

RESEARCH ARTICLE

## Desain *Attack Tree* Berdasarkan Eksploitasi Dan Analisis GraphQL Menggunakan Teknik *Injection* Dengan Metrik *Time*

Nabil Egan Valentino , Adityas Widjajarto\* and Umar Yunan Kurnia Septo

Fakultas Rekayasa Industri, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia

\*Corresponding author: [adtwjrt@telkomuniversity.ac.id](mailto:adtwjrt@telkomuniversity.ac.id)

### Abstrak

*Graph Query Language* adalah bahasa *query* yang menentukan klien berinteraksi dengan API. GraphQL juga memiliki titik kelemahan yang dimana dapat menimbulkan kerentanan seperti kerentanan injeksi. Penelitian ini dilakukan dengan tujuan untuk menemukan teknik injeksi mana yang paling efektif dengan dua mode keamanan yang berbeda dengan melakukan perbandingan teknik injeksi mana yang membutuhkan waktu lebih cepat dengan kedepannya dapat menemukan solusi keamanan. Dalam penelitian eksploitasi menggunakan tiga teknik injeksi yang berbeda yaitu *Command*, *Log* dan *Spoof Injection*. Eksploitasi terhadap GraphQL menggunakan teknik injeksi dalam bentuk *attack tree* dengan tujuan untuk mengetahui relasi eksploitasi *attack tree* berdasarkan metrik *time*. Dari waktu eksploitasi tiga teknik injeksi tersebut telah didapatkan bahwa *spoof injection* membutuhkan waktu paling singkat dengan mode sebelum *hardening* dengan total waktu 32,63s. Dan teknik injeksi yang membutuhkan waktu paling lama yaitu *command injection* dengan total waktu 60,15s. Pada keamanan setelah *hardening*, terjadi perubahan waktu yang dimana *log injection* berada di urutan pertama dengan jumlah waktu 38,19 s dan pada urutan terakhir, *command injection* dengan jumlah waktu 53,52s. Sehingga dapat disimpulkan bahwa serangan injeksi yang efektif sebelum *hardening* adalah *spoof injection* sedangkan serangan injeksi yang efektif setelah *hardening* adalah *log injection*, dimanaserangan injeksi tersebut dapat ditemukan solusi yang paling efektif.

**Key words:** *Attack Tree*, Eksploitasi, GraphQL, Injeksi, *Time*

### Pendahuluan

Di era digital saat ini, teknologi telah menjadi suatu bagian yang tidak dapat terpisahkan dari kehidupan sehari-hari. Salah satu teknologi yang telah menarik banyak perhatian yaitu *Application Programming Interface* (API). API memfasilitasi komunikasi antar suatu aplikasi dan berbagi data serta fungsionalitas. Akan tetapi, API yang sering dipakai sekarang saat ini yaitu REST API yang dimana API ini memiliki cukup banyak kemudahan yaitu dimana dia memiliki kinerja yang cukup baik, kompatibilitasnya terhadap banyak *platform* dan arsitektur yang terstruktur [1]. Namun semua itu tidak ada bandingnya dengan salah satu jenis API yang telah mendapatkan popularitas adalah GraphQL [2]. GraphQL merupakan bahasa *query* untuk API yang memberikan efisiensi, kekuatan, dan fleksibilitas yang luar biasa bagi pengembang yang dimana GraphQL mempunyai kelebihan yang tidak dimiliki oleh REST yaitu dapat melakukan pengambilan data yang lebih efisien dan dapat menghindari yang namanya *over-fetching* [3]. Namun, seperti semua teknologi yang ada. GraphQL juga mempunyai kerentanannya tersendiri. Salah satu kerentanan terbesar dalam penggunaan GraphQL adalah *injection vulnerabilities*. *Injection vulnerabilities* adalah salah satu jenis kerentanan keamanan yang dilakukan dengan memanipulasi *query* atau perintah yang dikirim ke aplikasi, dengan potensi untuk merusak ataupun mencuri data yang ada. Dalam penelitian ini,

penulis ingin mencari efektivitas dari teknik injeksi mana yang membutuhkan waktu secara efektif dalam melakukan eksploitasi terhadap GraphQL. Teknik injeksi yang digunakan dalam penelitian ini adalah *Log Injection*, *Spoof Injection*, dan *Command Injection*. Penelitian ini hanya berfokus pada serangan GraphQL. Dimana yang perlu diketahui serangan dengan menggunakan teknik injeksi sangat mengancam keamanan data aplikasi yang dimana pada penelitian ini waktu akan dijadikan sebagai perbandingan dalam melakukan eksploitasi.

### Tinjauan Pustaka

#### Kali Linux

Digunakan oleh profesional keamanan dalam berbagai bidang, seperti pengujian penetrasi, forensik, dan pengujian kerentanan, Kali Linux adalah *platform* pengujian penetrasi yang cukup paling populer dan terkena. Kali Linux Sekarang merupakan kerangka kerja pengujian penetrasi yang lengkap dan banyak memanfaatkan banyak fitur Debian GNU/Linux dari komunitas *open source* yang dinamis di seluruh dunia. Kali Linux dirancang untuk memenuhi kebutuhan tester penetrasi profesional, penggemar keamanan dalam melalui kerangka kerja yang fleksibel [4].

### Cyber Threat

*Cyber threat* adalah pelanggaran hukum yang dilakukan melalui internet. Berbagai bentuk ancaman siber ini termasuk *phishing*, *malware*, serangan pada perangkat IoT, serangan *denial of service*, *spam*, serangan pada jaringan atau perangkat seluler, penipuan finansial, dan *ransomware*. Ancaman ini dapat mencoba mencuri data, melanggar aturan integritas, dan merusak perangkat komputasi atau jaringan. Dengan meningkatnya ketergantungan pada ruang *cyber* dan kemajuan teknologi, metode deteksi dan perlindungan yang lebih canggih diperlukan untuk menghadapi ancaman *cyber* [5].

### SQL Injection

*SQL Injection* adalah metode serangan *cyber* di mana penyerang memasukkan *query* SQL berbahaya ke dalam *input* data ke dalam aplikasi web. Tujuan utama dari *SQL Injection* adalah untuk mengubah *query* SQL yang dieksekusi oleh sistem *database*, sehingga penyerang dapat mendapatkan akses tidak sah ke data, mengubah data, atau bahkan merusak *database* secara keseluruhan [6].

### GraphQL

GraphQL merupakan bahasa *query* yang dikembangkan oleh Facebook untuk menerapkan arsitektur layanan web, memungkinkan klien untuk mengidentifikasi data khusus yang mereka butuhkan dari penyedia layanan, menghindari masalah *over-fetching* [7].

### Damn Vulnerable GraphQL Application (DVGA)

*Damn Vulnerable GraphQL Application* (DVGA) dirancang dengan kelemahan keamanan untuk pembelajaran dan pengujian. DVGA digunakan sebagai bukti konsep untuk menunjukkan potensi serangan seperti serangan injeksi pada aplikasi GraphQL. Dalam penelitian ini, DVGA digunakan untuk mengevaluasi efektivitas metode dalam menemukan *query* GraphQL yang dapat digunakan untuk melakukan serangan injeksi. DVGA ini dapat di unduh pada *link* <https://github.com/dolevf/DamnVulnerableGraphQLApplication>. Dengan menggunakan DVGA, para peneliti dapat menguji dan mengembangkan metode keamanan tanpa mengancam merusak aplikasi yang sebenarnya.

### Log Injection

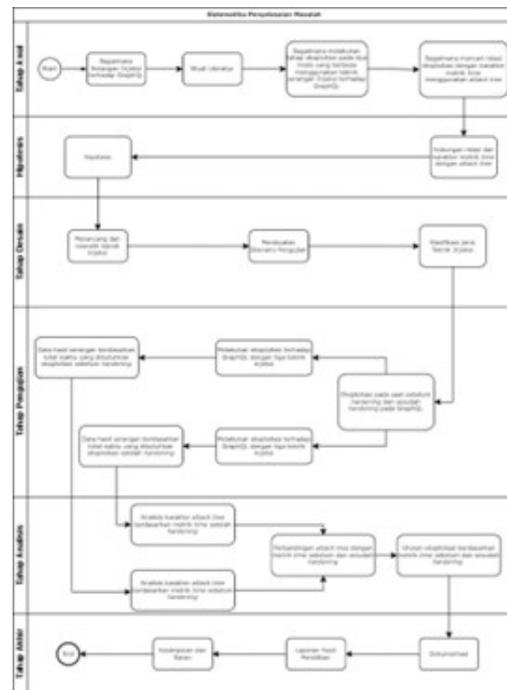
*Log Injection* adalah kerentanan keamanan yang terjadi pada aplikasi web ketika data berbahaya dimasukkan ke dalam *log* oleh pengguna. Ketika antarmuka web melihat *log*, data berbahaya yang disuntikkan akan dieksekusi sebagai kode, yang dapat menyebabkan kerusakan yang berkelanjutan pada aplikasi web dan memungkinkan pengguna yang tidak terotentikasi untuk menyuntikkan data berbahaya ke dalam *log*, sehingga menurunkan syarat eksploitasi kerentanan [8].

### Spoofing

*Spoofing* adalah jenis serangan *cyber* di mana pelaku kejahatan mencoba menipu sistem atau pengguna dengan menyamar sebagai orang lain. Tujuan dari *spoofing* adalah untuk mendapatkan akses tidak sah, mencuri data sensitif, menyebarkan *malware*, atau mengelabui pengguna agar memberikan informasi pribadi [9].

### Command Injection

*Command Injection* merupakan ancaman paling serius terhadap keamanan aplikasi web, yang dimana dapat memungkinkan penyerang untuk memasukkan perintah yang tidak diinginkan ke dalam server aplikasi untuk mengkompromikan aplikasi ataupun datanya. Serangan ini juga memungkinkan penyerang untuk mendapatkan akses tidak sah ke server aplikasi web yang rentan atau mengambil informasi [10].



Gambar 1. Sistematika Penyelesaian Masalah

### Metrik Time

Salah satu cara untuk mengukur waktu tanggap atau *respons* adalah dengan menggunakan metrik *time*, metrik ini mempunyai peran yang sangat penting karena waktu tanggap yang cepat menunjukkan seberapa efektif dan efisien sistem menangani permintaan [11].

### Attack Tree

*Attack Tree* adalah teknik pemodelan serangan yang digunakan untuk memvisualisasikan urutan dan/atau kombinasi peristiwa yang memungkinkan serangan *cyber* yang berhasil pada komputer atau jaringan. *Attack Tree* ini secara khusus digunakan untuk menampilkan serangan *cyber* secara grafis, dengan setiap *node* atau simpul dalam pohon menggambarkan langkah-langkah atau eksploitasi yang dilakukan oleh penyerang [12].

### Metodologi Penelitian

Sistematika penelitian digambarkan sebagai jalan yang tersusun yang membantu menyelesaikan masalah yang muncul selama penelitian. Awal, hipotesis, desain, pengujian, analisis, dan akhir adalah enam tahapan metodologi utama yang harus dilakukan. Sistematika penyelesaian masalah dijelaskan dalam diagram berikut:

#### 1. Tahap Awal

Tahapan awal dari penelitian adalah identifikasi bagaimana tahapan serangan injeksi terhadap GraphQL. Setelah itu, penelitian literatur dapat dilakukan untuk memastikan bahwa masalah yang ada relevan dan dapat memperdalam teori tentang tahapan serangan injeksi. Setelah tahapan studi literatur selesai, pengujian serangan injeksi dilakukan dengan metrik *time*. Kemudian masuk ke fase selanjutnya yaitu fase analisa terhadap relasi yang dimana mempunyai hubungan metrik *time* berdasarkan teknik serangan injeksi.

## 2. Tahap Hipotesis

Setelah tahap awal selesai, tahap hipotesis dimulai. Pada tahap ini, hipotesis dibuat untuk menghasilkan praduga terhadap hipotesis yang berkaitan dengan hubungan dan karakter metrik *time* dengan *attack tree*.

## 3. Tahap Desain

Setelah tahap hipotesis dilaksanakan, dilanjutkan ke tahap selanjutnya yaitu tahap desain. Dimana pada tahap ini merancang dan memilih teknik injeksi yang akan digunakan serta pembuatan skenario pengujian, setelah kedua itu selesai dilaksanakan, selanjutnya dapat melakukan klasifikasi jenis dari teknik injeksi yang akan digunakan untuk melakukan eksploitasi pada tahap pengujian.

## 4. Tahap Pengujian

Pada tahap pengujian ini akan dilakukan tahapan serangan injeksi dengan tiga teknik injeksi yang terpilih yang memiliki dua kondisi pengujian eksploitasi yaitu sebagai berikut:

- Serangan menggunakan teknik injeksi sebelum *hardening*.
- Serangan menggunakan teknik injeksi setelah *hardening*.

Setelah dilakukannya pengujian serangan injeksi dilaksanakan dengan dua kondisi yang berbeda, akan menghasilkan sebuah data berdasarkan ukuran pengujian serangan injeksi terhadap GraphQL. Data-data yang dihasilkan yaitu antara lain:

- Data hasil serangan berdasarkan teknik injeksi sebelum *hardening*.
- Data hasil serangan berdasarkan teknik injeksi setelah *hardening*.

## 5. Tahap Analisis

Pada tahap analisis akan dilakukan analisis karakter yang digambar dalam bentuk diagram *attack tree*. *Attack tree* dianalisis berdasarkan dengan data hasil metrik *time* dengan memiliki dua kondisi yaitu:

- Analisis eksploitasi sebelum *hardening*.
- Analisis eksploitasi setelah *hardening*.

Analisis akan dilakukan dengan tujuan untuk mengukur eksploitasi yang dimana akan menghasilkan metrik *time*. Setelah hasil analisis dapat menunjukkan karakteristik *attack tree* yang berkaitan dengan metrik *time*, akan dilakukan perbandingan teknik injeksi mana yang sangat cepat untuk di eksploitasi dengan sebelum dan sesudah *hardening*. Setelah hasil data sudah dibandingkan, data akan diolah menggunakan metrik *time*.

## 6. Tahap Akhir

Tahap akhir merupakan penarikan kesimpulan dari hasil pengujian eksploitasi yang dimana menghasilkan data akhir yang digunakan untuk analisis mengenai *attack tree* berdasarkan metrik *time* serta saran akan dituliskan pada laporan hasil penelitian.

## Hasil dan Pembahasan

Eksperimen dan Data yang digunakan seperti Skenario Percobaan, Implementasi Percobaan, dan Data Percobaan. Dalam pengujian dan percobaan kali ini digunakan juga perangkat keras, perangkat lunak dan spesifikasi terdapat pada tabel 1. Menurut tabel 1 & 2 beberapa spesifikasi yang digunakan dalam pengujian dan penelitian disebutkan. Fungsi masing-masing perangkat lunak yang digunakan dijelaskan di bawah ini:

**Table 1.** Spesifikasi Perangkat Keras

Spesifikasi Perangkat Keras		
Komponen	Informasi	
Core Hardware Specification	Processor	NVIDIA GeForce GTX 1650 with MaxQ Design
	Memory	16384 MB DDR4 RAM
	Hard Disk	1 TB SSD
	System Type	64-bit Operating System, x64-based processor
Virtual Machine Specification	Operating System	Windows 10 Home Single Language 64bit (10.0, Build 19045)
	Processor	NVIDIA GeForce GTX 1650 with MaxQ Design
	Memory	2 GB RAM
	Hard Disk	80.1 GB SSD
Virtual Machine Specification	System Type	64-bit Operating System, x64-based processor
	Operating System	Kali Linux 2023.3

**Table 2.** Spesifikasi Perangkat Lunak

Komponen	Informasi	
Core Hardware Specification	Processor	NVIDIA GeForce GTX 1650 with MaxQ Design
	Memory	16384 MB DDR4 RAM
	Hard Disk	1 TB SSD
	System Type	64-bit Operating System, x64-based processor
Virtual Machine Specification	Operating System	Windows 10 Home Single Language 64bit (10.0, Build 19045)
	Processor	NVIDIA GeForce GTX 1650 with MaxQ Design
	Memory	2 GB RAM
	Hard Disk	80.1 GB SSD
Virtual Machine Specification	System Type	64-bit Operating System, x64-based processor
	Operating System	Kali Linux 2023.3

**Table 3.** Pengukuran Waktu Eksploitasi *Command Injection* Sebelum Hardening

No.	Step	Time Eksploitasi 1 (s)		
		Real	User	System
1.	Web Browser Changing	18,01	6,22	2,67
2.	Graphw00f web scanning	0,27	0,09	0,07
3.	<i>Altair Inject Command</i>	41,87	5,91	1,23
<i>Total Time</i>		60, 15		

**Table 4.** Pengukuran Waktu Eksploitasi *Command Injection* Setelah *Hardening*

No.	Step	Time Eksploitasi 1 (s)		
		Real	User	System
1.	Web Browser Changing	24,25	6,56	0,14
2.	Gecurity Level			
3.	<i>Altair Inject Command</i>	19,00	5,03	1,10
	<i>Total Time</i>		53,52	

### Perating System

- Kali Linux  
Kali Linux adalah sistem operasi *open-source* yang berbasis Debian yang dibuat oleh *Offensive Security* sebagai pengganti *BackTrack* Linux pada tahun 2013. Dirancang untuk berbagai tugas keamanan informasi seperti pengujian penetrasi, riset keamanan, forensik komputer, dan rekayasa balik.

### Aplikasi Web

- DVGA  
*Damn Vulnerable GraphQL Application* (DVGA) adalah aplikasi web yang dirancang untuk memiliki kerentanan dan dimaksudkan untuk menjadi alat pembelajaran bagi pengembang dan profesional IT. DVGA merupakan implementasi dari GraphQL, sebuah bahasa *query* untuk API yang memiliki fitur untuk meminta data khusus dan mengumpulkan banyak sumber data. Aplikasi web ini menyediakan lingkungan yang aman dan terkendali di mana pengembang dapat mempelajari dan mempraktekkan teknik penyerangan pada aplikasi GraphQL.

### Attack Tools

- GraphW00f  
GraphW00F adalah alat yang digunakan untuk melakukan *fingerprinting* pada *endpoint* GraphQL. Tujuan utama Graphw00F adalah untuk memberikan wawasan tentang keamanan apa saja yang disediakan oleh setiap teknologi secara *default*, dari aktif atau tidak. *Query* yang dirancang khusus memudahkan *fingerprinting* pada mesin *backend* dan membedakan antara berbagai implementasi GraphQL karena berbagai implementasi server GraphQL merespons *query*, mutasi, dan langganan secara unik.
- Altair  
Altair adalah alat yang berfungsi sebagai *Integrated Development Environment* (IDE) untuk GraphQL dan dirancang untuk membantu *user* untuk berinteraksi dengan server GraphQL dengan menyediakan berbagai fitur yang membantu dalam proses *debugging* dan implementasi *query* GraphQL. Beberapa fitur utama Altair termasuk kemampuan untuk membuat dan beralih antara berbagai lingkungan kerja, dan kemampuan untuk mencari dokumen skema yang lebih lanjut.

### Analisa

#### 1. Pengukuran Waktu

Pengukuran waktu berdasarkan eksploitasi berfungsi untuk mengukur, mencatat, dan mengamati setiap ukuran waktu yang memiliki ukuran yang berbeda untuk setiap elemen. Waktu yang diukur untuk setiap elemen didasarkan pada jumlah waktu yang diperlukan untuk melakukan eksploitasi sebelumnya. Jenis jenis pengukuran *time* pada pengujian kali ini dikategorikan menjadi tiga metrik, yaitu:

**Table 5.** Pengukuran Waktu Eksploitasi *Log Injection* Sebelum *Hardening*

No.	Step	Time Eksploitasi 1 (s)		
		Real	User	System
1.	Web Browser Changing	15,22	5,63	2,62
2.	Gecurity Level	0,18	0,11	0,04
3.	<i>Altair Inject Command</i>	22,93	5,40	1,25
	<i>Total Time</i>	\mathbf{38,33}		

- *Real Time*

*Real time* adalah ukuran waktu yang diperlukan secara keseluruhan pada saat mengeksekusi suatu perintah. Dalam metrik *real time* sendiri sudah termasuk waktu yang dihitung pada *user time* dan *system time*. Pencatatan metrik *real time* dimulai dari eksekusi hingga program telah selesai digunakan ataupun dihentikan.

- *User Time*

*User time* mengacu pada jumlah waktu yang dihabiskan oleh CPU untuk menjalankan suatu perintah atau instruksi dalam mode *user*, yang dimana waktu yang dihabiskan untuk menjalankan tugas pengguna daripada tugas-tugas *system*. *User time* adalah jumlah waktu CPU yang sebenarnya digunakan selama menjalankan proses.

- *System Time*

*System time* adalah ukuran waktu yang dihabiskan oleh CPU untuk menjalankan suatu proses dari program yang sedang dilakukan di kernel. Pada pengukuran waktu ini akan ditampilkan tabel yang berisi pengukuran *time* dari setiap langkah langkah pada setiap eksploitasi.

- Metrik *time command injection*
- Metrik *time log injection*
- Metrik *time spoof injection*

Pengukuran waktu yang dilakukan berdasarkan eksploitasi *command injection*, *log injection* dan *spoof injection*. Dengan tujuan untuk mendapatkan waktu yang telah dihabiskan ketika pengujian. Waktu yang didapatkan sebanyak tiga metrik yang berbeda beda yaitu *real time*, *user time*, dan *system time*. Data waktu pada pengukuran ini terbagi menjadi dua data *time* yaitu sebelum *hardening* dan setelah *hardening*. Berikut merupakan tabel - tabel yang berisi data hasil dari pengukuran waktu sebelum *hardening* dan setelah *hardening*.

Dari pengukuran waktu eksploitasi *command injection* pada tabel 3 sebelum *hardening* telah didapatkan jumlah perhitungan waktu untuk proses program sebagai berikut: *Real time*: 60,15s Dari pengukuran waktu eksploitasi *command injection* pada tabel 4 setelah *hardening* telah didapatkan jumlah perhitungan waktu untuk proses program sebagai berikut: *Real time*: 53,52s Dari pengukuran waktu eksploitasi *log injection* pada tabel 8 sebelum *hardening* telah didapatkan jumlah perhitungan waktu untuk proses program sebagai berikut: *Real time*: 38,33s Dari pengukuran waktu eksploitasi *log injection* pada tabel 6 setelah *hardening* telah didapatkan jumlah perhitungan waktu untuk proses program sebagai berikut: *Real time* : 38,19s Dari pengukuran waktu eksploitasi *spoof injection* sebelum *hardening* pada tabel 7 telah didapatkan jumlah perhitungan waktu untuk proses program sebagai berikut: *Real time*: 32,63s. Dari pengukuran waktu eksploitasi *spoof injection* setelah *hardening* pada tabel 8 telah didapatkan jumlah perhitungan waktu untuk proses program sebagai berikut *Real time*: 40,85s.

**Table 6.** Pengukuran Waktu Eksploitasi *Log Injection* Setelah *Hardening*

No.	Step	Time Eksploitasi 1 (s)		
		Real	User	System
1.	Web Browser Changing	15,60	5,73	3,72
2.	Security Level	0,24	0,10	0,06
3.	<i>Altair Inject Command</i>	22,35	4,06	0,96
Total Time		38,19		

**Table 7.** Pengukuran Waktu Eksploitasi *Spoof Injection* Sebelum *Hardening*

No.	Step	Time Eksploitasi 1 (s)		
		Real	User	System
1.	Web Browser Changing Security Level	11,27	5,12	2,29
2.	Graphw00f web scanning	0,23	0,12	0,04
3.	<i>Altair Inject Command</i>	21,13	3,87	0,88
TotalTime		32,63		

**Table 8.** Pengukuran Waktu Eksploitasi *Spoof Injection* Setelah *Hardening*

No.	Step	Time Eksploitasi 1 (s)		
		Real	User	System
1.	Web Browser Changing	22,36	7,01	2,85
2.	Security Level	0,14	0,10	0,03
3.	<i>Altair Inject Command</i>	18,35	3,83	0,95
Total Time		40,85		

**Penyusunan Attack Tree**

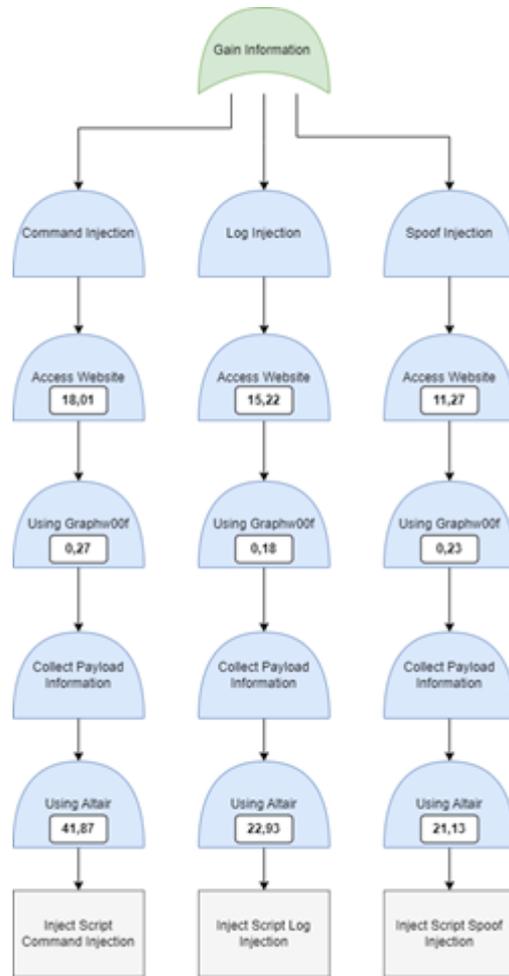
*Attack tree* adalah model grafis yang digunakan untuk mengevaluasi potensi ancaman dan risiko yang terlibat dalam suatu sistem keamanan. Dalam model ini menunjukkan berbagai cara yang dapat digunakan oleh penyerang untuk mencapai suatu tujuan tertentu, yang diwakili oleh *root node* di bagian atas pohon. Setiap cabang dari pohon menunjukkan langkah-langkah atau tujuan yang harus dicapai untuk mencapai tujuan utama. *Attack tree* yang ada di dalam penelitian ini dibuat berdasarkan referensi *attack tree* pada paper yang sudah dikembangkan oleh Terrance R Ingoldsby dengan judul “*Attack Tree-based Threat Risk Analysis*” [13]. *Attack tree* yang dibuat pada penelitian ini akan dibuat beberapa *node* yang mempunyai fungsi berbeda beda sebagai berikut:



- *Or* = Tujuan dari *node* ini adalah untuk menggambarkan pengujian pada sebuah eksploitasi dan dapat digambarkan sebagai variasi dalam langkah proses pengujian.



- *And* = Tujuan dari *node* ini adalah untuk menggambarkan sebuah rangkaian proses pada penyerangan yang dimana sebuah proses atau lebih harus terjadi secara bersamaan dengan



**Gambar 2.** Hasil Gabungan *Attack Tree* Eksploitasi *Injection* Pada GraphQL Sebelum *Hardening*

tujuan untuk mencapai sebuah tujuan utama atau hasil yang diinginkan oleh penyerang.

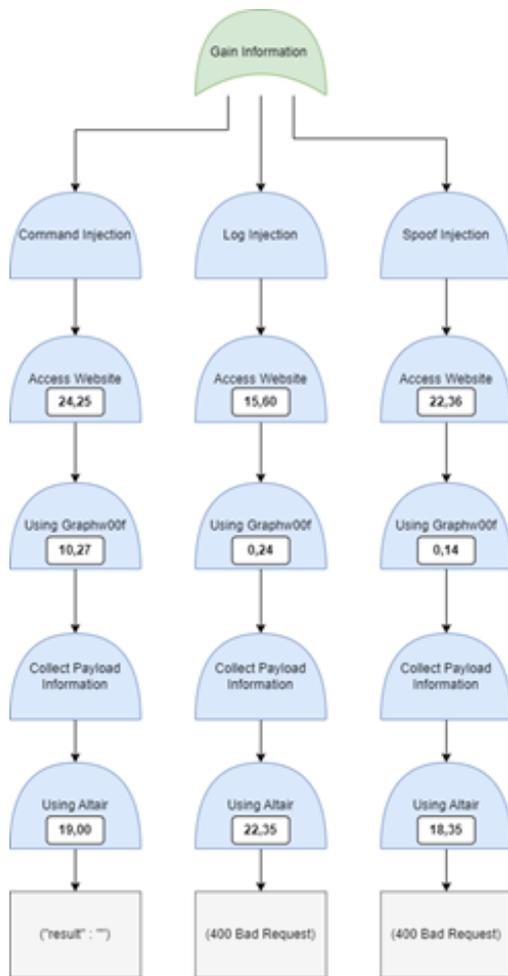
- Sub-goal = Tujuan dari *node* ini adalah untuk menggambarkan sebuah tujuan atau akhir dari langkah langkah serangan.

**Attack Tree Eksploitasi Injection Pada GraphQL**

*Attack tree* yang telah dibuat secara terpisah sebelumnya sesuai dengan serangannya masing masing akan ditampilkan menjadi satu diagram. Tujuan dari menyatukan diagram tersebut menjadi satu adalah untuk melihat secara keseluruhan dari hasil penelitian yang sudah dilaksanakan. Berikut adalah diagram yang telah disatukan dari gabungan *attack tree* eksploitasi *injection* pada GraphQL:

Pada gambar 2 menjelaskan sebuah diagram *attack tree* berdasarkan setiap eksploitasinya. Seluruh serangan diatas bertujuan untuk mendapatkan informasi secara ilegal. Pada diagram ini mempunyai tiga serangan yaitu *Command Injection*, *Log Injection*, *Spoof Injection* yang dimana setiap serangan memiliki langkah langkah sebelum *hardening*.

Pada gambar 3 menjelaskan sebuah diagram *attack tree* berdasarkan setiap eksploitasinya. Seluruh serangan diatas bertujuan untuk



**Gambar 3.** Hasil Gabungan *Attack Tree* Eksploitasi *Injection* Pada GraphQL Setelah Hardenig

mendapatkan informasi secara ilegal. Pada diagram ini mempunyai tiga serangan yaitu *Command Injection*, *Log Injection*, *Spoof Injection* yang dimana setiap serangan memiliki langkah langkah setelah *hardening*. Setiap serangan memiliki hasil yang berbeda dimana *Command Injection* mendapatkan hasil ("result" : ""). Untuk *Log Injection* dan *Spoof Injection* mendapatkan pesan yang sama yaitu (400 Bad Request).

## Kesimpulan

Metrik *time* dapat digunakan sebagai pengkategorian berbagai macam *attack tree*. Pada hasil penelitian ini, kategori eksploitasi teknik injeksi yang membutuhkan waktu paling efisien sebelum *hardening* adalah *attack tree spoof injection* dengan total waktu 32,63 s dengan urutan kedua paling efisien yaitu *attack tree log injection* dengan total waktu 38,33 s. Dan teknik injeksi yang terakhir yaitu *attack tree command*

*injection* dengan total waktu 60,15 s. Waktu tersingkat dalam proses eksploitasi GraphQL mode *hardening* terjadi perubahan waktu yaitu dimana *attack tree log injection* berada di urutan pertama yang tercepat dengan jumlah waktu 38,19s. Dan pada urutan kedua yaitu *attack tree spoof injection* dengan jumlah waktu 40,85 s. Pada urutan terakhir yaitu *attack tree command injection* dengan jumlah waktu 53,52 s. Sehingga dapat disimpulkan bahwa serangan injeksi yang paling efektif sebelum *hardening* adalah *spoof injection* sedangkan serangan injeksi yang efektif setelah *hardening* adalah *log injection*, dimana dari serangan injeksi tersebut kedepannya dapat ditemukan solusi keamanan yang paling efektif.

## Daftar Pustaka

1. Brito G, Valente MT. REST vs GraphQL: A Controlled Experiment; 2020. Accessed: 2024-09-05. <http://arxiv.org/abs/2003.04761>.
2. Serbout S, Di Lauro F, Pautasso C. Web APIs Structures and Data Models Analysis;. Accessed: 2024-09-05. <https://bigqueryconnection.googleapis>.
3. Belhadi A, Zhang M, Arcuri A. Evolutionarybased Automated Testing for GraphQL APIs. In: GECCO 2022 Companion - Proceedings of the 2022 Genetic and Evolutionary Computation Conference. Association for Computing Machinery, Inc.; 2022. p. 778-81.
4. Rusdi MI, Prasti D. Penetration Testing Pada Jaringan Wifi Menggunakan Kali Linux; 2019.
5. Shaukat K, Luo S, Chen S, Liu D. Cyber Threat Detection Using Machine Learning Techniques: A Performance Evaluation Perspective. In: Proceedings of ICCWS 2020 - 1st Annual International Conference on Cyber Warfare and Security. IEEE; 2020. .
6. Johny JHB, Nordin WAFB, Lahapi NMB, Leau YB. SQL Injection Prevention in Web Application: A Review. In: Communications in Computer and Information Science. Springer; 2021. p. 568-85.
7. Wiegel W, Keller J, Biß K, Schmidt T. Development of a GraphQL-based API for querying security advisories for Common Security Advisory Framework (CSAF); 2023.
8. Pan Z, Chen Y, Chen Y, Shen Y, Li Y. LogInjector: Detecting Web Application Log Injection Vulnerabilities. Applied Sciences (Switzerland). 2022 Aug;12(15).
9. Agarwal N. Content Spoofing via Compounded SQL Injection; 2017. Accessed: 2024-09-05. <https://www.researchgate.net/publication/32109687>.
10. Kohnfelder L, Heymann E, Miller BP. Introduction to Software Security Chapter 3.8.2: Command Injections; 2018.
11. Irawan AR, Widjajarto A, Fathinuddin M. Implementasi dan Analisis Attack Tree pada Aplikasi DVWA Berdasar Metrik Time dan Probability; 2023.
12. Lallie HS, Debattista K, Bal J. A Review of Attack Graph and Attack Tree Visual Syntax in Cyber Security; 2019. Accessed: 2024-09-05. <http://wrap.warwick.ac.uk/131160>.
13. Ingoldsby TR. Attack Tree-based Threat Risk Analysis; 2009. Accessed: 2024-09-05. <http://www.amenaza.com>.