Segurança de Redes e Sistemas de Computadores
2015/2016, 2º Semestre

Trabalho Prático nº 2 (v1.0)

Resumo

Neste trabalho pretende-se desenvolver um mini-sistema de auditoria de segurança a sistemas de computadores, para detectar anomalias de integridade de instalações de software, de conformidade do sistema de ficheiros (incluindo verificação de conformidade de permissões de controlo de acesso), bem como verificação de integridade de configurações críticas. O sistema será constituído por um cliente de auditoria que executa operações remotas de verificação e atestação num agente de auditoria (servidor), sendo estas operações suportadas em Java RMI/SSL ou em Invocações Rest/WS suportadas em HTTPS.

1. Introdução

No sistema a implementar o cliente procede às verificações e atestações em qualquer sistema destino (por exemplo, servidores críticos que precisam de ser monitorados) correspondendo estas a invocações suportadas em JAVA-RMI/SSL, com proteção TLS (podendo alternativamente e opcionalmente serem suportadas em REST/HTTPS). O suporte TLS, deve ser construído de forma a garantir autenticação mútua cliente/servidor com base no suporte que permite implicitamente em SSL ou TLS suportar autenticação mútua com certificados de chave pública, independentemente de complementarmente poder ainda haver autenticação do utilizador com base em passwords, de modo a autenticar o utilizador-operador que está a fazer uma auditoria.

O sistema deve ser construído de forma a suportar parametrização de segurança configurável, permitindo evitar do lado do cliente a utilização de configurações de *ciphersuites* hoje consideradas fracas ou potencialmente fracas em TLS ou evitar versões menos seguras do protocolo SSL – por exemplo SSLv1, SSLv2, TLSv1, TLSv1.1). A configuração deve permitir que o *handshake* TLS possa ser desencadeado a partir do servidor (ou seja, sendo o servidor RMI ou SOAP/REST a desempenhar o papel de cliente no protocolo *handshake* da conexão TLS), de modo que seja o cliente de auditoria a determinar a configuração de segurança TLS que impõe ao servidor nas respostas às operações de auditoria remota.

Para o efeito deverá ser tido em conta o suporte de configuração adequada e detalhada dos *endpoints* de *sockets* TLS no suporte JSSE (em Java 7 ou Java 8), subjacentes ao suporte JAVA-RMI/TLS ou REST/HTTPS/TLS, tal como discutido em detalhe na aula prática (laboratório).

2. Operação do sistema

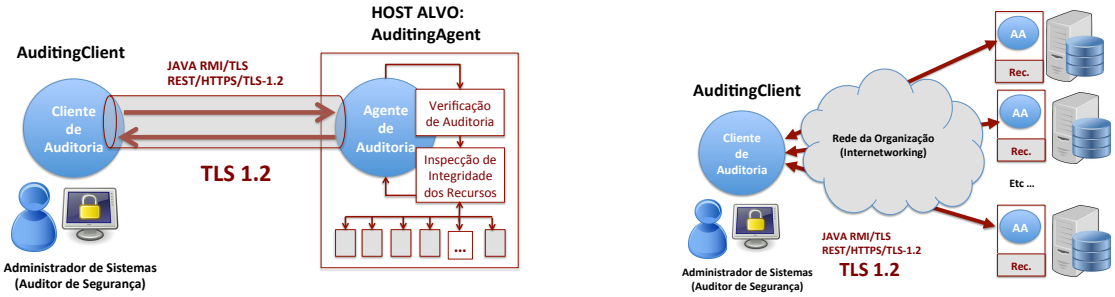
Os clientes têm em vista detectar remotamente situações anómalas, podendo estas resultar de eventuais intrusões ou ações maliciosas que tenham sido perpetradas ilicitamente por penetração nos sistemas alvo, mas também resultantes de operações anómalas ou erros humanos de utilizadores autorizados a operarem o sistema remoto, sejam estes conscientes ou inconscientes e que possam ter afectado o ambiente de execução ou as configurações que sejam considerados críticos do ponto de vista da sua atestação de autenticidade e integridade.

Como referência para o cenário e âmbito de operação, prova de conceito e demonstração do sistema a implementar, considera-se que os servidores a auditar são computadores semelhante sãos instalados nos ambientes dos laboratórios do DI/FCT/UNL a executar em ambiente LINUX (ex., distribuições Ubuntu ou Debian) e que iriam ser auditados a partir de estações de monitorização remota num Centro SOC (*Security Operation Center*) possivelmente instalado como subsector da Divisão de Informática da FCT/UNL. No caso concreto o cliente de auditoria poderia permitir verificar todas as instalações do Campus da FCT, como componente de uma estratégia mais alargada de um sistema de detecção de intrusões (*Intrusion Detection System*).

O sistema na sua funcionalidade base (associada aos requisitos obrigatórios - FASE 1) fará inspeções de integridade ao nível do File System, podendo acrescentar-se (como funcionalidade valorizável – FASE 2) outras operações de auditoria, podendo mesmo estas serem propostas pelos alunos.

3. Modelo e arquitetura do sistema

Para uma descrição mais detalhada deve consultar o Enunciado-Versão Estendida, com as especificações mais detalhadas (ficheiro SRSC-TP2-Enunciado-Detalhe.pdf) bem como a versão da Apresentação PPT (ficheiro SRSC-TP2.pdf).



Arquitetura e componentes do sistema: o cliente de auditoria (AuditingClient) faz pedidos de verificações a sistemas alvo de auditoria (onde executa um agende de auditoria - AuditingAgent) que inspeciona o sistema, devolvendo provas de integridade de recursos tal como solicitado, podendo estes envolver recursos do sistema operativo, configurações críticas a auditar recursos de comunicação ou estado de execução do sistema e aplicações. O objetivo é o cliente detectar estados anómalos que podem resultar de incorreções da operação do sistema ou resultantes de operação incorreta ou resultados de ataques por intrusão, por comparação com provas de integridade de estados corretos, por si conhecidos.

Operação do cliente (auditingclient)

O cliente obterá as provas de verificação por interações JAVA/RMI/TLS ou REST/HTTPS/TLS, com base em parametrizações de detalhe das propriedades de segurança a impor para o protocolo TLS.

Os requisitos para tais parametrizações pode ser visto no enunciado de detalhe.

O cliente fará as seguintes auditorias

Funcionalidade para FASE F1 (obrigatório: 3,5/5 pontos)

a) Java auditingclient -target <host> -nonce <nonce> -checkdir <path-diretoria>

Nesta caso o cliente executa testes de integridade na diretoria passada como argumento, mostrando se tudo está íntegro. As provas de integridade correspondem a computações de sínteses de segurança da diretoria e, no caso de anomalia, de cada um dos seus ficheiros, para se encontrar a origem da anomalia, de acordo com as imagens de teste de integridade. Estas provas serão calculadas e assinadas dinamicamente pelo servidor (agente) que executa no *target*, que as assinará para responder ao cliente, de acordo com a assinatura da resposta ao *nonce*. O *nonce* passado é iterado pelo cliente calculando um hash do mesmo concatenado com um valor aleatório (secure-random) e um *timestamp* e que corresponderá ao desafio (DESAFIO, enviado como *long integer* 64 bits) que o servidor de auditoria terá que iterar e incluir nas repostas a devolver ao cliente.

b) Java auditingclient -target <host> -nonce <nonce> -checkentry <path-diretoria/entry>

Idem, mas neste caso o cliente executa testes de integridade de uma entrada (ficheiro) da diretoria passada como argumento.

c) Java auditingclient -target <host> -nonce <nonce> -checkdirall <path-diretoria>

Idem, mas neste caso o cliente executa testes de integridade da diretoria indicada no argumento, obtendo uma lista das provas de integridade de todas as entradas

Funcionalidade para FASE F2.1 (valorativo, 1,5/5 pontos)

d) Java auditingclient -target <url> -check https-security

Neste caso o cliente será capaz de detectar se o target, admitindo que o mesmo tem um servidor WEB HTTPS disponível no url considerado, de modo a testar vulnerabilidades SSL ou TLS, nomeadamente:

- D1) Se está a admitir *handshakes* com versões de protocolo consideradas fracas, exemplo, SSLv1, SSLv2, TLSv1.1
- D2) Se está a admitir *ciphersuites* consideradas fracas, semi-fracas ou intermédias

- D3) Se o servidor está a enviar cadeias de certificação com certificados expirados ou revogados (em algum certificado da cadeia que envia no *handshake* do protocolo TLS)

O teste desta funcionalidade deverá ser feito não sobre o servidor que vai implementar no trabalho, mas sobre um servidor HTTPS. Obviamente, nesta opção do cliente, este funcionará para esta opção com configurações próprias, para que pudesse realizar este teste de *vulnerability-scanning* sobre qualquer servidor HTTPS.

- Autenticação TLS unilateral do servidor, apenas com autenticação do lado do servidor
- O teste de vulnerabilidade implica que o cliente fará tentativas de *scanning* com todas as *ciphersuites*, de modo a verificar se o servidor as aceita
- O *handshake* será sempre iniciado pelo cliente

Funcionalidade para FASE F2.2 (valorativo, 0,5/5 pontos de Bónus)

e) java auditingclient -target <url> -check heartbleed

Esta auditoria investiga se um servidor HTTPS está vulnerável por um potencial ataque do tipo ***Heartbleed***[1]

Valorização adicional do trabalho

Os alunos podem acrescentar outras auditorias de segurança que considerem relevantes (em adição à funcionalidade pedida), podendo vir a cobrir outras verificações ao nível de outros recursos, exemplo: verificação de padrões anómalos de tráfego no perímetro do sistema alvo de auditoria, verificações anómalas de logging de componentes Honeypot, logging relevante do sistema, informação de reporte sobre padrões anómalos de operação do sistema, etc. Para qualquer uma destas valorizações, sugere-se que os grupos interessados discutam as suas ideias com os docentes.

4. Entrega do trabalho

A entrega do trabalho deverá ser feita de acordo com os prazos indicados (de acordo com a informação no sistema CLIP). Envolverá a entrega do pacote de implementação com o código e respetivo relatório, nomeadamente: FASE 1 (Requisitos Obrigatórios) e FASE 2 (Requisitos Valorativos)

O formato do relatório bem como as instruções de entrega (que será feira por submissão por via electrónica), estão detalhados em documentos (*templates*) próprios.

Verificar os seguintes documentos no pacote do TP2:

- Instruções-Entrega-TP2
- Template-Relatório-TP2

[1] http://heartbleed.com, http://www.javatester.org/othertesters.html