Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB II

Tinjauan Pustaka

2.1. *Denial-Of-Service* (DOS)

Denial-of-Service merupakan sebuah metode penyerangan dalam dunia maya untuk mengacaukan atau bahkan menghentikan sistem layanan sebuah *server* dalam jaringan. Terdapat berbagai macam teknik untuk melakukan *Denial-of-Service* diantaranya merupakan *Distributed Denial-of-Service (DDOS)* yang mengandalkan pengkoordinasian *bot* atau *host* yang telah terinfeksi untuk mengacaukan layanan *server*.

DDOS dapat diklasifikasikan menjadi dua buah kelas yaitu *Bandwidth Depletion Attacks* dan *Resource Depletion Attacks* [9], dan mempunyai teknik penyerangan yang berbeda berdasarkan kelasnya seperti:

Bandwidth Depletion Attacks :

1. *Flood Attacks* : Penyerangan yang memanfaatkan host yang terinfeksi untuk mengirimkan lalu lintas dengan jumlah yang besar untuk memenuhi band*width* sebuah host/server.
2. *Amplification Attacks* : Penyerangan yang bekerja dengan cara mengirimkan sebuah paket yang mempunyai alamat tujuan host/server korban pada broadcast IP sebuah subnet, dengan begitu seluruh host pada subnet akan mengirimkan reply pada host/server korban dan memenuhi band*width*nya.

Resource Depletion Attacks :

1. *Protocol Exploit Attacks* : Penyerangan yang memanfaatkan kelemahan sebuah protocol. Contohnya seperti memanfaatkan *TCP SYN* pada Three-way handshake dengan tidak membalas

Implementasi Count-Min..., Nelson Wijaya, FTI UMN, 2018

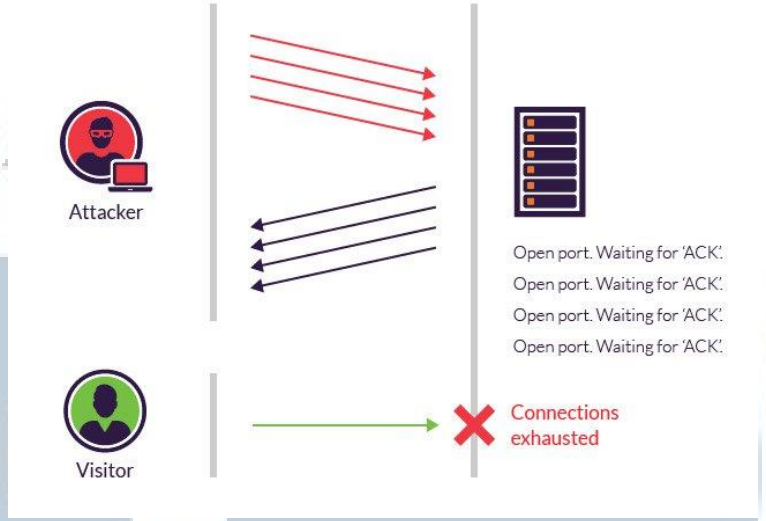
ACK Flag pada server setelah server mengirim *SYN+ACK Flag*, proses ini dilakukan secara berulang sehingga server akan terus menunggu respon dari penyerang dan tidak dapat menerima *client* baru karena mempunyai antrian koneksi dengan penyerang.

2. *Malformed Packet Attacks* : Penyerangan yang memanfaatkan IP paket's header. Contohnya seperti mengirimkan IP paket dengan mengaktifkan *quality-of-service* pada bagian *optional flag* di IP *Header*, dengan begitu server akan membutuhkan lebih banyak waktu untuk memproses koneksi dengan penyerang sehingga dapat memperlambat layanan untuk *host* yang lain.

2.2. *SYN Flood*

SYN Flood merupakan suatu teknik *Denial of Service* bertipe *Flood Attack* dan *Protocol Exploit Attack* yang telah menjadi salah satu teknik penyerangan yang populer [10].

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

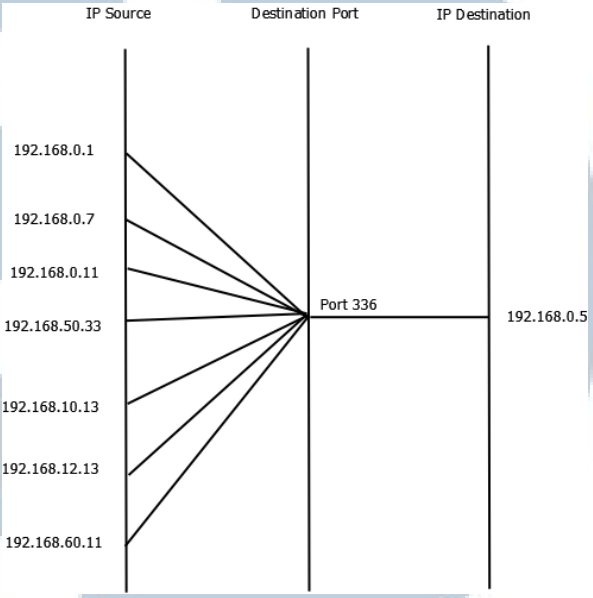


Gambar 2.1 Visualisasi SYN Flood Attack [23]

Penyerangan dilakukan dengan memanfaatkan kelemahan pada protocol Three-Way handshake seperti yang digambarkan pada Gambar 2.1. Pada prosedur Three-Way handshake yang normal, *client* mengirim paket *TCP SYN* kepada server, lalu server akan membuat sebuah koneksi pada *client* dengan membalasnya menggunakan sebuah paket *TCP SYN-ACK Flag* untuk menginformasikan pada *client* bahwa server dapat menerima koneksi *client*, dengan begitu *client* akan membalasnya dengan paket *TCP ACK* [11].

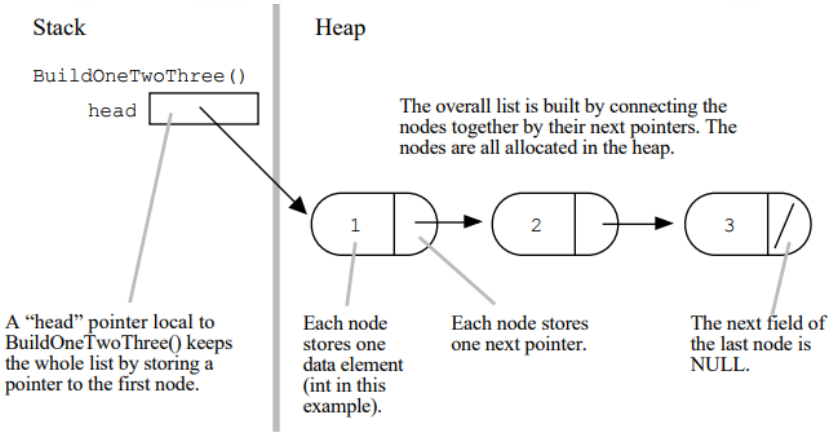
Namun terdapat kelemahan pada protokol disaat *client* tidak membalas paket *TCP ACK* pada server, maka server akan mempertahankan koneksi dengan *client* hingga waktu yang ditentukan untuk menerima paket *TCP ACK* sebelum memutuskan koneksi dengan *client*. Mengetahui kelemahan protokol tersebut, terciptalah sebuah teknik yang bertujuan untuk membuat server mempertahankan banyak koneksi pada *client* yang tidak menggunakan layanan server dengan tidak membalas paket *TCP SYN-ACK* dari server.

SYN Flood secara umumnya menyerang menggunakan berbagai Alamat IP Sumber kepada satu port pada satu Alamat IP Tujuan, sehingga dapat digambarkan seperti pada Gambar 2.2.



Gambar 2.2 Struktur Serangan SYN Flood

2.3. *Linked-List*



Gambar 2.3 Visualisasi Struktur Data Linked-List [11]

Gambar 2.3 merupakan visualisasi bagaimana struktur data *Linked-List* bekerja. *Linked-List* merupakan sebuah struktur data linear yang terdiri

dari sebuah *node* yang memiliki alamat ke *node* yang lainnya sehingga membentuk sebuah sekuensi. Jika dibandingkan dengan array yang juga merupakan struktur data linear, *Linked-List* mempunyai keunggulan dimana ukurannya dapat bertambah secara dinamik dibandingkan dengan array yang mempunyai ukuran yang statik [12].

2.3.1. Pemeriksaan Elemen Baru

Pemeriksaan elemen baru pada *Linked-List* melibatkan proses pemeriksaan setiap node pada *Linked-List*. Waktu yang dibutuhkan untuk memeriksa elemen baru bergantungan dengan seberapa panjang *Linked-List*, maka dapat dikatakan bahwa waktu yang dibutuhkan untuk memeriksa elemen baru adalah $O(n)$ dimana n merupakan banyaknya node pada *Linked-List*.

2.4. *Count-Min Sketch*

Count-Min Sketch merupakan sebuah struktur data probabilistik yang dipublikasikan oleh Graham Cormode dan S. Muthukrishnan dalam karya ilmiah yang berjudul *"An Improved Data Stream Summary: The Count-Min Sketch and its Applications"* [6]. *Count-Min Sketch* mempunyai struktur data yang mirip dengan struktur data *Bloom Filter* [7] namun *Count-Min Sketch* lebih digunakan untuk menjadi solusi *Heavy Hitters Problem*.

Pada dasarnya *Count-Min Sketch* digunakan sebagai sebuah *counter* dengan susunan dua dimensi $[d, w]$ yang disebut *hashtable*, dimana w (*width*) merupakan lebar susunan *counter* dan d (*depth*) merupakan kedalaman tabel. Pada setiap d ditentukan sebuah fungsi *hash* yang menghasilkan *hash* yang berbeda dengan fungsi *hash* lainnya dengan *input* yang sama [13],

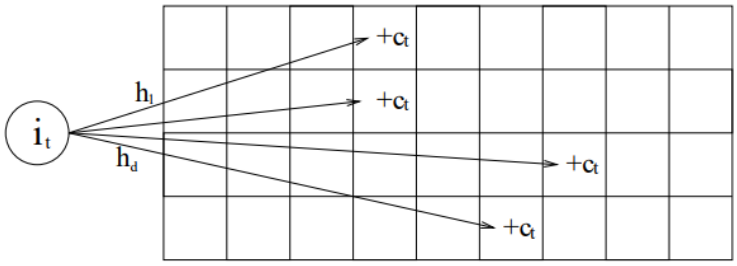
$$h_1 \dots h_d : \{1 \dots n\} \rightarrow \{1 \dots w\} \qquad (1)$$

besar w dan d yang efektif dapat bergantung pada parameter (ε, δ) dimana ε merupakan probabilitas terjadinya error yang diterima dengan δ yaitu konstan stokastik atau seberapa acak data yang ingin dihitung. Dengan parameter tersebut dapat digunakan pada persamaan $w = \left\lceil \frac{e}{\varepsilon} \right\rceil$ dan $d = \left\lceil ln \frac{1}{\delta} \right\rceil$.

2.4.1. Prosedur Update

Setiap aksi update dapat direpresentasikan sebagai (i_t, c_t) dimana i_t adalah datanya dan c_t adalah sebuah konstanta yang dapat atur sebagaimana keinginan alur kerja *counter*. Sehingga proses update dapat direpresentasikan sebagai Persamaan (2).

$$a_{i_t}(t) = a_{i_t}(t-1) + c_t \qquad (2)$$



Gambar 2.4 Visualisasi Prosedur Update [13]

Berdasarkan dari diagram Gambar 2.4 dapat dilihat bahwa untuk setiap proses *update* dapat dirumuskan menjadi Persamaan (3),

$$\left[CM\left[j, h_j(i_t)\right] + c_t\right] \rightarrow CM[j, h_j(i_t)] \qquad (3)$$

Dimana $h_j(i_t)$ merupakan sebuah fungsi *hash* yang dilakukan pada data yang didapatkan, dan kemudian hasil *hash* tersebut menjadi penentu kolom cell [13].

2.4.2. Point Query

Prosedur point *query* adalah prosedur yang digunakan untuk mendapatkan hasil *count* dari sebuah *key*word i, yang kemudian akan di*hash* dan untuk setiap fungsi *hash* akan diambil *count*nya dan dimasukan pada sebuah set, hasil *count* akan didapatkan dengan mencari nilai terkecil pada set tersebut. Sehingga prosedur *point query* dapat direpresentasikan secara matematis menjadi Persamaan (4).

$$min\{CM[1, h_1(i)], CM[2, h_2(i)], \dots, CM[j, h_j(i)]\} \qquad (4)$$

2.4.3. Periksa Elemen Baru

Prosedur pemeriksaan ketersediaan elemen baru dapat dilakukan dengan melakukan proses *hash* pada sebuah *key*word i yang ingin diketahui ketersediaan *counter*nya dalam *hashtable*, hasil *hash* yang digenerasikan dapat menjadi sebuah indeks pada *hashtable* untuk diambil hasil *count*nya dan jika nilai *min*imum dari *count* tersebut merupakan 0 maka *key*word i tidak tersedia dalam kumpulan *key*word A yang telah diupdate pada *hashtable*.

Asumsikan bahwa,

$$A = \{K_1, K_2, \dots, K_n\}$$

Sehingga,

$$(min\{CM[1, h_1(i)], CM[2, h_2(i)], \dots, CM[j, h_j(i)]\} = 0) \rightarrow (i \notin \mathrm{A})$$

Maka dapat disimpulkan bahwa waktu yang dibutuhkan untuk menentukan bahwa elemen yang di*input* merupakan elemen baru atau tidak adalah $O(d)$ dimana d adalah kedalaman *hashtable*.

2.5. *Universal Hash*

Universal Hash merupakan sebuah konsep dimana sebuah kumpulan *hash* function yang mempunyai kemungkinan terjadinya *Hash Collision* yang kecil. *Universal Hash* harus memenuhi sebuah persamaan matematis, dimana pada sebuah kumpulan *key* U ,fungsi *hash* $x \neq y \in U$ dan sebuah kumpulan *hash* yakni B dan mempunyai *hash* $s, t \in B$. Maka dapat disimpulkan probabilitas bahwa sebuah *hash* function h berdalam dalam kumpulan *hash* function H adalah $P_{h \in H}[h(x) = s, h(y) = t] = \frac{1}{R^2}$ [8] .

$$h(x) = \sum_{i=0}^{k-1} \left((a_i x^i) mod\ p \right) mod\ m \qquad (5)$$

Persamaan (5) merupakan persamaan fungsi *Universal Hash.* Dimana $x < p$ dan p merupakan sebuah bilangan prima Mersenne agar komputasi tidak memakan waktu yang besar, dan a_i merupakan sebuah angka yang dipilih secara acak dimana $0 < a_i < p$. Dengan begitu probabilitas terjadinya *hash* collision adalah $P_{h \in H}[h(x_i) = v_i] = \frac{1}{m^k}$,dimana $v_i \in [m], i \in (0,1, \dots, k-1)$ [8].

2.6. Murmur*Hash*

Murmur*Hash* merupakan sebuah fungsi *hashing non-cryptographic* yang diciptakan oleh Austin Appleby pada tahun 2008, dan sekarang source code *hash* function tersebut bisa didapatkan pada GitHub dengan nama SM*Hash*er [14]. Berikut merupakan pseudocode algoritma Murmur*hash* :

```
Murmurhash_32bit(key,length,seed)
        // Semua INTEGER yang digunakan pada versi ini   // bertipe 32 bit
unsigned integer
        c1 = 0xcc9e2d51
        c2 = 0x1b873593
        r1 = 15
        r2 = 13
        m = 5
        n = 0xe6546b64
        hash = seed
        foreach fourBytes of key
                k = fourBytes
                k = k * c1
                k = k ROL r1 times
                //ROL merupakan sebuah operasi yang
                //menggeserkan bit ke kiri, namun bit pada
                //posisi paling kiri dipindahkan ke bit paling
                //kanan
                // contoh : ROL 111010 2
                // Hasil : 101011
```

Pesudocode dilanjutkan dibawah….

```
            k = k * c2
            hash = hash XOR k
            hash = hash ROL r2 times
            hash = hash * m + n
    if there is still any remaining bytes in key
            remainingbytes = remainingbytes * c1
            remainingbytes = remainingbytes ROL r1 times
            remainingbytes = remainingbytes * c2
            hash = hash XOR remainingbytes
    hash = hash XOR length
    hash = hash XOR (hash shift right 16 times)
    hash = hash * 0x85ebca6b
    hash = hash XOR (hash shift right 13 times)
    hash = hash * 0xc2b2ae35
    hash = hash XOR (hash shift right 16 times)
```

Berikut merupakan tes banding performa *hash*ing dari berbagai algoritma *hash*ing menggunakan Hsieh's Test :

Tabel 2.1 Tes banding performa dengan berbagai algoritma hash [15]

CRC32	4.7810s
oneAtATime*Hash*	3.5470s
alphaNum*Hash*	2.2500s
FNV*Hash*	2.1870s
Jenkins Lookup3	1.2650s
SuperFast*Hash*	1.2970s
Murmur*Hash*V.2	0.9060s

UNIVERSITAS MULTIMEDIA NUSANTARA

2.7. Naïve Bayes Classifier

2.7.1. Teori Dasar

Naïve bayes classifier merupakan sebuah teknik *machine learning* yang digunakan untuk mengkelompokkan data baru pada sekumpulan data yang telah dipelajari. *Naïve Bayes Classifier* merupakan salah satu pengaplikasian Teori Bayes, yang merupakan sebuah teori probabilistik yang ditemukan oleh Thomas Bayes untuk mencari kemungkinan terjadinya sebuah kejadian berdasarkan dari probabilitas sebuah elemen yang berkaitan dengan kejadian.

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \quad (6)$$

Persamaan (6) merupakan persamaan Bayes. Dimana $P(A|B)$ merupakan probabilitas sebuah kejadian A saat terjadinya kejadian B.

Naïve bayes mengasumsikan bahwa setiap fitur mempunyai probabilitas yang independen atau tidak terpengaruh dengan probabilitas kejadian yang lain, maka persamaan Teori Bayes dapat dimodifikasi menjadi Persamaan (7).

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)} \quad (7)$$

Dimana X merupakan vektor variable-variable yang dapat diukur dari kerjadian yang terkait yang disebut *fitur* dan dapat direpresentasikan sebagai $X = (x_1, \dots, x_i)$. c merupakan sebuah kelas yang bagian dari kelas yang ingin diklasifikasikan. Dengan begitu persamaan dapat diubah menjadi Persamaan (8).

$$P(c|x_1, \dots, x_n) = \frac{P(x_1,\dots,x_n|c)P(c)}{P(x_1,\dots,x_n)} \quad (8)$$

Menggunakan asumsi bahwa setiap fitur adalah independen maka relasi dapat disingkat menjadi :

$$P(x_1, \dots, x_n | c) = P(x_i | c) \quad (9)$$

$$P(c | x_1, \dots, x_n) = \frac{P(c) \prod_{i=1}^{n} P(x_i | c)}{P(x_1, \dots, x_n)} \quad (10)$$

Dikarenakan $P(x_1, \dots, x_n)$ pada Persamaan (10) merupakan *input* yang konstan, maka dapat disimpulkan bahwa $P(c | x_1, \dots, x_n)$ mempunyai ratio yang sama dengan $P(c) \prod_{i=1}^{n} P(x_i | c)$ sehingga dapat disimpulkan menjadi Persamaan (11).

$$P(c | x_1, \dots, x_n) \propto P(c) \prod_{i=1}^{n} P(x_i | c) \quad (11)$$

Naïve bayes merupakan sebuah classifier bertipe *Supervised Machine Learning*, sehingga memerlukan prosedur pelatihan menggunakan data yang sudah terklasifikasikan agar sistem dapat mengklasfikasikan sebuah data secara independen.

2.7.2. Fase Training

Fase training dapat dilakukan dengan mencari *Mean* dan *Variance* dari nilai setiap fitur dan disimpan untuk digunakan pada proses prediksi.

$$\mu = \frac{\sum_{n}^{i=0} x_i}{n} \quad (12)$$

$$\sigma^2 = \frac{\sum_{n}^{i=0} (x_i - \mu)^2}{n} \quad (13)$$

Persamaan (12) dan Persamaan (13) merupakan persamaan *Mean(Simbol* μ*)* dan *Variance(Simbol* σ^2*).* Dimana x merupakan kumpulan data yang ingin dihitung *Mean* dan *Variancenya*, dan n merupakan jumlah kumpulan data.

Fase Prediksi

2.7.3. Dikarenakan data mengalir secara kontinu, maka diasumsikan bahwa setiap fitur kelas terdistribusi berdasarkan dari distribusi normal [16].

$$p(x\,|\sigma^2, \mu\,) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (14)$$

Persamaan (14) menunjukkan persamaan fungsi distribusi normal. Dimana nilai *Mean(μ)* dan *Variance(σ^2)* didapatkan dari fase training pada sebelumnya, dengan begitu dapat dihitung probabilitas dari nilai fitur data baru pada fungsi distribusi. Setelah dikumpulkan probabilitas fitur tersebut, dapat dibandingkan probabilitas bahwa nilai fitur dari data baru merupakan bagian dari sebuah kelas.

Lalu probabilitas setiap fitur dikumpulkan dan dihitung kemungkinannya bahwa data yang baru merupakan sebuah bagian dari kelas *c* sesuai dengan persamaan *Bayes*. Probabilitas yang telah dihitung dapat dibandingkan dengan probabilitas kelas lain yang dihitung menggunakan proses yang sama, jika probabilitas suatu kelas lebih besar dari kelas yang lainnya. Sehingga secara matematis dapat ditentukan kelas dari sebuah data *X* berdasarkan dari Persamaan (15).

$$X \in c_1$$

$$if:$$

$$P(c_1|x_1, \dots, x_n) > P(c_2|x_1, \dots, x_n) \qquad (15)$$