Simulador de Ataque DDoS com Mecanismos de Mitigação

Ryan Pimentel[1], **Leonam Sousa**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brasil

ryanpimentel52@gmail.com, leonam666lol@gmail.com

Abstract. *This work presents the development of a DDoS (Distributed Denial of Service) attack simulator with mitigation mechanisms for educational purposes in Operating Systems discipline. The system implements a complete web environment with real-time monitoring, rate limiting, and IP blocking mechanisms. The results demonstrate the effectiveness of implemented protections, achieving a 99% blocking rate during simulated attacks while maintaining system stability and monitoring capabilities.*

Resumo. *Este trabalho apresenta o desenvolvimento de um simulador de ataques DDoS (Distributed Denial of Service) com mecanismos de mitigação para fins educacionais na disciplina de Sistemas Operacionais. O sistema implementa um ambiente web completo com monitoramento em tempo real, limitação de taxa de requisições e mecanismos de bloqueio de IP. Os resultados demonstram a eficácia das proteções implementadas, alcançando uma taxa de bloqueio de 99% durante ataques simulados, mantendo a estabilidade do sistema e capacidades de monitoramento.*

1. Introdução

Os ataques DDoS representam uma das principais ameaças à disponibilidade de serviços web na atualidade. Estes ataques visam tornar um recurso indisponível aos seus usuários legítimos, sobrecarregando o sistema com um volume excessivo de requisições simultâneas [1].

Este projeto tem como objetivo desenvolver um simulador educacional que permita compreender tanto os mecanismos de ataque DDoS quanto as técnicas de mitigação empregadas para defendê-los. O sistema foi implementado utilizando tecnologias web modernas, incluindo Node.js, Express, Socket.io e Chart.js para visualização em tempo real.

A relevância deste trabalho está na necessidade de formar profissionais capazes de identificar, compreender e mitigar ataques DDoS, proporcionando um ambiente controlado e seguro para experimentação e aprendizado.

2. Fundamentação Teórica

2.1. Ataques DDoS

Os ataques de Negação de Serviço Distribuída (DDoS) consistem na coordenação de múltiplos sistemas para sobrecarregar um alvo específico. Diferentemente dos ataques

DoS tradicionais, que partem de uma única origem, os ataques DDoS utilizam redes de computadores comprometidos (botnets) para amplificar o impacto [2].

Os principais tipos de ataques DDoS incluem:

- **Ataques volumétricos**: Consomem largura de banda disponível
- **Ataques de protocolo**: Exploram vulnerabilidades em protocolos de rede
- **Ataques de aplicação**: Direcionados a vulnerabilidades específicas de aplicações

2.2. Mecanismos de Mitigação

As técnicas de mitigação de ataques DDoS podem ser categorizadas em:

2.2.1. Rate Limiting

Técnica que limita o número de requisições por IP em um determinado período de tempo, prevenindo o abuso de recursos do servidor.

2.2.2. IP Blocking

Bloqueio temporário ou permanente de endereços IP que apresentam comportamento malicioso, baseado em padrões de tráfego suspeitos.

2.2.3. Monitoramento em Tempo Real

Coleta e análise contínua de métricas do sistema (CPU, memória, conexões ativas) para detecção precoce de ataques.

3. Metodologia

O desenvolvimento do simulador seguiu uma abordagem modular, dividindo o sistema em componentes específicos:

3.1. Arquitetura do Sistema

O sistema foi estruturado em quatro módulos principais:

1. **Servidor Web**: Implementado em Node.js com Express, responsável por servir a aplicação e processar requisições HTTP
2. **Sistema de Monitoramento**: Coleta métricas de sistema em tempo real (CPU, memória, conexões)
3. **Mecanismo de Mitigação**: Implementa rate limiting e bloqueio de IPs suspeitos
4. **Interface de Visualização**: Dashboard web com gráficos em tempo real usando Chart.js

3.2. Implementação dos Componentes

3.2.1. Servidor Principal

O servidor principal foi implementado utilizando Express.js com suporte a WebSockets através do Socket.io:

```
const express = require("express");
const http = require("http");
const socketIo = require("socket.io");

const app = express();
const server = http.createServer(app);
const io = socketIo(server);

let mitigacaoAtiva = false;
const limites = new Map();
const bloqueados = new Map();
```

Listing 1. Estrutura básica do servidor

3.2.2. Sistema de Rate Limiting

O controle de taxa implementado monitora requisições por IP:

```
function firewall(req, res, next) {
    const ip = req.ip;

    if (!mitigacaoAtiva) return next();

    if (bloqueados.has(ip)) {
        if (Date.now() - bloqueados.get(ip) > BAN_TIME) {
            bloqueados.delete(ip);
        } else {
            return res.status(429).send("IP bloqueado");
        }
    }

    // Verifica  o de limite de requisi  es
    const now = Date.now();
    const timestamps = limites.get(ip) || [];
    timestamps.push(now);

    // Remove timestamps antigos (> 1 segundo)
    while (timestamps.length > 0 &&
           now - timestamps[0] > 1000) {
        timestamps.shift();
    }

    if (timestamps.length > LIMITE_REQ) {
```

```
        bloqueados.set(ip, Date.now());
        return res.status(429).send("Muitas requisi  es");
    }

    limites.set(ip, timestamps);
    next();
}
```

Listing 2. Implementação do firewall

3.2.3. Monitoramento de Sistema

O sistema coleta métricas de CPU e memória utilizando módulos nativos do Node.js:

```
function calcularUsoCpu() {
    const cpusAgora = os.cpus();
    let usoTotal = 0;

    cpusAgora.forEach((cpu, i) => {
        const anterior = ultimaMedidaCpu[i];
        const totalAnterior = Object.values(anterior.times)
            .reduce((acc, tv) => acc + tv, 0);
        const totalAgora = Object.values(cpu.times)
            .reduce((acc, tv) => acc + tv, 0);

        const totalDelta = totalAgora - totalAnterior;
        const idleDelta = cpu.times.idle - anterior.times.idle;

        const uso = totalDelta > 0 ?
            ((totalDelta - idleDelta) / totalDelta) * 100 : 0;
        usoTotal += uso;
    });

    ultimaMedidaCpu = cpusAgora;
    return usoTotal / cpusAgora.length;
}
```

Listing 3. Coleta de métricas do sistema

3.3. Metodologia de Testes

Os testes foram realizados utilizando ApacheBench (ab) para simulação de carga, comparando o comportamento do sistema com e sem mecanismos de mitigação ativos.

Parâmetros dos testes:

- Número de requisições: 2000
- Concorrência: 100 conexões simultâneas
- URL alvo: http://localhost:3000/ataque
- Limite de requisições: 20 req/s por IP
- Tempo de bloqueio: 10 segundos

4. Resultados e Discussão

Os testes realizados demonstraram a eficácia dos mecanismos de mitigação implementados.

4.1. Cenário sem Mitigação

No primeiro teste, com os mecanismos de mitigação desativados, o sistema processou todas as 2000 requisições sem falhas:

Tabela 1. Resultados sem mitigação ativa

Métrica	Valor
Requisições totais	2000
Requisições falharam	0
Taxa de sucesso	100%
Tempo total	1.267 segundos
Requisições/segundo	1578.99
Uso de CPU final	3.24%
Uso de memória	21.53 MB
IPs bloqueados	0

4.2. Cenário com Mitigação

Com os mecanismos de proteção ativos, o sistema bloqueou efetivamente 99% das requisições:

Tabela 2. Resultados com mitigação ativa

Métrica	Valor
Requisições totais	2000
Requisições falharam	1980
Taxa de bloqueio	99.00%
Tempo total	0.850 segundos
Requisições/segundo	2351.87
Uso de CPU final	20.52%
Uso de memória	21.03 MB
IPs bloqueados	1

4.3. Análise dos Resultados

Os resultados evidenciam que:

1. O sistema de mitigação foi altamente efetivo, bloqueando 99% das requisições maliciosas
2. O tempo de resposta médio foi reduzido de 63.332ms para 42.519ms com mitigação ativa
3. O uso de CPU aumentou de 3.24% para 20.52%, indicando o overhead dos mecanismos de proteção

4. O uso de memória permaneceu estável em ambos os cenários
5. Um único IP foi identificado e bloqueado durante o ataque

O aumento no uso de CPU demonstra que, embora os mecanismos de mitigação sejam eficazes, eles introduzem overhead computacional. No entanto, este custo é justificado pela proteção oferecida contra ataques DDoS.

4.4. Interface de Monitoramento

A interface web desenvolvida permite visualização em tempo real das métricas do sistema através de gráficos dinâmicos que se atualizam a cada segundo. Os gráficos incluem:

- Uso de CPU em percentual
- Consumo de memória RAM
- Número de conexões ativas

Esta funcionalidade é fundamental para a detecção precoce de ataques e monitoramento da eficácia das medidas de mitigação.

5. Limitações e Trabalhos Futuros

5.1. Limitações Identificadas

Durante o desenvolvimento e testes, algumas limitações foram identificadas:

1. O sistema atual não diferencia entre tráfego legítimo e malicioso baseado em padrões comportamentais
2. A mitigação é aplicada uniformemente, sem considerar diferentes tipos de requisições
3. Não há implementação de técnicas avançadas como CAPTCHA ou verificação de JavaScript
4. O sistema não persiste dados de bloqueio entre reinicializações

5.2. Propostas para Trabalhos Futuros

Para aprimoramento do sistema, propõem-se:

- Implementação de algoritmos de machine learning para detecção de padrões de ataque
- Desenvolvimento de um sistema de whitelist para IPs confiáveis
- Integração com serviços externos de inteligência de ameaças
- Implementação de técnicas de mitigação em camadas de rede (Layer 3/4)
- Adição de métricas de rede (latência, throughput) ao sistema de monitoramento

6. Conclusão

Este trabalho apresentou o desenvolvimento de um simulador completo de ataques DDoS com mecanismos de mitigação integrados. O sistema demonstrou alta eficácia na proteção contra ataques simulados, bloqueando 99% das requisições maliciosas durante os testes.

A implementação utilizando tecnologias web modernas permitiu criar uma solução educacional robusta e visualmente intuitiva, facilitando o entendimento dos conceitos relacionados a segurança de sistemas e ataques DDoS.

Os resultados obtidos confirmam que técnicas simples como rate limiting e bloqueio de IP podem ser altamente efetivas contra ataques DDoS básicos, embora introduzam overhead computacional que deve ser considerado em implementações em produção.

O sistema desenvolvido serve como uma ferramenta valiosa para o ensino de segurança computacional, proporcionando experiência prática com conceitos teóricos fundamentais da área.

Referências

[1] Zargar, S. T., Joshi, J., and Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (ddos) attacks. *IEEE Communications Surveys & Tutorials*, 15(4):2046–2069.

[2] Specht, S. M. and Lee, R. B. (2004). Distributed denial of service: Taxonomies of attacks, tools, and countermeasures. In *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems*, pages 543–550.

[3] Tilkov, S. and Vinoski, S. (2010). Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83.

[4] Williamson, C. L. (2001). Internet traffic measurement. *IEEE Internet Computing*, 5(6):70–74.