INSTITUTO POLITÉCNICO DE BRAGANÇA - CAMPUS MIRANDELA
LICENCIATURA EM INFORMÁTICA E COMUNICAÇÕES

TRABALHO DE SISTEMAS DISTRIBUÍDOS

SISTEMA DE TESTE DE VULNERABILIDADE COM SOCKET

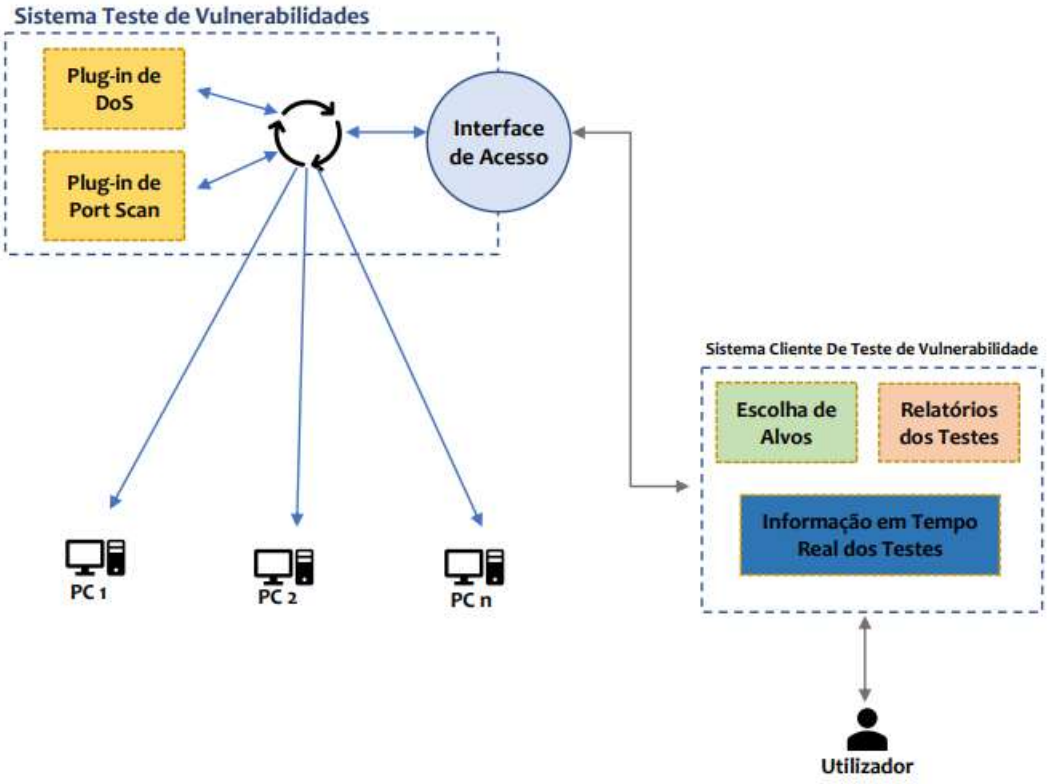
PATRICK SILVA MENEZES

MIRANDELA
2023

INTRODUÇÃO

O trabalho da disciplina de Sistemas Distribuídos sobre Sockets, consiste no desenvolvimento de um sistema de testes de vulnerabilidades de segurança fazendo uso da tecnologia de comunicação Socket, threads e interface gráfica para interatividade do cliente com as funcionalidades implementadas pelo servidor, podendo ainda mais de um cliente ser executado ao mesmo tempo devido ao paralelismo obtido através do uso de threads.

OBJETIVO



Breve descrição do cenário ilustrado

O cenário acima ilustrado representa um sistema de testes de vulnerabilidades de segurança baseado no modelo Cliente/Servidor. A componente servidor deverá ser capaz de dirigir dois tipos de testes a um ou mais computadores alvo em simultâneo. Os testes a efetuar serão do tipo DoS e Port Scan. A informação em tempo real de cada teste a cada computador deverá ser visível (também em tempo real) na componente cliente. No final de cada teste, deverá ser compilado no cliente um relatório/resultado de cada teste. Para os teste de Port Scan, deverá constar a informação das portas abertas por cada computador alvo. Para o teste de DoS, deverá constar, por computador alvo, o número de conexões que foi necessário estabelecer até que o computador deixou de responder. Este teste de DoS deverá ter como alvo um dado serviço (porta) escolhido pelo sistema cliente.

METODOLOGIA – CLASSE SERVER

Do lado do servidor foi criada a classe *TesteVunerabilidadeServer* a qual herda da classe Thread seus métodos que possibilitam fazer o uso de threads no decorrer do desenvolvimento do código.

Logo em seguida é criado um objeto do tipo Socket chamado *clientConnection* de maneira global para que além do escopo do *main*() o objeto possa ser visto também no escopo do método void *run*() da Thread que ainda será criada.

Em seguida foi criado um construtor onde é passado o objeto *clientConnection* como parametro. Já dentro do escopo do método *main*() é colocado o processo do servidor a escuta na porta 5000, através da classe ServerSocket é instanciado o objeto *listener* que fica na escuta dessa porta.

Dentro de um while(true) o servidor fica a espera de um cliente se conectar, a conexão é aceita apenas depois de criar o objeto *clientConnection* que representa o estabelecimento dessa ligação receber a aceitação de conexão ao servidor da seguinte forma: *Socket clientConnection = listener.accept();*.

Após o estabelecimento dessa ligação, para que mais de um cliente possa interagir com o sistema ao mesmo tempo, é criada uma Thread para atender cada ligação que se estabelece, e é passada na instanciação do objeto *t* da classe Thread a ligação recém estabelecida através do construtor: *TesteVulnerabilidadeServer(clientConnection)* e então é colocada em funcionamento a Thread criada através do método *start*() da classe Thread.

No método void *run*() são criados os canais de entrada e saída de dados através das classes *BufferedReader* e *PrintStream* respectivamente onde são instanciados os objetos “clientReceiver” representando o canal de entrada do servidor e *clientSender* o canal de saída do servidor.

Logo após isso é declarada e inicializada a variável *dataFromClient* do tipo String, que será responsável por receber os dados que o cliente está a enviar do outro lado da comunicação. Sendo assim, essa variável será importante para identificar no corpo dos dados enviados pelo cliente através do corte da String recebida usando a função *substring()* se o teste de vulnerabilidades escolhido foi <PS> para PortScan ou <DS> para Dos Attack. O resultado desse corte na informação recebida é guardada na variável também do tipo

String *testType* e através dela é feito 2 condicionais se a tag que identifica o tipo de teste que chegou for <PS> será realizado o Port Scan através do método portScan da classe PortScan, caso contrário se for <DS> então será realizado o Dos attack através do método da classe AttackDos. Para que esses métodos corram bem, devem ser passados os respectivos parametros que cada um requere e para isso foram feitos cortes no dado recebido no caso do Port Scan na String contida em *testType* e em *dataFromClient* no caso do Dos Attack , de forma a buscar receber para passar para os parametros os dados que correspondam corretamente. Para chamar os métodos de cada tipo de teste de vulnerabilidade do sistema, foi preciso apenas declarar um objeto da classe respectiva do tipo de teste e fazer a chamada do método.

Por fim, são fechados os canais de entrada e saída estabelecidos para o servidor e ainda é também fechado a conexão com o cliente.

METODOLOGIA – CLASSE PortScan

Na classe PortScan foi criado um construtor da classe para instanciação de objetos da mesma em outras classes, como é o caso da classe *TesteVunerabilidadeServer*, onde ocorre isso para fazer a chamada do método *portScan().*

O funcionamento do método *portScan()* ocorre de forma que há uma tentativa de conexão via Socket com o ip alvo e a porta “x” que varia seu valor dentro de um loop que vai de um valor inicial a um valor final de portas que se deseja fazer o port scan. Em caso de ser estabelecida uma conexão com a porta é informado ao cliente que a porta “x” está aberta, em caso contrário no catch do try-catch é informado que a porta está fechada devido a falha na conexão com o cliente na porta “x”. Além disso, vale ressaltar que antes de uma nova tentativa de conexão com uma porta diferente é colocado um delay de 1ms e se estabelecida a conexão a mesma deve ser fechada para a tentativa das portas posteriores.

METODOLOGIA – CLASSE AttackDos

Na classe AttackDos foi criado um construtor da classe para instanciação de objetos da mesma em outras classes, como é o caso da classe *TesteVunerabilidadeServer*, onde ocorre isso para fazer a chamada do método *dosAttack().*

Além disso, foram declaradas 3 variáveis globais estáticas *ip_target_received, port_target_received e sendLogToClient*, de respectivos tipos String, String e PrintStream, para que no método *void run()* da thread possam ser usadas com valores inalterados que são recebidos pelos parametros do método *dosAttack().*

No método dosAttack(), é implementado a lógica de ataque de negação de serviço onde são criadas e inicializadas múltiplas threads ao mesmo tempo através do loop infinito *while(true)*, para que seja possível perceber cada thread nova sendo criada é colocado um delay de 7ms.

No método *void run(),* da classe Thread são feitas múltiplas requisições de conexão a um determinado ip representado pela variável global *ip_target_received* e um determinado serviço determinado pela variável gloval *port_target_received*, em caso de haver conexão estabelecida é mostrada uma mensagem que ainda está sendo possível estabelecer conexão com a porta, caso contrário fora do try dessa tentativa de conexão é mostrado uma mensagem de que a conexão falhou devido a porta ter sido derrubada pelo ataque de negação de serviço.

METODOLOGIA – CLIENT

Do lado do cliente foi criado a classe *TesteVulnerabilidadeClient* que herda também métodos da classe Thread. Logo no início da classe foram declaradas variáveis globais sendo elas: *ligado, entrada, sendToServer* e *clientInterface* que serão usadas posteriormente no escopo do método *void run()* da thread. Além disso, foram criados dois construtores, o primeiro tem como parametro *entrada* que é do tipo *BufferedReader* representando o canal de entrada do cliente e o segundo tem como parametro o objeto do tipo da Classe *PortScanDos_Client* que representa a interface interativa do cliente.

Nessa classe *TesteVulnerabilidadeClient* ainda foram criados outros métodos além dos construtores, como é o caso do método *ConexaoServidor()*, que permite o cliente estabelecer uma conexão com o servidor ao clicar no botão de iniciar o teste *START*, e esse método é chamado no método do evento de mouse released do botão *btStartMouseReleased()*. Essa conexão ocorre de forma que se instancia um objeto da classe *Socket*, passando o endereço de ip de loopback e a porta na qual o servidor está escutando. Em seguida, são criados os canais de entrada *receivedFromServer* e saída *sendToServer* do lado do cliente, respectivamente do tipo *BufferedReader* e *PrintStream*.

Ao fim desse método *ConexaoServidor()* é instanciado um objeto da classe Thread sendo passado o canal de entrada *receivedFromServer* para ele, e então a thread é inicializada. O canal de entrada deve ser passado, pois a cada conexão estabelecida com o servidor, cada cliente tenha seu próprio canal de entrada, da mesma forma que do lado do servidor cada cliente terá uma thread que lidará com cada ligação, do lado do cliente também deve haver uma thread para separar o fluxo de entrada próprio de cada cliente.

Antes do método *void run()* da classe Thread ainda foram criados dois métodos responsáveis por enviar dados para o servidor, dados esses necessários para realizar os testes de vulnerabilidade. O método *enviarDadosPortScan* foi criado de forma que se envia para o servidor através da variável global de canal de saída *sendToServer* uma mensagem no seguinte formato: ("<PS>"+ipTarget), possuindo <PS> como identificador de um método de portScan concatenado com o ip inserido pelo cliente na interface ao ser selecionado o teste de PortScan e passado o ip da máquina alvo. O método *enviarDadosDos* tem o funcionamento parecido com o método explicado anteriormente, faz uso também da variável global de canal de saída *sendToServer* e envia para o servidor uma mensagem no seguinte formato: ("<DS>"+ipTarget+"|"+portTarget), possuindo <DS> como identificador de um método de attack Dos concatenado com o ip inserido na interface e após o pipe "|" é passado concatenado a porta do serviço alvo.

Por fim no método *void run()* da classe Thread é feito a chamada do método *recebeMensagemServidor()* da classe da interface do cliente por onde é passado como parametro os dados recebidos do servidor através da variável de canal de entrada *entrada* e do método *readLine()* da classe *BufferedReader*, sendo assim no método *recebeMensagemServidor()* é feito a adição das mensagem vindas do servidor no componente List chamado *reportList*.

METODOLOGIA – INTERFACE GRÁFICA CLIENT

A interação do cliente com o servidor foi implementada na classe *TesteVulnerabilidadeClient*, e então foi criada uma interface que pudesse fazer uso dos métodos desenvolvidos nessa classe. Ao criar uma interface no NetBeans é gerado um código automático de acordo com os elementos selecionados e eventos criados sobre os mesmos, a classe criada tem por nome *PortScanDos_Client*. No início da classe *PortScanDos_Client* foi declarado um objeto global da classe *TesteVulnerabilidadeClient* de nome *client* que pode ser utilizado no escopo de qualquer método da classe *PortScanDos_Client.* Além disso foi gerado um construtor automático pelo NetBeans onde é inicializado os componentes e onde é criado um objeto da classe *TesteVulnerabilidadeClient* passando para construtor o objeto da classe *PortScanDos_Client* através do *this.*

Logo em seguida foi criado o método *recebeMensagemServidor()* citado anteriormente na explicação da metodologia client, que recebe uma String do servidor e adiciona no componente *reportList* do tipo *List.*

Por fim, no botão *btPortScanMouse* ao ser realizado o evento de mouse realised, é setado o enabled para false desabilitando o mesmo através da função do componente *setEnabled()*, além de desabilitar o campo de porta alvo, pois para operação de PortScan não é necessário passar porta, apenas o ip alvo. Já no botão *btDosMouse* ao ser realizado o evento de mouse realised, é setado o enabled para false desabilitando o mesmo através da função do componente *setEnabled*. E no botão *btStart* primeiro é chamado o método *ConexaoServidor()* através do objeto client da classe *TesteVulnerabilidadeClient*, e então é feita uma verificação através de condicionais se o botão de Port Scan estiver desabilitado e tiver preenchido o campo de ip alvo é feito a chamada do método *enviarDadosPortScan()* da classe *TesteVulnerabilidadeClient* e é passado o valor de ip inserido no campo *txtIpTarget*, caso seja o botão de Dos Attack que esteja desabilitado e tiver preenchido o campo de ip alvo e porta alvo é feito a chamada do método *enviarDadosDos()* e é passado os valores de ip e porta inseridos no campo *txtIpTarget* e *txtPortTarget* respectivamente, e por fim é desabilitado o botão *btStart*.

CONCLUSÃO

Diante o desenvolvimento do trabalho, foi possível perceber melhor o conceito do uso de socket para comunicação em sistemas distribuídos, usando o modelo cliente/servidor. Além disso, foi possível perceber melhor o funcionamento e a importância da criação de threads ao desenvolver uma aplicação distribuída, que proporcionou o paralelismo na execução de processos e uma melhor performance tanto em tempo de execução quanto memória.