Sistem*

2.6. Alur Sistem

Sebagai gambaran umum dari sistem *alert*, maka dibuat alur kerja sistem monitoring. Alur sistem dimulai dengan menjalankan *Snort* yang sudah terkonfigurasi dengan rule untuk melakukan pemilahan terhadap paket data yang ditangkap oleh *SNORT*. Apabila ada paket data yang sesuai dengan rule, maka akan disimpan ke dalam *file /var/log/Snort/alert* berupa informasi mengenai jenis *rule* yang sesuai. Aplikasi *swatch* berjalan untuk memonitoring setiap perubahan yang terjadi dari *file log Snort*, dan mengirimkan notifikasi ke aplikasi *telegram* melalui *bot API* setiap menitnya. Adapaun alur sistem dapat dilihat sebagai berikut:



Gambar 3. *Alur Sistem*

2.7. Skenario Pengujian

Skenario pengujian dibagi menjadi beberapa skenario yang dilakukan untuk melakukan pengujian terhadap *Web Server* dan *IDS Raspberry Pi*, antara lain:

Tabel 1. *Skenario Pengujian*

Parameter Pengujian	Nilai
Alat	Raspberry Pi 3
Sistem Operasi	Arch Linux ARM
Memori RAM	992MB
Memori GPU	32 MB
Aplikasi IDS	Snort
Aplikasi Serangan SYN Flood	Hping3 [18]
Parameter Serangan SYN Flood	100, 1000, 10000 (dalam paket/detik)

2.8 *Rules Snort*

Rules Snort yang digunakan adalah *rules* yang sudah tersedia dari pengembang *Snort* itu sendiri, namun karena skenarionya sudah ditentukan, maka *rules* tersebut dispesifikkan untuk serangan *DOS* (*Denial of Service*), yang disimulasikan sebagai *syn-flood* menggunakan *tools hping3*. *Rules Snort* yang digunakan disederhanakan sebagai berikut:

```
alert tcp any any -> $HOME_NET 80 (flags: S; msg:"Possible TCP
DoS"; flow: stateless; threshold: type both, track by_dst,
count 70, seconds 10; sid:10001;rev:1;)
```

2.9. *Bot Telegram*

Bot telegram dijalan menggunakan *cURL* dan dipicu oleh perubahan isi dari *alert log Snort* dengan bantuan *file watcher swatch*. Adapun pengaturan *file bot telegram* sebagai berikut:

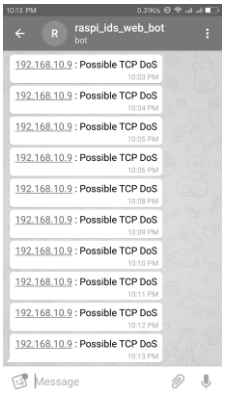
```
#!/bin/bash
message=$1
dt=`date '+%d/%m/%Y %H:%M:%S'`
IP=$(ip a s|sed -ne '/127.0.0.1/!{s/^[ \t]*inet[ \t]*\([0-9.]\
+\)\/.*$/\1/p}')
apiToken=<API_TOKEN_TELEGRAM>
userChatId=<USER_CHAT_ID>

sendTelegram() {
   curl -s -X POST
   https://api.telegram.org/bot$apiToken/sendMessage \
   -d text="$IP : $message" -d chat_id=$userChatId
   echo “$dt : $IP : $message” \
   >> /var/log/sendTelegramMessage.log
}
if  [[ -z "$message" ]]; then
   echo "Please add message to me!"
else
   sendTelegram
fi
```

Sedangkan untuk konfigurasi swatch dibuat untuk melakukan monitoring setiap menit untuk mengetahui perubahan yang terjadi pada *file /var/log/Snort/alert*. Adapun konfigurasi swatch adalah sebagai berikut:

```
watchfor /Possible TCP DoS/
       exec bash telegram-bot.sh “TCP Dos”
       echo red
       throttle 00:01:00
```

Hasil *output* untuk *alert bot* dapat diakses melalu aplikasi *telegram*, baik dari aplikasi *desktop*, *web*, maupun *mobile*. Gambar 4 adalah hasil dari *bot telegram* apabila *server web* diujicobakan pada saat menerima serangan *syn-flood* menggunakan *hping3*.



Gambar 4. *Hasil alert dari Telegram Bot pada Aplikasi Telegram*

2.10. Serangan SYN Flood

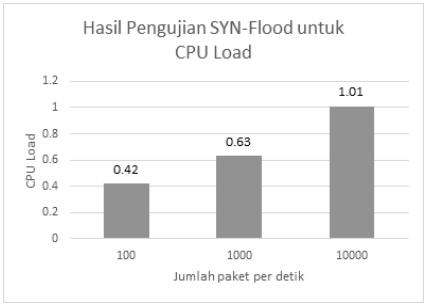
Serangan SYN flood dilakukan sebanyak 3 kali serangan dengan intensitas berbeda, yaitu dengan mengirimkan 100 paket per detik, 1000 paket per detik, dan 10000 paket per detik. Sintaks *hping3* yang digunakan dalam pengujian sebagai berikut:

```
hping3 -S -i u$X -p 80 $IP_TARGET
```

Variabel $X adalah jumlah paket yang dikirim dalam setiap detik, dimana nilai $X = 1000 untuk 100 paket perdetik, $X = 100 untuk 1000 paket perdetik, dan $X=10 untuk 10000 paket perdetik.

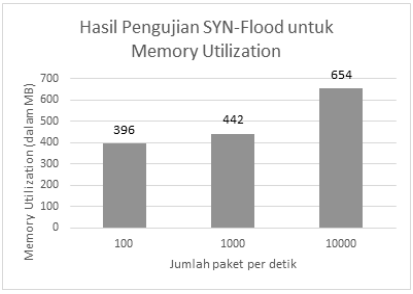
2.11 Hasil pengujian Sistem

Pengujian system dilakukan dengan cara melakukan uji coba serangan dengan intensistas yang berbeda, sesuai dengan skenario serangan, yakni 100 serangan, 1000 serangan dan 10000 serangan. Dari hasil pengujian diperoleh bahwa terjadi peningkatan intensitas dari *CPU Load*, *Memory Utilization*, dan *Network Throughput*. Adapun hasil peningkatan tersebut dapat dilihat pada Gambar 5, Gambar 6, dan Gambar 7.



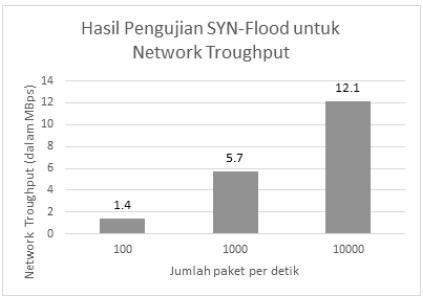
Gambar 5. *Hasil pengujian Serangan Syn-flood pada CPU Load*

Pada pengujian di sisi *CPU load*, terlihat bahwa beban serangan *syn-flood* 10000 paket per detik membuat *CPU Raspberi Pi* memanfaatkan seluruh *CPU* untuk memproses data dari paket yang diterima. Sedangankan pada pengujian 100 dan 1000 paket per detik, *CPU* masih terdapat sisa *resource* yang tersedia.



Gambar 6. *Hasil pengujian Serangan Syn-flood pada Memory Utilization*

Pada pengujian di sisi *Memory Utilization*, *trend* peningkatan penggunaan *memory* berbanding lurus dengan jumlah paket data yang diterima. Terlihat bahwa pengingkatan memori cukup signifikan terjadi pada pengujian 10000 paket per detik, sedangkan pada pengujian lainnya tidak terpaut jauh.



Gambar 7. *Hasil pengujian Serangan Syn-flood pada Network Throughput*

Pada pengujian di sisi *Network Throughput*, *trend* peningkatan yang sama dengan sisi *CPU Load* maupun *Memory Utilization*. Pengujian dengan jumlah paket data 10000 per detik menghabiskan semua *thoughput* yang

tersedia dari *port FastEthernet Raspberry Pi*, yaitu sebesar 12.1 *MBps*.

Dengan hasil yang didapat, maka diuji dengan menggunakan *bot telegram* sebagai *alert* kepada *user*. Pengujian akan diulang sebanyak 3 kali untuk masing-masing skenario serangan *Syn-flood*. Hasil pengujian *telegram bot alert* dapat dilihat pada Tabel 2.

Tabel 2. *Hasil pengujian pengiriman pesan dari bot Telegram*

No	Jumlah paket *syn-flood* (paket per detik)	Status Serangan pada SNORT	Status Pengiriman Pesan Bot	Delay Pengiriman Pesan (detik)
1	100	Terdeteksi	Terkirim	0
2	100	Terdeteksi	Terkirim	0
3	100	Terdeteksi	Terkirim	0
4	1000	Terdeteksi	Terkirim	2
5	1000	Terdeteksi	Terkirim	5
6	1000	Terdeteksi	Terkirim	3
7	10000	Terdeteksi	Tidak Terkirim	N/A
8	10000	Terdeteksi	Tidak Terkirim	N/A
9	10000	Terdeteksi	Tidak Terkirim	N/A

3. Kesimpulan

Berdasarkan pada penelitian yang telah dilakukan, *rule Snort* dapat mendeteksi simulasi serangan yang dilakukan. Akan tetapi hal yang berbeda pada saat ditambahkan *bot alert* berupa *telegram bot*. *Bot* dapat berjalan dengan baik hanya pada intensitas serangan yang kecil maupun menengah. Namun pada serangan dengan intensitas 10000 paket perdetik, dengan 3 kali pengulangan skenario, pesan dari *telegram bot* tidak dapat terkirim. Pada skenario 10000 paket per detik terjadi beban kinerja *CPU* dari *Raspberry Pi* mencapai angka 1.01 dan *network throughput* dari jaringan mencapai nilai 12.1 *MBps*, sehingga aplikasi *bot* tidak mendapatkan *resource* terutama *resource network* untuk mengirim pesan *alert* ke jaringan luar. Penelitian selanjutnya diharapkan sistem ini digabungkan dengan *IPS* (*Intrustion Prevention System*) dan algoritma deteksi *DDOS* yang lebih baik, sehingga apabila terjadi serangan dengan intensitas tinggi, maka serangan tersebut langsung terdeteksi dan diblokir oleh sistem.

Daftar Pustaka

[1] J. Scambray, V. Liu, and C. Sima, *Hacking Exposed Web Applications, Third Edition*. McGraw-Hill Education, 2010.

[2] S. S. Chapade, K. U. Pandey, and D. S. Bhade, "Securing Cloud Servers Against Flooding Based DDOS Attacks," in *2013 International Conference on Communication Systems and Network Technologies*, 2013, pp. 524–528.

[3] "Statistik Insiden Respon Domain .Go.Id | GovCSIRT – Kementerian Komunikasi dan Informatika." [Online]. Available: https://govcsirt.kominfo.go.id/statistik-insiden-respon-domain-go-id/. [Accessed: 21-Jun-2018].

[4] J. R. Vacca, *Computer and Information Security Handbook*. Elsevier Science, 2017.

[5] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.

[6] D. K. W. Tion, "Perancangan Bot untuk Remote Monitoring pada Server Menggunakan Telegram Bot API," Program Studi Teknik Informatika FTI-UKSW, 2016.

[7] V. A. Kozhevnikov, O. Y. Sabinin, and J. E. Shats, "LIBRARY DEVELOPMENT FOR CREATING BOTS ON SLACK, TELEGRAM AND FACEBOOK MESSENGERS," *Theor. Appl. Sci.*, no. 6, pp. 59–62, 2017.

[8] J. C. de Oliveira, D. H. Santos, and M. P. Neto, "Chatting with Arduino platform through Telegram Bot," in *Consumer Electronics (ISCE), 2016 IEEE International Symposium on*, 2016, pp. 131–132.

[9] Á. Alesanco, J. Sancho, Y. Gilaberte, E. Abarca, and J. Garcia, "Bots in messaging platforms, a new paradigm in healthcare delivery: application to custom prescription in dermatology," in *EMBEC & NBC 2017*, Springer, 2017, pp. 185–188.

[10] A. Palisse, A. Durand, H. Le Bouder, C. Le Guernic, and J.-L. Lanet, "Data Aware Defense (DaD): Towards a Generic and Practical Ransomware Countermeasure," in *Nordic Conference on Secure IT Systems*, 2017, pp. 192–208.

[11] G. Sastrawangsa, "Pemanfaatan Telegram Bot Untuk Automatisasi Layanan Dan Informasi Mahasiswa Dalam Konsep Smart Campus," *E-Proceedings KNS&I STIKOM Bali*, pp. 772–776, 2017.

[12] H. N. U. R. ROCHIM, S. T. Lukman Heryawan, and others, "RANCANG BANGUN TELEGRAM BOT PADA TELEGRAM MESSENGER DENGAN METODE LONG POLLING UNTUK KOPERASI KOPMA UGM," Universitas Gadjah Mada, 2016.

[13] A. NETHANEL SETIAWAN JUNIOR, A. HARIANTO, and A. ALEXANDER, "PERANCANGAN DAN IMPLEMENTASI INTRUSION DETECTION SYSTEM PADA JARINGAN WIRELESS BINUS UNIVERSITY," BINUS, 2009.

[14] A. Sforzin, F. G. Mármol, M. Conti, and J.-M. Bohli, "RPiDS: Raspberry Pi IDS—A Fruitful Intrusion Detection System for IoT," in *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, 2016, pp. 440–448.

[15] S. Mahajan, A. M. Adagale, and C. Sahare, "Intrusion Detection System Using Raspberry PI Honeypot in Network Security," *Int. J. Eng. Sci.*, vol. 6, no. 3, p. 2792, 2016.

[16] C. P. media, *Latest Mobile Apps and Technology: 6 Top Smartphones to Buy under 10K In 2015*. Cloudpeer Media Technologies, 2015.

[17] D. E. Frattura, R. W. Graham, and J. Roese, "Method for network traffic mirroring with data privacy." Google Patents, 2012.

[18] B. Buchanan, F. Flandrin, R. Macfarlane, and J. Graves, "A methodology to evaluate rate-based intrusion prevention system against distributed denial-of-service (DDoS)," *Cyberforensics 2011*, 2011.