Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Fábio Dias Francisco

Gestão de Recursos e Processos em Rede

Tese de Mestrado

Sistemas e Tecnologias de Informação para as Organizações

Professor Doutor Paulo Tomé

Janeiro de 2013

A toda a minha família e amigos, a todos os que me apoiaram, dedico este trabalho

RESUMO

Hoje em dia deparamo-nos com sistemas informáticos cada vez mais desenvolvidos e complexos, mas ao desenvolver mais uma certa tecnologia, abrem-se por vezes mais falhas na segurança o que torna o sistema mais vulnerável a tentativas de invasão, e por vezes à destruição do próprio sistema.

Nos sistemas operativos modernos, o problema reside, não na totalidade, mas na sua maioria, no controlo/monitorização de processos em execução em determinado momento. No sistemas baseados em Unix, este problema não surte tantos efeitos nefastos, pois o controlo é feito de uma forma mais elaborada, obrigando o utilizador a autorizar com as suas próprias credenciais a execução de determinado processo. No caso do Windows, como é simples ignorar este pedido, o problema cresce exponencialmente, podendo-se exemplificar, com a quantidade de malware existente para este ambiente. O problema mencionado multiplica-se ainda mais, quando se fala em redes Windows, e nas várias máquinas ligadas a uma destas.

Para a resolução deste problema, é necessário desenvolver um sistema que seja capaz de monitorizar, e atuar sobre os processos em execução nas máquinas de uma rede de forma remota, dando o controlo ao Administrador para ditar o que pode estar em execução em cada máquina.

Concluindo, o tema escolhido para esta tese de mestrado "Gestão de Recursos e Monitorização de Processos em Rede", visa a construção de uma ferramenta que consiga fazer esta gestão, trazendo melhorias em relação a software já existente e possivelmente características novas, tudo isto para que o gestor de uma rede informática assente em sistemas Windows, possa executar um trabalho complexo de forma simples e dinâmica.

ABSTRACT

Today, IT systems get more and more evolved and complex; but sometimes, while you develop a part of a system, you sometimes forget about the other ones, and you leave vulnerabilities that someone can explore, leading to the destruction of the system.

In modern operating systems, the problem resides, not totally, but partially, in the control/monitoring of processes executing in a given time. On Unix based systems, this problem isn't that major, due to the control done by the OS, that makes the user insert his credentials to execute a process. In Windows, as it is simple to skip this step, the problem grows exponentially, as an example, we have the massive quantity of malware that exists to bring this environment down. The problem grows even more, when we get into Windows networks, and the machines connected to one of these.

To solve this problem, a system that is able to monitor and act on the executing processes of all the machines of a network remotely is needed, giving the control do the Administrator to dictate what may and may not execute in each machine.

Therefore, the chosen theme for this master thesis, "Network Resources Management and Process Monitoring", searches to build a software that can manage the network, bringing up improvements to the already existent software, and possibly new characteristics; all this work is done, so that a Windows network Administrator, may do a complex job easily and dynamically.

PALAVRAS CHAVE

Processos
Windows
Rede
Ferramentas
Software

KEY WORDS

Processes
Windows
Network
Tools
Software

AGRADECIMENTOS

Pela paciência demonstrada, e pelo apoio incondicional, agradeço aos meus Pais, pois sem eles nada disto seria possível. Aos meus amigos, pelos momentos de descontração proporcionados, quando nem tudo corria bem, e à Diana Lucena, por se dar ao trabalho de me desenhar o ícone da aplicação. A todos, o meu muito obrigado.

ÍNDICE GERAL

Conteúdo

ÍNDICE DE ILUSTRAÇÕES

ABREVIATURAS E SIGLAS

AES Advanced Encryption Standard
API Application Programming Interface
CSS Cascading Style Sheets
IP Internet Protocol
P2P Peer-to-Peer
SCTP Stream Control Transmission Protocol
SDK Software Development Kit
TCP Transmission Control Protocol
UDP User Datagram Protocol
VB Visual Basic
VNC Virtual Networking Computing
WPF Windows Presentation Foundation

1. Introdução

1.1 Enquadramento

Nos dias que correm, com as constantes mudanças no mundo da informática, algo considerado seguro e impenetrável pode tornar-se obsoleto de um dia para o outro. Isto pode acontecer, por exemplo, com medidas de segurança embutidas em sistemas operativos. Nas versões modernas destes, talvez devido à sua complexidade, um dos maiores problemas detectado, é a execução de software malicioso, que apesar de ter necessidade de elevação (autorização de nível administrador), para poder efetuar as modificações para o qual foi desenhado, os utilizadores garantem-na. Por vezes por desconhecimento, outras vezes por este se fazer passar por algo conhecido, ou mesmo por falta de tempo para ler informação sobre o que está a autorizar, o utilizador garante ao malware, tudo o que este necessita para danificar o sistema. Nos sistemas baseados em Unix, como Linux e Mac OSX, existe uma medida de segurança, na qual os próprios utilizadores são obrigados a ler, e a dar autorização ao processo, sob a forma de autenticação, para este correr. Este método é mais seguro, que aquele utilizado em máquinas Windows, onde o utilizador apenas necessita de responder à pergunta sobre a execução de um processo que requer elevação, com um simples "Sim" ou "Não". É sobre este último sistema que esta tese se vai debruçar, pois sendo o Windows, o Sistema Operativo moderno que mais falhas de segurança apresenta, pode-se dar como exemplo a quantidade massiva de vírus existente para este, torna-se interessante contribuir com uma ferramenta que possa de alguma forma ajudar a colmatar esta falha. Posto isto, este trabalho enquadra-se no âmbito da segurança em sistemas informáticos, de uma forma geral, e na segurança em redes informáticas Windows no particular.

1.2 Objetivos

Tendo enquadrado o projeto na área da segurança de sistemas informáticos, torna-se fundamental identificar os objetivos desta tese, e de que forma será concluída. O objetivo principal, será produzir uma ferramenta que seja capaz de monitorizar/atuar sobre os processos em execução nas várias máquinas de uma qualquer rede Windows. Ou seja, tornar possível ao Administrador duma qualquer rede, filtrar os processos em execução em máquinas remotas e dar autorização, ou não, a certos processos para serem executados. O Administrador não será responsável por filtrar todos os processos existentes em todas as máquinas, pois o software irá trabalhar tendo em conta uma lista de conformidade. Esta lista contém todos os processos autorizados a executar em determinado computador, sendo que esta é fornecida pelos responsáveis da empresa que implementa o software. O papel do administrador será, portanto, o de interagir com a lista das várias máquinas e possivelmente efetuar alterações. Outro dos objetivos será também fazer um estudo do estado da arte, de forma a poder identificar ferramentas existentes no panorama atual que possam de alguma forma contribuir para a resolução do problema, e também para demonstrar que este tema é válido para tese de Mestrado, através da mostra de aplicações de software já existentes, que de alguma forma, poderão competir com a aplicação a desenvolver.

O documento será organizado em capítulos, sendo que no primeiro capítulo será feita uma abordagem introdutória, onde se encontra um enquadramento do projeto na área da informática, e se apresentam os objetivos da tese de mestrado. No segundo capítulo, fala-se sobre o estado da Arte, começando pelas várias técnicas utilizadas na área da segurança em informática, passando pelas arquiteturas que poderiam ser usadas para o desenvolvimento da aplicação e concluindo com as diferentes aplicações já existentes no mercado, que poderão de alguma forma estar relacionadas com a aplicação a desenvolver. No terceiro capítulo é feita a introdução à aplicação, falando-se das técnicas utilizadas, bem como demonstrando o funcionamento desta com esquemas e explicação de algoritmos utilizados. No quarto capítulo, falar-se-á da aplicação do projeto a uma situação real, e aos resultados obtidos. No quinto e último capítulo, está presente a conclusão, acompanhada de uma análise crítica e uma projeção de trabalho futuro que poderá ser feito para enriquecer a aplicação ainda mais.

2. Estado da Arte

Após uma abordagem inicial ao tema, apresentar-se-ão de seguida um estudo comparativo de algumas tecnologias que estão intimamente relacionadas com o tema, bem como algum software presente no mercado, que poderia de alguma forma, substituir a aplicação a desenvolver em alguns aspetos.

2.1 Mecanismos de segurança

- **Autenticação de utilizadores** (Rouse,2005) - As funções de autenticação estabelecem a identidade de utilizador e/ou de sistemas, tendo em vista a determinação das ações e das capacidades permitidas. Estas funções estão na base do controlo do tipo de acesso aos recursos. Depois de validada a palavra-chave durante o login, os utilizadores podem ter tipos de acesso diferenciados aos recursos. Por exemplo, alguns utilizadores poderão apenas consultar certos ficheiros, imprimir para determinadas impressoras ou modificar determinados ficheiros. Outros utilizadores poderão não ter capacidade para copiar ou apagar determinados ficheiros.

- **Cifragem** (Rouse, 2005) - Este é um processo que "modifica" os dados através de uma chave secreta, conhecida somente por partes autorizadas. O processo de modificação da mensagem denomina-se cifragem sendo a sua função transformar texto simples num criptograma. O processo de recuperação da mensagem original a partir do criptograma denomina-se decifragem. Os algoritmos de criptografia, também denominados cifras, são funções matemáticas responsáveis pela cifragem e decifragem. Existem portanto dois componentes nestes algoritmos, o da codificação e da descodificação.

- **Firewalls** (Microsoft, 2004-a) - Uma firewall é, basicamente, um equipamento computacional colocado na zona de fronteira de uma rede cujo principal objetivo é o controlo de acesso a essa rede por parte de utilizadores sediados noutras redes.
Tipicamente, as firewalls são utilizadas para controlar o acesso a uma intranet, protegendo-a de acessos não autorizados oriundos da internet. No entanto, podem ser usadas para proteger uma rede de acessos a partir de qualquer outra rede. Esta serve para proteger o sistema de ataques externos, sendo ineficaz para tentativas de acesso internas.

- **Assinatura Digital** (Rouse, 2005-c) - Consiste num conjunto de dados cifrados associados a um documento. As assinaturas digitais garantem a integridade do documento, e identificam quem o envia. Não garantem, no entanto confidencialidade, pois os documentos circulam não cifrados. Isto não constitui qualquer problema, pois na maior parte dos casos, o que se pretende garantir é a autenticidade do documento, independentemente de ser público ou não.

- **Logs** (techterms.com, 2008)- Os logs são medidas básicas de segurança, pelo facto de serem simples na sua natureza. São registos gerados pelos sistemas ou aplicações com informações dos eventos ocorridos. É uma ferramenta muito útil para auditorias de acessos, para verificação do que está a ser utlizado, possíveis falhas do sistema, etc. Dependendo do sistema e do hardware, a geração de logs poderá tornar-se lenta. A manutenção de logs detalhados é importante quando for necessário investigar uma invasão na rede, além de servir como prova contra um possível invasor detectado.

- **Antivírus**(Microsoft, 2004-b) - Trata-se de um software que verifica a existência de vírus em computadores, pastas ou ficheiros e, ao encontrá-lo, tenta removê-lo. O modo de remoção pode ser configurado pelo utilizador. Normalmente, o antivírus tenta remover apenas o vírus e, caso não o consiga, remove também o ficheiro, depois da autorização do utilizador. Uma vez instalado, o antivírus pode ser configurado, dependendo das suas características, para ficar ativo e analisar todos os ficheiros que forem abertos. Caso apareça algum vírus, avisa imediatamente o utilizador. Como surgem diariamente novos tipos de vírus, o utilizador deve atualizar esta ferramenta sempre que possível.

- **Políticas de Segurança -** Uma política de segurança é um conjunto formal de regras que devem ser seguidas pelos utilizadores dos recursos de uma organização. Deverá, também, ser suficientemente flexível para se adaptar a alterações desta. É necessário que as regras sejam facilmente acessíveis a todos os membros da empresa, para que estes não possam negar o conhecimento destas, e possam também segui-las com facilidade. É necessário definir objetivos de segurança, como listas de conformidade por exemplo, pois sem linhas guia, qualquer um pode cometer um erro crasso a nível de segurança sem sequer saber que o está a cometer. Definir objectivamente todos os aspetos abordados, não deixar a opinião pessoal enevoar a visão objetiva do problema. Definir a posição da empresa em cada questão, ou seja, esclarecer de forma simples e objetiva qual vai ser o comportamento da empresa perante cada situação prevista.

Definir circunstâncias em que é aplicada cada uma das regras, isto é, explicar como cada membro se deverá comportar perante certa situação, ou como o sistema irá reagir a cada momento, dependendo do que se está a passar naquele espaço de tempo. Definir os papéis de cada membro da organização, atribuindo comportamentos padrão que cada membro, segundo a suas características dentro da organização, tem de seguir. Especificar as consequências do não cumprimento das regras, isto engloba as consequências para os sistema, e para o membro que não as cumprir. Definir o nível de privacidade garantido aos utilizadores, atribuir um maior nível de privacidade àqueles que possuem um trabalho de maior sensibilidade. Identificar os contactos para o esclarecimento de questões, no caso de surgirem dúvidas de última hora.

2.2 Arquitetura P2P

P2P (An Overview of Peer-to-Peer Computing, 2003), é um conceito tecnológico que é aplicável a vários níveis de arquiteturas de sistemas. A sua característica típica é a existência de comunicação simétrica e troca de dados entre nós; cada nó é em simultâneo, cliente e servidor. É a base para computação distribuída descentralizada. Alguns benefícios da arquitetura incluem melhoria na dimensionalidade evitando pontos centralizados; a eliminação de infraestruturas custosas, permitindo a comunicação entre nós e a partilha de

informação e recursos. Tendo referenciado as suas características mais importantes, explicar-se-ão de seguida, esses termos de forma mais aprofundada.

Decentralização

Um dos principais objetivos desta característica é dar ênfase ao controlo de dados e recursos a cada utilizador. Num sistema completamente decentralizado, cada nó assume o mesmo papel de um qualquer outro nó. Isto torna a implementação deste modelo difícil, já que não existe um servidor central que tenha a visibilidade sobre toda a rede e possa fornecer informação sobre os dados disponíveis para partilha. Esta é a razão do porquê de muitas redes P2P serem na verdade híbridas, onde existe uma localização central, que contém a informação sobre todos os dados e recursos prontos a partilhar, mas cada nó faz o download dos ficheiros, não a partir da pasta central, mas sim dos outros nós que possuem tal informação.

Dimensionalidade

Um benefício imediato da decentralização é a dimensionalidade. Esta é limitada apenas pelo número de operações centralizadas que necessitam de execução, o paralelismo herdado que é demonstrado por uma aplicação, e o modelo de programação que é usado para representar o resultado da computação.

O Napster atacou o problema da dimensionalidade fazendo com que os nós descarregassem os ficheiros de música diretamente dos nós que possuíam os ditos ficheiros. Consequentemente, esta rede foi capaz de crescer até mais de 6 milhões de utilizadores no pico da sua utilização.

Tolerância a falhas

Como o objetivo principal da arquitetura passa pela descentralização, não existe um ponto central que, em caso de falha, afete toda a rede, como no paradigma cliente-servidor. Esta é uma vantagem enorme neste tipo de rede, pois é possível recuperar devido à redundância existente dentro da rede, podendo cada nó ligar-se a um qualquer outro que possua a informação desejada (ligações Ad-Hoc).

2.3 Arquitetura Cliente-Servidor

A arquitetura Cliente-Servidor (Mendes, 2002) permite, além de comunicação fluída e permanente entre as 2 aplicações, passar também toda a carga de processamento ou a maior parte desta, para o servidor, equipamento este, que possui normalmente hardware muito superior a uma simples workstation.
Passando a explicar de forma mais profunda, o modelo cliente servidor, é um modelo computacional que separa clientes e servidores, sendo interligados entre si, geralmente,

através de uma rede de computadores. Cada cliente envia pedidos ao servidor e espera por uma resposta deste. Enquanto isto o servidor recebe os pedidos, pode aceitar processá-los, e retornar o resultado para os clientes que efetuaram o pedido.
Apesar de ser utilizado em muitas aplicações, a arquitetura é praticamente a mesma em todas elas. Apesar desta levar a pensar que o cliente e o servidor residem em hardware distinto, estas aplicações podem residir na mesma máquina. A maior diferença entre as duas aplicações é o facto do servidor partilhar recursos com os clientes, enquanto que o cliente não partilha recursos com nenhum outro nó, só efetua pedidos ao servidor. Falando mais aprofundadamente nas características das diferentes aplicações, começar-se-á pelo cliente.
As características do cliente incluem, por exemplo, iniciar pedidos ao servidor, ou seja, como esta arquitetura pressupõe comunicação entre os dois tipos de aplicação, sendo suposto este possuir a menor capacidade de processamento, o cliente apenas executa pedidos ao servidor, para que este lhe forneça dados necessários à sua execução.
O cliente espera por respostas, após iniciar os pedidos, sendo que para poder continuar a sua execução necessita que o servidor lhe envie os dados pedidos. Recebe respostas estando à espera destas desde o momento que envia o pedido, esperando que o servidor processe o pedido, e lhe envie os dados.
Normalmente, este tipo de aplicação, liga-se a um, ou a um número reduzido de servidores de uma só vez, devido à sua configuração de fraco impacto na máquina onde está implementada, este possui características simples que apenas permitem a ligação a um número reduzido de aplicações servidor.
O usual neste tipo de aplicações é interagir diretamente com os utilizadores finais através de uma UI, pois é esta a aplicação que interage diretamente com o utilizador, embora exista exceção à regra em que se passa exatamente o contrário, sendo o servidor a interagir com o utilizador (o software desenvolvido nesta tese, é um desses casos); por fim, utiliza recursos de rede, pois esta arquitetura pressupõe comunicação entre as duas aplicações.

O servidor, ao contrário do cliente, que envia pedidos, está sempre à espera de receber pedidos de informação do cliente , para que consiga decidir o que é necessário e responder ao cliente com os dados solicitados. Caso o pedido do cliente tenha uma natureza complexa, pode comunicar com outros servidores, de forma a conseguir responder ao pedido do cliente corretamente. Por fim, fornece recursos de rede, pois a ligação do cliente é feita diretamente ao servidor.
Esta arquitetura apresenta vantagens tais como, na maioria dos casos, melhorar a capacidade de manutenção de uma rede, pois os papéis e responsabilidades do sistema, ficam subdivididos e distribuídos por vários computadores conhecidos entre si. Todos os dados importantes são guardados nos servidores, que por norma, possuem uma maior segurança que uma workstation cliente, podendo assim controlar melhor o acesso a recursos, a clientes cujas credenciais permitam acesso a estes.
Visto que o armazenamento de dados é centralizado, este sistema torna-se mais fácil de atualizar, ao contrário do paradigma P2P (Peer-to-Peer), que pressupõe a atualização de cada nó da rede, o que leva a grande consumo de tempo, e a um aumento na probabilidade de ocorrência de erros. Por último, funciona com clientes de várias capacidades, pois o poder de processamento tem de estar alocado no servidor.

Como foi visto nas características do servidor, este recebe comunicações do cliente, no caso destas excederem o limite para o qual o servidor está preparado, o programa poderá deixar de responder. Há uma clara falta de robustez, pois no caso de falhar o servidor, não existe mais ninguém que consiga processar os pedidos dos clientes, ao contrário das redes P2P, em que caso falhe um dos nós os outros conseguem processar os vários pedidos, ou seja, existe redundância, o que neste caso, não existe.

2.4 Técnicas de Cifragem

Sendo este um dos pontos fulcrais na segurança informática, procedeu-se à identificação e consequente descrição de algumas técnicas de cifragem utilizadas nos dias de hoje.
A cifragem, é a capacidade de transformar algo legível em algo que não é reconhecível com o uso de uma cifra específica.

2.4.1 AES

Em criptografia, o AES (McCaffrey,2002), é uma cifra de bloco adotada pelos EUA. Espera-se que a sua adopção seja geral e analisada extensivamente, de forma a poder descobrir se se trata de algo impenetrável.

Descrição da cifra

Esta cifra é baseada num princípio conhecido como rede de permutação-substituição**,** e é rápido, quer em software, quer em hardware. O AES tem um tamanho de bloco fixo de 128 bits, e uma chave que varia entre os 128, 192 e 256 bits. Esta cifra opera numa matriz de 4x4 bytes, apelidada de *estado.*

O tamanho da chave é importante, pois especifica o número de ciclos de repetição das rondas de transformação que convertem os dados de entrada, apelidado de texto simples, nos dados de saída, denominados de texto cifrado. O número de ciclos de repetição são os seguintes:

- dez ciclos de repetição para a chave de 128 bits;
- doze ciclos de repetição para chaves de 192 bits;
- catorze ciclos de repetição para chaves de 256 bits.

Cada ronda consiste em vários passos de processamento, incluindo um que depende da própria chave de cifragem. Um conjunto de rondas invertidas, são utilizadas, de forma a transformar texto cifrado em texto simples utilizando a mesma chave de cifragem.

Descrição de alto nível do algoritmo

Os passos para cifrar algo, utilizando esta técnica, encontram-se enunciados abaixo.

1 - ExpansãoChave - são geradas "round keys" derivadas da chave de cifragem, utilizando o algoritmo de escalonamento de chaves de Rjandael's;
2 - Ronda inicial
    1 - AdicionaChave - Cada byte do *estado* é combinado com a roundKey utilizando uma operação XOR;
3 - Rondas
    3.1 - SubBytes - um passo de substituição não-linear onde cada byte é substituído por outro de acordo com uma tabela;
    3.2 - TrocaLinhas - um passo de transposição, onde cada linha do *estado* é mudada de sítio ciclicamente um certo número de passos;
    3.3 - MisturaColunas - uma operação de mistura que opera nas colunas do estado, combinando os 4 bytes em cada coluna;
    3.4 - Adiciona Chave
4 - Ronda final (sem MisturaColunas)
    4.1 - SubBytes;
    4.2 - TrocaLinhas;
    4.3 - AdicionaChave;

O passo de SubBytes

Neste passo cada byte da matriz de *estado* é substituído com um SubByte utilizando uma caixa de substituição de 8-bit, também conhecida como caixa-S de Rijndael. Esta operação fornece não linearidade à cifra.

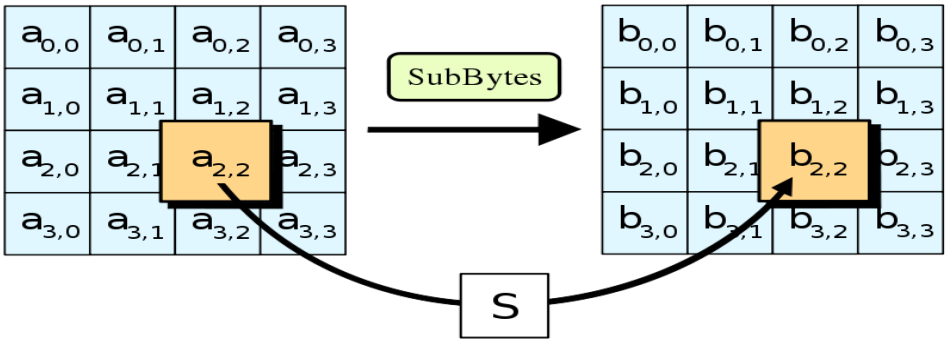

Ilustração 1 - passo SubBytes (McCaffrey,2002)

O passo TrocaLinhas

Este passo opera sobre as linhas do *estado;* ele muda ciclicamente os bytes de cada linha um certo número posições. No AES, a primeira linha é deixada inalterada. Cada byte da segunda linha, é mudado uma posição para a esquerda, da mesma forma os bytes da terceira e quarta linha são mudados duas e três posições para a esquerda, respetivamente, Para tamanhos de blocos de 128 e 192 bits, o padrão de mudança é o mesmo. A linha n é mudada para a esquerda n-1 bytes. Desta forma, cada coluna do estado de saída passo ShiftRows é composto de bytes de cada coluna do estado de estrada. Para cada bloco de 256 bit, a primeira fila fica inalterada, e a mudança para a segunda, terceira e quarta fila é de 1 byte, 3 bytes e 4 bytes respetivamente - esta mudança aplica-se apenas para a cifra de Rijndael quando se usa um bloco de 256 bit, já que a AES não usa blocos deste tamanho.

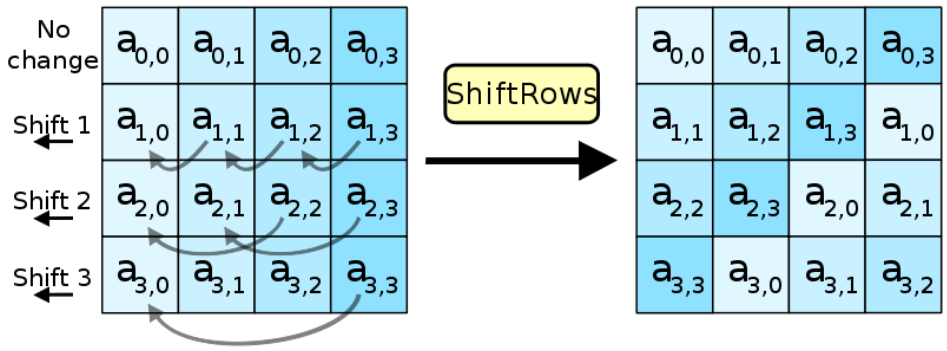

Ilustração 2 - Passo TrocaLinhas (McCaffrey,2002)

O passo MisturaColunas

Neste passo, os quatro bytes de cada coluna do *estado*, são combinados utilizando uma transformação linear invertível. A função MixColumns, pega em quatro bytes de entrada, e tranforma-os em 4 bytes de saída, onde cada byte afecta cada um dos 4 bytes de saída. Juntamente com o ShiftRows, este passo fornece difusão à cifra.

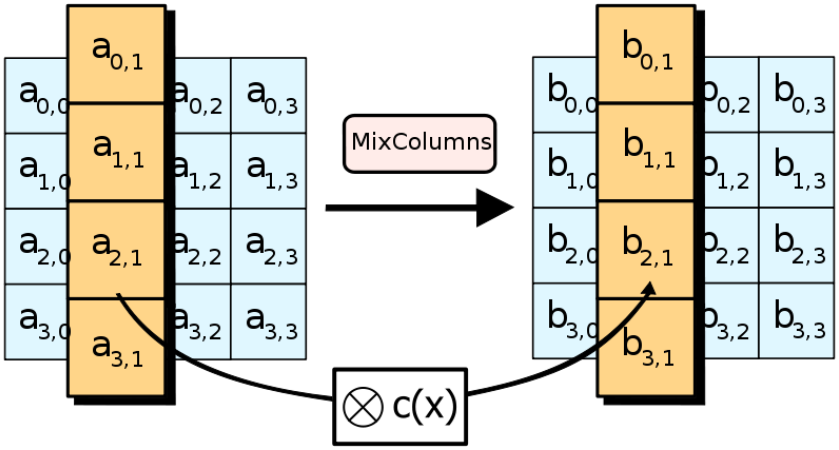

Ilustração 3 - Passo MisturaColunas (McCaffrey,2002)

O passo AdicionaChave

Neste passo, a subchave é combinada com o *estado.* Para cada ronda, a subchave é derivada da chave principal; cada subchave é do mesmo tamanho do *estado*. A subchave é somada combinando cada byte do *estado* com o byte correspondente da subchave utilizando uma operação XOR bit a bit.

2.4.2 Merkle-Helmann

Esta cifra é um cripto sistema de chave assimétrica (Agarwal, Ashish. 2011), significando que para a comunicação, são necessárias duas chaves: uma chave privada, e uma chave pública. Este sistema é baseado no problema da soma dos subconjuntos. O problema é o seguinte: dado um conjunto de números A e um número b, encontrar um subconjunto de A, cuja soma entre os seus elementos seja b. No geral, este problema é conhecido como sendo NP completo. Contudo, se o conjunto de números for super crescente - isto é, cada elemento do conjunto é maior que a soma dos anteriores - o problema é fácil e solúvel em tempo polinomial com um simples algoritmo "greedy".

Geração da chave

Na cifra de Merkle-Hellman, as chaves são "mochilas" (do problema complexo da mochila). A chave pública é uma "mochila" hard, e a chave privada é uma "mochila" 'easy', ou supercrescente, combinado com dois números adicionais, um multiplicador e um módulo, que são usados para converter a mochila supercrescente, na mochila "hard". Estes mesmos números são usados para transformar a soma do subconjunto da mochila hard, na soma do subconjunto da mochila 'easy', que é solúvel em tempo polinomial.

Cifragem

A ideia por trás da cifragem Merkle-Hellman (Agarwal, Ashish. 2011), é criar um subconjunto que possa ser resolvido facilmente e esconder a natureza supercrescente, utilizando multiplicação modular e permutações. O array transformado forma uma mensagem cifrada e o vetor original supercrescente, forma a chave privada e é utilizada para decifrar a mensagem.

Decifragem

A decifragem (Princeton, 2007) é possível devido ao multiplicador e módulo utilizados para transformar a mochila 'easy' supercrescente, na chave pública, também podem ser usados para transformar o número representando o texto cifrado na soma dos elementos correspondentes da mochila supercrescente. Depois, utilizando um simples algoritmo 'greedy', este pode ser resolvido utilizando operações aritméticas de ordem n, que decifram a mensagem.

Exemplo:

Primeiro cria-se uma sequencia supercrescente, por exemplo:

```
w = {2, 7, 11, 21, 42, 89, 180, 354}
```

Esta é a base para a chave privada. Daqui calcula-se o somatório:

$$\sum \mathbf{w} = \mathbf{706}$$

Em seguida, escolhe-se um número q, maior que a soma, por exemplo q = 881. Escolhe-se também um número r, que seja primo entre si com q (um número diz primo entre si com outro numero se o único divisor comum entre os dois for 1), e se encontre entre [1,q[, por exemplo r=588. A chave privada consiste em q, w e r. Para calcular a chave pública, gera-se a sequencia B, multiplicando cada elemento em w, por r mod q, obtendo-se o seguinte:

```
B = {295, 592, 301, 14, 28, 353, 120, 236}
```

A sequencia B, é a chave pública. Digamos que se pretende cifrar a letra 'a'. Primeiro é necessário converter essa letra para binário:

```
a = 01100001
```

Em seguida, multiplica-se cada bit correspondendo cada um deles com o numero do conjunto B respetivo, e somando os vários produtos:

```
a = 01100001
0 * 295
+ 1 * 592
+ 1 * 301
+ 0 * 14
+ 0 * 28
+ 0 * 353
+ 0 * 120
+ 1 * 236
= 1129
```

Este número, 1129, é o resultado da cifragem de 'a'. para decifrar, mutiplica-se 1129 pelo inverso do módulo utilizado para obter a chave pública, ou seja $1129 * r^{-1} \bmod q$.

```
1129 * 442 mod 881 = 372
```

Agora, decompõe-se 372 selecionando o maior número em w que é menor ou igual a 372. Selecionando posteriormente o elemento maior, menor ou igual ao resultado, até que a diferença seja 0:

```
372 - 354 = 18
18 - 11 = 7
7 - 7 = 0
```

Os elementos que foram utilizados da chave privada, correspondem aos bits com valor 1 na mensagem:

```
01100001
```

Quando traduzido para binário, este 'a' será a mensagem final decifrada.

2.5 Sockets de Rede

Um socket de rede (Princeton, 2009) é a paragem final de um fluxo de processos de comunicação que atravessa uma rede de computadores. Hoje em dia, a maior parte da das comunicações feitas entre computadores, tem como base o IP; daí a maioria dos sockets serem sockets de internet. Um endereço de socket, é portanto, a combinação de um endereço IP, e um número de porto, tal como uma ligação telefónica é a combinação de um número, com o código da área onde se encontra o número. Baseado neste endereço, os sockets entregam a informação, ao processo ou thread da aplicação apropriada.

2.5.1 Vista Geral

Um socket de internet, é composto por uma combinação única de um endereço IP e um porto; endereço remoto de socket, apenas para sockets TCP estabelecidos. Tal como será mostrado abaixo na parte de cliente servidor, muitos servidores TCP servem múltiplos clientes de forma concorrente, o que leva à necessidade do servidor criar um socket para cada cliente, e estes partilharem o mesmo endereço local. O protocolo de transporte é também um fator a ter em conta, podendo ser TCP, UDP, ou até mesmo IP. A porta TCP 53 é diferente da Porta UDP 53, daí a serem sockets completamente diferentes.

Dentro do sistema operativo e da aplicação que criou o socket, este é referenciado por um identificador único constituído por um inteiro, apelidado de identificador de socket ou número de socket.

2.5.2 Tipos de Socket

Existem vários tipos de sockets, sendo a diferença entre estes, o protocolo utilizado. Existem portanto sockets Datagram, também conhecidos por sockets "sem ligação", que utilizam o protocolo UDP; os sockets de tipo stream, também conhecidos por sockets orientados à ligação, que utilizam o protocolo TCP ou o SCTP; por fim existem os sockets Raw, tipicamente disponíveis em router e outros equipamentos de rede. Aqui salta-se a camada de transporte, e os cabeçalhos dos pacotes são disponibilizados à aplicação.

2.5.3 Estados de Socket e o modelo Cliente-Servidor

Processos que fornecem serviços aplicacionais são chamados de servidores, e criam sockets ao iniciar que estão num estado de escuta. Estes sockets estão à espera da iniciativa dos programas clientes.
Um servidor TCP (Figueira, 2010), pode servir vários clientes de forma concorrente, criando um processo filho para cada cliente e estabelecendo uma ligação TCP entre o cliente e o processo filho. Sockets únicos e dedicados, são criados para cada ligação. Estes encontram-se em estado estabelecido, quanto uma ligação ou circuito virtual, é estabelecido com o socket remoto, fornecendo um stream de bytes duplex (comunicação nos dois sentidos, um terminal de cada vez).
Um servidor pode criar vários sockets concorrentes com o mesmo numero de porto e endereço IP local, cada um mapeado para o seu próprio processo filho, servindo o seu próprio terminal. Estes são tratados pelo sistema operativo como sockets diferentes, pois o endereço remoto de cada um é diferente (endereço cliente ou porto).
Um socket UDP (Figueira, 2010), não pode estar num estado estabelecido, pois o UDP não estabelece ligações. Logo o netstat, não mostra o estado de um socket UDP. Um socket UDP, não cria processos filhos para cada nova ligação, este responde aos vários pedidos de forma sequencial através do mesmo socket. Isto implica que os sockets UDP não são identificados pelo endereço remoto, mas apenas pelo endereço local, embora cada mensagem tenha um endereço remoto associado.

2.6 Linguagens de Programação

Entre uma infinidade de linguagens de programação que poderiam ser tidas em conta para este projeto, faria sentido que se escolhesse algo que funcionasse bem em conjunto com os API do Windows. Sendo que foi escolhida a plataforma .NET da Microsoft por esta razão, poder-se-ia escolher entre C++, C# (Microsoft, 2002) e Visual Basic. Pela simplicidade da sua programação, e por permitir um desenvolvimento rápido de aplicações robustas, a escolha recaiu sobre a segunda.

Passando a explicar um pouco as origens desta linguagem, como já foi referenciado, esta linguagem está incluída num grupo de linguagens utilizadas na plataforma .NET da Microsoft, sendo que esta foi criada pela própria empresa, para concorrer com outras soluções, criadas por outros. Assim a Microsoft ficaria com uma solução caseira para a criação rápida de aplicações. Criada tendo por base o C++, a sua sintaxe recebe influências de outras, tais como o Java e o Object Pascal.

Entre as suas características, sobressaem as operações aritméticas típicas (adição, subtracção, etc), o uso de ponteiros, embora não aconselhado, pois os blocos de código que os usam

necessitam muitas vezes de permissões de segurança muito altas para serem executados. Suporta também coletores de lixo, sendo a função destes recuperar blocos de memória que já não estão a ser utilizados pelo programa, de forma a impedir paragens na execução dos programas devido à falta de recursos.

Falando de bibliotecas, esta linguagem, ao contrário de outras muito utilizadas, não inclui qualquer conjunto de implementação de classes ou funções de execução; isto é, o código é organizado em conjuntos de "namespaces" (do Português, espaços de nome), que agrupam as classes com funções semelhantes. Por exemplo, System.Console, é utilizado para entrada/saída de dados. Existe contudo um sistema de referências, em que se podem incluir nomes de espaço através da chamada de ficheiros .dll (dynamic link library).

A política de gestão desta linguagem, ao contrário de outras linguagens proprietárias da Microsoft, tal como o Visual Basic, inclui a submissão do C# a um processo de normalização. No entanto é a empresa que continua a ser a maior força por trás da inovação da linguagem. Além disso, a Microsoft fez questão de frisar a importância do C#, na sua estratégia de software.

2.7 Visual Studio

De todos os IDE (integrated development environment, ambiente de desenvolvimento integrado) existentes no mercado, foi escolhido o Visual Studio da Microsoft, pois seria o único a permitir o desenvolvimento de forma simples e dinâmica na linguagem escolhida, o C#. Este IDE inclui um editor de código, que além de suportar IntelliSense, suporta também code refactoring, ou refactoração de código. O debugger incluído, suporta tanto código fonte das aplicações como código máquina. Outra ferramenta incluída, e bastante utilizada no desenvolvimento da aplicação informática presente nesta tese, é o designer gráfico de janelas, utilizado primariamente para dar uma interface visual ao utilizador, de forma a este poder interagir com o programa de forma simples.

Esta ferramenta suporta muitas linguagens de programação além da escolhida, através de serviços de linguagem, que permitem ao editor de código e ao debugger suportar praticamente qualquer linguagem, desde que exista um serviço específico à linguagem escolhida. Nas linguagens incluídas na ferramenta, figuram o C/C++ (através do Visual C++), o VB .NET, C# e F#, este último só apareceu a partir da versão 2010 do Visual Studio. Existe suporte a linguagens como Ruby e Python, através da instalação de pacotes de linguagem. Suporta também XML/XSLT, HTML/XHTML, JavaScript e CSS.

Como esta ferramenta possui imensas características, vão ser explicadas algo por alto, as que foram necessárias ao desenvolvimento da aplicação, começando pelos designers de janelas. Existem vários tipos de designer, como o Windows Forms designer, o WPF designer, Web

Designer, etc. Aquele que foi utilizado para o desenvolvimento desta aplicação foi o designer WPF. Este designer, nome de código Cider, foi introduzido pela primeira vez na versão 2008 do Visual Studio. É utilizado para editar interfaces gráficos de aplicações, gerando código XAML no processo de edição, para cada elemento novo colocado na interface, através do paradigma "drag and drop", introduzido no Windows Forms aquando da sua criação.

Este paradigma permite ao utilizador, selecionar de uma caixa de ferramentas os elementos que pretende colocar na janela da aplicação, tais como, botões, listas, checkboxes, etc. Ao clicar duas vezes em cima deste elemento, uma classe é automaticamente criada pelo editor de código, permitindo facilmente identificar o local onde as alterações ao código terão de ser feitas, de forma àquele elemento efetuar alguma ação quando se interage com ele, como por exemplo, clicar num botão.

Outra característica muito importante, além do editor de código e debugger já referenciado no início deste subcapítulo, é o explorador da solução, pois é o que permite navegar entre os vários ficheiros .cs, que possuem o código editado no editor de código, sem esta funcionalidade, ter-se-ia de ter tantas instancias do Visual Studio abertas, quantas classes isoladas existissem (ficheiros .cs).

Por fim, e não menos importante, a característica que permitiu o desenvolvimento da aplicação móvel, o SDK Windows Phone 7. Esta ferramenta, é um pacote adicional, que é instalado após a instalação do Visual Studio. Este permite, tal como o próprio nome indica, utilizar o Visual Studio como ferramenta de desenvolvimento de aplicações móveis para Windows Phone. Este possibilita o desenvolvimento de diversas aplicações para este sistema operativo móvel, como pode ser vislumbrado na Ilustração 4, abaixo.

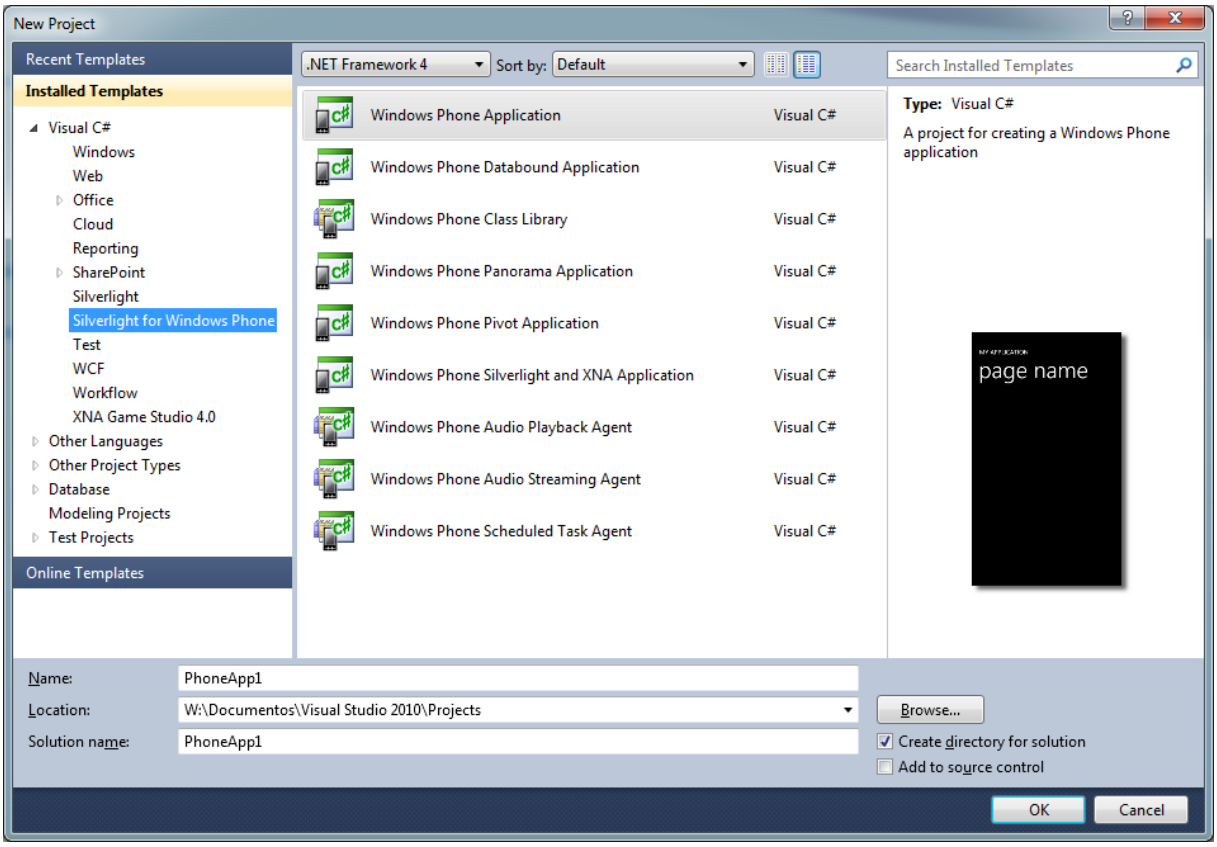

Ilustração 4 - Tipos de projetos Windows Phone

Entre estes projetos, destaca-se o Windows Phone Silverlight, que permite, entre outras coisas, a utilização de Sockets na aplicação móvel, para a comunicação entre as várias aplicações presentes no projeto desenvolvido nesta tese.

2.8 Software existente no mercado

Existe uma miríade de programas que se propõem a resolver este problema da gestão de recursos remota. Uns mais completos que outros, uns mais caros que outros (alguns como freeware), todos resolvem o problema à sua maneira. Começar-se-á pelos programas com custo fixo, neste caso o Remote Task Manager (DeviceLock, 2003). Este permite, por exemplo, monitorizar tarefas e serviços que se encontram a correr em máquinas remotas, saber as características dos processos, terminar corretamente tarefas, modificar prioridades de processos, arquivar logs de eventos, e oferece a possibilidade de monitorizar os processos à distância.

Pela Ilustração 5 em baixo, pode-se ver que é possível criar logs sombra, ou seja, monitorizar o sistema com o registo de eventos de forma a que o utilizador não se aperceba, verificar processos em execução, etc. Contudo, o ambiente apresentado , é algo confuso, o que pode levar o utilizador a não se sentir tão atraído por este software, quando existem programas que oferecem uma apresentação mais cuidada tornando mais simples a sua utilização.

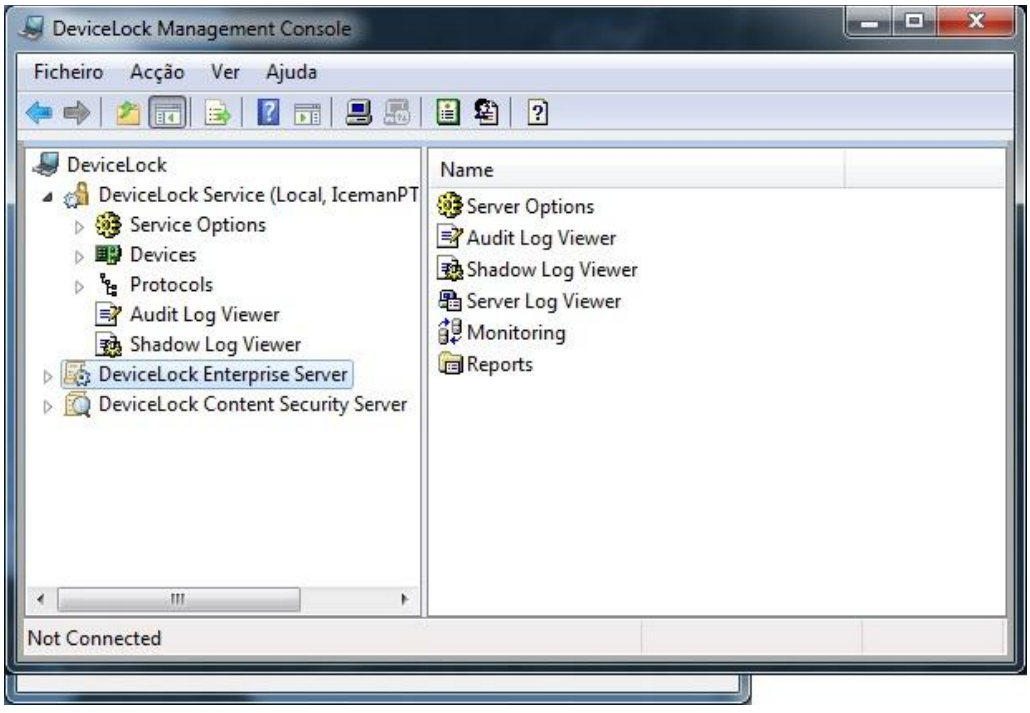

Ilustração 5 - Aspeto geral do programa

O segundo programa chama-se Remote utilities(Usoris systems, 2009), este programa possui características mais complexas, permitindo desligar/reiniciar computadores remotos, efetuar a gestão de inventários, realizar instalação remota, execução remota de programas, monitorização remota de processos, linha de comandos remota. Possui também a característica de Remote Desktop, ou seja ver o ambiente de trabalho do computador remoto em questão. Para utilizar o programa, primeiramente, procede-se à instalação do cliente nas máquinas remotas, configurando-se para tal, uma palavra passe, de forma a atribuir um nível de segurança à comunicação entre estas e o servidor, tal pode ser vislumbrado na Ilustração 6 abaixo.

Ilustração 6 - Escolha de password no cliente

Após isto, é necessário proceder ao adicionamento dos clientes ao programa servidor, clicando no botão "add computer", para adicionar uma nova máquina, botão esse que pode ser identificado, no canto superior esquerdo da imagem 7 em baixo.

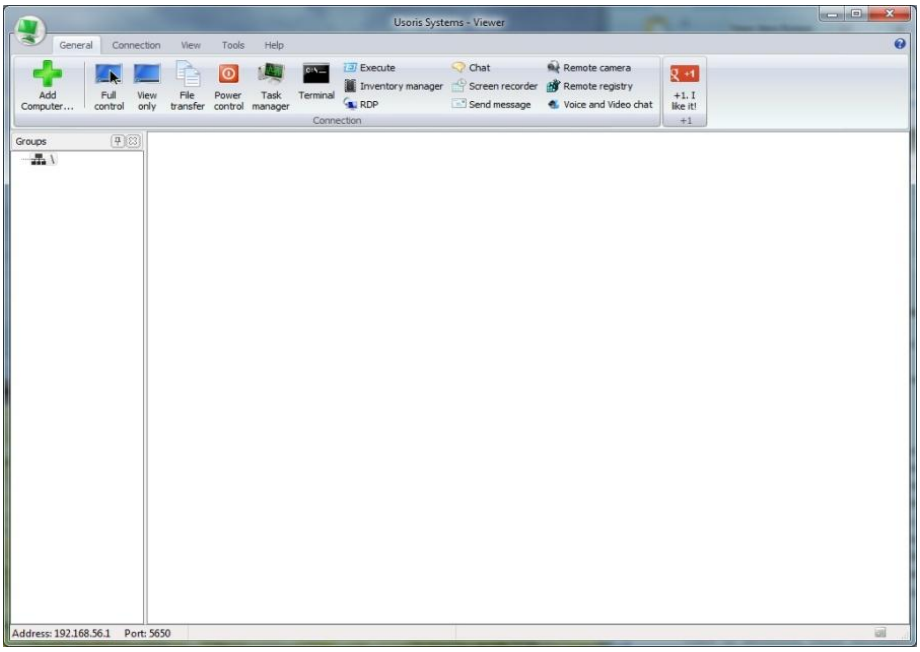

Ilustração 7 - Aspeto geral do programa

Após clicar no botão, é necessário introduzir o endereço da máquina que se pretende adicionar, no campo IP Address, como se pode ver na Ilustração 8.

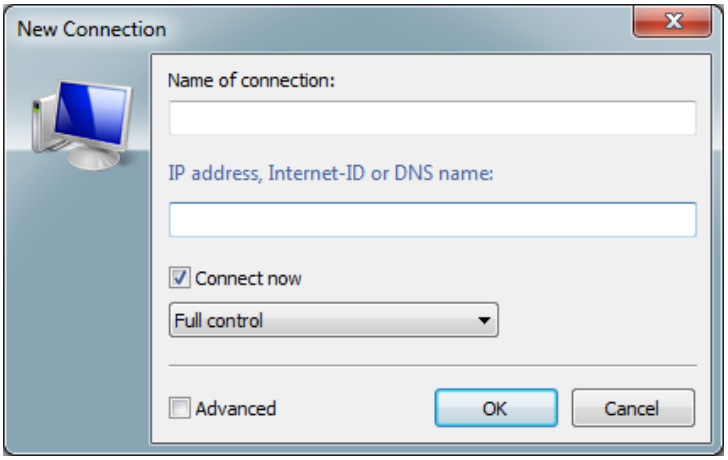

Ilustração 8 - Introdução do endereço IP da máquina remota.

No final, o nome da máquina irá aparecer em baixo da barra de ferramentas (Ilustração 9), e poderá fazer uma ligação a esta, para controlar o desktop remoto, ou então, controlar os processos remotamente (Ilustração 10), ou ainda abrir o terminal remoto (Ilustração 11), entre outras.

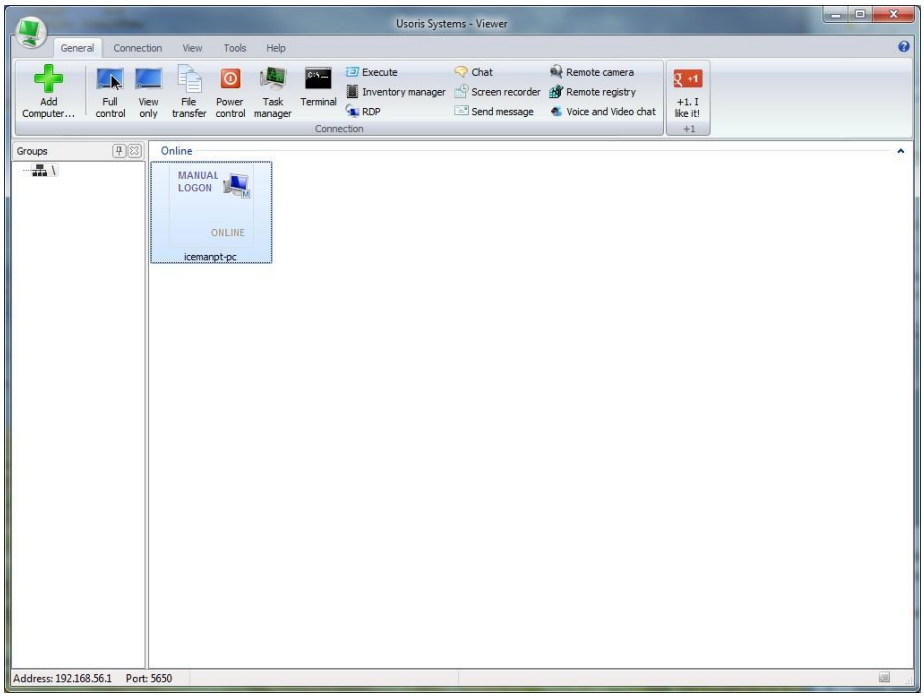

Ilustração 9 - Apresentação do ecrã principal, com uma máquina inserida

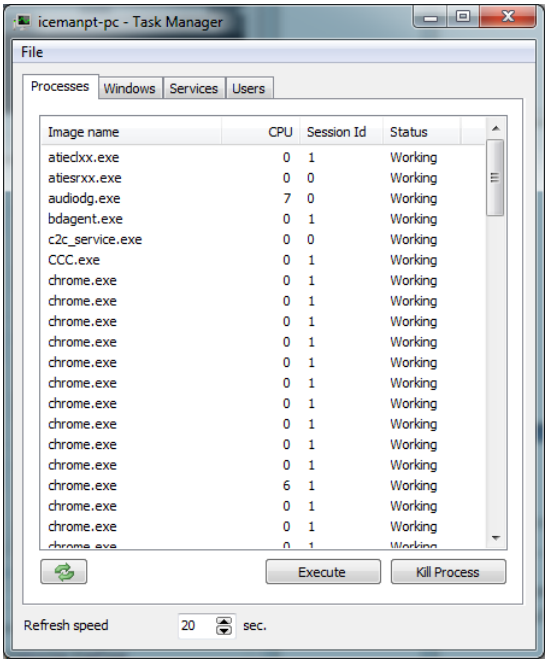

Ilustração 10 - Remote task manager

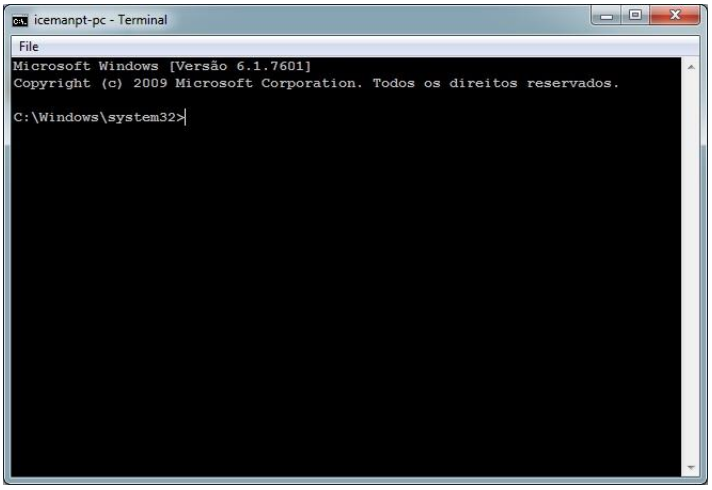

Ilustração 11 - Terminal remoto

Como se pode ver pelas Ilustrações apresentadas, é possível ver e executar processos, e também pará-los; também é possível modificar o tempo até à próxima atualização dos processos em execução, já que a recepção desta informação é feita quando se clica no botão e vai sendo atualizada conforme o tempo que esta estiver aberta, ou seja, neste caso está selecionado o valor de 20 segundos, se esta janela se fechar em menos de 20 segundos, esta nunca será atualizada.

Neste exemplo, devem-se reter as seguintes características:

- ➔ Desligar/Reiniciar computadores remotos; é possível ordenar o encerramento de máquinas sem a presença física no local;
- ➔ Gestão de processos remota (lista de processos, aplicações e serviços em execução); Permite fazer exatamente tudo o que vimos na ferramenta anterior;
- ➔ Linha de comandos remota; uma das características mais importantes, pois torna possível o acesso à linha de comando de uma máquina da Lan, de forma remota, dando acesso ao gestor do sistema, a programas e a batch files, que só seria possível ter acesso no caso de ter acesso físico à máquina em questão.

Este programa é bem mais completo e intuitivo na sua utilização que o primeiro apresentado. O que pode levar também a uma escolha mais fiável e a uma tomada de decisão mais rápida no caso de se estar indeciso entre os dois.

Aplicações freeware

Passando às aplicações sem custos, por norma estas não são tão completas quanto as pagas, sendo portanto necessário, neste caso, instalar e manter vários programas ao mesmo tempo, de forma a poder ter parte das características presentes nos softwares mencionados anteriormente. Consultou-se portanto um sítio na internet, que possuía uma lista de ferramentas de administração grátis, chamado de TalkTechToMe (Software, 2005). Neste sítio encontraram-se ferramentas interessantes, algumas descritas de seguida. O programa Process Monitor (Microsoft, 2002-c)' é uma ferramenta de monitorização em tempo real, que mostra a actividade de processos, serviços e sistema de ficheiros. Serve acima de tudo, como ferramenta para obter informação acerca destas características, mostrando por exemplo a arvore de processos, de forma a saber a relação entre eles. As características mais importantes a reter, são:

- ➔ Captura detalhada e confiável, da informação de cada processo, incluindo linha de comandos e identificação de utilizador e sessão;
- ➔ Árvore de processos, mostra a relação entre todos os processos referenciados numa procura;
- ➔ Log de todas as operações aquando do boot do sistema operativo.

Outro programa sem custos, também interessante, é o ManagePC (ManagePC, 2006), este é construído em .Net, e pretende-se que seja capaz de inventariar todos os aspetos dos computadores presentes numa rede, tais como hardware, software, serviços, processos, utilizadores locais e grupos, etc. O destaque deste programa, vai para a sua capacidade de efetuar o que foi acima mencionado, através de interfaces familiares, tais como:

- ➔ remote desktop;
- ➔ VNC;

Oferece também a possibilidade de, por exemplo, parar serviços remotamente. Após a enumeração destas características, é possível elaborar um quadro comparativo, de forma a classificar os programas quanto às suas características. Esta análise comparativa, tem como objetivo relacionar variáveis que se tornam cada vez mais importantes na vida ativa de uma empresa, que são as características globais de uma aplicação, o preço pago por esta aplicação

(pois nem todas as empresas possuem o mesmo orçamento) e compatibilidades com sistemas operativos. Quanto às características, estas serão divididas em, remote desktop, gestor de tarefas remoto, terminal remoto, identificação de processos desconhecidos.

Nome	RD[1]	GTR[2]	TR[3]	IPD[4]	Compatibilidade	Preço
Remote Task Manager [1]	n	s	n	n	1 Fraca (só até Windows 2000)	1 (Para uma licença global, cerca de 4000€)
Remote Utilities [2]	s	s	s	n	4 Boa (até Windows 7 64bit)	3 (Versão completa helpdesk $700)
Process Monitor[4]	n	s	s	n	5 Muito boa, desde Windows XP SP2	5 Sem custos
ManagePC[5]	s	s	n	n	4 Boa, requer .Net 3.5	5 Sem custos

Legenda: 1 - Remote Desktop; 2 - Gestor de Tarefas Remoto; 3 - Terminal Remoto; 4 - Identificação de processos desconhecidos;

Como era de esperar, as ferramentas pagas são por sua vez as ferramentas mais completas e mais apelativas, não fosse o preço algo elevado, mesmo a menos custosa apresenta um valor algo elevado para muitas empresas médias/pequenas. Já nas aplicações freeware, apesar de até possuírem características interessantes, pecam por terem de recorrer a programas secundários tais como VNC, para efetuarem as mudanças no sistema de que são capazes.

Sendo assim, é possível afirmar que a ideia desta tese é válida, pois propõe a criação de uma ferramenta, que mesmo podendo não possuir algumas características mencionadas anteriormente, utilizará tecnologias presentes na plataforma .Net da Microsoft, sem ser necessário recorrer a programas de terceiros, para se fazer valer das características mencionadas, o que se traduz em menos um inconveniente para o gestor de rede/sistemas. Além de avisar o próprio administrador para a presença de processos que estão a atuar fora do âmbito da lista de conformidade da empresa, pode fechá-los e proibir por completo a sua utilização.

Pode-se afirmar, que das características referidas, apenas a característica de Desktop Remoto não está presente na aplicação desenvolvida na tese, pois tratando-se de uma aplicação desenvolvida para melhorar a segurança de uma rede, e não para servir de help desk, não faria muito sentido possuir esta característica.

3. Aplicação Quimera

3.1 Tecnologias e ferramentas escolhidas

Tendo em conta o objetivo do projeto, ou seja, a projeção e construção de uma aplicação que permita a gestão de recursos e processos de forma remota, dando o poder ao Administrador, seria necessário construir o software utilizando uma arquitetura cliente-servidor, pois apesar da carga atribuída ao servidor, no caso de múltiplas comunicações em simultâneo, ser maior do que no paradigma P2P, não era do interesse do projeto dar a capacidade de troca de informação entre os clientes, só se pretendia dar a possibilidade de comunicação entre o servidor e os vários clientes. Como a segurança neste projeto é levada muito a sério, foi utilizada cifragem nos logs de sistema criados pela aplicação servidor, utilizando o algoritmo de cifragem AES, bem como na lista de conformidade presente nos clientes, pois ainda se está longe de uma técnica que permita aceder à informação cifrada com este algoritmo em tempo útil, sem conhecer a chave. Na segurança de comunicação, ou seja, na troca de informação entre os clientes e o servidor, foi utilizada a cifra Merkle-Hellman, diminuindo assim a possibilidade de injeção de comandos, caso se conheça o IP do servidor. Sendo este software desenvolvido para ser utilizado em ambiente Windows, faria sentido utilizar uma tecnologia que funcionasse de forma harmoniosa com os API deste ambiente. Para tal utilizou-se a conjunção Visual Studio 2010 (IDE) e C# (Linguagem de Programação). Esta dupla, permite o desenvolvimento rápido de aplicações robustas e poderosas. No caso do tipo de ligação, ao selecionar o tipo de sockets, como é preferível haver a necessidade de estabelecer uma ligação ao cliente/servidor para comunicar algo, ao invés de enviar informação e não ter certeza se esta foi entregue, escolheram-se os sockets do tipo TCP/IP em detrimento dos sockets de tipo UDP.

3.2 Explicação da cifragem utilizada, no contexto do programa.

Como já foi explicado no capítulo introdutório, foram utilizados processos de cifragem de forma a poder haver comunicação segura entre as aplicações, e a poder guardar-se ficheiros importante sem que estes possam ser lidos por outras aplicações. Depois de estes terem sido explicados de forma mais teórica anteriormente, estes serão explicados agora no contexto do programa, ou seja, onde e quando são utlizados e porquê.

3.2.1 AES

Este algoritmo foi utlizado de forma a proteger os ficheiros gerados pela aplicação, impedindo-os de serem lidos por utilizadores maliciosos, utilizando outras ferramentas. Este ficheiros são os logs de sistema, que guardam toda a atividade do programa servidor e os processos autorizados e negados (aplicação cliente). Os logs, vão sendo criados num ficheiro, cifrado e fechado ao encerrar o programa, sendo lidos do ficheiro cifrado ao abrir o programa. Os processos autorizados e negados funcionam da mesma forma, estes são adicionados e/ou removidos de listas durante a operação da aplicação, quando esta é fechada, são guardados em ficheiro, sendo lidos do ficheiro, e colocados nas respetivas listas, quando o programa inicia. Sendo assim, pode-se resumir o funcionamento desta cifragem a dois eventos, quer no programa cliente, quer no programa servidor, Iniciação do programa, e encerramento do programa.

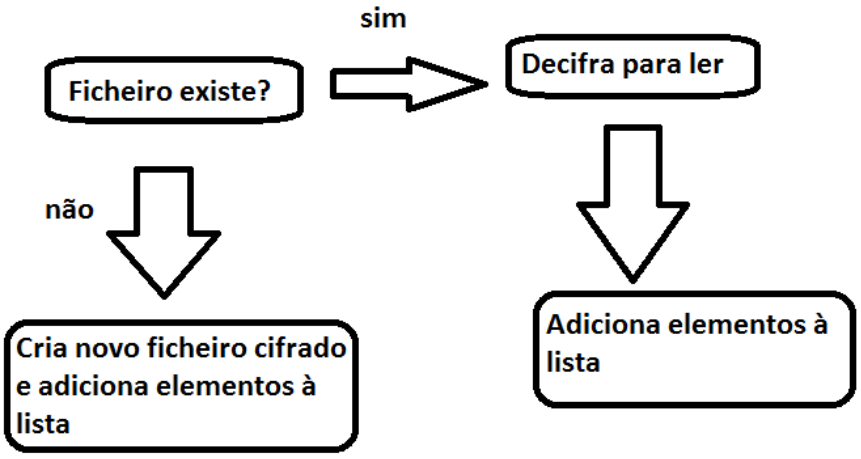

Ilustração 12 - Iniciação do progrma

Na Ilustração 12, pode-se ver que ao iniciar qualquer um dos programas (menos o programa móvel), é feita uma verificação da existência do ficheiro cifrado, caso este não exista, um

ficheiro é criado com os processos que estão de momento a correr, para criar a lista de conformidade (caso o utilizador não deseje criar uma lista manualmente, isto não se aplica aos processos não autorizados, já que estes só existirão se existirem processos autorizados em primeiro lugar), sendo os dados adicionados à lista. No caso do ficheiro existir, este é decifrado, e os dados obtidos, são adicionados à lista.

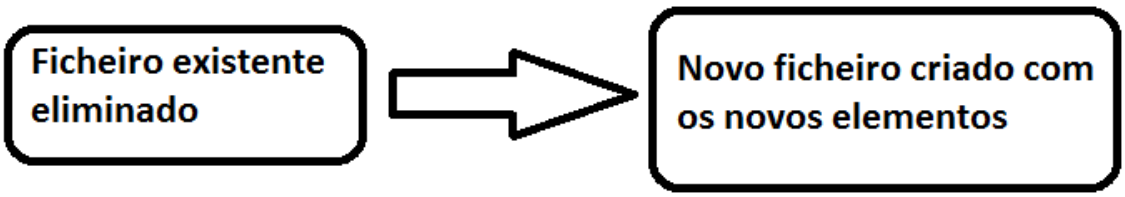

Ilustração 13 - Encerramento do programa

Na Ilustração 13, pode-se ver que o ficheiro cifrado existente, é eliminado, e é criado outro no seu lugar com os dados já atualizados, pois poderão ter existido alterações à lista de conformidade, com adições ou remoções a esta, bem como novos processos negados, e também novos eventos nos logs.

3.2.1 Merkle-Hellman

Este algoritmo foi utilizado para cifrar as comunicações entre as várias aplicações existentes nesta tese (cliente, servidor, móvel). Antes de qualquer texto ser enviado, este é cifrado, sendo transformado apenas em números, ao chegar ao destino, este é decifrado e interpretado. De referir, que sendo este algoritmo de chave assimétrica, seria necessário, para maior segurança, mudar as chaves a cada execução, mas existiria um grande problema de sincronização entre aplicações, e correr-se-ia o risco de haver interceção da transmissão da mensagem que continha as chaves, daí se ter tomado a decisão de recorrer a uma chave fixa, ou seja, imutável, e que se encontra presente nas várias aplicações elaboradas nesta tese.

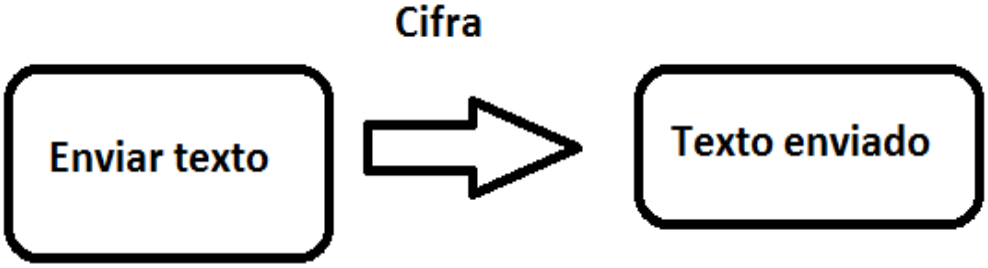

Ilustração 14 - Enviar Texto

Na Ilustração 14, pode-se ver o processo que ocorre quando o texto é enviado de uma aplicação para outra, todo ele é cifrado, e só depois enviado.

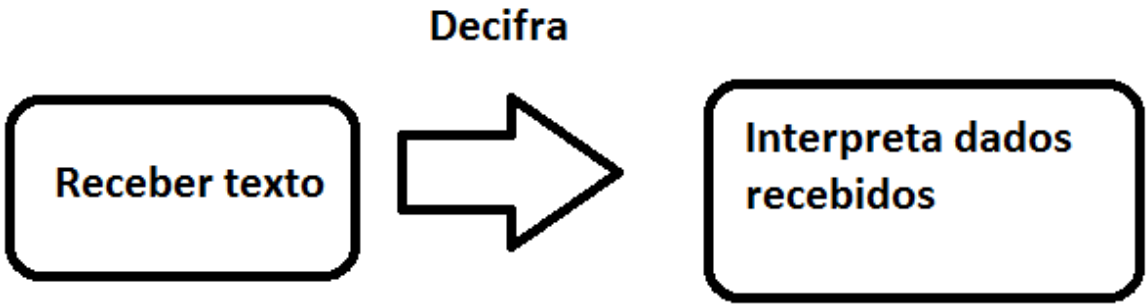

Ilustração 15 - Receber texto

Na Ilustração 15, consegue-se perceber o método empregue na receção do texto. Todo o texto recebido é decifrado, e só depois interpretado, de forma ao servidor poder responder ao cliente, e vice versa.

3.3 Diagrama de Caso de uso.

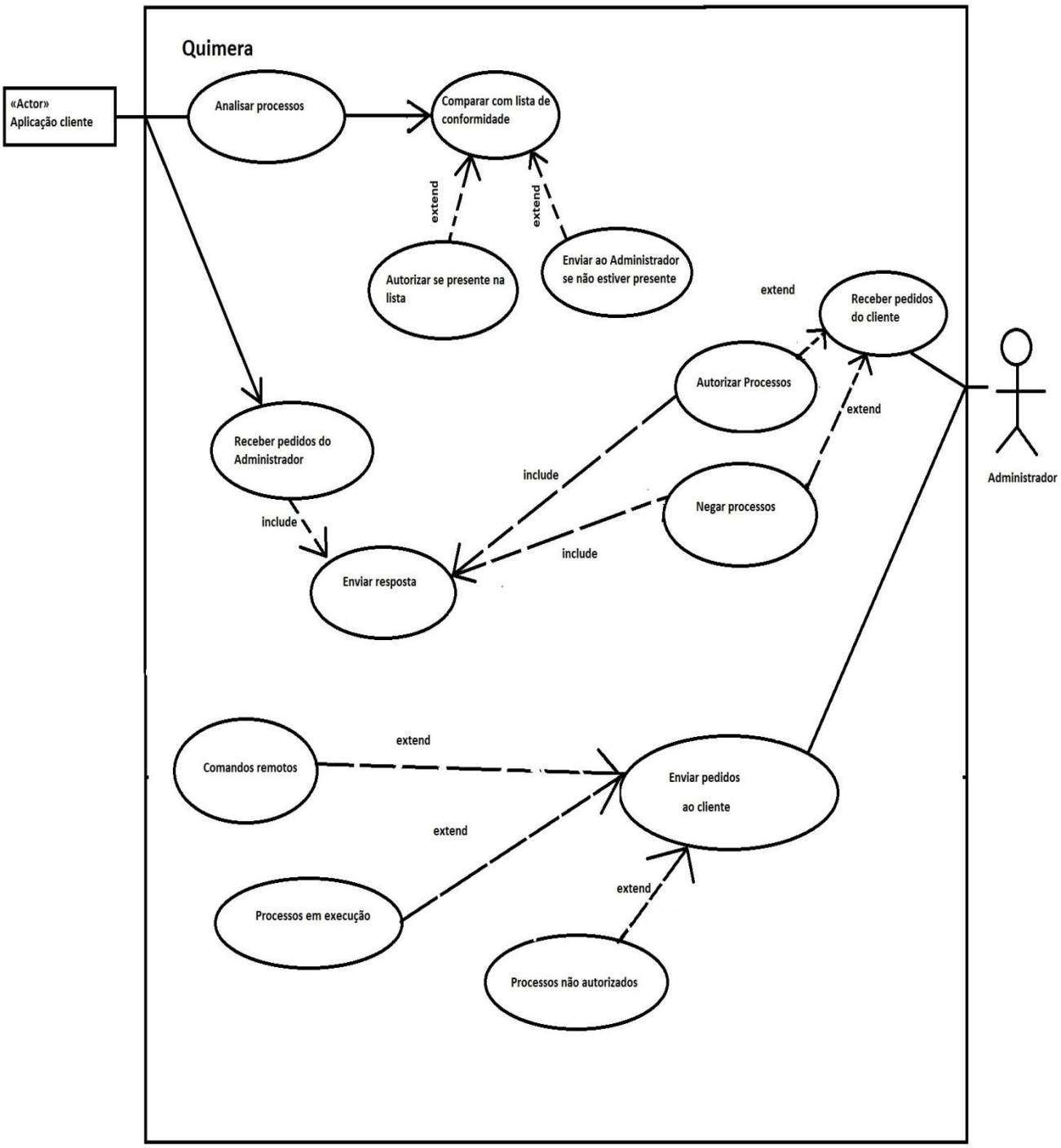

Ilustração 16 - Diagrama de caso de uso 1

A partir do diagrama anterior, ilustração 16, pode-se perceber o funcionamento que se pretende dar ao software criado. Passando a explicar, pretende-se que a aplicação cliente se encontre num constante estado de análise aos processos em execução na máquina onde esta aplicação se encontra em execução. Esta análise resumir-se-á a uma comparação com a lista de conformidade gerada. No caso de o processo se encontrar presente na lista, este é autorizado a correr automaticamente, não sendo efetuada qualquer modificação, no caso de este não estar presente, é enviado ao administrador para avaliação. A aplicação cliente também recebe pedidos do Administrador, que são respondidos na hora. Na parte do administrador, este pode receber pedidos do cliente, e enviar pedidos ao cliente. No caso de receber pedidos, como só podem ser pedidos de autorização de determinado processo, o Administrador poderá negar ou autorizar, acções essas que incluem obrigatoriamente, o envio de uma resposta. No caso de enviar pedidos ao cliente, estes só poderão ser, ou pedido de processos em execução, ou os processos não autorizador, ou então, o envio de um comando remoto.

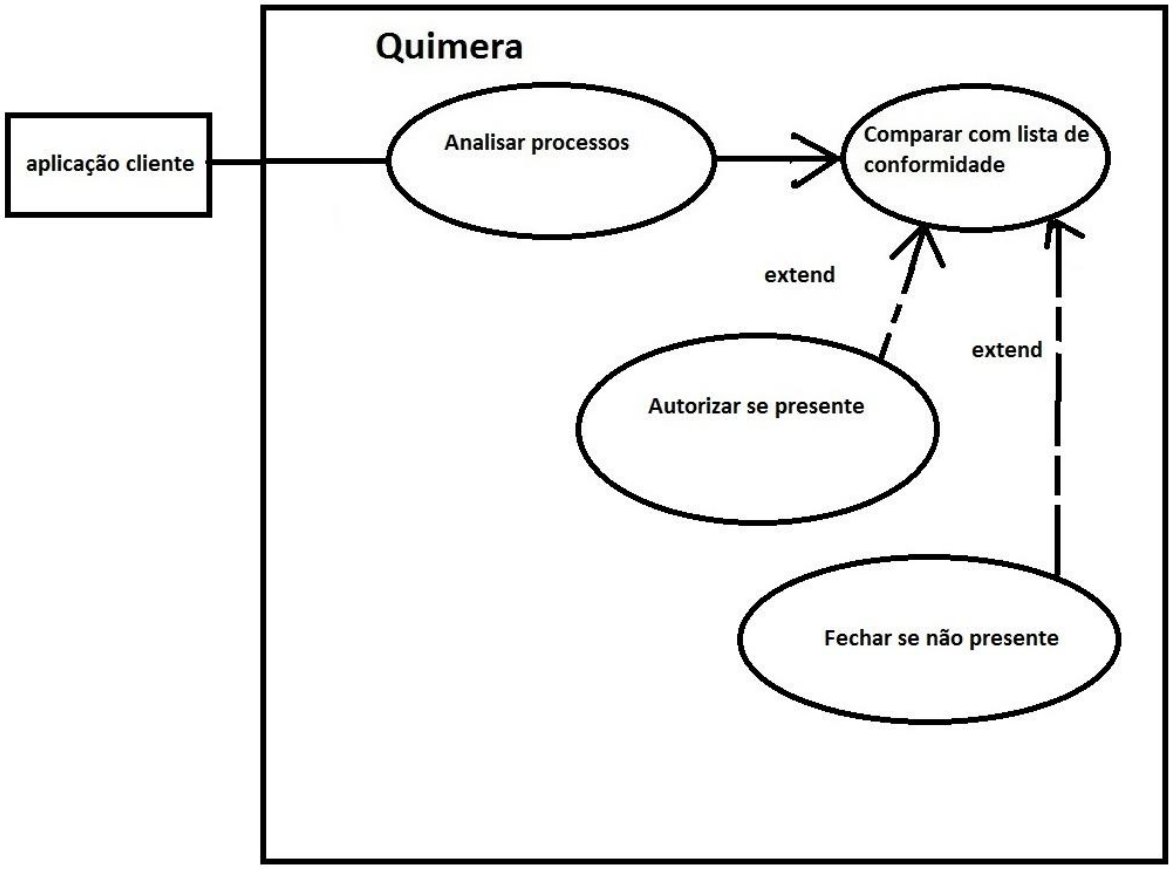

Ilustração 17 - Diagrama caso de uso 2

No diagrama presente na ilustração 17, pode-se distinguir a segunda função deste programa, que é funcionar apesar do administrador não estar presente, sendo capaz de tomar decisões sozinho, tendo em conta a lista de conformidade. Neste modo de funcionamento, o programa cliente encontrar-se-á também numa constante análise dos processos em execução. Caso o processo encontrado esteja presente na lista, este é autorizado, não havendo alterações ao sistema. No caso do processo não se encontrar na lista, este é fechado automaticamente, não podendo ser aberto.

3.4 Aspeto geral e funcionamento do programa servidor.

A aplicação servidor, contém certas medidas de segurança de forma a dificultar o acesso às ferramentas presentes nesta aplicação, por partes não autorizadas. Como tal, de forma a poder começar a utilizar esta ferramenta, é necessário proceder à autenticação do utilizador, como se pode ver pela Ilustração 18 abaixo:

Ilustração 18 - Janela Log in

Aqui será necessário ao Administrador inserir as suas credenciais, informações essas que são, nada mais, nada menos, que as informações relativas à sua conta Windows, aquela que utilizou para ter acesso ao desktop da sua máquina. Isto é possível recorrendo a um dll de sistema, o advapi32.dll. Após uma autenticação com sucesso, é apresentada a janela principal do programa, janela essa que irá servir de interface ao utilizador enquanto procede ou não a alterações nas listas de conformidade de cada máquina.

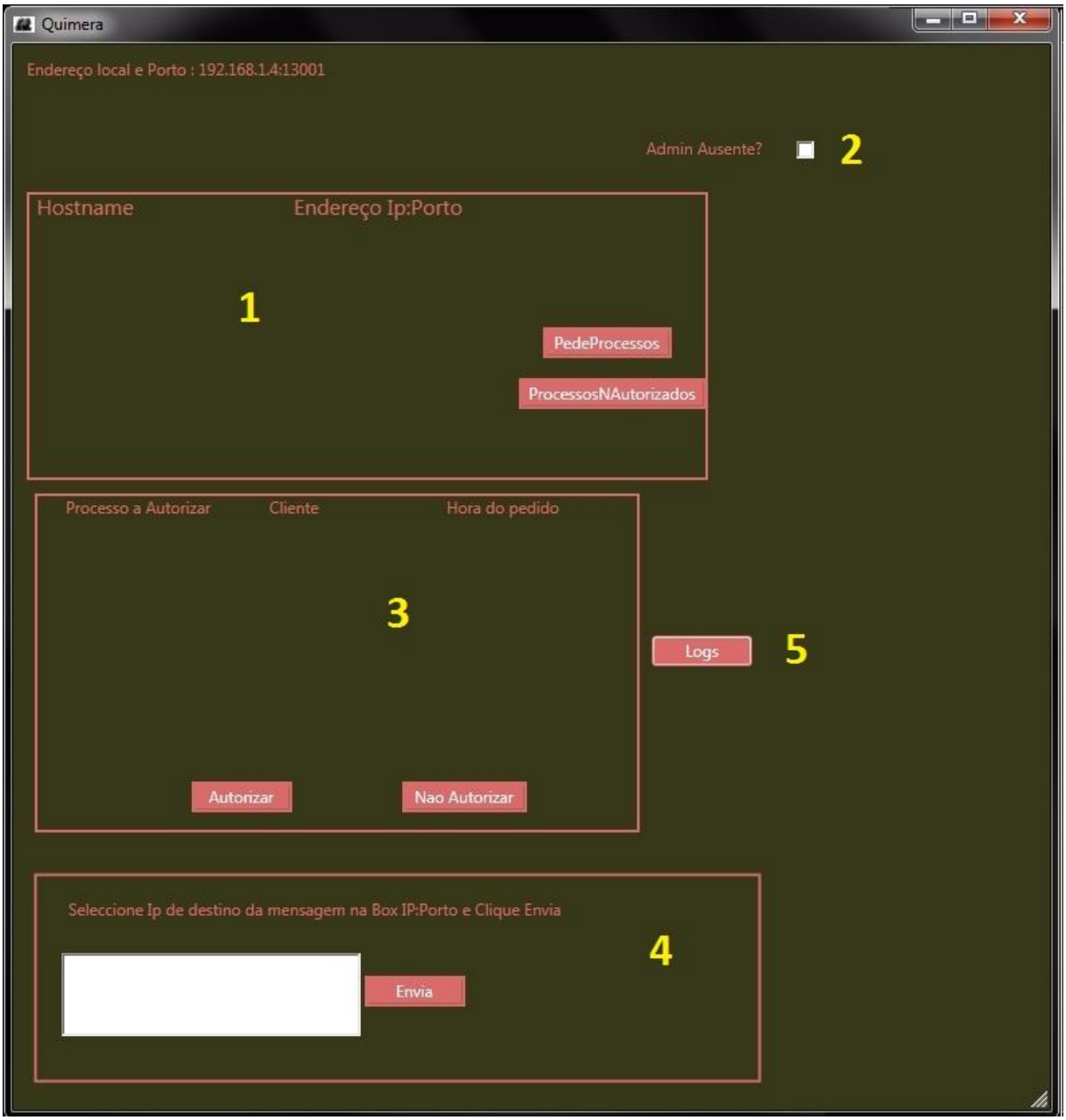

Ilustração 19 - Janela Principal

Como se pode ver na Ilustração 19, esta aplicação encontra-se dividida por zonas, neste caso cinco zonas. A zona 1, contém a informação das máquinas que se encontram de momento ligadas na rede, inclui os botões "PedeProcessos" e "ProcessosNAutorizados". Enquanto que o primeiro serve para pedir à máquina selecionada os processos em execução nesta, o segundo servirá para pedir os processos que foram desautorizados pelo administrador naquela máquina.

Ao pedir os processos, uma mensagem é enviada à máquina alvo, sendo que esta é interpretada pela aplicação cliente, respondendo ao servidor com os processos em execução. Estes são apresentados na janela da Ilustração 20.

Ilustração 20 - Janela Processos em Execução

Na janela de processos podem ver-se várias características pertencentes aos processos, como o nome do processo, o nome do dono do processo (o próprio utilizador, ou o sistema), a memória ocupada pelo processo, e por fim o local de onde este se encontra a executar. Aqui é possível também matar processos, ou desautorizar a sua futura execução, para tal basta clicar em Kill, para matar, ou em desautorizar. No caso de ser desautorizado, este será adicionado à lista negra de processos, lista essa que contém as informações de todos os processos, cuja execução foi rejeitada pelo administrador, sendo impossível voltar a ser executado sem que o Administrador o retire desta lista. A janela que contém os processos desautorizados, pode ser acedida através do clique no botão "ProcessosNAutorizados", presente na zona 1 da janela principal da aplicação. Janela essa que se pode ver na Ilustração 21.

Ilustração 21 - Janela Processos Não Autorizados

Voltando à explicação da janela principal, passando à zona 2 (Ilustração 19), verifica-se a existência de uma checkbox, esta serve para ativar o "piloto automático". O Administrador clicará aqui se por alguma razão estiver ausente da sua estação. Isto fará com que o programa servidor transmita a todos os programas clientes a informação de que o administrador se encontra ausente. Desta forma todos os pedidos de autorização de novos processos, que não se encontrem presentes nas listas de conformidade da empresa, serão automaticamente fechados nas máquinas clientes. Esta situação só irá mudar, no caso do administrador retirar o visto da checkbox, dando a informação aos clientes que o administrador se encontra de novo na sua estação.

Na zona 3 (Ilustração 19), existem 3 listas e 2 botões, essas listas irão conter a informação sobre os vários pedidos de autorização de processos que provêm dos computadores clientes (da esquerda para a direita, o nome do processo, o computador de origem, e a data do pedido). Selecionando um processo, será possível autorizar a execução, com o botão autorizar, e desautorizar, adicionando o processo à lista negra de processos, com o botão Desautorizar.

Na zona 4 (Ilustração 19), existem três componentes, uma caixa de texto, um botão e um bloco de texto, este último só é visível quando se encontra preenchido com texto. Ao clicar numa máquina na zona 1, este bloco é preenchido com o nome da máquina selecionada. Aqui é possível enviar desde simples mensagens aos utilizadores, ou comandos, como desligar, reiniciar, etc. Para tal basta preencher a caixa de texto com o que se deseja enviar, e clicar no botão enviar. Importante referir de novo, que a mensagem será enviada para a máquina, cujo nome é apresentado no bloco de texto imediatamente à direita do botão "enviar".

Na última zona, a 5 (Ilustração 19), encontra-se apenas um botão, apelidado de "Logs". Como o próprio nome indica, é possível, ao clicar neste, ter acesso aos logs gerados pela aplicação servidor.

É importante referir, que a aplicação cliente não se encontra demonstrada em imagem, nem descrita neste texto, pois sendo algo cuja visibilidade ao cliente não é do interesse do administrador, esta se encontra invisível, correndo em background. A explicação do seu funcionamento será feita no ponto 3.5.1 deste documento.

3.5 Aspeto geral e funcionamento da aplicação móvel.

Neste projeto, optou-se também por construir uma aplicação móvel, e como a escolha de linguagem e IDE, recaiu sobre o C# e Visual Studio, respetivamente, a plataforma Windows Phone 7 seria a escolha óbvia. Na Ilustração 22, é apresentado o aspeto geral da aplicação móvel.

Ilustração 22 - Aspeto aplicação móvel

Na página inicial da aplicação móvel, podem-se deslindar duas listas (máquinas e endereço), bem como dois botões (PedeProc e ProcNAuto). A lista máquinas será onde o nome das máquinas irão aparecer, bem como o seu endereço IP. O Botão PedeProc, como o próprio nome indica, serve para pedir os processos, que estão a correr em determinada máquina, já o botão ProcNauto, serve para solicitar os processos que não estão autorizados a correr, e por ventura, se quererá garantir acesso aos recursos em questão.

Ao selecionar um endereço na lista de endereços, e clicar no botão PedeProc, o utilizador depara-se com a janela da Ilustração 23.

Ilustração 23 - Janela Processos em Execução

Como se pode ver pela Ilustração 23, o utilizador poderá vislumbrar todos os processos em execução na máquina que tinha selecionado anteriormente. São-lhe também apresentados dois botões, "Fechar" e "Desautorizar". Estes botões têm exatamente a mesma função do botões presentes na janela de processos do programa servidor, janela essa aberta ao clicar no botão "PedeProcessos", na zona 1(Ilustração 19) da janela principal do programa servidor. Se ao invés de clicar no botão "PedeProc", na janela principal da aplicação móvel, o utilizador decidir consultar os processos não autorizados de determinada máquina, clicando no botão "PrcoNAuto", será apresentado ao utilizador a janela que contém os processos da lista negra daquela máquina, tal como acontece com o programa servidor(Ilustração 19).

3.6 Explicação a nível funcional

Nesta secção, passar-se-á a explicar o funcionamento de cada pedaço de software duma forma simples.

3.6.1 Programa Cliente

A aplicação cliente é constituída por várias classes. São elas a classe cifra, responsável pela cifragem da comunicação; Ficheiro, responsável pela criação de ficheiros cifrados; Listener, responsável pela criação do servidor de sockets, que escuta a comunicação no porto escolhido do endereço local; Operações, responsável por efetuar determinada ação ou chamar determinada função de uma classe para responder a um pedido específico do servidor; Processo, efetua todas as ações determinadas pelos Administradores da rede que estejam diretamente relacionados com processos e por fim, a classe SocketCliente, que permite o envio de mensagens, de forma a responder a pedidos do servidor.

Passando à explicação do funcionamento (Ilustração 24), ao iniciar o pc cliente, a aplicação cliente é também iniciada, dando início por sua vez à classe listener, para poder ouvir a comunicação que vem na direção desta, e à classe SocketCliente, de forma a invocar a classe cifra, para cifrar o texto da mensagem e enviar as suas informações ao servidor, que serão posteriormente utilizadas para este comunicar com o cliente em questão. Após este passo, caso não exista uma lista de conformidade criada pelo responsável da empresa, esta é gerada utilizando para o efeito a classe Ficheiro, que irá popular a lista com os nomes dos processos em execução no computador.

Ao mesmo tempo que a aplicação cliente é iniciada, existe uma rotina dentro da classe processo, que é iniciada num novo thread, e que existe para verificar quando há um novo processo a correr.
Caso encontre um novo processo, verifica-se se se encontra autorizado.
Se sim, continua aberto, se não verifica-se se o Administrador está presente; caso não esteja, este é fechado imediatamente. No caso do Administrador estar presente invoca-se a classe SocketCliente que por sua vez pede à classe Cifra para codificar o texto da mensagem.
Envia-se o nome deste novo processo ao servidor para aprovação. Ao receber a informação do feedback do Administrador, o texto é passado à classe operações, que antes de passar a ordem de autorização ou de desautorização à classe Processo, passa o texto pela classe Cifra de forma a ter acesso ao texto simples da mensagem, podendo assim tratar de adicionar o processo à lista dos autorizados mantendo-o aberto, ou dos desautorizados, fechando a aplicação alvo no processo.

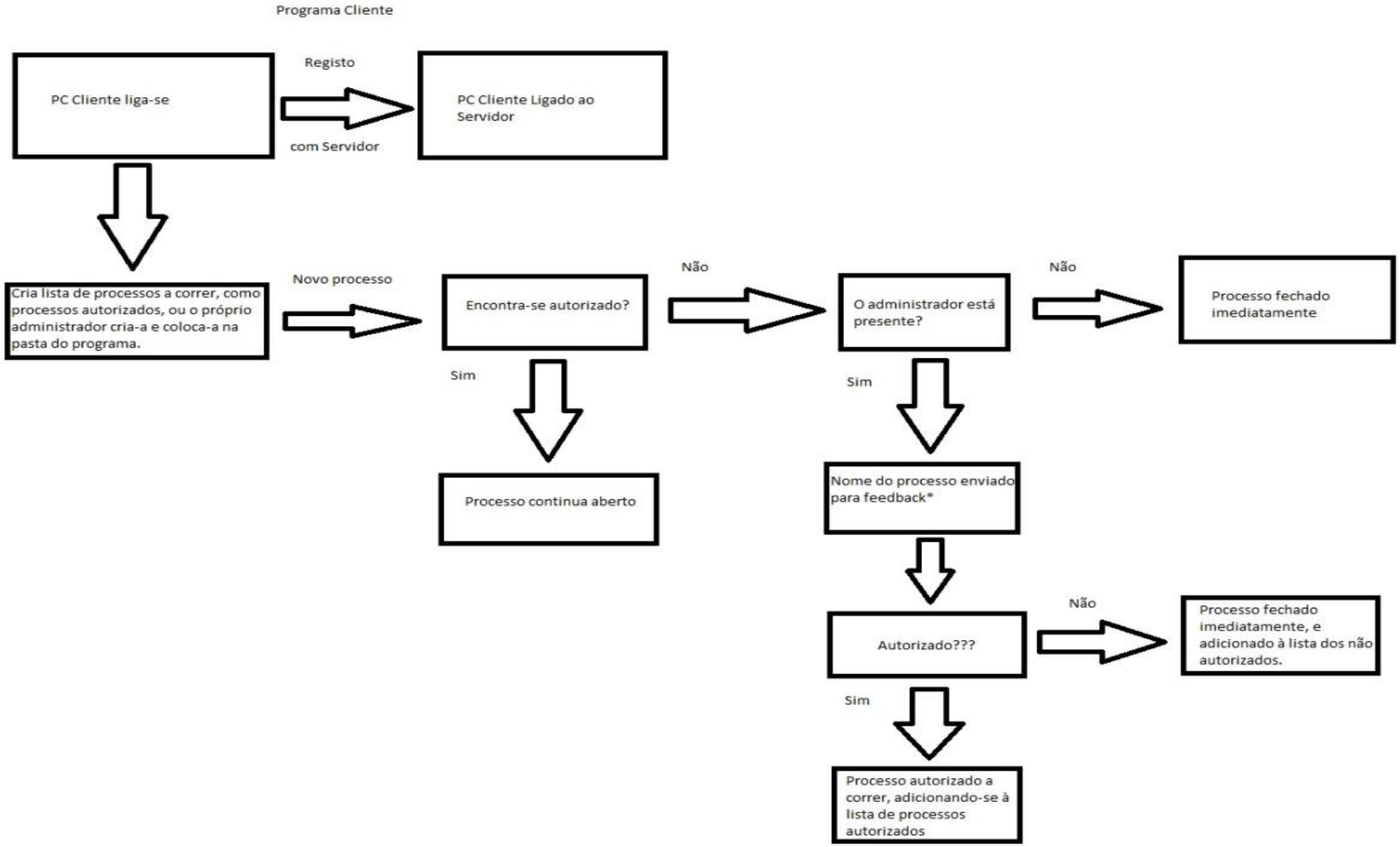

Ilustração 24 - Funcionamento Programa Cliente

3.6.2 Programa Servidor

O programa Servidor, é mais simples de explicar, já que os vários eventos não se encontram encadeados, estando estes isolados uns dos outros. Este possui várias classes na sua constituição, pois tal como no programa cliente, teve-se o cuidado de desenvolver a aplicação de forma a que fosse fácil efetuar futuras alterações. As classes incluem, a classe Cifra, responsável pela comunicação segura entre aplicações; a classe ClienteSocket, para envio de mensagens às outras aplicações; a classe listener, que cria o servidor de Sockets TCP; a classe operações, responsável por decidir que ação efetuar quando recebe informação das aplicações clientes;

O servidor recebe os valores do cliente (hostname, ip), e guarda-os para uso futuro. Isto processa-se da seguinte forma, a classe listener escuta uma comunicação, decifra a mensagem utilizando a classe Cifra que passa o texto simples para a classe operações, que o interpreta, e efetua a ação associada ao pedido feito, neste caso, guardar as informações do cliente.

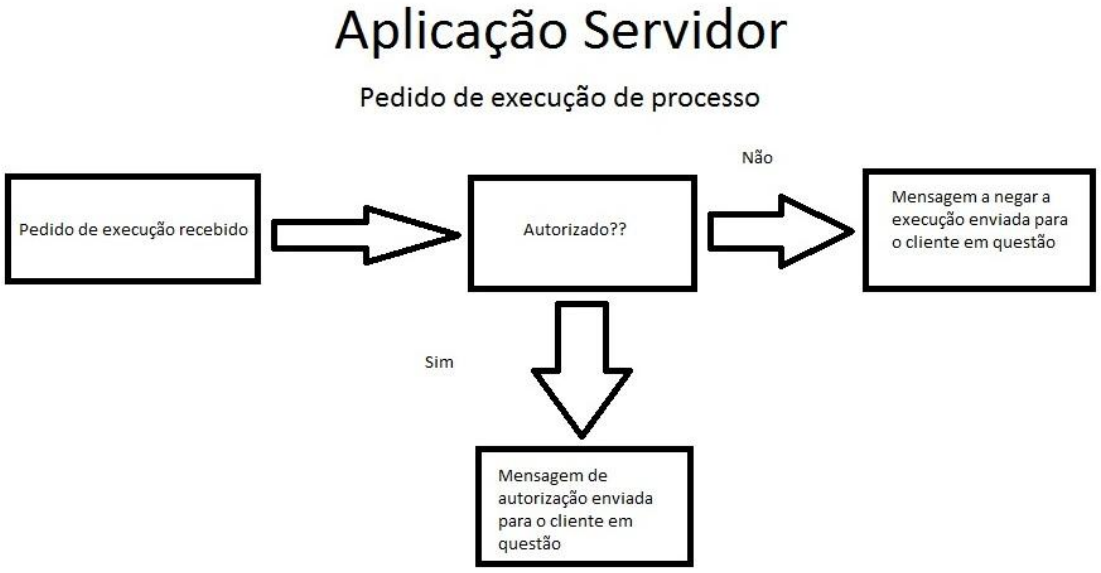

Ilustração 25 - Pedido de execução de processo

Na imagem acima (Ilustração 25), apresenta-se o exemplo do pedido de autorização de um processo por parte do cliente ao servidor. O cliente envia o nome do processo a autorizar, se este for autorizado, o servidor envia uma mensagem de autorização com o nome do processo, se não for autorizado, é enviada uma mensagem de desautorização. Programaticamente, isto processa-se da seguinte maneira, o servidor recebe a informação e decifra-a com a classe Cifra, depois passa-a à classe operações, que por sua vez a coloca na lista de processos a

autorizar. Quando o administrador se decide por fazer algo quanto ao processo, ou autoriza, ou nega a execução deste. Se autorizar, invoca-se a classe SocketCliente que utiliza a classe Cifra novamente para transformar o texto em texto cifrado e envia uma mensagem com o nome do processo, e uma ordem de autorização do mesmo; se negar, repetem-se os mesmos passos, mas com uma ordem desautorização do processo.

Ilustração 26 - Pedido de processos

Como se pode ler na Ilustração 26, o servidor pede os processos ao cliente, e este responde enviando os processos em execução de volta ao servidor, onde este os apresenta numa nova janela (ilustração 26). Falando em classes, a classe SocketCliente pede à classe Cifra para transformar o texto simples, e envia o pedido ao Cliente, este ao responder com a informação pedida, a informação é decifrada novamente pela classe Cifra, e a janela de Processos, é invocada para mostrar os processos em execução.

O utilizador , além de poder pedir os processos em execução e negar a execução a algum deles, pode também pedir os nomes dos processos não autorizados, e autorizá-los. Isto permite que haja uma certa margem de erro para o Administrador, podendo alterar na hora a lista de conformidade da empresa, se houver ordens superiores para estas alterações claro está.
Por fim, proceder-se-á à explicação da aplicação móvel. A aplicação móvel, ao ser muito mais simples que as aplicações desktop, além das classes obrigatórias, ou seja, as classes que representam as janelas, possui apenas uma extra, que é responsável pela comunicação por sockets.

Ao contrário do servidor de sockets presente nas outras aplicações, devido a limitações neste sistema operativo móvel, apenas é possível receber informação, se antes se proceder a uma ligação a um servidor remoto, enviar alguma informação, e receber a resposta do servidor na forma daquilo que foi pedido. Esta é uma forma de ultrapassar esta dificuldade, que não irá ser necessário uma vez que esteja disponível o Windows Phone 8, pois este irá partilhar o Kernel com o Windows 8, possibilitando assim, a utilização de exatamente as mesmas bibliotecas que no Sistema Operativo principal.

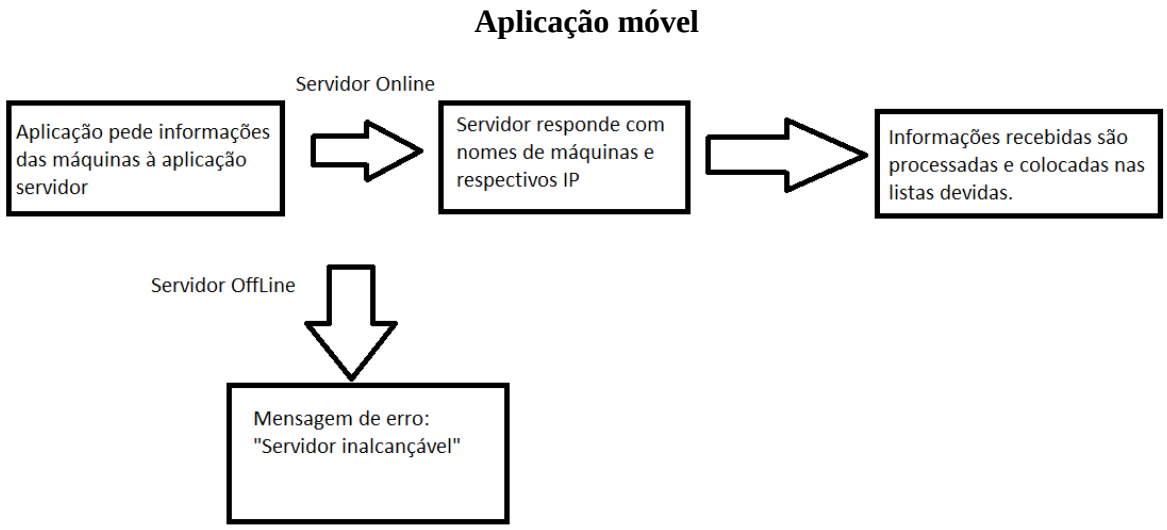

Ilustração 27 - funcionamento aplicação móvel

Este evento, demonstrado acima na Ilustração 27, descreve os passos efectuados aquando do inicio da aplicação. Mal o utilizador abre o software no seu smartphone, este irá comunicar com o servidor, pedindo as informações relativas às máquinas que se encontram ligadas a este, se o servidor estiver ligado irá responder com as informações pedidas, e após o processamento desta informação ser feito, as características de cada máquina, serão colocadas na lista devida para utilização posterior do utilizador (Ilustração 27).

Pedido de processos

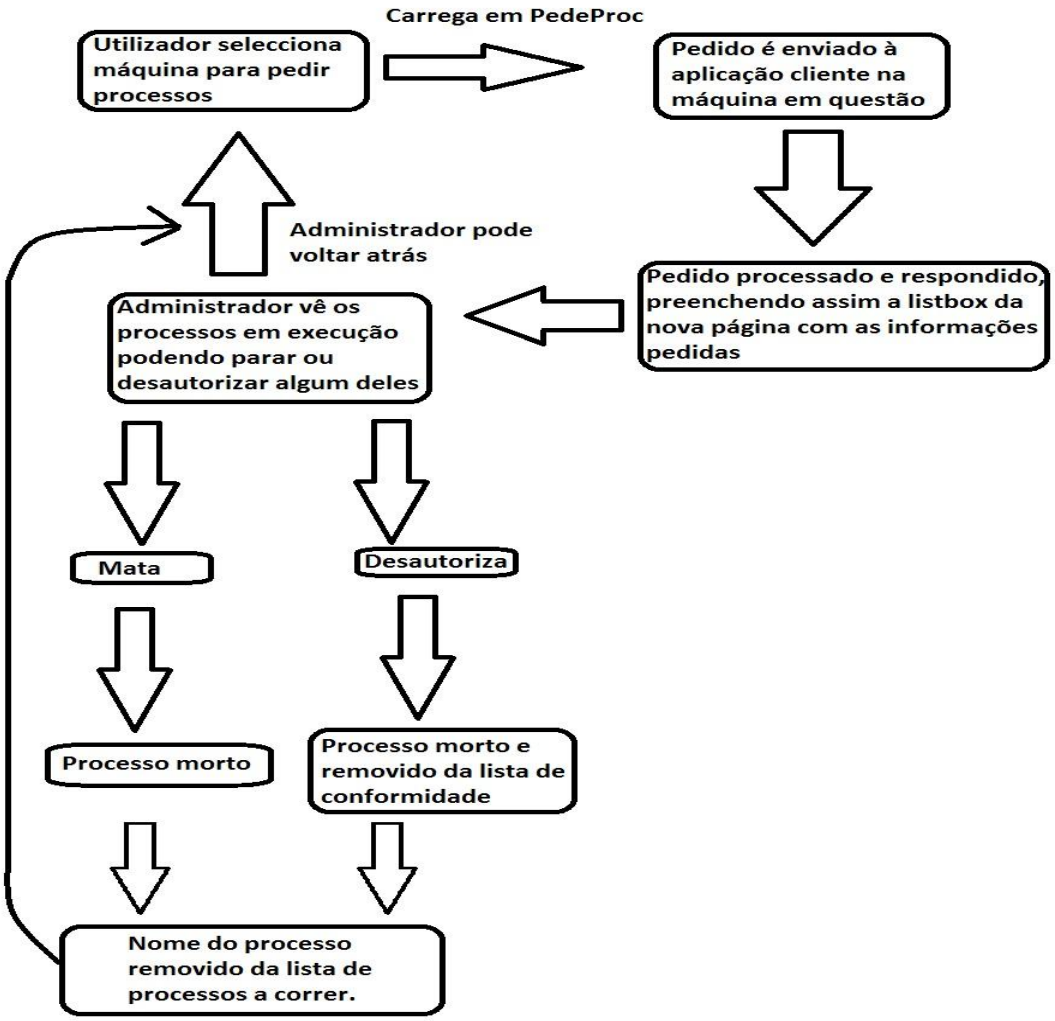

Ilustração 28 - Pedido de processos

Este esquema (Ilustração 28) demonstra de forma muito resumida o que acontece quando um utilizador decide pedir os processos a correr em determinada máquina naquele momento. Para o fazer, como é óbvio, o Administrador necessita de selecionar previamente a máquina à qual vai fazer o pedido. De seguida a aplicação móvel irá efetuar um pedido ao servidor, para este lhe responder com os nomes dos vários processos, esta irá receber a resposta e irá processá-la apresentando os nomes dos vários processos a correr numa nova página, preenchendo uma lista, como se pode ver na Ilustração 28. Depois de apresentados os processos, o utilizador pode voltar atrás e selecionar uma nova máquina, ou fechar e/ou desautorizar um processo. No caso de voltar atrás, este voltará a ser apresentado com o ecrã de seleção de máquinas. Caso decida parar (matar) um processo, este irá ser fechado na máquina alvo.
Na hipótese de querer desautorizar um dos processos, este será fechado, e removido da lista de conformidade presente em determinada máquina. No final, todos os processos fechados e/ou desautorizados serão removidos da lista apresentada acima. Por fim, o utilizador pode voltar para a página anterior, e selecionar uma nova máquina para interagir.

4. Aplicação a um caso real e apresentação de resultados.

Vivemos no tempo da informação, numa era em que aceder a determinado conteúdo, está a segundos de distância, já longe vão os tempos em que pesquisas para trabalhos e afins, eram feitas em Bibliotecas, recorrendo a livros; esta porta entreaberta denominada internet, ou também denominada autoestrada da informação, dá-nos acesso instantâneo a conteúdos, conversas com familiares distantes, etc. Todas estas vantagens trazem também perigos, como se pode ver nos telejornais, e meios de comunicação no geral, em crianças que são raptadas e/ou maltratadas depois de combinarem encontros com estranhos através de redes sociais. Este problema, e para não arcarem com responsabilidades acrescidas, levou certas escolas, senão mesmo todas, a colocar sistemas de filtração por proxy nas suas redes internas, para que os alunos que utilizam a rede interna da escola para aceder a conteúdos online, não tenham a possibilidade de aceder a redes sociais, e plataformas semelhantes, de forma a poder minimizar o risco de tais situações virem a ocorrer no futuro. Ora, recorrendo à imensa quantidade de informação que circula online, é possível arranjar forma de contornar o proxy, com um qualquer programa que crie tuneis VPN (Virtual Private Network). O Bibliotecário da Escola EB 2,3/S de Tarouca, reparou na utilização de um programa apelidado de Ultrasurf, programa este que permite o acesso a todos os sítios na internet através de um túnel VPN.

4.1 Explicação da experiência.

Foi depois de uma longa conversa com o bibliotecário dessa Escola, que surgiu a ideia de fazer uma pequena experiência. Na primeira parte desta, colocar-se-ia um programa que faria uma espécie de log, de alguns dos computadores presentes na biblioteca, sendo que este registaria todos os processos executados durante este tempo, de forma a poder proceder-se a uma análise do tipo de programas executados pelos alunos. O registo foi efectuado da seguinte forma, ao iniciar a máquina, o programa regista todos os processos em execução, sendo que a cada novo processo que surge, é adicionado à lista já existente, ou seja, se um aluno abrir o Microsoft Word, e este ainda não estiver presente na lista gerada, este será adicionado ao registo. No final dos 3 dias, apenas estarão presentes os vários nomes/donos/caminhos dos processos que foram executados em determinada máquina, sem repetição destes campos. Na segunda parte, colocar-se-ia nos vários computadores da biblioteca o programa cliente, já com a lista de conformidade criada tendo em conta os resultados obtidos na primeira parte da experiência, e o programa servidor na máquina do bibliotecário em modo automático, já que interessa saber a capacidade de negação de processos fora da lista de conformidade, voltando-se a monitorizar os processos em execução. Desta forma poder-se-ia fazer um estudo comparativo entre o que os alunos executavam sem controlo, e aquilo que eles conseguiam abrir após a instalação da Quimera nos computadores requisitados por estes.

4.2 Análise de resultados da Parte 1. Elaboração da lista de conformidade.

Durante 3 dias registaram-se os programas executados pelos alunos, sendo estes apresentados nos quadros seguintes. Começa-se pelo computador "Biblio1".

Processos Biblio1	Dono do processo	Caminho do processo
crss.exe	System	C:\Windows\System32\
dwm.exe	Biblio1	C:\Windows\System32\
explorer.exe	Biblio1	C:\Windows\
lsass.exe	System	C:\Windows\System32\
lsm.exe	System	C:\Windows\System32\
Idle	System	C:\Windows\System32\
SearchIndexer.exe	System	C:\Windows\System32\
Services.exe	System	C:\Windows\System32\
Smss.exe	System	C:\Windows\System32\
Iexplore.exe	Biblio1	C:\Program Files\Internet Explorer
Spoolsv.exe	System	C:\Windows\System32\
Svchost.exe	System/Network_Service/Local Service	C:\Windows\System32\
system	System	
U1201.exe	Biblio1	C:\Users\Biblio1\Desktop
Winit.exe	System	C:\Windows\System32\
WinWord.exe	Biblio1	C:\Program Files\Microsoft Office\Office12
Winlogon.exe	System	C:\Windows\System32

Quadro Biblio1

De seguida apresenta-se o quadro para o computador Biblio2.

Processos Biblio2	Dono do processo	Caminho do processo
crss.exe	System	C:\Windows\System32\
dwm.exe	Biblio2	C:\Windows\System32\
explorer.exe	Biblio2	C:\Windows\
lsass.exe	System	C:\Windows\System32\
lsm.exe	System	C:\Windows\System32\
Iexplore.exe	Biblio1	C:\Program Files\Internet Explorer
Idle	System	C:\Windows\System32\
SearchIndexer.exe	System	C:\Windows\System32\
Services.exe	System	C:\Windows\System32\
Smss.exe	System	C:\Windows\System32\
Chrome.exe	Biblio2	C:\Users\Biblio2\AppData\Local\Google\Chrome\Application
Spoolsv.exe	System	C:\Windows\System32\
Svchost.exe	System/Network_Service/Local Service	C:\Windows\System32\
system	System	
WinWord.exe	Biblio2	C:\Program Files\Microsoft Office\Office12
Winit.exe	System	C:\Windows\System32\
Winlogon.exe	System	C:\Windows\System32
Wmplayer.exe	Biblio2	C:\Program Files\Windows Media Player
Wmpnetwk.exe	Network Service	C:\Program Files\Windows Media Player

Quadro Biblio2

Processos Biblio2	Dono do processo	Caminho do processo
crss.exe	System	C:\Windows\System32\
dwm.exe	Biblio3	C:\Windows\System32\
Excel.exe	Biblio3	C:\Program Files\Microsoft Office\Office12
explorer.exe	Biblio3	C:\Windows\
lsass.exe	System	C:\Windows\System32\
lsm.exe	System	C:\Windows\System32\
Idle	System	C:\Windows\System32\
SearchIndexer.exe	System	C:\Windows\System32\
Services.exe	System	C:\Windows\System32\
Smss.exe	System	C:\Windows\System32\
Chrome.exe	Biblio3	C:\Users\Biblio3\AppData\Local\Google\Chrome\Application
Spoolsv.exe	System	C:\Windows\System32\
Svchost.exe	System/Network_Service/Local Service	C:\Windows\System32\
system	System	
U1201.exe	Biblio3	C:\Users\Biblio3\Desktop
WinWord.exe	Biblio3	C:\Program Files\Microsoft Office\Office12
Winit.exe	System	C:\Windows\System32\
Winlogon.exe	System	C:\Windows\System32
Wmplayer.exe	Biblio3	C:\Program Files\Windows Media Player
Wmpnetwk.exe	Network Service	C:\Program Files\Windows Media Player

Quadro Biblio 3

Ao analisar os vários quadros, pode-se chegar à conclusão que todos os processos presentes nestes, parecem legítimos. Observando com mais cuidado, verifica-se a presença de um

processo que executa do Ambiente de Trabalho (Desktop), e que tem o nome de U1201.exe. Este processo, é aquele que dá a possibilidade de criar um túnel VPN, de forma a poder ignorar o proxy presente na rede da escola. O programa chama-se Ultrasurf (ultrasurf, 2011) , tal como já foi referenciado, e é devido a este e outros, que se decidiu fazer esta experiência nesta escola. Após ter analisado os resultados, decidiu-se elaborar a lista de conformidade, lista essa que iria ser colocada em conjunto com o programa cliente, presente em cada computador analisado. Posto isto, apresenta-se a lista de processos, presenta na lista de conformidade.

Processos	Caminho do processo
crss.exe	C:\Windows\System32\
dwm.exe	C:\Windows\System32\
Excel.exe	C:\Program Files\Microsoft Office\Office12
explorer.exe	C:\Windows\
lsass.exe	C:\Windows\System32\
lsm.exe	C:\Windows\System32\
Iexplore.exe	C:\Program Files\Internet Explorer
Idle	C:\Windows\System32\
SearchIndexer.exe	C:\Windows\System32\
Services.exe	C:\Windows\System32\
Smss.exe	C:\Windows\System32\
Chrome.exe	C:\Users\%username%\AppData\Local\Google\Chrome\Application
Spoolsv.exe	C:\Windows\System32\
Svchost.exe	C:\Windows\System32\
system	
WinWord.exe	C:\Program Files\Microsoft Office\Office12
Winit.exe	C:\Windows\System32\
Winlogon.exe	C:\Windows\System32
Wmplayer.exe	C:\Program Files\Windows Media Player
Wmpnetwk.exe	C:\Program Files\Windows Media Player

Quadro lista de conformidade utilizada

4.3 Análise de resultados da Parte 2.

Recorrendo à utilização do mesmo sistema de monitorização utilizado anteriormente, pode-se ter um registo dos processos executados ao longo do tempo. Em conjunção com o sistema de logs do programa cliente, pode-se cruzar essa informação, com todos os processos fechados pelos vários clientes, tendo assim uma ideia do bom funcionamento, ou não, do programa.

Sendo que o programa cliente permite guardar no log, o nome do programa em execução, bem como o caminho de onde este se encontra a executar, poder-se-á negar mesmo processos que estejam dentro da lista, desde que estes se encontrem a executar de uma localização diferente daquela que está autorizada a executar. Por exemplo, se alguém tentar mascarar o programa ultrasurf com o nome "svchost", este será negado, pois o utilizador não tem permissões para colocar esse programa a executar dentro da pasta system32. Passando agora aos resultados, começa-se por apresentar os programas executados durante o tempo que o programa cliente esteve ligado nos computadores em questão (Biblio1, Biblio2 e Biblio3).

Processos	Caminho do processo
crss.exe	C:\Windows\System32\
dwm.exe	C:\Windows\System32\
Excel.exe	C:\Program Files\Microsoft Office\Office12
explorer.exe	C:\Windows\
lsass.exe	C:\Windows\System32\
lsm.exe	C:\Windows\System32\
Iexplore.exe	C:\Program Files\Internet Explorer
Idle	C:\Windows\System32\
SearchIndexer.exe	C:\Windows\System32\
Services.exe	C:\Windows\System32\
Smss.exe	C:\Windows\System32\
Chrome.exe	C:\Users\%username%\AppData\Local\Google\Chrome\Application

Spoolsv.exe	C:\Windows\System32\
Svchost.exe	C:\Windows\System32\
system	
WinWord.exe	C:\Program Files\Microsoft Office\Office12
Winit.exe	C:\Windows\System32\
Winlogon.exe	C:\Windows\System32
Wmplayer.exe	C:\Program Files\Windows Media Player
Wmpnetwk.exe	C:\Program Files\Windows Media Player

Quadro representativo dos processos executados nos vários computadores.

Pelo quadro anterior, é possível ver que todos os processos em execução nos vários computadores teste, não incluem nenhum programa que não se encontra dentro da lista de conformidade. Mas a questão que resta responder, é se algum dos utilizadores tentou executar algo fora da lista. Para tal, construiu-se um quadro onde se representam os processos que foram negados pelo programa cliente, bem como o computador onde foi negado, e o caminho da execução.

Processos Biblio2	Computador	Caminho do processo
cmd.exe	Biblio1	C:\Windows\System32\
iexplore	Biblio3	C:\Users\Biblio3\Desktop
gta-vc.exe	Biblio2	F:\GTA
uTorrent.exe	Biblio2	F:\torrent
U1201.exe	Biblio3	G:\Desktop

Quadro Processos negados

Dos programas negados, devem-se reter três, o programa iexplore, cujo nome está normalmente associado ao navegador Internet Explorer, sendo portanto algo autorizado a correr, o programa de torrents uTorrent que executa de uma pendrive, por fim, o programa ultrasurf com o nome u1201.exe. Depois de enunciar as situações mais importantes, há-que explicar o porquê desta escolha. No primeiro caso, apesar do nome iexplore estar associado a

um programa que se encontra presente na lista de conformidade, este foi negado a executar, pois o caminho da aplicação era diferente do caminho previamente configurado e tido como normal para este processo, logo este foi negado a executar. No segundo ponto, o programa uTorrent, apesar de executar de uma pendrive, foi prontamente identificado como não autorizado e fechado pelo programa. Por fim, a razão pela qual se fez esta pequena experiência, o programa ultrasurf, foi prontamente identificado, apesar de ser executado de uma pendrive, sendo a sua execução prontamente negada pelo programa cliente.

4.4 Conclusão da experiência.

Apesar do objetivo principal ter sido verificar a capacidade do programa em identificar processos fora da lista de conformidade, este permitiu também identificar certos alunos que agiam fora das regras da escola, como a proibição de efectuar Downloads ilegais, ou mesmo executar jogos de computador (programa gta-vc.exe presente no quadro dos negados). Portanto, e tendo apresentado os resultados obtidos por este pequeno ensaio, pode-se concluir que o programa criado conseguiu melhorar em muito a qualidade do trabalho do bibliotecário, permitindo-lhe de forma praticamente instantânea identificar processos não autorizados a correr nos computadores da biblioteca, podendo ele tomar a decisão sobre a autorização da execução destes, ou a utilização do chamado "piloto automático" que entrega a tomada de decisão ao programa cliente, presente em cada computador da biblioteca. Esta última, foi talvez a característica mais apreciada por este profissional.

5. Conclusão e trabalho futuro

Este capítulo servirá, de uma forma geral, a efetuar uma análise crítica ao trabalho efetuado, tecendo, para isso, comentários sobre possíveis alterações e melhorias que podem ser implementadas no futuro, de forma a enriquecer a ideia que levou a um projeto de tese.

5.1 Análise crítica

Após uma análise cuidada do que foi conseguido ao longo deste projeto, verifica-se que praticamente todos os objetivos propostos foram cumpridos, sendo que pode-se considerar como um sucesso o resultado final obtido. Conseguiu-se um programa que controla todos os processos executados remotamente em máquinas da mesma rede, avisando o administrador quando algo não autorizado está a ser executado, passando a decisão de autorizar ou não a sua execução para este. É possível enviar comandos remotamente para as máquinas com o programa cliente instalado, sendo possível por exemplo, ordenar o seu reinicio, o seu encerramento, terminar sessão, etc. É possível enviar mensagens para o cliente, de forma a dar avisos sobre o seu comportamento, ou mesmo comunicar algo necessário no momento.

Tudo isto foi conseguido com grande nível de segurança, impedindo assim a injeção de comandos por parte de aplicações desconhecidas, bem como acesos indevido aos ficheiros de log, etc.

É também possível obrigar as aplicações cliente a funcionarem e a tomarem decisões por elas próprias, fechando tudo o que não seja autorizado, caso o administrador não esteja presente na sua área de trabalho. Por fim, é também possível ao administrador utilizar o seu smartphone, caso esteja ausente do seu terminal, para controlar o que se passa com as várias máquinas ligadas. É claro que existem também certos problemas, como por exemplo a não deteção do servidor por parte dos clientes, caso este se desligue e volte a ligar, pois é suposto o servidor estar ligado antes de o cliente iniciar. As possíveis soluções a este problema e melhorias, serão apresentadas no subcapítulo seguinte.

5.2 Trabalho futuro

Como foi referido na análise crítica, existem vários problemas na aplicação. Talvez o mais grave seja mesmo o facto de, caso o servidor se desligue enquanto há máquinas ligadas, e se voltar a ligar, este não receberá informações das aplicações clientes. Este problema pode ser resolvido, aplicando um simples multicast udp, em que se enviará uma mensagem para a gama de IP onde o servidor está inserido, e cada cliente que ouvir a referida mensagem, enviará os seus dados ao servidor. Resolve-se assim o problema de tentar enviar uma mensagem para cada endereço de rede, presente na gama de IP do servidor, enviando assim uma mensagem geral, que poderá ser escutada pelos clientes presentes na rede. Outra solução possível, e talvez aquela que será adoptada, será transformar a aplicação servidor numa aplicação web, e a aplicação cliente num serviço windows. Com a aplicação web será possível comunicar com o cliente apenas quando se pretende efectuar algum pedido, e não será necessário manter uma ligação permanente com o cliente, tal como é necessário com sockets do tipo TCP/IP. Ao transformar a aplicação cliente num serviço Windows, será possível passar esta forma de segurança despercebida ao utilizador, tornando esta medida ainda mais difícil de ser contornada.

Quanto a possíveis melhorias no programa, poderia começar-se pela janela de processos, aquando do recebimento destes, poderia colocar-se uma forma de atualiza-los automaticamente mantendo essa janela aberta, ou selecionando o intervalo de tempo desejado. Poderia também dar-se acesso ao controlo de serviços em execução, utilizando o mesmo processo usado na obtenção dos processos. Outra sugestão seria introduzir também mais comandos, passíveis de serem utilizados remotamente pelo administrador, como por exemplo, desfragmentação de disco, mudança dos perfis de energia, etc. Uma outra característica, seria por exemplo colocar uma rotina de aprendizagem, para que o programa cliente saiba, por exemplo, quais os valores normais para a memória utilizada por determinado programa, que executa duma certa localização, tornando assim ainda mais difícil contornar esta medida de segurança. Em suma, este projeto é válido para o futuro, pois possui muita potencialidade para novas características, e melhorias das características existentes. Sendo esta uma área pouco explorada nas novas tecnologias, considera-se o desenvolvimento desta aplicação uma mais valia para o mercado no caso, quem sabe, de vir a ser comercializado.

REFERÊNCIAS

Encrypting Messages using Merkle-Hellman Knapsack Cryptosystem. **Agarwal, Ashish. 2011.** Computer Science and Network Security, Simon Fraser University : s.n., 2011, International Journal of Computer Science and Network Security, Vol. 11. 5.

Princeton. (2007). Merkle-Helman Cypher. Retrieved September 29, 2012, from https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Merkle-Hellman.html

McCaffrey, J. (2002). AES. Retrieved September 29, 2012, from http://msdn.microsoft.com/en-us/magazine/cc164055.aspx

Rouse, M. (2005). Authentication. Retrieved August 20, 2012, from http://searchsecurity.techtarget.com/definition/authentication

Rouse, M. (2005). Encryption. Retrieved August 21, 2012, from http://searchsecurity.techtarget.com/definition/encryption

Microsoft. (2004). FireWalls. Retrieved from http://technet.microsoft.com/en-us/library/cc700820.aspx

Rouse, M. (2005). Digital Signature. Retrieved from http://searchsecurity.techtarget.com/definition/digital-signature

techterms.com. (2008). Logs. Retrieved from http://www.techterms.com/definition/logfile

Microsoft. (2004). O que são antivírus? Retrieved from http://www.microsoft.com/pt-pt/security/resources/antivirus-whatis.aspx

Mendes, A. (2002). *Arquitetura de Software: desenvolvimento orientado para arquitetura.* (E. Campus, Ed.). Rio de Janeiro.

Usoris systems, L. (2009). Remote utilities. Retrieved from http://www.remoteutilities.com/

DeviceLock, I. (2003). Remote Task Manager. Retrieved from http://www.devicelock.com/rtm/

Software, G. (2005). TalkTechToMe. Retrieved from http://www.gfi.com/blog/

Microsoft. (2002). Process Monitor. Retrieved from http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

ManagePC. (2006). ManagePC. Retrieved from http://managepc.net

Microsoft. (2002). C#. Retrieved from http://msdn.microsoft.com/library/orm-9780596521066-01-03.aspx

Princeton. (2009). Sockets. Retrieved from http://en.wikipedia.org/wiki/Network_socket

Corp., UltraReach Internet. 2011. ultrasurf. [Online] 2011

Figueira,Paulo Marques / Hernâni Pedroso / Ricardo. 2010 C# 3.5. s.l. : FCA,2010. 978-972-722-403-6.

An Overview of Peer-toPeer Computing. Oladele, R.O. 2003. Computer Science, Nigeria : University of llorin, 2003.