Segurança em Engenharia de Software

Pedro Fernando Moreira Silva Antunes
Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
Porto, Portugal
up201507254@edu.fc.up.pt

Sílvia Andreia Ribeiro Maia
Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
Porto, Portugal
up201605209@edu.fc.up.pt

I. Introdução

Este trabalho está inserido na unidade curricular de Segurança em Engenharia de Software do Mestrado de Segurança Informática da Faculdade de Ciências da Universidade do Porto. Vamos abordar conceitos de Segurança em Engenharia de Software, como forma de assegurar a segurança do sistema, dos intervenientes do sistema e da comunicação entre os intervenientes e o sistema. O nosso objetivo foi perceber de que forma é que se constrói o design, o planeamento e a implementação de software destacando a segurança como primitiva para cada etapa.

Para isso iremos construir um software numa arquitetura cliente-servidor e os clientes podem comunicar com os outros clientes do servidor numa arquitetura cliente-cliente. Os clientes têm de efetuar um pré registo presencial onde são validados os dados nacionais do utilizador (como o nome, identificação nacional, etc). Posto isto, os clientes podem então registar uma conta de utilizador e autenticar-se no servidor. O servidor fornece serviços de acordo com o nível de autorização de cada utilizador, ou seja, não permite a utilização de alguns serviços aos utilizador que não estão autorizados para a utilização de serviços com elevadas permissões de acesso.

O serviço principal é a comunicação cliente-cliente para realizar trocas de mensagens. Primeiramente através do servidor, os clientes podem pedir uma lista de clientes, escolhem um cliente da lista e iniciam uma comunicação cliente-cliente com o cliente escolhido.

Começaremos o artigo com a apresentação da ideia do software, bem como os requisitos de segurança. Depois, iremos apresentar o design e a arquitetura do sistema. De seguida, relatamos como é que o software será implementado e por fim descrevemos a fase de testagem da implementação, a análise ao trabalho realizado e aos resultados obtidos.

II. Descrição do sistema

O objetivo do trabalho é construir um software com prevenções e respostas enquadradas em políticas de segurança, modelações de ameaças e avaliações de risco. Para a construção do software é necessário desenhar e implementar um sistema com um servidor web confiável que fornece serviços às aplicações do cliente.

A aplicação cliente irá ser utilizada pelos seus colaboradores e será uma interface no terminal que acede aos recursos desenvolvidos no servidor. Para o canal de comunicação dos serviços provenientes do servidor utilizaremos a tecnologia REST(REpresentational State Transfer), ou seja, cada recurso do servidor web é acedido através de uma solicitação através do URI(Uniform Resource Identifier) que provoca uma resposta formatada em JSON(JavaScript Object Notation).

Cada recurso do lado do servidor tem um nível de segurança associado e os clientes só conseguem aceder a cada recurso se tiverem sido associados com um nível de segurança igual ou superior ao nível de segurança do recurso pedido.

Posto isto, os colaboradores podem enviar e receber mensagens de outros colaboradores utilizando para este efeito a sua aplicação cliente (o servidor não envia mensagens produzidas por um cliente para outro).

A. Requisitos de Segurança

As políticas de segurança impostas estão sobre uma estratégia de segurança Informática de forma a garantirmos:

- ***Confidencialidade:***
  Asseguramos que a informação confidencial não é divulgada a indivíduos não autorizados e que os indivíduos autorizados controlam e influenciam a informação que é recolhido e armazenada.
- ***Integridade:***
  Asseguramos que a informação e os programas são alterados apenas de um forma específica e autorizada, e asseguramos que o sistema executa as suas operações de forma irrepreensível e sem ser influenciado por qualquer indivíduo não autorizado.
- ***Disponibilidade:***
  Garantimos que o sistema funciona prontamente e que o serviço não é negado a indivíduos autorizados.
- ***Autenticidade:***
  A propriedade de um indivíduo ser genuíno e de poder ser verificado e confiado. A avaliação da validade de confiança de uma transmissão, de uma mensagem e do seu autor.
- ***Responsabilização (não-repúdio):***
  O objetivo que gera a exigência de que as ações de um indivíduo autorizado sejam rastreadas de forma única até esse indivíduo. Assim garantimos o não-repúdio das mensagens, o isolamento de falhas que possam ocorrer e recuperação das mesmas. O sistema mantém registos de atividades para detetar violações de segurança e para possibilitar uma análise forense no caso desta ser necessária.

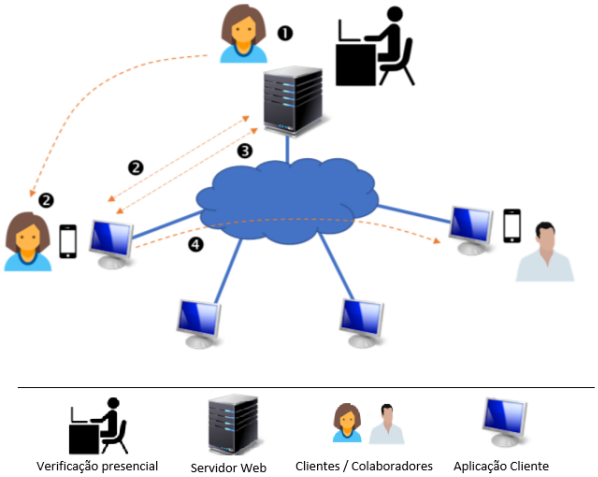

Fig. 1. Ilustração do fluxo de comunicação

Para um colaborador poder ser integrado no nosso sistema deve cumprir certos requisitos de segurança:

1) ***Pré-registo presencial (Fig.1 , Ponto 1)****:* O colaborador tem de realizar um registo presencial onde fornece os dados referentes à sua identidade e o serviço presencial tem de ser capaz de verificar se os dados fornecidos são validados. Em caso afirmativo, o colaborador está identificado e é-lhe fornecido um nome de utilizador temporário(tamanho máximo de 12 caracteres) e um identificador temporário(aleatório com 12 caracteres, com letras pequenas, com letras grandes e com dígitos). Por fim, é-lhe associado um nível de segurança entre 1 e 3.

2) ***Registo e Autenticação no sistema (Fig.1 , Ponto 2)****:* O colaborador descarrega e instala a aplicação cliente e a partir disto faz o registo no servidor utilizando os dados temporários provenientes do pré-registo (nome de utilizador e identificador único). Neste processo de registo, o colaborador escolhe um novo nome de utilizador e os restantes dados necessário para o registo de uma conta. O servidor valida os dados do colaborador e define os meios criptográficos referentes ao colaborador de forma a garantir confidencialidade, integridade, autenticidade e não-repúdio. Para garantirmos disponibilidade dos serviços prestados pelo servidor, a aplicação cliente é responsável por identificar e autenticar o colaborador localmente, e só em caso de sucesso é que autentica o colaborador no servidor.

3) ***Invocar serviços disponíveis no servidor (Fig.1 , Ponto 3)****:* O colaborador pode invocar os serviços disponíveis no servidor caso a autorização aos mesmos seja confirmada pelo veracidade do maior ou igual nível de segurança imposto ao cliente em relação ao nível de segurança imposto ao serviço.

4) ***Enviar e receber mensagens de um colaborador para outro (Fig.1 , Ponto 4)****:* Como serviço adicional, o colaborador pode pedir uma lista de todos os utilizadores com o nível de segurança menor ou igual ao seu. A partir do servidor, este serviço fornece os dados que estão nos níveis de seguranças iguais ou inferiores ao pedido do colaborador. Estão são os nomes do utilizadores, os meios para que a comunicação entre os colaboradores seja segura, estados, endereços ips públicos e portas atualizados. Posto isto, as mensagens são produzidas e enviadas a partir da aplicação cliente para a aplicação cliente requerida do respetivo colaborador. Assim, as mensagens nunca serão enviadas através do servidor, o servidor apenas funciona como um pré-requisito para que essa comunicação seja possível.

5) ***Toda a informação que circula na rede entre clientes e servidor****:* A informação a circular tem de ter garantias de confidencialidade e integridade. Cada extremo de uma comunicação deve estar identificado e autenticado e cada mensagem recebida deve ter meios de verificação não-repúdio e responsabilização.

6) ***As mensagens enviadas ou recebidas pelos colaboradores devem ser guardadas****:* As mensagens devem ser guardadas localmente no dispositivo do colaborador, de forma confidencial e cifrada. Devem existir meios para que possamos decifrar as mensagens guardadas e identificar qualquer tipo de alteração e garantir a verificação de não-repúdio e responsabilização.

III. Design e Arquitetura do Sistema

Para efetuarmos o design e a arquitetura do nosso sistema, temos de ter presentes conceitos como a superfície de ataque, os princípios de design e mecanismos de segurança. Na Fig. 1, identificamos o diagrama de fluxo da comunicação do nosso sistema. Através das políticas de segurança, podemos definir que a nossa superfície de ataque vai estar influenciada por ataques sobre a rede por toda a Internet (visto que as comunicações com o servidor web passaram pelos routers subjacentes à Internet). Por ataques sobre o hardware devido a possíveis vulnerabilidades existentes nos componentes físicos do servidor. Por ataques sobre o software pelas possíveis vulnerabilidades instauradas no sistema operativo e nas aplicações instaladas nele. Por ataques sobre os dados em função de entradas dos utilizadores no sistema ou entradas na base de dados. Por ataques de humanos através de pessoas não confiáveis que têm acesso direto ou indireto ao local físico do sistema.

A. Modelação de ameaças e vulnerabilidades

Nesta secção deveriam estar presentes diagramas de fluxos de dados, contudo não foi possível realizar este tipo de diagramas devido a falta de tempo. No entanto, como trabalho futuro iremos realizar os diagramas de fluxo de dados(DFD) através de ferramentas de modelação de ameaças tais como o ”Pytm”. O ”Pytm” é um projeto desenvolvido pela OWASP que segue a modelação de ameaças de segurança STRIDE(Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) e a metodologia de avaliação de risco DREAD(Damage Potential, Reproducibility, Exploitability, Affected users, Dicoverability). O ”Pytm” consegue automatizar e gerar o processo da construção de DFDs, Diagramas Sequênciais e Diagramas de Ameaças para o nosso sistema.

Após uma análise teórica do nosso sistema, identificamos as ameaças de ataques com maior risco para o nosso sistema:

- ***Side-channels Attacks***
  Estes ataques exploram aspetos da arquitetura do sistema para retirar informação adicional sobre os dados. A princípal característica destes ataques é que eles são bastante difíceis de prevenir e mitigar porque exigem uma análise estatística para provar comportamentos do sistema e calcular os dados privados em função dos comportamentos.
- ***Buffer Overflow***
  Enquadrado na violação da segurança da memória, o buffer overflow é provocado por entradas dos utilizadores. Este ataque é causado por re-escrever a memória através da ultrapassagem dos limites de um buffer. Provoca alterações no funcionamento do programa de modo a colocar toda a nossa segurança de dados e de recursos em risco.
- ***SQL Injection***
  É uma ameaça de segurança provocada pelas entradas dos utilizadores que ataca as interações com a Base de Dados através da inserção de instruções SQL maliciosas dentro de uma consulta provenientes de entradas BD. Pode provocar modificações ou eliminações nos dados privados.
- ***Cross-Site Scripting (XSS)***
  Este ataque é um tipo de injeção em servidores web. É provocado pela aplicação web sobre a forma de um script lateral que contém código malicioso e que tem como alvo um utilizador final diferente. O utilizador diferente pensa que o script é confiável por vir do servidor de confiança e no entanto, estará a ser infetado. Depois de infetado, as informações partilhadas com o servidor web estarão expostas pondo em causa os controlos de acesso ao servidor.

B. Princípios de design

Após as modelações de ataques e definições de segurança, foi feito o desenho do sistema. Para garantirmos que cada utilizador da aplicação cliente esteja identificado e que é único, passamos o utilizador por um processo presencial onde verificamos físicamente a pessoa em questão e as correspondentes informações identificadoras governamentais.

Depois da comunicação presencial estar assegurada, o utilizador pode registar-se no sistema informático com os dados gerados pelo passo anterior (utilizador e identificador único temporários correspondentes a um pré-registo presencial). Se o registo for bem sucedido, o cliente gera certificados de chave pública e partilha o certificado e a chave privada utilizando o protocolo de transferência de ficheiros seguro SCP(Secure Copy). Com estes certificados o cliente e o servidor têm os meios criptográficos necessários para estabelecerem conexões entre si seguras.

Depois do registo informático estar concluído, a aplicação cliente é responsável por verificar e autenticar a identidade dos clientes localmente antes de iniciar qualquer comunicação. Com isto estamos a prevenir tipos de ataques de negação de serviços e não confiámos em ninguém que não tenha sido verificado com acesso ao sistema. Desde que esteja registado no sistema, o utilizador garantidamente está identificado e responsabilizado pelas suas ações.

Os registos e os pré-registos são guardados numa Base de Dados em que apenas o servidor tem acesso. As autenticações são verificadas através da palavra-chave que estam concatenadas com números aleatórios (salt) armazenadas na base de dados por meios criptográficos. Em caso da autenticação ser sucedida, o servidor gera um token de sessão para o utilizador. Isto garante-nos uma autenticação de dois fatores.

Qualquer pedido de serviço ao servidor tem de vir proveniente de um utilizador com uma token de sessão válida e não pode haver utilizadores com mais do que uma sessão. Depois de ser validada a sessão do utilizador, o serviço é controlado por permissões de acesso. Um utilizador só poderá obter o serviço requerido se tiver permissões superiores ou iguais às permissões do serviço em questão. Assim garantimos que nenhum utilizador autenticado mas não autorizado tenha acesso ao recurso.

Qualquer comunicação proveniente da aplicação cliente com destino ao servidor é estabelecida através do protocolo HTTPS(Hyper Text Transfer Protocol Secure) utilizando certificados verificados por uma CA(Certification Authority) confiável. Ou seja está a ser transportada por camadas de seguranças por protocolos como o TLS(Transport Layer Security), que garantem confidencialidade e integridade na informação que circula pela rede até chegar ao servidor.

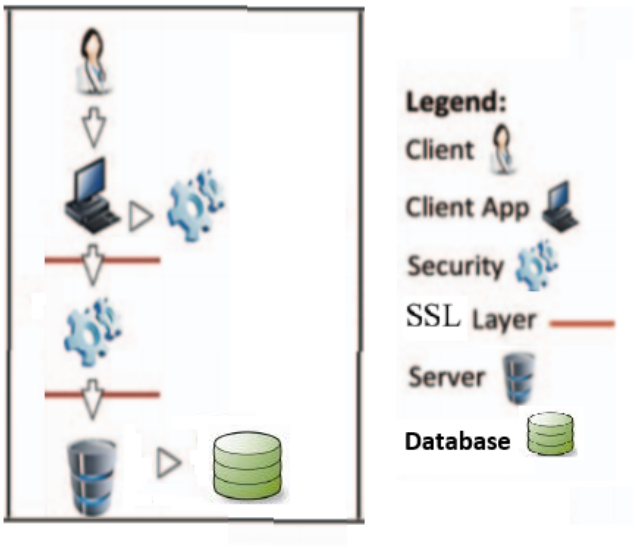

Fig. 2. Comunicação cliente-servidor

Qualquer comunicação entre aplicações cliente é estabelecida por sockets com o auxílio de terceiros, através do protocolo SSL(Secure Sockets Layer), que utiliza os certificados gerados pelo servidor e que autenticam os utilizadores do nosso sistema. O SSL encapsula o tráfego da comunicação dando-nos garantias de confidencialidade, integridade, disponibilidade e não-repúdio. Cada aplicação cliente pode pedir ao servidor os certificados referentes ao extremo do utilizador que quer estabelecer a comunicação, e com isto, pode verificar a autenticidade e decifrar/cifrar a informação que quer comunicar.

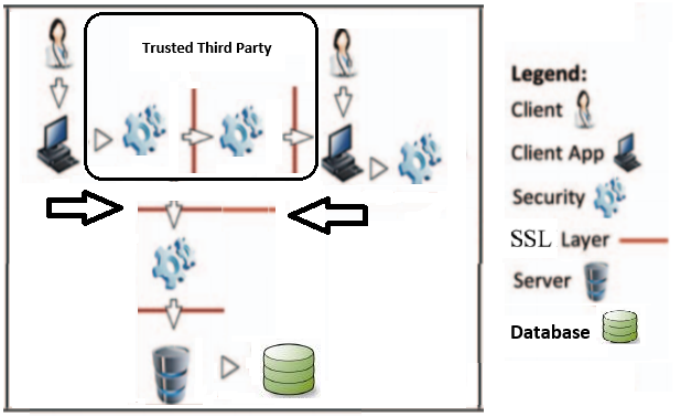

Fig. 3. Comunicação cliente-cliente

Para armazenar as mensagens no dispositivo local do colaborador confidencialmente iremos utilizar criptografia simétrica onde a chave de cifrar ficheiros será a chave pública fornecida pelo servidor. Para decifrar e detetar alterações dos ficheiros utilizamos a chave privada fornecida igualmente pelo servidor. Esta chave privada é gerada pelo algoritmo matemático RSA e é protegida localmente por cifrações por palavra-chave em que a palavra-chave é a palavra-passe utilizada na altura do registo no servidor.

C. Mecanismos de Segurança

Como mecanismos de segurança utilizamos cifração, assinaturas digitais, integridade dos dados, autenticação mútua, firewalls e controlos de acessos, uso de uma TTP(Trusted Third Party) e relatórios de segurança para detetar, relatar e recuperar de possíveis ataques.

D. Mitigações e Contra-medidas

Para os ataques de buffer overflow podemos tentar prevenir inserindo valores aleatórios extras na pilha do programa e podemos criar formas de que os endereços de memória de retorno estejam protegidas fora da pilha do programa.

Para os ataques de SQL injection podemos verificar os inputs provenientes do utilizador através de técnicas de normalização, sanitização e validação de modelos de expressões regulares.

Para os ataques de side-channels podemos tornar aleatório a utilização dos endereços de memória e a utilização de semáforos para tentar esconder os fluxos e tempos de execução de cada funcionalidade.

Para os ataques de Cross-Site Scripting podemos fazer uma análise através de testes de penetração, análise estatística da implementação e tentar mitigar algumas públicas vulnerabilidades comuns da comunidade de segurança de CVEs (Common Vulnerability Exploit).

IV. Implementação

A implementação do sistema será feita em função de tecnologias para auxiliar o desenvolvimento. O desenvolvimento será todo feito na linguagem de programação Python, o web server em Nginx com uma framework de desenvolvimento web chamada de Flask, desenvolvida em Python.

Para a implementação da infraestrutura de chaves públicas, a CA escolhida tem de ser confiável(desde que tenhamos um domínio registado, conseguimos adquirir uma CA confiável), e foi utilizado o protocolo SSL para gerar e verificar uma cadeia de certificados confiável. O desenvolvimento dos sockets é também em Python e são usados em paralelo com o SSL para que os certificados de cada utilizador esteja presente em todas as comunicações.

V. Testagem - Mitigações e Prevenções de ataques conhecidos

Para a testagem da segurança do nosso sistema iremos implementar testes de entradas do utilizador para que possamos fazer uma análise estatística de ameaças encontradas e dos consequentes riscos que o nosso sistema tem.

A. Fuzz Testing

Fuzzing é uma técnica de testes de software, frequentemente automatizada ou semi-automatizada, que envolve prever inválidos, inesperados e aleatórios dados como entradas para programas. O programa é então monitorizado, analisando exceções como erros em tempo de execução. Este é um tipo de teste open-source desenvolvido pela Google e que está disponível para a linguagem Python através do módulo ”fuzzing”. Este módulo será utilizado para abordarmos A relização dos testes não foram possíveis de realizar devido a falta de tempo. No entanto, como trabalho futuro iremos utilizar o módulo ”fuzzing” do python para tirarmos conclusões dos testes ao nosso sistema.

B. Análise

Ao analisarmos o design do nosso sistema, conseguimos argumentar que conseguimos prevenir várias ameaças de possíveis ataques e garantimos que o nosso sistema defende ataques como man-in-the-middle, utiliza firewalls para prevenir ataques de negação de serviço, cifração nas comunicações, autenticidade dos cliente por password e tokens evitando ataques de repetição, através de infraestruturas de chave pública garantimos que as mensagens têm autoria e não-repúdio.

No entanto, não temos dados estatísticos e testes efetuados para garantirmos analisamos suficientemente o nosso sistema

para termos um nível de confiança aceitável. Muitas vezes a fase de testes e análise levam à correção e prevenção de várias vulnerabilidades que possam escapar no início do planeamento de um software.

As garantias apresentadas estão sendo apresentadas assumindo que não existem vulnerabilidades que se saiba até agora pelas tecnologias utilizadas na arquitetura do projeto. Caso surja alguma vulnerabilidade imposta em alguma destas tecnologias, a segurança do nosso sistema também ficará em causa.

VI. Conclusões e trabalhos futuros

Começamos este artigo por identificar o sistema e os requisitos de segurança. Vimos de forma geral a comunicação e os intervenientes da comunicação. Depois conseguimos definir a superfície de ataque do sistema. Posto isto, identificamos algumas ameaças ao nosso sistema e a partir disto definimos o design e os mecanismos de segurança que temos de ter para o sistema. Apresentamos as tecnologias a ser utilizadas na implementação e tentamos analisar algumas mitigações e contra-medidas que devemos ter na implementação.

Como existem vulnerabilidades a cada dia, temos de estar constantemente a atualizar o nosso conhecimento de ameaças e vulnerabilidades encontradas pela comunidade de segurança para que possamos detetar qualquer falha que possa colocar o nosso sistema em risco.

Como trabalhos futuros, tencionamos desenvolver os testes fuzzing para que possamos ter os nossos desenhos e conceitos de segurança testados por testes para esse efeito. Também tencionamos aprimorar ainda mais o nosso desenho e arquitetura do sistema acrescentando diagramas de fluxos de dados e diagramas de ameaças e riscos para o sistema.

References

[1] Common Vulnerabilities Exploitation, URL https://cve.mitre.org/
[2] Threat Modelling: Designing for Security, Adam Shostack, Wiley, 2014
[3] Threat Modelling: A Pratical Guide for Development Teams, Izar Tarandach Matthew Coles, O'Reilly, 2020
[4] Risk Centric Threat Modelling, PASTA, Tony Velez, Marco Morana, Wiley, 2015
[5] Secure Software Lifecycle, CyBok chapter by Laurie Williams, 2019
[6] Software Security, CyBok chapter by Frank Piessens, 2019
[7] US CyberSecurity and Infrastructure Security Agency, URL, https://us-cert.cisa.gov/
[8] Fuzzing Forum, URL, https://github.com/google/fuzzing
[9] Software Assurance Maturity Model, URL, https://www.opensamm.org/
[10] The Open Web Application Security Project, URL, https://owasp.org/