BAB 2
Tinjauan Pustaka

Pada bab ini, kami merangkum penelitian-penelitian terdahulu yang berkaitan dengan DNS tunneling sebagai landasan dalam penelitian kami. Selain itu, ada landasan teori tentang bagaimana cara kerja DNS, DNS *tunneling*, dan *tools* yang kami gunakan dalam DNS *tunneling*. Disini juga dipaparkan tentang Elasticsearch, Packetbeat, dan Watcher yang kami gunakan untuk mendeteksi DNS *tunneling*. Kami juga menjelaskan tentang DNS *sinkhole* yang nantinya akan kami gunakan untuk memblokir DNS *tunneling* dari *output* yang dihasilkan oleh pendeteksian menggunakan Elasticsearch.

2.1 Penelitian Terdahulu

Kui Xu, Patrick Butler, Sudip Saha, dan Danfeng Yao (2013), meneliti tentang penggunaan DNS sebagai *command and control* skala masif. Penyerang, khususnya yang menggunakan botnet, memanfaatkan sistem pesan tersembunyi untuk mengatur *command and control* skala besar. Untuk memahami secara sistematis kemampuan potensial penyerang, mereka menyelidiki kelayakan menggunakan servis DNS sebagai saluran untuk *command and control botnet* yang tersembunyi. Mereka menjelaskan dan menganalisis secara kuantitatif beberapa teknik yang dapat digunakan secara efektif menyembunyikan aktivitas DNS jahat di tingkat jaringan. Penelitian ini menggunakan kumpulan data jaringan kampus sebesar 4.6GB selama dua bulan dan 1 juta nama domain yang diperoleh dari *alexa.com*. Mereka menyimpulkan bahwa saluran *command and control* berbasis DNS sangat kuat untuk menyerang.

Abdulla Dakhgan, Ali Hadi, Jaafer Al-Saraireh, Doaa Rababah (2018), menjelaskan protokol DNS dapat dimanfaatkan untuk banyak dimanfaatkan kelemahannya untuk tujuan jahat. Contohnya adalah *Fast Flux,* DNS *cache poisoning* dan *spoofing*, dan DNS *Amplification* untuk *DDoS Attack*. Paper mereka mencoba menjawab pertanyaan, mengapa Administrator jaringan harus mempertimbangkan untuk memonitor lalu lintas DNS mereka dan bagaimana mereka dapat memperoleh manfaat dari penerapan sistem *passive* DNS. Mereka menggunakan *Bro*-IDS dan *Microsoft Excel* untuk mengkalkulasi dan mengekstrak data *passive* DNS. Hasil analisis menunjukan statistik yang berfokus pada *host, port*, protokol, *domain* yang dikunjungi, dan jumlah *query*.

Penelitian tentang DNS *tunnel* pernah dilakukan oleh Irvin Homem (2017), Irvin mendeteksi DNS *tunneling* dengan menganalisis struktur paket DNS *tunneling* dan mengkarakterisasi *entropi* informasi dari protokol jaringan berbeda dan dengan DNS *tunnel* yang ekuivalen, analisis mereka menghasilkan metode rata-rata distribusi entropi, kemudian diterapkan di datasheet dan menghasilkan tingkat akurasi prediksi sebesar 75%. Metode Irvin juga tetap menjaga privasi karena tidak memparsing konten yang di tunnel, melainkan hanya menghitung entropi informasi.

Jingkun Liu, Shuhao Li, Yongzheng Zhang, Jun Xiao, Peng Chang, dan Chengwei Peng (2017), meneliti bagaimana mendeteksi DNS *tunnel* dengan metode *klasifikasi biner* berdasarkan perilaku. Mereka berpendapat metode deteksi berbasis *signature* dan *threshold-based* yang biasa ada dalam *Firewall* atau IDS tidak efektif dan sering menimbulkan *false alarm*. Pendekatan berdasarkan fitur distribusi karakter juga tidak berkinerja baik, karena penyerang dapat memodifikasi metode *encoding* untuk mengganggu distribusi karakter. Mereka mengusulkan metode yang lebih efektif yakni dengan menggunakan beberapa parameter yaitu *time-interval*, *packet size*, *record type*, dan *subdomain entropy*.

Anirban Das, Min-Yi Shen, Madhu Shashanka, dan Jisheng Wang (2018), mengembangkan *machine learning* untuk mendeteksi exfiltrasi data dari *command and control server*. Penelitian ini divalidasi dengan eksperimen dimana mereka berhasil mendeteksi *malware* yang digunakan dalam beberapa serangan *Advaced Persistent Threat*. Kebaruan dari metode mereka adalah ketahanan, kesederhanaan, skalabilitas dan kemudahan penyebaran dalam lingkungan produksi.

Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzzana Chitcyan, M. Ali Babar, Awais Rashid (2017), membuat penelitian yang bertujuan untuk mengidentifikasi dan menganalisis secara kritis serangan DNS exfiltrasi dan penanggulangannya, membuat laporan kebaharuan, dan mengurangi jarak untuk penelitian selanjutnya. Metodenya adalah analisis tematik dari 108 paper. Kesimpulan yang mereka dapatkan adalah:

a. Sebagian besar difokuskan pada tindakan pencegahan
b. Beberapa penanggulangan exfiltrasi data tidak dapat merespon secara real-time.
c. Sejumlah penanggulangan exfiltrasi data tidak mengambil privasi pengguna.
d. Penelitian saat ini berfokus hanya pada melindungi data yang sedang digunakan, bukan data yang tidak digunakan dan data transit.
e. Tidak ada standar atau framework untuk mengevaluasi penanggulangan exfiltrasi data.

Penelitian yang dilakukan oleh Guy Bruneau (2010) menyatakan DNS adalah layanan inti untuk mengakses internet, sehingga mengontrol DNS sama dengan mengontrol sebagian *traffic* Internet. Dengan mengintersepsi *traffic* DNS *request*, organisasi dapat mencegah komputer yang mencoba menghubungi *malicious* domain, seperti *botnet, spyware,* dan antivirus palsu.

Salah satu cara untuk mengontrol *traffic* DNS adalah dengan menggunakan DNS *sinkhole*, DNS *sinkhole* bekerja dengan menipu (*spoofing*) atau memberikan jawaban palsu dari sebuah *request* DNS. Ketika sebuah *client merequest* sebuah domain yang ada dalam daftar *blacklist* DNS *sinkhole*, maka DNS *sinkhole* akan memberikan jawaban berupa alamat IP yang bukan sebenarnya, seperti alamat IP 0.0.0.0 atau alamat IP *apapun* yang bukan alamat IP sesungguhnya.

DNS *sinkhole* membutuhkan daftar nama domain yang akan di *blacklist*, Administrator jaringan harus selalu memperbaharui daftar tersebut. Daftar nama domain *blacklist* bisa didapatkan di situs-situs seperti malwaredomains.com, malwareurl.com dan urlblacklist.com dan masih banyak lagi.

Logging DNS query dari beberapa sumber yang berbeda dapat menciptakan data dalam jumlah besar yang dapat diinvestigasi. *Tool* bernama *Splunk* dapat membantu membaca data dalam jumlah besar dan mencari informasi siapakah yang menggunakan DNS *tunneling* pada jaringan. Makalah dari Steve Jaworski (2019), akan memandu kita dalam membangun jaringan lab untuk menguji dan memahami berbagai *tools tunneling* DNS. Kemudian menggunakan *Splunk* dan *Splunk Stream* untuk mengumpulkan data dan mendeteksi *tunneling* DNS.

Hamad AL-Mohannadi, Irfan Awan, Jassim Al Hamar, Andrea Cullen, Jules Pagan Disso dan Lorna Armitage (2018), menjelaskan ada sejumlah teknologi yang digunakan untuk mengatasi serangan cyber, seperti *Intrusion Detection Systems* (IDS), *Intrusion Prevention Systems* (IPS), *Firewall, router* yang aktif sepanjang waktu, sistem ini menghasilkan peringatan dan mencegah serangan *cyber*. Namun itu bukan solusi langsung, karena IDS menghasilkan sejumlah besar peringatan yang mungkin tidak akurat dan berpotensi menghasilkan sejumlah alarm *false positive*. Mereka mengusulkan teknik mendeteksi serangan dengan menganalisis data *log honeypot* untuk mengidentifikasi perilaku penyerang untuk menemukan pola serangan. Untuk mencapai tujuan tersebut, mereka menggunakan *honeypot* pada *cloud AWS* untuk mengumpulkan data *log insiden*

cyber. Data *log* dianalisis dengan menggunakan teknologi Elasticsearch yang dinamakan ELK (Elasticsearch, Logstash, dan Kibana) Stack.

Greg Farnham (2013), mengulas tentang utilitas atau *tools* yang digunakan untuk tunneling DNS dan membahas teknik praktis untuk mendeteksi *tunneling* DNS. Dua kategori deteksi yang dipertimbangkan adalah *payload analysis* dan *traffic analysis*. Teknik deteksi *payload* telah digunakan untuk mendeteksi utilitas *tunneling* DNS tertentu saja, sedangkan teknik *traffic analysis* dapat digunakan untuk mendeteksi DNS *tunneling* secara universal. dengan teknik deteksi ini dapat menekan resiko yang terkait dengan DNS *tunneling*.

Literatur review diatas kami rangkum kedalam sebuah tabel dibawah ini.

Tabel 2.1 Literatur Review

No.	Peneliti	Topik Penelitian	Hasil Penelitian
1	Xu, Butler, Saha, & Yao, 2013	Meneliti kelayakan DNS sebagai saluran untuk command and control	Protokol DNS sangat kuat untuk melakukan penyerangan
2	Dakhgan, Hadi, Saraireh, & Alrababah, 2018	Menganalisis trafik DNS dengan *Bro-IDS*	Protokol DNS sangat perlu untuk dimonitoring, karena banyak dimanfaatkan untuk tujuan jahat seperti *Fast Flux*, DNS *Cache Poisoning,* DNS *Amplification DDoS*
3	Homem, Papapetrou, & Dosis, 2017	Analisis forensik DNS *tunnel* berbasis prediksi *entropi*	Analisis struktur paket DNS *tunneling* dan mengkarakterisasi entropi informasi dari protokol jaringan berbeda dan dengan DNS *tunnel* yang ekuivalen, analisis mereka menghasilkan metode rata-rata distribusi entropi, kemudian diterapkan di datasheet, menghasilkan tingkat akurasi 75%.
4	Jingkun Liu, Shuhao Li, Yongzheng Zhang, Jun Xiao, Peng Chang, dan	Deteksi DNS *tunnel* dengan metode klasifikasi	Dapat mendeteksi DNS *tunneling* dengan tingkat akurasi

No.	Peneliti	Topik Penelitian	Hasil Penelitian
	Chengwei Peng, 2017	biner berdasarkan perilaku.	hampir 99%
5	Anirban Das, Min-Yi Shen, Madhu Shashanka, dan Jisheng Wang, 2018	Deteksi DNS *exfiltration* menggunakan *machine learning.*	Mengembangkan *machine learning* untuk mendeteksi *data exfiltration* dari c*ommand and control Server*. Penelitian ini divalidasi dengan eksperimen dimana mereka berhasil mendeteksi *malware* yang digunakan dalam beberapa serangan *advaced persistent threat*.
6	Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzzana Chitcyan, M. Ali Babar, Awais Rashid, 2017	Review serangan *data exfiltration* dari luar dan penanggulangannya.	Mengidentifikasi dan menganalisis secara kritis serangan *data exfiltration* dan penanggulangannya, membuat laporan kebaharuan, dan mengurangi jarak untuk penelitian selanjutnya.
7	Guy Bruneau, 2010	DNS *sinkhole* menggunakan BIND untuk mengontrol *traffic* DNS	Membuat DNS sinkhole dengan BIND untuk mencegah *malicious activity* antara jaringan di dalam organisasi dengan Internet.
8	Steve Jaworski, 2019	Deteksi DNS *tunneling* dengan *Splunk*.	Mendeteksi DNS *tunneling* dengan *Splunk*
9	Hamad AL-Mohannadi, Irfan Awan, Jassim Al Hamar, Andrea Cullen, Jules Pagan Disso dan Lorna Armitage, 2018	Analisis ancaman *cyber* dengan *Elasticsearch* dan *Honeypot*	Perilaku dan pola penyerangan cyber dapat dideteksi dari honeypot menggunakan *elasticsearch*

No.	Peneliti	Topik Penelitian	Hasil Penelitian
10	Greg Farnham, 2013	Mendeteksi DNS *tunneling* dengan *payload analysis* dan *traffic analysis.*	Mengulas tentang *tools* yang digunakan untuk tunneling DNS dan membahas teknik untuk mendeteksi tunneling DNS. Dua kategori deteksi yang dipertimbangkan adalah *Payload Analysis* dan *Traffic Analysis*. Teknik deteksi *payload* telah digunakan untuk mendeteksi utilitas *tunneling* DNS tertentu saja, sedangkan teknik *traffic analysis* dapat digunakan untuk mendeteksi DNS *tunneling* secara universal. dengan teknik deteksi ini dapat menekan resiko yang terkait dengan *tunneling* DNS.

Dari *literature review* diatas, kami mencoba melengkapi penelitian dengan mengusulkan penelitian deteksi DNS *tunneling* dengan Elasticsearch. Pendekatan yang kami lakukan dalam permasalahan ini adalah memanfaatkan metode *traffic analysis* dengan jumlah *unique hostname* sebagai *indicator of compromise* menggunakan *elasticsearch* untuk mendeteksi DNS *tunneling*. Elasticsearch mempunyai beberapa komponen yang dapat digunakan pada penelitian ini, yaitu Packetbeats, Kibana, dan Watcher. Packetbeats berperan sebagai *sniffer realtime* yang akan melakukan *capture traffic* DNS, Watcher akan memberikan notifikasi berupa *email* kepada Administrator jaringan jika terjadi DNS *tunneling*, dan Kibana akan digunakan sebagai *dashboard panel visualisasi* dan akan menampilkan grafik *bar* nama *domain* yang paling banyak jumlah *unique hostnamenya*. Kombinasi tersebut diharapkan dapat membantu Administator dalam memantau dan mengamankan jaringan.

2.2 Landasan Teori

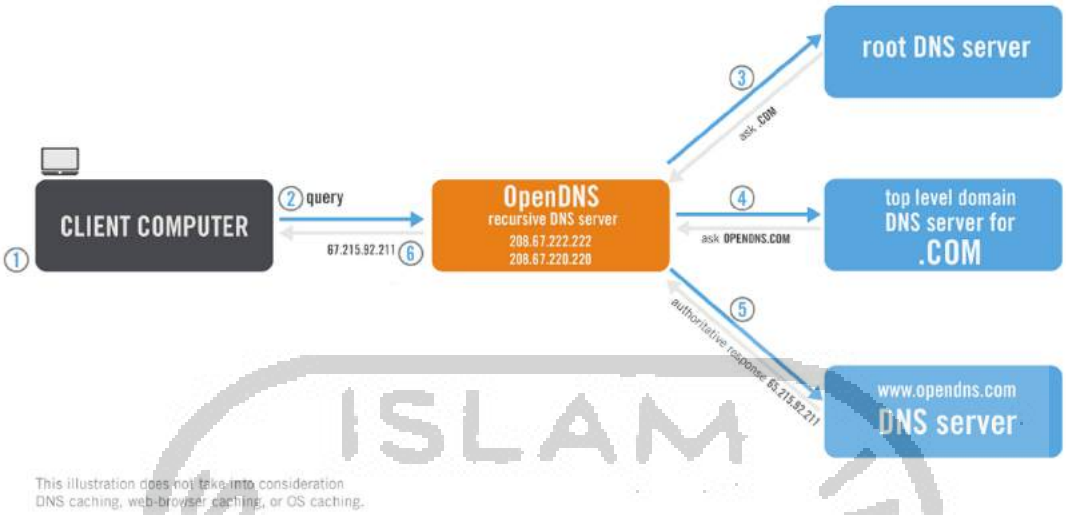
2.2.1 *Domain Name System* (DNS)

Protokol DNS adalah sistem penamaan untuk sebuah *host* komputer. DNS *server* bertugas mentranslasikan sebuah alamat IP menjadi sebuah nama hirarki yang mudah diingat, dan sebaliknya mengubah alamat *domain* menjadi alamat IP dari sebuah *host*.

Jumlah nama *domain* dan *subdomain* di internet saat ini sangat banyak dan melebihi kemampuan basis data yang kecil dan sederhana, oleh karena itu DNS dirancang sebagai basis data yang terdistribusi.

Proses bekerjanya DNS terdiri dari tiga komponen. Yang pertama adalah DNS *resolver*, yaitu sebuah program yang berjalan di komputer pengguna yang akan membuat *request* DNS. Kedua adalah DNS *server recursive*, yang bertugas melakukan pencarian melalui DNS sebagai tanggapan permintaan dari *resolver*, dan mengembalikan jawaban kepada para *resolver* tersebut. Ketiga adalah DNS *server authoritative (name server)*, yang bertugas memberikan jawaban terhadap permintaan dari DNS *server recursive*, baik dalam bentuk sebuah jawaban alamat IP, maupun dalam bentuk delegasi (misalkan: mereferensikan ke *authoritative* DNS *server* lainnya).

Proses request DNS dari sebuah *client* hingga mendapatkan jawaban dari DNS *server* diilustrasikan dalam gambar berikut.



Gambar 2.1 Proses *request* DNS dari *client* hingga mendapat jawaban dari DNS *server*

Pada gambar 2.1, komputer *client* akan mencari berapa alamat IP dari nama domain www.opendns.com (1). Kemudian *resolver* komputer client akan melakukan *query* ke *recursive* DNS *server* (2). *Recursive* DNS *server* akan mencari ke *root dns server* untuk menanyakan dimana DNS *server* untuk *domain .com* (3), kemudian DNS *server domain .com* akan menanyakan dimana *authoritative* DNS *server opendns.com* (4). DNS *server authoritative* kemudian memberikan jawaban berupa IP *address* domain www.opendns.com ke *recursive* DNS *server* (5) yang selanjutnya diteruskan ke komputer *client* yang melakukan *request* (6).

DNS memilik lebih dari 30 tipe *record*, beberapa yang sering digunakan adalah *A record*, berfungsi memetakan nama domain ke alamat IP versi 4. *AAAA record* digunakan untuk memetakan nama domain ke alamat IP versi 6. *Canonical Name record* (*CNAME record)* digunakan untuk memetakan nama domain ke nama domain lain (alias). *Mail Exchange record* (*MX* record) digunakan untuk menunjuk alamat *email server* suatu nama *domain*. *Name Server record* (*NS record)* digunakan untuk menentukan *authoritative name server* pada sebuah domain. *Pointer record* (*PTR record)* digunakan untuk memetakan alamat IP ke nama domain (*reverse* DNS). *Text record* (*TXT record)* digunakan untuk mengirimkan data berupa *text*, biasanya *record* ini digunakan oleh *Sender Policy Framework (SPF)* untuk menghindari *email* masuk *spam.* (Wong, 2006).

DNS menggunakan *port* 53 baik protokol *UDP* maupun *TCP*. *UDP* lebih sering digunakan, *TCP* hanya digunakan untuk *zone transfer* atau muatan lebih dari *512 byte*. Ada

juga mekanisme *Extension* DNS (EDNS) yang memungkinkan muatan lebih dari *512 byte* dapat menggunakan protokol *UDP* (Vixie, 1999). *E*DNS adalah fitur yang dapat dimanfaatkan untuk meningkatkan bandwidth untuk tunneling DNS.

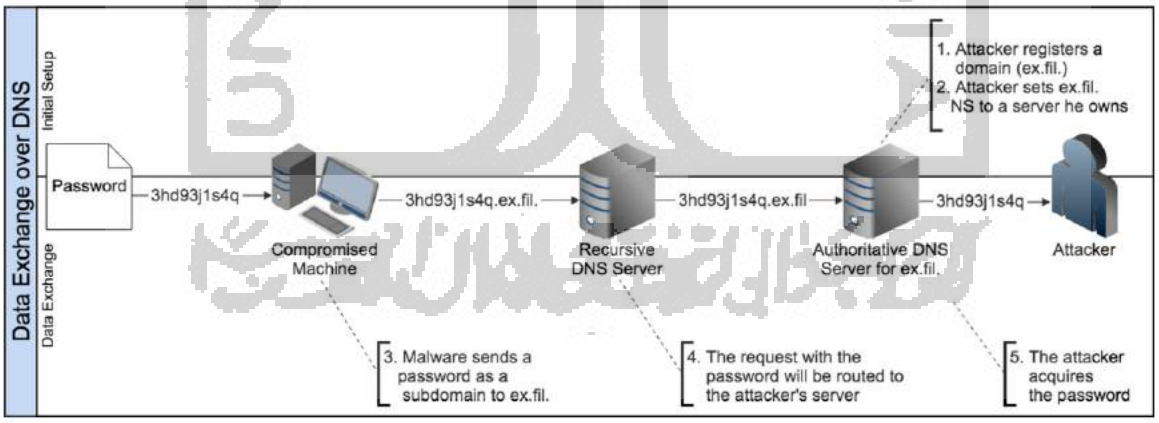
DNS dapat melakukan *caching* ketika DNS nilai *Time To Live (TTL)* diset. Server perantara DNS dapat menggunakan nilai *TTL* tersebut untuk menentukan berapa lama domain di-*cache* oleh *server*, sehingga *request* dari *client* dapat dilayani dari hasil *cache* daripada me*request* ke DNS *server* luar.

2.2.2 DNS *Tunneling*

Layanan DNS digunakan pada hampir semua jaringan Internet, namun layanan DNS biasanya tidak dimonitoring dengan baik karena dipercaya secara penuh (*blindly trusted*)(Branscombe, n.d.). Hal itulah yang kemudian dimanfaatkan oleh penyerang untuk penyalahgunaan protokol DNS, salah satunya adalah DNS *tunneling*.

Dengan DNS *tunneling*, protokol lain dapat dilewatkan melalui tunneling DNS. DNS *tunnel* dapat digunakan untuk *command and control, data exfiltration* ataupun *tunneling traffic* IP.

Pada presentasi di konfrensi RSA tahun 2012, Ed Skoudis mengidentifikasi bahwa Command and Control *malware* berbasis DNS sebagai salah satu dari enam serangan baru paling berbahaya.Ed Skoudis menyatakan penyerang menggunakan teknik ini untuk mencuri jutaan *account* (Skoudis, 2012). DNS *tunneling* juga telah terbukti dapat mencapai bandwidth 110 KB/s dengan latensi 150 ms (Leijenhorst et al., 2008).



Gambar 2.2 Proses pertukaran data melalui DNS.

Gambar diatas menjelaskan proses terjadinya pertukaran data melalui protokol DNS. Pada gambar tersebut mengilustrasikan penyerang mencuri *password* "3hd93j1s4q" dari sebuah komputer *client* atau korban *(compromised machine)*. Penyerang menggunakan

nama domain "*ex.fill"* yang akan digunakan sebagai *authoritative* DNS *tunnel server*. *Password* yang dicuri disisipkan kedalam *subdomain "ex.fill"*, kemudian komputer korban akan me-*request* ke *recursive* DNS *server*, karena tidak ditemukan jawaban dari *recursive* DNS *server*, maka akan diteruskan ke *authoritative* DNS *server* milik penyerang yang berujung *password* sampai ke tangan penyerang.

2.2.3 Komponen DNS tunnel

Diskusi yang diketahui pertama tentang DNS *tunneling* adalah dari Oskar Pearson di mailing list Bugtraq pada bulan April 1998 (Pearson, 1998). Sejak saat itu sejumlah tools DNS *tunneling* dikembangkan. Semua *tool* menggunakan teknik inti yang serupa tetapi memiliki variasi pada *encoding* dan detil implementasi lainnya. Teknik inti yang digunakan oleh *tools* tersebut adalah *domain* atau *subdomain* yang dikontrol, aplikasi sisi *server*, aplikasi sisi *client*, dan data yang di-*encoding* dalam muatan DNS (*Payload* DNS).

DNS *tunneling* membutuhkan komponen berupa DNS *tunnel server* dan DNS *tunnel client*. DNS *tunnel server* biasanya merupakan *server* yang dapat diakses internet (IP *Public*) untuk dijadikan *authoritative name server* dan dapat dikendalikan oleh pengguna *tunnel*. Sedangkan sisi *client* akan merequest DNS menggunakan ke DNS *server* dengan *name server authoritative*.

2.2.4 Encoding

Teknik *encoding* merupakan teknik untuk menyandikan data ke dalam muatan DNS. Ada bermacam-macam teknik *encoding*, dari yang sederhana hingga paling rumit. Sebagai contoh, ketika ada sebuah client DNS *tunnel* ingin mengirim data ke DNS *tunnel* server, client akan meminta *A record* dan data akan di-*encoding* ke dalam *hostname : MRZGS3TLEBWW64TFEBXXMYLMORUW4ZI.t.contoh.com*.
Kemudian server akan merespon dengan jawaban dengan *CNAME record* : *NVWW2IDPOZQWY5DJNZSQ.t.contoh.com*. Dengan cara tersebut, data apapun dapat di-*encoding* dan dikirimkan ke *server*. *Server* juga dapat merespon dengan data apapun.

Beberapa teknik *encoding* yang sering digunakan adalah *Base32* dan Base64. *Base 32* atau *5 bit encoding* biasanya digunakan oleh client untuk *request* ke *server* dengan *A record. Base32* menggunakan 32 set karakter yang berupa huruf A-Z, dan angka 2-7. Kita bisa membuat *string subdomain* dari data yang disandikan. DNS mengijinkan sampai 255 karakter total setiap *subdomain* menjadi 63 karakter atau kurang.

Base64 atau *6 bit encoding* digunakan oleh *server* untuk merespon *request* dari *client* dengan menggunakan *TXT record. TXT record* dapat menggunakan huruf besar dan kecil sehingga menjadi total 52 karakter ditambah dengan angka 10 karakter.

2.2.5 Tools DNS tunnel

Tools yang kami gunakan dalam penelitian ini sudah mewakili beberapa metode yang digunakan untuk DNS *tunneling*, yaitu DNS *tunnel, Command and Control*, dan *Data Exfiltration*. Dibawah ini adalah *tools* yang kami gunakan:

a. *Iodine*

   Iodine adalah perangkat lunak yang memungkinkan kita melakukan *tunnel* data *IPv4* melalui *server* DNS. *Tools* ini dapat digunakan di mana akses internet di-*firewall*, tetapi request DNS diizinkan ("Iodine DNS tunneling Tools," n.d.). Untuk menggunakan aplikasi ini, kita membutuhkan nama *domain* sungguhan dan sebuah *server* dengan *IP publik* yang akan dijadikan sebagai *authoritative name server*. Pada beberapa sistem operasi seperti *Linux Ubuntu* dan *CentOS*, *Iodine* dapat diinstall dengan mudah menggunakan software manager.

b. *Dnscat2*

   Tool ini dirancang untuk membuat saluran *command and control* (C&C) terenkripsi melalui protokol DNS, yang merupakan *tunnel* efektif dari hampir setiap jaringan ("DNScat2 C&C tunneling tools," n.d.). dnscat2 juga membutuhkan nama domain sungguhan dan server dengan ip public untuk dijadikan authoritative name server.

c. *Malware* DNSExfiltrator

   Pada percobaan ini kami juga menggunakan *malware* bernama DNS*Exfiltrator* . *Malware* ini terdaftar pada situs virustotal.com dengan nilai hash "*ed937bcd5dc05f1021aa83afdb47af266083ef47228e23a32292bad577c53191*". *Malware* ini dapat mengirimkan file melalui protokol DNS. Pada sisi *server*, *malware* ini menggunakan bahasa *python*, sedangkan sisi *client* (sisi korban) versi yang kami gunakan adalah *powershell windows*.

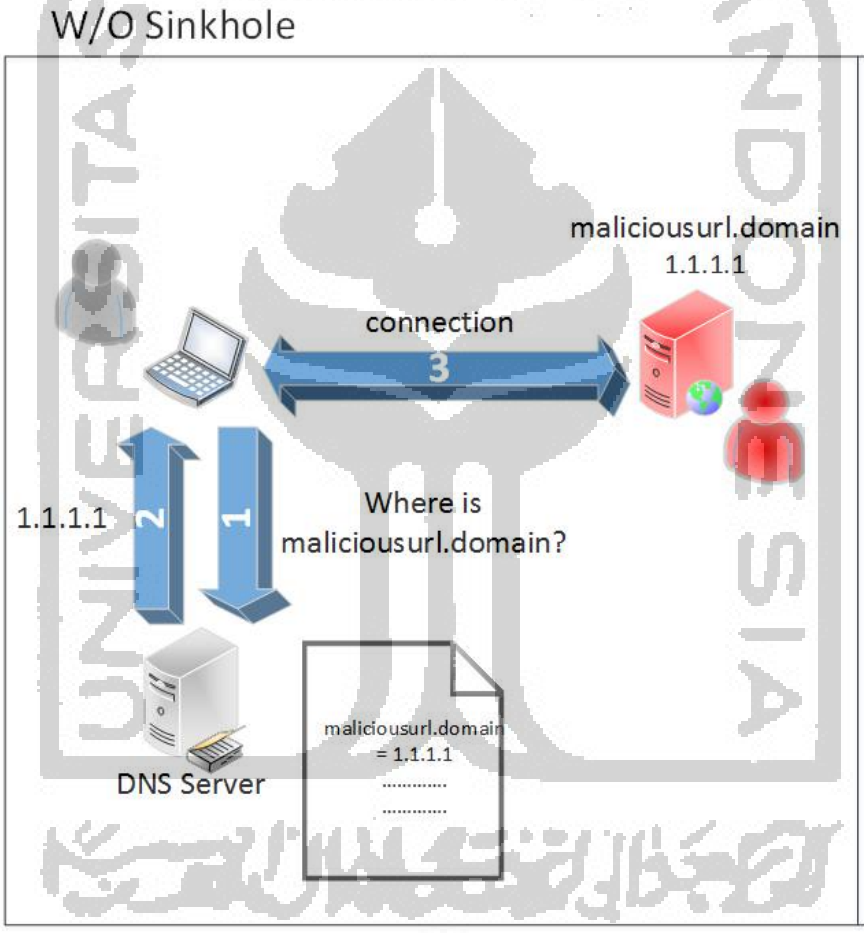
2.2.6 DNS Sinkhole

DNS *sinkhole* adalah sebuah DNS *server* yang bekerja dengan menipu (*spoofing*) atau memberikan jawaban palsu dari sebuah *request* DNS. Ketika sebuah *client* me-*request* sebuah domain yang ada dalam daftar *blacklist* DNS *sinkhole*, maka DNS *sinkhole* akan

memberikan jawaban berupa alamat IP yang bukan sebenarnya, seperti alamat IP 0.0.0.0 atau alamat IP apapun yang bukan alamat IP sesungguhnya (Bruneau, 2010).

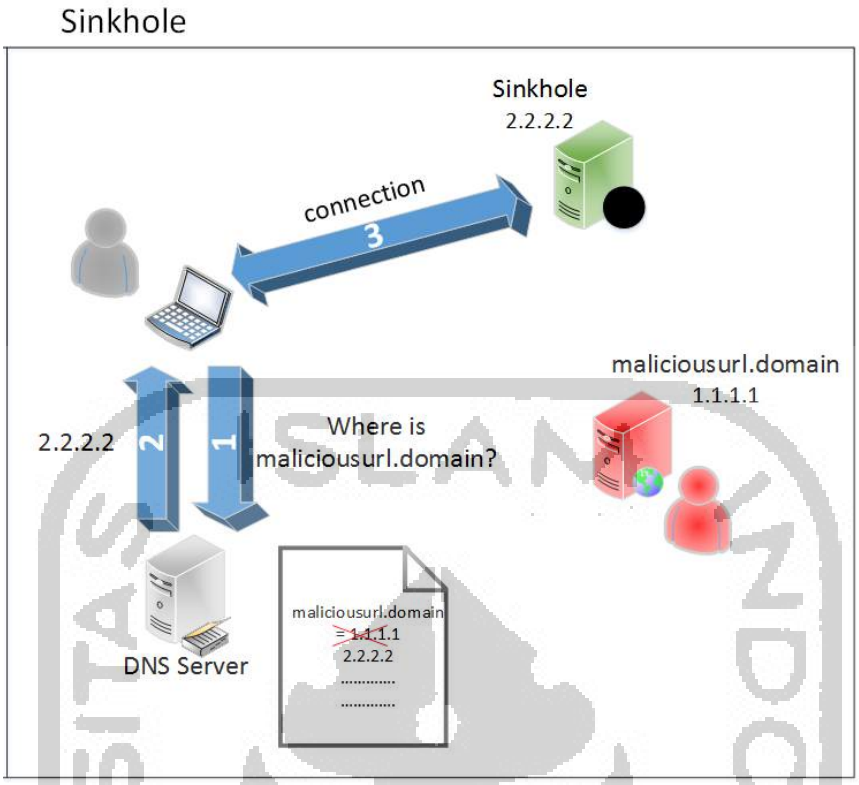
Dengan DNS *sinkhole*, komputer client tidak bisa mengakses nama domain yang tidak diperbolehkan seperti *malware*, *botnet*, *command* and *control*. DNS *sinkhole* dapat dibuat menggunakan aplikasi DNS *server* seperti BIND. Selain itu fitur DNS *sinkhole* juga dapat ditemukan pada aplikasi *firewall* seperti Palo Alto Network, dan Fortinet.

DNS *sinkhole* juga bisa dimanfaatkan untuk hal lain seperti yang dilakukan oleh aplikasi Pi-Hole, yakni sebuah DNS *sinkhole* yang difokuskan untuk memblokir iklan dan konten yang tidak diinginkan (Pi-Hole, 2019).



Gambar 2.2 Proses *request* sebuah nama *domain* oleh *client* ke DNS *server* tanpa *sinkhole*

Gambar 2.2 menunjukan sebuah proses *request* dan *response* dari *client* yang akan menghubungi *malicious domain* ke DNS *server* tanpa DNS *sinkhole*. (1) client me-*request* ke DNS *server* berapakah alamat IP dari *domain* maliciousrul.domain. (2) DNS *server* memberikan jawaban kepada *client* bahwa IP dari domain maliciousurl.domain adalah 1.1.1.1. (3) client terhubung ke *server* malicious.domain yang sesungguhnya.



Gambar 2.3 Proses request sebuah nama domain oleh client ke DNS server dengan sinkhole

Gambar 2.3 menunjukkan proses *request* dan *response* dari *client* yang akan menghubungi *malicious domain* ke *server* dengan menambahkan DNS *sinkhole*. (1) *client* me-*request* ke DNS *server* berapakah alamat IP dari *domain* maliciousrul.domain. (2) DNS *server* memberikan jawaban kepada *client* bahwa IP dari *domain* maliciousurl.domain adalah 2.2.2.2. (3) *client* tidak terhubung ke *server* malicious.domain yang sesungguhnya, namun diarahkan ke *server* lain.

DNS *sinkhole* membutuhkan daftar nama *domain blacklist* yang akan di blokir. Daftar nama *domain* bisa didapatkan di beberapa situs seperti urlblacklist.com, malwaredomains.com dan lainnya. Pada penelitian ini, kami akan menggunakan DNS *sinkhole* untuk memblokir nama *domain* yang kami peroleh dari hasil pendeteksian DNS *tunneling* dengan Elasticsearch.

2.2.7 Elasticsearch

Elasticsearch adalah mesin pencari yang dibangun berbasis Apache Lucene dan merupakan produk *opensource* dan dikembangan dengan Java. Elasticsearch dapat digunakan untuk melakukan analisis secara *realtime* dan terdistribusi dan dapat melakukan berbagai jenis mekanisme pencarian. Elasticsearch juga dapat digunakan untuk manajemen berbagai

macam *log*, seperti *log* sistem operasi, *web server*, *log traffic, log* aplikasi, dan *log* pada Amazon Web Service (Elasticsearch, n.d.).

Data mentah mengalir ke Elasticsearch dari berbagai sumber, termasuk *log*, metrik sistem, dan aplikasi web. *Data Ingestion* adalah proses di mana data mentah ini diuraikan, dinormalisasi, dan diperkaya sebelum di-indeks dalam Elasticsearch. Setelah di-indeks di Elasticsearch, pengguna dapat menjalankan *query* kompleks terhadap data mereka dan menggunakan agregasi untuk mengambil ringkasan kompleks dari data mereka. Dari Kibana, pengguna dapat membuat visualisasi data, berbagi dasbor, dan mengelola Elastic Stack.

Kecepatan dan skalabilitas Elasticsearch dan kemampuannya untuk mengindeks berbagai jenis konten sehingga ia dapat digunakan untuk sejumlah kasus penggunaan:

- Pencarian aplikasi
- Pencarian situs web
- Pencarian perusahaan
- Logging dan log analytics
- Metrik infrastruktur dan pemantauan wadah
- Pemantauan kinerja aplikasi
- Analisis dan visualisasi data geospasial
- Analisis keamanan
- Analisis bisnis

Elasticsearch mempunyai beberapa *Add-on* dengan berbagai macam fungsi, seperti Beat untuk *logging* (Filebeat, Packetbeat, Topbeat, Libbeat, Winlogbeat), Watcher untuk *alerting*, Kibana untuk *dashboard visualisasi*, dan masih banyak lagi.

2.2.8 Packetbeat

Packetbeat adalah pengirim dan penganalisis data paket jaringan yang terintegrasi ke dalam Elastic Stack. Sebagai anggota keluarga Elastic pengirim *log* (Filebeat, Topbeat, Libbeat, Winlogbeat), Packetbeat menyediakan metrik pemantauan *real time* di *web*, *database*, DNS dan protokol jaringan lainnya dengan memantau paket-paket aktual yang ditransfer melalui media jaringan seperti kabel .

Memantau paket data dengan ELK Stack dapat membantu mendeteksi tingkat lalu lintas jaringan dan karakteristik paket yang tidak biasa, mengidentifikasi sumber paket dan tujuan, mencari *string* data spesifik dalam paket, dan membuat dasbor ramah pengguna dengan statistik yang mendalam. Pemantauan paket dapat melengkapi langkah-langkah

keamanan lainnya dan membantu meningkatkan waktu respons kita terhadap serangan berbahaya. Berikut gambar contoh *capture packet* DNS dengan Packetbeat.

Gambar 2.4 Contoh *capture* DNS dengan Packetbeat

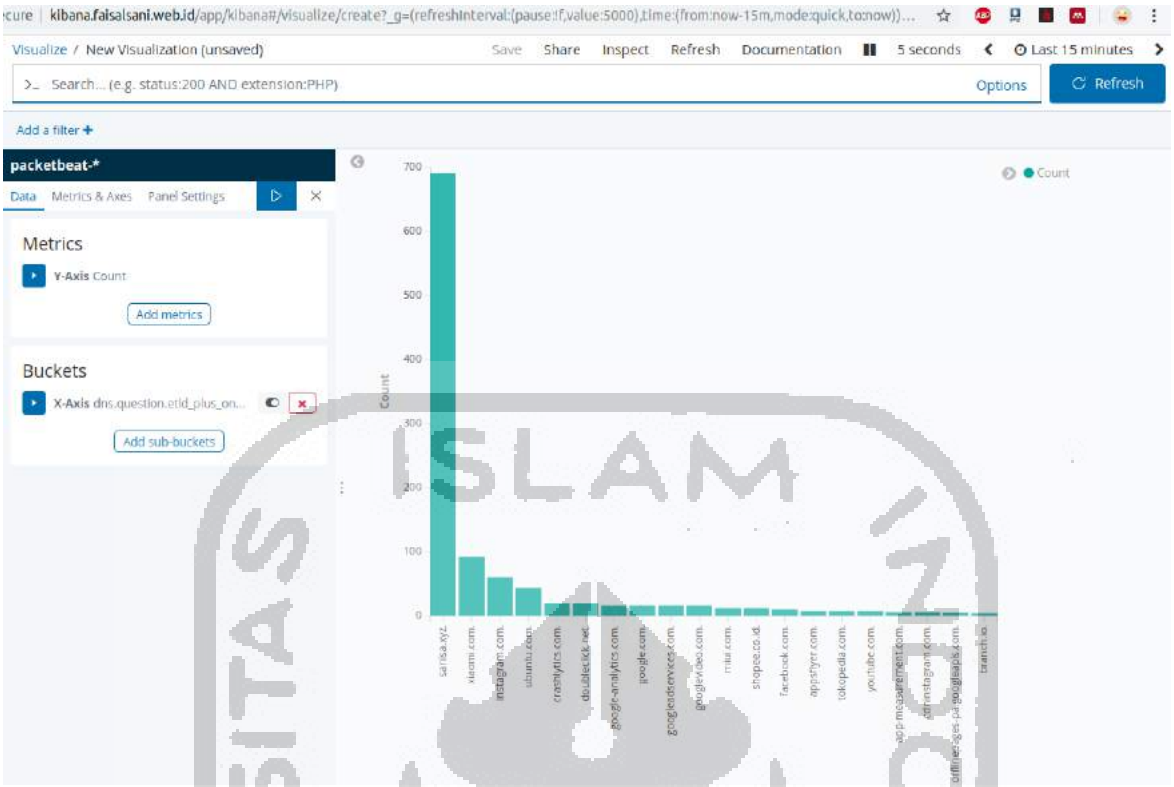
2.2.9 Watcher

Watcher adalah fitur Elasticsearch yang dapat Anda gunakan untuk membuat tindakan berdasarkan kondisi, yang secara berkala dievaluasi menggunakan *query* pada data Anda. Pada penelitian ini, kami memanfaatkan Watcher untuk mengirimkan notifikasi berupa *email* kepada Administrator jaringan jika nilai *unique hostname* lebih dari 300.

2.2.10 Kibana

Kibana adalah plugin visualisasi data *open source* untuk Elasticsearch. *plugin* ini mempunyai kemampuan visualisasi untuk konten yang diindeks pada *cluster* Elasticsearch. Pengguna dapat membuat grafik *bar*, garis, dan sebaran plot, atau diagram *pie* dan peta di atas volume data yang besar. (Elastic, n.d.)

Kibana juga menyediakan alat presentasi, yang disebut sebagai *canvas*, yang memungkinkan pengguna untuk membuat *slide deck* yang menarik data langsung langsung dari Elasticsearch.



Gambar 2.5 Contoh tampilan *dashboard kibana*

Pada penelitian ini, hasil dari pendeteksian menggunakan Elasticsearch, Packetbeat, dan Watcher akan ditampilkan pada dashboard Kibana. Kibana akan menampilkan grafik *bar* nama *domain* yang paling banyak jumlah *unique hostname*.