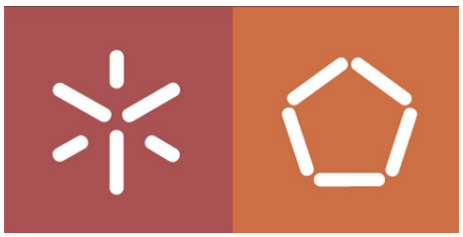
Universidade do Minho

Departamento de Informática



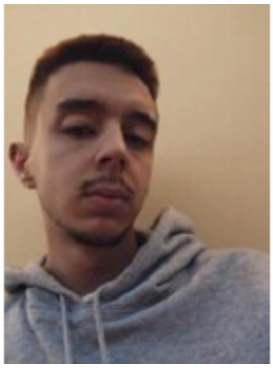
MEI - Mestrado em Engenharia Informática

Criptografia e Segurança da Informação

Tecnologias de Segurança

Grupo 13

Trabalho Prático 3



Hugo Marques PG47848

Nuno Mata PG44420

12 de Junho
2021/2022

Conteúdo

1 Introdução

Este relatório surge como documento de apoio para o projeto desenvolvido associado ao trabalho prático 3, da unidade curricular Tecnologias de Segurança. O âmbito do documento é descrever o software desenvolvido, tal como as bibliotecas usadas, como instalar e executar o software e também ainda outros aspetos que devem ter sidos em consideração.

Para contextualização, é pretendido que seja desenvolvida a implementação de um sistema/serviço, para o sistema operativo Linux, capaz de detetar modificações não autorizadas num determinado ficheiro ou num conjunto de ficheiros (uma pasta). Quando este é iniciado deve recolher periodicamente informações sobre os meta-dados relativos aos ficheiros que está a monitorizar, de forma a ser capaz de identificar quando existem modificações, estas informações devem ainda ser guardadas (num ficheiro de log por exemplo). Quando é verificado um alerta, através de uma alteração a um ficheiro que esta a ser monitorizado, o serviço deve ainda reportar esse acontecimento, alertando o responsável via e-mail ou mensagem telefónica.

Inicialmente, no relatório vão ser descritas as tecnologias, bibliotecas e abordagens utilizadas para o desenvolvimento do projeto. Em seguida, vai ser descrito o desenvolvimento do software, onde o objetivo é explicar a implementação das funcionalidades mais relevantes, e por fim são apresentados alguns dos resultados obtidos com o serviço desenvolvido.

2 Descrição do Sistema

O projeto é um monitorizador de ficheiros que pode "observar"qualquer tipo de ficheiro e verificar quando existem alterações. Pode ser escolhido observar 1 único ficheiro, vários ficheiros ou todos os ficheiros contidos num diretório. Quando é detetada uma alteração considerada *alerta* para o sistema, é enviado um e-mail e SMS para o destinatário desejado. As mensagens/e-mails enviados são configurados de acordo com o *alerta* produzido, são enviadas informações relativas à hora do *alerta*, ficheiro no qual foi detetada uma alteração, entre outras. Um *alerta* depende do que for designado no ficheiro de configurações do serviço para ser observado, caso a situação que está observada realmente ocorra então é um *alerta*, alguns exemplos de alertas que podem ser definidos são:

- É considerado *alerta* quando se identifica qualquer tipo de modificação detetada no ficheiro/ficheiros em monitorização;
- É considerado *alerta* quando se identifica que uma palavra específica é introduzida no ficheiro/ficheiros em monitorização;
- É considerado *alerta* quando se identifica que existe qualquer tipo de adição de informação ao ficheiro/ficheiros em monitorização;
- É considerado *alerta* quando se identifica qualquer tipo de remoção de informação ao ficheiro/ficheiros em monitorização;

Após ser disparado um alerta o serviço cria um ficheiro "log.txt", caso ele não exista, e guarda as informações associadas a ele (hora, ficheiro, tipo de alerta, nome do *host* que fez a modificação no ficheiro, etc).

A configuração do serviço pode ser feita no ficheiro de configurações, neste ficheiro podemos alterar parâmetros como: os ficheiros que vão ser observados pelo serviço, o número de segundos entre as verificações dos ficheiros, configurações associadas ao envio dos alertas (autenticação dos serviços externos usados para o envio de e-mails e SMS), estrutura da mensagem enviada pelo alerta, tipo de modificação que queremos que seja observada (como descrito acima), o caminho do ficheiro log que vai ser gerado pelos alertas, se queremos produzir um som quando é identificado um alerta, se queremos um limite de mensagens/e-mails enviados por dia/hora, etc.

3 Tecnologias Usadas

Nesta secção são abordadas as bibliotecas e tecnologias que foram usadas na implementação do serviço.

A tecnologia utilizada foi o Nodejs que executa linguagem JavaScript, pois era familiar a ambos os elementos do grupo e também dá acesso a várias bibliotecas que vão ser úteis no desenvolvimento do software, tais como:

- O *module* ***fs*** que vem pré instalado com o Node.js vai facilitar o acesso ao *file system*, ajudando a aceder, guardar e gerir ficheiros no nosso sistema;
- O *module* ***os*** vai ajudar a ter acesso a informações sobre o próprio sistema operativo (*hostname, home directory*, etc.);
- O *module* ***child process***, dá-nos a capacidade de fazer sub-processos associados ao nosso serviço;
- O *module* ***path*** também nos permite aceder e gerir ficheiros mais facilmente;
- O *module* ***request*** vai permitir fazer pedidos *HTTP GET*, ou *POST*, usado para aceder à *API* de envio de SMS por exemplo;
- O *module* ***logger*** vai ajudar criar o ficheiro de *log* associado aos alertas;
- O *module* ***mail*** vai permitir que seja possível o envio de e-mails;
- O *module* ***cli*** vai facilitar o processamento e pedido de argumentos passados pelo utilizador através da linha de comandos/terminal;
- O *module* ***daemon*** vai permitir tornar um *script* num *daemon*, para que este permaneça em execução e para que realize as suas monitorizações periódicas.

Usou-se ainda uma *application programming interface* (API) para o envio de alertas por SMS, o **Twilio**. Este serviço permitia através da criação de uma conta gratuita, o acesso a um número de telemóvel que podia ser utilizado para o envio de SMS com a utilização da sua API (apenas uma subscrição paga permitia o uso de um número pessoal).

4 Instalação e Configuração

Nesta secção vai ser descrito, não só como o serviço pode ser instalado e executado, mas também como se pode fazer diferentes configurações para ele.

Quanto à "instalação", como se trata de um projeto em Node.js, é necessário instalar esta tecnologia na máquina e em seguida devem ser corridos os seguintes comandos:

- Para criar a pasta que vai conter as bibliotecas: **npm i**;
- Para instalar as bibliotecas necessárias para o serviço: **npm i os**; **npm i child_process**; **npm i path**; **npm i pixl-request**; **npm i pixl-logger**; **npm i pixl-mail**; **npm i pixl-cli**; **npm i daemon**;
- Para executar o serviço: **node index.js**.

O projeto contém apenas 2 ficheiros, o **index.js** e o **config.json**, para a configuração do serviço vamo-nos focar mais neste segundo, que existe precisamente para podermos ser capazes de alterar parâmetros do nosso serviço que sejam desejados alterar. Aqui podem ser definidos os ficheiros a monitorizar e diferentes monitorizações para dois ficheiros diferentes. Por exemplo para a monitorização de um dos ficheiros apenas alertar quando são adicionadas informações ao ficheiro e para o outro alertar quando detetar qualquer atividade. No exemplo a baixo criamos 2 monitorizadores, um para o ficheiro "notas.txt"e outro monitorizador para todos os ficheiros dentro da pasta "configs". O tipo de alerta que queremos estabelecer para cada monitorizador depende do parâmetro **match**, onde podemos adicionar expressões regulares/*keywords* para o monitorizador detetar e produzir o alerta. Ambos os monitorizadores produzem ainda mensagens com destinos diferentes, um para o e-mail abaixo indicado e outro para o número de telefone indicado (o que temos acesso com a conta gratuita no serviço Twilio).

```
"monitors":
            {
                "name": "Monitorizar notas.txt",
                "path": "/home/tp3/notas.txt",
                "match": ".*",
                "email": "TS_TP3@gmail.com"
            },{
                "name": "Monitorizar todos os ficheiros",
                "path": "/home/tp3/configs/*",
                "match": "ALERT",
                "sms": "+15863333441"
            }
```

Neste ficheiro podemos ainda alterar o serviço associado ao envio do e-mail, no nosso caso usamos o software **sendmail** que estava em execução localmente na nossa máquina, por isso o nosso ficheiro de configurações tinha a seguinte instrução:

```
"mail_settings": {
            "sendmail": true,
            "newline": "unix",
            "path": "/usr/sbin/sendmail",
            "from": "admin@localhost"
            }
```

No entanto poderíamos usar um serviço com *SMTP servers* a correr localmente, ou não, para o envio de emails com o nosso endereço de email mais pessoal. Para isso poderíamos fazer a configuração, especificando o host do serviço, porta e a autenticação do e-mail associado desta forma:

```
"mail_settings": {
        "host": "smtp.gmail.com",
        "port": 587,
        "secure": false,
        "auth": {       "user": "grupo13",  "pass": "password_tp3"       },
        "from": "ts_tp3@gmail.com"
},
```

Relativamente ao envio de SMS, as configurações necessárias são apenas os dados de autenticação do serviço, para que este nos dê acesso às suas informações quando fazemos um pedido através da sua API. E o número de telemóvel que temos acesso. A configuração destas credenciais pode ser feita no ficheiro de configurações, são necessários os códigos de *sid* e *auth* atribuídos à nossa conta de Twilio. Nosso exemplo:

```
"twilio_settings": {
        "sid": "ACed514cceb5a29eb7cb7e4598b82c4a8b",
        "auth": "f642a8bbc734ee3e2b24bd0186056dc4",
        "from": "+15863333441"
    }
```

Estando a configuração dos serviços externos de e-mail e SMS estabelecidos, fazemos o template das mensagens enviadas. Consideramos que as informações importantes para conter na mensagem/e-mail enviado são o nome atribuído ao monitorizador que observou um alarme, o nome do ficheiro onde ocorreu tal como o seu "caminho"(*path*) e a data e hora. Esta mensagem pode ser alterada no ficheiro de configurações nos parâmetros **email_template** e **sms_template**. Então o nosso template para o envio de mensagem seria deste tipo:

```
"sms_template": "Files Monitor: [name]:
[file]\n\n[lines]\n \nDate/Time: [date]"
```

Existe ainda um pequeno áudio que é produzido quando é observado um alarme, recorrendo ao **vlc** mas sem abrir o seu GUI (graphical user interface).

```
"exec": "cvlc --no-video /home/tp3/alert.wav"
```

Por fim pode ser ainda referido que é possível alterar aqui o local onde o ficheiro de log criado pelo serviço para guardar todos os alertas observados vai ficar.

Existem configurações adicionais que podem ser definidas para o serviço de monitorização, no entanto aqui ficaram descritas as principais.

5 Experiência Adicional

Dado que os dois elementos do grupo dividiram as tarefas entre si e usaram duas linguagens de programação distintas, no fim optou-se por considerar a implementação de um mecanismo de controlo de acesso a ficheiros através da introdução de um *token*, como uma experiência adicional, tendo sido implementada à parte do programa em si.

Para tal recorreu-se à biblioteca *libfuse*, aliada em todo o processo, dado que através desta conseguimos criar um Sistema de Ficheiros próprio sem a necessidade interna de alterar o sistema de ficheiros predefinido pelo *Linux* em si. Sendo assim através deste mesmo sistema de ficheiros e do mecanismo de autorizações, sempre que um utilizador requisita uma operação de abertura de um ficheiro, através da chamada da função **open**, recebe um *token de segurança* para verificar a autenticação do utilizador.

5.1 Execução do programa

Para a utilização do sistema de ficheiros implementado, é necessário:

- Instalar as bibliotecas **python3**, **pip3**, **fuse3**, **fusepy**, **libfuse3** e **libfuse3-dev**.
- Aceder ao ficheiro *"users.txt"* e inserir os dados necessários para receber o *token* no telemóvel. Para tal insere-se o nome e o número no formato ('nome/número'). O ficheiro contêm os dados pessoais de um dos elementos do grupo, podendo o mesmo ser alterado.
- Para testar o programa, devem primeiramente ser criadas duas diretorias, uma diretoria que contêm o ficheiro original a ser executado e uma diretoria que irá permitir a montagem do *filesystem* através da replicação do ficheiro original, podendo este ser manipulado como se estivéssemos no *filesystem* original. Com isto, apenas é necessário correr o seguinte programa:

  python3 passthrough.py /diretoriaFicheiros /diretoriaMount

- Imediatamente após correr o ficheiro *passthrough*, deve ser inicializado um outro cliente numa nova janela do terminal. Ao aceder à diretoria **mount**, o objetivo é tentar aceder ao ficheiro dos registo dos utilizadores, que pode ser realizado através do comando *cat*, por exemplo. O terminal do servidor vai receber a solicitação e irá ativar as medidas de segurança e autenticação necessárias para a abertura do ficheiro.
- O servidor irá então solicitar o *username* do utilizador e vai averiguar se este é um dos donos do ficheiro, caso seja, é lhe enviado, por SMS, um *token* de autenticação que deve ser validado de forma a poder abrir o ficheiro.

5.2 Implementação

No que diz respeito à implementação da solução descrita, o grupo escolheu a linguagem **python** e recorreu a um sistema de ficheiros totalmente modelado nessa mesma linguagem. O *script* desenvolvido é baseado na biblioteca **fusepy** onde se escreve toda uma classe com as várias chamadas ao sistema pelo Sistema de Ficheiros que são atribuídos para correr diretamente pelo *Sistema Operativo* em si.

Com este ponto de partida, alterou-se a função **open()**, de modo a que fosse solicitada a identificação de um utilizador sem permissões ao tentar abrir um ficheiro.

```
fl = 1
#Verificação
try:
    print("Insira o seu username:")
    username = input()
    os.chmod("users.txt", 400)
    with open("users.txt", 'r') as ficheiroUsers:
        for line in ficheiroUsers:
            user = line.split('/')
            if(user[0] == username):
                #trata de enviar sms com o token
                fl = 0
                tokenSend = str(random.randint(100000, 999999))

                send = sendSMS(tokenSend, user[1])
                os.chmod("users.txt", 000)
                if send :
                    tempoEspera = 35
                    try:
                        signal.signal(signal.SIGALRM, timeout)
                        signal.alarm(tempoEspera)
                        print("Introduza o token recebido:")
                        tokenR = input()
                        if(tokenR == tokenSend):
                            signal.alarm(0)
                            os.chmod("users.txt", 444)
                            return os.open(full_path, flags)
                        else:
                            print("Codigo Incorreto!")
                            signal.alarm(0)
                            return 0
                    except IOError:
                        print("\nToken nao chegou a tempo\n")
                else :
                    print("\nErro no envio da mensagem")
    if fl==1:
        os.chmod("users.txt", 000)
        print("Utilizador desconhecido!")
```

Pela figura anterior, vemos que as permissões do utilizador são inicialmente definidos a 000, ou seja, não pode ser lido por ninguém, exceto a *root*. Se o utilizador quiser ler, terá de se identificar.

As permissões são novamente alteradas, para 400, após a solicitação do nome do utilizador, para ir de encontro à validação por parte do utilizador que está a invocar a operação de abertura do ficheiro. Se este se encontrar registado, os dados do registo serão usados no envio do *token* gerado, por SMS. Após a validaçáo do utilizador, as permissões do ficheiro retornam a 000, de modo a evitar tentativas de acesso.

É também validado a existência de um *timeout* que limita a validação do *código* por 35 segundos, utilizando bibliotecas do *Python* que permitem criar sinais e alarmes associados então a este timeout para a confirmação do código enviado por SMS. Se o *token* for inserido antes do *timeout* terminar, o *signal* é desativado, caso contrário é imprimida uma mensagem de erro.

As permissões são alteradas para 444 se houver sucesso na validação do código e respetivamente abertura do ficheiro. Se o *token* for inválido, o utilizador recebe a mensagem de "Código incorreto!"e o alarme é desativado, não sendo possível inserir o *token* corretamente, tendo o utilizador que solicitar o acesso novamente.

5.3 Arquitetura da solução

Tal como já foi explicado no tópico **6.2**, toda a parte da gestão de fica ao cargo do sistema montado, tendo em conta que este será o grande responsável por verificar o *username* do utilizador e aguardar os 35 segundos até que seja validada a operação requisitada após o envio do *token* por SMS. O seguinte diagrama evidencia a arquitetura da solução através da troca de ações entre utilizador e sistema de ficheiros.

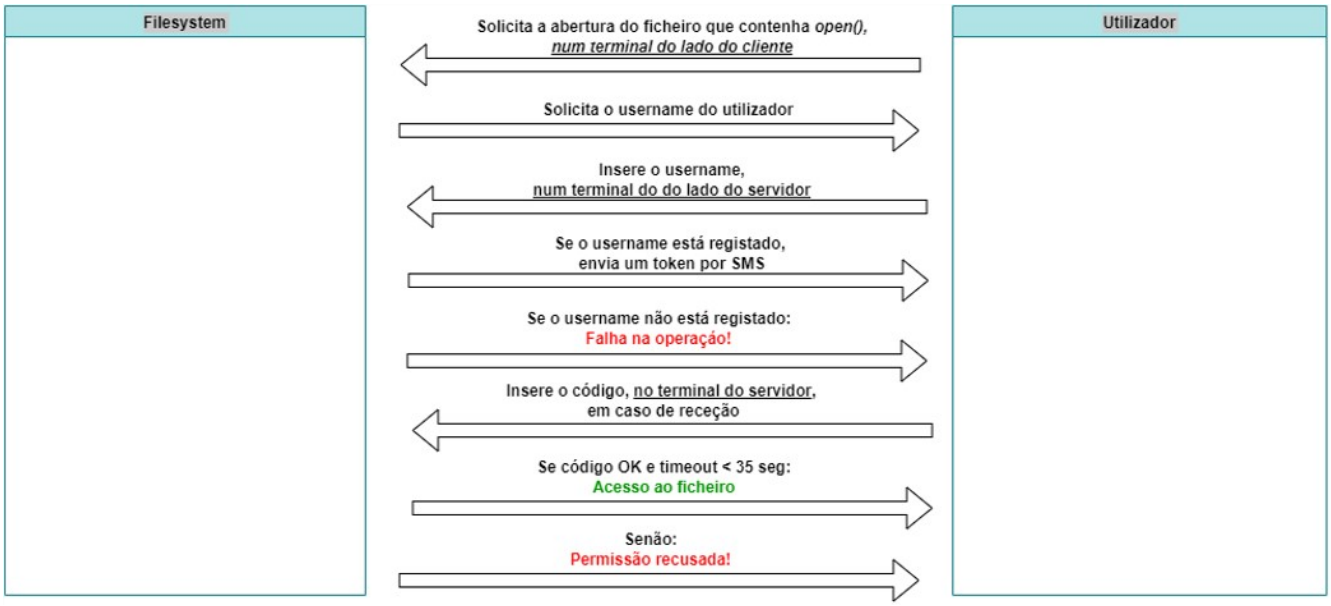


Figura 2: Arquitetura da Solução

5.4 Vulnerabilidades

Fazendo uma pequena análise de possíveis vulnerabilidades que possam estar associadas à solução apresentada, conclui-se que possa existir uma vulnerabilidade no sistema escolhido para o envio de *tokens* de autenticação.

Há ainda outra vulnerabilidade associada ao facto de que, sendo possível escolher uma diretoria para realizar a montagem do sistema, poderá ser uma oportunidade para fazer algo malicioso.

6 Conclusão

Ao longo do desenvolvimento deste trabalho foi-nos possível estudar e implementar funcionalidades de monitorização mais direcionadas para o sistema operativo Linux, o que é algo que normalmente não fazemos. Conseguimos implementar um serviço que funcionasse como daemon, execução constante no "fundo"para fazer as suas análises periódicas, e observasse efetivamente o tipo de modificação escolhida para o ficheiro/pasta escolhido para monitorização. Foi ainda implementado o envio de alertas por e-mail e SMS.

Como melhorias futuras, a mais evidente seria a implementação da experiência realizada de autenticação por token enviado por SMS. Esta funcionalidade poderia estar associada à execução do próprio serviço e ao acesso ao ficheiro de log gerado por ele, o que traria uma componente de segurança que o projeto atual não tem.