REPOSITOR, Vol. 2, No. 10, Oktober 2020, Pp. 1380-1389
ISSN : 2714-7975
E-ISSN : 2716-1382

Integrasi Modern Honey Network Dengan Grafana Untuk Visualisasi

Novandha Yudyanto[1], Syaifuddin[2], Yufis Azhar[3]
[1,2,3]Jurusan Teknik Informatika, Universitas Muhammadiyah Malang
e-mail: no.panda94@gmail.com[*1], saifuddin@umm.ac.id[2], yufis@umm.ac.id[3]

Abstrak

Internet merupakan komoditas utama dalam hal komunikasi pada saat ini. Namun seiring dengan perkembangan zaman, penyerangan dalam internet semakin bertambah. Demi mencegah serangan tersebut banyak sistem atau program yang telah dikembangakan salah satunya ialah Honeypot . Honeypot mampu menahan, mendeteksi, dan mencatat serangan yang diterima guna mencari informasi mengenai serangan tersebut. Dengan menggunakan informasi tersebut akan lebih mudah untuk mencegah kerusakan apabila terjadi serangan yang serupa. Akan tetapi Honeypot masih belum banyak diminati oleh komunitas keamanan jaringan dikarenakan oleh rumitnya pengelolaan dan pemeliharaannya. Modern Honey Network atau MHN merupakan program open-source yang mampu menjalankan beberapa Honeypot sekaligus mengumpulkan log serangan yang diterima Honeypot tersebut. MHN bertujuan untuk mengatasi masalah rumitnya pengelolaan dan pemeliharaan Honeypot . Namun MHN tidak memiliki visualisasi untuk menganalisis pola serangan yang diterima oleh Honeypot . Hal ini tentu menyulitkan pengguna terlebih dengan banyaknya log yang dapat dihasilkan oleh Honeypot . Karena itulah dibutuhkan program untuk visualisasi log tersebut. Dalam penelitian ini Grafana digunakan untuk visualisasi log pada MHN. Dengan mengintegrasikan Grafana dengan MHN diharapkan pengguna MHN tidak kesulitan dalam membaca dan menganalisa log Honeypot .

Kata Kunci: Honeypot , Modern Honey Network, Visualisasi, Grafana

Abstract

The internet is the main commodity in terms of communication at the moment. But along with the times, attacks on the internet are increasing. In order to prevent these attacks many systems or programs have been developed, one of which is Honeypot . Honeypot can resist, detect, and record attacks received to find information about these attacks. Using those information will be easier to prevent damage in case of a similar attack. However, Honeypot is still not much in term of usage by the network security community due to the complexity of its management and maintenance. Modern Honey Network or MHN is an open-source program that is capable of running multiple Honeypot s while simultaneously gathering logs of attacks received by the Honeypot . MHN aims to overcome the complex problem of managing and maintaining of a Honeypot . However, MHN does not have a visualization to analyze the attack patterns received by the Honeypot . This becomes a difficulty for users especially with the number of logs that can be generated by Honeypot . That's why a program is needed for the log visualization. In this study Grafana is used for log visualization. By integrating Grafana with MHN, it is expected that MHN users will have no difficulty in reading and analyzing the log from Honeypot .

Keywords: Honeypot , Modern Honey Network, Visualization, Grafana

1. Pendahuluan

Pada saat ini kebutuhan akan komunikasi sangat tinggi. Tidak hanya komunikasi dekat tetapi komunikasi jarak jauh bahkan mencapai global. Dengan internet kita mampu untuk berkomunikasi tidak hanya satu kota, pulau, atau Negara, akan tetapi kita mampu berkomunikasi dengan orang lain di balik bumi ini. Perkembangan internet sangat cepat, bahkan pada tahun 2014 sudah ada lebih dari 1 milyar domain yang ada di internet[1] dan pada tahun yang sama terdapat 3 milyar pengguna internet[2]. Internet menjadi alat komunikasi yang inovatif dalam sejarah manusia bahkan menjadi inovasi yang paling penting yang pernah dikembangkan oleh manusia.

Akan tetapi, terdapat sebagian orang yang memanfaatkan internet untuk kepentingan mereka sendiri yang menyebabkan orang lain mengalami kerugian. Mereka menyerang komputer dengan berbagai cara seperti *malware, scanning , brute force, DDoS* yang menyebabkan kerugian terhadap yang diserang. Meski begitu sebagian orang yang menerima serangan tersebut tidak mengetahui adanya serangan. Mereka biasanya kekurangan informasi mengenai dari mana serangan itu berasal, siapa yang menyerang, bagaimana cara menyerang, dan kapan serangan tersebut dilakukan. Karena itulah dibutuhkan sebuah alat atau sistem bantu untuk mendeteksi serangan yang mereka terima, salah satunya ialah *honeypot*

Honeypot merupakan sebuah alat keamanan jaringan dengan tujuan untuk diserang oleh peretas[3][4]. *Honeypot* mempunyai tujuan untuk mengumpulkan semua informasi mengenai peretas yang nantinya dianalisa dan membuat penanggulangan terhadap serangan serupa[5]. *Honeypot* berbeda dari alat keamanan jaringan yang lain seperti *Firewall* atau *Intrusion Detection Systems* (IDS) karena honeypot tidak hanya mampu menyelesaikan sebuah masalah namun bisa menyelesaikan beberapa masalah sekaligus. Contohnya adalah *honeypot* bisa menghalangi sebuah serangan jaringan layaknya sebuah *Firewall* dan *honeypot* mampu mendeteksi sebuah serangan layaknya IDS pada saat yang bersamaan[6]. Akan tetapi honeypot masih kurang digunakan oleh organisasi ataupun perusahaan karena mereka beranggapan honeypot sulit untuk penerapan, konfigurasi maupun pemeliharaannya.

Pada tahun 2014 Threatstream (sekarang Anomali) mengembangkan sebuah sistem terpusat yang mempunyai sekumpulan *sensor honeypot* yaitu *Modern Honey Network* atau MHN. *Modern Honey Network* merupakan sebuah sistem manajemen yang mampu menjalankan banyak sensor dalam satu waktu yang singkat[7]. Selain itu di dalam sistem tersebut terdapat web interface yang berfungsi untuk men-*deploy honeypot,* dan melihat log penyerangan. Tetapi pengguna kesulitan untuk membaca pola penyerangan dikarenakan data yang dihasilkan hanya berupa log tanpa visualisasi seperti diagram batang atau diagram garis yang mepresentasikan pola serangan tersebut. Terlebih log yang ditampilkan oleh MHN hanya 10 log per halaman, dibandingkan dengan ribuan log yang dapat dihasilkan dari honeypot, hal ini tentu menyulitkan pengguna.. maka dari itu dibutuhkan sebuah perangkat lunak yang dapat memvisualisasikan untuk mempermudah melihat statistik dan juga mempermudah dalam menganalisa log yang dihasilkan *sensor honeypot.* Dalam penelitian ini hasil dari penyerangan akan divisualisasikan dengan menggunakan *Grafana.*

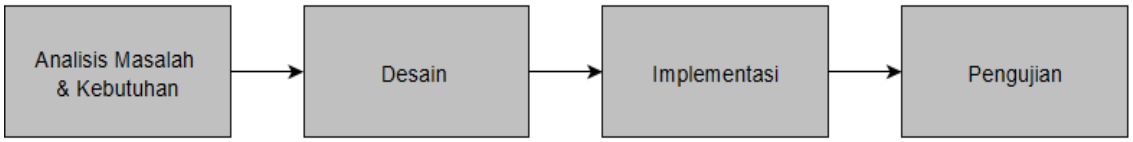
Grafana merupakan perangkat lunak open source yang mempunyai fungsi untuk memvisualisasikan, memberi peringatan, dan men-query data[8]. Dengan mengintegrasikan *Modern Honey Network* dengan *Grafana*, pembacaan log yang telah direkam oleh sensor Honeypot dapat divisualisasikan.

Berdasarkan penelitian sebelumnya yang dilakukan oleh Hibatul Wafi dkk. dengan judul "Implementation of a Modern Security Systems Honeypot Honey Network on Wireless Network" membahas bagaimana menerapkan sistem *Modern Honey Network* pada jaringan *wireless* dengan menggunakan beberapa *Honeypot*[7], sedangkan peneliti akan menerapkan *Modern Honey Network* dan akan diintegrasikan dengan *Grafana* untuk memvisualisasikan *log* yang didapat dari *Honeypot*. Sedangkan pada penelitian yang dilakukan oleh Vesselin Bontchev dan Veneta Yosifosa pada penelitian "Analysis of the Global Attack Landscape Using Data from a Telnet Honeypot"[9], membahas bagaimana menggabungkan *Grafana* dengan sebuah *Honeypot,* tetapi peneliti menggabungkan *Grafana* dengan beberapa *Honeypot* melalui *Modern Honey Network*.

Berdasarkan latar belakang diatas, peneliti bertujuan untuk mengimplementasikan *Modern Honey Network* dan mengintegrasikannya dengan *Grafana*.yang nantinya peneliti berharap bisa memudahkan seorang administrator dalam menganalisa *log* yang dihasilkan oleh *Modern Honey Network*.

2. Metode Penelitian

Pada penelitian ini peneliti menggunakan model proses penelitian sekuensial linier. Berikut adalah gambaran bagaimana alur sekuensial linier menurut Pressman[10]:



Gambar 1 *Model Proses Sekuensial Linier*

Analisis masalah merupakan langkah awal agar peneliti mengerti akan masalah yang akan dihadapi. Masalah yang dihadapi pada penelitian ini ialah banyak pengguna kurang memahami akan serangan yang diterima seperti darimana serangan itu datang, serangan apa yang dilakukan, kapan penyerangan itu terjadi. *Honeypot* menjadi salah satu solusi untuk masalah tersebut tetapi *honeypot* tersebut jarang digunakan karena sulit untuk penerapan, konfigurasi ataupun pemeliharaannya. *Modern Honey Network* atau *MHN* menjadi salah satu cara supaya pengguna mampu menggunakan honeypot – honeypot tersebut. Tetapi MHN kurang dalam hal mempresentasikan statistik serangan. MHN hanya menampilkan log – log serangan dalam bentuk daftar dan dalam satu halaman ditampilkan hanya 10 daftar saja, padahal log – log yang diterima oleh MHN mampu mencapai ribuan. hal ini tentu saja merepotkan pengguna MHN yang ingin menganalisa serangan tersebut.

Setelah analisis masalah selesai dilanjutkan dengan analisis kebutuhan untuk menganalisis kebutuhan apa saja yang diperlukan pada penelitian ini. Berikut adalah tabel kebutuhan yang dibutuhkan pada penelitian ini, **Tabel 1** berisi kebutuhan perangkat keras, **Tabel 2** perangkat lunak untuk MHN Honeypot serta **Tabel 3** berisi kebutuhan perangkat lunak untuk penyerang.

Tabel 4 *Kebutuhan Perangkat Keras*

	Laptop 1	Laptop 2	Laptop 3	Komputer 1	Komputer 2
Sistem Operasi	Ubuntu 18.04 Server	Ubuntu 18.04 Desktop	Ubuntu 18.04 Desktop	Kali Linux 2020.1b	Windows 7
Processor	AMD 2 Core	Inter 4 Core	Intel 2 Core	Intel 4 Core	Intel 4 Core
RAM	4 GB	4 GB	2 GB	8 GB	8 GB
Storage	30 GB	30 GB	20 GB	30 GB	120 GB
IP	192.168.1.20/24	192.168.1.21/24	192.168.1.22/24	192.168.1.35/24	192.168.1.30/24
Peran	Server MHN dan *Grafana*	Honeypot Dionaea	Honeypot Cowrie	Penyerang	Akses Grafana

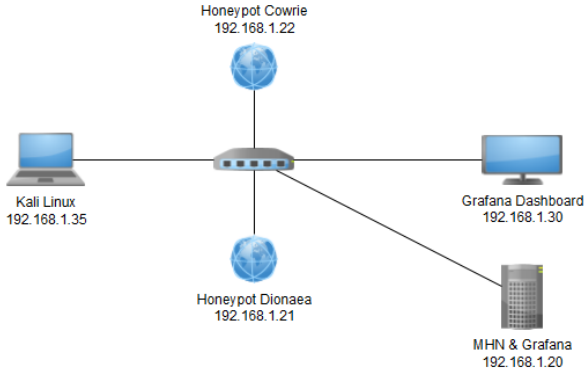
Tabel 5 *Kebutuhan Perangkat Lunak sisi MHN*

Nama	**Deskripsi**	Versi
Modern Honey Network	Perangkat Lunak yang berperan sebagai pusat untuk mengumpulkan data dari honeypot	
Grafana	Perangkat lunak untuk visualisasi data yang didapatkan dari database	6.7
MySQL	Database *open source* berbasis *Structured Query Language* yang digunakan untuk menyimpan data dari database MHN	8.0.20
Python	Bahasa Pemrograman untuk menjalankan program berbasis python	3.8.3
Cron	Perangkat Lunak untuk menjalankan perintah berdasarkan jeda waktu yang telah ditentukan	
Dionaea	Honeypot untuk menangkap exploitasi dengan cara membuka layanan yang kerap menjadi sasaran	0.8.0
Cowrie/Kippo	Honeypot untuk menangkap serangan brute force dan kegiatan penyerang pada layanan SSH dan Telnet	2.1.0

Tabel 6 *Kebutuhan Perangkat Lunak Penyerang*

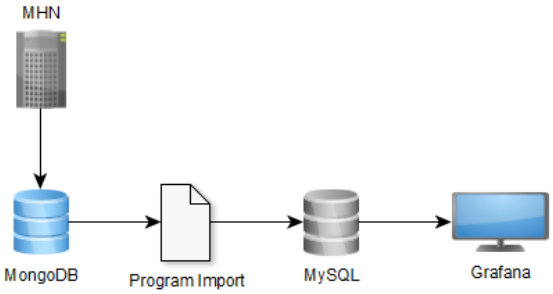
Nama	Deskripsi	Versi
Nmap	Perangkat lunak bertujuan untuk mencari informasi tentang target mulai dari IP target dalam jaringan sampai layanan yang terbuka pada target	7.80
Hping3	Perangkat lunak yang mampu memanipulasi paket yang dikirimkan seperti besar dan jumlah	6.7
Metasploit	Perangkat lunak bertujuan untuk penetration testing atau uji coba penetrasi pada kelemahan yang dimiliki suatu layanan	8.0.20

Tahap berikutnya ialah desain. Pada tahap ini akan dibahas bagaimana secara garis besar cara kerja dari sistem yang akan dibangun. Ada dua hal yang akan dibahas yang pertama adalah topologi sistem dan yang kedua adalah alur data log.



Gambar 2 *Desain Arsitektur Sistem*

Pada **Gambar 3** terlihat ada empat unsur penting yaitu komputer penyerang, honeypot cowrie dan dionaea, server MHN yang terintegrasi dengan *Grafana* , dan sebuah PC untuk melihat hasil visualisasi dari *Grafana* . Komputer penyerang akan dipasang sistem operasi Kali karena dalam sistem operasi tersebut sudah terpasang perangkat lunak untuk menyerang honeypot. Honeypot dionaea dan cowrie terpasang pada laptop terpisah agar mudah untuk melihat hasil serangan dari dua honeypot tersebut. Serangan yang diterima oleh honeypot akan direkam dan dikirimkan ke MHN. MHN akan menerima dan menampung data dari honeypot kedalam database MHN yaitu MongoDB dan dapat dilihat pada Web Apps milik MHN. Data tersebut kemudian akan dikirikan menuju *Grafana* , akan tetapi *Grafana* tidak mendukung penggunaan *MongoDB* sebagai sumber data. Karena itu dibutuhkan *database* yang didukung oleh *Grafana* , disini peneliti memilih *database MySQL.*

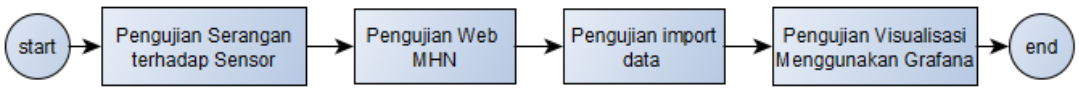


Gambar 3 *Alur Data log Honeypot*

Untuk mengirimkan data dari *MongoDB* menuju *MySQL* peneliti membuat sebuah program yang mampu membaca *record* dari *MongoDB,* kemudian mengirimkannya menuju *MySQL* dan akan memperbarui atau *update record* tersebut jika *record* tersebut sudah ada. *Record* pada *MySQL* akan dibaca oleh *Grafana* untuk divisualisasikan pada *panel* - *panel* di

dalam suatu *dashboard.* PC dapat melihat hasil visualisasi dari dashboard tersebut menggunakan *web browser*.

Setelah desain selesai dibuat maka dilakukanlah tahap implementasi. Peneliti akan mengimplementasi sesuai dengan desain yang dibuat pada tahap sebelumnya. Dimulai dari menyiapkan perangkat keras dan perangkat lunak yang dibutuhkan, sampai memasang aplikasi yang dibutuhkan. Terakhir adalah tahap pengujian. Tahap ini merupakan tahapan dimana sistem yang telah didesain dan diimplementasi diuji untuk melihat apakah sistem sudah sesuai dengan harapan pada penelitian ini. Proses pengujian terlihat pada **Gambar 4**



***Gambar 4** Alur Pengujian*

Pengujian dimulai dari serangan terhadap sensor. Penyerangan pertama ialah port scanning, kemudian dilanjutkan dengan Denial of Service dengan cara pengiriman paket secara terus menerus dan terakhir adalah exploitasi port yang terbuka. Pengujian selanjutnya adalah memeriksa apakah server MHN menerima log dari sensor. Pengujian dilakukan dengan menggunakan Web Apps yang dimiliki MHN. Setelah log berhasil ditampilkan pada *Web App* pengujian selanjutnya adalah mengimpor log tersebut menuju database *MySQL* karena database MHN, *MongoDB* tidak disupport oleh *Grafana* . Disini peneliti membuat program sendiri dengan cara kerja program pertama akan mengambil *record* dari *MongoDB* terlebih dahulu kemudian disimpan pada suatu *object*. Setelah itu program akan memasukkan record pada object menuju *MySQL* dan akan meng-update jika pada tabel telah terisi *record* dari *record MongoDB* yang lama. Terakhir adalah visualisasi pada *Grafana* . *Panel* pada *Grafana* akan memvisualisasikan data dengan menggunakan *query* yang sebelumnya telah diatur sedemikian rupa agar dapat memilih data apa saja yang akan ditampilkan pada *panel*. *Panel* tersebut akan ditempatkan pada *dashboard* bersama dengan *panel* lain yang menampilkan data berbeda.

.

3. Hasil Penelitian dan Pembahasan

3.1 Hasil Pengujian Serangan

Dalam pengujian serangan terhadap *Honeypot* , ada tiga jenis serangan yang dilakukan, *Port* Scanning, *Port* Exploit, Denial of Service

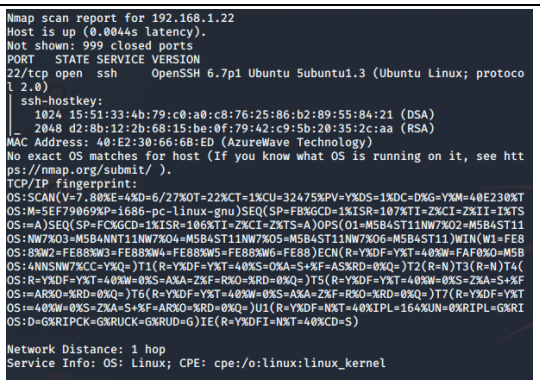
a) *Port Scanning*

Uji coba *Scanning* dilakukan dengan menggunakan program *Nmap* yang terpasang pada sistem operasi Kali. target pertama dari *Scanning* ialah *Honeypot Dionaea* pada IP 192.168.1.21 dengan perintah "*Nmap* –sS –p 1-1000 –sC 192.168.1.21", dan pada **Gambar 6** adalah hasilnya

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-27 18:46 UTC
Nmap scan report for 192.168.1.21
Host is up (0.0044s latency).
Not shown: 992 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
23/tcp  open  telnet
42/tcp  open  nameserver
53/tcp  open  domain
80/tcp  open  http
135/tcp open  msrpc
443/tcp open  https
|_http-title: Directory listing for /
| ssl-cert: Subject: commonName=Nepenthes Development Team/organizationName
=dionaea.carnivore.it/countryName=DE
| Not valid before: 2020-06-27T01:13:48
|_Not valid after:  2021-06-27T01:13:48
|_ssl-date: ERROR: Script execution failed (use -d to debug)
445/tcp open  microsoft-ds
MAC Address: C0:38:96:10:EC:3D (Hon Hai Precision Ind.)
```

***Gambar 6** Hasil dari Port Scanning pada Dionaea*

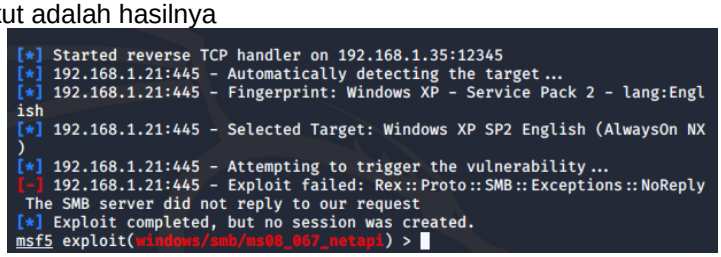
Target kedua ialah *Honeypot* Cowrie. Perintah untuk *Port Scanning* adalah "*Nmap* –sS –p 1-1000 –O –sC 192.168.1.22" dan berikut **Gambar 7** adalah hasilnya



Gambar 7 *Hasil dari Port Scanning pada Cowrie*

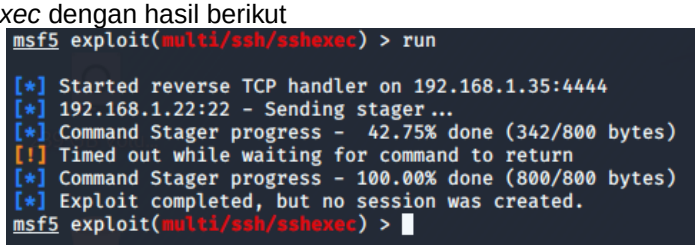
b) *Exploit Port*

Pengujian berikutnya ialah eksploitasi kepada *Port* dengan menggunakan program metasploit. Target pertama ialah *Honeypot Dionaea* pada *Port* 445 dimana *Port* tersebut sering menjadi target untuk eksploitasi. Exploit yang digunakan ialah *ms08_067_netapi,* dan berikut adalah hasilnya



Gambar 8 *Hasil exploitasi Port 445 pada DIonaea*

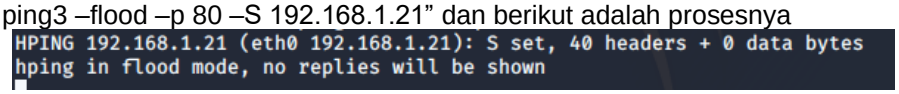
Berikutnya eksploitasi terhadap *Honeypot Cowrie* pada *Port* 22. Exploit yang digunakan ialah *sshexec* dengan hasil berikut



Gambar 9 *Hasil Exploitasi pada Cowrie*

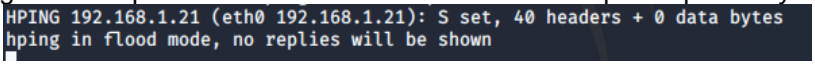
c) *Denial of Service* (DoS)

Pengujian terakhir ialah *Denial of Service* dengan menggunakan program Hping3. Target yang diserang pertama ialah *Dionaea* pada *Port* 80. Perintah yang digunakan ialah " Hping3 –flood –p 80 –S 192.168.1.21" dan berikut adalah prosesnya



Gambar 10 *Proses Denial of Service pada Dionaea*

Dilanjutkan dengan serangan terhadap *Cowrie* dengan perintah yang sama yakni "Hping3 –flood –p 22 –S 192.168.1.22" dan berikut merupakan prosesnya



Gambar 11 *Proses Denial of Service pada Cowrie*

3.2. Hasil Pengujian Serangan

MHN akan menerima *log* serangan yang diterima *Honeypot* . Dari serangan dari sub bab pengujian, *Dionaea* menangkap banyak serangan terutama dari serangan berupa scanning yang dilakukan program Nmap. *Dionaea* juga menangkap serangan exploit yang dikirimkan oleh metasploit pada *Port* 445 dan serangan DoS yang dikirimkan pada *Port* 80 seperti yang terlihat pada **Gambar 12** dan **Gambar 13:**

	Date	Sensor	Country	Src IP	Dst port	Protocol	Honeypot
1	2020-06-28 02:16:58	dionaea	?	192.168.1.35	445	smbd	dionaea
2	2020-06-28 02:13:53	dionaea	?	192.168.1.35	8085	pcap	dionaea
3	2020-06-28 02:13:53	dionaea	?	192.168.1.35	1300	pcap	dionaea
4	2020-06-28 02:13:53	dionaea	?	192.168.1.35	4242	pcap	dionaea
5	2020-06-28 02:13:53	dionaea	?	192.168.1.35	1042	pcap	dionaea
6	2020-06-28 02:13:53	dionaea	?	192.168.1.35	90	pcap	dionaea
7	2020-06-28 02:13:53	dionaea	?	192.168.1.35	1055	pcap	dionaea
8	2020-06-28 02:13:53	dionaea	?	192.168.1.35	57797	pcap	dionaea
9	2020-06-28 02:13:53	dionaea	?	192.168.1.35	1914	pcap	dionaea
10	2020-06-28 02:13:53	dionaea	?	192.168.1.35	3690	pcap	dionaea

Gambar 12 *Serangan Nmap dan Metaslpoit pada Dionaea*

	Date	Sensor	Country	Src IP	Dst port	Protocol	Honeypot
1	2020-06-27 11:47:01	dionaea	?	192.168.1.35	80	httpd	dionaea

Gambar 13 *Serangan DoS pada Dionaea*

Sebaliknya dengan *Honeypot Cowrie* menangkap sedikit serangan yang telah dikirmkan. Bahkan pada suatu hari *Cowrie* hanya menangkap satu serangan sedangkan serangan yang dikirmkan ada tiga

	Date	Sensor	Country	Src IP	Dst port	Protocol	Honeypot
1	2020-06-27 11:39:35	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
2	2020-06-27 11:31:05	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
3	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
4	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
5	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
6	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
7	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
8	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
9	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie
10	2020-06-27 06:43:19	cowrie-panda	?	192.168.1.35	22	ssh	cowrie

Gambar 14 *Serangan yang diterima Cowrie*

Sensor: All | Honeypot: cowrie | Date: 06-28 | Port: 445 | IP Address: 192.168.1.35 | GO

	Date	Sensor	Country	Src IP	Dst port	Protocol	Honeypot
1	2020-06-28 02:17:26	cowrie-panda	?	192.168.1.35	22	ssh	cowrie

Gambar 15 *Serangan yang diterima hanya satu*

3.3. *Import log* dari MHN menuju *MySQL*

Sebelum melakukan visualisasi di grafana, data dari MHN akan di *import* terlebih dahulu menuju *MySQL*. Disini peneliti membuat sebuah program dengan bahasa Python yang berfungsi mengambil data dari database MHN dan mengirimnya menuju database *MySQL*. Program memiliki dua method yang pertama berfungsi mengambil data dari *MongoDB* dan menyimpannya dalam variable dan yang kedua berfungsi untuk membaca variable tersebut dan memasukkannya ke dalam *MySQL*. Berikut adalah screenshot yang digunakan

```
def ambilhasil():
      # ~ membuat koneksi dengan MongoDB
      koneksi = pymongo.MongoClient("mongodb://localhost:27017")
      # ~ memilih database dan tabel yang akan diambil recordnya
      database = koneksi["mnemosyne"]
      tabel = database["session"]

      # ~ mengambil record dimana kolom honeypot berisi cowrie dan dionaea
      query = {"honeypot": {"$in": ["cowrie", "dionaea"]}}
      # ~ memilih kolom yang akan diambil recordnya
      query2 = {"protocol":1, "timestamp":1, "source_ip":1,
"source_port":1, "destination_port":1, "identifier":1 ,"honeypot":1,
"_id":0}
      # ~ eksekusi query dan masukkan ke dalam object "hasil"
      hasil = tabel.find(query, query2)
      return hasil
      # ~ menutup koneksi
      koneksi.close()
```

Gambar 16 *method pertama pada program*

```
      # ~ membuat sebuah objek cursor agar bisa menjalankan query layaknya query pada
MySQL
      cursor = db.cursor()
      # ~ memasukkan record ke dalam tabel dengan kolom sebagai berikut
      # ~ jika terjadi duplikasi pada primary key maka akan update record tersebut
      sql = "INSERT INTO session (ID_serangan, tanggal, honeypot, protocol, port,
IP_penyerang)\
      VALUES(%s, %s, %s, %s, %s, %s)\
      ON DUPLICATE KEY UPDATE\
      tanggal=values(tanggal), honeypot=values(honeypot), protocol=values(protocol),
port=values(port), IP_penyerang=values(IP_penyerang);"

      # ~ untuk setiap isi dari query akan dilakukan pengulangan
      for index, isi in enumerate(query):
            ID = index+1
            # ~ merubah format tanggal dan zona yang didapat dari record MongoDB
            tanggal = isi['timestamp']
            gmt = datetime.timedelta(hours=7)
            tanggalgmt = tanggal + gmt
            tanggalbaru = tanggalgmt.strftime('%Y-%m-%d %H:%M:%S')

            honeypot = isi['honeypot']
            protocol = isi['protocol']
            destinasi_port = isi['destination_port']
            IP_penyerang = isi['source_ip']

            values = (ID, tanggalbaru, honeypot, protocol, destinasi_port,
IP_penyerang)

            # ~ eksekusi sql dengan record yang telah dimodifikasi
            cursor.execute(sql, values)
            db.commit()

      # ~ menutup koneksi agar tidak terjadi penumpukan sesi untuk MySQL
      cursor.close()
      db.close()
```

Gambar 17 *method kedua pada program*

Hasil dari program tersebut dapat dilihat pada database *MySQL*

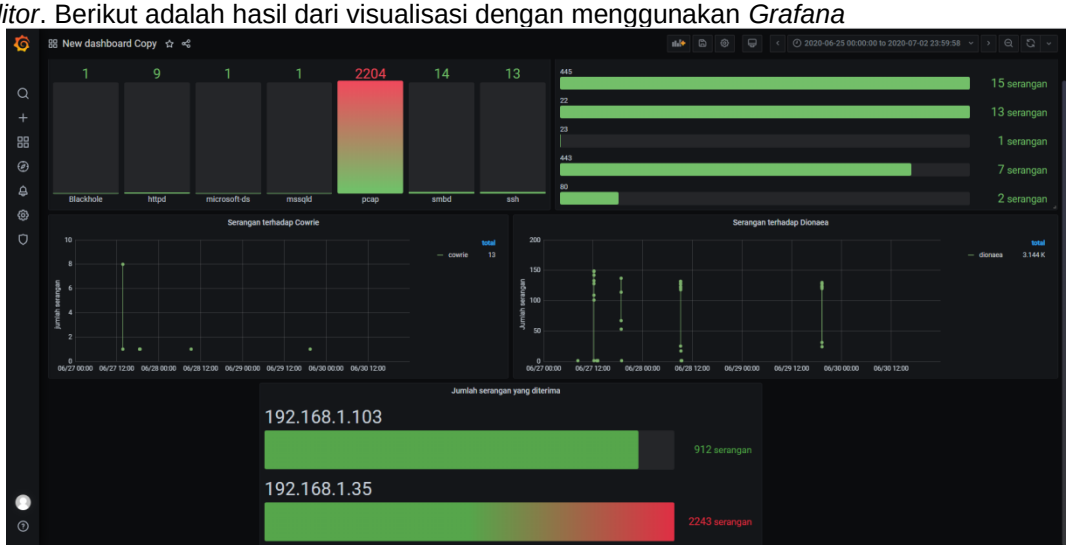
```
mysql> select * from session limit 10;
+-------------+---------------------+----------+----------+------+--------------+
| ID_serangan | tanggal             | honeypot | protocol | port | IP_penyerang |
+-------------+---------------------+----------+----------+------+--------------+
|           1 | 2020-06-27 13:43:07 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           2 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           3 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           4 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           5 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           6 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           7 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           8 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|           9 | 2020-06-27 13:43:19 | cowrie   | ssh      | 22   | 192.168.1.35 |
|          10 | 2020-06-27 18:31:05 | cowrie   | ssh      | 22   | 192.168.1.35 |
+-------------+---------------------+----------+----------+------+--------------+
10 rows in set (0.00 sec)

mysql>
```

Gambar 18 *Record hasil import dari MongoDB*

3.4 Visualisasi menggunakan Grafana

Tahap terakhir dari pengujian ini adalah memvisualisasikan data pada *MySQL* dengan *Grafana*. *Grafana* harus dihubungkan dengan sumber data terlebih dahulu, yakni database *MySQL* untuk mengambil data yang akan divisualisasikan. Setelah *Grafana* terhubung dengan *MySQL* berikutnya ialah membuat *dashboard* dan *panel. Dashboard* merupakan tempat diperlihatkannya

panel - panel, sedangkan *panel* adalah tempat untuk memvisualkan data yang diambil dari sumberdata. *Panel* menggunakan *Query* yang sama dengan sumber data untuk mengambil data. Pada panel terdapat *Query Editor* yang dapat membantu pengguna untuk menuliskan *Query* agar tampil pada panel. Pengguna dapat juga menuliskan *Query* sendiri tanpamenggunakan *Query Editor*. Berikut adalah hasil dari visualisasi dengan menggunakan *Grafana*



Gambar 19 *Hasil Visualisasi menggunakan Grafana*

pada **Gambar 20** terlihat *Grafana* menampilkan berbagai macam data seperti serangan terhadap *honeypot*, IP penyerang, *port* yang diserang serta *protocol* yang berjalan

4. Kesimpulan dan Saran

4.1 Kesimpulan

Berdasarkan dari hasil pengmplementasian dan pengujian sistem yang dibangun pada penelitian " *Modern Honey Network dengan Grafan untuk Visualisasi*" dapat ditarik kesimpulan antara lain

1. Dibutuhkan database lain untuk mengintegrasikan *Modern Honey Network* dengan *Grafana* dikarenakan *Grafana* tidak mendukung MongoDB sebagai sumber data
2. *Grafana* mampu mevisualisasikan data yang diberikan ke dalam bentuk grafik sesuai dengan keinginan peneliti

4.2 Saran

Berdasarkan dari hasil penelitian ini masih terdapat kekurangan sehingga memungkinkan digunakan sebagai acuan untuk pengemabangan penelitian selanjutnya

1. Dapat mengintegrasikan *Modern Honey Network* dan *Grafana* secara langsung tanpa menggunakan Database lain seperti *MySQL* atau Prometheus
2. Menambahkan jumlah *Honeypot* dan Sensor yang digunakan dalam penelitian untuk mendapatkan hasil yang lebih beragam.
3. Pemasangan *Honeypot* dan Sensor pada jaringan yang berbeda tetapi tetap bisa berkomunikasi dengan Modern Honey Network.

Refrensi

[1] U. Gasser, J. Zittrain, R. Faris, and R. H. Jones, "Internet Monitor 2014: Reflections on the Digital World: Platforms, Policy, Privacy, and Public Discourse," vol. 2014–17, p. 155, 2014.

[2] B. B. Gupta, D. P. Agrawal, and H. Wang, *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*, no. 9. Boca Raton: CRC Press, 2019.

[3] LanceSpitzner, *Honeypots: Tracking Hackers*, 2nd ed. Addison Wesley, 2002.

[4] P. Kenealy and W. J. Noonan, *Virtualization for Security*, 1st ed. Burlington: Syngress Publishing, Inc., 2009.

[5] S. M. Jigneshkumar, "Modern Honey Network," *Int. J. Res. Advent Technol.*, no. Special Issue, pp. 156–162, 2016.

[6] M. Mohammed and H. Rehman, *Honeypots and Routers, Collecting Internet Attacks*. Boca Raton: CRC Press, 2016.

[7] H. Wafi, A. Fiade, N. Hakiem, and R. B. Bahaweres, "Implementation of a Modern Security Systems Honeypot Honey Network on Wireless Networks," *Int. Young Eng. Forum*, pp. 91–96, 2017.

[8] "What is Grafana? | Grafana Labs." [Online]. Available: https://grafana.com/docs/grafana/latest/guides/what-is-grafana/. [Accessed: 04-Apr-2020].

[9] V. Bontchev and V. Yosifova, "Analysis of the Global Attack Landscape Using Data from a Telnet Honeypot," vol. 43, no. 2, pp. 264–282, 2019.

[10] R. S. Pressman, *Book review: Software Engineering: a Practitioner's Approach*, 5th ed., vol. 10, no. 6. Thomas Cason, 1995.