Inspetor HTTP

Gabriel de Oliveira Estevam

17/0142591

Thiago Santana Marques

14/0164049

Junho 2019

1 Introdução

Este trabalho tem como objetivo desenvolver um servidor PROXY HTTP para inspecionar o tráfego Web. Na inspeção, o usuário envia um URL HTTP e o programa desenvolvido captura o tráfego, com isso, pode realizar as funções de SPIDER e DUMP.

2 Servidor Proxy

Um servidor proxy trabalha como intermediador a comunicação computador-Internet. Essa ferramenta possui funcionalidades que melhoram a comunicação entre computador e internet, seguem abaixo as principais funcionalidades:

- **Cache de páginas Web**: o servidor proxy armazena páginas e arquivos já acessados, caso seja solicitada uma página já acessada, então o servidor proxy recupera essa página do cache. Porém, antes de entregar o conteúdo da página para o computador, o proxy verifica se a página foi alterada, utilizando envio de pacote, caso a página tenha sido alterada, então atualiza o conteúdo da página no cache. Essa funcionalidade evita o constante acesso, melhorando o fluxo da rede e a velocidade de resposta para o cliente.
- **Controle do acesso a páginas Web**: permite a opção de disponibilizar ao cliente o acesso ou não a determinadas páginas. Essa funcionalidade é utilizada normalmente em empresas para evitar que os funcionários acessem páginas indevidas (Exemplo: redes sociais, notícias, compras, etc).

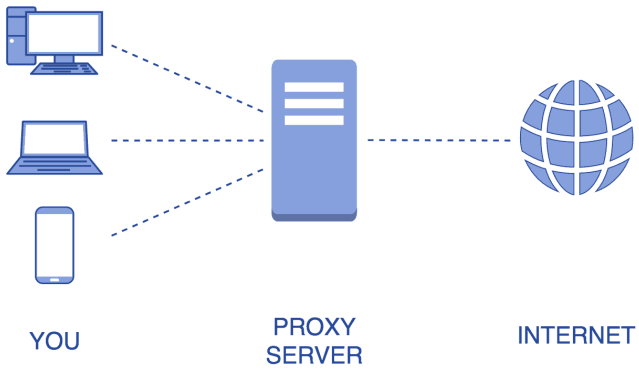

Figura 1: Servidor Proxy

- **Compartilhar o acesso a Internet**: O servidor proxy é quem está conectado a internet e os computadores acessam ele para acessarem os conteúdos Web, com isso, o servidor oculta as informações de identificação do computador que está acessando o conteúdo Web.

Segue abaixo figura mostrando as várias partes realacionadas com o servidor proxy:

3 Protocolos

Os protocolos utilizados para desenvolvimento do projeto foram: TCP e HTTP.

3.1 Protocolo TCP

O protocolo TCP é orientado a conexão, isso significa que antes de iniciar a transmissão de dados as aplicações estabelecem uma conecção que serve para definir os parâmetros de transmissão de dados.É um protocolo full-duplex, pois pode realizar o envio de segmentos, e também a recepção de pacotes transmitidos pelo outro lado ao mesmo tempo. Também é caracterizado por ser point-to-point, ou seja, trabalha entre um único sender e um único receiver, não permitindo multicasting.

3.2 Protocolo HTTP

- O protocolo HTTP (Hypertext transfer Protocol) é um protocolo de comunicação definido pela especificação RFC2616. É um protocolo baseado em requisições e respostas entre clientes e servidores. Tais mensagens são padronizadas e utilizam o procolo de transporte TCP.

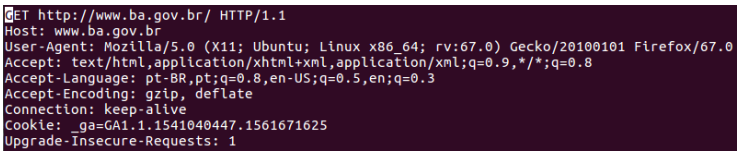

Figura 2: Exemplo de requisição HTTPS

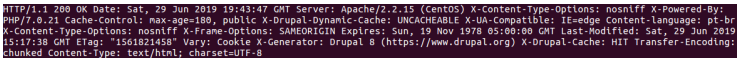

Figura 3: Exemplo de header de response HTTPS

- O cliente — navegador ou dispositivo que fará a requisição também é conhecido como user agent — solicita um determinado recurso (resource), enviando um pacote de informações contendo alguns cabeçalhos (headers) a um URI ou, mais especificamente, URL. O servidor recebe estas informações e envia uma resposta, que pode ser um recurso ou um simplesmente um outro cabeçalho.
- Como este protocolo não possui estado (stateless), não armazena informações dos clientes. Para capturar informações precisamos dos cookies, sessões, campos de formulários ou variáveis na própria URL.

4 Materiais teóricos

4.1 Livro: Redes de computadores e a internet: uma abordagem top-down

O livro Redes de computadores e a internet: uma abordagem top-down de Kurose e Ross foi utilizado para aprender os conceitos necessários para o desenvolvimento do projeto.

4.2 Slides da aula

Foram utilizados também os slides da aula de Redes de Computadores para sanar as dúvidas que surgiram durante o desenvolvimento do projeto.

5 Tecnologias

O código foi desenvolvido na linguagem de programação C e versionado no usuário GitHub de um dos componentes da dupla, também foi utilizado o brownser do navegador Mozilla Firefox para executar a função de cliente.

6 Arquitetura do Software

O software desempenha três funcionalidades: inspetor, spider e dump, por isso, foram criados um arquivo .c para cada funcionalidade, são eles: server.c (inspetor), spider.c e dump.c.

7 O código

7.1 Inspetor

Essa parte se refere ao arquivo server.c. Primeiramente são implementados sockets para a conexão com o servidor na porta 8228 (determinada no início do software). Para a criação do socket utilizamos a biblioteca sys/socket.h. Nessa biblioteca utilizamos as funções de criação do socket (socket()), atribuir um endereço a um socket (bind()) e executar o socket (listen()). Quando uma requisição é feita, então um socket é criado e reservado para uso do cliente e a conexão é aceita. Assim que a conexão é aberta recebemos a resposta enviada do servidor ao browser.

Quando a conexão cliente-servidor é feita, a mensagem request é gravada em um arquivo .txt para, então, realizar o tratamento dessa mensagem. Sendo assim, o usuário pode editar a mensagem request.

Função - tratamentoUrlHost: Esta função recebe a mensagem request e verifica onde estão as informações de URL e HOST, quando encontra, inclui essas informações nas variáveis url e host.

Após o fim do trartamento da mensagem request, é chamada a função obterIP.

Função - obterIP: Esta função recebe o Host e URL retirados da mensagem request, então, verifica se é possível avançar como Host. Em caso positivo, é criado um novo socket para esse cliente e é realizada a conexão. Com a conexão estabelecida a requisição com o método GET é criada para essa URL e Host, após isso, a requisição é enviada pelo socket.

Após o passo anterior é retornada uma mensagem que é escrita em um arquivo .txt, essa mensagem possui o .html da página. Agora, como o proxy deve ser transparente e permitir o fluxo de ida e volta, o código reenvia esse .html para browser, uma vez que todo o tratamento foi finalizado.

Logo após, é mostrado ao usuário um menu contendo as opções de SPIDER, DUMP e SAIR DO PROGRAMA.

7.2 SPIDER

O spider monta uma árvore hiertextual com todas as url's subjacentes a uma página baixada. Para isso, foi criada uma estrutura de dados árvore, para ser definido uma herança entre as url's. Inicialmente é necessário percorrer todo o html da página para encontrar as url's subjacentes, ou seja, é preciso procurar todos os href's contidos em uma página.

Função - spider: A função spider recebe o URL e o HOSTNAME caputurados no response HTTP recebido. Ela também recebe como parâmetro uma string com o nome do diretório que o html baixado será armazenado, a árvore a ser utilizada e, por fim, o diretório raiz que nada mais é que o nome do diretório da página a qual o spider será feito.

Nessa função, são abertos dois arquivos: index.txt e html tree.txt. O index.txt possui o html baixado de cada página, então, é nesse arquivo que será realizada a busca por href's. O arquivo tree.txt servirá como armazenador dos href's encontrados no index.txt.

Função - verificaArvore: Essa função recebe uma string com o href de uma página, então, ela realiza a busca na árvore para verificar se esse href existe na árvore. Em caso positivo, então a função retorna TRUE. Em caso negativo, ou seja, se não encontrou o href na árvore, então é retornado FALSE.

Função - construiArvore: A função construiArvore recebe uma string contendo o href, recebe a raiz da árvore, a string com o nome do arquivo a ser criado para salvar as referencias (href's), o HOSTNAME, a string contendo o diretorio atual e o diretorio raiz desse SPIDER. Essa função cria novos nós na árvore com a href passada no parâmetro. Feito isso, essa função cria um request HTTP solicitando o objeto html referente a um determinado href. Após isso, é criado um novo diretório para o armazenamento do html no arquivo tree.txt dessa referência encontrada. Por fim, a função spider é chamada até no máximo o número de níveis N (setado no arquivo functions.h).

Função - inicializaNo: Essa função inicializa (inclui NULL) em todos os nós filhos de um nó pai, ou seja, do href pai.

7.3 DUMP

A funcionalidade dump consiste em realizar o download dos arquivos descritos no spider para visualizar a URL e sua árvore localmente. O arquivo dump.c basicamente recebe a árvore do spider e realiza o download de todas as refêrencias contidas no URL.

Função - dump: Esta função recebe a string da URL do objeto que será baixado e a string do HOSTNAME da URL raíz.O arquivo html tree.txt é aberto e varrido para obter todos as URL's referentes a URL raiz. Após isso, é criado um socket para realizar a comunicação cliente-servidor. Com isso, os requests foram feitos tratando strings e utilizando os dados dos URL's obtidos no arquivo html tree.txt e sempre utilizando o HOSTNAME que foi passado como parâmetro da função. No final da função dump, utilizamos a função write, da biblioteca de sockets para enviar a requisição ao servidor, via TCP. Agora, utilizamos a função read da biblioteca de sockets para continuamente ler o buffer e extrair as informações passados pelo servidor. Essas informações são extraídas e salvas de acordo com sua extensão utilizando um fopen, também é válido salientar que cada arquivo é salvo no diretório de acordo com o nome presente em sua url.

8 Considerações Finais

Por meio desse trabalho foi possível colocar em prática muitos conceitos aprendidos durante as aulas de Redes de Computadores de 01/2019, esses conceitos são: sevidor proxy, socket, protocolo HTTPS, protocolo TCP, etc. Existem algumas limitações, pois, durante a execução da função SPIDER houve o problema de falha de segmentação, porém, boa parte da árvore foi imprimida. Acreditamos que isso ocorre, pois existe algum erro de lógica na função de impressão da árvore. Essa limitação também afeta o DUMP já que não serão baixados todos os arquivos, pois, não é possível percorrer toda a árvore, mesmo assim, a função DUMP está correta, já que está baixando os arquivos que ela consegue. Outro ponto é que não foi implemetado uma persistência no software, ou seja, para inspecionar outra página é necessário comandar novamente o início do software.

9 Observação

Para executar o código: Acesse o link https://github.com/ThiagoMarques/$RC_{12}019$ e clone o repositório para seu computador. Após isso, abra o terminal na pasta criada e digite *gcc -o server server.c spider.c dump.c*, então o programa será compilado. Uma vez compilado, execute o programa com o comando *./server*. Sites para teste:

- www.ba.gov.br;
- http://flaviomoura.mat.br/paa.html

Referências

KUROSE, James F.; ROSS, Keith W.; ZUCCHI, Wagner Luiz. Redes de Computadores ea Internet: uma abordagem top-down. Pearson Addison Wesley, 2007.