Guião de demonstração da Segurança
Grupo T64-Komparator
Francisco Teixeira de Barros, 85069, LETI
Sistemas Distribuídos, Ano Lectivo de 2016-2017

Obter o código

1. Comece por digirir-se a página do GitHub deste grupo, cujo o endereço é o https://github.com/tecnico-distsys/T64-Komparator de seguida, no menu secundário horizontal clique *releases* e faça download da versão mais atual. Alternativamente poderá escrever no seu terminal a seguinte linha git clone -b NomeDaReleaseDesejada http://github.com/tecnico-distsys/T64-Komparator, desde que tenha as devidas autorizações.
2. Dirija-se à pasta de *downloads* e com o botão direito do rato, clique na opção de extrair. Deverá obter uma pasta com o nome T64-Komparator-SD_P3_v1.4 ou outra versão mais atualizada.

Instalar e compilar

3. Assumindo que optou por fazer download do zip. Lance uma instância do seu terminal, dirija-se para a diretoria aonde guardou o projecto que acabou de extrair e após entrar nessa mesma diretoria, insira o comando *mvn clean install -DskipITs*, isto eliminará possíveis conflictos com versões anteriores do programa e instalará as dependências entre os módulos de forma automática sem que sejam obtidos problemas derivados de classes de testes. Prossiga para o passo 4. assim que for impressa a mensagem "*BUILD SUCESS"* no *standart output* do seu terminal.

Demonstração funcional

4. Recorrendo agora ao atalho *ctrl+shift+T*, abra quatro instâncias novas do terminal. Estas por defeito estarão na diretoria da instância do terminal a partir da qual utilizou o atalho.
5. No primeiro terminal extra que abriu, escreva o seguinte comando *cd supplier-ws* e assim que a diretoria tiver mudado escreva *mvn compile exec:java*, quando vir a mensagem "*Awaiting connections"* nessa janela do terminal poderá prosseguir.
6. Análogamente para o segundo terminal escreva *cd supplier-ws* e depois *mvn exec:java -Dws.i=2*
7. No terceiro terminal escreva *cd mediator-ws* e depois *mvn compile exec:java*
8. Quando todos os terminais tiverem impresso a mensagem "*Awaiting connections",* o que deverá ser imediato, poderá escrever no quarto terminal cd *mediator-ws-cli* e de seguida o comando *mvn install*, isto correrá vários testes de integração que não tem conflictos com o código ao contrário dos testes de IT do módulo *supplier-ws-cli.*
9. Quando a mensagem "*BUILD SUCCESS"* for imprimida no terminal, poderá navegar o mesmo e inspecionar entre os vários terminais as mensagens que forma trocadas entre os vários módulos.

Breve explicação

Enquanto percorre as mensagens SOAP que foram imprimidas durante a passagem dos 44 testes de integração. Note-se que estas alteram o seu conteúdo consoante a operação invocada e que o conteúdo as várias mensagens respeitantes à mesma operação alteram o seu contéudo ao longo do tempo. Isto deve-se a intercepção das mensagens SOAP à saída dos emissores e à chegada dos receptores por parte das cadeias de *Handler.*

Às mensagens trocadas entre *mediator-ws-cli* e o *supplier-ws* é possível reparar numa camada de segurança que as protege de ataques por repetição, através da adição ao cabeçalho um requestID

aleatório, utilizado para verificar que pedidos já foram respondidos pela instância alvo da operação Supplier e ainda uma marca temporal que marca o tempo entro o despacho da mensagem e a sua receção. Se este diferencial for maior do que três segundos a chegar ao Supplier após o seu envio é lançada um RuntimeException e a mensagem é rejeitada. Esta exceção poderá ser vista numa das outras janelas do terminal, que representa o outro fim do canal de comunicação desta intereção entre entidades. Em alguns casos são impressas mensagens auxiliares extra não lançadas em exceção apenas com o objetivo de melhor notar alguns acontecimentos. Para além destes dois SOAPElements são ainda adicionados outros dois, por parte dos handlers de assinatura. Um deles contem o nome do remessor e outro a sua assinatura. O nome do remessor permite que os *Handlers* de assinatura que estao do outro lado do canal e que vão receber a mensagem, possam obter a chave pública do remessor e testar de a mensagem foi realmente enviada por este através das operações de digest. Quando um Supplier envia uma resposta (Outbound) para o remessor inicial, este último vai receber também a identificação e a assinatura do Supplier por forma garantir que a resposta que obteve foi também, de modo análogo, enviada pelo Supplier a quem pediu que fosse realizado algum trabalho e com quem comunicou anteriormente comunicou anteriormente. Contudo, neste caminho de regresso, o tempo que a mensagem demora a chegar não foi considerado importante, pelo que os cabeçalhos são menos extensos. Caso fosse se deseje tal funcionalidade, bastará juntar os dois Handlers de frescura num só, fazendo com que ambos os intervenientes da comunicação SOAP passem a ter handlers de frescura que atuam sobre a frescura das mensagens, ao inves de definerem comportamento apenas para um dos casos, consoante a direção da comunicação.

Nas comunicações entre o Mediador-ws e o mediator-cli-ws, é possível verificar que sempre que a operação invocada na mensagem SOAP é uma ação de compra com a *tag buyCart*, esta vê o conteúdo da sua mensagem a ser alterado no campo cartão de crédito, obtido pela procura do node creditCardNr no SOAPBody da mensagem SOAP trocada entre os intervenientes, mas apenas quando a mensagem sai do mediator-ws-cli em direção ao mediator-ws. Sendo decifrada apenas à chegada do Mediator-ws, momento a partir do qual a informação é descartada e não é reincluída na mensagem de resposta pois não existe necessidade de reenviar a informação do cartão de crédito de volta para o mediator-ws-cli. É mais seguro desta forma, mais eficiente e menos trabalhoso. Pelo que no caminho de regresso os Handlers limitam-se a fazer *logging* do conteúdo das mensagens, à semelhança do que já era feito até aqui, que tende a manter-se a menos que algo de errado ocorra. Para melhor visualizar a intereção entre os clientes com operações de compra, pode correr na diretoria $ *mediator-ws-cli*, o comando mvn *-Dtest=BuyCartIT.java test*, este comando fará com que todos os testes respetivos a esta operação sejam executados. Naturalmente poderá efetuar o mesmo para outras classes de teste que não a *BuyCart*. Caso deseje correr apenas um teste específico pode optar ainda por escrever *-Dtest=ClassITDesejada#Metododesejado test*.

Demonstração de casos de erro:

1. Por falta de tempo, só foi desenvolvido um teste representador do comportamento do programa. Neste é executado sempre que na diretoria $ security é escrito o comando *mvn install* ou *mvn verify*

Notas finais para o leitor / avaliador:

1. Não é provável que venham a ser incluídos mais testes no módulo security, pois faltam apenas sessenta minutos para o *deadline* do projecto. Ainda assim vale a pena executar o comando indicado pelo sim pelo não.

2. Existe uma pequena parte deste projeto que não está implementada, pois ao longo dos últimos dois dias não conseguir implementar nenhuma das soluções para o problema em causa. Esta parte está ainda assim escrita nos respetivas classes (SignatureSupplierClientHandler e

SignatureSupplierHandler), mas está comentada. Se por acaso não estiver comentada é porque ainda mudei a tempo. O motivo pelo qual não tentei solucionar o problema de outra forma mais simples e directa, deve-se ao facto de eu não saber que os testes de integração da segunda entrega iam ser corridos com o código dos *handlers* comentados, caso fosse necessário, como dito pelo responsável da cadeira via e-mail, pelo que perdi tempo a tentar implementar uma solução mais díficil, aparentemente trivial, mas nada trivial para mim pois não tenho a prática a mexer nesses campos da programação e porque também estou a fazer o projecto sozinho