

Packet Marking Algorithm for IP Traceback

Abstrak

Tulisan ini menjelaskan sebuah teknik untuk menelusuri serangan *flooding* paket yang bersifat *anonymous* di Internet, dari korban serangan sampai ke sumber serangannya. Hal yang menjadi motivasi adalah semakin meningkatnya frekuensi dan kecanggihan serangan *denial-of-service* (DoS-attack), dan sulitnya cara penelusuran paket-paket dengan alamat sumber yang tidak benar (di-*spoof*). Pada tulisan ini, dijelaskan sebuah mekanisme *traceback*, berdasarkan penandaan paket (*packet marking*) pada *network*. Teknik ini memungkinkan “korban” mengidentifikasi jalur (*path traffic*) serangan yang terbentang di *network*, tanpa memerlukan dukungan interaktif dari *Internet Service Provider* (ISP). Lebih jauh lagi, teknik *traceback* ini dapat dilakukan secara “*post mortem*”—setelah suatu serangan terjadi. Disajikan pula implementasi dari teknik ini, yang (kebanyakannya) *backward compatible*, dan dapat diimplementasikan secara efisien dengan menggunakan teknologi yang konvensional.

Daftar Isi

1	Pendahuluan	2
2	Beberapa teknik	3
	A. Ingress Filtering	3
	B. Link Testing	4
	1) Input debugging	4
	2) Controlled flooding	5
	C. Logging	6
	D. ICMP traceback	6
3	Gambaran dasar	6
	A. Beberapa definisi	7
	B. Beberapa asumsi dasar	8
4	Beberapa algoritma dasar	9
	A. Node append	9
	B. Node sampling	10
	C. Edge sampling	11
5	Pengkodean	13
	A. Compressed Edge Fragment Sampling	13
	B. IP Header Encoding	17
6	Keterbatasan	19
	A. Backward compability	19
	B. Serangan tersebar	20
	C. Validasi jalur	20
	D. Teknik penentuan asal serangan	21
7	Simpulan	21
	Pustaka	22

1 Pendahuluan

Serangan *denial-of-service* (DoS-attack) menghabiskan sumberdaya dari suatu *remote host* atau suatu jaringan komputer, sehingga mengurangi kualitas pelayanan, bahkan sampai menolak melayani *user* yang legal. Serangan semacam ini termasuk masalah keamanan yang paling sukar ditangani, sebab serangan semacam ini sangat mudah dilakukan, sukar ditanggulangi dan untuk ditelusuri. DoS-attack yang dilakukan di Internet telah bertambah dalam frekuensi, keganasan, dan kecanggihannya. Antara tahun 1989 sampai 1995, jumlah serangan semacam ini yang dilaporkan ke badan Computer Emergency Response Team (CERT) bertambah 50% per tahun. Pada laporan survey CSI/FBI tahun 1999, 32% responden mendeteksi adanya serangan ini ke situs mereka. Bahkan lebih mengkhawatirkan lagi, laporan terakhir mengindikasikan bahwa para penyerang telah membangun *tools* yang dapat mengkoordinasikan serangan tersebar (*distributed attacks*) dari banyak situs yang berbeda.

Sayangnya, mekanisme untuk menangani DoS-attack tidak berkembang pada waktu yang bersamaan. Kebanyakan usaha pada wilayah ini terfokus pada mentoleransi serangan dengan cara mengurangi akibat yang ditimbulkan serangan tersebut pada ‘korban’. Teknik ini memang dapat—untuk sementara—memberikan penanganan darurat yang cukup efektif, akan tetapi tidak akan dapat menghilangkan masalahnya sama sekali, apalagi menakutkan bagi penyerang. Saran lain, dan fokus dari tulisan ini, adalah dengan menelusuri serangan sampai ke asalnya—yang idealnya dapat menghentikan serangan pada sumbernya.

Solusi sempurna dari masalah ini dipersulit pencapaiannya karena penggunaan yang potensial dari ‘*launder*’ (‘pencucian’) yang merupakan inti dari sebuah serangan. Sebagai contoh, sebuah serangan mungkin terdiri dari paket-paket yang dikirim dari berbagai mesin *slave* yang berbeda, yang kesemua mesin tersebut dikendalikan oleh sebuah mesin *master* dari jauh. Ke-tertidaklangsung-an semacam ini dapat tercapai baik secara eksplisit (dengan menyuruh pengelola *slave* secara individual) atau secara implisit (dengan mengirim *request* palsu atas nama korban—hal ini disebut *reflector*). Lebih menantang lagi, asal serangan sesungguhnya dan identitas penyerang dapat disembunyikan lewat serangkaian *computer account* palsu, *call forwarding*, dan sebagainya.

Yang akan dibahas di sini adalah cara mengidentifikasi mesin-mesin yang digunakan untuk menghasilkan serangan secara langsung, dan jalur jaringan (*network path*) yang dilalui oleh serangan. Pada *setting* ini, informasi yang tidak lengkap, atau bahkan informasi teknik, merupakan hal yang berharga. Akan tetapi, bahkan untuk masalah yang terbatas ini, menentukan sumber yang menghasilkan serangan merupakan hal yang luar biasa sukar. Hal ini disebabkan karena sifat *stateless* dari Internet *routing*. Para penyerang secara rutin menyembunyikan lokasi mereka dengan cara menggunakan alamat IP yang salah (*spoofing*). Bersamaan dengan menjalarnya paket-paket serangan di Internet, maka alamat asli sumber paket akan hilang dan korban tinggal memiliki sedikit informasi yang berguna. Pada saat ini telah terdapat beberapa teknik *traceback* khusus yang digunakan, teknik-teknik ini memiliki beberapa kekurangan yang membatasi penggunaan praktisnya pada Internet saat ini.

Teknik yang akan dibahas pada tulisan ini membutuhkan kerjasama antara korban dan operator jaringan (ISP). Solusinya adalah dengan menandai paket-paket data—secara probabilistik—dengan informasi sebagian *path* pada saat informasi ini tiba di *router-router*. Pendekatan ini memanfaatkan kenyataan bahwa serangan biasanya terdiri dari sejumlah besar paket. Dengan demikian, setiap paket yang telah ditandai cukup berisi ‘secuil’

informasi jalur yang telah dilaluinya, dan dengan menyatukan sejumlah paket semacam ini—dalam jumlah yang cukup—korban dapat merekonstruksi keseluruhan jalur serangan. Hal ini memungkinkan korban menentukan kira-kira lokasi sumber serangan tanpa membutuhkan bantuan interaktif dari operator-operator jaringan yang lain. Lebih jauh lagi, penentuan sumber serangan ini dapat dilakukan bahkan setelah serangan terjadi.

2 Beberapa teknik

Telah lama diketahui, bahwasanya protokol IP memungkinkan adanya *anonymous attacks*. Pada tulisannya mengenai kelemahan TCP/IP tahun 1985, Morris menulis:

“Kelemahan skema ini (Internet Protokol) adalah bahwa *source host* sendiri yang mengisi identifikasi IP-nya, dan tidak ada mekanisme pada ...TCP/IP untuk mengungkapkan sumber sebenarnya dari suatu paket.”

Sebagai tambahan pada DoS-attack, IP *spoofing* dapat digunakan bersamaan dengan kelemahan-kelemahan protokol ini lainnya, untuk melakukan *anonymous one-way TCP channels* dan menutupi *port scanning*.

Sebenarnya telah terdapat beberapa usaha untuk mengurangi ke-anonimitas-an yang diperoleh dengan cara IP *spoofing*. Tabel 1 menunjukkan karakteristik setiap teknik dari sudut biaya manajemen (*management cost*), beban tambahan pada jaringan, *overhead* yang terjadi di *router*, kemampuan menelusuri serangan-serangan yang dilakukan secara bersamaan, kemampuan menelusuri serangan setelah serangan selesai dilakukan, dan apakah teknik-teknik tersebut bersifat preventatif atau reaktif. Juga disajikan pada tabel tersebut, karakteristik dari teknik yang diajukan, berdasarkan kriteria yang sama.

	<i>Overhead</i> Manajemen	<i>Overhead</i> Jaringan	<i>Overhead</i> <i>Router</i>	Kemampuan Terdistribusi	Kemampuan <i>Post-mortem</i>	Preventatif / reaktif
Ingress filtering	Moderat	Rendah	Moderat	Tidak ada	Tidak ada	Preventatif
Link testing						
Input debugging	Tinggi	Rendah	Tinggi	Baik	Tidak ada	Reaktif
Controlled flooding	Rendah	Tinggi	Rendah	Buruk	Tidak ada	Reaktif
Logging	Tinggi	Rendah	Tinggi	Sangat baik	Sangat baik	Reaktif
ICMP traceback	Rendah	Rendah	Rendah	Baik	Sangat baik	Reaktif
Marking	Rendah	Rendah	Rendah	Baik	Sangat baik	Reaktif

Tabel 1. Perbandingan kualitatif skema yang ada untuk melawan serangan *anonymous*

A. Ingress Filtering

Salah satu cara untuk mengatasi masalah serangan anonim adalah dengan menghilangkan kemampuan untuk menutupi/mengganti alamat sumber serangan. Teknik ini, sering disebut sebagai *ingress filtering*, dilakukan dengan cara mengkonfigurasi *router-router* agar menahan paket-paket yang datang dengan alamat sumber paket yang tidak legal (*illegitimate*). Teknik semacam ini membutuhkan *router* dengan sumber daya yang cukup untuk memeriksa alamat sumber setiap paket dan memiliki *knowledge* yang cukup besar agar dapat membedakan antara alamat yang legal dan yang tidak. Sebagai konsekuensinya, *ingress filtering* paling

mungkin diterapkan hanya pada jaringan yang kepemilikan alamatnya relatif jelas dan beban *traffic*-nya rendah.

Sebenarnya gerakan *traffic* itu menjalar dari berbagai ISP ke jaringan transit, sehingga informasi yang ada (pada setiap paket data yang diterima) tidak akan cukup untuk digunakan sebagai penentu secara pasti, apakah suatu paket yang tiba memiliki alamat sumber yang legal. Lebih jauh lagi, pada kebanyakan arsitektur *router* yang ada, *overhead* dari *ingress filtering* terasa sangat membebani.

Masalah utama dari *ingress filtering* adalah tingkat ke-efektif-annya sangat tergantung pada penggunaannya yang harus luas—bahkan kalau bisa universal. Malangnya, sebagian besar ISP tidak menerapkan pelayanan ini—entah karena beban administratif¹, *overhead* pada *router* yang potensial, atau komplikasi yang mungkin muncul dengan beberapa jenis pelayanan sebelumnya yang tergantung pada *spoofing* alamat sumber (misalkan beberapa versi dari *Mobile IP* dan beberapa arsitektur satelit komunikasi *hybrid*). Masalah kedua adalah, bahkan jika *ingress filtering* dipasang secara universal pada tingkat pelanggan-ke-ISP, para penyerang tetap dapat mengganti alamat mereka dengan alamat milik beratus bahkan beribu *host* yang merupakan pelanggan valid jaringan tersebut.

Jelaslah bahwasanya penggunaan yang lebih luas dari *ingress filtering* secara dramatis akan meningkatkan kekokohan Internet dari DoS-attack. Pada saat yang sama, layak untuk dibuat asumsi bahwa sistem semacam ini tidak akan pernah benar-benar manjur—dan dengan demikian teknologi *traceback* tetap diperlukan.

B. Link Testing

Kebanyakan teknik *traceback* yang ada saat ini dimulai dari *router* yang terdekat dari korban serangan dan secara interaktif menguji *link-link upstream*-nya sampai dapat ditentukan mana yang digunakan untuk membawa *traffic* serangan. Idealnya, prosedur ini dilakukan berulang secara rekursif pada *router-router upstream* sampai ke sumbernya. Teknik ini mengasumsikan bahwa suatu serangan tetap aktif sampai selesainya proses *trace* dan dengan demikian tidak cocok untuk serangan-serangan yang baru terdeteksi setelah kejadiannya berlangsung, serangan-serangan yang terjadi secara *intermittent* (sekali-sekali), atau serangan-serangan yang mengubah-ubah perilakunya sebagai respon terhadap suatu usaha *traceback* (layak diasumsikan bahwa penyerang memiliki informasi yang lengkap). Berikut disajikan dua jenis skema *link testing*, yaitu *input debugging*, dan *controlled flooding*.

1) Input debugging

Banyak *router* yang dilengkapi dengan fitur yang disebut *input debugging*, sehingga memungkinkan seorang operator menyaring paket-paket tertentu pada *port* keluaran dan menentukan dari *port* masukan mana paket-paket tersebut berasal. Kemampuan ini digunakan untuk menerapkan suatu penelusuran dengan cara sebagai berikut. Pertama-tama, korban serangan harus menyadari bahwa dia sedang diserang dan membangun suatu *attack signature* yang menjelaskan suatu ciri umum yang terdapat pada seluruh paket serangan. Korban akan memberitahukan *tanda tangan* ini ke operator jaringan, biasanya lewat telepon, yang

¹ Beberapa *router* modern mempermudah beban administratif dari *ingress filtering* dengan memberikan fungsi untuk secara otomatis menguji alamat sumber paket berdasarkan tabel *routing* yang *destination-based* (sebagai contoh: *IP verify unicast reverse-path* pada IOS keluaran Cisco). Teknik ini hanya tepat jika jalur ke dan dari pelanggan simetris—umumnya pada batas dari *single-homed stub network*.

kemudian akan meng-*install input debugging filter* yang sesuai pada *port upstream* keluaran si korban. Saringan ini akan mengungkapkan informasi *port* masukan yang berhubungan, dan dengan demikian mengungkapkan *router upstream* mana yang menghasilkan *traffic* tersebut. Proses tersebut kemudian diulang secara rekursif pada *router upstream* berikutnya, sampai ke situs awalnya atau penelusuran tersebut telah melampaui batas ISP (dan dengan demikian kendali administratifnya sudah diluar *router*). Pada kasus terakhir, ISP *upstream* harus dihubungi dan prosedur tadi akan diulangi terus. Walaupun umumnya hal ini dilakukan secara manual, beberapa ISP telah membangun *tools* yang dapat mengotomatiskan proses penelusuran serangan di dalam jaringan mereka. Sistem semacam ini—disebut CenterTrack—memberikan suatu peningkatan pada *hop-by-hop backtracking* dengan cara me-*route*-ulang secara dinamis seluruh *traffic* korban agar mengalir lewat suatu *tracking router* yang tersentralisasi. Pada saat usaha me-*route*-ulang ini selesai, operator jaringan dapat menggunakan *input debugging* pada *tracking router* untuk menyelidiki tempat masuknya serangan ke jaringan ISP tersebut.

Masalah yang paling jelas terlihat pada teknik *input debugging* adalah: bahkan dengan *tools* yang otomatis, akan terdapat *overhead* manajemen yang cukup besar. Komunikasi dan koordinasi antar operator jaringan dari beberapa ISP membutuhkan waktu, perhatian, dan komitmen baik antara korban dan pegawai-pegawai itu—yang tidak mendapatkan insentif ekonomis langsung saat memberikan bantuan. Jika operator jaringan yang tepat tidak ada, jika mereka tidak mau membantu, atau jika mereka tidak memiliki keahlian dan kapabilitas teknis yang cukup, maka suatu proses *traceback* semacam ini akan berjalan lambat atau bahkan tidak mungkin dikerjakan sama sekali.

2) Controlled flooding

Teknik lain adalah teknik *link-testing traceback* yang tidak memerlukan bantuan apapun dari operator jaringan. Teknik ini disebut *controlled flooding* karena teknik ini menguji *link-link* dengan cara membanjirinya dengan *traffic* yang besar dan mengamati bagaimana hal ini mengganggu *traffic* si penyerang. Dengan menggunakan suatu ‘peta’ topologi Internet yang telah dibuat sebelumnya, korban serangan memaksa beberapa *host* pilihan di sepanjang *upstream route* untuk secara iteratif membanjiri setiap *incoming link* pada *router* yang terdekat dari korban. Karena *buffer* dari *router* di-*share*, maka paket-paket yang bergerak di sepanjang *link* yang dimuati—termasuk yang dikirim oleh penyerang—memiliki kemungkinan untuk di-*drop* yang bertambah. Dengan mengamati perubahan pada laju paket-paket yang diterima oleh penyerang, korban kemudian dapat menentukan *link* yang merupakan asal serangan. Seperti halnya skema penelusuran *link* lainnya, prosedur dasar ini kemudian diulang secara rekursif pada *router-router upstream* berikutnya sampai sumber serangan tercapai.

Walaupun ide ini cukup cemerlang—sekaligus pragmatis—ternyata skema ini memiliki beberapa kekurangan dan keterbatasan. Yang paling problematis adalah *controlled flooding*-nya sendiri merupakan DoS-attack yang mengeksploitasi kelemahan pada *host-host* yang tidak bersalah. Kelemahan ini saja telah menyebabkan skema ini tidak cocok untuk penggunaan yang terus-menerus. Hal yang lain, *controlled flooding* mengharuskan korban serangan memiliki suatu peta topologis dari Internet yang baik, sebagai daftar *host-host* yang ‘mau’ dibanjiri. *Controlled flooding* juga sangat jelek untuk diimplementasikan pada kasus DoS-attack yang tersebar/terdistribusi, karena mekanisme *link-testing* dasarnya sangat rumit (*noisy*) dan bisa jadi sulit untuk mendeteksi sekumpulan *path* yang akan dieksploitasi pada saat sejumlah *upstream link* ikut ambil bagian dalam serangan. Akhirnya, seperti semua

skema *link-testing*, *controlled flooding* hanya efektif saat menelusuri suatu serangan yang sedang berlangsung dan tidak dapat digunakan untuk kasus yang bersifat ‘*post mortem*’—setelah serangan berakhir.

C. Logging

Teknik lainnya adalah dengan mencatat paket-paket di *router* kunci dan kemudian dengan menggunakan teknik-teknik *data mining* ditentukan *path* yang dilalui oleh paket-paket tersebut. Skema ini memiliki sifat yang berguna, yaitu bahwa skema ini dapat menelusuri suatu serangan, lama setelah serangan tersebut selesai. Akan tetapi, skema ini juga memiliki kekurangan yang jelas terlihat, yaitu dibutuhkannya sumberdaya yang besar dan masalah integrasi *database* antar *provider* yang juga tidak kalah sulitnya.

D. ICMP traceback

Ide dasar dari skema ini adalah untuk semua *router* yang di-sampel, dengan kemungkinan yang rendah (misalkan 1/20.000), salah satu dari paket akan di-*forward* dan disalinkan isinya ke suatu pesan ICMP *traceback* khusus, termasuk informasi tentang *router-router* sebelumnya di sepanjang *path* ke tujuan. Selama suatu serangan yang berbentuk *flooding*, *host* korban kemudian dapat menggunakan pesan-pesan ini untuk merekonstruksi suatu *path*, mundur sampai ke si penyerang. Skema ini memiliki banyak keuntungan, dibandingkan dengan skema sebelumnya dan mirip dengan teknik *packet marking* yang diusulkan.

Akan tetapi terdapat beberapa kerugian pada disain ini yang menyebabkan sulitnya penggunaan, diantaranya adalah:

- *traffic* ICMP semakin dapat dibedakan, sehingga bisa jadi akan difilter pada jaringan yang sedang diserang;
- pesan ICMP *traceback* bergantung pada kemampuan *input debugging* (yaitu kemampuan untuk menghubungkan suatu paket dengan *port* masukan dan/atau alamat MAC tempat paket tersebut berasal) yang tidak dapat diperoleh pada beberapa arsitektur *router*;
- jika hanya sedikit *router* yang ikut berpartisipasi, kelihatannya sukar untuk secara positif menghubungkan pesan *traceback* dari *router-router* yang berpartisipasi, yang terpisah oleh *router* yang tidak ikut berpartisipasi;
- skema ini membutuhkan infrastruktur khusus—yang dapat menanggulangi masalah pengiriman pesan-pesan ICMP *traceback* palsu yang dikirimkan oleh penyerang.

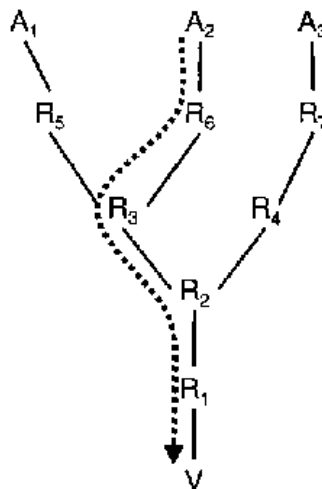
3 Gambaran dasar

Burch dan Cheswick menyebutkan kemungkinan penelusuran serangan *flooding* dengan ‘menandai’ paket-paket, baik secara probalistik maupun deterministik, dengan alamat *router* yang dilalui paket-paket tersebut. Korban kemudian menggunakan informasi ini untuk menelusuri suatu serangan sampai ke sumbernya. Teknik ini belum pernah dieksplorasi, tapi memiliki banyak keuntungan potensial. Skema ini tidak membutuhkan kerjasama interaktif dengan operator ISP dan dengan demikian menghindari *overhead* manajemen yang tinggi dari *input debugging*. Tidak seperti *controlled flooding*, skema ini tidak memerlukan *traffic* tambahan yang signifikan dan dapat secara potensial digunakan untuk menelusuri serangan-serangan berganda. Lebih lanjut lagi, seperti halnya teknik *logging*, teknik penandaan

paket—*packet marking*—dapat digunakan untuk menelusuri serangan, lama setelah serangan tersebut berhenti. Akhirnya, algoritma penandaan dapat diimplementasikan tanpa membutuhkan *overhead* yang berarti pada jaringan *router*.

A. Beberapa definisi

Gambar 1. mengilustrasikan jaringan seperti yang dilihat oleh korban serangan V . V dapat berupa *host* tunggal yang diserang, atau suatu perangkat batas jaringan seperti *firewall* atau *intrusion detection system* yang mewakili banyak *host*. Setiap asal serangan potensial A_i adalah suatu daun pada *tree* yang berakar pada V , dan setiap *router* R_i adalah *node* internal di sepanjang jalur antara A_i dan V . jalur serangan dari A_i adalah daftar *router* berurut antara A_i dan V . sebagai contoh jika suatu serangan berasal dari A_2 , maka untuk mencapai V pertama-tama serangan itu harus melewati jalur R_6 , R_3 , R_2 , dan R_1 —seperti yang ditunjukkan dengan garis putus-putus dalam Gambar 1.



Gambar 1. Jaringan seperti yang dilihat oleh korban suatu serangan V , *router-router* dinyatakan dalam R_i , dan penyerang potensial oleh A_i . Garis putus-putus menyatakan suatu jalur serangan saat itu, yang menghubungkan si korban dengan penyerangnya

Masalah *traceback* sesungguhnya adalah menentukan jalur serangan dan sumber asli setiap serangan untuk setiap penyerang. Akan tetapi, usaha untuk menyelesaikan masalah ini dipersulit oleh beberapa keterbatasan praktis. Asal serangan sesungguhnya mungkin tidak akan pernah diketahui (bahkan MAC *source address* juga dapat di-*spoof*) dan seorang penyerang yang cerdas dapat mengirimkan sinyal palsu untu ‘menciptakan’ *router-router* tambahan pada jalur *traceback*. Hal ini akan dibahas pada Bagian 6, tapi untuk saat ini, kita batasi pembahasan pada penyelesaian yang lebih terbatas. Kita definisikan masalah *approximate traceback* sebagai usaha untuk menemukan suatu kandidat jalur serangan untuk setiap penyerang yang mengandung jalur serangan sesungguhnya sebagai suatu *suffix*. Kita sebut hal ini sebagai *valid suffix* dari suatu kandidat jalur. Sebagai contoh, $(R_5, R_6, R_3, R_2, R_1)$ adalah sebuah solusi valid untuk Gambar 1. sebab merupakan jalur serangan yang valid. Kita akan katakan solusi untuk masalah ini kokoh, jika seorang penyerang tidak dapat mencegah korban serangan menemukan jalur-jalur kandidat yang berisi *suffix* yang valid.

Seluruh algoritma penandaan memiliki dua komponen: sebuah prosedur penandaan yang dieksekusi oleh *router-router* di jaringan dan sebuah prosedur rekonstruksi jalur (*path*

reconstruction procedure) yang diterapkan oleh korban. Sebuah *router* ‘menandai’ satu atau lebih paket dengan menempelkan, suatu tanda yang berisi informasi tambahan mengenai jalur yang dilalui oleh paket. Korban kemudian mencoba untuk merekonstruksi jalur serangan dengan hanya menggunakan informasi ini. Waktu konvergensi (*convergence time*) dari sebuah algoritma didefinisikan sebagai jumlah paket yang harus diamati oleh korban agar dapat direkonstruksi jalur serangannya dengan benar.

B. Beberapa asumsi dasar

Ruang disain dari algoritma-algoritma penandaan sangatlah luas, dan untuk menempatkan skema ini pada konteksnya, kita identifikasi asumsi-asumsi yang akan memotivasi dan membatasi disain skema ini:

- Seorang penyerang bisa men-*generate* paket apapun.
- Bisa jadi ada konspirasi antara beberapa penyerang.
- Para penyerang bisa jadi mengetahui kalau mereka sedang ditelusuri jalurnya.
- Paket-paket bisa jadi hilang atau disusun ulang.
- Para penyerang mengirimkan banyak sekali paket
- Jalur antara penyerang dan korban cukup stabil.
- *Router-router* memiliki keterbatasan sumber daya: CPU dan memori yang terbatas.
- *Router-router* tidak bisa saling berkompromi dalam skala yang dalam.

Keempat asumsi pertama merepresentasikan anggapan konservatif pada kemampuan penyerang moderen dan keterbatasan jaringan. Mendisain sistem *traceback* untuk lingkungan Internet cukup sukar, sebab sedikit sekali yang bisa dipercaya di Internet. Khususnya kemampuan penyerang untuk membuat paket-paket bebas yang secara signifikan membatasi solusi-solusi potensial. Pada saat suatu *router* menerima sebuah paket, *router* ini tidak memiliki cara untuk menentukan apakah paket tersebut telah ditandai oleh *router upstream*-nya atau apakah penyerang telah menipu informasi ini. Pada kenyataannya, satu-satunya kepastian yang kita pegang adalah: sebuah paket dari penyerang harus menjalar di seluruh *router* di antara penyerang dan si korban.

Asumsi sisanya mencerminkan dasar untuk disain skema ini. Pertama, DoS-attack hanya efektif selama serangan ini menguasai sumberdaya si korban. Sebagai konsekuensinya, kebanyakan serangan terdiri dari ribuan atau bahkan jutaan paket. Teknik kita didasarkan pada sifat ini, sebab kita akan menandai setiap paket dengan hanya sedikit bagian dari *path state* dan si korban harus mengamati paket-paket ini untuk merekonstruksi keseluruhan jalur—sampai ke si penyerang. Jika banyak serangan yang muncul hanya membutuhkan paket tunggal untuk menghancurkan suatu *host* (seperti serangan *ping-o-death*), maka asumsi ini tidak dapat berlaku (walaupun telah diperhatikan bahwa bahkan serangan semacam ini tetap memerlukan beberapa paket untuk tetap membuat suatu mesin *down*).

Kedua, pengamatan terhadap bukti-bukti memberikan informasi bahwa memang *router-router* di Internet berubah, akan tetapi sangat jarang terjadi paket-paket mengikuti jalur yang berbeda selama waktu yang skalanya singkat pada suatu operasi *traceback* (dalam hitungan detik). Asumsi ini sangat menyederhanakan tugas si korban, karena dengan demikian membatasi perhatian pada sebuah jalur primer untuk setiap serangan. Jika teknologi Internet berkembang sehingga memungkinkan *routing* yang multijalur pada tingkat yang signifikan, maka asumsi ini tidak dapat berlaku.

Ketiga, selama terjadinya peningkatan teknologi implementasi *router*, kecepatan sambungan/*link* juga bertambah secara dramatis. Sebagai konsekuensinya, kita nyatakan bahwa untuk implementasi manapun, harus memiliki *overhead* per paket yang rendah. Skema yang jauh lebih sederhana dapat diimplementasikan jika kita asumsikan bahwa *router-router* tidak memiliki keterbatasan pada sumberdayanya.

Akhirnya, karena sebuah *router* yang bisa diajak kerjasama oleh penyerang dapat secara efektif menghilangkan informasi apapun yang diberikan *router-router upstream*-nya. Pada kasus semacam ini pelanggaran keamanan di *router* harus diselesaikan terlebih dahulu, sebelum proses *traceback* apapun dilakukan. Jika infrastruktur *routing* yang menyembunyikan informasi suatu saat menjadi lebih populer, maka asumsi ini harus ditinjau ulang.

4 Beberapa algoritma dasar

Pada bagian ini disajikan beberapa algoritma penandaan—mulai dari yang paling sederhana sampai yang paling kompleks. Setiap algoritma mencoba menyelesaikan masalah *traceback* dengan cara yang konsisten dengan asumsi-asumsi yang kita buat pada bagian sebelumnya.

A. Node append

Algoritma paling sederhana—secara konseptual sama dengan metode *IP Record Route*—adalah dengan cara menambahkan alamat *node* pada akhir paket selama paket tersebut menjalar di sepanjang jaringan dari si penyerang sampai ke korban (lihat Gambar 2.). konsekuensinya, setiap paket yang diterima oleh si korban akan datang dengan suatu daftar berurut yang lengkap dari *router-router* yang dilaluinya—suatu jalur serangan yang *built in* pada paket.

Algoritma ini kokoh dan sangat cepat mengkonvergen (hanya membutuhkan sebuah paket tunggal), akan tetapi memiliki beberapa keterbatasan yang serius. Yang paling prinsip adalah *overhead* pada *router* yang sangat besar sehingga tidak dapat diterima yang terjadi karena proses penambahan data ke paket pada saat berjalan (*in-flight*). Lebih jauh lagi, karena panjang dari suatu jalur tidak dapat diketahui secara *a-priori*, sangatlah tidak mungkin untuk menjamin bahwa terdapat cukup tempat (*field*) yang tidak terpakai yang dapat menampung keseluruhan daftar secara lengkap pada paket tersebut. Hal ini dapat menyebabkan fragmentasi yang tidak perlu dan interaksi yang buruk dengan pelayanan-pelayanan seperti penentuan MTU. Masalah ini tidak dapat diselesaikan dengan mencadangkan *field* “yang selalu cukup”, sebab penyerang kemudian dapat mengisi *field* ini dengan informasi jalur yang salah atau menyesatkan.

Prosedur penandaan pada router R:
Untuk setiap paket w , tambahkan R ke w

Prosedur rekonstruksi jalur pada korban v:
Untuk sembarang paket w dari penyerang
Ekstrak jalur $(R_i \dots R_j)$ dari suffix-nya w .

Gambar 2. Algoritma *node append*

B. Node sampling

Untuk mengurangi *overhead* pada *router* dan permintaan lebar *field* di tiap paket untuk menyimpan informasi *node*, kita dapat mengambil contoh/sampel satu *node* pada jalur sekali-sekali saja. Suatu *field* dicadangkan untuk menyimpan *single static "node"* di *header* paket, cukup besar untuk menyimpan satu alamat *router* (misal 32 bit untuk IPv4). Selama menerima sebuah paket, setiap *router* memilih apakah akan menulis alamatnya pada *node field* dengan kemungkinan p . Setelah paket-paket dengan jumlah yang cukup dikirimkan, si korban akan telah menerima setidaknya satu sampel untuk setiap *router* di jalur serangan. Seperti yang telah dibicarakan di Bagian 3, kita asumsikan bahwa penyerang mengirimkan sejumlah cukup paket dan jalurnya cukup stabil sehingga metode *sampling* ini dapat berhasil.

Walaupun terlihat tidak mungkin untuk merekonstruksi sebuah jalur berurut dengan hanya diberikan suatu kumpulan dari sampel-sampel *node* yang tidak berurutan, ternyata dengan sejumlah cukup usaha coba-coba, urutan yang benar dapat diperoleh dari sejumlah sampel tiap *node*. Karena *router-router* disusun secara seri, kemungkinan sebuah paket akan ditandai oleh sebuah *router* dan kemudian dibiarkan tidak diganggu oleh semua *router-router downstream*-nya adalah sebuah fungsi yang monoton turun. Jika kita membatasi p agar harus identik pada setiap *router*, maka kemungkinan penerimaan sebuah paket yang telah ditandai dari *router* yang berjarak d buah *hop* jauhnya adalah :

$$p (1 - p)^{d-1}$$

Karena fungsi tersebut merupakan fungsi monoton turun terhadap jarak dari si korban, usaha untuk mengurutkan setiap *router* berdasarkan nomer sampel yang diberikannya akan menghasilkan jalur serangan yang akurat. Algoritma lengkapnya disajikan dalam Gambar 3.

Sejenuk kita kesampingkan kesulitan pada usaha mengubah *IP header* untuk menambahkan sebuah *node field* 32 bit, algoritma ini efisien untuk diimplementasi sebab hanya membutuhkan penambahan pada usaha menuliskan informasi dan *checksum update* ke jalur berikutnya. *Router-router* dewasa ini telah melakukan operasi ini secara efisien untuk meng-*update field: time-to-live* pada setiap *hop*. Lebih jauh lagi, jika $p > 0,5$ maka algoritma ini kokoh terhadap penyerang tunggal, karena tidak ada cara bagi penyerang untuk menyisipkan sebuah *router* 'palsu' ke dalam *suffix* valid dari jalur dengan memberikan lebih banyak sampel dibandingkan sebuah *router downstream*, atau juga tidak dengan cara mengurutkan ulang urutan *router* pada jalur dengan memberikan lebih banyak sampel dibandingkan beda antara dua *router downstream* manapun.

Akan tetapi terdapat juga dua batasan yang serius. Pertama, mengganggu urutan total *router* dari penyebaran sampel adalah proses yang lambat. *Router-router* yang jauh dari korban memberikan kontribusi sampel yang relatif sedikit (khususnya sejak p haruslah besar) dan variabilitas acak dapat dengan mudah berakhir pada salah urutan, kecuali suatu jumlah sampel yang sangat besardiamati. Sebagai contoh, jika $d = 15$ dan $p = 0,51$, penerima harus menerima lebih dari 42000 paket rata-rata sebelum menerima sebuah sampel tunggal dari *router* terjauh. Untuk menjamin bahwa urutannya benar dengan kepastian 95%, membutuhkan lebih dari tujuh kali angka tersebut.

Kedua, jika terdapat beberapa penyerang, maka beberapa *router* mungkin berada pada jarak yang sama—dan dengan demikian akan disampel dengan kemungkinan sampel. Dengan demikian, teknik ini tidak kokoh terhadap beberapa penyerang yang bekerja sekaligus.

```

Prosedur penandaan pada router R:
Untuk setiap paket w
  Misal x adalah nilai acak dari [0..1)
  If x < p, then
    Tuliskan R pada w.node

Prosedur rekonstruksi jalur pada korban v:
  Misal NodeTbl adalah sebuah tabel dari tuple (node,count)
Untuk setiap paket w dari si penyerang
  z := lookup w.node pada NodeTbl
  If z !=NIL then
    Tambahkan z.count
  Else
    Insert tuple (w.node,1) pada NodeTbl
Urutkan NodeTbl berdasarkan count
Ekstrak jalur (Ri..Rj) dari field node berurut pada NodeTbl

```

Gambar 3. Algoritma *node sampling*

C. Edge sampling

Sebuah jawaban langsung atas masalah sebelumnya adalah dengan secara eksplisit mengkodekan *edge-edge* pada jalur serangan daripada mengkodekan *node-node* secara individual. Untuk melakukan hal ini, kita harus memesan dua *field* statis seukuran alamat *node*, yaitu awal dan akhir, pada setiap paket untuk merepresentasikan *router-router* di setiap akhir *link*, juga sebagai sebuah *field* tambahan kecil untuk merepresentasikan jarak suatu sampel *edge* dari si korban.

Pada saat suatu *router* memutuskan untuk menandai sebuah paket, maka *router* tersebut akan menuliskan alamatnya sendiri ke *field* awal dan menuliskan sebuah nol ke *field* jarak. Atau jika *field* jarak telah bernilai nol, maka hal ini mengindikasikan bahwa paket tersebut telah ditandai oleh *router* sebelumnya. Pada kasus ini, *router* akan menuliskan alamatnya sendiri ke *field* akhir—sehingga merepresentasikan *edge* antara dirinya sendiri dengan *router* sebelumnya—dan mengubah nilai *field* jarak menjadi satu. Akhirnya, jika *router* tersebut tidak menandai paket, maka dia akan selalu menambahkan isi *field* jarak. Hal ini merupakan mekanisme pensinyalan simetris, yang membolehkan proses *sampling* dikeluarkan secara bertambah-tambah—*edges* dikonstruksi hanya antara *router-router* yang berpartisipasi. Proses penambahan mandatoris ini sangat penting untuk meminimalkan usaha *spoofing* yang dilakukan oleh seorang penyerang. Pada saat paket tiba di korban, *field* jarak merepresentasikan jumlah *hop* yang dilalui sejak *edge* yang dikandungnya disampe². Paket apapun yang ditulis oleh penyerang akan memiliki nilai jarak yang lebih besar atau sama dengan panjang dari jalur serangan sesungguhnya. Karena hal inilah, seorang penyerang tunggal tidak dapat menutup-tutupi *edge* manapun di antara mereka dengan korban (untuk serangan tersebar, hal ini tentu berlaku hanya untuk penyerang terdekat). Konsekuensinya, karena kita tidak lagi menggunakan teknik *sampling rank* untuk membedakan sample-sample yang ‘salah’, kita bebas menggunakan nilai sembarang untuk kemungkinan penandaan *p*.

² Adalah hal yang penting untuk selalu meng-*update field* jarak dengan menggunakan suatu proses penjumlahan. Jika *field* jarak dibolehkan untuk di-*wrap*, maka penyerang dapat men-*spoof edges* yang dekat ke korban dengan mengirim paket-paket dengan nilai jarak yang mendekati maksimum.

```

Prosedur penandaan pada router R:
Untuk setiap paket  $w$ 
  Misal  $x$  adalah nilai acak dari  $[0..1)$ 
  If  $x < p$ , then
    Tuliskan  $R$  pada  $w.awal$  dan  $0$  pada  $w.jarak$ 
  Else
    Jika  $w.jarak=0$  then
      Tuliskan  $R$  pada  $w.akhir$ 
      Tambahkan nilai  $w.jarak$ 

Prosedur rekonstruksi jalur pada korban  $v$ :
Misal  $G$  adalah sebuah tree dengan akar  $v$ 
Misal edge pada  $G$  adalah tuple dari (awal, akhir, jarak)
Untuk setiap paket  $w$  dari si penyerang
  If  $w.jarak = 0$  then
    Insert edge ( $w.awal, v, 0$ ) pada  $G$ 
  Else
    Insert edge ( $w.awal, w.akhir, w.jarak$ ) pada  $G$ 
Hapus semua edge ( $x, y, d$ ) dengan  $d \neq jarak$  dari  $x$  ke  $v$  pada  $G$ 
Ekstrak jalur ( $R_i..R_j$ ) dengan menentukan jalur acyclic pada  $G$ 

```

Gambar 4. Algoritma *edge sampling*

Korban menggunakan *edge-edge* yang disampel pada paket-paket tersebut untuk membentuk sebuah *graph* (tidak jauh berbeda dengan Gambar 1) yang terurut sampai sumber serangannya (atau sumber-sumber serangan). Algoritma lengkapnya ditampilkan dalam Gambar 4. Karena kemungkinan untuk menerima sebuah sampel secara geometris lebih kecil jika semakin jauh dari korban, maka waktu yang diperlukan algoritma ini untuk menkonvergen didominasi oleh waktu untuk menerima sebuah sampel dari *router* terjauh, dengan kemungkinan $1/p(1-p)^{d-1}$, untuk sebuah *router* dengan jarak sejauh d buah *hop*. Akan tetapi, terdapat juga kemungkinan yang kecil bagi kita untuk menerima sebuah sampel dari *router* terjauh, tapi bukan dari *router* yang lebih dekat. Kita dapat mengikat efek ini ke suatu faktor sebesar $\ln(d)$ dengan alasan berikut. Secara konservatif kita asumsikan bahwa sampel-sampel dari seluruh d buah *router* muncul dengan kesamaan yang mirip dengan *router* terjauh. Karena kemungkinan ini *disjoint*, kemungkinan dikirimkannya sebuah sampel pada suatu paket tertentu dari sembarang *router* setidaknya sebesar $dp(1-p)^{d-1}$. Akhirnya, nilai percobaan yang dibutuhkan untuk menyeleksi satu dari setiap *item* d yang ber-kemungkinan-sama (*equiprobable*) adalah $d(\ln(d)+O(1))$ ³.

Dengan demikian jumlah paket X yang dibutuhkan oleh korban untuk merekonstruksi sebuah jalur dengan panjang d memiliki ekspektasi terbatas berikut:

$$E(X) < \frac{\ln(d)}{p(1-p)^{d-1}}$$

Sebagai contoh, jika $p = 1/10$, dan jalur serangan memiliki panjang 10, maka korban serangan dapat secara umum merekonstruksi jalur ini setelah menerima paket sebanyak 75 buah dari si

³ Lebih pasti lagi, pernyataannya adalah $d(\ln(d)+\gamma)$, dengan γ merepresentasikan konstanta Euler. Untuk sederhananya, kita abaikan konstanta kecil ini pada saat menjelaskan ekspektasi, walaupun kita masukkan efeknya pada saat kita melakukan perhitungan.

penyerang. Selama $p \leq 1/d$, hasil yang diperoleh berada pada rentang yang tidak jauh dari nilai optimal. Pada tulisan ini selebihnya, akan digunakan $p = 1/25$, karena hanya sedikit sekali jalur yang melebihi nilai ini. Sebagai perbandingan, contoh sebelumnya mengkonvergen dengan hanya 108 paket dengan menggunakan $p = 1/25$.

Algoritma yang sama ini dapat secara efektif mendeteksi serangan-serangan berganda karena para penyerang dari sumber-sumber yang berbeda menghasilkan *edge-edge* yang *disjoint* pada struktur *tree* yang digunakan selama rekonstruksi. Jumlah paket yang dibutuhkan untuk merekonstruksi setiap jalur serangan berbeda dan saling tak bergantung, sehingga jumlah paket yang dibutuhkan untuk merekonstruksi seluruh jalur serangan merupakan fungsi linier dari jumlah penyerang. Akhirnya, *edge sampling* juga kokoh, yaitu untuk *edge* manapun yang lebih dekat daripada penyerang terdekat tidak mungkin di-*spoof*, karena determinasi jarak yang kokoh. Kebalikannya, pada suatu serangan tersebar, hal ini juga berarti bahwa tidak mungkin mempercayai isi *edge* manapun yang lebih jauh dari penyerang terdekat.

Tentu saja, terdapat suatu batasan praktis yang signifikan pada teknik ini, yaitu diperlukannya *field* tempat penyimpanan data tambahan pada *header* paket IP dan dengan demikian tidak *backward compatible*. Pada bagian berikutnya, akan dibahas suatu versi modifikasi dari *edge sampling* yang menjawab masalah ini, walaupun ada sejumlah *cost* yang mengurangi performa dan kekokohan untuk serangan tersebar (*distributed attacks*) yang harus dikorbankan.

5 Pengkodean

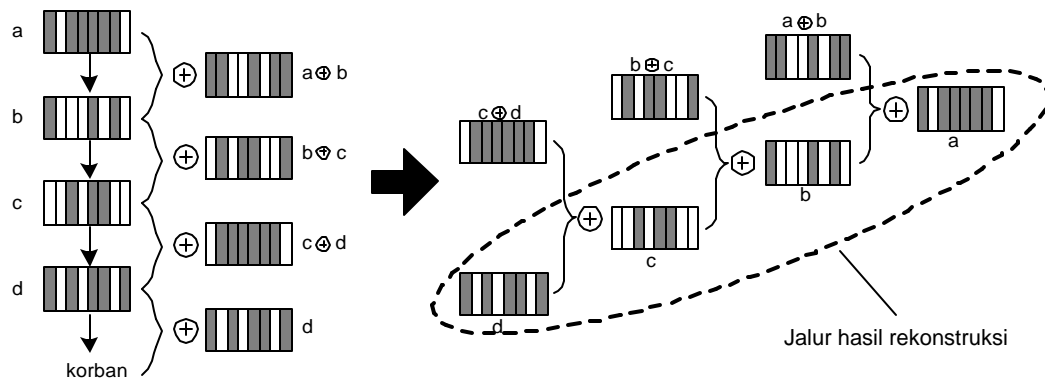
Algoritma *edge-sampling* membutuhkan *field* sebesar 72 bit di setiap paket IP-nya (dua buah 32 bit Alamat IP dan 8 bit “jarak” yang merepresentasikan jumlah *hop* maksimum teoritis yang dibolehkan jika menggunakan IP). Salah satu cara yang mungkin dilakukan adalah dengan menyimpan data *edge sample* ini pada suatu opsi IP, tapi cara ini merupakan pilihan yang buruk dengan alasan yang cukup mirip dengan alasan bahwa algoritma *node append* tidak cukup *feasible*, yaitu menambahkan data tambahan pada sebuah paket secara *in flight* membutuhkan *resource* yang mahal, dan kemungkinan besar tidak terdapat cukup *field* untuk menyimpan data tersebut.

Kita juga dapat mengirimkan data tambahan ini secara *out-of-band*—pada paket yang terpisah—akan tetapi hal ini akan menambah *overhead* pada *router* dan *network*, ditambah dengan kompleksitas protokol baru untuk mengatur hal ini, yang belum tentu kompatibel dengan protokol yang telah ada.

Cara lain yang dikembangkan pada penelitian ini adalah suatu modifikasi dari *edge sampling* yang dapat mengurangi kebutuhan *field* penyimpanan data secara drastis dengan sedikit pengorbanan pada sedikit peningkatan di waktu peng-konvergen-an dan suatu pengurangan nilai kekokohan untuk kasus adanya beberapa penyerang sekaligus.

A. Compressed Edge Fragment Sampling

Penelitian ini menggunakan 3 teknik untuk mengurangi permintaan *field* penyimpanan tiap paket tanpa mengurangi nilai kekokohnya. Pertama, kita kodekan setiap *edge* pada setengah *field* yang tersedia dengan cara merepresentasikannya sebagai *eksklusif-OR* (XOR) dua alamat IP yang membentuk *edge* tersebut, seperti yang ditunjukkan dalam Gambar 5.

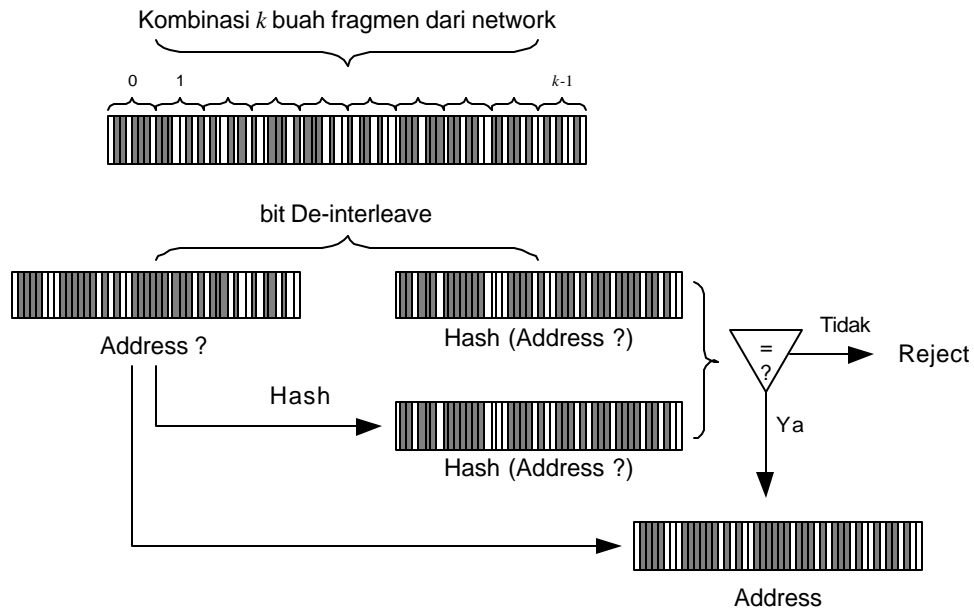


Gambar 5. Data mengenai *edge* dapat dipersingkat menjadi $\frac{1}{2}$ ukuran asli dengan cara melakukan proses logika XOR dari dua *node* (alamat IP-nya dua buah *router*), sehingga membentuk suatu *edge*, daripada mengirimkan data setiap *node* secara terpisah. Si korban akan menerima paket dengan data d , $c \oplus d$, $b \oplus c$, dan $b \oplus a$. Dengan meng-XOR-kan data-data tersebut, jalur aslinya dapat direkonstruksi.

Pada saat suatu *router* memutuskan untuk menandai sebuah paket, maka *router* tersebut akan menulis alamatnya, a , pada paket. *Router* berikutnya, b , sadar bahwa *field* penyimpanan jarak bernilai 0 dan membaca a dari paket tersebut (diasumsikan b tidak menandai paket itu lagi), maka b akan men-XOR nilai ini dengan alamatnya sendiri, dan menuliskan nilai hasil operasi ini, $a \oplus b$, pada paket, menggantikan nilai a . Kita sebut nilai hasil operasi $a \oplus b$, adalah *edge-id* untuk *edge* antara a dan b . Semua *edge-id* yang diterima oleh korban serangan selalu berisikan hasil XOR dari 2 *router* berurutan, kecuali untuk *sampel-sampel* dari *router* yang jaraknya hanya 1 *hop* dari si korban. Perhatikan bahwa $b \oplus a \oplus b = a$, paket-paket bertanda dari *router* final dapat digunakan untuk men-dekode-kan *edge-id* sebelumnya, dan demikian terus-menerus, *hop-demi-hop* sampai kita peroleh *router* pertama.

Teknik kedua yang dikembangkan lebih mengurangi permintaan *field* penyimpanan di tiap paket dengan cara membagi setiap *edge-id* menjadi beberapa k buah fragmen yang tidak saling tumpuk-yang lebih kecil. Pada saat sebuah *router* memutuskan untuk menandai suatu paket, *router* ini akan memilih secara acak salah satu dari fragmen yang ada dan menyimpannya pada paket. Kita gunakan beberapa bit tambahan ($\log_2 k$) untuk menyimpan nilai *offset* dari fragmen ini, hal ini penting agar nantinya fragmen-fragmen *edge-id* yang berbeda dapat dikombinasikan kembali dengan benar sehingga menghasilkan alamat yang benar dari *router* tersebut (berupa alamat IP). Jika jumlah paket yang dikirimkan oleh penyerang cukup, maka pada akhirnya si korban akan menerima seluruh fragmen dari seluruh *edge-id*.

Akhirnya, tidak seperti alamat IP yang lengkap, fragmen-fragmen *edge-id* tidaklah unik dan beberapa fragmen dari *edge-id* yang berbeda bisa saja memiliki nilai yang sama. Pada kasus adanya beberapa penyerang sekaligus, si korban bisa jadi menerima beberapa fragmen *edge* dengan *offset* dan jarak yang sama. Untuk mengurangi kemungkinan rekronstruksi suatu *edge-id* dari fragmen-fragmen yang berasal dari jalur yang berbeda, kita coba memberikan kode pendeteksi kesalahan sederhana pada algoritma yang kita buat. Caranya, kita menambahkan ukuran setiap alamat *router* dengan cara melakukan *bit-interleaving address* IP-nya dengan suatu *random hash* miliknya sendiri. Hal ini digambarkan dalam Gambar 7.



Gambar 7. Pada saat merekonstruksi suatu kandidat *edge*, si korban mengkombinasikan *k* buah fragmen untuk menghasilkan suatu *bit string*. Dengan men-*De-interleaving* string tersebut, bagian alamat dan bagian *hash* akan diperoleh. Kita kembali melakukan *hash* terhadap bagian alamat dengan fungsi *hash* yang sama dengan yang digunakan oleh *router*. Jika hasil *hash* nya sama dengan bagian *hash* yang dihasilkan dari proses ekstraksi, maka alamat tersebut diterima sebagai alamat yang valid. Prosedur ini digunakan agar mencegah proses kombinasi fragmen-fragmen yang berasal dari *edges* yang berbeda.

Seperti yang diperlihatkan dalam Gambar 7, suatu kandidat *edge-id* hanya diterima jika bagian *hash* -nya cocok dengan bagian data di setiap dua *nodes*. Jika kita tambahkan ukuran *hash*, kemungkinan terjadinya *collision* menjadi berkurang. Prosedur lengkap untuk hal ini dijelaskan dalam Gambar. 8.

Jumlah paket yang diharapkan untuk algoritma ini agar meng-konvergen sama dengan teknik *edge sampling*, kecuali sekarang kita memerlukan *k* buah fragmen untuk setiap *edge-id*-nya, dengan total *kd* buah fragmen. Jika kita kembali mengasumsikan bahwa setiap fragmen yang dikirimkan ini dengan ber-kemungkinan-sama (*equiprobable*) dengan besarnya kemungkinan $p(1-p)^{d-1}$, maka jumlah paket yang dibutuhkan untuk rekonstruksi jalur dibatasi oleh batas atas:

$$E(X) < \frac{k \cdot \ln(kd)}{p(1-p)^{d-1}}$$

Sebagai contoh, jika terdapat 8 buah fragmen untuk setiap *edge-id*, seorang penyerang terpisahkan sejauh 10 *hop*, dan $p = 1/25$, maka si korban dapat merekonstruksi jalurnya secara keseluruhan setelah memperoleh rata-rata kurang dari 1300 buah paket. Dengan menggunakan teknik yang sama dengan yang digunakan untuk menunjukkan hasil konsentrasi pada masalah *coupon collector*, kita dapat menunjukkan bahwa jumlah (teknik) paket yang dibutuhkan untuk memastikan bahwa suatu jalur dapat direkonstruksi dengan kemungkinan $1 - 1/c$ adalah:

$$\frac{k \cdot \ln(kdc)}{p(1-p)^{d-1}} \text{ buah paket}$$

untuk merekonstruksi secara lengkap, jalur sebelumnya dengan nilai kepastian sebesar 95% akan membutuhkan tidak lebih dari 2500 paket. Kebanyakan DoS-attack mengirimkan paket sebanyak ini dalam hanya beberapa detik saja.

```

Prosedur penandaan pada router R:
  misal  $R' = \text{BitIntereave}(R, \text{Hash}(R))$ 
  misal  $k$  adalah nomer dari non-overlapping fragmen pada  $R'$ 
  untuk setiap paket  $w$ 
    misal  $x$  adalah angka acak dari  $[0..1)$ 
    if  $x < p$  then
      misal  $\sigma$  adalah a random integer from  $[0..k - 1]$ 
      misal  $f$  adalah the fragmen of  $R'$  at offset  $\sigma$ 
      tulis  $f$  ke dalam  $w.\text{frag}$ 
      tulis  $0$  ke dalam  $w.\text{jarak}$ 
      tulis  $\sigma$  ke dalam  $w.\text{offset}$ 
    else
      if  $w.\text{jarak} = 0$  then
        misal  $f$  adalah fragmen dari  $R'$  pada offset  $w.\text{offset}$ 
        tulis  $f \oplus w.\text{frag}$  ke dalam  $w.\text{frag}$ 
        increment  $w.\text{jarak}$ 

Prosedur rekonstruksi jalur pada korban v:
  misal  $\text{FragTbl}$  adalah a table of tuples (flag,offset,jarak)
  misal  $G$  adalah a tree with root  $v$ 
  misal  $\text{edges}$  in  $G$  adalah tuples (awal,akhir,jarak)
  misal  $\text{maxd} := 0$ 
  misal  $\text{last} := v$ 
  untuk setiap paket  $w$  from penyerang
     $\text{FragTbl.Insert}(w.\text{frag}, w.\text{offset}, w.\text{jarak})$ 
    if  $w.\text{jarak} > \text{maxd}$  then
       $\text{maxd} := w.\text{jarak}$ 
  for  $d := 0$  to  $\text{maxd}$ 
    untuk seluruh kombinasi berurut dari fragmen dengan jarak  $d$ 
      konstruksikan  $\text{edge } z$ 
      if  $d \neq 0$  then
         $z := z \oplus \text{last}$ 
      if  $\text{Hash}(\text{EvenBits}(z)) = \text{OddBits}(z)$  then
        sisipkan  $\text{edge } (z, \text{EvenBits}(z), d)$  ke dalam  $G$ 
         $\text{last} := \text{EvenBits}(z)$ ;
  buang semua  $\text{edge } (x, y, d)$  dengan  $d \neq \text{jarak}$  dari  $x$  ke  $v$  pada  $G$ 
  ekstrak jalur  $(R_i..R_j)$  dengan me mberi nomer jalur-jalur acyclic pada  $G$ 

```

Gambar 8. Algoritma *edge fragment sampling* yang dikompres

Akhirnya, kita selidiki bagaimana kekokohan algoritma ini terhadap banyak serangan sekaligus. Untuk suatu *random hash* dengan panjang h , kemungkinan diterimanya kandidat *edge-id* bebas terkonstruksi adalah $1/2^h$. Pada kejadian adanya m orang penyerang, maka

berapapun jarak d , pada kasus terburuk, mungkin akan ada sebanyak-banyaknya m buah *router* yang berbeda⁴. Sebagai konsekuensinya, kemungkinan *edge-id* manapun dengan jarak d menerima kesalahan, paling banyak adalah:

$$1 - \left(1 - \frac{1}{2^h}\right)^m$$

karena terdapat m^k buah kombinasi fragmen yang mungkin pada kasus terburuknya. Untuk $h = 32$ dan $k = 4$, hal ini berarti bahwa 100 buah *router* yang berbeda pada jarak yang sama (jalur serangan yang *disjoint*) akan diselesaikan tanpa kesalahan dengan kemungkinan yang lebih baik dari 97%. Untuk $h = 32$ dan $k = 8$, nilai kepastian yang sama hanya dapat dihasilkan untuk 10 buah *router* berbeda pada jarak yang sama. Penggunaan fungsi XOR semakin memperumit proses rekonstruksi, sebab seluruh kombinasi dari nilai XOR harus dicoba karena jalur serangan yang menyebar (*diverge*). Hal ini sesungguhnya meringankan, sebab kemungkinan menjalarnya kesalahan dari sebuah *edge* tunggal ke seluruh jalur menuju penyerang sangatlah kecil, karena hasil *edge-id*, saat di-XOR dengan *edge-id* sebelumnya, pasti menghasilkan sebuah *hash* yang benar.

Kelemahan paling penting yang ada pada teknik ini adalah besarnya jumlah kombinasi yang harus diperhitungkan. Sementara kombinasi-kombinasi ini dapat dihitung secara *off-line*, untuk nilai k dan m yang besar, perhitungan kombinasi ini dapat menjadi hal yang sukar untuk dikendalikan (lama). Sebagai contoh, bahkan dengan $k=8$ dan $m=10$, jika jalur-jalur serangan yang berbeda berpecah, misalkan ada 10 buah *edge* independen di tiap penyerangnya, hal ini akan membutuhkan setidaknya satu milyar kombinasi yang harus diperhitungkan.

B. IP Header Encoding

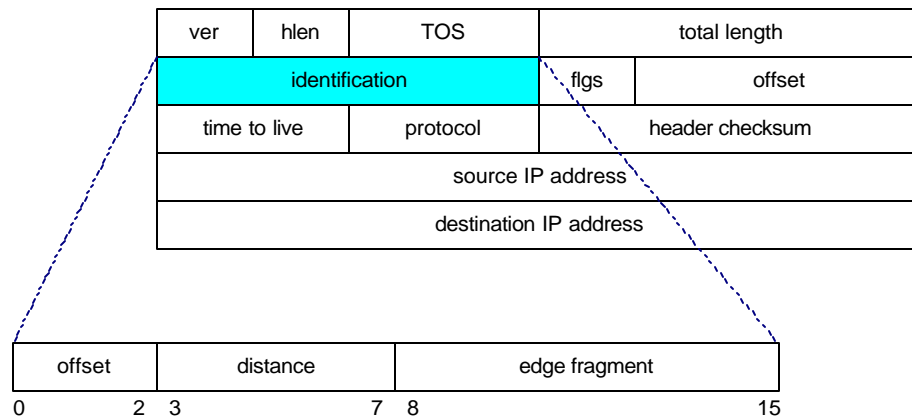
Agar algoritma yang ditawarkan dapat diaplikasikan pada tataran praktis, kita harus dapat menyisipkan informasi pada IP *header* dalam cara dan ukuran yang sedemikian rupa sehingga dapat meminimalkan efek buruk pada *user-user* yang telah ada sebelumnya.

Hal ini adalah tugas yang sulit, terlebih karena ternyata kita membutuhkan ruang sebesar 16 bit. Akan tetapi, kita percaya bahwa informasi sebesar ini dapat disisipkan pada *field* identifikasi IP pada IP *header*. *Field* ini biasanya digunakan untuk membedakan fragmen-fragmen IP yang berasal dari paket-paket yang berbeda. Berikut ini akan disajikan pengkodean yang kita ajukan, kemudian akan dibahas juga masalah *backward-compatibility* yang muncul, walaupun masalah *backward-compatible* sebenarnya merupakan pokok bahasan yang berbeda cukup jauh dari pokok bahasan algoritma *traceback* yang kita bahas.

Gambar 9. menjelaskan cara untuk mempartisi *identification field*: 3 bit *offset* menyatakan 8 macam *fragmen* yang mungkin ada, 5 bit menyatakan 'jarak', dan 8 bit menyatakan fragmen *edge*. Kita gunakan *hash* sebesar 32 bit, yang akan melipat-gandakan ukuran alamat *router* menjadi 64 bit. Hal ini berimplikasi pada dibutuhkannya 8 fragmen terpisah untuk menyatakan setiap *edge*—setiap fragmen diberi tanda nomer offset yang unik. Lima bit yang

⁴ Pada praktiknya, jumlah *router* yang berbeda biasanya lebih kecil dari panjang jalur yang terdekat dengan penerima, karena banyak penyerang akan tetap membagi jalur serangan mereka dalam porsi yang signifikan antara satu dengan yang lainnya.

digunakan untuk menyatakan ‘jarak’, yaitu jumlah *hop* maksimum sebesar 31 *hop*, merupakan nilai yang cukup, sebab melebihi jumlah maksimum *hop* di Internet umumnya⁵.



Gambar 9. Tempat menuliskan *edge fragments* di dalam IP *identification field*

Teknik ini dipilih agar implementasi perangkat lunak memiliki performa yang paling tinggi. Pada kasus yang umum, satu-satunya modifikasi pada paket adalah menambah *distance field*-nya (*field* ‘jarak’), hal ini berdampak secara langsung pada pengurangan nilai pada *time-to-live* (TTL)*field* yang dilakukan oleh setiap *router*—dengan hanya melakukan penjumlahan di *distance field*, pengurangan di TTL *field*, dan perbandingan untuk memeriksa apakah di antara kedua *field* ini ada yang sudah *overflow*. Pada kasus terburuk, algoritma kita ini harus membaca IP *identification field*, mengacu sebuah *edge fragment* dan meng-XOR-nya, kemudian melakukan prosedur *checksum*.

Karena kita menggunakan IP *identification field*, maka kita harus dapat menjawab permasalahan *backward compability* pada *traffic*-nya IP *fragment*. Sebetulnya tidak ada solusi sempurna untuk masalah ini, sehingga kita terpaksa melakukan kompromi yang merugikan *traffic* yang terfragmentasi. Untung saja, penelitian baru-baru ini menyatakan bahwa kurang dari 0,25% paket-paket saja yang terfragmentasi. Bahkan, telah lama dimengerti bahwa fragmentasi *network layer* sangat merusak performa *end-to-end*, sehingga *stack* pada jaringan komputer moderen menerapkan MTU *discovery* otomatis untuk menghindari fragmentasi. Dengan demikian, kita yakin pengkodean yang diajukan ini akan beroperasi secara baik dengan protokol yang sudah ada. Walaupun demikian, memang terdapat sebagian kecil *traffic* legal yang terfragmentasi.

Normalnya, jika suatu paket terfragmentasi, *identification field*-nya akan disalin ke setiap fragmen sehingga si penerima akan dapat menyusun kembali fragmen-fragmen tersebut menjadi paket aslinya. Prosedur penandaan yang kita buat akan merusak sifat tersebut lewat salah satu dari dua cara berikut: dengan menuliskan nilai-nilai yang berbeda pada *identification field* fragmen-fragmen yang berasal dari datagram yang sama, atau dengan menuliskan nilai yang sama pada *identification field* fragmen-fragmen yang berasal dari datagram yang berbeda. Kedua masalah ini membutuhkan cara penanganan dan solusi yang berbeda.

⁵ Merupakan hal yang masuk akal untuk mematikan mekanisme penandaan pada *router-router* yang tidak dapat dihubungkan secara langsung dengan sebuah *attacking host* (misalnya *core routers*). Hal ini sekaligus akan mengurangi waktu konvergensi dan menambah ‘jangkauan’ *distance field*.

Pertama, sebuah datagram dapat saja merupakan *upstream* terfragmentasi dari suatu *marking router* (*router* penanda). Jika fragmen tersebut telah ditandai dan fragmen berikutnya dari datagram yang sama tidak ditandai secara konsisten, maka proses pengaturan ulang fragmen bisa jadi gagal atau datanya menjadi *corrupted*. Solusi yang paling sederhana adalah hanya dengan cara tidak menandai fragmen-fragmen. Akan tetapi hal ini akan menjadi kelemahan yang mudah dieksploitasi, bahkan sesungguhnya beberapa DoS-attack saat ini telah memanfaatkan fragmen-fragmen IP untuk mengeksploitasi kesalahan pada fungsi pengaturan ulang fragmen di IP *host*. Dengan demikian, kita ajukan mekanisme penandaan lain—khusus untuk menandai fragmen. Kita gunakan suatu probabilitas penandaan lain, q , untuk fragmen. Pada saat kita memutuskan untuk menandai suatu fragmen, kita berikan suatu ICMP “*echo reply*” *header* baru, bersama-sama dengan data *edge* lengkap—yang ditempelkan di akhir paket. Paket ICMP ini akan dianggap sebagai paket yang ditandai dan *field* ‘jarak’-nya diset nol (yang akan menjamin bahwa *field* ini mencerminkan jumlah *edge* yang dilalui paket menuju ke si korban). Paket ini ‘hilang’ dari sudut pandang penerima, tapi informasi *edge*-nya dikirimkan dengan suatu cara yang tidak akan memberikan efek merusak pada *legacy host*. Karena kita dapat menggunakan algoritma *edge sampling* secara lengkap, maka q dapat bernilai lebih dari suatu tingkat yang lebih kecil dari p dan dengan demikian dapat mencapai waktu konvergensi yang sama. Solusi ini meningkatkan nilai *loss rate* dari *fragmented flow* (lebih terlihat pada jalur-jalur yang lebih panjang) tapi menjamin integritas data pada *flow-flow* tersebut.

Masalah yang lebih pelik adalah fragmentasi yang terjadi *downstream* dari suatu *router* penanda. Jika suatu paket yang ditandai terfragmentasi, tapi satu dari fragmennya hilang, maka fragmen sisanya bisa jadi berada pada *buffer* penyusunan ulang si korban untuk waktu yang cukup lama, tetapi tidak bisa diolah. Paket-paket berikutnya yang ditandai oleh *router* yang sama dapat memiliki nilai identifikasi IP yang sama dan konsekuensinya akan disusun secara salah dengan fragmen sebelumnya. Salah satu kemungkinannya adalah untuk membiarkan hal ini ditangani oleh *layer checksum* yang lebih tinggi. Akan tetapi, tidak semua protokol *layer* yang lebih tinggi memberlakukan *checksum*, dan pada kasus apapun, sangatlah berbahaya untuk mengandalkan pada *checksum* semacam ini karena biasanya mekanismenya didisain hanya untuk kesalahan yang kecil. Solusi lainnya adalah dengan men-set *flag* ‘Jangan di-fragmentasi’ pada setiap paket yang ditandai.

6 Keterbatasan

Pada teknik yang dibahas ini masih terdapat sejumlah keterbatasan. Kita bahas beberapa keterbatasan yang paling penting, yaitu:

- *Backward compability*;
- Serangan tersebar;
- Validasi jalur;
- Teknik penentuan asal serangan;

A. *Backward compability*

Pengkodean *header* IP yang telah kita bahas sebelumnya memiliki sejumlah keterbatasan praktis. Hal ini akan mempengaruhi (secara negatif) pengguna yang membutuhkan datagram IP yang terfragmentasi, dan tidak kompatibel dengan beberapa IPsec (*header* autentifikasi

yang menghasilkan proteksi kriptografi untuk *identification field* dan dengan demikian *field* ini tidak bisa dengan mudah dimodifikasi oleh *router*). Salah satu usaha untuk menjawab masalah ini adalah dengan secara selektif memberlakukan dukungan *traceback* sebagai respons terhadap kebutuhan operasional. Suatu “permintaan untuk *traceback*” dari suatu jaringan dapat dikodekan sebagai atribut BGP di ‘iklan’ *route* jaringan-jaringan di Internet. *Router-router* yang menerima ‘iklan’ semacam ini akan memberlakukan dukungan *traceback* pada paket-paket yang ditujukan untuk jaringan yang meminta fasilitas tersebut. Karena jaringan yang meminta dukungan semacam ini dapat dianggap telah mengalami serangan, degradasi pelayanan yang relatif kecil terhadap *flow-flow* terfragmentasi seharusnya bisa diterima.

Akhirnya, skema kita ini tidak dapat diterapkan langsung pada IPv6, yang tidak memiliki suatu *identification field*. Skema ini harus dimodifikasi, mungkin dengan cara menulisi *flow label field* yang berukuran 24 bit (yang jika dilakukan tanpa modifikasi lebih lanjut akan mengakibatkan jumlah paket yang dibutuhkan untuk merekonstruksi jalur meningkat).

B. Serangan tersebar

Untuk serangan tersebar pada umumnya, implementasi teknik ini memiliki keterbatasan serius yang disebabkan karena sulitnya cara pengelompokan fragmen-fragmen dengan benar. Sebagai konsekuensinya, probabilitas dari kesalahan pemberian atribut suatu *edge*, demikian pula dengan jumlah *state* yang dibutuhkan untuk mengevaluasi keputusan ini, bertambah sangat cepat berdasarkan tersebarannya suatu serangan.

C. Validasi jalur

Sejumlah paket yang dikirimkan oleh penyerang tidak akan ditandai oleh *router* yang terlibat (ikut-ikutan menyerang). Si korban tidak dapat membedakan paket-paket ini dengan paket-paket bertanda yang asli. Oleh karena itu, seorang penyerang dapat menyisipkan *edge-edge* ‘palsu’ dengan secara hati-hati memanipulasi *identification field* pada paket-paket yang dikirimkannya. Semenjak *field* ‘jarak’ menghalangi penyerang dari men-*spoof edge-edge* diantara penyerang dan korban—yang kita sebut dengan istilah *valid suffix*—tidak ada yang dapat mencegah penyerang dari men-*spoof edge-edge* tambahan di akhir jalur serangan sesungguhnya.

Ada beberapa cara untuk mengidentifikasi *valid suffix* di dalam suatu jalur yang dihasilkan oleh prosedur rekonstruksi. Dengan pengetahuan topologi Internet yang minimal, seseorang dapat membedakan *router* yang termasuk ke dalam jaringan transit (misalnya ISP) dengan *router-router* yang termasuk ke dalam jaringan *stub* (misalnya jaringan perusahaan). Umumnya, suatu jalur valid tidak akan pernah memasuki jaringan *stub* dan kemudian terus memasuki jaringan transit. Lebih jauh lagi, *tool-tool* pengecekan sederhana seperti traceroute dapat digunakan oleh korban serangan untuk menentukan apakah dua buah jaringan benar-benar saling terhubung. Peta jaringan yang lebih lengkap dapat menyelesaikan masalah ini pada sejumlah kasus yang semakin lama semakin bertambah jumlahnya.

Mekanisme yang lebih umum adalah dengan memberi setiap *router* dengan suatu ‘kode rahasia’ yang bervariasi terhadap waktu yang digunakan untuk mengautentifikasikan setiap paket yang ditandai (setidaknya satu bit pada *header IP*). Pada saat si korban ingin memvalidasi sebuah *router* pada jalur, maka korban dapat menghubungi jaringan yang bersangkutan (biasanya menggunakan telepon atau e-mail) dan memperoleh ‘kode rahasia’ yang digunakan oleh *router* tersebut pada saat serangan. Untuk menghindari penggunaan

ulang, ‘kode rahasia’ ini harus bervariasi relatif cepat dan di-*hash* dengan isi paket. Karena penyerang tidak akan tahu ‘kode rahasia’ *router*, maka fragmen-fragmen *edge-id* yang dimanipulasi oleh si penyerang tidak akan memiliki suatu kode autentifikasi yang tepat. Dengan menghilangkan *edge-id* yang fragmen-fragmennya tidak dapat divalidasi, maka kandidat jalur serangan dapat dipangkas menjadi hanya yang mengandung *valid suffix*.

D. Teknik penentuan asal serangan

Walaupun algoritma *traceback* pada tingkat IP yang kita bahas ini bisa jadi merupakan bagian penting dari solusi untuk menghentikan DoS-attack, akan tetapi tidak berarti merupakan solusi yang lengkap. Algoritma kita ini mencoba untuk menentukan kira-kira sumber *traffic* suatu serangan, khususnya *router* paling awal yang dapat dilacak, yang terlibat dalam penyaluran *traffic* serangan dari sumber yang secara langsung menghasilkannya. Seperti yang telah dijelaskan sebelumnya, terdapat sejumlah alasan mengapa hal ini berbeda dengan sumber sesungguhnya dari suatu serangan: para penyerang dapat menyembunyikan identitas asli mereka dengan cara ‘mencuci’ serangan tersebut lewat pihak ketiga, baik itu secara tidak langsung (misalnya lewat *smurf attack*, atau DNS *refectors*) atau secara langsung lewat mesin-mesin “*stepping stone*” yang telah berkompromi, atau lewat IP-in-IP *tunnels*. Sebuah kemungkinan yang menarik yang dihasilkan oleh teknik penandaan paket ini adalah dengan mengembangkan penelusuran lewat “titik-titik pencucian”. Sebagai contoh, penentuan tanda dapat disalin dari sebuah paket *request* DNS ke *reply* DNS yang berhubungan, sehingga membolehkan korban serangan menelusuri jalur penyebabnya secara lengkap. Akan tetapi, hal ini juga akan menambah panjang jalur yang dibutuhkan untuk direkonstruksi—bisa jadi lebih panjang dari *field* ‘jarak yang terbatas.

Bahkan jika tidak ada kasus ‘pencucian’, teknik kita ini tidak mengungkapkan *host* sesungguhnya yang menghasilkan suatu serangan. Lebih jauh lagi, karena *hosts* dapat menyembunyikan baik alamat IP maupun alamat MAC mereka, maka sumber dari sebuah paket mungkin tidak akan bisa diungkap. Pada shared media seperti FDDI *ring*, masalah ini hanya dapat diselesaikan dengan pengecekan secara langsung. Pada media *point-to-point*, *port* masukan-tempat kedatangan-suatu paket biasanya cukup untuk menentukan sumber asli paket tersebut. Pada media lain, mungkin terdapat suatu alamat MAC, nomer *cell*, *channel*, atau informasi lain yang akan menolong penentuan asal serangan. Secara prinsip, algoritma kita ini dapat dimodifikasi untuk melaporkan informasi semacam itu dengan cara sekali-sekali menandai paket dengan suatu *edge-id* yang merepresentasikan *link* antara *router* dan *port* masukan tempat datangnya paket (atau informasi-informasi lainnya).

Akhirnya, metode *traceback* hanya efektif untuk menemukan sumber dari *traffic* serangan, tidak mesti si penyerangnya. Bahkan dengan dukungan *traceback* yang sempurna sekalipun, usaha pengidentifikasian penyerang yang sangat lihai dan hati-hati, besar kemungkinannya memerlukan kerjasama yang baik antara penegak hukum dan organisasi telekomunikasi.

7 Simpulan

Pada tulisan ini, telah dibahas algoritma-algoritma *traceback* berdasarkan pada proses penandaan paket di jaringan. Telah ditunjukkan pula bahwa jenis algoritma semacam ini dapat menghasilkan upaya penelusuran serangan yang bersifat efisien dan kokoh, yang dapat diluncurkan lebih dari satu upaya penelusuran pada satu waktu dan dapat diimplementasikan secara efisien. Demikian pula telah dibangun beberapa algoritma variasi yang mengorbankan

waktu kekonvergenan dan kekokohan untuk mengurangi permintaan tempat penyimpanan informasi pada *field* di tiap paketnya. Akhirnya, telah ditunjukkan pula sebuah strategi peluncuran potensial menggunakan algoritma berdasarkan penulisan *field-field* pada *header* IP yang sudah ada, dan telah ditunjukkan pula bahwa implementasi strategi ini dapat melakukan penelusuran secara lengkap suatu serangan setelah hanya menerima beberapa ribu paket saja. Beberapa area masih harus diteliti lebih lanjut, misalnya untuk menambah tingkat kekokohan pada serangan yang tersebar.

Pustaka

- A.C. Snoeren, et.al., *Hash -based IP Traceback*, SIGCOMM 2001
- M. Adler, *Tradeoffs in Probabilistic Packet Marking for IP Traceback*, to appear in Proceedings of 34th ACM Symposium on Theory of Computing (STOC) 2002
- Naugle, Matthew, *Network Protocol Handbook*, McGraw-Hill International, 1994
- S. Savage, et.al., *Network support for IP traceback*, ACM/IEEE Transactions on Networking, 9(3):226-237, June 2001