BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian Harja, dkk [6] pada tahun 2019 yang berjudul "Implementasi Metode *Deep Packet Inspection* untuk Meningkatkan Keamanan Jaringan pada *Software Defined Networks*". Penelitian ini meggunakan modul DPI dengan *tools* ntopng yang dipasang pada *control plane* di jaringan SDN dengan tujuan agar seluruh paket yang melintas pada jaringan SDN tidak dianlisis secara menyeluruh oleh DPI dikarenakan sebelum DPI bekerja paket terlebih dahulu ditangani langsung oleh protokol *OpenFlow*. Algoritma yang digunakan pada pengimplementasian metode DPI adalah algoritma *Aho-Corasick* dengan menganalisis performa berdasarkan *pattern matching* dengan menggunakan *tree* yang terdiri dari *set of strings*. Hasil dari penelitian ini bahwa serangan DDoS dapat di blokir dengan menggunakan metode DPI yang memanfaatkan *tools* ntopng dan *floodlight* sebagai kontroler, dan *tools* ntopng dapat mengurangi pemrosesan data yang tidak perlu sehingga *bandwidth* dan *latency* tidak mengalami penurunan [6].

Pada tahun 2020 penelitian yang dilakukan oleh Rizal dan Sumaryana [7] yang berjudul "Peningkatan Keamanan Aplikasi *Web* Menggunakan *Web Application Firewall* (WAF) Pada Sistem Informasi Manajemen Kampus Terintegrasi". Penelitian ini membahas tentang cara meningkatkan sistem keamanan pada aplikasi berbasis *web* dengan menggunakan *modsecurity*. Serangan yang digunakan pada penelitian ini adalah *cross site scripting*. Pada tahap pegujian, peneliti menggunakan 2 skenario yaitu pada uji coba pertama dengan tidak mengaktifkan *modsecurity* terhadap serangan *cross site scripting*, pada ujicoba kedua dengan mengaktifkan *modsecurity* terhadap serangan *cross site scripting*. Pada hasil pengujian skenario pertama, *web server* dapat tereksploitasi. Pada skenario kedua menunjukan keberhasilan penggunaan *modsecurity* yaitu dengan setiap permintaan yang terdeksi sebagai ancaman dialihkan ke halaman kosong. Berdasarkan hasil pengujian penggunaan WAF *modsecurity* terbukti dapat meningkatan perlindungan terhadap *server* dengan melakukan penolakan dan pengalihan terhadap permintaan HTTP yang berbahaya [7].

Penelitian yang dilakukan oleh Mukhtar dan Azer [8] pada tahun 2020 yang berjudul " *Evaluating the Modsecurity Web Application Firewall Against SQL Injection Attack*" yang membahas tentang upaya untuk mendeteksi dan mencegah serangan SQL *Injection*. Penelitian ini bertujuan untuk menilai efesiensi dari WAF *modsecurity* dalam upaya pencegahan serangan SQL *injection*. Alur penelitian ini dimulai pada fase 1 yaitu melakukan pengujian DVWA menggunakan SQLMAP. Fase 2 melakukan pemasangan *modsecurity* pada *web server apache*. Pada fase 3, peneliti menambahkan beberapa opsi pada SQLMAP. Hasil pengujian fase 1 pada *website* terinjeksi 3 jenis serangan SQL. Pada fase 2 didapatkan hasil bahwa parameter serangan yang diuji tidak dapat mengesploitasi situs *web* DVWA yang berarti *modsecurity* telah memperbaiki kerentanan pada *web server*. Pada fase 3 didapatkan hasil bahwa penambahan parameter serangan tetap tidak dapat bekerja pada web DVWA yang telah dilindungi oleh *modsecurity*. Hasil dari penelitian menunjukan bahwa *modsecurity* efektif dalam menangani serangan SQL *injection* [8].

Pada tahun 2021 penelitian yang dilakukan oleh Ayunda, dkk [9] yang berjudul "*Implementation and Analysis modsecurity on Web-Based Application with* OWASP *Standards*" membahas mengenai kemampuan pengamanan jaringan mod *security* yang dikembangkan oleh OWASP pada aplikasi *web* terhadap serangan jaringan berjenis *brute force*. Penelitian ini dimulai dari mengindetifikasi masalah untuk dilanjutkan ke tahap pengintaian dengan melakukan pemindaian pada target menggunakan OWASP-ZAP dan ditemukan 31 kerentanan pada aplikasi *web* dengan rincian 8 risiko tinggi, 8 risiko medium, 15 risiko rendah. Pada tahap pengujian terhadap serangan *brute force* dengan mengaktifkan *modsecurity* dengan hasil serangan tersebut tidak berhasil memperoleh *password* dari daftar *password* acak yang berarti *modsecurity* berhasil memblokir serangan tersebut. Hasil dari penelitian menunjukan bahwa *rules* bawaan *modsecurity* yang dikembangkan oleh OWASP dapat menangani serangan *brute force* tetapi tidak dapat mengatasi serangan *session hijacking* [9].

Penelitian dari Pahlawan dan Ulum [5] pada tahun 2021 yang berjudul "Perbandingan Penerapan Metode Pengaman *ModSecurity* dan *Mod Evasive* pada *Web Server* terhadap serangan *Slow Headers*". Penelitian ini bertujuan untuk

mengetahui metode pengamanan terbaik terhadap serangan DOS *slow headers.* Pada penelitian ini menggunakan tiga sistem *web server*, setiap sistem dijalankan satu per satu dengan tujuan agar tidak terjadi pembebanan pada perangkat uji. Pada *web server* 1 dengan uji skenario melancarkan serangan DOS *slow header tools switchblade* dengan tidak menggunakan *modsecurity* dan *mod evasive*. Pada *web server* 2 dengan uji skenario melancarkan serangan DOS *slow headers* dengan menggunakan mod *security*. Pada *web server* 3 dengan uji skenario melancarkan serangan DOS *slow header* dengan menggunakan *mod* evasive. Berdasarkan hasil uji dari penelitian, *modsecurity* dinyatakan memiliki kinerja yang lebih unggul dibandingkan dengan *mod evasive*, *modsecurity* yang telah terpasang pada *server* 2 berhasil mempertahankan *web server* agar dapat tetap dikases dan *modsecurity* mengalami kenaikan penggunaan *memory* paling sedikit dibandingkan *server* 1 dan *server* 3. Sedangkan pada *server* 2 dengan menggunakan *mod evasive* tidak dapat melindungi *server* secara optimal dikarenakan *mod evasive* hanya dapat melakukan pembatasan terhadap permintaan paket HTTP utuh, sedangkan serangan *slow headers* bekerja dengan cara mengirimkan permintaan paket HTTP dengan tidak utuh [5].

Berikut pada Tabel 2.1 disajikan tabel informasi pembanding berdasarkan jurnal referensi yang bertujuan untuk mengetahui perbedaan penggunaan alat dan parameter yang digunakan oleh penelitian terdahulu dengan penelitian saat ini.

Tabel 2.1 Perbandingan Jurnal Kajian Pustaka

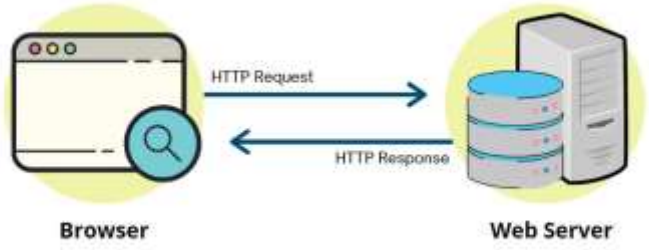
Penelitian Oleh	DDoS *Slow Headers*	Pemasangan *Firewall* Pada *Web Server*	*Modsecurity*	DPI	Implementasi	Simulasi	Parameter CPU *Usage*	Parameter *Delay*	Parameter *Request Time Out*	Parameter *Throughput*
Harja, dkk (2019) [6]				√		√				√
Rizal, Sumaryana (2020) [7]		√	√		√					
Mukhtar, Azer (2020) [8]		√	√		√					
Ayunda, dkk (2021) [9]		√	√			√				
Pahlawan, Ulum (2021) [5]	√	√	√			√				
Febriansyah (2022)	√	√	√	√	√		√	√	√	

2.2 TEORI DASAR PENDUKUNG

Pada sub bab teori dasar pendukung dipaparkan teori yang berkaitan dengan penelitian ini.

2.2.1 *Web Server*

Web server adalah sebuah perangkat lunak yang memberikan layanan berbasis data. *Web server* dapat melayanai permintaan berupa permintaan HTTP dan HTTPS yang berasal dari *client* pada sebuah *browser* untuk kemudian memproses permintaan tersebut dan memberikan hasil pemrosesan berupa halamaan *web* yang berbentuk dokumen HTML pada *browser* [10].



Gambar 2.1 Simulasi *Web Server* [11]

2.2.2 *Apache Web Server*

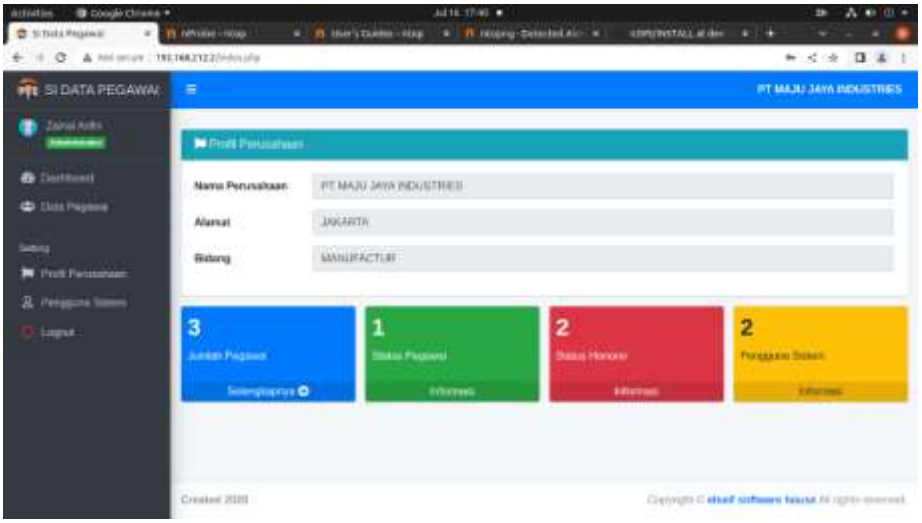
Salah satu *web server open source* adalah *apache web server* yang merupakan *unixbased web server*. *Apache* adalah *web server* dengan pengguna terbanyak di dunia dengan persentase sebesar 42% yang bersumber dari *domain website* yang terdapat pada internet. *Apache web server* memiliki program pendukung dengan jumlah banyak, hal tersebut dapat memberikan layanan yang cukup lengkap bagi para penggunanya [12].

Apache pertama kali dikembangkan berbasiskode pada NCSA HTTPD 1.3 yang diprogram ulang menjadi sebuah *web server*. *Apache* memiliki fitur yang sangat lengkap mulai dari performa yang tinggi, fungsionalitas, efisiensi, serta kecepatan [12]. *Apache* memiliki performansi lebih baik dari *web server open source* lainnya pada segi parameter *throughput* dan *response time* [13].

2.2.3 *Web* Aplikasi

Aplikasi berbasis *web* adalah sebuah program yang dapat dikases menggunakan *web browser*, aplikasi *web* dikembangkan dalam bahasa HTML, PHP, CSS, JS yang membutuhkan *web browser* untuk menampilkan konten dari aplikasi *web*. Pada umumnya aplikasi berbasis *web* memliki tingkat efisiensi yang tinggi dari penggunaan aplikasi diukur dari segi konumsi sumber daya dari sisi perangkat keras dan perangkat lunak [14].

Web aplikasi memungkinkan terjadi interaksi di dalam *web* applikasi tersebut seperti pengguna *web* aplikasi dapat mengisi dan mengirim sebuah formular, menggunakan fitur *chat*, melakukan transaksi pembayaran *online*. *Web* aplikasi berbeda dengan *website* pada umumnya, perbedaan tersebut terletak pada tugas utama yaitu *web* aplikasi dapat melakukan pengolahan data dan informasi secara otomatis, sedangkan *website* biasa hanya memberikan informasi kepada pembaca. Pada komposisi pengembangan terdapat perbedaan *web* aplikasi dengan *website* biasa yaitu *web* aplikasi memiliki kebutuhan dasar PHP (*Hypertext Preprocessor*) dan *python* sedangkan *website* tidak memerlukan PHP dan *python* [15].



Gambar 2.2 Contoh *Web* Aplikasi

2.2.4 MySQL

Pada perkembangannya, MYSQL disebut juga SQL yang merupakan singkatan dari *Structured Query Languange*. SQL merupakan bahasa terstruktur yang khusus digunakan untuk mengolah *database*. SQL pertama kali didefinisikan

oleh *American National Standards Institute* (ANSI) pada tahun 1986. MYSQL adalah sebuah sistem manajemen *database* yang bersifat *open source* [16].

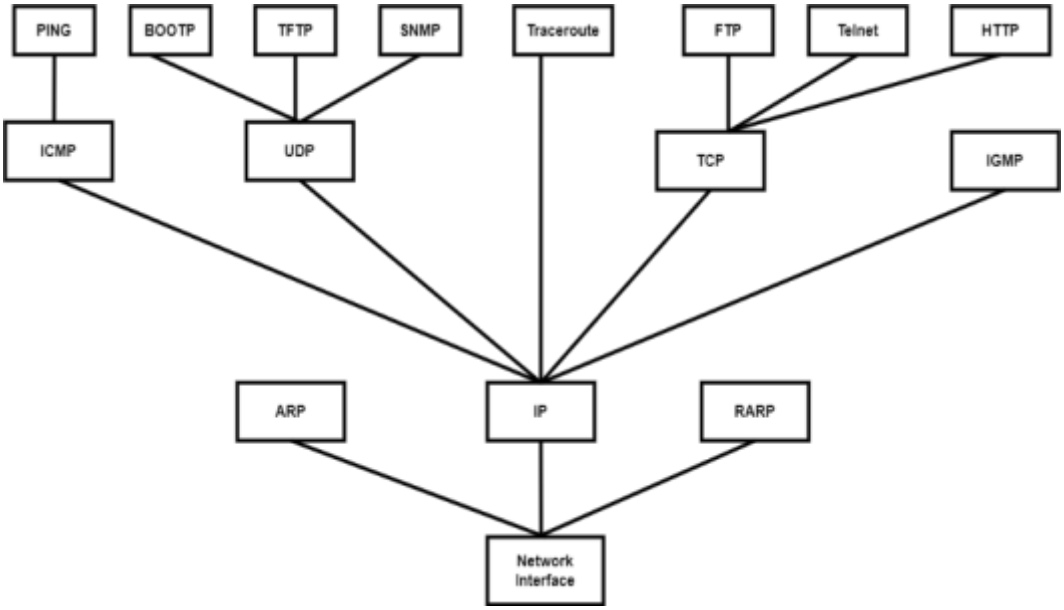
MYSQL merupakan sistem manajemen *database* yang bersifat *relational*. Artinya, data yang dikelola dalam *database* yang akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan jauh lebih cepat. MYSQL dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar [16].

SQL juga merupakan bahasa pemrograman yang dirancang khusus untuk mengirimankan suatu perintah *query* (pengaksesan data berdasarkan pengalamatan tertentu) terhadap sebuah *database*. Kebanyakan *software database* mengimplementasikan SQL secara sedikit berbeda, tapi seluruh database SQL mendukung subset standar yang ada. Jadi, SQL adalah permintaan yang melekat pada suatu database atau SMBD tertentu [16].

2.2.5 *Hypertext Transfer Protocol* (HTTP)

Hypertext Transfer Protocol (HTTP) adalah protokol pada lapisan aplikasi yang berfungsi untuk membantu proses pemindahan data antar komputer, protokol HTTP dapat mengirim data dalam berbagai media seperti dokumen, video, gambar, audio. Protokol HTTP berhubungan dengan *hypertext* yang memungkinkan protokol ini mengambil sumber daya pada sebuah tautan [17].

Pada awal perkembangannya, protokol HTTP berbasis *client server* sederhana yang diubagun atas dasar *Transnission Control Protocol* (TCP), dalam kerjanya protokol HTTP menggunakan *port* 80. Pada dasarnya protokol HTTP merupakan protokol yang beroperasi pada lapisan teratas TCP, dan protokol HTTP memiliki peran sebagai jalan permintaan dari client menuju ke server. Pada protokol HTTP tersedia beragam perintah dalam komunikasi antar *client* dengan *server*. Dalam komunikasi tersebut client melakukan permintaan akses dengan alamat IP atau mengakses *domain* (URL) untuk kemudian *web server* menerima dan mengelola permintaan tersebut [18].



Gambar 2.3 Hierarki Antar Muka Jaringan Komputer

2.2.6 *Port* 80

Port 80 merupakan jenis *port* dalam jaringan komputer yang awam digunakan berfungsi dalam menghubungkan peramban/*browser* ke halaman *website*, *port* 80 menunggu permintaan klien untuk kemudian permintaan tersebut dikirim menuju *web server* [19].

2.2.7 Keamanan Jaringan

Keamanan Jaringan adalah sebuah cara atau metode identifikasi yang digunakan dalam upaya pencegahan akses yang tidak sah dengan tujuan merusak jaringan, melakukan pencurian data, menyalahgunakan akses, memodifikasi data yang mampu merusak merusak jaringan [20]. Sistem keamanan jaringan melindungi aset *digital* termasuk lalu lintas jaringan. Keamanan jaringan dapat berbentuk perangkat lunak dan perangkat keras yang di desain untuk merespon berbagai serangan jaringan [21].

2.2.8 OWASP

OWASP merupakan organisasi internasional non *profit* asal Amerika Serikat yang secara resmi dibuka pada bulan Desember 2001. Pembentukan OWASP bertujuan agar membuka komunitas secara terbuka untuk mendedikasikan, mengembangkan aplikasi keamanan jaringan yang bersifat *open source* untuk diterapkan pada aplikasi *web* [22] .

OWASP memiliki standard untuk melakukan penilian dan pengujian keamanan pada sebuah *website* yaitu *information gathering, configuration management, secure transmission, authentication, session management, authorization, cryptography, data validation, denial of service, error handling* [23].

Gambar 2.4 Logo OWASP [24]

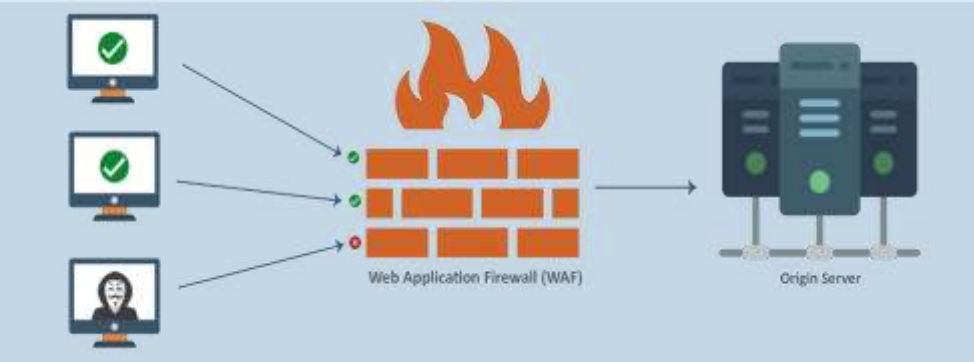
2.2.9 WAF *MODSECURITY*

Web Application Firewall (WAF) adalah sebuah modul *open source* yang berfungsi untuk menyaring, memantau, dan memblokir upaya peretasan dan ancaman yang dikririm dari *client* yang tidak bertanggung jawab pada sebuah *website.* Sebuah WAF terdiri dari 3 jenis yaitu *network based* WAF, *host based* WAF dan *cloud based* WAF. Cara kerja dari WAF hampir sama dengan cara kerja dari *firewall* tradisional biasa, *firewall* tradisonal bekerja berdasarkan pada suatu set aturan yang telah dikonfigurasi pada *setting firewall*, aturan tersebut dirancang unutk menyaring lalu lintas sebuah jaringan menggunakan protokol HTTP [25].

Sebagai alat pada sebuah jaringan, WAF memeriksa setiap data untuk menganalisis secara logika pada OSI *layer* 7 dan menyaring lalu lintas yang berbahaya. WAF mampu mendeteksi serangan seperti SQL *injection*, *cross site scripting* (XSS), DoS. WAF adalah salah satu sistem pertahanan pertama pada sebuah *website*. WAF sering digunakan pada lingkungan perusahaan yang menyediakan layanana interkoneksi seperti *e-commerce, online banking,* dan masih banyak yang lainnya [25].

Terdapat berbagai macam aplikasi WAF yang salah satunya adalah *modsecurity*. *Modsecurity* adalah aplikasi *firewall* yang bersifat *open source*. *Modsecurity* biasa dipasang pada apache *web server* dan merupakan aplikasi WAF yang paling sering digunakan sebagai *tools* keamanan jaringan terhadap serangan pada aplikasi *web*. *Modsecurity* mampu melakukan 80% pertahan pada aplikasi

web, *modsecurity* dapat digunakan sebagai *proxy* yang di *embed*. Pada *modsecurity* terdapat sebuah *rule* konfigurasi yang disebut dengan '*SecRules*' yang berfungsi untuk memantau lalu lintas HTTP secara *real time*, melakukan pencatatan, dan memfilter paket HTTP untuk dicocokan apakah paket HTTP tersebut sesuai dengan '*SecRules*' yang telah dikonfigurasi, apabila terdapat paket HTTP yang tidak sesuai maka *modsecurity* akan melakukan pemblokiran permintaan yang dianggap membahayakan [26].



Gambar 2.5 Mekanisme Kerja WAF [27]

WAF bekerja dengan cara menganilisis permintaan yang telah dikirimkan oleh *client* berupa HTTP (*Hypertext Transfer Protocol*) dan mencocokan permintaan tersebut dengan aturan untuk menentukan apakah permintaan tersebut berbahaya atau tidak, apabila permintaan tersebut berbahaya dan tidak sesuai dengan aturan WAF maka WAF dapat memblokir permintaan tersebut dan sebaliknya apabila permintaan tersebut aman dan sesuai dengan aturan WAF maka WAF akan mengizinkan paket tersebut untuk sampai ke *web server* tujuan. WAF menganalisis bagian utama pada HTTP yaitu pada permintaan *GET* dan *POST* [28].

Pada *web server* yang telah terinstall oleh *modsecurity*, setiap lalu lintas yang masuk melalui *web server* melewati lima langkah atau fase. Pada setiap fase, *modsecurity* akan melakukan beberapa pemanggilan set aturan yang berlaku untuk bekerja pada fase tersebut untuk melanjutkan ke fase berikutnya. Pada fase pertama yaitu permintaan *header* yang bertujuan untuk menilai permintaan sebelum melakukan pembacaan isi paket. Pada fase kedua yaitu permintaan *body*, *body* merupakan isi pada paket HTTP. Pada fase permintaan *body*, seluruh set aturan memiliki data permintaan yang telah tersedia. Pada fase ketiga yaitu *response header* yang bekerja setelah meminta *body* paket, pada fase ini set aturan perlu

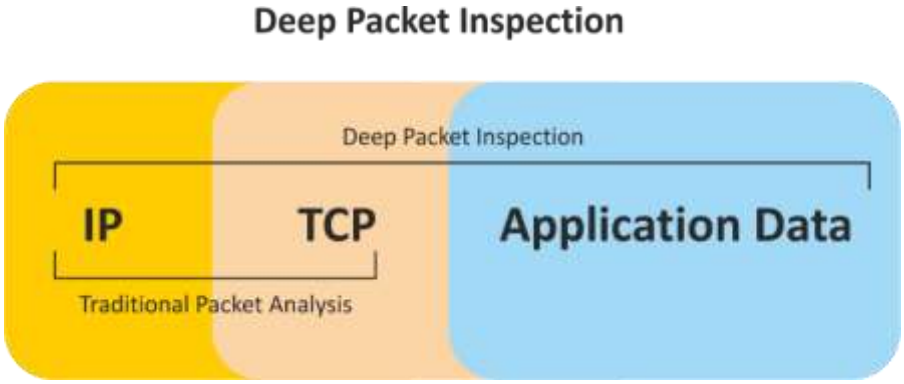
menentukan apakah akan menginspek *header* paket pengirim untuk mengecek kelengkapan header paket pengirim. Pada fase keempat yaitu *response body*, pada fase ini *body* paket pengirim dibaca secara keseluruhan, dengan semua itu *modsecurity* akan mencocokan set aturan dengan *body* paket tersebut untuk kemudian menentukan apakah paket yang telah dikirimkan oleh pengirim termasuk ke dalam paket yang membahayakan atau aman, apabila paket yang dikirimkan aman maka *modsecurity* akan mengizinkan paket tersebut melewati *web server*, dan apabila paket tersebut berbahaya maka *modsecurity* dapat memblokirnya berdasarkan set aturan yang berlaku. Pada fase 5 yaitu *logging* atau pencatatan, dalam fase ini *modsecurity* akan mencatatan segala aktivitas pada *web server* baik dalam upaya pengirim mengirimkan paket menuju *web server* atau pencatatan upaya penggagalan yang telah dilakukan oleh *modsecurity* [29].

2.2.10 *Deep Packet Inspection* (DPI)

Deep Packet Inspection (DPI) merupakan sebuah teknik *packet filtering* dalam lalu lintas jaringan yang akan melakukan *scaning* terhadap *header* dan muatan paket data dengan pola tertentu. DPI mengambil sebagian paket dari keseluruhan paket yang berada dalam lalu lintas dan memeriksa paket dalam *frame* TCP/UDP untuk menentukan tindakan berdasarkan informasi yang telah dikumpulkan. DPI dapat digunakan sebagai sitem pada keamanan jaringan, dengan adanya *network-based* DPI yang dapat mengidentifikasi serangan DoS [30].

DPI mengandalkan perbandingan dari dua bagian yaitu pada muatan paket dengan IP *header*. DPI membandingkan keduanya dengan aturan DPI untuk memutuskan apakah paket yang dikirimkan berbahaya atau aman, apabila paket tersebut berbahaya maka DPI dapat menghapus atau memblokir paket tersebut. DPI berkaitan dengan konten paket dibawah osi *layer* ke 4 yang mencakup *port*, sumber dan tujuan pengiriman, dan jenis protokol yang dikirimkan. DPI mengklasifikasikan jenis aplikasi berdasarkan nomor *port* ytang dilewati [31].

DPI bekerja dan diaplikasikan pada osi *layer* ke 7. DPI membuat penyaringan pada jaringan dengan memeriksa muatan paket baik menggunakan algoritma *Wu-Manber*, *Aho-corasick*, dan SBOM atau dengan algoritma pencocokan ekspresi *regular* yang digunakan pada NIDS dari snort, Bro, dan L7-*filter* [31].



Gambar 2.6 *Deep Packet Inspection* [32]

Terdapat dua cara untuk mengumpulkan packet dalam DPI yaitu *port mirroring* dan *optical splitter*. *Port mirroring* adalah sebuah mekanisme yang menginspeksi setiap lapisan pada OSI *layer* lalu mengirimkan tujuan informasi tersebut menuju pengaturan jaringan, sebuah cermin terbentuk pada masing masing lapisan lalu menyaring data tersebut pada lapisan aplikasi. Pada *optical splitter*, informasi paket dikumpulkan dan dikirimkan menuju pengaturan jaringan yang digunakan untuk penyaringan data [31].

Pada metode DPI menggunakan tools ntopng yang telah terpasang pada *web server*, ntopng akan menangkap paket dengan menggunakan PF_RING, paket yang telah tertangkap akan dilakukan pencatatan memanfaatkan n2disk dalam format PCAP. Pada ntopng akan melakukan pemeriksaan isi konten memanfaatkan nDPI, dengan nDPI ntopng dapat mengidentifikasi IP yang mencoba mengkases *web server*, membedah isi dari paket pengirim. Mengetahui port dan metode pengiriman yang digunakan oleh pengirim. Ntopng menggunakan teknologi HTML5/AJAX untuk melakukan *logging*/pencatatan untuk menghasilkan statistik lalu lintas jaringan [33].

2.2.11 *Denial of Service* (DoS)

DoS adalah salah satu jenis serangan *cyber* yang menyerang *website*, jaringan, perangkat jaringan. Metode serangan DoS dengan membanjiri permintaan palsu, *fake traffic*, ke *server* dengan jumlah sangat banyak sehingga *server* tidak dapat melayani permintaan dari pengguna lain yang berujung pada tidak beroperasinya *server*. Serangan DoS menjadi serangan yang sangat berbahaya ketika serangan ini terdistribusi dalam jumlah penyerang menggunakan lebih dari

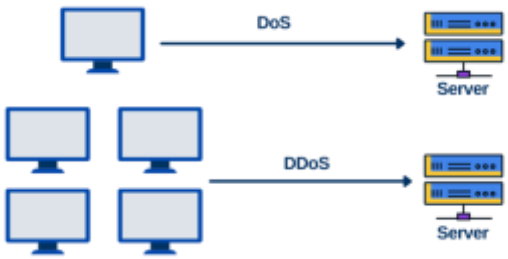
satu komputer, serangan ini disebut dengan serangan DDoS (*Distributed Denial of Service*) [34].

DoS memiliki beragam macam serangan, salah satunya adalah serangan *slow* HTTP DoS yaitu serangan DoS *layer* 7 dengan cara kerja mengirim permintaan HTTP yang tidak lengkap, serangan ini tergolong kedalam kelas baru dalam serangan DoS, serangan ini sulit untuk dilacak dikarenakan :

1. Jenis serangan DoS ini tidak memakan banyak *bandwidth*.
2. Serangan ini bertujuan untuk menciptakan kemacetan sehingga terjadi penurunan sumber daya.
3. Serangan ini terdistribusi dengan menunggangi paket http untuk menyembunyikan asal serangan ini.

Serangan *slow* HTTP adalah serangan yang berbahaya, salah satunya adalah *slow headers*. Serangan *slow headers* menggunakan *tools slowrois* dalam melancarkan serangannya, serangan ini mengirimkan *header* HTTP dengan paket yang tidak lengkap, hanya menambahkan tetapi tidak menyelesaikan isi paket HTTP dengan tujuan memaksa *web server* untuk tetap menjaga koneksi tetap terbuka sehingga tidak terputus, serangan *slowrlois* akan mengeruk seluruh sumber daya dari *web server* target, dengan ini klien lain tidak dapat terpenuhi permintaan dan akses *web server* [2].

Pada skala yang lebih besar serangan DoS dapat menjadi sangat membahayakan Ketika menjadi serangan DDoS (*Distributed Denial of Service*). Serangan DDoS adalah DoS yang dilakukan lebih dari satu penyerang. Serangan DDoS juga dapat dilakukan oleh sekelompok pengguna aktif dengan skala yang besar dengan menggunakan *tools* sederhana. Kelompok tersebut menggunakan *tools* serangan DoS bersifat *open source* yang biasa disebut LOIC (*Low Orbit Ion Canon*) untuk menyerang target [35].



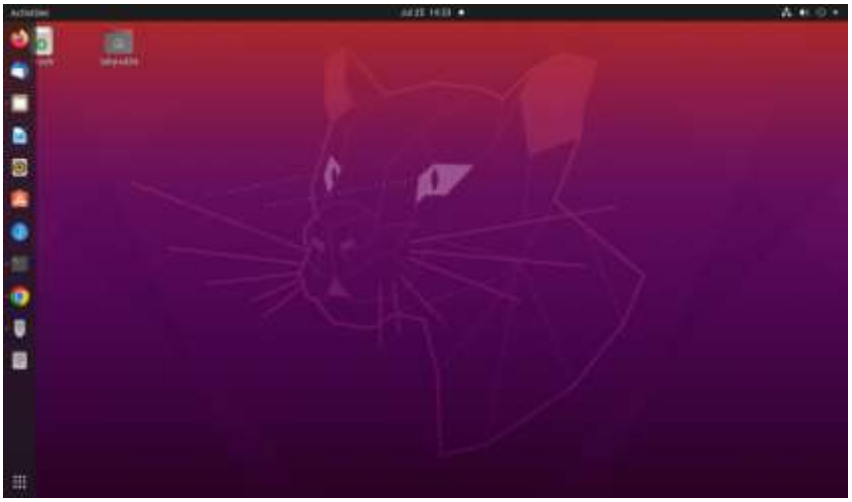
Gambar 2.7 Perbedaan Serangan DoS dan DDoS [36]

2.2.12 *Ubuntu*

Ubuntu merupakan sistem operasi yang dikembangkan dari program inti *Linux*. Sistem operasi *ubuntu* pertama kali dikenalkan ke publik pada tahun 2004 oleh perusahaan canonical, canocinal adalah perusahaan yang walanya menjadi *developer* dari sistem operasi *debian* [37].

Sistem operasi *ubuntu* bersifat *open source* yang sesuai dengan cita cita dari penemunya yaitu Mark richard shuttleworth yang menyatakan sebuah perangkat lunak harus tersedia tanpa biaya, perangkat lunak harus dapat digunakan oleh pengguna dalam bahasa lokal mereka sendiri dan untuk mereka yang memiliki kekurangan, dan pengguna harus memiliki kebebasan untuk menyesuaikan dan mengubah perangkat lunak mereka menurut dengan apa yang mereka inginkan. Sistem operasi *ubuntu* telah menyediakan banyak paket ytang telah diuji dan dipilih secara teliti dari distro *debian* [38].

Ubuntu merupakan salah satu sistem operasi yang rutin merilis *update* yang stabil dengan jadwal pemnbaruan dilakukan setiap enam bulan sekali untuk versi reguler dan dua tahun sekali untuk versi LTS (*Long Term Service*). Sistem operasi ubuntu banyak digunakan dalam berbagai lintas perangkat seperti pada komputer hingga penggunaan pada *server* [38].



Gambar 2.8 Tampilan Awal *Ubuntu*

2.2.13 *Kali Linux*

Kali Linux adalah sistem operasi bebrbasis *debian linux*, *kali linux* merupakan pembangunan kembali *backtrack* linux yang disempurnakan. *Kali linux* menjadi salah satu distribusi *Linux* tingkat lanjut untuk *Penetration Testing* dan

audit keamanan. *Kali linux* beroperasi dengan sistem operasi berbasis *open source*, dengan begitu *kali linux* dapat dioperasikan secara global serta dapat melakukan modifikasi tanpa membutuhkan lisensi [39].

Gambar 2.9 Tampilan Awal *Kali Linux*

2.2.14 *Slowhttptest*

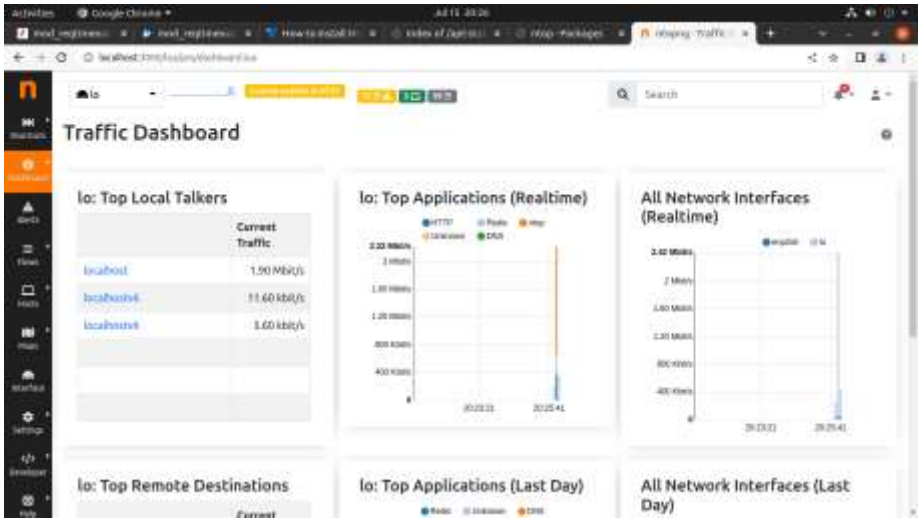
Slowhttptest merupakan *tools* serangan jaringan yang digunakan dalam simulasi serangan *application layer Denial of Service* berjenis *slow headers* dari *prologging* HTTP *connections.* Seorang pengguna dapat mensimulasikan uji kerentanan *web server* dari serangan DoS atau DDoS *slow headers* atau dapat mengecek kekuatan situs saat menangani *concurrent connections* dalam waktu yang bersamaan [40].

Gambar 2.10 Serangan DoS dengan *Tools Slowhttptest*

2.2.15 Ntopng

Ntop *next generation* (Ntopng) adalah salah satu *tools* yang digunakan untuk memonitoring lalu lintas jaringan hingga ke *layer* 7, *tools* ini bersifat *open source*. *Tools* ini dapat menampilkan informasi mengenai lalu lintas jaringan dan daftar *host* yang terhubung dengan jaringan internal secara detail. Ntopng juga merupakan *tool* keamanan jaringan yang diinstalilasi pada metode *Deep Packet Inspection* (DPI) dengan kolaborasi ntopng dengan DPI , pemantauan lalu lintas dapat dilakukan secara *remote* ke *server gateway* [41].

Dalam menjalankan fungsinya sebagai pemantau lalu lintas dan keamanan jaringan, ntopng bekerja secara bersamaan dengan *tools* nDPI yang telah menjadi satu paket instalasi dari ntopng. Ntopng menggunakan teknologi nDPI untuk melakukan pendeteksian secara otomatis. *Tools* nDPI menyediakan protocol pemantauan seperti HTTP, syslog, ICMP, DNS [42].



Gambar 2.11 Tampilan Awal *Tools* Ntopng

2.2.16 CPU *Usage*

CPU (*Central Processing Unit*) termasuk ke dalam jenis perangkat keras komputer yang bertugas menerima, mengeksekusi dan menyelesaikan perintah yang dikirimkan dari perangkat lunak. CPU menjadi pusat pengolahan data dalam sistem komputer atau sering dikenal dengan sebutan prosesor [43].

CPU *usage* atau dalam Bahasa Indonesia disebut dengan penggunaan CPU merupakan istilah yang digunakan dalam menggambarkan seberapa besar prosesor bekerja. Pada umumnya penggunaan CPU direpresentasikan dalam satuan persen,

besaran penggunaan CPU bergantung pada sebearap banyak tugas yang ditangani oleh CPU dalam operasi komputer [44].

2.2.17 *Response Time*

Response Time adalah lamanya waktu respon yang dibutuhkan pada satu *node* system dalam menanggapi permintaan *client*. Waktu respon dimulai pada saat *client* mengirim permintaan hingga permintaan tersebut telah selesai [45].

2.2.18 *Delay*

Delay merupakan lamanya waktu yang dibutuhkan data dalam transmisi dari pengirim ke penerima. Beberapa faktor dapat mempengaruhi nilai *delay* seperti jarak, media yang digunakan, *congestion* atau lamanya waktu proses [46].

Tabel 2.1 Standar *Delay* Berdasarkan TIPHON [46].

Kategori *Latency*	*Delay*
Sangat Bagus	< 150 ms
Bagus	150 s/d 300 ms
Medium	300 s/d 450 ms
Buruk	> 450 ms