Segurança de Redes e Sistemas de Computadores 2017/2018
Trabalho Prático nº 2 (v1.0), 15/Maio/2018)

ENUNCIADO DE REFERÊNCIA

Resumo

Este trabalho visa a concepção, desenvolvimento e ensaio experimental de uma plataforma confiável de software como serviço (aproximando uma solução do tipo PaaS ou Platform as a Service). A plataforma será concebida de forma a suportar genericamente micro-serviços disponibilizados em ambientes isolados (em ambiente sandboxing), virtualmente executados em contentores docker e asism beneficiando das garantias de isolamento e autocontenção neste tipo de tecnologia. Os serviços disponibilizados pela plataforma podem então ser utilizados por clientes remotos que podem realizar operações sobre os mesmos, com garantias de privacidade (com preservação de confidencialidade de dados trocados e mantidos nesses serviços), bem como com garantias de auditabilidade providenciada por módulos do tipo TPM (Tamper Proof Modules) que permitirão aos clientes obter provas de autenticidade e integridade dos serviços docker em execução na plataforma, incluindo ainda componentes selecionados da pilha de software que constitui a plataforma no seu todo. Como contexto de aplicação, a concepção da plataforma disponibilizará serviços de gestão e armazenamento de dados com base em contentores de serviços para suporte de repositório de dados do tipo chave-valor, tendo por base a tecnologia REDIS.

Referências iniciais a ter em conta:
SRSC-TP2 Background (bem como materiais e referencias constantes neste documento).
Apresentação TP2-Initial-Overview.ppt (apresentação em aula prática)

1. Vertentes de desenvolvimento do trabalho TP2

1.1 Desenvolvimento de um serviço baseado no *Key-Value-Store* REDIS suportado na plataforma

Este desenvolvimento visa suportar um serviço REDIS (como micro-serviço em contentor virtualizado – *docker* – como nível de aplicação da plataforma e desenvolvimento. O restante desenvolvimento passa por uma de duas diferentes variantes do trabalho (a selecionar livremente por cada grupo) como contexto de aplicação específico:

- **Variante 1: Aplicação cliente/serviço REDIS** com garantias de autenticidade, confidencialidade e integridade na operação dos dados que são mantidos ao nível do repositório chave-valor REDIS
  - Este contexto de aplicação como prova de conceito de aplicação da plataforma concebida é indicado para grupos que não tenham realizado a FASE 2 do trabalho prático nº 1.
- **Variante 2: Serviço de autenticação, controlo de acesso e estabelecimento de chaves/associações de segurança,** como iteração do serviço de autenticação e controlo de acessos inicialmente implementado no trabalho prático nº 1, e que usará um repositório REDIS com garantias de autenticação, confidencialidade e integridade dos dados armazenados e operados no contexto desse serviço.
  - Este contexto de aplicação como prova de conceito de aplicação da plataforma concebida corresponderá a uma reimplementação da Fase 2 do trabalho prático nº1, cuja solução será concebida, implementada e integrada (com os restantes desenvolvimentos do trabalho prático nº 1), por parte dos grupos que tenham realizado essa fase do trabalho prático nº 1.

1.1.1 Variante 1 - Aplicação cliente/serviço REDIS (até 8 valores)

Para concretização desta variante de aplicação, deverá ser concebida e materializada uma base de dados REDIS, numa tabela com 100 entradas. A base de dados REDIS deverá poder suportar no *key-vaue-store* REDIS uma tabela de pelo menos 100 linhas com o mínimo de 6 colunas. O conteúdo da base de dados corresponderá à concretização de um cenário de aplicação livre, concebido por cada grupo para prova de conceito. A ideia para o cenário a criar é criar um conteúdo da base de dados que pressupõe que a sua operação e armazenamento exigissem garantias de autenticação, integridade e confidencialidade dos dados.

Sugestões indicativas para o efeito podem ser (como exemplos):

- Tabela com movimentos bancários de contas de clientes de diversas entidades bancárias, sendo as entradas pesquisáveis por número de cartão de cidadão do cliente (string), data (string), montante (valor inteiro), identificador do cliente na entidade bancária (string) e código IBAN da conta (string), campos

estes que devem também corresponder a colunas da tabela. Cada entrada poderá conter outras colunas complementares.

- Tabela com salários de empregados de uma organização, sendo as entradas pesquisáveis por número de cartão de cidadão do empregado (string), data de emissão (string), salário bruto (valor inteiro), código do departamento ao qual o empregado pertence (string) e número de contribuinte fiscal do empregado (string), campos estes que devem também corresponder a colunas da tabela. Cada entrada poderá conter colunas complementares.
- Tabela com registo de clientes de uma dada empresa, sendo as entradas pesquisáveis por número de cliente (inteiro), cartão de cidadão do cliente (string), data de emissão (string), morada (string), telefone (string), número de contribuinte fiscal do cliente (string), campos estes que devem também corresponder a colunas da tabela. Cada entrada poderá conter colunas complementares.
- Outros exemplos (similares) que podem ser criados: tabela com dados de análises médicas (exemplo, indicadores de records com valores de análises sanguíneas associados a utentes do serviço nacional de saúde ou de pacientes de um hospital), tabelas de preços de stocks de fornecedores de uma empresa, com indicação do fornecedor, nº de contribuinte do fornecedor, contacto do fornecedor, produto, preços de aquisição do produto, etc. Cada entrada poderá conter colunas complementares.

Requisitos base para a concretização do cenário de aplicação: De acordo com o cenário de aplicação implementado por cada grupo e que deve ter em conta o modelo de adversário definido na arquitetura de referência da plataforma, cada grupo deverá:

- Criar o *dataset* (100 entradas) que permite popular a base de dados com dados iniciais para serem operados, devendo estes ser armazenados de forma que se assegure confidencialidade, autenticidade e integridade de cada entrada. A confidencialidade será assegurada com base em criptografia simétrica. As provas de autenticidade serão estabelecidas com assinaturas digitais (usando criptografia de chave publica) e as provas de integridade podem usar sínteses de segurança, MACs ou CMACs. Para o efeito o clienet deverá poder ser parametrizável, de modo a poder usar qualquer algoritmo/método criptográfico para estes efeitos;
- Deverá ser desenvolvido uma aplicação cliente (Java / JEDIS), que funcione como um *benchmark* para realizar 100 operações (com um híbrido de operações SET / GET e ERASE (ou REMOVE). Estas operações deverão poder ser realizadas com base em chaves associadas a qualquer um dos atributos pesquisáveis da tabela. A ideia é que deste modo se possam obter indicadores de desempenho médio das diversas operações (em operações/segundo), de modo a avaliar o desempenho (*throughput*) das operações sobre o repositório REDIS devidamente protegido.
- Todas as chaves ou parâmetros criptográficos devem ser apenas mantidos no cliente já que os utilizadores e todo o software (cliente) são considerados confiáveis, estando fora do modelo de adversário.

Opcionalmente, serão consideradas valorizações adicionais (até 2 valores adicionais) em relação aos requisitos de implementação anteriores, os seguintes:

- A interação (operações cliente/serviço REDIS) suportadas em canal TLS (o que pode ser feito com base na análise do suporte REDIS para este efeito)
- Parametrização da criptografia usada pelo cliente, nomeadamente keystores ou configurações criptográficas que possam ser usadas de forma flexível pelo cliente.

1.1.2 Variante 2 - Serviço de autenticação, controlo de acesso e estabelecimento de chaves/associações de segurança (valorização semelhante ao da variante 1)

No caso dos grupos que selecionem esta variante de implementação, estes deverão discutir com o docente as opções e requisitos que irão implementar, de modo a que sejam perviamente validados. De qualquer modo, os dados que sejam mantidos na solução REDIS deverão estar armazenados ocm garantais de autenticidade, confidencialidade e integridade, de modo a que se suporte o modelo de adversário definido para a plataforma (e que é comum à variante 1).

2. Desenvolvimento do suporte de confiabilidade com módulos TPM (até 12 valores)

Neste desenvolvimento, serão adicionados os módulos TPM definidos na arquitetura da plataforma, designados por VMS-TPM e GOS-TPM (conforme representados no documento TP2-Initial-Overview.ppt).

Estes módulos deverão ser implementados em JAVA e depois serão contentorizados em imagens *docker* – aspecto valorativo (embora possam ser considerados arquivos executáveis *jar* durante o processo de desenvolvimento ou mesmo para efeitos de demonstração). De acordo com a arquitetura indicada (TP2-Initial-Overview.ppt) estes módulos permitirão proceder à auditoria da configuração e estado de execução da plataforma, com base num protocolo de atestação disponibilizado para clientes. A implementação passa por garantir os seguintes requisitos:

- **Módulo VMS-TPM**. Este módulo atestará a confiabilidade de módulos de serviço (nível aplicação) disponibilizados. A atestação será baseada na inspeção do estado dos processos iniciados em contentores dos serviços (exemplo contentores docker executados como serviços em execução na plataforma, ao nível do sistema operativo GUEST OS Linux. As evidências que o módulo permite auditar basear-se-á no output típico de informação de comandos nomeadamente: monitorizações via os comandos docker (monitorização de contentores e imagens de referência em execução), ps – process status e ls – list directory contentes, de modo que se possa auditar o contexto de execução desses serviços como processos na plataforma, face a um contexto de execução confiável de referencia pré-conhecido pelos clientes. Tal inclui a correção do estado dos processos em execução bem como a integridade e consistência do fie system, nomeadamente em relação a ficheiros críticos que possam conter configurações ou, desde logo, os próprios ficheiros com código binário correspondente aos processos em execução e que devem ser verificados em relação às evidências de integridade no sistema de ficheiros. Estas evidência deverão basear-se no conteúdo íntegro e tamanhos desses ficheiros, datas de referência de alteração bem como permissões e atributos de controlo de acesso no *filesystem*. A atestação far-se-á com base num protocolo de atestação, por comparação da captura destas evidências obtidas dinamicamente (e as provas inicialmente conhecidas pelo cliente), como cerificação de confiança de configuração e estado de execução corretos da plataforma.

- **Módulo GOS-TPM**. Este módulo atestará a confiabilidade do ambiente GUEST-OS, ao nível do runtime de processos nativos em execução no GUEST-OS, de modo a atestar-se a conformidade do estado de execução no ambiente de processos do GUEST-OS, face a um contexto confiável de referencia.

- **Protocolo de atestação implementado pelos módulos GOS-TPM e VMS-TPM.** Este protocolo será implementado pelos módulos GOS-TPM e VMS-TPM. No essencial é um protocolo do tipo pergunta/resposta e que será protegido num canal TLS. O protocolo tem uma especificação comum aos dois módulos, que executarão em *endpoints* TLS em portos distintos bem conhecidos e que serão definidos em cada implementação. A implementação do protocolo pode fazer-se com sockets TLS ou com base numa invocação REST/TLS, com parametrização flexível da versão TLS e *ciphersuites* da conexão TLS impostas por pelos módulos (no *handshake* TLS). Notar que se deve implementar o necessário suporte de modo a poder executar-se o protocolo TLS no protocolo de atestação, o que inclui keystores e certificados de chave pública, como certificados confiáveis por parte do cliente.

  O protocolo será implementado com base na seguinte especificação de referência (que deve ser complementada na especificação concreta por parte de cada grupo) e que deve usar um acordo de Diffie-Hellman, como se referencia:

  O cliente envia no pedido (protegido na conexão TLS):
  > ATTESTATION REQUEST CODE || PUB- DH (1024 bits) || Secure RANDOM NONCE

  ATTESTATION REQUEST CODE: é um mero código de operação (ex: 0x00)
  PUB-DH: é um número público Diffie-Hellman gerado pelo cliente para o pedido de atestação
  NONCE: número aleatório gerado pelo cliente para o pedido de atestação.

  Cada módulo (em causa) responde ao cliente da seguinte forma:

> ATTESTATION RESPONSE CODE || ATTESTATION SIGNATURE || ATTESTATION STATUS

ATTESTATION RESPONDE: é um mero código de operação, ex: 0x01
ATTESTATION SIGNATURE: é uma assinatura digital cobrindo:

- Um número público Diffie-Hellman gerado pelo módulo em causa para a resposta
- A resposta o NONCE do cliente (exemplo, NONCE+1)

ATTESTATION STATUS: é uma lista com um conjunto (lista) de provas de síntese correspondentes ao estado auditado. A lista é enviado cifrada com AES, usando como chave K a chave derivada do acordo Diffie Hellman resultante da troca de números públicos DH.

O restantes parâmetros necessários ao estabelecimento da chave K com base no acordo de Diffie-Hellman podem ser considerados fixos (pré-fixados na implementação), embora possam também ser enviados no pedido de atestação.

Note-se que de acordo com o protocolo de atestação tal como indicado como referência:

- O suporte TLS pode ser feito com autenticação unilateral do servidor, embora fosse susceptível de ser implementado com autenticação mútua, o que no entanto não se sugere.
- A parametrização da cifra simétrica (algoritmo, tamanho de chave, modo ou *padding*) para se cifrar a prova de atestação pode estabelecer-se de forma fixa. De qualquer modo o cliente pode indicar a sua preferência (na forma de uma *string:* ALG/MODO/PADDING – por ex., AES256/CBC/PKCS7Padding, como um campo suplementar na mensagem de REQUEST.

A especificação indicada deixa propositadamente em aberto quais as provas de integridade (referentes aos estados auditáveis de execução) serão auditáveis pelos módulos TPM. Os grupos devem analisar e propor essas provas (a obter para a comprovação do estado de correção e de integridade da plataforma), sabendo que as mesmas têm que ser do conhecimento do cliente (que as usará como inputs para a comprovação de atestação).

- **Cliente com protocolo de atestação.** Após implementação dos módulos TPM e protocolo de atestação, bastará adicionar ao cliente que foi inicialmente implementado uma fase inicial de atestação, antes de iniciar as operações sobre o serviço REDIS da plataforma que só será usado se o cliente garantir essa prova de confiabilidade. Deverá então ser repetido agora o *benchmark* (inicialmente feito), de modo a tirarem-se elações sobre o impacto do protocolo de atestação, comparativamente à primeira observação. Note que a atestação através dos dois módulos TPM da plataforma será feita em paralelo (com base em duas *threads* de atestação).