MENINGKATKAN KEMAMPUAN DARI *SELF-ADAPTIF* SSH *HONEYPOT* UNTUK BERINTERAKSI DENGAN *ATTACKER*

TUGAS BESAR
KEAMANAN JARINGAN

Oleh
BAIK BUDI
NIM : 23214338

dosen

Ir. BUDI RAHARDJO, MSc., PhD.

PROGRAM STUDI TEKNIK TELEMATIKA DAN
JARINGAN TELEKOMUNIKASI
INSTITUT TEKNOLOGI BANDUNG
2015

ABSTRAK

MENINGKATKAN KEMAMPUAN DARI *SELF-ADAPTIF* SSH *HONEYPOT* UNTUK BERINTERAKSI DENGAN *ATTACKER*

oleh
Baik Budi
NIM: 23214338
(Program Studi Magister Teknik Telematika
dan Jaringan Telekomunikasi)

Penggunaan system computer dan teknologi internet yang sangat besar telah membuat keamanan informasi mendapat perhatian yang sangat besar pada saat sekarang ini. Salah satu target dari *Attacker* yang paling menonjol saat ini adalah *server* yang di dalamnya telah diset up *remote access service*, contohnya *Secure Shell* (SSH). Mengindikasikan dan mengklasifikasikan serangan ini serta aktor dibalik serangan tersebut merupakan langkah yang sangat penting dalam mengembangkan sebuah kebijakan keamanan dan juga merupakan sebuah strategi yang efektif dalam mempertahankan diri dari serangan *attacker* tersebut. *Honeypots*, merupakan mekanisme baru dalam keamanan, membantu dalam memonitor dan mempelajari serangan. *Honeypots,* dapat memikat pengguna jahat dengan cara bertindak sebagai sistem yang mengandung data yang berharga atau layanan yang menarik. Saat ini, para peneliti mencoba meningkatkan kemampuan *honeypot* dengan mengintegrasikannya dengan kemampuan *self-adaptif* atau kecerdasan buatan. Pada makalah ini akan dibahas peningkatan kemampuan dari *self-adaptif honeypot* dengan memperbaiki kekurangan dari *Heliza* dengan menggunakan acuan dari *honeypot Kippo* dan fungsi dari *Pybrain Library. Honeypot* ini dibuat dengan menirukan layanan SSH *server.* Dari hasil penelitian didapatkan bahwa *reinforcement learning* pada *self-adaptif* SSH *honeypot* ini dapat meningkatkan skalabilitas, kemampuan dalam menentukan asal tempat *attacker* berdasarkan IP, dan dengan penggunaan *machine learning* yang dikembangkan dengan *python library* dapat membuat *adaptif honeypot* yang memiliki kemampuan tinggi untuk dapat berinteraksi dengan *attacker.*

Kata Kunci: *Honeypot,* SSH, *Reinforcement Learning, Machine Learning*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

1. Pendahuluan

Pada era global seperti sekarang ini, keamanan sistem informasi berbasis Internet menjadi suatu keharusan untuk lebih diperhatikan, karena jaringan internet yang sifatnya publik dan global pada dasarnya tidak aman. Perlu kita sadari bahwa untuk mencapai suatu keamanan itu adalah suatu hal yang sangat sulit, seperti yang ada dalam dunia nyata sekarang ini. Tidak ada satu daerah pun yang betul-betul aman kondisinya, walau penjaga keamanan telah ditempatkan di daerah tersebut, begitu juga dengan keamanan sistem komputer. Namun yang bisa kita lakukan adalah untuk mengurangi gangguan keamanan tersebut.

Penggunaan system komputer dan teknologi internet yang sangat besar telah membuat keamanan informasi mendapat perhatian yang sangat besar pada saat sekarang ini. *Attacker* menggunakan internet untuk mencari *server* atau target yang akan mereka jadikan korban kejahatan mereka [1]. Motivasi dibalik serangan terhadap sebuah system ini bervariasi, mulai dari tindakan sederhana dari perusak cyber, melakukan spionase pada sebuah perusahaan untuk mencari keuntungan berupa finansial atau sebuah serangan yang berdampak pada *geopolitics* sebuah negara.

Salah satu target dari *Attacker* yang paling menonjol adalah *server* yang didalamnya telah diset up *remote access service*, contohnya *Secure Shell* (SSH). Telah banyak *server* yang diserang oleh pengguna yang berniat jahat jika didalam mekanisme otentikasinya terdapat *password* yang lemah. Seriap kali seorang *attacker* menemukan calon korban, yaitu sebuah server yang menjalankan layanan tertentu, ia akan mencoba untuk terhubung ke server tersebut dengan menggunakan berbagai kombinasi dari autentifikasi [1]. Jika *attacker* tersebut berhasil login, maka ia akan memperoleh akses remote ke server dan kemudian menggunakannya untuk kegiatan yang berbahaya, seperti menginstal *malware,* dan bahkan *attacker* tersebut juga bisa menggunakan server yang telah dikuasainya itu untuk menyerang system yang lain.

Saat ini untuk mengamankan jaringan komputer digunakan *Intrusion Detection System* (IDS) [2]. Namun IDS ini memiliki kelemahan, yaitu tidak mampu dalam menangani serangan baru yang belum diketahui sebelumnya. Laporan terbaru dari kasus *SSH Brute Force Attack* adalah serangan dengan tujuan memperoleh *credentials* dan *critical information,* seperti akses *administrator* dari target, dan gambaran sistem yang

digunakannya yang kemudian semua informasi tersebut akan di internet melalui *underground website* [3].

Honeypots, merupakan mekanisme baru dalam keamanan, membantu dalam memonitor dan mempelajari serangan. *Honeypots* adalah alat yang digunakan menjebak *Attacker. Honeypots,* dapat memikat pengguna jahat dengan cara bertindak sebagai sistem yang mengandung data yang berharga atau layanan yang menarik [1]. *Honeypots* memungkinkan untuk dieksploitasi oleh *Attacker.* Hal inilah yang kemudian dapat membantu pakar keamanan profesional dan peneliti dalam proses pembelajaran terhadap teknik dan metode yang dilakukan oleh para *attacker. Honeypots* tidak bisa mencegah serangan *cyber* terhadap jaringan sendiri, tetapi mereka dapat membantu dalam mengidentifikasi dan melakukan deteksi terhadap serangan ketika mereka digunakan bersama dengan perangkat pertahanan lainnya seperti *firewall. Honeypot* dapat menghasilkan sejumlah data yang memiliki nilai yang tinggi dan dapat juga menjadi tantangan bagi para pakar keamanan profesional.

Saat ini para peneliti mencoba untuk meningkatkan kemampuan dari *honeypot,* salah satunya adalah dengan mengintegrasikannya dengan kemampuan *self-adaptive*, dengan artian teknik kecerdasan buatan. Salah satu sistem yang paling baru dikembangkan adalah *Heliza,* yaitu sistem *honeypot* yang mampu berintegrasi dengan para *attacker* dengan memanfaatkan *machine learning technique* [4]. *Heliza* dikembangkan oleh G. Wagner, R. State, A. Dalaunoy, dan T. Engeland. *Heliza* di implementasikan dengan menggunakan kernel *User Mode Linux* dan *Python* yang telah dimodifikasi dan diimplementasikan pada *machine learning algorithm.* Implementasi dari sistem ini memiliki beberapa kekurangan tertentu, seperti kompleksitas penyebaran sistem UML, dan juga kerugian berupa penggunaan sistem operasi secara penuh yang digunakan sebagai target yang akan diserang oleh penyerang. Dalam makalah ini, akan disajikan implementasi baru dari sistem *adaptive honeypot* yang mengikuti pendekatan yang digunakan oleh *Heliza* dengan mengatasi beberapa kekurangannya. Sistem yang akan dikembangkan ini menggunakan acuan kode dari sistem *honeypot Kippo* dan fungsi dari *Pybrain Library* serta *honeypot* ini dibuat dengan menirukan layanan SSH *server.*

Salah satu pendekatan yang digunakan untuk untuk membuat *self-adaptive honeypot* adalah berdasarkan tingkat adaptasinya. Sistem *self-adaptive honeypot* ini pada dasarnya memanfaatkan teknik kecerdasan buatan *(Artificial Inteligence),* seperti *reinforcement learning, game theory* atau *Case Based Reasoning* untuk berinteraksi dengan *attacker.*

2. Tinjauan Pustaka

2.1 Pengenalan *Honeypot*

Sistem *honeypot* merupakan sistem komputer yang digunakan untuk memikat atau mengundang *attacker* dengan ruang lingkup untuk mendeteksi, dan memantau tindakan yang mereka lakukan [5]. Sistem *honeypot* sengaja dibuat untuk diserang oleh *attacker.* Setelah sistem tersebut diserang oleh *attacker,* kita bisa melihat bagaimana teknik yang digunakan oleh *attacker* tersebut untuk menyerang, dan juga kita bisa mengetahui penyebab *attacker* menyerang sistem tersebut. Pada saat yang sama, *honeypot* juga dapat digunakan untuk menguping kegiatan *hacker,* mengumpulkan *tools* yang digunakan oleh *hacker,* dan mengetahui hubungan dari jaringan *master hacker* [6].

Sumber dari teknologi *honeypot* dapat berupa salinan dari sistem operasi [6]. Sistem *honeypot* terbuat dari komputer atau *server* biasa atau segmen jaringan terisolasi yang membuat *attackers* berpikir itu adalah bagian dari jaringan yang sah *(legitimate network)* yang didalamnya terdapat data yang berharga bagi mereka para *attackers* [5]. Sistem *honeypot* ini bertujuan untuk membangun lingkungan penipuan, menginduksi serangan *attacker,* memeriksa semua hal yang dilakukan oleh *attacker,* dan pada saat yang sama merekam dan menghasilkan laporan yang kemudian akan dipelajari dan dianalisis untuk mengetahui metode dan tujuan dari *attacker* tersebut.

2.2 Klasifikasi *Honeypot*

2.2.1 Berdasarkan lokasi [5]:

1. *Production Honeypot*

   Production Honeypot digunakan terutama oleh perusahaan besar dan sistem ini ditempatkan dibagian dalam jaringan organisasi mereka.

2. *Research Honeypot*

   Honeypot ini dikembangkan oleh organisasi yang bertujuan untuk mengumpulkan informasi sebanyak mungkin tentang alasan dan metode terkini yang digunakan oleh *hacker* dalam melakukan serangan.

2.2.2 Berdasarkan fungsi dasarnya [5]

1. *Monitor Honeypot*

   Honeypot yang mendengarkan *port* jaringan dan memberikan jawaban pada jaringan.

2. *Emulator Honeypot*

   Honeypot yang berinteraksi dengan penyerang pada *real system.* Misalnya meniru fungsi server *e-mail*.

3. *Simulator Honeypot*

   Honeypot yang mensimulasikan berbagai sistem operasi dengan mensimulasikan *banner* mereka, *TCP Stack,* dan lainnya.

4. *Detector Honeypot*

   Honeypot yang digunakan dalam hubungannya dengan *Intrusion Detection System* (IDS).

2.2.3 Berdasarkan level interaksinya [7]:

1. *Low Interaction*

   Sistem ini hanya mengekspos layanan spesifik seperti FTP, SSH, HTTP dan oleh karena itulah *attacker* tidak berinteraksi dengan sistem operasi dari *host.*

2. *Medium Interaction*

   Sistem ini hanya mengekspos layanan spesifik tapi dengan tingkat level tertentu memberikan akses ke dasar sistem operasi, seperti contoh layanan SSH yang memberikan akses untuk *emulated file system.*

3. *High Interaction*

   Sistem yang secara penuh mengekspos sistem operasi, dimana menawarkan sepenuhnya dirinya untuk di serang

2.2.4 Berdasarkan tingkat adaptasinya:

1. *Static*

   Honeypot yang selalu mempunyai konfigurasi yang sama dan mengekspos tampilan yang sama pada *attacker.*

2. *Dynamic*

   Honeypot yang akan mengubah tampilannya secara otomatis berdasarkan tingkat serangan yang dihadapi atau berdasarkan konfigursi jaringan saat ini yang mengelilinginya [7].

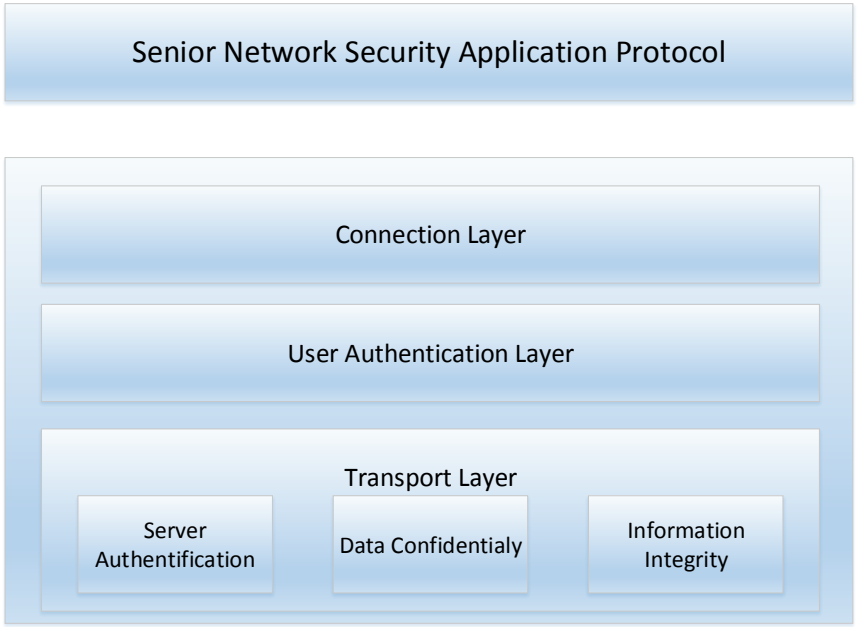
2.3 Secure Shell (SSH)

Secure Shell merupakan mekanisme *remote connection* yang dieknkripsi, yang umumnya digunakan pada sistem operasi berbasis *Linux* dan *Unix* Protolol SSH ini idefenisikan oleh Ylonen dan Lonvick pada *Internet Engineering Task Force's* RFC 4254 9 [8]. SSH-1 dirancang oleh Tatu Ylonen, seorang peneliti di *Helsinki University of Technology,* Finlandia pada tahun 1995. Dasar dari pembuatan protokol SSH ini adalah karena peristiwa pembongkaran sandi di jaringan universitas. Tujuan pembuatan SSH adalah untuk menggantikan fungsi *rlogin, Telnet,* dan *rsh* protokol yang tidak memberikan otentifikasi kuat dan menjamin kerahasiaan. SSH-2 dirancang pada tahun 1996, dengan peningkatan fitur keamanan dan peningkatan dari SSH-1 [9].

SSH protokol sangat handal dan dirancang untuk memberikan sebuah protokol keamanan untuk sesi *remote login,* dan layanan lainnya. Protokol SSH efektif dalam mencegah kebocoran informasi pada proses *remote management.* Enkripsi semua data yang dikirim melalui SSH, membuat serangan seperti *"middleman"* susah untuk mencapainya, dan SSH juga dapat mencegah *Spoofing* DNS dan IP. Keuntungan lain dari penggunaan SSH ini adalah dapat meningkatkan kecepatan data transfer karena data yang dikirimkan dikompres [8]. SSH memiliki banyak fungsi, ia dapat menggantikan *Telnet,* dan menyediakan saluran yang aman untuk FTP atau POP, dan juga untuk PPP [8].

Bagian utama dari *framework* SSH protokol adalah terdiri dari tiga layer, yaitu *Connection Layer, User Authentifiction Layer,* dan

Transport Layer [9], seperti yang ditunjukkan pada gambar 2.1 dibawah ini.



Gambar 2.1 SSH *Basic Framework* [8]

1. *Connection Layer*

   Pada lapisan ini menyediakan fungsi otentikasi server, enkripsi data, dan keaslian dan integritas data, serta ada pilihan lainnya seperti kompresi.
2. *User Authentication Layer*

   Pada lapisan ini menangani otentikasi *client* dan sejumlah otentikasi lainnya.
3. *Connect Layer*

   Lapisan ini membagi sejumlah *logical channel* pada *channel* terenkripsi.

2.4 *Reinforcement Learning* (RL)

Reinforcement Learning adalah salah satu paradigma baru di dalam *learning theory*. RL dibangun dari proses pemetaan *(mapping)* dari situasi yang ada di lingkungan *(states)* ke bentuk aksi *(behavior)* agar dapat memaksimalkan keuntungan *(reward)* [10]. Agen yang bertindak sebagai *learner* tidak perlu diberitahukan tindakan apakah yang akan seharus nya dilakukan, atau dengan kata lain, membiarkan *learner* belajar sendiri dari pengalamannya. Ketika *learner* melakukan sesuatu yang benar berdasarkan aturan yang kita

tentukan, *learner* tersebut akan mendapatkan *reward*, dan begitu juga sebaliknya. RL secara umum terdiri dari 4 komponen dasar, yaitu [10]:

1. Kebijakan *(Policy)*

   Berfungsi untuk membuat keputusan dari *learner* yang menspesifikasikan tindakan apakah yang mungkin dilakukan dalam berbagai situasi yang ditemukan. *Policy* inilah yang bertugas memetakan situasi lingkungan ke dalam bentuk aksi.

2. *Reward function*

   Mendefinisikan tujuan dari kasus atau masalah yang dihadapi. *Reward function* juga yang menentukan *reward* atau *punishment* yang diterima *learner* saat berinteraksi dengan lingkungan.

3. Fungsi Nilai *(Value function)*

   Menspesifikasikan fungsi akumulasi dari total *reward* yang didapatkan oleh *learner*.

4. Model Lingkungan *(Model of environment)*

   Sesuatu yang menggambarkan aksi dari lingkungan. *Model of environment* ini sangat berguna untuk mendesain dan merencanakan tindakan yang tepat pada situasi mendatang yang memungkinkan, sebelum *learner* mempunyai pengalaman dengan situasi tersebut.

Salah satu keunggulan RL dibandingkan teori-teori *learning* yang lain adalah kemampuannya dalam mengadopsi proses *exploitation* dan *exploration* yang memang biasanya dilakukan oleh manusia. *Exploitation* dan *exploration* inilah yang menjadi kunci keberhasilan proses *learning* dari RL. Dalam hal ini seimbang tidak berarti sama, atau dengan kata lain perbandingannya 50:50, akan tetapi persentase keduanya akan berfluktuasi sesuai dengan berbagai macam keadaan yang jelas dan beragam.

2.5 *Heliza-self-adaptif honeypot system*

Heliza merupakan *honeypot* yang memiliki kemampuan beradaptasi dan berinterasksi dengan *attacker* menggunakan *reinforcement learnin*. Heliza menggunakan teknik *machine learning* dan *game theory.* Heliza mengumpulkan sejumlah informasi ketika berinteraksi dengan penyerang. Heliza dapat secara permanen

meningkatkan kemampuan, perilaku, dan mengevaluasi beberapa strategi untuk berinteraksi dengan penyerang. Heliza diimplementasikan pada *server* SSH berkonfigurasi rendah. Heliza memiliki kemampuan yang cepat dalam belajar dan beradaptasi. Heliza dioperasikan berdasarkan proses keputusan markov *(markovian decision process).*

2.6 *Kippo*

Kippo adalah sistem *adaptif honeypot* kelas menengah, dikembangkan dengan Python yang mencatat semua *brute force attack* pada SSH dan mempunyai kemampuan untuk meniru setiap perintah yang dilakukan oleh penyerang [11]. *Honeypot* ini banyak digunakan oleh organisasi yang aktif dalam bidang keamanan *cyber* seperti, CERT, vendor antivirus, dan universitas. Sistem ini mampu mengumpulkan informasi yang berguna bagi pakar keamanan *cyber,* karena mempunyai beberapa fitur yang sangat menarik seperti:

1. Fitur *Debian Shell* palsu]
2. File sistem palsu
3. Perintah yang menawarkan untuk mendownload seperti *wget.*

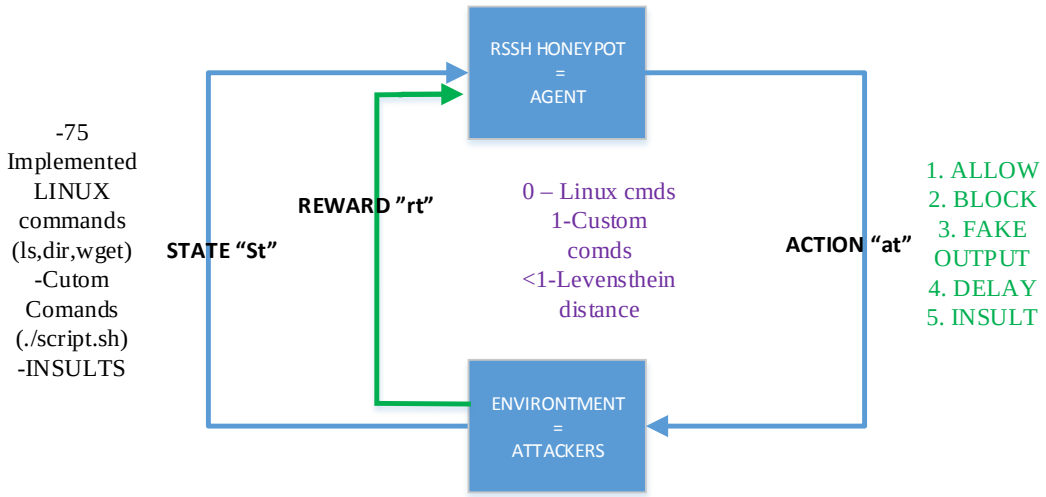
Semua informasi dari *honeypot* dicatat dan disimpan dalam *database MySQL,* sehingga nantinya informs tersebut dapat dibuka kembali untuk dianalisa dengan menggunakan *tools* seperti *playlog.py*.

2.7 *Pybrain*

Pybrain adalah *machine learning library* untuk *python* [12]. *Pybrain* sangat fleksibel, dan memiliki kemampuan algoritma yang kuat untuk *machine learning.* *Pybrain library* memiliki ketergantungan dengan *SciPy library* dan diimplementasikan pada algoritma pembelajaran mulai dari *supervised learning* sampai ke *reinforcement learning*. Salah satu fitur yang menarik dari. Salah satu fitur yang menarik dari *pybrain* adalah kemampuan untuk menghubungkan berbagai jenis arsitektur dan algoritma.

3. *Reinforced* SSH *Honeypot*

Reinforcement Leaning (RL) adalah bagian dari *machine learning* yang mencoba untuk memecahkan masalah yang dihadapi agen atau *learner* ketika harus belajar berprilaku dengan cara berinteraksi dengan lingkungan yang dinamis dengan menggunakan mekanisme *trial-and-error*. Sebuah model *reinforcement learning* dapat digambarkan seperti gambar 3.1 dibawah ini



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha r_t + 1 + \gamma Q(s_t + 1, a_t + 1) - Q(s_t, a_t)$$

Gambar 3.1 Model *Reinforcement Learning* [14]

Pada gambar 3.1 diatas dapat dilihat bahwa agen *(learner)* terhubung dengan lingkungan oleh persepsi dan aksi. Adapun pada setiap langkah agen [13]:

1. Menerima beberapa masukan, yang dinotasikan dengan *i,* dimana memberikan rincian tentang keadaan sekarang dari linkungan.
2. Memilih suatu tindakan *a,* dimana akan merubah keadaan dari lingkungan.
3. Menghitung nilai dari transisi keadaan dengan menghasilkan sinyal penguatan *(reinforcement)* yang dinotasikan dengan *r.*

Tujuan dari keseluruhan proses ini adalah untuk menemukan kebijakan *S,* yang memetakan semua keadaan untuk suatu tindakan, dan memaksimalkan pengukuran jangka panjang dari *reinforcement.* Lingkungan dapat bersifat *nondeterministic,* yang berarti bahwa tindakan yang sama dalam suatu keadaan yang sama tapi memiliki langkah dan prosedur yang berbeda dapat menghasilkan *reinforcement* yang berbeda.

3.1 Lingkungan *(environment)*

Dalam *reinforcement learning honeypot* ini para *attacker* yang mengelola dan berinteraksi dengan *honeypot* disebut dengan *environment.* Setelah para penyerang berinteraksi dengan SSH *server,* penyerang tersebut akan mengeluarkan urutan perintah yang dimulai dengan *starts* dengan *sshd,* dan diakhiri dengan perintah *exit.* Urutan perintah tersebut dinotasikan dengan *string* $i_0 i_1 i_2 \dots i_n$, dimana $i_n =\in S^*$. *String* i_n akan masuk dalam salah satu kategori :

- Linux *commands* yang diimplementasikan dalam aplikasi $L = \{S_0 S_1 S_2 \dots S_n\}$;
- *Customized tool* yang dikembangkan atau didownload dari *local system:* $C =\in S^*$;
- *Insults* – Perintah yang tidak termasuk dalam kategori diatas dianggap perintah ENTER
- $I = S^* - L - \{empty\} - C$

Lingkungan dimodelkan dengan Rantai Markov dengan asosiasi proses *reward* pada *state* yang terbatas. Oleh karena itu pada impolementasi ini, dibatasi dengan 75 *commands*: $S = \{S_1 S_2 S_3 \dots S_{75}\}$. Dalam pemodelan lingkungn ini, *insults* juga dianggap sebagai bagian dari lingkungan, dengan perintah yang telah di *custom* dimana:

$$S = L > \cup \{insults, custom, empty\}$$

Akhir dari *state* adalah *absorbing state*, karena dalam keadaan akhir ini, penyerang telah meninggalkan *honeypots*.

3.2 *Action*

RASSH menerapkan serangkaian tindakan ketika *honeypot* berinteraksi dengan penyerang. Adapun tindakan tersebut adalah sebagai berikut [13]:

1. *Allow Action*

   Memiliki fungsi dan implementasi yang sama dengan *Heliza,* memungkinkan eksekusi dari perintah yang dimasukkan oleh penyerang.

2. *Block Action*

   Memiliki fungsi yang sama dengan *Heliza* (memblokir eksekusi perintah), tetapi memili implementasi berbeda, dimana untuk setiap perintah tertentu, memiliki pesan pemblokiran tertentu.

3. *Fake Output Action*

   Memiliki fungsi yang sama dengan *Heliza* (memalsukan keluaran dari perimtah), tetapi memiliki implementasi berbeda dimana untuk setiap perintah tertentu akan dipalsukan.

4. *Insult Action*

   Memiliki fungsi yang sama dengan *Heliza,* tetapi dalam pelaksanaannya berbeda dalam arti setiap penyerang mendapat *geo-lokal* dan *insult message* dalam bahasa *correspondent.*

5. *Delay Action*

   Merupakan aksi yang baru diimplementasikan yang mempunyai target untuk menunda eksekusi perintah dari penyerang. Dengan utilitas ini berada dalam kenyataan bahwa penyerang mungkin mempertimbangkan sistem menjadi lelah dan mencoba untuk menggunakan *tool* lain yang sedikit menggunakan sumber daya.

3.3 *Rewards*

Heliza mempunyai dua fungsi *reward*, sesuai dengan tujuan yang ingin dicapai, yaitu:

1. Mengalihkan *attacker* untuk men-*download custom software.*
2. Membuat *attacker* sebanyak mungkin berada pada *honeypot.*

Dengan menambahkan *action* ke lima, yaitu *delay function,* maka fungsi *reward* ke dua ini menjadi tidak relevan. Oleh karena itu dalam implementasi RASSH ini, digunakan fungsi *reward* "mengumpulkan informasi yang terkait dengan *attacker*".

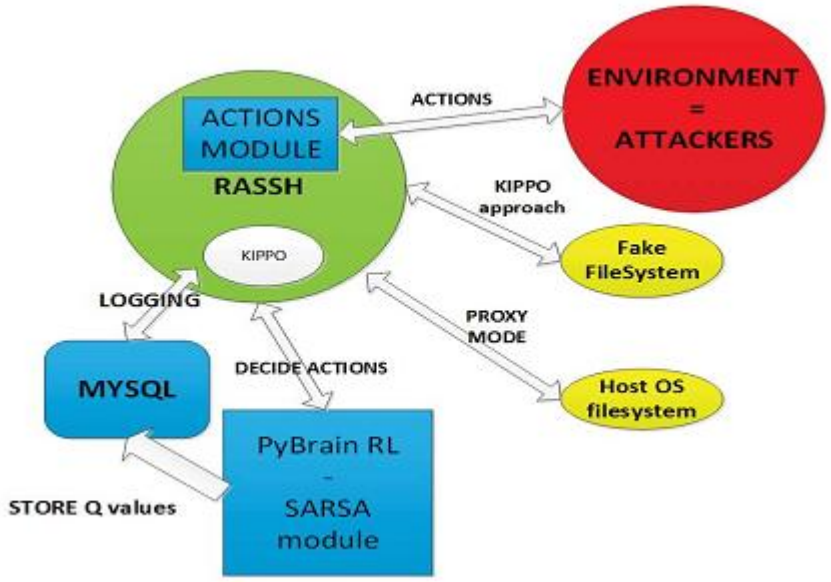
3.4 *Learning Agent*

Pada dasarnya *adaptif honeypot* bertujuan untuk menemukan kebijakan dari setiap kejadian yang diberikan. Jenis pembelajaran atau *learning* ini berada dalam kategori *on-policy.* Setiap serangan dari penyerang dimulai dengan perintah *sshd,* dan berakhir dengan perintah *exit,* metoda pembelajaran ini dapat dipecah menjadi

beberapa episode, dimana setiap kebijakan akan dievaluasi di akhir setiap episode. Pada implementasinya, digunakan algoritma *State Action Reward Action* (SARSA), karena algoritma ini merupakan metode sederhana untuk pembelajaran berbasis kebijakan *(on-policy learning)*.

4. Arsitektur RASHH *Honeypot*

Reinforced Adaptif SSH honeypot yang dirancang ini merupakan pengembangan dari *Kippo,* yang diimplementasikan dengan *Python* yang memanfaatkan metoda *reinforcement learning.* Adapun arsitektur dari RASSH dapat dilihat pada gambar 3.2 dibawah ini:



Gambar 3.2 Arsitektur RASSH [13]

4.1 Modul *Honeypot*

Modul *honeypot* ini merupakan implementasi dari kode *Kippo* yang telah ditingkatkan kemampuannya dengan menambahkan bebrapa perintah baru, seperti *useradd, userdell,* dan lainnya. Selain itu juga ada perintah *ifconfig, wget, ping,* dan perintah lainnya. Dalam *honeypot* ini juga diimplementasikan suatu pendekatan baru yang dinamakan *proxu mode.* Dalam *proxy mode* ini, modul *honeypot* bertindak sebagai *proxy* antara sistem operasi (Linux) dengan *shell* Linux yang dapat diakses oileh penyerang. Dengan cara ini, sebelum penyerang mengeksekusi perintah pada tingkat sistem

operasi, maka modul *honeypot* akan menunggu aksi yang akan diputuskan oleh modul RL.

4.2 Modul *Action*

Modul ini sebagai penengah antara modul *honeypot* dengan modul RL. Modul ini berfungsi untuk melakukan *action* atau tindakan yang dilakukan ketika *attacker* menyerang modul *honeypot.* Tindakan yang dilakukan dipicu dari keputusan yang diberikan oleh modul RL. Modul *action* ini berdiri sendiri dan pendekatan yang dilakukan oleh modul ini adalah:

- *Allow Action*

  Memungkinkan modul *action* untuk melaksanakan perintah.

- *Block Action*

  Jika modul RL memicu suatu tindakan dimana perintah tersebut memiliki pesan kesalahan dan telah tersimpan dalam *database,* maka perintah tersebut tidak akan dieksekusi.

- *Fake Output Action*

  Jika modul RL memicu suatu tindakan dimana perintah tersebut memiliki pesan bahwa perintah tersebut palsu dan telah tersimpan dalam *database,* maka perintah tersebut tidak akan dieksekusi.

- *Delay Action*

  Jika modul RL memicu tindakan ini, maka perintah akan dijalankan setelah interval waktu tertentu.

- *Insult Action*

  Jika modul RL memicu tindakan ini, maka modul akan melakukan *geo-localizes* alamat IP dan *insult message* akan disimpan dalam *database.*

4.3 Modul *Reinforcement Learning* (RL)

Modul ini berisi algoritma SARSA dari *Pybrain* dan model *reinforcement learning. Pybrain* menyediakan implementasi yang sederhana dan menyediakan semua komponen yang dibutuhkan oleh algoritma RL. Modul ini disimpan dalam *database* MySQL.

5. Hasil *Reinforcement Learning* pada *Adaptif-Honeypot*

Titik awal dari implementasi RL pada *adaptif-honeypot* ini adalah untuk menciptakan sistem *honeypot* baru untuk mengatasi kelemahan dari *adaptif-honeypot* yang telah ada yaitu *Heliza* [13]. Adapun perbaikan yang dilakukan meliputi:

1. Peningkatan skalabilitas dari sistem:

   Heliza bergantung pada pelaksanaan dari *Unified Modelling Language* (UML) yang terdapat pada sistem Linux.

2. Peningkatan kemampuan *Reverse Turing Test*

   Heliza memiliki masalah dimana *insult* hanya dalam bahasa inggris dan penciptanya mengamati bahwa penyerang cenderung meninggalkan sistem ketika kebangsaan mereka berbeda.

3. Meningkatkan kemampuan *learning* (pembelajaran)

   Heliza hanya menggunakan algoritma SARSA dalam penerapannya.

5.1 Peningkatan Skalabilitas dari *Adaptif-Honeypot*

Dalam perancangan RASSH ini, *Kippo* digunakan sebagai acuan, karena *Kippo* merupakan *honeypot* yang sudah terbukti sangat *scalable* dan digunakan oleh banyak ahli dan organisasi yang terlibat dalam masalah keamanan *cyber.* Dalam perancangannya, diharapkan RASSH menjadi *self-adaptif honeypot* yang *scalable.* Adapun skalabilitas dari RASSH *honeypot* ini juga berasal dari pelaksanaannya, dimana: *Heliza* menggunakan kernel UML dan ini membutuhkan upaya lebih besar dalam penerapannya, sedangkan RASSH menggunakan *Python libraries* yang portabel dan tidak membutuhkan upaya berlebih dalam implementasinya. Adapun perbandingan dari *RASSH Honeypot* dengan *Heliza* tampak pada tabel 3.1 dibawah ini:

Fitur	*RASSH*	*Heliza*
Fake File System	X	X
Adaptive Action (adaptasi *honeypot* dengan asal *attacker* dan tipe *command* yang digunakan)	X	
Low Overhead (penggunaan CPU yang rendah)	X	
Portability (sistem *honeypot* ini bisa berjalan di sistem operasi yang berbeda)	X	
Tested RL Algorithms (benchmark dari implementasi RL)		X
Comparative Tests (perbandingan dengan sistem *honeypot* yang umum)	X	X
Proxy Mode Commands (menawarkan akses langsung ke *hosting* dari sistem operasi)	X	

Tabel 3.1 Perbandingan RASSH *Honeypot* dengan *Heliza* [13]

5.2 Peningkatan Kemampuan *Reverse Turing Test*

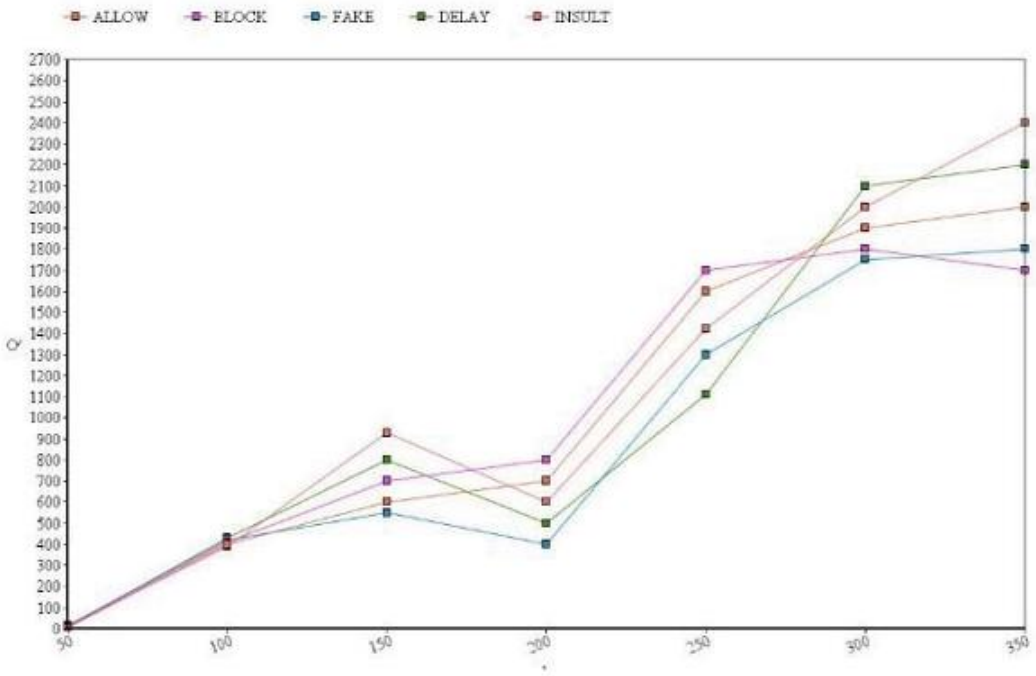
Setiap serangan yang terjadi pada SSH *honeypot* akan dikumpulkan secara otomatis. Pada saat tes sederhana yang dilakukan, seperti mengaktifkan fitur *logging* pada layanan SSH menunjukkan bahwa pada internet ada "*bots*" yang memindai seluruh *range* dari alamat IP dengan satu tujuan yaitu melakukan *brute force* pada *account* pengguna. Beberapa *bots* telah dikonfigurasi, sehingga ketika mereka telah masuk dalam sistem, mereka akan mengeluarkan suatu set perintah untuk mengumpulkan informasi terkait sistem tersebut. Pada saat yang sama juga dilakukan penyerangan oleh manusia yang juga mencoba untuk menemukan rincian informasi dari sistem. Adapun salah satu cara untuk membedakan serangan dari *bots* atau *humans*, menurut pencipta *Heliza* adalah dengan menggunakan konsep *Reverse Tes Turing*.

Pada pengujian RASSH, dihasilkan bahwa dengan mengeluarkan *insult messages,* dalam bahasa yang sesuai dengan alamat IP sumber, dapat meningkatkan kemampuan untuk menentukan asal tempat dari *attacker* [13]. Jika dibandingkan dengan *Heliza,* maka 15% penyerang akan meninggalkan sistem setelah menerima *insult messages,* sedangkan dengan menggunakan

RASSH, 10% penyerang akan meninggalkan sistem setelah menerima *insult messages.*

5.3 Peningkatan kemampuan belajar *(learning capability)*

Studi menunjukkan bahwa ada ketertarikan nyata dalam menggunakan kecerdasan buatan *(artificial intelligence)* untuk mengembangkan sistem *adaptif-honeypot.* RASSH merupakan salah satu *honeypot* yang menggunakan *machine learning library,* dimana menawarkan pada para ahli untuk mengintegasikan algoritma tersebut ke dalam sistem *honeypot* [13]. Salah satu tujuan RASSH ini adalah uintuk menguji kelayakan atau optimasi dari algoritma *machine leaning* ini. Adapun perbandingan antara RASSH yang menggunakan algoritma SARSA dengan *datasheet* yang sama dengan *Heliza* tampak pada gambar 3.3 dibawah ini [13]:



Gambar 3.3 RASSH *test-action evolution for wget (*$r_a$reward) [13]

Penggunaan RASSH sangat menjanjikan, seperti pada gambar 3.3 diaats, bahwa perilaku dari RASSH serupa dengan *Heliza* [13]. Dengan hasil ini, dapat dilihat bahwa penggunaan *machine learning* yang dikembangkan dengan *python library* dapat membuat *self-adaptif honeypot* memiliki kemampuan tinggi untuk dapat berinteraksi dengan *attacker.*

6. Penutup

6.1 Kesimpulan

1. *Honeypots*, merupakan mekanisme baru dalam keamanan, membantu dalam memonitor dan mempelajari serangan.
2. Kemampuan dari honeypot dapat ditingkatkan, salah satunya adalah dengan mengintegrasikannya dengan kemampuan *self-adaptive*, dengan artian teknik kecerdasan buatan *(artificial intelligence)*
3. RASSH merupakan sebuah *self-adaptif honeypot* yang lebih dalam skalabilitas dan menciptakan sebuah *framework* yang potensial untuk dikembangkan di masa depan*,* dimana RASSH mempunyai kemampuan belajar *(learning)* berprilaku dengan berinteraksi dengan *attacker.*
4. RASSH sangat ringan *(lighter)* dan mudah digunakan, bahkan untuk orang teknikal yang belum berpengalaman dengan ini.
5. Dengan menggunakan *machine learning libraries,* para pakar yang ahli dalam bidang kecerdasan buatan akan menjadi lebih mudah untuk membuat algoritma yang handal untuk setiap tipe *honeypot.*

6.2 Saran

Untuk pengembangan dimasa depan, dicoba untuk membuat adaptif-honeypot dengan Multiagent Learning dan Accelerating Multiagent Reinforcement learning by Equilibrium Transfer [14].

DAFTAR PUSTAKA

[1] I. Koniaris, G. Papadimitriou and P. Nicopolitidis, "Analysis and Visualization of SSH Attacks Using Honeypot," *2013 IEEE EUROCON,* pp. 65-72, 2013.

[2] M. S. Zemene and P. S. Avadhani, "Implementing High Interaction Honeypot to Study SSH Attacks," *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI),* pp. 1898-1903, 2015.

[3] A. K. Soud, R. Bansal and R. J. Enbody, "Cybercrime: Dissecting the State of Underground Enterprise," *IEEE Internet Computing,* vol. 17, no. 1, pp. 60-68, 2013.

[4] G. Wagener, R. State, A. Dulaunoy and T. Engle, "Heliza: Talking Dirty to the Attackers," *Computer Virology,* vol. 7, pp. 221-232, 2011.

[5] A. Pauna, "Improved Self Adaptive Honeypots Sapable of Detecting Rootkit Malware," *2012 9th International Conference on Communications (COMM),* pp. 281-284, 2012.

[6] X. Suo, X. Han and y. Gao, "Research on the Application of Honeypot Technology in Intrusion Detection System," *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA),* pp. 1030-1032, 2014.

[7] W. Z. A. Zakaria and M. L. M. Kiah, "A Review of Dynamic and Intelligent Honeypot," *ScienceAsia 39S,* pp. 1-5, 2013.

[8] L. Zheng, C. Zhao and Z. Wang, "The SSH Protocol Audit System Based on Proxy Technology," *2013 Fifth International Conference on Computational and Information Sciences (ICCIS),* pp. 1489-1492, 2013.

[9] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Connection Protocol," January 2006. [Online]. Available: https://www.ietf.org/rfc/rfc4254.txt. [Accessed 12 Desember 2015].

[10] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Massachusetts : The MIT Press, 1998.

[11] M. Oosterhof, "https://github.com/desaster/kippo," 28 May 2014. [Online]. Available: https://github.com/desaster/kippo. [Accessed 12 12 2015].

[12] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Ruckstie and J. Schmidhuber, "Pybrain," *The Journal of Machine Learning Research,* vol. 11, pp. 743-746, 2010.

[13] A. Pauna and I. Bica, "RASSH - Reinforced Adaptive SSH Honeypot," *2014 10th International Conference on Communications (COMM),* pp. 1-6, 2014.

[14] Y. Hu, Y. Gao and B. An, "Accelerating Multiagent Reinforcement Learning by Equilibrium Transfer," *IEEE TRANSACTIONS ON CYBERNETICS,* vol. 45, pp. 1289-1302, 2015.