TUGAS AKHIR – KI141502

PREDIKSI SERANGAN PADA JARINGAN KOMPUTER SECARA *REAL-TIME* MENGGUNAKAN METODE *HIDDEN MARKOV MODEL* (HMM)

KHARISMA NUR ANNISA
NRP 5112100026

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KI141502

PREDIKSI SERANGAN PADA JARINGAN KOMPUTER SURABAYA SECARA *REAL-TIME* MENGGUNAKAN METODE *HIDDEN MARKOV MODEL* (HMM)

KHARISMA NUR ANNISA
NRP 5112100026

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

REAL-TIME PREDICTION OF INTRUSION AT COMPUTER NETWORKS USING HIDDEN MARKOV MODEL METHODE (HMM)

KHARISMA NUR ANNISA
NRP 5113100026

Supervisor I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Supervisor II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Prediksi Serangan pada Jaringan Komputer Secara *Real-Time* Menggunakan Metode *Hidden Markov Model* (HMM)

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Arsitektur dan Jaringan Komputer
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
KHARISMA NUR ANNISA
NRP : 5113 100 026

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Henning T C,S.Kom.,M.Kom. .....
   NIP: 19840708 201012 2 004 (Pembimbing 1)

2. Royyana M I, S.Kom., M.Kom., PhD. .....
   NIP: 19770824 200304 1 001 (Pembimbing 2)

SURABAYA
JULI, 2017

[Halaman ini sengaja dikosongkan]

PREDIKSI SERANGAN PADA JARINGAN KOMPUTER SECARA *REAL-TIME* MENGGUNAKAN METODE *HIDDEN MARKOV MODEL* (HMM)

Nama Mahasiswa	**: Kharisma Nur Annisa**
NRP	**: 5113100026**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.**

Abstrak

Penyerangan yang akan dilakukan oleh *intruder* biasanya tidak terduga oleh sistem administrator. Namun kondisi suatu jaringan komputer dapat diprediksi dengan mengamati perubahan urutan kejadian penyerangan. *SQL Injection* adalah jenis serangan yang akan diprediksi dan dilakukan uji coba untuk Tugas Akhir ini.

Pada tugas akhir ini, diterapkan metode pemodelan statistika yaitu *Hidden Markov Model* (HMM) untuk memprediksi serangan yang mungkin terjadi pada jaringan komputer. Percobaan prediksi dilakukan pada sistem jaringan Institut Teknologi Sepuluh Nopember. Pemakaian metode HMM pada Tugas Akhir ini dikarenakan HMM memiliki karakteristik berupa perubahan pada *state* internal yang berdasarkan waktu (Markov Model) akan tetapi tidak nampak (*hidden*) dari luar sistem sehingga tidak dapat langsung dilakukan pengamatan. Namun karena jumlah *state* internal yang terbatas dan *state* saat ini (*current state*) bergantung pada *state* sebelumnya, sehingga dapat melakukan pengamatan pada suatu yang berhubungan (misalnya variabel yang berkorelasi) dengan *state* sebelumnya.

Dari hasil penerapan metode *Hidden Markov Model* untuk memprediksi kondisi suatu server atau jaringan komputer

dari serangan *SQL Injection* dihasilkan rata-rata akurasi 25.54%. Sedangkan untuk hasil prediksi serangan DDoS dari data DARPA 2000 rata-rata akurasi adalah 49.04%. Oleh karena itu, penerapan metode *Hidden Markov Model* dirasa lebih cocok untuk prediksi serangan DDoS dari pada serangan *SQL Injection* pada suatu server atau jaringan komputer.

Kata kunci: HMM, Intrusi, Deteksi, Prediksi, SQL Injection.

REAL-TIME PREDICTION OF INTRUSION AT COMPUTER NETWORKS USING HIDDEN MARKOV MODEL METHODE (HMM)

Nama Mahasiswa	**: Kharisma Nur Annisa**
NRP	**: 5113100026**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.**

Abstract

Attacks to be committed by intruders are usually unexpected by the system administrator. However, the condition of a computer network can be predicted from the sequence of attacks. SQL Injection is a type of attack that will be predicted and tested for this Final Project.

In this final project, Hidden Markov Model (HMM) is used to predict the attacks that occur on computer network. The system prediction is tested on the network system of Institut Teknologi Sepuluh Nopember. HMM has the characteristic of a change in the internal state based on time (Markov Model) but it is not visible (hidden) from the outside of the system so that it can not be directly observed. However, due to the limited number of internal states and the current state depends on the previous state, so as to observe a corresponding (eg correlated variable) with the previous state.

From the implementation of Hidden Markov Model method, it results 25,54% as an average of accuaration prediction for SQLInjection attack. And for DDoS attack it results average of accuration predisction is 49.04%. DDoS attacks are the result of intrusion detection form DARPA 2000. Therefore, implementattion of Hidden Markov Model method

is more suitable for DDoS attack prediction than SQL Injection attack on a server or computer network.

Keywords: HMM, Intrusion, Detection, Preditcion, SQLInjection.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"PREDIKSI SERANGAN PADA JARINGAN KOMPUTER SECARA *REAL-TIME* MENGGUNAKAN METODE *HIDDEN MARKOV MODEL* (HMM)".** Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Abah Cholik, Abah Yak, Ibu Istiqomah, Mas Aris, Mas Risqo, Mbak Rista, dan keluarga di Tulungagung untuk segala do'a dan semangat yang diberikan.
3. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD**.** selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS.

6. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
7. Nela, Nida, Sari, Tities, Okta, Demsy, dkk atas segala support sebagai teman seperjuangan TA.
8. Tante Dian dan keluarga yang selalu memberikan semangat serta do'a dalam menyelesaikan Tugas Akhir dan membantu lancarnya administrasi kelulusan.
9. Teman-teman administrator laboratorium Arsitektur dan Jaringan Komputer(AJK) Mas Sam, Mas Uyung, Mas Romen, Mas Pur, Mas Surya, Mas Thiar, Mas Agus, Mbak Nisa, Mbak Eva, Daniel, Wicak, Uul, Setiyo, Nindy, Zaza, Fatih, Oing, Syukron, Afif, Thoni, Vivi, Bebet, Fuad, Didin, Awan, dan Satria .
10. Teman-teman TC angkatan 2013 yang yang telah berbagi ilmu, dan memberi motivasi kepada penulis.
11. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juli 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Cyber Attack atau serangan pada dunia maya saat ini menjadi ancaman utama bagi kemanan jaringan [1]. Dilakukan dengan cara mencari kelemahan pada sistem keamanan jaringan pada suatu komputer atau server, kemudian memanfaatkan kelemahan tersebut untuk menyerang komputer atau server tersebut. Penyerangan yang akan dilakukan oleh *intruder* biasanya tidak terduga oleh sistem administrator. Namun kondisi suatu jaringan komputer dapat diprediksi dengan mengamati perubahan urutan kejadian penyerangan.

Untuk menangani penyerangan yang tak terduga dari intruder, beberapa penelitian telah dilakukan untuk mempelajari serangan yang telah terjadi dan melakukan deteksi intrusi pada jaringan komputer. Seperti halnya penggunaan algoritma *AdaBoost* untuk mendeteksi intrusi pada jaringan komputer [2]. Kemudian pengembangan dilakukan agar tidak hanya bisa mendekteksi serangan intrusi, namun agar bisa memprediksi serangan yang mungkin akan terjadi. Seperti penelitian yang dilakukan dengan menggunakan *framework short term forecasting* dengan algoritma *Neuro-genetic ensemble* [3].

Pada tugas akhir ini, diterapkan metode pemodelan statistika yaitu *Hidden Markov Model* (HMM) untuk memprediksi serangan yang mungkin terjadi pada jaringan komputer. Percobaan sistem prediksi dilakukan pada jaringan komputer Institut Teknologi Sepuluh Nopember Surabaya.

Pemakaian metode HMM pada Tugas Akhir ini dikarenakan HMM memiliki karakteristik berupa perubahan pada *state* internal yang berdasarkan waktu (Markov Model) akan tetapi tidak nampak (*hidden*) dari luar sistem sehingga tidak dapat langsung dilakukan pengamatan. Namun karena jumlah *state* internal yang terbatas dan *state* saat ini (*current state*) bergantung pada *state* sebelumnya [4]

sehingga dapat melakukan pengamatan pada suatu yang berhubungan (misalnya variabel yang berkorelasi) dengan *state* sebelumnya. Karakteristik yang demikian membuat HMM menjadi metode yang tepat dalam memprediksi serangan dalam kondis *real-time*, karena HMM mengamati serangan sebelumnya yang terjadi dan memperkirakan serangan yang akan terjadi berdasarkan kondisi sebelumnya. Dari karakteristik yang demikian, membuat HMM menjadi metode yang tepat untuk membuat prediksi pada suatu jaringan komputer.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana menerapkan *Hidden Markov Model* dalam memprediksi serangan secara *real-time* pada jaringan komputer di ITS?
2. Bagaimana hasil dari prediksi serangan jaringan dari implementasi *Hidden Markov Model* ?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Metode pemodelan yang digunakan untuk prediksi serangan adalah *Hidden Markov Model* (HMM).
2. Prediksi dilakukan secara *real-time* sesuai kondisi dan keadaan jaringan ITS pada saat pengujian.
3. Jenis serangan yang diprediksi adalah SQL *Injection.*

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah untuk memprediksi serangan yang mungkin terjadi pada suatu jaringan komputer

secara *real-time* berdasarkan data serangan jaringan menggunakan metode pemodelan *Hidden Markov Model* (HMM).

1.5 Manfaat

Dengan dibuatnya Tugas Akhir ini maka akan dapat diprediksi dengan lebih akurat tentang serangan yang mungkin terjadi pada suatu jaringan komputer secara *real-time* berdasarkah hasil prediksi dari implementasi metode pemodelan *Hidden Markov Model*.

1.6 Metodologi

Tahapan yang digunakan dalam mengerjakan Tugas Akhir ini adalah sebagai berikut:

1. **Penyusunan Proposal Tugas Akhir**
   Proposal Tugas Akhir berisi mengenai ringkasan latar belakang, rumusan masalah serta apa saja yang dibutuhkan untuk memprediksi serangan secara *real-time* dari hasil prediksi data serangan pada jaringan komputer ITS menggunakan metode *Hidden Makov Model* (HMM).

2. **Studi Literatur**
   Untuk Tugas Akhir ini menggunakan literatur *paper* dan artikel dari internet. *Paper* yang digunakan adalah hasil riset dari Sendi, dkk [1] mengenai prediksi intrusi secara *real-time* berdasarkan sinyal/tanda yang telah dioptimalkan dengan menggunakan *Hidden Markov Model* (HMM). Dan juga *paper* hasil riset dari Haslum, dkk [5] mengenai *distributed monitoring* dari percobaan intrusi menggunakan metode pemodelan *Hidden Markov Model* (HMM).

3. **Analisis dan Desain Perangkat Lunak**
   Pada tahap ini, dimulai menganalisa sistem yang akan dibuat berdasarkan serangan yang terjadi pada jaringan ITS serta algoritma metode yang digunakan (*Hidden*

Markov Model/ HMM) guna mencapai tujuan untuk dapat memprediksi serangan secara *real-time.* Hasil analisa kemudian akan menjadi gambaran kasar/ desain dari proses-proses yang selanjutnya dapat diimplementasikan.

4. **Implementasi Perangkat Lunak**
   Pada tahap implementasi merupakan tahap membangun sistem dengan algoritma yang telah dirancang sebelumnya. Tahapan ini merupakan realisasi rancangan algoritma yang telah dibuat.

5. **Pengujian dan Evaluasi**
   Pada tahap ini dilakukan pengujian terhadap hasil implementasi dengan menggunakan percobaan intrusi yang terjadi pada jaringan ITS secara *real-time* sekaligus evaluasi dilakukan dengan melihat kesesuaian perencanaan.

6. **Penyusunan Buku Tugas Akhir**
   Dalam tahap akhir ini, penulis melakukan penyusunan laporan yang berisikan dokumen pembuatan dan hasil pengerjaan pada perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan
Bab yang berisi latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu perumusan masalah, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II **Tinjauan Pustaka**

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III **Analisis dan Perancangan Perangkat Lunak**

Bab ini berisi tentang dasar dari algoritma yang akan diimplementasikan pada Tugas Akhir ini.

Bab IV **Implementasi**

Bab ini membahas mengenai implementasi dari rancangan yang telah dibuat pada bab sebelumnya.

Bab V **Uji Coba Dan Evaluasi**

Bab ini menjelaskan mengenai kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari perangkat lunak yang telah dibuat sesuai dengan data yang diujikan.

Bab VI **Kesimpulan dan Saran**

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang telah dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 *Hidden Markov Model*

Hidden Markov Model adalah sebuah metode hasil pengembangan dari *Markov Model*. Dalam *Markov Model*, suatu keadaan (*state*) dapat langsung diamati, oleh karenanya probabilitas transisi dari *state* adalah satu-satunya parameter. Sedangkan dalam *Hidden Markov Model*, suatu keadaan (*state*) tidak dapat langsung diamati, tapi *output* bergantung pada *state* yang terlihat. Proses yang terjadi dalam HMM merupakan *finite-state* yang homogen dari *Markov Model* dan tidak dapat diamati secara langsung.

Hidden Markov Model memiliki himpunan *state* (Q), sebuah output alfabet atau yang biasa disebut himpunan observasi (O), matriks probabilitas transisi (A), matriks probabilitas emisi (B), dan matriks probabilitas awal (π). *State* saat ini tidak dapat dilakukan pengamatan secara langsung. Namun, setiap *state* saat juga memiliki output alfabet dengan probabilitas tertentu (B). Biasanya *state* (Q) dan output (O) berdasarkan pemahaman. Oleh karena itu, sebuah *Hidden Markov Model* memiliki 3 komponen utama yang dihitung yaitu (A, B, π) [6].

Secara umum, terdapat 3 aturan untuk penyelesainya masalah dengan *Hidden Markov Model* [6], yaitu :

1. Diberikan model parameter, menghitung probabilitas urutan output tertentu. Diselesaikan dengan algoritma *Forward* dan *Backward*

2. Diberikan model parameter, menemukan urutan yang paling mungkin dari *state* yang menghasilkan urutan keluaran yang diberikan. Diselesaikan dengan algoritma Viterbi. Dalam tugas akhir ini menggunakan solusi ini dikarenakan solusi kedua ini diharuskan melakukan pencatatan *state* dan output.
3. Diberikan model parameter, menemukan kemungkinan himpunan dari *state* dan probabilitas output.

2.1.1 *Markov Model*

Markov model [6] atau juga bisa disebut *Markov chain* adalah sebuah pemodelan dari transisi antar *state*. Dimana setiap transisi memiliki *weight* yang berupa probabilitas transisi tiap dan antar *state*. Misal, terdapat N kejadian, maka akan terdapat N *state*. Selanjutnya didefinisikan seperti dalam persamaan (2.1) berikut :

$$\boldsymbol{Q} = \{ q_i \},\ i = 1, \ldots, N \tag{2.1}$$

Probabilitas dari transisi *state* dihitung berdasarkan waktu [7]. Probabilitas untuk *state* saat ini berdasarkan pada tapat satu *state* sebelumnya, dan begitu seterusnya hingga t waktu. Penghitungan transisi dengan cara ini disebut dengan istilah *fisrt order*. Seperti yang dirumuskan pada persamaan (2.2) berikut :

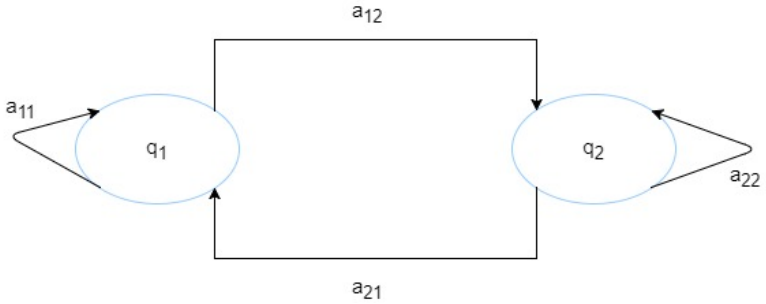
$$a_{ij} = P[q_t = j | q_{t-1} = i] \tag{2.2}$$

Keterangan dari rumus di atas adalah sebagai berikut :

- i, j adalah jenis *state* termasuk dalam himpunan $\boldsymbol{Q}$
- $a_{ij} \geq 0$ merupakan suatu komponen matriks dari indeks i dan j
- $\sum_i^j a = 1$ untuk setiap i, i=1 hingga ke N.

Markov model inilah yang nantinya akan menjadi *hidden state* dalam *Hidden Markov Model.* Berikut adalah gambar dari

pemodelan *Markov model*, dengan contoh jumlah *state* 2 (N = 2) pada **Gambar 2.1**



Gambar 2. 1 Pemodelan *Markov Chain*

2.1.2 Output (O), Probabilitas Emisi (B) dan *Initial State* (π)

Untuk membuat *Markov Model* menjadi *Hidden Markov Model* maka diperlukan tambahan komponen, yaitu output alfabet atau yang biasa disebut himpunan observasi (O), probabilitas emisi (B), serta probabilitas *state* awal (π). Himpunan observasi merupakan sekumpulan kondisi yang dapat teramati dan mempengaruhi perubahan *state*. Misal, terdapat M kondisi yang mempengaruhi, maka akan terdapat M observasi. Selanjutnya didefinisikan seperti dalam persamaan (2.3) berikut :

$$\boldsymbol{O} = \{ o_k \}, k = 1, \ldots, M . \qquad (2.3)$$

Probabilitas dari sebuah kondisi untuk mempengaruhi perubahan suatu *state* disebut sebagai probabilitas emisi. Seperti yang dirumuskan dalam persamaan (2.4) berikut :

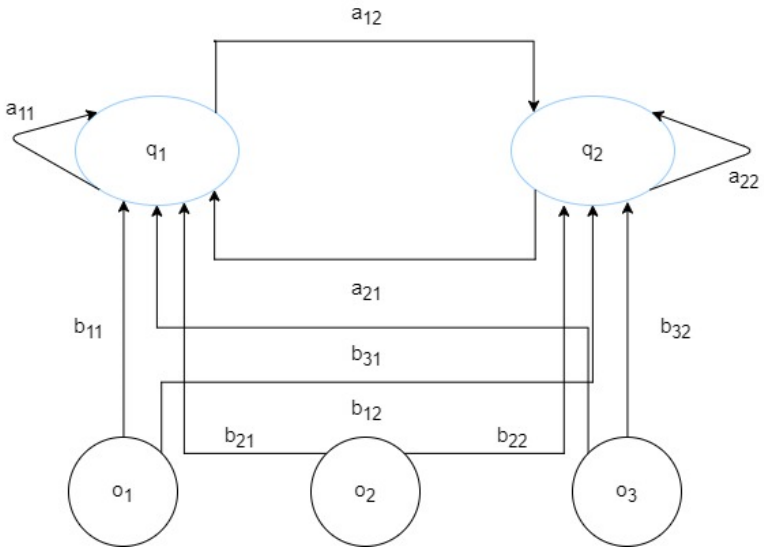
$$\boldsymbol{B} = \{ b_{ik} = b_i(o_k) = P(o_k \mid q_i) \} \qquad (2.4)$$

Keterangan dari rumus di atas adalah sebagai berikut :

- i adalah indeks dari jenis *state* dalam himpunan Q, i = 1 sampai dengan N.

- k adalah indeks dari jenis observasi dalam himpunan O, k = 1 sampai dengan M.
- o adalah anggota himpunan O
- q adalah anggota himpunan Q

Berikut terdapat gambaran dari hubungan Q, O, A, dan B pada **Gambar 2.2**



Gambar 2. 2 Pemodelan *Hidden Markov Model*

Sedangkan probabilitas *initial state* adalah probabilitas kejadian awal, sebelum dilakukan *Hidden Markov Model.* Dinotasikan seperti pada persamaan (2.5) berikut :

$$\boldsymbol{\Pi} = \{p_i = P(q_i \text{ at } t = 1)\} \qquad (2.5)$$

Dengan keterangan rumus di atas adalah sebagai berikut :

- i = 1 sampai dengan N
- p_i adalah probabilitas untuk *state* ke-i

- q_i adalah *state* ke-i
- t adalah waktu

2.1.3 Algoritma Viterbi

Algoritma Viterbi adalah algoritma *dynamic programming* untuk menemukan kemungkinan rangkaian *state* yang tersembunyi yang dihasilkan pada rangkaian pengamatan kondisi [8].

Algoritma Viterbi mencari nilai paling besar dari hasil perhitungan probabilitas awal (π), probabilitas transisi (A), serta probabilitas emisi (B). Saat pertama kali berjalan, Viterbi akan mencari nilai terbesar dari perkalian probabilitas awal dan probabilitas emisi dari kondisi yang mempengaruhi kejadian, seperti pada persamaan (2.6) berikut :

$$P = P_{start(state)} \; x \; P_{emisi(kondisi|state)} \quad (2.6)$$

Nilai terbesar akan menjadi *state* selanjutnya, dan termasuk dalam himpunan rangkaian *state* yang diprediksi. Kemudian, nilai terbesar dari probabilitas *state* saat itu, probabilitas transisi dari *state* saat itu ke *state* selanjutnya, serta probablitas emisi dari kondisi yang mempengaruhi kejadian, seperti pada persamaan (2.7) berikut :

$$P = P_{state} \; x \; P_{transisi(now \rightarrow next)} x \; P_{emisi(kondisi|state)} \quad (2.7)$$

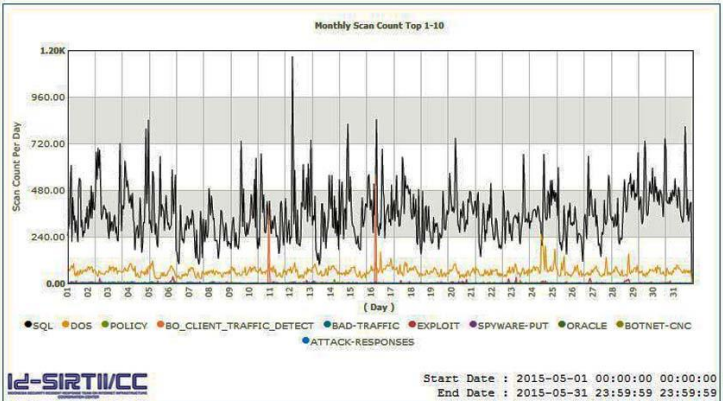
Perhitungan dan pencarian nilai terbesar dilakukan terus hingga input observasi selesai.

2.2 *SQLInjection*

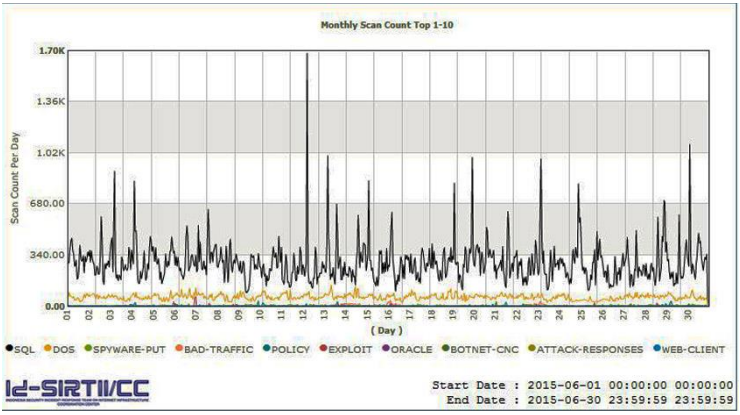
SQLInjeciton adalah sebuah serangan dengan cara memasukkan atau menambahkan kode SQL ke dalam aplikasi atau parameter pada input user yang mana kemudian akan diteruskan ke back-end SQL server untuk parsing dan eksekusi [9]. Dengan

serangan melalui SQL, penyerang dapat melalui autentikasi, mengakses, memodifikasi, serta menghapus data dalam *database*.

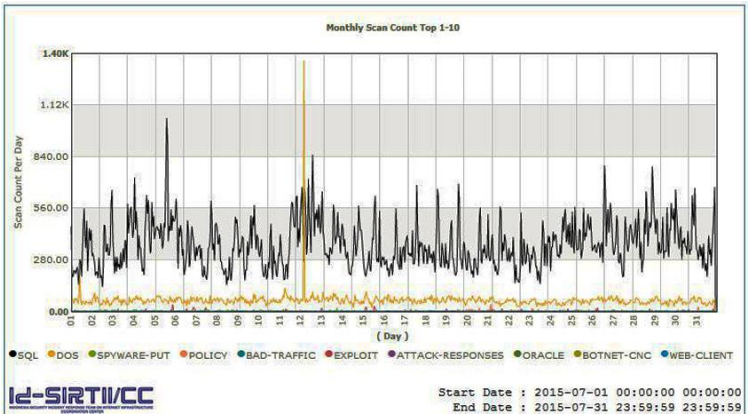
SQLInjection dipilih sebagai serangan yang akan diprediksi karena merupakan salah satu jenis serangan jaringan yang paling sering terjadi. Menurut laporan anomali trafik jaringan hasil riset ID-SIRTII/CC pada bulan Mei 2015, Juni 2015, dan Juli 2015 serangan SQL menjadi serangan yang paling sering terjadi [10] [11] [12].



Gambar 2. 3 Grafik Data Anomali Trafik Jaringan Bulan Mei 2015



Gambar 2. 4 Grafik Data Anomali Trafik Jaringan Bulan Juni 2015



Gambar 2. 5 Grafik Data Anomali Trafik Jaringan Bulan Juli 2015

2.3 *Python*

Python adalah sebuah bahasa pemrograman tingkat tinggi dan bersifat *open-source*. Karena bersifat *open-source* ini *library* Python sangat berkembang, biasanya *development* dari *library* dibangun menggunakan bahasa C/C++. Python memiliki modul, *class*, *exception*, serta tipe data paling tinggi dan *dynamic typing* [13].

Pada tugas akhir ini, untuk membangun komponen prediksi hasil implementasi dari *Hidden Markov Model* dengan menggunakan Python 2.7.

2.4 *Hmmlearn*

Hmmlearn merupakan library *open-source* Python yang menerapkan algoritma *Hidden Markov Model*. Dibangun dengan *library* scikit-learn, NumPy, SciPy, dan matplotlib [14].

Penggunaan *library* hmmlearn karena termasuk *library* yang sederhana dalam implementasi model dari HMM, *open-source* sehingga dapat digunakan secara komersial.

2.5 *Numpy*

Numpy adalah paket dasar untuk komputasi ilmiah pada bahasa pemrograman Python. Selain kegunaannya dalam hal ilmiah, Numpy juga dapat digunakan sebagai wadah multi dimensi yang efisien [15].

Pada tugas akhir ini Numpy dipanggil sebagai *library* untuk menggunakan *array multi-dimensi* yang berguna sebagai penyimpanan probabilitas transisi, probabilitas emisi, dan probabilitas *state* awal.

2.6 *InfluxDB*

InfluxDB adalah sebuah DBMS *open-source* untuk data yang berbentuk *time-series*.

Data *time-series* adalah bentuk data dari hasil sekumpulan pengamatan suatu *item* yang terdefinisi dengan baik melalui pengukuran berulang dari waktu ke waktu. Misal mengukur tingkat pengangguran setiap bulan dalam satu tahun dan hasilnya terdiri atas deret waktu. Sebelumnya, pengangguran telah terdefinisi dengan baik (untuk membedakan dengan yang bukan pengangguran), kemudian dihitung jumlahnya secara konsisten pada interval yang sama [16]. Data *time-series* digunakan untuk mencatat *state* dan jumlah serangan yang telah terdeteksi oleh komponen pendeteksi serangan. Berikut adalah contoh data *time-series* untuk catatan kondisi sebuah server atau jaringan komputer pada rentang waktu 10:00 sampai 10:05 WIB dengan interval pencatatan setiap 1 menit pada **Tabel 2.1**.

Tabel 2. 1 Contoh Bentuk Data *Time-Series*

time	code_state	state	num_attack
10:00:00	0	Normal	0
10:01:00	0	Normal	0
10:02:00	1	Attempt	673
10:03:00	1	Attempt	987
10:04:00	2	Progress	1500

Dari penjelasan bentuk data *time*-series, untuk satu *record* membutuhkan waktu sebagai penanda, oleh karena itu digunakan InfluxDB sebagai penyimpanan data *time*-series. InfluxDB ditujukan sebagai penyimpanan dari beberapa contoh kasus yang mencakup data dalam jumlah besar seperti *timestamped data*, matriks aplikasi, sensor data IoT dan analisis *real-time* [17].

2.7 *NodeJS*

NodeJS adalah sebuah JavaScript runtime yang dibangun dari mesin Chrome's V8 JavaScript. NodeJs menggunakan sebuah *event-driven*, model *non-blocking* I/O yang membuat NodeJS menjadi ringan dan efisien. Paket ekosistem dari NodeJS adalah npm. Npm merupakan ekosistem terbesar dari *open-source library*.

Pada tugas akhir ini NodeJS digunakan untuk membangun komponen deteksi serangan jaringan.

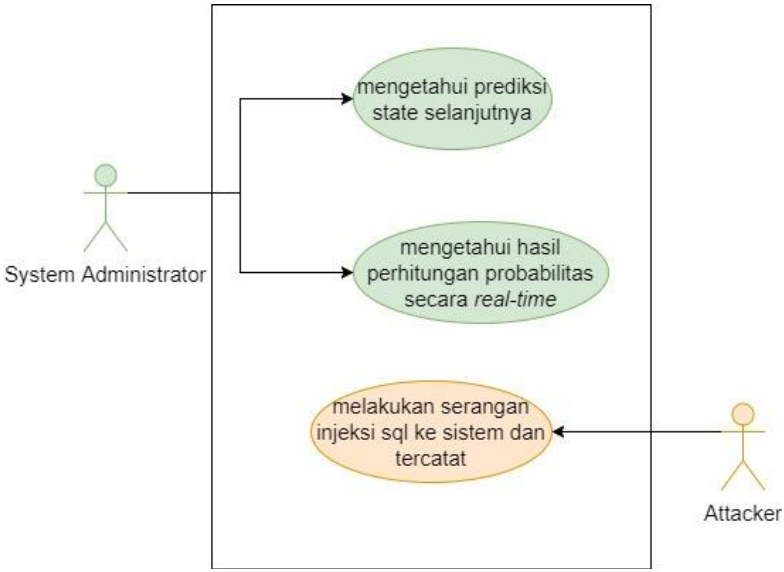
[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan perancangan sistem prediksi dengan implementasi dari algortima *hidden markov model*.

3.1 Kasus Penggunaan

Terdapat satu aktor dalam sistem yaitu sistem administrator. Sistem administrator adalah aktor yang mengoperasikan sistem,. Diagram kasus penggunaan menggambarkan kebutuhan-kebutuhan yang harus dipenuhi sistem. Diagram kasus penggunaan digambarkan pada **Gambar 3.1**.



Gambar 3. 1 Diagram Kasus Penggunaan

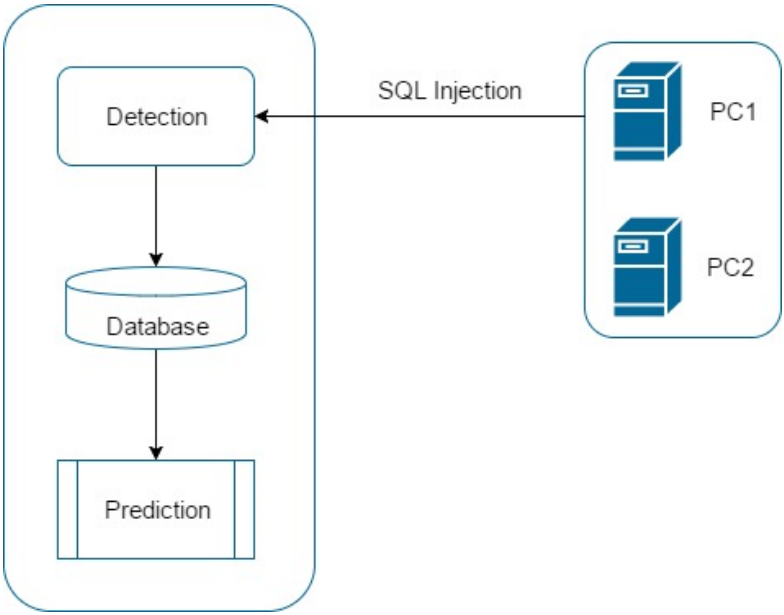
Diagram kasus penggunaan pada Gambar 3.1 dideskripsikan masing-masing pada Tabel 3.1.

Tabel 3. 1 Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0001	Mengetahui prediksi state selanjutnya	Sistem administrator dapat mengetahui hasil prediksi state selanjutnya
UC-0002	Mengetahui hasil perhitungan probabilitas secara *real-time*	Sistem administrator mendapatkan hasil perhitungan probabilitas secara *real-time*
UC-0003	Melakukan serangan injeksi sql ke sistem dan tercatat	Attacker atau penyerang dapat melakukan serangan injeksi sql ke sistem dan tercatat

3.2 Perancangan Sistem Prediksi

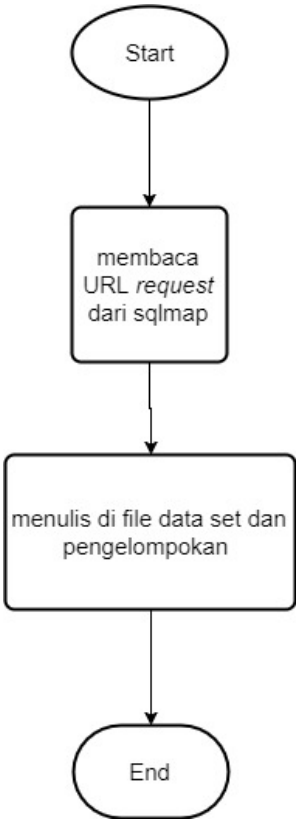
Pada sub-bab ini dijelaskan mengenai sistem prediksi yang akan dibangun. Sistem prediksi terdiri atas beberapa komponen, seperti : *Intrusion Detection System*, pengoleksi data, serta prediksi serangan. Struktur sistem prediksi yang digunakan seperti pada **Gambar 3.2.**



Gambar 3. 2 Desain Sistem Prediksi Intrusi

Berdasarkan gambar di atas, terdapat komponen untuk mendeteksi serangan. Komponen pendeteksi ini adalah *Intrusion Detection System* (IDS). IDS dibangun dengan menggunakan bahasa NodeJS. IDS yang dibangun mendeteksi *vulnerable* URL yang berusaha berkali-kali dikirimkan oleh penyerang. Pendeteksian *vulnerable* URL dilakukan dengan cara mencocokkan URL yang telah dibaca oleh IDS dengan data set dari *vulnerable* URL yang sebelumnya telah disiapkan. Penyiapan data set untuk IDS adalah dengan cara membaca URL *request* yang diterima dari serangan sqlmap, kemudian disimpan dalam file data set sesuai dengan tingkatan serangan dari sqlmap. Serangan dari sqlmap memiliki tingkatan berdasarakan *level* dan *risk*. Terdapat 5 tingkatan *level* dan 3 tingkatan *risk*. Jika *risk* 1 dan *level* kurang dari 3 maka serangan termasuk dalam *state Attempt*, jika *risk* 2 atau 3 dan *level* serangan kurang dari 3 maka serangan termasuk dalam

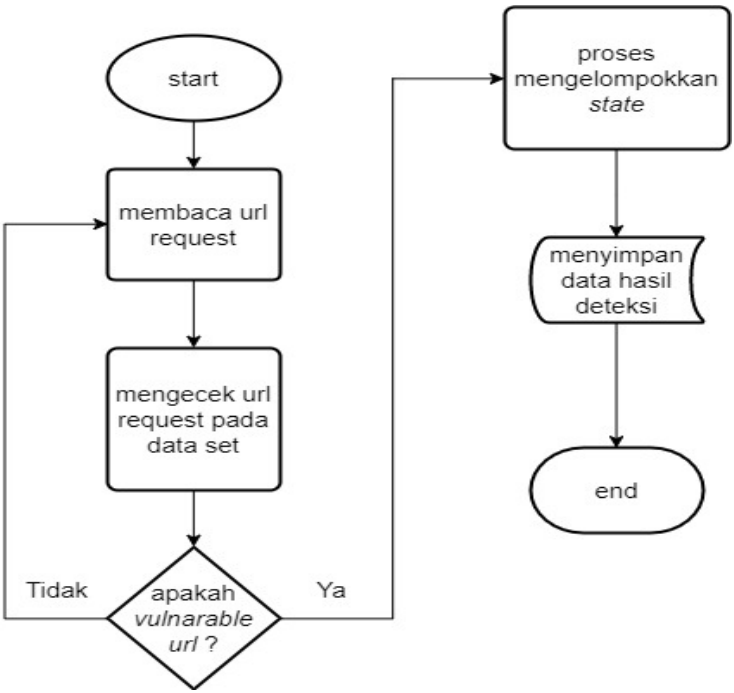
state Progress, dan berapapun *risk*-nya jika *level* lebih dari sama dengan 3 maka termasuk *state Compromise*. **Gambar 3.3** menjelaskan tentang diagram alur dari pembuatan data set IDS.



Gambar 3. 3 Diagram Alur Pmebuatan Data Set IDS

IDS juga melakukan perhitungan jumlah serangan setiap menitnya serta mengelompokkan serangan yang terjadi ke dalam beberapa *state*. *State* tersebut adalah *Normal, Attempt, Progress, Compromise*.

Setelah serangan dideteksi, jumlah serangan dan *state* disimpan dalam bentuk data *time-series* ke dalam *database*. *Database* yang digunakan adalah InfluxDB, karena dapat menyimpan banyak data dari hasil deteksi dan terdapat *timestamp* data sehingga cocok untuk data *real-time*. Diagram alur kerja dari IDS seperti tertera pada **Gambar 3.4.**



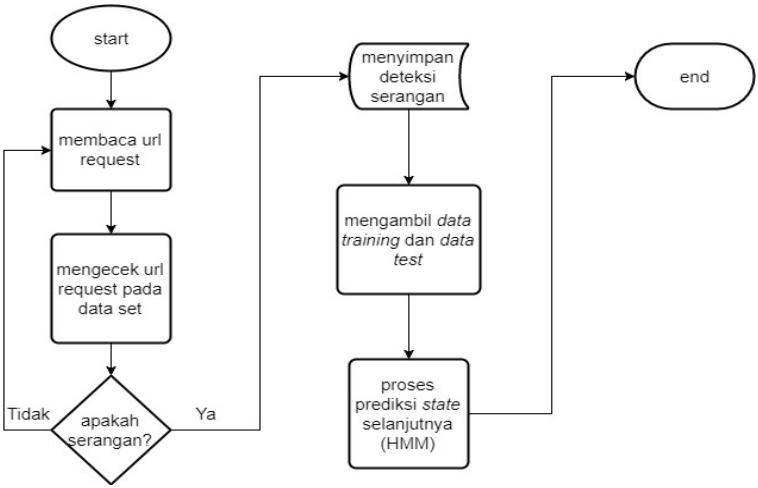
Gambar 3. 4 Diagram Alur Intrusion Detection System

Satu data yang disimpan dalam database InfluxDB memiliki atribut waktu (*timestamp*), *state*, nomor *state* (*state_num*), serta jumlah serangan (*jml_attack*). Rancangan dari penyimpanan pada InfluxDB seperti tertera pada **Tabel 3.2.**

Tabel 3. 2 Contoh Penyimpanan Data Serangan pada InfluxDB

timestamp	code_state	state	jml_attack
2017/6/6 20:30:09	0	normal	0
2017/6/6 20:31:09	0	normal	0
2017/6/6 20:32:09	1	attempt	250
2017/6/6 20:33:09	1	attempt	500
2017/6/6 20:34:09	2	progress	1500
2017/6/6 20:35:09	3	compromise	2500

Data dari InfluxDB kemudian diambil oleh komponen prediksi setiap 1 jam terakhir untuk dijadikan sebagai *data training* agar prediksi bisa memiliki probabilitas dari *log state* sebelumnya dan bisa digunakan secara *real-time* karena probabilitas yang digunakan oleh *Hidden Markov Model* akan selalu *update* setiap waktunya sesuai waktu penyerangan. Pengambilan *data test* juga diambil dari database dan berupa data *time-series*. Diagram alur kerja sistem prediksi serangan jaringan seperti pada **Gambar 3.5**.



Gambar 3. 5 Diagram Alur Sistem Prediksi Intrusi

3.3 Perancangan Penerapan Algoritma Hidden Markov Model

Pada sub-bab ini dijelaskan mengenai perancangan penerapan algoritma *Hidden Markov Model*. Dimulai dari pengambilan *data traning* dan *data test* dari *database* InfluxDB. *Data training* digunakan untuk membuat probabilitas yang *real-time*. Kemudian *data test* digunakan untuk diprediksi *state* selanjutnya.

Penggunaan *Hidden Markov Model* sebagai metode yang digunakan karena, *Hidden Makorv Model* dapat menangani permasalahan *streaming mode*, yaitu tidak seringnya terjadi suatu pola pada berjalannya waktu. Sehingga akan sulit diprediksi apakah terjadi serangan pada waktu selanjutnya atau tidak. HMM bekerja dengan baik terhadap input *streaming*. *Hidden Markov Model* adalah *Markov Model* statistik dengan *state* yang unobserved atau *state* yang tidak dapat di observasi secara langsung. Pada HMM *state* tidak dapat diamati secara langsung, namun hasil keluaran/ *output* bergantung pada *state* yang nampak. HMM berguna untuk menilai resiko dan memprediksi keadaan yang akan dating pada sistem deteksi intrusi.

Berikut adalah penjelasan mengenai elemen-elemen dari HMM yang digunakan pada tugas akhir ini.

1. *State*

   Sistem diasumsikan sedang berada pada salah satu *state* berikut. *State* merupakan kondisi yang menjelaskan keadaan dari sistem itu. Pada tugas akhir ini terdapat 4 *state* yang dapat menjadi gambaran keadaan dari sistem. *State* tersebut antara lain : *Normal, Attempt, Progress*, serta *Compromise*.

   - Normal (N) : merupakan *state* yang mengindikasikan sistem berjalan baik-baik saja, tidak terdapat aktivitas mencurigakan atau usaha mauapun percobaan apapun yang mecoba untuk masuk ke sistem.
   - Attempt (A) : mengindikasikan bahwa telah terjadi aktivitas mencurigakan yang telah dicoba untuk

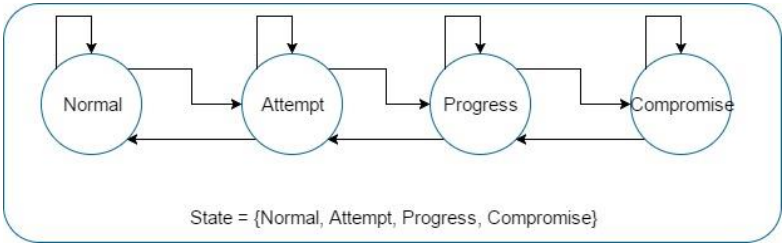
menyerang sistem. Pada tugas akhir ini *state Attempt* ditandai dengan penyerang mulai mencoba untuk mencari-cari celah agar bisa melakukan sqlinjection.

- Progress (P) : mengindikasikan bahwa serangan telah dimulai dan saat ini sedang dalam keadaan berkembang. Pada tugas akhir ini *state Progress* ditandai dengan penyerangan sudah mulai melakukan *sqlinjection* berbasis waktu yang presisi.
- Compromise (C): mengindikasikan bahwa serangan telah sukses membobol sistem. Pada tugas akhir ini *state Compromise* ditandai dengan penyerang sudah melakukan query dml seperti INSERT, UPDATE, DELETE.

Pada tugas akhir ini digunakan N,A,P,C untuk merepresantikan *state* di atas. Sehingga dalam notasi HMM adalah seperti persamaan (3.1) berikut :

$$S_i = \{s_1 = N, s_2 = A, s_3 = P, s_4 = C\} \quad (3.1)$$

Gambar 3.6 menjelaskan hubungan antar *state* pada *Hidden Markov Model* yang digunakan.



Gambar 3. 6 Hubungan antar state untuk *Hidden Markov Model*

2. Observasi ($O_i = \{O_1, O_2, O_3, \dots, O_n\}$)
   Observasi merupakan model yang dapat langsung di observasi untuk prediksi. Observasi yang membuat *State* dapat berpindah dari satu *state* ke *state* lain. Pada tugas akhir

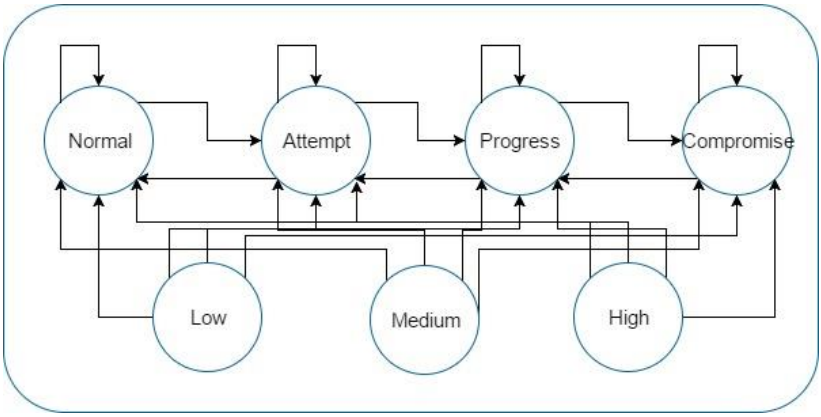
ini observasi adalah banyak jumlah serangan atau sql yang diinjeksikan dalam satuan waktu tertentu. Dalam x waktu dihitung berapa jumlah serangan berupa sql. Kemudian dari jumlah serangan tersebut dikelompokkan menjadi *Low*, *Medium*, dan *High*. *Low* untuk jumlah serangan 0-1000. *Medium* untuk jumlah serangan 1001-10000. *High* untuk jumlah serangan lebih dari 10000. Perhitungan jumlah serangan dilakukan setiap satu menit sekali.

3. Probabilitas Transisi *State* (Λ)
   Probabilitas Transisi *State* merupakan matriks yang menggambarkan probabilitas dari perubahan antar *State*. Probabilitas Transisi *State* berubah setiap kali sistem prediksi dijalankan bergantung sesuai dengan log *state* terakhir yang diambil oleh sistem prediksi untuk menjadi probabilitas.

4. Probabilitas Emisi Observasi (ϕ)
   Probabilitas Emisi Observasi merupakan matriks yang menggambarkan probabilitas kejadian suatu *state* berdasarkan pada observasi. Probabilitas Emisi Observasi berubah setiap kali sistem prediksi dijalankan sesuai dengan log jumlah serangan terakhir yang diambil oleh sistem prediksi untuk menjadi probabilitas.

5. *Initial State Distribution* (π)
   Initial State Distribution menjelaskan tentang probabilitas dari *state* ketika sistem pertama kali dijalankan. *Initial State Distribution* berubah setiap kali sistem prediksi dijalankan.

Gambar 3.7 menjelaskan tentang hubungan antara *state* dan observasi.

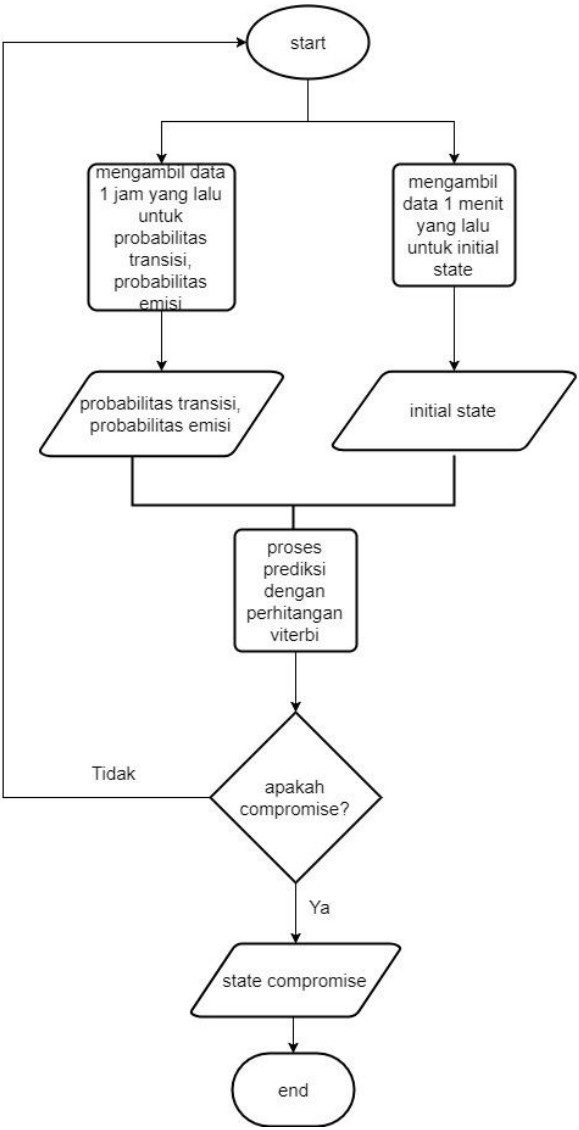


Gambar 3. 7 Hubungan antara State dan Observasi

Dari sistem prediksi yang dicari adalah status saat *Compromise State*. Karena *state* tersebut menggambarkan kondisi komputer yang telah dalam keadaan terserang secara sukses.

Dalam proses prediksi HMM, digunakan algoritma Viterbi untuk memprediksi *state* selanjutnya dari server yang ingin dipasangi alat prediksi. Algoritma Viterbi digunakan oleh HMM untuk melakukan perhitungan pada "jalan" dari *state* yang paling mungkin terjadi. Dengan kata lain, algoritma Viterbi menentukan *state* selanjutnya dengan melihat probabilitas paling besar selanjutnya mengarah ke *state* apa untuk waktu yang akan datang. Algoritma Viterbi melakukan perhitungan dengan menggunakan probabilitas yang telah disediakan oleh algoritma *Hidden Markov Model*, yaitu probabilitas awal dari *state*, probabilitas transisi *state*, serta probabilitas transisi obervasi. Inputan berupa log dari jumlah serangan yang tercatat oleh IDS setiap menitnya.

Berikut adalah gambar diagram alur dari algoritma HMM dengan perhitungan algoritma Viterbi yang digunakan untuk memprediksi serangan jaringan, sesuai dengan **Gambar 3.8.**



Gambar 3. 8 Diagram Alur Prediksi dengan Hidden Markov Model

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Lingkungan yang akan digunakan untuk melakukan proses implementasi menggunakan komputer fisik dengan spesifikasi Intel(R) Core(TM) i3-2120 CPU @ 3.20 Hz dengan memori 4 GB di lantai 4 Teknik Informatika ITS. Perangkat Lunak yang digunakan dalam pengembangan adalah sebagai berikut :

- Sistem Operasi Linux Mate
- Editor Sublime text 3
- Git versi 1.9.1 untuk pengolahan versi program
- NodeJS versi 4.2.3 untuk pengembangan aplikasi *Intrusion Detection System*
- Python versi 2.7 untuk membuat komponen prediksi *Hidden Markov Model*
- InfluxDB versi 1.2 untuk menyimpan data *time-series* secara *real-time*
- Sqlmap versi 1.0.0.15 untuk melakukan serangan kepada sistem prediksi yang telah dibuat

4.2 Implementasi Sistem Prediksi

Sistem prediksi yang terdiri atas 3 komponen utama yaitu komponen IDS (*Intrusion Detection System*), komponen penyimpanan data *time-series* dengan menggunakan InfluxDB, serta komponen prediksi hasil implementasi *Hidden Markov Model*.

4.2.1 Implementasi *Intrusion Detection System*

IDS yang digunakan pada tugas akhir ini dibangun dengan mneggunakan NodeJS. Pada NodeJS diperlukan beberapa modul untuk diinstal agar bisa memabaca URL *request*, mengecek pada data set *vulnerable* URL serta memasukkan data pada InfluxDB.

Pada mulanya IDS membuat data set untuk URL yang *vulnerable* dari serangan sqlmap. Data set tersebut berisi *string query* yang diinjeksikan dari sqlmap ke server dan disimpan dalam bentuk file. Penyimpanan data set dibedakan menjadi 3 tingkatan sesuai dengan *risk* dari sqlmap. Setiap *risk* disimpan dalam 1 file dataset tersendiri. Sehingga terdapat 3 file data set untuk IDS dari serangan sqlmap.

Rangka alur kerja untuk pembuatan data set *risk* 1 seperti pada *pseudocode* **Gambar 4.1.**

```
1  get request url
2
3  open file risk1
4  append file risk1(request.query)
5      if error
6            print error
7      else
8            print success save to
             file
```

Gambar 4. 1 Pseudocode pembuatan data set risk 1

Rangka alur kerja untuk pembuatan data set *risk* 2 seperti pada *pseudocode* **Gambar 4.2.**

```
1  get request url
2
3  open file risk2
4  append file risk2(request.query)
```

```
5       if error
6             print error
7       else
8             print success save to
              file
```

Gambar 4. 2 Pseudocode pembuatan data set risk 2

Rangka alur kerja untuk pembuatan data set *risk* 3 seperti pada *pseudocode* **Gambar 4.3.**

```
1   get request url
2
3   open file risk3
4   append file risk3(request.query)
5       if error
6             print error
7       else
8             print success save to
              file
```

Gambar 4. 3 Pseudocode pembuatan data set risk 3

Penjelasan dari *pseudocode* pembuatan data set IDS adalah sebagai berikut :

- *Request url* adalah url yang dikirimkan oleh klien dan diterima kemudian dibaca untuk selanjutnya dapat dibalas dengan *response url*.
- *Request url* tersebut dibaca oleh IDS
- IDS mengambil bagian *query* pada *request url* kemudian di simpan pada file *risk1, risk2, risk3*.

IDS ini bekerja dengan cara membaca URL *request* dari klien, kemudian mengecek URL *request* tersebut apakah ada pada file data set, jika ada berarti URL *request* termasuk serangan SQL

Injection. Kemudian menghitung *request url* tersebut sebagai serangan. Rangka alur kerja untuk pengecekan apakah *request* URL yang didapatkan oleh IDS merupakan *vulnerable* URL seperti pada *pseudocode* **Gambar 4.4**.

```
 1  vulnarable1 = read file risk 1
 2  vulnarable2 = read file risk 2
 3  vulnarable3 = read file risk 3
 4
 5  get request url
 6      if request.query == vulnarable1
 7          state1 = state1 + 1
 8      if request.query == vulnarable2
 9          state2 = state2 + 1
10      if request.query == vulnarable3
11          state2 = state2 + 1
12      if request.query == INSERT or
        request.query == UPDATE or
        request.query == DELETE
13          state3 = state3 + 1
14      total = total + 1
```

Gambar 4. 4 Pseudocode Pengecekan Request URL pada IDS

Penjelasan dari *pseudocode* pengecekan *request url* IDS adalah sebagai berikut :

- *Vulnarable1* adalah variabel yang menyimpan hasil membaca file *risk1/* data set IDS.
- *Vulnarable2* adalah variabel yang menyimpan hasil membaca file *risk2/* data set IDS.
- *Vulnarable3* adalah variabel yang menyimpan hasil membaca file *risk3/* data set IDS.

- *State1* merupakan jumlah *request query* yang *vulnerable* hasil mencocokkan dengan file *risk1*.
- *State2* merupakan jumlah *request query* yang *vulnerable* hasil mencocokkan dengan file *risk2* dan *risk3*.
- *State3* merupakan jumlah *request query* yang *vulnerable* dimana terdapat aksi INSERT, UPDATE ataupun DELETE.
- *Total* merupakan jumlah *request url* yang terdeteksi sebagai serangan baik dari data set *risk1, risk2,* ataupun *risk3.*

Selanjutnya IDS mengelompokkan serangan tersebut kepada tingkatan serangan serta menghitung total jumlah serangan yang diterima setiap satu menit. Atribut yang dicari pada IDS antara lain adalah jumlah serangan dalam tiap menit, tingkatan serangan, kondisi komputer dalam menit tersebut (*Normal, Attempt, Progress, Compromise*), serta nomor dari *state* saat itu (*Normal = 1, Attempt = 2, Progress = 3, Compromise = 4*). Rangka alur kerja untuk mengelompokkan serangan yang berhasil dideteksi IDS pada *state* seperti pada *pseudocode* **Gambar 4.5**.

```
15 Fungsi hitung
16      normal = state0 / total
17      attempt = state1/ total
18      progress = state2 / (2*total)
19      compromise = state3 / total
20
21      state = "normal"
22      state_num = 0
23      if compromise and progress > attempt
24          state = "compromise"
25          state_num = 3
26      else if progress > attempt
27          state = "progress"
```

```
28          state_num = 2
29      else if attempt > 0
30          state = "attempt"
31          state_num = 1
32
33      if not attempt
34          attempt = 0
35      if not progress
36          progress = 0
```

Gambar 4. 5 Pseudocode Pengelompokan Serangan pada IDS

Penjelasan dari *pseudocode* pengelompokan serangan pada IDS adalah sebagai berikut :

- Awalnya state0, state1, state2, state3 memiliki nilai 0 sebelum memulai pengecekan *request query*.
- Kemudian setelah melalui pengecekan dan penambahan nilai sesuai dengan hasil deteksi, dilakukan perhitungan agar bisa dipetakan pada *state* dari HMM.
- *State Normal* terjadi ketika pada state0, state1, state2, state3 memiliki nilai 0. Atau dengan kata lain tidak terdeteksi serangan sama sekali. Sehingga jumlah serangan bernilai 0.
- *State Attempt* terjadi ketika state1 memiliki nilai namun hasil pembagiannya dengan total serangan lebih besar dibandingkan dengan hasil pembagian state2 dengan total serangan. Dengan catatan state3 tidak memiliki nilai atau bernilai 0.
- *State Progress* terjadi ketika state2 memiliki nilai namun hasil pembagiannya dengan total serangan lebih besar dibandingkan dengan hasil pembagian state1 dengan total serangan. Dengan catatan state3 tidak memiliki nilai atau bernilai 0.

- *State Compromise* terjadi ketika state2 dan state3 memiliki nilai atau lebih besar dari pada 0. Dengan catatan hasil pembagian state2 dengan total serangan pada waktu tersebut lebih banyak dibandingkan dengan hasil pembagian state1 dengan total serangan pada waktu tersebut.
- *State_num* merupakan variable yang menyimpan representasi *state* berupa angka. *Normal* = 0, *Attempt* = 1, *Progress* = 2, *Compromise* = 3.

Kemudian data dari atribut yang telah didapatkan oleh IDS disimpan dalam database InfluxDB dengan format data JSON.

4.2.2 Implementasi Penyimpanan Data *Time-Series*

Penyimpanan data secara *time-series* dipilih karena dapat memudahkan bagi komponen prediksi untuk membaca dan mengolah data yang dibutuhkan. Selain itu karena pada sistem yang dirancang dari tugas akhir ini dapat memprediksi secara *real-time* sehingga penyimpanan secara *time-series* dirasa lebih mudah.

Implementasi dari penyimpanan dari data *time-series* dilakukan dengan menggunakan *database* InfluxDB. Pertama yang harus dilakukan adalah menginstall InfluxDB pada komputer implementasi. Jika sudah berhasil maka influxDB berjalan di *localhost* komputer pada *port* 8086.

Data yang dikirimkan oleh IDS, disimpan pada *database* 'IDS' dan *measurement* 'states' pada InfluxDB. Rangka alur kerja dari IDS sesuai dengan *pseudocode* **Gambar 4.6.**

```
37  write to states //menulis data pada
    series states
38       state = state
39       state_num = state_num
40       jml_attack = total
```

Gambar 4. 6 Pseudocode Insert Data dari NodeJS ke InfluxDB

Hasil inputan dari IDS pada InfluxDB seperti yang tertera pada **Gambar 4.7** berikut

```
name: states_5
--------------
time                    jml_attack      state           state_num
1499142453793571774     0               normal          0
1499142513817585217     0               normal          0
1499142573826814979     665             attempt         1
1499142633839252517     10603           attempt         1
1499142693867973317     17966           progress        2
1499142753889233425     7130            compromised     3
1499142813902879705     10219           compromised     3
```

Gambar 4. 7 Tampilan Penyimpanan Hasil Deteksi Serangan

4.3 Implementasi Penerapan Algoritma *Hidden Markov Model*

Data yang telah tersimpan di InfluxDB kemudian akan digunakan oleh komponen prediksi yaitu dengan metode statistik *Hidden Markov Model*. Dari InfluxDB komponen prediksi membutuhkan *log state* dari komputer yang diserang. *Log state* berguna untuk menentukan probabilitas yang dibutuhkan oleh HMM. Seperti probabilitas transisi dari *state* dalam jangka kurun waktu tertentu, probabilitas emisi yaitu kemungkinan kondisi *state* bergantung pada jumlah serangan yang berhasil dideteksi dan probabilitas inisiasi yaitu probabilitas dari state diambil dari kondisi *state* terakhir. Data untuk probabilitas disimpan dalam bentuk matriks array. Rangka alur kerja dari perhitungan probabilitas transisi dan emisi sesuai dengan *pseudocode* 4.8 dan *pseudocode* 4.9.

```
1  result = select from states where
   time > now - 1h
2  now = select last insert from
   states
3
4  for data in result
```

```
5       tentukan kategori jumlah
        serangan tiap data
6               jumlah++
7   return jumlah/len(data)
```

Gambar 4. 8 Pseudocode Perhitungan Probabilitas Emisi

```
1   result = select from states where
    time > now - 1h
2   now = select last insert from
    states
3
4   for data in result
5           hitung jumlah state
6                   jumlah++
7   return jumlah/len(data)
```

Gambar 4. 9 Pseudocode Perhitungan Probabilitas Transisi

Setelah mengambil data untuk probabilitas, kemudian ambil *record* jumlah serangan yang terjadi dalam x waktu dengan hitungan jumlah per menit. *Record* jumlah serangan akan diprediksi kondisi *state* selanjutnya dengan menggunakan algoritma *Veterbi*. Rangka alur kerja dari IDS sesuai dengan *pseudocode* 4.10.

```
1   hasil prediksi = fungsi
    viterbi(emisi_prob, trans_prob,
    start_prob, data_predict)
2   return hasil prediksi
```

Gambar 4. 10 Pseudocode Prediksi

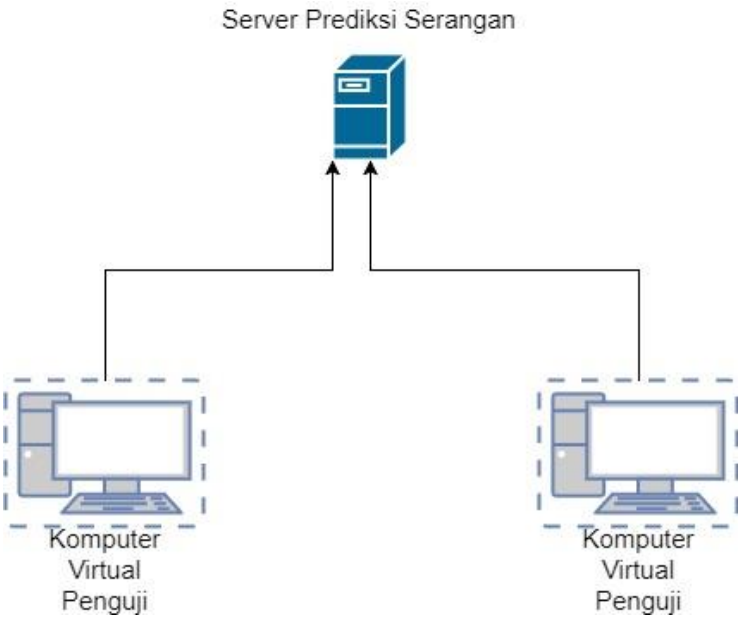
Implementasi algoritma HMM menggunakan bahasa Pyhton dan menggunakan library hmmlearn untuk *hidden markov model*. Kemudian data probabilitas yang disimpan dalam bentuk array menggunakan *library* Numpy.

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

5.1 Lingkungan Uji Coba

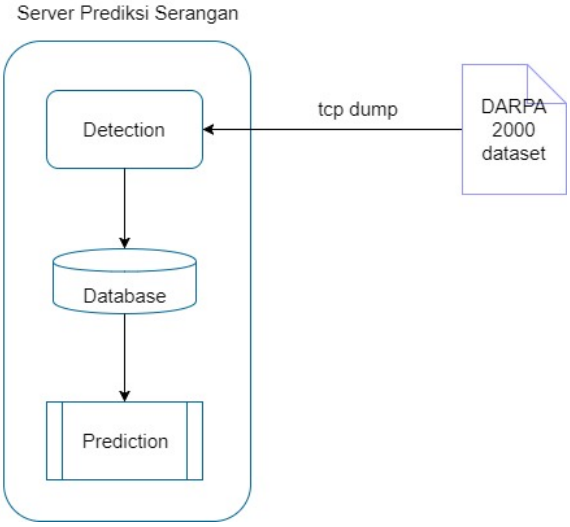
Lingkungan pengujian menggunakan sebuah server untuk IDS, *database*, serta Prediksi; serta 2 buah komputer virtual untuk melakukan serangan terhadap server secara *real-time*. Server berada pada Lantai 4 Teknik Informatika ITS, sedangkan komputer pengujian dilakukan di Laboratorium Arsitektur dan Jaringan Komputer Jurusan Teknik Informatika ITS. **Gambar 5.1** menggambarkan tentang arsitektur dari scenario uji coba yang akan dilakukan.



Gambar 5. 1 Arsitektur Skenario Uji Coba dengan Data *Real-Time*

Spesifikasi untuk setiap komponen yang digunakan adalah sebagai berikut:

- Server Prediksi Serangan:
    - Perangkat Keras:
        - Intel(R) Core(TM) i3-2120 CPU @ 3.20 Hz
        - RAM 4 GB
        - Ethernet interface
    - Perangkat Lunak:
        - Sistem Operasi Linux Mate
        - NodeJS versi 4.2.3
        - Python versi 2.7
        - InfluxDB versi 1.2
- Komputer Virtual Penguji:
    - 2 buah
    - Perangkat Lunak:
        - Sistem Operasi Ubuntu 16.04 LTS
        - Sqlmap versi 1.0.0.15
        - RAM 512 MB



Gambar 5. 2 Arsitektur Skenario Uji Coba dengan Data DARPA 2000

5.2 Skenario Uji Coba

Uji coba dilakukan untuk mengetahui keberhasilan sistem yang telah dibangu. Skenario pengujian dibedakan menjadi 2 bagian yaitu :

1. **Uji Fungsionalitas**
   Pengujian ini didasarkan pada fungsionalitas yang disajikan sistem. Uji coba yang akan dilakukan adalah uji dari fungsionalitas dari IDS, fungsi perhitungan probabilitas secara *real-time* serta fungsionalitas implementasi program prediksi.

2. **Uji Peforma (Akurasi)**
   Pengujian ini untuk menguji seberapa akurat implementasi dari Hidden Markov Model untuk melakukan prediksi *state* atau kondisi selanjutnya baik secara *real-time* maupun dengan menggunakan dataset dari DARPA 2000.

5.2.1 Skenario Uji Fungsionalitas

Uji fungsionalitas dibagi menjadi 2 bagian yaitu uji funsgionalitas prediksi dan uji fungsionalitas IDS.

5.2.1.1 Uji Fungsionalitas Prediksi

Pengujian yang dilakukan meliputi mengetahui hasil perhitungan probabilitas secara *real-time* dan hasil prediksi serangan jaringan dari implementasi Hidden Markov Model.

5.2.1.2 Uji Fungsionalitas IDS

Pengujian yang dilakukan bertujuan untuk mengetahui apakah IDS dapat mendeteksi serangan injeksi sql dari penyerang dan mencatatnya pada InfluxDB. Serangan akan diklasifikasikan menjadi 4 kondisi yaitu *Normal, Attempt, Progress, Compromise*.

5.2.2 Skenario Uji Peforma (Akurasi)

Uji performa dibagi menjadi 2 skenario yaitu uji dari dataset tcp dump dan uji secara *real*-time.

5.2.2.1 Uji Performa LLDDoS 1.0

Skenario pertama dengan dataset dari LLDDoS 1.0 milik DARPA 2000 yang berupa tcp dump hasil dari penangkapan paket-paket yang masuk pada sebuah jaringan dengan melakukan serangan DDoS.

Terdapat 5 tahap dari skenario serangan DDoS dari DARPA 2000, yang dijelaskan dengan **Tabel 5.1** berikut :

Tabel 5. 1 Skenario Serangan DDoS dari DARPA 2000

Tahap	Nama	Keterangan Serangan	*State*
1	IP Sweep	Penyerangan mengirim ICMP *echo-request* dan menunggu ICMP *echo-reply* untuk menentukan server yang "up"	*Normal*
2	Sadmind Ping	Penyerangan mencari server yang memiliki celah dan memungkinkan untuk di eksploitasi.	*Attempt*
3	Break into	Penyerang mencoba mematahkan	*Attempt*

Tahap	Nama	Keterangan Serangan	*State*
		pertahanan dari server.	
4	Installation	Instalasi Trojan mstream DDoS	*Progress*
5	Launch	Melancarkan serangan DDoS	*Compromise*

5.2.2.2 Uji Performa *Real-Time*

Skenario yang kedua pengujian secara *real-time* dengan cara mendeteksi serangan injeksi sql pada server, kemudian IDS akan mencatat hasil deteksi pada *database*. Serangan akan dilakukan dengan menggunakan aplikasi sqlmap. Sqlmap melakukan penyerangan dengan parameter *risk* dan *level* yang berbeda. Terdapat 5 tingkatan *level* dan 3 tingkatan *risk*. Jika *risk* 1 dan *level* kurang dari 3 maka serangan termasuk dalam *state Attempt*, jika *risk* 2 atau 3 dan *level* serangan kurang dari 3 maka serangan termasuk dalam *state Progress*, dan berapapun *risk*-nya jika *level* lebih dari sama dengan 3 maka termasuk *state Compromise*. Hasil yang didapat akan dibandingkan dengan klasifikasi serangan dari IDS. Kemudian dihitung akurasi dari sistem yang telah dibangun.

5.3 Hasil Uji Coba dan Evaluasi

Berikut dijelaskan hasil uji coba dan evaluasi berdasarkan skenario yang sudah dijelaskan pada bab 5.2.

5.3.1 Uji Fungsionalitas

Berikut dijelaskan hasil pengujian fungsionalitas pada sistem yang sudah dibangun.

5.3.1.1 Uji Fungsionalitas Prediksi

Pengujian dilakukan dengan menguji fungsionalitas dari proses prediksi. Proses prediksi dapat melakukan perhitungan probabilitas secara *real-time*¸ dan memprediksi kondisi server dari serangan jaringan. Hasil uji coba tertera pada **Tabel 5.2**.

Tabel 5. 2 Hasil Uji Coba Fungsionalitas Prediksi

No	Fungsional	Uji Coba	Hasil
1	Mengetahui hasil perhitungan probabilitas secara *real-time*	Mengambil data baru disetiap sistem prediksi dijalankan untuk dihitung menjadi probabilitas yang *real-time*.	OK
2	Mengetahui prediksi state selanjutnya	Menjalankan algortima hidden markov model untuk memprediksi state selanjutnya	OK

Berikut adalah gambar saat sistem menghasilkan prediksi tentang kondisi *state* selanjutnya dari *server* untuk jumlah data tes 60 maka akan dihasilkan pula 60 data prediksi seperti pada **Gambar 5.2**.

```
thiar@karamel:~/intrusion_prediction$ python prediction2.py
[[ 0.033  0.033  0.     0.   ]
 [ 0.017  0.35   0.133  0.   ]
 [ 0.     0.     0.     0.133]
 [ 0.     0.     0.     0.133]]
[[ 0.083  0.     0.   ]
 [ 0.133  0.133  0.233]
 [ 0.017  0.     0.133]
 [ 0.133  0.133  0.   ]]
[[0 0 0 1 2 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 0 1 2
  0 1 2 2 0 0 0 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 1]]
Prediksi: Compromise, Progress, Progress, Normal, Attempt, Progress, Normal, Att
empt, Attempt, Progress, Normal, Attempt, Progress, Normal, Attempt, Attempt, Pr
ogress, Normal, Attempt, Progress, Normal, Attempt, Attempt, Progress, Normal, A
ttempt, Progress, Normal, Attempt, Attempt, Progress, Normal, Attempt, Progress,
 Progress, Normal, Attempt, Progress, Normal, Attempt, Attempt, Progress, Compro
mise, Progress, Progress, Normal, Attempt, Attempt, Progress, Normal, Attempt, P
rogress, Normal, Attempt, Attempt, Progress, Normal, Attempt, Progress, Normal
accuracy :  0.25
error rate :  0.75
sesitivity :  0.483870967742
```

Gambar 5. 3 Hasil Prediksi Menggunakan HMM

Probabilitas transisi dan emisi dihitung secara *real-time* karena HMM memiliki karakter dimana dia melakukan prediksi *state* selanjutnya berdasarkan catatan *state* sebelumnya. Untuk itulah probabilitas transisi dan emisi selalu dihitung ulang dengan data paling baru setiap sistem prediksi dijalankan. Hasilnya seperti pada **Gambar 5.4**.

```
thiar@karamel:~/intrusion_prediction$ python getProb.py
[[ 0.033  0.033  0.     0.   ]
 [ 0.017  0.35   0.133  0.   ]
 [ 0.     0.     0.     0.133]
 [ 0.     0.     0.     0.133]]
[[ 0.083  0.     0.   ]
 [ 0.133  0.133  0.233]
 [ 0.017  0.     0.133]
 [ 0.133  0.133  0.   ]]
[[0 0 0 1 2 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 0 1 2
  0 1 2 2 0 0 0 0 1 2 2 0 1 2 0 1 2 2 0 1 2 0 1]]
```

Gambar 5. 4 Hasil Perhitungan Probabilitas Secara Real-Time

5.3.1.2 Uji Fungsionalitas IDS

Pengujian dilakukan dengan menguji fungsionalitas dari IDS. IDS dapat mendeteksi serangan injeksi sql serta mencatatnya dalam database. Serangan akan diklasifikasikan menjadi 4 kondisi yaitu *Normal, Attempt, Progress, Compromise*. Hasil uji coba tertera pada **Tabel 5.3**.

Tabel 5. 3 Hasil Uji Coba Fungsionalitas IDS

No	Fungsional	Uji Coba	Hasil
1	Melakukan serangan injeksi sql ke sistem dan tercatat	Melakukan serangan injeksi sql	OK

IDS dapat mencatat dan mengelompokkan serangan injeksi sql yang dilancarkan oleh penyerang ke sistem. Kemudian hasil deteksi dicatat atau disimpan di database influxDB. Hasil pencatatan kondisi atau *state* dari server yang diserang seperti pada **Gambar 5.5.**

```
name: states_5
--------------
time                    jml_attack      state           state_num
1499142453793571774     0               normal          0
1499142513817585217     0               normal          0
1499142573826814979     665             attempt         1
1499142633839252517     10603           attempt         1
1499142693867973317     17966           progress        2
1499142753889233425     7130            compromised     3
1499142813902879705     10219           compromised     3
```

Gambar 5. 5 Pencatatan Data Serangan ke Sistem

5.3.2 Uji Performa (Akurasi)

Seperti yang telah dijelaskan pada bab 5.2 mengenai pengujian performa dilakukan dengan 2 skenario. Yang pertama dengan dataset dari LLDDoS 1.0 milik DARPA 2000 atau tidak *real-time* dengan jenis serangan DDoS. Yang kedua secara *real-time* dengan menggunakan 2 komputer virtual untuk melakukan penyerangan pada sistem.

5.3.2.1 Uji Performa LLDDoS 1.0

IDS membaca file tcp dump dari LLDDoS 1.0 milik DARPA 2000. Kemudian mendeteksi aktivitas-aktivitas yang

dilakukan selanjutnya dikategorikan dalam *state* tertentu sesuai dengan tingkat serangan yang lancarkan.
Hasil pengujian sesuai dengan **Tabel 5.4** sebagai berikut :

Tabel 5. 4 Hasil Uji Coba Dengan DARPA 2000

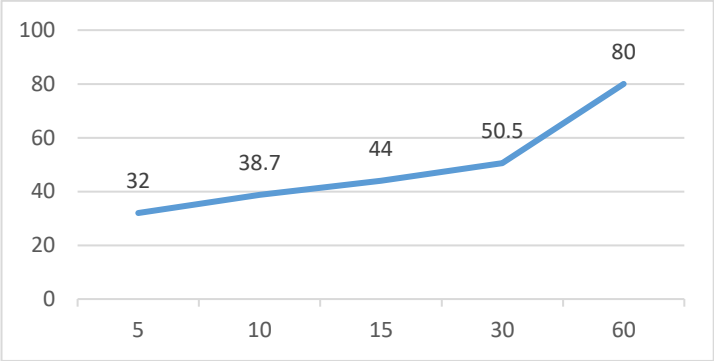
Jumlah Data Training	**Jumlah Data Tes**	**Akurasi (%)**	**Presentase Kesalahan (%)**	**Sensitifitas (%)**
60	60	80	20	70.5
60	30	50.5	49.5	50.7
60	15	44	56	49.6
60	10	38.7	61.3	40.2
60	5	32	68	29.7

Dari data pengujian, didapatkan bahwa akurasi untuk prediksi serangan DDoS dari DARPA 2000 dengan menggunakan Hidden Markov Model memiliki persentase akurasi mencapai 80% dengan jumlah data *training* 60 data dan jumlah data tes 60 data. Persentase akurasi paling rendah adalah 32% dengan jumlah data *traning* masih sama yaitu 60 data dan jumlah data tes 5 data.

Persentase kesalahan atau *error rate* mengalami peningkatan seiring dengan dikuranginya jumlah data tes. Presentase kesalahan paling kecil adalah 20% dengan data *training* berjumlah 60 data dan data tes berjumlah 60 data. Persentase kesalahan paling tinggi adalah 68% dengan jumlah data *training* 60 data dan data tes 5 data.

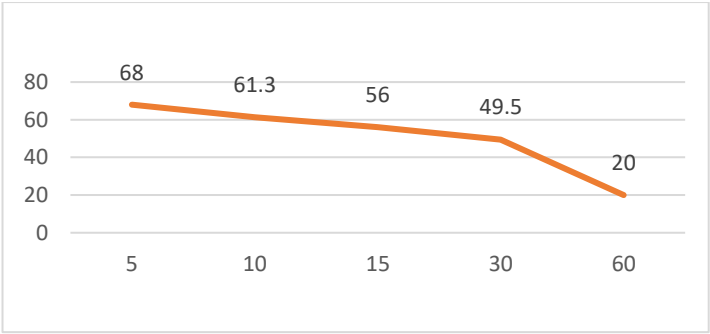
Untuk sensitifitas menunjukkan penurunan angka persentase seiring dengan dikuranginya jumlah data tes. Sensitifitas paling tinggi saat jumlah data *training* sebanyak 60 data dan data tes sebanyak 60 data adalah 70.5%. Sedangkan sensitifitas paling kecil ketika jumlah data tes menjadi 5 data namun dengan jumlah data *training* tetap berjumlah 60 data adalah 29.7%. Semakin lebar jarak perbandingan antara jumlah data training dan jumlah data tes mempengaruhi bertambahnya akurasi dan sesitifitas maupun mengurangi persentase kesalahan.

Grafik akurasi dari prediksi hidden markov model menunjukkan bahwa semakin besar jumlah data tes maka semakin bertambah pula persentase akurasi HMM untuk prediksi serangan DDoS. Grafik akurasi dapat dilihat pada **Gambar 5.6.**



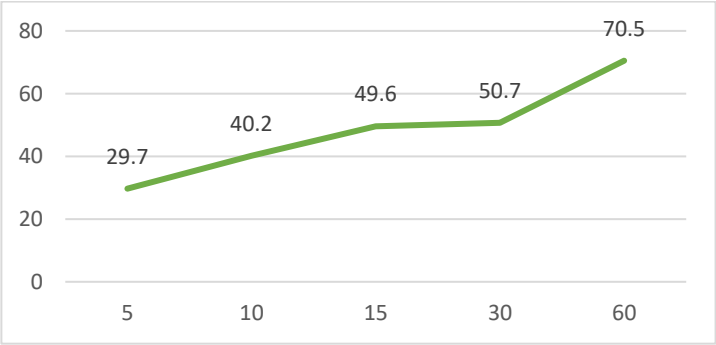
Gambar 5. 6 Gambar Akurasi HMM untuk serangan DDoS

Begitu pula pada grafik presentase kesalahan yang hasilnya berbanding lurus dengan akurasi untuk serangan DDoS, semakin menurun seiring dengan bertambahnya jumlah data tes. Grafik presentase kesalahan dapat dilihat pada **Gambar 5.7.**



Gambar 5. 7 Persentase Kesalahan HMM untuk serangan DDoS

Sensitifitas merupakan persentase dari hasil *True-Positive*. Dimana jika prediksi benar dan kenyataan juga benar. Dalam grafik ini senstifitas bergerak hamper searah dengan akurasi dari HMM namun dengan angka persentase yang berbeda. Grafik Sensitifitas dapat dilihat pada **Gambar 5.8.**



Gambar 5. 8 Grafik Sensitifitas HMM untuk Serangan DDoS

5.3.2.2 Uji Performa *Real-Time*

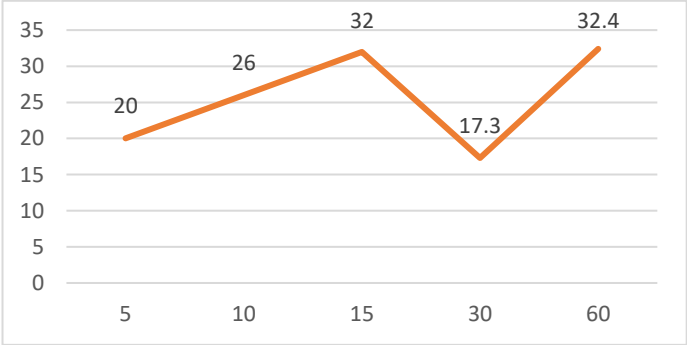
Penyerangan dilakukan dengan menggunakan aplikasi sqlmap sebegai alat untuk injeksi sql. Bentuk injeksi sql yang dikirim berupa request query melalui url.
Hasil pengujian sesuai dengan **Tabel 5.5** sebagai berikut :

Tabel 5. 5 Hasil Uji Coba Dengan 60 Data Training *Real-Time*

Jumlah Data Training	Jumlah Data Tes	Akurasi (%)	Presentase Kesalahan (%)	Sensitifitas (%)
60	60	32.4	49.5	59.5
60	30	17.3	82.7	48.3
60	15	32	68	65.5
60	10	26	74	53.7
60	5	20	80	36.7

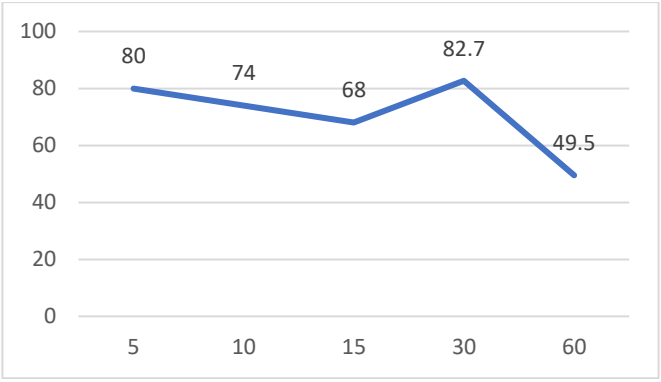
Dari data pengujian, didapatkan bahwa akurasi untuk prediksi serangan injeksi sql dengan menggunakan Hidden Markov Model memiliki persentase akurasi dibawah 50% . Semakin lebar jarak perbandingan antara jumlah data training dan jumlah data tes tidak mempengaruhi bertambahnya akurasi maupun mengurangi persentase kesalahan.

Grafik akurasi dari prediksi hidden markov model menunjukkan bahwa semakin besar jumlah data tes maka semakin bertambah pula persentase akurasi HMM untuk prediksi serangan injeksi sql. Namun ada saat dimana data lebih tinggi namun pada saat jumlah data tes sebanyak 30 mengalami penurunan angka akurasi. Grafik akurasi dapat dilihat pada **Gambar 5.9.**



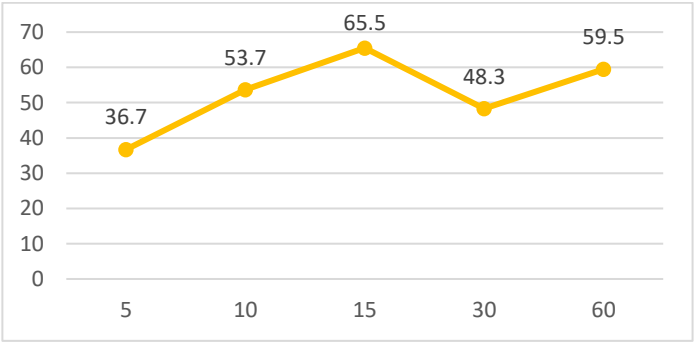
Gambar 5. 9 Grafik Akurasi HMM untuk SQLInjection

Begitu pula pada grafik presentase kesalahan, pada jumlah data tes dari 5 sampai 15 persetase kesalahan cenderung menurun, namun pada saat jumlah data tes menyentuh angkan 30 perserntase kesalahan meningkat dan kembali turun saat jumlah data tes berjumlah 60. Grafik presentase kesalahan dapat dilihat pada **Gambar 5.10.**



Gambar 5. 10 Grafik Presentase Keselahan HMM untuk SQLInjection

Sensitifitas merupakan persentase dari hasil *True-Positive*. Dimana jika prediksi benar dan kenyataan juga benar. Dalam grafik ini senstifitas bergerak hamper searah dengan akurasi dari HMM namun dengan angka persentase yang berbeda. Grafik Sensitifitas dapat dilihat pada **Gambar 5.11.**



Gambar 5. 11 Grafik Sensitifitas HMM untuk SQLInjection

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan pada bab sebelumnya, yaitu Bab Pengujian dan Evaluasi. Bab ini juga digunakan sebagai jawaban dari rumusan masalah yang dikemukakan pada Bab Pendahuluan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil ujicoba yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut :

1. HMM diimplementasikan dengan cara melakukan deteksi serangan terlebih dahulu, pada proses deteksi dilakukan juga pengelompokan untuk *state* dari serangan yang telah terdeteksi ke dalam *state Normal, Attempt, Progress, Compromise*. Kemudian disimpan dalam database time-series yaitu InfluxDB, setiap menitnya tercatat jumlah serangan, *state* serta nomor dari *state* hasil deteksi. Selanjutnya diambil data log dari InfluxDB untuk dibuat probabilitas transisi serta probabilitas emisi agar hasilnya *real-time*. Selanjutnya dilakukan proses prediksi dengan perhitungan Viterbi untuk menentukan *state* selanjutnya.
2. Dengan metode HMM yang telah diimplementasikan untuk memprediksi serangan diperoleh hasil seperti berikut:
    a. Dataset DARPA 2000 dengan jenis serangan DDoS diperoleh rata-rata akurasi sebesar 49.04%, rata-rata persentase kesalahan sebesar 46.96%, dan rata-rata persentase sensitifitas sebesar 48.14%.
    b. Serangan injeksi sql secara *real-time* diperoleh rata-rata akurasi sebesar 25.54%, rata-rata

persentase kesalahan sebesar 70.4%, dan rata-rata persentase sensitifitas sebesar 52.74%.

6.2 Saran

1. Sistem prediksi yang dibuat ini bisa dikembangkan menjadi sistem prevensi untuk melakukan pencegahan terhadap serangan yang akan datang, sesuai dengan tipe jenis serangannya.
2. Metode HMM lebih baik digunakan untuk prediksi serangan yang memiliki tahapan menyerangan seperti DDoS.

DAFTAR PUSTAKA

[1] A. S. Sendi, M. Dagenais dan M. Jabbarifar, "Real Time Intrusion Prediction based on Optimization Alerts with Hidden Markov Model," *Journal of Networks,* vol. 7, pp. 311-321, February 2012.

[2] W. Hu, W. Hu dan S. Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection," *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics,* vol. 38, no. 2, pp. 577-583, April 2008.

[3] S. S. Sindhu, S. Geetha, S. S. Sivanath dan D. A. Kannan, "A Neuro-genetic ensemble Short Term Forecasting Framework for Anomaly Intrusion Prediction," dalam *International Conference on Advanced Computing and Communication,* Mangalore, 2006.

[4] M. V. Eswara, "What is a simple explanation of the Hidden Markov Model algorithm?," Quora, 6 October 2015. [Online]. Available: https://www.quora.com/What-is-a-simple-explanation-of-the-Hidden-Markov-Model-algorithm. [Diakses 20 July 2016].

[5] K. Haslum, A. Abraham dan S. Knapskog, "DIPS: A Framework for Distributed Intrusion Prediction and Prevention Using Hidden Markov Models and Online Fuzzy Risk Assessment," dalam *Third International Symposium on Information Assurance and Security*, Montreal, Canada, 2008.

[6] S. N, "Hidden Markov Models," 2010. [Online]. Available: http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html. [Diakses Juli 2017].

[7] A. Irfani, A. Ratih dan D. Saptanti P, "Algoritma Viterbi dalam Metode Hidden Markov Models pada Teknologi".

[8] N. F. Almeida Jr, E. N. Caceres, C. E. R. Alves dan S. W. Song, "Comparison of Genomes using High-Performance Parallel Computing," dalam *Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing*, 2003.

[9] J. Clarke-Salt, "SQL Injection Attacks and Defense," dalam *First Edition: Winner of the Best Book Bejtlich Read Award*, Elsevier, 2009.

[10] ID-SIRTII/CC, "ID-SIRTII/CC," ID-SIRTII/CC, July 2015. [Online]. Available: http://www.idsirtii.or.id/bulanan/bulan/Juli/2015.html. [Diakses 29 June 2016].

[11] IDSIRTII/CC, "IDSIRTII/CC," IDSIRTII/CC, May 2015. [Online]. Available: http://www.idsirtii.or.id/bulanan/bulan/Mei/2015.html. [Diakses 29 June 2016].

[12] IDSIRTII/CC, "IDSIRTII/CC," IDSIRTII/CC, June 2015. [Online]. Available: http://www.idsirtii.or.id/bulanan/bulan/Juni/2015.html. [Diakses 29 June 2016].

[13] admin, "Python," [Online]. Available: https://www.python.org/doc/. [Diakses 10 Juni 2017].

[14] admin, "Docs hmmlearn," 2016. [Online]. Available: https://hmmlearn.readthedocs.io/en/latest/. [Diakses 10 Juni 2017].

[15] admin, "Numpy," 2017. [Online]. Available: http://www.numpy.org/. [Diakses 10 Juni 2017].

[16] admin, "Autralian Bureau of Statistics," 3 Juli 2013. [Online]. Available: http://www.abs.gov.au/websitedbs/a3121120.nsf/home/statistical+language+-+time+series+data. [Diakses 10 Juni 2017].

[17] admin, “InfluxDB,” 2017. [Online]. Available: https://www.influxdata.com/. [Diakses 10 Juni 2017].

LAMPIRAN

KODE SUMBER

1. Kode Sumber IDS bagian deteksi url vulnerable

Kode sumber berikut akan melakukan pengecekan jenis serangan yang terjadi menggunakan data *log* yang telah diperoleh sebelumnya sebagai kamus data serangan, sebagai acuan untuk menetukan tahapan serangan.

```
if(vulnerable_risk1.indexOf(JSON.stringify(req
uest.query))>-1)state1++;
if(vulnerable_risk2.indexOf(JSON.stringify(req
uest.query))>-1)state2++;
if(vulnerable_risk3.indexOf(JSON.stringify(req
uest.query))>-1)state2++;
if(JSON.stringify(request.query).indexOf("INSE
RT")>-1 ||
JSON.stringify(request.query).indexOf("UPDATE"
)>-1 ||
JSON.stringify(request.query).indexOf("DELETE"
)>-1)state3++;
total++;
 response.send('Hello from Express!')
})

setInterval(function(){

 normal = state0/total
 attempt = state1/total
 progress = state2/(2*total)
 compromised = state3/total
 console.log("normal = " + state0/total)
 console.log("risk1 = " + state1/total)
 console.log("risk2+risk3 = " +
state2/(2*total))
 console.log("compromised = " +state3/total)
```

Kode Sumber 1 Kode Sumber IDS bagian deteksi url vulnarable

2. Kode Sumber IDS pembagian state
   Kode sumber berikut akan menentukan tahapan serangan yang terjadi dalam satu menit.

```
state = "normal"
 if(compromised && progress>attempt){
  state="compromised"
 }
 else if(progress>attempt){
  state="progress"
 }
 else if(attempt>0){
  state="attempt"
 }
```

Kode Sumber 2 Kode Sumber IDS pembagian state

3. Kode Sumber IDS insert InfluxDB
   Kode sumber berikut akan menyimpan hasil penggolongan tahapan serangan yang terjadi kedalam InfluxDB.

```
client.write('states')
 .field({
     risk1:attempt,
     risk2_3:progress,
     state_num:state_num,
     state:state
  })
 .then(() => console.info('write point
success'))
 .catch(console.error)
```

Kode Sumber 3 Kode Sumber IDS insert InfluxDB

4. Kode Sumber HMM memperoleh data dari influxDB

Kode sumber berikut mengambil data dari InfluxDB yang kemudian akan digunakan sebagai parameter masukan algoritma HMM.

```
client = InfluxDBClient('localhost', 8086,
'admin', 'admin', 'IDS')
result = client.query('select * from states
where time > now() - 2h and time < now() -
1h;')
predict = client.query('select * from states
where time > now() - 1h;')
now = client.query('select * from states
where time > now() - 1m;')
array_result = list(result)[0]
array_predict = list(predict)[0]
array_now = list(now)[0]
```

Kode Sumber 4 Kode Sumber HMM memperoleh data dari influxDB

5. Kode Sumber HMM
   Kode sumber untuk mendapatkan probabilitas tahapan serangan sebagai masukan fungsi HMM.

```
ttransProb = np.zeros((4,4))
emissProb = np.zeros((4,3))

nn = 0
na = 0
nc = 0
an = 0
aa = 0
ap = 0
ac = 0
pn = 0
pa = 0
pp = 0
pc = 0
cn = 0
ca = 0
cc = 0
cp = 0
state = []
```

```
# data_predict = np.zeros((1,60), dtype=int)
data_predict = np.zeros((1,60),dtype=int)
numAttack = np.zeros((4,3))

for data in array_result:
      state.append(data['state_num'])

      if data['state_num'] == 0:
            if data['attack_number'] >= 0 and
data['attack_number'] <= 250:
                  numAttack[0,0] =
numAttack[0,0] + 1
            elif data['attack_number'] >= 251
and data['attack_number'] <= 1500:
                  numAttack[0,1] =
numAttack[0,1] + 1
            else:
                  numAttack[0,2] =
numAttack[0,2] + 1
      if data['state_num'] == 1:
            if data['attack_number'] >= 0 and
data['attack_number'] <= 250:
                  numAttack[1,0] =
numAttack[1,0] + 1
            elif data['attack_number'] >= 251
and data['attack_number'] <= 1500:
                  numAttack[1,1] =
numAttack[1,1] + 1
            else:
                  numAttack[1,2] =
numAttack[1,2] + 1
      if data['state_num'] == 2:
            if data['attack_number'] >= 0 and
data['attack_number'] <= 250:
                  numAttack[2,0] =
numAttack[2,0] + 1
            elif data['attack_number'] >= 251
and data['attack_number'] <= 1500:
```

```
                        numAttack[2,1] =
numAttack[2,1] + 1
                else:
                        numAttack[2,2] =
numAttack[2,2] + 1
        if data['state_num'] == 3:
                if data['attack_number'] >= 0 and
data['attack_number'] <= 250:
                        numAttack[3,0] =
numAttack[3,0] + 1
                elif data['attack_number'] >= 251
and data['attack_number'] <= 1500:
                        numAttack[3,1] =
numAttack[3,1] + 1
                else:
                        numAttack[3,2] =
numAttack[3,2] + 1
j = 0
for prd in array_predict:
        if prd['attack_number'] >= 0 and
prd['attack_number'] <= 250:
                data_predict[0,j] = 0
        elif prd['attack_number'] >= 251 and
prd['attack_number'] <= 1500:
                data_predict[0,j] = 1
        else:
                data_predict[0,j] = 2
        j = j + 1

#print numAttack
for i in range(0,len(state)-1):
        if state[i]==0 and state[i+1]==0:
                nn = nn + 1
        elif state[i]==0 and state[i+1]==1:
                na = na + 1
        elif state[i]==1 and state[i+1]==1:
                aa = aa + 1
        elif state[i]==1 and state[i+1]==2:
                ap = ap + 1
```

```
        elif state[i]==2 and state[i+1]==2:
                pp = pp + 1
        elif state[i]==2 and state[i+1]==3:
                pc = pc + 1
        elif state[i]==3 and state[i+1]==3:
                cc = cc + 1
        elif state[i]==3 and state[i+1]==2:
                cp = cp + 1
        elif state[i]==2 and state[i+1]==1:
                pa = pa + 1
        elif state[i]==1 and state[i+1]==0:
                an = an + 1

for i in range(0,4):
        for j in range(0,4):
                if i == 0:
                        if j == 0:
                                transProb[i,j] = nn /
float(len(array_result))
                        elif j == 1:
                                transProb[i,j] = na /
float(len(array_result))
                if i == 1:
                        if j == 0:
                                transProb[i,j] = an /
float(len(array_result))
                        elif j == 1:
                                transProb[i,j] = aa /
float(len(array_result))
                        elif j == 2:
                                transProb[i,j] = ap /
float(len(array_result))
                if i == 2:
                        if j == 1:
                                transProb[i,j] = pa /
float(len(array_result))
                        elif j == 2:
                                transProb[i,j] = pp /
float(len(array_result))
```

```
                    elif j == 3:
                          transProb[i,j] = pc /
float(len(array_result))
              if i == 3:
                    if j == 2:
                          transProb[i,j] = cp /
float(len(array_result))
                    elif j == 3:
                          transProb[i,j] = cc /
float(len(array_result))
              if j != 3:
                    emissProb[i,j] =
numAttack[i,j] / float(len(array_result))

np.set_printoptions(precision=3)
print transProb
print emissProb
```

Kode Sumber 5 Kode Sumber Mendapat Probabilitas Tahapan

6. Kode Sumber Pemanggilan Pustaka
   Kode sumber berikut akan melakukan pemanggilan pustaka HMM dan numpy.

```
from __future__ import division
import numpy as np
from hmmlearn import hmm
import getProb
```

Kode Sumber 6 Kode Sumber Pemanggilan Pustaka

7. Kode Sumber Prediksi HMM
   Kode sumber untuk melakukan prediksi terhadap tahapan yang akan terjadi selanjutnya.

```
states =
["Normal","Attempt","Progress","Compromise"
]
n_state = len(states)

observations = ["Low", "Medium", "High"]
n_observation = len(observations)
```

```
start_prob = np.zeros((1,4))
for now in getProb.array_now:
  if now['last'] == 0:
    start_prob[0,0] = 1
  elif now['last'] == 1:
    start_prob[0,1] = 1
  elif now['last'] == 2:
    start_prob[0,2] = 1
  else:
    start_prob[0,3] = 1

lvl = getProb.data_predict
lvl2 = lvl.T

model = model.fit(lvl2)
logprob, prediction = model.decode(lvl2,
algorithm="viterbi")
print "Prediksi:", ", ".join(map(lambda x:
states[x], prediction))
```

Kode Sumber 7 Kode Sumber Prediksi HMM

8. Kode Sumber Evaluasi Kinerja HMM

Evaluasi kinerja HMM diuji dengan menggunakan *confusion matrix*. Matrik tersebut akan menghasilkan data seperti *accuracy*, *error rate*, *sensitivity*, *false positife rate*, *specificity*, *precision*, *prevalence* yang digunakan sebagai parameter untuk mengetahui kinerja prediksi yang dilakukan oleh algoritma HMM.

```
i=0, tp = 0, tn = 0, fp = 0, fn = 0
for data in getProb.array_predict:
  if data['state_num'] == 3:
    if prediction[i] == data['state_num']:
      tn = tn + 1
    else:
      fn = fn + 1
  else:
    if prediction[i] == data['state_num']:
```

```
        tp = tp + 1
      else:
        fp = fp + 1
    i = i + 1
accuracy =
(tp+tn)/float(len(getProb.array_predict))
error_rate =
(fp+fn)/float(len(getProb.array_predict))
sesitivity = tp/float(fn+tp)
falsepost_rate = fp/float(tn+fp)
specificity = tn/float(tn+fp)
precision = tp/float(fp+tp)
prevalence                                    =
(fn+tp)/float(len(getProb.array_predict))
```

Kode Sumber 8 Kode Sumber Evaluasi Kinerja

9. Skema tabel penyimpanan data InfluxDB
   Skema yang digunakan sebagai tempat penyimpanan data pada basis data InfluxDB. Tabel tersebut kemudian akan digunakan untuk menyimpan data yang telah terdeteksi sebagai serangan setelah dibandingkan terhadap kamus data serangan.

Tabel 1 Skema Penyimpanan Data Influx DB

Nama kolom	Tipe data	Keterangan
Time	Timestamp	Waktu data masuk
Jml_attack	Integer	Jumlah serangan yang terjadi dalam satu menit
State	String	Tahapan serangan
State_num	Integer	Kode tahapan serangan

10. Contoh data
    Berikut merupakan contoh data yang disimpan pada basis data InfluxDB. Data diambil berdasarkan serangan yang terjadi setiap satu menit. Data yang disimpan telah digolongkan menjadi tahapan-tahapan yang sesuai berdasarkan pola serangannya.

Tabel 2 Contoh Data Log 1

Time	Jml_attack	State	State_num
1499142453793571774	0	Normal	0
1499142513817585217	665	normal	0
1499142573826814979	10603	Attempt	1
1499142633839252517	17966	Attempt	1
1499142693867973317	7130	Progress	2
1499142753889233425	10219	Compromised	3
1499142813902879705	22118	Compromised	3
1499142873921679099	17133	Attempt	1
1499142933947995413	665	Attempt	1
1499142993969680435	10508	Attempt	1
1499143053985278305	18061	Attempt	1
1499143114009520956	7130	Progress	2
1499143174026497050	10189	Compromised	3
1499143234039183971	21979	Compromised	3
1499143294056591477	17302	Attempt	1

INSTALASI

11. Instalasi InfluxDB

    Instalasi InfluxDB dilakukan sesuai dengan langkah-langkah berikut ini:

    - Tambahkan repository InfluxDB dengan menjalankan perintah:

```
curl -sL
https://repos.influxdata.com/in
fluxdb.key |
sudo apt-key add -
source /etc/lsb-release
echo "deb
https://repos.influxdata.com/${
DISTRIB_ID,,}
${DISTRIB_CODENAME} stable" |
sudo tee
```

```
/etc/apt/sources.list.d/influxd
b.list
```

- Setelah itu lakukan instalasi paket InfluxDB dengan menjalankan perintah:

```
# apt-get install influxdb
```

- Membuat *database* yang akan diguanakan untuk menyimpan data dengan perintah:

```
> create database IDS
> use IDS
> create series state
> show series
```

12. Instalasi NodeJS
    NodeJS digunakan sebagai IDS untuk mendeteksi serangan yang terjadi. NodeJS diinstal menggunakan langkah-langkah sebagai berikut:
    - Tambahkan perangkat lunak pendukung agar NodeJS dapat berjalan dengan baik dan benar. Perintah yang digunakan sebagai berikut:

```
apt-get install python-software-
properties
```

    - Tambahkan repository dari NodeJS agar instalasi dapat dilakukan. Perintah yang digunakan sebagai berikut:

```
curl -sL
https://deb.nodesource.com/setup_6
.x | sudo -E bash -
```

    - Lakukan instalasi paket NodeJS dengan melakukan instalasi dengan perintah:

```
apt-get install nodejs
```

13. Instalasi pustaka NodeJS
    Pustakan NodeJS yang digunakan untuk membangun sistem IDS antara lain:
    - Sebelum melakukan instalasi pustaka yang diperlukan, perlu diinstal paket manajer npm. Perintah yang digunakan adalah:
      ```
      apt-get install npm
      ```
    - ExpreesJS, pustaka tersebut digunakan sebagai *honeypot* untuk menangkap serangan yang terjadi. Langkah untuk melakukan instalasi digunakan perintah berikut:
      ```
      npm install express
      ```
    - InfluxDB-nodeJS, pustaka tersebut digunakan untuk membangun koneksi antara nodeJS dengan basis data InfluxDB. Perintah yang digunakan untuk melakukan instalasi adalah sebagai berikut:
      ```
      npm install influxdb-nodejs
      ```

14. Instalasi pustaka python
    Pustakan python dibutuhkan agar InfluxDB dan pustaka HMM dapat berjalan dengan baik. Pustakan yang harus diinstal antara lain:
    - Sebelum melakukan instalasi pustaka yang diperlukan, perlu diinstal paket manajer pip. Untuk melakukan instalasi digunakan perintah:
      ```
      python -pip
      ```
    - InfluxDB, jalankan perintah berikut untuk menginstall pustaka InsfluxDB.
      ```
      # menggunakan PIP
      pip install influxdb
      pip install –upgrade influxdb
      ```

```
# menggunakan apt
apt-get install python-influxdb
```

- Numpy, perintah yang digunakan untuk melakukan instalasi numpy adalah:

```
# menggunakan PIP
pip install numpy
pip install scipy
```

- HMM, pustakan tersebut digunakan agar algoritma hmm dapat digunakan dengan baik. Perintah yang digunakan untuk melakukan instalasi pustaka tersebut adalah:

```
pip install scikit-learn
git clone
https://github.com/hmmlearn/hmm
learn.git
python setup.py install
```

BIODATA PENULIS

Kharisma Nur Annisa dilahirkan di Tulungagung pada tanggal 31 Mei 1995. Penulis menempuh pendidikan mulai dari SD Islam Al-Munawwar (2001-2007), SMPN 1 Tulungagung (2007-2010), SMAN 1 Kedungwaru (2010-20013), dan S1 Teknik Informatika ITS (2013-2017).

Selama kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika ITS (HMTC). Diantaranya adalah sebagai Staff Departemen Kesejahteraan Mahasiswa (2014-2015) dan menjadi Sekretaris Departemen Kesejahteraan Mahasiswa (2015-2016). Selain itu, penulis ikut aktif dalam kegiatan kepanitiaan Schematics dengan menjadi staff National Logic Competition pada Schematics 2014 dan Schematics 2015. Penulis juga pernah menjadi Staff Departemen Bismika (2014-2015) dan Staff Ahli Departemen An-Nisa' (2015-2016) dari Keluarga Muslim Informatika (KMI). Penulis juga merupakan seorang administrator di Laboratorium Arsitektur dan Jaringan Komputer dan pernah menjadi asisten mata kuliah Sistem Operasi dan Jaringan Komputer serta coordinator asisten untuk mata kuliah Sistem Operasi.

Kritik dan saran sangat diharapkan guna peningkatan kualitas dan penulisan selanjutnya. Untuk itu, silahkan kirim kritik dan saran ke : **kharismana31@gmail.com**