Concórdia
Serviço Local de Troca de Mensagens

Segurança de Sistemas Informáticos, LEI, UMinho, 2023-24

1 Descrição geral (obrigatório, 100% da cotação)

Pretende-se desenvolver um serviço de conversação entre utilizadores locais de um sistema Linux. A este respeito, considere que os utilizadores já existem no sistema operativo (o serviço não terá necessidade de os criar). Na realização do trabalho, tenha como referência uma distribuição que permita, se necessário, utilizar listas estendidas de controlo de acesso (por exemplo, distribuições baseadas em Debian ou Ubuntu).

1.1 Aspetos funcionais

Do ponto de vista funcional, o serviço deverá suportar o envio de mensagens para um utilizador, bem como a sua leitura pelo respetivo destinatário.

O serviço deverá suportar comunicação assíncrona, como no caso do mail. Por uma questão de simplicidade, sugere-se que considere que o tamanho das mensagens nunca excede os 512 caracteres.

O serviço deverá também suportar a gestão de utilizadores do serviço. Em concreto, deverá permitir a adição e remoção de utilizadores (já existentes no sistema operativo).

Deverá também suportar a noção de grupos privados de conversação. Nesse sentido, deverá oferecer mecanismos para criação e remoçao de grupos, e de gestão dos seus membros. Considere que um grupo é gerido exclusivamente pelo utilizador que o criou. Ou seja, só ele pode adicionar ou remover novos membros e só ele poder remover o próprio grupo. Contudo, todos os seus membros podem enviar e receber mensagens dentro do grupo. Por uma questão de simplicidade, a resposta é sempre enviada para todo o grupo. Um utilizador que seja removido de um grupo perderá imediatamente acesso ao conjunto de mensagens prévias e futuramente trocadas no grupo.

1.2 Aspetos de segurança

Tendo em conta que o foco desta unidade curricular é em tecnologias da segurança (no sentido lato), as equipas deverão ter como preocupação central a proteção da confidencialidade e integridade das mensagens trocadas entre os utilizadores, bem como, obviamente, a disponibilidade do próprio serviço.

No sentido de exercitar os mecanismos de controlo de acesso do Linux, as equipas deverão suportar – pelo menos em parte – o seu modelo de segurança no modelo de segurança do próprio sistema operativo. Ou seja, deverão ter em conta: as noções de utilizador e grupo de utilizadores Linux; as permissões definidas para cada objeto do sistema de ficheiros necessário ao funcionamento do serviço; as noções de processo e a eventual execução de programas com os privilégios associados ao utilizador ou grupo donos do mesmo.

A este respeito, sugere-se o estudo da abordagem ao desenvolvimento seguro do `qmail`, um agente de transferência de mail (MTA), desenvolvido por Daniel J. Bernstein, entre 1995 e 1998. Nele poderão encontrar um exemplo de uma arquitetura modular focada na segurança, em que componentes de software executam como processos com acesso restrito aos recursos do sistema.

2 Aspetos de valorização (opcional, bonificação)

No desenho deste serviço, cada equipa poderá querer suportar aspetos funcionais adicionais que considere relevantes, porventura em consequência de decisões relativas ao modelo e arquitetura de

segurança adotados no seu desenho e desenvolvimento.

Nesse sentido, as equipas poderão, por exemplo, querer integrar o serviço com a infraestrutura de registo de eventos (syslog[1]) e/ou a de gestão de serviços do sistema (systemctl[2]), ou ainda, poderão querer (re)confirmar a identidade de um utilizador, exigindo, por exemplo, a introdução de um código de autenticação OTP (oauthtool[3]).

3 Observações e sugestões

A descrição dos aspetos funcionais e de segurança do serviço permite, intencionalmente, o desenho e a implementação de soluções muito diferentes. Nesse sentido, a equipa deverá ter bem em conta que, tão ou mais importante do que a implementação do serviço, é o seu modelo e arquitetura de segurança.

Com vista a ajudar a perceber alguns dos aspetos que deverão ser endereçados no trabalho, seguem-se um conjunto de observações e de sugestões de pontos que a considerar no desenho da arquitetura e implementação do serviço.

Sem prejuízo de ajustes que os grupos poderão querer realizar, o serviço deverá oferecer o seguinte conjunto de programas de linha de comando:

`concordia-enviar dest msg`
: Enviar uma mensagem de texto para um destinatário. O destinatário poderá ser um utilizador individual ou um grupo do serviço.

`concordia-listar [-a]`
: Listar as novas mensagens de um utilizador. As mensagens serão apresentadas com um índice numérico de modo a possibilitar a resposta num momento posterior. Se o comando for invocado com o argumento `-a`, imprimará a totalidade das respostas recebidas. A listagem deverá incluir a data de receção, o remetente e o tamanho da mensagem.

`concordia-ler mid`
: Ler o conteúdo de uma mensagem identificada pelo seu índice numérico.

`concordia-responder mid msg`
: Responder ao remetente de uma mensagem previamente recebida e indicada pelo seu índice numérico.

`concordia-remover mid`.
: Apagar uma mensagem recebida indicada pelo seu índice numérico.

`concordia-grupo-criar nome`
: Criar um grupo de utilizadores do serviço.

`concordia-grupo-remover`
: Remover um grupo de utilizadores do serviço.

`concordia-grupo-listar`
: Listar os utilizadores membros de um grupo.

`concordia-grupo-destinario-adicionar uid`
: Adicionar um utilizador a um grupo.

`concordia-grupo-destinario-remover uid`
: Remover um utilizador de um grupo.

[1] Manual/tutorial de utilização da API C do serviço syslog: `https://ftp.gnu.org/old-gnu/Manuals/glibc-2.2.3/html_chapter/libc_18.html`.

[2] Tutorial de escrita de um serviço que integra com systemctl: `https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6`.

[3] Manual/tutorial de utilização da ferramenta oauthtool`https://manpages.ubuntu.com/manpages/jammy/man1/oathtool.1.html`.

`concordia-ativar`
: Adicionar o utilizador (que invoca o programa) ao serviço. O serviço criará as estruturas necessárias (diretorias, ficheiros, etc. . .) necessárias à interação do utilizador com o serviço.

`concordia-desativar`
: Remover o utilizador (que invoca o programa) do serviço. Por uma questão de simplicidade, as mensagens anteriormente recebidas também deverão ser removidas.

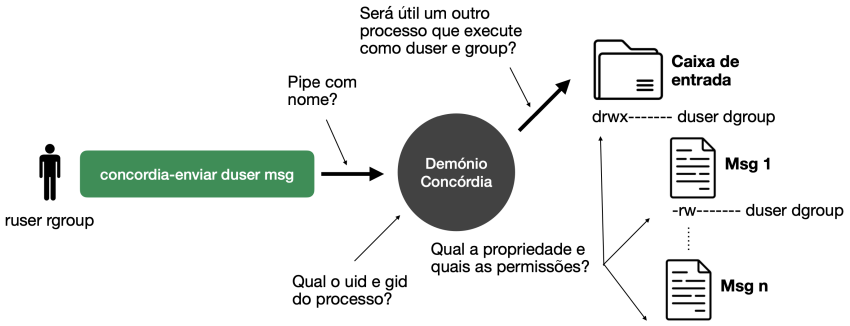
Os programas que acima estão apresentados como programas totalmente independentes. Contudo, caso faça sentido do ponto de vista da solução desenhada (especialmente em termos de segurança), poderá haver espaço para alguma consolidação dos mesmos. Por exemplo, poderá ser razoável considerar um programa `concordia-mensagem` que agregue a funcionalidade dos comendos `concordia-enviar`, `-listar`, `-ler`, `-responder`, e `-remover`.

Alguns aspectos adicionais que deverá endereçar:

- Como representar e armazenar as mensagens trocadas entre os utilizadores do serviço? Cada utilizador tem um ficheiro que representa a caixa de entrada das mensagens? Cada utilizador tem uma caixa de entrada estruturada como uma pasta contendo mensagens individuais sob a forma de ficheiros? Onde será guardada informação como a data de receção de uma mensagem, o seu destinatário, o tamanho, e se foi ou não lida? Como são geridos os índices que identificam as mensagens? Em todas estas questões, deverá ter em conta o controlo de acesso a estas estruturas, ou seja, quem é utilizador e grupo donos e quais as permissões a elas associadas.

- O serviço poderá ou não necessitar de uma componente *"demónio"*, um processo servidor que execute em pano de fundo, interagindo exclusivamente com os programas executados pelos utilizadores (e.g. `concordia-enviar`). Caso opte por um processo demónio, idealmente este deverá ser integrado na infraestrutura de gestão de serviço do sistema operativo (`systemctl`).

Segue-se um exemplo de uma reflexão inicial sobre o desenho de uma possível arquitetura do serviço. Neste exemplo, o serviço socorre-se de um demónio que recebe uma mensagem de um utilizador `ruser` destinada a um `duser`. No diagrama, chama-se a atenção para algumas das questões que deverão ser tidas em conta. Sublinha-se, mais uma vez, que este é apenas um exemplo das opções que poderão ser consideradas em termos de desenho da arquitetura do serviço.



4 Critérios de avaliação e a sua ponderação

Relatório (40-50%): O relatório, deverá ser preparado em formato PDF, em A4, 11pt, espaçamento simples. Não deverá ter mais do que 10 páginas (sem capas e índices), podendo incluir outra documentação numa secção opcional de anexos. Deverá ser estruturado da seguinte forma:

- A secção de introdução descreverá os contornos da solução desenhada, as suas preocupações e decisões arquiteturais mais relevantes.

- A secção de arquitetura funcional descreve os programas, os processos envolvidos na operação do serviço, aspetos de interoperabilidade (por exemplo, formatos e APIs internas, e mecanismos de comunicação), e, por último, as componentes de sofware desenvolvidas e eventuais dependências de componentes. Deverá ainda identifiar e descrever as estruturas de dados necessárias ou funcionamento do serviço. Estes aspetos arquiteturais deverão ser ilustrados com base em diagramas simples mas rigorosos e completos, de modo a facilitar a compreensão das suas ideias.
- Partindo da arquitetura funcional, uma nova secção deverá descrever e explicar agora as decisões tomadas no domínio da segurança do serviço. Por exemplo, quem são os donos e quais as permissões definidas em cada objeto do sistema de ficheiros, bem como dos processos necessários à execução do serviço. Se necessário, os diagramas usados nesta secção poderão omitir alguns detalhes apresentados na secção anterior (por exemplo, os formatos de dados) e/ou expandi-los com as anotações necessárias.
- Uma secção de reflexão justificará detalhadamente os aspetos funcionais e, em particular, de segurança que considere mais relevante. Nomeadamente, a necessidade do conjunto particular de utilizadores e dos grupos de utilizador do sistema, permissões bem como outros mecanismos complementares que possam ter sido empregues. Deverá também explicar eventuais aspetos de modularidade e encapsulamento das componentes de software desenvolvidas, que ferramentas foram utilizadas para identificar problemas no código dos programas (por exemplo, Valgrind, flags de compilação, estratégia de testes, etc. . .). Se for conveniente esta secção poderá anteceder ou ser agregada à anterior.
- Uma secção de conclusão resume não só os pontos fundamentais das decisões tomadas, identifica as eventuais limitações e formas de as superar.

Implementação (20-35%): Deverá ser entregue uma Makefile, com objetivos (*targets*) para a compilação, teste de componentes (se aplicável) e instalação do serviço. Deverá também ser fornecido um README com instruções relevantes, identificando, eventualmente, dependências de componentes externas que devem ser instaladas ou que terão de ser compiladas (se código-fonte incluído), bem como, exemplos de utilização dos vários programas do serviço. Deverá ser também disponibilizado todo o código-fonte e outros ficheiros necessários à instalação e utilização do serviço.

Discussão (30%): Em sessão presencial com a equipa docente a equipa apresentará a solução desenhada e desenvolvida, focando a atenção nos aspetos de segurança do serviço. Deverá ser capaz de demonstrar todo o processo de compilação, de teste (se aplicável), de instalação e de operação do serviço. Relembra-se que, sendo o trabalho desenvolvido em grupo, a avaliação é, no entanto, individual. Todos membros da equipa deverão ser capazes de participar na discussão do trabalho e justificar todas as decisões tomadas.

Valorizações: Os aspetos de valorização, opcionais, funcionarão como bonificação sobre a nota base. Dito de outra forma, somam-se à classificação relativa aos requisitos obrigatórios do trabalho. A bonificação não poderá ultrapassar os três valores e, naturalmente, a nota final (base + bonificação) não poderá ultrapassar os 20 valores[4].

[4]FAQ: 1) Posso ter uma avaliação de 20 valores se só tratar os requisitos obrigatórios do trabalho? Sim, claro. 2) Então a valorização só serve para bonificar o trabalho? Sim, é isso.