BAB IV
PERANCANGAN

Penelitian ini menciptakan suatu proses perancangan untuk aplikasi yang akan dibangun. Perencanaan aplikasi sangat penting karena merupakan titik awal untuk melakukan pengembangan. Jika perencanaan sudah dilakukan dengan baik, pengembangan aplikasi akan dapat tercapai.

4.1 Analisis Sistem Terdahulu

Analisis sistem sebelumnya bertujuan untuk menganalisis fitur, kelebihan, kekurangan, dan kebutuhan sistem sebelumnya. Pada tahap ini, peneliti dapat menggunakan observasi untuk memahami apa yang dibutuhkan sistem. Analisis sistem pendahuluan memungkinkan perubahan sistem dikembangkan secara andal dan menutupi kekurangan pada sistem sebelumnya.

Pada penelitian yang dilakukan oleh Wiguna *et al* (2020), melakukan implementasi modsecurity pada ubuntu yang berfokus pada pencegahan serangan pada website. Topik yang diteliti pada sistem terdahulu yaitu pengamanan terhadap serangan *command injection*. Maka dibutuhkanlah aplikasi *monitoring* serangan dengan peringatan sistem deteksi intrusi menggunakan telegram *bot*.

4.2 Spesifikasi Kebutuhan Sistem Baru

Setelah menemukan hasil dari analisis pemasalahan tersebut, diperlukannya kebutuhan sistem baru dari sistem terdahulunya. Sistem baru ini dibutuhkan untuk membantu penerapan sistem keamanan untuk mencapai tujuan penelitian. Berikut merupakan kebutuhan spesifikasi sistem.

4.2.1 Spesifikasi Kebutuhan Perangkat Lunak

Dalam mengatasi masalah dalam sistem sebelumnya dan mengembangkan sistem aplikasi monitoring serangan pada aplikasi *web*, diperlukan beberapa kebutuhan perangkat lunak yang harus disiapkan agar proses pengembangan sistem dapat berjalan dengan baik. Beberapa spesifikasi yang harus dimiliki oleh perangkat lunak pendukung pengembangan sistem ini tercantum dalam tabel di bawah ini.

Tabel 4. 1 Tabel kebutuhan piranti lunak

No	Kebutuhan *software*	Keterangan
1	*Proxmox Virtual Environment*	Sebagai sistem operasi *server development*.
2	Sistem Operasi *Ubuntu-LiveServer* 22.04 LTS	Sebagai sistem operasi *server modsecurity*.
3	*Web Browser*	Sebagai alat yang digunakan untuk menampilkan halaman web.
4	*Mikrotik Cloud*	Berfungsi untuk meneruskan jaringan agar bisa diakses dari luar.
5	*Visual Studio Code*	Sebagai pendukung pengembangan perangkat lunak.
6	ZAP-OWASP	Sebagai *tools* yang berfungsi untuk *scanning* dan *attacking.*

Tabel 4.1 merupakan kebutuhan piranti lunak yang digunakan peneliti untuk mengembangan sistem ini. Proxmox VE digunakan untuk menampung mesin virtual. Ubuntu LiveServer digunakan untuk dijadikan sebagai webserver dan penempatan WAF. Web Browser digunakan untuk mengoperasikan proxmox dan aplikasi monitoring. Sistem keamanan jaringan menggunakan Mikrotik Cloud yang diaktifkan fitur VPN dan Port Forwarding untuk menambahkan lapisan keamanan. Lingkup pengembangan skrip dan aplikasi menggunakan Visual Studio Code. Website di scanning dan penetration testing menggunakan ZAP-OWASP untuk menguji WAF yang dipasang.

4.2.2 Spesifikasi Kebutuhan Perangkat Keras

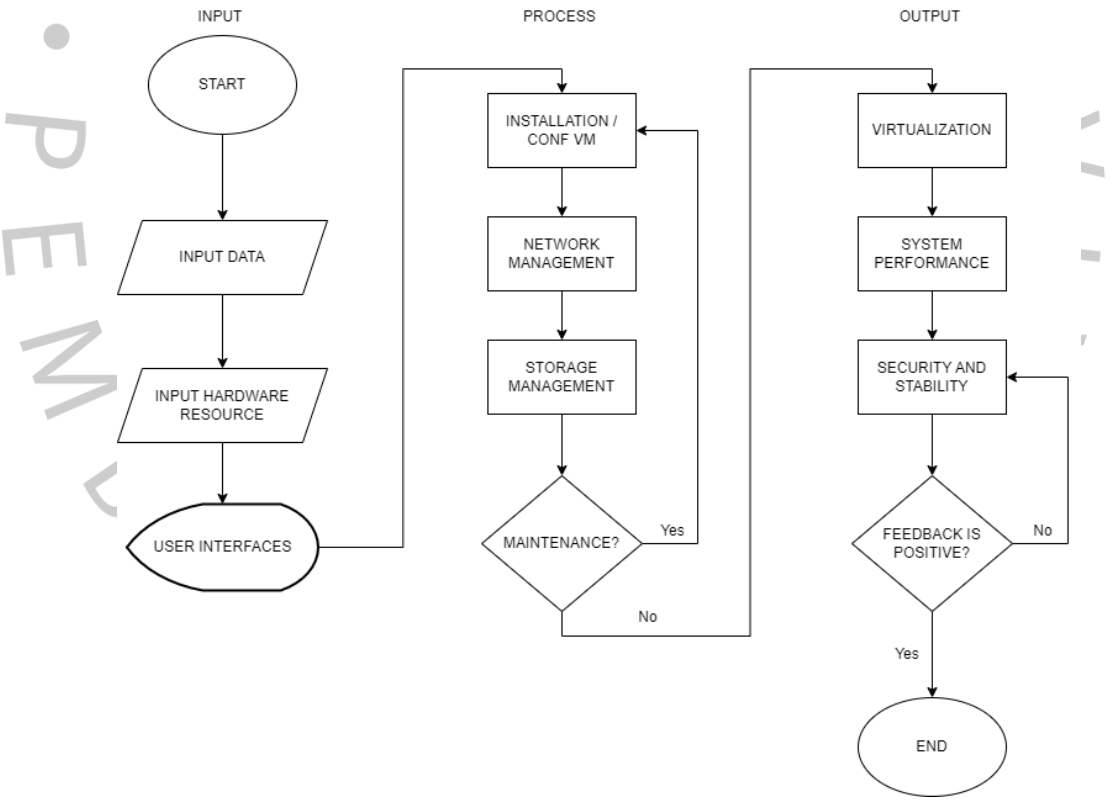
Perangkat keras mendukung proses perancangan sistem aplikasi monitoring serangan dan konfigurasi ModSecurity, selain kebutuhan perangkat lunak, juga dibutuhkan perangkat keras yang memadai agar proses yang dihasilkan dari mesin atau komputer yang digunakan dapat berjalan dengan

optimal dan tidak mengalami hambatan. Dalam hal ini, dibutuhkan spesifikasi perangkat keras tertentu untuk mendukung pengembangan sistem ini, yang tercantum dalam daftar berikut.

Tabel 4. 2 Kebutuhan *hardware*

No	*Hardware*	Spesifikasi
1	*Processor*	Intel Core I3-12100F
2	RAM	16 GB
3	Storage	SSD 256GB & 30GB VHD
4	*Display*	24 inch 144Hz
5	*Router*	Mikrotik Rb750

4.2.3 Proses Instalasi Server



Gambar 4. 1 *Flowchart* Instalasi *Server*

Gambar 4.1 merupakan *flowchart* instalasi *server* dengan menggunakan *Proxmox Virtual Environtment.* Proses membuat *server* meliputi *input*, proses, dan *output*, bagian ini merupakan bagian terpenting untuk *deployment server*. Tahap

inisialisasi adalah input data yang memasukkan permintaan pembuatan *virtual machine*, *update* data, dan pembuatan *backup*, lalu *input hardware resource* yaitu memasukkan spesifikasi *hardware* yang digunakan untuk *Proxmox VE*, jika sudah *server* akan menampilkan tampilan *proxmox* di *browser*.

Selanjutnya, ketika *proxmox* telah berhasil dibuat maka diarahkan untuk pembuatan *virtual machine* didalam *proxmox*, sistem operasi yang digunakan adalah Ubuntu *Live Server* 22.04 berbasis *CLI* (*Command Line Interfaces*) diakses bisa melalui *console proxmox* atau *SSH* (*Secure Shell*) pada *Windows* atau sejenisnya. Setelah sistem operasi sudah ditentukan pengguna juga harus mengonfigurasi jaringan yang digunakan dan media penyimpanan VHD (*Virtual Hard Disk*) jika sudah maka lakukan instalasi yang dibutuhkan seperti *Apache2 Server*, SSH *Client*, *Modsecurity*, *Python*, *MySQL*, *Django*, dan yang dibutuhkan lainnya. Jika terjadi kendala pada salah satu instalasi maka pengguna harus mengonfigurasi ulang pada Ubuntu tersebut dan jika tidak ada masalah maka pengguna dapat menggunakan *virtual machine* tersebut.

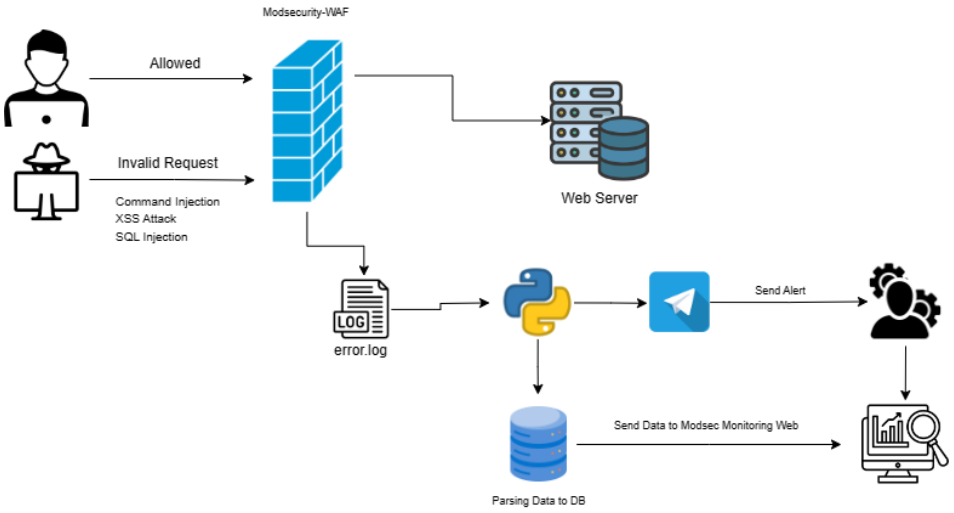
Output yang didapatkan pengguna jika sudah melewati proses sebelumnya yaitu mendapatkan virtualisasi mesin yang diinginkan dan bisa melihat secara performa apakah sesuai dengan kebutuhkan, serta stabilitas dan keamanan sesuai konfirgurasi. Umpan balik yang didapatkan pengguna bisa berupa *log* atau catatan vm, notifikasi melalui *web console*, dan *monitoring tools* seperti Grafana.

4.3 Perancangan Sistem

Bagian perancangan sistem merupakan bagian penting dalam menyajikan rancangan teknis dan bentuk sistem yang akan dibangun. Rancangan ini meliputi arsitektur sistem, *use case* diagram, *activity* diagram, *sequence* diagram, dan rancangan antar-muka.

4.3.1 Arsitektur Sistem

Modsecurity dan *Telegram* digunakan sebagai dasar penelitian ini. Perancangan sistem ini akan terus memantau paket data yang mencurigakan dan aliran data yang masuk ke *server* komputer melalui *modsecurity log*. Gambar 4.3 dibawah ini mengilustrasikan dan menjelaskan dari arsitektur sistem yang dibuat.

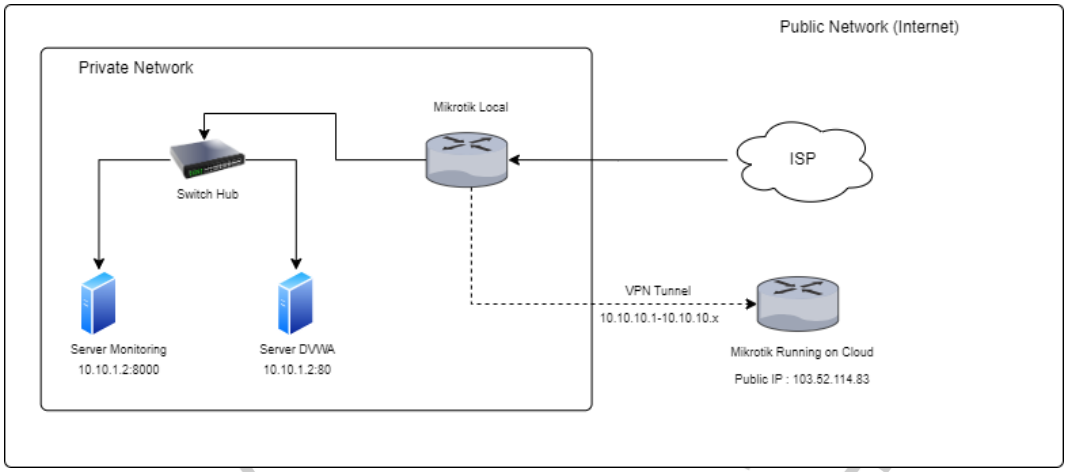


Gambar 4. 2 Arsitektur Sistem

Pertama, *secure user* (tidak membawa paket mencurigakan) akan melewati proses *screening* oleh *modsecurity.* Jika terbukti aman maka user tersebut diizinkan memasuki *webserver*. Jika ada user tidak dikenal dan dicurigai membawa paket yang berbahaya (serangan siber) maka permintaan akses akan diblokir dan informasi itu akan masuk ke dalam *error*. *Log* dan skrip *Telegram* akan mengolah informasi tersebut menjadi 2 yaitu informasi akan diekstrak kemudian dimasukkan ke dalam *database* dan mengirim notifikasi ke telegram jika adanya intrusi terhadap sistem. *Administrator* akan mendapatkan pesan dari sistem bahwa adanya serangan dan telah diblokir dan *administrator* dapat melihat catatan serangan pada aplikasi *monitoring*.

4.3.2 Topologi Jaringan

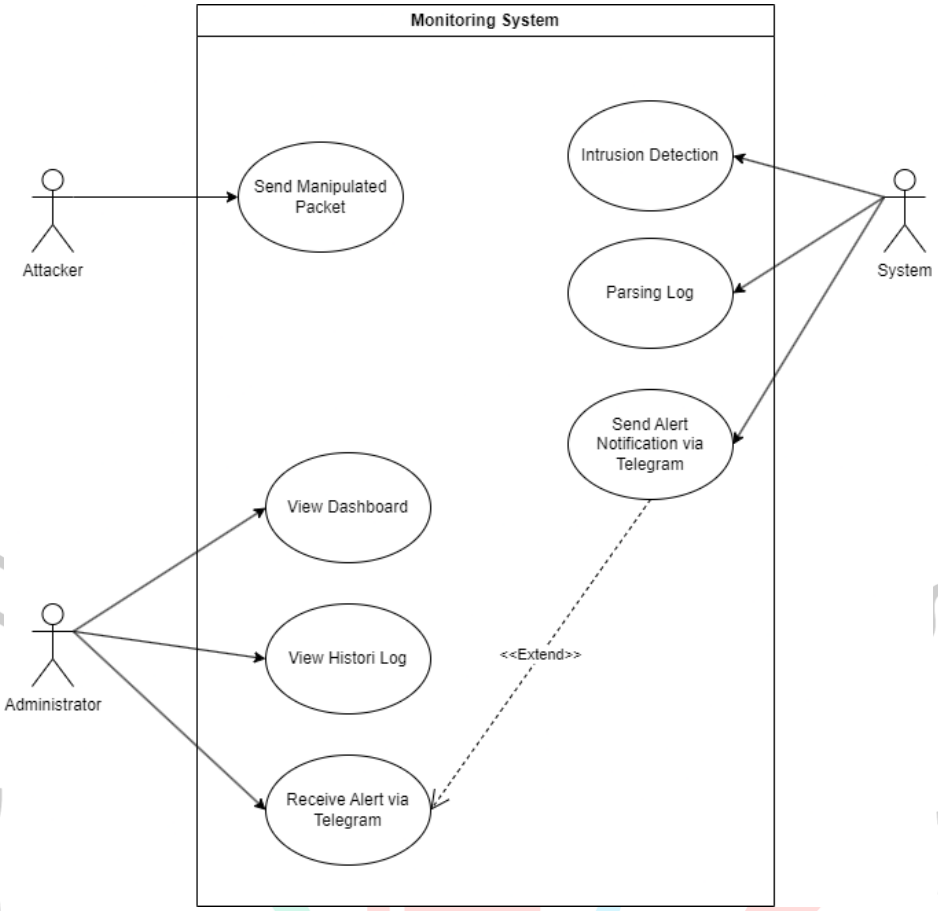
Mikrotik merupakan komponen penting untuk mendukung *web server* agar bisa diteruskan atau diakses dari jaringan luar. *Router* ini mempunyai fitur *port forwading* yang dapat membuka *port web server* yang dijalankan, dengan catatan samarkan *port* agar tidak disalahgunakan oleh pihak lain.



Gambar 4. 3 Topologi Jaringan

Gambar 4.4 merupakan topologi jaringan yang dirancang untuk memenuhi kebutuhan dalam penelitian ini. Topologi ini diawali dengan menghubungkan jaringan *internet* ke *router* mikrotik lokal untuk dapat menyebarkan *internet* ke setiap perangkat. Jika internet sudah terhubung maka tugas *router* meneruskan koneksi internet melalui *switch* ke perangkat *server* (*proxmox*) hingga mendapatkan IP (*Internet Protocol*), lalu *server* dapat dikonfigurasi dan mengunduh paket-paket yang diperlukan. Selanjutnya *router* mikrotik lokal dihubungkan dengan *Mikrotik Cloud* melalui VPN *Tunnel* supaya *server* lokal dapat diakses melalui jaringan luar dengan IP *Public* yang dimiliki oleh *Mikrotik Cloud*. Pengguna dapat mengakses dari luar jaringan dengan memasukkan *IP Public* dan *Port* yang sudah disesuaikan keamanannya. *IP Public Mikrotik Cloud* ini mudah dan efisien secara pengunaan dan biaya karena harga VPS (*Virtual Private Server*) lebih murah dari penyedia layanan lain.

4.3.3 Use Case



Gambar 4. 4 *Use Case* Diagram

Pada Gambar 4.5 *use case* ini menjelaskan, ada 3 (tiga) *actor* dalam *monitoring system* ini. *Attacker* berperan menyerang *website* dengan memasukkan paket yang telah dimanipulasi, sistem akan mendeteksi intrusi yang disematkan oleh *attacker* dan menyimpan catatan serangan ke dalam *database* serta mengirimkan notifikasi kepada *administrator* sebagai peringatan bahwa terjadinya *attackeran*. *Administrator* mempunyai peran untuk melihat secara visual informasi serangan terhadap *web* yang dimiliki dan *administrator* mendapatkan notifikasi telegram yang dikirim oleh sistem.

4.3.4 Use Case Narrative

Use case Narrative digunakan untuk mendokumentasikan u*se case* yang ada dalam pengembangan aplikasi. Berikut u*se case narrative* yang dibuat berdasarkan perancangan.

1. *Use Case Narrative Send Manipulated Packet*

Tabel 4. 3 *Use Case Narrative Send Manipulated Packet*

Use case name	*Send Manipulated Packet*	
Actor	*Attacker*	
Description	*Use case* ini menjelaskan tentang bagaimana attacker melakukan manipuilasi paket untuk menyerang sistem.	
Precondition	Sistem menunjukkan halaman aplikasi untuk objek serang	
Typical Course of Event	*Action*	*System Response*
	1. *Attacker* melakukan akses objek 2. Attacker melakukan Serangan	1.1 Sistem akan menampilkan halaman yang berisikan inputan data seperti sql dan command
Alternate Course	Tampilan tidak akan muncul bila belum akses web melakukan aktivitas dengan klik halaman yang tersedia	
Post Condition	*Attacker* tidak berhasil memanipulasi paket karena terblokir oleh modsecurity	

2. *Use Case Narrative Intrusion Detection*

Tabel 4. 4 *Use Case Narrative Dashboard*

Use case name	*Intrusion Detection*	
Actor	*System*	
Description	*Use case* ini menjelaskan tentang bagaimana sistem mendeteksi adanya serangan	
Precondition	Sistem sudah mengaktifkan modsecurity	
Typical Course of Event	*Action*	*System Response*
	1. Fitur modsecurity diaktifkan melalui apache2.conf 2. Sistem mencatat log di error.log	1.1 Sistem akan menyimpan log di error.log yang belum dipilah alias log dengan catatan lengkap
Alternate Course	Log tidak akan tercatat apabila belum mendefinisikan tempat penyimpanan log.	

Post Condition	Web tetap terjaga dari serangan yang diblokir modsecurity

3. *Use Case Narrative Parsing Log*

Tabel 4. 5 *Use Case Narrative Dashboard*

Use case name	*Parsing Log*	
Actor	*System*	
Description	*Use case* ini menjelaskan tentang bagaimana melakukan parsing log	
Precondition	Sistem sudah memiliki skrip aktif untuk parsing log yang sudah terintegrasi dengan mysql.	
Typical Course of Event	*Action*	*System Response*
	1. Skrip dijalan melalui daemon-service	1.1 Sistem akan memilah log sesuai kriteria untuk disimpan kedalam database
Alternate Course	Parsing log tidak akan berjalan jika skrip belum dijalankan dalam daemon-service	
Post Condition	Log yang berisikan catatan modsecurity yang sudah dipilah dan berhasil tersimpan dalam database	

4. *Use Case Narrative Send Alert*

Tabel 4.6 *Use Case Narrative Send Alert*

Use case name	*Send Alert*	
Actor	*System*	
Description	*Use case* ini menjelaskan tentang bagaimana sistem melakukan pengiriman notifikasi serangan.	
Precondition	Sistem sudah menajalan skrip bot di server	
Typical Course of Event	*Action*	*System Response*
	1. Skrip dijalankan melalui daemon-service	1.1 Sistem akan mengirimkan id dan token ke Telegram Gateway dan mengirimkan pesan sesuai id yang disematkan.

Alternate Course	Notifikasi tidak akan muncul jika log belum dibersihkan sebelumnya
Post Condition	*Administrator* mendapatkan notifikasi melalui Telegram

5. *Use Case Narrative Dashboard*

Tabel 4.7 *Use Case Narrative Dashboard*

Use case name	*Dashboard*	
Actor	*Administrator*	
Description	*Use case* ini menjelaskan tentang bagaimana administrator melakukan monitoring serangan	
Precondition	Sistem menunjukkan halaman utama	
Typical Course of Event	*Action*	*System Response*
	1. *User admin* melakukan akses halaman utama 2. User Admin melakukan klik *Dashboard*	1.1 Sistem akan menampilkan halaman yang berisikan data-data hasil serangan yang masuk dan menyajikan informasi terkait informasi keamanan
Alternate Course	Tampilan tidak akan muncul bila administrator melakukan aktivitas dengan klik halaman yang tersedia	
Post Condition	*User* admin dapat melihat informasi modsecurity	

6. *Use Case Narrative* Histori Log

Tabel 4.8 *Use Case Narrative* Histori Log

Use case name	*Histori Log*	
Actor	*Administrator*	
Description	Use case ini menjelaskan tentang bagaimana admin melakukan Histori Log	
Precondition	Sistem menunjukkan halaman utama	
Typical Course of Event	*Action*	*System Response*
	1. *User* melakukan klik filter	1.1 Sistem menampilkan log dan dapat mencari log yang dipilih oleh user

Alternate Course	Administrator bisa melakukan monitoring data tanpa menambahkan filter	
Post Condition	*User* admin sukses melakukan filter	

7. Use Case Narrative Receive Alert

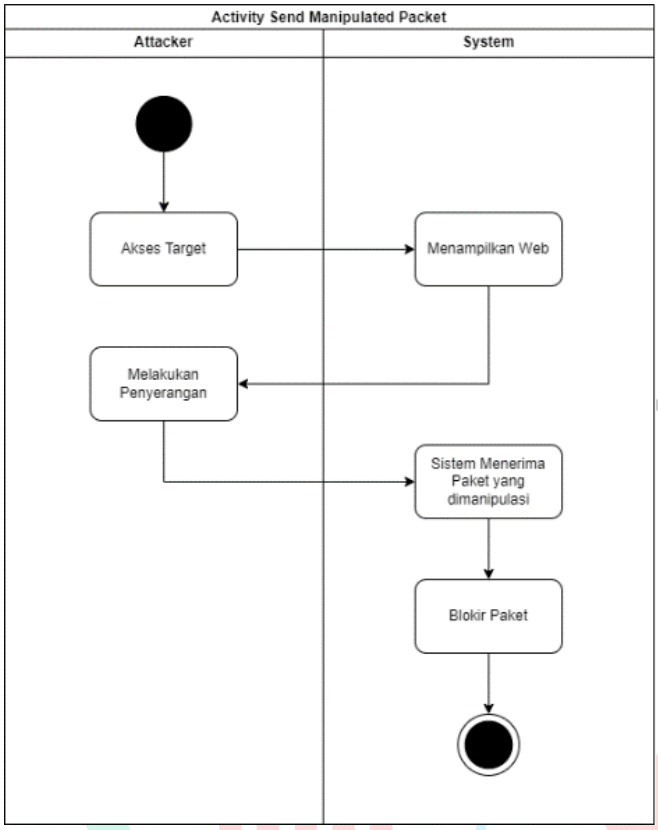
Tabel 4.9 *Use Case Narrative Receive Alert*

Use case name	*Receive Alert*	
Actor	*Administrator*	
Description	Use case ini menjelaskan tentang bagaimana admin mendapatkan notifikasi intrusi melalui telegram	
Precondition	Sistem WAF sudah berjalan	
Typical Course of Event	*Action*	*System Response*
	1. Serangan masuk ke server	1.1. Sistem akan memblokir, mencatat, dan mengirimkan notifikasi ke Administrator
Alternate Course	Administrator bisa melakukan monitoring data tanpa akses web	
Post Condition	Administrator mendapatkan notifikasi di telegram	

4.3.5 Activity Diagram

Activity diagram ini merupakan diagram yang menjelaskan alur admin dalam menggunakan aplikasi monitoring. Berikut gambar dibawah ini yang sudah dibuat.

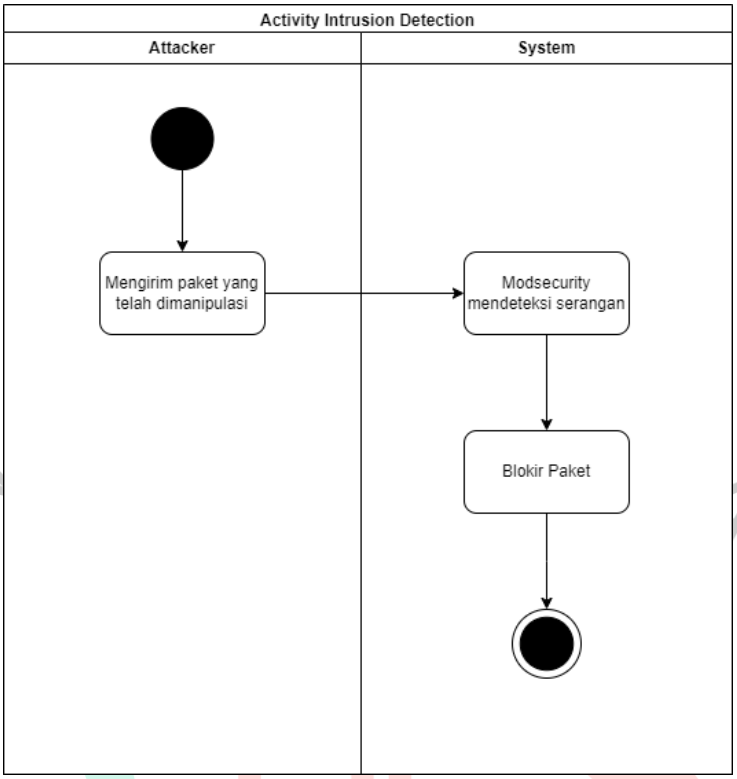
1. *Activity Diagram Send Manipulated Packet*



Gambar 4. 5 *Activity Diagram Send Manipulated Packet*

Pada Gambar 4.6 menjelaskan *Attacker* sudah mendapatkan akses ke sistem *target*. Setelah diakses, *attacker* dapat memanipulasi paket dengan mengubah konten atau *header*-nya. Misalnya, *attacker* dapat memasukkan kode berbahaya ke dalam paket untuk mengambil kendali sistem atau mengelabui sistem dengan menyembunyikan sumber atau tujuan paket. Sistem dapat memblokir paket yang dirusak untuk mencegah terjadinya kerusakan.

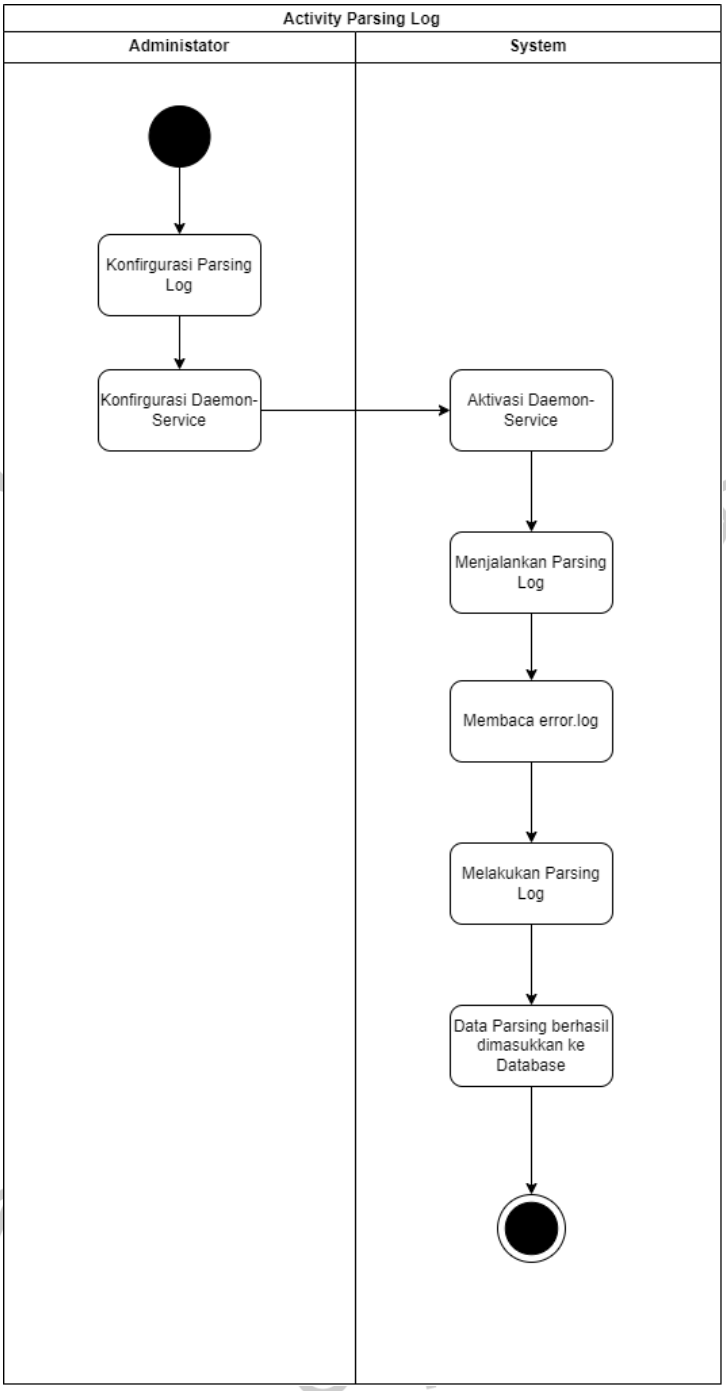
2. *Activity Diagram Intrusion Detection*



Gambar 4. 6 *Activity Diagram Intrusion Detection*

Pada Gambar 4.7 menjelaskan tentang *modsecurity* memblokir serangan yang dilakukan oleh *attacker*, jika paket tersebut terdeteksi sebagai paket yang tidak sah maka *modsecurity* secara langsung memblokir akses tersebut.

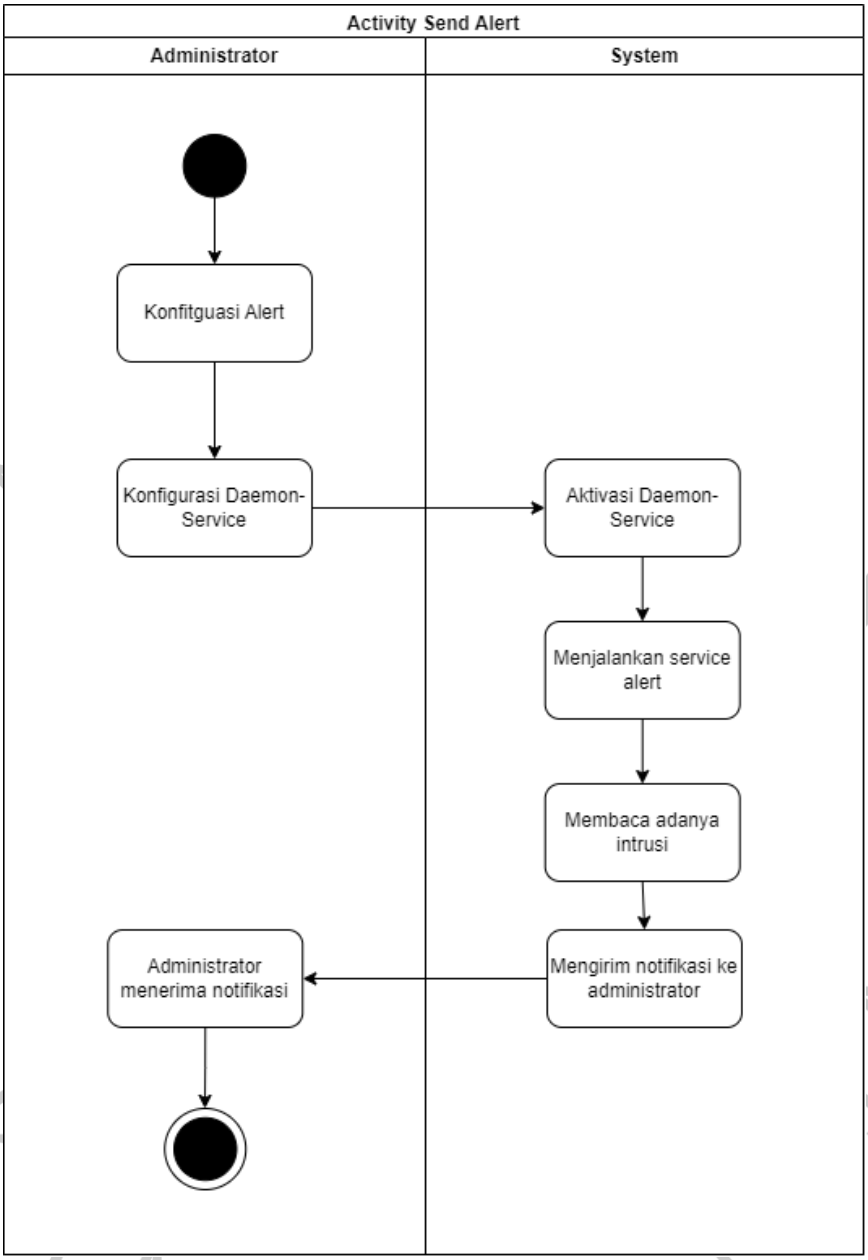
3. *Activity Send Parsing Log*



Gambar 4. 7 *Activity Diagram Dashboard*

Pada Gambar 4.8 menjelaskan memilah catatan yang ditulis oleh *modsecurity* untuk dimasukkan ke dalam *database*. Parsing dilakukan setelah *administrator* mengonfigurasi program parsing dan *daemon-service* di *server*, lalu sistem akan menjalankan program, membaca *log*, memilah *log*, dan menyimpannya ke dalam *database*.

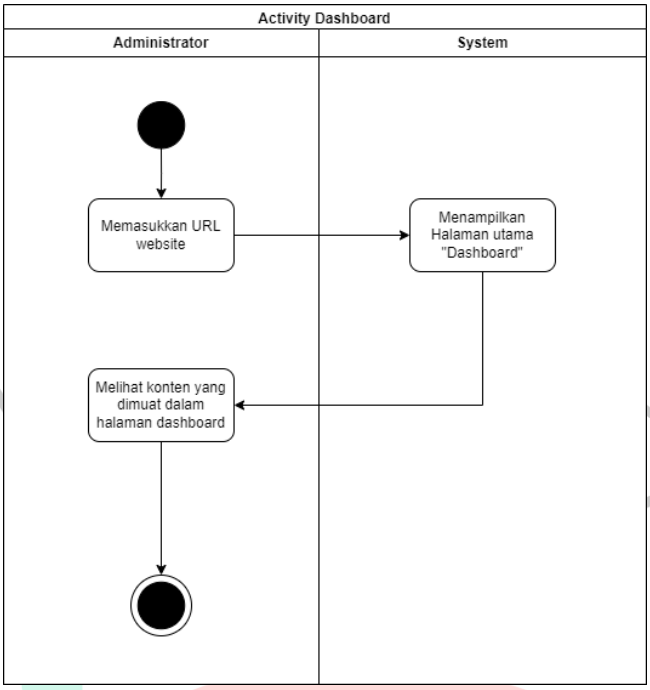
4. *Activity Diagram Send Alert*



Gambar 4. 8 *Activity Diagram Send Alert*

Pada Gambar 4.9 menjelaskan sistem mengirim notifikasi ke *administrator* menggunakan Telegram. *Administrator* akan mengonfigurasi Telegram *Bot* dan *daemon-service* agar sistem dapat berjalan secara *background*. Setelah itu sistem menjalankan program, membaca *log* apabila tercetak intrusi maka sistem akan mengirimkan notifikasi hingga pesan tersebut sampai kepada *adminstrator*.

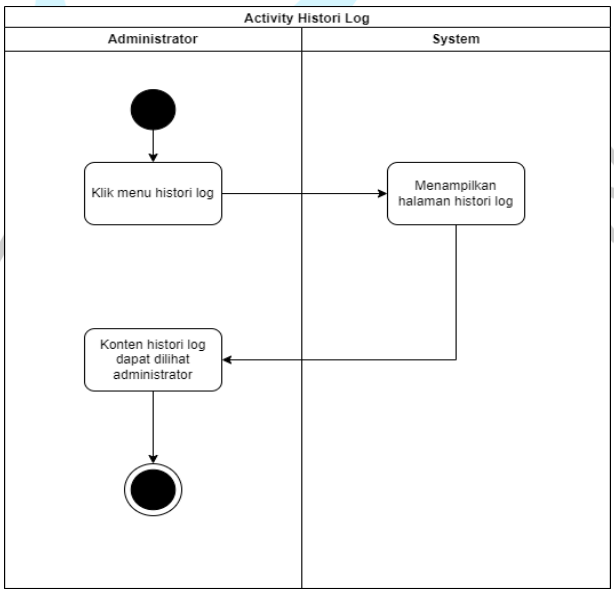
5. *Activity Dashboard*



Gambar 4. 9 *Activity Diagram Dashboard*

Pada Gambar 4.10 menjelaskan tentang kondisi user *admin* dapat melihat visualisasi data dalam bentuk grafik yang disediakan.

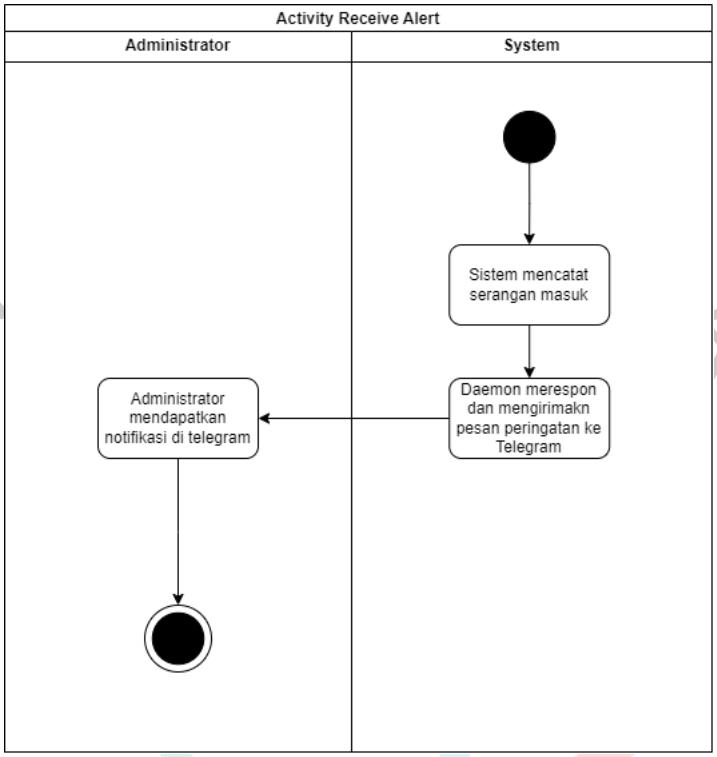
6. *Activity Histori Log*



Gambar 4. 10 *Activity Diagram Histori Log*

Aktivitas kedua dijelaskan pada Gambar 4.11 Diagram aktivitas “Histori *Log*” menunjukkan admin dapat melihat dan memfilter identitas dan jenis serangan yang sudah diblokir oleh *Modsecurity*.

7. *Activity Receive Alert*



Gambar 4. 11 *Activity Diagram Receive Alert*

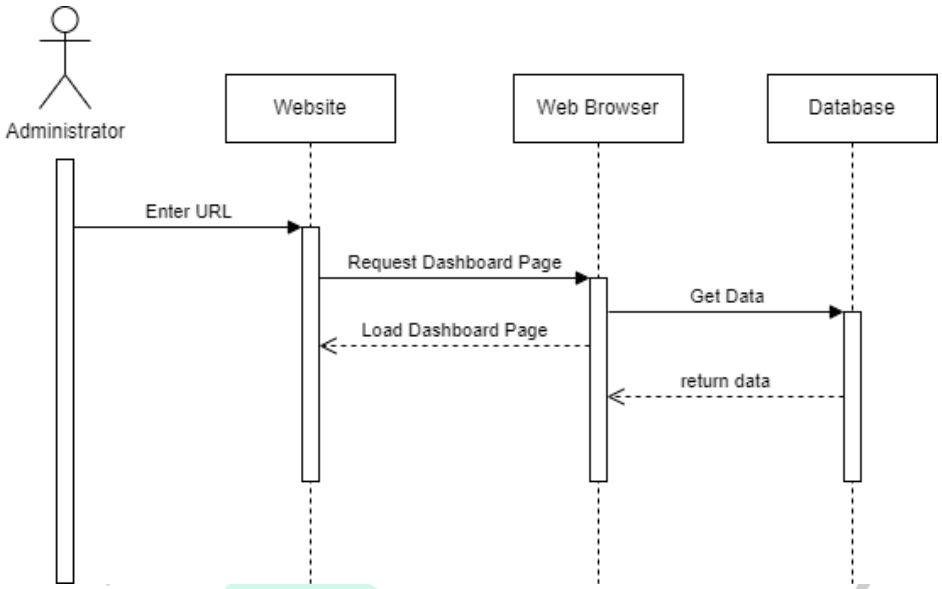
Aktivitas terakhir dijelaskan pada Gambar 4.12 Diagram aktivitas “*Receive Alert*” menunjukkan admin mendapat pesan notifikasi intrusi yang dikirimkan melalui aplikasi pesan daring telegram.

4.3.6 Sequence Diagram

Berikut merupakan Sequece Diagram yang telah dibuat berdasarkan pengembangan yang terdapat pada arsitektur sistem.

1. *Sequence Diagram* Halaman *Dashboard*

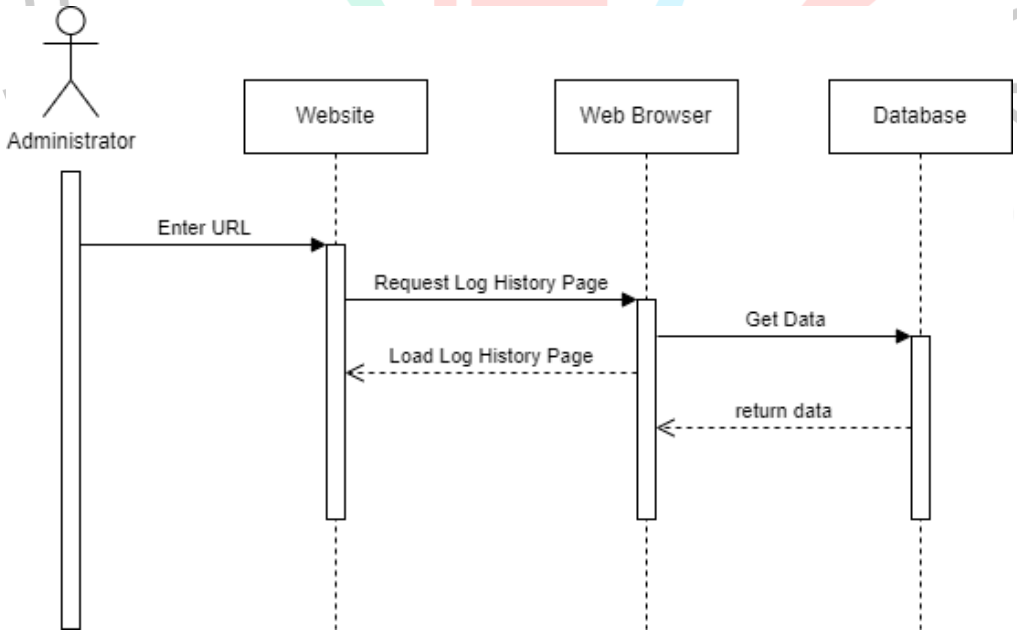
Sequence diagram dibawah ini merupakan rangkaian langkah yang terjadi saat pengguna membuka halaman *dashboard*. Objek-objek ini terdiri dari *Website*, *Web Browser*, *Database*, dan bagaimana permintaan dikirim untuk dapat memuat halaman *dashboard*.



Gambar 4. 12 *Sequence diagram dashboard*

2. Sequence Diagram Halaman Histori Log

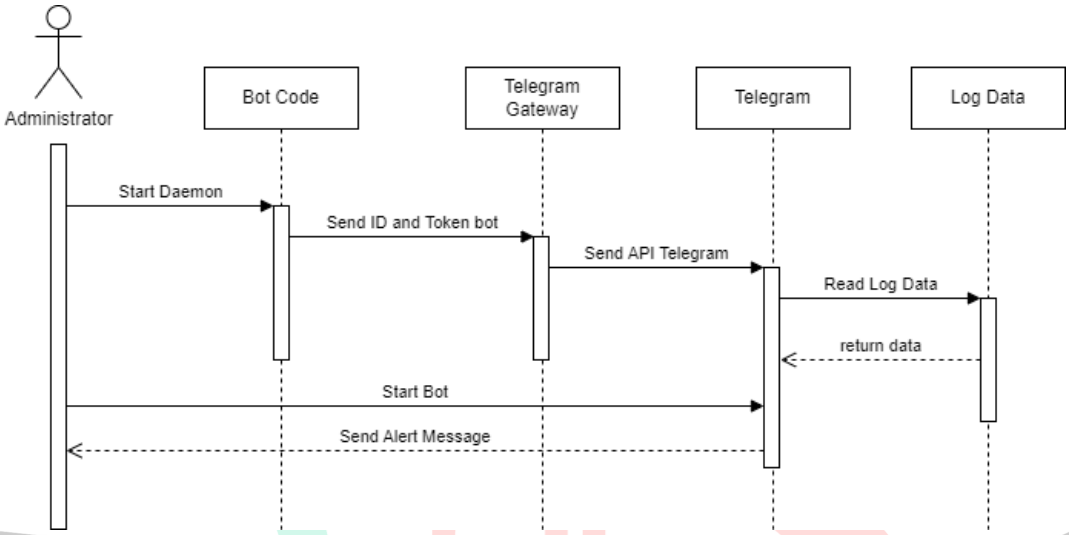
Sequence diagram dibawah ini merupakan rangkaian langkah yang terjadi saat pengguna membuka halaman histori log. Objek-objek ini terdiri dari Website, Web Browser, Database, dan bagaimana permintaan dikirim untuk dapat memuat halaman histori log.



Gambar 4. 13 *Sequence* diagram histori log

3. *Sequence* Diagram Telegram Bot

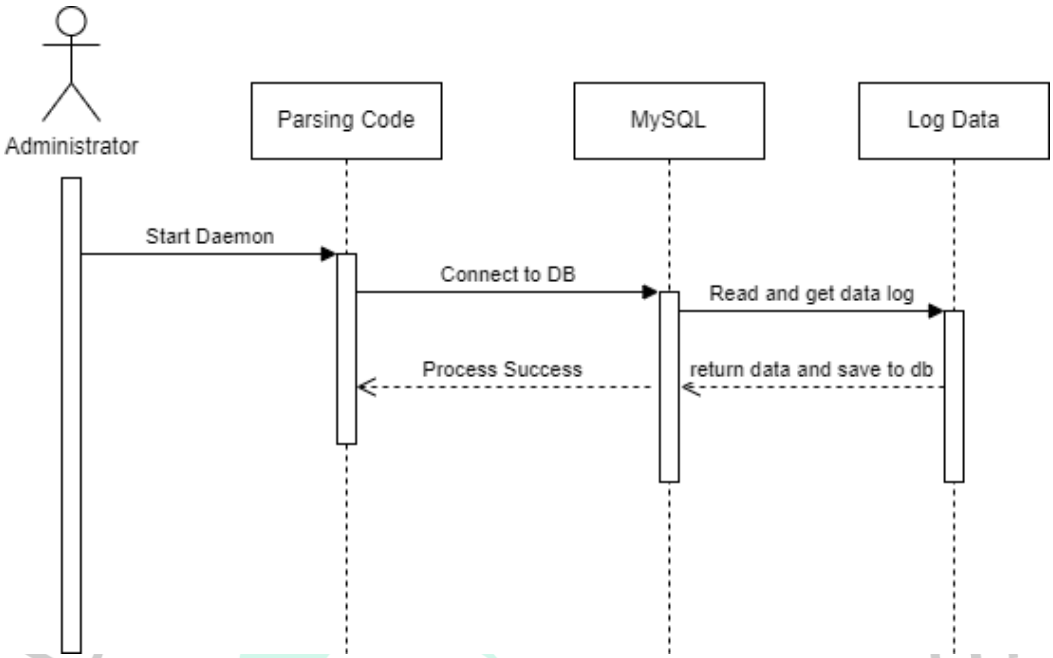
Sequence diagram dibawah ini merupakan rangkaian langkah Telegram *Bot* berkeja. Objek-objek ini terdiri dari kode *bot*, telegram *gateway*, telegram, dan *log* data yang di *server*. Ini membentuk serangkaian proses untuk mengirim dan menerima pesan notifikasi intrusi kepada pengguna.



Gambar 4. 14 *Sequence diagram telegram bot*

4. Sequence Diagram Parsing Log

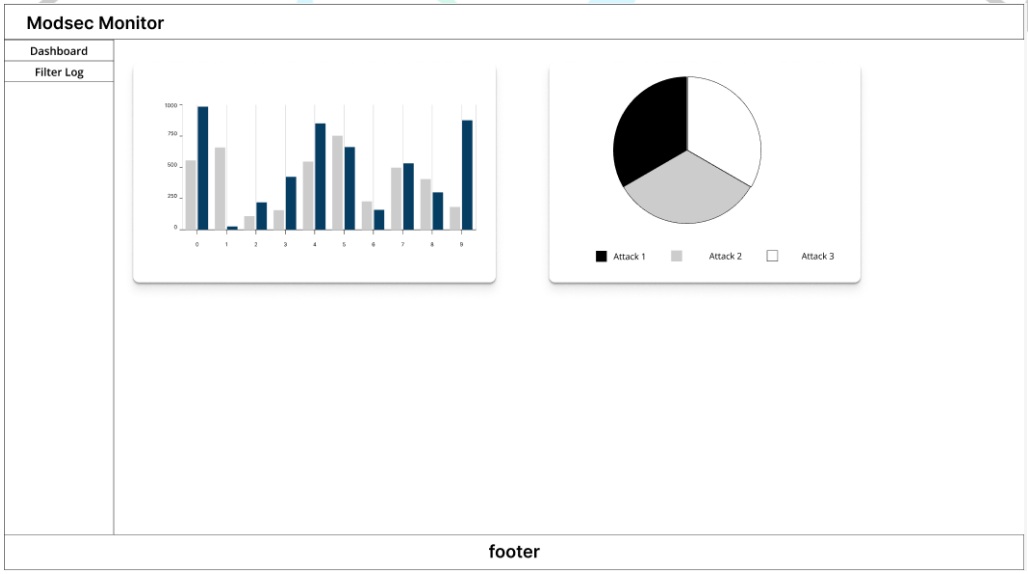
Sequence diagram dibawah ini merupakan rangkaian langkah yang terjadi saat pengguna melakukan *parsing log*. Objek-objek ini terdiri dari kode parsing, mysql, dan log data. Rangkaian ini membentuk proses Dimana kode yang dijalan dapat memilah data dari log yang digunakan oleh web server.



Gambar 4. 15 *Sequence diagram parsing log*

4.3.7 Rancangan Antar Muka

UI atau antarmuka pengguna adalah antarmuka yang digunakan oleh pengguna untuk berinteraksi dengan suatu sistem atau aplikasi. Tujuan dari *UI* adalah untuk menyederhanakan dan memudahkan interaksi antara pengguna dan sistem, sehingga lebih intuitif dan efisien.



Gambar 4. 16 *Halaman Dashboard*

Gambar 4.17 diatas menunjukkan pengguna ketika sudah berhasil memasuki aplikasi akan disajikan halaman utama (*dashboard*), pengguna dapat

melihat seluruh informasi modsecurity dan status server di halaman utama (dashboard).

Gambar 4. 17 Halaman *Filter*

Gambar 4.18 menunjukkan pengguna dapat memfilter serangan yang didapat kepada aplikasi *web* yang sudah diamankan menggunakan *modsecurity*. Halaman ini bertujuan agar pengguna dapat menentukan kriteria serangan yang didapat.

4.3.8 Rancangan Pengujian

Perancangan pengujian adalah tahapan dalam pengembangan perangkat lunak atau sistem yang meliputi perencanaan dan perancangan rencana pengujian yang sistematis dan terperinci. Tujuannya adalah untuk menguji perangkat lunak atau sistem secara menyeluruh. Dalam penelitian ini, perancangan pengujian dilakukan dengan menggunakan metode pengujian *white box* dan *black box*.

4.3.8.1 Rancangan Pengujian *Black Box*

Metode pengujian *black box* digunakan untuk menguji fungsi dari rancangan aplikasi *web* dan objek yang terpasang *ModSecurity* apakah fungsi sistem berjalan sesuai yang diharapkan atau tidak. Berikut merupakan metode pengujian *black box* .

Tabel 4. 6 Skenario Pengujian *Black Box*

No.	Skenario Pengujian	Hasil yang diharapkan
1.	Pengguna mengakses aplikasi web	Dapat masuk ke dalam aplikasi *web*
2.	Pengguna mengakses objek	Masuk ke dalam *web* objek dengan menggunakan alamat *IP*
3.	Menguji Serangan melalui objek	*Modsecurity* dapat menangkal serangan
4.	Menguji Skrip *Parsing Log*	*Log Modsecurity* dapat masuk ke dalam database
5.	Menguji Skrip *Telegram Bot*	Telegram dapat mengirimkan notifikasi intrusi

4.3.8.2 Rancangan Pengujian *White Box*

Metode pengujian *white box* digunakan untuk menguji fungsi dari *OBJEK* dengan *ModSecurity* apakah *rules* yang diterapkan berjalan sesuai yang diharapkan atau tidak. Berikut merupakan metode pengujian *white box*.

Tabel 4. 7 Skenario Pengujian *White Box*

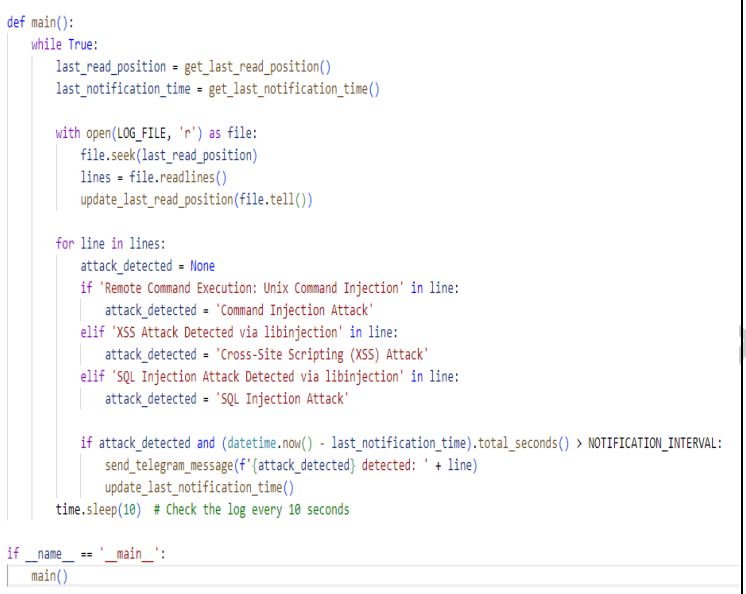
No.	Skenario Pengujian	Hasil yang diharapkan

1.	`class IndexView(APIView):`
	`  def get(self, request, *args, **kwargs):`
	`  return render(request, "theme/index.html", {})`
	
	`  def post(self, request, *args, **kwargs):`
	
	`  return Response({'status': 200})`
	
	`@api_view(['GET'])`
		Aplikasi akan memuat halaman *dashboard*

No.	Skenario Pengujian	Hasil yang diharapkan

2.	`class TablesView(APIView):`
	`  #def get(self, request, *args, **kwargs):`
	`  #return render(request, "theme/tables.html", {})`
	
	`  def post(self, request, *args, **kwargs):`
	
	`  return Response({'status': 200})`
	`  def get(self, request, *args, **kwargs):`
	`  logs = LogTable.objects.all()  # Mengambil semua data log`
	`  return render(request, 'theme/tables.html', {'logs': logs})`
		Aplikasi akan memuat page Histori *Log*

3.	`with open(log_file_path, 'r') as file:`
	`  for line in file:`
	`  # Ekspresi reguler diperbarui untuk mengekstrak jenis serangan`
	`  match = re.search(r'\[([A-Za-z]{3} \w{3} \d{2} \d{2}:\d{2}:\d{2}\.\d{6} \d{4})\]`
	`  if match:`
	`  timestamp_str, client_ip, message = match.groups()`
	`  attack_type = None`
	
	`  # Identifikasi jenis serangan berdasarkan pesan`
	`  if "Remote Command Execution: Unix Command Injection" in message:`
	`  attack_type = "Unix Command Injection"`
	`  elif "XSS Attack Detected via libinjection" in message:`
	`  attack_type = "XSS Attack"`
	`  elif "SQL Injection Attack Detected via libinjection" in message:`
	`  attack_type = "SQL Injection"`
	
	`  # Jika jenis serangan dikenal, lakukan penyimpanan ke dalam database`
	`  if attack_type:`
	`  # Konversi format timestamp`
	`  log_timestamp = datetime.strptime(timestamp_str, '%a %b %d %H:%M:%S.%f %Y`
	`  mysql_timestamp = log_timestamp.strftime('%Y-%m-%d %H:%M:%S')`
	
	`  # Simpan semua informasi yang diinginkan ke database`
	`  insert_query = "INSERT INTO logs1 (timestamp, client_ip, attack_type, mes`
	`  values = (mysql_timestamp, client_ip, attack_type, message)`
	`  cursor.execute(insert_query, values)`
	
	`connection.commit()`
	`print("Parser Log Berhasil.")`
		Skrip ini dapat mengekstrak *error.log* dan memilah isi untuk disimpan ke dalam *database*

No.	Skenario Pengujian	Hasil yang diharapkan
4		Skrip ini dapat mengirim notifikasi ke telegram dengan membaca posisi *read position* agar tidak terjadinya redundan notifikasi

```python
def main():
    while True:
        last_read_position = get_last_read_position()
        last_notification_time = get_last_notification_time()

        with open(LOG_FILE, 'r') as file:
            file.seek(last_read_position)
            lines = file.readlines()
            update_last_read_position(file.tell())

        for line in lines:
            attack_detected = None
            if 'Remote Command Execution: Unix Command Injection' in line:
                attack_detected = 'Command Injection Attack'
            elif 'XSS Attack Detected via libinjection' in line:
                attack_detected = 'Cross-Site Scripting (XSS) Attack'
            elif 'SQL Injection Attack Detected via libinjection' in line:
                attack_detected = 'SQL Injection Attack'

            if attack_detected and (datetime.now() - last_notification_time).total_seconds() > NOTIFICATION_INTERVAL:
                send_telegram_message(f'{attack_detected} detected: ' + line)
                update_last_notification_time()
        time.sleep(10)  # Check the log every 10 seconds

if __name__ == '__main__':
    main()
```