RANCANG BANGUN INTRUSION DETECTION SYSTEM ZEEK UNTUK MONITORING JARINGAN DENGAN VISUALISASI ELK STACK DAN NOTIFIKASI TELEGRAM

LAPORAN SKRIPSI

FAIZ WATSIQUL UMAM 1807422015

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

POLITEKNIK NEGERI JAKARTA

2022

RANCANG BANGUN INTRUSION DETECTION SYSTEM ZEEK UNTUK MONITORING JARINGAN DENGAN VISUALISASI ELK STACK DAN NOTIFIKASI TELEGRAM

LAPORAN SKRIPSI

Dibuat untuk Melengkapi Syarat-Syarat yang Diperlukan untuk Memperoleh Diploma Empat Politeknik

FAIZ WATSIQUL UMAM

1807422015

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

POLITEKNIK NEGERI JAKARTA

2022

HALAMAN PERNYATAAN ORISINALITAS SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini :

Nama : Faiz Watsiqul Umam

NIM : 1807422015

Jurusan/Program/Studi : T.Informatika dan Komputer / T. Multimedia dan Jaringan

Judul Skripsi :Rancang Bangun IDS Zeek Untuk Monitoring Jaringan Dengan Visualiasi ELK Stack dan Notifikasi Telegram

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok, 14 Juli 2022

Faiz Watsiqul Umam

NIM. 1807422015

LEMBAR PENGESAHAN

Skripsi diajukan oleh

Nama	: Faiz Watsiqul Umam
NIM	: 1807422015
Program Studi	: Teknik Multimedia dan Jaringan
Judul Skripsi	: Rancang Bangun Intrusion Detection System Zeek untuk Monitoring Jaringan dengan Visualisasi ELK Stack dan Notifikasi Telegram

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari Selasa, tanggal 9, bulan Agustus, Tahun 2022, dan dinyatakan **LULUS**

Disahkan oleh:

Pembimbing I	: Ayu Rosyida Zain S.ST, M.T.	( )
Penguji I	: Nur Fauzi Soelaiman, S.T., M.Kom.	( )
Penguji II	: Maria Agustin S.Kom., M.Kom.	( )
Penguji III	: Fachroni Arbi Murad, S.Kom., M.Kom	( )

Mengetahui:
Jurusan Teknik Informatika dan Komputer
Ketua

Mauldy Laya, S.Kom, M.Kom

NIP. 197802112009121003

KATA PENGANTAR

Puji Syukur saya panjatkan kepada Tuhan Yang Maha Esa, atas berkat dan rahmat nya, penulis dapat menyelesaikan laporan skripsi ini. Penulisan laporan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Terapan Politeknik. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan laporan skripsi, sangatlah sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada :

a. Ibu Ayu Rosyida Zain S.T., M.Kom. selaku Dosen Pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis dalam penyusunan skripsi ini;
b. Ayah Dr. Ir. Yayan Apriyana M.Sc, Ibu Ai Salamah S.H, Kakak Fathul Alim Al-Faruqi, Fachry Syifaurrahman, Fikri Dhiyaul Ilmi yang telah memberikan dukungan penulis dari kecil hingga sekarang
c. Teman Kelas satu Perjuangan dari awal hingga akhir masa perkuliahan yang tidak pernah putus memberi semangat kepada penulis
d. Teman-Teman perkopian Muhammad Rifqy Al – A'dzomy, Fadhli Naufal, Syahrizal Daffa Aditya, Andika Dwi Febrian, Hasbi Akbar, Reza Pratama yang sudah memberikan canda tawa dan semangat untuk penulis.
e. Teman – Teman Seperidolan terutama Amel, Zaki, Zesky, Daffa, Rafi, Rayhan, Rafli, Riski yang memberikan kelucuan dan canda tawa yang membahagiakan bagi penulis.

Dalam penulisan skripsi ini masih banyak kekurangan dan kesalahan, karena itu segala kritik dan saran yang membangun akan menyempurnakan penulisan skripsi ini serta bermanfaat bagi penulis dan para pembaca.

Bogor, 14 Juli 2022

Penulis

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya bertanda tangan dibawah ini :

Nama : Faiz Watsiqul Umam

NIM : 1807422015

Jurusan/Program Studi : T. Informatika dan Komputer / Teknik Multimedia Jaringan

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Ekslusif atas karya ilmiah saya yang berjudul :

RANCANG BANGUN INSTRUTION DETECTION SYSTEM ZEEK UNTUK MONITORING JARINGAN DENGAN VISUALISASI ELK STACK DAN NOTIFIKASI TELEGRAM.

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini Politeknik Negeri Jakarta Berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data (*database)*, merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta..

Demikian pernyataan ini saya buat dengan sebenarnya.

Depok, 14 Juli 2022

Faiz Watsiqul Umam

NIM. 1807422015

Rancang Bangun Instrution Detection System Zeek untuk Monitoring Jaringan dengan Visualiasi ELK Stack dan Notifikasi Telegram

ABSTRAK

Keamanan jaringan komputer menjadi hal yang perlu diperhatikan seiring berkembangnya teknologi yang pesat. Menjadi tanggung jawab bagi seorang administrator jaringan untuk memonitor keamanan sistem sewaktu-waktu. Mengingat banyaknya ancaman yang bisa masuk kedalam sistem kapan saja, dibutuhkan aplikasi yang dapat mendeteksi adanya ancaman tersebut secara realtime. Permasalahan tersebut menimbulkan gagasan bagi penulis untuk memanfaatkan salah satu aplikasi, yaitu Zeek yang di dalamnya terdapat metode IDS (Intruction Detection System) yang akan berfungsi sebagai pendeteksi attacker. Zeek akan menampilkan alert ketika ada paket yang mencurigakan. Alert yang dihasilkan akan disimpan didalam log file. Tujuan dari penelitian ini adalah untuk menerapkan sistem deteksi Zeek, menganalisis log alert yang masuk kedalam sistem jaringan dengan menggunakan visualisasi ELK (Elasticsearch, Logstash, Kibana), serta mempermudah admin jaringan dalam membaca dan menganalisa log tersebut. Alert yang tampil pada ELK nantinya akan dikirimkan ke handphone administrator jaringan melalui pesan telegram. Administrator jaringan akan memperoleh informasi terkait dengan serangan yang terjadi pada jaringan secara realtime. Metode yang digunakan dalam penelitian ini adalah eksperimental. Metode Eksperimental meliputi analisa kebutuhan, pembuatan desain topologi jaringan, installasi sistem (Zeek dan ELK), software pendukung lainnya, konfigurasi sistem dan pengujian serangan pada sistem. Berdasarkan hasil pengujian dari implementasi penelitian ini, sistem IDS dapat mendeteksi serangan yang terjadi ke dalam jaringan komputer. Telegram bot yang dibuat berhasil mengirimkan notifikasi secara realtime dengan waktu rata-rata kepada aplikasi telegram serta ELK Stack dapat mengolah log Zeek dengan memberikan statistik yang mudah dipahami oleh admin jaringan.

Kata Kunci: *Alert, Attacker, ELK, IDS, Zeek, Telegram.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengunaan jaringan internet berkembang sangat pesat dan sudah menjadi kebutuhan pokok bagi masyarakat, internet sangat memungkinkan kita untuk berkomunikasi dengan pihak lain. Seiring berkembangnya internet, banyak instansi kantor dan perusahaan yang mengunakan internet untuk memperlancar arus komunikasi informasi didalamnya (Sutarti dkk 2018). Data-data perusahaan merupakan informasi yang bersifat rahasia dan dijaga keamanannya. Disisi lain, mudahnya akses tersebut bisa menyebabkan dengan memanfaatkannya informasi atau data penting oleh pihak yang tidak bertanggung jawab.

Fitur *Intrusion Detection System (IDS)* dapat meningkatkan keamanan jaringan server sehingga dapat mempersulit serangan yang dilakukan terhadap Server. Intrusion Detection System adalah suatu sistem yang dapat mendeteksi adanya serangan sekaligus bisa menampilkan dan mengirim kepada admin jaringan ketika terjadi penyusupan/penyerangan (Wahyu F, 2016). Admin jaringan bertanggung jawab terhadap semua kondisi yang terjadi pada jaringan yang dikelolanya, terutama untuk sistem keamanan jaringan tersebut. Meskipun umumnya sebuah jaringan sudah dilengkapi dengan *firewall*, seorang admin harus *standby* dalam memonitoring *log service* secara berkala. Hal tersebut administrator harus rajin memeriksa aktivitas yang berjalan pada system operasi server yang dicatat pada *log service*. namun pada kenyataannya administrator tidak dapat standby selama 24 Jam (Akhyar dkk 2020)

Zeek adalah salah satu dari Program *Intrusion Detection System (IDS)* yang bersifat open source yang dapat memonitoring jaringan pada server, mendeteksi serangan dari attacker yang tidak bertanggung jawab, dan mencegah adanya ancaman yang masuk ke dalam jaringan secara realtime (Wahyu F, 2016). Zeek dapat melakukan konfigurasi dengan menyediakan fungsi scripting, dengan banyak dukungan library. Secara default, Zeek menghasilkan output Log yang tampil pada terminal linux berupa aktifitas pada server . Tentu saja untuk seorang admin butuh penyesuaian untuk memahami output tersebut. Log yang dihasilkan Zeek akan

angsung diterapkan dengan penerapan Elasticsearch Logstash dan Kibana Stack (ELK Stack). ELK Stack merupakan sekumpulan aplikasi open source yang digunakan dalam pengolahan data dengan volume yang besar (Elang dkk 2020). Elasticsearch sebagai tempat menyimpan, mencari, dan memproses data log, nantinya bisa menghasilkan output dan bisa ditampilkan secara visual di browser dengan Kibana (Walidatush dkk 2020). Selain dari IDS Zeek, model monitoring akan memanfaatkan Aplikasi telegram sebagai notifikasi adanya ancaman serangan. Aplikasi *instant messanging* Telegram saat ini popular digunakan oleh berbagai kalangan, karena mempuntai fitur-fitur sangat canggih dalam hal keamanannya. Salah satu fitur dari telegram yaitu *Secret Chat.* Secret Chat dienkripsi dengan prosedur *end-to-end* sehingga isi pesan tersebut tidak bisa diakses oleh siapapun di perangkat lain hanya pengirim dan penerima sajalah yang dapat mengaksesnya (Febriyanti, 2019).

Dari latar belakang diatas akhirnya penulis mengambil judul "Rancang Bangun Intrusion Detection System Zeek Untuk Monitoring Jaringan Dengan Visualisasi ELK Stack dan Notifikasi Dengan Telegram". Pada penelitian sebelumnya, Zeek hanya menulis log ketika serangan terjadi tanpa mengirimkan alert kepada administrator saat tidak ditempat dan ELK Stack hanya memvisualisasikan program Program IDS lain seperti IDS Suricata. Pada penelitian ini, penulis menawarkan penelitian dari IDS Zeek yang diintegrasikan kepada ELK Stack dengan memberikan alert kepada telegram.

1.2 Perumusan Masalah

Dilihat dari latar belakang tersebut adapun perumusan masalahnya sebagai berikut

a. Bagaimana membangun sebuah sistem server yang dapat mendeteksi penyerang yang tidak bertanggung jawab?
b. Bagaimana penerapan IDS dengan menggunakan Zeek?
c. Bagaimana penerapan ELK Stack untuk memvisualisasikan hasil laporan Zeek?
d. Bagaimana Telegram menotifikasikan kepada pengguna ketika terjadi serangan?

1.3 Batasan Masalah

Adapun batasan masalah yang disebutkan bertujuan agar pembahasan dapat lebih terarah. Pembatasan masalah tersebut antara lain :

a. Penelitian ini berfokus bagaimana cara implementasi *Zeek* dan memvisualisasikan hasil *Log Zeek* pada ELK Stack

b. User menggunakan Virtual Machine (Server IDS Zeek, Internet, Attacker)

c. Pengujian Serangan menggunakan Port Scanning dan Brute Force Attack

d. Notifikasi Serangan menggunakan telegram

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Adapun Tujuan dari penelitian ini adalah :

a. Mengimplementasikan Intrusion Detected System Menggunakan Zeek dengan ELK Stack

b. Mendeteksi Serangan Port Scanning dan Brute Force dari Attacker dengan bantuan Notifikasi Telegram

1.4.2 Manfaat

Adapun Manfaat dari penelitian ini adalah :

a. Memvisualisasikan hasil dari Log Zeek yang dapat mudah dipahami oleh administrator

b. Mempermudah Administrator Jaringan untuk memperoleh informasi terkait dengan serangan Port Scanning dan Brute Force yang terjadi pada jaringan secara realtime.

1.5 Sistimatika Penulisan

Penelitian ini dilakukan dengan Sistimatika Penulisan sebagai berikut:

a. BAB I PENDAHULUAN

Pada bab ini membahas tentang latar belakang, perumusan masalah, perumusan masalah, tujuan dan manfaat pada penelitian ini.

b. BAB II TINJAUAN PUSTAKA

Pada bab ini membahas tentang materi-materi yang mendukung dan membantu penelitian ini.

c. BAB III PERANCANGAN DAN REALISASI

Pada bab ini membahas metode pengerjaan, rancangan penelitian, tahapan penelitian dan objek penelitian.

d. BAB IV PEMBAHASAN

Pada bab ini berisi mengenai pengujian dan hasil dari analisis pengujian IDS seperti deskripsi pengujian, prosedur pengujian, data hasil pengujian dan analisis data/evaluasi

e. BAB V KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dan saran dari keseluruhan penelitian.

BAB V
PENUTUP

5.1. Kesimpulan

Setelah merancang dan membangun Intrusion Detection System Zeek dengan Visualisasi ELK Stack dan Notifikasi Telegram, dapat ditarik kesimpulan :

a. IDS Zeek mampu mendeteksi terjadinya serangan *port scanning* dengan serangan *brute force attack.*
b. Rata-rata waktu notifikasi pada Telegram ketika terjadi sepuluh kali Serangan Port Scanning menggunakan *nmap* yaitu 3.95 detik dan rata-rata notifikasi Telegram yang dihasil Serangan Brute Force menggunakan *hydra* yaitu 3.4 detik
c. Jumlah *hits* yang didapat pada ELK Stack Ketika terjadi serangan sepuluh kali Port Scanning sebesar 710.084 Hits

5.2 Saran

Untuk pengembangan selanjutnya dari sistem ini dapat menambahkan fitur-fitur atau rules yang dapat menambah kinerja IDS Zeek lebih baik :

a. Untuk pengujian selanjutnya dapat ditambahkan dengan serangan lainnya seperti SQL Injection, Ping Attack, phising dan lain sebagainya
b. Notifikasi pada Zeek tidak hanya melalui Telegram saja tetapi juga bisa ditambahkan melalui Email, atau menambahkan API Token Media Sosial lainnya

DAFTAR PUSTAKA

Albert Yakobus Chandra. (2019) Analisis Performansi Antara Apache & Nginx Web Server dalam Menangani Client Request. Jurnal Sistem dan Informatika (JSI)

Bayu, WS. Firdiansyah, S. Azhari, M. (2016) Implementasi Algoritma Brute Force Sebagai Mesin Pencari (Search Engine) Berbasis Web Pada Database

Corelight, 2022. How Zeek Works [Online] Avaiable at: https://corelight.com/about-zeek/how-zeek-works [Accessed 18 Febuary 2022].

Elang Putra Sartika, A.B. (2020) Implementasi Elasticsearch Logstash Kibana Stack pada Sistem Portal Pengembangan dan Pembinaan Sumber Daya Manusia. Jurnal UMJ.

Elastic, "What is Elasticsearch ?," 2019. [Online]. Available: https://www.elastic.co/what-is/elasticsearch . [Accessed 20 April 2022].

Feriana Istining Tiyas, Feriana (2011) APLIKASI WEB UNTUK METODE FUZZY NEURAL NETWORK PADA INTRUSION DETECTION SYSTEM BERBASIS SNORT.

Graham Barbour, A. N. (2021). Evasion of Port Scan Detection in Zeek and Snort and its Mitigation. Journal ECCWS 20th European Conference. 25-34

Hashem Alyami (2022). Effectiveness Evaluation of IDSs Using Integrated Fuzzy MCDM Model. MDPI Eletronics. Journal MDPI

Hindung, Pramu Andono (2021). Penerapan Keamanan Jaringan Menggunakan Zeek Sebagai Intrusion Detection System (IDS). Diploma Thesis, Institut Teknologi Telkom Purwokerto.

Sofana, Iwan (2018). Network Security dan Cyber Security. Penerbit Informatika

Maria Ulfa. (2013). Implementasi Intrusion Detection System (IDS) Di Jaringan Universitas Bina Darma. Jurnal Imiah MATRIK Vol.15 No.2, Agustus 2013., 105-118.

Nanda Fernando, H. E. (2020). Monitoring Jaringan dan Notifikasi dengan Telegram pada Dinas Komunikasi dan Informatika Kota Padang . Jurnal Ilmiah Teknologi Sistem Informasi, Volume 1 No 4, Desember 2020 , 121-126.

Panjaitan, F. (2019). PEMANFAATAN NOTIFIKASI TELEGRAM UNTUK MONITORING JARINGAN. Jurnal SIMETRIS, Vol. 10 No. 2 November 2019, 725-732.

Ruli Dimas Prakoso. (2018). IMPLEMENTASI DAN PERBANDINGAN PERFORMA PROXMOX DALAM VIRTUALISASI DENGAN TIGA VIRTUAL SERVER. Jurnal UNESA, Vol. 8 No.1, 79-85

Santoso, R.N. (2017). Perencanaan dan Pengembangan Aplikasi Absensi Mahasiswa Menggunakan Smart Card Guna Pengembangan Kampus Cerdas (Studi Kasus Politeknik Negeri Tanah Laut). Jurnal Integrasi, Vol.9 No.1, April 2017, 84-91.

Sugiyono, (2016). SISTEM KEAMANAN JARINGAN KOMPUTER MENGGUNAKAN METODE WATCHGUARD FIREBOX PADA PT GUNA KARYA INDONESIA. Jurnal CK1 On SPOT, Vol.9, No. 1 1-8

Sutarti, A. P. (2018). IMPLEMENTASI IDS (INTRUSION DETECTION SYSTEM) PADA SISTEM KEAMANAN JARINGAN SMAN 1 CIKEUSAL. Jurnal PROSISKO Vol. 5 No. 1 Maret 2018, 1-8.

Wahyu Dwi Romadhon (2021). Implementasi Suricata IDPS Untuk Monitoring Jaringan Dengan Visualisasi ELK (Elasticsearch, Logstash, Kibana) dan Notifikasi Melalui Bot Telegram. Proyek Akhir. Universitas Muhammadiyah Surakarta

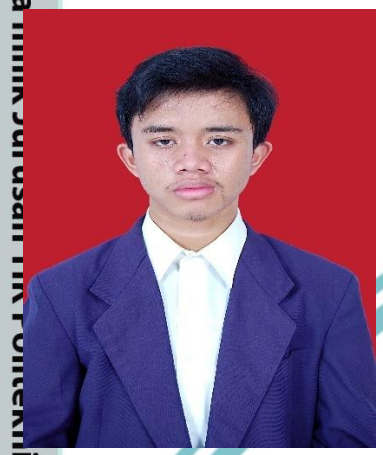
Wahyu Febriyan Ramadhan (2016). Implementasi IPS dan IDS Menggunakan Aplikasi BRO-IDS (Intrusion Detection System) Yang Terintegrasi Dengan SMS Gateway. Proyek Akhir. Universitas Telkom

Walidatush Sholihah, S. A. (2020). Log Event Management Server Menggunakan Elastic Search Logstash Kibana (ELK Stack). JTIM : Jurnal Teknologi Informasi dan Multimedia., 12-20.

Zaki Akhyar, H. A. (2018). Rancang Bangun Sistem Pengiriman Alert Instrusion Detection System Suricata Telegram. Seminar Nasional Politeknik Negeri Lhokseumawe., A-175-A-181.

Zeek, 2022. About Zeek [Online] Avaiable at: https://docs.zeek.org/en/master/about.html [Accessed 18 Febuary 2022].

LAMPIRAN
DAFTAR RIWAYAT HIDUP PENULIS



Faiz Watsiqul Umam

Lulus dari SDIT Aliya tahun 2010, SMP Negeri 7 Bogor 2013, dan SMA Bina Bangsa Sejahtera pada tahun 2016

Lampiran 2 Script dari detect-port.zeek

```
@load base/frameworks/notice
@load base/utils/active-http
@load base/frameworks/sumstats
@load base/utils/time

module Notice;

export {
    redef enum Action += {
        ACTION_TELEGRAM,
    };

    const telegram_endpoint = "https://api.telegram.org";
    option telegram_token = "5496216098:AAEf-nYbUgokT3KvZandCK8cGg701boOEUU";
    option telegram_chat_id = "-1001517872303";

    global telegram_payload: function(n: Notice::Info): string;
    global telegram_send_notice: function(text: string);
}

export {
        redef enum Notice::Type += {
                ## Address scans detect that a host appears to be scanning some
                ## number of destinations on a single port. This notice is
                ## generated when more than :zeek:id:`Scan::addr_scan_threshold`
                ## unique hosts are seen over the previous
                ## :zeek:id:`Scan::addr_scan_interval` time range.
                Address_Scan,

                ## Port scans detect that an attacking host appears to be
                ## scanning a single victim host on several ports.  This notice
                ## is generated when an attacking host attempts to connect to
                ## :zeek:id:`Scan::port_scan_threshold`
                ## unique ports on a single host over the previous
                ## :zeek:id:`Scan::port_scan_interval` time range.
                Port_Scan,
        };
```

```
        ## Failed connection attempts are tracked over this time interval for
        ## the address scan detection.  A higher interval will detect slower
        ## scanners, but may also yield more false positives.
        const addr_scan_interval = 5min &redef;

        ## Failed connection attempts are tracked over this time interval for
        ## the port scan detection.  A higher interval will detect slower
        ## scanners, but may also yield more false positives.
        const port_scan_interval = 1min &redef;

        ## The threshold of the unique number of hosts a scanning host has to
        ## have failed connections with on a single port.
        const addr_scan_threshold = 25.0 &redef;

        ## The threshold of the number of unique ports a scanning host has to
        ## have failed connections with on a single victim host.
        const port_scan_threshold = 15.0 &redef;

        global Scan::addr_scan_policy: hook(scanner: addr, victim: addr, scanned_port: port);
        global Scan::port_scan_policy: hook(scanner: addr, victim: addr, scanned_port: port);
}

event zeek_init() &priority=5
        {
        local r1: SumStats::Reducer = [$stream="scan.addr.fail", $apply=set(SumStats::UNIQUE), $unique_max=double_to_count(addr_scan_threshold+2)];
        SumStats::create([$name="addr-scan",
                          $epoch=addr_scan_interval,
                          $reducers=set(r1),
                          $threshold_val(key: SumStats::Key, result: SumStats::Result) =
                              {
                              return result["scan.addr.fail"]$unique+0.0;
                              },
                          #$threshold_func=check_addr_scan_threshold,
                          $threshold=addr_scan_threshold,
                          $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
                              {
                              local r = result["scan.addr.fail"];
                              local side = Site::is_local_addr(key$host) ? "local" : "remote";
                              local dur = duration_to_mins_secs(r$end-r$begin);
                              local message=fmt("%s scanned at least %d unique hosts on port %s in %s", key$host, r$unique, key$str, dur);
                              NOTICE([$note=Address_Scan,
                                      $src=key$host,
                                      $p=to_port(key$str),
                                      $sub=side,
                                      $msg=message,
```

```
        if ( hook Scan::port_scan_policy(scanner, victim, scanned_port) )
                SumStats::observe("scan.port.fail", [$host=scanner, $str=cat(victim)], [$str=cat
(scanned_port)]);
        }

function is_failed_conn(c: connection): bool
        {
        # Sr || ( (hR || ShR) && (data not sent in any direction) )
        if ( (c$orig$state == TCP_SYN_SENT && c$resp$state == TCP_RESET) ||
             (((c$orig$state == TCP_RESET && c$resp$state == TCP_SYN_ACK_SENT) ||
               (c$orig$state == TCP_RESET && c$resp$state == TCP_ESTABLISHED && "S" in c$history
)
              ) && /[Dd]/ !in c$history )
           )
                return T;
        return F;
        }

function is_reverse_failed_conn(c: connection): bool
        {
        # reverse scan i.e. conn dest is the scanner
        # sR || ( (Hr || sHr) && (data not sent in any direction) )
        if ( (c$resp$state == TCP_SYN_SENT && c$orig$state == TCP_RESET) ||
             (((c$resp$state == TCP_RESET && c$orig$state == TCP_SYN_ACK_SENT) ||
               (c$resp$state == TCP_RESET && c$orig$state == TCP_ESTABLISHED && "s" in c$history
)
              ) && /[Dd]/ !in c$history )
           )
                return T;
        return F;
        }

event connection_attempt(c: connection)
        {
        local is_reverse_scan = F;
        if ( "H" in c$history )
                is_reverse_scan = T;

        add_sumstats(c$id, is_reverse_scan);
        }

event connection_rejected(c: connection)
        {
        local is_reverse_scan = F;
```

```
        if ( "s" in c$history )
                is_reverse_scan = T;

        add_sumstats(c$id, is_reverse_scan);
        }

event connection_reset(c: connection)
        {
        if ( is_failed_conn(c) )
                add_sumstats(c$id, F);
        else if ( is_reverse_failed_conn(c) )
                add_sumstats(c$id, T);
        }

event connection_pending(c: connection)
        {
        if ( is_failed_conn(c) )
                add_sumstats(c$id, F);
        else if ( is_reverse_failed_conn(c) )
                add_sumstats(c$id, T);
        }

hook Notice::policy(n: Notice::Info)
    {
    add n$actions[Notice::ACTION_TELEGRAM];
    }


function telegram_send_notice(text: string)
    {
    if (telegram_token == "REDEF-TOKEN" || telegram_chat_id == "REDEF-ID")
        {
        Reporter::warning("Notice::telegram_token and Notice::telegram_chat_id must be redef'd t
o use Notice::ACTION_TELEGRAM");
        return;
        }
    local url = cat_sep("/", "", telegram_endpoint, cat("bot", telegram_token), "sendMessage");
    local request: ActiveHTTP::Request = ActiveHTTP::Request(
        $url=url,
        $method="POST",
        $client_data=fmt("chat_id=%s&text=%s", telegram_chat_id, text)
    );

    when ( local result = ActiveHTTP::request(request) )
        {
```

```
    when ( local result = ActiveHTTP::request(request) )
        {
        if ( result$code != 200 )
            Reporter::warning(fmt("Telegram notice failed (%d): %s", result$code, result$body));
        }
    }

function telegram_payload(n: Notice::Info): string
    {
    local text = fmt("%s: %s", n$note, n$msg);
    if ( n?$sub )
        {
        text = string_cat(text,
            fmt(" (%s)", n$sub));
        }
    if ( n?$id )
        {
        text = string_cat(text, ", Connection: ",
            fmt("%s", n$id$orig_h), ":", fmt("%d", n$id$orig_p), " -> ",
            fmt("%s", n$id$resp_h), ":", fmt("%d", n$id$resp_p));
        if ( n?$uid )
            text = string_cat(text, ", Connection uid: ", n$uid);
        }
    else if ( n?$src )
        text = string_cat(text, fmt(", Source: %s", n$src));

    return text;
    }

hook notice(n: Notice::Info)
    {
    if ( ACTION_TELEGRAM in n$actions )
        telegram_send_notice(telegram_payload(n));
    }
```

Lampiran 3 detect-FTP.zeek

```
load base/frameworks/notice
load base/utils/active-http
load base/protocols/ftp
load base/frameworks/sumstats
load base/utils/time

module Notice;
export {
    redef enum Action += {
        ACTION_TELEGRAM,
    };

    const telegram_endpoint = "https://api.telegram.org";
    option telegram_token = "5496216098:AAEf-nYbUgokT3KvZandCK8cGg701boOEUU";
    option telegram_chat_id = "-1001517872303";

    global telegram_payload: function(n: Notice::Info): string;
    global telegram_send_notice: function(text: string);


function telegram_send_notice(text: string)
    {
    if (telegram_token == "REDEF-TOKEN" || telegram_chat_id == "REDEF-ID")
        {
        Reporter::warning("Notice::telegram_token and Notice::telegram_chat_id must be redef'd t
o use Notice::ACTION_TELEGRAM");
        return;
        }
    local url = cat_sep("/", "", telegram_endpoint, cat("bot", telegram_token), "sendMessage");
    local request: ActiveHTTP::Request = ActiveHTTP::Request(
        $url=url,
        $method="POST",
        $client_data=fmt("chat_id=%s&text=%s", telegram_chat_id, text)
    );

    when ( local result = ActiveHTTP::request(request) )
        {
        if ( result$code != 200 )
            Reporter::warning(fmt("Telegram notice failed (%d): %s", result$code, result$body));
        }
    }
```

```
function telegram_payload(n: Notice::Info): string
    {
    local text = fmt("%s: %s", n$note, n$msg);
    if ( n?$sub )
        {
        text = string_cat(text,
            fmt(" (%s)", n$sub));
        }
    if ( n?$id )
        {
        text = string_cat(text, ", Connection: ",
            fmt("%s", n$id$orig_h), ":", fmt("%d", n$id$orig_p), " -> ",
            fmt("%s", n$id$resp_h), ":", fmt("%d", n$id$resp_p));
        if ( n?$uid )
            text = string_cat(text, ", Connection uid: ", n$uid);
        }
    else if ( n?$src )
        text = string_cat(text, fmt(", Source: %s", n$src));

    return text;
    }

hook notice(n: Notice::Info)
    {
        if ( ACTION_TELEGRAM in n$actions )
            telegram_send_notice(telegram_payload(n));
    }


module FTP;

export {
    redef enum Notice::Type += {
        ## Indicates a host bruteforcing FTP logins by watching for too
        ## many rejected usernames or failed passwords.
        Bruteforcing
    };

    ## How many rejected usernames or passwords are required before being
    ## considered to be bruteforcing.
    const bruteforce_threshold: double = 20 &redef;

    ## The time period in which the threshold needs to be crossed before
    ## being reset.
```

```
    ## The time period in which the threshold needs to be crossed before
    ## being reset.
    const bruteforce_measurement_interval = 15mins &redef;
}


event zeek_init()
    {
    local r1: SumStats::Reducer = [$stream="ftp.failed_auth", $apply=set(SumStats::UNIQUE), $uni
que_max=double_to_count(bruteforce_threshold+2)];
    SumStats::create([$name="ftp-detect-bruteforcing",
                      $epoch=bruteforce_measurement_interval,
                      $reducers=set(r1),
                      $threshold_val(key: SumStats::Key, result: SumStats::Result) =
                          {
                          return result["ftp.failed_auth"]$num+0.0;
                          },
                      $threshold=bruteforce_threshold,
                      $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
                          {
                          local r = result["ftp.failed_auth"];
                          local dur = duration_to_mins_secs(r$end-r$begin);
                          local plural = r$unique>1 ? "s" : "";
                          local message = fmt("%s had %d failed logins on %d FTP server%s in %s"
, key$host, r$num, r$unique, plural, dur);
                          NOTICE([$note=FTP::Bruteforcing,
                                  $src=key$host,
                                  $msg=message,
                                  $identifier=cat(key$host)]);
                          }]);
    }

event ftp_reply(c: connection, code: count, msg: string, cont_resp: bool)
    {
    local cmd = c$ftp$cmdarg$cmd;
    if ( cmd == "USER" || cmd == "PASS" )
        {
        if ( FTP::parse_ftp_reply_code(code)$x == 5 )
            SumStats::observe("ftp.failed_auth", [$host=c$id$orig_h], [$str=cat(c$id$resp_h)]);
        }
    }
```

```
hook Notice::policy(n: Notice::Info)
    {
    add n$actions[Notice::ACTION_TELEGRAM];
    }
```