NetProbe

Gabriel Almeida Portela
825233281@ulife.com.br
Universidade São Judas Tadeu
Ciência da Computação

Daniel Almeida Portela
825234443@ulife.com.br
Universidade São Judas Tadeu
Ciência da Computação

1 Introdução

1.1 Tema

Este software é sobre segurança, eficiência, e confiabilidade em redes de computadores—as fundações onde toda solução digital moderna é construída. E tem em mente o objetivo 9 da ONU, que visa construir infraestruturas resilientes, promover a industrialização inclusiva e sustentável, e fomentar a inovação.

1.2 Objetivo

Este software tem como objetivo desenvolver um analisador de tráfego de rede que seja capaz de monitorar o tráfego de rede em tempo real, e tem como publico-alvo estudantes de ciência/engenharia da computação, entusiastas de redes, e educadores. O software terá um backend em C++ e sem o uso de bibliotecas externas, com o fim de proporcionar uma experiência de aprendizado mais simples e prática. Também será dividido em três etapas principais: captura de pacotes, análise de pacotes, e visualização de dados por meio de uma dashboard web.

1.3 Escopo Principal

1.3.1 Atores do Sistema

- **Usuário**: Entusiasta de redes que irá se registrar, autenticar e utilizar o sistema para monitorar e analisar o tráfego da rede local.

1.3.2 Casos de Uso

1.3.2.1 Cadastrar-se no Sistema

- **Código**: UC01
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário não possui uma conta de acesso ao sistema.
- **Gatilho**: O usuário acessa a página inicial e decide criar uma nova conta para acessar o dashboard.
- **Cenário de Sucesso Principal**: **1.** O usuário acessa a interface web do sistema. **2.** O sistema exibe a página de login e apresenta uma opção para "Cadastro". **3.** O usuário seleciona a opção de cadastro. **4.** O sistema exibe um formulário solicitando informações como e-mail e senha. **5.** O usuário preenche o formulário com dados válidos e o submete. **6.** O sistema valida os dados, cria a conta de usuário e armazena as informações de forma segura. **7.** O sistema redireciona o usuário para a página de login com uma mensagem de sucesso, indicando que o cadastro foi concluído.
- **Exceções**: **6a.** Se o e-mail fornecido já estiver cadastrado, o sistema exibe uma mensagem de erro informando que o usuário já existe. **6b.** Se os dados fornecidos forem inválidos (ex: formato de e-

mail incorreto, senha fora do padrão exigido), o sistema exibe uma mensagem de erro e solicita a correção.

1.3.2.2 Autenticar-se no Sistema (Login)

- **Código**: UC02
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário deve ter uma conta previamente cadastrada no sistema.
- **Gatilho**: O usuário deseja acessar o dashboard de monitoramento de rede.
- **Cenário de Sucesso Principal**: **1.** O usuário acessa a página de login do sistema. **2.** O usuário insere suas credenciais (e-mail e senha) nos campos correspondentes. **3.** O usuário submete o formulário de login. **4.** O sistema valida as credenciais. **5.** Após a autenticação bem-sucedida, o sistema concede acesso e exibe o dashboard principal de monitoramento de rede.
- **Exceções**: **4a.** Se as credenciais estiverem incorretas, o sistema exibe uma mensagem de "Usuário ou senha inválidos" e permite que o usuário tente novamente.

1.3.2.3 Monitorar Tráfego de Rede em Tempo Real

- **Código**: UC03
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário está autenticado e na página do dashboard web. O backend em C++ está ativo e capturando pacotes da rede local.
- **Gatilho**: O usuário acessa o dashboard para visualizar a atividade da rede.
- **Cenário de Sucesso Principal**: **1.** Após o login, o usuário visualiza o dashboard principal. **2.** O backend em C++, integrado ao frontend via WebAssembly, captura os pacotes da rede. **3.** O dashboard exibe uma tabela ou lista que é atualizada em tempo real com os pacotes capturados. **4.** Para cada pacote, o sistema exibe informações essenciais como: endereço IP de origem, endereço IP de destino, porta de origem, porta de destino, protocolo (TCP, UDP, ICMP, etc.) e o volume de dados (tamanho do pacote).
- **Exceções**: **3a.** Caso o backend em C++ não consiga capturar o tráfego (ex: falta de permissões), o dashboard exibe uma mensagem de erro informando sobre a falha na captura de pacotes.

1.3.2.4 Filtrar Tráfego de Rede

- **Código**: UC04
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário está autenticado e visualizando o tráfego em tempo real no dashboard.
- **Gatilho**: O usuário deseja isolar e visualizar pacotes específicos para uma análise mais focada.
- **Cenário de Sucesso Principal**: **1.** No dashboard, o usuário localiza os campos ou a seção de filtros. **2.** O usuário preenche um ou mais critérios de filtro, como: protocolo (ex: "HTTP"), endereço IP específico (origem ou destino) ou número de porta. **3.** O usuário aplica o filtro. **4.** O sistema processa a regra de filtro e atualiza a exibição do tráfego, mostrando apenas os pacotes que correspondem aos critérios definidos. **5.** O usuário tem a opção de modificar ou limpar os filtros para retornar à visualização completa.
- **Exceções**: **4a.** Se o usuário inserir um valor de filtro inválido (ex: um endereço IP mal formatado), o sistema exibe uma notificação de erro e não aplica o filtro até que seja corrigido.

1.3.2.5 Analisar Pacote de Dados

- **Código**: UC05
- **Ator Principal**: Usuário
- **Pré-condições**: O usuário está autenticado e visualizando a lista de pacotes no dashboard.
- **Gatilho**: O usuário identifica um pacote de interesse e deseja inspecionar seus detalhes técnicos.
- **Cenário de Sucesso Principal**: **1.** O usuário seleciona (clica em) um pacote específico na lista de tráfego. **2.** O sistema exibe uma visualização detalhada do pacote selecionado em uma janela modal ou em um painel lateral. **3.** Esta visualização apresenta o conteúdo decodificado do pacote, dividido por camadas, como o cabeçalho Ethernet, cabeçalho IP, cabeçalho do protocolo de transporte (TCP/UDP) e a carga útil (payload) dos dados. **4.** O usuário analisa as informações detalhadas para entender a natureza da comunicação. **5.** O usuário fecha a visualização de detalhes para retornar à lista principal de tráfego.
- **Exceções**: **3a.** Se o conteúdo do pacote estiver criptografado ou em um formato que não possa ser decodificado, o sistema exibirá as informações que conseguir interpretar (como os cabeçalhos) e indicará que a carga útil não é legível.

1.4 Fatores de Qualidade

Os fatores de qualidade de McCall é um modelo que fornece uma estrutura para inspecionar e garantir a qualidade de um produto de software. O modelo separa qualidade em 3 categorias—Operação, Revisão e Transição do produto—essas categorias são dividias em 11 fatores de qualidades. Essa seção applica o modelo de qualidade de McCall para o software NetProbe.

1.4.1 Operação do Produto

1.4.1.1 Correção (Correctness)

- O sistema exibe corretamente os IPs de origem e destino, portas e protocolos de cada pacote?
- A função de cadastro e login operam exatamente como especificado nos requisitos (RF001 a RF004)?
- Os filtros aplicados pelo usuário retornam com precisão apenas os pacotes que correspondem aos critérios (filtros)?
- A análise de pacotes decodifica e exibe corretamente os dados dos cabeçalhos do respectivo pacote?

1.4.1.2 Confiabilidade (Reliability)

- O backend em C++ consegue capturar pacotes continuamente por longos períodos (horas ou dias) sem falhar?
- O sistema tem um jeito “gracioso” de lidar com erros, como a perda de acesso à interface de rede, informando o usuário sem travar?
- A conexão via WebAssembly entre o frontend e o backend é estável?

1.4.1.3 Eficiência (Efficiency)

- O backend em C++ tem baixo consumo de CPU e memória para não impactar o desempenho da máquina onde está rodando? (Fundamental para uma ferramenta de monitoramento).
- A atualização em tempo real do dashboard é otimizada para consumir poucos recursos do navegador?
- A aplicação de filtros é processada rapidamente, mesmo com um grande volume de pacotes sendo exibido?

1.4.1.4 Integridade (Integrity)

- O sistema impede que usuários não autenticados acessem o dashboard?
- As senhas dos usuários são armazenadas de forma segura (criptografadas)?
- O sistema garante que a captura de tráfego não abre brechas de segurança no sistema hospedeiro (*host machine*)?

1.4.1.5 Usabilidade (Usability)

- A interface do dashboard é clara e intuitiva para um "entusiasta de redes"?
- É fácil para o usuário encontrar e aplicar os filtros desejados?
- A visualização dos dados de um pacote é detalhada, organizada e fácil de ler?

1.4.2 Revisão do Produto

1.4.2.1 Manutenibilidade (Maintainability)

- O código-fonte é bem estruturado, comentado e segue padrões de programação?
- É fácil para um novo desenvolvedor entender a lógica de captura e processamento de pacotes no backend?
- Se um bug for encontrado na renderização do dashboard, quão rápido ele pode ser identificado e resolvido?

1.4.2.2 Flexibilidade (Flexibility)

- Qual seria a dificuldade para adicionar um novo critério de filtro (e.g. por tamanho do pacote)?
- Se no futuro for decidio adicionar a funcionalidade de "Gerar Relatórios" (que foi explicitamente excluída), a arquitetura atual facilita essa adição?
- Quão fácil é modificar a interface para suportar um novo tipo de gráfico ou visualização?

1.4.2.3 Testabilidade (Testability)

- O backend em C++ pode ser testado de forma automatizada, talvez usando arquivos de captura (`.pcap`) como entrada?
- Os componentes do frontend são isolados, permitindo a criação de testes unitários para a interface?
- Existem procedimentos claros para realizar testes de ponta a ponta (end-to-end), simulando o login, a aplicação de um filtro e a análise de um pacote?

1.4.3 Transição do Produto

1.4.3.1 Portabilidade (Portability)

- O backend em C++ pode ser compilado e executado em diferentes sistemas operacionais com pouca ou nenhuma modificação?
- O frontend web, por sua natureza, já é altamente portátil, mas ele funciona corretamente em todos os navegadores modernos?

1.4.3.2 Reusabilidade (Reusability)

- O módulo de autenticação de usuários poderia ser reutilizado em outro sistema web?
- A biblioteca C++ de captura de pacotes foi projetada de forma que possa ser usada como base para outra ferramenta de rede?

1.4.3.3 Interoperabilidade (Interoperability)

- O sistema poderia, futuramente, exportar os dados capturados em um formato padrão (como CSV ou JSON) para que possam ser importados por outras ferramentas de análise?
- Seria possível integrar o sistema com uma API externa para, por exemplo, verificar se um IP de origem está em uma lista de ameaças conhecidas (blacklist)?

2 Requisitos do Sistema de Software

2.1 Requisitos Funcionais

2.1.1 Autenticação e Gerenciamento de Usuários

- **RF001**: O sistema deve permitir que novos usuários se cadastrem fornecendo um e-mail e uma senha.
- **RF002**: O sistema deve permitir que usuários cadastrados sejam autenticados utilizando seu e-mail e senha.
- **RF003**: O sistema deve permitir que um usuário autenticado encerre sua sessão de forma segura.
- **RF004**: O sistema deve garantir que apenas usuários autenticados possam acessar as páginas do dashboard de visualização de dados.

2.1.2 Monitoramento de Tráfego

- **RF005**: O backend em C++ deve capturar pacotes da rede local em tempo real.
- **RF006**: O sistema deve processar os pacotes capturados para extrair informações relevantes.
- **RF007**: O backend deve enviar os dados processados para o frontend para visualização.
- **RF008**: O dashboard web deve exibir os dados de tráfego de rede recebidos do backend de forma dinâmica.
- **RF009**: Os dados na interface devem ser apresentados de maneira clara e organizada.

2.1.3 Filtragem de Tráfego

- **RF010**: O sistema deve fornecer ao usuário a capacidade de filtrar os dados de tráfego exibidos no dashboard.

2.1.4 Análise de Pacotes

- **RF011**: O sistema deve permitir que o usuário inspecione os detalhes de um pacote específico.
- **RF012**: O sistema deve detectar e informar ao usuário sobre erros que impeçam a captura de dados.

2.2 Requisitos Não-Funcionais

2.2.1 Desempenho

- **RNF001**: O backend em C++ deve capturar e processar pacotes com baixa latência com o intuito de garantir a visualização dos dados em tempo real.
- **RNF002**: A *UI* (Interface do Usuário) do dashboard web deve ser responsiva a atualizar os dados de tráfego de forma fluida, ou seja, sem travamentos.
- **RNF003**: A filtragem de dados exibidos pelo dashboard web deve produzir resultados de forma ideal—rápido, e suave ou harmonizada.

2.2.2 Segurança

- **RNF004**: As senhas dos usuários cadastrados devem ser guardadas de forma segura no banco de dados, utilizando técnicas e algoritmos avançados de criptografia.
- **RNF005**: A comunicação entre o frontend e o backend deve ser segura, especialmente durante o processo de autenticação.
- **RNF006**: O acesso ao dashboard web para visualização de dados deve ser estritamente controlado por sessão de usuário autenticada.

2.2.3 Usabilidade

- **RNF007**: A *UI* do dashboard web deve ser intuitiva e de fácil compreensão para usuários com conhecimento em redes.

- **RNF008**: Os dados de tráfego da rede devem ser apresentados de forma clara e organizada para fácil visualização.
- **RNF009**: A filtragem de dados e a inspeção de pacotes devem ser de fácil acesso e utilização.

2.2.4 Compatibilidade

- **RNF010**: O dashboard web deve ser compatível com as versões mais recentes dos navegadores mais utilizados no mercado (e.g. Google Chrome, Mozilla Firefox, Brave).
- **RNF011**: O backend em C++ deve ser multiplataforma, ou seja, compilável para o Kernel dos sistemas operacionais mais utilizados (e.g. Unix-based, Windows).

2.2.5 Confiabilidade

- **RNF012**: O sistema de captura de dados (backend) deve operar de forma estável e contínua, sem interrupções inesperadas.
- **RNF013**: O sistema deve apresentar *Exception handling*—i.e. lidar de forma adequada com possíveis erros de captura de dados da rede (e.g. falta de permissão na interface de rede), informando o usuário sobre o problema.

2.2.6 Tecnologia

- **RNF014**: O backend do sistema deve ser desenvolvido em C++ para garantir alto desempenho na manipulação de pacotes.
- **RNF015**: A integração e comunicação entre o frontend e o backend em C++ deve ser realizada utilizando WebAssembly.