Pemodelan HTTP *Request* dengan *Regular Expression* untuk Deteksi Serangan *Slow* HTTP DoS

Modeling of HTTP Request with Regular Expression for Slow HTTP DoS Attack Detection

Ramlan[1)], Avinanta Tarigan[2)]

[1]Program Studi S2 Teknik Informatika Universitas AMIKOM Yogyakarta
[2]Departemen Computer Engineering Universitas Gunadarma

ramlankctux@gmail.com[1)], avinanta@gmail.com[2)]

Diterima : 18 Februari 2019 || Revisi : 15 Maret 2019 || Disetujui: 16 Maret 2019

Abstrak – Menjamin ketersediaan layanan berbasis *web* merupakan hal yang sangat penting, namun kenyataannya ada banyak jenis ancaman terhadap layanan *web*. Hal ini terutama yang berasal dari internet, salah satu jenis ancaman yang terkenal adalah serangan *Slow HTTP DoS*. Penelitian tentang *Intrusion Detection System* (*IDS*) terhadap serangan *Slow HTTP DoS* sudah dilakukan oleh beberapa peneliti sebelumnya, namun selalu terdapat kemungkinan untuk meningkatkan tingkat akurasi deteksi dan meminimalkan tingkat *False Positive*, yang juga harus dipertimbangkan adalah metode tersebut seharusnya dapat digunakan pada skala jaringan kecil dan menengah. Pada penelitian ini digunakan metode alternatif yang lain untuk memperoleh hasil deteksi yang lebih baik untuk serangan *Slow HTTP DoS*, yaitu dengan merumuskan pola deteksi dengan metode pemodelan *HTTP Request* berbasis *Regular Expression*. Pola deteksi yang dibuat didasarkan pada trafik *HTTP Request* dari *Web Browser* yang populer digunakan, lalu dicari kemiripannya menggunakan algoritma Needleman Wunsch. Pola *HTTP Request* dari *Web Browser* yang dihasilkan kemudian dinegasikan pada beberapa bagian penting dari *header HTTP Request*, lalu diubah ke dalam bentuk *Regular Expression*. Pola tersebut kemudian dimasukkan pada konfigurasi *L7-Filter* yang merupakan bagian dari *Netfilter*. Metode ini terbukti berhasil digunakan untuk mendeteksi serangan *Slow HTTP DoS* dengan tingkat akurasi 100% dan tingkat *False Positive* sebesar 0%.

Kata Kunci: *slow HTTP DoS, IDS, Needleman Wunsch, regular expression, netfilter*

Abstract *– The availability of Web Service is the most important thing to be guaranteed, but there are many threats to the Web Service particularly from the internet, one of the famous methods is Slow HTTP DoS Attack. There are many research projects about this topic before, but there is always the possibility to increase the accuracy rate and minimizing the False Positive Rate and should be considered to use it at the small and medium scale of network infrastructure. In this research, another IDS method was used to find a better result for Slow HTTP DoS Attack detection through modeling HTTP Request based on Regular Expression. Detection patterns made from HTTP Request Traffic of some popular Web Browsers then looked for the similarity of the HTTP Request Traffic using Needleman Wunsch algorithm. This pattern was negated at the important part of HTTP Request headers, then converted to Regular Expression. New Pattern in Regular Expression was inserted to the L7-Filter that part of Netfilter. This method has been proved to detect Slow HTTP DoS Attack with 100% accuracy and False Positive Rate 0%.*

Keywords: *Needleman Wunsch, netfilter, regular expression, slow HTTP DoS, IDS*

PENDAHULUAN

Penyediaan informasi dalam bentuk halaman web pada layanan *Hyper Text Transfer Protocol* (HTTP) saat ini sudah merupakan sebuah kebutuhan, karena akan memudahkan dan mempercepat penyebaran informasi di ruang publik. Dalam prosesnya ternyata ada saja hambatan yang dialami, tidak hanya berasal dari dari dalam, misalnya minimnya infrastruktur dan koneksi, namun juga dari luar, misalnya gangguan berupa serangan yang ditujukan pada layanan HTTP yang berasal dari internet. Hal tersebut akan mengganggu ketersediaan akses layanan informasi. Menurut Stewart dan kawan-kawan (2012), salah satu jenis serangan yang populer digunakan untuk melumpuhkan layanan HTTP adalah serangan *Denial of Service* (DoS).

Informasi yang dihimpun oleh Kaspersky (2018) di akhir tahun 2017 menjelaskan bahwa telah terjadi serangan DoS yang cukup signifikan dan menargetkan sekitar 84 negara di dunia. Sekitar 12,70% serangan DoS tersebut ditujukan kepada layanan HTTP. Serangan DoS menjadi pilihan utama karena teknik ini tidak mengharuskan penyerang memiliki keahlian tinggi, bermodalkan aplikasi serangan DoS yang banyak bertebaran di internet maka layanan HTTP

dapat dilumpuhkan (Kumar, 2016; Sathwara & Parekh, 2017). Salah satu jenis serangan DoS yang ditujukan pada layanan HTTP dan cukup populer adalah serangan *Slow HTTP DoS* (Giralte dkk., 2013; Yevsieieva & Helalat, 2017). Gupta & Grover (2016) menyatakan bahwa teknik serangan menggunakan *Slow HTTP DoS* menjadi pilihan, karena tidak membutuhkan sumber daya koneksi (*Bandwidth*) yang besar dan cukup sulit untuk dideteksi.

Ada dua jenis metode *Intrusion Detection System* (IDS) yang direkomendasikan oleh para peneliti untuk deteksi *Slow HTTP DoS*. Metode pertama adalah metode *Anomaly-Based* yang akan mendeteksi serangan berdasarkan penyimpangan yang terjadi terhadap pola trafik normal, salah satu bentuk metode ini adalah dengan *Machine Learning* (Cambiaso dkk., 2016). Metode *Signature-Based* digunakan untuk mendeteksi serangan berdasarkan pola unik dari trafik serangan, umumnya menggunakan *Snort* (Santoso dkk., 2016) dan *Suricata* (Akbar dkk., 2016).

Idhammad dan kawan-kawan (2018) dalam penelitiannya terkait metode deteksi serangan *Slow HTTP DoS*, melakukan penghitungan estimasi entropi untuk membedakan trafik normal dan trafik serangan. Hasil penelitian menunjukkan bahwa tingkat akurasi deteksi sebesar 99.54% dan tingkat *False Positive Rate* (FPR) terjadi fluktuasi antara 0,4% hingga 0,7%. Kelemahan dari deteksi berbasis anomali adalah implementasi secara *real-time* sulit dilakukan pada skala jaringan kecil dan menengah karena membutuhkan mesin dengan spesifikasi tinggi.

Hampir serupa dengan penelitian Idhammad dkk. (2018), penelitian yang dilakukan oleh Bansal & Kaur (2018) menerapkan metode *Extreme Gradient Boosting Based Tuning* (XGBoost) untuk melakukan deteksi serangan *Slow HTTP DoS* dan beberapa serangan *HTTP DoS* lainnya. Hasil penelitian menunjukkan tingkat akurasi deteksi sebesar 99,54% dan tingkat FPR sekitar 0,12%. Seperti halnya penelitian sebelumnya yang berbasis anomali, akan sulit diterapkan pada jaringan skala kecil dan menengah.

Metode deteksi *Anomaly-Based* memiliki kelebihan mampu mendeteksi jenis serangan yang baru, namun metode ini harus didukung dengan *dataset* yang besar agar menghasilkan tingkat akurasi yang tinggi, dampak dari *dataset* yang besar tersebut maka dibutuhkan pula mesin dengan kemampuan komputasi yang tinggi (Gangwar & Sahu, 2014). Lebih lanjut Gangwar & Sahu (2014) menjelaskan bahwa metode deteksi *Signature-Based* bisa memperbaiki masalah akurasi yang merupakan isu utama pada metode deteksi *Anomaly-Based*, namun kelemahan dari *Signature-Based* adalah tidak mampu mendeteksi jenis serangan di luar pola spesifik yang telah ditentukan sebelumnya berdasarkan trafik serangan yang telah diketahui.

Deteksi *Signature-Based* yang selama ini digunakan hanya fokus pada pola *HTTP Request* dari aplikasi serangan *HTTP DoS*, sehingga setiap jenis serangan harus dibuatkan pola deteksinya dan ini kurang efektif, karena untuk jenis serangan *HTTP DoS* yang belum dibuatkan pola deteksinya tentu saja akan gagal terdeteksi, maka pada penelitian ini kelemahan tersebut diperbaiki dengan menerapkan pola deteksi berdasarkan paket *HTTP Request* normal dari sejumlah *browser* yang digunakan oleh pengguna (W3Counter, 2017) lalu diubah ke dalam bentuk *Regular Expression* (Prithi dkk., 2017) yang disusun berdasarkan algoritma *Needleman Wunsch* (Kozik dkk., 2014) dan kemudian digunakan pada *Netfilter* (Satrya dkk., 2016) di atas sistem operasi *RouterOS* (Sagala & Pardosi, 2017) untuk proses deteksinya. Dengan menerapkan metode ini, diharapkan dapat melakukan deteksi terhadap serangan *Slow HTTP DoS* dengan lebih baik dan memungkinkan melakukan deteksi jenis serangan *HTTP DoS* yang lain, dengan tingkat akurasi yang lebih tinggi dan tingkat *False Positive* (FP) yang lebih rendah.

METODOLOGI PENELITIAN

Pendekatan penelitian yang digunakan adalah penelitian kuantitatif, hal ini karena masalah dalam penelitian ini telah jelas, yaitu adanya serangan *Slow HTTP DoS*. Jenis penelitian yang digunakan adalah penelitian *Quasi-Experimental*, dikarenakan sulit memperoleh kelompok kontrol akibat beberapa hal terkait perangkat yang digunakan dalam proses eksperimen, salah satunya adalah masalah spesifikasi. Bentuk *Quasi-Experimental* yang dipilih adalah desain *Time-Series*, oleh karenanya kelompok eksperimen akan diberikan tes awal (*Pre-Test*) sebanyak empat kali untuk menguji kestabilannya sebelum diberi perlakuan. Setelah diberi perlakuan maka dilakukan tes akhir (*Post-Test*) untuk kebutuhan evaluasi hasil uji sistem deteksi serangan *Slow HTTP DoS* (Sugiyono, 2015).

Penelitian dilakukan di Lab *Data Center* Pemerintah Kota Palopo dalam rentang waktu antara bulan Oktober 2018 hingga bulan Desember 2018. Sistem operasi yang digunakan untuk kebutuhan

pengumpulan data paket *HTTP Request browser* menggunakan sistem operasi Windows 10, Linux Ubuntu 18.04, Android Jelly Bean, Apple iOS iPad dan Apple macOS Macbook. Spesifikasi perangkat untuk kebutuhan pengujian deteksi IDS adalah RouterBoard RB2011UiAS, CPU 600MHz, memori 128MB, sedang sistem operasi yang digunakan adalah RouterOS versi 6.43.2.

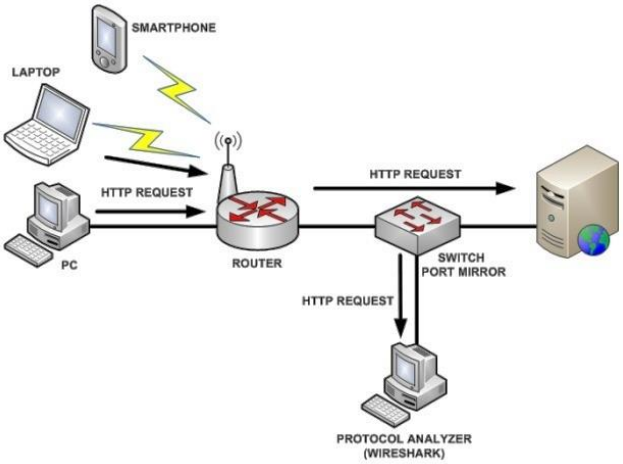
Berikut adalah tahapan penelitian yang dilakukan:

1. Studi Literatur

Referensi yang terkait erat dengan topik penelitian dikumpulkan dalam rangka identifikasi masalah serta mencari teori-teori yang mendukung pemecahan masalah penelitian.

2. Mengumpulkan dan Analisis Data *HTTP Request*

Data trafik HTTP yang dikumpulkan meliputi trafik *HTTP Request* dari beberapa *browser* dan trafik *HTTP Request* dari aplikasi *Slow HTTP DoS*. Gambar 1 adalah topologi jaringan yang digunakan untuk menangkap trafik *HTTP Request*.



Gambar 1 Topologi Menangkap Data *HTTP Request*

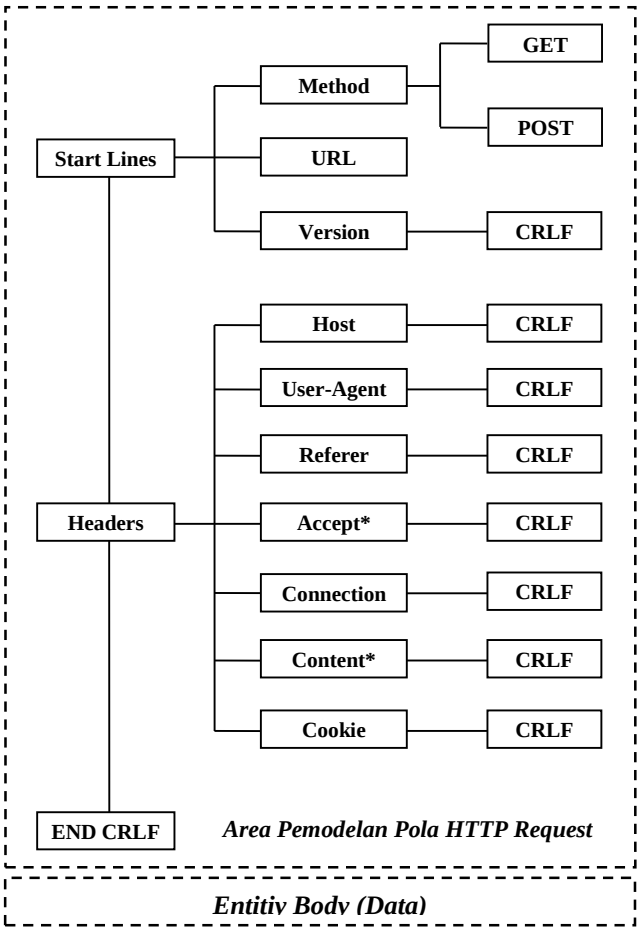
Setelah trafik *HTTP Request* dari *browser* dikumpulkan, selanjutnya dilakukan analisis pada setiap *HTTP Request*. Analisa trafik *HTTP Request* dibantu oleh aplikasi *Protocol Analyzer Wireshark* (Ndatinya dkk., 2015).

3. Membuat Model Pola Deteksi

Sebelum merumuskan pola *HTTP Request*, terlebih dahulu peneliti menentukan batasan area dari struktur protokol HTTP yang akan dimodelkan (Gourley & Totty, 2002), seperti yang terlihat pada gambar 2. Hal ini menjadi landasan dalam proses perumusan dan pembuatan pola deteksi serangan *Slow HTTP DoS*.

Sekumpulan *HTTP Request* dari beberapa *browser* kemudian dikelompokkan menjadi dua bagian, yaitu: (1) *HTTP Request GET* dan (2) *HTTP Request POST*, karena berdasarkan RFC2616 keduanya memiliki komposisi yang berbeda (Fielding dkk., 1999). Prioritas dalam pengambilan komponen dari *HTTP Request* pada pola juga akan diperhatikan.

a. Bagian *Start Lines* (*Method – URL – Version*) harus dimasukkan dalam pola karena merupakan bagian awal dalam setiap *HTTP Request*.
b. Bagian *Headers* terdapat dua kategori prioritas, komponen *Header* dengan isi yang selalu tetap pada setiap *HTTP Request* di setiap *browser,* akan mendapatkan prioritas lebih tinggi dibanding komponen *Header* dengan isi yang selalu berubah.
c. Bagian *End CRLF* juga harus ada pada pola karena bagian ini merupakan bagian akhir pada setiap *HTTP Request*.



Gambar 2 Area Pemodelan *HTTP Request*

4. Merumuskan Pola Deteksi

Setiap *HTTP Request GET* dan *HTTP Request POST* dikelompokkan dan disejajarkan untuk memperoleh pola kemiripan menggunakan algoritma Needleman Wunsch (Likic, 2008). Untuk memudahkan dalam proses menyejajarkan, maka setiap komponen *HTTP Request* dikodekan agar tampak lebih sederhana dan memudahkan dalam proses pencarian kemiripan pola. Hal ini terlihat pada Tabel 1 dan Tabel 2.

Tabel 1 Pengkodean *HTTP Request GET*

Field	Kode	Isi Tidak Berubah
Start Lines	*T*	*GET <> HTTP/1.1*
Host	S	
Connection	*U*	*keep-alive*
User-Agent	N	
Accept	A	
Accept-Encoding	M	
Accept-Language	I	

Tabel 2 Pengkodean *HTTP Request POST*

Field	Kode	Isi Tidak Berubah
Start Lines	*B*	*POST <> HTTP/1.1*
Host	I	
Connection	*G*	*keep-alive*
Content-Length	C	
User-Agent	O	
Content-Type	M	
Accept	P	
Referer	U	
Accept-Encoding	T	
Accept-Language	E	
Cookie	R	

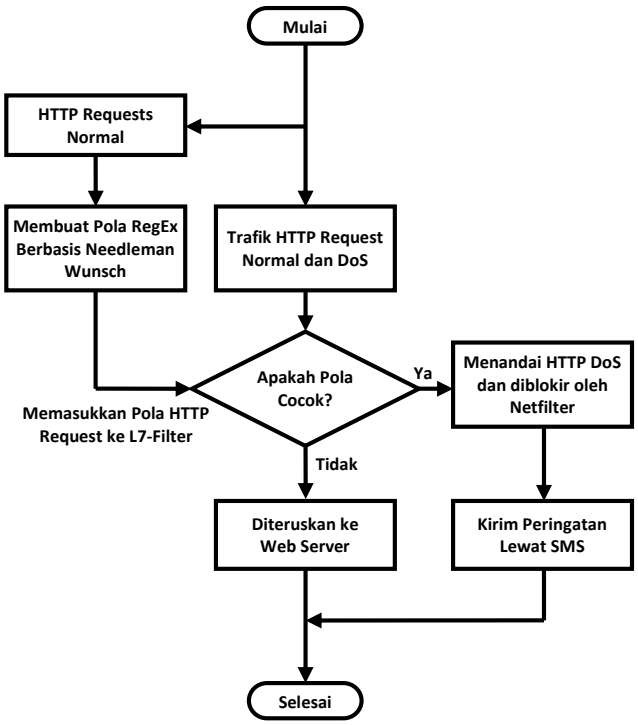
Tabel 3 Rumusan *Regular Expression*

Syntax	Arti	Contoh
^	Awal baris	^abc : abc ada di awal baris
|	OR (atau)	x|y : karakter x atau y
.	Mewakili 1 karakter apapun	a. : setelah a adalah 1 karakter apapun
\xHEX	Menyisipkan kode heksadesimal	\x20\x0d\x0a : kode heksadesimal menyatakan karakter "<spasi>\r\n"
*	Mengulangi sebanyak 0 atau lebih karakter sebelumnya	.* : mengulangi karakter apapun sebanyak 0 atau lebih
( )	Grup karakter tertentu	(abc) : grup karakter abc
[ ]	Mencocokkan karakter yang ada dalam kurung	[abc] : karakter a, b atau c
[^]	Selain karakter dalam kurung	[^abc] : selain karakter a, b atau c
$	Akhir baris	a$: karakter a berada di akhir baris

Setelah ditemukan pola kemiripan dari *HTTP Request* pada setiap *browser*, selanjutnya dilakukan konversi ke model *Regular Expression* menggunakan kode seperti pada Tabel 3 (Friedl, 1997; Goyvaerts, 2006).

5. Pengujian Model Deteksi

Gambar 3 menunjukkan sistem kerja IDS yang digunakan dalam penelitian ini. Setiap ***10 paket pertama*** atau ***2kB*** trafik HTTP akan dicek oleh *L7-Filter* untuk pencocokan pola (Gheorghe, 2006).



Gambar 3 Model Sistem Deteksi *Intrusi Netfilter*

6. Evaluasi Hasil Pengujian

Setelah pengujian sistem deteksi, selanjutnya dilakukan evaluasi hasil pengujian dengan melakukan penghitungan jumlah trafik *HTTP Request normal* dan *HTTP Request* dari serangan *Slow HTTP DoS* baik yang terdeteksi oleh sistem IDS maupun yang tidak terdeteksi dan kemudian dihitung tingkat *akurasi* dan tingkat *False Positive* (Zhu dkk., 2010).

Berikut adalah rumus umum yang digunakan.

$$Akurasi = \frac{(TN+TP)}{(TN+TP+FN+FP)} \; x \; 100\% \quad (1)$$

$$False\ Positive\ Rate = \frac{(FP)}{(FP+TN)} \; x \; 100\% \quad (2)$$

Untuk memperoleh nilai akurasi dalam penelitian IDS, maka paket data diklasifikasikan sebagai berikut:

a. *True Positive (TP)* : paket HTTP yang masuk kategori serangan dan berhasil dideteksi oleh IDS.

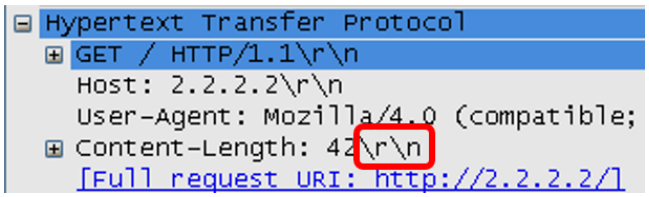
b. *True Negative (TN)* : paket HTTP normal dan tidak terdeteksi oleh IDS sebagai serangan.
c. *False Positive (FP)* : paket HTTP normal namun terdeteksi oleh IDS sebagai serangan.
d. *False Negative (FN)* : paket HTTP yang masuk kategori serangan namun tidak terdeteksi oleh IDS.

HASIL DAN PEMBAHASAN

Ada 3 (tiga) metode yang digunakan dalam serangan *Slow HTTP DoS* (Cambiaso dkk., 2013):

1. *Slow HTTP Header*

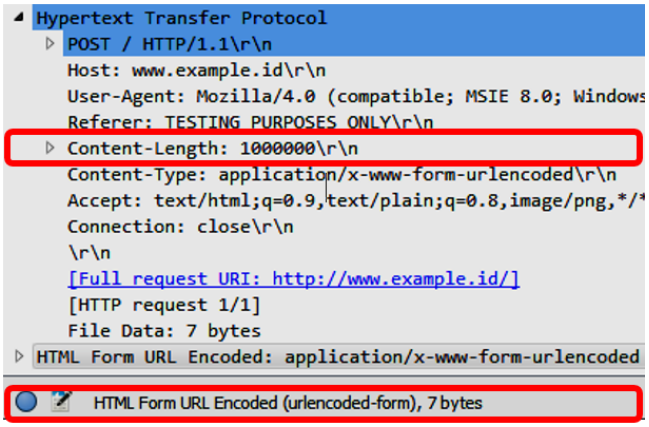
Setiap *HTTP Request GET* yang dikirimkan tidak memiliki *End CRLF* (\r\n) yang memisahkan antara *Header* dan *Entity Body* (Data). Hal ini mengakibatkan Web Server menganggap bahwa *HTTP Request* belum lengkap dan akan menunggu hingga batas waktu yang ditentukan, sehingga proses menjadi lambat.



Gambar 4 *HTTP Request Slow HTTP Header*

2. *Slow HTTP Body*

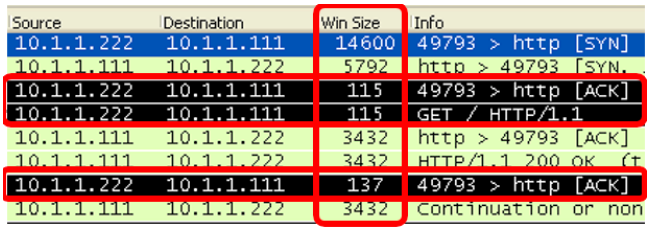
Serangan ini akan mengirimkan paket *HTTP POST* dalam jumlah banyak dengan memodifikasi *field Content-Length* dengan nilai yang sengaja dibuat berbeda dengan ukuran data yang dibawanya. Hal ini akan membuat *web server* menunggu kiriman data lengkap hingga tidak dapat melayani akses pengguna yang lain.



Gambar 5 *HTTP Request Slow HTTP Body*

3. *Slow HTTP Read*

Serangan ini mengirimkan trafik *HTTP Request* dan memodifikasi nilai ***Window Size*** dengan nilai yang rendah. Hal ini mengakibatkan *Web Server* dipaksa untuk mengurangi ukuran paket yang dikirimkannya sehingga proses pengiriman data menjadi lambat.



Gambar 6 *HTTP Request Slow HTTP Read*

Berikut adalah *HTTP Request GET* yang merupakan *HTTP Request* normal dari tiga *browser* berbeda. Bagian yang disajikan hanya pada komponen *HTTP Request* yang akan dimodelkan saja.

HTTP GET Browser Chrome (Android):

```
=======================================================
GET  /  HTTP/1.1\r\n
Host             : 192.168.43.230\r\n
Connection       : keep-alive\r\n
User-Agent       : Mozilla/5.0 (Linux; Android 4.4.4;
                   2014817 Build/KTU84P)
                   AppleWebKit/537.36 (KHTML, like
                   Gecko) Chrome/61.0.3163.98 Mobile
                   Safari/537.36\r\n
Accept           : text/html,application/xhtml+xml,
                   application/xml;q=0.9,
                   image/webp,image/apng,*/*;q=0.8\r\n
Accept-Encoding  : gzip, deflate\r\n
Accept-Language  : en-US,en;q=0.8,id;q=0.6
\r\n
=======================================================
```

HTTP GET Browser Firefox (Windows):

```
=======================================================
GET  /  HTTP/1.1\r\n
Host             : 10.1.1.1\r\n
User-Agent       : Mozilla/5.0 (Windows NT 6.1; Win64;
                   x64; rv:62.0) Gecko/20100101
                   Firefox/62.0\r\n
Accept           : text/html,application/xhtml+xml,
                   application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language  : en-US,en;q=0.5\r\n
Accept-Encoding  : gzip, deflate\r\n
Connection       : keep-alive\r\n
\r\n
=======================================================
```

HTTP GET Browser Safari (Mac-OS):

```
=======================================================
GET  /  HTTP/1.1\r\n
Host             : 192.168.1.212\r\n
Accept           : text/html,application/xhtml+xml,
                   application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language  : id\r\n
Connection       : keep-alive\r\n
Accept-Encoding  : gzip, deflate\r\n
User-Agent       : Mozilla/5.0 (Macintosh; Intel Mac OS
                   X 10_10_2) AppleWebKit/600.3.18
                   (KHTML, like Gecko) Version/8.0.3
                   Safari/600.3.18\r\n
\r\n
=======================================================
```

Berikut adalah hasil pencarian kemiripan pola menggunakan metode Needleman Wunsch.

```
Chrome  : T S - - - - U N A M I
Firefox : T S N A I M U - - - -
Safari  : T S A I - - U M N - -
Pola    : T S - - - - U - - - -
```

Setelah ditemukan komponen dari *HTTP Request* yang mirip, selanjutnya dibuatkan pola awal dalam bentuk *Regular Expression*.

```
^.*GET.*HTTP/1.1\x0d\x0aHost:.*\x0d\x0a
Connection:\x20keep.*\x0d\x0a\x0d\x0a$
```

Dari pola *Regular Expression* di atas, kemudian dibagi menjadi empat model pola dan **dinegasikan** (**[^]**) di beberapa bagian untuk deteksi serangan.

```
^.*GET.*HTTP/1.[^1].*$
```

Pola *Regular Expression* di atas untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* dengan *versi* ***HTTP/1.1***, selain itu akan dideteksi sebagai serangan.

```
^.*GET.*HTTP/1.1\x0d\x0a[^H][^o][^s][^t]
[^:].*$
```

Pola *Regular Expression* di atas untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang mengandung *header* "***Host:***".

```
(^.*GET.*HTTP/1.1.*\x0d\x0a[^C][^o][^n][^
n][^e][^c][^t][^i][^o][^n]:\x20keep.*$)|(
^.*GET.*HTTP/1.1.*\x0d\x0aConnection:\x20
[^k][^e][^e][^p].*$)
```

Pola *Regular Expression* di atas untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang mengandung *header* "***Connection:***" dengan isi "***keep-alive***".

```
^.*GET.*HTTP/1.1.*[^\x0d][^\x0a]\x0d\x0a$
```

Pola *Regular Expression* di atas untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang berakhir dengan ***End CRLF*** **(\r\n)**. Pola akhir *HTTP Request GET* dalam bentuk *Regular Expression* akan menjadi sebagai berikut:

```
(^.*GET.*HTTP/1.[^1].*$)|(^.*GET.*HTTP/1.
1\x0d\x0a[^H][^o][^s][^t][^:].*$)|(^.*GET
.*HTTP/1.1.*\x0d\x0a[^C][^o][^n][^n][^e][
^c][^t][^i][^o][^n]:\x20keep.*$)|(^.*GET.
*HTTP/1.1.*\x0d\x0aConnection:\x20[^k][^e
][^e][^p].*$)|(^.*GET.*HTTP/1.1.*[^\x0d][
^\x0a]\x0d\x0a$)
```

HTTP Request POST dari 3 jenis *browser*.

HTTP POST Browser Chrome (Linux):

```
=====================================================
POST /administrator/tambah_berita.php HTTP/1.1\r\n
Host            : www.example.id\r\n
Connection      : keep-alive\r\n
Content-Length  : 125\r\n
User-Agent      : Mozilla/5.0 (X11; Linux i686)
                  AppleWebKit/535.19 (KHTML, like
                  Gecko) Ubuntu/12.04
                  Chromium/18.0.1025.151
                  Chrome/18.0.1025.151
                  Safari/535.19\r\n
Content-Type    : application/x-www-form-
                  urlencoded]\r\n
Accept          : text/html,application/xhtml+xml,
                  application/xml;q=0.9,*/*;q=0.8\r\n
Referer         : http://www.example.id/administrator/
                  tambah_berita.php\r\n
Accept-Encoding : gzip,deflate,sdch\r\n
Accept-Language : en-US,en;q=0.8\r\n
Cookie          : PHPSESSID=7f944f27f2df6af10ad42be1
                  e8484a9b\r\n
\r\n
=====================================================
```

HTTP POST Browser Firefox (Windows):

```
=====================================================
POST /jsproxy  HTTP/1.1\r\n
Host            : 10.1.1.1\r\n
User-Agent      : Mozilla/5.0 (Windows NT 6.1;
                  rv:15.0) Gecko/20100101
                  Firefox/15.0.1\r\n
Accept          : text/html,application/xhtml+xml,
                  application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language : id,en-us;q=0.7,en;q=0.3\r\n
Accept-Encoding : gzip, deflate\r\n
Connection      : keep-alive\r\n
Referer         : http://10.1.1.1/webfig/\r\n
Content-Length  : 64\r\n
Content-Type    : text/plain; charset=UTF-8\r\n
Cookie          : username=admin\r\n
\r\n
=====================================================
```

HTTP POST Browser Safari (Mac-OS):

```
=====================================================
POST /injector/administrator/ HTTP/1.1\r\n
Host            : 192.168.1.212\r\n
Content-Type    : application/x-www-form-urlencode\r\n
Cookie          : PHPSESSID=4rgoltsjmaa7hd355gaqp
                  cugf\r\n
Content-Length  : 46\r\n
Connection      : keep-alive\r\n
Accept          : text/html,application/xhtml+xml,
                  application/xml;q=0.9,*/*;q=0.8
User-Agent      : Mozilla/5.0 (Macintosh; Intel Mac OS
                  X 10_10_2) AppleWebKit/600.3.18
                  (KHTML, like Gecko) Version/8.0.3
                  Safari/600.3.1) \r\n
Referer         : http://192.168.1.212/injector/
                  administrator/\r\n
Accept-Language : id\r\n
Accept-Encoding : gzip, deflate\r\n
\r\n
=====================================================
```

Model kemiripan pola *HTTP Request POST* dengan algoritma Needleman Wunsch:

```
Chrome : B I - - - - G C O M P U T E R
Firefox: B I O P E T G - - - - U C M R
Safari : B I M R C - G P O - - U E T -
Pola   : B I - - - - G - - - - U ! ! -
```

Pola awal *HTTP Request POST* dalam bentuk *Regular Expression*.

```
^.*POST.*HTTP/1.1\x0d\x0aHost:.*Connectio
n:\x20keep.*Referer:\x20http.*\x0d\x0a\x0
d\x0a$
```

Dari pola *Regular Expression* sebelumnya, kemudian dibagi menjadi lima model pola dan **dinegasikan** (**`[^ ]`**) di beberapa bagian untuk deteksi serangan *Slow HTTP DoS*.

```
^.*POST.*HTTP/1.[^1].*$
```

Pola *Regular Expression* di atas untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* dengan versi ***HTTP/1.1***, selain itu akan dideteksi sebagai serangan *Slow HTTP DoS*.

```
^.*POST.*HTTP/1.1\x0d\x0a[^H][^o][^s][^t]
[^:].*$
```

Untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang mengandung *header* "***Host:***".

```
(^.*POST.*HTTP/1.1.*\x0d\x0a[^C][^o][^n][
^n][^e][^c][^t][^i][^o][^n]:\x20keep.*$)|
(^.*POST.*HTTP/1.1.*\x0d\x0aConnection:\x
20[^k][^e][^e][^p].*$)
```

Untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang mengandung *header* "***Connection:***" dengan isi "***keep-alive***".

```
(^.*POST.*HTTP/1.1.*\x0d\x0a[^R][^e][^f][
^e][^r][^e][^r][^:]\x20.*$)|(^.*POST.*HTT
P/1.1.*\x0d\x0aReferer:\x20[^h][^t][^t][^
p].*$)
```

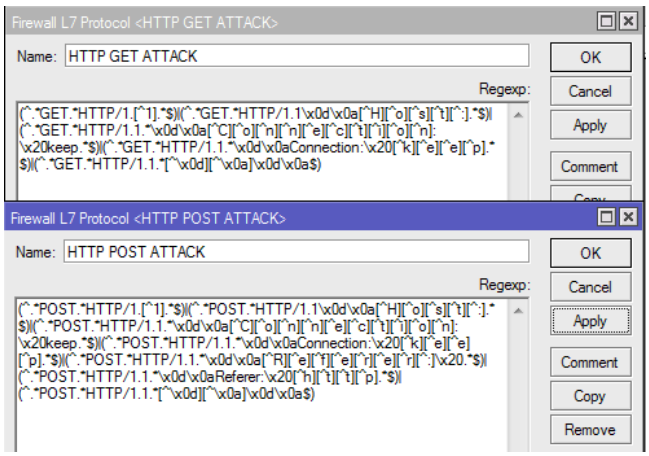
Untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang mengandung *header* "***Referer:***" dengan isi "***http***".

```
^.*GET.*HTTP/1.1.*[^\x0d][^\x0a]\x0d\x0a$
```

Untuk memastikan bahwa *HTTP Request* yang diizinkan adalah *HTTP Request* yang berakhir dengan ***End CRLF***. Model akhir *Regular Expression* ketika digabungkan dan digunakan pada *L7-Filter*.

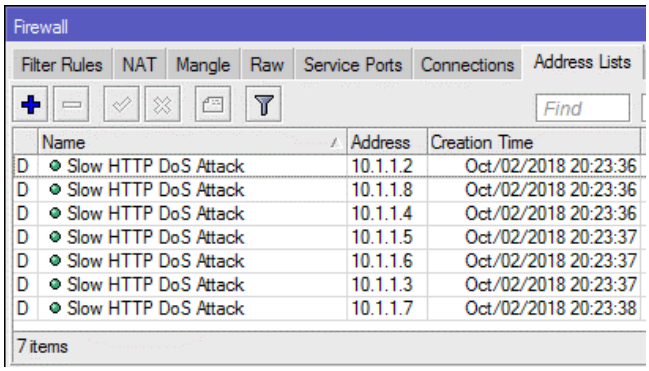
```
(^.*POST.*HTTP/1.[^1].*$)|(^.*POST.*HTTP/
1.1\x0d\x0a[^H][^o][^s][^t][^:].*$)|(^.*P
OST.*HTTP/1.1.*\x0d\x0a[^C][^o][^n][^n][^
e][^c][^t][^i][^o][^n]:\x20keep.*$)|(^.*P
OST.*HTTP/1.1.*\x0d\x0aConnection:\x20[^k
][^e][^e][^p].*$)|(^.*POST.*HTTP/1.1.*\x0
d\x0a[^R][^e][^f][^e][^r][^e][^r][^:]\x20
.*$)|(^.*POST.*HTTP/1.1.*\x0d\x0aReferer:
\x20[^h][^t][^t][^p].*$)|(^.*POST.*HTTP/1
.1.*[^\x0d][^\x0a]\x0d\x0a$)
```

Konfigurasi Pola *Regular Expression* di ***L7-Protocols*** (*L7-Filter*) tampak pada Gambar 7.

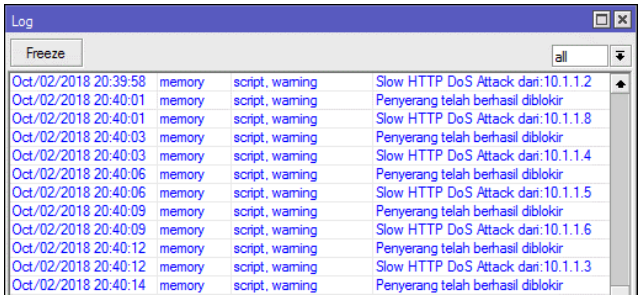


Gambar 7 Konfigurasi Pola di *L7-Filter*

Gambar 8 adalah hasil deteksi *Netfilter* terhadap serangan *Slow HTTP DoS*. Gambar 9 adalah informasi deteksi serangan yang disimpan di *Log sistem IDS*.

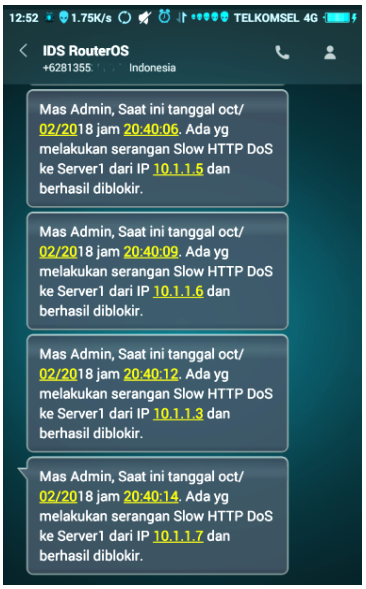


Gambar 8 Hasil Deteksi *L7-Filter* di *Netfilter*



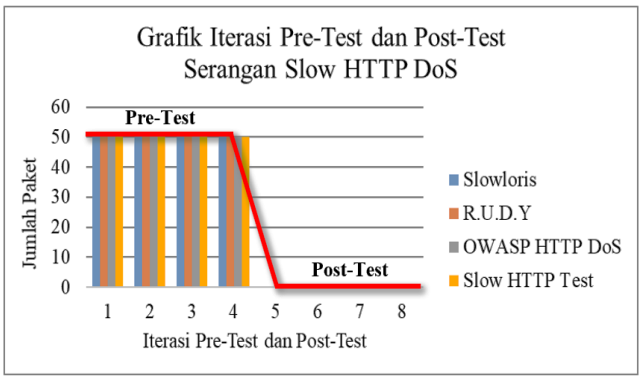
Gambar 9 Informasi Serangan di *Log IDS*

Informasi serangan *Slow HTTP DoS* juga dikirimkan ke Admin lewat SMS, seperti pada gambar 10.



Gambar 10 Laporan SMS Serangan *Slow HTTP DoS*

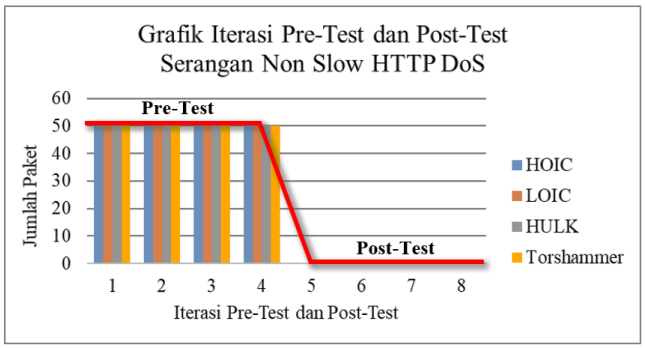
Gambar 11 adalah grafik iterasi *Pre-Test* sebanyak empat kali (sebelum diterapkan sistem deteksi) dan *Post-Test* sebanyak empat kali (setelah diterapkan sistem deteksi) dari serangan *Slow HTTP DoS*.



Gambar 11 Grafik Iterasi *Pre-Test* dan *Post-Test* Serangan *Slow HTTP DoS*

Gambar 12 adalah grafik iterasi *Pre-Test* sebanyak empat kali (sebelum diterapkan sistem deteksi) dan *Post-Test* sebanyak empat kali (setelah diterapkan sistem deteksi) dari serangan *Non Slow HTTP DoS*.

Didasarkan pada grafik iterasi *Pre-Test* dan *Post-Test* sebelumnya maka dapat diketahui bahwa sejumlah masing-masing 50 paket yang dikirimkan oleh setiap jenis serangan, dapat dengan baik terdeteksi oleh *IDS* dan hal ini juga dipertegas pada Tabel 4.



Gambar 12 Grafik Iterasi *Pre-Test* dan *Post-Test* Serangan *Non Slow HTTP DoS*

Dari jumlah 20 paket *HTTP Request* yang dikirimkan oleh masing-masing *browser* akan digunakan untuk kebutuhan pengujian dan pengukuran *False Positive Rate* (FPR), dalam hal ini baik *HTTP GET* maupun *HTTP POST*. Dari Tabel 5 dapat diketahui bahwa paket *HTTP Request* dari semua *browser* yang digunakan tidak terdeteksi sebagai serangan.

Dari jumlah total 550 paket dan semuanya terdeteksi sebagai serangan, baik dari aplikasi serangan *Slow HTTP DoS* maupun yang berasa dari aplikasi serangan *Non Slow HTTP DoS*, kemudian dikategorikan sebagai nilai *True Positive* (TP).

Lalu total 960 paket dan semuanya tidak terdeteksi sebagai serangan dari aplikasi *browser*, kemudian dikategorikan sebagai nilai *True Negative* (TN). Sehingga dapat dihitung nilai akurasi dan tingkat *False Positive* (FP) dengan rumus sebagai berikut:

$$Akurasi = \frac{(TP+TN)}{(TP+TN+FP+FN)} \; x \; 100\% \qquad (1)$$

$$Akurasi = \frac{(550+960)}{(550+960+0+0)} \; x \; 100\% = 100\%$$

$$False\ Positive\ Rate = \frac{(FP)}{(FP+TN)} \; x \; 100\% \qquad (2)$$

$$False\ Positive\ Rate = \frac{(0)}{(0+960)} \; x \; 100\% = 0\%$$

Dari hasil perhitungan di atas dapat diketahui bahwa tingkat akurasi dari metode deteksi yang digunakan adalah sebesar 100% dan tingkat *False Positif (FP)* yang diperoleh adalah sebesar 0%. Hal ini menunjukkan bahwa metode deteksi yang diterapkan menghasilkan tingkat akurasi yang lebih baik daripada penelitian sebelumnya, demikian dengan tingkat *False Positive (FP)* yang dihasilkan juga lebih rendah dibanding penelitian sebelumnya.

Tabel 4 Hasil Deteksi Aplikasi *Slow HTTP DoS* dan *Non Slow HTTP DoS*

Aplikasi HTTP DoS	Slow HTTP DoS			Nonlow HTTP DoS	Hasil Deteksi
	Slow Header	Slow Body	Slow Read		
Slowloris	+	-	-	-	✓
R.U.D.Y	-	+	-	-	✓
OWASP HTTP DoS	+	+	-	-	✓
Slow HTTP Test	+	+	+	-	✓
HOIC	-	-	-	+	✓
LOIC	-	-	-	+	✓
HULK	-	-	-	+	✓
Torshammer	-	-	-	+	✓

Tabel 5 Hasil Deteksi Aplikasi *Web Browser*

Sistem Operasi	Aplikasi Browser	Jenis HTTP Request		Hasil Deteksi
		HTTP GET	HTTP POST	
Windows 10	EDGE	+	+	✗
	Internet Explorer	+	+	✗
	Chrome	+	+	✗
	Firefox	+	+	✗
	Opera	+	+	✗
	UC Browser	+	+	✗
	KMeleon	+	+	✗
Linux Ubuntu 18.04	Chrome	+	+	✗
	Firefox	+	+	✗
	Opera	+	+	✗
Android Jelly Bean	Default	+	+	✗
	Brave	+	+	✗
	Chrome	+	+	✗
	Firefox	+	+	✗
	Opera Mini	+	+	✗
	UC Browser	+	+	✗
	Dolphin	+	+	✗
iOS iPad Apple	Safari	+	+	✗
	Chrome	+	+	✗
	Firefox Focus	+	+	✗
	Opera Mini	+	+	✗
macOS Macbook Apple	Safari	+	+	✗
	Chrome	+	+	✗
	Firefox	+	+	✗

KESIMPULAN

Dari hasil penelitian yang telah dilakukan dapat disimpulkan bahwa Serangan *Slow HTTP DoS* dapat dideteksi dengan metode *Signature-Based Detection* berbasis pola *Regular Expression* dengan tingkat akurasi sebesar 100% dan tingkat *False Positive (FP)* sebesar 0%. Serangan lain yang tidak termasuk dalam serangan *Slow HTTP DoS* dan dapat terdeteksi oleh IDS antara lain *HOIC, LOIC, HULK* dan *Torshammer.*

Jenis serangan *Slow HTTP DoS* dengan metode *Slow Header* dapat dideteksi dengan sempurna dan tidak memungkinkan penyerang melakukan manipulasi pada komponen *HTTP Request* agar dapat mengelabui IDS, karena pada dasarnya teknik ini menghilangkan komponen dasar dan wajib dari protokol HTTP, yaitu dengan tanpa akhiran *End CRLF* pada setiap *HTTP Request* yang dikirimkannya. Sedangkan jenis serangan *Slow HTTP DoS* dengan metode *Slow Body* dan *Slow Read* masih memungkinkan bagi penyerang untuk mengelabui IDS karena komponen yang dimanipulasi melibatkan komponen di luar area pemodelan pada *header HTTP Request*.

Deteksi untuk jenis serangan *Slow Body* masih dapat dikembangkan lebih lanjut, misalnya dengan cara membandingkan nilai pada *field Content-Length* dengan kapasitas data sebenarnya, jika berbeda maka bisa dipastikan hal tersebut adalah serangan *Slow Body*. Untuk jenis serangan *Slow Read* mungkin hanya dapat dideteksi dengan baik dengan metode berbasis anomali, karena kelemahan yang dimanfaatkan berada di luar area pemodelan *header HTTP Request*.

UCAPAN TERIMA KASIH

Terima kasih banyak kami ucapkan kepada para *reviewer*, Kepala Balitbang Kominfo, redaksi dan seluruh pihak yang telah membantu dalam proses penelitian ini.

DAFTAR PUSTAKA

Akbar, S., Endroyono, Wibawa, A. D. (2016). The impact analysis and mitigation of DDoS attack on local government electronic procurement service (LPSE). *Intelligent Technology and Its Applications (ISITIA), 2016 International Seminar on* (pp. 405-410).

Bansal, A., & Kaur, S. (2018). Extreme Gradient Boosting Based Tuning for Classification in Intrusion Detection Systems. *In International Conference on Advances in Computing and Data Sciences*, (pp. 372-380).

Cambiaso, E., Papaleo, G., Chiola, G., & Aiello, M. (2013). Slow DoS attacks: definition and categorization. *International Journal of Trust Management in Computing and Communications*, Vol.1(3-4), 300-319.

Cambiaso, E., Papaleo, G., Chiola, G., & Aiello, M. (2016). A Network Traffic Representation Model for Detecting Application Layer Attacks. *International Journal of Computing and Digital System 5*, No.1.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). *Hypertext transfer protocol--HTTP/1.1* (No. RFC 2616).

Friedl, J.E.F. (1997). *Mastering Regular Expressions: Powerful Techniques for Perl and Other Tools.* O'Reilly & Associates.

Gangwar, A., Sahu, S. (2014). A survey on anomaly and signature based intrusion detection system (IDS). *International Journal of Engineering Research and Applications* ISSN : 2248-9622, Vol. 4, Issue 4 (Version 1), April 2014, pp.67-72.

Gheorghe, L. (2006). *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter*. Packt Publishing Ltd.

Giralte, L. C., Conde, C., De Diego, I. M., & Cabello, E. (2013). Detecting denial of service by modelling web-server behaviour. *Computers & Electrical Engineering*, 39(7), 2252-2262.

Gourley, D., Totty, B. (2002). HTTP: *The Definitive Guide*. O'Reilly Media, Inc.

Goyvaerts, J. (2006). *Regular Expressions: The Complete Tutorial*. Lulu Press.

Gupta, S., Grover, D. (2016). Application Layer DDOS Attack :A Big Threat. *Streamed Info-Ocean*, Volume 1, Issue 1.

Idhammad, M., Afdel, K., & Belouch, M. (2018). Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest. *Security and Communication Networks.*

Kaspersky. (6 Februari 2018). DDoS Attack in Q4 2017. Diakses dari https://securelist.com/ddos-attacks-in-q4-2017/83729/ tanggal 25 Juli 2018.

Kozik, R., Choraś, M., Renk, R., & Hołubowicz, W. (2014). Modelling HTTP requests with regular expressions for detection of cyber attacks targeted at web applications. *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14* (pp. 527-535).

Kumar, G. (2016). Denial of service attacks–an updated perspective. *Systems Science & Control Engineering*, *4*(1), 285-294.

Likic, V. (2008). The Needleman-Wunsch algorithm for sequence alignment. *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, 1-46.

Ndatinya, V., Xiao, Z., Manepalli, V. R., Meng, K., & Xiao, Y. (2015). Network forensics analysis using Wireshark. *International Journal of Security and Networks*, *10*(2), 91-106.

Prithi, S., Sumathi, S., & Amuthavalli, C. (2017). A Survey on Intrusion Detection System using Deep Packet Inspection for Regular Expression Matching. *International Journal of Electronics, Electrical and*

Computational System (IJEECS), ISSN 2348-117X, Volume 6, Issue 1.

Sagala, A. & Pardosi, R. (2017). Improving SCADA Security using IDS and MikroTIK, *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* 9(1-4), pp.133-137.

Santoso, B. I., Idrus, M. R. S., & Gunawan, I. P. (2016). Designing Network Intrusion and Detection System using signature-based method for protecting OpenStack private cloud. *Engineering Seminar (InAES), International Annual* (pp. 61-66).

Sathwara, S., & Parekh, C. (2017). Distributed Denial of Service Attacks: TCP Syn Flooding Attack Mitigation. *International Journal of Advanced Research in Computer Science*, Vol. 8 Issue 5, p.2392-2396.

Satrya, G.B., Nugroho, F. E., Brotoharsono, T. (2016). Improving Network Security – A Comparison between nDPI and L7-Filter. *International Journal on ICT* Vol. 2, Issue. 2, pp. 11-26.

Stewart, J. M., Chapple, M., & Gibson, D. (2012). *CISSP: Certified Information Systems Security Professional Study Guide 6th Edition*. John Wiley & Sons.

W3Counter. (12 Desember 2017). Browser and Platfrom Market Share. Diakses dari https://www.w3counter.com/globalstats.php?year=2017&month=12 tanggal 25 Juli 2018.

Yevsieieva, O., & Helalat, S. M. (2017). Analysis of the impact of the slow HTTP DOS and DDOS attacks on the cloud environment. *Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), 2017 4th International* (pp. 519-523).

Zhu, W., Zeng, N., & Wang, N. (2010). Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland, 19*, 67.

Halaman ini sengaja dikosongkan