Universidade de Aveiro

Mecanismos de Proteção de Software

Departamento de Eletrónica, Telecomunicações e Informática

Discentes: Rafael José da Silva Martins, Ricardo Jorge Martins Lucas

Disciplina: Segurança Informática nas Organizações

Docentes: João Paulo Barraca e Hélder Gomes

Licenciatura em Engenharia Informática

29 de Dezembro de 2015

Mecanismos de Proteção de Software

Relatório de Segurança Informática e nas Organizações da Licenciatura em Engenharia Informática da Universidade de Aveiro, realizado por Rafael Martins, Ricardo Lucas.

Resumo

Um aspeto muito importante do desenvolvimento de aplicações é o controlo de integridade e o controlo de execução das aplicações desenvolvidas. Considerando que qualquer desenvolvimento possui um custo não desprezável é simples considerar que muitos autores pretendem disponibilizar as suas aplicações de forma não gratuita.

Considerando esta situação e todos os mecanismos que estão incorporados nas aplicações para garantir uma distribuição fidedigna a um cliente foi-nos proposta a construção de um sistema/mecanismo diferente, embora com objectivos similares, que facultasse uma distribuição entre autor e cliente de uma forma mais eficiente e segura.

Para a construção deste sistema foi então necessário desenvolver alguns elementos importantes, uns integrados na própria aplicação e outros apenas para o autor, que ajudassem o autor a manter uma rede de distribuição de aplicações segura e de certa forma confidencial.

Estes elementos vão ser descritos ao longo deste relatório tal como todos os aspectos de segurança que estão presentes no sistema. É também descrito todo o processo de funcionamento geral o inicio de uma distribuição até á validação de uma licença válida de utilização da aplicação.

Conteúdo

Lista de figuras

Capitulo 1

Introdução

Problemas de distribuição de programas pagos são bastante relevantes e importantes aquando o desenvolvimento de uma aplicação. Um programador tem de ter em conta a forma de distribuição da aplicação que está a construir de forma a oferecer ao cliente um programa consistente e seguro. Ou seja, é necessário existir um controlo de versões, de licenças e uma protecção contra ataques.

Contudo, todas as aplicações são diferentes e servem objectivos diferentes, daí existir um grande investimento por parte de quem quer vender uma aplicação para garantir esta não é utilizada de forma errada/maliciosa por parte de outros.

A nossa solução foi implementar (tal como proposto) alguns mecanismos diferentes para protecção da aplicação. Tais como o Cartão de Cidadão para autenticidade tanto do cliente como do autor e mecanismos de hardware para autenticidade da máquina/computador.

Contexto

Este projecto foi desenvolvido no contexto na disciplina de Segurança Informática e nas Organizações com o intuito de fornecer um novo tipo de segurança e protecção para autores de aplicações.

Motivação

A motivação para este projecto foi o facto de podermos criar um programa para que os programadores possam partilhar o seu trabalho de forma segura. Sendo nós programadores conseguimos ter uma visão interior dos requisitos necessários para uma partilha de software e dai termos em atenção alguns aspectos relevantes e importantes de autenticidade e segurança.

Objetivos

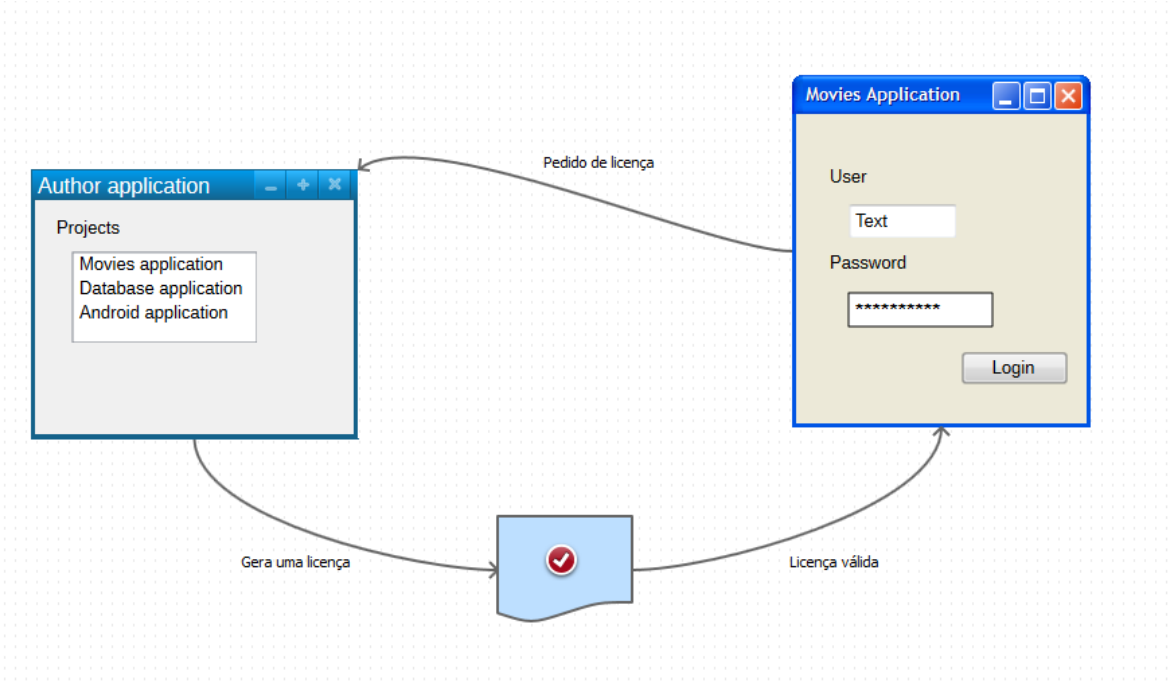

Fig. 1: Arquitectura geral

Com o intuito de alcançar o objectivo final nós criamos 2 componentes importantes, uma aplicação de gestão para o autor e uma biblioteca que contém os métodos necessários para que o cliente consiga gerar um pedido de licença para a utilização da aplicação criada pelo autor.

Para trabalharmos de forma coerente criamos um projecto no code.ua e um repositório git para podermos trabalhar de forma consistente. No code.ua criamos também uma lista de tarefas e traçamos algumas datas e objectivos importantes, desta forma poderíamos acompanhar o trabalho de ambos sem ser necessário estarmos juntos.

Capitulo 2

Arquitetura geral do sistema

O nosso software funciona de uma forma bastante linear e consistente. Tudo começa no lado do autor, este quando pretende fazer uma distribuição do seu software gera um ficheiro zip com o software e com a sua chave pública do cartão de cidadão. Depois disto o zip é enviado para o cliente alvo.

Quando o cliente recebe o zip vai descompacta-lo e retirar o software e a chave publica. Assim que este executar o software a biblioteca interna vai automaticamente começar as suas verificações para ver se este está registado e caso não esteja então vai iniciar o processo de registo. A biblioteca vai buscar os dados de hardware do PC do cliente, vai buscar os dados da aplicação e do cartão de cidadão do cliente. Depois trata de encriptar estes dados e assiná-los. Deste processo resulta outro ficheiro zip que contém alguns ficheiros importantes para o autor.

Depois desse zip ser enviado para o autor este vai ter de os validar e criar uma licença válida para utilização do seu software registada para aquele utilizador. De seguida a licença é enviada para o cliente que ao coloca-la na pasta no software este já vai executar sem problemas. Damos também ao cliente alguma liberdade de troca de dispositivos de hardware, nomeadamente 3. Caso este altere mais que 3 dispositivos a licença deixa de ser válida.

Ambiente de desenvolvimento

O nosso projecto foi desenvolvido em Java no NetBeans IDE e em Windows, é de salientar que o projecto apenas funciona em ambientes Windows devido á forma como o software foi construído.

Requisitos para utilização

- Java 8;
- SmartCard Reader;
- Cartão de Cidadão;
- Windows 7 64 bits;

Elementos do sistema

Aplicação do autor

A nossa aplicação do autor é um simples software de gestão para permitir ao autor uma gestão dinâmica da distribuição dos seus projectos. Consiste num conjunto de ecrãs com autenticação inicial feita pelo Cartão de Cidadão.

Biblioteca

Esta biblioteca é constituída por vários packages que contêm vários módulos com classes que nós criamos conforme a nossa necessidade. Um desses packages contém uma classe principal (securityaplication.AppBible.java) com 4 métodos que foram criados como requisito da construção do projecto que são:

- void init(string nomeDaApp, string versão);
- bool isRegistered();
- bool startRegistration();
- void showLicenseInfo().

Estes métodos são os que vão ser chamados na quando a aplicação que queremos proteger é executada e são estes mesmos que vão fazer toda o processo de registo e verificação de licenças. É de relevar que o método init(…) corresponde ao construtor da nossa classe.

Todos os módulos restantes foram criados consoante a nossa necessidade de desenvolvimento, estes são:

- User:
  - ClientInfo.java
- Hardware:
  - Bios.java
  - Cpu.java
  - HardDrive.java
  - Motherboard.java
  - Network.java
  - OsInfo.java
  - HardwareRecognizer.java (módulo central que agrupa todos os acima descritos)
- Utilities:
  - CCProvider.java
  - ReadCC.java
  - SecurityTools.java
- Registration:
  - AplicationData.java
  - UserInformation.java

Foram também utilizadas algumas livrarias externas para ajudar em algumas partes do desenvolvimento. Estas são:

- Pteidlib.jar (cartão de cidadão);
- Bouncy Castle (criação de certificados);
- Jai_imageio-1.1.jar (tratamento da imagem do CC).

Aplicação a proteger

A aplicação que vamos proteger é uma aplicação de filmes que contém um login que dá acesso a uma área reservada por utilizador onde estes podem colocar os filmes que viram entre outras coisas.

É uma aplicação simples feita em java que serve apenas o propósito deste projecto.

Capitulo 3

Processos e conteúdos dos ficheiros de licença

Nesta parte vamos falar dos conteúdos do ficheiro de licença e do ficheiro de pedido de licença. Vamos explicar também como é feito o processo de criação de ambos os ficheiros.

Pedido de licença

O pedido de licença é um ficheiro xml que contém todos os dados necessários de registo do cliente e do seu hardware. Estes dados são retirados pela biblioteca com recurso a alguns mecanismos externos á aplicação.

No caso dos dados de hardware são criados alguns scripts em visual basic que vão buscar os dados e os retornam para o java, seguidamente é criado um digest destes resultados todos. Os dados do cliente são retirados do Cartão de Cidadão e os da aplicação são passados como argumentos.

Depois destes dados serem todos recolhidos são encriptados com uma chave simétrica criada apenas para este objectivo e de seguida são guardados num ficheiro xml. Este ficheiro é depois assinado pelo cliente usando o seu cartão de cidadão. Por sua vez a chave simétrica é encriptada com uma chave pública resultante de um par de chaves assimétricas que é enviada pelo autor aquando da partilha da aplicação.

Deste processo, no final, é gerado um ficheiro zip que contém quatro ficheiros importantes para o registo da aplicação. Estes são, o xml com os dados de hardware, cliente e aplicação, o ficheiro com o certificado, o ficheiro com a assinatura digital e o ficheiro da chave simétrica.

Finalizado este processo basta o cliente enviar o zip para o autor para este proceder ao registo.

Ficheiro de licença

O ficheiro de licença é criado pelo autor depois de este validar os dados do cliente a quem distribui-o a aplicação, este ficheiro contém apenas algumas informações necessárias para a posterior validação do lado do cliente. Estas informações são o nome do cliente, a validade da licença e as hash's criadas através de uma função digest dos dados de hardware do cliente.

Estas hash's são provenientes do ficheiro de pedido de registo e são depois utilizadas do lado do cliente para verificar se foi ou não modificado algum hardware no computador do cliente.

Integridade e autenticidade

Para garantir a integridade tanto da aplicação como da biblioteca são criados digests dos ficheiros de ambos e enviados na troca de licenças para serem validados e se estes não foram iguais então alguma das partes foi modificada.

Quanto à autenticidade esta é feita tanto do lado do cliente como do autor. Do lado do cliente como o ficheiro de pedido de licença é assinado conseguimos garantir a autenticidade deste antes do registo. Do lado do autor utilizamos o cartão de cidadão para que este tenha acesso ao software de gestão. São duas formas simples de autenticidade de ambas as partes.

Capitulo 4

Neste capítulo vamos falar dos mecanismos de segurança utilizado no nosso projecto. Vamos também explicar os algoritmos que escolhemos utilizar tal como a utilização do cartão de cidadão.

Mecanismos de segurança

Para atingir os nossos objectivos foi necessário fazer algumas pesquisas para vermos quais as melhores estratégias para assegurar um bom nível de segurança não só na transição dos ficheiros mas também dos dois lados independentes, cliente e autor.

Cifra mista

A utilização da cifra mista neste projecto vem como solução para a troca confidencial de mensagens entre autor e cliente. Consiste em usar cifras simétricas e assimétricas, nomeadamente uma simétrica para cifrar um texto e posteriormente uma chave assimétrica para cifrar a chave simétrica.

Ao utilizar este tipo de cifras garantimos uma distribuição de chaves segura e eficiente entre autor e cliente.

Keystore

No nosso projecto utilizamos também um keystore (chaveiro) do lado do autor para guardarmos a chave privada proveniente do par de chaves assimétrico utilizado na troca de mensagens entre ambas as partes.

Desta forma quando for necessário gerar a licença válida o autor apenas necessita de aceder ao keystore e retirar a chave privada para poder aceder aos dados do ficheiro de pedido de licença.

BouncyCastle

A BouncyCastle é uma API que nos fazer enumeras coisas e uma delas é gerar certificados e cadeias de certificados. No nosso projecto nós utilizamos esta API para criarmos uma cadeia de certificados ligada ao par de chaves assimétrico gerado para distribuição de chaves.

Esta cadeia de certificados é muito importante porque sem ela o autor não consegue guardar a chave privada do par de chaves assimétrico no seu keystore.

Chaves utilizadas no projecto

- **Chave de sessão**: chave simétrica que é usada para cifrar os dados do pedido de licença.

- **Par de chaves assimétricas RSA**: par de chaves utilizado para a distribuição de chaves entre ambas as partes. A pública é partilhada juntamente com a aplicação e a privada fica no keystore do autor.
- **Chave pessoal**: esta chave pessoal é a chave de acesso ao keystore do autor, cada contém uma chave pessoal diferente associada.

Algoritmos utilizados

Neste ponto vamos explicar o porque da utilização de alguns algoritmos criptográficos e também dos modos de cifra.

- **RSA**: é um algoritmo de criptografia muito conhecido e o mais utilizado na geração de chaves assimétricas e em assinaturas digitais. Escolhemos este algoritmo mesmo pelo facto de ser o mais utilizado e tirando o facto de conter algumas vulnerabilidades serve bem para o nosso propósito/objectivo sem causar problemas. Utilizações:
    - Geração do par de chaves assimétrica;
    - Cifra da chave de sessão simétrica.
- **DESede**: o DES é um algoritmo de cifra por blocos muito utilizado que usa chaves de 56 bits. Como utiliza chaves pequenas a sua criptanálise torna-se de certa forma fácil daí nós utilizarmos o triple DES (DESede) pois assim desta forma garantimos um reforço de segurança na cifra dos dados. Utilizações:
    - Geração da chave simétrica;
    - Cifra do xml de pedido de licença.
- **SHA256withRSA**: este algoritmo é utilizado na assinatura digital do documento xml de pedido de licença. O RSA vem apenas inserir mecanismos de controlo de integridade próprios que permitem validar se a assinatura é ou não um valor válido.
- **SHA-1**: é uma função de síntese que neste momento é a mais recomendada face ás vulnerabilidades existentes. Utilizações:
    - Digest dos dados de hardware do cliente.
- **SHA-1PNRG**:

Cartão de cidadão

No nosso projecto o cartão de cidadão tem um papel importante pois este está presente em quase todos os aspectos relacionados com a geração de licenças.

Lado do autor

Do lado do autor o cartão de cidadão é utilizado para efectuar o login dentro do nosso software de gestão de projectos e licenças. É utilizado o PIN de autenticação para, tal como o nome indica, autenticar o autor.

O CC é também utilizado para assinar o ficheiro de licença válido de forma a garantir a veracidade e legitimidade do autor que passou a licença. Para este efeito o autor tem também de inserir o seu PIN de assinatura.

O certificado de chave publica do CC é enviado para o cliente para que este possa comprovar a assinatura do ficheiro de licença.

Lado do cliente

Do lado do cliente o CC é utilizado para autenticação/validação do utilizador, isto é, se não for o utilizador que registou a aplicação então este não vai ter acesso ao software. É também utilizado para assinar digitalmente o ficheiro de pedido de licença, para que isto aconteça o cliente tem de colocar o seu PIN de assinatura digital.

Tal como no autor o certificado de chave pública do cliente vai junto com o xml do pedido para posterior validação da assinatura.

Capitulo 5

Neste capitulo vamos explicar a aplicação de autor com imagens e de certa forma com mais algum pormenor.

Aplicação do autor

Fig. 2: Janela de entrada

Assim que ligamos a aplicação aparece a janela indicada em cima e quando carregamos no login é nos pedido o PIN de autenticação do CC.

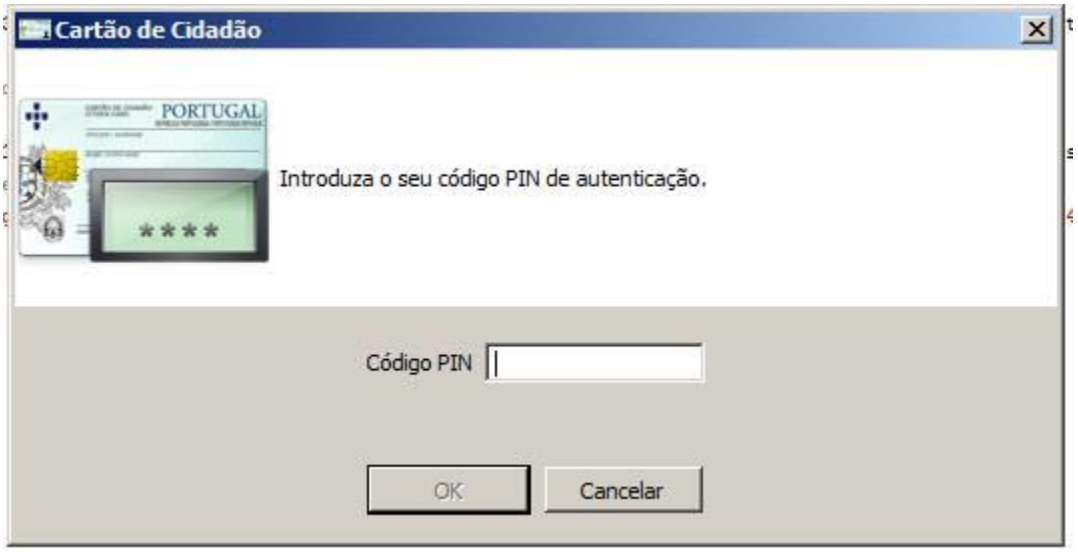

Fig. 3: Autenticação

De seguida, no fim do login, passamos para a janela dos projectos onde podemos criar novos projectos e novas licenças para estes.

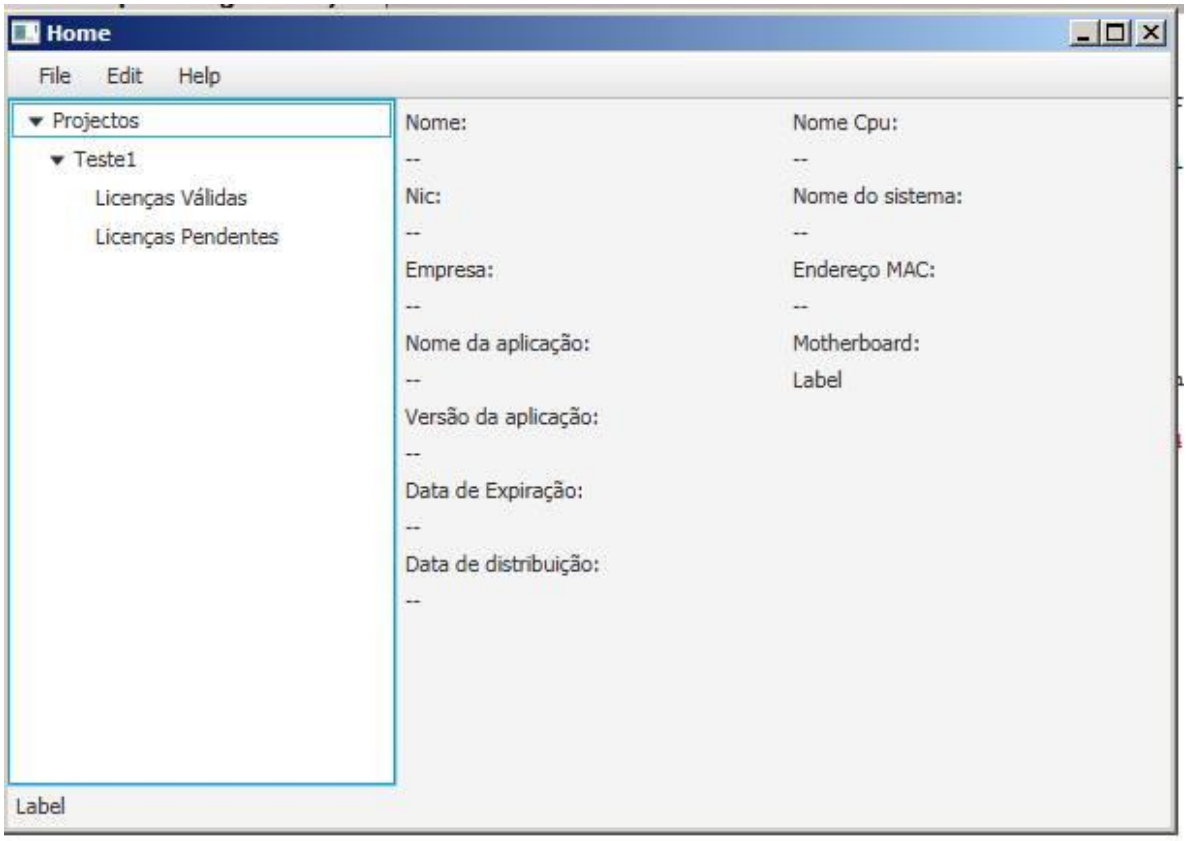

Fig. 4: Janela dos projetos

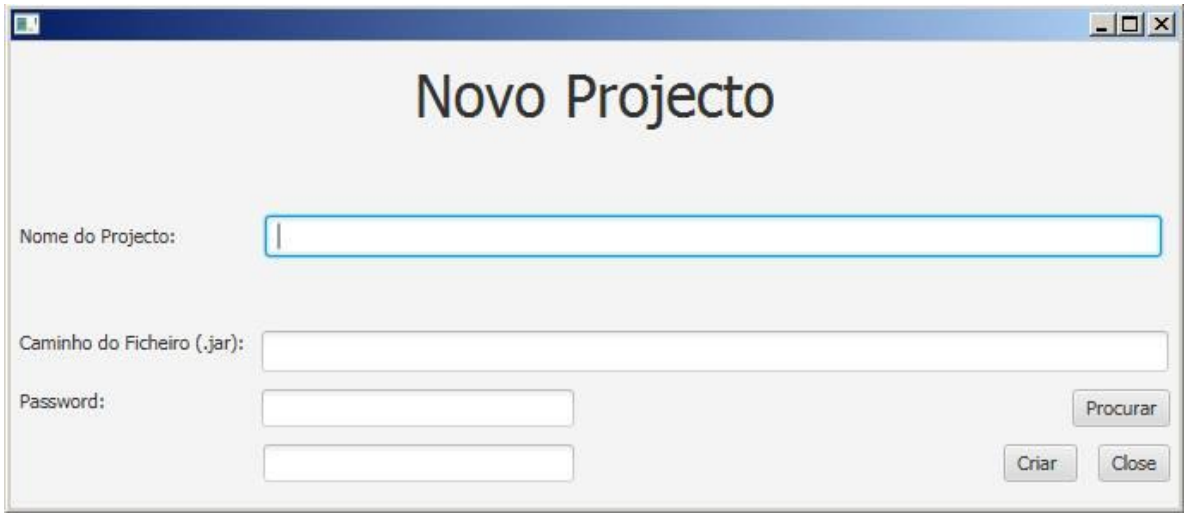

Fig. 5: Novo projecto

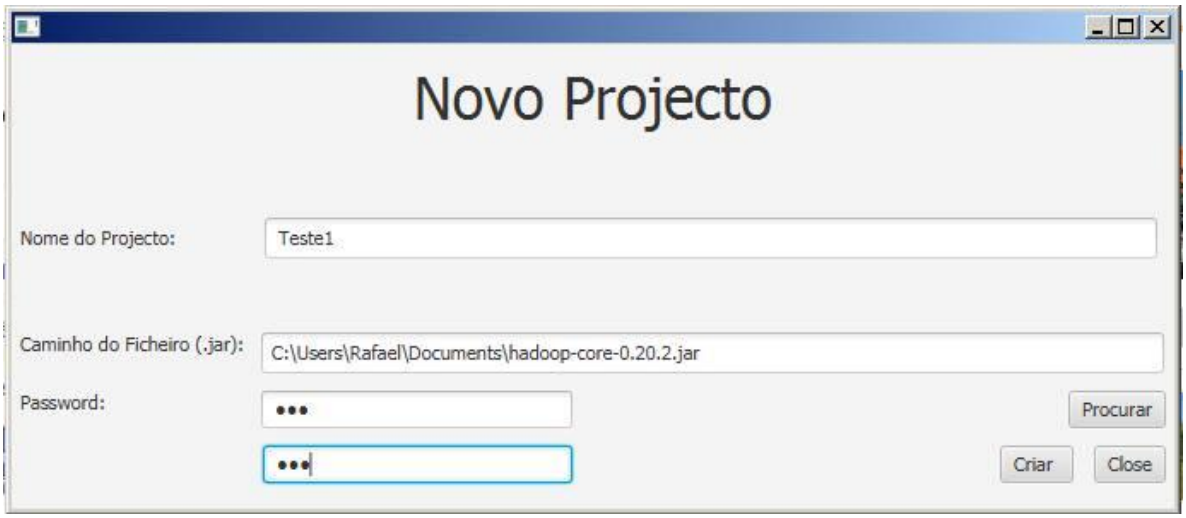

Fig. 6: Novo projecto com dados

Depois de ser criado um novo projecto podemos criar uma licença válida para uma distribuição previamente feita.

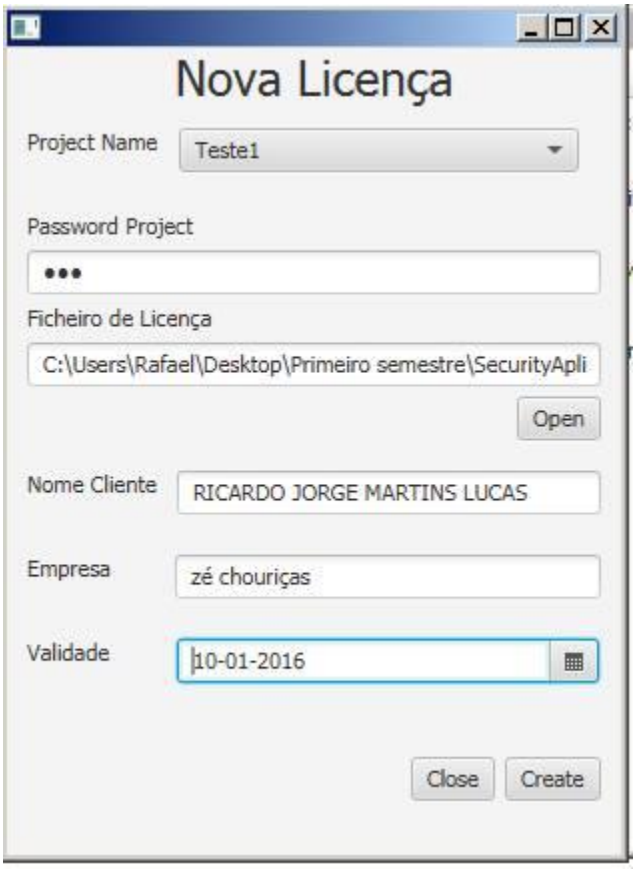

Fig. 7: Nova licença

Quando carregamos no botão “create” é nos pedido o PIN de assinatura para assinarmos a licença.

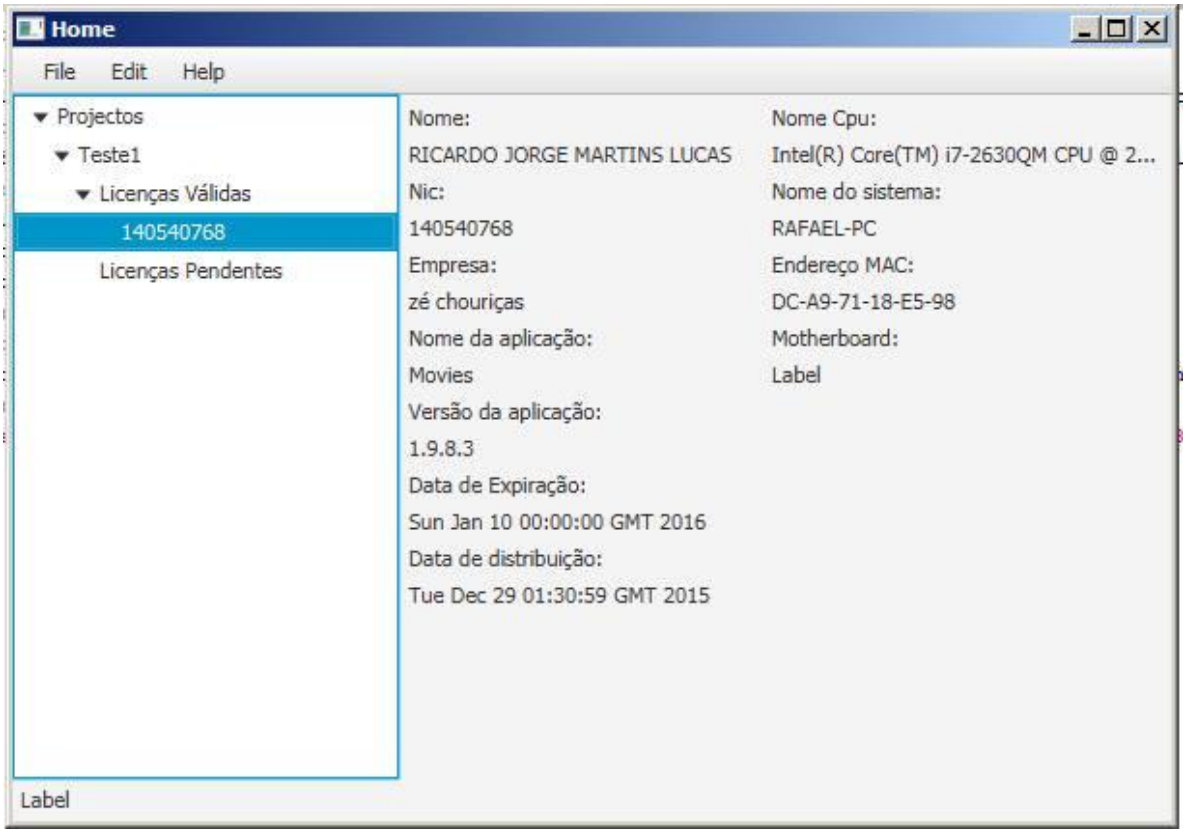

Fig. 8: Janela dos projectos com detalhes

Testes e protecção contra ataques

Criação de um pedido de registo com dados arbitrários

Caso seja criado um ficheiro de pedido de registo com dados arbitrários se o ficheiro de pedido não conter uma assinatura válida não vai ser possível criar uma licença válida.

Obtenção dos dados do pedido de registo

Se um atacante obter os dados de pedido de registo do cliente este não vai poder fazer nada pois como não tem a chave privada para decifrar a chave de sessão só vai ver dados encriptados.

Manipulação do pedido de registo, alterando campos chave

Caso o pedido de registo seja manipulado quando o autor for validar a assinatura dos dados a sua integridade não vai ser válida e como tal não vai ser passada uma nova licença válida.

Manipulação da biblioteca e aplicação de forma a evitar a verificação da licença

No caso de alguém por ventura conseguir aceder ao código e evitar a verificação da licença não existe nenhuma medida de segurança que não o deixe executar o software. Tornando este tipo de problema uma vulnerabilidade do sistema.

Obtenção dos dados da licença com vista a duplicação do sistema

Caso um atacante obtenha o ficheiro de licença é difícil poder utilizá-lo para executar no seu software, pois não tendo o cartão de cidadão da pessoa a quem se destina a licença a validação da licença logo no inicio da execução do software retorna um erro e não deixa o atacante abrir o software.

Capitulo 6

Trabalho futuro

Este projecto ainda contém arestas por limar e algumas possibilidades de serem inseridas novas funcionalidades, nomeadamente do lado do autor. A aplicação de gestão de projectos pode ser melhorada tendo uma BD por trás a guardar os dados dos projectos em vez de ser tudo guardado localmente num xml.

No que toca ao cartão de cidadão também existem algumas validações que podem ser adicionadas, no entanto, nós tivemos em conta alguns possíveis cenários de problemas envolvidos com a utilização do CC, mas claramente não todos.

Conclusão

Com a execução deste projecto pode-se observar que conseguimos alcançar os objectivos proposto inicialmente e até com algumas partes a mais que não eram pedidas. Concluímos então que a utilização do cartão de cidadão e o seu midleware são uma opção de trabalho nova que oferece um grau de segurança elevado através da sua integração em projectos de programação.

Bibliografia

ActiveExperts Software. (s.d.). *List of WMI*. Obtido de http://activexperts.com/admin/scripts/wmi/vbscript/

Agencia para a Modernização Administrativa. (s.d.). *SVN GOV*. Obtido de https://svn.gov.pt/projects/ccidadao

Bouncy Castle. (s.d.). *The Legion of the Bouncy Castle*. Obtido de https://www.bouncycastle.org/java.html

Zúquete, A. (2014). *Segurança em Redes Informáticas.* Aveiro: FCA.