JOISIE Journal Of Information System And Informatics Engineering
Vol. 5, No.2 Desember 2021, Hlm 98-108

p- ISSN: 2503-5304
e- ISSN: 2527-3116

PENINGKATAN KEAMANAN JARINGAN WIRELESS DENGAN MENERAPKAN SECURITY POLICY PADA FIREWALL

Muhammad Jufri[1)], Heryanto[2*)]
[1,2]Fakultas Ilmu Komputer, Universitas Internasional Batam, Batam
email: [1]jufri@uib.edu, [2*]**1831150.heryanto@uib.edu**

Abstract

Current technological developments have a significant impact on human performance in fulfilling daily activities. The development of technology makes crime in the network more widespread so that network security is very influential on the prevention of attacks carried out by attackers. Examples of crimes such as theft of company data that cause harm to the company as well as other crimes. For that, a system that can detect attacks such as Denial of Service, Ping Attack, and Port Scanning. Intrusion Detection System (IDS) serves to monitor every network traffic that passes. The purpose of the study is to define the security of wireless networks and get results from the use of IDS. This implementation is carried out on the Linux Ubuntu operating system version 18.04 LTS using snort and Iptables as an attack deterrent. After that, use the opensource wireshark tool as an analysis of attacks that occur in the network. The research method used is the Security Policy Development Life Cycle (SPDLC) by passing through several stages, namely: Analysis, Design, Implementation, Enforcement, Enchancement. The results concluded that attacks carried out by attackers on the network can be known and handled before wider damage occurs as well as the use of wireshark that can analyze attacks properly through the flow graph provided.

Keywords: *Linux, Iptables, Network Security, Security Policy Development Life Cycle (SPDLC), Intrusion Detection System (IDS)*

Abstrak

Perkembangan teknologi saat ini memiliki dampak signifikan terhadap kinerja manusia dalam memenuhi aktivitas sehari-hari. Perkembangan teknologi membuat kejahatan dalam jaringan semakin meluas sehingga keamanan jaringan sangat berpengaruh terhadap pencegahan serangan yang dilakukan oleh penyerang. Contoh kejahatan seperti pencurian data perusahaan yang menyebabkan kerugian bagi perusahaan serta kejahatan lainnya. Untuk itu, diperlukan sistem yang dapat mendeteksi serangan Seperti Denial of Service, Ping Attack, dan Port Scanning yaitu Intrusion Detection System (IDS) berfungsi memantau setiap lalu lintas jaringan yang berlalu. Tujuan penelitian yaitu untuk mendefinisikan keamanan jaringan wireless dan mendapatkan hasil dari pemakaian IDS. Implementasi ini dilakukan pada sistem operasi Linux Ubuntu versi 18.04 LTS menggunakan snort dan Iptables sebagai pencegah serangan. Setelah itu, menggunakan tools opensource wireshark sebagai analisa serangan yang terjadi dalam jaringan tersebut. Metode penelitian yang digunakan adalah Security Policy Development Life Cycle (SPDLC) dengan melewati beberapa tahapan, yaitu:. Analysis, Desain, Implementation, Enforcement, Enchancement. Hasil penelitian menyimpulkan bahwa serangan yang dilakukan oleh penyerang pada jaringan dapat diketahui dan ditangani sebelum kerusakan yang lebih luas terjadi serta penggunaan wireshark yang dapat menganalisa serangan dengan baik melalui flow graph yang disediakan.

Kata Kunci: *Linux, Iptables, Keamanan Jaringan, Siklus Hidup Pengembangan Kebijakan Keamanan (SPDLC), Sistem Deteksi Intrusi (IDS)*

1. PENDAHULUAN

Dalam perkembangan teknologi saat ini, kajian ilmu dan terapan teknologi jaringan *wireless* telah berkembang pesat. Perkembangan teknologi selalu membawa perubahan untuk kita. Seiring berkembangnya teknologi, semua jenis informasi dapat diakses dengan mudah (Mukti & Junikhah, 2019). Salah satu teknologi yang dikembangkan adalah teknologi media transmisi tanpa kabel atau biasa disebut *Wireless*. *Wireless* merupakan jaringan yang memanfaatkan gelombang radio sebagai

media transmisi dalam pengiriman data (Samsumar & Gunawan, 2017). Namun perlu diketahui bahwa *wireless* masih cukup rentan terhadap pencurian hak akses seperti pencurian *username* dan *password*. Contoh lainnya yaitu serangan DOS (*Denial of Service*) yang dapat menghabiskan sumber daya dan juga melakukan pencurian data sehingga mengakibatkan kerugian. Maka dari itu, Keamanan jaringan *wireless* sebagai bagian dari sistem menjadi sangat penting dalam menjaga validalitas dan integritas data (Alamsyah et al., 2020). Jika tidak ditangani dengan benar, *hacker* akan dengan mudah mengeksploitasi kemampuan mereka dalam jaringan dan pada akhirnya menimbulkan kerugian bagi pengguna.

Firewall merupakan suatu pelindungan yang diterapkan dalam software maupun hardware yang bersifat melindungi baik dengan menyaring atau membatasi suatu segmen jaringan yang berlalu (Widianto & Sulistyo, 2021). Pengaplikasian *firewall* ini sangatlah beragam, salah satunya yaitu *packet filtering* dengan mengawasi jalannya lalu lintas jaringan. Fitur ini digunakan dalam *iptables* pada linux. Hal ini disebabkan bahwa *iptables* berfungsi sebagai pemblokiran akses menuju *server* sehingga tidak terjadi penghabisan sumber daya pada *server*

Dalam sebuah jaringan, tentunya, memerlukan *administrator* sebagai pemantauan *server* dengan melihat kinerja *server* yang sedang berjalan(Suwanto et al., 2019). Apabila tidak ada sistem yang baik pada jaringan, maka serangan dapat dengan mudah menyerang jaringan tersebut. Beberapa serangan yang sering terjadi di jaringan sebagai berikut: DOS (*Denial of Service*), *Port Scanning*, dan *Ping Attack*. Serangan berikut dilakukan oleh penyerang melalui paket jaringan untuk mendapatkan informasi dari jaringan yang diserang dan juga menyebabkan komputer menjadi *crash* (Marta et al., 2020). Serangan dilakukan dengan mencari *port* terbuka di *server* dan kemudian melakukan serangan pada jaringan tersebut.

Beberapa penelitian yang sudah dilakukan dalam *Intrusion Detection System* adalah penelitian Santoso ,(2019), yang mengimplementasi keamanan jaringan menggunakan IDS dan *Iptables* sebagai antisipasi serangan *ping attack*, *port scanning*, serta *Denial of Service*. Hasil yang diperoleh semua serangan berhasil di blokir dan Keseluruhan sistem IDS dapat berjalan dengan baik. Kemudian penelitian selanjutnya menurut Pratama & Handayani, (2019), yang juga menggunakan IDS untuk dapat mengetahui mendeteksi serangan yang terjadi dalam jaringan tersebut. Dari hasil penelitian diatas, dapat disimpulkan bahwa peran IDS dan *Iptables* sangat penting dalam antisipasi serangan.

Penerapan *Intrusion Detection System* (IDS) yang merupakan alat sebagai pemantauan lalu lintas jaringan baik *inbound* maupun *outbond*. Jika aktivitas yang mencurigakan ditemukan dalam jaringan, IDS akan memberikan peringatan kepada sistem (Husain et al., 2018). Hal ini menjadi penyebab penulis menggunakan IDS dalam penelitian ini. IDS menggunakan 2 teknik dalam mendeteksi serangan, yaitu: menggunakan *Anomaly Based* dan *Signature Based* (Fauzi et al., 2019). Teknik *Anomaly Based* melibatkan pola lalu lintas terhadap serangan oleh penyerang. Lalu lintas tersebut adalah penggunaan *bandwidth* yang biasa digunakan oleh protokol, *port*, dan perangkat yang terhubung. Sedangkan *Signature Based* adalah teknik yang mendeteksi serangan baru yang belum ada dalam *database*. Teknik ini akan menganggap serangan apabila memiliki paket yang sama dalam database (Alviana & Sumitra, 2018). Sistem IDS hanya berfungsi pada lalu lintas jaringan dan hanya memberikan peringatan. Untuk itu diperlukan sebuah alat *opensource* yang dapat memberikan penjelasan lebih jelas mengenai serangan yang terjadi dalam jaringan.

Wireshark adalah alat yang digunakan oleh *administrator* dalam menangkap paket data yang sedang berjalan (Susianto & Rachmawati, 2018). Alat-alat ini berkembang berkat kontribusi para ahli jaringan dari seluruh dunia dan merupakan kelanjutan dari *Gerald Combs* dimulai pada tahun 1998. *Network Analyzer* sering kali menjadikan *wireshark* sebagai alat dalam menangkap dan menganalisa paket jaringan yang berlalu. *Tools opensource* ini juga dilengkapi dengan tampilan grafis yang cukup menarik sehingga memudahkan dalam pemantauan aktifitas jaringan. *Wireshark* sendiri juga digunakan sebagai penyadapan data pada jaringan komputer (Irawan, 2017). Dalam peningkatan keamanan jaringan *wireless*, diperlukan sebuah metode pengembangan sistem yaitu

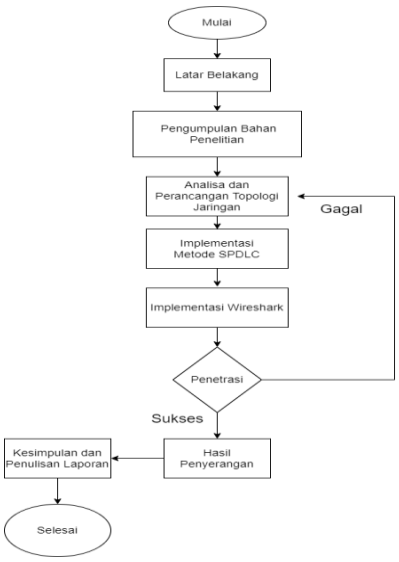
Security Policy Development Life Cycle (SPDLC).

Metode SPDLC merupakan metode yang melakukan tahap secara berskala untuk memperbarui organisasi jaringan (Lukman & Suci, 2020). Pengembangan sistem SPDLC dilakukan dengan 5 tahap, yaitu: Tahap *Analysis* yang berfungsi untuk menganalisa spesifikasi sistem yang akan dibangun, Tahap *Desain* yang berfungsi untuk menciptakan desain topologi jaringan dari sistem keamanan, Tahap *Implementation* berfungsi untuk menerapkan detail dari rancangan topologi pada sistem, Tahap *Enforcement* yang berfungsi untuk melakukan proses pengujian untuk memastikan bahwa sistem telah sesuai dengan spesifikasi rancangan, dan yang terakhir Tahap *Enchancement* berfungsi untuk melakukan perbaikan terhadap sistem yang telah dibangun (Dahlan & Zulianto, 2019). Penulis menggunakan metode ini dikarenakan metode ini tepat untuk penyajian tahapan development system yang berhubungan dengan keamanan jaringan.

2. METODE PENELITIAN

2.1 ALUR PENELITIAN

Berikut merupakan tahapan penelitian yang dirancang dalam sebuah alur penelitian. Alur penelitian ini bertujuan untuk memberikan gambaran mengenai tahapan yang akan dilalui dalam penelitian. Berikut adalah gambaran alur penelitian yang dapat dilihat pada Gambar 1 sebagai berikut:



Gambar 1. Alur Penelitian

Berikut penjelasan mengenai tahapan alur penelitian diatas yaitu:

1. Tahapan Latar Belakang, tahap ini penulis merangkum masalah dan mencari solusi atau penyelesaian dari permasalahan tersebut.
2. Tahap Penyusunan Pengumpulan Bahan Penelitian, penulis mencari jurnal yang akan dijadikan sebagai referensi dalam pembuatan penelitian serta mengumpulkan bahan-bahan yang akan digunakan selama penelitian berlangsung.
3. Analisa dan Perancangan Topologi Jaringan, tahap ini penulis akan menganalisa alat serta merancang topologi jaringan yang akan digunakan untuk penelitian.
4. Implementasi Metode SPDLC, tahap ini penulis akan melakukan konfigurasi sistem yang diperlukan selama penelitian.
5. Implementasi *Wireshark*, pada tahap ini dilakukan instalasi software untuk memantau protocol jaringan yang sedang berlangsung.
6. Penetrasi, tahap ini merupakan tahap penentuan keberhasilan dan kegagalan dari perancangan yang telah dibuat. Jika hasilnya gagal, maka akan kembali kesistem perancangan dan analisa.
7. Hasil Penyerangan, setelah melakukan penetrasi kita bisa melihat hasil analisa serangan pada tahap ini melalui *Packet Capture* pada *Wireshark.*
8. Kesimpulan dan Penulisan Laporan, pada tahap ini, penulis akan memberikan penjelasan mengenai hasil kegiatan yang telah dilakukan serta penulisan laporan.

2.2 METODE SECURITY POLICY LIFE CYCLE (SPDLC)

Dalam penelitian ini, metode yang digunakan adalah Metode *Security Policy Development Life Cycle* (SPDLC) yang melakukan proses secara berskala dan melakukan perbaruan dari sistem jaringan. Berikut ini penulis menjelaskan mengenai tahapan metode yang digunakan yaitu:

1. Analysis

Tahap ini melakukan Perumusan masalah dan Pengumpulan data berupa informasi tentang IDS dalam peningkatan

keamanan jaringan dan serangan yang terjadi. Hasil informasi yang di dapatkan dijadikan sebagai landasan dalam pemecahan masalah Ilmiah.

2. *Design*
   Membuat perancangan berupa topologi jaringan yang akan dibangun dan merancang penggunaan sistem operasi dan aplikasi pada *server* dan *client*.
3. *Implementation*
   Kemudian tahap ini akan dilakukan implementasi dari rancangan topologi jaringan yang telah dibuat yaitu dengan melakukan instalasi perangkat yang dibutuhkan dan menkonfigurasi *software* yang diperlukan. Detail Rancangan akan digunakan sebagai intruksi agar pembangunan sistem dapat berjalan dengan relavan.
4. *Enforcement*
   Pada tahap ini, akan dilakukan pengujian sistem sesuai dengan perencanaan awal. Pertama, sistem intruksi akan dilakukan percobaan yang bertujuan untuk mengetahui serangan yang dilakukan serta port yang digunakan oleh *attacker*. Kedua, pengujian *wireshark* untuk mendapatkan hasil bagaimana sebuah *packet* data dengan *protocol* tertentu bergerak pada *flow graph*.
5. *Enchancement*
   Pada tahap ini, penulis melakukan perbaikan sistem yang telah dibangun meliputi kesalahan yang terjadi pada penelitian sebelumnya, melakukan penginkatan fungsionalitas atas komponen spesifik dan melakukan pembaruan sistem.

3. HASIL DAN PEMBAHASAN

3.1 ANALYSIS

Seperti yang diketahui, saat ini banyak sekali kasus-kasus penyerangan server perusahaan yang mengakibatkan hilangnya data dan *server* menjadi *down*. Maka dari itu, penulis akan mencoba membuat analisis tentang antisipasi serangan yang menyebabkan kerugian bagi kita semua. Hal pertama yang dilakukan yaitu:

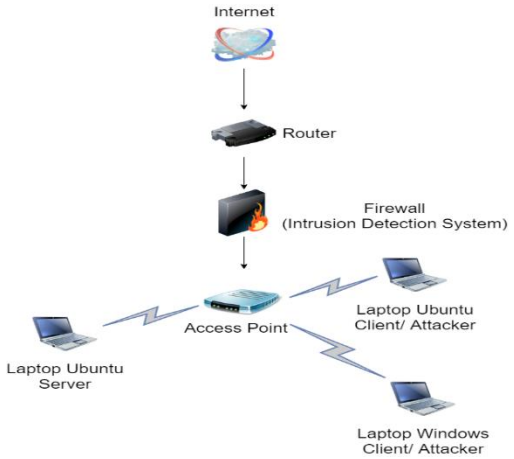
1. Spesifikasi Software dan Hardware

Spesifikasi Software		Spesifikasi Hardware	
OS Ubuntu	Versi 18.04LTS	Laptop ASUS K45VD PC Server	-Processor Intel(R) Core (TM) i5-3230M -Ram 6GB -*Harddisk* 223.57 GB
Vmware Workstation	15 Layer	PC Client Ubuntu	II
Snort	Versi 2.9.18.1	PC Client Windows	II
Wireshark	Versi 3.4.7	Access Point	Huawei HG8245H5
-	-	Router dan Kabel UTP	Huawei WS319

Gambar 2. Spesifikasi *Software* dan *Hardware*

3.2 DESIGN

Perancangan topologi ini dibuat berdasarkan konsep dan gambaran dari perangkat yang ada. Rancangan topologi dapat dilihat pada gambar 3 dibawah ini:



Gambar 3. Topologi Jaringan

Seperti yang terlihat pada gambar diatas, bahwa jaringan akan melewati perangkat *router* yang berfungsi sebagai penghubung antar jaringan komputer untuk meneruskan data dari satu jaringan ke jaringan lain. Kemudian disaring oleh *firewall* IDS untuk memastikan jaringan dalam keadaan baik. Setelah itu akan melewati *Access point* yang bersifat sebagai pemancar jaringan ke laptop yang ada. Lalu dapat kita lihat 3 laptop yang yang terhubung dalam satu jaringan. Laptop *server* akan dilakukan penginstallan *vmware*, *ubuntu*, dan *snort*. Kemudian pada laptop *attacker* akan dilakukan penginstallan *hping3* sebagai *tools* dalam melakukan penyerangan.

Berikut IP Address dari sistem yang akan dibangun dapat dilihat pada gambar 4.

Device	IP Address	Subnet Mask	Gateway
Server	192.168.220.135	255.255.255.0	192.168.220.1
Client Ubuntu	192.168.220.134	255.255.255.0	192.168.220.1
Client Windows	192.168.220.1	255.255.255.0	-
Access Point	-	255.255.255.0	-
Router	192.168.100.4	255.255.255.0	192.168.100.1

Gambar 4. IP Address

3.3 IMPLEMENTATION

IDS atau sistem intruksi deteksi yang digunakan dalam penelitian ini adalan *snort* menggunakan *ubuntu* versi 18.04 LTS. Berikut adalah beberapa proses yang harus dilakukan sebelum memulai tes yang terlihat pada Gambar 5 sebagai berikut:

Menginstalasi Paket-Paket dari Repositori Ubuntu
sudo apt-get install -y build-essential libpcre3-dev zlib1g-dev libluajit-5.1-dev libpcap-dev openssl libssl-dev bison flex libdumbnet -dev liblzma-dev libnghttp2-dev libdnet autoconf libtool
Membuat Direktori
mkdir ~/snort_src
cd ~/snort_src
#Instalasi DAQ
wget https://snort.org/downloads/snort/daq-2.0.7.tar.gz
tar -xvzf daq-2.0.7.tar.gz
cd daq-2.0.7 dan autoreconf -f -i
./configure && make && sudo make install

Gambar 5. Instalasi Paket Snort dan DAQ Source

Dari gambar diatas, terlihat perintah *apt-get install* yang bertujuan untuk menginstal paket baru pada *root* agar semua sistem dapat diakses oleh *root*. Setelah itu melakukan pembuatan *folder* baru dengan perintah mkdir. Selanjutnya penginstallan DAQ (*Data Acquisition Library*) yang berfungsi sebagai pengaktif semua fitur inline pada *snort*. Namun pastikan untuk menginstall DAQ dengan menggunakan *wget* terlebih dahulu.

Jika terjadi masalah pada penginstallan prasyarat *snort* seperti dibawah ini:
E: Could not get lock /var/lib/dpkg/lock - open (11 Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/) is another process using it?
Maka lakukan cara berikut ini: (jika tidak ada masalah, lanjutkan ke konfigurasi *snort*)

Penanganan masalah diatas
ps aux | grep apt
sudo rm /var/lib/dpkg/lock
sudo dpkg –configure -a dan reboot ubuntu

Gambar 6. Solusi Masalah *Unable to Lock*

Setelah menginstal paket dan sumber DAQ pada *snort*, langkah selanjutnya adalah memasang *snort*. Beberapa versi *snort* dapat ditemukan pada *website* berikut ini *https://www.snort.org/downloads*. Namun, dalam penelitian ini, penulis menggunakan versi *snort* 2.9.18. Seluruh instalasi sebagai *root* sehingga setiap akses atau izin di diakses oleh *admin*. Berikut adalah proses untuk menginstal *snort* terlihat di Gambar 7 sebagai berikut:

#Instalasi *Snort*
cd ~/snort_src
wget https://snort.org/downloads/snort/snort-2.9.18.1.tar.gz
tar -xvzf snort-2.9.18.1.tar.gz
cd snort-2.9.18.1
./configure --enable-sourcefire && make sudo install
sudo apt-get install snort
(*Interface* sesuaikan dengan *ip addr* dan *ip address* komputer *server* anda)
cd ~/snort_src/snort-2.9.18.1
snort -V (Menunjukkan versi snort) dan kembali ke cd ~/snort_src
#Memperbarui *Shared Library*
sudo ldconfig
sudo ln -s /usr/local/bin/snort /usr/sbin/snort

Gambar 7. Konfigurasi *Snort*

Selanjutnya, buat pengguna baru yang tidak memiliki hak istimewa dan kelompok pengguna baru untuk menjalankan *snort* tanpa akses *root*. Kemudian buat struktur *folder* untuk mengakomodasi konfigurasi *snort* dan memberikan izin ke direktori / *folder* yang baru dibuat. Setelah itu, buat file baru ke dalam *white_list.rules* dan *black_list.rules* dalam aturan lokal serta salin file konfigurasi dari *folder* unduhan pada Gambar 8 di bawah ini:

#Melakukan Settingan *Username* dan Struktur *Folder*
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
sudo mkdir -p /etc/snort/rules
sudo mkdir /var/log/snort
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules
sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
sudo touch /etc/snort/rules/white_list.rules
sudo touch /etc/snort/rules/black_list.rules
sudo touch /etc/snort/rules/local.rules
sudo cp ~/snort_src/snort-2.9.18.1/etc/*.conf* /etc/snort
sudo cp ~/snort_src/snort-2.9.18.1/etc/*.map /etc/snort

Gambar 8. *Settingan Username dan Folder Structure*

Setelah menyelesaikan konfigurasi *folder* dan nama pengguna, lalu konfigurasikan aturan komunitas. Hal ini membuat *snort* lebih mudah digunakan. Setelah semua perintah telah dilaksanakan, maka kita dapat memeriksa dengan perintah Sudo *snort* -T -c / *snort* / *snort.conf* (Gambar 9).

#Konfigurasi *Community Rules Snort*
cd ~/snort_src
wget https://www.snort.org/rules/community -O ~/community.tar.gz
sudo tar -xvf ~/community.tar.gz -C ~/
sudo cp ~/community-rules/* /etc/snort/rules
sudo sed -i 's/include \$RULE_PATH/#include \$RULE_PATH/' /etc/snort/snort.conf
#Konfigurasi Jaringan dan *Rules Sets* (cari perintah dibawah ini dan sesuaikan dengan *snort* anda)
sudo nano /etc/snort/snort.conf
ipvar HOME_NET 192.168.109.0/24)
ipvar EXTERNAL_NET !$HOME_NET)
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
(Cari unified2 kemudian hilangkan tanda # dan ganti ke perintah ini pada output unified2: filename snort.log, limit 128)
(Cari dan hilangkan tanda # pada include $RULE_PATH/local.rules)
(Tambahkan include $RULE_PATH/community.rules)
#Testing Konfigurasi *Snort*
sudo snort -T -c /etc/snort/snort.conf (jika terdapat direktori yang hilang, masuk ke snort-2.9.18.1 dan coba lagi perintah ./configure –enable-sourcefire && make && sudo make install)

Gambar 9. Instalasi dan Konfigurasi *Community Rules*

Berikut tampilan setelah *snort* berhasil terinstall:

```
server@server:~/snort_src/snort-2.9.18.1$ snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11
```

Gambar 10. Tampilan Versi *Snort*

```
           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.2  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>

Total snort Fixed Memory Cost - MaxRss:55700
Snort successfully validated the configuration!
```

Gambar 11. Tampilan *Snort.conf* Berhasil Ter-Install

3.4 ENFORCEMENT

Selanjutnya, penulis akan mensimulasikan komponen *snort* yang terpasang dan konfigurasi sebelumnya. pengujian *snort* dilakukan dengan menggunakan *rules* yang ditulis dalam perintah sederhana. Berikut adalah perintah yang digunakan untuk memeriksa fungsi *snort*: *alert icmp any -> $HOME_NET any (msg:"Testing"; sid:10000001; rev:001;).* Aturan di atas menginstruksikan *snort* untuk mengeluarkan peringatan tentang lalu lintas protokol *ICMP* dari mesin dan *port* apa pun. Kemudian mulai *snort* dengan opsi konsol -A untuk mencetak peringatan ke *stdouts*. Kita diminta untuk memilih *interface* jaringan yang benar dengan alamat Ip *server public* Anda misalnya. ens33. Perintah untuk mencetak peringatan adalah: sudo *snort* -A console -i ens33 -u *snort* -g *snort* -c /etc/*snort*/*snort.conf*. jika Anda tidak tahu *interface* jaringan anda atau lupa dapat memeriksanya dengan perintah ip *addr* di konsol.

```
2.168.220.129 -> 192.168.220.1
10/12-14:23:34.539013  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.1 -> 192.168.220.129
10/12-14:23:34.539257  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.129 -> 192.168.220.1
10/12-14:23:35.545469  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.1 -> 192.168.220.129
10/12-14:23:35.545667  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.129 -> 192.168.220.1
10/12-14:23:36.551827  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.1 -> 192.168.220.129
10/12-14:23:36.552426  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.129 -> 192.168.220.1
10/12-14:24:40.827497  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.254 -> 192.168.220.130
10/12-14:24:40.827632  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.130 -> 192.168.220.254
10/12-14:24:40.828257  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.254 -> 192.168.220.130
10/12-14:24:40.828324  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.130 -> 192.168.220.254
10/12-14:24:44.018659  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.254 -> 192.168.220.131
10/12-14:24:44.718343  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.254 -> 192.168.220.131
10/12-14:24:44.718712  [**] [1:10000001:1] Testing [**] [Priority: 0] {ICMP} 19
2.168.220.131 -> 192.168.220.254
```

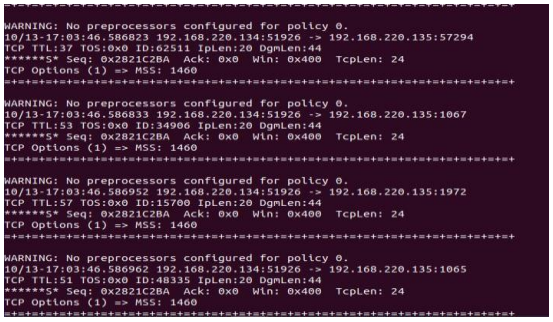
Gambar 12. *Test Alert*

Setelah itu, penulis menguji efektivitas fungsi interkoneksi IDS secara keseluruhan. sistem ini berfungsi untuk memberikan *alert* apabila mendeteksi serangan pada lalu lintas jaringan. Tes ini dilakukan dengan beberapa percobaan serangan sebagai berikut:

1. NMAP (*Port Scanning*)
   Kasus ini dilakukan pengujian Nmap dengan memindai *port* untuk menampilkan port yang terbuka di *snort*. Setelah mendapatkan *port* yang terbuka, penyerang/*Attacker* akan masuk ke *server* dan menyerang dengan *IP Address* 192.168.220.135. Setelah melakukan perintah tersebut, terdapat *port* yang terbuka yaitu *port* 22. Namun, sebelum itu kita harus menginstal Nmap terlebih dahulu untuk melakukan *port scanning*. Berikut adalah hasil yang terjadi ketika serangan dilakukan yaitu:

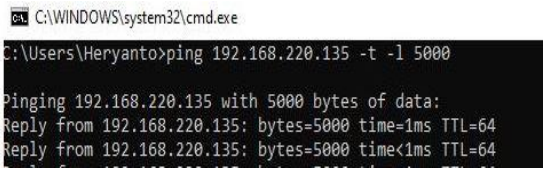
```
client@client:~/snort_src/snort-2.9.18.1$ sudo nmap 192.168.220.135
[sudo] password for client:

Starting Nmap 7.60 ( https://nmap.org ) at 2021-10-13 17:04 WIB
Nmap scan report for 192.168.220.135
Host is up (0.00049s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:0C:29:E1:E5:CD (VMware)

Nmap done: 1 IP address (1 host up) scanned in 8.49 seconds
client@client:~/snort_src/snort-2.9.18.1$
```

Gambar 13. Perintah Serangan NMAP (*Port Scanning*) dari *Client*

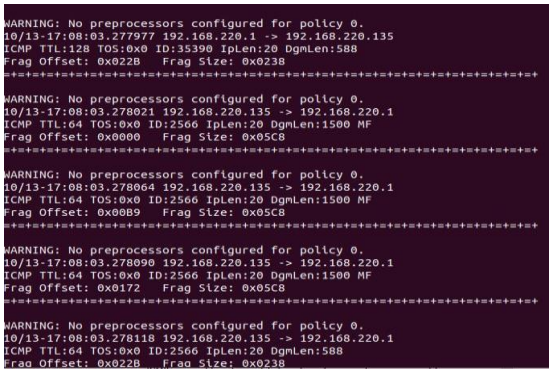


Gambar 14. Snort Menganalisis *IP Address* Serangan Nmap (*Port Scanning*)

2. Ping Attack (*ICMP Traffic*)
   Kasus ini dilakukan analisa jenis serangan berprotokol *ICMP*. Traffic ini sangat berpotensi untuk digunakan sebagai senjata penyerangan oleh penyerang karena memproduksi perintah *ping*. Bisa dilihat pada Gambar 15, *Client* melakukan serangan ping attack ke alamat IP IDS yaitu 192.168.220.135 menggunakan *Command Prompt* pada OS *Windows*. Serangan ini juga dilakukan untuk mengetahui atau memastikan apakah *host* dalam keadaan aktif.



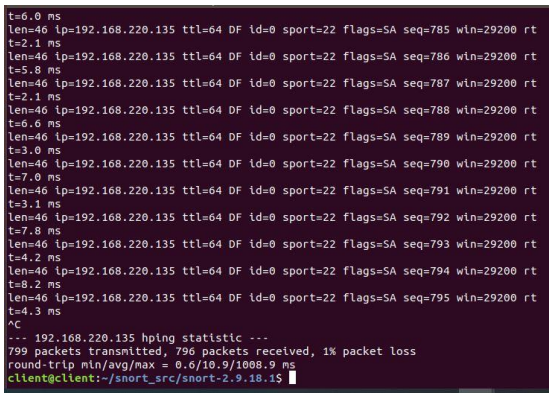
Gambar 15. Perintah Serangan *Ping Attack* (*IMCP Traffic*) dari *Client Windows*



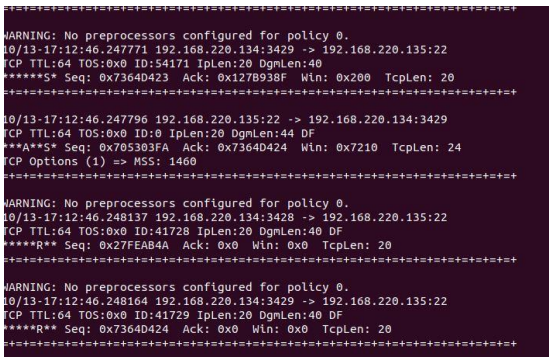
Gambar 16. *Snort* Menganalisis *IP Address* Serangan *Ping Attack* (*ICMP Traffic)*

3. DOS (*Denial of Service*)
   Pada kasus ini, Komputer yang telah terpasang *snort* akan dicoba dengan melakukan penyerangan DOS dari komputer *client*. Serangan ini akan membahayakan *user* karena mengirim sejumlah paket berbahaya dalam jaringan yang membuat layanan jaringan menjadi terhambat atau melemah. Sebelum itu, kita harus menginstall *hping3* terlebih dahulu agar bisa melakukan serangan. Setelah itu, melakukan percobaan penyerangan dengan perintah sudo *hping3* -I u3000 -S -p 22 192.168.220.135. Hasil yang terjadi ketika serangan dilakukan yaitu:

Gambar 17. Perintah Serangan DOS (*Denial of Service*) dari *Client*



Gambar 18. Proses *Client* Me-*request* DOS secara Terus Menerus Kepada *Server*



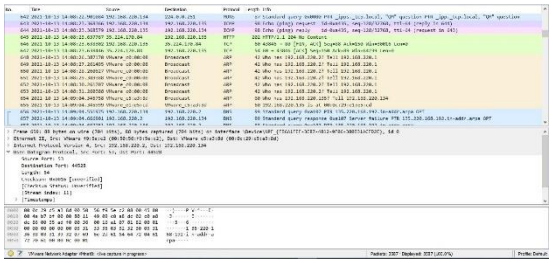
Gambar 19. *Snort* Menganalisis *IP Address* Serangan DOS (*Denial of Service*)

Setelah menyelesaikan tes serangan melalui sistem *snort*, selanjutnya penulis akan melakukan pengujian *wireshark* terhadap serangan. Saat membuka *wireshark*, pastikan untuk menjalankannya dengan *administrator* atau jalankan sebagai *Administrator*. Setelah itu, pilih *interface* jaringan yang terhubung ke

jaringan internet. Jaringan ini akan dianalisis untuk melihat aktivitas di dalamnya.

A. Penggunaan *Wireshark*

Berikut merupakan tampilan *Packet Capture wireshark*.



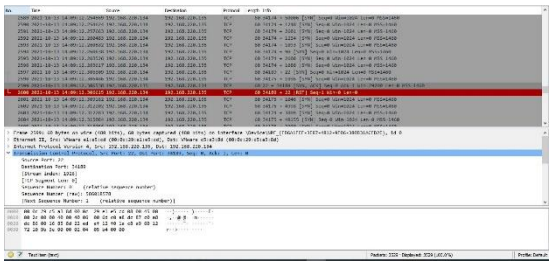
Gambar 20. Tampilan Hasil *Packet Capture Wireshark*

B. Pengujian Serangan Pada *Wireshark*

Berikut ini serangan yang di uji pada *wireshark* sebagai berikut:

1. Nmap (*Port Scanning*)

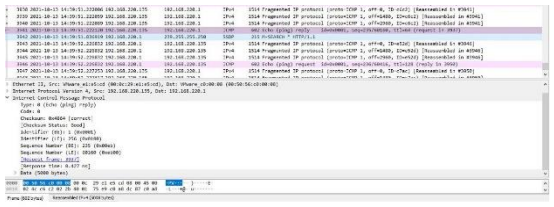
   Dalam *packet capture Wireshark*, terdapat lalu lintas yang tidak biasa dengan menunjukkan adanya tindakan serangan *port scanning* Pada kolom *source* terdapat IP dari komputer penyerang/*client* yaitu 192.168.220.134 yang melakukan serangan. Sedangkan pada kolom *destination* terdapat IP target/*server*. Dari hasil analisa, terdapat pernyataan yang memberitahukan bahwa *port* dari penyerang sedang melakukan *scanning* pada komputer target. Protokol yang digunakan yaitu TCP (*Transmission Control Protokol*). Lalu bisa dilihat pada *port* 34189 (penyerang) menuju ke *port* 22 (*Client*/Target) dan juga sebaliknya. Arti dari pengiriman paket melalui *port* tadi yaitu bahwa *port* 22 dalam keadaan terbuka dengan mengirim umpan balik ke penyerang dan siap untuk menerima koneksi dari luar.



Gambar 21. *Wireshark* Menganalisa Serangan Nmap

2. Ping Attack (*ICMP Traffic*)

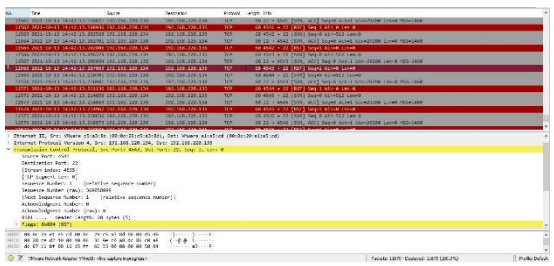
   Serangan ICMP yang dilakukan oleh penyerang dengan mengirim paket *echo request ICMP* secara berlebihan kedalam jaringan, dapat mengakibatkan sistem menjadi crash, hang ataupun *reboot*. Seperti pada Gambar 22 bahwa client terus mengirim *ICMP ping request* kepada *server*. Apabila paket-paket tersebut disatukan Kembali, maka akan melebihi batasan. Jenis ini mengakibatkan penghabisan *bandwidth*. *Server* akan terus mengirim paket *ICMP Echo Reply* dan pada akhirnya sistem server akan mengalami perlambatan.



Gambar 22. *Wireshark* Menganalisa Serangan *Ping Attack* (*ICMP Attack*)

3. DOS (*Denial of Service*)

   Dalam proses ini, penyerang mengirim ping request secara terus menerus ke *server* dengan IP *Address* 192.168.220.135. Penyerang mengirim permintaan SYN (*Synchronization*) ke *server*. Kemudian *server* dengan IP Address 192.168.220.135 akan memberi jawaban *SYN-ACK* ke *client*/penyerang dengan menggunakan *port* 22 sebagai *packet* yang dilalui. Dari hal ini, *client* tidak menerima ACK sehingga menyebabkan *server* menjadi *down*.



Gambar 23. *Wireshark* Menganalisa Serangan DOS (*Denial of Service*

C. Solusi Mengatasi Serangan pada Jarringan *Wireless*

Setelah melakukan beberapa proses penyerangan ke *server*, penulis mendapatkan data yang digunakan oleh penyerang. Data yang diperoleh yaitu penyerang menggunakan

metode alamat internet protokol, alamat MAC, dan *port* yang digunakan untuk menyerang. Untuk mengatasi serangan dari *instruder*/penyerang, penulis menggunakan sebuah *rule* yaitu *IPTABLES* dengan melakukan konfigurasi terhadap komputer server dimana konfigurasi yang dilakukan berupa pemasukan *source* komputer penyerang. Dengan melakukan konfigurasi tersebut, penyerang tidak dapat lagi melakukan aktivitas penyerangan ke komputer *client* seperti melakukan *PING* dan lain-lain.

```
server@server:~$ sudo iptables -A INPUT -s 192.168.220.134 -p icmp -j DROP
[sudo] password for server:
server@server:~$ sudo iptables -A INPUT -s 192.168.220.134 -p tcp -j DROP
server@server:~$ sudo iptables -A INPUT -s 192.168.220.134 -p udp -j DROP
server@server:~$ sudo iptables -A INPUT -s 192.168.220.1 -p icmp -j DROP
server@server:~$ sudo iptables -A INPUT -s 192.168.220.1 -p tcp -j DROP
server@server:~$ sudo iptables -A INPUT -s 192.168.220.1 -p udp -j DROP
server@server:~$
```

Gambar 24. Perintah Pemblokiran Ip *Address*

Pada Gambar 24 merupakan perintah pemblokiran *Ip address* 192.168.220.134 dan 192.168.220.1 yang menyerang komputer *server* dengan protokol *ICMP*, *TCP*, dan *UDP*. Perintah -A *INPUT* digunakan oleh penulis untuk menambah perintah *table INPUT* sehingga proses akan berada pada *table chain input*. Kemudian perintah -s 192.168.220.134 digunakan untuk menyatakan sumber komunikasi. Dan yang terakhir -p *ICMP* -j *DROP* menyatakan bahwa komunikasi dari sumber protokol *ICMP* akan dijatuhkan begitu juga dengan protokol *TCP* dan *UDP*. Setelah melakukan tersebut tulis perintah sudo *iptables* -L agar dapat melihat tampilan *block* IP *address* seperti pada Gambar 25 dibawah ini

```
server@server:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ufw-before-logging-input  all  --  anywhere             anywhere
ufw-before-input  all  --  anywhere             anywhere
ufw-after-input  all  --  anywhere             anywhere
ufw-after-logging-input  all  --  anywhere             anywhere
ufw-reject-input  all  --  anywhere             anywhere
ufw-track-input  all  --  anywhere             anywhere
DROP       icmp --  192.168.220.134      anywhere
DROP       tcp  --  192.168.220.134      anywhere
DROP       udp  --  192.168.220.134      anywhere
DROP       icmp --  192.168.220.1        anywhere
DROP       tcp  --  192.168.220.1        anywhere
DROP       udp  --  192.168.220.1        anywhere
```

Gambar 25. Tampilan Ip *Address* yang Telah di Blokir

Pada Gambar 26 ini, dapat dilihat bahwa serangan *port scanning* yang dilakukan penyerang sudah tidak bisa lagi melakukan *scanning* terhadap komputer *server* lagi. Sesuai pernyataan dari *console* komputer penyerang bahwa semua *port* yang ingin di *scanning* telah *terfiltered.*

```
client@client:~/snort_src/snort-2.9.18.1$ sudo nmap 192.168.220.135
[sudo] password for client:

Starting Nmap 7.60 ( https://nmap.org ) at 2021-10-14 13:46 WIB
Note: Host seems down. If it is really up, but blocking our ping probes, try -P
n
Nmap done: 1 IP address (0 hosts up) scanned in 0.66 seconds
client@client:~/snort_src/snort-2.9.18.1$
```

Gambar 26. Tampilan *Client* yang Berhasil terblokir Saat Melakukan Serangan Nmap

Pada Gambar 27, menandakan bahwa proses pemblokiran IP *Address* telah berhasil dilakukan. Terlihat saat melakukan *ping attack* ke komputer target, *console* mengeluarkan tulisan RTO atau *Request Time Out* yang artinya penyerang sudah tidak bisa lagi melakukan *ping* ke alamat tujuan. Begitu pula dengan Gambar 28 yang sudah tidak bisa lagi melakukan serangan DOS (*Denial of Service*) ke komputer target.

```
client2@client2:~/snort_src/snort-2.9.18.1$ ping 192.168.220.131
PING 192.168.220.131 (192.168.220.131) 56(84) bytes of data.
^C
--- 192.168.220.131 ping statistics ---
61 packets transmitted, 0 received, 100% packet loss, time 61425ms
```

Gambar 27. Tampilan *Client* yang Berhasil Terblokir Saat Melakukan *Ping Attack*

```
client@client:~/snort_src/snort-2.9.18.1$ sudo hping3 -i u3000 -S -p 22 192.168
.220.135
HPING 192.168.220.135 (ens33 192.168.220.135): S set, 40 headers + 0 data bytes
^C
--- 192.168.220.135 hping statistic ---
3177 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
client@client:~/snort_src/snort-2.9.18.1$
```

Gambar 28. Tampilan *Client* yang Berhasil Terblokir Saat Melakukan DOS (*Denial of Service*)

Berikut merupakan kesimpulan data hasil penyerangan yang telah dilakukan sebagai berikut:

Tipe Serangan	Source	Destination	Hasil Serangan	Berhasil/Gagal	Blokir	Terdeteksi Oleh Wireshark
Nmap (Port Scanning)	192.168.220.134	192.168.220.135	Mendapatkan 1 Port SSH yang terbuka	Berhasil	Berhasil	Berhasil
Ping Attack (ICMP Attack)	192.168.220.1	192.168.220.135	Pengiriman ping berupa 5000 Byte kepada server	Berhasil	Berhasil	Berhasil
DOS (Denial of Service)	192.168.220.134	192.168.220.135	Terus menerima request dari client	Berhasil	Berhasil	Berhasil

Gambar 29. Data Hasil Penyerangan

3.5 ENHANCEMENT

Fase ini meliputi aktivitas perbaikan sistem dimana memiliki sejumlah tujuan seperti:

1. Perbaikan kesalahan dari sistem yang telah dibangun.

2. Melakukan pembaruan terhadap sistem baru
3. Melakukan adaptasi terhadap sejumlah platform dan teknologi baru.

4. SIMPULAN

Dari hasil penelitian yang telah diimplementasikan oleh penulis menggunakan *Instrusion Detection System* dengan *wireshark*, secara langsung penulis mendapatkan pengetahuan tentang cara pemakaian dan penerapan dalam peningkatan keamanan jaringan wireless. Penulis juga menyimpulkan bahwa sistem IDS (*Intrusion Detection System*) dalam mendeteksi serangan yaitu dengan melakukan pemantauan *traffic* jaringan melalui hasil *capture* sistem *snort* dapat berjalan dengan baik. Konfigurasi *snort* telah berhasil diimplementasikan dengan melakukan percobaan serangan *Ping Attack* (*ICMP Traffic*), Nmap (*Port Scanning Attack*), dan DOS (*Denial of Services*).

Pencegahan yang dilakukan menggunakan *Iptables* untuk mengatasi serangan dari penyerang/*Instruder* dimana *rule* tersebut akan melakukan pemblokiran IP *Address* penyerang dapat berjalan dengan baik. dan yang terakhir *Wireshark* dapat melakukan pemantauan *protocol* jaringan dan menganalisa setiap paket-paket melalui *flow graph* sehingga keamanan data dan informasi dapat terjamin. Selain itu juga terdapat saran seperti perlunya peningkatan kinerja sistem IDS *snort* dan perlunya peningkatan *wireshark* pada *OS Windows* sehingga mempermudah Pengguna dalam pemakaiannya.

5. DAFTAR PUSTAKA

Alamsyah, H., Riska, & Al Akbar, A. (2020). Analisa Keamanan Jaringan Menggunakan Network Intrusion Detection and Prevention System. *JOINTECS (Journal of Information Technology and Computer Science)*, *5*(1), 17.

Alviana, S., & Sumitra, I. D. (2018). Analisis Pengukuran Penggunaan Sumber Daya Komputer pada Intrusion Detection System dalam Meminimalkan Serangan Jaringan. *Komputa : Jurnal Ilmiah Komputer Dan Informatika*, *7*(1), 27–34.

Dahlan, & Zulianto, A. (2019). Perancangan Keamanan Jaringan Komputer pada Layer Application Berbasis Intrusion Prevention System (IPS) yang Di Integrasikan dengan Acess Control List (ACLs). *Scientia Regendi*, *I*(1), 86–96.

Fauzi, M. A., Hanuranto, A. T., & Setianingsih, C. (2019). Sistem Deteksi Intrusi Menggunakan Algoritma Genetik Pada Intrusion Detection System Using Genetic Algorithm on Dos Attack in Tcp and Udp Protocol. *6*(2), 4800–4807.

Husain, M. S., Aksara, L. F., & Ransi, N. (2018). Implementasi Keamanan Server Pada Jaringan Wireless Menggunakan Metode Intrusion Detection and Prevention System (Idps) (Studi Kasus : Techno'S Studio). *SemanTIK*, *4*(2), 11–20.

Irawan, D. (2017). Analisis dan Penyadapan Transmisi Paket Data Jaringan Komputer Menggunakan Wireshark. *Analisis Dan Penyadapan Transmisi Paket Data Jaringan Komputer Menggunakan Wireshark*, *7*(1), 1–5.

Lukman, & Suci, M. (2020). Analisis Perbandingan Kinerja Snort dan Suricata Sebagai Intrusion Detection System dalam Mendeteksi Serangan Syn Flood Pada Web Server Apache. *Jurnal Teknologi Informasi*, *XV*(2), 6–15.

Marta, I. K. K. A., Hartawan, I. N. B., & Satwika, I. K. S. (2020). Analisis Sistem Monitoring Keamanan Server Dengan Sms Alert Berbasis Snort. *Information System and Emerging Technology Journal*, *1*(1), 25.

Mukti, F. S., & Junikhah, A. (2019). Prediksi Cakupan Area untuk Jaringan Wireless Indoor Kampus berdasarkan Penempatan Access Point. *Jurnal Intake : Jurnal Penelitian Ilmu Teknik Dan Terapan*, *10*(2), 73–78.

Pratama, I. P. A. E., & Handayani, N. K. M. (2019). Implementasi Ids Menggunakan Snort Pada Sistem. *3*(1), 176–181.

Samsumar, L. D., & Gunawan, K. (2017). Analisis dan Evaluasi Tingkat Keamanan Jaringan Komputer Nirkabel (Wireless Lan); Studi. *Ilmiah Teknologi Informasi Terapan*, *IV*(1), 73–82.

Santoso, J. D. (2019). Keamanan Jaringan Nirkabel Menggunakan Wireless

Intrusion Detection System. *Infos*, *1*(3), 44–50.

Susianto, D., & Rachmawati, A. (2018). Implementasi dan Analisis Jaringan Menggunakan Wireshark, Cain and Abels, Network Minner (Studi Kasus : AMIK Dian Cipta Cendikia). *Jurnal Cendikia*, *XVI*, 120–125.

Suwanto, R., Ruslianto, I., & Diponegoro, M. (2019). Implementasi Intrusion Prevention System (IPS) Menggunakan Snort dan IPTable Pada Monitoring Jaringan Lokal Berbasis Website. *Jurnal Komputer Dan Aplikasi*, *07*(1), 97–107. http://jurnal.untan.ac.id/index.php/jcskommipa/article/view/32690/75676581071

Widianto, T. K., & Sulistyo, W. (2021). Implementasi Iptables Firewall dan Intrusion Detection System Untuk Mencegah Serangan DDoS Pada Linux Server. *MEANS (Media Informasi Analisa Dan Sistem)*, *6*(1), 19–23.