FRAMEWORK UNTUK MENDETEKSI BOTNET KRAKEN DAN CONFICKER PADA JARINGAN KOMPUTER

GuruhFajar Shidik[1], Aisyatul Karima[2]
[1,2]Faculty of Information and Communication Technology , University Technical Malaysia Melaka 1752
E-mail : fajar_gro@yahoo.com[1], E-mail : aisyatul.karima@gmail.com[2]

ABSTRAK

Botnet adalah malware yang dapat melakukan serangan terhadap suatu jaringan secara tergorganisir dimana malware ini juga dapat dikendalikan dari pusat atau command and control (C&C). Dengan membedakan kondisi normal dan abnormal traffic pada jaringan komputer, dapat digunakan sebagai indikasi keberadaan Botnet khususnya Kraken dan Conficker. Untuk membedakan kondisi normal dan abnormal jaringan komputer, dapat dilakukan dengan menggunakan anomaly based detection. Dimana dengan anomaly based detection kita dapat mendeteksi Botnet secara dini dengan membandingkan suatu traffic pada jaringan komputer secara visual. Akan tetapi metode anomaly based detection masih belum dapat mendeteksi Botnet secara tepat, masih terdapat dugaan false rate yang tinggi. Untuk mengarahkan metode ini agar terfokus untuk mendeteksi Botnet, diperlukan sebuah framework yang dapat memberi penjelasan akan tahapan yang harus dilakukan. Paper ini memberikan sebuah framework yang berisi langkah-langkah kerja guna mendeteksi Botnet Kraken dan Conficker dengan memanfaatkan metode anomaly based detection.

Kata kunci : *Botnet*, *flow traffic*, anomali *traffic*

1. LATAR BELAKANG

Botnet adalah ancaman yang paling serius dalam keamanan di dunia *cyber*, karena *botnet* menyediakan platform yang didistribusikan untuk beberapa kegiatan ilegal seperti penyebaran serangan *Denial of Service* (*DoS*), penyebaran *malware*, *phising*, dan *click fraud* [1]. Secara teknis, *botnet* merupakan jaringan komputer *zombie* atau komputer yang telah diambil alih kendali dan dikontrol oleh user yang tidak bertanggung jawab tanpa sepengetahuan pengguna komputer yang sah. Penjahat bisa melakukan berbagai macam tidak pidana bermotif finansial maupun pencurian informasi dengan jaringan *botnet*.

Untuk mendeteksi suatu jaringan yang telah terinfeksi oleh *botnet*, dapat dideteksi dengan menggunakan *anomaly based detection*. Dengan metode ini *network traffic* dapat dibedakan baik yang normal maupun abnormal. Karena *anomaly based detection* menggunakan pendekatan untuk mendeteksi *botnet* berdasarkan pada *flow traffic* di jaringan seperti *volume traffic*, *average lattency, port traffic,* dan dari *behaviour botnet* yang biasa muncul pada jaringan yang terinfeksi [5].

Namun masih terdapat kelemahan pada pendeteksian *botnet* dengan metode *anomaly based detection*, yaitu masih terdapat banyak kemungkinan dugaan *false rate* yang tinggi [11]. Oleh karena itu terkadang keadaan *traffic* normal bisa dianggap sebagai abnormal *traffic*. Untuk mengurangi kesalahan dalam mendeteksi *botnet* khususnya *kraken* dan *conficker*, kami membuat *framework* yang khusus agar lebih terfokus dan terarah dalam mendeteksi *botnet kraken* dan *conficker*.

Dalam *framework* ini terdapat empat proses utama untuk mendeteksi *botnet* yaitu proses identifikasi *flow DNS, flow broadcast address, flow communication port* dan proses klasifikasi. Ketiga proses awal ialah proses untuk mendeteksi keberadaan *botnet* di jaringan, sedangkan proses kalsifikasi ialah proses untuk menemukan grup *local host* atau sejumlah *local* ip yang terinfeksi *botnet.*

Paper ini terbagi menjadi lima bagian, bagian pertama merupakan latar belakang. Bagian kedua menjelaskan karakteristik *Botnet kraken* dan *conficker*. Bagian ketiga penjelasan metode *anomaly based detection*. Pada bagian empat merupakan *framework* yang kami usulkan serta didalamnya kami

memberikan penjelasan dan hasil dari penerapan *framework* kami dalam mendeteksi *botnet*. Bagian terakhir ialah kesimpulan.

2. KARAKTERISTIK BOTNET KRAKEN DAN CONFICKER

Karakter dari *botnet* pada umumnya adalah menyebarkan aplikasi yang akan menginfeksi *host* yang rentan terhadap serangan dengan mengeksploitasi aktivitas untuk memperluas jangkauan mereka [1]. *Bot* menyerang dengan memanfaatkan kerentanan perangkat lunak dan penyisipan trojan seperti teknik *social engineering* untuk mendownload kode bot yang berbahaya. Namun, di antara kelas-kelas lain dari malware, mendefinisikan karakteristik *botnet* adalah penggunaan chanel *command dan control* (*C&C*) yang dapat diupdate dan diarahkan. Arsitektur multi-tier *C&C Botnet* memberikan anonimitas untuk botmaster tersebut. *Chanel C&C* dapat beroperasi pada area dari *logical network topologies* dan menggunakan protokol komunikasi yang berbeda. *Botnet* biasanya digolongkan menurut perintah mereka dan arsitektur kontrol.

Kraken adalah *botnet* yang tidak jauh beda dengan *botnet bobax*. Malware yang digunakan oleh *botnet* ini muncul dengan berbagi kode atau menjadi varian dari malware sebelumnya. Meskipun beberapa sampel mungkin dapat menghindari *scaning* yang dilakukan oleh antivirus, mayoritas beberapa sampel dapat dideteksi dengan deteksi yang memperhatikan *behavior* atau perilakunya. Tujuan utama dari *kraken bot* digunakan sebagi *relay spam,* juga biasa dikenal sebagai spam trojan,dengan nama lain *Oderoor* [10]. Host yang terinfeksi *botnet*, akan menerima template spam dan daftar penerima spam dari C&C. Host yang terinfeksi ini dapat mengirimkan spam yang telah terinfeksi ke host yang lain atau mungkin juga mendownload file berbahaya lainnya. Host yang terinfeksi dengan *kraken* (atau o*deroor*) melakukan download trojan berdasarkan pada pesan instan atau link *peer-to-peer*. Berdasarkan referensi [10] minimal terdapat satu *IRC botnet* yang menggunakan *port* tersebut untuk mendistribusikan malware ini. Setelah *botnet* berhasil mendapatkan alamat remote hostname, malware mengirimkan datagram *UDP* ke *server kraken* pada *port* tujuan 447 untuk mengidentifikasi mesin korban. Ukuran payload untuk datagram tergantung pada varian malware yaitu antara 24 dan 74 byte. Host yang terinfeksi kemudian mendapatkan template spam dan mengirimkan spam berdasarkan *command* tertentu. Secara periodik, malware ini membuat koneksi ke *server kraken* pada *UDP* / *TCP port* 447, untuk *update* mendapatkan template baru.

Conficker adalah sejenis virus jaringan yang menggunakan metode canggih untuk *cracking passworrd administrator*. Conficker membangun *framework* bot yang digunakan untuk spam atau mencuri informasi rahasia dari user [9]. Target dari *confikcer* adalah *end user*. Salah satu karakter dari *conficker* adalah memperbanyak diri melalui USB *flash drive*. Ciri - ciri serangan *conficker* adalah koneksi internet menjadi semakin lambat atau bahkan sampai terputus. *Conficker* akan menonaktifkan akses ke *system restore* windows, windows defender dan lainnya, serta pada jaringan yang terinfeksi oleh *malware* ini akan terdapat peningkatan aktifitas pada *port UDP* yang mengakses multiple IP[7]. Metode penyebaran *conficker* adalah dengan mengeksploitasi layanan widows *server* yang masih rentan [9]. *Conficker* dirancang untuk melakukan *Remote Procedure Call* (RPC) yang bekerja pada *port* 445/*TCP* yang dapat menyebabkan windows 2000, XP, 2003 *server*, dan vista untuk menjalankan suatu segmen kode tanpa otentikasi [7]. Untuk *generate* suatu *domain name, conficker* akan berusaha untuk menghubungi suatu IP address menggunakan *port* 80 dengan mengirimkan *single call HTTP query*. Agar *conficker* tidak mudah terdeteksi di saaat berkomunikasi dengan C&C *server* maupun host lain yang terinfeksi, *malware* ini membuat koneksi *P2P* dengan menggunakan berbagai *port* secara acak [7]. Akibatnya *local host* yang terinfeksi sulit diklasifikasikan ke dalam satu grup aktivitas yang memiliki aktivitas yang sama.

3. METHODOLOGY

Ada beberapa teknik yang digunakan untuk mendeteksi *Botnet* yang berdasarkan pada analisa dan *monitoring network traffic*[1].Teknik tersebut terbagi dalam empat teknik, diantaranya signature *based detection*, *anomaly based detection*, *DNS based detection* dan *mining based detection*.

Beberapa penelitian sebelumnya yang menggunakan *anomaly-based detection* ini. Diantaranya, Binkley et al. [2][3] mengusulkan sebuah sistem berbasis anomali yang menggabungkan statistik *IRC* dan *TCP* untuk mendeteksi *botnet*. Karasaridis et al. [4] mengembangkan algoritma analisis anomali berbasis pasif yang mampu mendeteksi controller *IRC botnet* yang berjalan pada salah satu *port* acak tanpa memerlukan signature yang diketahui atau kode binari yang diterima.

Dalam paper ini, penulis menggunakan metode deteksi *botnet* dengan *anomaly-based detection*. *Anomaly-based detection* mencoba untuk mendeteksi *botnet* berdasarkan pada lalu lintas jaringan dan melihat beberapa anomali dari latensi jaringan yang tinggi, volume lalu lintas yang tinggi, lalu lintas di *port* yang tidak biasa, dan perilaku sistem yang tidak biasa yang dapat mengindikasikan adanya bot berbahaya dalam suatu jaringan [5]. Dalam *anomaly-based detection* terbagi dalam dua teknik, yaitu *host based technique* dan *network based technique*.

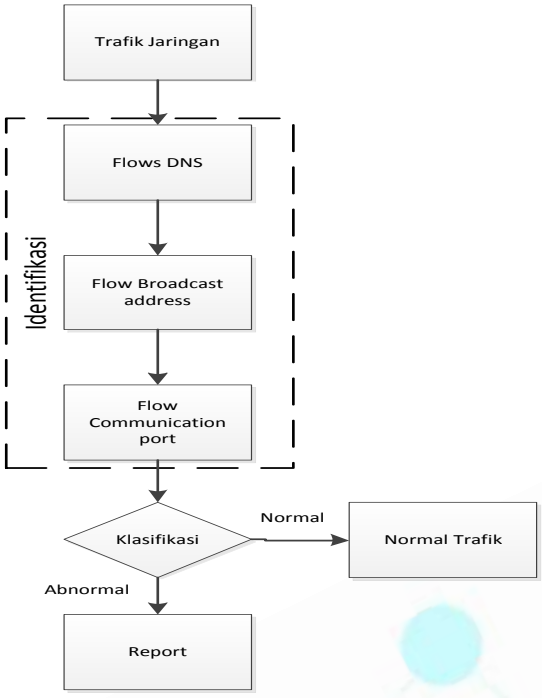
Host based technique adalah teknik yang memonitor dan menganalisa bagian internal sistem komputer sebagai pengganti dari *network traffic* pada *interface external* [6]. Sedangkan *network based technique* adalah teknik deteksi yang mencoba untuk mendeteksi *botnet* dengan monitoring network *traffic*.

Network based technique ini terbagi lagi menjadi dua yaitu *active monitoring* dan *passive monitoring*. Pada *active monitoring* didasarkan pada kemampuan untuk menginjeksi paket tes ke dalam jaringan, *server* atau aplikasi untuk mengukur reaksi dari jaringan tersebut [6], namun dengan cara ini akan memproduksi ekstra *traffic* pada jaringan. Sedangkan *passive network monitoring* menggunakan beberapa peralatan untuk mengamati *flow network traffic* yang melewati jaringan tersebut. Teknik ini tidak meningkatkan *traffic* di jaringan saat melakukan inspeksi Teknik ini memerlukan waktu yang cukup lama untuk mengamati tahapan-tahapan dari komunikasi *botnet* dan aktivitas untuk mendeteksi *botnet*. Kami menggunakan *network based technique* dengan *passive monitoring* karena mayoritas untuk mendeteksi *botnet* metode inilah yang paling mudah untuk diterapkan.

4. FRAMEWORK UNTUK MENDETEKSI BOTNET

Pada bagian ini kami menjelaskan *framework* yang diusulkan untuk mendeteksi *botnet kraken* dan *conficker* secara *visual*, serta *framework* ini didasarkan pada monitoring jaringan secara pasif. Pada dasarnya terdapat empat proses utama, yaitu identifikasi *DNS flows, local broadcast address, flow communication port* dan proses klasifikasi. Ketiga proses awal ialah proses untuk mendeteksi keberadaan *botnet kraken* dan *conficker* di jaringan, sedangkan proses kalsifikasi ialah proses untuk menemukan grup *local host* atau sejumlah *local* ip yang terinfeksi *botnet.*

Dengan menerapkan anomaly based detection kami menganalisa flow *traffic* pada jaringan dan membandingkan antara flow jaringan normal dengan abnormal, setiap proses di dalam *framework* dituangkan kedalam bentuk grafik sehingga perbedaan antara keaadan normal dan abnormal jaringan dapat dibedakan secara *visual*. Gambar 1 menjelaskan komponen *framework* yang kami usulkan.



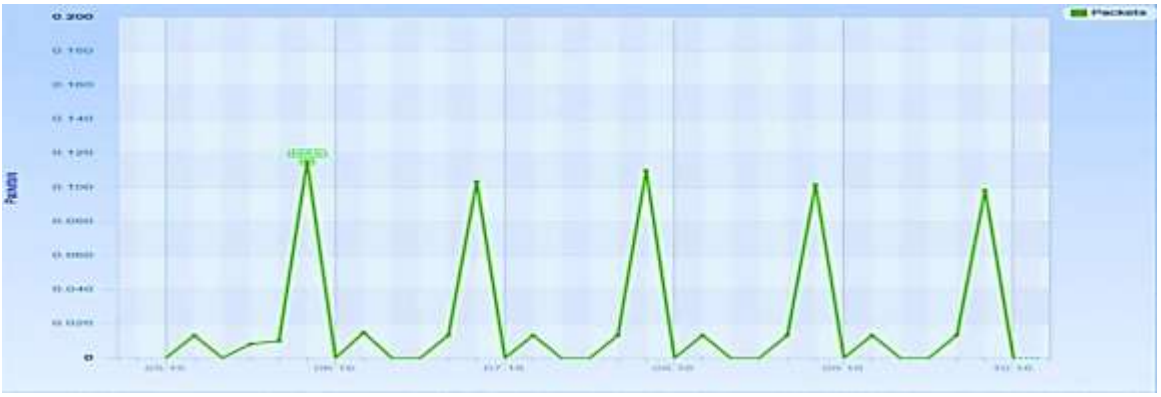
Gambar 1: Framework untuk mendeteksi Botnet Kraken dan Conficker

Kami mengkaji *framework* yang kami usulkan dengan melakukan studi ekperimen pada *network traffic* yang telah dipastikan terinfeksi oleh *botnet kraken* dan *conficker* serta melakukan perbandingan dengan *traffic* yang normal. Adapun *tools* yang kami gunakan didalam eksperimen yang kami lakukan adalah dengan menggunakan *cace pilot* untuk meneliti *flow traffic* secara *visual* dan *wireshark* guna analisis packet frame lebih dalam.

4.1 Identifikasi *DNS* Flows

Proses awal dalam mendeteksi keberadaan *botnet,* ialah dengan menganalisa paket frame *DNS*. Dalam proses ini kami menganalisa *DNS* (*port* 53) sebagai langkah awal dalam mendeteksi *botnet*. Hal ini didasari oleh penelitian sebelumnya yang telah menerangkan bahwa *botnet* secara periodik dan terus menerus mencari alamat *Remote Host* dengan mengakses *DNS server*[8], sehingga *botnet* dapat terdeteksi dengan meneliti isi frame pada jaringan yang diduga terinfeksi, khususnya oleh *botnet kraken*. Pada tahap ini kita belum dapat mengetahui jenis *botnet* yang telah menginfeksi jaringan dan masih terdapat kemungkinan false positif untuk *botnet conficker*.

Seluruh *flow packet DNS* di dalam jaringan ditampilkan dalam bentuk grafik. Kami menggunakan sample *flow DNS* selama 5 jam dengan perbandingan jumlah *packet* yang dikirim per detik, sehingga didapatkan *flow DNS* dalam *packet*/detik. Pada jaringan yang normal *flow DNS traffic* akan seperti pada gambar 2, sedangkan pada jaringan yang telah terinfeksi oleh *botnet* baik *kraken* maupun *conficker* akan muncul seperti pada gambar 3.



Gambar 2: Normal *traffic DNS*

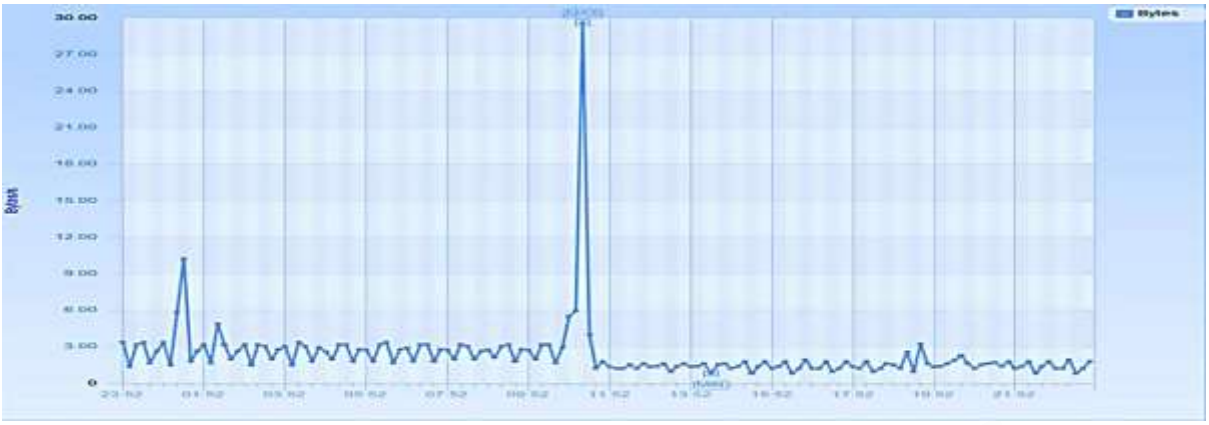
Gambar 3: Abnormal *traffic DNS*

Dalam keadaan normal *flow traffic DNS* pada gambar 2 menunjukan bahwa akses ke *DNS server* tidak lah stabil atau fluktuatif dan terkadang tidak terdapat aktifitas, sehingga jumlah *packet* yang dikirim dan diterima oleh *host* bisa lebih besar dan lebih kecil dari kondisi abnormal. Hal ini terjadi karena dalam keadaan normal user tidak selalu me-*request* halaman web. Sedangkan dalam keadaan abnormal, *flow traffic DNS server* cenderung lebih stabil, hal ini terjadi karena *Botnet* secara periodik dan *realtime* terus menerus melakukan *request DNS* untuk menemukan alamat *remote* hostnya.

4.2 Identifikasi Flow Broadcast Address

Tahapan ini dilakukan untuk manganalisa *broadcast address* pada *local* area jaringan komputer. Pada jaringan yang telah terinfeksi oleh *botnet,* setiap host yang terinfeksi pada *local* jaringan akan mengirimkan suatu *packet* frame guna menginfeksi host yang belum terinfeksi melalui *broadcast address*, khususnya melalui *port* 137.

Traffic broadcast flow tersebut divisualisasikan dengan menetapkan sample waktu selama 12 jam, sehingga didapatkan *flow broadcast size data* per detik, sehingga didapatkan satuan bytes/detik . Untuk lebih jelas dapat dilihat normal *traffic* pada gambar 4 dan abnormal *traffic* pada gambar 5.



Gambar 4 : Normal *Broadcast traffic*

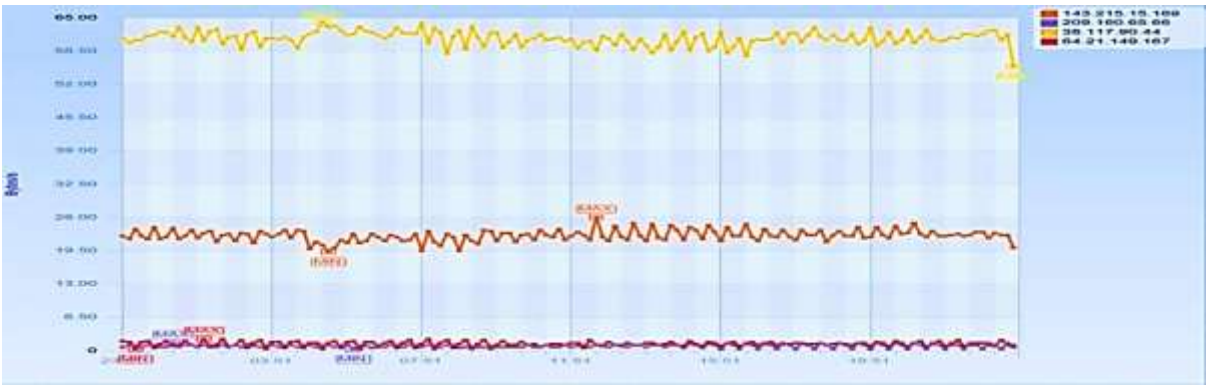
Gambar 5: Abnormal *Broadcast traffic*

Flow broadcast *traffic* pada abnormal jaringan akan jauh lebih tinggi dari keadaan normal, hal ini bisa diperhatikan jumlah jumlah *flow data* yang dikirim per detik. Hal karena host yang terinfeksi terus menerus melakukan *broadcast* dengan ukuran *packet* yang cukup besardan relative sama. Akibat dari *broadcast flow* yang tidak berhenti akan menurunkan performa jaringan sehingga akan menjadi lebih lambat.

4.3 Identifikasi Flow Protokol Komunikasi

Tahap ini dilakukan untuk membedakan normal dan abnormal *traffic* pada suatu jaringan apabila *botnet* melakukan komunikasi dengan *C&C* servernya. Dari protokol komunikasi ini kita dapat membedakan *botnet* yang menyerang jaringan, baik *conficker* maupu *kraken*. Setiap *botnet* akan berkomunikasi dengan *C&C server* guna menerima perintah dari *C&C server* atau sekedar mengupdate informasi.

Dalam hal ini *kraken* menggunakan Port *UDP* 447 untuk mengirimkan informasi dari host yang menjadi korban, adapun *payload data* yang dikirim bergantung pada varian *botnet* serta berukuran sekitar 24 dan 74 bytes[10]. Gambar 6 menunjukan *outbound* abnormal *traffic* pada *port* 447 yang digunakan oleh sejumlah *Remote Host C&C* untuk berkomunikasi dengan local host yang menjadi korban. Gambar 7 menunjukan besar *payload data* yang dikirim oleh *local host* yang terinfeksi untuk berkomunikasi dengan *C&C botnet kraken*. Sedangkan untuk jaringan normal sangat jarang menggunakan *port* 447 bahkan tidak pernah sama sekali, dan hasil eksperimen kami tidak terdapat *flow traffic* pada normal *port* 447.

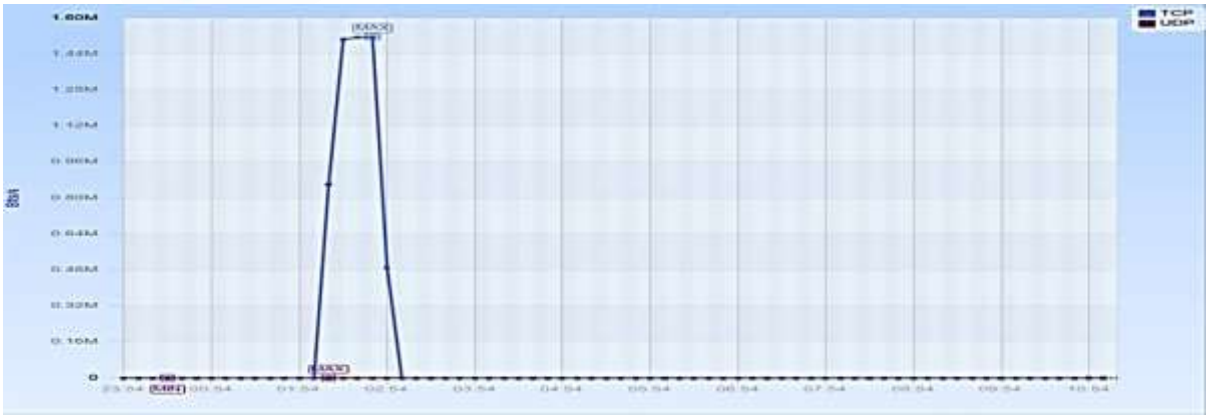


Gambar 6: Abnormal *traffic Kraken* pada *port* 447

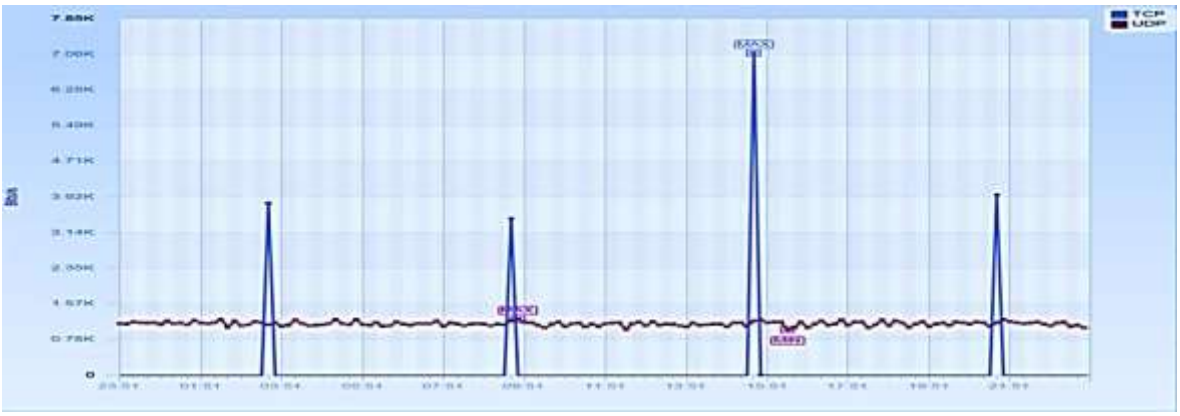
No.	Time	Source	Destination	Protocol	Data Size	Source Port	Destination Port	Info
1	0.000000	192.168.3.105	38.117.90.44	UDP	24	tpcsrvr	ddm-dfm	Source port: tpcsrvr Destination port: ddm-dfm
2	1.437608	192.168.3.105	143.215.15.189	UDP	74	idware-router	ddm-dfm	Source port: idware-router Destination port: ddm-dfm
3	1.705758	192.168.3.102	38.117.90.44	UDP	74	igcp	ddm-dfm	Source port: igcp Destination port: ddm-dfm
4	4.479496	192.168.3.104	38.117.90.44	UDP	74	netmon	ddm-dfm	Source port: netmon Destination port: ddm-dfm
5	6.780206	192.168.3.102	38.117.90.44	UDP	24	veritas-udp1	ddm-dfm	Source port: veritas-udp1 Destination port: ddm-dfm
6	7.322381	192.168.3.113	38.117.90.44	UDP	74	nimreg	ddm-dfm	Source port: nimreg Destination port: ddm-dfm
7	7.411906	192.168.3.100	143.215.15.189	UDP	74	s3db	ddm-dfm	Source port: s3db Destination port: ddm-dfm
8	9.887762	192.168.3.117	38.117.90.44	UDP	74	taskmaster2000	ddm-dfm	Source port: taskmaster2000 Destination port: ddm-dfm
9	11.890093	192.168.3.101	38.117.90.44	UDP	74	cr-websystems	ddm-dfm	Source port: cr-websystems Destination port: ddm-dfm
10	14.157723	192.168.3.105	38.117.90.44	UDP	74	radsec	ddm-dfm	Source port: radsec Destination port: ddm-dfm
11	14.737843	192.168.3.104	38.117.90.44	UDP	74	wag-service	ddm-dfm	Source port: wag-service Destination port: ddm-dfm
12	15.569589	192.168.3.104	38.117.90.44	UDP	24	system-monitor	ddm-dfm	Source port: system-monitor Destination port: ddm-dfm
13	15.620632	192.168.3.101	38.117.90.44	UDP	24	precise-sft	ddm-dfm	Source port: precise-sft Destination port: ddm-dfm
14	15.705855	192.168.3.102	143.215.15.189	UDP	74	btprjctrl	ddm-dfm	Source port: btprjctrl Destination port: ddm-dfm
15	16.987503	192.168.3.106	38.117.90.44	UDP	74	oracle-em1	ddm-dfm	Source port: oracle-em1 Destination port: ddm-dfm
16	17.650746	192.168.3.102	38.117.90.44	UDP	24	dvr-esm	ddm-dfm	Source port: dvr-esm Destination port: ddm-dfm
17	17.669744	192.168.3.100	143.215.15.189	UDP	74	lnvpoller	ddm-dfm	Source port: lnvpoller Destination port: ddm-dfm
18	23.170269	192.168.3.115	38.117.90.44	UDP	74	veracity	ddm-dfm	Source port: veracity Destination port: ddm-dfm
19	25.013816	192.168.3.101	38.117.90.44	UDP	74	attachmate-g32	ddm-dfm	Source port: attachmate-g32 Destination port: ddm-dfm
20	25.339137	192.168.3.104	143.215.15.189	UDP	74	lionhead	ddm-dfm	Source port: lionhead Destination port: ddm-dfm
21	25.682395	192.168.3.100	38.117.90.44	UDP	24	lnvalarm	ddm-dfm	Source port: lnvalarm Destination port: ddm-dfm
22	26.053157	192.168.3.101	38.117.90.44	UDP	24	cadencecontrol	ddm-dfm	Source port: cadencecontrol Destination port: ddm-dfm
23	26.812135	192.168.3.105	38.117.90.44	UDP	74	gnunet	ddm-dfm	Source port: gnunet Destination port: ddm-dfm

Gambar 7: Payload ukuran data *Kraken*

Sedangkan *conficker* menggunakan *port* 80 untuk meng-update dirinya dan menggunakan protokol secara *P2P* untuk berkomunikasi dengan *C&C*. Sehingga saat *conficker* berkomunikasi secara *P2P*, akan terdapat banyak *port* yang terbuka secara acak oleh *botnet* ini. Menurut riset sebelumnya pada jaringan yang terinfeksi oleh *conficker*, flow packet pada *UDP* akan mengalami peningkatan secara terus menerus [9] dan akan terdapat pola flow tertentu khususnya pada *port* 80/*TCP*[9]. Karena dalam setiap beberapa periode dengan frekuensi yang sama *conficker* akan melakukan *pulling* dari *remote host* melalui *port* 80/TCP. Untuk lebih jelas dapat dilihat pada gambar 8 dan gambar 9 yang membedakan normal dan abnormal *traffic flow* pada *TCP/ UDP* yang terserang *Conficker*.



Gambar 8: Normal *traffic TCP* / *UDP*



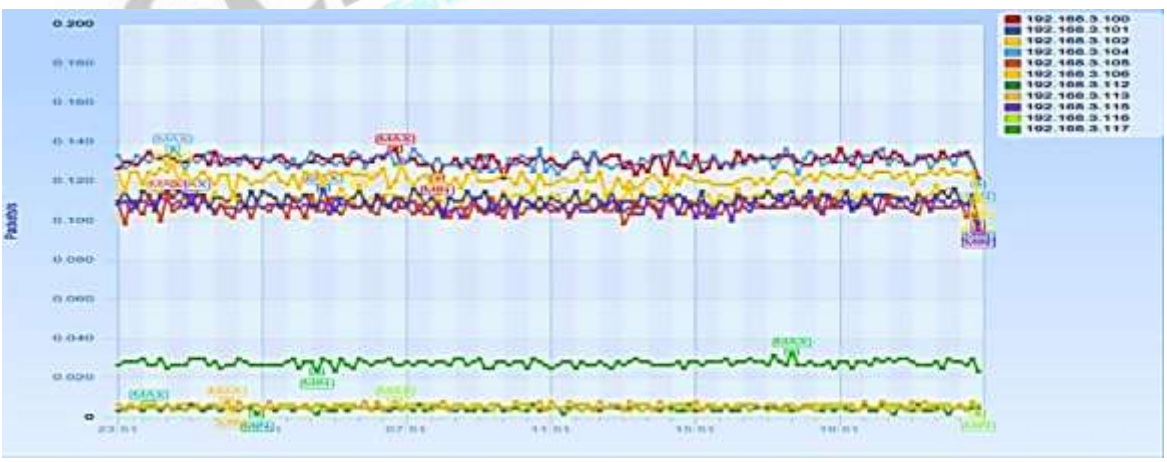
Gambar 9: Abnormal *traffic TCP* / *UDP* Conficker

4.4 Klasifikasi

Proses klasifikasi bertujuan untuk mencari grup host yang beraktivistas sama, dengan tujuan untuk menentukan host mana saja yang diperkirakan telah terinfeksi oleh *botnet.* Untuk menemukan host yang terinfeksi terlebih dahulu harus menemukan *protocol* komunikasi yang biasa diakses oleh *botnet* untuk berkomunikasi dengan *C&C.*

Dalam hal ini, untuk mengklasifikasikan *host* yang terinfeksi *kraken*, kami menggunakan *port* 447 sebagai *port* utama bagi konfiker untuk berkomunikasi. Kemudian lakukan filter untuk seluruh paket *frame* yang melewati jaringan berdasarkan pada ukuran data dari *host* yang mengakses *port* 447, dengan besar data 24 bytes dan 74 bytes. Sehingga akan didapat jumlah *host* yang terinfeksi dengan jumlah paket yang dikirim per detik. Untuk lebih jelas, bisa dilihat pada gambar 10.

Sedangkan untuk *botnet conficker* karena menggunakan protokol *P2P* untuk berkomunikasi, maka ada sedikit kendala dalam mengklasifikasikan host yang terinfeksi dan host yang normal. Hal ini disebabkan protokol *P2P* digunakan secara random oleh *conficker*. Sehingga, sulit untuk membedakan *local host* yang terinfeksi *conficker.* Pada proses klasifikasi ini kami belum dapat membedakan local host yang terinfeksi oleh *kraken*.



Gambar 10: *Local host* yang mengakses *port* 447

5. KESIMPULAN

Botnet adalah suatu malware yang dapat dikontrol oleh botmaster melalui *C&C server*. Untuk mendeteksi *botnet* paper ini menggunakan *anomaly based detection* dimana *botnet* dideteksi dengan meneliti *flow traffic, port traffic, volume traffic*, *botnet behaviour* dan membandingkan antar normal dan abnormal *traffic*.

Di dalam *framework* yang kami usulkan terdapat empat proses utama, yaitu identifikasi *DNS* flows, *local broadcast address*, *flow communication port* dan proses klasifikasi.Pada tiga proses awal ialah untuk mendeteksi keberadaan *botnet kreken* dan *conficker* di jaringan, sedangkan proses kalsifikasi ialah proses untuk menemukan grup *local host* atau sejumlah *local* ip yang terinfeksi *botnet.*

Framework kami mampu untuk mendeteksi *botnet kraken* dan *conficker* yang menginfeksi jaringan. Selain itu, *framework* ini juga dapat mengklasifikasikan grup *local host* yang terinfeksi oleh *botnet kraken* menggunakan *port 447* sebagai *filter* utama. Namun *framework* ini memiliki keterbatasan dalam mengklasifikasikan grup *local host* yang terinfeksi oleh *conficker*, karena sulitnya membedakan *port* komunikasi *P2P*, hal ini disebabkan terdapat ribuan *port* yang dapat digunakan secara acak oleh *conficker* untuk melakukan komunikasi secara *P2P*.

Untuk riset berikutnya kami akan membuat *framework* yang dapat mengklasifikasikan *local host* yang terinfeksi oleh berbagai varian *botnet* yang menggunakan koneksi *P2P* untuk berkomunikasi dengan *C&C* maupun dengan sesama *botnet*.

DAFTAR PUSTAKA

[1] Feily Maryam, Shahrestani A and Ramadass S, "A Survey of Botnet and Botnet Detection", IEEE, 2009, pp 268-273.

[2] J. R. Binkley, "Anomaly-based Botnet Server Detection", June 2006, pp 1-4.

[3] J. R. Binkley and S. Singh. "An algorithm for anomaly-based Botnet detection", In Proceedings of USENIX Steps to Reducing Unwanted Traffic on thfe Internet Workshop (SRUTI), July 2006, pp. 43–48.

[4] Karasaridis A, Rexroad B, Hoeflin D. "Wide-Scale Botnet detection and haracterization", In: Proc. of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots 2007). 2007

[5] B. Saha and A, Gairola, "Botnet: An overview," CERT-In White PaperCIWP-2005-05, 2005.

[6] Zeidanloo HR, Zadeh MJ, Safari dan Zamani M, "A Taxonomy of Botnet Detection Techniques", IEEE, 2010. Pp 158-162.

[7] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran, "An Analysis of Conficker's Logic and Rendezvous Points", http://mtc.sri.com/Conficker.

[8] Hyunsang choi, Hanwoo Lee, Hyogon Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic", IEEE 2007

[9] Mark Yason, IBM x-force" IBM internet security system protection alert" 2009, www.iss.net/threats/Conficker.html

[10] Jose Nazario" IBM internet security system protection alert" 2008, www.iss.net/threats/Kraken.html

[11] Chao Li, Wei jiang, Xin Zou, "Botenet: Survey and Case Study" IEEE 2009