SISTEM DETEKSI INTRUSI MENGGUNAKAN N-GRAM DAN COSINE SIMILARITY

Baskoro A. Pratomo[1)] dan Royyana M. Ijtihadie[2)]
[1,2)]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo
e-mail: baskoro@if.its.ac.id[1)], roy@its.ac.id[2)]

ABSTRAK

Serangan atau intrusi yang masuk ke dalam sebuah sistem adalah sesuatu yang hampir pasti terjadi dalam dunia teknologi informasi saat ini. Untuk mengatasi hal tersebut ada beberapa teknologi yang dapat digunakan, seperti firewall atau sistem deteksi intrusi (intrusion detection system/IDS). Tidak seperti firewall yang hanya menyeleksi paket yang masuk berdasarkan alamat IP dan port, IDS bekerja dengan cara memantau isi paket yang masuk ke dalam sebuah komputer untuk kemudian memutuskan apakah rangkaian paket yang masuk tersebut merupakan sebuah serangan atau tidak. Salah satu aplikasi open sources dari IDS adalah Snort yang menggunakan pencocokan string untuk mengambil keputusan.

Kelemahan dari IDS yang berbasis pencocokan string adalah kemunculan string dalam sebuah paket harus sama persis, sehingga sulit untuk mendeteksi serangan yang mirip tetapi memiliki pola string yang berbeda. Oleh karena itu paper ini mengusulkan sebuah metode deteksi intrusi menggunakan n-gram dan cosine similarity untuk mencari kemiripan dari serangkaian paket, sehingga yang dicari bukan yang sama persis, tetapi seberapa mirip dengan signature yang ada. Berbeda dengan Snort, paket tidak dicocokkan dengan pola serangan, tetapi dengan pola pengaksesan sebuah halaman web oleh pengguna yang sesungguhnya, sehingga yang memiliki kemiripan tinggi akan dianggap sebagai paket yang sah, sedangkan yang rendah akan dianggap sebagai serangan.

Dari hasil ujicoba dengan berbagai variasi nilai ambang batas, maka didapatkan nilai 0.8 dengan n = 3 memberikan akurasi yang terbaik. Sistem deteksi intrusi ini juga mampu mendeteksi berbagai jenis serangan tanpa harus mendefinisikan serangan yang ada sebelumnya, sehingga lebih tahan terhadap zero-day attack.

***Kata Kunci**: anomaly IDS, cosine-similarity, n-gram, sistem deteksi intrusi*

ABSTRACT

Attack or intrusion into a system is something that is almost certainly happened in today's world of information technology. To overcome this, there are several technologies that can be used, such as firewalls or intrusion detection systems (IDS). Unlike firewalls that only inspect incoming packets based on IP address and port, IDS works by monitoring the payloads of the packet that come into a computer to then decide whether the incoming packet is malicious or not. An example of IDS application is Snort IDS, an open source application which uses string matching to detect malicious activity.

One weakness of string matching IDS is the occurrence of a string in a packet must be an exact match, just a slight difference can make an attack comes undetected, making it difficult to detect attacks that have similar flow but different pattern. Therefore, this paper proposed an intrusion detection method using n-gram and cosine similarity to seek similarity of a couple of packet sequences, thus the searching is conducted by looking for similarity between payload and existing signature. In contrast to Snort, those packets are not matched with the pattern of attacks, but rather the pattern of legitimate access to a web page done by legitimate users, so packets which have a high similarity are regarded as benign, while the low ones will be regarded as an attack.

From the test results with different alue of threshold, then we obtained the value of 0.8 with n = 3 gave the best accuracy. This intrusion detection system is also capable of detecting various types of attacks without having to define existing attacks in advance, making it more resistant to zero-day attacks.

***Keywords**: anomaly-based IDS, cosine-similarity, intrusion detection system, n-gram.*

I. PENDAHULUAN

SISTEM deteksi intrusi (IDS) adalah sebuah sistem yang digunakan untuk mendeteksi adanya serangan pada sebuah komputer atau server. Secara garis besar, ada dua jenis sistem deteksi intrusi berdasarkan bagaimana cara untuk mendeteksi serangan tersebut, yaitu berbasis aturan (rule-based/signature-based) dan berbasis anomali. IDS yang berbasis aturan akan menggunakan daftar aturan atau daftar pola isi paket yang dimilikinya untuk dicocokkan dengan rangkaian paket data yang melalui sebuah jaringan komputer. Metode ini cukup ampuh untuk serangan-serangan yang sudah dikenali sebelumnya dan metode ini sampai sekarang masih sering digunakan.

Kelemahan dari metode berbasis aturan ini adalah jika serangan yang masuk belum pernah ada sebelumnya atau daftar aturan yang digunakan sudah terlalu lama tidak berubah, sehingga tidak bisa mendeteksi jenis-jenis serangan terbaru. Oleh karena itu metode deteksi berbasis anomali digunakan. Metode ini memanfaatkan kebiasaan-kebiasaan yang terjadi dalam sebuah sistem dan menganggap apabila ada penyimpangan dari

kebiasaan yang terjadi, maka itu adalah sebuah aktivitas serangan. Sehingga jenis-jenis serangan yang belum pernah ada sebelumnya dapat lebih mudah ditangkap karena dianggap memiliki pola-pola yang berbeda dari yang umum diterima atau dikirim.

Namun metode ini cukup sulit diterapkan di dunia nyata karena akurasinya masih rendah, terutama false positive yang dihasilkan masih cukup tinggi. Selain itu proses deteksi yang biasanya membutuhkan pelatihan (training) memakan waktu yang tidak sebentar, sehingga sulit diterapkan pada dunia nyata. Oleh karena itu pada paper ini diusulkan sebuah metode hibrid yang berusaha memanfaatkan kelebihan dari masing-masing metode untuk meminimalisir kelemahan metode yang lain. Sistem deteksi intrusi yang dibangun memanfaatkan n-gram yang biasa digunakan pada penggalian teks (text mining) dan cosine similarity untuk mencari kemiripan antara paket data yang masuk dengan aturan yang ada. Selain itu aturan yang dibangun bukan berdasarkan paket-paket yang berbahaya, tetapi sebaliknya, dari paket-paket yang sah.

Bagian berikutnya dari paper ini tersusun sebagai berikut, bab 2 akan mendaftar metode-metode yang sudah ada yang berhubungan dengan IDS. Bab 3 menjelaskan tentang desain dari sistem deteksi intrusi yang diusulkan yang kemudian diujicoba dan hasilnya akan dibahas pada bab 4. Kemudian kesimpulan dari metode yang diusulkan serta hasil ujicoba akan masuk di bab 5.

II. STUDI LITERATUR

A. *Snort*

Snort[1] adalah sebuah sistem deteksi intrusi berbasis open source yang cukup populer. Snort termasuk dalam jenis IDS yang berbasis aturan, artinya Snort sangat tergantung dengan basis data aturan yang digunakan, semakin lengkap basis data tersebut, semakin besar juga kemungkinan Snort untuk mendeteksi suatu serangan. Dan karena Snort merupakan IDS yang berbasis aturan, maka dia membutuhkan algoritma pencocokan string, yaitu Aho-Corrasick[2]untuk mencari kemunculan suatu pola string dalam sebuah pesan. Contoh dari aturan yang digunakan oleh Snort dapat dilihat pada Gambar 1.

```
alert tcp any any -> 10.1.1.0/24 80 (content: "/cgi-bin/phf"; msg: "PHF
probe!";)
alert tcp any any -> 10.1.1.0/24 6000:6010 (msg: "X traffic";)
alert tcp !10.1.1.0/24 any -> 10.1.1.0/24 6000:6010 (msg: "X traffic";)
```

GAMBAR 1 CONTOH ATURAN-ATURAN PADA SNORT

Kelebihan dari Snort adalah dia termasuk dalam IDS yang sudah cukup matang dan memiliki dukungan komunitas yang kuat, sehingga banyak aturan-aturan yang dibuat oleh komunitas tersebut. Selain itu, sejak Sourcefire, perusahaan yang menaungi Snort, dibeli oleh Cisco, pengembangan aturan-aturan ini menjadi semakin besar. Namun karena Snort masih fokus di IDS berbasis aturan, maka dia belum bisa mendeteksi zero-day attack.

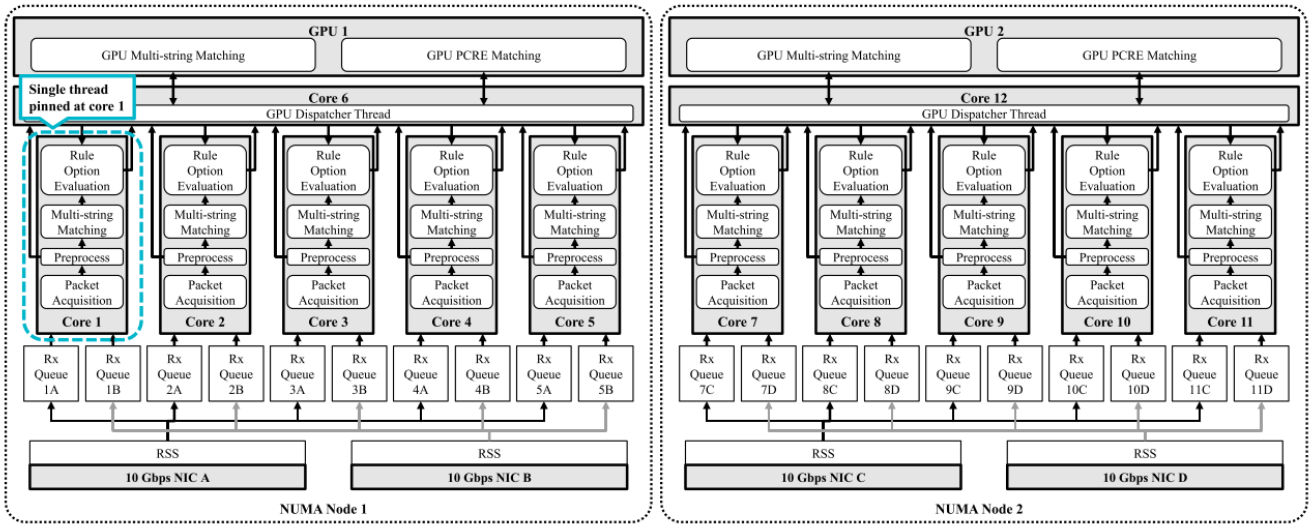
B. *Suricata*

Kekurangan yang lain dari Snort adalah desain aplikasinya yang masih menggunakan single thread. Sehingga performanya tidak akan bertambah signifikan meskipun jumlah core prosesor dalam sebuah komputer bertambah. Hal ini dapat mengakibatkan beberapa paket dilewatkan dari pemeriksaan ketika beban dari IDS terlalu besar, seperti yang diujicoba oleh [3].

Oleh karena itu [4] mengembangkan aplikasi deteksi intrusi yang memiliki kemampuan multithreading, sehingga mampu mengurangi jumlah paket yang dilewatkan dan menyebabkan false negatives. Suricata mampu memanfaatkan core prosesor yang ada seoptimal mungkin berkat fitur multithreading ini. Walaupun Suricata juga masih bergantung terhadap basis data aturan seperti Snort, bahkan Suricata dapat menggunakan aturan-aturan yang dimiliki oleh Snort karena memang format yang digunakan sama.

C. *Kargus*

Selain Suricata, Kargus[5] juga menerapkan multithreading pada deteksi intrusi. Namun selain memanfaatkan CPU untuk melakukan pencocokan string, Kargus juga memanfaatkan Graphical Processing Unit berbasis NVidia Cuda [6] untuk mempercepat. Tidak hanya itu, Kargus juga memperhatikan aspek-aspek lain yang mempengaruhi kecepatan pada pemrosesan paket, seperti pengambilan paket dari network interface dan passing parameter pada fungsi-fungsi secara batch untuk mengurangi overhead akibat terlalu banyaknya system call yang dipanggil. Arsitektur umum dari Kargus dapat dilihat pada Gambar 2.



GAMBAR 2 ARSITEKTUR KARGUS

Akan tetapi, karena Kargus dibangun berdasarkan kode sumber dari Snort versi 2.9.2.1, maka Kargus juga masih bergantung sekali dengan pola-pola string yang sudah didefinisikan sebelumnya. Sehingga belum mampu untuk menangani kejadian-kejadian yang sebetulnya merupakan variasi dari serangan yang sudah ada.

D. Snort AD

[7] mengembangkan sebuah plugin untuk Snort agar mampu melakukan deteksi anomali. Mereka memanfaatkan preprosesor yang ada di dalam Snort untuk membangun sebuah model time series dari aliran data yang keluar dan masuk di dalam jaringan yang selanjutnya akan digunakan untuk mendeteksi apakah ada anomali yang terjadi pada paket yang datang. Namun sayangnya proyek ini terhenti dan tidak ada perkembangan lebih lanjut selain pengembangan preprosesor pada Snort tsb.

E. PHAD

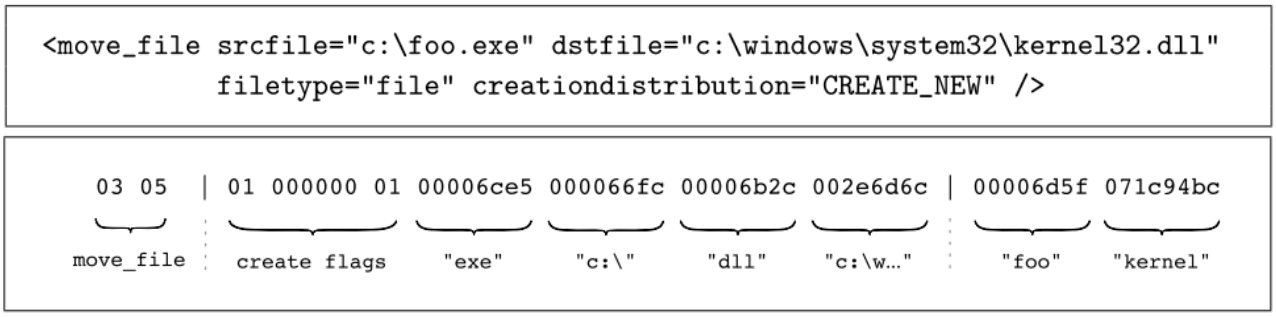
Salah satu pengembangan awal dari sebuah detektor anomali adalah PHAD (Packet Header Anomaly Detection) [8] yang bekerja dengan cara memeriksa fields dari header paket Ethernet, IP, TCP, UDP, dan ICMP dan mencari yang tidak wajar dari paket-paket tersebut. PHAD mencari kemunculan r nilai yang berbeda dari serangkaian n paket sejenis yang diperiksa, maka berarti ada r anomali yang terjadi. Dan jika kemunculannya melebihi ambang batas tertentu, maka paket tersebut akan dikategorikan sebagai serangan.

PHAD adalah salah satu dari IDS berbasis anomaly yang mampu bekerja secara real-time, sehingga benar-benar dapat digunakan untuk mendeteksi serangan pada paket yang keluar masuk di sistem pada saat itu juga. Namun PHAD hanya memeriksa informasi pada header paket, dia mengabaikan apa saja yang ada dalam sebuah payload. Sehingga jika tidak ada yang aneh di header paket, tetapi mengandung payload berbahaya seperti SQL Injection atau Buffer Overflow, maka PHAD tidak mampu mendeteksinya.

F. Malheur

Meskipun bukan untuk mendeteksi adanya malicious software (malware), namun Malheur[9] mengimplementasikan intelegensia mesin (machine learning) pada pengkategorian malware dengan memanfaatkan klastering dan klasifikasi untuk mengetahui apakah sebuah malware itu merupakan varian malware yang ada sebelumnya atau sebuah jenis malware baru.

Malheur memanfaatkan Q-Gram, sejenis N-Gram, untuk menunjukkan rangkaian system call yang dieksekusi oleh sebuah malware dengan asumsi bahwa malware-malware yang sejenis akan memiliki urutan eksekusi system call yang sama juga. Oleh karena itu urutan eksekusi system call tersebut dimodelkan dalam bentuk Q-Gram. Untuk menghemat ruang penyimpanan, system call tersebut disimpan dalam bentuk Malware Instruction SeT (MIST) yang terinspirasi dari kode instruksi dalam bahasa mesin. Contoh dari MIST ini dapat dilihat pada Gambar 3, dimana sebuah system call move_file diubah ke dalam bentuk integer MIST.



GAMBAR 3 KONVERSI SYSTEM CALL KE MIST PADA MALHEUR

G. N-Gram

N-Gram adalah serangkaian n-item yang berurutan dari sebuah data, biasanya berupa teks. Nilai N tersebut bisa bervariasi tergantung dari kebutuhan, mulai dari satu hingga sepanjang teks yang ada. Sebagai contoh, jika ada sebuah teks "INTRUSION", maka berbagai variasi N-Gram yang bisa dihasilkan dapat dilihat pada Tabel 1 berikut.

TABEL 1 CONTOH N-GRAM DARI KATA INTRUSION

Kata	N	N-Gram yang dihasilkan
INTRUSION	1	(I), (N), (T), (R), (U), (S), (I), (O), (N)
	2	(IN), (NT), (TR), (RU), (US), (SI), (IO), (ON)
	3	(INT), (NTR), (TRU), (RUS), (USI), (SIO), (ION)
	4	(INTR), (NTRU), (TRUS), (RUSI), (USIO), (SION)
	5	(INTRU), (NTRUS), (TRUSI), (RUSIO), (USION)

N-Gram dapat digunakan untuk berbagai masalah, salah satu yang paling sering digunakan adalah untuk prediksi kata yang akan digunakan, seperti pada fitur auto-correct pada keyboard telepon genggam berlayar sentuh atau pada pencarian kata kunci tertentu di dalam Google Search untuk membantu pengguna. Namun pada paper ini, N-Gram digunakan untuk mendeteksi kesamaan antara dua buah string.

GAMBAR 4 GAMBARAN UMUM PENGEMBANGAN SISTEM

III. METODOLOGI

Secara umum, ada empat tahapan yang dilakukan untuk membangun sebuah sistem deteksi intrusi yang memanfaatkan n-gram dan cosine similarity. Diawali dengan memodelkan paket-paket yang masuk ke dalam

sistem menjadi n-gram. Kemudian file tcpdump hasil dari pengaksesan ke sebuah website yang disediakan secara sah dibaca dan diekstrak n-gram-nya untuk menjadi dataset dari paket-paket yang tidak berbahaya.

Lalu masuk pada bagian inti dari sistem deteksi intrusi, yaitu ekstraksi paket dari antarmuka jaringan dan melakukan pra pemrosesan untuk membentuknya rangkaian paket yang sekuensial menjadi satu dan mendapatkan n-gram dari pesan yang terbentuk. Di akhir, n-gram dari paket-paket yang datang akan dibandingkan kemiripannya dengan n-gram dari tcpdump yang ada sebelumnya. Gambaran umum dari langkah-langkah ini ada pada Gambar 4, sedangkan penjelasan detilnya dibahas pada subbab-subbab berikut ini.

A. Pemodelan Paket dalam N-Gram

Sistem deteksi intrusi ini dibangun khusus untuk protokol HTTP di port 80, dimana distribusi karakter yang muncul lebih banyak ke karakter-karakter yang dapat dicetak (huruf, angka, tanda baca). Dalam metode yang diusulkan ini akan digunakan beberapa variasi n pada n-gram untuk nantinya dilihat efek dari perubahan tersebut.

p adalah paket yang diterima dan ditangkap oleh IDS, dimana p berisi karakter m.

$$p_i = m_{i_1} m_{i_2} m_{i_3} \cdots m_{i_y} \quad (1)$$

Dimana *y* bervariasi nilainya untuk setiap *i*

P adalah sebuah pesan yang dibuat dari rangkaian paket $p_1 \ldots p_X$ yang berurutan.

$$P = p_1 + p_2 + p_3 + \cdots + p_X \quad (2)$$

Sehingga P dapat juga dituliskan sebagai berikut :

$$P = m_{i_1} m_{i_2} m_{i_3} \cdots m_{i_y} m_{i+1_1} m_{i+1_2} m_{i+1_3} \cdots m_{i+1_y} \cdots\cdots m_{x_1} m_{x_2} m_{x_3} \cdots m_{x_y} \quad (3)$$

Kemudian untuk memudahkan, indeks m akan diubah seperti berikut :

$$m_{i_1} \rightarrow m_1, m_{i_2} \rightarrow m_2, m_{i_3} \rightarrow m_3, \cdots, m_{x_y} \rightarrow m_z \quad (4)$$

Sedangkan sekumpulan N-Gram N dengan panjang n akan dibentuk dari substring dari M. Seperti yang terlihat pada persamaan berikut :

$$N = \{g_1, g_2, \cdots g_k = (m_1 m_2 \cdots m_n), (m_2 m_3 \cdots m_{n+1}), \cdots, (m_{z-n} m_{z-n+1} \cdots m_z) \mid 1 \leq n \leq z, z > 0\} \quad (5)$$

$$N' = \{(g_1', frequency(g_1', N)), (g_2', frequency(g_2', N)), \cdots, (g_l', frequency(g_1^l, N)) \mid g_h' \epsilon\, distinct(N), h \geq 1\} \quad (6)$$

Dan karena yang dibutuhkan adalah distribusi kemunculan dari setiap gram, maka untuk setiap gram yang sama akan dijumlahkan nominalnya, sehingga *N'* berisi setiap elemen yang berbeda pada *N*, lengkap dengan kemunculannya pada *N*. Sebagai contoh, dapat dilihat pada Gambar 5 dimana ada beberapa *n* yang dapat digunakan untuk sebuah pesan permintaan HTTP dengan isi pesan *P* = 'GET HTTP/1.1\r\n' dan menghasilkan gram yang berbeda-beda juga.

n	N'
1	(G, 1), (E, 1), (T, 3), (spasi, 1), (H,1), (P,1), (/,1), (1, 2), (.,1), (\r,1), (\n,1)
2	(GE,1), (ET,1), (Tspasi,1), (spasiH,1), (HT,1), (TT,1), (TP,1), (P/,1), (/1,1), (1.,1), (.1,1), (1\r,1), (\r\n,1)

GAMBAR 5 HASIL PEMODELAN DARI SEBUAH STRING 'GET HTTP/1.1\R\N'

B. Pembuatan Benign Signatures

Berbeda dengan IDS berbasis aturan yang ada, seperti Snort, IDS yang diusulkan menggunakan basis data yang berisi daftar paket-paket yang sah dalam bentuk n-gram. Dengan asumsi bahwa format string yang keluar masuk dari sebuah web server cenderung tidak banyak berubah dan memiliki distribusi huruf yang sama, maka paket-paket yang berbahaya akan memiliki kemiripan distribusi yang cukup jauh dibanding dengan yang sah. Sehingga

serangan-serangan jenis baru pun juga tetap memiliki distribusi gram yang berbeda, meskipun menggunakan beberapa teknik penghindaran yang ada.

Untuk keperluan ujicoba nanti, disediakan sebuah website yang baru ter-install dalam sebuah server. Website ini akan diakses oleh mahasiswa dengan satu ketentuan, tidak boleh melakukan aktivitas serangan pada website ini. Semua aktivitas pada website akan direkam dengan menggunakan tcpdump dan hasilnya disimpan dalam file pcap.

Kemudian, file pcap tersebut akan dibaca oleh sebuah aplikasi untuk membuat Benign Signatures (BS's), yaitu n-gram dari paket-paket yang sah, yang sudah direkonstruksi menjadi sebuah pesan. Dan untuk setiap pesan yang terbuat, akan menjadi sebuah signature tersendiri. Sebagai ilustrasi, salah satu signature yang berhasil dibuat dengan n = 2 dapat dilihat pada Gambar 6. Hasil dari aktivitas ini akan disimpan dalam sebuah file dengan ekstensi .rules.

```
3233:1;6962:1;796e:1;6874:2;6572:1;6573:1;6570:3;6574:1;6578:4;206c:2;772d:1;656e:3;6d2f:2;722d
:1;0a0d:1;3932:1;322e:5;3136:1;696e:2;3132:1;3139:1;0a75:1;706c:1;7561:1;3031:1;2068:1;7032:1;2
062:1;2063:1;2065:1;2067:2;702c:1;702f:1;312e:3;7874:4;2f20:1;0a68:1;0a61:3;0d0a:7;742d:2;2a3b:1
;2d6c:1;7365:1;7367:1;627a:1;6c2d:1;6c2c:2;2073:1;677a:1;2074:4;2e38:2;2f2a:1;2e32:2;2e30:2;2e31
:5;713d:1;2e34:2;382e:2;2f31:2;202a:1;743a:3;6277:1;7373:3;6775:1;6e2c:1;7374:1;7420:1;3d30:1;30
2e:1;6373:1;3120:1;666d:1;652e:1;330d:1;6f6d:1;6169:1;6163:3;6167:2;202f:1;3638:1;7265:2;6765:3;
320d:2;3134:1;6363:3;2f32:3;6365:3;300d:1;653a:1;3870:1;6d6c:2;616e:1;6e74:1;6e75:1;6e78:1;6d70
:1;6f64:1;2a2f:1;636f:2;2c20:6;676e:1;676d:1;673a:1;2f63:1;6f73:1;686f:1;782f:1;2f68:1;2d65:1;2d66
:1;7a69:2;2d61:1;6e67:2;6e63:1;6c79:1;7573:1;7574:1;7074:3;6c73:1;7072:2;746d:1;7777:2;2031:1;7
470:1;7474:1;342e:1;2d6d:1;2f70:1;2f73:1;6970:2;742f:4;6d6d:1;6e0d:1;3a20:5;6469:1;6c69:1;736c:1
;3b71:1;6c61:2;3420:2;746c:1;732f:1;732c:2;310d:1;7465:4;
```

GAMBAR 6 CONTOH BARIS DALAM BENIGN SIGNATURE

Untuk penyimpanan gram, yang harus diperhatikan adalah keberadaan karakter-karakter tertentu yang bisa mengacaukan pola. Sebagai contoh, gram untuk setiap pesan ditulis dalam satu baris dan antara gram satu dengan yang lain akan dipisahkan dengan titik koma (;). Sedangkan pemisah antara gram dengan kuantitasnya adalah titik dua (:) dan karakter-karakter tersebut pasti muncul dalam sebuah pesan HTTP. Untuk mengatasi hal tersebut, maka karakter-karakter dalam gram akan disimpan dalam bentuk heksadesimal, meskipun berarti mengorbankan ruang tambahan di harddisk untuk menyimpan BS.

C. Penangkapan Paket dan Pra Pemrosesan

Poin penting yang ingin dicapai di sini adalah bagaimana agar sistem ini dapat berjalan secara real-time. Karena dari beberapa metode sistem deteksi intrusi yang ada [10], [11], [12] tidak dapat melakukannya karena proses perhitungan yang terlalu lama dan membutuhkan proses pelatihan yang berulang-ulang. Oleh karena itu, penggunaan n-gram di sini diharapkan dapat mengatasi masalah tersebut.

Langkah pertama tentu adalah membaca BS yang ada di dalam file dan menyimpannya dalam memori untuk dicocokkan nantinya dengan pesan-pesan yang datang. Tidak lupa karakter gram yang tersimpan dalam file dikembalikan ke bentuk ASCII-nya.

Langkah kedua yang perlu dilakukan adalah melakukan penangkapan paket dan menyusunnya kembali. Ada dua masalah yang bisa timbul pada langkah ini, yang pertama adalah rangkaian paket dari sebuah alamat IP asal bisa saja tidak sampai pada waktu yang berurutan. Masalah kedua adalah bisa saja penyerang mengirimkan sebuah paket yang sangat panjang dalam beberapa potongan segmen TCP dengan tujuan membuat sibuk IDS dengan menunggu paket sebelum memrosesnya.

Untuk mengatasi masalah tersebut, penangkapan paket pada IDS yang diusulkan memanfaatkan teknik multithreading. Setiap thread akan menangani request dari sebuah 4-tuple alamat IP asal, alamat IP tujuan, port asal, dan port tujuan. Setiap paket yang berasal dari tuple tersebut akan ditangani oleh sebuah thread yang merekonstruksi paket tersebut. Selain itu setiap thread juga diberi batasan waktu, jika dalam batas waktu tertentu paket dari suatu tuple masih terus berdatangan, maka paket yang datang setelah batas waktu akan dihiraukan dan apapun pesan yang ada akan diproses.

D. Deteksi Kemiripan

N-gram yang tersimpan dalam BS's dan yang didapat dari paket yang masuk dapat direpresentasikan juga dalam bentuk vektor n-dimensi dimana karakter gram-nya adalah arah dan kuantitas dari gram adalah besarannya. Sehingga untuk mencari kemiripan dari dua buah vektor n-dimensi adalah menggunakan cosine-similarity, seperti yang tertera pada persamaan :

$$similarity_i = \frac{I \,.\, R_i}{\|I\|\|R_i\|} = \frac{\sum_{j=1}^{k} I_j \times R_{i_j}}{\sqrt{\sum_{j=1}^{k} I_j} \times \sqrt{\sum_{j=1}^{k} R_{i_j}}} \quad (7)$$

Dimana :
I : n-gram dari paket yang datang (Incoming)
R_i : n-gram dari Benign Signatures ke-i

I adalah koleksi N-gram dari paket yang masuk dan R_i adalah koleksi N-gram dari BS. Dan setiap elemen I akan dikalikan jumlahnya dengan jumlah dari setiap elemen di R_i yang sama. Untuk kemudian dibagi dengan penjumlahan besaran dari setiap elemen di I dengan besaran dari setiap elemen di R_i.

Salah satu keuntungan menggunakan cosine-similarity ini adalah kecepatan perhitungannya, karena pada dasarnya kita tidak akan menghitung sebanyak elemen dari I ataupun R_i. Hal ini disebabkan oleh apabila ada gram di I yang tidak ada di R_i, maka hasil perkaliannya pasti akan 0, sehingga yang perlu dihitung hanya setiap gram yang ada baik di I maupun R_i. Dan dari implementasi pada aplikasi deteksi intrusi, metode pencocokan ini hanya membutuhkan kompleksitas $O(kmn)$, dimana k adalah jumlah baris pada file signature, m adalah banyaknya variasi gram pada R_i, dan n adalah banyaknya variasi gram pada I.

Seperti yang terlihat pada potongan kode dalam Python di Gambar 7, setelah proses pencocokan dilakukan, maka akan didapatkan beberapa nilai *similarity* dan apabila dari semua *similarity* tersebut tidak ada yang lebih besar dari nilai ambang batas yang ditentukan, maka paket tersebut akan dianggap sebagai sebuah serangan dan sebuah peringatan akan dikeluarkan. Yang perlu diingat adalah semakin besar nilai hasil cosine similarity, maka paket yang masuk akan semakin mirip dengan paket-paket yang sah.

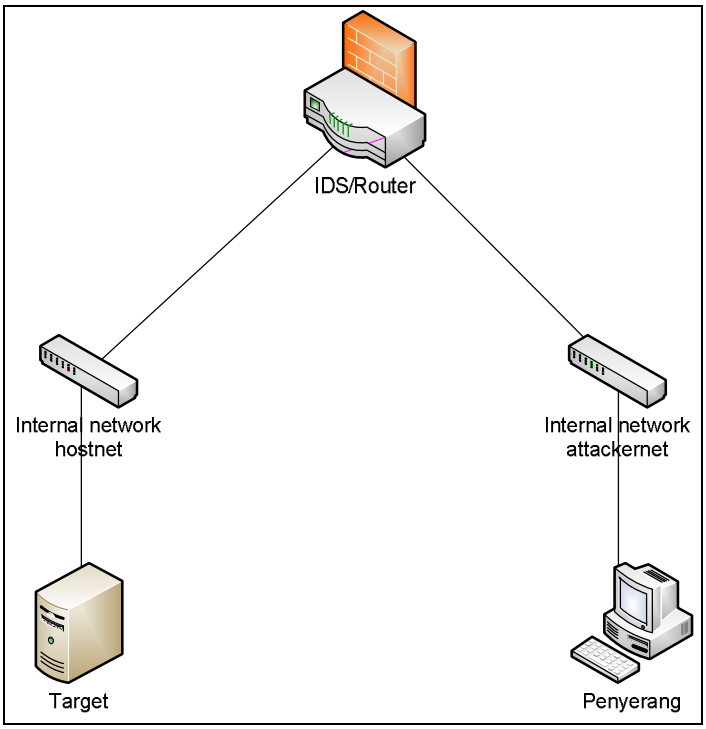
```
def similarities(self, incoming_gram, algorithm='cosine'):
    sims = []
    for key, rule in self.list_rules.iteritems():
      if algorithm == 'cosine':
        sims.append(self.cosine_sims(incoming_gram, rule))

    return sims

def cosine_sims(self, incoming_gram, gram_rule):
    up_sum = 0
    down_sum_a = 0
    down_sum_b = 0

    for key, a in incoming_gram.iteritems():
      b = 0 if gram_rule.rule.get(key) is None else gram_rule.rule.get(key)
      up_sum += a*b
      down_sum_a += math.pow(a,2)

    for key, b in gram_rule.rule.iteritems():
      down_sum_b += math.pow(b,2)

    return up_sum / (math.sqrt(down_sum_a) * math.sqrt(down_sum_b))
```

GAMBAR 7 POTONGAN KODE MENCARI KEMIRIPAN ANTARA PAKET YANG DATANG DENGAN BENIGN SIGNATURES

IV. UJI COBA

Untuk mengetahui efektivitas dari sistem deteksi intrusi yang diusulkan maka perlu dilakukan uji coba. Uji coba ini dilakukan dengan memanfaatkan Oracle VM VirtualBox untuk membuat mesin virtual yang tersusun

seperti Gambar 8. Ada tiga mesin virtual yang dibutuhkan, masing-masing digunakan sebagai target, penyerang, dan router sekaligus IDS. Mesin penyerang akan mengirimkan beberapa variasi serangan pada aplikasi web yang terletak di mesin korban. Kemudian akan dihitung berapa serangan yang mampu terdeteksi oleh sistem dengan nilai ambang batas tertentu.

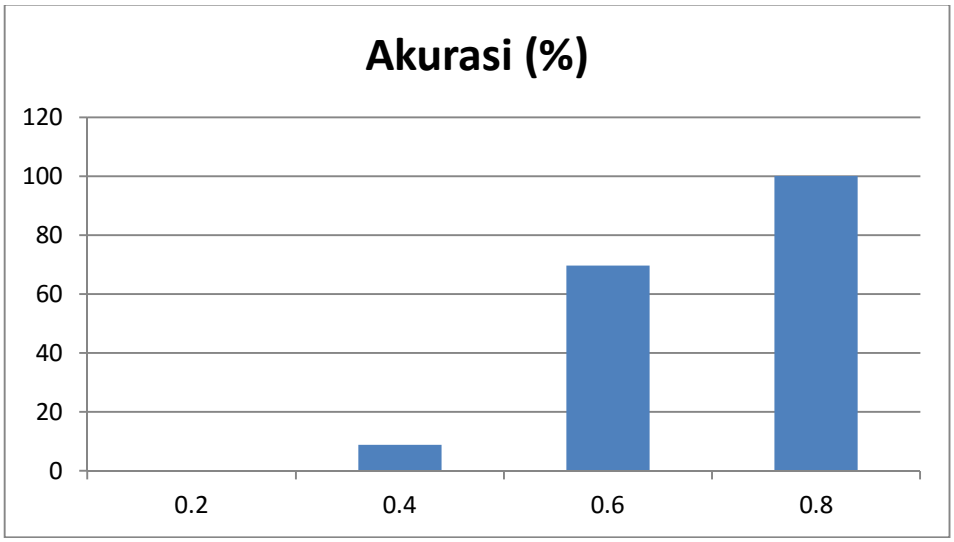


GAMBAR 8 TOPOLOGI UJI COBA

Untuk ujicoba penyerangan, digunakan aplikasi SQLMap untuk membuat serangan ke komputer target. Sedangkan nilai n yang digunakan adalah 3, dengan nilai ambang batas bervariasi dari 0,2 sampai 0,8. Selain itu diketahui juga ada 474 serangan yang dilakukan oleh SQLMap dalam satu sesi. Rekapitulasi hasil dari uji coba ini dapat dilihat pada Tabel 2 dan Gambar 9. Dari sini dapat diambil kesimpulan bahwa nilai ambang batas yang paling cocok adalah 0.8.

TABEL 2 HASIL UJICOBA SERANGAN

Nilai ambang batas	Jumlah serangan terdeteksi
0.2	0
0.4	42
0.6	330
0.8	474



GAMBAR 9 DIAGRAM PENGARUH NILAI AMBANG BATAS TERHADAP AKURASI

V. KESIMPULAN

Penggunaan n-gram dan pencarian kemiripan menggunakan cosine similarity terbukti dapat digunakan pada sistem deteksi intrusi dan IDS yang diimplementasikan dapat berjalan secara real-time. Sehingga pemeriksaan paket dapat berjalan tanpa harus menunggu waktu pelatihan yang terlalu lama. Selain itu dengan penggunaan Benign Signatures, dibandingkan dengan Malicious Signatures, dapat membuat IDS mampu mendeteksi jenis-jenis serangan yang belum pernah terdeteksi sebelumnya.

Untuk ke depan, pra pemrosesan dari IDS ini dapat diperbaiki lagi agar bisa mencakup lebih banyak protokol. Selain itu, penyimpanan Benign Signatures juga perlu diperbaiki agar tidak memakan terlalu banyak ruang dan Signature yang mirip, sama, atau sekelompok bisa digabungkan saja. Penggunaan metode lain untuk deteksi kemiripan juga dapat dicoba untuk mendapatkan hasil yang terbaik.

DAFTAR PUSTAKA

[1] Roesch, M. (1999, November). Snort: Lightweight Intrusion Detection for Networks. In LISA (Vol. 99, No. 1, pp. 229-238).
[2] Aho, A. V., & Corasick, M. J. (1975). Efficient string matching: an aid to bibliographic search. Communications of the ACM, 18(6), 333-340.
[3] Day, D. J., & Burns, B. M. (2011, February). A performance analysis of snort and suricata network intrusion detection and prevention engines. In The Fifth International Conference on Digital Society.
[4] Suricata, I. D. S. (2014). open-source IDS/IPS/NSM engine (http://suricata-ids.org/).
[5] Jamshed, M. A., Lee, J., Moon, S., Yun, I., Kim, D., Lee, S., ... & Park, K. (2012, October). Kargus: a highly-scalable software-based intrusion detection system. In Proceedings of the 2012 ACM conference on Computer and communications security (pp. 317-328). ACM.
[6] Nvidia, C. U. D. A. (2008). Cublas library. NVIDIA Corporation, Santa Clara, California, 15.
[7] Szmit, M., Wężyk, R., Skowroński, M., & Szmit, A. (2007). Traffic Anomaly Detection with Snort. Information Systems Architecture and Technology. Information Systems and Computer Communication Networks, Wydawnictwo Politechniki Wrocławskiej, Wrocław, 181-187.
[8] Mahoney, M. V., & Chan, P. K. (2001). PHAD: Packet header anomaly detection for identifying hostile network traffic.
[9] Rieck, K., Trinius, P., Willems, C., & Holz, T. (2009). Automatic analysis of malware behavior using machine learning. TU, Professoren der Fak. IV.
[10] Lee, W., Stolfo, S. J., & Mok, K. W. (1999). A data mining framework for building intrusion detection models. In Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on (pp. 120-132). IEEE.
[11] Lee, W., & Stolfo, S. J. (1998, January). Data mining approaches for intrusion detection. In Usenix Security.
[12] Portnoy, L., Eskin, E., & Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001.