LAPORAN TUGAS KEAMANAN SISTEM DAN SIBER

Implementasi *Distributed Denial of Service (DDoS)* pada *Web Server* Python Berbasis Flask

Disusun oleh

Fatimah Tuzahra 24/550163/NPA/19970

Isna Nur Amalia 24/550106/NPA/19963

Rahmad Ramadhan 22/494516/PA/21278

Departemen Ilmu Komputer Dan Elektronika

Fakultas Matematika Dan Ilmu Pengetahuan Alam

Universitas Gadjah Mada

2024

I. Pendahuluan

Dalam era digitalisasi, aplikasi berbasis *web* menjadi peranan yang sangat penting dalam menjalankan berbagai bisnis. Oleh karena itu, pengembangan *web server* menjadi suatu hal yang penting untuk memastikan layanan berbasis *web* dapat berjalan dengan baik. Namun disisi lain, keamanan sistem menjadi tantangan utama dalam lingkungan internet yang semakin kompleks. Salah satu ancaman yang sering muncul yaitu serangan *Denial of Service* (DoS), dimana serangan ini dirancang untuk membuat sebuah layanan atau situs *web* yang tidak dapat diakses oleh pengguna dengan cara menyerang *server* dengan intensitas yang sangat tinggi, sehingga dapat menyebabkan *overload* pada sistem sehingga membuat situs *web* tersebut tumbang.

Tujuan dari eksperimen ini kita dapat memahami bagaimana serangan *Distributed* DoS (DDoS) mempengaruhi performa *web server*, termasuk dampaknya terhadap waktu respon dan ketersediaan layanan. Hasil dari eksperimen ini dapat membantu mengetahui seberapa besar gangguan yang dapat terjadi akibat serangan dari DDoS, serta mengetahui langkah mana yang efektif dalam melindungi *server* dari serangan DDoS dan dapat mengevaluasi kesiapan dan kekuatan *server* terhadap serangan DDoS. Experimen ini memberikan kita pemahaman yang lebih baik mengenai efek dari serangan DDoS dan efektivitas metode mitigasi yang diterapkan, sehingga kita dapat lebih siap dalam melindungi layanan web dari ancaman siber.

II. Metodologi

A. Inisiasi Web Server dengan Flask

Dalam inisiasi *web server*, digunakan skrip python `app.py` yang mengutilisasi *framework* Flask. *Web server* ini dijalankan di *localhost* pada *port* 5000 dengan beberapa aset telah dibuat sebelumnya. Skrip *web server* akan memiliki *endpoint* sederhana yang akan merespon dengan pesan tertentu saat dijalankan *request* padanya..

B. Implementasi DoS dengan *Threading*

Dalam simulasi serangan DoS, digunakan skrip python `dos.py` yang memanfaatkan modul *threading* yang akan mengirimkan *request* berulang kali ke URL target. Skrip memiliki parameter jumlah *thread* yang dapat diatur sesuai dengan kebutuhan pengguna. Dalam eksperimen ini, jumlah *thread* per skrip adalah 500.

C. Implementasi *Multithreading*

Untuk mensimulasikan serangan DoS dengan lebih nyata, *multithreading* diimplementasikan untuk menjalankan beberapa skrip `dos.py` sekaligus menggunakan skrip `calldos.py`. Skrip `dos.py` akan diduplikat menjadi beberapa file serupa –dalam eksperimen ini, menjadi `dos1.py`, `dos2.py`, dst. hingga menjadi sejumlah 5 skrip. Skrip-skrip tersebut kemudian dipanggil secara bersamaan secara paralel oleh `calldos.py` untuk menyerang *web server* secara bersamaan dan memperbesar intensitas dan kemungkinan *web server* tumbang.

D. *Monitoring* Performa *Web Server*

Performa web server dipantau oleh skrip `monitor.py`. Skrip akan melakukan *request* setiap beberapa detik ke *web server*, lalu akan memberikan kembalian berupa waktu respon yang diperlukan *web server* untuk membalas *request* tersebut. Skrip juga akan memberikan pesan tertentu apabila *web server* sedang berstatus *down* dan tidak dapat menjawab *request*.

III. Hasil dan Analisis

Analisis dilakukan dengan menjalankan `app.py` terlebih dahulu, diikuti dengan `monitor.py`. Kemudian, `calldos.py` akan dipanggil untuk menjalankan beberapa `dos.py` secara bertahap (dari 1 sampai 5 sekaligus) sebagai simulator serangan DDoS. Waktu kembalian *request* dari `monitor.py` akan dicatat dari sebelum serangan dilakukan, saat serangan dilakukan, dan setelah serangan dihentikan.

Jumlah `dos.py`	Rerata waktu respon (s)			Request `monitor.py` sebelum *server down*
	sebelum DDoS	saat DDoS	setelah DDoS	
0	2.041	-	-	tidak down
1	2.038	2.081	2.054	12 (*)
2	2.033	2.654	2.042	8
3	2.043	3.748	2.048	7
4	2.039	- (**)	2.047	0
5	2.044	- (**)	2.052	0

Catatan:

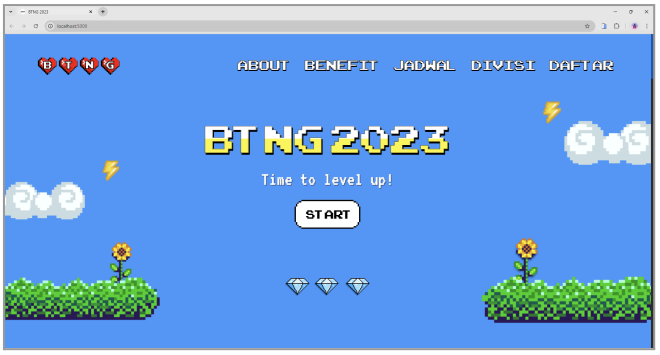
- *: *web server* kembali merespon setelah 2 *request* dengan waktu respon 5.081 detik

- **: *web server* tidak dapat memberikan respon karena langsung *down*

Berdasarkan hasil pengujian, didapatkan analisis sebagai berikut:

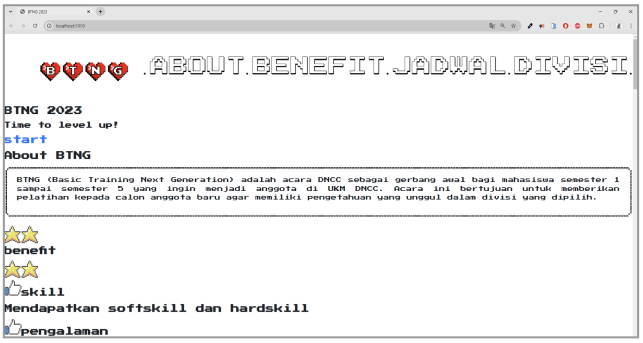
1. Sebelum serangan, *web server* memiliki waktu respon yang cukup stabil, sekitar 2.04 detik, menunjukkan bahwa *server* berjalan dengan baik dan dapat menangani *request* dari monitor.py tanpa masalah.
2. Saat serangan dilakukan, *web server* menunjukkan respon yang berbeda namun linear, bergantung jumlah dos.py yang dijalankan.
    1) Ketika 1 skrip dijalankan, terdapat sedikit peningkatan pada waktu respon rata-rata (dari 2.038 detik menjadi 2.081 detik), tetapi *server* masih tetap berfungsi dan tidak mengalami *downtime* sampai *request* ke-12.
    2) Ketika 2 skrip dijalankan, peningkatan waktu respon lebih signifikan (menjadi 2.654 detik), namun *server* tetap dapat bertahan tanpa mengalami *downtime*, sampai *request* ke-8.
    3) Ketika 3 skrip dijalankan, waktu respon meningkat drastis menjadi 3.748 detik, dan *server* hanya mampu berada dalam *state responding* sampai *request* ke-7.
    4) Ketika 4 dan 5 skrip dijalankan secara bersamaan, server langsung mengalami *downtime*, dilihat dari tidak adanya respon yang diterima oleh monitor.py, yang menandakan server *overload* dan tidak mampu menerima *request* lebih lanjut.
3. Setelah serangan dihentikan, waktu respon *server* kembali normal menjadi sekitar 2.04 detik, menunjukkan bahwa *server* dapat pulih dengan baik setelah beban serangan dihilangkan.

Untuk memberikan gambaran yang lebih jelas mengenai dampak serangan DDoS pada web server, berikut adalah beberapa screenshot yang menunjukkan kondisi web server sebelum dan setelah serangan dilakukan:

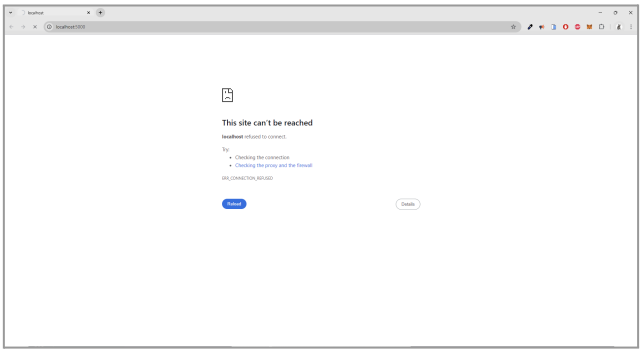


Gambar 1. Kondisi Web Server Sebelum Serangan DDoS

Gambar 1 menunjukkan tampilan web server di browser ketika kondisi normal, sebelum serangan DDoS dijalankan. Web server dapat merespon permintaan dengan cepat dan menampilkan konten tanpa ada masalah. Hal ini sesuai dengan hasil pengamatan yang menunjukkan rata-rata waktu respon sekitar 2.04 detik.



Gambar 2. Kondisi Web Server Sebelum Serangan DDoS



Gambar 3. Kondisi Web Server Sebelum Serangan DDoS

Pada gambar 1 dan gambar 2 menunjukkan kondisi web server ketika serangan DDoS sedang berlangsung dengan beberapa skrip dos.py dijalankan secara bersamaan menggunakan calldos.py. Tampilan ini menunjukkan bahwa server tidak dapat diakses atau mengalami *downtime*, yang mana sesuai dengan hasil analisis yang menunjukkan bahwa server mengalami *overload* dan tidak mampu menangani permintaan lebih lanjut.

Gambar-gambar tersebut membantu untuk memvisualisasikan efek nyata dari serangan DDoS terhadap ketersediaan dan performa web server. Dengan demikian, terlihat jelas bahwa intensitas serangan yang tinggi akan menyebabkan server tidak dapat berfungsi dengan baik, mengakibatkan *downtime* dan penurunan kualitas layanan yang signifikan.

IV. Kesimpulan dan Saran

Eksperimen ini menunjukkan bahwa serangan DDoS dapat secara signifikan mempengaruhi ketersediaan dan kinerja *web server*. Ketika intensitas serangan meningkat, waktu respon *server* melonjak drastis, dan pada titik tertentu, *server* tidak mampu menangani *request* dan mengalami *downtime*. Akan tetapi, *server* dapat bekerja dengan normal kembali ketika serangan dihentikan.

Eksperimen ini dapat diperluas menggunakan skenario serangan yang lebih kompleks, termasuk menggunakan jaringan terdistribusi (botnet) untuk mensimulasikan serangan DDoS yang lebih realistis. Selain itu, *server* yang lebih *scalable* yang mengutilisasi *load balancer* atau *microservices* dapat diuji untuk melihat bagaimana performa *server* ketika menghadapi serangan yang besar.

V. Lampiran

Seluruh source code yang digunakan dalam eksperimen ini dapat diakses pada tautan berikut: s.id/JPNYj