PERBANDINGAN SUMBER DAYA HONEYPOT BERBASIS COWRIE DAN OPENCANARY TERHADAP SERANGAN SSH BRUTE-FORCE DAN PORT-SCANNING

(*Comparison Of Cowrie And Opencanary Based Honeypot Resources Against SSH Brute-force And Port-scanning Attacks*)

Yoga Tri Pramana*[1], Raphael Bianco Huwae [1], Andy Hidayat Jatmika [1]
[1]Dept Informatics Engineering, Mataram University
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: yoga.pra.87.91@gmail.com, [Raphael.bianco.huwae, andy]@unram.ac.id

Abstract

The rapid advancement of information technology demands good security. One approach that can be taken and is considered effective for identifying and analyzing cyberattacks is to implement a honeypot security layer. In a comparative study of two types of honeypots, namely Opencanary and Cowrie, which examined the form of honeypot response and the main focus of resource requirements to attacks along with the use of two methods of brute-force and port-scanning attacks. The attack was carried out virtually with the self-attack method, with the aim of comparing each honeypot in terms of resource requirements. The results show that in brute-force attacks, Opencanary has lower resource requirements with the highest CPU/RAM requirements of only 14% / 1.3%, while Cowrie requires more resources with the highest CPU/RAM requirements of 17% / 1.3%. While port-scanning attacks have lower resource requirements with the highest CPU requirements in Opencanary at 3% and Cowrie at 2% and similar RAM requirements at 1.28%. This study is expected to be a benchmark in selecting a honeypot that is tailored to the existing resource requirements.

Keywords*: Honeypot*, *OpenCanary*, *Cowrie*, *Brute-Force*, *Port-Scanning*

**Corresponding Author*

1. PENDAHULUAN

Pada perkembangan era modern saat ini, memiliki sistem operasi atau trafik jaringan yang selalu aman menjadi kebutuhan paling mendasar yang di butuh kan Indonesia baik dalam lembaga atau organisasi pemerintahannya. Memiliki struktur lapisan keamanan yang buruk dapat menyebabkan masalah seperti menghambat pelayanan publik, hingga bocornya akses data penting. Namun penggunaan internet yang terus meluas secara global menyebabkan naiknya risiko serangan yang mengancam keamanan sistem, baik pada metode serangan umum seperti *port-scanning* dengan memindai layanan *port* yang berpotensi untuk dieksploitasi dan *brute-force* yang menyerang dengan kombinasi *password* secara berulang untuk mendapatkan akses sistem [1].

Salah satu penyelesaian yang diberikan adalah penanaman *firewall* dan sistem deteksi penyerangan seperti *IDS* (*Intrusion Detection System*) yang dikatakan mampu sebagai pelindungan awal, namun *IDS* sendiri memiliki banyak permasalahan seperti kurangnya pengenalan pola serangan dan tingkah laku dari penyerangan yang tidak bisa di investigasi. Sementara untuk *firewall* sendiri bertugas sebagai penyaring (*filter*) di dalam jaringan tanpa adanya pencegahan lainnya, sehingga keterbatasan ini dapat memunculkan kerentanan pada keamanan sebab tidak adanya proteksi lebih, baik dalam mengamankan sistem ataupun pengenalan perilaku dari penyerang yang siap dalam melakukan eksploitasi. Nantinya serangan yang berhasil menembus pertahanan dasar akan menanggung kerugian baik secara ekonomi dan struktur dalam jaringan [1].

Banyaknya permasalahan seperti serangan *brute-force* pada protokol *SSH* menjadi ancaman yang sering terjadi, khususnya pada sistem yang menjaga keamanan data ataupun server. Keberhasilan serangan dapat menimbulkan pencurian data serta kendali penuh atas server. Sementara serangan umum lainnya yang sering digunakan adalah *port-scanning* yang sering digunakan sebagai pemetaan dalam menentukan layanan yang memiliki potensi untuk di eksploitasi [1].

Berdasarkan laporan tahunan oleh Badan Siber dan Sandi Negara (BSSN), pada tahun 2020 telah ter data lebih dari 495 juta serangan siber yang menyerang jaringan Indonesia, seperti serangan *brute-force*, eksploitasi sistem hingga *port-scanning* pada lingkungan jaringan publik. Ancaman sejenis tidak hanya terjadi di Indonesia saja namun telah terjadi secara global, berdasarkan laporan *FBI* tahun 2020 kerugian yang terjadi karena serangan siber mencapai 4,2 miliar US *dollar*, dengan menggunakan serangan tahap awal seperti *brute-force* dan *port-scanning*. Dengan ini menunjukkan kebutuhan sistem deteksi dini dan lapisan keamanan semakin dibutuhkan [2]. Oleh sebab itu diperlukan solusi lapisan keamanan yang fleksibel yang mampu memenuhi kebutuhan dasar hingga kompleks. Salah satu lapisan keamanan yang sering digunakan dalam menjaga keamanan sistem operasi atau jaringan adalah *Honeypot*, dengan menggunakan *Honeypot* proses pemantauan pola dari serangan dapat dilakukan, karena honeypot merupakan sistem umpan yang dirancang untuk menarik, mendeteksi dan menganalisis serangan [2].

Berdasarkan kesimpulan tersebut maka dalam penelitian ini akan dilakukannya implementasi *honeypot*, pemilihan lapisan keamanan ini dilatar belakangi oleh kemampuannya sebagai *decoy* dengan melindungi sistem asli secara tidak langsung, seolah-olah menjadi sistem sesungguhnya yang mengarahkan ataupun tergoda karena kerentanan sistem sehingga sistem asli tetap aman karena setiap jenis serangan yang dilakukan hanya berada di dalam simulasi. Nantinya akan dilakukan pemilihan dua jenis *honeypot*, yaitu cowrie dan opencanary. Pemilihan cowrie didasari oleh kemampuannya dalam mencatat serangan *SSH* secara mendetail seperti tingkah laku dan sifat dari penyerang, sementara opencanary dipilih karena ringan serta mampu memberikan *alert* yang lebih cepat dan fleksibel. Adapun metode serangan yang digunakan SSH *brute-force* dan *port-scanning* sebab metode serangan ini paling umum digunakan di tahap awal intrusi jaringan, alasan lainnya lebih relevan dengan layanan SSH yang di tawarkan oleh Cowrie dan Opencanary.

Pemilihan *honeypot* demi memastikan keamanan trafik dapat melalui proses uji coba terlebih dahulu sehingga implementasi dapat dikatakan baik dan sesuai dengan kebutuhan, dengan acuan ini penulisan jurnal memiliki tujuan untuk melakukan penelitian perbandingan performa *honeypot* dengan fokus utama kebutuhan sumber daya *honeypot*, menggunakan dua jenis *honeypot* yaitu Cowrie dan Opencanary, serta dua metode serangan *SSH brute-force* dan *port-scanning,* nantinya hasil akhir penelitian diharapkan dapat digunakan dalam memilih *honeypot* dari sisi kebutuhan sumber daya yang di miliki atau kelengkapan log yang dibutuhkan [3].

2. TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian milik D. S. Nugroho, [4] berfokus pada pola serangan yang dilakukan di dalam *SSH* menggunakan serangan umum seperti *brute-force* yang menuju log cowrie dalam virtual, juga dalam penelitian melakukan berbagai klasifikasi pada tingkah laku percobaan *login* dengan algoritma Naive Bayes. Pada hasilnya mengarah pada percobaan *login* secara terus menerus dengan berbagai variasi alamat *IP*, yang menyatakan bahwa cowrie efektif dalam mengumpulkan data penyerangan. Kekurangan dalam penelitian tidak adanya penelitian kebutuhan sumber daya *CPU*/*RAM* pada kondisi *honeypot* terserangan.

Penelitian milik Michal Ambrozkiewicz, [5] menggambarkan struktur atau anatomi serangan SSH menuju *honeypot* dalam sistem operasi Linux, yang juga menggunakan metode serangan *brute-force* juga menunjukkan berbagai jenis serangan yang berasal dari banyaknya alamat *IP* dalam penelitian menggunakan alat bantu pemindaian yang tersusun dalam pemrograman Go. Dalam hasil penelitian menemukan beban serangan sering muncul dengan kombinasi *brute-force* dan pemindaian otomatis, selain itu menilai pertahanan *honeypot* dan biaya komputasi ketika serangan memuncak. Kelemahan dari penelitian ini terletak tidak menyertakan metrik performa *host*.

Penelitian milik L. Moessner, [6] fokus pada pengalaman dalam penggunaan *honeypot* opencanary, penelitian meliputi pemalsuan SSH, juga berbagai jenis serangan yang sering dilakukan berdasarkan kajian dari NSA/CISA. Hasil dari penelitian memperjelas bahwa opencanary sebagai *low-interaction honeypot* mampu memenuhi kelebihannya seperti ringan dan kemudahan dalam implementasi. Kelemahannya terletak tidak adanya penelitian kebutuhan sumber daya *CPU*/*RAM* terhadap risiko serangan *brute-force* dan *port-scanning*.

Penelitian milik A. A. Karo Karo, dkk, [7] fokus melakukan peningkatan pada opencanary yang bertujuan memberikan gambaran terhadap tingkah laku penyerang. Lingkungan *honeypot* dalam penelitian ini memiliki domain protokol yang berbeda, dengan memperluas fitur logging tanpa penambahan kompleksitas. Kelemahan pada penelitian terletak tidak adanya penelitian dalam kebutuhan sumber daya CPU/RAM dan tidak mengunakan serangan umum

seperti brute-force dan port-scanning karena penelitian bersifat khusus pada port dan layanan MS SQL.

Penelitian milik N. Ilga, dkk [8] melakukan pemetaan pada *honeypot open-source* salah satunya cowrie, dengan melakukan berbagai macam perbandingan seperti model interaksi, protokol dan ekosistem alat. Dalam penelitian menyatakan cowrie sebagai salah satu *honeypot* dengan kemampuan pencatatan *log shell* terinformatif, kelemahan pada tidak membandingkan kebutuhan sumber daya *CPU*/*RAM* pada kondisi server *honeypot* bebas dalam serangan.

Berdasarkan Referensi-referensi *honeypot* sebelumnya khususnya dalam pola serangan, klasifikasi *brute-force*, analisa *shell*, dan pembahasan lainnya dalam efektivitas pencatatan *log*. Namun, belum ada penelitian yang membahas dalam pengujian kebutuhan sumber daya (*CPU* dan *RAM*) terhadap *honeypot* Cowrie dan Opencanary dalam serangan *brute-force* dan *port-scanning*. Oleh karena itu *state of the art* pada penelitian ini berpusat dalam pengukuran performa dan efisiensi sumber daya terhadap dua *honeypot* yang berbeda dalam tingkat interaksi dan skenario serangan umum.

2.2. Teori Dasar

2.2.1. Definisi *Honeypot*

Honeypot adalah lapisan keamanan yang memang dirancang sebagai penjebak para pelaku serangan siber [3] dengan menyajikan berbagai jenis layanan ataupun sistem yang terlihat rentan secara keamanan, adapun hasil penelitian yang mengatakan, *honeypot* adalah alat yang memiliki fungsi sebagai pendeteksi dan pengalihan yang digunakan sebagai salah satu cara dalam mempelajari serangan dalam berupaya masuk ke dalam akses sistem operasi secara tidak sah. *Honeypot* sendiri dibagai menjadi 3 jenis yaitu *low-interaction*, *medium-interaction*, dan *high-interaction* [9],[10]. *Low-interaction* hanya meniru layanan ataupun protokol tertentu tanpa menyediakan sebuah sistem secara utuh, namun karena ini honeypot menjadi relatif ringan dan mudah untuk digunkan, *medium-interaction* memberikan lebih banyak interaksi simulasi layanan seperti *plugin* dan visual *UI*, meskipun tetap terbatas dalam penggunaan ataupun layanan yang dapat dideteksi, *high-interaction* honeypot menyediakan sistem secara nyata untuk diamati atau digunakan dalam penelitian. Namun dalam konteks jurnal ini, pembahasan yang dilakukan hanya sebatas pada *low-interaction honeypot* karena efisiensi penggunaananya dalam menangani serangan layanan berbasis *SSH*, tanpa menimbulkan risiko tinggi terhadap sistem utama [11].

2.2.2. *Honeypot* Opencanary

Opencanary merupakan salah satu *honeypot* yang memiliki tingkat interaksi *low-interaction* yang dikembangkan Thinkst Applied Research sebagai bentuk pendeteksian pergerakan atau serangan asing berada dalam sistem dan jaringan melalui pembuatan simulasi protokol. Opencanary digolongkan sebagai *low-interaction* sebab keterbatasan seperti tidak memiliki simulasi *shell* dan *user*. Walaupun kekurangan tersebut opencanary dapat dikonfigurasi sehingga mampu melakukan berbagai jenis *alert* seperti *file log*, *syslog*, dan email. Tidak hanya itu kelebihan lainnya terletak pada kemudahan dalam implementasi dan konfigurasi ke dalam sistem serta kebutuhan sumber daya tergolong rendah [6].

2.2.3. *Honeypot* Cowrie

Cowrie merupakan *honeypot* yang dikembangkan pertama kali di tahun 2014-2015 yang tergolong sebagai *medium-interaction* dibuat dan dikembangkan sebagai penganti honeypot kippo, tujuannya sendiri karena kippo berhenti dikembang dan menggantinya menjadi cowrie agar lebih modern, stabil serta penambahan fitur-fitur baru seperti simulasi shell dan *user*. Keunggulan dari cowrie terletak pada pendeteksian serangan yang mampu mencatat percobaan *login*, perintah serangan serta *file* yang ter-*download* ataupun ter-*upload* dapat diketahui, dan semuanya hanya terjadi di dalam *shell* simulasi yang dimiliki cowrie, sehingga dapat digunakan sebagai salah satu cara dalam mengenal jenis serangan, tingkah laku penyerang ataupun cara penanganan awal yang dapat dilakukan oleh pengguna sistem [4], namun pada penelitian di jalankan secara *low-interaction* karena pengujian dilakukan pada *honeypot* itu sendiri tanpa penambahan *plugin*.

2.2.4. Jenis Serangan

a. *SSH Brute-Force*

SSH (*Secure Shell*) adalah protocol jaringan yang memungkinkan pertukaran data melalui saluran yang aman antara kedua perangkat jaringan. *Brute-force* adalah metode *trial* dan *error* yang digunakan oleh program aplikasi untuk memecahkan data yang telah dienkripsi seperti *password* atau standar data enkripsi (DES). Metode ini akan mencoba semua kemungkinan yang ada dari pada menggunakan strategi yang lebih baik, berdasarkan pengertian *SSH* dan *Brute-force* yang telah dijelaskan sebelumnya dapat diambil kesimpulan bahwa *SSH Brute-force attack* adalah sebuah jenis

serangan pada *SSH* dengan mencoba semua kombinasi yang memungkinkan untuk mendapatkan akses pada sebuah *SSH* [12].

b. *Port-scanning*

Port-scanning adalah salah satu teknik yang penyerangan untuk mengidentifikasi port terbuka, tertutup, atau terfilter dan memastikan apakah terdapat layanan yang berjalan pada suatu host. Teknik serangan ini sering kali menjadi tahap awal dalam serangan untuk memastikan ataupun memetakan kerentanan sistem. Menurut Nmap Project pada tahun 2015, *port-scanning* pada umumnya dapat dilakukan menggunakan metode seperti TCP SYN Scan, UDP Scan, atau FIN Scan, dengan sistem kerja pemindai *port* mengirimkan permintaan jaringan untuk terhubung ke *port* TCP atau UDP tertentu pada komputer serta akan mencatat respons yang diberikan, nantinya pemindai *port* akan mengirimkan paket data jaringan ke suatu *port* untuk memeriksa status terkini. Deteksi aktivitas ini sangat penting karena sering kali menandai adanya percobaan serangan lanjutan [13].

2.2.5. Perangkat Lunak Pendukung

a. Python

Python adalah bahasa pemrograman yang di golongkan sebagai tingkat tinggi atau dapat dikatakan sebagai bahasa komputer yang sudah mendekati atau menyerupai bahasa manusia. Mendukung pemrograman terstruktur dan berorientasi objek. Penggunaan python untuk penelitian lebih berfokus pada pendataan kebutuhan sumber daya seperti *RAM* dan *CPU* ketika *honeypot* berjalan dalam *guest OS* dengan melakukan pengambilan *PID*.

b. Kali Linux dan Ubuntu

Kali Linux dan Ubuntu adalah salah satu distribusi linux yang berbasis *Debian* dan didistribusikan menjadi perangkat lunak sistem operasi yang bebas. Secara singkat dan jelasnya yaitu ubuntu adalah sejenis sistem operasi berbasiskan linux Debian. Linux sendiri merupakah sistem operasi *open-source* dimana mengunakan Unix yang pada dasarnya digunakan untuk server, jaringan dan riset keamanan. Pemilihan linux karena memiliki kestabilan dalam melakukan implementasi *honeypot*.

c. Virtual Machine

Virtual Machine adalah program untuk virtualisasi komputer yang ditujukan untuk komputer desktop, server, maupun laptop. Dengan menggunakan VMware yang dapat memvirtualisasikan OS 32bit dan 64bit pada sebuah komputer yang menggunakan prosesor Inter dan AMD, baik virtualisasi perangkat lunak maupun perangkat keras. Alasan utama memilih VMware merupakan perangkat lunak yang stabil sehingga memudahkan dalam melakukan virtualisasi, serta kemampuannya dalam membuat virtual *appliance* secara *native*.

2.3. Metodologi Pengujian *Honeypot SSH*

Percobaan pengujian *honeypot* dalam penelitan jurnal akan menggunakan skenario serangan layanan *SSH* yang dilakukan secara *self-attack* dengan berbagai jenis cara dan metode sebagai bentuk penelitian perbandingan antara kedua honeypot, di mana saat penelitian akan menggunakan sistem operasi Kali Linux secara virtual sebagai penyerang sistem lapisan keamanan *honeypot* melalui layanan *SSH* [14], [15]. Tujuan dilakukannya pengujian digunakan sebagai nilai ukur terhadap efektivitas tiap-tiap lapisan keamanan *honeypot* dalam merespons ataupun mencatat segala jenis upaya penyerangan yang dilakukan, parameter yang diukur sepeti data *logging*, interaksi, kebutuhan sumber daya dan penggunaan idealnya seperti apa [16]. Dengan melakukan pendekatan perbandingan seperti ini, diharapkan penelitian dapat digunakan sebagai bentuk penilaian sejauh mana pelaporan hasil yang dapat diberikan pada tiap-tiap *honeypot* jika digunakan dalam situasi nyata. Opencanary diharapkan dapat menunjukkan keunggulannya seperti kecepatan dan kesederhanaannya dalam melakukan pendataan serangan dan Cowrie dapat menunjukkan kelebihannya berupa analisis yang mendalam. Pelu diketahui pengujian hanya dilakukan secara lokal dan menggunakan metode *self-attack* secara virtual demi keamanan dan kontrol terhadap lalu lintas jaringan [17].

3. METODE PENELITIAN

3.1. Alur Penelitian

Pada tahap ini merupakan bagian identifikasi berbagai kebutuhan yang diperlukan dalam merancang lapisan keamanan yang digunakan yang dapat dilihat pada gambar 1

Gambar 1. Diagram Penelitian

Dapat dilihat pada gambar diagram tersebut langkah-langkah yang diperlukan sebelum melakukan penambahan lapisan keamanan, yang akan dijelaskan sebagai berikut:

a. Identifikasi permasalahan, di mana pada bagian ini merupakan masalah yang timbul di dalam keamanan jaringan yang ingin diperbaiki baik secara efisiensi ataupun keamanan dengan cara pengumpulan informasi penting secara literatur seperti buku ataupun jurnal mengenai cara alternatif dalam meningkatkan keamanan jaringan dan bagaimana penyelesaian tersebut dapat hasil yang lebih baik dari sebelumnya.
b. Studi literatur pada tahap ini kurang lebih memiliki kemiripan dengan tahap sebelumnya yaitu melakukan analisis, namun lebih mengarah pencarian ide secara literatur melalui buku, jurnal, dll, serta lebih mengarah kepada teknik penelitian. Teknik ini mengarah kepada bagaimana cara konfigurasi lapisan keamanan baik secara pemahaman ataupun metode apa yang digunakan untuk menciptakan solusi akhir di dalam masalah tersebut, karena tanpa adanya analisis berdasarkan studi literatur yang mendukung akan menyebabkan munculnya permasalahan lainnya.
c. Analisa kebutuhan, kebutuhan yang diperlukan seperti perangkat keras dan lunak yang dibutuhkan dalam pembangunan lapisan keamanan yang akan digunakan dalam penyelesaian masalah.
d. Arsitektur lapisan keamanan, pada tahap ini berfokus pada pembuatan arsitektur *honeypot* yang akan digambarkan secara asli sesuai dengan lingkungan penelitian sistem yang digunakan dengan tujuan memberikan visual serta penjelasan dari penelitian dan cara kerja lapisan keamanan *honeypot*.
e. Pengujian dan evaluasi sistem, tahap ini akan menentukan bagaimana lapisan keamanan yang telah dirancang akan di teliti, bertujuan mencari sistem yang dapat saling mendukung dan mencapai penyelesaian yang telah di tentukan. Tahap ini adalah pokok bahasan penelitian karena terdiri dari poin-poin penting seperti pengujian lapisan keamanan yang digunakan opencanary dan cowrie beserta hasil dari *monitoring*.
f. Pada tahap terakhir merupakan tahap dokumentasi dan pelaporan hasil, dengan melakukan penyusunan laporan secara rinci dan mendetail beserta di perkuat oleh buku dan jurnal ilmiah yang digunakan untuk membuktikan keaslian dari hasil dan argumentasi yang di tulis pada jurnal ini.

3.2. Studi Literatur

pada tahap ini merupakan proses awal yaitu pengumpulan studi literatur dengan tujuan sebagai analisis metode yang digunakan seperti konfigurasi dan pemasangan lapisan keamanan melalui literatur dan sumber terpercaya seperti buku dan jurnal yang digunakan untuk mendukung argumentasi rancangan ataupun melalui teknik metode yang digunakan dalam penelitian ini.

3.3. Analisis Kebutuhan *Hardware* dan *Software*

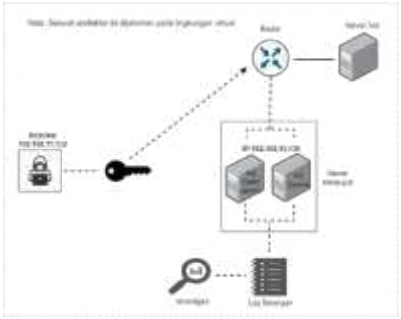
Pada tahap ini terdiri atas 2 kebutuhan perangkat keras (*hardware*) dan perangkat lunak (*software*) yang dibutuhkan dalam pengembangan masalah yang diuji sebagai berikut:

a. Perangkat keras yang dibutuhkan hanya sebuah laptop dengan spesifikasi yang cukup untuk menjalankan aplikasi seperti *virtual box*. Pada kasus ini menggunakan laptop dengan spesifikasi *processor AMD Ryzen* 5, memiliki *RAM* 8 *GB* dan menjalankan sistem operasi *Windows* 11 .
b. Aplikasi *virtual-machine* yang digunakan untuk menjalankan sistem operasi dan server yang akan digunakan dalam kasus lapisan keamanan ini. Sementara aplikasi yang dapat digunakan adalah VMware versi 17 pro.
c. Ubuntu versi 22.04, penggunaan sistem operasi Ubuntu dikarenakan aplikasi seperti Opencannary dan Cowrie yang hanya didukung oleh *OS* tersebut, sementara faktor lainya *OS* Ubuntu stabil dalam menjalankan aplikasi, serta komunitas lebih aktif.
d. Kali Linux versi 25, Sistem Operasi ini digunakan sebagai tempat penyerangan ke lapisan keamanan yang digunakan. Di mana pada dasarnya Kali Linux digunakan sebagai *Ethical Hacking*, dengan menggunakan layanan yang sudah disediakan.
e. Python versi 3.12 adalah bahasa pemrograman yang berperan pada

penelitian pendataan kebutuhan sumber daya seperti RAM dan CPU yang dibutuhkan dalam menjalankan *honeypot*.

3.4. Rancangan Lapisan Keamanan

Arsitektur lapisan keamanan, pada tahap ini berfokus pada pembuatan arsitektur *honeypot* yang akan digambarkan secara asli sesuai dengan lingkungan penelitian sistem dengan tujuan memberikan visual serta penjelasan dari penelitian dan cara kerja lapisan keamanan *honeypot*.



Gambar 2. Sistem Kerja *Honeypot*

Pada gambar di atas merupakan kerangka sistem kerja pada penelitian yang dilakukan secara virtual. Pada arsitektur ini dapat di lihat bahwa penyerang akan melakukan penyerangan ke dalam server utama, namun dilindungi secara tidak langsung oleh *honeypot* yang memang berperan sebagai *decoy* dengan melakukan pendeteksian serangan melalui pembuatan simulasi yang tampak nyata seperti menyajikan layanan sesungguhnya yang terlihat rentan, hanya saja hasil serangan tidak mengarah pada server asli pengguna namun pada server palsu *honeypot* di mana nantinya seluruh *log* serangan bisa di rubah menjadi *file*, yang nantinya *file* berisi seluruh serangan yang dapat dianalisis sebagai bentuk investigasi terhadap serangan terhadap PC target.

3.5. Pengujian dan Evaluasi Sistem

Gambar 3. Rancangan Pengujian

Pada gambar di atas merupakan rancangan pengujian yang akan dilakukan dalam penelitian *honeypot*, yang menjadi 3 bagian yaitu pengujian serangan, interaksi *honeypot* dan kebutuhan sumber daya. Nantinya hasil pengujian akan digunakan sebagai nilai ukur antara opencanary dan cowrie sebagai pembanding kelebihan dan keterbatasan dari kedua *honeypot* tersebut.

3.5.1. Pengujian Serangan

Pada bagian pengujian serangan akan dilakukan secara *self-attack* dengan berfokus pada penyerangan layanan SSH, pemilihan dua metode *SSH brute-force* dan *port-scanning* yang secara langsung mendata serta memastikan serangan pada tiap-tiap *honeypot*, opencanary dan cowrie mampu mendeteksi upaya koneksi SSH. Pada penelitian ini akan berfokus pada koneksi secara tunggal dengan berisikan variasi serangan yang nantinya di uji sesuai dengan pengujian Tabel 1.

TABEL I. PENGUJIAN SERANGAN

Jenis *Honeypot*	Deskripsi Serangan	Hasil yang Diharapkan
Opencanary dan Cowrie	Upaya koneksi *SSH* tanpa proses login	*honeypot* mencatat koneksi TCP pada *port* SSH, alamat IP sumber, *timestamp*, dan upaya *login* jika muncul
	Upaya *login* dengan *username* `*root*` dan *password* salah	*Log* mencatat *username* yang dicoba, status gagal *login*, IP, dan *timestamp*
	Login dengan *user* dan *password* yang disiapkan untuk masuk ke sistem dan menjalankan perintah sederhana	*Log* mencatat *login* sukses, *session start*, perintah yang diketik, *output*/aktivitas *shell*, dan *IP* penyerang
	setelah *login*, melakukan transfer *file*	*Log* mencatat upaya transfer, nama *file*, *IP* sumber, dan status transfer
	Upaya login dengan metode *brute-force*	*honeypot* mencatat koneksi upaya login SSH, alamat IP

Jenis *Honeypot*	Deskripsi Serangan	Hasil yang Diharapkan
		sumber, *timestamp*, dan upaya *login*
	Upaya mendeteksi kondisi port dengan metode *port-scanning*	*Honeypot* mencatat koneksi port, IP sumber dan *timestamp*

3.5.2. Interaksi *Honeypot*

Pada bagian interaksi *honeypot* akan dilakukan setelah pendataan hasil serangan pada layanan *SSH* berhasil dilakukan, bentuk penelitian interaksi akan nilai berdasarkan bagaimana hasil log dan kelebihan dari *honeypot* dalam memberikan *alert* yang didapat, bagaimana interaksi penyerang dengan *honeypot* dalam *shell*, jumlah perintah yang dapat di eksekusi dan pengujian *port* yang dilakukan.

TABEL II. PENGUJIAN KONEKSI PORT

Jenis Honeypot	Jenis Serangan	Pengujian Port	Hasil yang Diharapkan
Opencanary dan Cowrie	*Port-Scanning*	21	Kondisi/status *port* dalam penyerangan dan jumlah kebutuhan sumber daya pada CPU dan RAM dan persenan peningkatan.
		22	
		80	
		2222	

3.5.3. Kebutuhan Sumber Daya *Honeypot*

Pada bagian akhir adalah penelitian kebutuhan sumber daya *honeypot* yang berfokus pada *CPU* dan *RAM* sebagai parameter pengukur sumber daya yang digunakan oleh *opencanary* dan *cowrie*. Keduanya akan diukur selama 5 menit dengan bahasa pemrograman python sebagai *code* yang membantu pengukuran penggunaan *CPU* dan *RAM*, penelitian akan dibagi menjadi 2 tahap, tahap pertama akan mengukur kebutuhan sumber daya ketika posisi server *honeypot* tidak diserang dan tahap kedua mengukur kebutuhan sumber daya ketika *sever honeypot* diserang yang dilakukan selama 300 detik sebagai efisiensi penelitian dan hasil berfokus pada puncak kebutuhan. Hasil akhir penggunaan sumber daya akan dicantumkan dalam bentuk statistik.

4. HASIL DAN PEMBAHASAN

Selanjutnya pada perancangan lapisan keamanan akan berfokus pada 2 Sistem Operasi yaitu Ubuntu dan Kali Linux secara *virtual*. Ubuntu akan berperan sebagai lapisan keamanan *honeypot* yang tentunya dijalankan pada lingkungan *guest OS* juga memastikan pengambilan data menggunakan *PID honeypot* sehingga data yang dihasilkan valid. Kali Linux berperan sebagai penyerang dengan metode serangan *brute-force* dan *port-scanning*.

4.1. Hasil Pengujian

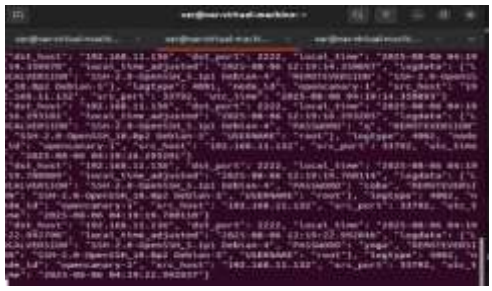
TABEL III. HASIL PENGUJIAN SERANGAN

Perintah	Hasil	Status	
		Opencanary	Cowrie
ssh -o BatchMode=yes user@192.168.11.136	*honeypot* mencatat koneksi TCP pada *port* SSH, alamat IP sumber, *timestamp*, dan upaya *login*	Berhasil	Berhasil
ssh user@192.168.11.136 (*password* salah)	*Log* mencatat *username* yang dicoba, status gagal *login*, IP, dan *timestamp*	Berhasil	Berhasil
ssh user@192.168.11.136	*Log* mencatat *login* sukses, *session start*, perintah yang diketik, *output*/aktivitas *shell*, dan *IP* penyerang	Gagal	Berhasil

Perintah	Hasil	Status	
		Open canary	Cow rie
sftp user@192.168.11.136 (put localfile.txt /tmp/)	*Log* mencatat upaya transfer, nama *file*, *IP* sumber, dan status transfer	Gagal	Berha sil
hydra -l root -P /usr/share/wordlist/rockyou.txt ssh://192.168.11.136:2222	*honeypot* mencatat koneksi upaya login SSH, alamat IP sumber, *timestamp*, dan upaya *login*	Berhasil	Berha sil
nmap -sS -sV -P 2222 192.168.11.136	*Honeypot* mencatat koneksi port, IP sumber dan *timestamp*	Berhasil	Berha sil

Pada pengujian menggunakan opencanary, saat dilakukan koneksi *SSH* dengan opsi *BatchMode=yes*, *honeypot* berhasil mencatat adanya koneksi *TCP* ke *port SSH*, alamat *IP* sumbernya, waktu kejadian, dan upaya *login* yang dilakukan. Jika dicoba *login* dengan kata sandi yang salah, *log* bisa merekam *username* yang dipakai, alamat *IP* penyerang, waktu kejadian meskipun gagal dalam menentukan status *login* dan simulasi transfer file. Meskipun begitu opencanary cukup efektif untuk mendeteksi serangan awal, walaupun rekamannya masih terbatas. Di sisi lain, cowrie mencatat data yang lebih lengkap. Saat *login* sukses, *log* menampilkan status berhasil *login*, perintah yang diketik, *output* dari *shell*, dan alamat *IP* penyerang. Selain itu, saat percobaan transfer *file* menggunakan *sftp*, *honeypot* bisa mendeteksi dan mencatat nama *file* yang dipindahkan, alamat *IP* sumbernya, dan status transfer tersebut. Dengan kemampuan ini, cowrie tidak hanya mendeteksi serangan, tapi juga bisa merekam aktivitas dan pola perilaku penyerang secara lebih mendetail, cocok digunakan sebagai analisis serangan secara lebih mendalam.

4.2. Hasil Interaksi Honeypot

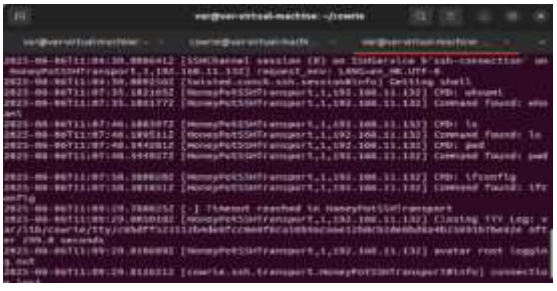


Gambar 4. Interaksi Server Opencanary

Pada gambar di atas membuktikan keberhasilan deteksi lapisan *honeypot* pada aplikasi Opencannary dalam bentuk *log*, di mana menunjukkan berbagai poin penting seperti alamat penyerang, *port*, *username*, *password*, dll. Interaksi yang diberikan terlihat monoton, di mana tidak ada pembeda yang menerangkan serangan seperti apa yang dilakukan dan keberhasilan atau kegagalan serangan terlihat sama.

Gambar 5. Interaksi Opencanary dengan *Email*

Pada gambar di atas menunjukkan bentuk interaksi lain dari opencanary, meskipun dari hasil uji sebelumnya membuktikan bahwa keterbatasan dari opencanary adalah ketidakmampuan membuat *shell* dan *user* palsu, namun kelebihannya terletak pada interaksi *alert* yang diberikan tidak hanya dalam bentuk *log system* tetapi juga dapat terhubung pesan *email* sehingga menjadi nilai tambah dalam bentuk interaksi.



Gambar 6. Interaksi Server Cowrie

Pada gambar di atas adalah hasil *log* perintah SSH yang terdeteksi oleh cowrie, dapat dilihat cowrie langsung menjalankan *shell* palsu dan mencatat setiap perintah yang digunakan penyerang seperti "*ls*", "*whoami*", "*ifconfic*", dll yang nantinya dapat digunakan sebagai investigasi dan penanganan yang

dapat dilakukan oleh pengguna *OS*. Sehingga poin tambah terletak dari kekayaan informasi yang diberikan.

TABEL IV. HASIL KONEKSI PORT

Jenis HP	Port	CPU (%)	RAM (%)	Status Port	
				Open	Close
Open canary	21	1.0	1.28	✓	
	22	2.0	1.28	✓	
	80	1.0	1.28	✓	
	2222	3.0	1.28	✓	
Cowrie	21	0.0	1.28		✓
	22	2.0	1.28	✓	
	80	0.0	1.28		✓
	2222	2.0	1.28	✓	

Pada tabel di atas merupakan hasil uji koneksi pada tiap *port*, pengujian sebagai bentuk pendataan kebutuhan sumber daya pada serangan *port-scanning*. Dapat dilihat pada hasil penelitian seluruh *port* memiliki status terbuka kecuali pada cowrie karena batasan layanan yang hanya berupa *SSH*, sehingga hanya dapat membuka *port* 22 dan 2222. Terdapat kesamaan hasil baik opencanary atau cowrie yaitu pada bagian *RAM* 1.28% dan pada *port* 22 membutuhkan *CPU* 2.0%. Perbedaan terletak pada opencanary *port* 21, 80 dan 2222 yang mana tiap *port* membutuhkan *CPU* 1%/1%/3%. Kebutuhan sumber daya yang rendah ini normal, sebab bentuk serangan *port-scanning* yang sangat singkat dan hanya bertugas melihat status *port*.

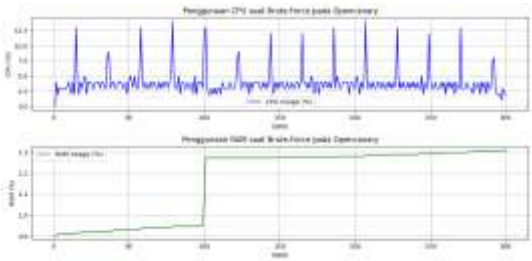
4.3. Hasil Kebutuhan Sumber Daya

4.3.1. Grafik CPU dan RAM Posisi Tidak Diserang

Gambar 7. Statistik Sumber Daya Tanpa Serangan

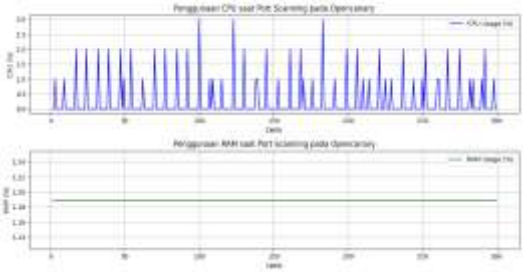
Pada gambar di atas bentuk statistik kebutuhan sumber daya yang menunjukkan kebutuhan *CPU* dan *RAM* pada opencanary dan cowrie saat tidak ada penyerangan terhadap server *honeypot*. Dapat di lihat bahwa opencanary dan cowrie memiliki kebutuhan sumber daya yang sama pada *CPU* dengan kebutuhan 0% dan perbedaan hanya terletak pada *RAM* di mana opencanary membutuhkan sekitar 1,18% dan cowrie membutuhkan sekitar 1,28%. Pada hasil statistik tersebut dapat disimpulkan bahwa kebutuhan sumber daya ketika keadaan server *honeypot* tidak diserang adalah rendah.

4.3.2. Kebutuhan Sumber Daya Opencanary



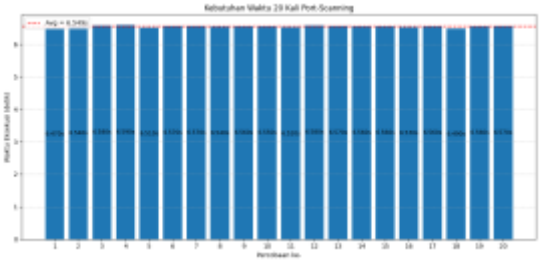
Gambar 8. Statistik *Brute-Force* Opencanary

Pada gambar di atas bentuk statistik kebutuhan sumber daya pada opencanary ketika menggunakan metode *brute-force* di mana dapat dilihat kebutuhan *CPU* tertinggi adalah 14% dan kebutuhan *RAM* tertinggi adalah 1.3%. dapat dikatakan metode penyerangan lebih membebani *CPU* dibandingkan dengan *RAM*, namun masih digolongkan kebutuhan sumber daya yang rendah.



Gambar 9. Statistik *Port-Scanning* Opencanary

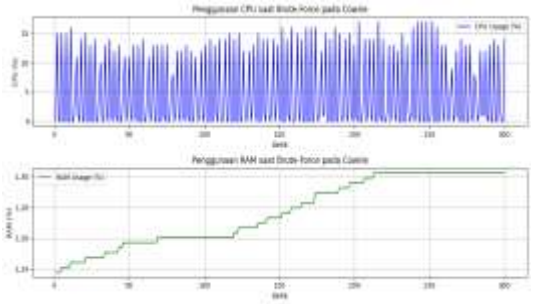
Pada gambar di atas bentuk statistik kebutuhan sumber daya pada cowrie ketika menggunakan metode *port-scanning* di mana dapat dilihat kebutuhan *CPU* tertinggi adalah 3% dan kebutuhan *RAM* tertinggi adalah 1.28%. dapat dikatakan metode penyerangan lebih membebani *CPU* dibandingkan dengan *RAM*, namun masih digolongkan kebutuhan sumber daya yang rendah.



Gambar 10. *Mean* Waktu Serangan *Port-Sacanning* Opencanary

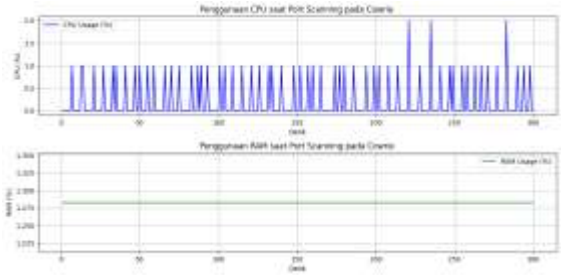
Pada gambar di atas merupakan bentuk rata-rata (*mean*) dari kebutuhan waktu dalam penyerangan *port-scanning* sebanyak 20 serangan yang diambil pada hasil serangan sebelumnya, di mana hasil rata-rata waktu pada seluruh serangan 6.549 detik.

4.3.3. Kebutuhan Sumber Daya Cowrie



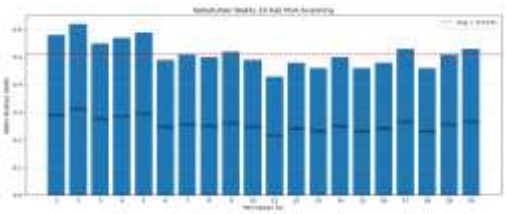
Gambar 11. Statistik *Brute-Force* Cowrie

Pada gambar di atas bentuk statistik kebutuhan sumber daya pada cowrie ketika menggunakan metode *brute-force* di mana dapat dilihat kebutuhan *CPU* tertinggi adalah 17% dan kebutuhan *RAM* tertinggi adalah 1.3%. dapat dikatakan metode penyerangan lebih membebani *CPU* dibandingkan dengan *RAM*, namun masih digolongkan kebutuhan sumber daya yang rendah.



Gambar 12. Statistik *Port-Scanning* Cowrie

Pada gambar di atas bentuk statistik kebutuhan sumber daya pada cowrie ketika menggunakan metode *port-scanning* di mana dapat dilihat kebutuhan *CPU* tertinggi adalah 2% dan kebutuhan *RAM* tertinggi adalah 1.28%. dapat dikatakan metode penyerangan lebih membebani *CPU* dibandingkan dengan *RAM*, namun masih digolongkan kebutuhan sumber daya yang rendah.

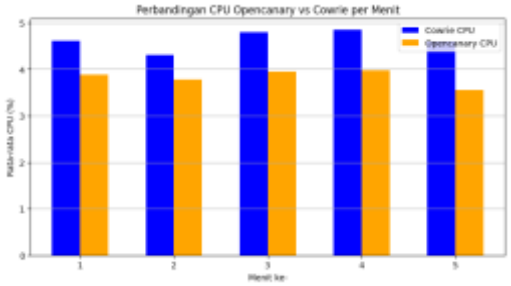


Gambar 13. *Mean* Waktu Serangan *Port-Sacanning* Cowrie

Pada gambar di atas merupakan bentuk rata-rata (*mean*) dari kebutuhan waktu dalam penyerangan *port-scanning* sebanyak 20 serangan yang diambil pada hasil serangan sebelumnya, di mana hasil rata-rata waktu pada seluruh serangan 0.513 detik.

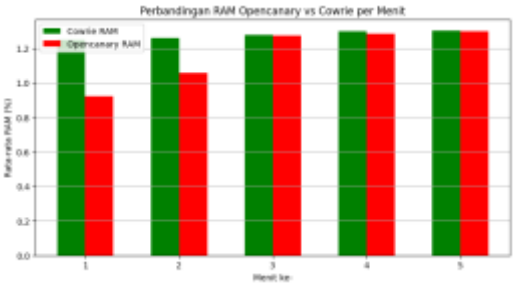
4.4. Hasil Akhir Perbandingan Opencanary dan Cowrie

4.4.1. Serangan *Brute-Force*



Gambar 14. Perbandingan *Mean CPU* Pada Serangan *Brute-Force*

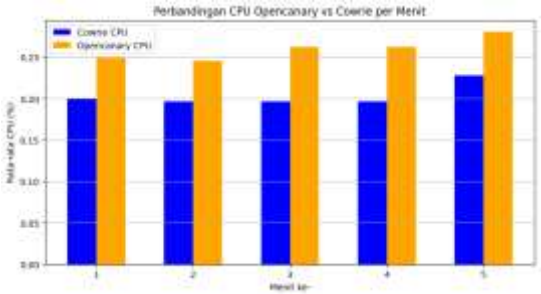
Pada gambar di atas merupakan perhitungan rata-rata kebutuhan *CPU* tiap menit yang terbagi atas 5 bagian dengan metode *brute-force* dimana dapat disimpulkan kebutuhan paling tinggi diduduki oleh cowrie dengan kebutuhan tertinggi mendekati 5% pada menit ke-4 dan kedudukan tertinggi opencanary mendekati 4% pada menit ke-4.



Gambar 15. Perbandingan *Mean RAM* Pada Serangan *Brute-Force*

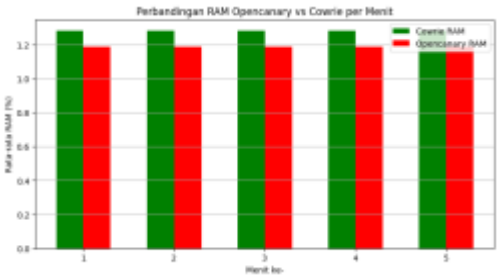
Pada gambar di atas merupakan perhitungan rata-rata kebutuhan *RAM* tiap menit yang terbagi atas 5 bagian dengan metode *brute-force* dimana dapat disimpulkan kebutuhan paling tinggi tetap diduduki oleh cowrie dengan kebutuhan tertinggi 1.2% lebih, meskipun kedudukan kebutuhan sumber daya opencanary hampir mendekati pada menit ke-3, 4 dan 5.

4.4.2. Serangan *Port-Scanning*



Gambar 16. Perbandingan *Mean CPU* Pada Serangan *Port-Scanning*

Pada gambar di atas merupakan perhitungan rata-rata kebutuhan *CPU* tiap menit yang terbagi atas 5 bagian dengan metode *port-scanning* dimana dapat disimpulkan kebutuhan paling tinggi diduduki oleh opencanary dengan kebutuhan tertinggi 0.25% lebih pada menit ke-5, sementara cowrie 0,2% lebih pada menit ke-5.



Gambar 17. Perbandingan *Mean RAM* Pada Serangan *Port-Scanning*

Pada gambar di atas merupakan perhitungan rata-rata kebutuhan *RAM* tiap menit yang terbagi atas 5 bagian dengan metode *port-scanning* dimana dapat disimpulkan kebutuhan paling tinggi diduduki oleh cowrie dengan kebutuhan tertinggi 1.2% lebih pada tiap menitnya, sementara opencanary hanya membutuhkan 1.2%.

TABEL V. KELEBIHAN DAN KEKURANGAN MENURUT HASIL AKHIR

Jenis Honeypot	Kelebihan	Kekurangan
Opencanary	Opencanary memiliki kelebihan pada kecepatan baik dalam mendeteksi serangan ataupun konfigurasi serta fleksibel dalam notifikasi serangan berupa *log system* dan *email* serta layanan yang lengkap speperti *SSH* dan *HTTP*, meskipun membutuhkan sumber daya yang lebih pada serangan *port-scanning* dibandingkan cowrie namun masih tergolong sangat rendah.	Kekuranganny a terletak pada log yang dihasilkan ambigu dan monoton informasi lebih yang sebaiknya ada seperti apakah serangan berhasil atau tidak dan tidak adanya kelebihan lain seperti shell simulasi.
Cowrie	Cowrie memiliki kelebihan pada kekayaan informasi *log* yang diberikan seperti pola serangan dan tingkah laku dari penyerang, pola ini dapat diketahui karena layaan *shell* palsu yang diberikan sehingga segala kegitan penyerang langsung terdata oleh *log.*	Kekurangan terletak pada sulitnya konfigurasi dan terbatasnya layanan yang hanya memiliki *SSH*, notifikasi hanya pada *log system* dan membutuhka n sumber daya lebih khususnya pada serangan *brute-force*.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil dari penelitian, dapat ditarik sebuah kesimpulan bahwa Opencanary dan Cowrie memiliki beberapa kesamaan di antaranya pada kebutuhan sumber daya yang tergolong rendah dan memiliki fokus ancaman serangan layanan *SSH*, namun keduanya memiliki ciri khas dan fungsinya tersendiri. Opencanary lebih baik digunakan pada pendeteksian awal karena ringan dan mudah untuk di konfigurasi meskipun hanya mencatat

percobaan *login* dan pada dasarnya dikembangkan untuk *monitoring* secara pasif yang memiliki keterbatasan sumber daya, dan memang berdasarkan penelitian kebutuhan sumber daya Opencanary tergolong rendah. berbeda dengan Cowrie, kemampuan memberikan data yang lebih teliti melalui simulasi *shell* palsu dan *filesystem* virtual, sehingga cocok sebagai analisis penyerang pasca *login* meskipun konfigurasi yang kompleks serta membutuhkan sumber daya yang lebih dibandingkan Opencanary, namun masih tergolong rendah berdasarkan penelitian. Pada akhirnya pemilihan *honeypot* harus didasari oleh kebutuhan pribadi seperti Opencanary dengan kebutuhan sumber daya yang rendah dan Cowrie yang lebih unggul dalam pendataan penyerang untuk mempelajari tingkah laku penyerang. Sebagai bentuk saran pengoptimalan *honeypot* dalam meningkatkan keamanan, khususnya pada jaringan yang terhubung pada publik dapat melakukan enkripsi dan menggunakan simulasi keamanan ataupun server yang dapat menjalankan *honeypot* secara otomatis yang sehingga mengoptimalkan keamanan jaringan dan sistem.

DAFTAR PUSTAKA

[1] M. Alharbi, H. Alhussain, and A. Alghamdi, “A survey of security threats and defense mechanisms in cloud computing,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2556–2570, Jun. 2022.

[2] Z. Mardiansyah, Y. M. Abdussyakur, and A. H. Jatmika, "Optimasi Port Knocking dan Honeypot Menggunakan IPTables sebagai Keamanan Jaringan pada Server" JTIKA (Jurnal Teknologi Informasi, Komputer, dan Aplikasinya), vol. 3, no. 2, pp. 189–199, 2021.

[3] M. R. Amal and V. Perumal, “Review of cyber attack detection: Honeypot system,” *Webology*, 2022.

[4] D. S. Nugroho, “Detection of SSH Brute Force Attacks Using Naïve Bayes Classification on Cowrie Honeypot Logs in a Virtualized Environment,” *Jurnal Ilmiah Teknologi Informasi dan Komunikasi*, Jun. 2025.

[5] Michal Ambrozkiewicz, SANS Internet Storm Center, “[Guest Diary] Anatomy of a Linux SSH Honeypot Attack,” *ISC Diary*, Jun. 13, 2025.

[6] L. Moessner, “The OpenCanary Experience,” in *SIGS 22nd SOC Forum*, 2023.

[7] A. A. Karo Karo, C. Lim, and K. E. Silaen, “Enhancing OpenCanary Honeypot to Profile Attackers Behavior on MS SQL,” in *2024 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia, Dec. 2024, pp. 268–273.

[8] N. Ilga, P. Farley, S. Bennett, and T. Welsh, “A survey of contemporary open-source honeypots,” *Journal of Network and Computer Applications*, vol. 225, no. 103675, Feb. 2023.

[9] Y. Sun, Z. Tian, M. Li, S. Su, X. Du, and M. Guizani, “Honeypot identification in softwarized industrial cyber–physical systems,” *IEEE Trans. Ind. Informatics*, vol. 17, no. 8, pp. 5542–5551, 2021.

[10] S. Lee, A. Abdullah, and N. Z. Jhanjhi, “A review on honeypot-based botnet detection models for smart factory,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 6, 2020.

[11] C. Hetzler, Z. Chen, and T. M. Khan, “Analysis of SSH honeypot effectiveness,” in *Proc.* (conference chapter / Springer chapter listing), 2023.

[12] G. K. Sadasivam, A. Jain, and N. Mahapatra, “Detection of Severe SSH Attacks Using Honeypot Servers,” *River Publishers Journal of Cyber Security*, vol. 1, no. 1, pp. 1–15, Oct. 2017.

[13] A. Subhan, Y. N. Kunang, and I. Z. Yadi, “Analyzing the Attack Pattern of Brute Force Attack on SSH Port,” in *2023 Int. Conf. Inf. Technol. Comput. (ICITCOM),* Palembang, Indonesia, Dec. 2023

[14] V. S. D. Priya and S. S. Chakkaravarthy, “Containerized cloud-based honeypot deception for tracking attackers,” *Scientific Reports*, 2023.

[15] X. Yang, J. Yuan, H. Yang, Y. Kong, H. Zhang, and J. Zhao, “A highly interactive honeypot-based approach to network threat management,” *Future Internet*, vol. 15, 2023.

[16] A. H. Anwar, C. A. Kamhoua, N. O. Leslie, et al., “Honeypot allocation for cyber deception under uncertainty,” *IEEE Trans. (paper/proceeding listing)*, 2022.

[17] P. S. Negi, A. Garg, and R. Lal, “Intrusion detection and prevention using honeypot network for cloud security,” in *Proc. on Cloud Computing, Data Science & …*, 2020.