Zumbis, teias e negação de serviço: uma introdução a DoS

Por Gabriel Cruz

Descrição

Ataques Oh, ataques De negação? Não, de serviço Serviço? Isso é nada mais que um DEServiço Um DoServiço

Ataques de negação de serviço, ou Denial of Service (DoS) em inglês, estão cada vez mais comuns na internet. Seja pelo aspecto ubíquo que a internet tomou nos últimos anos, seja pela facilitação de implementar ataques desse tipo, ou mesmo a dificuldade de evitá-los, é visível que hoje mais do que nunca ataques de negação de serviço são uma ameaça crescente.

Então bora derrubar o site do seu primo... da padaria!

Duração: 4h

Currículo do Ganesh: Ganesh é um grupo de alunos vinculado ao ICMC-USP sem fins lucrativos focado no estudo, discussão e desenvolvimento de técnicas para a segurança de sistemas computacionais e redes de computadores. O grupo visa à aplicação direta dos conhecimentos adquiridos nos cursos de graduação do ICMC e da sua integração com as mais diversas tecnologias disponíveis no mercado. Por meio do estudo e do desenvolvimento de técnicas e algoritmos voltados para a segurança de sistemas computacionais e redes de computadores, o grupo visa o aprendizado, difusão e troca de conhecimentos e experiências entre os membros. Além disto, também visa a integração e interação com outros grupos externos a Universidade, bem como a difusão de suas experiências e conhecimentos junto a comunidade externa.

Introdução e histórico

DoS: Denial of Service

Um ataque de negação de serviço ou DoS (Denial of Service) tem como objetivo impedir o acesso a um determinado recurso, comprometendo assim sua disponibilidade. Entenda recurso aqui como qualquer coisa que seja útil para alguém. Ou seja, recursos podem ser arquivos, páginas web, estatísticas, números, etc.

A maneira mais comum de se realizar um ataque de negação de serviço é inundando ("floodando") a rede de um servidor cujo serviço queremos negar. Pense no site do enem logo que saem os resultados, é muito comum ele ficar instável (você tem dificuldade para acessá-lo). Isso acontece porque há muitas pessoas acessando o site ao mesmo tempo, e a rede ou o servidor não têm capacidade para lidar com tantos acessos simultâneos. Os ataques DoS mais comuns realizam acessos em massa justamente para tentar desestabilizar seus alvos.

Em geral, então, negação de serviço é feita inundando redes com requisições, mas nem sempre esse é o caso! Como veremos a seguir, esses ataques podem ser realizados de maneiras muito mais simples e com resultados tão eficazes quanto.

Historinhas

1.Estônia (2007): A Estônia é um país que, desde muito cedo, investiu na digitalização tanto de seus sistemas governamentais. Lá, mesmo as eleições são feitas pela internet. A suspeita é de que o ataque tenha sido motivado por um atrito político entre Rússia e Estônia a respeito da realocação de um importante monumento histórico da Segunda Guerra Mundial, o "Soldado de Bronze de Tallinn".

2.Github China (2015): Esse ataque, também com prováveis causas políticas, durou vários dias e atingiu alguns repositórios específicos do Github -- que continham códigos e ferramentas usadas para burlar censura imposta pelo governo chinês. A ferramenta de busca do Baidu, muito usada pelos usuários chineses, continha um código javascript que fazia com que os browsers desses usuários disparassem torrentes de requisições HTTP para os repositórios-alvo.

3.Mirai botnet (2016): Em outubro de 2016 os servidores de DNS do Dyn foram atacados por uma botnet de dispositivos IoT -- sobretudo câmeras de vigilância com conexão wifi. O malware tomava vantagem de uma vulnerabilidade muito simples desses dispositivos: muitos deles ainda usavam senhas padrão para acesso de administrador. A botnet então fazia com que as máquinas infectadas espalhassem o malware através da internet e, finalmente, que elas atacassem um alvo (no caso, o Dyn DNS Provider).

- Mirai malware (Wikipedia)
- Código-fonte da Mirai

Internet, Web e outras coisinhas

Internet vs Web

Muitas vezes, principalmente no linguajar informal do dia-a-dia, confundimos os conceitos de Internet e Web. No entanto, esses dois conceitos, apesar de extremamente relacionados, são sim diferentes, e aprender as principais diferenças entre eles é fundamental.

Internet é uma **rede de computadores**, ou seja, é a infraestrutura que permite que vários computadores se conectem. Quando falamos de Internet estamos nos referindo a, por exemplo, cabos, endereços IP, protocolos de roteamento, etc.

Web é uma **aplicação distribuída que roda através da Internet**, isto é, é o conjunto de páginas HTML ligadas por hyperlinks (ou apenas links) que são transmitidas através da Internet por meio dos protocolos HTTP/HTTPS.

Então, quando ouvir falar em Web, pense no seu browser, na página *web* do ICMC, e em HTML/CSS/JavaScript. Quando pensar em Internet, lembre-se de tudo que roda em cima dela: Skype/Discord (VoIP), Dispositivos IoT, Wi-fi e até a própria Web!

Protocolos

Se você se sentiu meio perdido quando falamos de protocolos de roteamento, VoIP, IP ou HTTP/HTTPS, fique sossegado, vamos esclarecer essa coisa de protocolo agora.

Protocolos, em especial protocolos de comunicação, são nada mais que **padrões de comunicação**, ou **acordos de comunicação**. Se você já brincou de Walkie Talkie você deve saber bem como funciona um protocolo de comunicação.

O Walkie Talkie usa um canal de comunicação Half-Duplex, ou seja, um canal em que a informação pode correr em qualquer sentido, mas apenas em um sentido de cada vez. Quer dizer que apenas uma pessoa pode falar por vez, se as duas tentam falar, dá ruim (em geral nenhuma consegue transmitir, mas às vezes alguém começa a falar um pouquinho antes e ela "ganha o canal").

Para resolver o problema das pessoas tentando falar ao mesmo tempo no Walkie Talkie foi inventado um protocolo de comunicação: quando alguém termina de falar, diz "câmbio", o que significa que agora sim outros podem começar a falar; quando alguém vai desligar o aparelho, diz "câmbio, desligo"; quando você entende o que o outro disse, diz "copiei, câmbio".

Nas redes de computadores é exatamente a mesma coisa: existem os protocolos (IP, TCP, HTTP, NTP, ICMP, etc.) e cada um deles define um padrão que ele deve seguir. Por exemplo, no protocolo IP sabemos que os primeiros quatro bits a chegar dirão a versão do protocolo (IPv4 ou IPv6), os próximos quatro -- se for IPv4 -- dirão o tamanho total do cabeçalho, e assim por diante.

Três pontos muito importantes: 1. Protocolos funcionam pois há um **acordo** entre os computadores, e eles seguem esse acordo porque, caso contrário, seria impossível de se comunicar!

2. Protocolos são padrões, mas, desde que o que foi especificado seja obedecido, a implementação fica a critério do programador. Por exemplo, é possível implementar as funções do protocolo IP tanto em Assembly quanto em C, e, se o padrão continuar a ser seguido, essas duas implementações, mesmo que completamente diferentes, conseguirão se comunicar perfeitamente!

3. Protocolos em geral trabalham em conjunto, criando *stacks*, ou pilhas, de protocolos. Assim, um pacote de um protocolo pode levar, dentro dele, um pacote de outro protocolo (daí o famoso termo TCP/IP).

RFCs: A maioria dos protocolos de comunicação são definidos pela IETF (Internet Engineering TaskForce) através de documentos chamados RFCs, ou **Requests For Comments**. Esses documentos podem ser facilmente encontrados no site da IETF

- RFC do protocolo IP
- RFC do protocolo TCP

Pacotes

Os protocolos de transmissão de mensagens em geral definem **pacotes**, isso acontece porque a internet em geral é uma rede de *comutação de pacotes*, ou seja, que troca pacotes.

Um pacote é uma unidade de dado bem definida que viaja pela rede que, em geral, carrega parte de um dado maior. Por exemplo, uma imagem de 30MB é grande demais para ser carregada de uma só vez pela internet, isso porque podem existir eventuais gargalos na rede como trechos do caminho que suportem apenas pequenos pacotes. Então, o que fazemos é quebrar essa imagem em vários pacotinhos menores que podem, quando chegam no destino, ser ordenados corretamente, fazendo com que a imagem seja montada novamente.

Obs: Pacotes podem também ser chamados de datagramas, ou, dependendo do nível em que estejamos, de quadros (*frames*). Não vamos nos ater a esses detalhes, mas eles são de fato diferentes em teoria.

Arquitetura Cliente/Servidor

A Web como conhecemos hoje ainda funciona majoritariamente seguindo a arquitetura Cliente/Servidor (apesar de isso estar mudando).

Nessa arquitetura, um cliente demanda algum serviço de um servidor. Um exemplo comum disso são os browsers, que atuam como clientes, pedindo serviços, no caso páginas HTML, para os servidores Web, por exemplo o da Google.

Mas não se engane: um servidor não é necessariamente uma máquina enorme, cheia de cabos e tão complexa que precisa de toda uma equipe para operá-la. Esses servidores existem sim, mas atualmente você pode rodar um servidor web, o Apache por exemplo, na sua própria máquina. Na verdade, você pode rodar vários servidores na sua máquina -- tudo depende de quantos acessos serão feitos nela (como veremos a seguir hehe :).

Um Primeiro Ataque

SYN Flood

Quando sua máquina conversa com outra através da internet ela quase sempre usa o protocolo TCP. Esse protocolo permite que duas máquinas abram uma **conexão**, ou seja, um canal de comunicação preestabelecido. Para que essa conexão seja feita é realizado um procedimento chamado *Three-way handshake*, ou, numa tradução livre, *Aperto de mão em três vias*.

O Three-way handshake é um processo em que, **normalmente**, a máquina A (ou *host*) que deseja se conectar envia um pacote TCP com uma flag *SYN* (de Synchronize) para a máquina B, indicando que ela quer se conectar. Se a máquina B aceitar a conexão, ela então responde com um pacote *SYN/ACK* (Synchronization acknowledgement, ou Confirmação de sincronização). Finalmente, a máquina A precisa dizer que ela recebeu o pacote *SYN/ACK*, então ela envia um terceiro (e último) pacote *ACK*.

Mas repare que esse processo todo ocorre **normalmente**, e nosso primeiro ataque toma vantagem justamente de um comportamento inusitado nessa transação.

Imagine agora que a máquina A manda o *SYN*, mas ela não recebe um *SYN/ACK* como resposta. Isso pode acontecer por diversos fatores: problemas na rede, firewals mal configurados, quedas de energia, etc. É por isso que o protocolo TCP tem os pacotes *ACK*, trata-se de um mecanismo de confirmação de recebimento de mensagens. Nesse caso, a máquina A entenderia que algo de errado aconteceu e ela reenviaria o pacote *SYN*. Da mesma forma, se a máquina B envia o *SYN/ACK* mas recebe um *SYN* novamente de A, ela entende que A não recebeu o seu *SYN/ACK* e reenviou o *SYN*.

O nosso ataque então faz o seguinte: nós enviamos pacotes *SYN* sem parar, não importando qual seja a resposta do alvo. Ele pensará que nós realmente não estamos recebendo suas respostas e manterá várias conexões abertas por um tempo considerável. Isso faz com que a tabela de conexões do alvo fique lotada com as nossas conexões, impedindo-o de abrir conexões com outras máquinas.

Para isso, usaremos uma ferramenta chamada Hping3:

```
sudo hping3 -V -c <quantidade de pacotes> -d <tamanho do pacote> -S -p <porta alvo> -i u<tempo> --flood <ip do alvo>
```

`-V` : Verboso, faz a saída ser mais explícita (opcional) `-c <quantidade>` : Quantidade de pacotes a serem enviados `-d <tamanho do pacote>` : Tamanho de cada pacote a ser enviado `-S` : Usar a flag *SYN* do TCP `-p <porta alvo>` : Porta da máquina destino (em geral não importa, desde que seja uma porta onde há um socket em modo LISTEN) `-i u<tempo>` : tempo entre envios (em milissegundos: u100 == 100 milissegundos entre pacotes) `--flood` : Envia pacotes o mais rápido possível

- RFC sobre SYN flood
- Arigo sobre SYN Flood da Phrack Magazine
- Repositório do Hping3 no Github
- Wiki do Hping3

IP Spoofing

Em geral é muito fácil se defender de um desses ataques, pois basta bloquear quaisquer pacotes advindos do IP do atacante. No entanto, uma prática comum, e que torna o ataque muito mais poderoso, é a de IP Spoofing.

A ideia por trás do IP spoofing é de enviar pacotes com IP de origem diferente do real, fazendo parecer que os pacotes vêm de outras máquinas.

No Hping3, podemos fazer o IP Spoofing especificando mais uma flag: `--rand-source`, que coloca um valor de IP aleatório como origem do pacote, ou `--rand-dest`, que coloca um IP aleatório como destino do pacote.

Um...Segundo...Ataque

Slow Loris

A ideia por trás do Slow Loris é abrir várias conexões TCP com um servidor web e, em cada uma delas, enviar de tempos em tempos uma parte de uma requisição GET. O servidor acha que as conexões estão lentas, mas que são legítimas, então ele as mantêm, impedindo outras conexões de usuários comuns.

Para esse ataque, vamos usar esse código maravilindo. Primeiro clonamos o repositório:

```
$ git clone https://github.com/gkbrk/slowloris
```

Depois entramos no repositório clonado:

```
$ cd slowloris
```

Obs: É possível que você precise especificar um caminho diferente ao `cd`, dependendo da pasta para onde o repositório foi clonado e da pasta onde você está.

Finalmente, basta usar o python para executar o código, passando o IP do alvo (nesse caso, nosso alvo é `10.10.10.10`):

```
$ python3 slowloris.py 10.10.10.10
```

Ou apenas o domínio, caso exista:

```
$ python3 slowloris.py example.com
```

Obs: Existe uma outra versão do Slow Loris que se chama R U Dead Yet, que envia requisições POST ao invés de GET.

ADoS, DDoS e Botnets

Imagine que você quer derrubar um servidor, o da Google por exemplo. Se você fizer um monte de requisições no servidor da Google com a internet de joça de 1Mb/s da sua casa ele não vai cair, ou não devia. Isso porque o link da Google (a largura de banda deles) é gigantescamente maior do que a da sua casa, ou melhor, provavelmente do que toda a sua cidade.

Para derrubar a Google usando ataques de flood, é preciso achar maneiras de controlar mais máquinas que não só a sua. Isso pode ser feito, fundamentalmente, de duas maneiras:

1. Com um **ADoS (Amplified DoS, ou DoS amplificado)**: o ICMP Storm, ataque que veremos a seguir, explora uma vulnerabilidade no protocolo ICMP que nos permite fazer todas as máquinas de uma rede mandarem pacotes para o nosso alvo. Outras formas de fazer ADoS existem, algumas envolvem servidores DNS ou até mesmo o protocolo NTP, mas nos ateremos a esse exemplo
2. Um **DDoS (Distributed DoS, DoS distribuído) através de uma Botnet**: Outra forma de conseguir controle de outras máquinas é criar um malware (*malicious software*, ou software malicioso) como um vírus, que infecte outras máquinas se espalhando pela rede. Depois, as máquinas infectadas, chamadas de **bots**, criam uma rede de *zumbis*, a **botnet**, controlada por um mestre, o **botmaster**. O botmaster pode, daí, mandar os bots atacarem algum site ou servidor, esses ataques em geral são extremamente difíceis de mitigar do ponto de vista do servidor alvo pois não há muitos indícios para diferenciar acessos advindos da botnet de acessos legítimos.

o tercEIRO ATAQUE

ICMP Storm ou Smurf Attack

O ICMP (Internet Control Message Protocol) é o protocolo usado para trocar mensagens sobre a situação da rede. Se isso parece difícil de imaginar, basta pensar no famoso `ping`. O `ping` é um pacote do ICMP para verificar o tempo de ida e volta (RTT: Round Trip Time) de um pacote entre duas máquinas na rede.

O que o ICMP Storm faz é enviar um ping em nome do alvo, ou seja, realizando um spoofing no endereço IP do alvo, no endereço de broadcast da rede (o endereço que redireciona para todas as máquinas da rede). Todas as máquinas da rede responderão para o alvo, que, caso haja máquinas suficientes conectadas, ficará lotado de respostas ICMP.

O comando para realizar isso no `hping3` é:

```
sudo hping3 --icmp --spoof <ip do alvo ("origem")> <ip de broadcast (e.g. 192.168.1.255)>
```

DDoS, Botnets e C&C

É comum ouvimos falar de negação de serviço atrelada à sigla DDoS, e não DoS, isso porque a grande maioria dos ataques de negação de serviço hoje são também **distribuídos**, daí o "D" a mais: Distributed Denial of Service (DDoS).

Como já vimos, uma prática comum para se realizar ataques DDoS é construir malwares, softwares maliciosos, que infectam outros computadores e, em geral, se replicam através da rede para outras máquinas. As máquinas infectadas são comumente chamados de **bots**, e a rede desses bots chama-se **botnet**.

Anonymous

Uma botnet nem sempre é construída de computadores infectados por malware. Alguns ataques DDoS são feitos com bots voluntários, é o caso dos ataques feitos pelos Anonymous.

Uma ferramenta frequente para realizar DDoS coordenado é o LOIC (Low Orbit Ion Cannon), que possui uma interface gráfica intuitiva e pode ser controlado por um mestre, o botmaster, via canais IRC.

Command and Control em Botnets

Botnets requerem um mecanismo para que o botmaster controle os bots. Esse mecanismo é chamado de **Command and Control (C&C)**.

C&C centralizado: HTTP e IRC

A forma mais simples de controlar uma botnet é fazer os bots checarem um site ou um chat do IRC (controlados pelo botmaster) regularmente e executar os comandos lá encontrados. Assim, o botmaster colocaria lá os comandos que deseja enviar aos bots e eles então os executariam.

O lado negativo dessa abordagem é, claro, que, caso o site ou o chat caiam, será impossível controlar os bots. Uma estratégia de mitigação dessa abordagem seria simplesmente bloquear qualquer comunicação entre os computadores de uma rede (no caso, a rede que se quer proteger) e o canal (site ou chat do IRC).

C&C descentralizado: DHT (Distributed Hash Table)

Você provavelmente está pensando que DHT é algo super complexo e longe da sua realidade. Bom, não é muito intuitivo não, mas você provavelmente já usou uma DHT antes: torrents!

A DHT é uma estrutura de dados distribuída, o que quer dizer que ela fica dividida em várias máquinas e usa a rede para comunicação, busca e inserção de dados.

Não vamos entrar em detalhes, só saiba que é possível usar DHTs para enviar mensagens em redes P2P (Peer-to-Peer), que pode muito bem ser uma botnet.

Leituras Adicionais

- Slides do minicurso
- Notas da Cloudflare sobre DDoS
- Página da Wikipedia sobre arquitetura Cliente/Servidor
- RFC 4732: Internet Denial-of-Service Considerations
- Post sobre SYN flooding, ICMP flooding e LAND attack
- Vários ataques de flood com Hping3
- Mais exemplos com Hping3
- Smurf attack (ICMP Storm)
- Artigo sobre Command and Control (C&C) da IEEE
- Montando ambientes de teste para ataques DoS
- Tipos de ataques DoS (teórico)