Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 1, Februari 2024, pp 116-123
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Deteksi *Tunneling Domain Name System* dengan Fitur *Cache Property*

Zalfa Hilmi Abdilah[1], Atep Aulia Rahman[2]
[1,2]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Widyatama, Bandung, JawaBarat, Indonesia
Email: zalfa.hilmi@widyatama.ac.id[1], atep.aulia@widyatama.ac.id[2]

Abstract

Many companies and agencies are being attacked with data exfiltration. The attack was carried out through malware from the target by exploiting secret channels and abusing the domain system (DNS). By creating a virus by an attacker that can infect the target malware, it will generate a client tunelling, so that the attacker can enter through the communication channels he has created to the target Malware. Attackers can control malware remotely and steal data that leaks data from targets, which will affect the profitability of companies and agencies. Therefore, the author prioritizes the traces left behind of DNS tuneling that cannot be hidden by proposing the property cache feature as a method of detecting DNS tuning.

Keywords*: DNS Tunelling, Cache property features, data exfiltration, DNS*

Abstrak

Banyak perusahaan dan instansi mendapat serangan dengan adanya eksfiltrasi data. Serangan dilakukan melalui malware dari target tersebut dengan mengeksploitasi saluran rahasia dan menyalahgunakan sistem domain (DNS). Dengan dibuatnya virus oleh penyerang yang dapat menginfeksi malware target, maka akan menghasilkan client tunelling, sehingga penyerang dapat masuk melalui saluran komunikasi yang dibuatnya sampai ke malware target. Penyerang dapat mengontrol malware dari jarak jauh dan mencuri data yang menimbulkan kebocoran data dari target, hal itu akan mempengaruhi profitabilitas perusahaan dan instansi. Oleh karena itu, penulis mengutamakan jejak yang ditinggalkan dari tunelling DNS yang tidak dapat disembunyikan dengan mengusulkan fitur cache property sebagai metode deteksi tunelling DNS.

Kata kunci*: Tunelling DNS, Fitur cache property, eksfiltrasi data, DNS*

1. PENDAHULUAN

Dengan cara mengirimkan email berupa *malware* oleh penyerang terhadap target yang tampak tidak berbahaya, dan ketika target membuka email tersebut, maka target akan menginfeksif *device* mereka dengan *malware* yang dikirim oleh penyerang, sehingga membuat saluran komunikasi antara penyerang dan targetnya. Dengan demikian, para penyerang dapat mengontrol *malware* dan mencuri informasi rahasia dari target yang terinfeksi dengan adanya penyalahgunakan sistem nama *domain* (DNS) yang disebut sebagai DNS *tunneling* [1]. Adapun serangan DNS *tunelling* yang diluncurkan oleh penyerang dengan memberikan link berupa suatu akses *website* yang palsu, dan saat target mengakases link tersebut, maka akan membuat sebuah saluran lintasan berupa terowongan antara target dan penyerang. Sehingga penyerang dapat mengakses, mengontrol dan mencuri data dari target tersebut [2].

Pada tahun 1998, DNS tunelling untuk transmisi data pertama kali ditemukan [3], pada awalnya sistem tersebut dirancang untuk melewati tepi jaringan untuk mendapatkan akses internet secara gratis [4]. Karena efisien untuk

melewati mekanisme keamanan mencatat bahwa *malware remote control* berbasis DNS akan menjadi salah satu dari enam serangan jaringan yang paling berbahaya di masa depan [5]. Selain itu, *tunneling* DNS dapat digunakan untuk perintah jarak jauh dan kontrol transmisi informasi *malware*, kebocoran data, dan sebagainya [4]. Pada tahun 2014 tepatnya diantara bulan april dan desember ritel Amerika Home Depot sebelumnya telah diserang oleh *malware* yang disebut *FrameworkPos* [6], yang digunakan untuk menangkap data kartu pembayaran melalui permintaan DNS. Lebih dari 56 juta akun kartu kredit dan informasi debit pelanggan dicuri.

Untuk mendeteksi serangan DNS *tunneling* yang terjadi, beberapa tindakan telah diusulkan [7][8]. Terdapat beberapa metode yang cukup efektif untuk mendeteksi lalu lintas dari *malware*, seperti *Morto worm* [9] dan peralatan pembuat DNS Tunelling seperti *dnscat2* [10]. Namun langkah-langkah penanggulangan tersebut dibangun dengan menggunakan fitur-fitur yang dapat dengan mudah dirusak oleh teknik menggali DNS *Tunelling*. Misalnya, *steganography* dapat menyembunyikan dan membocorkan data pada nama domain yang sepenuhnya akan memenuhi syarat (FQDN) dari pernyataan *tunneling* yang membuat FQDN terlihat sah, sehingga membatalkan filter yang bergantung pada karakteristiknya. Oleh karena itu, gangguan ini mengakibatkan kebocoran data. Untuk mengatasi masalah ini, penulis akan fokus pada sifat dari DNS *Tunneling*-nya. Dengan mengekstrak data yang terkait dengan nama domain dari *query* DNS, server *cache* DNS yang terhubung dengan *malware* secara langsung harus menghindari pembuatan *cache hit* dalam server, Jika tidak data tidak akan bocor. kebocoran data melalui tuneling DNS akan memicu miss *cache* di server *cache* DNS [11].

Membuat *tunneling* pada DNS akan melanggar perilaku manusia normal karena dibutuhkan *cache* untuk dilewati, yang jelas bahwa kekurangan *cache (miss cache)* adalah penyebab sebenarnya dari DNS *Tunneling*. Oleh karena itu, dari masalah diatas, maka diusulkan tiga fitur yang berasal dari properti cache, yaitu: CHR (*Cache Hit Ratio)*, kemudian AHR (*Access Hit Ratio), dan* AMC *(Access Miss Count)*. Dengan ditunjukannya akses *miss count* akan mengarahkan beberapa kekurangan dari *hit ratio* pada kedua *cache*, maka jelas hal tersebut merupakan karakterisasi lalu lintas DNS *Tunellng*. Oleh karena itu, hal tersebut dapat berguna untuk merancang dan menerapkan *firewall* DNS yang baik dan terhindar dari DNS *Tunneling*.

2. METODOLOGI PENELITIAN

Tahapan proses peneletian ini merujuk pada penggunaan fitur *cache property* yang dapat menganalisis *query* yang masuk ke dalam server. Pada server DNS terdapat DNS *cache* yang merupakan proses penyimpanan hasil pencarian domain dalam database berdasarkan atas *nameserver* yang diminta [12]. Pada kasus yang diangkat oleh peneliti, saat terjadinya serangan *tunneling* DNS, rasio *hit cache* yang terjadi diharapkan dapat menurun, karena *query* DNS berbahaya yang dihasilkan dapat mengakibatkan ekstraksi data yang menyebabkan hilangnya cache. Dengan diusulkan tiga fitur berdasarkan *cache property*, diharapkan dapat menanggulangi serangan DNS *tunneling* dengan mengidentifikasi lalu lintas *tunneling* DNS [2] [3].

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 1, Februari 2024, pp 116-123
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Berikut terdapat beberapa fitur dari *cache property* untuk membantu proses IDS terhadap seangan DNS *tunneling*.

2.1. *Fitur Cache Property*

2.1.1. *Cache Hit Ratio*

Fitur diusulkan adalah *cache hit ratio* (CHR^n) pada server *cache* DNS, yang didefinisikan sebagai berikut:

$$\boldsymbol{CHR^n == \frac{1}{n}.\ N^n_{CH}} \quad (1)$$

n adalah jumlah *query* yang diamati, dan N^n_{CH} adalah jumlah *hit cache* yang berhasil dijalankan dalam n *query*, maka untuk menghitung *cache hit ratio* (CHR^n) pada server cache DNS adalah jumlah *cache hit* yang berhasil dijalankan dalam *query* dibagi dengan jumlah *query* yang diamati. *Hit cache* pada *server cache* DNS tersebut dapat didefinisikan sebagai keadaan di mana respons terhadap *query* dari *client* DNS ditemukan di server cache DNS yang terhubung tanpa mengirimkan *query* apa pun ke server DNS otoritatif. Oleh karena itu, *Cache Hit Ratio* sendiri merupakan fitur yang diturunkan dari *cache property* untuk mengidentifikasi lalu lintas *tunneling* DNS [2]F.

2.1.2. *Access Hit Ratio*

Untuk mengkompensasi kekurangan CHR, maka diusulkan entri akses yang memeriksa *query client* dan menyimpan informasi FQDN. Ketika FQDN dalam *query client* ditemukan dalam daftar entri akses tertentu, hal tersebut dapat dianggap sebagai hit akses. Dengan mengusulkan fitur kedua yaitu *access hit ratio* (AHR^n) , maka dapat ditentukan oleh rumus berikut:

$$AHR^n = \frac{\mathbf{1}}{\boldsymbol{n}} N\ ^nAH \quad (2)$$

n adalah jumlah *query* yang diamati dan $N\ ^nAH$ adalah adalah jumlah hits akses yang berhasil dalam n *query*. Maka untuk menghitung *access hit ratio* pada server cache DNS adalah jumlah hits akses yang berhasil dalam n *query* dibagi dengan jumlah *query* yang diamati [4].

Access hit ratio juga digunakan untuk mengevaluasi kinerja dari *cache*. Ini menunjukkan seberapa efektif *cache* dalam menyimpan data yang sering diakses dan meningkatkan performa sistem secara keseluruhan. Rasio hit yang tinggi akan meunjukkan bahwa *cache* efektif untuk menyimpan data yang paling sering diakses, sedangkan rasio hit yang rendah menunjukkan bahwa *cache* tidak efektif menyimpan data yang paling sering diakses. *Access hit ratio* juga digunakan untuk mengidentifikasi masalah yang mungkin terjadi dalam sistem *cache* seperti yang terjadi pada kasus penyerangan DNS *tunneling*, dengan menganalisis rasio hit, administrator sistem dapat mengambil tindakan untuk meningkatkan kinerja *cache* dan keseluruhan sistem pada *server cache* DNS [2].

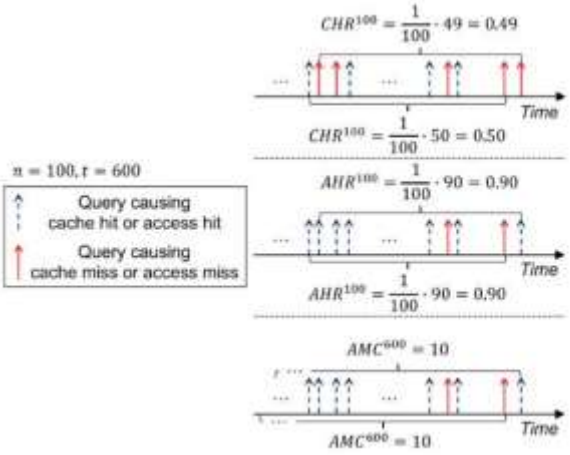
Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 1, Februari 2024, pp 116-123
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

2.1.3. *Access Miss Count*

Untuk mengkompensasi kekurangan AHR untuk difokuskan pada akses yang terlewat, maka diusulkan fitur *access miss count* AMC^t, yang ditentukan dengan rumus berikut:

$$AMC^t == N_{AM}^t \quad (3)$$

t adalah interval waktu yang satuannya detik, dan N_{AM}^t adalah jumlah *query miss* akses dalam interval waktu tersebut.



Gambar 1. Contoh perhitungan CHR, AHR, dan AMC

Berdasarkan perhitungan CHR, AHR, dan AMC pada Gambar 2, dijelaskan tidak hanya *query tunneling* tetapi juga *query* sebenarnya yang menyebabkan *cache* hilang, dan sulit untuk membedakan apakah penurunan CHR disebabkan *tunneling* DNS. Dengan mengusulkan entri akses dan AHR, maka dapat dikompensasi kekurangan CHR. Namun, *malware* dapat dengan sengaja memutar ulang *query* DNS yang sebenarnya untuk meningkatkan AHR. Selain itu, rasio masing-masing menormalkan jumlah *hit cache* dan *hit access*, yang berarti bahwa jumlah akses yang hilang itu sendiri tidak dapat dievaluasi berdasarkan fitur-fitur yang digunakan. Sehingga, AMC dapat mengatasi kekurangan hal tersebut [13].

2.2. *Technitium DNS*

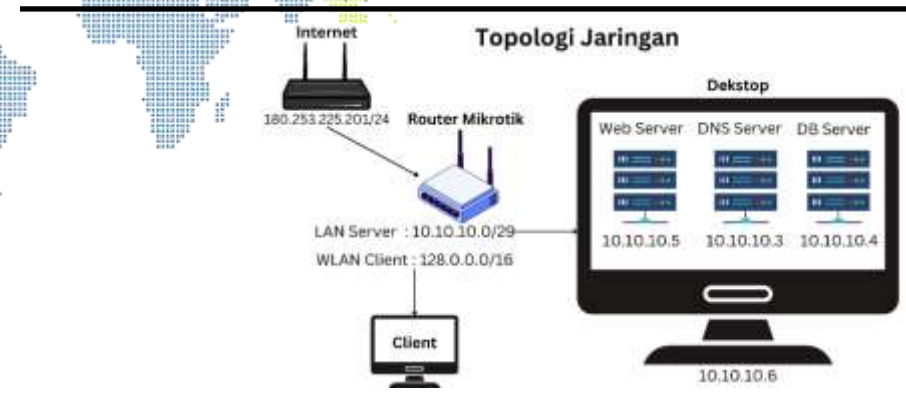
Technitium DNS Server merupakan *open source* dan *recursive* DNS yang dapat digunakan untuk *self-hosting* server DNS untuk privasi dan keamanan. *Technitium* DNS bekerja *out-of-the-box* dengan tidak ada atau minimal konfigurasi, dan menyediakan konsol web ramah pengguna yang dapat diakses menggunakan *browser web modern*. *Technitium* DNS dapat menerima dan memproses tanggapan dalam *PostProcessQueryAsync* [14].

3. HASIL DAN PEMBAHASAN

3.1. *Experimen Setup*

Untuk percobaan pemantauan lalu lintas DNS, peneliti menginstal server *cache* DNS di jaringan *local* menggunakan DNS *Technitium*.

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 1, Februari 2024, pp 116-123
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



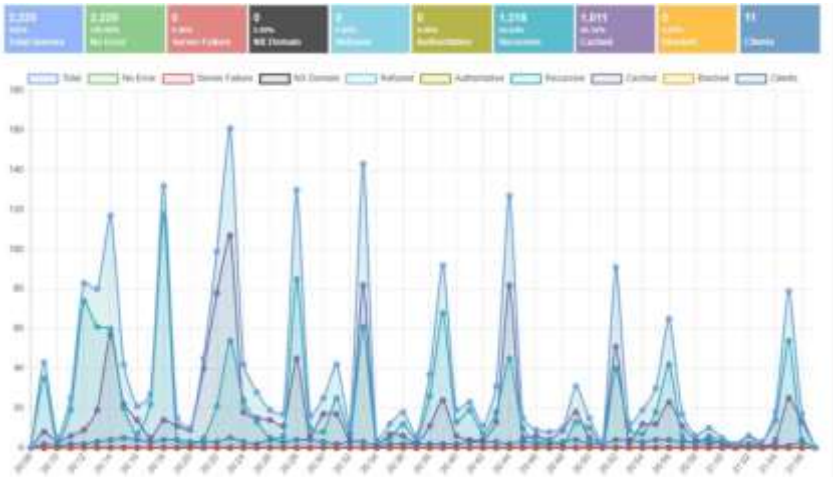
Gambar 3. Topologi Jaringan

Pada Gambar 3 merupakan topologi jaringan yang digunakan dalam proses penelitian pada jaringan *local*. Server yang digunakan adalah *ubuntu server* dan dilengkapi IPS (*Intrusion Prevention System*) *Suricata. Intrusion Prevention System* (IPS) *suricata* merupakan salah satu *too*l pengamanan pada jaringan melindungi DNS server dan server yang lainnya dari serangan *malware* [15].

Untuk menghasilkan lalu lintas *tunneling* DNS pada jaringan diatas, maka perlu mengatur server DNS dan *client tunneling* DNS. Peneliti berasusmsi bahwa *client tunneling* yang masuk ke server itu sah. Namun, sayangnya terinfeksi oleh *malware*. Oleh karena itu, *clien* akan menghasilkan lalu lintas yang sah dan *tunneling* pada DNS Server. Dalam percobaan ini, sementara menghasilkan lalu lintas *tunneling*, *client tunneling* juga akan menghasilkan trafik DNS yang sah dengan menjelajahi *Web* dan DNS yang telah terdaftar pada server. Untuk melakukan percobaan *tunel*, peneliti membuat daftar *entry cache* dan *entry log* dengan menangkap lalu lintas DNS dari 11 *client* di jaringan *local* yang sudah di *set up* selama 3600 *second*.

3.2. Monitoring DNS *Statistic*

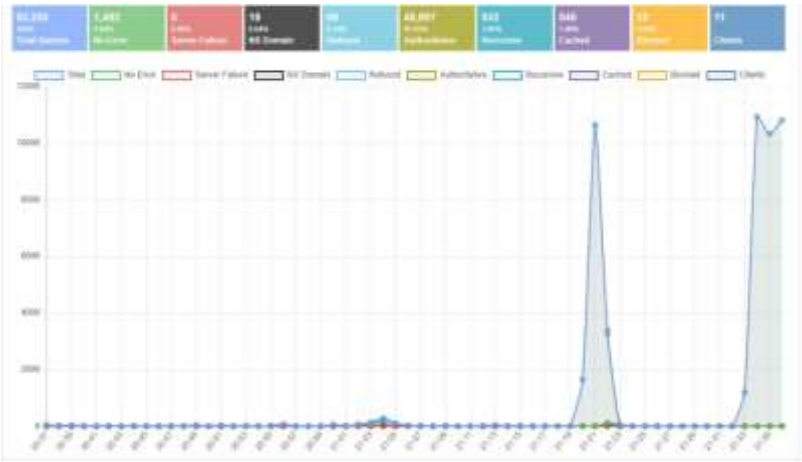
Pada penelitian ini mencoba dua kali percobaan yaitu akses normal dan akses saat terjadi DNS *Tunelling* dengan dengan kurun waktu 3600 *second*.



Gambar 4. Akses Normal

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 1, Februari 2024, pp 116-123
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

` Pada Gambar 4 merupakan akses *client* yang masuk ke server DNS secara normal tanpa terjadi *tunneling*. Dilihat hasil dari grafik diatas total *query* yang diamati 2229 (100%) dan *no error* yang dihasilkan sama dengan *query* yang masuk yaitu 2229 (100%). Kemudian *query* yang masuk tersebut berhasil di jawab oleh *cache* yaitu 1011 (45,36%) dan *recursive* sebesar 1218 (54,64%).



Gambar 5. Akses *tunneling client*

Pada Gambar 5 merupakan akses *tunneling client*. Berdasarkan grafik yang diamati dan hasil yang tertera, jumlah *query* 50393 (100%) dan no error 1482 (2,94%). *No error* tersebut sudah terjawab dari *recursive* dan *cache* yang dihasilkan, kemudian terdapat nx *domain* sebesar 18 (0,04%), *refused* 69 (0,14%), *blocked* 18 (0,04%) yang sama dengan nx *domain* dan kemungkinan *blocked* teresebut bersumber dari nx *domain*, yang terakhir *authoritative* yang dihasilkan sebesar 48897 (97,03%). Akan tetapi jika dihitung dari *query* total yang diamati sebesar 50393 dan *query* yang masuk 50446, maka *query* yang diamati < *query* masuk. Pada Gambar 4 dan 5 ditunjukan *query logging* yang dihasilkan pada DNS dengan TTL 3600 *second* sebagai berikut.

Tabel 1. *Query Logging*

Query Logging	**TTL**	**Total *query***	**Total *Client***
Query Logging Akses Normal	3600 *second*	2229	11
Query Logging Akses *Tunneling*	3600 *second*	50393	11

3.3. *Cache Property*

a) *Cache Hit Ratio* (CHR)

Hasil perhitungan yang dihasilkan pada *Cache* Hit Ratio (CHR) pada Gambar 4 dan 5 sebagai berikut.

Tabel 2. Hasil Perhitungan CHR.

Gambar	***n***	N_{CH}^{n}	CHR^{n}
6	2229	1011	0,45
7	50393	546	0,01

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 1, Februari 2024, pp 116-123
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

b) *Acces Hit Ratio* (AHR)

Hasil perhitungan yang dihasilkan dari *Access Hit Ratio* (AHR) yang ditunjukan pada Gambar 4 dan 5 sebagai berikut.

Tabel 3. Hasil Perhitungan AHR.

Gambar	*n*	N^nAH	AHR^n
6	2229	2229	1
7	50393	1482	0,02

c) *Access Miss Count* (AMC)

Hasil Perhitungan *Access Miss Count* (AMC) yang didapatkan pada Gambar 4 dan 5 sebagai berikut.

Tabel 4. Hasil Perhitungan AMC

Gambar	*n*	N^t_{AM}	AMC^t
6	2229	0	0
7	50393	73	73

4. SIMPULAN

Beberapa tindakan terhadap *tunneling* DNS telah diusulkan beberapa metode yang cukup efektif untuk mendeteksi lalu lintas dari *malware*, seperti *Morto worm* [9], peralatan pembuat DNS *Tunneling* seperti *dnscat2* [10] dan *metaploits*. Namun langkah-langkah penanggulangan tersebut dibangun dengan menggunakan fitur-fitur yang dapat dengan mudah dirusak oleh teknik menggali DNS *Tunneling*. Misalnya, *steganography* dapat menyembunyikan dan membocorkan data pada nama domain yang sepenuhnya akan memenuhi syarat (FQDN) dari pernyataan *tunneling* yang membuat FQDN terlihat sah, sehingga membatalkan filter yang bergantung pada karakteristiknya. Oleh karena itu, pendekatan konvensional tidak kuat terhadap *obfuscation* fitur. Untuk memecahkan masalah ini berfokus pada sifat *tunel* DNS. Ketika *client tunneling* mengirimkan *query* berbahaya ke server *tunneling*, *query* tersebut pasti menyebabkan kehilangan *cache* terhadap server *cache* DNS yang terhubung pada *client*. Berdasarkan pengamatan ini diusulkan fitur *cache-property-aware* untuk deteksi *tunneling* DNS. Berdasarkan hasil penelitian bahwa perhitungan akses yang hilang (*Access Miss Count*) dapat dengan jelas mengungkap lalu lintas *tunneling* DNS yang menghasilkan pertanyaan *tunneling* dalam interval waktu yang wajar, dibandingkan dengan lalu lintas pertanyaan yang sah secara umum.

DAFTAR PUSTAKA

[1] Ishikura, Naotake, et al. "Cache-property-aware features for dns tunneling detection" 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). IEEE, 2020.
[2] M. Al-Kasassbeh and T. Khairallah, “Winning tactics with DNS tunnelling” Netw. Security, vol. 2019, no. 12, pp. 12–19, 2019.
[3] Oskar Pearson, DNS Tunnel - through bastion hosts, 1998, [Online]. Available: https://seclists.org/bugtraq/1998/Apr/79.

[4] Greg Farnham, Detecting DNS tunneling, 2013, https://www.sans.org/readingroom/whitepapers/dns/detecting-dns-tunneling-34152.
[5] E. Skoudis, The six most dangerous new attack techniques and what is coming next? [Online]. Availabel: https://blogs.sans.org/pentesting/files/2012/ 03/RSA-2012-EXP-108-Skoudis-Ullrich.pdf.
[6] (2014). New FrameworkPOS Variant Exfiltrates Data via DNS Requests. [Online]. Available: https://www.gdatasoftware.com/blog/ 2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-
[7] K. Born and D. Gustafson, "Detecting DNS tunnels using charac-ter frequency analysis" 2010. [Online]. Available: https://arxiv.org/abs/ 1004.4358.
[8] S. Chen, B. Lang, H. Liu, D. Li, and C. Gao, "DNS covert channel detection method using the LSTM model" Comput. Security, vol. 104, May 2021, Art. no. 102095.
[9] (2011). Morto Worm Sets a (DNS) Record. [Online]. Available: https://community.broadcom.com/symantecenterprise/communities/ community home/librarydocuments/viewdocument?DocumentKey= 268f079a-2bb8-4775-9ef9-1b02e32ca55d&CommunityKey=1ecf5f55- 9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocumentsdnscat2. Accessed: Mar. 18, 2021. [Online]. Available: https://github. com/iagox86/dnscat2.
[10] G. K. Zipf, Human Behavior and the Principle of Least Effort. Cambridge, MA, USA: Addison-Wesley, 1949.
[11] Wijaya, Sandi. 2004, Mekanisme dan Implementasi Cache Poisoning Pada DNS Server. Bandung: ITB.
[12] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A bigram based real time DNS tunnel detection approach" Procedia Comput. Sci., vol. 17, pp. 852–860, 2013.
[13] Technitium DNS. PostProcessQueryAsync. https://github.com/ TechnitiumSoftware/DnsServer/blob/v11.0.1/DnsServerCore/Dns/ DnsServer.cs#L985, 2023.
[14] Anugrah, Faula Tanang, Syariful Ikhwan, dan Jafaruddin Gusti AG. "Implementasi Intrusion Prevention System (IPS) Menggunakan Suricata Untuk Serangan SQL Injection." Teknik: Jurnal Ilmiah Elektroteknika 21.2 (2022): 199-210.