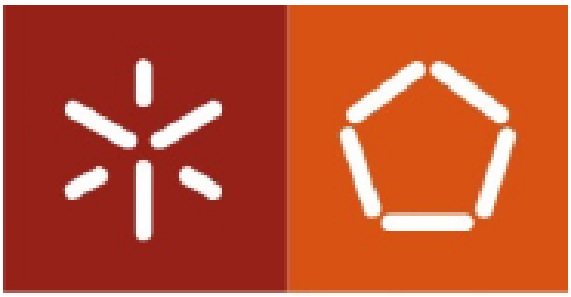

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Engenharia de Segurança

Aplicação CMD-SOAP

Adriana Meireles a82582

Carla Cruz a80564

July 6, 2020

Contents

1 Introdução

No âmbito da Unidade Curricular Engenharia de Segurança foi-nos proposto desenvolver, em ruby, uma aplicação comando linha que permita testar as operações do serviço SCMD, fazendo reverse engineer da aplicação fornecida pelo docente, **CMD-SOAP**.

Ao longo deste relatório iremos abordar a passagem dos ficheiros fornecidos em python para ruby,as técnicas para o desenvolvimento seguro de software e modo de testar o código desenvolvido. No fim é feita uma apreciação crítica.

2 Desenvolvimento

Para o desenvolvimento do Projeto 3 foi necessário basearmo-nos no que fora previamente aprendido com o Projeto 1 e Projeto 2, bem como, nas aulas teóricas desenvolvendo desta forma, em **Ruby**, uma aplicação comando linha (CLI) que nos permite testar as operações do serviço SCMD (Signature CMD), onde foi feito o reverse engineer da aplicação CMD-SOAP fornecida como exemplo.

2.1 CMD-SOAP - Teste das operações do serviço SCMD (Signature CMD)

Com a implementação deste projeto, é possível encontrar 3 ficheiros que permitem testar as operações do serviço SCMD (Signature CMD), de acordo com a versão 1.6 da "CMD - Especificação dos serviços de Assinatura", utilizando a linguagem de programação Ruby.

Encontramos destas forma os seguinte ficheiros:

- *cmd_soap_msg.rb* - Neste ficheiro é possível encontrar as funções que preparam e executam os comandos SOAP do SCMD, nomeadamente:
  - GetCertificate
  - CCMovelSign
  - CCMovelMultipleSign
  - ValidateOtp
- *cmd_config.rb* - Este ficheiro deve conter o ApplicationId fornecido pela AMA.
- *test_cmd_wsdl.rb* - Aqui é apresentado um Menu cheio de opções que nos permite testar vários comandos SOAP do SCMD.

2.2 Utilização da aplicação de testes

Quando se faz a execução de ruby test_cmd_wsdl é possível observar as várias opções da command line interface (CLI) da aplicação de testes.

Desta forma, com a utilização de um menu, é possível realizar várias opções, que são as seguintes:

```
[(master)⚡ % ruby test_cmd_wsdl.rb

############################ Command Line Program ############################
   1  Mostrar a Versão atual do Command Line Program
   2  GetCertificate
   3  CC Movel Sign
   4  CCMovelMultipleSign
   5  Validate OTP
   6  TestAll
   0  Fechar Programa

##############################################################################
Insira a sua Opção:
```

Figure 1: Menu inicial das funcionalidades

Com a apresentação deste Menu, podemos então realizar o seguinte:

1. Mostrar a versão atual do Command Line Program
2. GetCertificate - testa o comando SOAP GetCertificate do SCMD
3. CCMovelSign - testa o comando SOAP CCMovelSign do SCMD
4. CCMovelMultipleSign - testa o comando SOAP CCMovelMultipleSign do SCMD
5. ValidateOtp - testa o comando SOAP ValidateOtp do SCMD
6. TestAll - testa automaticamente a sequência de comandos GetCertificate, CCMovelSign e ValidateOtp, verificando no final a assinatura, baseado na assinatura recebida, na hash gerada e na chave pública do certificado recebido.

Por defeito fazemos sempre a utilização do serviço SCMD de produção.

2.3 Técnicas de Desenvolvimento Seguro de Software Utilizadas

Para um bom desenvolvimento e implementação segura do software, foram necessárias ter em conta diversas técnicas que permitem garantir e assegurar uma boa prática. Desta forma, foram utilziadas as seguintes:

2.3.1 Recuperação

Foi possível garantir, que caso o programa não possa continuar, existe uma recuperação adequada.

2.3.2 Conhecer os limites

Como o tamanho do tipo de dados é dependente de máquina e compilador é uma boa ideia familiarizar-se com os limites na máquina onde o programa vai executar.

2.3.3 Tipos de Dados

Foi realizada a escolha do tipo de inteiro mais adequado para os valores que vai conter, na linguagem de programação utilizada.

2.3.4 Validação dos Inputs

- **Tipo**

É necessário validar que o input tem o tipo de dados expectável. Neste casos, à semelhança de muitos programas, lidamos com os dados de input assumindo que são uma string e posteriormente é feita a verificação dessa string sabendo se contém os caracteres apropriados, e convertendo-a para o tipo de dados desejado.

- **Tamanho**

É importante proceder à validação de que o input tem o tamanho expectável (por exemplo, o número de telefone tem 9 digitos, fora o "+351" inicial).

- **Intervalo**

Validar que o input se encontra dentro do intervalo expectável. Por exemplo, o userId tem que começar obrigatoriamente por "+", seguido de 3 números(de 0-9), seguido de 1 espaço e terminando com 9 números;

- **Razoabilidade**

Validamos que o input tem um valor razoável (por exemplo, o pin do contém apenas alfanuméricos, excluíndo qualquer tipo de caracter que não este).

- **Formato**

Validar que os dados de input estão no formato adequado Por exemplo, o userId, pin e otp encontram-se no formato correto;

- **Dados obrigatórios**

Foi necessário garantir que o utilizador insere os dados obrigatórios tais como o userId, pin,entre outros;

2.4 Ferramentas e Indicadores de Qualidade de Software

De forma a analisar e perceber a qualidade do software desenvolvido, a linguagem **Ruby** tem ao seu dispôr diversas ferramentas que auxiliam nesta análise.

2.4.1 Brakeman

Uma que pretendemos explorar mais foi a ferramenta **Brakeman**.

O **Brakeman** é uma ferramenta focada na localização de potenciais falhas de segurança no seu código. Ele também permite visualizar alertas especificamente relacionados ao Rails, tais como falhas de segurança que já foram corrigidas numa versão do Rails mais recente em relação à que é utilizada por nós.

Como tem como principal foco a segurança, é muito importante manter a *gem* sempre atualizada, para que seja possível detectar falhas descobertas mais recentemente.

2.4.2 Flay

Esta ferramenta procura similaridades na estrutura do código (ex: dois métodos que possuem o código muito semelhante). Neste caso em concreto, os métodos *ccmovelsign* e *ccmovelmultiplesign* tem algumas semelhanças no hash. **request_data**

```
request_data = {
    'request' => {
        'ApplicationId' => Base64.encode64(args["applicationId"]),
        'DocName' => args["docName"],
        'Hash' => args["hash"],
        'Pin' => args["pin"],
        'UserId' => args["userId"]
    }
}

return client.call(:cc_movel_sign, message: request_data)
```

```
request_data = {
    'request' => {
        'ApplicationId' => Base64.encode64(args["applicationId"]),
        'Pin' => args["pin"],
        'UserId' => args["userId"]
    },
    'documents' => {
        'HashStructure' => [
            {'Hash' => Digest::SHA256.hexdigest('Nobody inspects the spammish repetition').bytes,
             'Name' => 'docname teste1', 'id': '1234'},
            {'Hash' => Digest::SHA256.hexdigest('Always inspect the spammish repetition').bytes,
             'Name' => 'docname teste2', 'id': '1235'}
            ]}
}
return client.call(:cc_movel_multiple_sign, message: request_data)
```

Figure 2: Local onde podia ser aplicada a ferramenta *Flay*

2.5 Teste do Código Desenvolvido

Para o teste da implementação criada, nem tudo foi possível testar.

Neste sentido, mostramos a tentativa de testar que seria de esperar e se tudo acontecesse normalmente, iriam ser obtidos os dados e informações requisitadas.

Relativamente à versão, selecionando a opção número 1 e transpondo esse valor para a linha de comandos, obtemos:

```
[(master)⚡ % ruby test_cmd_wsdl.rb

############################ Command Line Program ############################
   1  Mostrar a Versão atual do Command Line Program
   2  GetCertificate
   3  CC Movel Sign
   4  CCMovelMultipleSign
   5  Validate OTP
   6  TestAll
   0  Fechar Programa

##############################################################################
Insira a sua Opção:
1
Versão do Programa: Version 1.0
```

Figure 3: Versao da Command Line

Relativamente às outras funcionalidades, infelizmente não foi possível obter o resultado pretendido devido à dificuldade sentida na utilização da API Savon. Na figura podemos ver como seria obtido o certificado. Para as outras funcionalidades, a forma de testar seria semelhante, apenas diferindo nos inputs.

```
############################ Command Line Program ############################
   1  Mostrar a Versão atual do Command Line Program
   2  GetCertificate
   3  CC Movel Sign
   4  CCMovelMultipleSign
   5  Validate OTP
   6  TestAll
   0  Fechar Programa

##############################################################################
Insira a sua Opção:
2
https://cmd.autenticacao.gov.pt/Ama.Authentication.Frontend/CCMovelDigitalSignature.svc?wsdl

+++ GetCertificate +++
Insira o seu número de telemóvel (+XXX NNNNNNNNN):
+351 *********
```

Figure 4: Input para GetCertificate

Dada a situação explicada em cima, seriam os seguintes inputs apresentados na figura de forma a fazer o testAll da aplicação. Assim, podemos observar uma pré-visualização do que seria esperado caso a aplicação estivesse funcional, desde a solicitação dos argumentos até à verificação da assinatura.

```
################################################################################
Insira a sua Opção:
6

+++ Test All inicializado +++
Insira o path do ficheiro a assinar:
../LICENSE
Insira o seu número de telemóvel (+XXX NNNNNNNNN):
+351
Insira o seu pin CMD Signature:

Initializing Test of All Commands
 0% ... Leitura de argumentos - file: ../LICENSE user: +351 [redacted] pin:
10% ...  A contactar servidor SOAP CMD para operação GetCertificate
20% ...  Certificado emitido para "
         pela Entidade de Certificação "
         na hierarquia do "
30% ...  Leitura do ficheiro "../LICENSE"
40% ...  Geração de hash do ficheiro "../LICENSE"
50% ... Hash gerada (em base64):

60% ...  A contactar servidor SOAP CMD para operação CCMovelSign
70% ...  ProcessID devolvido pela operação CCMovelSign:

80% ...  A iniciar operação ValidateOtp
Introduza o OTP recebido no seu dispositivo:
90% ...  A contactar servidor SOAP CMD para operação ValidateOtp:
100% ... Assinatura (em base 64) devolvida pela operação ValidateOtp:

110% ... A validar assinatura ...
Assinatura verificada com sucesso, baseada na assinatura recebida, na hash gerada e na chave pública do certificado de

+++ Test All finalizado +++
```

Figure 5: Exemplo de como executaria a testAll

3 Conclusão

A realização deste trabalho prático permitiu-nos aprender uma nova linguagem de programação, o **Ruby**.

Também nos permitiu aprofundar o conhecimento sobre as técnicas de desenvolvimento de Software, ficando com uma melhor noção sobre a sua importância para a construção de código seguro.

O grupo sentiu grandes dificuldades a entender o protocolo *SOAP* e como seria possível implementá-lo em Ruby. Através de uma intensa pesquisa, descobrimos a API *Savon* que serviu para o propósito.

Apesar de não terem sido concluídos os objetivos propostos, sentimos que estivemos muito próximas de alcançar os mesmos. Contudo, o trabalho de pesquisa e tentativa intensivo permitiu-nos aprender diversos conteúdos e pôr em prática algumas funcionalidades que podem vir a ser úteis em implementações futuras.