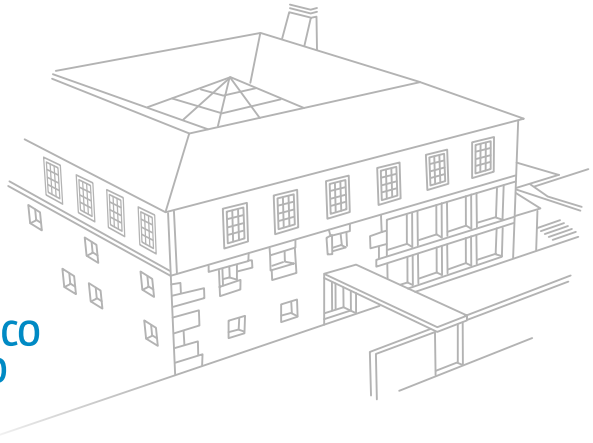

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Recolha de informação de estado para análise forense digital

DESIGNAÇÃO DO MESTRADO
Mestrado em Engenharia Informática

AUTOR
Fábio André Soares de Freitas

ORIENTADOR(ES) Prof. Doutor António Alberto dos Santos Pinto

ANO
2016

Recolha de informação de estado para análise forense digital

Fábio Freitas
Mestrado em Engenharia Informática
ESTGF-IPP
8080176@estgf.ipp.pt

Novembro de 2016

Resumo

No contexto de uma análise forense a um computador, é usual que o técnico queira obter o máximo de informação possível. Em particular, sempre que um técnico se depara com um computador ligado, este deverá tentar recolher o máximo de informação de estado possível. Dispositivos USB apresentam-se como veículos interessantes para construir mecanismos automatizados para esta recolha de informação, pois permitem armazenar as aplicações necessárias para a recolha da informação, o resultado da recolha da informação e facilitar a recolha de forma automática após a sua inserção no PC. Este trabalho tece uma proposta de solução USB para facilitar a recolha de informação de estado com garantia de integridade e multi-plataforma.

Palavras Chave: Resposta a incidentes informáticos, Recolha informação, Multi-plataforma, USB, Informação homogénea, Integridade.

Abstract

In the context of a digital forensic analysis, the technician usually wants to get as much information as possible. In particular, whenever a technician encounters a computer that is turned on, it should try to collect as much status information as possible. USB devices present themselves as interesting vehicles for the automated collection of such information, as it can store the applications required for the collection of the information, can store the results of the information collection and can also facilitate the information collection by enabling its automatic operation, where available. This paper proposes a USB solution to facilitate the collection of state information with integrity guarantees and multi-platform operation.

Key Words: Computer incident response, Information collection, Multi-platform, USB, Homogeneous information, Integrity.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Acrónimos

AES: Advanced Encryption Standard
APOP: Authenticated Post Office Protocol
ARP: Address Resolution Protocol
CD-ROM: Compact Disc Read-Only Memory
DNS: Domain Name System
exFAT: Extended File Allocation Table
FIPS: Federal Information Processing Standard
GCC: GNU Compiler Collection
GHz: Gigahertz
HMAC: Hash-based Message Authentication Code
ID: Identifier
IP: Internet Protocol
IR: Incident Response
LiME: Linux Memory Extractor
MAC: Medium Access Control
MD5: Message-Digest algorithm 5
NIST: National Institute of Standards and Technology
NTFS: New Technology File System
PID: Process Identifier
RAM: Random Access Memory
RFC: Request for Comments
RPM: Red Hat Package Manager
SHA-1: Secure Hash Algorithm 1
SHA-2: Secure Hash Algorithm 2
USB: Universal Serial Bus

Capítulo 1

Introdução

Em situações de resposta a incidentes de segurança informática, ou *Incident Response (IR)*, em inglês [5], o analista forense tenta recolher toda a informação possível. Situações de IR podem incluir atividades que passam por: confirmar a real existência de um incidente; providenciar uma rápida deteção e contenção; identificar factos e informações reais; minimizar interrupções comerciais e de operação da rede; minimizar danos para a organização; recuperar e gerir a perceção pública do incidente; recolher provas que possibilitem ações legais ou civis contra os autores do crime; informar a gestão de topo; e, por fim, melhorar a postura da organização perante futuros incidentes de segurança [6].

A análise de informação de estado surge como um aspeto importante da análise forense digital, especialmente quando se tratam de situações de IR que envolvam vários equipamentos ligados em rede. Este tipo de procedimento de análise forense digital é usualmente designado de *live forensics*. O objetivo deste tipo de análise forense é o de recolher dados voláteis antes de se desligar o sistema a ser analisado, recolhendo informações como, por exemplo, uma cópia da memória *RAM* ou a lista de processos em execução. Esta informação de estado é considerada como volátil, pois será perdida com o desligar do equipamento.

A informação de estado armazenada na memória RAM pode conter evidências de interesse para o analista, tais como [7]: a lista de processos em execução; o histórico de comandos executados na consola; palavras passe utilizadas recentemente (algumas em texto legível); mensagens instantâneas; endereços IP; quem tem sessão iniciada no sistema; que portas de rede e aplicações estão à escuta de ligações IP; informação do sistema; lista e histórico de dispositivos ligados, entre outras.

A informação recolhida numa análise forense digital *live* não deve ser vista como um substituto daquela que é obtida por uma análise forense tradicional. Deve ser vista como complementar que permitirá recolher informação não disponível de outra forma. A análise forense tradicional, que

envolve a cópia *bit* a *bit* dos suportes de armazenamento de dados e a sua análise posterior, é normalmente dirigida a um conjunto reduzido de equipamentos já que é um processo moroso. Por outro lado, a recolha de informação de estado, podendo ser automatizada, torna-se numa fonte de informação complementar que pode ser muito útil. Permite, por exemplo, perceber que ligações estão estabelecidas entre os vários equipamentos e daí verificar a existência de vírus, *spyware*, *malware* ou outro tipo de programas maliciosos que façam uso da rede para comunicar com um servidor de controlo, por exemplo.

A recolha de dados voláteis não está isenta de risco. Por um lado, numa recolha de informação de estado é necessário utilizar-se o sistema em análise que poderá estar comprometido e, eventualmente, comprometendo também a informação recolhida. Por outro, a recolha de informação de estado é um processo intrusivo, no sentido em que a recolha de informação de estado altera o estado do sistema em análise. Alteração esta que é imperioso evitar numa análise forense digital tradicional.

Uma abordagem poderá ser a de se usar um processo automatizado, previamente conhecido, que permitirá prever o seu impacto na informação de estado. De outra forma, poder-se-á causar alterações excessivas, desnecessárias e não previsíveis para o sistema em análise. Possivelmente, interrompendo processos de negócio, destruindo provas, ou alertando um atacante da sua presença. Em alguns casos, realizar uma recolha de dados voláteis pode causar a falha do sistema. O investigador deverá estar bem preparado e ser diligente na recolha de dados voláteis.

Dispositivos de armazenamento *Universal Serial Bus (USB)* apresentam-se como veículos interessantes para construir mecanismos automatizados para esta recolha de informação, pois permitem armazenar tanto as aplicações necessárias para a recolha da informação como o resultado da recolha, são facilmente transportáveis, podem ter grandes capacidades de armazenamento e facilitam a recolha de forma automática após a sua inserção no PC.

Atualmente, não estão disponíveis soluções que permitam efectuar a recolha de informação de estado num contexto de análises forense que, simultaneamente, seja multi-plataforma, exaustivo na sua recolha de informação de estado, homogéneo quanto à qualidade e quantidade de informação recolhida, independentemente do sistema operativo em análise, capaz de garantir a integridade de comandos e da informação recolhida e portável.

1.1 Objetivos do trabalho

O objetivo deste trabalho consiste na elaboração de uma proposta de solução que facilite a recolha de informação de estado de forma simples, automatizada, portável e que seja capaz de ser utilizada em contextos de investigação forense digital, de que são exemplo as investigações em situações de IR.

1.2 Resultados relevantes

A solução proposta deverá, em particular, ser:

1. **Multi-plataforma:** Recolher informação de estado independentemente do sistema operativo em uso.
2. **Portável:** Deverá ser transportada num suporte de armazenamento USB.
3. **Auto-executável:** Deverá ser executado, sempre que possível, de forma automática.
4. **Exaustiva:** Deverá recolher o máximo de informação possível de cada sistema à qual for ligado.
5. **Integra:** Deverá registar o resultado de funções criptográficas de resumo (ex: SHA512) sobre toda a informação recolhida e sobre os comandos utilizados.
6. **Documentada:** Deverá guardar um registo da toda a sua atividade para ficheiro de registo.
7. **Homogénea:** Deverá recolher o mesmo tipo de informação para os vários sistemas operativos suportados (Windows, Linux e macOS).

1.3 Estrutura do relatório

O relatório está organizada em capítulos. O Capítulo 2 apresenta uma descrição do que é a criptografia, descreve também o funcionamento das funções criptográficas de resumo bem como alguns exemplos de funções criptográficas. O Capítulo 3 identifica as soluções existentes que possibilitam a recolha de informação de estado em situações de IR. O funcionamento de cada uma destas soluções é descrito de forma geral e é identificada a lista de artefactos que estas recolhem. Terminando o capítulo, é efetuada uma análise comparativa das várias soluções. O Capítulo 4 apresenta resultado do levantamento de requisitos para a solução proposta, que inclui as características necessárias para o suporte de armazenamento USB, bem como descreve a proposta de solução encontrada e apresenta um protótipo da aplicação. O Capítulo 5 descreve a metodologia de trabalho desenvolvida, os testes que foram realizados, bem como apresenta e analisa os resultados obtidos. O Capítulo 6 conclui o trabalho realizado.

Capítulo 2

Criptografia

A criptografia possibilita que se escrevam mensagens ocultando o seu conteúdo. A criptografia tem como objetivo possibilitar que um grupo restrito de entidades, normalmente duas, consigam trocar informação de forma a que esta apareça como incompreensível a terceiros [8].

A utilização da criptografia revela a informação, isto é, um texto que esteja criptografado é obviamente incompreensível para quem não conhece métodos para o decifrar. Este modo de operação pode proporcionar a troca de informação sensível que foi escondida por ser ilegal, levando a que surjam leis ou regulamentos que impedem a criação *ad hoc* de conteúdos criptografados. A introdução de legislação limitadora do uso de criptografia, tipicamente nos EUA, levou ao aparecimento de uma outra técnica conhecida como esteganografia.

A finalidade da esteganografia é possibilitar que um conteúdo sensível seja escondido no interior de outro conteúdo de teor aparentemente inocente. Um exemplo muito conhecido é o de escrever com tinta invisível. Como exemplo mais atual temos a capacidade para se esconder conteúdos dentro de imagens, utilizando os *bits* com menos significado de cada *pixel* para transportar a informação escondida. Em ambos os casos, o conteúdo escondido é invisível ao olho humano, mas quem conhecer como a informação foi escondida conseguirá descobri-la e poderá obtê-la com muita facilidade.

A criptanálise possibilita a descoberta da informação que está criptografada. Uma motivação para o desenvolvimento de técnicas criptográficas é a criação de métodos de criptanálise para impedir as intenções das primeiras. Estes métodos tem como objetivo encontrar, por meio de processos variados, os elementos que foram escondidos através da criptografia ou técnicas ou aspetos particulares utilizados para a produzir (algoritmos, chaves, etc.).

Por fim, denomina-se por criptologia a área do conhecimento que se propõe estudar a criptografia e a criptanálise. Um criptólogo é uma pessoa que se propõe estudar os problemas que surgem tanto na criptografia como na criptanálise. Em casos práticos esse estudo é inseparável, visto que a

criação ou a utilização das técnicas criptográficas corretamente pressupõe-se que exista algum conhecimento dos riscos de criptanálise das mesmas.

2.1 Funções Criptográficas de Resumo

Funções criptográficas de resumo, ou função de *hash*, convertem uma sequência de bits de qualquer tamanho (como ficheiros e mensagens) numa pequena sequência de tamanho fixo, a que se chama resumo. Não se pode garantir que nunca vão existir dois ficheiros (ou mensagens) que gerem o mesmo resumo como resultado (colisão), devido ao facto de o número de bits utilizados no resumo ser menor que o número de bits utilizados como mensagens. No entanto, as funções criptográficas de resumo são preparadas para que as colisões não aconteçam [9]. Por exemplo, na Fórmula 2.1 temos uma representação do funcionamento de uma função criptográfica de resumo. A função H, da fórmula 2.1, devolve um *hash* de valor h, para uma determinada mensagem [10]:

$$h = H(M) \tag{2.1}$$

M é uma mensagem de tamanho aleatório. Como h é igual a H(M), h é um valor de tamanho fixo. O valor do *hash* (h) quando acrescentado à mensagem original, permite validar se a mensagem está correta, isto é, igual ao original. Para validar se a mensagem está correta, o recetor calcula o valor do *hash* e compara o resultado obtido com o valor recebido [10, 11].

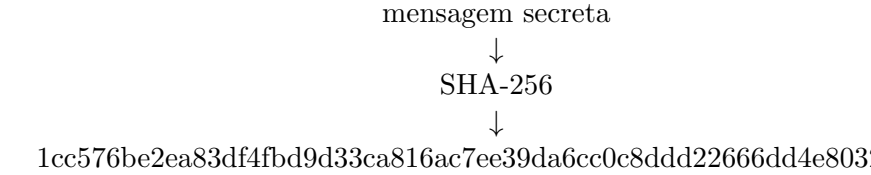

Figura 2.1: Exemplo de SHA-256 [1]

Na Figura 2.1 observa-se um exemplo da utilização da função SHA-256 [1]. O tamanho do *output* da função não vai depender do tamanho da mensagem de entrada, visto que o tamanho do *hash* resultante será sempre o mesmo (256 bits). Com a alteração da mensagem de entrada, mesmo que ligeira, o *hash* resultante deverá ser completamente diferente. As funções de *hash* tem de ser obrigatoriamente difíceis de inverter em tempo útil. Funções de *hash* são úteis em situações onde se tenta manter a integridade de dados, ou como elementos de novas ferramentas criptográficas [9].

As funções criptográficas de resumo podem também ser utilizadas em mecanismos de autenticação. Um exemplo comum são os mecanismos Hash-based Message Authentication Code (HMAC) [12], que são mecanismos de

autenticação de mensagens que recorrem a funções criptográficas de resumo. A força criptográfica do HMAC depende das propriedades da função hash subjacente. HMAC-MD5 [13] define casos de teste para HMAC-MD5.

2.1.1 MD5

Message-Digest algorithm 5 (MD5) [14] é uma função criptográfica de resumo que gera um resultado de 128 *bits* de comprimento, para qualquer mensagem de entrada. É uma função de resumo muito utilizada mas que, atualmente, já não o deveria ser. Os ataques publicados contra MD5 mostram que não é prudente utilizar MD5 pois já não tem a resistência necessária contra colisões [15].

O MD5 foi publicado em 1992 como um Request for Comments (RFC) informativo. Desde essa altura que tem sido estudado, surgindo novos ataques. Tal levou também a que surgissem recomendações adicionais para o seu uso [16, 17]. Alternativas para o HMAC-MD5 incluem HMAC-SHA256 [18] e AES-CMAC [19] quando o Advanced Encryption Standard (AES) está disponível por *hardware*.

Pseudo-colisões foram descritas pela primeira vez para o MD5 em 1993 [20]. Em 1996 demonstrou-se uma colisão para a função de MD5 com recurso a um valor inicial conhecido[21]. O primeiro trabalho que demonstrou duas colisões com MD5 foi publicado em 2004 [22]. As técnicas de ataque ao MD5 conhecidas na altura foram publicadas na EUROCRYPT 2005 [23]. Desde então, múltiplos trabalhos de investigação, onde se detalham outros ataques de colisão ao MD5, foram publicados. O trabalho apresentado em [24] pode encontrar colisões MD5 em cerca de um minuto num PC padrão (Intel Pentium, 1,6 Gigahertz (GHz)). Em [25] afirma-se que se consegue encontrar colisões MD5 em 10 segundos, ou menos, com um Pentium 4 de 2,6 GHz. Adicionalmente, foram descobertos e aplicados com sucesso, ataques a certificados X.509 [26] que usem o algoritmo MD5 [25, 27, 28, 29].

2.1.2 SHA-1

Secure Hash Algorithm 1 (SHA-1) [30] é uma função criptográfica de resumo que gera um resultado de 160 *bits* de comprimento, para qualquer mensagem de entrada. A função SHA-1 foi emitida pelo National Institute of Standards and Technology (NIST) em 1995 como uma Federal Information Processing Standard (FIPS). Desde sua publicação, o SHA-1 tem sido adotado por muitas normas do governo e na indústria, em normas específicas sobre assinaturas digitais para as quais é necessária uma função de resumo resistente a colisões. Além de seu uso em assinaturas digitais, o SHA-1 também foi implantado como um componente importante em vários esquemas e protocolos de criptografia, como autenticação do utilizador, acordo de chave, e geração de números pseudo-aleatórios. O SHA-1 tem sido amplamente adotado em

muitos sistemas e produtos de segurança [31].

Ao longo do tempo, e tal como surgiu com o MD5, a comunidade científica debruçou-se também sobre a análise de falhas e problemas do SHA-1 [32]. Um componente que surge em alguns ataques é o de colisão local, introduzido por Chabaud e Joux em 1998 em ataques anteriores[33]. A ideia subjacente a uma colisão local é o de primeiro introduzir uma diferença numa das palavras do estado intermediário da função. Para um estado interno feito de j palavras (j = 5 no caso de SHA-1), o atacante usa então diferenças subsequentes em apenas algumas das palavras da mensagem, a fim de cancelar qualquer contribuição da diferença no cálculo de um novo estado interno.

O obstáculo principal quando se usam as colisões locais é que o atacante não controla todas as palavras da mensagem, já que algumas são geradas pela função interna de extensão de mensagem. Uma importante observação de Chabaud e Joux foi mostrar como encadear múltiplas colisões locais ao longo do *disturbance vector*[1], criando um caminho semelhante em que o estado final de uma função não contem diferenças e que o padrão das colisões locais é compatível com a extensão da mensagem.

Uma melhoria importante para um ataque original, inteiramente baseado em colisões locais, foi introduzida por Wang, Yin e Yu em 2005, num primeiro ataque teórico completo ao SHA-1 [31]. Este ataque baseava-se na construção de caminhos diferentes não lineares. Construção esta foi inicialmente feita à mão por Wang, Yin e Yu. A elaboração de algoritmos eficientes para gerar tais caminhos diferentes surge mais tarde com a contribuição de De Cannière e Rechberger em 2006, que propôs uma abordagem *guess-and-determine* [34]. Uma abordagem diferente, baseada num método *meet-in-the-middle*, foi proposta por Stevens et al. [35, 36].

2.1.3 SHA-2

Desde 2005, têm surgido ataques de colisão para funções de resumo em uso. Em particular, as falhas detetadas ao MD5 e ao SHA-1, convenceram muitos criptógrafos de que estas funções não podem mais ser consideradas seguras. Como consequência, o NIST propôs a transição de SHA-1 para a família Secure Hash Algorithm 2 (SHA-2). Muitas empresas e organizações seguiram este conselho e já migraram para SHA-2. Atualmente já se trabalha na próxima versão, o Keccak [37] que ainda não foi padronizado como SHA-3 porque que o SHA-2 é mais rápido em várias plataformas. Em particular, o SHA-512 é muito mais rápido do que ambos SHA-256 e Keccak na maioria das plataformas 64 bits [38]. Tal levou a que aparece-se a sugestão de utilização de uma versão truncada de SHA-512, mesmo para valores de hash

[1] A própria mensagem, numa das iteração de extensão com a função interna de extensão de mensagem

de 256 bits [39]. O próprio NIST também define esta variante, denominada SHA-512/256, no FIPS 180-4 [40, 41].

Poucos trabalhos científicos sobre ataques ao SHA-512 têm sido publicados nos últimos anos. A segurança de SHA-512 contra ataques *preimage* foi estudado pela primeira vez por Aoki et al. em [42]. Estes apresentaram um ataque *preimage* em 46 das 80 etapas do SHA-2. Sendo mais tarde estendido para 50 etapas por Khovratovich et al. em [43]. Recentemente, Li et al. mostraram que determinados ataques *preimage* também podem ser utilizados para construir para um ataque de colisão *free-collision* até 57 etapas do SHA-512 em [44]. No entanto, todos os ataques são apenas ligeiramente mais rápido do que as respetivas complexidades dos ataque genéricos [41].

2.2 Conclusão

A criptografia e, em particular, as funções criptográficas de resumo são uma componente essencial nas ferramentas automatizadas de investigação digital. Resultados de funções criptográficas de resumo sobre ficheiros podem servir para identificar conteúdos (inofensivo, ilegais, vírus, ...), para autenticar utilizadores ou como garantia de integridade de dados. Existem várias funções criptográficas de resumo, como por exemplo: MD5, SHA-1, SHA-2. Algumas com problemas já bem identificados que levam à recomendação da sua não utilização.

Capítulo 3

Trabalho Relacionado

Estão disponíveis *online* várias aplicações que visam a recolha de artefactos, num contexto de uma investigação forense digital de resposta a um incidente de segurança, e que podem ser obtidas livremente. Este capítulo enumera e descreve tais aplicações, indicando como funcionam e identificando a lista de artefactos recolhidos por cada uma. As aplicações a testar serão um misto de aplicações *open source*, gratuitas e pagas. Concluindo o capítulo, é apresentada uma comparação entre as várias aplicações.

3.1 ir-triage-toolkit

A aplicação *ir-triage-toolkit* consiste num conjunto de pequenas aplicações e *scripts*. O autor, por incluir aplicações de outras entidades na sua aplicação, afirma que não são quebrados quaisquer contratos de licença ou restrições de direitos de autor. Estes *scripts* visam automatizar a recolha de artefactos para facilitar a sua posterior análise, bem como a triagem de eventos em cenários de respostas a incidentes. Pode ser utilizada em sistemas operativos Windows e Linux. Possibilita ainda a obtenção de cópias da memória RAM e inclui *scripts* para criar *kits* de ferramentas para Linux e Windows [45]. A necessidade de criar tais *scripts* surge porque algumas das aplicações que compõem esta aplicação não permitem a sua distribuição, embora estejam livremente disponíveis *online*.

Para se utilizar esta aplicação, deve-se primeiro fazer o seu *download* e depois extraí-lo para o suporte de armazenamento USB onde será utilizado. Após isto, abre-se um terminal e muda-se para o diretório para onde este foi extraído. No diretório utiliza-se o comando constante da Listagem 3.1, passando-lhe o caminho para o directório de instalação como argumento. O procedimento para Windows é executado de forma análoga.

Listagem 3.1: Criar kit de aplicações no ir-triage-toolkit (Linux)

```
1 ./create-toolkit /store/it/here
```

Este passo inicial deve ser executado, de preferência, antes que uma situação IR surja. O *kit* de aplicações resultante deve ser armazenado num suporte de armazenamento USB, ou similar, para que possa ser utilizado em situações de IR. Após a criação do *kit*, este poderá ser executado, num terminal, como demonstrado na Listagem 3.2.

Listagem 3.2: Executar ir-triage-toolkit no Linux

```
1 ./run
```

O comando apresentado na Listagem 3.2 deve ser executado com privilégios de administração. O argumento utilizado é o nome da pasta onde será guardada toda a informação recolhida. Na ausência do argumento, o *script* solicita-o ao utilizador.

O *scritp* inicia a recolha da informação, executando tarefas como: obter uma cópia da memória RAM; recolha de informação de rede; recolha de informação sobre ficheiros abertos e processos em execução; recolha de informação do utilizador/sistema e recolha de informação de dispositivos. Durante o processo de recolha, são criados registos detalhados, como o tempo demorado pelos comandos executados, e são calculadas funções de resumo (SHA256) sobre os ficheiros guardados no diretório de saída [45].

Listagem 3.3: Script run.sh do ir-triage-toolkit no Linux

```
1  log() {
2  echo "$(date +"%b %d %H:%M:%S") $(hostname) irscript: $1" | tee
       -a "$logfile"
3  }
4
5  # Start the log.
6  echo -n > "$logfile"
7  log "# Incident response volatile data collection script."
8  log "# Starting data collection..."
9
10 # 1. Acquire a full memory dump.
11 log "# Starting LiME to dump system memory..."
12 memfile="$saveto/memdump-$(hostname)-linux-$(uname -m).lime"
13 log "insmod $bin/lime.ko \"path='$memfile' format=lime\""
14 insmod "$bin/lime.ko" "path=\"$memfile\" format=lime"
15 log "rmmod lime"
16 rmmod lime
17 log "# LiME finished."
18
19 # 2. Collect network information.
20 log "# Collecting network information..."
21 log "netstat -ap > $saveto/network.txt 2>&1"
22 netstat -ap > "$saveto/network.txt" 2>&1
23
24 # 3. Collect information about opened files and running
       processes.
25 log "# Collecting information about opened files and running
       processes."
```

```
26 log "lsof > $saveto/opened_files.txt 2>&1"
27 lsof > "$saveto/opened_files.txt" 2>&1
```

A Listagem 3.3 apresenta um excerto do *script run.sh* usado pelo *ir-triage-toolkit* para recolher informação num sistema Linux. A função **log()** (linhas 1-3) é a função que é utilizada para guardar, num ficheiro log.txt, todas as mensagens que vão sendo mostradas ao utilizador no decorrer da aplicação. Este ficheiro é limpo a cada execução (linha 6). A cópia da memória RAM é conseguida recorrendo à instalação do módulo Linux Memory Extractor (LiME), como se pode verificar na linha 12. Para recolher informação de rede usa o comando netstat, usa o comando lsof para recolher informação sobre os ficheiros abertos.

Listagem 3.4: Exemplo de um output do ir-triage-toolkit no Linux

```
1 3eff669c359c9a91828addd778090bacf7c30cc98080edc55be05e8b3b1b6df4
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/mounted_devices.
    txt
2 911432fd3d8c935063d8170b5c2301927b7ac5b6e12b8589f23a5895c6c6b58d
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/network.txt
3 d346e0ed572b88280ad3bfe9621fb83fb89a1a449a05dffe6ba215ea84cc3ed2
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/opened_files.txt
4 b369d4999e0f620c8650369cc910959d7e5aaa9fd6473873af7df7cc31c0aea1
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/system_dmesg.txt
5 5a1d52bc8b48578760955a8708f07931ab8518361a252074860fff1468f2a97a
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/system_uname.txt
6 94881f7a7ede7854c72af8c5e24ad2b83d4fa4cee0548d730cb13fbac5c69a04
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/users_id.txt
7 529826fc99860076ea60346bae719b1e014d24d54ccc5a32206f00e625fd859e
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/users_who.txt
8 a791eaf769195fa2cc1d67cfa98aa0630b3d83e29192e4534d7d61ebabffaa5a
      teste/Fabio-PC.Linux-2015.10.27-16.11.19/users_w.txt
```

Na Listagem 3.4 é apresentado um resumo do *output* do ir-triage-toolkit no Linux. Em particular, é possível ver os resumos SHA256 de alguns dos ficheiros criados pela aplicação. O propósito desta informação é poder saber se os ficheiros criados pela aplicação são alterados posteriormente caso seja alterado e gerado um novo *hash*. Pelo nome do ficheiro é percetível que tipo de informação foi guardada nesse ficheiro.

A aplicação foi testada também num sistema Windows e concluiu-se, em função da informação recolhida, que há falta de homogeneidade quanto aos dados recolhidos pela mesma aplicação em ambos os sistemas. É recolhida mais informação nos sistemas Windows sobre a rede e sobre dispositivos ligados, nomeadamente dispositivos USB. Já relativamente à informação sobre o sistema em si e sobre os seus utilizadores, é recolhida mais informação em sistemas Linux.

3.2 IRKIT

A aplicação IRKIT foi criada por Bill Dean para uso próprio. Juntou um conjunto de aplicações livres para a recolha de dados voláteis num *script*. Todas as aplicações devem caber num suporte de armazenamento USB pronto para ser utilizada em situações de IR. Pode ser utilizada apenas em sistemas operativos Windows [46].

Esta aplicação é constituída por um *script* e por uma pasta com pequenas aplicações que ajudam a recolher informação. Só são incluídas aplicações que podem ser distribuídas livremente, as restantes são identificadas e descritas no ficheiro *!!MUST READ ME!!.txt*. É importante que esta ferramenta seja configurada num sistema onde não tenha ocorrido um IR. Uma vez configurada, poderá ser utilizada abrindo um terminal no diretório onde a aplicação está armazenada. Uma vez dentro do diretório utiliza-se o comando demonstrado na Listagem 3.5.

Listagem 3.5: Executar IRKIT no Windows

```
1 IncidentResponseAuto.cmd
```

Como se pode observar na listagem 3.5, para executar a aplicação basta usar-se o nome do *script* sem parâmetros. Este inicia a execução de imediato e vai mostrando o que está a acontecer no momento da recolha. Esta aplicação pode demorar algum tempo na recolha de toda a informação já que faz a cópia integral da memória RAM. Quando esta aplicação terminar vai comprimir toda a informação, criando assim 3 ficheiros distintos: 1) ficheiro de registo de erros; 2) a cópia da memória RAM; e 3) listagem de toda a informação recolhida pelo *script*. Caso se analise este último ficheiro, pode-se observar que cada comando executado pelo *script* gera um ficheiro de registo próprio, como se pode verificar na Listagem 3.6.

Listagem 3.6: Script IRKIT para Windows

```
1  :BEGIN_ipconfig_all
2  SET /a CURR_CMD_NUM+=1
3  REM Grab network information
4  TITLE [%CURR_CMD_NUM%/%TOTAL_NUM_CMDS%]%mode% Mode: NETWORK --
       ipconfig all
5  ECHO [%CURR_CMD_NUM%/%TOTAL_NUM_CMDS%] %mode% Mode: NETWORK --
       ipconfig all
6  ipconfig.exe /all > %cmd_resultsdir%\ipconfig_all_results.txt 2>
       %error_dir%\ipconfig_all_errors.txt
7  ECHO %date% %time% INFO : NETWORK -- ipconfig all >> %resultsdir
       %\incidentresponseauto.log
8  :END_ipconfig_all
9
10 :BEGIN_netstat_a
11 SET /a CURR_CMD_NUM+=1
12 REM Grab network connection and routing information (normal)
```

```
13 TITLE [%CURR_CMD_NUM%/%TOTAL_NUM_CMDS%]%mode% Mode: NETWORK --
       netstat -a
14 ECHO [%CURR_CMD_NUM%/%TOTAL_NUM_CMDS%] %mode% Mode: NETWORK --
       netstat -a
15 netstat.exe -a > %cmd_resultsdir%\netstat_a_results.txt 2> %
       error_dir%\netstat_a_errors.txt
16 ECHO %date% %time% INFO : NETWORK -- netstat -a >> %resultsdir%\
       incidentresponseauto.log
17 :END_netstat_a
18
19 :BEGIN_netstat_abvon
20 SET /a CURR_CMD_NUM+=1
21 REM Grab network connection and routing information (extended)
22 TITLE [%CURR_CMD_NUM%/%TOTAL_NUM_CMDS%]%mode% Mode: NETWORK --
       netstat -abvon - This may take a while...
23 ECHO [%CURR_CMD_NUM%/%TOTAL_NUM_CMDS%] %mode% Mode: NETWORK --
       netstat -abvon - This may take a while...
24 netstat.exe -abvon > %cmd_resultsdir%\netstat_abvon_results.txt
       2> %error_dir%\netstat_abvon_errors.txt
25 ECHO %date% %time% INFO : NETWORK -- netstat -abvon >> %
       resultsdir%\incidentresponseauto.log
26 :END_netstat_abvon
```

Na listagem 3.6 temos um extrato do *script IncidentResponseAuto.cmd* usado para recolher informação em sistemas Windows. Na linha 2 existe uma variável que é incrementada há medida que vai terminando os comandos que o compõe. Esta variável e utilizada para mostrar quanto falta para terminar o *script*. A linha 4 altera o título da consola em uso (comando *TITLE*). A linha 5 mostra na consola o número atual de comandos já executados (*%CURR_CMD_NUM%*), o número total de comandos (*%TOTAL_NUM_CMDS%*), e também mostra qual é o tipo de informação que está a ser recolhida (*Mode: NETWORK*) e o comando em uso no momento (*ipconfig all*). A linha 6 do *script* executa o comando. A linha 7 é utilizada para datar a execução (*%date%*, *%time%*). A informação é ainda guardada num ficheiro de logs ($\gg$ *%resultsdir%\incidentresponseauto.log*).

Listagem 3.7: Exemplo de um output do IRKIT no Windows

```
1 Drive    Type         File System  Path                    Free Space
2 -----    -----        -----------  -----                   ----------
3 C:\      Fixed        NTFS                                 79.04 GB
4 D:\      CD-ROM                                            0.00
5 E:\      Network      VBoxSharedFolderFS \\vboxsrv\
      Pasta_Partilhada_windows                               107.51 GB
6 F:\      CD-ROM                                            0.00
```

Na listagem 3.7 pode ver-se um exemplo de um *output* do IRKIT no Windows. O *output* apresentado é o do comando *di.exe* que mostra informação sobre as unidade de dados (*drives*) disponíveis no sistema. Observa-se que existe uma unidade que é o disco duro onde está instalado o sistema operativo (linha 3), com o tipo de ficheiros New Technology File System (NTFS).

Também temos 2 unidades que permitem a leitura de Compact Disc Read-Only Memory (CD-ROM) (linhas 4 e 6). E por fim temos mais uma unidade do tipo *Network* que permite trocar ficheiros entre o sistemas operativos (linha 5). A unidade *E:* aparece neste *output* porque a aplicação foi testada numa máquina virtual (VirtualBox) e serve para que a máquina virtual tenha acesso a essa pasta do sistema operativo real, permitindo a troca de ficheiros ente a máquina virtual e o sistema operativo real.

A aplicação não recolhe informação noutros sistemas operativos que não Windows.

3.3 Tr3Secure

O Tr3Secure's Data Collection é outro *script* para recolha de evidências forenses digitais em sistemas ligados. É desenvolvida por Corey Harrell, disponibilizada no seu blogue *Journey Into Incident Response*. Harrell necessitava de um conjunto de aplicações para responder a sistemas durante as simulações de ataque e uma das aplicações tinha que recolher rapidamente dados voláteis num sistema. Os comandos necessários para o bom funcionamento do Tr3Secure não são disponibilizados diretamente pelo mesmo por salvaguarda de direitos de autor. Este *script* foi desenvolvido para atender às necessidades da comunidade DFIR e pode ser utilizado em sistemas operativos Windows[47].

Esta aplicação é constituída por 2 *scripts* e uma pasta com pequenas aplicações para ajudar na recolha de informação. Um dos *scripts* é utilizado para recolher todo o tipo de informação, já o outro é utilizado para recolher informação de um determinado utilizador do sistema. Foi analisado o *script* que recolhe toda a informação do sistema por ser mais próximo das restantes aplicações em análise. Para se utilizar esta aplicação deve-se fazer *download* desta e extrai-la para o suporte de armazenamento USB. O passo seguinte passa por fazer o *download* das dependências (descritas no ficheiro *read_me.txt*). Após isto, terá de se abrir um terminal como administrador e mudar para a pasta para onde foi extraído. Nesse diretório, executa-se o comando demonstrado na Listagem 3.8.

Listagem 3.8: Executar Tr3Secure no Windows

```
1 tr3-collect.bat [case number] [drive letter for storing
    collected data] [menu selection #]
```

O parâmetro *[case number]* permite indicar um identificador único para a recolha que será feita, esse identificador será concatenado com *Data-* para formar o nome da pasta onde será guardada a informação recolhida. O segundo parâmetro é a letra da unidade de armazenaemnto onde a recolha de informação será guardada. O terceiro parâmetro, opcional, pode ser usado para escolher que tipo de informação deve ser recolhida. As opções

disponíveis para o terceiro parâmetro são: 1 - recolhe uma cópia da memória RAM; 2 - recolhe dados voláteis; 3 - recolhe dados não voláteis; 4 - recolhe dados voláteis e não voláteis; 5 - recolhe só artefactos NTFS; 6 - recolhe uma cópia da memória RAM, dados voláteis e não voláteis. Caso não se passe este parâmetro, o seu valor por omissão é 4.

Tal como a aplicação IRKIT, esta também só foi desenvolvida para Windows, assim, todos os exemplos mostrados para esta aplicação são em Windows.

Listagem 3.9: Script Tr3Secure para Windows

```
1  :: Creating Log
2  :: ------------------------------------------------------------
3  :: This section sets up the logging function of the script
4  ::
5  :createlog
6  cls
7  :: Creates the directory on the collection drive to store data
       if it isn't already present
8  if not exist %c_drive%:\Data-%case% (
9     echo %DATE% %TIME% - Running tools\mkdir.exe on %COMPUTERNAME
          % to create the case output folder Data-%case%
10    echo:
11    tools\mkdir.exe %c_drive%:\Data-%case%
12    )
13 :: Will create the log file if it's not present and start
       logging
14 if not exist %c_drive%:\Data-%case%\Collection.log (
15    echo ************************************************** >
          %c_drive%:\Data-%case%\Collection.log
16    echo Collection Log for Case %case% >> %c_drive%:\Data-%case
          %\Collection.log
17    echo ************************************************** >>
          %c_drive%:\Data-%case%\Collection.log
18    echo Log Created at %DATE% %TIME% >> %c_drive%:\Data-%case%\
          Collection.log
19    echo: >> %c_drive%:\Data-%case%\Collection.log
20    echo: >> %c_drive%:\Data-%case%\Collection.log
21    )
22 echo ------------------------------------------------------------ >>
       %c_drive%:\Data-%case%\Collection.log
23 echo %DATE% %TIME% - Logging initiated for %COMPUTERNAME% >> %
       c_drive%:\Data-%case%\Collection.log
24 echo %DATE% %TIME% - Logging initiated for %COMPUTERNAME%
25 echo %DATE% %TIME% - Logging started on %COMPUTERNAME% by user
       account %USERDOMAIN%\%USERNAME%  >> %c_drive%:\Data-%case%\
       Collection.log
26 echo %DATE% %TIME% - The drive letter for the volume storing the
       collection data is %c_drive%: >> %c_drive%:\Data-%case%\
       Collection.log
27 echo:
28 cls
29    :: Logs the selection made
```

```
30    if %selection% == 1 echo %DATE% %TIME% - Selection was made
          to acquire memory forensic image for %COMPUTERNAME% >> %
          c_drive%:\Data-%case%\Collection.log
31    if %selection% == 2 echo %DATE% %TIME% - Selection was made
          to acquire volatile data for %COMPUTERNAME% >> %c_drive%:\
          Data-%case%\Collection.log
32    if %selection% == 3 echo %DATE% %TIME% - Selection was made
          to acquire non-volatile data for %COMPUTERNAME% >> %
          c_drive%:\Data-%case%\Collection.log
33    if %selection% == 4 echo %DATE% %TIME% - Selection was made
          to acquire volatile and non-volatile data for %
          COMPUTERNAME% >> %c_drive%:\Data-%case%\Collection.log
34    if %selection% == 5 echo %DATE% %TIME% - Selection was made
          to acquire NTFS artifacts for %COMPUTERNAME% >> %c_drive
          %:\Data-%case%\Collection.log
35    if %selection% == 6 echo %DATE% %TIME% - Selection was made
          to acquire memory forensic image, volatile, and non-
          volatile data %COMPUTERNAME% >> %c_drive%:\Data-%case%\
          Collection.log
36    goto :main
```

A Listagem 3.9 mostra um excerto do *script tr3-collect.bat*. O comando *cls* (linha 5) limpa o terminal. Entre as linhas 8 e 12 , verifica-se se a pasta, onde será guardada toda a informação a recolher, já existe e, se não existir, é criada. Também será verificado se o ficheiro de logs da aplicação existe, se não existir, este é criado e é lhe adicionado um cabeçalho (linhas 14 a 21). Após isto, verifica qual foi a opção de recolha de informação pedida pelo utilizador e dá inicio a respetiva recolha (linhas 30-36).

Listagem 3.10: Exemplo de output do Tr3Secure no Windows

```
1  Command Executed: tasklist.exe
2
3  Image Name                     PID Session Name        Session#    Mem Usage
4  ========================= ======== ================ =========== ============
5  System Idle Process              0 Services                   0         24 K
6  System                           4 Services                   0      7.892 K
7  smss.exe                       364 Services                   0        808 K
8  csrss.exe                      440 Services                   0      3.644 K
9  csrss.exe                      488 Console                    1      6.464 K
10 wininit.exe                    496 Services                   0      3.304 K
11 winlogon.exe                   536 Console                    1      4.784 K
12 services.exe                   584 Services                   0      7.076 K
13 lsass.exe                      592 Services                   0     10.120 K
14 lsm.exe                        600 Services                   0      3.012 K
15 svchost.exe                    696 Services                   0      7.216 K
16 VBoxService.exe                760 Services                   0      4.764 K
17 svchost.exe                    824 Services                   0      6.344 K
18 MsMpEng.exe                    912 Services                   0    127.388 K
19 svchost.exe                    980 Services                   0     16.216 K
20 svchost.exe                   1012 Services                   0     11.928 K
21 svchost.exe                   1036 Services                   0     32.648 K
22 svchost.exe                   1172 Services                   0     12.016 K
```

```
23 svchost.exe              1252 Services    0     18.016 K
24 spoolsv.exe              1400 Services    0      9.000 K
25 svchost.exe              1476 Services    0     12.784 K
26 svchost.exe              1592 Services    0      7.264 K
27 svchost.exe              1624 Services    0     13.072 K
28 FCUpdateService.exe      1648 Services    0      4.064 K
29 svchost.exe              1776 Services    0      8.052 K
30 taskhost.exe             2024 Console     1      7.032 K
31 dwm.exe                  2168 Console     1      4.640 K
32 explorer.exe             2220 Console     1     87.360 K
33 NisSrv.exe               2264 Services    0      1.716 K
34 VBoxTray.exe             2756 Console     1      5.752 K
35 msseces.exe              2772 Console     1     11.080 K
36 PWRISOVM.EXE             3000 Console     1      3.964 K
37 taskeng.exe              3072 Console     1      4.276 K
38 CCleaner.exe             3104 Console     1      2.164 K
39 SearchIndexer.exe        3128 Services    0     46.064 K
40 wmpnetwk.exe             3248 Services    0      7.500 K
41 svchost.exe              3472 Services    0     11.724 K
42 audiodg.exe              3924 Services    0     14.168 K
43 svchost.exe              3684 Services    0      5.404 K
44 cmd.exe                  1612 Console     1      2.968 K
45 conhost.exe              1940 Console     1      4.752 K
46 tasklist.exe             3088 Console     1      4.264 K
47 WmiPrvSE.exe             2908 Services    0      4.832 K
```

A Listagem 3.10 apresenta um excerto do *output* do Tr3Secure em Windows. Neste caso em concreto apresenta-se o *output* do comando *tasklist.exe* (linha 1). Este comando, para cada processo em execução no PC em análise, detalha as seguintes características: nome do processo, Process Identifier (PID) do processo, nome da sessão, Identifier (ID) da sessão e memória usada.

Esta também só foi desenvolvida para Windows, assim, todos os exemplos mostrados para esta aplicação são em Windows.

3.4 Tr3Secure Master

TR3Secure master é outro *script* para recolha de evidências forenses digitais em sistemas ligados. É um *fork* do projeto Tr3Secure desenvolvido por Corey Harrell (ver Secção 2.3). Esta aplicação foi desenvolvida porque este analista sentiu necessidade de adaptar o Tr3Secure à forma de trabalho da sua equipa de resposta e triagem em situações de deteção de *malwares* [48].

Tal como a aplicação Tr3Secure, esta aplicação também é constituída por dois *scripts* e uma pasta com aplicações para auxiliar na recolha de informação. Um dos *scripts* é utilizado para recolher informação do sistema alvo e o outro é para recolher informação de uma determinada conta do sistema, mas só se vai testar o *script* que recolhe a informação do sistema alvo. Para ser utilizada, é necessário descarregá-la da Internet e extrai-la

para o suporte de armazenamento USB, de seguida, têm de se descarregar também as suas dependências. Estas dependências estão descritas no ficheiro *README.md*, que contém os respetivos *links*, e não são distribuídas em conjunto com a aplicação porque os seus autores colocam restrições à sua distribuição. As dependências devem ser guardadas numa pasta com nome *tools* e mantendo o nome que é utilizado no *script*. Após isto, terá de se abrir um terminal como administrador e mudar para a pasta para onde foi extraído. Nesse diretório, executa-se o comando demonstrado na Listagem 3.11.

Listagem 3.11: Executar Tr3Secure Master no Windows

```
1 tr3-collect.bat [case number] [drive letter for storing
      collected data] [menu selection #]
```

O comando requer três parâmetros sendo o último opcional. O parâmetro *[case number]* deve ser um identificador único para a situação em análise; será concatenado com *Data-* para formar o nome da pasta onde será guardada a informação recolhida. O parâmetro *[drive letter for storing collected data]* deve ser a letra da unidade de armazenamento de destino da recolha de informação. O parâmetro *[menu selection #]* deve indicar o tipo de informação a recolher. As possibilidades para este último parâmetro são:

1. Recolhe apenas uma cópia da memória RAM
2. Recolhe dados voláteis
3. Recolhe dados não voláteis
4. Recolhe dados voláteis e não voláteis
5. Recolhe uma cópia da memória RAM, dados voláteis e dados não voláteis

Se este parâmetro for suprimido, será efetuada a recolha 4. Tal como a aplicação original, esta também foi apenas desenvolvida para Windows.

Listagem 3.12: Script Tr3Secure Master para Windows

```
1  :: Main Processing Area
2  :: ------------------------------------------------------------
3  :: This section performs the data collection from the system
4  ::
5  :main
6      :: The main function creates the output folder, preserves the
           computer's prefetch files, and preserves the user account
           ntuser.dat file
7      :: Preserving the prefetch files and ntuser.dat file prevents
           them from being overwritten
8      ::
```

```
9     :: The collection output folder's name is based on the
          computer's name and the timestamp of when the data was
          collected
10    ::
11    :: Setting up a timestamp variable because %date% and %time%
          variables contain characters that cannot be used in folder
          names
12    :: Below sets up variables for the date by getting the month,
          day, and year in two digits
13    set m=%date:~4,2%
14    set d=%date:~7,2%
15    set y=%date:~12,2%
16    :: Below sets up variables for the time by getting the hour
          and minute in two digits
17    set hh=%time:~0,2%
18    set mm=%time:~3,2%
19    :: below accounts for single digit hours since Microsoft uses
          a blank and not zero
20    if "%hh:~0,1%" == " " set hh=0%hh:~1,1%
21    :: Creates the date/time variable for naming folders
22    set timestamp=%m%.%d%.%y%-%hh%.%mm%
23    :: Creating directory for output data. The naming convention
          allows the script to be
24    :: executed numerous times without overwriting previous
          output data
25    if not exist %c_drive%:\Data-%case%\%computername%-%timestamp
          % (
26       echo %DATE% %TIME% - Running tools\mkdir.exe on %
             COMPUTERNAME% to create the output directory %
             computername%-%timestamp%
27       echo %DATE% %TIME% - Running tools\mkdir.exe on %
             COMPUTERNAME% to create the output directory %
             computername%-%timestamp% >> %c_drive%:\Data-%case%\
             Collection.log
28       tools\mkdir.exe %c_drive%:\Data-%case%\%computername%-%
             timestamp%
29       )
30 :: ************************************************************
```

Na Listagem 3.12 temos um excerto do *script tr3-collect.bat* adaptado pelo analista da Tr3Secure Master. Entre as linhas 13 e 18 vão ser configuradas as variáveis utilizadas para obter o dia (linha 14), o mês (linha 13), o ano (linha 15), as horas (linha 17) e os minutos (linha 18) com 2 casas decimais. A linha 22 configura uma variável com a hora e a data que mais tarde será utilizada como parte do nome da pasta de *output*. As linhas 25 a 29 verificam a existência de tal pasta, caso não exista, procede-se à sua criação.

Listagem 3.13: Exemplo de um output do Tr3Secure Master no Windows

```
1 Command Executed: arp.exe -a
2
3
```

```
4  Interface: 10.0.2.15 --- 0xb
5  Endereco Internet      Endereco fisico       Tipo
6  10.0.2.2               52-54-00-12-35-02     dinamico
7  10.0.2.3               52-54-00-12-35-03     dinamico
8  10.0.2.255             ff-ff-ff-ff-ff-ff     estatico
9  224.0.0.22             01-00-5e-00-00-16     estatico
10 224.0.0.252            01-00-5e-00-00-fc     estatico
11 239.255.255.250        01-00-5e-7f-ff-fa     estatico
12 255.255.255.255        ff-ff-ff-ff-ff-ff     estatico
```

A Listagem 3.13 mostra um dos *outputs* do comando, neste caso, o resultado do comando *arp.exe -a*. Este comando mostra as entradas atuais da tabela de Address Resolution Protocol (ARP). Na linha 4 observa-se o Internet Protocol (IP) da máquina que foi analisada. Entre as linhas 6 e 12 pode observar-se que o comando lista as entradas da tabela ARP com os seguintes detalhes: endereço IP, endereço físico e tipo de endereço. Com o endereço Internet (IP) pode-se saber o IP da máquina com que foi comunicado, no endereço físico (Medium Access Control (MAC)) mostra o endereço MAC desse mesmo IP e no tipo mostra se o IP é dinâmico ou estático. Se o tipo for dinâmico significa que o IP pode mudar ao longo do tempo. Esta aplicação só foi desenvolvida para Windows.

3.5 Live Response BriMor

A aplicação Live Response desenvolvida pela Brimorlabs é também uma coleção de comandos e de *scipts* para a recolha de artefactos em cenários de investigação forense digital. Esta tem a particularidade de poder ser utilizada em sistemas operativos Windows, Linux e Mac. A aplicação foi testada numa ampla variedade de sistemas operativos [49]. Esta aplicação é constituída por alguns *scripts*, nomeadamente um *script* principal que executa outros *scripts*, armazenados dentro da pasta *Modules*, e aplicações, armazenados dentro da pasta *Tools*. Contém ainda um *script* para sistemas Linux e um outro para sistemas Windows que, depois de instalado, executa outros *scripts* de recolha de informação. Após a descarga da aplicação da Internet, esta está pronta a ser utilizada. O autor permite a sua distribuição. Após isto, terá de se abrir um terminal como administrador e mudar para a pasta para onde foi extraído. Nesse diretório, executa-se o comando demonstrado na Listagem 3.14.

Listagem 3.14: Executar Live Response BriMor no Linux

```
1 ./nix_Live_Response.sh
```

A aplicação não requer parâmetros adicionais para executar, basta simplesmente executar o comando para dar inicio à recolha da informação. Ao contrário do que acontece em sistemas Windows e Mac, em sistemas Linux a aplicação não recolhe uma cópia da memória RAM. Esta aplicação

não é homogénea em termos de recolha de informação nos vários sistemas operativos suportados.

Listagem 3.15: Script Live Response BriMor no Linux

```
1  #BASIC INFORMATION
2
3  date >> $computername/LiveResponseData/BasicInfo/date.txt
4  echo "date"
5  hostname >> $computername/LiveResponseData/BasicInfo/hostname.
       txt
6  echo "hostname"
7  who >> $computername/LiveResponseData/BasicInfo/Logged_In_Users.
       txt
8  echo "who"
9  ps aux --forest >> $computername/LiveResponseData/BasicInfo/
       List_of_Running_Processes.txt
10 echo "ps aux --forest"
11 pstree -ah >> $computername/LiveResponseData/BasicInfo/
       Process_tree_and_arguments.txt
12 echo "pstree -ah"
13 mount >> $computername/LiveResponseData/BasicInfo/Mounted_items.
       txt
14 echo "mount"
15 diskutil list >> $computername/LiveResponseData/BasicInfo/
       Disk_utility.txt
16 echo "diskutil"
17 kextstat -l >> $computername/LiveResponseData/BasicInfo/
       Loaded_Kernel_Extensions.txt
18 echo "kextstat -l"
19 uptime >> $computername/LiveResponseData/BasicInfo/System_uptime
       .txt
20 echo "uptime"
21 uname -a >> $computername/LiveResponseData/BasicInfo/
       System_environment.txt
22 echo "uname -a"
23 printenv >> $computername/LiveResponseData/BasicInfo/
       System_environment_detailed.txt
24 echo "prinenv"
25 cat /proc/version >> $computername/LiveResponseData/BasicInfo/
       OS_kernel_version.txt
26 echo "cat /proc/version"
27 top -n 1 -b >> $computername/LiveResponseData/BasicInfo/
       Process_memory_usage.txt
28 echo "top -n 1 -b"
29 service --status-all | grep + >> $computername/LiveResponseData/
       BasicInfo/Running_services.txt
30 echo "service --status-all | grep +"
31 lsmod | head >> $computername/LiveResponseData/BasicInfo/
       Loaded_modules.txt
32 echo "lsmod | head"
33 last >> $computername/LiveResponseData/BasicInfo/Last_logins.txt
34 echo "last"
```

A Listagem 3.15 apresenta um extrato do *script nix_Live_Response.sh* para Linux. No caso em concreto, apresenta-se a informação básica que o *script* guarda na pasta *BasicInfo*, gerando um ficheiro para cada comando executado pelo *script*. Os comandos usados para recolher a informação básica são: date (linha 3), hostname (linha 5), who (linha 7), ps (linha 9), pstree (linha 11), mount (linha 13), diskutil (linha 15), kextstat (linha 17), uptime (linha 19), uname (linha 21), printenv (linha 23), cat (linha 25), top (linha 27), service (linha 29), lsmod (linha 31) e last (linha 33). Depois de executar um comando e guardar o seu output num ficheiro, será mostrado no terminal o comando que acabou de ser executado. Por exemplo, a linha 3 guarda num ficheiro a data e hora do sistema, já a linha 4 mostra que acabou de executar o comando date.

Listagem 3.16: Exemplo de um output do Live Response BriMor no Linux

```
1  eth0      Link encap:Ethernet  HWaddr f0:bf:97:03:5c:68
2            UP BROADCAST MULTICAST  MTU:1500  Metric:1
3            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
4            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
5            collisions:0 txqueuelen:1000
6            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
7
8  eth1      Link encap:Ethernet  HWaddr 08:2e:5f:29:0e:69
9            inet addr:192.168.1.12  Bcast:192.168.1.15  Mask
                 :255.255.255.252
10           inet6 addr: fe80::a2e:5fff:fe29:e69/64 Scope:Link
11           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
12           RX packets:3598185 errors:159 dropped:0 overruns:0
                 frame:159
13           TX packets:2879133 errors:0 dropped:0 overruns:0
                 carrier:0
14           collisions:0 txqueuelen:1000
15           RX bytes:4387661278 (4.3 GB)  TX bytes:585479827
                 (585.4 MB)
16
17 lo        Link encap:Local Loopback
18           inet addr:127.0.0.1  Mask:255.0.0.0
19           inet6 addr: ::1/128 Scope:Host
20           UP LOOPBACK RUNNING  MTU:65536  Metric:1
21           RX packets:47687 errors:0 dropped:0 overruns:0 frame:0
22           TX packets:47687 errors:0 dropped:0 overruns:0 carrier
                 :0
23           collisions:0 txqueuelen:0
24           RX bytes:4997351 (4.9 MB)  TX bytes:4997351 (4.9 MB)
25
26 vboxnet0  Link encap:Ethernet  HWaddr 0a:00:27:00:00:00
27           BROADCAST MULTICAST  MTU:1500  Metric:1
28           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
29           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
30           collisions:0 txqueuelen:1000
31           RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
32
```

```
33 wlan0     Link encap:Ethernet  HWaddr 48:5d:60:a4:b8:6b
34           inet6 addr: fe80::4a5d:60ff:fea4:b86b/64 Scope:Link
35           UP BROADCAST MULTICAST  MTU:1500  Metric:1
36           RX packets:25236 errors:0 dropped:0 overruns:0 frame:0
37           TX packets:14918 errors:0 dropped:0 overruns:0 carrier
                 :0
38           collisions:0 txqueuelen:1000
39           RX bytes:19726394 (19.7 MB)  TX bytes:2533816 (2.5 MB)
```

Na Listagem 3.16 pode observar-se *output* do comando *ifconfig*, que pode ser usado para ver e alterar a configuração de rede num sistema Linux. Onde eth0, lo e wlan0 são os nomes da redes ativas no sistema. A eth0 (linha 1) é a primeira *ethernet* ativa. A lo (linha 17) é a interface de *loopback*, sendo uma interface de rede especial que o sistema usa para comunicar com ele próprio. A vboxnet0 (linha 26) é a primeira rede configura para ser usada no VirtualBox. Por fim, menciona-se a wlan0 (linha 33), sendo esta a primeira interface de rede *wireless* no sistema.

A aplicação foi testada nos três sistemas operativos (Windows, Linux e MacOS) de referência e conclui-se que não há homogeneidade na informação recolhida. Na informação de rede o MacOS recolhe mais informação. Na informação de estado, sistema e utilizador, a versão para Windows recolhe mais informação. Quanto a informação sobre dispositivos e à cópia da RAM há um equilíbrio nos sistemas Windows e MacOS, mas bastante mais deficitário no caso do Linux.

3.6 Live Response Nekyia

A aplicação Nekyia Live Response Scripts assume a forma de três *scripts* de recolha de informação volátil. Um para cada sistema operativo de referência. Os *scripts* para Linux e macOS não dependem de aplicações externas [50]. Após a descarga da Internet, a aplicação está pronta a ser executada. Para tal, terá de se abrir um terminal como administrador e mudar para a pasta para onde foi descarregado. Nesse diretório, executa-se o comando demonstrado na Listagem 3.17.

Listagem 3.17: Executar Live Response Nekyia no Linux

```
1 ./lie.sh [> lie_'hostname'.txt]
```

Tal como demonstrado na Listagem 3.17, caso se queira guardar o *output* do *script* num ficheiro, terá de se utilizar técnicas de redirecção para ficheiro.

Listagem 3.18: Script Live Response Nekyia para Linux

```
1 console "++ Storing uptime\n"
2 echo -n "Uptime: "
3 uptime
4 echo ""
5
```

```
6  console "\n+ Obtaining Network Information\n"
7  echo "## Network Information"
8
9  console "++ Storing interfaces\n"
10 echo "### Interfaces"
11 ifconfig -a
12
13 console "++ Storing open connections\n"
14 echo "### Netstat"
15 netstat -ano
16 echo ""
17
18 console "++ Storing routing table\n"
19 echo "### Routing table"
20 netstat -rn
21 echo ""
22
23 console "\n+ Obtaining User Information\n"
24 echo "## User Information"
25
26 console "++ Storing logged in users\n"
27 echo "### Currently logged in users"
28 who -aH
29 echo ""
30
31 echo "### Current activies by user"
32 w
33 echo ""
```

A Listagem 3.18 apresenta um excerto de código do *script lie.sh* para Linux. Esta aplicação não faz divisão da recolha de informação por mais do que um *script*. Na linha 1 observa-se a utilização da função *console* que é uma função que envia mensagens para a consola através de STDERR, informando o utilizador do progresso da aplicação enquanto redireciona o STDOUT para relatório. Na linha 2 o comando *echo* mostra a mensagem que aparece no terminal. Na linha 3, surge então o comando que vai recolher a informação do sistema. O *script* adota esta abordagem para a restante recolha de informação.

Listagem 3.19: Exemplo de um output do Live Response Nekyia no Linux

```
1  Kernel Version:
2  Linux Fabio-PC 4.3.3-2-ARCH #1 SMP PREEMPT Wed Dec 23 20:09:18
       CET 2015 x86_64 GNU/Linux
3
4  Proc Version:
5  Linux version 4.3.3-2-ARCH (builduser@tobias) (gcc version 5.3.0
       (GCC) ) #1 SMP PREEMPT Wed Dec 23 20:09:18 CET 2015
6
7  Uptime:  19:59:53 up  6:51,  1 user,  load average: 0,44, 0,32,
       0,26
8
9  ## Network Information
```

```
10 ### Interfaces
11 enp19s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
12         ether f0:bf:97:03:5c:68  txqueuelen 1000  (Ethernet)
13         RX packets 0  bytes 0 (0.0 B)
14         RX errors 0  dropped 0  overruns 0  frame 0
15         TX packets 0  bytes 0 (0.0 B)
16         TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
17
18 lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
19         inet 127.0.0.1  netmask 255.0.0.0
20         inet6 ::1  prefixlen 128  scopeid 0x10<host>
21         loop  txqueuelen 0  (Local Loopback)
22         RX packets 42260  bytes 87627382 (83.5 MiB)
23         RX errors 0  dropped 0  overruns 0  frame 0
24         TX packets 42260  bytes 87627382 (83.5 MiB)
25         TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
26
27 wlp7s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
28         inet 192.168.1.2  netmask 255.255.255.0  broadcast
               192.168.1.255
29         inet6 fe80::4a5d:60ff:fea4:b86b  prefixlen 64  scopeid 0
               x20<link>
30         ether 48:5d:60:a4:b8:6b  txqueuelen 1000  (Ethernet)
31         RX packets 194988  bytes 234548406 (223.6 MiB)
32         RX errors 0  dropped 0  overruns 0  frame 0
33         TX packets 141539  bytes 20631183 (19.6 MiB)
34         TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

A Listagem 3.19 apresenta um excerto do *output* do *script Live Response Nekyia*. Aqui pode se observar o output do comando *ifconfig* (linhas 9 a 34) mostrando as redes ativas no sistema operativo. Também se pode observar o comando *uname -a* para se saber a versão do kernel, da arquitetura, a data e o nome do computador (linha 2). Foi ainda usado o comando *cat /proc/version* para ver versão do *kernel* e do GNU Compiler Collection (GCC) (linha 5). Por fim, pode se observar o *output* do comando *uptime* (linha 7), que indica à quanto tempo o computador está ligado.

Esta aplicação foi testada nos três sistemas operativos de referência (Windows, Linux e macOS) e conclui-se que não há homogeneidade quanto aos dados obtidos pela aplicação nos vários sistemas. Na informação de rede e de sistema, a aplicação gera mais informação na sua versão para Windows. Há diferenças também na informação recolhida sobre utilizadores e dispositivos.

3.7 Triage-Responder

Esta aplicação não foi testada. É uma aplicação comercial que não tem livremente disponíveis *online* versões de demonstração [51]. Foi solicitada uma versão de demonstração ao fabricante da aplicação, mas não foi recebida

nenhuma resposta. A análise a esta aplicação incidiu sobre a informação comercial disponível *online*.

Esta aplicação é constituída por uma mala de transporte, um suporte de armazenamento USB, um CD-ROM de arranque e uma agulha com cabo de plástico[1]. A aplicação suporta múltiplas plataformas, incluindo Windows, Linux e macOS, em análise tradicional. A recolha em sistemas ligados só é possibilitada para sistemas Windows.

3.8 Comparação

A comparação das várias aplicações descritas neste capítulo está vertida na Tabela 3.1. Em particular, foram analisados 3 fatores principais. Analisou-se se as aplicações têm suporte multi-plataforma, a quantidade de informação recolhida por estas e, por fim, se a informação recolhida pelas aplicações era homogénea entre os vários sistemas operativos suportados.

Tabela 3.1: Taxa de recolha de informação (em %)

	Ir-triage			IRKIT			Tr3Secure			Tr3Secure Master			LiveResponseBriMor			Nekyia		
Informação	W	L	M	W	L	M	W	L	M	W	L	M	W	L	M	W	L	M
Rede	60	20	-	80	-	-	100	-	-	100	-	-	60	80	100	80	60	60
Estado	67	67	-	67	-	-	67	-	-	67	-	-	100	67	67	67	67	67
Sistema	-	50	-	75	-	-	50	-	-	50	-	-	100	50	50	75	50	50
Utilizador	33	67	-	100	-	-	100	-	-	100	-	-	67	33	33	33	67	67
Dispositivos	100	50	-	100	-	-	50	-	-	50	-	-	100	50	100	-	100	100
Integridade	100	100	-	-	-	-	-	-	-	-	-	-	100	100	100	-	-	-
Logs	-	-	-	-	-	-	100	-	-	100	-	-	50	100	100	-	-	-
Autorun	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Cópia RAM	100	100	-	100	-	-	100	-	-	100	-	-	100	-	100	-	-	-

A Tabela 3.1 compara o nível de informação que cada aplicação recolhe em cada sistema operativo (Windows (W), Linux (L) e macOS (M)). A informação a recolher foi agrupada em informação de: rede, estado, sistema, utilizador, dispositivos, integridade e registos.

A informação de rede considera a configuração de interfaces de rede, endereços IP e MAC, tabela de encaminhamento, tabela de ARP, *cache* DNS; ligações de rede e processos associados. A informação de estado considera os ficheiros abertos e outros identificadores, processos em execução e seus detalhes (ex.: número de identificação do processo, tempo em execução) e lista completa de sistemas de ficheiros. A informação de sistema considera a data e hora do sistema, incluindo o fuso horário, versão do sistema operativo, lista de serviços e programas configurados para iniciar automaticamente no arranque, dados de configuração do sistema e a lista do *software* instalado. A informação de utilizador considera a lista de tarefas agendadas para executar automaticamente em determinados momentos ou intervalos, a lista de contas de utilizadores locais e de grupos e o histórico de acessos ao sistema,

[1] Esta agulha pode ser utilizada em situações de IR quando é necessário mover pequenos objetos ou cabos numa cena em investigação.

incluindo nome do utilizador, fonte e duração. A informação de dispositivos considera os *drivers* ou módulos carregados e a capacidade da memória, discos rígidos e sistemas de ficheiros montados. Informação de integridade considera a operação de funções criptográficas de resumo (MD5 [14], SHA-1 [52], SHA512 [1]), sobre os resultados gerados pela própria aplicação, visando a garantia da sua integridade. Por último, a informação de registos (ou *logs*) considera os *logs* do sistema operativo e *logs* de aplicações específicas (ex.: histórico *web browser*)[6].

A cada grupo de informação foi atribuída a percentagem que cada aplicação recolhe desse grupo de informação. Para tal, calcula-se o número de itens recolhidos para cada grupo de informação sobre o total de itens de cada grupo de informação, obtendo assim a percentagem recolhida pela aplicação desse grupo de informação.

3.9 Conclusão

Neste capítulo foram apresentadas várias aplicações que possibilitam a recolha de artefactos de sistemas ligados em cenários de IR. Constata-se que existem várias aplicações para este propósito, o que demonstra o interesse no assunto pela comunidade. As várias aplicações foram analisadas e testadas. Foi elaborada uma comparação das várias aplicações. A informação que estas aplicações recolhem não é homogénea e algumas só suportam um ou dois dos sistemas operativos de referência.

Capítulo 4

DFIRU

Num contexto de uma investigação forense digital ou de resposta a um incidente de segurança informática é importante que o técnico esteja dotado de aplicações adequadas, que cumpra um procedimento de recolha padronizado e que garanta a integridade dos artefactos recolhidos. A solução proposta visa dotar o técnico de uma aplicação que lhe permita recolher artefactos de forma padronizada, com garantia de integridade e tão automatizada quanto possível.

A análise ao atual estado da arte demonstra que nenhuma solução existente cumpre com os requisitos identificados de seguida. Razão que motivou o presente trabalho, denominado de *Digital Forensic and Incident Response* USB (DFIRU).

4.1 Requisitos

A solução proposta tem como principais requisitos:

1. **Multi-plataforma:** a aplicação a desenvolver deve ser capaz de recolher artefactos e outra informação volátil na generalidade dos sistemas operativos mais comuns.

2. **Portabilidade:** a aplicação a desenvolver deverá ser transportável num dispositivo de reduzidas dimensões físicas.

3. **Auto-executável:** a aplicação deverá, sempre que possível, iniciar-se de forma automática.

4. **Alcance:** a aplicação deverá recolher o máximo de informação possível de cada sistema à qual for ligado.

5. **Registo de atividade:** a aplicação deverá guardar um registo da sua atividade para ficheiro.

6. **Integridade:** a aplicação deverá utilizar funções criptográficas de resumo sobre toda a informação recolhida ou gerada.

4.2 Arquitetura da solução

A aplicação é constituída por uma aplicação principal, desenvolvida na linguagem Java, que executa *scripts* para Windows, Linux e macOS. Por sua vez, estes *scipts* vão executar os comandos que farão a recolha de informação no sistema alvo.

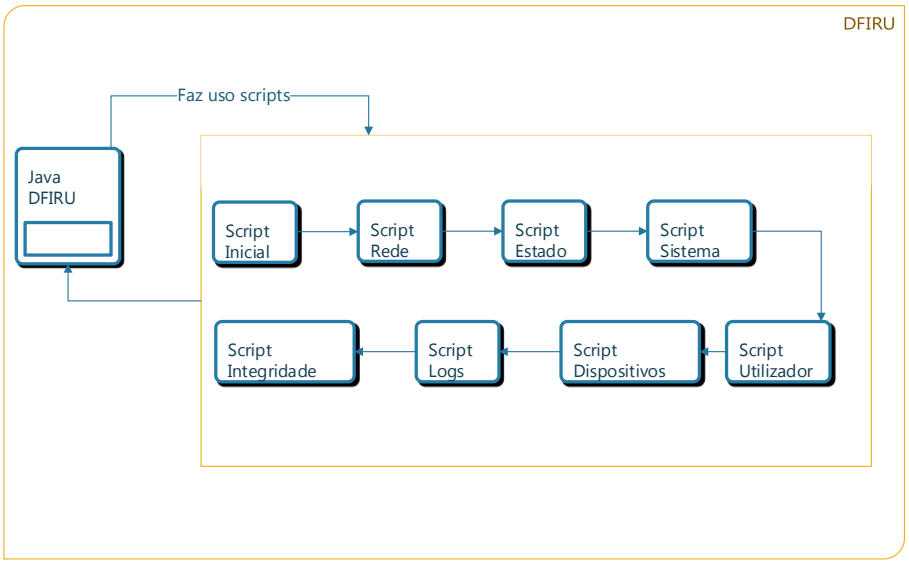

Figura 4.1: Arquitetura da aplicação DFIRU

Como se pode observar na Figura 4.1, foram criados *scripts* distintos para os vários sistemas operativos com o intuito destes serem homogéneos no que diz respeito à informação recolhida. Os *scripts* estão agrupados em 7 tipos de informação a recolher. Estes grupos de informação são:

1. Rede
2. Estado
3. Sistema
4. Utilizador
5. Dispositivos
6. Logs

7. Integridade.

Para recolher a informação desses grupos foram criados 21 *scripts*, que foram distribuídos pelos grupos da seguinte maneira: 2 para Rede, 2 para Estado, 2 para Sistema, 2 para Utilizador, 2 para Dispositivos, 2 para Logs e 8 para Integridade e sobrando 1 *script* que é onde será criado as pastas e ficheiros necessários para guardar a informação recolhida (*script* inicial). A necessidade de 2 *scripts* por grupo está relacionado com o nível de privilégios do utilizador em uso na recolha (com ou sem privilégios de administrador).

Para o grupo de Integridade foram criados *scripts* adicionais já que este grupo tem funcionalidades extras, selecionáveis no menu, que o tornam muito extenso. Estes estão ainda divididos entre *scripts* para utilizadores com e sem privilégios de administrador.

Tabela 4.1: Relação informação/comando - Windows[2]

Informação	Windows
Rede	ipconfig /all netstat -rn arp.exe -a ipconfig /displaydns netstat -ano
Estado	OpenedFilesView.exe /stext x.txt pslist.exe /accepteula dir /S /B /AHD %HOMEDRIVE%
Sistema	time/T date/T w32tm /tz ver wmic /Output:"x.txt"startup list full wmic /Output:"x.txt"product get Name, Version
Utilizador	schtasks /query /fo LIST /v dumpsec.exe /rpt=users /saveas=fixed /outfile=x.txt logonsessions.exe /accepteula NetUsers.exe /h /v
Dispositivos	wmic.exe /Output:"x.txt"diskdrive list brief /format:list di.exe
Logs	psloglist.exe /accepteula RawCopy.exe %WINDIR%\System32\winevt\Logs\Application.evtx %LOGS%
Integridade	md5deep.exe -r ficheiro(s) sha1deep.exe -r ficheiro(s) sha256deep.exe -r ficheiro(s) winpmem_2.1.exe -o ficheiro

Nas tabelas 4.1, 4.2 e 4.3 observam-se os comandos utilizados pelos *scripts* para recolher a informação relativa aos vários grupos, para cada

Tabela 4.2: Relação informação/comando - Linux[3]

Informação	Linux
Rede	ifconfig -a netstat -rn arp -a netstat -anp
Estado	lsof top -n 1 -b ls -Rlsah /
Sistema	date uname -a ls -l /home/$USER/.config/autostart/ dpkg -l
Utilizador	crontab -l cat /etc/passwd last
Dispositivos	mount df -k
Logs	cp /var/log/*.log* $LOGS
Integridade	find / -type f -exec md5sum \; find / -type f -exec sha1sum \; find / -type f -exec sha256sum \; sha256sum ficheiro insmod lime-*.ko ”path=../../$NOME_MEM_DUMP format=lime”

Tabela 4.3: Relação informação/comando - macOS[4]

Informação	macOS
Rede	ifconfig -a netstat -rn arp -a killall -INFO mDNSResponder netstat
Estado	lsof ps auxwww ls -Rlsah /
Sistema	systemsetup -getdate systemsetup -gettime systemsetup -gettimezone uname -a find /Applications/ -name LogInItems -exec ls -lsct \; ls -l /Applications/
Utilizador	crontab -l cat /etc/passwd last
Dispositivos	mount df -k
Logs	cp /var/log/*.log* $LOGS cp –r /etc/cron* $LOGS
Integridade	find / -type f -exec md5sum \; find / -type f -exec sha1sum \; find / -type f -exec shasum -a 256 \; shasum -a 256 ficheiro ./osxpmem $NOME_MEM_DUMP

sistema operativo. Na tabela 4.2 também se pode observar que os comandos exibidos são para o sistema operativo Linux baseado em Debian. Nos *scripts* também existem comandos alternativos para outras distribuições Linux tais como Arch, Red Hat, Slackware e Gentoo.

Listagem 4.1: Script para recolha de informação de rede - Linux

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  REDE=$PCNAME/Rede
7
8  #Nomes Ficheiros
9  INTERFACE_REDE=$REDE/interface_rede.txt
10 TABELA_ROTEAMENTO=$REDE/tabela_roteamento.txt
11 TABELA_ARP=$REDE/tabela_arp.txt
12 CACHE_DNS=$REDE/cache_dns.txt
13 LiGACOES_REDE=$REDE/ligacoes_rede.txt
14
15 ###REDE
16 #Interface Rede
17 printf "Interface de Rede\n\n" >> $INTERFACE_REDE
18 printf "Comando: ifconfig -a\n" >> $CALCULO_SHA256_COMANDOS
19 printf "\nOutput:\n" >> $INTERFACE_REDE
20 ifconfig -a >> $INTERFACE_REDE 2> /dev/null
21 echo concluido
22
23 #Tabela Roteamento
24 printf "Tabela de Roteamento\n\n" >> $TABELA_ROTEAMENTO
25 printf "Comando: netstat -rn\n" >> $TABELA_ROTEAMENTO
26 printf "\nOutput:\n" >> $TABELA_ROTEAMENTO
27 netstat -rn >> $TABELA_ROTEAMENTO 2> /dev/null
28 echo concluido
29
30 #Tabela ARP
31 printf "Tabela Arp\n\n" >> $TABELA_ARP
32 printf "Comando: arp -a\n" >> $TABELA_ARP
33 printf "\nOutput:\n" >> $TABELA_ARP
34 arp -a >> $TABELA_ARP 2> /dev/null
35 echo concluido
36
37 #Ligações de Rede
38 printf "Ligações de Rede\n\n" >> $LiGACOES_REDE
39 printf "Comando: netstat -anp\n" >> $LiGACOES_REDE
40 printf "\nOutput:\n" >> $LiGACOES_REDE
41 netstat -anp >> $LiGACOES_REDE 2> /dev/null
42 echo concluido
```

Na listagem 4.1 pode observar-se o *script* de recolha de informação de rede para o sistema operativo Linux e está subdividido nos seguintes blocos: Interface Rede; Tabela de Roteamento e de ARP; Cache Domain Name

System (DNS) e Ligações de Rede. O *script* contém uma secção inicial de processamento de argumentos (linhas 3 a 6). Na secção seguinte (linhas 9 a 14) definem-se as variáveis que contêm os caminhos para os ficheiros de armazenamento de informação. As secções seguintes dizem respeito à recolha de informação para cada um dos blocos. Estas secções estão todas organizadas da mesma forma, iniciando-se pela indicação do bloco (ex.: linha 17), do comando a executar (ex.: linha 18), do *output* do comando (ex.: 20) e da indicação de que o comando terminou (ex.: linha 21). Esta informação de término é utilizada para avançar a barra de progressão da aplicação principal em Java. Os restantes blocos de informação assumem um procedimento equivalente.

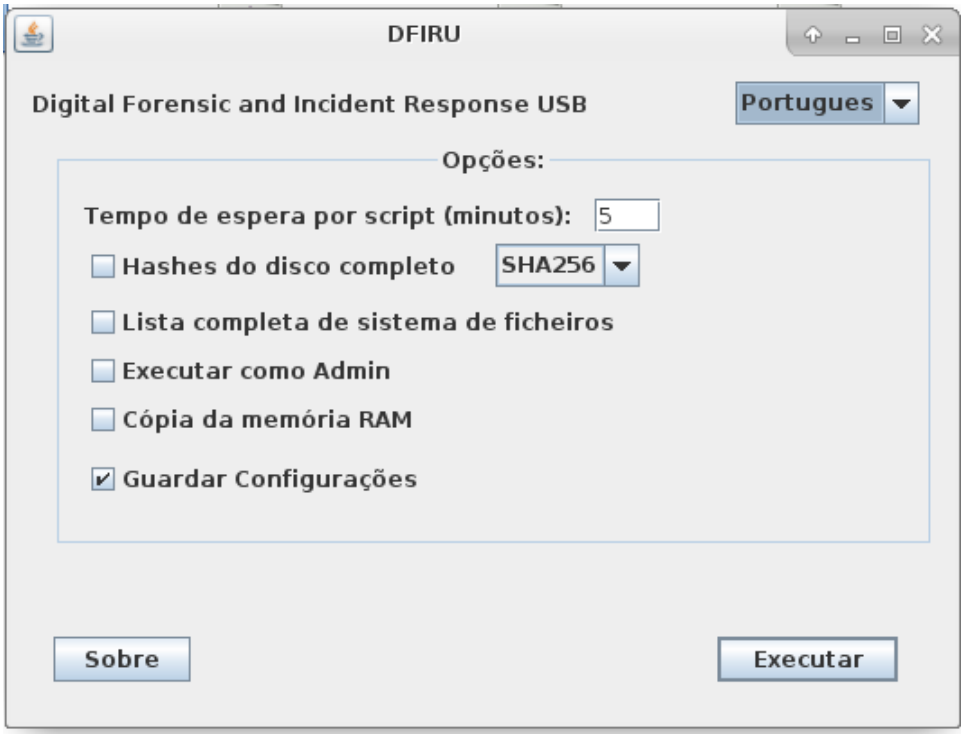

Figura 4.2: Janela Principal da aplicação DFIRU

A Figura 4.2 apresenta a janela principal da solução proposta. Esta contém uma *comboBox* que permite a seleção do idioma a utilizar pela aplicação (Português, Inglês e Espanhol). O suporte multi-língua foi consigo através de ficheiros de tradução (um para cada idioma) onde se definem constantes que depois são utilizadas para substituir os textos dos elementos do ambiente gráfico.

A aplicação permite ao utilizador escolher quais as funcionalidades que quer executar. As funcionalidades que o utilizador pode ativar/desativar são:

- Tempo de espera por *script*
- *Hashes* do disco completo
- Lista completa do sistema de ficheiros
- Executar como administrador
- Cópia da memória RAM
- Guardar configurações

A opção tempo de espera por *script* permite ao utilizador definir um tempo máximo, em minutos, permitido para a execução de cada *script*. Quando ultrapassado, a aplicação pergunta ao utilizador se este quer cancelar a execução do *script*. Esta opção tempo de espera por *script* foi criado devido ao facto de se utilizarem comandos que podem ser muito demorados.

A opção de *hashes* do disco completo, quando selecionada, leva a que a aplicação construa uma listagem de *hashes* de todos os ficheiros do disco rígido, usando o algoritmo escolhido.

A opção de execução como administrador possibilita que a aplicação se adapte ao nível de privilégios que o utilizador que executa a aplicação tem. A cópia da memória RAM, por exemplo, só é permitida caso o utilizador tenha privilégios de administração.

Listagem 4.2: Ficheiro de configuração DFIRU

```
1 time=5
2 hashs=false
3 typeHash=SHA256
4 list=false
5 runAdmin=true
6 dump=true
```

A opção guardar configurações, quando selecionada, faz com que a aplicação armazene a configuração atual. A aplicação, caso exista um ficheiro configuração, arranca com a configuração já existente. A listagem 4.2 mostra um exemplo de uma configuração armazenada. Neste caso pode-se observar que, na última configuração da aplicação, o tempo de espera foi definido para 5 minutos (linha 1), que não se quer *hashes* do disco rígido (linha 2), não se quer a listagem dos ficheiro do disco (linha 4), que o utilizador tinha privilégios de administração (linha 5) e que se quer uma cópia da memória RAM (linha 6).

A imagem 4.3 apresenta a janela que aparece quando o utilizador quer executar a aplicação como administrador. É validado se a *password* de administrador inserida está correta ou não.

A imagem 4.4 mostra a execução da aplicação DFIRU. Também se observa que esta aplicação está a ser executada em Linux, que já recolheu toda

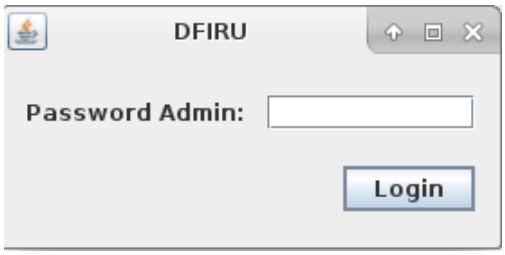

Figura 4.3: Janela para marcar password do administrador

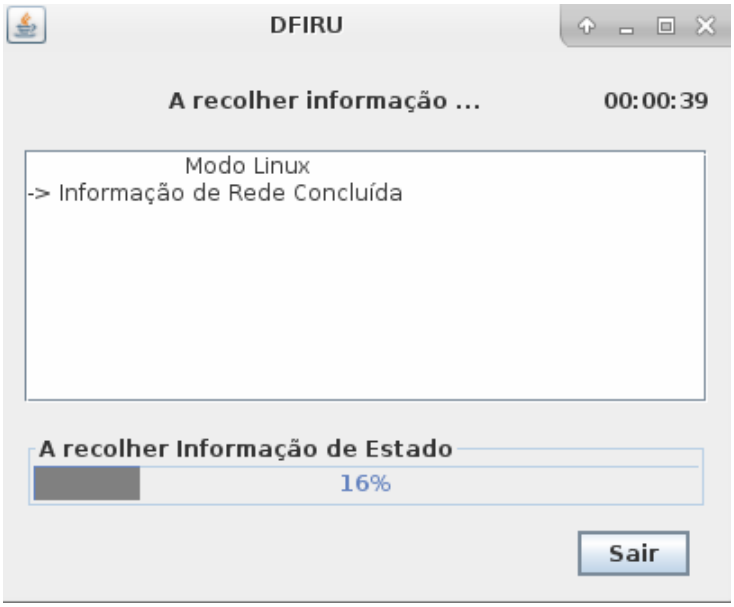

Figura 4.4: Janela de execução

a informação de Rede, que está a recolher a informação de Estado e que já se passaram 39 segundos desde que a aplicação foi iniciada.

Listagem 4.3: Extrato de um output do DFIRU no Linux

```
1 Lista de software instalado
2
3 Comando: sudo rpm -qa
4
5 Output:
6 texlive-uhc-doc-svn16791.0-19.fc23.noarch
7 texlive-ltabptch-svn17533.1.74d-19.fc23.noarch
8 (...)
```

A listagem 4.3 mostra um exemplo do *output* da aplicação DFIRU após execução num sistema operativo Linux. É apresentado um excerto de um ficheiro gerado pelo *script* de recolha de informação do Sistema (lista do software instalado). Pode observar-se que o sistema operativo usa pacotes *Red Hat Package Manager (RPM)* e que se trata do sistema operativo Linux Fedora (pacotes contêm *.fc23*). Na linha 3 surge o comando utilizado.

4.3 Conclusão

A solução proposta consiste numa aplicação desenvolvida em Java que executa um conjunto de *scripts* (Windows, Linux e macOS). A opção pelo Java possibilitou a execução da solução proposta em múltiplas plataformas. Os *scripts*, diferentes para cada plataforma suportada, vão ser executados para recolher a informação do sistema alvo. Tendo em vista o alcance e homogeneidade da informação recolhida, os *scripts* foram desenvolvidos para recolherem tanta informação quanto possível, desde que tal informação pudesse ser recolhida nas várias plataformas suportadas. A aplicação termina com a execução de um *script* que gera resumos criptográficos de todos os ficheiros de informação gerados pelos demais *scripts* e dos comandos utilizados por estes. A informação gerada é armazenada no suporte onde é executada a aplicação.

Capítulo 5

Avaliação do trabalho

Neste capítulo serão tratados os resultados que se obteve com a utilização da aplicação desenvolvida, bem como das aplicações descritas no capítulo 3, comparando-as e avaliando os resultados obtidos.

A proposta de solução, para ser executada de acordo com a sua especificação, deverá ser instalada num dispositivo de armazenamento USB. O processo de instalação passa por vários passos que incluem a formatação do dispositivo de armazenamento USB, a cópia da aplicação e *scripts* associados para as pastas corretas dentro do dispositivo de armazenamento USB e a cópia ou *download* dos utilitários necessários aos vários *scripts*.

Foi desenvolvido um conjunto de *scripts* para facilitar a instalação da aplicação no dispositivo de armazenamento USB a ser utilizado pelo técnico. Tal permitiu agilizar todo o processo de testes e validação da aplicação, executando-se este o processo para os vários sistemas operativos suportados.

5.1 *Script* de instalação

O *script* de instalação tem variantes para os vários sistemas operativos (Linux, Windows e macOS). O principal objectivo é o de agilizar todo o processo de preparação de um dispositivo de armazenamento de USB, mas também permite contornar problemas relacionados com licenciamento de *software*.

Em particular, alguns dos utilitários utilizados pelos *scripts* são de utilização gratuita mas de distribuição condicionada. Tal significa que não poderiam ser incluídos num pacote de *software* a distribuir por outros, por exemplo, *online*. A forma encontrada para contornar esta limitação, consistiu em fazer com que o *script* de instalação fizesse o *download* do *site* original.

Listagem 5.1: Exemplo do *script instalação Windows*

```
1 @echo off
2
3 set SITE=site
```

```
4  set PASTA=software.zip
5  set NOME_PEN=DFIRU
6
7  :start
8  set /p drive=Enter the drive letter to be formatted:
9
10 if '%drive%' == 'D' goto proceed
11 if '%drive%' == 'E' goto proceed
12 if '%drive%' == 'F' goto proceed
13 if '%drive%' == 'G' goto proceed
14 if '%drive%' == 'H' goto proceed
15 if '%drive%' == 'I' goto proceed
16 if '%drive%' == 'J' goto proceed
17 if '%drive%' == 'K' goto proceed
18 if '%drive%' == 'L' goto proceed
19 if '%drive%' == 'M' goto proceed
20 if '%drive%' == 'N' goto proceed
21 if '%drive%' == 'O' goto proceed
22 if '%drive%' == 'P' goto proceed
23 if '%drive%' == 'Q' goto proceed
24 if '%drive%' == 'R' goto proceed
25 if '%drive%' == 'S' goto proceed
26 if '%drive%' == 'T' goto proceed
27 if '%drive%' == 'U' goto proceed
28 if '%drive%' == 'V' goto proceed
29 if '%drive%' == 'W' goto proceed
30 if '%drive%' == 'X' goto proceed
31 if '%drive%' == 'Y' goto proceed
32 if '%drive%' == 'Z' goto proceed
33 ...
34 cls
35 goto error
36
37 :error
38 echo.
39 echo The drive letter you entered was not recognized
40 echo.
41 pause
42 cls
43 goto start
44
45 :proceed
46 cls
47 echo.
48 echo Would you like to format this flash drive?
49 echo.
50 echo Type Y for Yes or N for NO then press Enter
51 set /p ok=
52 if '%ok%' == 'y' goto yes
53 if '%ok%' == 'Y' goto yes
54 if '%ok%' == 'n' goto no
55 if '%ok%' == 'N' goto no
56 cls
57
```

```
58 :yes
59 cls
60
61 format %drive%: /fs:EXFAT /v:%NOME_PEN%
62 goto no
63
64 :no
65 wget.exe %SITE%
66 move %PASTA% "%drive%:\"
67 xcopy unzip.exe %drive%:\
68 xcopy wget.exe %drive%:\
69 cd %drive%:\
70 unzip.exe %PASTA%
71 del %PASTA%
72
73 cd tools\windows
74
75 :: logonsessions download
76 %drive%:\wget.exe https://download.sysinternals.com/files/
       logonSessions.zip
77 %drive%:\unzip.exe logonSessions.zip
78
79 :: pslist download
80 %drive%:\wget.exe https://download.sysinternals.com/files/
       PSTools.zip
81 %drive%:\unzip.exe PSTools.zip
82
83 :: psloglist
84 %drive%:\wget.exe https://download.sysinternals.com/files/
       PSTools.zip
85 %drive%:\unzip.exe PSTools.zip
86
87 del %drive%:\wget.exe
88 del %drive%:\unzip.exe
```

A Listagem 5.1 mostra um excerto do *script* de instalação do DFIRU para sistemas operativos Windows. Foram desenvolvidas versões equivalentes para a instalação em sistemas operativos Linux e macOS. Na Listagem 5.1 podemos observar que existem variáveis para guardar informação geral (PASTA e NOME_PEN) que visam facilitar a sua utilização ao longo do *script* e, se necessário, facilitar futuras adaptações aos *scripts*(linhas 3-5). O utilizador é questionado sobre a letra da unidade do dispositivo de armazenamento USB a ser utilizado (linhas 8), depois de introduzida, esta será validada (linhas 10 a 33). O bloco de *if*s existe em duplicado (letras maiúsculas e minúsculas), tendo o bloco de letras minúsculas sido suprimido por questões de legibilidade (linha 33). O utilizador é, de seguida, questionado se pretende formatar o dispositivo de armazenamento de USB no formato Extended File Allocation Table (exFAT) (linhas 48-51). Caso aceite, o dispositivo de armazenamento USB será formatado; caso contrário, o *script* tentará mesmo assim fazer o *download* da aplicação e tentará instalá-la no

dispositivo de armazenamento USB (linhas 65-69). Como existem softwares que não podem ser distribuídos, o *script* de instalação faz download desse softwares através dos seus sites originais (linhas 75-85).

Este *script* de instalação é uma das funcionalidades que facilita o dia-a-dia do técnico de investigação de situações de IR. Quando comparada com as demais aplicações referidas no Capítulo 3, constata-se que esta é a única com tal funcionalidade.

5.2 Testes ao protótipo

Neste sub capítulo são demonstrados os principais testes realizados à aplicação. Os testes demonstram a sua execução no sistema operativo Linux, contudo estes foram também repetidos noutros sistemas operativos.

A solução proposta foi primeiramente validada quanto à sua portabilidade e capacidade de auto-execução. Dois dos requisitos identificados na Secção 4.1. Começa-se por verificar se o *script* de instalação faz a instalação correta dos ficheiro no dispositivo de armazenamento USB, verificando assim o pré-requisito da portabilidade. A verificação do pré-requisito de auto-execução será verificado em sistema operativo que permita esta funcionalidade.

```
fabio@fabio-PC ~/Aulas 2015-2016/Tese/scripts meus/scripts instalacao $ sudo sh
scriptInstall.sh
Path pen usb:
/dev/sdd1
Would you like to format this flash drive?
Type y for Yes or n for NO then press Enter
y
scriptInstall.sh: 96: [: ii: unexpected operator
Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Obter:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [94,5 kB]
Ign:3 http://dl.google.com/linux/chrome/deb stable InRelease
Obter:4 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [95,7 kB]
Hit:5 http://dl.google.com/linux/chrome/deb stable Release
Hit:6 http://archive.canonical.com/ubuntu xenial InRelease
Obter:8 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [92,2 kB]
Ign:9 http://packages.linuxmint.com sarah InRelease
Hit:10 http://packages.linuxmint.com sarah Release
Obtidos 282 kB em 24s (11,4 kB/s)
A ler as listas de pacotes... Pronto
A ler as listas de pacotes... Pronto
A construir árvore de dependências
A ler a informação de estado... Pronto
exfat-fuse is already the newest version (1.2.3-1).
exfat-utils is already the newest version (1.2.3-1).
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  g++-5 icoutils libcublas7.5 libcudart7.5 libcufft7.5 libcufftw7.5
  libcuinj64-7.5 libcurand7.5 libcusolver7.5 libcusparse7.5 libnppc7.5
  libnppi7.5 libnpps7.5 libnvblas7.5 libnvrtc7.5 libnvtoolsext1 libnvvm3
  libstdc++-5-dev libthrust-dev nvidia-cuda-dev nvidia-profiler opencl-headers
Utilize 'sudo apt autoremove' para os remover.
0 pacotes actualizados, 0 pacotes novos instalados, 0 a remover e 29 não actuali
zados.
mkexfatfs 1.2.3
Creating... done.
Flushing... done.
File system created successfully.
FUSE exfat 1.2.3
```

Figura 5.1: Teste formatação em Linux

A Figura 5.1 retrata o teste de instalação. Este teste visa garantir que o

script de instalação faz a formatação do dispositivo de armazenamento USB no formato exFAT e que faz de seguida o *download* da aplicação da Internet diretamente para o dispositivo de armazenamento USB. Uma qualidade boa deste *script* é que é executado fora da raiz do dispositivo de armazenamento USB e que quando necessário muda-se para a raiz do dispositivo de armazenamento USB.

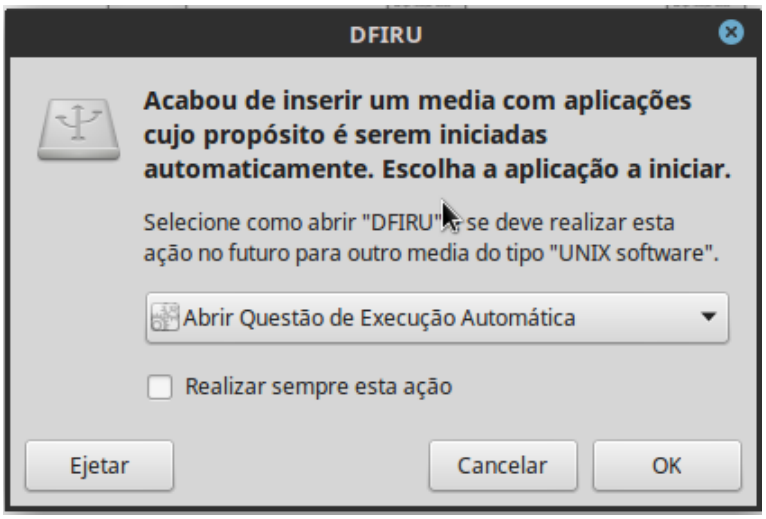

Figura 5.2: Teste de auto-execução em Linux

A Figura 5.2 retrata o teste de verificação da auto-execução. Nem todos os sistemas operativos suportam (ou permitem) esta funcionalidade. A tendência vai no sentido desta funcionalidade deixar de funcionar. O teste realizado demonstra que esta está a funcionar corretamente visto que pede para executar a aplicação que está definida no ficheiro de configurações próprio para o efeito. Como o ficheiro de configurações auto-executável do Linux é diferente do Windows, também foi testado para garantir que a aplicação é auto-executável sempre que possível.

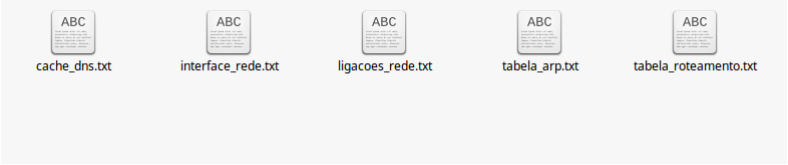

Figura 5.3: Informação recolhida pela aplicação em Linux

A Figura 5.3 mostra o *output* gerado pela aplicação quando recolhe a informação do grupo Rede. Todos os ficheiros foram abertos para garantir que a informação recolhida era a informação que se esperava que a aplicação recolhesse para o grupo Rede. Com este teste garante-se que a aplicação

está a gerar os ficheiros corretamente para o grupo Rede. Também foram realizados testes para os outros grupos e estes também foram analisados para garantir que a aplicação recolhe toda a informação prevista e que esta é homogénea entre os sistemas operativos testados.

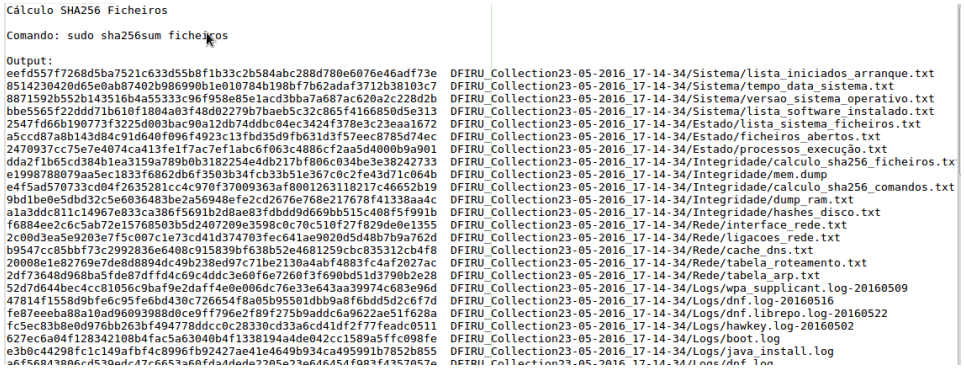

Figura 5.4: Teste para verificar integridade no Linux

A Figura 5.4 retrata o teste realizado para validar que a aplicação gera resumos criptográficos (SHA256, neste caso) para todos os ficheiros criados pela a aplicação. Este teste permite saber se todo a informação foi recolhida e se foi calculado todos os *hashes* para todos os ficheiros gerados pela a aplicação.

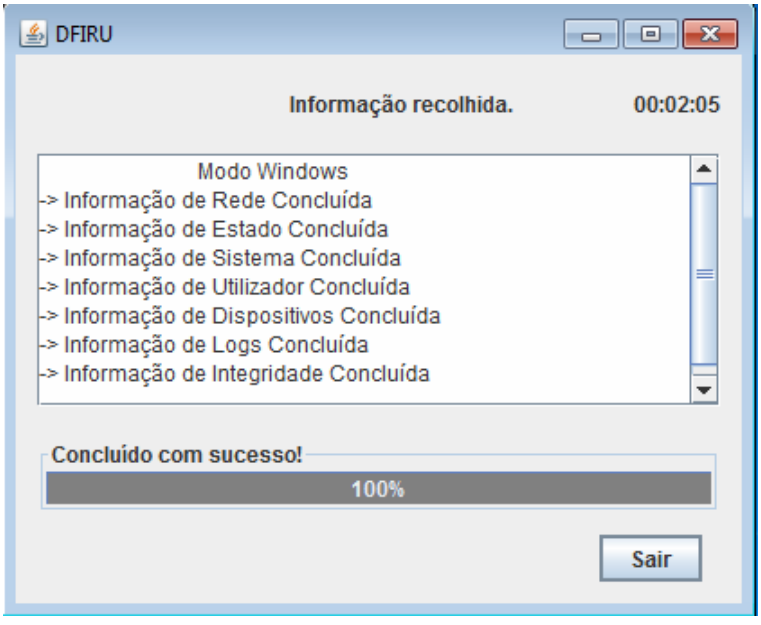

Figura 5.5: Teste de execução no Windows

Na Figura 5.5 pode observar-se a execução da aplicação no sistema ope-

rativo Windows. Neste caso foi executado sem funcionalidades extra e em modo de utilizador comum. Também se observa que a aplicação demora pouco tempo para concluir a sua execução.

O teste de verificação da execução multi-plataforma consistiu na sua execução nos vários sistemas operativos suportados (Linux, Windows e macOS). Os testes foram realizados com sucesso. Testou-se a funcionalidade de calculo de resumos criptográficas de resumo de todos os ficheiros do disco rígido. Foram testados os algoritmos MD5, SHA-1 ou SHA256. Testou-se a funcionalidade de recolha da lista de todos os ficheiros do sistema de ficheiros. Testou-se ainda a funcionalidade de execução como administrador e a funcionalidade de recolha de uma cópia da memória RAM. Todos os testes foram concluídos com sucesso. Estando todas as funcionalidades testadas, passou-se para os testes da aplicação principal, responsável pela interface com o utilizador. Testou o armazenamento para ficheiro das configurações da aplicação, bem como o suporte para múltiplos idiomas.

Tendo os testes de funcionamento da solução proposta sido concluídos com sucesso, efectuaram-se alguns testes de desempenho. Em particular avaliou-se o desempenho das funcionalidades de listagem completa do sistema de ficheiros, de cálculo de resumos criptográficos de todos os ficheiros no disco, de criação de uma cópia da memória RAM. Foi elaborado um *script* para contabilizar o tempo que demorava a funcionalidade do calculo das funções criptográficas de resumo, facilitando a sua repetição com diferentes algoritimos criptográficos. Este *script* é o apresentado na Listagem 5.2

Listagem 5.2: *Script* de contabilização de tempos para cálculo de *hashes*

```
1  #!/bin/bash
2
3
4  #Nomes Ficheiros
5  FICHEIRO_MD5=dados_md5.txt
6  FICHEIRO_SHA1=dados_sha1.txt
7  FICHEIRO_SHA256=dados_sha256.txt
8  FICHEIRO_TEMPOS=tmp.txt
9
10
11 #Cálculo md5 Ficheiros
12
13 printf "MD5" >> $FICHEIRO_TEMPOS
14 (time $(find / -type f -exec md5sum {} \; >> $FICHEIRO_MD5 2> /
       dev/null)) &>> $FICHEIRO_TEMPOS
15
16
17 printf "\nSHA1" >> $FICHEIRO_TEMPOS
18 (time $(find / -type f -exec sha1sum {} \; >> $FICHEIRO_SHA1 2>
       /dev/null)) &>> $FICHEIRO_TEMPOS
19
20
```

```
21 printf "\nSHA256" >> $FICHEIRO_TEMPOS
22 (time $(find / -type f -exec sha256sum {} \; >> $FICHEIRO_SHA256
       2> /dev/null)) &>> $FICHEIRO_TEMPOS
```

O *script* recorre ao comando *time* para obter o tempo execução. O comando *find /* é utilizado para procurar em todas as diretorias dentro da diretoria /. A opção *-type f* limita a procura a ficheiros e com a opção *-exec* indica-se a execução da função de resumo. Neste caso especifico vai variar de acordo com a função criptográfica passado por parâmetro e poderá ser: MD5, SHA-1 ou SHA256. Os resultados vão ser guardados para ficheiro, em de acrescento.

Listagem 5.3: Execução do *script* de contabilização de tempo gasto em cálculo de resumos

```
1  MD5
2
3  real    24m40.618s
4  user    0m30.487s
5  sys     0m59.553s
6
7  SHA1
8
9  real    24m47.773s
10 user    0m31.883s
11 sys     1m3.207s
12
13 SHA256
14
15 real    25m5.241s
16 user    1m9.397s
17 sys     1m5.067s
```

Na listagem 5.3 podem observar-se os tempos que as funções criptográficas de resumo demoraram para calcular os resumos de todos os ficheiros de um disco rígido. Também pode se observar que o tempo entre as funções criptográficas de resumo não variam assim tanto, por isso, conclui-se que a melhor função criptográfica de resumo para ser utilizada por omissão na aplicação é a SHA-256, já que existem registos de problemas com os algoritmos MD5[25] e SHA-1[32]. Tendo por base este teste, optou-se por atribuir, por omissão, um valor máximo de execução do *script* de 60 minutos sempre que o utilizador selecionar esta opção.

Na Figura 5.6 pode observa-se o teste que foi realizado para testar se todos os ficheiros do disco rígido eram listados. Para testar esta funcionalidade só se selecionou as opções lista completa de sistema de ficheiros e executar como Admin. Como observado neste teste o tempo gasto para realizar a funcionalidade lista completa de sistema de ficheiros mais as funcionalidades normais foi apenas 1 minuto e 37 segundos num sistema operativo Linux. É uma funcionalidade que demora pouco tempo.

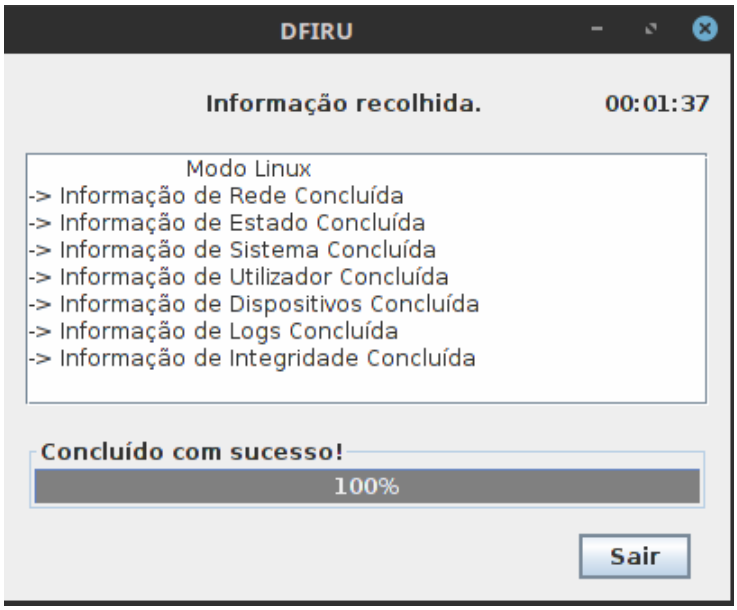

Figura 5.6: Teste da listagem de todos os ficheiros do disco

O teste de duração da operação de listagem de todos os ficheiros do disco demonstrou ser pouco demorada. O valor por omissão a utilizar pela aplicação, como limite máximo de tempo para a execução deste *script*, quando o utilizador a seleciona foi definido em 5 minutos.

Na Figura 5.7 pode observar o teste que foi realizado para testar se fazia a cópia da memória RAM do sistema operativo alvo. Para testar esta funcionalidade selecionou-se a funcionalidade extra de cópia da memória RAM e executar como Admin. A cópia da RAM requer privilégios de administração. Como se observa no teste, o tempo gasto para realizar a funcionalidade cópia da memória RAM e funcionalidades normais foi de apenas 4 minutos e 26 segundos, num sistema operativo Linux com 4 Gigabytes de memória RAM.

O teste de duração da operação de obtenção de uma cópia da memória RAM demonstrou ser algo demorada. O valor por omissão a utilizar pela aplicação, como limite máximo de tempo para a execução deste *script*, quando o utilizador a seleciona foi definido em 60 minutos.

As demais funcionalidades selecionáveis pelo utilizador concluem-se em segundos pelo que o valor por omissão a utilizar pela aplicação como limite máximo de tempo para a execução destas, quando o utilizador as seleciona, foi definido em 5 minutos.

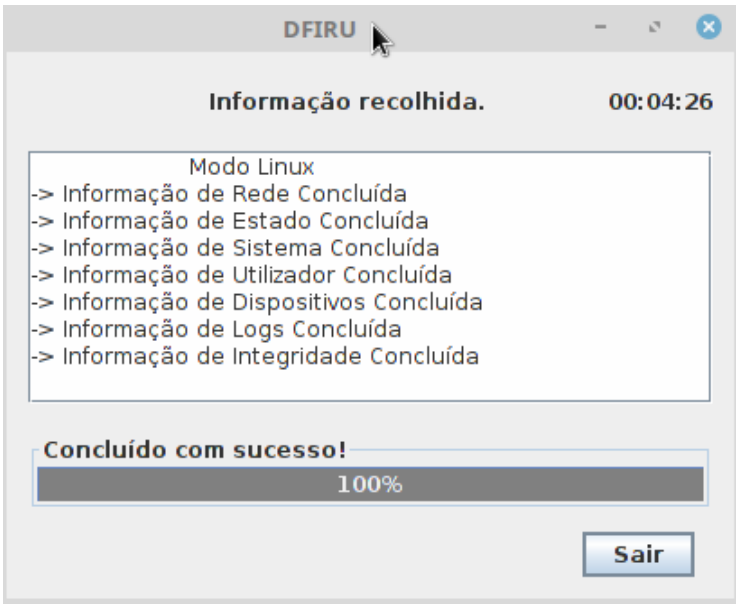

Figura 5.7: Teste da memória RAM

5.3 Conclusão

A solução proposta satisfaz todos os requisitos identificados. Recolhe artefactos e outra informação em diferentes sistemas operativos. Tanto a solução com a informação recolhida são armazenados num suporte de armazenamento portátil. Sempre que permitido pelo sistema operativo do equipamento em análise, a solução proposta faz uso dos mecanismos de auto-execução para automatizar o processo de recolha de artefactos. Recolhe informação de vários grupos, onde se inclui a rede, o estado do sistema operativo, utilizadores, dispositivos e registos de atividade. A solução proposta regista ainda a sua própria atividade e calcula resumos criptográficos de toda a informação recolhida e de todos os ficheiros gerados.

Capítulo 6

Conclusões

Em situações de resposta a incidentes de segurança informática, o analista forense tenta recolher toda a informação possível. A análise de informação de estado surge como um aspeto importante deste tipo de análise forense digital, especialmente quando se tratam de situações de IR que envolvam vários equipamentos ligados em rede. Este tipo de procedimento de análise forense digital é usualmente designado de *live forensics*. A informação recolhida numa análise forense digital *live* não deve ser vista como um substituto daquela que é obtida por uma análise forense tradicional. Deve ser vista como complementar que permitirá recolher informação não disponível de outra forma.

Note-se ainda que a recolha de dados voláteis não está isenta de risco. Uma abordagem poderá ser a de se usar um processo automatizado, previamente conhecido, que permitirá prever o seu impacto na informação de estado. De outra forma, poder-se-á causar alterações excessivas, desnecessárias e não previsíveis para o sistema em análise. Dispositivos de armazenamento *USB* apresentam-se como veículos interessantes para construir mecanismos automatizados para esta recolha de informação, pois permitem armazenar tanto as aplicações necessárias para a recolha da informação como o resultado da recolha, são facilmente transportáveis, podem ter grandes capacidades de armazenamento e facilitam a recolha de forma automática após a sua inserção no PC.

As aplicações existente que possibilitam a recolha de artefactos de sistemas ligados em cenários de IR foram identificadas, analisadas e testadas. Com base nessa analise foi desenvolvida uma tabela de comparação que permitiu concluir que não estão disponíveis soluções que permitam efectuar a recolha de informação de estado num contexto de análises forense que, simultaneamente, seja multi-plataforma, exaustiva na sua recolha de informação de estado, homogénea quanto à qualidade e quantidade de informação recolhida, independentemente do sistema operativo em análise, capaz de garantir a integridade de comandos e da informação recolhida e portável.

A solução proposta, de nome DFIRU, é constituída por uma aplicação principal, desenvolvida na linguagem Java, que executa *scripts* para Windows, Linux e macOS. Por sua vez, estes *scripts* vão executar os comandos que farão a recolha de informação no sistema alvo. A aplicação pode ser executada em modo gráfico e em modo consola. A informação recolhida quando executada em diferentes sistemas operativos (Linux, Windows e macOS) é homogénea. Foram desenvolvidos testes garantindo que a aplicação desenvolvida funciona de acordo com o esperado.

6.1 Resultados relevantes

A solução proposta consiste numa aplicação desenvolvida em Java que recolhe informação de estado tendo em vista tanto o alcance como a homogeneidade da informação recolhida. A solução proposta recolhe artefactos e outra informação em diferentes sistemas operativos. Tanto a solução como a informação recolhida são armazenados num suporte de armazenamento portátil. Sempre que permitido pelo sistema operativo do equipamento em análise, a solução proposta faz uso dos mecanismos de auto-execução para automatizar o processo de recolha de artefactos. A solução proposta regista a sua própria atividade e calcula resumos criptográficos de toda a informação recolhida e de todos os ficheiros gerados.

6.2 Trabalho Futuro

A título de próximos passos, o presente trabalho será publicado como *open source*. Tal permitirá que esta possa ganhar novos idiomas e novas atualizações, quer através do *feedback* de novos utilizadores, quer pela inclusão de desenvolvimentos da comunidade *open source*.

Tentar-se-á ainda que a utilização da aplicação seja monitorizada afim de se verificar da sua aplicabilidade. Tal monitorização assentará na solicitação de *feedback* a utilizadores específicos que façam uso da ferramenta no terreno. Tendo por base este *feedback*, poder-se-á evoluir a versão atual ou criar versões mais específicas, conforme o caso.

Bibliografia

[1] J. Jansen. Use of sha-2 algorithms with rsa in dnskey and rrsig resource records for dnssec @ONLINE, October 2009. https://tools.ietf.org/html/rfc5702 Accessed: 2016-01-03.

[2] Marcos Elias Picão. Prompt de comando do windows @ONLINE, August 2007. http://www.miqueiasreale.com.br/wp-content/uploads/2010/08/revistagdh_07.pdf Accessed: 2016-10-6.

[3] Alessandro Vivas Andrade, Leonardo Carneiro de Araújo, Cristiano Grijó Pitangui, and Luciana Pereira de Assis. Linux: Comandos básicos e avançados, November 2015. http://www.andarilho.pro.br/docs/Linux.pdf Accessed: 2016-10-6.

[4] Alessandro Vivas Andrade, Luciana P. Assis, and André L. Maravilha. Mac os x: comandos básicos e avançados, 2016. http://www.andarilho.pro.br/docs/Mac.pdf Accessed: 2016-10-6.

[5] Marwan Al-Zarouni and Haitham Al-Hajri. A proof-of-concept project for utilizing u3 technology in incident response. In *Australian Digital Forensics Conference*, page 15, 2007.

[6] Jason T. Luttgens, Kevin Mandia, and Mathew Pepe. *Incident response & computer forensics*. McGraw-Hill Education, 2014.

[7] Todd G Shipley, CFCE CFE, and Henry R Reeve. Collecting evidence from a running computer. *SEARCH, The National Consortium for Justice and Internationals Standards*, page 6, 2006.

[8] André Zúquete. *Segurança em Redes Informáticas*. FCA - Editora de Informática, Lda., April 2013.

[9] Jerônimo C. Pellefrini. Introdução à criptografia e seus fundamentos notas de aula - versão 90 @ONLINE, February 2016. http://aleph0.info/cursos/ic/notas/cripto.pdf Accessed: 2016-10-6.

[10] Fábio Freitas. Relatório de projeto final da licenciatura em engenharia informática, June 2014.

[11] W. Stallings. *Criptografia e segurança de redes Princípios e práticas.* São Paulo, Pearson Prentice Hall, 2008.

[12] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication, February 1997. https://tools.ietf.org/html/rfc2104 Accessed: 2016-10-6.

[13] P. Cheng and R. Glenn. Test cases for hmac-md5 and hmac-sha-1 @ONLINE, September 1997. https://tools.ietf.org/html/rfc2202 Accessed: 2016-01-03.

[14] R. Rivest. The md5 message-digest algorithm @ONLINE, April 1992. https://tools.ietf.org/html/rfc1321 Accessed: 2016-01-03.

[15] S. Turner and L. Chen. Updated security considerations for the md5 message-digest and the hmac-md5 algorithms @ONLINE, March 2011. https://tools.ietf.org/html/rfc6151#ref-MD5 Accessed: 2016-01-03.

[16] National Institute of Standards and Technology (NIST). *Recommendation for Key Management - Part 1 (Revised).* Special Publication 800-57, March 2007.

[17] National Institute of Standards and Technology (NIST). *DRAFT Recommendation for the Transitioning of Cryptographic Algorithms and Key Sizes.* Special Publication 800-131, June 2010.

[18] M. Nystrom. Identifiers and test vectors for hmac-sha-224, hmac-sha-256, hmac-sha-384, and hmac-sha-512 @ONLINE, December 2005. https://tools.ietf.org/html/rfc4231 Accessed: 2016-01-03.

[19] JH. Song, R. Poovendran, J. Lee, and T. Iwata. The aes-cmac algorithm @ONLINE, June 2006. https://tools.ietf.org/html/rfc4493 Accessed: 2016-01-03.

[20] B. den Boer and A. Bosselaers. Collisions for the compression function of md5. *Eurocrypt*, 1993.

[21] H. Dobbertin. Cryptanalysis of md5 compress. *Eurocrypt*, 1996.

[22] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for hash functions md4, md5, haval-128 and ripemd @ONLINE, 2004. http://eprint.iacr.org/2004/199.pdf Accessed: 2016-01-03.

[23] X. Wang and H. Yu. How to break md5 and other hash functions. In *Advances in Cryptology - EUROCRYPT2005*, Lecture Notes in Computer Science 3494. Springer, 2005.

[24] V. Klima. Tunnels in hash functions: Md5 collisions within a minute @ONLINE. Cryptology ePrint Archive, Report 2006/105, 2006. http://eprint.iacr.org/2004/199.pdf Accessed: 2016-01-03.

[25] M.J. Stevens. On collisions for md5. Master's thesis, Eindhoven University of Technology, June 2007. http://www.win.tue.nl/hashclash/On%20Collisions%20for%20MD5%20-%20M.M.J.%20Stevens.pdf Accessed: 2016-10-6.

[26] Arjen Lenstra, Xiaoyun Wang, and Benne de Weger. Colliding x.509 certificates. Cryptology ePrint Archive, Report 2005/067, 2005. http://eprint.iacr.org/2005/067 Accessed: 2016-01-03.

[27] M. Stevens, A. Lenstra, and B. de Weger. Chosen-prefix collisions for md5 and colliding x.509 certificates for different identities. *EuroCrypt*, 2007.

[28] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. Osvik, and B. de Weger. Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. *Crypto*, 2009.

[29] M. Stevens, A. Lenstra, and B. de Weger. Chosen-prefix collisions for md5 and applications. *Journal of Cryptology*, 2009.

[30] NIST. *Secure hash standard*. Federal Information Processing Standard, FIPS-180-1, April 1995.

[31] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1 @ONLINE. http://gauss.ececs.uc.edu/Courses/c6053/lectures/Hashing/sha1-crypto-auth-new-2-yao.pdf Accessed: 2016-01-03.

[32] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart collision for full sha-1 @ONLINE. https://marc-stevens.nl/research/papers/KPS_freestart80.pdf Accessed: 2016-10-6.

[33] Florent Chabaud and Antoine Joux. Differential collisions in sha-0. In *CRYPTO (Hugo Krawczyk, ed.)*, volume 1462 of *Lecture Notes in Computer Science*, page 56–71. Springer, 1998.

[34] Christophe De Cannière and Christian Rechberger. Finding sha-1 characteristics: General results and applications. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, page 1–20. Springer, 2006.

[35] Marc Stevens. Project hashclash. https://marc-stevens.nl/p/hashclash/index.php Accessed: 2016-01-03.

[36] Marc Stevens. *Attacks on Hash Functions and Applications*. PhD thesis, Leiden University, June 2012.

[37] National Institute of Standards and Technology. Sha-3 selection announcement, October 2012. http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf Accessed: 2016-01-03.

[38] D. J. Bernstein and T. Lange. ebash: Ecrypt benchmarking of all submitted hashes, January 2011. http://bench.cr.yp.to/ebash.html Accessed: 2016-01-03.

[39] J. W. Shay Gueron and Simon Johnson. Sha-512/256. Technical Report Report 2010/548, Cryptology ePrint Archive, 2010.

[40] National Institute of Standards and Technology. Fips pub 180-4: Secure hash standard, August 2015. http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf Accessed: 2016-01-03.

[41] Maria Eichlseder, Florian Mendel, and Martin Schläffer. Branching heuristics in differential collision search with applications to sha-512. *IACR*, 2014.

[42] K. Aoki, J. Guo, K. Matusiewicz, Y. Sasaki, and L. Wang. Preimages for step-reduced sha-2. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, page 578–597. Springer, 2009.

[43] D. Khovratovicha, C. Rechberger, and A. Savelieva. Bicliques for preimages: Attacks on skein-512 and the sha-2 family. In A. Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, page 244–263. Springer, 2012.

[44] J. Li, T. Isobe, and K. Shibutani. Converting meet-in-the-middle preimage attack into pseudo collision attack: Application to sha-2. In A. Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, pages 264–286. Springer, 2012.

[45] Ryan Shipp. ir-triage-toolkit @ONLINE, July 2013. https://github.com/rshipp/ir-triage-toolkit Accessed: 2016-01-03.

[46] Adam Compton Adrian Sanabria and Bill Dean. Free volatile data collection kit@ONLINE, September 2010. https://www.swordshield.com/2010/09/free-volatile-data-collection-kit/ Accessed: 2016-01-03.

[47] Corey Harrell. jiir updates @ONLINE, December 2011. http://journeyintoir.blogspot.pt/2011/12/jiir-updates.html Accessed: 2016-01-03.

[48] Kevin. Tr3secure@ONLINE, October 2014. https://github.com/ktneely/Tr3Secure Accessed: 2016-01-03.

[49] Brian Moran. Live response collection@ONLINE, September 2015. https://www.cybercompex.org/topic/live-response-collection Accessed: 2016-01-10.

[50] Erik Musick. Live response scripts@ONLINE, July 2011. http://erikmusick.com/live-response-scripts/ Accessed: 2016-01-10.

[51] Inc. ADF Solutions. Quickly and easily scan computers for cyber-related crimes@ONLINE, 2016. http://www.adfsolutions.com/products/triage-responder Accessed: 2016-01-03.

[52] D. Eastlake and P. Jones. Us secure hash algorithm 1 (sha1) @ONLINE, September 2001. https://tools.ietf.org/html/rfc3174 Accessed: 2016-01-03.

Apêndice A

Scripts Desenvolvidos

A.1 *Scripts* de Instalação

Listagem A.1: *Script* de instalação Linux

```
1  #!/bin/bash
2
3  SITE=site
4  PASTA=dfiru.zip
5  MONTADO=/mnt/exfat
6  NAME_PEN=DFIRU
7  DISTRO=`uname`
8  CONTINUAR=no
9  DISTRO_LINUX="lsb_release -dr"
10
11
12 if [ "$DISTRO" = "Darwin" ]; then
13 diskutil list
14 echo "Enter the IDENTIFIER of Disk USB?"
15 read CAMINHO
16
17 else
18 echo "Path pen usb:"
19 read CAMINHO
20 fi
21
22
23 until [ "$CONTINUAR" = "yes" ];
24 do
25 echo "Would you like to format this flash drive?"
26 echo "Type y for Yes or n for NO then press Enter"
27 read RESPOSTA
28
29 if [ "$RESPOSTA" = "y" ] || [ "$RESPOSTA" = "n" ]; then
30 CONTINUAR=yes
31 fi
32 done
33
```

```
34
35 if [ "$RESPOSTA" = "y" ]; then
36 #echo "Desmonte a PEN"
37 sudo umount $CAMINHO
38
39 if [ "$DISTRO" = "Linux" ]; then
40 if  which rpm >/dev/null 2>/dev/null; then
41 INSTALADO_FUSE_RPM=`rpm -qa | grep fuse-exfat`
42 INSTALADO_EXFAT_RPM=`rpm -qa | grep exfat-utils`
43
44
45 if [ $INSTALADO_EXFAT_RPM >/dev/null ] && [ $INSTALADO_FUSE_RPM
       >/dev/null ]; then
46 echo "Installed dependencies!";
47 else
48 if $DISTRO_LINUX | grep -i "opensuse" >> /dev/null; then
49 sudo zypper update
50 sudo zypper install fuse-exfat exfat-utils
51
52 elif $DISTRO_LINUX | grep -i "mageia" >> /dev/null; then
53 sudo urpmi.update -a
54 sudo urpmi exfat-utils
55
56 elif $DISTRO_LINUX | grep -i "korora" >> /dev/null; then
57 sudo dnf update
58 sudo dnf install fuse-exfat exfat-utils
59
60 elif $DISTRO_LINUX | grep -i "fedora" >> /dev/null; then
61 su -c 'dnf install http://download1.rpmfusion.org/free/fedora/
       rpmfusion-free-release-$(rpm -E %fedora).noarch.rpm http://
       download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-
       release-$(rpm -E %fedora).noarch.rpm'
62 sudo dnf update
63 sudo dnf install fuse-exfat exfat-utils
64
65 elif $DISTRO_LINUX | grep -i "centos" >> /dev/null; then
66 versao=`uname -r`
67 arquitetura=`uname -i`
68
69
70 if echo "$versao" | egrep 'el6' >/dev/null; then
71 sudo yum update
72
73 if [ "$arquitetura" = "x86_64" ]; then
74 sudo yum -y install epel-release && rpm -Uvh http://li.nux.ro/
       download/nux/dextop/el6/x86_64/nux-dextop-release-0-2.el6.nux
       .noarch.rpm
75 else
76 sudo yum -y install epel-release && rpm -Uvh http://li.nux.ro/
       download/nux/dextop/el6/i386/nux-dextop-release-0-3.el6.nux.
       noarch.rpm
77 fi
78 fi
79
```

```
80
81  if echo "$versao" | egrep 'el7' >/dev/null; then
82  sudo yum update
83  sudo yum -y install epel-release && rpm -Uvh http://li.nux.ro/
        download/nux/dextop/el7/x86_64/nux-dextop-release-0-5.el7.nux
        .noarch.rpm
84
85  fi
86
87  sudo yum install exfat-utils fuse-exfat
88  fi
89  fi
90
91  elif  which dpkg >/dev/null 2>/dev/null ; then
92  INSTALADO_FUSE_DEB=`dpkg -l | grep exfat-fuse`
93  INSTALADO_EXFAT_DEB=`dpkg -l | grep exfat-utils`
94
95
96  if [ $INSTALADO_EXFAT_DEB >/dev/null ] && [ $INSTALADO_FUSE_DEB
        >/dev/null ]; then
97  echo "Installed dependencies!";
98  else
99  sudo apt-get update
100 sudo apt-get install exfat-fuse exfat-utils
101 fi
102
103 elif which pacman >/dev/null 2>/dev/null ; then
104 INSTALADO_EXFAT_PACMAN=`pacman -Q | grep exfat-utils`
105
106
107 if [ $INSTALADO_EXFAT_PACMAN >/dev/null ]; then
108 echo "Installed dependencies!";
109 else
110 sudo sh -c 'echo -e "[community]\nInclude = /etc/pacman.d/
        mirrorlist" >> /etc/pacman.conf'
111 sudo pacman -Syu
112 sudo pacman -Sy exfat-utils
113 fi
114
115 elif which slpkg >/dev/null 2>/dev/null ; then
116 INSTALADO_FUSE_SLACK=`ls -l /var/log/packages | grep fuse-exfat`
117 INSTALADO_EXFAT_SLACK=`ls -l /var/log/packages | grep exfat-
        utils`
118
119
120 if [ $INSTALADO_EXFAT_SLACK >/dev/null ] && [
        $INSTALADO_FUSE_SLACK >/dev/null ]; then
121 echo "Installed dependencies!";
122 else
123 wget https://github.com/relan/exfat/releases/download/v1.2.1/
        fuse-exfat-1.2.1.tar.gz
124 sudo installpkg fuse-exfat-1.2.1.tar.gz
125 sudo rm fuse-exfat-1.2.1.tar.gz
126
```

```
127 wget https://github.com/relan/exfat/releases/download/v1.2.1/
        exfat-utils-1.2.1.tar.gz
128 sudo installpkg exfat-utils-1.2.1.tar.gz
129 sudo rm exfat-utils-1.2.1.tar.gz
130 fi
131
132
133 else
134 echo "Linux don't support";
135 fi
136
137 sudo mkfs.exfat -n $NAME_PEN $CAMINHO
138
139 if [ ! -d $MONTADO ]; then
140 sudo mkdir -p $MONTADO
141 fi
142
143 sudo mount $CAMINHO $MONTADO
144
145 wget $SITE
146 mv $PASTA $MONTADO
147 cd $MONTADO
148 unzip $PASTA
149 rm $PASTA
150 sudo umount $MONTADO
151 else
152 if [ "$DISTRO" = "Darwin" ]; then
153 diskutil unmount $CAMINHO
154 diskutil eraseDisk ExFAT $NAME_PEN MBRFormat $CAMINHO
155 diskutil mount $CAMINHO
156
157 wget $SITE
158 mv $PASTA /Volumes/$NAME_PEN
159 cd /Volumes/$NAME_PEN
160 unzip $PASTA
161 rm $PASTA
162 fi
163 fi
164
165 else
166 if [ "$DISTRO" = "Linux" ]; then
167
168 if [ ! -d $MONTADO ]; then
169 sudo mkdir -p $MONTADO
170 fi
171
172 sudo mount $CAMINHO $MONTADO
173
174 wget $SITE
175 mv $PASTA $MONTADO
176 cd $MONTADO
177 unzip $PASTA
178 rm $PASTA
179 sudo umount $MONTADO
```

```
180 else
181 if [ "$DISTRO" = "Darwin" ]; then
182 wget $SITE
183 NAME=`diskutil info $CAMINHO | grep "Volume Name:" | awk '{print
      $3}'`
184 mv $PASTA /Volumes/$NAME
185 cd /Volumes/$NAME_PEN
186 unzip $PASTA
187 rm $PASTA
188 fi
189 fi
190
191 fi
```

Listagem A.2: *Script* de instalação Windows

```
1  @echo off
2
3  set SITE=site
4  set PASTA=dfiru.zip
5  set NOME_PEN=DFIRU
6
7  :start
8  set /p drive=Enter the drive letter to be formatted:
9
10 if '%drive%' == 'D' goto proceed
11 if '%drive%' == 'E' goto proceed
12 if '%drive%' == 'F' goto proceed
13 if '%drive%' == 'G' goto proceed
14 if '%drive%' == 'H' goto proceed
15 if '%drive%' == 'I' goto proceed
16 if '%drive%' == 'J' goto proceed
17 if '%drive%' == 'K' goto proceed
18 if '%drive%' == 'L' goto proceed
19 if '%drive%' == 'M' goto proceed
20 if '%drive%' == 'N' goto proceed
21 if '%drive%' == 'O' goto proceed
22 if '%drive%' == 'P' goto proceed
23 if '%drive%' == 'Q' goto proceed
24 if '%drive%' == 'R' goto proceed
25 if '%drive%' == 'S' goto proceed
26 if '%drive%' == 'T' goto proceed
27 if '%drive%' == 'U' goto proceed
28 if '%drive%' == 'V' goto proceed
29 if '%drive%' == 'W' goto proceed
30 if '%drive%' == 'X' goto proceed
31 if '%drive%' == 'Y' goto proceed
32 if '%drive%' == 'Z' goto proceed
33 if '%drive%' == 'd' goto proceed
34 if '%drive%' == 'e' goto proceed
35 if '%drive%' == 'f' goto proceed
36 if '%drive%' == 'g' goto proceed
37 if '%drive%' == 'h' goto proceed
38 if '%drive%' == 'i' goto proceed
```

```
39 if '%drive%' == 'j' goto proceed
40 if '%drive%' == 'k' goto proceed
41 if '%drive%' == 'l' goto proceed
42 if '%drive%' == 'm' goto proceed
43 if '%drive%' == 'n' goto proceed
44 if '%drive%' == 'o' goto proceed
45 if '%drive%' == 'p' goto proceed
46 if '%drive%' == 'q' goto proceed
47 if '%drive%' == 'r' goto proceed
48 if '%drive%' == 's' goto proceed
49 if '%drive%' == 't' goto proceed
50 if '%drive%' == 'u' goto proceed
51 if '%drive%' == 'v' goto proceed
52 if '%drive%' == 'w' goto proceed
53 if '%drive%' == 'x' goto proceed
54 if '%drive%' == 'y' goto proceed
55 if '%drive%' == 'z' goto proceed
56 cls
57 goto error
58
59 :error
60 echo.
61 echo The drive letter you entered was not recognized
62 echo.
63 pause
64 cls
65 goto start
66
67 :proceed
68 cls
69 echo.
70 echo Would you like to format this flash drive?
71 echo.
72 echo Type Y for Yes or N for NO then press Enter
73 set /p ok=
74 if '%ok%' == 'y' goto yes
75 if '%ok%' == 'Y' goto yes
76 if '%ok%' == 'n' goto no
77 if '%ok%' == 'N' goto no
78 cls
79
80 :yes
81 cls
82
83 format %drive%: /fs:EXFAT /v:%NOME_PEN%
84 goto no
85
86 :no
87 wget.exe %SITE%
88 move %PASTA% "%drive%:\"
89 xcopy unzip.exe %drive%:\
90 xcopy wget.exe %drive%:\
91 cd %drive%:\
92 unzip.exe %PASTA%
```

```
93  del %PASTA%
94
95  cd tools\windows
96
97  :: logonsessions download
98  %drive%:\wget.exe https://download.sysinternals.com/files/
        logonSessions.zip
99  %drive%:\unzip.exe logonSessions.zip
100
101 :: pslist download
102 %drive%:\wget.exe https://download.sysinternals.com/files/
        PSTools.zip
103 %drive%:\unzip.exe PSTools.zip
104
105 :: psloglist
106 %drive%:\wget.exe https://download.sysinternals.com/files/
        PSTools.zip
107 %drive%:\unzip.exe PSTools.zip
108
109 del %drive%:\wget.exe
110 del %drive%:\unzip.exe
```

A.2 *Scripts* Inicial

Listagem A.3: *Script* Inicial Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  REDE=$PCNAME/Rede
7  ESTADO=$PCNAME/Estado
8  SISTEMA=$PCNAME/Sistema
9  UTILIZADOR=$PCNAME/Utilizador
10 DISPOSITIVOS=$PCNAME/Dispositivos
11 INTEGRIDADE=$PCNAME/Integridade
12 LOGS=$PCNAME/Logs
13
14
15 #criar pastas
16 mkdir -p $PCNAME
17 mkdir -p $REDE
18 mkdir -p $ESTADO
19 mkdir -p $SISTEMA
20 mkdir -p $UTILIZADOR
21 mkdir -p $DISPOSITIVOS
22 mkdir -p $INTEGRIDADE
23 mkdir -p $LOGS
24
25 echo concluido
```

Listagem A.4: *Script* Inicial macOS

```
#!/bin/bash


#Nome Pastas e caminhos
PCNAME=$1
REDE=$PCNAME/Rede
ESTADO=$PCNAME/Estado
SISTEMA=$PCNAME/Sistema
UTILIZADOR=$PCNAME/Utilizador
DISPOSITIVOS=$PCNAME/Dispositivos
INTEGRIDADE=$PCNAME/Integridade
LOGS=$PCNAME/Logs



#criar pastas
mkdir -p $PCNAME
mkdir -p $REDE
mkdir -p $ESTADO
mkdir -p $SISTEMA
mkdir -p $UTILIZADOR
mkdir -p $DISPOSITIVOS
mkdir -p $INTEGRIDADE
mkdir -p $LOGS

echo concluido
```

Listagem A.5: *Script* Inicial Windows

```
echo off


::Nome Pastas e caminhos
set PCNAME=%1
set PASS=%2
set REDE=%PCNAME%\Rede
set ESTADO=%PCNAME%\Estado
set SISTEMA=%PCNAME%\Sistema
set UTILIZADOR=%PCNAME%\Utilizador
set DISPOSITIVOS=%PCNAME%\Dispositivos
set INTEGRIDADE=%PCNAME%\Integridade
set LOGS=%PCNAME%\Logs
set TOOLS=tools\windows
set CAMINHO=nomePasta.txt


::criar pastas
mkdir %PCNAME%
mkdir %REDE%
mkdir %ESTADO%
mkdir %SISTEMA%
mkdir %UTILIZADOR%
mkdir %DISPOSITIVOS%
```

```
25 mkdir %INTEGRIDADE%
26 mkdir %LOGS%
27
28
29 mkdir %REDE%
30 mkdir %ESTADO%
31 mkdir %SISTEMA%
32 mkdir %UTILIZADOR%
33 mkdir %DISPOSITIVOS%
34 mkdir %INTEGRIDADE%
35 mkdir %LOGS%
36
37 echo concluido
```

A.3 *Scritps* de Rede

Listagem A.6: *Script* de Rede Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  REDE=$PCNAME/Rede
8
9
10 #Nomes Ficheiros
11 INTERFACE_REDE=$REDE/interface_rede.txt
12 TABELA_ROTEAMENTO=$REDE/tabela_roteamento.txt
13 TABELA_ARP=$REDE/tabela_arp.txt
14 CACHE_DNS=$REDE/cache_dns.txt
15 LiGACOES_REDE=$REDE/ligacoes_rede.txt
16
17
18
19 ###REDE
20
21 #Interface Rede
22 printf "Interface de Rede\n\n" >> $INTERFACE_REDE
23 printf "Comando: ifconfig -a\n" >> $CALCULO_SHA256_COMANDOS
24 printf "\nOutput:\n" >> $INTERFACE_REDE
25 ifconfig -a >> $INTERFACE_REDE 2> /dev/null
26 #echo "-> Interface de Rede - Concluído"
27 echo concluido
28
29
30 #Tabela Roteamento
31 printf "Tabela de Roteamento\n\n" >> $TABELA_ROTEAMENTO
32 printf "Comando: netstat -rn\n" >> $TABELA_ROTEAMENTO
33 printf "\nOutput:\n" >> $TABELA_ROTEAMENTO
34 netstat -rn >> $TABELA_ROTEAMENTO 2> /dev/null
```

```
35 #echo "-> Tabela de Roteamento - Concluído"
36 echo concluido
37
38
39 #Tabela ARP
40 printf "Tabela Arp\n\n" >> $TABELA_ARP
41 printf "Comando: arp -a\n" >> $TABELA_ARP
42 printf "\nOutput:\n" >> $TABELA_ARP
43 arp -a >> $TABELA_ARP 2> /dev/null
44 #echo "-> Tabela Arp - Concluído"
45 echo concluido
46
47
48 #Cache DNS
49 #echo "-> Cache DNS - Concluído"
50 echo concluido
51
52
53 #Ligações de Rede
54 printf "Ligações de Rede\n\n" >> $LiGACOES_REDE
55 printf "Comando: netstat -anp\n" >> $LiGACOES_REDE
56 printf "\nOutput:\n" >> $LiGACOES_REDE
57 netstat -anp >> $LiGACOES_REDE 2> /dev/null
58 #echo "-> Ligações de Rede - Concluído"
59 echo concluido
```

Listagem A.7: *Script* de Rede Admin Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  REDE=$PCNAME/Rede
8
9
10 #Nomes Ficheiros
11 INTERFACE_REDE=$REDE/interface_rede.txt
12 TABELA_ROTEAMENTO=$REDE/tabela_roteamento.txt
13 TABELA_ARP=$REDE/tabela_arp.txt
14 CACHE_DNS=$REDE/cache_dns.txt
15 LiGACOES_REDE=$REDE/ligacoes_rede.txt
16
17
18
19 ###REDE
20
21 #Interface Rede
22 printf "Interface de Rede\n\n" >> $INTERFACE_REDE
23 printf "Comando: sudo ifconfig -a\n" >> $INTERFACE_REDE
24 printf "\nOutput:\n" >> $INTERFACE_REDE
25 echo $PASS | sudo -S ifconfig -a >> $INTERFACE_REDE
26 #echo "-> Interface de Rede - Concluído"
```

```
27 echo concluido
28
29
30 #Tabela Roteamento
31 printf "Tabela de Roteamento\n\n" >> $TABELA_ROTEAMENTO
32 printf "Comando: sudo netstat -rn\n" >> $TABELA_ROTEAMENTO
33 printf "\nOutput:\n" >> $TABELA_ROTEAMENTO
34 echo $PASS | sudo -S netstat -rn >> $TABELA_ROTEAMENTO
35 #echo "-> Tabela de Roteamento - Concluído"
36 echo concluido
37
38
39 #Tabela ARP
40 printf "Tabela Arp\n\n" >> $TABELA_ARP
41 printf "Comando: sudo arp -a\n" >> $TABELA_ARP
42 printf "\nOutput:\n" >> $TABELA_ARP
43 echo $PASS | sudo -S arp -a >> $TABELA_ARP
44 #echo "-> Tabela Arp - Concluído"
45 echo concluido
46
47
48 #Cache DNS
49 #echo "-> Cache DNS - Concluído"
50 echo concluido
51
52
53 #Ligações de Rede
54 printf "Ligações de Rede\n\n" >> $LiGACOES_REDE
55 printf "Comando: sudo netstat -anp\n" >> $LiGACOES_REDE
56 printf "\nOutput:\n" >> $LiGACOES_REDE
57 echo $PASS | sudo -S netstat -anp >> $LiGACOES_REDE
58 #echo "-> Ligações de Rede - Concluído"
59 echo concluido
```

Listagem A.8: *Script* de Rede macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  REDE=$PCNAME/Rede
8
9
10 #Nomes Ficheiros
11 INTERFACE_REDE=$REDE/interface_rede.txt
12 TABELA_ROTEAMENTO=$REDE/tabela_roteamento.txt
13 TABELA_ARP=$REDE/tabela_arp.txt
14 CACHE_DNS=$REDE/cache_dns.txt
15 LiGACOES_REDE=$REDE/ligacoes_rede.txt
16
17
18
```

```
19 ###REDE
20
21 #Interface Rede
22 printf "Interface de Rede\n\n" >> $INTERFACE_REDE
23 printf "Comando: ifconfig -a\n" >> $INTERFACE_REDE
24 printf "\nOutput:\n" >> $INTERFACE_REDE
25 ifconfig -a >> $INTERFACE_REDE
26 #echo "-> Interface de Rede - Concluído"
27 echo concluido
28
29
30 #Tabela Roteamento
31 printf "Tabela de Roteamento\n\n" >> $TABELA_ROTEAMENTO
32 printf "Comando: netstat -rn\n" >> $TABELA_ROTEAMENTO
33 printf "\nOutput:\n" >> $TABELA_ROTEAMENTO
34 netstat -rn >> $TABELA_ROTEAMENTO
35 #echo "-> Tabela de Roteamento - Concluído"
36 echo concluido
37
38
39 #Tabela ARP
40 printf "Tabela Arp\n\n" >> $TABELA_ARP
41 printf "Comando: arp -a\n" >> $TABELA_ARP
42 printf "\nOutput:\n" >> $TABELA_ARP
43 arp -a >> $TABELA_ARP
44 #echo "-> Tabela Arp - Concluído"
45 echo concluido
46
47
48 #Cache DNS
49 printf "Cache DNS\n\n" >> $CACHE_DNS
50 printf "Comando: sudo killall -INFO mDNSResponder\n" >>
       $CACHE_DNS
51 printf "\nOutput:\n" >> $CACHE_DNS
52 printf "Só disponível como root!\n"
53 #echo "-> Cache DNS - Concluído"
54 echo concluido
55
56
57 #Ligações de Rede
58 printf "Ligações de Rede\n\n" >> $LiGACOES_REDE
59 printf "Comando: netstat\n" >> $LiGACOES_REDE
60 printf "\nOutput:\n" >> $LiGACOES_REDE
61 netstat >> $LiGACOES_REDE
62 #echo "-> Ligações de Rede - Concluído"
63 echo concluido
```

Listagem A.9: *Script* de Rede Admin macOS

```
1 #!/bin/bash
2
3
4 #Nome Pastas e caminhos
5 PCNAME=$1
```

```
6  PASS=$2
7  REDE=$PCNAME/Rede
8
9
10 #Nomes Ficheiros
11 INTERFACE_REDE=$REDE/interface_rede.txt
12 TABELA_ROTEAMENTO=$REDE/tabela_roteamento.txt
13 TABELA_ARP=$REDE/tabela_arp.txt
14 CACHE_DNS=$REDE/cache_dns.txt
15 LiGACOES_REDE=$REDE/ligacoes_rede.txt
16
17
18
19 ###REDE
20
21 #Interface Rede
22 printf "Interface de Rede\n\n" >> $INTERFACE_REDE
23 printf "Comando: sudo ifconfig -a\n" >> $INTERFACE_REDE
24 printf "\nOutput:\n" >> $INTERFACE_REDE
25 echo $PASS | sudo -S ifconfig -a >> $INTERFACE_REDE
26 #echo "-> Interface de Rede - Concluído"
27 echo concluido
28
29
30 #Tabela Roteamento
31 printf "Tabela de Roteamento\n\n" >> $TABELA_ROTEAMENTO
32 printf "Comando: sudo netstat -rn\n" >> $TABELA_ROTEAMENTO
33 printf "\nOutput:\n" >> $TABELA_ROTEAMENTO
34 echo $PASS | sudo -S netstat -rn >> $TABELA_ROTEAMENTO
35 #echo "-> Tabela de Roteamento - Concluído"
36 echo concluido
37
38
39 #Tabela ARP
40 printf "Tabela Arp\n\n" >> $TABELA_ARP
41 printf "Comando: sudo arp -a\n" >> $TABELA_ARP
42 printf "\nOutput:\n" >> $TABELA_ARP
43 echo $PASS | sudo -S arp -a >> $TABELA_ARP
44 #echo "-> Tabela Arp - Concluído"
45 echo concluido
46
47
48 #Cache DNS
49 printf "Cache DNS\n\n" >> $CACHE_DNS
50 printf "Comando: sudo killall -INFO mDNSResponder\n" >>
       $CACHE_DNS
51 printf "\nOutput:\n" >> $CACHE_DNS
52 echo $PASS | sudo -S killall -INFO mDNSResponder
53 echo $PASS | sudo -S cat /var/log/system.log | grep "mDNS.*Addr"
       >> $CACHE_DNS
54 #echo "-> Cache DNS - Concluído"
55 echo concluido
56
57
```

```
58 #Ligações de Rede
59 printf "Ligações de Rede\n\n" >> $LiGACOES_REDE
60 printf "Comando: sudo netstat\n" >> $LiGACOES_REDE
61 printf "\nOutput:\n" >> $LiGACOES_REDE
62 echo $PASS | sudo -S netstat >> $LiGACOES_REDE
63 #echo "-> Ligações de Rede - Concluído"
64 echo concluido
```

Listagem A.10: *Script* de Rede Windows

```
1  @echo off
2
3  ::Nome Pastas e caminhos
4  set PCNAME=%1
5  set PASS=%2
6  set REDE=%PCNAME%\Rede
7  set TOOLS=tools\windows
8
9  ::Nomes Ficheiros
10 set INTERFACE_REDE=%REDE%\interface_rede.txt
11 set TABELA_ROTEAMENTO=%REDE%\tabela_roteamento.txt
12 set TABELA_ARP=%REDE%\tabela_arp.txt
13 set CACHE_DNS=%REDE%\cache_dns.txt
14 set LiGACOES_REDE=%REDE%\ligacoes_rede.txt
15
16
17 ::Interface Rede
18 echo Interface de Rede >> %INTERFACE_REDE%
19 echo. >> %INTERFACE_REDE%
20 echo Comando: ipconfig /all >> %INTERFACE_REDE%
21 echo. >> %INTERFACE_REDE%
22 echo Output: >> %INTERFACE_REDE%
23 ipconfig /all >> %INTERFACE_REDE%
24 ::echo "-> Interface de Rede - Concluído"
25 echo concluido
26
27 ::Tabela Roteamento
28 echo Tabela Roteamento >> %TABELA_ROTEAMENTO%
29 echo. >> %TABELA_ROTEAMENTO%
30 echo Comando: netstat -rn >> %TABELA_ROTEAMENTO%
31 echo. >> %TABELA_ROTEAMENTO%
32 echo Output: >> %TABELA_ROTEAMENTO%
33 netstat -rn >> %TABELA_ROTEAMENTO%
34 ::echo "-> Tabela Roteamento - Concluído"
35 echo concluido
36
37 ::Tabela ARP
38 echo Tabela ARP >> %TABELA_ARP%
39 echo. >> %TABELA_ARP%
40 echo Comando: arp.exe -a >> %TABELA_ARP%
41 echo. >> %TABELA_ARP%
42 echo Output: >> %TABELA_ARP%
43 arp.exe -a >> %TABELA_ARP%
44 ::echo "-> Tabela ARP - Concluído"
```

```
45 echo concluido
46
47 ::Cache DNS
48 echo Cache DNS >> %CACHE_DNS%
49 echo. >> %CACHE_DNS%
50 echo Comando: ipconfig /displaydns >> %CACHE_DNS%
51 echo. >> %CACHE_DNS%
52 echo Output: >> %CACHE_DNS%
53 ipconfig /displaydns >> %CACHE_DNS%
54 ::echo "-> Cache DNS - Concluído"
55 echo concluido
56
57 ::Ligações de Rede
58 echo Ligações de Rede >> %LiGACOES_REDE%
59 echo. >> %LiGACOES_REDE%
60 echo Comando: netstat -ano >> %LiGACOES_REDE%
61 echo. >> %LiGACOES_REDE%
62 echo Output: >> %LiGACOES_REDE%
63 netstat -ano >> %LiGACOES_REDE%
64 ::echo "-> Ligações de Rede - Concluído"
65 echo concluido
```

Listagem A.11: *Script* de Rede Admin Windows

```
1  @echo off
2
3  ::Nome Pastas e caminhos
4  set PCNAME=%1
5  set REDE=%PCNAME%\Rede
6  set TOOLS=tools\windows
7
8  ::Nomes Ficheiros
9  set INTERFACE_REDE=%REDE%\interface_rede.txt
10 set TABELA_ROTEAMENTO=%REDE%\tabela_roteamento.txt
11 set TABELA_ARP=%REDE%\tabela_arp.txt
12 set CACHE_DNS=%REDE%\cache_dns.txt
13 set LiGACOES_REDE=%REDE%\ligacoes_rede.txt
14
15
16 ::::::REDE
17
18 ::Interface Rede
19 echo Interface de Rede >> %INTERFACE_REDE%
20 echo. >> %INTERFACE_REDE%
21 echo Comando: ipconfig /all >> %INTERFACE_REDE%
22 echo. >> %INTERFACE_REDE%
23 echo Output: >> %INTERFACE_REDE%
24 ipconfig /all >> %INTERFACE_REDE%
25 ::echo "-> Interface de Rede - Concluído"
26 echo concluido
27
28 ::Tabela Roteamento
29 echo Tabela Roteamento >> %TABELA_ROTEAMENTO%
30 echo. >> %TABELA_ROTEAMENTO%
```

```
31 echo Comando: netstat -rn >> %TABELA_ROTEAMENTO%
32 echo. >> %TABELA_ROTEAMENTO%
33 echo Output: >> %TABELA_ROTEAMENTO%
34 netstat -rn >> %TABELA_ROTEAMENTO%
35 ::echo "-> Tabela Roteamento - Concluído"
36 echo concluido
37
38 ::Tabela ARP
39 echo Tabela ARP >> %TABELA_ARP%
40 echo. >> %TABELA_ARP%
41 echo Comando: arp.exe -a >> %TABELA_ARP%
42 echo. >> %TABELA_ARP%
43 echo Output: >> %TABELA_ARP%
44 arp.exe -a >> %TABELA_ARP%
45 ::echo "-> Tabela ARP - Concluído"
46 echo concluido
47
48 ::Cache DNS
49 echo Cache DNS >> %CACHE_DNS%
50 echo. >> %CACHE_DNS%
51 echo Comando: ipconfig /displaydns >> %CACHE_DNS%
52 echo. >> %CACHE_DNS%
53 echo Output: >> %CACHE_DNS%
54 ipconfig /displaydns >> %CACHE_DNS%
55 ::echo "-> Cache DNS - Concluído"
56 echo concluido
57
58 ::Ligações de Rede
59 echo Ligações de Rede >> %LiGACOES_REDE%
60 echo. >> %LiGACOES_REDE%
61 echo Comando: netstat -ano >> %LiGACOES_REDE%
62 echo. >> %LiGACOES_REDE%
63 echo Output: >> %LiGACOES_REDE%
64 netstat -ano >> %LiGACOES_REDE%
65 ::echo "-> Ligações de Rede - Concluído"
66 echo concluido
```

A.4 *Scripts* de Estado

Listagem A.12: *Script* de Estado Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  LISTA=$5
8  ESTADO=$PCNAME/Estado
9
10
11 #Nomes Ficheiros
```

```
12 FICHEIROS_ABERTOS=$ESTADO/ficheiros_abertos.txt
13 PROCESSOS_EXECUCAO=$ESTADO/processos_execução.txt
14 LISTA_SISTEMA_FICHEIROS=$ESTADO/lista_sistema_ficheiros.txt
15
16
17
18 ###Estado
19
20 #Ficheiros Abertos
21 printf "Ficheiros Abertos\n\n" >> $FICHEIROS_ABERTOS
22 printf "Comando: lsof\n" >> $FICHEIROS_ABERTOS
23 printf "\nOutput:\n" >> $FICHEIROS_ABERTOS
24 lsof >> $FICHEIROS_ABERTOS 2> /dev/null
25 #echo "-> Ficheiros Abertos - Concluído"
26 echo concluido
27
28
29 #Processos em execução
30 printf "Processos em execução\n\n" >> $PROCESSOS_EXECUCAO
31 printf "Comando: top -n 1 -b\n" >> $PROCESSOS_EXECUCAO
32 printf "\nOutput:\n" >> $PROCESSOS_EXECUCAO
33 top -n 1 -b >> $PROCESSOS_EXECUCAO 2> /dev/null
34 #echo "-> Processos em execução - Concluído"
35 echo concluido
36
37
38 if $LISTA; then
39 #Lista completa de sistema de ficheiros
40 printf "Lista completa de sistema de ficheiros\n\n" >>
       $LISTA_SISTEMA_FICHEIROS
41 printf "Comando: ls -Rlsah / \n" >> $LISTA_SISTEMA_FICHEIROS
42 printf "\nOutput:\n" >> $LISTA_SISTEMA_FICHEIROS
43 ls -Rlsah / >> $LISTA_SISTEMA_FICHEIROS 2> /dev/null
44 #echo "-> Lista completa de sistema de ficheiros - Concluído"
45 fi
46
47 echo concluido
```

Listagem A.13: *Script* de Estado Admin Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  LISTA=$5
8  ESTADO=$PCNAME/Estado
9
10
11 #Nomes Ficheiros
12 FICHEIROS_ABERTOS=$ESTADO/ficheiros_abertos.txt
13 PROCESSOS_EXECUCAO=$ESTADO/processos_execução.txt
14 LISTA_SISTEMA_FICHEIROS=$ESTADO/lista_sistema_ficheiros.txt
```

```
15
16
17
18 ###Estado
19
20 #Ficheiros Abertos
21 printf "Ficheiros Abertos\n\n" >> $FICHEIROS_ABERTOS
22 printf "Comando: sudo lsof\n" >> $FICHEIROS_ABERTOS
23 printf "\nOutput:\n" >> $FICHEIROS_ABERTOS
24 echo $PASS | sudo -S lsof >> $FICHEIROS_ABERTOS
25 #echo "-> Ficheiros Abertos - Concluído"
26 echo concluido
27
28
29 #Processos em execução
30 printf "Processos em execução\n\n" >> $PROCESSOS_EXECUCAO
31 printf "Comando: sudo top -n 1 -b\n" >> $PROCESSOS_EXECUCAO
32 printf "\nOutput:\n" >> $PROCESSOS_EXECUCAO
33 echo $PASS | sudo -S top -n 1 -b >> $PROCESSOS_EXECUCAO
34 #echo "-> Processos em execução - Concluído"
35 echo concluido
36
37
38 if $LISTA; then
39 #Lista completa de sistema de ficheiros
40 printf "Lista completa de sistema de ficheiros\n\n" >>
       $LISTA_SISTEMA_FICHEIROS
41 printf "Comando: sudo ls -Rlsah / \n" >>
       $LISTA_SISTEMA_FICHEIROS
42 printf "\nOutput:\n" >> $LISTA_SISTEMA_FICHEIROS
43 echo $PASS | sudo -S ls -Rlsah / >> $LISTA_SISTEMA_FICHEIROS
44 #echo "-> Lista completa de sistema de ficheiros - Concluído"
45 fi
46
47 echo concluido
```

Listagem A.14: *Script* de Estado macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  LISTA=$5
8  ESTADO=$PCNAME/Estado
9
10
11 #Nomes Ficheiros
12 FICHEIROS_ABERTOS=$ESTADO/ficheiros_abertos.txt
13 PROCESSOS_EXECUCAO=$ESTADO/processos_execução.txt
14 LISTA_SISTEMA_FICHEIROS=$ESTADO/lista_sistema_ficheiros.txt
15
16
```

```
17
18 ###Estado
19
20 #Ficheiros Abertos
21 printf "Ficheiros Abertos\n\n" >> $FICHEIROS_ABERTOS
22 printf "Comando: lsof\n" >> $FICHEIROS_ABERTOS
23 printf "\nOutput:\n" >> $FICHEIROS_ABERTOS
24 lsof >> $FICHEIROS_ABERTOS
25 #echo "-> Ficheiros Abertos - Concluído"
26 echo concluido
27
28
29 #Processos em execução
30 printf "Processos em execução\n\n" >> $PROCESSOS_EXECUCAO
31 printf "Comando: ps auxwww\n" >> $PROCESSOS_EXECUCAO
32 printf "\nOutput:\n" >> $PROCESSOS_EXECUCAO
33 ps auxwww >> $PROCESSOS_EXECUCAO
34 #echo "-> Processos em execução - Concluído"
35 echo concluido
36
37
38 if $LISTA; then
39 #Lista completa de sistema de ficheiros
40 printf "Lista completa de sistema de ficheiros\n\n" >>
       $LISTA_SISTEMA_FICHEIROS
41 printf "Comando: ls -Rlsah / \n" >> $LISTA_SISTEMA_FICHEIROS
42 printf "\nOutput:\n" >> $LISTA_SISTEMA_FICHEIROS
43 ls -Rlsah / >> $LISTA_SISTEMA_FICHEIROS
44 #echo "-> Lista completa de sistema de ficheiros - Concluído"
45 fi
46
47 echo concluido
```

Listagem A.15: *Script* de Estado Admin macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  LISTA=$5
8  ESTADO=$PCNAME/Estado
9
10
11 #Nomes Ficheiros
12 FICHEIROS_ABERTOS=$ESTADO/ficheiros_abertos.txt
13 PROCESSOS_EXECUCAO=$ESTADO/processos_execução.txt
14 LISTA_SISTEMA_FICHEIROS=$ESTADO/lista_sistema_ficheiros.txt
15
16
17
18 ###Estado
19
```

```
20 #Ficheiros Abertos
21 printf "Ficheiros Abertos\n\n" >> $FICHEIROS_ABERTOS
22 printf "Comando: sudo lsof\n" >> $FICHEIROS_ABERTOS
23 printf "\nOutput:\n" >> $FICHEIROS_ABERTOS
24 echo $PASS | sudo -S lsof >> $FICHEIROS_ABERTOS
25 #echo "-> Ficheiros Abertos - Concluído"
26 echo concluido
27
28
29 #Processos em execução
30 printf "Processos em execução\n\n" >> $PROCESSOS_EXECUCAO
31 printf "Comando: sudo ps auxwww\n" >> $PROCESSOS_EXECUCAO
32 printf "\nOutput:\n" >> $PROCESSOS_EXECUCAO
33 echo $PASS | sudo -S ps auxwww >> $PROCESSOS_EXECUCAO
34 #echo "-> Processos em execução - Concluído"
35 echo concluido
36
37
38 if $LISTA; then
39 #Lista completa de sistema de ficheiros
40 printf "Lista completa de sistema de ficheiros\n\n" >>
       $LISTA_SISTEMA_FICHEIROS
41 printf "Comando: sudo ls -Rlsah / \n" >>
       $LISTA_SISTEMA_FICHEIROS
42 printf "\nOutput:\n" >> $LISTA_SISTEMA_FICHEIROS
43 echo $PASS | sudo -S ls -Rlsah / >> $LISTA_SISTEMA_FICHEIROS
44 #echo "-> Lista completa de sistema de ficheiros - Concluído"
45 fi
46
47 echo concluido
```

Listagem A.16: *Script* de Estado Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set PASS=%2
7  set LISTA=%5
8  set ESTADO=%PCNAME%\Estado
9  set TOOLS=tools\windows
10
11
12 ::Nomes Ficheiros
13 set FICHEIROS_ABERTOS=%ESTADO%\ficheiros_abertos.txt
14 set PROCESSOS_EXECUCAO=%ESTADO%\processos_execucao.txt
15 set LISTA_SISTEMA_FICHEIROS=%ESTADO%\lista_sistema_ficheiros.txt
16
17
18
19 ::Ficheiros Abertos
20 echo Ficheiros Abertos >> %FICHEIROS_ABERTOS%
21 echo. >> %FICHEIROS_ABERTOS%
```

```
22 echo Comando: openedfilesview.exe /stext >> %FICHEIROS_ABERTOS%
23 echo. >> %FICHEIROS_ABERTOS%
24 echo Output: >> %FICHEIROS_ABERTOS%
25 echo Comando disponivel só para Administrador! >> %
       FICHEIROS_ABERTOS%
26 ::echo "-> Ficheiros Abertos - Concluído"
27 echo concluido
28
29
30 ::Processos em execução
31 echo Processos em execução >> %PROCESSOS_EXECUCAO%
32 echo. >> %PROCESSOS_EXECUCAO%
33 echo Comando: pslist.exe /accepteula >> %PROCESSOS_EXECUCAO%
34 echo. >> %PROCESSOS_EXECUCAO%
35 echo Output: >> %PROCESSOS_EXECUCAO%
36 %TOOLS%\pslist.exe /accepteula >> %PROCESSOS_EXECUCAO%
37 ::echo "-> Processos em execução - Concluído"
38 echo concluido
39
40
41 if "%LISTA%" == "true" (
42 goto lista
43 ) else (
44 goto fim
45 )
46
47
48 :lista
49 ::Lista completa de sistema de ficheiros
50 echo Lista completa de sistema de ficheiros >> %
       LISTA_SISTEMA_FICHEIROS%
51 echo. >> %LISTA_SISTEMA_FICHEIROS%
52 echo Comando: dir /S /B /AHD >> %LISTA_SISTEMA_FICHEIROS%
53 echo. >> %LISTA_SISTEMA_FICHEIROS%
54 echo Output: >> %LISTA_SISTEMA_FICHEIROS%
55 dir /S /B /AHD %HOMEDRIVE%\ >> %LISTA_SISTEMA_FICHEIROS%
56 ::echo "-> Lista completa de sistema de ficheiros - Concluído"
57 ::echo concluido
58
59
60 :fim
61 echo concluido
```

Listagem A.17: *Script* de Estado Admin Windows

```
1 @echo off
2
3
4 ::Nome Pastas e caminhos
5 set PCNAME=%1
6 set LISTA=%5
7 set ESTADO=%PCNAME%\Estado
8 set TOOLS=tools\windows
9
```

```
10
11 ::Nomes Ficheiros
12 set FICHEIROS_ABERTOS=%ESTADO%\ficheiros_abertos.txt
13 set PROCESSOS_EXECUCAO=%ESTADO%\processos_execucao.txt
14 set LISTA_SISTEMA_FICHEIROS=%ESTADO%\lista_sistema_ficheiros.txt
15
16
17
18 ::::::Estado
19
20 ::Ficheiros Abertos
21 echo Ficheiros Abertos >> %FICHEIROS_ABERTOS%
22 echo. >> %FICHEIROS_ABERTOS%
23 echo Comando: openedfilesview.exe /stext >> %FICHEIROS_ABERTOS%
24 echo. >> %FICHEIROS_ABERTOS%
25 echo Output: >> %FICHEIROS_ABERTOS%
26 %TOOLS%\OpenedFilesView.exe /stext x.txt
27 type x.txt >> %FICHEIROS_ABERTOS%
28 del x.txt
29 ::echo "-> Ficheiros Abertos - Concluído"
30 echo concluido
31
32
33 ::Processos em execução
34 echo Processos em execução >> %PROCESSOS_EXECUCAO%
35 echo. >> %PROCESSOS_EXECUCAO%
36 echo Comando: pslist.exe /accepteula >> %PROCESSOS_EXECUCAO%
37 echo. >> %PROCESSOS_EXECUCAO%
38 echo Output: >> %PROCESSOS_EXECUCAO%
39 %TOOLS%\pslist.exe /accepteula >> %PROCESSOS_EXECUCAO%
40 ::echo "-> Processos em execução - Concluído"
41 echo concluido
42
43
44 if "%LISTA%" == "true" (
45 goto lista
46 ) else (
47 goto fim
48 )
49
50
51 :lista
52 ::Lista completa de sistema de ficheiros
53 echo Lista completa de sistema de ficheiros >> %
       LISTA_SISTEMA_FICHEIROS%
54 echo. >> %LISTA_SISTEMA_FICHEIROS%
55 echo Comando: dir /S /B /AHD >> %LISTA_SISTEMA_FICHEIROS%
56 echo. >> %LISTA_SISTEMA_FICHEIROS%
57 echo Output: >> %LISTA_SISTEMA_FICHEIROS%
58 dir /S /B /AHD %HOMEDRIVE%\ >> %LISTA_SISTEMA_FICHEIROS%
59 ::echo "-> Lista completa de sistema de ficheiros - Concluído"
60 ::echo concluído
61
62
```

```
63 :fim
64 echo concluido
```

A.5 *Scripts* de Sistema

Listagem A.18: *Script* de Sistema Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SISTEMA=$PCNAME/Sistema
8  INTEGRIDADE=$PCNAME/Integridade
9
10
11 #Nomes Ficheiros
12 TEMPO_DATA_SISTEMA=$SISTEMA/tempo_data_sistema.txt
13 VERSAO_SISTEMA_OPERATIVO=$SISTEMA/versao_sistema_operativo.txt
14 LISTA_INICIADOS_ARRANQUE=$SISTEMA/lista_iniciados_arranque.txt
15 LISTA_SOFTWARE_INSTALADO=$SISTEMA/lista_software_instalado.txt
16 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
17
18 #Outras Variaveis
19 USER=`whoami`
20
21
22 ###Sistema
23
24 #Tempo e data do sistema
25 printf "Tempo e data do sistema\n\n" >> $TEMPO_DATA_SISTEMA
26 printf "Comando: date\n" >> $TEMPO_DATA_SISTEMA
27 printf "\nOutput:\n" >> $TEMPO_DATA_SISTEMA
28 date >> $TEMPO_DATA_SISTEMA 2> /dev/null
29 #echo "-> Tempo e data do sistema - Concluído"
30 echo concluido
31
32
33 #versão do sistema operativo
34 printf "Versão do sistema operativo\n\n" >>
       $VERSAO_SISTEMA_OPERATIVO
35 printf "Comando: uname -a\n" >> $VERSAO_SISTEMA_OPERATIVO
36 printf "\nOutput:\n" >> $VERSAO_SISTEMA_OPERATIVO
37 uname -a >> $VERSAO_SISTEMA_OPERATIVO 2> /dev/null
38 #echo "-> Versão do sistema operativo - Concluído"
39 echo concluido
40
41
42 #Lista de serviços e programas iniciados no arranque
43 printf "Lista de serviços e programas iniciados no arranque\n\n"
       >> $LISTA_INICIADOS_ARRANQUE
```

```
44 printf "Comando: ls -l /home/$USER/.config/autostart/\n" >>
       $LISTA_INICIADOS_ARRANQUE
45 printf "\nOutput:\n" >> $LISTA_INICIADOS_ARRANQUE
46 ls -l /home/$USER/.config/autostart/ >>
       $LISTA_INICIADOS_ARRANQUE 2> /dev/null
47 #echo "-> Lista de serviços e programas iniciados no arranque -
       Concluído"
48 echo concluido
49
50
51 #Lista de software instalado
52 printf "Lista de software instalado\n\n" >>
       $LISTA_SOFTWARE_INSTALADO
53
54 if  which rpm >/dev/null 2>/dev/null; then
55 printf "Comando: rpm -qa\n" >> $LISTA_SOFTWARE_INSTALADO
56 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
57 rpm -qa >> $LISTA_SOFTWARE_INSTALADO 2> /dev/null
58
59 elif  which dpkg >/dev/null 2>/dev/null ; then
60 printf "Comando: dpkg -l\n" >> $LISTA_SOFTWARE_INSTALADO
61 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
62 dpkg -l >> $LISTA_SOFTWARE_INSTALADO 2> /dev/null
63
64 elif which pacman >/dev/null 2>/dev/null ; then
65 printf "Comando: pacman -Q\n" >> $LISTA_SOFTWARE_INSTALADO
66 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
67 pacman -Q >> $LISTA_SOFTWARE_INSTALADO 2> /dev/null
68
69 elif which slpkg >/dev/null 2>/dev/null ; then
70 printf "Comando: ls -l /var/log/packages\n" >>
       $LISTA_SOFTWARE_INSTALADO
71 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
72 ls -l /var/log/packages >> $LISTA_SOFTWARE_INSTALADO 2> /dev/
       null
73
74 elif which emerge >/dev/null 2>/dev/null ; then
75 printf "Comando: qlist -IC\n" >> $LISTA_SOFTWARE_INSTALADO
76 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
77 qlist -IC >> $LISTA_SOFTWARE_INSTALADO 2> /dev/null
78
79 else
80 printf "Comando: \n\n" >> $LISTA_SOFTWARE_INSTALADO
81 printf "Output:\n" >> $LISTA_SOFTWARE_INSTALADO
82 printf "Não disponivel" >> $LISTA_SOFTWARE_INSTALADO
83
84 fi
85 #echo "-> Lista de software instalado - Concluído"
86 echo concluido
```

Listagem A.19: *Script* de Sistema Admin Linux

```
1 #!/bin/bash
2
```

```
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SISTEMA=$PCNAME/Sistema
8
9
10 #Nomes Ficheiros
11 TEMPO_DATA_SISTEMA=$SISTEMA/tempo_data_sistema.txt
12 VERSAO_SISTEMA_OPERATIVO=$SISTEMA/versao_sistema_operativo.txt
13 LISTA_INICIADOS_ARRANQUE=$SISTEMA/lista_iniciados_arranque.txt
14 LISTA_SOFTWARE_INSTALADO=$SISTEMA/lista_software_instalado.txt
15
16 #Outras Variaveis
17 USER=`whoami`
18
19
20 ###Sistema
21
22 #Tempo e data do sistema
23 printf "Tempo e data do sistema\n\n" >> $TEMPO_DATA_SISTEMA
24 printf "Comando: sudo date\n" >> $TEMPO_DATA_SISTEMA
25 printf "Versão:\n" >> $TEMPO_DATA_SISTEMA
26 echo $PASS | sudo -S date --version >> $TEMPO_DATA_SISTEMA
27 printf "\nSHA256 Comando: " >> $TEMPO_DATA_SISTEMA
28 echo $PASS | sudo -S date | sha256sum >> $TEMPO_DATA_SISTEMA
29 printf "\nOutput:\n" >> $TEMPO_DATA_SISTEMA
30 echo $PASS | sudo -S date >> $TEMPO_DATA_SISTEMA
31 #echo "-> Tempo e data do sistema - Concluído"
32 echo concluido
33
34
35 #versão do sistema operativo
36 printf "Versão do sistema operativo\n\n" >>
       $VERSAO_SISTEMA_OPERATIVO
37 printf "Comando: sudo uname -a\n" >> $VERSAO_SISTEMA_OPERATIVO
38 printf "\nOutput:\n" >> $VERSAO_SISTEMA_OPERATIVO
39 echo $PASS | sudo -S uname -a >> $VERSAO_SISTEMA_OPERATIVO
40 #echo "-> Versão do sistema operativo - Concluído"
41 echo concluido
42
43
44 #Lista de serviços e programas iniciados no arranque
45 printf "Lista de serviços e programas iniciados no arranque\n\n"
       >> $LISTA_INICIADOS_ARRANQUE
46 printf "Comando: sudo ls -l /home/$USER/.config/autostart/\n" >>
       $LISTA_INICIADOS_ARRANQUE
47 printf "\nOutput:\n" >> $LISTA_INICIADOS_ARRANQUE
48 echo $PASS | sudo -S ls -l /home/$USER/.config/autostart/ >>
       $LISTA_INICIADOS_ARRANQUE
49 #echo "-> Lista de serviços e programas iniciados no arranque -
       Concluído"
50 echo concluido
51
```

```
52
53 #Lista de software instalado
54 printf "Lista de software instalado\n\n" >>
       $LISTA_SOFTWARE_INSTALADO
55
56 if  which rpm >/dev/null 2>/dev/null; then
57 printf "Comando: sudo rpm -qa\n" >> $LISTA_SOFTWARE_INSTALADO
58 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
59 echo $PASS | sudo -S rpm -qa >> $LISTA_SOFTWARE_INSTALADO
60
61 elif  which dpkg >/dev/null 2>/dev/null ; then
62 printf "Comando: sudo dpkg -l\n" >> $LISTA_SOFTWARE_INSTALADO
63 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
64 echo $PASS | sudo -S dpkg -l >> $LISTA_SOFTWARE_INSTALADO
65
66 elif which pacman >/dev/null 2>/dev/null ; then
67 printf "Comando: sudo pacman -Q\n" >> $LISTA_SOFTWARE_INSTALADO
68 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
69 echo $PASS | sudo -S pacman -Q >> $LISTA_SOFTWARE_INSTALADO
70
71 elif which slpkg >/dev/null 2>/dev/null ; then
72 printf "Comando: sudo ls -l /var/log/packages\n" >>
       $LISTA_SOFTWARE_INSTALADO
73 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
74 echo $PASS | sudo -S ls -l /var/log/packages >>
       $LISTA_SOFTWARE_INSTALADO
75
76 elif which emerge >/dev/null 2>/dev/null ; then
77 printf "Comando: sudo qlist -IC\n" >> $LISTA_SOFTWARE_INSTALADO
78 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
79 echo $PASS | sudo -S qlist -IC >> $LISTA_SOFTWARE_INSTALADO
80
81 else
82 printf "Comando: \n\n" >> $LISTA_SOFTWARE_INSTALADO
83 printf "Output:\n" >> $LISTA_SOFTWARE_INSTALADO
84 printf "Não disponivel" >> $LISTA_SOFTWARE_INSTALADO
85 fi
86 #echo "-> Lista de software instalado - Concluído"
87 echo concluido
```

Listagem A.20: *Script* de Sistema macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SISTEMA=$PCNAME/Sistema
8
9
10 #Nomes Ficheiros
11 TEMPO_DATA_SISTEMA=$SISTEMA/tempo_data_sistema.txt
12 VERSAO_SISTEMA_OPERATIVO=$SISTEMA/versao_sistema_operativo.txt
```

```
13 LISTA_INICIADOS_ARRANQUE=$SISTEMA/lista_iniciados_arranque.txt
14 LISTA_SOFTWARE_INSTALADO=$SISTEMA/lista_software_instalado.txt
15
16
17
18 ###Sistema
19
20 #Tempo e data do sistema
21 printf "Tempo e data do sistema\n\n" >> $TEMPO_DATA_SISTEMA
22 printf "Comando: \n systemsetup -getdate\n systemsetup -gettime\
       n systemsetup -gettimezone\n" >> $TEMPO_DATA_SISTEMA
23 printf "\nOutput:\n" >> $TEMPO_DATA_SISTEMA
24 systemsetup -getdate >> $TEMPO_DATA_SISTEMA
25 systemsetup -gettime >> $TEMPO_DATA_SISTEMA
26 systemsetup -gettimezone >> $TEMPO_DATA_SISTEMA
27 #echo "-> Tempo e data do sistema - Concluído"
28 echo concluido
29
30
31 #versão do sistema operativo
32 printf "Versão do sistema operativo\n\n" >>
       $VERSAO_SISTEMA_OPERATIVO
33 printf "Comando: uname -a\n" >> $VERSAO_SISTEMA_OPERATIVO
34 printf "\nOutput:\n" >> $VERSAO_SISTEMA_OPERATIVO
35 uname -a >> $VERSAO_SISTEMA_OPERATIVO
36 #echo "-> Versão do sistema operativo - Concluído"
37 echo concluido
38
39
40 #Lista de serviços e programas iniciados no arranque
41 printf "Lista de serviços e programas iniciados no arranque\n\n"
       >> $LISTA_INICIADOS_ARRANQUE
42 printf "Comando: find /Applications/ -name LogInItems -exec ls -
       lsct {} \; \n" >> $LISTA_INICIADOS_ARRANQUE
43 printf "\nOutput:\n" >> $LISTA_INICIADOS_ARRANQUE
44 find /Applications/ -name LogInItems -exec ls -lsct {} \; >>
       $LISTA_INICIADOS_ARRANQUE
45 #echo "-> Lista de serviços e programas iniciados no arranque -
       Concluído"
46 echo concluido
47
48
49 #Lista de software instalado
50 printf "Lista de software instalado\n\n" >>
       $LISTA_SOFTWARE_INSTALADO
51 printf "Comando: ls -l /Applications/\n" >>
       $LISTA_SOFTWARE_INSTALADO
52 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
53 ls -l /Applications/ >> $LISTA_SOFTWARE_INSTALADO
54 #echo "-> Lista de software instalado - Concluído"
55 echo concluido
```

Listagem A.21: *Script* de Sistema Admin macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SISTEMA=$PCNAME/Sistema
8
9
10 #Nomes Ficheiros
11 TEMPO_DATA_SISTEMA=$SISTEMA/tempo_data_sistema.txt
12 VERSAO_SISTEMA_OPERATIVO=$SISTEMA/versao_sistema_operativo.txt
13 LISTA_INICIADOS_ARRANQUE=$SISTEMA/lista_iniciados_arranque.txt
14 LISTA_SOFTWARE_INSTALADO=$SISTEMA/lista_software_instalado.txt
15
16
17
18 ###Sistema
19
20 #Tempo e data do sistema
21 printf "Tempo e data do sistema\n\n" >> $TEMPO_DATA_SISTEMA
22 printf "Comando: \n sudo systemsetup -getdate\n sudo systemsetup
       -gettime\n sudo systemsetup -gettimezone\n" >>
     $TEMPO_DATA_SISTEMA
23 printf "\nOutput:\n" >> $TEMPO_DATA_SISTEMA
24 echo $PASS | sudo -S systemsetup -getdate >> $TEMPO_DATA_SISTEMA
25 echo $PASS | sudo -S systemsetup -gettime >> $TEMPO_DATA_SISTEMA
26 echo $PASS | sudo -S systemsetup -gettimezone >>
     $TEMPO_DATA_SISTEMA
27 #echo "-> Tempo e data do sistema - Concluído"
28 echo concluido
29
30
31 #versão do sistema operativo
32 printf "Versão do sistema operativo\n\n" >>
     $VERSAO_SISTEMA_OPERATIVO
33 printf "Comando: sudo uname -a\n" >> $VERSAO_SISTEMA_OPERATIVO
34 printf "\nOutput:\n" >> $VERSAO_SISTEMA_OPERATIVO
35 echo $PASS | sudo -S uname -a >> $VERSAO_SISTEMA_OPERATIVO
36 #echo "-> Versão do sistema operativo - Concluído"
37 echo concluido
38
39
40 #Lista de serviços e programas iniciados no arranque
41 printf "Lista de serviços e programas iniciados no arranque\n\n"
       >> $LISTA_INICIADOS_ARRANQUE
42 printf "Comando: sudo find /Applications/ -name LogInItems -exec
       ls -lsct {} \; \n" >> $LISTA_INICIADOS_ARRANQUE
43 printf "\nOutput:\n" >> $LISTA_INICIADOS_ARRANQUE
44 echo $PASS | sudo -S find /Applications/ -name LogInItems -exec
     ls -lsct {} \; >> $LISTA_INICIADOS_ARRANQUE
45 #echo "-> Lista de serviços e programas iniciados no arranque -
     Concluído"
46 echo concluido
```

```
47
48
49 #Lista de software instalado
50 printf "Lista de software instalado\n\n" >>
       $LISTA_SOFTWARE_INSTALADO
51 printf "Comando: sudo ls -l /Applications/\n" >>
       $LISTA_SOFTWARE_INSTALADO
52 printf "\nOutput:\n" >> $LISTA_SOFTWARE_INSTALADO
53 echo $PASS | sudo -S ls -l /Applications/ >>
       $LISTA_SOFTWARE_INSTALADO
54 #echo "-> Lista de software instalado - Concluído"
55 echo concluido
```

Listagem A.22: *Script* de Sistema Windows

```
1  @echo off
2
3  ::Nome Pastas e caminhos
4  set PCNAME=%1
5  set PASS=%2
6  set SISTEMA=%PCNAME%\Sistema
7  set TOOLS=tools\windows
8
9  ::Nomes Ficheiros
10 set TEMPO_DATA_SISTEMA=%SISTEMA%\tempo_data_sistema.txt
11 set VERSAO_SISTEMA_OPERATIVO=%SISTEMA%\versao_sistema_operativo.
       txt
12 set LISTA_INICIADOS_ARRANQUE=%SISTEMA%\lista_iniciados_arranque.
       txt
13 set LISTA_SOFTWARE_INSTALADO=%SISTEMA%\lista_software_instalado.
       txt
14
15 ::::::Sistema
16 ::Tempo e data do sistema
17 echo Tempo e data do sistema >> %TEMPO_DATA_SISTEMA%
18 echo. >> %TEMPO_DATA_SISTEMA%
19 echo Comando: >> %TEMPO_DATA_SISTEMA%
20 echo time/T >> %TEMPO_DATA_SISTEMA%
21 echo date/T >> %TEMPO_DATA_SISTEMA%
22 echo w32tm /tz >> %TEMPO_DATA_SISTEMA%
23 echo. >> %TEMPO_DATA_SISTEMA%
24 echo Output: >> %TEMPO_DATA_SISTEMA%
25 time/T >> %TEMPO_DATA_SISTEMA%
26 date/T >> %TEMPO_DATA_SISTEMA%
27 w32tm /tz >> %TEMPO_DATA_SISTEMA%
28 ::echo "-> Tempo e data do sistema - Concluído"
29 echo concluido
30
31
32 ::versão do sistema operativo
33 echo Versão do sistema operativo >> %VERSAO_SISTEMA_OPERATIVO%
34 echo. >> %VERSAO_SISTEMA_OPERATIVO%
35 echo Comando: ver >> %VERSAO_SISTEMA_OPERATIVO%
36 echo. >> %VERSAO_SISTEMA_OPERATIVO%
```

```
37 echo Output: >> %VERSAO_SISTEMA_OPERATIVO%
38 ver >> %VERSAO_SISTEMA_OPERATIVO%
39 ::echo "-> Versão do sistema operativo - Concluído"
40 echo concluido
41
42 ::Lista de serviços e programas iniciados no arranque
43 echo Lista de serviços e programas iniciados no arranque >> %
       LISTA_INICIADOS_ARRANQUE%
44 echo. >> %LISTA_INICIADOS_ARRANQUE%
45 echo Comando: wmic startup list full >> %
       LISTA_INICIADOS_ARRANQUE%
46 echo. >> %LISTA_INICIADOS_ARRANQUE%
47 echo Output: >> %LISTA_INICIADOS_ARRANQUE%
48 wmic /Output:"x.txt" startup list full
49 type x.txt >> %LISTA_INICIADOS_ARRANQUE%
50 del x.txt
51 ::echo "-> Lista de serviços e programas iniciados no arranque -
        Concluído"
52 echo concluido
53
54 ::Lista de software instalado
55 echo Lista de software instalado >> %LISTA_SOFTWARE_INSTALADO%
56 echo. >> %LISTA_SOFTWARE_INSTALADO%
57 echo Comando: wmic product get Name, Version >> %
       LISTA_SOFTWARE_INSTALADO%
58 echo. >> %LISTA_SOFTWARE_INSTALADO%
59 echo Output: >> %LISTA_SOFTWARE_INSTALADO%
60 wmic /Output:"x.txt" product get Name, Version >> %
       LISTA_SOFTWARE_INSTALADO%
61 type x.txt >> %LISTA_SOFTWARE_INSTALADO%
62 del x.txt
63 ::echo "-> Lista de software instalado - Concluído"
64 echo concluido
```

Listagem A.23: *Script* de sistema Admin Windows

```
1  @echo off
2
3  ::Nome Pastas e caminhos
4  set PCNAME=%1
5  set SISTEMA=%PCNAME%\Sistema
6  set TOOLS=tools\windows
7
8  ::Nomes Ficheiros
9  set TEMPO_DATA_SISTEMA=%SISTEMA%\tempo_data_sistema.txt
10 set VERSAO_SISTEMA_OPERATIVO=%SISTEMA%\versao_sistema_operativo.
       txt
11 set LISTA_INICIADOS_ARRANQUE=%SISTEMA%\lista_iniciados_arranque.
       txt
12 set LISTA_SOFTWARE_INSTALADO=%SISTEMA%\lista_software_instalado.
       txt
13
14 ::::::Sistema
15 ::Tempo e data do sistema
```

```
16 echo Tempo e data do sistema >> %TEMPO_DATA_SISTEMA%
17 echo. >> %TEMPO_DATA_SISTEMA%
18 echo Comando: >> %TEMPO_DATA_SISTEMA%
19 echo time/T >> %TEMPO_DATA_SISTEMA%
20 echo date/T >> %TEMPO_DATA_SISTEMA%
21 echo w32tm /tz >> %TEMPO_DATA_SISTEMA%
22 echo. >> %TEMPO_DATA_SISTEMA%
23 echo Output: >> %TEMPO_DATA_SISTEMA%
24 time/T >> %TEMPO_DATA_SISTEMA%
25 date/T >> %TEMPO_DATA_SISTEMA%
26 w32tm /tz >> %TEMPO_DATA_SISTEMA%
27 ::echo "-> Tempo e data do sistema - Concluído"
28 echo concluido
29
30 ::versão do sistema operativo
31 echo Versão do sistema operativo >> %VERSAO_SISTEMA_OPERATIVO%
32 echo. >> %VERSAO_SISTEMA_OPERATIVO%
33 echo Comando: ver >> %VERSAO_SISTEMA_OPERATIVO%
34 echo. >> %VERSAO_SISTEMA_OPERATIVO%
35 echo Output: >> %VERSAO_SISTEMA_OPERATIVO%
36 ver >> %VERSAO_SISTEMA_OPERATIVO%
37 ::echo "-> Versão do sistema operativo - Concluído"
38 echo concluido
39
40 ::Lista de serviços e programas iniciados no arranque
41 echo Lista de serviços e programas iniciados no arranque >> %
       LISTA_INICIADOS_ARRANQUE%
42 echo. >> %LISTA_INICIADOS_ARRANQUE%
43 echo Comando: wmic startup list full >> %
       LISTA_INICIADOS_ARRANQUE%
44 echo. >> %LISTA_INICIADOS_ARRANQUE%
45 echo Output: >> %LISTA_INICIADOS_ARRANQUE%
46 wmic /Output:"x.txt" startup list full
47 type x.txt >> %LISTA_INICIADOS_ARRANQUE%
48 del x.txt
49 ::echo "-> Lista de serviços e programas iniciados no arranque -
        Concluído"
50 echo concluido
51
52 ::Lista de software instalado
53 echo Lista de software instalado >> %LISTA_SOFTWARE_INSTALADO%
54 echo. >> %LISTA_SOFTWARE_INSTALADO%
55 echo Comando: wmic product get Name, Version >> %
       LISTA_SOFTWARE_INSTALADO%
56 echo. >> %LISTA_SOFTWARE_INSTALADO%
57 echo Output: >> %LISTA_SOFTWARE_INSTALADO%
58 wmic /Output:"x.txt" product get Name, Version >> %
       LISTA_SOFTWARE_INSTALADO%
59 type x.txt >> %LISTA_SOFTWARE_INSTALADO%
60 del x.txt
61 ::echo "-> Lista de software instalado - Concluído"
62 echo concluido
```

A.6 *Scripts* de Utilizador

Listagem A.24: *Script* de Utilizador Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  UTILIZADOR=$PCNAME/Utilizador
8  INTEGRIDADE=$PCNAME/Integridade
9
10
11 #Nomes Ficheiros
12 LISTA_TAREFAS_AGENDADAS=$UTILIZADOR/lista_tarefas_agendadas.txt
13 LISTA_CONTAS_UTILIZADORES=$UTILIZADOR/lista_contas_utilizadores.
       txt
14 HISTORICO_LOGIN_UTILIZADOR=$UTILIZADOR/
       historico_login_utilizador.txt
15 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
16
17
18
19 ###Utilizador
20
21 #Lista de tarefas agendadas
22 printf "Lista de tarefas agendadas\n\n" >>
       $LISTA_TAREFAS_AGENDADAS
23 printf "Comando: crontab -l\n" >> $LISTA_TAREFAS_AGENDADAS
24 printf "\nOutput:\n" >> $LISTA_TAREFAS_AGENDADAS
25 crontab -l >> $LISTA_TAREFAS_AGENDADAS 2> /dev/null
26 #echo "-> Lista de tarefas agendadas - Concluído"
27 echo concluido
28
29
30 #Lista de contas de utilizadores
31 printf "Lista de contas de utilizadores\n\n" >>
       $LISTA_CONTAS_UTILIZADORES
32 printf "Comando: cat /etc/passwd\n" >>
       $LISTA_CONTAS_UTILIZADORES
33 printf "\nOutput:\n" >> $LISTA_CONTAS_UTILIZADORES
34 cat /etc/passwd >> $LISTA_CONTAS_UTILIZADORES 2> /dev/null
35 #echo "-> Lista de contas de utilizadores - Concluído"
36 echo concluido
37
38
39 #Histórico do login do utilizador
40 printf "Histórico do login do utilizador\n\n" >>
       $HISTORICO_LOGIN_UTILIZADOR
41 printf "Comando: last\n" >> $HISTORICO_LOGIN_UTILIZADOR
42 printf "\nOutput:\n" >> $HISTORICO_LOGIN_UTILIZADOR
43 last >> $HISTORICO_LOGIN_UTILIZADOR 2> /dev/null
44 #echo "-> Histórico do login do utilizador - Concluído"
```

```
45 echo concluido
```

Listagem A.25: *Script* de Utilizador Admin Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  UTILIZADOR=$PCNAME/Utilizador
8  INTEGRIDADE=$PCNAME/Integridade
9
10
11 #Nomes Ficheiros
12 LISTA_TAREFAS_AGENDADAS=$UTILIZADOR/lista_tarefas_agendadas.txt
13 LISTA_CONTAS_UTILIZADORES=$UTILIZADOR/lista_contas_utilizadores.
       txt
14 HISTORICO_LOGIN_UTILIZADOR=$UTILIZADOR/
       historico_login_utilizador.txt
15 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
16
17
18
19 ###Utilizador
20
21 #Lista de tarefas agendadas
22 printf "Lista de tarefas agendadas\n\n" >>
       $LISTA_TAREFAS_AGENDADAS
23 printf "Comando: sudo crontab -l\n" >> $LISTA_TAREFAS_AGENDADAS
24 printf "\nOutput:\n" >> $LISTA_TAREFAS_AGENDADAS
25 echo $PASS | sudo -S crontab -l >> $LISTA_TAREFAS_AGENDADAS
26 #echo "-> Lista de tarefas agendadas - Concluído"
27 echo concluido
28
29
30 #Lista de contas de utilizadores
31 printf "Lista de contas de utilizadores\n\n" >>
       $LISTA_CONTAS_UTILIZADORES
32 printf "Comando: sudo cat /etc/passwd\n" >>
       $LISTA_CONTAS_UTILIZADORES
33 printf "\nOutput:\n" >> $LISTA_CONTAS_UTILIZADORES
34 echo $PASS | sudo -S cat /etc/passwd >>
       $LISTA_CONTAS_UTILIZADORES
35 #echo "-> Lista de contas de utilizadores - Concluído"
36 echo concluido
37
38
39 #Histórico do login do utilizador
40 printf "Histórico do login do utilizador\n\n" >>
       $HISTORICO_LOGIN_UTILIZADOR
41 printf "Comando: sudo last\n" >> $HISTORICO_LOGIN_UTILIZADOR
42 printf "\nOutput:\n" >> $HISTORICO_LOGIN_UTILIZADOR
43 echo $PASS | sudo -S last >> $HISTORICO_LOGIN_UTILIZADOR
```

```
44 #echo "-> Histórico do login do utilizador - Concluído"
45 echo concluido
```

Listagem A.26: *Script* de Utilizador macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  UTILIZADOR=$PCNAME/Utilizador
8
9
10 #Nomes Ficheiros
11 LISTA_TAREFAS_AGENDADAS=$UTILIZADOR/lista_tarefas_agendadas.txt
12 LISTA_CONTAS_UTILIZADORES=$UTILIZADOR/lista_contas_utilizadores.
       txt
13 HISTORICO_LOGIN_UTILIZADOR=$UTILIZADOR/
       historico_login_utilizador.txt
14
15
16
17 ###Utilizador
18
19 #Lista de tarefas agendadas
20 printf "Lista de tarefas agendadas\n\n" >>
       $LISTA_TAREFAS_AGENDADAS
21 printf "Comando: crontab -l\n" >> $LISTA_TAREFAS_AGENDADAS
22 printf "Versão:\n" >> $LISTA_TAREFAS_AGENDADAS
23 ## falta versão
24 printf "\nSHA256 Comando: " >> $LISTA_TAREFAS_AGENDADAS
25 crontab | shasum -a 256 >> $LISTA_TAREFAS_AGENDADAS
26 printf "\nOutput:\n" >> $LISTA_TAREFAS_AGENDADAS
27 crontab -l >> $LISTA_TAREFAS_AGENDADAS
28 #echo "-> Lista de tarefas agendadas - Concluído"
29 echo concluido
30
31
32 #Lista de contas de utilizadores
33 printf "Lista de contas de utilizadores\n\n" >>
       $LISTA_CONTAS_UTILIZADORES
34 printf "Comando: cat /etc/passwd\n" >>
       $LISTA_CONTAS_UTILIZADORES
35 printf "\nOutput:\n" >> $LISTA_CONTAS_UTILIZADORES
36 cat /etc/passwd >> $LISTA_CONTAS_UTILIZADORES
37 #echo "-> Lista de contas de utilizadores - Concluído"
38 echo concluido
39
40
41 #Histórico do login do utilizador
42 printf "Histórico do login do utilizador\n\n" >>
       $HISTORICO_LOGIN_UTILIZADOR
43 printf "Comando: last\n" >> $HISTORICO_LOGIN_UTILIZADOR
```

```
44 printf "\nOutput:\n" >> $HISTORICO_LOGIN_UTILIZADOR
45 last >> $HISTORICO_LOGIN_UTILIZADOR
46 #echo "-> Histórico do login do utilizador - Concluído"
47 echo concluido
```

Listagem A.27: *Script* de Utilizador Admin macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  UTILIZADOR=$PCNAME/Utilizador
8
9
10 #Nomes Ficheiros
11 LISTA_TAREFAS_AGENDADAS=$UTILIZADOR/lista_tarefas_agendadas.txt
12 LISTA_CONTAS_UTILIZADORES=$UTILIZADOR/lista_contas_utilizadores.
       txt
13 HISTORICO_LOGIN_UTILIZADOR=$UTILIZADOR/
       historico_login_utilizador.txt
14
15
16
17 ###Utilizador
18
19 #Lista de tarefas agendadas
20 printf "Lista de tarefas agendadas\n\n" >>
       $LISTA_TAREFAS_AGENDADAS
21 printf "Comando: sudo crontab -l\n" >> $LISTA_TAREFAS_AGENDADAS
22 printf "\nOutput:\n" >> $LISTA_TAREFAS_AGENDADAS
23 echo $PASS | sudo -S crontab -l >> $LISTA_TAREFAS_AGENDADAS
24 #echo "-> Lista de tarefas agendadas - Concluído"
25 echo concluido
26
27
28 #Lista de contas de utilizadores
29 printf "Lista de contas de utilizadores\n\n" >>
       $LISTA_CONTAS_UTILIZADORES
30 printf "Comando: sudo cat /etc/passwd\n" >>
       $LISTA_CONTAS_UTILIZADORES
31 printf "\nOutput:\n" >> $LISTA_CONTAS_UTILIZADORES
32 echo $PASS | sudo -S cat /etc/passwd >>
       $LISTA_CONTAS_UTILIZADORES
33 #echo "-> Lista de contas de utilizadores - Concluído"
34 echo concluido
35
36
37 #Histórico do login do utilizador
38 printf "Histórico do login do utilizador\n\n" >>
       $HISTORICO_LOGIN_UTILIZADOR
39 printf "Comando: sudo last\n" >> $HISTORICO_LOGIN_UTILIZADOR
40 printf "\nOutput:\n" >> $HISTORICO_LOGIN_UTILIZADOR
```

```
41 echo $PASS | sudo -S last >> $HISTORICO_LOGIN_UTILIZADOR
42 #echo "-> Histórico do login do utilizador - Concluído"
43 echo concluido
```

Listagem A.28: *Script* de Utilizador Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set PASS=%2
7  set UTILIZADOR=%PCNAME%\Utilizador
8  set TOOLS=tools\windows
9
10
11 ::Nomes Ficheiros
12 set LISTA_TAREFAS_AGENDADAS=%UTILIZADOR%\lista_tarefas_agendadas
       .txt
13 set LISTA_CONTAS_UTILIZADORES=%UTILIZADOR%\
       lista_contas_utilizadores.txt
14 set HISTORICO_LOGIN_UTILIZADOR=%UTILIZADOR%\
       historico_login_utilizador.txt
15
16
17
18 ::Lista de tarefas agendadas
19 echo Lista de tarefas agendadas >> %LISTA_TAREFAS_AGENDADAS%
20 echo. >> %LISTA_TAREFAS_AGENDADAS%
21 echo Comando: schtasks /query /fo LIST /v >> %
       LISTA_TAREFAS_AGENDADAS%
22 echo. >> %LISTA_TAREFAS_AGENDADAS%
23 echo Output: >> %LISTA_TAREFAS_AGENDADAS%
24 schtasks /query /fo LIST /v >> %LISTA_TAREFAS_AGENDADAS%
25 ::echo "-> Lista de tarefas agendadas - Concluído"
26 echo concluido
27
28
29 ::Lista de contas de utilizadores
30 echo Lista de contas de utilizadores >> %
       LISTA_CONTAS_UTILIZADORES%
31 echo. >> %LISTA_CONTAS_UTILIZADORES%
32 echo Comando: dumpsec.exe >> %LISTA_CONTAS_UTILIZADORES%
33 echo. >> %LISTA_CONTAS_UTILIZADORES%
34 echo Output: >> %LISTA_CONTAS_UTILIZADORES%
35 %TOOLS%\dumpsec.exe /rpt=users /saveas=fixed /outfile=x.txt
36 type x.txt >> %LISTA_CONTAS_UTILIZADORES%
37 del x.txt
38 ::echo "-> Lista de contas de utilizadores - Concluído"
39 echo concluido
40
41
42 ::Histórico do login do utilizador
43 echo Histórico do login do utilizador >> %
```

```
    HISTORICO_LOGIN_UTILIZADOR%
44 echo. >> %HISTORICO_LOGIN_UTILIZADOR%
45 echo Comando: >> %HISTORICO_LOGIN_UTILIZADOR%
46 echo logonsessions.exe /accepteula >> %
    HISTORICO_LOGIN_UTILIZADOR%
47 echo NetUsers.exe /h /v >> %HISTORICO_LOGIN_UTILIZADOR%
48 echo. >> %HISTORICO_LOGIN_UTILIZADOR%
49 echo Output: >> %HISTORICO_LOGIN_UTILIZADOR%
50 echo Local: >> %HISTORICO_LOGIN_UTILIZADOR%
51 %TOOLS%\logonsessions.exe /accepteula >> %
    HISTORICO_LOGIN_UTILIZADOR%
52 echo. >> %HISTORICO_LOGIN_UTILIZADOR%
53 %TOOLS%\NetUsers.exe /h /v >> %HISTORICO_LOGIN_UTILIZADOR%
54 echo concluido
55 ::echo "-> Histórico do login do utilizador - Concluído"
```

Listagem A.29: *Script* de Utilizador Admin Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set UTILIZADOR=%PCNAME%\Utilizador
7  set TOOLS=tools\windows
8
9
10 ::Nomes Ficheiros
11 set LISTA_TAREFAS_AGENDADAS=%UTILIZADOR%\lista_tarefas_agendadas
       .txt
12 set LISTA_CONTAS_UTILIZADORES=%UTILIZADOR%\
       lista_contas_utilizadores.txt
13 set HISTORICO_LOGIN_UTILIZADOR=%UTILIZADOR%\
       historico_login_utilizador.txt
14
15
16
17 ::::::Utilizador
18
19 ::Lista de tarefas agendadas
20 echo Lista de tarefas agendadas >> %LISTA_TAREFAS_AGENDADAS%
21 echo. >> %LISTA_TAREFAS_AGENDADAS%
22 echo Comando: schtasks /query /fo LIST /v >> %
       LISTA_TAREFAS_AGENDADAS%
23 echo. >> %LISTA_TAREFAS_AGENDADAS%
24 echo Output: >> %LISTA_TAREFAS_AGENDADAS%
25 schtasks /query /fo LIST /v >> %LISTA_TAREFAS_AGENDADAS%
26 ::echo "-> Lista de tarefas agendadas - Concluído"
27 echo concluido
28
29
30 ::Lista de contas de utilizadores
31 echo Lista de contas de utilizadores >> %
       LISTA_CONTAS_UTILIZADORES%
```

```
32 echo. >> %LISTA_CONTAS_UTILIZADORES%
33 echo Comando: dumpsec.exe >> %LISTA_CONTAS_UTILIZADORES%
34 echo. >> %LISTA_CONTAS_UTILIZADORES%
35 echo Output: >> %LISTA_CONTAS_UTILIZADORES%
36 %TOOLS%\dumpsec.exe /rpt=users /saveas=fixed /outfile=x.txt
37 type x1.txt >> %LISTA_CONTAS_UTILIZADORES%
38 del x1.txt
39 ::echo "-> Lista de contas de utilizadores - Concluído"
40 echo concluido
41
42
43 ::Histórico do login do utilizador
44 echo Histórico do login do utilizador >> %
       HISTORICO_LOGIN_UTILIZADOR%
45 echo. >> %HISTORICO_LOGIN_UTILIZADOR%
46 echo Comando: >> %HISTORICO_LOGIN_UTILIZADOR%
47 echo logonsessions.exe /accepteula >> %
       HISTORICO_LOGIN_UTILIZADOR%
48 echo NetUsers.exe /h /v >> %HISTORICO_LOGIN_UTILIZADOR%
49 echo. >> %HISTORICO_LOGIN_UTILIZADOR%
50 echo Output: >> %HISTORICO_LOGIN_UTILIZADOR%
51 echo Local: >> %HISTORICO_LOGIN_UTILIZADOR%
52 %TOOLS%\logonsessions.exe /accepteula >> %
       HISTORICO_LOGIN_UTILIZADOR%
53 echo. >> %HISTORICO_LOGIN_UTILIZADOR%
54 %TOOLS%\NetUsers.exe /h /v >> %HISTORICO_LOGIN_UTILIZADOR%
55 echo concluido
56 ::echo "-> Histórico do login do utilizador - Concluído"
```

A.7 *Scripts* de Dispositivos

Listagem A.30: *Script* de Dispositivos Linux

```
1  #!/bin/bash
2  #Nome Pastas e caminhos
3  PCNAME=$1
4  PASS=$2
5  DISPOSITIVOS=$PCNAME/Dispositivos
6  #Nomes Ficheiros
7  DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/dispositivos_montados.txt
8  INFORMACOES_DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/
       informacoes_dispositivos_montados.txt
9
10 ###Dispositivos
11 #Dispositivos montados
12 printf "Dispositivos montados\n\n" >> $DISPOSITIVOS_MONTADOS
13 printf "Comando: mount\n" >> $DISPOSITIVOS_MONTADOS
14 printf "\nOutput:\n" >> $DISPOSITIVOS_MONTADOS
15 mount >> $DISPOSITIVOS_MONTADOS 2> /dev/null
16 #echo "-> Dispositivos montados - Concluído"
17 echo concluido
18
```

```
19 #Informações dos dispositivos montados
20 printf "Informações dos dispositivos montados\n\n" >>
       $INFORMACOES_DISPOSITIVOS_MONTADOS
21 printf "Comando: df -k\n" >> $INFORMACOES_DISPOSITIVOS_MONTADOS
22 printf "\nOutput:\n" >> $INFORMACOES_DISPOSITIVOS_MONTADOS
23 df -k >> $INFORMACOES_DISPOSITIVOS_MONTADOS 2> /dev/null
24 #echo "-> Informações gerais do sistema - Concluído"
25 echo concluido
```

Listagem A.31: *Script* de Dispositivos Admin Linux

```
1  #!/bin/bash
2  #Nome Pastas e caminhos
3  PCNAME=$1
4  PASS=$2
5  DISPOSITIVOS=$PCNAME/Dispositivos
6  #Nomes Ficheiros
7  DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/dispositivos_montados.txt
8  INFORMACOES_DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/
       informacoes_dispositivos_montados.txt
9
10 ###Dispositivos
11 #Dispositivos montados
12 printf "Dispositivos montados\n\n" >> $DISPOSITIVOS_MONTADOS
13 printf "Comando: sudo mount\n" >> $DISPOSITIVOS_MONTADOS
14 printf "\nOutput:\n" >> $DISPOSITIVOS_MONTADOS
15 echo $PASS | sudo -S mount >> $DISPOSITIVOS_MONTADOS
16 #echo "-> Dispositivos montados - Concluído"
17 echo concluido
18
19 #Informações dos dispositivos montados
20 printf "Informações dos dispositivos montados\n\n" >>
       $INFORMACOES_DISPOSITIVOS_MONTADOS
21 printf "Comando: sudo df -k\n" >>
       $INFORMACOES_DISPOSITIVOS_MONTADOS
22 printf "\nOutput:\n" >> $INFORMACOES_DISPOSITIVOS_MONTADOS
23 echo $PASS | sudo -S df -k >> $INFORMACOES_DISPOSITIVOS_MONTADOS
24 #echo "-> Informações gerais do sistema - Concluído"
25 echo concluido
```

Listagem A.32: *Script* de Dispositivos macOS

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  DISPOSITIVOS=$PCNAME/Dispositivos
7
8  #Nomes Ficheiros
9  DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/dispositivos_montados.txt
10 INFORMACOES_DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/
       informacoes_dispositivos_montados.txt
11
```

```
12 ###Dispositivos
13 #Drivers carregados
14 printf "Dispositivos montados\n\n" >> $DISPOSITIVOS_MONTADOS
15 printf "Comando: mount\n" >> $DISPOSITIVOS_MONTADOS
16 printf "\nOutput:\n" >> $DISPOSITIVOS_MONTADOS
17 mount >> $DISPOSITIVOS_MONTADOS
18 #echo "-> Drivers carregados - Concluído"
19 echo concluido
20
21 #Informações dos dispositivos montados
22 printf "Informações dos dispositivos montados\n\n" >>
       $INFORMACOES_DISPOSITIVOS_MONTADOS
23 printf "Comando: df -k\n" >> $INFORMACOES_DISPOSITIVOS_MONTADOS
24 printf "\nOutput:\n" >> $INFORMACOES_DISPOSITIVOS_MONTADOS
25 df -k >> $INFORMACOES_DISPOSITIVOS_MONTADOS
26 #echo "-> Informações gerais do sistema - Concluído"
27 echo concluido
```

Listagem A.33: *Script* de Dispositivos Admin macOS

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  DISPOSITIVOS=$PCNAME/Dispositivos
7
8  #Nomes Ficheiros
9  DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/dispositivos_montados.txt
10 INFORMACOES_DISPOSITIVOS_MONTADOS=$DISPOSITIVOS/
       informacoes_dispositivos_montados.txt
11
12 ###Dispositivos
13 #Drivers carregados
14 printf "Dispositivos montados\n\n" >> $DISPOSITIVOS_MONTADOS
15 printf "Comando: sudo mount\n" >> $DISPOSITIVOS_MONTADOS
16 printf "\nOutput:\n" >> $DISPOSITIVOS_MONTADOS
17 echo $PASS | sudo -S mount >> $DISPOSITIVOS_MONTADOS
18 #echo "-> Drivers carregados - Concluído"
19 echo concluido
20
21 #Informações dos dispositivos montados
22 printf "Informações dos dispositivos montados\n\n" >>
       $INFORMACOES_DISPOSITIVOS_MONTADOS
23 printf "Comando: sudo df -k\n" >>
       $INFORMACOES_DISPOSITIVOS_MONTADOS
24 printf "\nOutput:\n" >> $INFORMACOES_DISPOSITIVOS_MONTADOS
25 echo $PASS | sudo -S df -k >> $INFORMACOES_DISPOSITIVOS_MONTADOS
26 #echo "-> Informações gerais do sistema - Concluído"
27 echo concluido
```

Listagem A.34: *Script* de Dispositivos Windows

```
1  @echo off
2
3  ::Nome Pastas e caminhos
4  set PCNAME=%1
5  set DISPOSITIVOS=%PCNAME%\Dispositivos
6  set TOOLS=tools\windows
7
8
9  ::Nomes Ficheiros
10 set DISPOSITIVOS_MONTADOS=%DISPOSITIVOS%\dispositivos_montados.
       txt
11 set INFORMACOES_DISPOSITIVOS_MONTADOS=%DISPOSITIVOS%\
       informacoes_dispositivos_montados.txt
12
13
14
15 ::Dispositivos Montados
16 echo Dispositivos montados >> %DISPOSITIVOS_MONTADOS%
17 echo. >> %DISPOSITIVOS_MONTADOS%
18 echo Comando: wmic.exe diskdrive list brief /format:list >> %
       DISPOSITIVOS_MONTADOS%
19 echo. >> %DISPOSITIVOS_MONTADOS%
20 echo Output: >> %DISPOSITIVOS_MONTADOS%
21 wmic /Output:"x.txt" diskdrive list brief /format:list
22 type x.txt >> %DISPOSITIVOS_MONTADOS%
23 del x.txt
24 ::echo "-> Dispositivos montados - Concluído"
25 echo concluido
26
27
28 ::Informações dos dispositivos montados
29 echo Informações dos dispositivos montados >> %
       INFORMACOES_DISPOSITIVOS_MONTADOS%
30 echo. >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
31 echo Comando: di.exe >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
32 echo. >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
33 echo Output: >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
34 %TOOLS%\di.exe >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
35 ::echo "-> Informações gerais do sistema - Concluído"
36 echo concluido
```

Listagem A.35: *Script* de Dispositivos Admin Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set DISPOSITIVOS=%PCNAME%\Dispositivos
7  set TOOLS=tools\windows
8
9
10 ::Nomes Ficheiros
11 set DISPOSITIVOS_MONTADOS=%DISPOSITIVOS%\dispositivos_montados.
```

```
    txt
12  set INFORMACOES_DISPOSITIVOS_MONTADOS=%DISPOSITIVOS%\
    informacoes_dispositivos_montados.txt
13
14
15
16  :::::: Dispositivos
17
18  :: Dispositivos Montados
19  echo Dispositivos montados >> %DISPOSITIVOS_MONTADOS%
20  echo. >> %DISPOSITIVOS_MONTADOS%
21  echo Comando: wmic.exe diskdrive list brief /format:list >> %
    DISPOSITIVOS_MONTADOS%
22  echo. >> %DISPOSITIVOS_MONTADOS%
23  echo Output: >> %DISPOSITIVOS_MONTADOS%
24  wmic.exe /Output:"x.txt" diskdrive list brief /format:list
25  type x.txt >> %DISPOSITIVOS_MONTADOS%
26  del x.txt
27  ::echo "-> Dispositivos montados - Concluído"
28  echo concluido
29
30
31  ::Informações dos dispositivos montados
32  echo Informações dos dispositivos montados >> %
    INFORMACOES_DISPOSITIVOS_MONTADOS%
33  echo. >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
34  echo Comando: di.exe >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
35  echo. >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
36  echo Output: >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
37  %TOOLS%\di.exe >> %INFORMACOES_DISPOSITIVOS_MONTADOS%
38  ::echo "-> Informações gerais do sistema - Concluído"
39  echo concluido
```

A.8 *Scripts* de Logs

Listagem A.36: *Script* de Logs Linux

```
1   #!/bin/bash
2
3   #Nome Pastas e caminhos
4   PCNAME=$1
5   PASS=$2
6   LOGS=$PCNAME/Logs
7
8   #Nomes Ficheiros
9   LOGS_SISTEMA=$LOGS/logs_sistema.txt
10
11  ###Logs
12  #Logs do sistema
13  printf "Logs do sistema\n\n" >> $LOGS_SISTEMA
14  printf "Comando: cp /var/log/*.log*\n" >> $LOGS_SISTEMA
15  printf "\nOutput:\n" >> $LOGS_SISTEMA
```

```
16 cp /var/log/*.log* $LOGS
17 printf "Ficheiros movidos para a pasta Logs\n" >> $LOGS_SISTEMA
18 #echo "-> Logs do sistema - Concluído"
19 echo concluido
```

Listagem A.37: *Script* de Logs Admin Linux

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  LOGS=$PCNAME/Logs
7
8  #Nomes Ficheiros
9  LOGS_SISTEMA=$LOGS/logs_sistema.txt
10
11 ###Logs
12 #Logs do sistema
13 printf "Logs do sistema\n\n" >> $LOGS_SISTEMA
14 printf "Comando: sudo cp /var/log/*.log*\n" >> $LOGS_SISTEMA
15 printf "\nOutput:\n" >> $LOGS_SISTEMA
16 echo $PASS | sudo -S cp /var/log/*.log* $LOGS
17 #wait $!
18 printf "Ficheiros movidos para a pasta Logs\n" >> $LOGS_SISTEMA
19 #echo "-> Logs do sistema - Concluído"
20 echo concluido
```

Listagem A.38: *Script* de Logs macOS

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  LOGS=$PCNAME/Logs
7
8  #Nomes Ficheiros
9  LOGS_SISTEMA=$LOGS/logs_sistema.txt
10
11 ###Logs
12 #Logs do sistema
13 printf "Logs do sistema\n\n" >> $LOGS_SISTEMA
14 printf "Comando: \n cp /var/log/*.log* \n cp -r /etc/cron* \n"
       >> $LOGS_SISTEMA
15 printf "\nOutput:\n" >> $LOGS_SISTEMA
16 cp /var/log/*.log* $LOGS
17 wait $!
18 cp -r /etc/cron* $LOGS
19 printf "Ficheiros movidos para a pasta Logs\n" >> $LOGS_SISTEMA
20 #echo "-> Logs do sistema - Concluído"
21 echo concluido
```

Listagem A.39: *Script* de Logs Admin macOS

```
#!/bin/bash

#Nome Pastas e caminhos
PCNAME=$1
PASS=$2
LOGS=$PCNAME/Logs

#Nomes Ficheiros
LOGS_SISTEMA=$LOGS/logs_sistema.txt

###Logs
#Logs do sistema
printf "Logs do sistema\n\n" >> $LOGS_SISTEMA
printf "Comando: \n sudo cp /var/log/*.log* \n sudo cp -r /etc/cron* \n" >> $LOGS_SISTEMA
printf "\nOutput:\n" >> $LOGS_SISTEMA
echo $PASS | sudo -S cp /var/log/*.log* $LOGS
wait $!
echo $PASS | sudo -S cp -r /etc/cron* $LOGS
#wait $!
printf "Ficheiros movidos para a pasta Logs\n" >> $LOGS_SISTEMA
#echo "-> Logs do sistema - Concluído"
echo concluido
```

Listagem A.40: *Script* de Logs Windows

```
@echo off
::Nome Pastas e caminhos
set PCNAME=%1
set PASS=%2
set LOGS=%PCNAME%\Logs
set TOOLS=tools\windows
::Nomes Ficheiros
set LOGS_SISTEMA=%LOGS%\logs_sistema.txt

::::::Logs
::Logs do sistema
echo Logs do sistema >> %LOGS_SISTEMA%
echo. >> %LOGS_SISTEMA%
echo Comando: >> %LOGS_SISTEMA%
echo psloglist.exe /accepteula >> %LOGS_SISTEMA%
echo RawCopy.exe %WINDIR%\System32\winevt\Logs\Application.evtx >> %LOGS_SISTEMA%
echo. >> %LOGS_SISTEMA%
echo Output: >> %LOGS_SISTEMA%
%TOOLS%\psloglist.exe /accepteula >> %LOGS_SISTEMA%
echo. >> %LOGS_SISTEMA%
echo Ficheiros só podem ser movidos para a pasta Logs como Administrador >> %LOGS_SISTEMA%
::echo "-> Logs do sistema - Concluído"
echo concluido
```

Listagem A.41: *Script* de Logs Admin Windows

```
@echo off
::Nome Pastas e caminhos
set PCNAME=%1
set LOGS=%PCNAME%\Logs
set TOOLS=tools\windows
::Nomes Ficheiros
set LOGS_SISTEMA=%LOGS%\logs_sistema.txt

::::::Logs
::Logs do sistema
echo Logs do sistema >> %LOGS_SISTEMA%
echo. >> %LOGS_SISTEMA%
echo Comando: >> %LOGS_SISTEMA%
echo psloglist.exe /accepteula >> %LOGS_SISTEMA%
echo RawCopy.exe %WINDIR%\System32\winevt\Logs\Application.evtx
    >> %LOGS_SISTEMA%
echo. >> %LOGS_SISTEMA%
echo Output: >> %LOGS_SISTEMA%
%TOOLS%\psloglist.exe /accepteula >> %LOGS_SISTEMA%
echo. >> %LOGS_SISTEMA%
%TOOLS%\RawCopy.exe %WINDIR%\System32\winevt\Logs\Application.
    evtx %LOGS%
echo Ficheiros movidos para a pasta Logs >> %LOGS_SISTEMA%
::echo "-> Logs do sistema - Concluído"
echo concluido
```

A.9 *Scripts* de Integridade

Listagem A.42: *Script* de Integridade Linux

```
#!/bin/bash


#Nome Pastas e caminhos
PCNAME=$1
PASS=$2
SELECIONADO=$3
HASH=$4
DUMP=$6
INTEGRIDADE=$PCNAME/Integridade


#Nomes Ficheiros
CALCULO_SHA256_FICHEIROS=$INTEGRIDADE/calculo_sha256_ficheiros.
    txt


###Integridade

#Cálculo SHA256 Comandos
```

```
20 sh scripts/linux/integridadeComandos.sh $PCNAME
21
22 echo concluido
23
24
25 #Cálculo Hashes disco
26 if $SELECIONADO; then
27 #printf "hashes.sh" > tmp.txt
28 sh scripts/linux/hashes.sh $PCNAME $PASS $HASH
29 fi
30
31 echo concluido
32
33
34 #Dump memória RAM
35 if $DUMP; then
36 sh scripts/linux/dump.sh $PCNAME $PASS
37 fi
38
39 echo concluido
40
41
42 #Cálculo SHA256 Ficheiros
43 printf "Cálculo SHA256 Ficheiros\n\n" >>
       $CALCULO_SHA256_FICHEIROS
44 printf "Comando: sha256sum ficheiros\n" >>
       $CALCULO_SHA256_FICHEIROS
45 printf "\nOutput:\n" >> $CALCULO_SHA256_FICHEIROS
46 find $PCNAME -type f -exec sha256sum {} \; >>
       $CALCULO_SHA256_FICHEIROS 2> /dev/null
47 #echo "-> Cálculo SHA256 - Concluído"
48 echo concluido
```

Listagem A.43: *Script* de Integridade Admin Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SELECIONADO=$3
8  HASH=$4
9  DUMP=$6
10 INTEGRIDADE=$PCNAME/Integridade
11
12
13 #Nomes Ficheiros
14 CALCULO_SHA256_FICHEIROS=$INTEGRIDADE/calculo_sha256_ficheiros.
       txt
15
16
17 ###Integridade
18
```

```
19 #Cálculo SHA256 Comandos
20 sh scripts/linux/integridadeComandosAdmin.sh $PCNAME $PASS
21
22 echo concluido
23
24
25 #Cálculo Hashes disco
26 if $SELECIONADO; then
27 #printf "hashesAdmin.sh" > tmp.txt
28 sh scripts/linux/hashesAdmin.sh $PCNAME $PASS $HASH
29 fi
30
31 echo concluido
32
33
34 #Dump memória RAM
35 if $DUMP; then
36 sh scripts/linux/dumpAdmin.sh $PCNAME $PASS
37 fi
38
39 echo concluido
40
41
42 #Cálculo SHA256 Ficheiros
43 printf "Cálculo SHA256 Ficheiros\n\n" >>
       $CALCULO_SHA256_FICHEIROS
44 printf "Comando: sudo sha256sum ficheiros\n" >>
       $CALCULO_SHA256_FICHEIROS
45 printf "\nOutput:\n" >> $CALCULO_SHA256_FICHEIROS
46 echo $PASS | sudo -S find $PCNAME -type f -exec sha256sum {} \;
       >> $CALCULO_SHA256_FICHEIROS
47 #echo "-> Cálculo SHA256 - Concluído"
48 echo concluido
```

Listagem A.44: *Script* de Integridade Dump Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  INTEGRIDADE=$PCNAME/Integridade
7  TOOLS=tools/linux
8
9
10 #Nomes Ficheiros
11 DUMP_RAM=$INTEGRIDADE/dump_ram.txt
12 FILE=$TOOLS/LiME-master.zip
13
14
15 ###Integridade
16
17 #Dump Memória RAM
18 printf "Dump da memória RAM\n\n" >> $DUMP_RAM
```

```
19 printf "Comando: insmod\n" >> $DUMP_RAM
20 printf "\nOutput:\n" >> $DUMP_RAM
21 printf "Só disponível como root!" >> $DUMP_RAM
```

Listagem A.45: *Script* de Integridade Dump Admin Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  INTEGRIDADE=$PCNAME/Integridade
8  TOOLS=tools/linux
9  KERNEL=`uname -r`
10
11
12 #Nomes Ficheiros
13 FILE=$TOOLS/LiME-master.zip
14 NOME_MEM_DUMP=$INTEGRIDADE/mem.dump
15
16
17 ###Integridade
18
19
20 if  which rpm >/dev/null 2>/dev/null; then
21
22 if $DISTRO_LINUX | grep -i "opensuse" >> /dev/null; then
23 echo $PASS | sudo -S zypper update
24 echo $PASS | sudo -S zypper install kernel-source
25
26 elif $DISTRO_LINUX | grep -i "mageia" >> /dev/null; then
27 echo $PASS | sudo -S urpmi.update -a
28 echo $PASS | sudo -S urpmi kernel-devel
29
30 elif $DISTRO_LINUX | grep -i "korora" >> /dev/null; then
31 echo $PASS | sudo -S dnf update
32 echo $PASS | sudo -S dnf install kernel-headers-$KERNEL
33
34 elif $DISTRO_LINUX | grep -i "fedora" >> /dev/null; then
35 echo $PASS | sudo -S dnf update
36 echo $PASS | sudo -S dnf install kernel-headers-$KERNEL
37
38 elif $DISTRO_LINUX | grep -i "centos" >> /dev/null; then
39 echo $PASS | sudo -S yum update
40 echo $PASS | sudo -S yum install kernel-headers-$KERNEL
41 fi
42
43 elif  which dpkg >/dev/null 2>/dev/null ; then
44
45 echo $PASS | sudo -S apt-get update
46 echo $PASS | sudo -S apt-get install linux-headers-$KERNEL
47
48 elif which pacman >/dev/null 2>/dev/null ; then
```

```
49
50 if $KERNEL | grep -i "arch" >> /dev/null; then
51 echo $PASS | sudo -S pacman -Syu
52 echo $PASS | sudo -S pacman -S linux-headers
53
54 elif $KERNEL | grep -i "manjaro" >> /dev/null; then
55 echo $PASS | sudo -S pacman -Syu
56 versao=$(echo $KERNEL | awk -F "." '{print $1,$2}')
57 set -- $versao
58 X=$1
59 Y=$2
60 echo $PASS | sudo -S pacman -S linuxXY-headers
61 fi
62
63 elif which slpkg >/dev/null 2>/dev/null ; then
64
65 echo $PASS | sudo -S slackpkg install kernel-headers
66 fi
67
68 #Dump Memória RAM
69 echo $PASS | sudo -S rmmod lime
70 unzip $FILE
71 cd LiME-master/src/
72 make
73 echo $PASS | sudo -S insmod lime-*.ko "path=../../$NOME_MEM_DUMP
       format=lime"
74 cd ..
75 cd ..
76 echo $PASS | sudo -S rm -R LiME-master/
```

Listagem A.46: *Script* de Integridade Hashes Linux

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  HASH=$3
7  INTEGRIDADE=$PCNAME/Integridade
8
9
10 #Nomes Ficheiros
11 HASHES_DISCO=$INTEGRIDADE/hashes_disco.txt
12
13
14 ###Integridade
15
16 if [ "$HASH" = "MD5" ]; then
17 #Cálculo MD5 Ficheiros Disco
18 printf "Cálculo MD5 Ficheiros Disco\n\n" >> $HASHES_DISCO
19 printf "Comando: md5sum ficheiros\n" >> $HASHES_DISCO
20 printf "\nOutput:\n" >> $HASHES_DISCO
21 find / -type f -exec md5sum {} \; >> $HASHES_DISCO 2> /dev/null
22
```

```
23 elif [ "$HASH" = "SHA1" ]; then
24 #Cálculo SHA1 Ficheiros Disco
25 printf "Cálculo SHA1 Ficheiros Disco\n\n" >> $HASHES_DISCO
26 printf "Comando: sha1sum ficheiros\n" >> $HASHES_DISCO
27 printf "\nOutput:\n" >> $HASHES_DISCO
28 find / -type f -exec sha1sum {} \; >> $HASHES_DISCO 2> /dev/null
29
30 elif [ "$HASH" = "SHA256" ]; then
31 #Cálculo SHA256 Ficheiros Disco
32 printf "Cálculo SHA256 Ficheiros Disco\n\n" >> $HASHES_DISCO
33 printf "Comando: sha256sum ficheiros\n" >> $HASHES_DISCO
34 printf "\nOutput:\n" >> $HASHES_DISCO
35 find / -type f -exec sha256sum {} \; >> $HASHES_DISCO 2> /dev/
       null
36
37 else
38 printf "Comando: \n\n" >> $HASHES_DISCO
39 printf "Output:\n" >> $HASHES_DISCO
40 printf "Não disponivel" >> $HASHES_DISCO
41 fi
```

Listagem A.47: *Script* de Integridade Hashes Admin Linux

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  HASH=$3
7  INTEGRIDADE=$PCNAME/Integridade
8
9
10 #Nomes Ficheiros
11 HASHES_DISCO=$INTEGRIDADE/hashes_disco.txt
12
13
14 ###Integridade
15
16 if [ "$HASH" = "MD5" ]; then
17 #Cálculo MD5 Ficheiros Disco
18 printf "Cálculo MD5 Ficheiros Disco\n\n" >> $HASHES_DISCO
19 printf "Comando: md5sum ficheiros\n" >> $HASHES_DISCO
20 printf "\nOutput:\n" >> $HASHES_DISCO
21 echo $PASS | sudo -S find / -type f -exec md5sum \{\} \; >>
       $HASHES_DISCO 2> /dev/null
22
23 elif [ "$HASH" = "SHA1" ]; then
24 #Cálculo SHA1 Ficheiros Disco
25 printf "Cálculo SHA1 Ficheiros Disco\n\n" >> $HASHES_DISCO
26 printf "Comando: sha1sum ficheiros\n" >> $HASHES_DISCO
27 printf "\nOutput:\n" >> $HASHES_DISCO
28 echo $PASS | sudo -S find / -type f -exec sha1sum \{\} \; >>
       $HASHES_DISCO 2> /dev/null
29
```

```
30 elif [ "$HASH" = "SHA256" ]; then
31 #Cálculo SHA256 Ficheiros Disco
32 printf "Cálculo SHA256 Ficheiros Disco\n\n" >> $HASHES_DISCO
33 printf "Comando: sha256sum ficheiros\n" >> $HASHES_DISCO
34 printf "\nOutput:\n" >> $HASHES_DISCO
35 echo $PASS | sudo -S find / -type f -exec sha256sum \{\} \; >>
       $HASHES_DISCO 2> /dev/null
36
37 else
38 printf "Comando: \n\n" >> $HASHES_DISCO
39 printf "Output:\n" >> $HASHES_DISCO
40 printf "Não disponivel" >> $HASHES_DISCO
41 fi
```

Listagem A.48: *Script* de Integridade Comandos Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  INTEGRIDADE=$PCNAME/Integridade
7
8
9  #Nomes Ficheiros
10 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
11
12
13 ###Integridade
14
15 #Cálculo SHA256 Comandos
16 printf "Cálculo SHA256 Comandos\n\n" >> $CALCULO_SHA256_COMANDOS
17
18 #bash
19 printf "Comando: bash\n" >> $CALCULO_SHA256_COMANDOS
20 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
21 bash --version >> $CALCULO_SHA256_COMANDOS
22 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
23 sha256sum $(echo `whereis -b bash` | head -n 1 | cut -d " " -f
       2) >> $CALCULO_SHA256_COMANDOS
24 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
25
26
27 #mkdir
28 printf "Comando: mkdir\n" >> $CALCULO_SHA256_COMANDOS
29 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
30 mkdir --version >> $CALCULO_SHA256_COMANDOS
31 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
32 sha256sum $(echo `whereis -b mkdir` | head -n 1 | cut -d " " -f
       2) >> $CALCULO_SHA256_COMANDOS
33 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
34
```

```
35
36 #ifconfig
37 printf "Comando: ifconfig\n" >> $CALCULO_SHA256_COMANDOS
38 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
39 ifconfig --version 2>> $CALCULO_SHA256_COMANDOS
40 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
41 sha256sum $(echo `whereis -b ifconfig` | head -n 1 | cut -d " "
       -f 2) >> $CALCULO_SHA256_COMANDOS
42 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
43
44
45 #netstat
46 printf "Comando: netstat\n" >> $CALCULO_SHA256_COMANDOS
47 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
48 netstat --version >> $CALCULO_SHA256_COMANDOS
49 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
50 sha256sum $(echo `whereis -b netstat` | head -n 1 | cut -d " " -
       f 2) >> $CALCULO_SHA256_COMANDOS
51 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
52
53
54 #arp
55 printf "Comando: arp\n" >> $CALCULO_SHA256_COMANDOS
56 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
57 arp --version 2>> $CALCULO_SHA256_COMANDOS
58 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
59 sha256sum $(echo `whereis -b arp` | head -n 1 | cut -d " " -f 2)
        >> $CALCULO_SHA256_COMANDOS
60 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
61
62
63 #cat
64 printf "Comando: cat\n" >> $CALCULO_SHA256_COMANDOS
65 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
66 cat --version >> $CALCULO_SHA256_COMANDOS
67 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
68 sha256sum $(echo `whereis -b cat` | head -n 1 | cut -d " " -f 2)
        >> $CALCULO_SHA256_COMANDOS
69 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
70
71
72 #lsof
73 printf "Comando: lsof\n" >> $CALCULO_SHA256_COMANDOS
74 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
75 lsof -v 2>> $CALCULO_SHA256_COMANDOS
76 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
77 sha256sum $(echo `whereis -b lsof` | head -n 1 | cut -d " " -f
       2) >> $CALCULO_SHA256_COMANDOS
78 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
```

```
79
80
81 #top
82 printf "Comando: top\n" >> $CALCULO_SHA256_COMANDOS
83 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
84 top -v >> $CALCULO_SHA256_COMANDOS
85 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
86 sha256sum $(echo `whereis -b top` | head -n 1 | cut -d " " -f 2)
       >> $CALCULO_SHA256_COMANDOS
87 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
88
89
90 #ls
91 printf "Comando: ls\n" >> $CALCULO_SHA256_COMANDOS
92 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
93 ls --version >> $CALCULO_SHA256_COMANDOS
94 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
95 sha256sum $(echo `whereis -b ls` | head -n 1 | cut -d " " -f 2)
       >> $CALCULO_SHA256_COMANDOS
96 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
97
98
99 #date
100 printf "Comando: date\n" >> $CALCULO_SHA256_COMANDOS
101 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
102 date --version >> $CALCULO_SHA256_COMANDOS
103 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
104 sha256sum $(echo `whereis -b date` | head -n 1 | cut -d " " -f
       2) >> $CALCULO_SHA256_COMANDOS
105 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
106
107
108 #uname
109 printf "Comando: uname\n" >> $CALCULO_SHA256_COMANDOS
110 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
111 uname --version >> $CALCULO_SHA256_COMANDOS
112 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
113 sha256sum $(echo `whereis -b uname` | head -n 1 | cut -d " " -f
       2) >> $CALCULO_SHA256_COMANDOS
114 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
115
116
117 #last
118 printf "Comando: last\n" >> $CALCULO_SHA256_COMANDOS
119 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
120 last --version >> $CALCULO_SHA256_COMANDOS
121 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
122 sha256sum $(echo `whereis -b last` | head -n 1 | cut -d " " -f
       2) >> $CALCULO_SHA256_COMANDOS
123 printf "\n-----------------------------------\n\n" >>
```

```
    $CALCULO_SHA256_COMANDOS
124
125
126 #mount
127 printf "Comando: mount\n" >> $CALCULO_SHA256_COMANDOS
128 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
129 mount --version >> $CALCULO_SHA256_COMANDOS
130 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
131 sha256sum $(echo `whereis -b mount` | head -n 1 | cut -d " " -f
        2) >> $CALCULO_SHA256_COMANDOS
132 printf "\n-----------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
133
134
135 #df
136 printf "Comando: df\n" >> $CALCULO_SHA256_COMANDOS
137 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
138 df --version >> $CALCULO_SHA256_COMANDOS
139 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
140 sha256sum $(echo `whereis -b df` | head -n 1 | cut -d " " -f 2)
        >> $CALCULO_SHA256_COMANDOS
141 printf "\n-----------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
142
143
144 #cp
145 printf "Comando: cp\n" >> $CALCULO_SHA256_COMANDOS
146 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
147 cp --version >> $CALCULO_SHA256_COMANDOS
148 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
149 sha256sum $(echo `whereis -b cp` | head -n 1 | cut -d " " -f 2)
        >> $CALCULO_SHA256_COMANDOS
150 printf "\n-----------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
151
152
153 #find
154 printf "Comando: find\n" >> $CALCULO_SHA256_COMANDOS
155 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
156 find --version >> $CALCULO_SHA256_COMANDOS
157 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
158 sha256sum $(echo `whereis -b find` | head -n 1 | cut -d " " -f
        2) >> $CALCULO_SHA256_COMANDOS
159 printf "\n-----------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
160
161
162 #sha256sum
163 printf "Comando: sha256sum\n" >> $CALCULO_SHA256_COMANDOS
164 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
165 sha256sum --version >> $CALCULO_SHA256_COMANDOS
166 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
167 sha256sum $(echo `whereis -b sha256sum` | head -n 1 | cut -d " "
        -f 2) >> $CALCULO_SHA256_COMANDOS
```

```
168 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
169
170
171 #rpm || dpkg || pacman || slpkg || qlist
172 if  which rpm >/dev/null 2>/dev/null; then
173 printf "Comando: rpm\n" >> $CALCULO_SHA256_COMANDOS
174 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
175 rpm --version >> $CALCULO_SHA256_COMANDOS
176 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
177 sha256sum $(echo `whereis -b rpm` | head -n 1 | cut -d " " -f 2)
        >> $CALCULO_SHA256_COMANDOS
178 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
179
180 elif  which dpkg >/dev/null 2>/dev/null ; then
181 printf "Comando: dpkg\n" >> $CALCULO_SHA256_COMANDOS
182 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
183 dpkg --version >> $CALCULO_SHA256_COMANDOS
184 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
185 sha256sum $(echo `whereis -b dpkg` | head -n 1 | cut -d " " -f
        2) >> $CALCULO_SHA256_COMANDOS
186 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
187
188 elif which pacman >/dev/null 2>/dev/null ; then
189 printf "Comando: pacman\n" >> $CALCULO_SHA256_COMANDOS
190 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
191 pacman --version >> $CALCULO_SHA256_COMANDOS
192 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
193 sha256sum $(echo `whereis -b pacman` | head -n 1 | cut -d " " -f
        2) >> $CALCULO_SHA256_COMANDOS
194 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
195
196 elif which slpkg >/dev/null 2>/dev/null ; then
197 printf "Comando: slpkg\n" >> $CALCULO_SHA256_COMANDOS
198 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
199 slpkg --version >> $CALCULO_SHA256_COMANDOS
200 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
201 sha256sum $(echo `whereis -b slpkg` | head -n 1 | cut -d " " -f
        2) >> $CALCULO_SHA256_COMANDOS
202 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
203
204 elif which emerge >/dev/null 2>/dev/null ; then
205 printf "Comando: qlist\n" >> $CALCULO_SHA256_COMANDOS
206 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
207 qlist --version >> $CALCULO_SHA256_COMANDOS
208 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
209 sha256sum $(echo `whereis -b qlist` | head -n 1 | cut -d " " -f
        2) >> $CALCULO_SHA256_COMANDOS
210 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
```

```
211
212 else
213 printf "Comando: \n\n" >> $LISTA_SOFTWARE_INSTALADO
214 printf "Output:\n" >> $LISTA_SOFTWARE_INSTALADO
215 printf "Não disponivel" >> $LISTA_SOFTWARE_INSTALADO
216 fi
217
218
219 #crontab
220 printf "Comando: crontab\n" >> $CALCULO_SHA256_COMANDOS
221 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
222
223
224 if  which rpm >/dev/null 2>/dev/null; then
225 rpm -qa | grep crontab >> $CALCULO_SHA256_COMANDOS
226 elif  which dpkg >/dev/null 2>/dev/null ; then
227 dpkg -l | grep crontab>> $CALCULO_SHA256_COMANDOS
228 elif which pacman >/dev/null 2>/dev/null ; then
229 pacman -Q | grep crontab >> $CALCULO_SHA256_COMANDOS
230 elif which slpkg >/dev/null 2>/dev/null ; then
231 slpkg -l sbo --installed | grep crontab >>
    $CALCULO_SHA256_COMANDOS
232 elif which emerge >/dev/null 2>/dev/null ; then
233 qlist -IC | grep crontab >> $CALCULO_SHA256_COMANDOS
234 else
235 printf "Não disponivel" >> $CALCULO_SHA256_COMANDOS
236 fi
237
238
239 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
240 sha256sum $(echo `whereis -b crontab` | head -n 1 | cut -d " " -
    f 2) >> $CALCULO_SHA256_COMANDOS
241 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
```

Listagem A.49: *Script* de Integridade Comandos Admin Linux

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  INTEGRIDADE=$PCNAME/Integridade
8
9
10 #Nomes Ficheiros
11 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
12
13
14 ###Integridade
15
16 #Cálculo SHA256 Comandos
17 printf "Cálculo SHA256 Comandos\n\n" >> $CALCULO_SHA256_COMANDOS
```

```
18
19 #bash
20 printf "Comando: bash\n" >> $CALCULO_SHA256_COMANDOS
21 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
22 bash --version >> $CALCULO_SHA256_COMANDOS
23 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
24 echo $PASS | sudo -S sha256sum $(echo `whereis -b bash` | head -
       n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
25 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
26
27
28 #mkdir
29 printf "Comando: mkdir\n" >> $CALCULO_SHA256_COMANDOS
30 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
31 mkdir --version >> $CALCULO_SHA256_COMANDOS
32 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
33 echo $PASS | sudo -S sha256sum $(echo `whereis -b mkdir` | head
       -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
34 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
35
36
37 #ifconfig
38 printf "Comando: ifconfig\n" >> $CALCULO_SHA256_COMANDOS
39 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
40 ifconfig --version 2>> $CALCULO_SHA256_COMANDOS
41 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
42 echo $PASS | sudo -S sha256sum $(echo `whereis -b ifconfig` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
43 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
44
45
46 #netstat
47 printf "Comando: netstat\n" >> $CALCULO_SHA256_COMANDOS
48 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
49 netstat --version >> $CALCULO_SHA256_COMANDOS
50 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
51 echo $PASS | sudo -S sha256sum $(echo `whereis -b netstat` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
52 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
53
54
55 #arp
56 printf "Comando: arp\n" >> $CALCULO_SHA256_COMANDOS
57 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
58 arp --version 2>> $CALCULO_SHA256_COMANDOS
59 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
60 echo $PASS | sudo -S sha256sum $(echo `whereis -b arp` | head -n
       1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
61 printf "\n----------\n\n" >>
       $CALCULO_SHA256_COMANDOS
```

```
62
63
64 #cat
65 printf "Comando: cat\n" >> $CALCULO_SHA256_COMANDOS
66 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
67 cat --version >> $CALCULO_SHA256_COMANDOS
68 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
69 echo $PASS | sudo -S sha256sum $(echo `whereis -b cat` | head -n
       1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
70 printf "\n----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
71
72
73 #lsof
74 printf "Comando: lsof\n" >> $CALCULO_SHA256_COMANDOS
75 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
76 lsof -v 2>> $CALCULO_SHA256_COMANDOS
77 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
78 echo $PASS | sudo -S sha256sum $(echo `whereis -b lsof` | head -
       n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
79 printf "\n----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
80
81
82 #top
83 printf "Comando: top\n" >> $CALCULO_SHA256_COMANDOS
84 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
85 top -v >> $CALCULO_SHA256_COMANDOS
86 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
87 echo $PASS | sudo -S sha256sum $(echo `whereis -b top` | head -n
       1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
88 printf "\n----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
89
90
91 #ls
92 printf "Comando: ls\n" >> $CALCULO_SHA256_COMANDOS
93 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
94 ls --version >> $CALCULO_SHA256_COMANDOS
95 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
96 echo $PASS | sudo -S sha256sum $(echo `whereis -b ls` | head -n
       1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
97 printf "\n----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
98
99
100 #date
101 printf "Comando: date\n" >> $CALCULO_SHA256_COMANDOS
102 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
103 date --version >> $CALCULO_SHA256_COMANDOS
104 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
105 echo $PASS | sudo -S sha256sum $(echo `whereis -b date` | head -
       n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
106 printf "\n----------------------------------\n\n" >>
```

```
    $CALCULO_SHA256_COMANDOS
107
108
109 #uname
110 printf "Comando: uname\n" >> $CALCULO_SHA256_COMANDOS
111 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
112 uname --version >> $CALCULO_SHA256_COMANDOS
113 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
114 echo $PASS | sudo -S sha256sum $(echo `whereis -b uname` | head
    -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
115 printf "\n------------------------------------\n\n" >>
    $CALCULO_SHA256_COMANDOS
116
117
118 #last
119 printf "Comando: last\n" >> $CALCULO_SHA256_COMANDOS
120 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
121 last --version >> $CALCULO_SHA256_COMANDOS
122 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
123 echo $PASS | sudo -S sha256sum $(echo `whereis -b last` | head -
    n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
124 printf "\n------------------------------------\n\n" >>
    $CALCULO_SHA256_COMANDOS
125
126
127 #mount
128 printf "Comando: mount\n" >> $CALCULO_SHA256_COMANDOS
129 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
130 mount --version >> $CALCULO_SHA256_COMANDOS
131 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
132 echo $PASS | sudo -S sha256sum $(echo `whereis -b mount` | head
    -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
133 printf "\n------------------------------------\n\n" >>
    $CALCULO_SHA256_COMANDOS
134
135
136 #df
137 printf "Comando: df\n" >> $CALCULO_SHA256_COMANDOS
138 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
139 df --version >> $CALCULO_SHA256_COMANDOS
140 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
141 echo $PASS | sudo -S sha256sum $(echo `whereis -b df` | head -n
    1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
142 printf "\n------------------------------------\n\n" >>
    $CALCULO_SHA256_COMANDOS
143
144
145 #cp
146 printf "Comando: cp\n" >> $CALCULO_SHA256_COMANDOS
147 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
148 cp --version >> $CALCULO_SHA256_COMANDOS
149 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
150 echo $PASS | sudo -S sha256sum $(echo `whereis -b cp` | head -n
    1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
```

```
151 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
152
153
154 #find
155 printf "Comando: find\n" >> $CALCULO_SHA256_COMANDOS
156 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
157 find --version >> $CALCULO_SHA256_COMANDOS
158 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
159 echo $PASS | sudo -S sha256sum $(echo `whereis -b find` | head -
        n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
160 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
161
162
163 #sha256sum
164 printf "Comando: sha256sum\n" >> $CALCULO_SHA256_COMANDOS
165 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
166 sha256sum --version >> $CALCULO_SHA256_COMANDOS
167 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
168 echo $PASS | sudo -S sha256sum $(echo `whereis -b sha256sum` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
169 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
170
171
172 #rpm || dpkg || pacman || slpkg || qlist
173 if  which rpm >/dev/null 2>/dev/null; then
174 printf "Comando: rpm\n" >> $CALCULO_SHA256_COMANDOS
175 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
176 rpm --version >> $CALCULO_SHA256_COMANDOS
177 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
178 echo $PASS | sudo -S sha256sum $(echo `whereis -b rpm` | head -n
         1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
179 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
180
181 elif  which dpkg >/dev/null 2>/dev/null ; then
182 printf "Comando: dpkg\n" >> $CALCULO_SHA256_COMANDOS
183 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
184 dpkg --version >> $CALCULO_SHA256_COMANDOS
185 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
186 echo $PASS | sudo -S sha256sum $(echo `whereis -b dpkg` | head -
        n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
187 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
188
189 elif which pacman >/dev/null 2>/dev/null ; then
190 printf "Comando: pacman\n" >> $CALCULO_SHA256_COMANDOS
191 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
192 pacman --version >> $CALCULO_SHA256_COMANDOS
193 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
194 echo $PASS | sudo -S sha256sum $(echo `whereis -b pacman` | head
        -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
```

```
195 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
196
197 elif which slpkg >/dev/null 2>/dev/null ; then
198 printf "Comando: slpkg\n" >> $CALCULO_SHA256_COMANDOS
199 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
200 slpkg --version >> $CALCULO_SHA256_COMANDOS
201 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
202 echo $PASS | sudo -S sha256sum $(echo `whereis -b slpkg` | head
        -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
203 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
204
205 elif which emerge >/dev/null 2>/dev/null ; then
206 printf "Comando: qlist\n" >> $CALCULO_SHA256_COMANDOS
207 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
208 qlist --version >> $CALCULO_SHA256_COMANDOS
209 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
210 echo $PASS | sudo -S sha256sum $(echo `whereis -b qlist` | head
        -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
211 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
212
213 else
214 printf "Comando: \n\n" >> $LISTA_SOFTWARE_INSTALADO
215 printf "Output:\n" >> $LISTA_SOFTWARE_INSTALADO
216 printf "Não disponivel" >> $LISTA_SOFTWARE_INSTALADO
217 fi
218
219
220 #crontab
221 printf "Comando: crontab\n" >> $CALCULO_SHA256_COMANDOS
222 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
223
224
225 if  which rpm >/dev/null 2>/dev/null; then
226 echo $PASS | sudo -S rpm -qa | grep crontab >>
        $CALCULO_SHA256_COMANDOS
227 elif  which dpkg >/dev/null 2>/dev/null ; then
228 echo $PASS | sudo -S dpkg -l | grep crontab>>
        $CALCULO_SHA256_COMANDOS
229 elif which pacman >/dev/null 2>/dev/null ; then
230 echo $PASS | sudo -S pacman -Q | grep crontab >>
        $CALCULO_SHA256_COMANDOS
231 elif which slpkg >/dev/null 2>/dev/null ; then
232 echo $PASS | sudo -S slpkg -l sbo --installed | grep crontab >>
        $CALCULO_SHA256_COMANDOS
233 elif which emerge >/dev/null 2>/dev/null ; then
234 echo $PASS | sudo -S qlist -IC | grep crontab >>
        $CALCULO_SHA256_COMANDOS
235 else
236 printf "Não disponivel" >> $CALCULO_SHA256_COMANDOS
237 fi
238
```

```
239
240 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
241 echo $PASS | sudo -S sha256sum $(echo `whereis -b crontab` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
242 printf "\n----------\n\n" >>
        $CALCULO_SHA256_COMANDOS
```

Listagem A.50: *Script* de Integridade macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SELECIONADO=$3
8  HASH=$4
9  DUMP=$6
10 INTEGRIDADE=$PCNAME/Integridade
11
12
13 #Nomes Ficheiros
14 CALCULO_SHA256_FICHEIROS=$INTEGRIDADE/calculo_sha256_ficheiros.
       txt
15
16
17
18 ###Integridade
19
20 #Cálculo SHA256 Comandos
21 sh scripts/mac/integridadeComandos.sh $PCNAME
22
23 echo concluido
24
25
26 #Cálculo Hashes disco
27 if $SELECIONADO; then
28 sh scripts/mac/hashes.sh $PCNAME $PASS $HASH
29 fi
30
31 echo concluido
32
33
34 #Dump memória RAM
35 if $DUMP; then
36 sh scripts/mac/dump.sh $PCNAME $PASS
37 fi
38
39 echo concluido
40
41
42 #Cálculo SHA256 Ficheiros
43 printf "Cálculo SHA256 Ficheiros\n\n" >>
       $CALCULO_SHA256_FICHEIROS
```

```
44 printf "Comando: sha256sum ficheiros\n" >>
       $CALCULO_SHA256_FICHEIROS
45 printf "\nOutput:\n" >> $CALCULO_SHA256_FICHEIROS
46 find $PCNAME -type f -exec shasum -a 256 {} \; >>
       $CALCULO_SHA256_FICHEIROS
47 #echo "-> Cálculo SHA256 - Concluído"
48 echo concluido
```

Listagem A.51: *Script* de Integridade Admin macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  SELECIONADO=$3
8  HASH=$4
9  DUMP=$5
10 INTEGRIDADE=$PCNAME/Integridade
11
12
13 #Nomes Ficheiros
14 CALCULO_SHA256_FICHEIROS=$INTEGRIDADE/calculo_sha256_ficheiros.
       txt
15
16
17
18 ###Integridade
19
20 #Cálculo SHA256 Comandos
21 sh scripts/mac/integridadeComandosAdmin.sh $PCNAME
22
23 echo concluido
24
25
26 #Cálculo Hashes disco
27 if $SELECIONADO; then
28 sh scripts/mac/hashesAdmin.sh $PCNAME $PASS $HASH
29 fi
30
31 echo concluido
32
33
34 #Dump memória RAM
35 if $DUMP; then
36 sh scripts/mac/dumpAdmin.sh $PCNAME $PASS
37 fi
38
39 echo concluido
40
41
42 #Cálculo SHA256 Ficheiros
43 printf "Cálculo SHA256 Ficheiros\n\n" >>
```

```
    $CALCULO_SHA256_FICHEIROS
44 printf "Comando: sudo sha256sum ficheiros\n" >>
    $CALCULO_SHA256_FICHEIROS
45 printf "\nOutput:\n" >> $CALCULO_SHA256_FICHEIROS
46 echo $PASS | sudo -S find $PCNAME -type f -exec shasum -a 256 {}
     \; >> $CALCULO_SHA256_FICHEIROS
47 #echo "-> Cálculo SHA256 - Concluído"
48 echo concluido
```

Listagem A.52: *Script* de Integridade Dump macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  INTEGRIDADE=$PCNAME/Integridade
7  TOOLS=tools/mac
8
9
10 #Nomes Ficheiros
11 DUMP_RAM=$INTEGRIDADE/dump_ram.txt
12 FILE=$TOOLS/OSXPMem.tar.gz
13
14
15 ###Integridade
16
17 #Dump Memória RAM
18 printf "Dump da memória RAM\n\n" >> $DUMP_RAM
19 printf "Comando: osxpmem\n" >> $DUMP_RAM
20 printf "\nOutput:\n" >> $DUMP_RAM
21 printf "Só disponível como root!" >> $DUMP_RAM
```

Listagem A.53: *Script* de Integridade Dump Admin macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  INTEGRIDADE=$PCNAME/Integridade
8  TOOLS=tools/mac
9
10
11 #Nomes Ficheiros
12 FILE=$TOOLS/OSXPMem.tar.gz
13 NOME_MEM_DUMP=$INTEGRIDADE/mem.dump
14
15
16 ###Integridade
17
18 #Dump Memória RAM
19 echo $PASS | sudo -S tar xvf $FILE
```

```
20 cd OSXPMem/
21 echo $PASS | sudo -S ./osxpmem $NOME_MEM_DUMP
22 cd ..
23 echo $PASS | sudo -S rm -R OSXPMem/
```

Listagem A.54: *Script* de Integridade Hashes macOS

```
1  #!/bin/bash
2
3  #Nome Pastas e caminhos
4  PCNAME=$1
5  PASS=$2
6  HASH=$3
7  INTEGRIDADE=$PCNAME/Integridade
8
9
10 #Nomes Ficheiros
11 HASHES_DISCO=$INTEGRIDADE/hashes_disco.txt
12
13
14
15 if [ "$HASH" = "MD5" ]; then
16 #Cálculo MD5 Ficheiros Disco
17 printf "Cálculo MD5 Ficheiros Disco\n\n" >> $HASHES_DISCO
18 printf "Comando: md5sum ficheiros\n" >> $HASHES_DISCO
19 printf "\nOutput:\n" >> $HASHES_DISCO
20 find / -type f -exec md5sum {} \; >> $HASHES_DISCO 2> /dev/null
21
22 elif [ "$HASH" = "SHA1" ]; then
23 #Cálculo SHA1 Ficheiros Disco
24 printf "Cálculo SHA1 Ficheiros Disco\n\n" >> $HASHES_DISCO
25 printf "Comando: sha1sum ficheiros\n" >> $HASHES_DISCO
26 printf "\nOutput:\n" >> $HASHES_DISCO
27 find / -type f -exec sha1sum {} \; >> $HASHES_DISCO 2> /dev/null
28
29 elif [ "$HASH" = "SHA256" ]; then
30 #Cálculo SHA256 Ficheiros Disco
31 printf "Cálculo SHA256 Ficheiros Disco\n\n" >> $HASHES_DISCO
32 printf "Comando: sha256sum ficheiros\n" >> $HASHES_DISCO
33 printf "\nOutput:\n" >> $HASHES_DISCO
34 find / -type f -exec shasum -a 256 {} \; >> $HASHES_DISCO 2> /
       dev/null
35
36 else
37 printf "Comando: \n\n" >> $HASHES_DISCO
38 printf "Output:\n" >> $HASHES_DISCO
39 printf "Não disponivel" >> $HASHES_DISCO
40 fi
```

Listagem A.55: *Script* de Integridade Hashes Admin macOS

```
1 #!/bin/bash
2
3 #Nome Pastas e caminhos
```

```
4  PCNAME=$1
5  PASS=$2
6  HASH=$3
7  INTEGRIDADE=$PCNAME/Integridade
8
9
10 #Nomes Ficheiros
11 HASHES_DISCO=$INTEGRIDADE/hashes_disco.txt
12
13
14
15 if [ "$HASH" = "MD5" ]; then
16 #Cálculo MD5 Ficheiros Disco
17 printf "Cálculo MD5 Ficheiros Disco\n\n" >> $HASHES_DISCO
18 printf "Comando: md5sum ficheiros\n" >> $HASHES_DISCO
19 printf "\nOutput:\n" >> $HASHES_DISCO
20 echo $PASS | sudo -S find / -type f -exec md5sum {} \; >>
       $HASHES_DISCO 2> /dev/null
21
22 elif [ "$HASH" = "SHA1" ]; then
23 #Cálculo SHA1 Ficheiros Disco
24 printf "Cálculo SHA1 Ficheiros Disco\n\n" >> $HASHES_DISCO
25 printf "Comando: sha1sum ficheiros\n" >> $HASHES_DISCO
26 printf "\nOutput:\n" >> $HASHES_DISCO
27 echo $PASS | sudo -S find / -type f -exec sha1sum {} \; >>
       $HASHES_DISCO 2> /dev/null
28
29 elif [ "$HASH" = "SHA256" ]; then
30 #Cálculo SHA256 Ficheiros Disco
31 printf "Cálculo SHA256 Ficheiros Disco\n\n" >> $HASHES_DISCO
32 printf "Comando: sha256sum ficheiros\n" >> $HASHES_DISCO
33 printf "\nOutput:\n" >> $HASHES_DISCO
34 echo $PASS | sudo -S find / -type f -exec shasum -a 256 {} \; >>
        $HASHES_DISCO 2> /dev/null
35
36 else
37 printf "Comando: \n\n" >> $HASHES_DISCO
38 printf "Output:\n" >> $HASHES_DISCO
39 printf "Não disponivel" >> $HASHES_DISCO
40 fi
```

Listagem A.56: *Script* de Integridade Comandos macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  INTEGRIDADE=$PCNAME/Integridade
7
8
9  #Nomes Ficheiros
10 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
11
```

```
12
13 ###Integridade
14
15 #Cálculo SHA256 Comandos
16 printf "Cálculo SHA256 Comandos\n\n" >> $CALCULO_SHA256_COMANDOS
17
18 #bash
19 printf "Comando: bash\n" >> $CALCULO_SHA256_COMANDOS
20 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
21 bash --version >> $CALCULO_SHA256_COMANDOS
22 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
23 shasum -a 256 $(echo `whereis -b bash` | head -n 1 | cut -d " "
       -f 2) >> $CALCULO_SHA256_COMANDOS
24 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
25
26
27 #mkdir
28 printf "Comando: mkdir\n" >> $CALCULO_SHA256_COMANDOS
29 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
30 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
31 shasum -a 256 $(echo `whereis -b mkdir` | head -n 1 | cut -d " "
        -f 2) >> $CALCULO_SHA256_COMANDOS
32 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
33
34
35 #ifconfig
36 printf "Comando: ifconfig\n" >> $CALCULO_SHA256_COMANDOS
37 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
38 ifconfig --version >> $CALCULO_SHA256_COMANDOS
39 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
40 shasum -a 256 $(echo `whereis -b ifconfig` | head -n 1 | cut -d
       " " -f 2) >> $CALCULO_SHA256_COMANDOS
41 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
42
43
44 #netstat
45 printf "Comando: netstat\n" >> $TABELA_ROTEAMENTO
46 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
47 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
48 shasum -a 256 $(echo `whereis -b netstat` | head -n 1 | cut -d "
        " -f 2) >> $CALCULO_SHA256_COMANDOS
49 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
50
51
52 #arp
53 printf "Comando: arp\n" >> $CALCULO_SHA256_COMANDOS
54 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
55 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
56 shasum -a 256 $(echo `whereis -b arp` | head -n 1 | cut -d " " -
       f 2) >> $CALCULO_SHA256_COMANDOS
```

```
57 printf "\n----------\n\n" >> $CALCULO_SHA256_COMANDOS
58
59
60 #scutil
61 printf "Comando: scutil\n" >> $CALCULO_SHA256_COMANDOS
62 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
63 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
64 shasum -a 256 $(echo `whereis -b scutil` | head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
65 printf "\n----------\n\n" >> $CALCULO_SHA256_COMANDOS
66
67
68 #lsof
69 printf "Comando: lsof\n" >> $FICHEIROS_ABERTOS
70 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
71 lsof -v >> $CALCULO_SHA256_COMANDOS
72 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
73 shasum -a 256 $(echo `whereis -b lsof` | head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
74 printf "\n----------\n\n" >> $CALCULO_SHA256_COMANDOS
75
76
77 #ps
78 printf "Comando: ps\n" >> $CALCULO_SHA256_COMANDOS
79 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
80 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
81 shasum -a 256 $(echo `whereis -b ps` | head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
82 printf "\n----------\n\n" >> $CALCULO_SHA256_COMANDOS
83
84
85 #ls
86 printf "Comando: ls\n" >> $CALCULO_SHA256_COMANDOS
87 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
88 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
89 shasum -a 256 $(echo `whereis -b ls` | head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
90 printf "\n----------\n\n" >> $CALCULO_SHA256_COMANDOS
91
92
93 #systemsetup
94 printf "Comando: systemsetup\n" >> $CALCULO_SHA256_COMANDOS
95 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
96 systemsetup -version >> $CALCULO_SHA256_COMANDOS
97 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
98 shasum -a 256 $(echo `whereis -b systemsetup` | head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
99 printf "\n----------\n\n" >> $CALCULO_SHA256_COMANDOS
```

```
100
101
102 #uname
103 printf "Comando: uname\n" >> $CALCULO_SHA256_COMANDOS
104 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
105 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
106 shasum -a 256 $(echo `whereis -b uname` | head -n 1 | cut -d " "
      -f 2) >> $CALCULO_SHA256_COMANDOS
107 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
108
109
110 #find
111 printf "Comando: find\n" >> $CALCULO_SHA256_COMANDOS
112 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
113 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
114 shasum -a 256 $(echo `whereis -b find` | head -n 1 | cut -d " "
    -f 2) >> $CALCULO_SHA256_COMANDOS
115 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
116
117
118 #crontab
119 printf "Comando: crontab\n" >> $CALCULO_SHA256_COMANDOS
120 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
121 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
122 shasum -a 256 $(echo `whereis -b crontab` | head -n 1 | cut -d "
     " -f 2) >> $CALCULO_SHA256_COMANDOS
123 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
124
125
126 #cat
127 printf "Comando: cat\n" >> $CACHE_DNS
128 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
129 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
130 shasum -a 256 $(echo `whereis -b cat` | head -n 1 | cut -d " " -
    f 2) >> $CALCULO_SHA256_COMANDOS
131 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
132
133
134 #last
135 printf "Comando: last\n" >> $CALCULO_SHA256_COMANDOS
136 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
137 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
138 shasum -a 256 $(echo `whereis -b last` | head -n 1 | cut -d " "
    -f 2) >> $CALCULO_SHA256_COMANDOS
139 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
140
141
142 #mount
143 printf "Comando: mount\n" >> $CALCULO_SHA256_COMANDOS
```

```
144 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
145 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
146 shasum -a 256 $(echo `whereis -b mount` | head -n 1 | cut -d " "
      -f 2) >> $CALCULO_SHA256_COMANDOS
147 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
148
149
150 #df
151 printf "Comando: df\n" >> $CALCULO_SHA256_COMANDOS
152 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
153 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
154 shasum -a 256 $(echo `whereis -b df` | head -n 1 | cut -d " " -f
      2) >> $CALCULO_SHA256_COMANDOS
155 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
156
157
158 #cp
159 printf "Comando: cp\n" >> $CALCULO_SHA256_COMANDOS
160 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
161 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
162 shasum -a 256 $(echo `whereis -b cp` | head -n 1 | cut -d " " -f
      2) >> $CALCULO_SHA256_COMANDOS
163 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
164
165
166 #shasum
167 printf "Comando: shasum\n" >> $CALCULO_SHA256_COMANDOS
168 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
169 shasum --version >> $CALCULO_SHA256_COMANDOS
170 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
171 shasum -a 256 $(echo `whereis -b shasum` | head -n 1 | cut -d "
    " -f 2) >> $CALCULO_SHA256_COMANDOS
172 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
```

Listagem A.57: *Script* de Integridade Comandos Admin macOS

```
1  #!/bin/bash
2
3
4  #Nome Pastas e caminhos
5  PCNAME=$1
6  PASS=$2
7  INTEGRIDADE=$PCNAME/Integridade
8
9
10 #Nomes Ficheiros
11 CALCULO_SHA256_COMANDOS=$INTEGRIDADE/calculo_sha256_comandos.txt
12
13
14 ###Integridade
```

```
15
16 #Cálculo SHA256 Comandos
17 printf "Cálculo SHA256 Comandos\n\n" >> $CALCULO_SHA256_COMANDOS
18
19 #bash
20 printf "Comando: bash\n" >> $CALCULO_SHA256_COMANDOS
21 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
22 echo $PASS | sudo -S bash --version >> $CALCULO_SHA256_COMANDOS
23 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
24 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b bash` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
25 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
26
27
28 #mkdir
29 printf "Comando: mkdir\n" >> $CALCULO_SHA256_COMANDOS
30 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
31 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
32 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b mkdir` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
33 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
34
35
36 #ifconfig
37 printf "Comando: ifconfig\n" >> $CALCULO_SHA256_COMANDOS
38 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
39 ifconfig --version >> $CALCULO_SHA256_COMANDOS
40 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
41 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b ifconfig`
       | head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
42 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
43
44
45 #netstat
46 printf "Comando: netstat\n" >> $TABELA_ROTEAMENTO
47 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
48 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
49 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b netstat` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
50 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
51
52
53 #arp
54 printf "Comando: arp\n" >> $CALCULO_SHA256_COMANDOS
55 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
56 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
57 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b arp` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
58 printf "\n-----------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
```

```
59
60
61 #scutil
62 printf "Comando: scutil\n" >> $CALCULO_SHA256_COMANDOS
63 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
64 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
65 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b scutil` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
66 printf "\n--------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
67
68
69 #lsof
70 printf "Comando: lsof\n" >> $FICHEIROS_ABERTOS
71 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
72 lsof -v >> $CALCULO_SHA256_COMANDOS
73 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
74 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b lsof` |
       head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
75 printf "\n--------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
76
77
78 #ps
79 printf "Comando: ps\n" >> $CALCULO_SHA256_COMANDOS
80 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
81 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
82 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b ps` | head
       -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
83 printf "\n--------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
84
85
86 #ls
87 printf "Comando: ls\n" >> $CALCULO_SHA256_COMANDOS
88 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
89 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
90 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b ls` | head
       -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
91 printf "\n--------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
92
93
94 #systemsetup
95 printf "Comando: systemsetup\n" >> $CALCULO_SHA256_COMANDOS
96 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
97 systemsetup -version >> $CALCULO_SHA256_COMANDOS
98 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
99 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b
       systemsetup` | head -n 1 | cut -d " " -f 2) >>
       $CALCULO_SHA256_COMANDOS
100 printf "\n--------------------------------\n\n" >>
       $CALCULO_SHA256_COMANDOS
101
```

```
102
103 #uname
104 printf "Comando: uname\n" >> $CALCULO_SHA256_COMANDOS
105 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
106 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
107 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b uname` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
108 printf "\n-------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
109
110
111 #find
112 printf "Comando: find\n" >> $CALCULO_SHA256_COMANDOS
113 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
114 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
115 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b find` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
116 printf "\n-------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
117
118
119 #crontab
120 printf "Comando: crontab\n" >> $CALCULO_SHA256_COMANDOS
121 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
122 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
123 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b crontab` |
         head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
124 printf "\n-------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
125
126
127 #cat
128 printf "Comando: cat\n" >> $CACHE_DNS
129 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
130 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
131 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b cat` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
132 printf "\n-------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
133
134
135 #last
136 printf "Comando: last\n" >> $CALCULO_SHA256_COMANDOS
137 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
138 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
139 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b last` |
        head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
140 printf "\n-------------------------------\n\n" >>
        $CALCULO_SHA256_COMANDOS
141
142
143 #mount
144 printf "Comando: mount\n" >> $CALCULO_SHA256_COMANDOS
145 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
```

```
146 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
147 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b mount` |
    head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
148 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
149
150
151 #df
152 printf "Comando: df\n" >> $CALCULO_SHA256_COMANDOS
153 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
154 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
155 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b df` | head
     -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
156 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
157
158
159 #cp
160 printf "Comando: cp\n" >> $CALCULO_SHA256_COMANDOS
161 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
162 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
163 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b cp` | head
     -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
164 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
165
166
167 #shasum
168 printf "Comando: shasum\n" >> $CALCULO_SHA256_COMANDOS
169 printf "Versão:\n" >> $CALCULO_SHA256_COMANDOS
170 shasum --version >> $CALCULO_SHA256_COMANDOS
171 printf "\nSHA256 Comando: " >> $CALCULO_SHA256_COMANDOS
172 echo $PASS | sudo -S shasum -a 256 $(echo `whereis -b shasum` |
    head -n 1 | cut -d " " -f 2) >> $CALCULO_SHA256_COMANDOS
173 printf "\n----------\n\n" >>
    $CALCULO_SHA256_COMANDOS
```

Listagem A.58: *Script* de Integridade Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set PASS=%2
7  set SELECIONADO=%3
8  set HASH=%4
9  set DUMP=%6
10 set INTEGRIDADE=%PCNAME%\Integridade
11 set TOOLS=tools\windows
12
13
14 ::Nomes Ficheiros
15 set CALCULO_SHA256_FICHEIROS=%INTEGRIDADE%\
```

```
    calculo_sha256_ficheiros.txt
16
17
18
19 ::Cálculo SHA256 Comandos
20 cmd /c scripts\windows\integridadeComandos.bat %PCNAME%
21
22 echo concluido
23
24
25 ::Cálculo Hashes disco
26 if "%SELECIONADO%" == "true" (
27 cmd /c scripts\windows\hashes.bat %PCNAME% %HASH%
28 )
29
30 echo concluido
31
32
33 ::Dump memória RAM
34 if "%DUMP%" == "true" (
35 cmd /c scripts\windows\dump.bat %PCNAME%
36 )
37
38 echo concluido
39
40
41 ::Cálculo SHA256 Ficheiros
42 echo Cálculo SHA256 Ficheiros >> %CALCULO_SHA256_FICHEIROS%
43 echo. >> %CALCULO_SHA256_FICHEIROS%
44 echo Comando: sha256deep.exe >> %CALCULO_SHA256_FICHEIROS%
45 echo. >> %CALCULO_SHA256_FICHEIROS%
46 echo Output: >> %CALCULO_SHA256_FICHEIROS%
47 %TOOLS%\sha256deep.exe -r %PCNAME%\* >> %
    CALCULO_SHA256_FICHEIROS%
48 ::echo "-> Cálculo SHA256 - Concluído"
49 echo concluido
```

Listagem A.59: *Script* de Integridade Admin Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set SELECIONADO=%3
7  set HASH=%4
8  set DUMP=%6
9  set INTEGRIDADE=%PCNAME%\Integridade
10 set TOOLS=tools\windows
11
12
13 ::Nomes Ficheiros
14 set CALCULO_SHA256_FICHEIROS=%INTEGRIDADE%\
    calculo_sha256_ficheiros.txt
```

```
15
16
17
18 ::::::Integridade
19
20 ::Cálculo SHA256 Comandos
21 cmd /c scripts\windows\integridadeComandosAdmin.bat %PCNAME%
22
23 echo concluido
24
25
26 ::Cálculo Hashes disco
27 if "%SELECIONADO%" == "true" (
28 cmd /c scripts\windows\hashesAdmin.bat %PCNAME% %HASH%
29 )
30
31 echo concluido
32
33
34 ::Dump memória RAM
35 if "%DUMP%" == "true" (
36 cmd /c scripts\windows\dumpAdmin.bat %PCNAME%
37 )
38
39 echo concluido
40
41
42 ::Cálculo SHA256
43 echo Cálculo SHA256 Ficheiros >> %CALCULO_SHA256_FICHEIROS%
44 echo. >> %CALCULO_SHA256_FICHEIROS%
45 echo Comando: sha256deep.exe >> %CALCULO_SHA256_FICHEIROS%
46 echo. >> %CALCULO_SHA256_FICHEIROS%
47 echo Output: >> %CALCULO_SHA256_FICHEIROS%
48 %TOOLS%\sha256deep.exe -r %PCNAME%\* >> %
       CALCULO_SHA256_FICHEIROS%
49 ::echo "-> Cálculo SHA256 - Concluído"
50 echo concluido
```

Listagem A.60: *Script* de Integridade Dump Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set INTEGRIDADE=%PCNAME%\Integridade
7  set TOOLS=tools\windows
8
9
10 ::Nomes Ficheiros
11 set DUMP_RAM=%INTEGRIDADE%\dump_ram.txt
12
13 ::Dump memória RAM
14 echo Dump da memória RAM >> %DUMP_RAM%
```

```
15 echo. >> %DUMP_RAM%
16 echo Comando: winpmem_2.1.exe >> %DUMP_RAM%
17 echo. >> %DUMP_RAM%
18 echo Output: >> %DUMP_RAM%
19 echo. >> %DUMP_RAM%
20 echo Só disponível como Admin! >> %DUMP_RAM%
```

Listagem A.61: *Script* de Integridade Dump Admin Windows

```
1  @echo off
2
3  :: BatchGotAdmin
4  :-------------------------------------
5  REM  --> Check for permissions
6  IF "%PROCESSOR_ARCHITECTURE%" EQU "amd64" (
7  >nul 2>&1 "%SYSTEMROOT%\SysWOW64\cacls.exe" "%SYSTEMROOT%\
       SysWOW64\config\system"
8  ) ELSE (
9  >nul 2>&1 "%SYSTEMROOT%\system32\cacls.exe" "%SYSTEMROOT%\
       system32\config\system"
10 )
11
12 REM --> If error flag set, we do not have admin.
13 if '%errorlevel%' NEQ '0' (
14 echo Requesting administrative privileges...
15 goto UACPrompt
16 ) else ( goto gotAdmin )
17
18 :UACPrompt
19 echo Set UAC = CreateObject^("Shell.Application"^) > "%temp%\
       getadmin.vbs"
20 set params = %*:"=""
21 echo UAC.ShellExecute "cmd.exe", "/c ""%~s0"" %1 %params%", "",
       "runas", 1 >> "%temp%\getadmin.vbs"
22
23 "%temp%\getadmin.vbs"
24 del "%temp%\getadmin.vbs"
25 exit /B
26
27 :gotAdmin
28 pushd "%CD%"
29 CD /D "%~dp0"
30 :--------------------------------------
31
32
33 ::Nome Pastas e caminhos
34 cd ..
35 cd ..
36 set PCNAME=%1
37 set INTEGRIDADE=%PCNAME%\Integridade
38 set TOOLS=tools\windows
39
40
41 ::Nomes Ficheiros
```

```
42 set DUMP_RAM_IMAGE=%INTEGRIDADE%\mem.dump
43 set CALCULO_SHA256_FICHEIROS=%INTEGRIDADE%\
       calculo_sha256_ficheiros.txt
44
45
46 ::Dump memória RAM
47 %TOOLS%\winpmem_2.1.exe -o %DUMP_RAM_IMAGE%
48 %TOOLS%\sha256deep.exe %DUMP_RAM_IMAGE% >> %
       CALCULO_SHA256_FICHEIROS%
```

Listagem A.62: *Script* de Integridade Hashes Windows

```
1  @echo off
2
3  ::Nome Pastas e caminhos
4  set PCNAME=%1
5  set HASH=%2
6  set INTEGRIDADE=%PCNAME%\Integridade
7  set TOOLS=tools\windows
8
9  ::Nomes Ficheiros
10 set HASHES_DISCO=%INTEGRIDADE%/hashes_disco.txt
11
12
13 if "%HASH%" == "MD5" (
14 goto md5
15 )
16
17
18 if "%HASH%" == "SHA1" (
19 goto sha1
20 )
21
22
23 if "%HASH%" == "SHA256" (
24 goto SHA256
25 )
26
27
28 :md5
29 ::Cálculo MD5 Ficheiros Disco
30 echo Cálculo MD5 Ficheiros Disco >> %HASHES_DISCO%
31 echo. >> %HASHES_DISCO%
32 echo Comando: md5.exe ficheiros >> %HASHES_DISCO%
33 echo. >> %HASHES_DISCO%
34 echo Output: >> %HASHES_DISCO%
35 %TOOLS%\md5deep.exe -r %SystemDrive% >> %HASHES_DISCO%
36 goto fim
37
38
39 :sha1
40 ::Cálculo SHA1 Ficheiros Disco
41 echo Cálculo SHA1 Ficheiros Disco >> %HASHES_DISCO%
42 echo. >> %HASHES_DISCO%
```

```
43 echo Comando: sha1sum.exe ficheiros >> %HASHES_DISCO%
44 echo. >> %HASHES_DISCO%
45 echo Output: >> %HASHES_DISCO%
46 %TOOLS%\sha1deep.exe -r %SystemDrive% >> %HASHES_DISCO%
47 goto fim
48
49
50 :sha256
51 ::Cálculo SHA256 Ficheiros Disco
52 echo Cálculo SHA256 Ficheiros Disco >> %HASHES_DISCO%
53 echo. >> %HASHES_DISCO%
54 echo Comando: sha256sum.exe ficheiros >> %HASHES_DISCO%
55 echo. >> %HASHES_DISCO%
56 echo Output: >> %HASHES_DISCO%
57 %TOOLS%\sha256deep.exe -r %SystemDrive% >> %HASHES_DISCO%
58
59 :fim
```

Listagem A.63: *Script* de Integridade Hashes Admin Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set HASH=%2
7  set INTEGRIDADE=%PCNAME%\Integridade
8  set TOOLS=tools\windows
9
10 ::Nomes Ficheiros
11 set HASHES_DISCO=%INTEGRIDADE%/hashes_disco.txt
12
13
14 if "%HASH%" == "MD5" (
15 goto md5
16 )
17
18
19 if "%HASH%" == "SHA1" (
20 goto sha1
21 )
22
23
24 if "%HASH%" == "SHA256" (
25 goto SHA256
26 )
27
28
29 :md5
30 ::Cálculo MD5 Ficheiros Disco
31 echo Cálculo MD5 Ficheiros Disco >> %HASHES_DISCO%
32 echo. >> %HASHES_DISCO%
33 echo Comando: md5.exe ficheiros >> %HASHES_DISCO%
34 echo. >> %HASHES_DISCO%
```

```
35 echo Output: >> %HASHES_DISCO%
36 %TOOLS%\md5deep.exe -r %SystemDrive% >> %HASHES_DISCO%
37 goto fim
38
39
40 :sha1
41 ::Cálculo SHA1 Ficheiros Disco
42 echo Cálculo SHA1 Ficheiros Disco >> %HASHES_DISCO%
43 echo. >> %HASHES_DISCO%
44 echo Comando: sha1sum.exe ficheiros >> %HASHES_DISCO%
45 echo. >> %HASHES_DISCO%
46 echo Output: >> %HASHES_DISCO%
47 %TOOLS%\sha1deep.exe -r %SystemDrive% >> %HASHES_DISCO%
48 goto fim
49
50
51 :sha256
52 ::Cálculo SHA256 Ficheiros Disco
53 echo Cálculo SHA256 Ficheiros Disco >> %HASHES_DISCO%
54 echo. >> %HASHES_DISCO%
55 echo Comando: sha256sum.exe ficheiros >> %HASHES_DISCO%
56 echo. >> %HASHES_DISCO%
57 echo Output: >> %HASHES_DISCO%
58 %TOOLS%\sha256deep.exe -r %SystemDrive% >> %HASHES_DISCO%
59
60
61 :fim
```

Listagem A.64: *Script* de Integridade Comandos Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5
6  set PCNAME=%1
7
8  set INTEGRIDADE=%PCNAME%/Integridade
9
10 set TOOLS=tools\windows
11
12
13
14 ::Nomes Ficheiros
15
16 set CALCULO_SHA256_COMANDOS=%INTEGRIDADE%/
       calculo_sha256_comandos.txt
17
18
19
20 ::::::Integridade
21
22
23 ::Cálculo SHA256 Comandos
```

```
24
25 echo Cálculo SHA256 Comandos >> %CALCULO_SHA256_COMANDOS%
26
27 echo. >> %CALCULO_SHA256_COMANDOS%
28
29
30 ::cmd
31
32 echo Comando: cmd >> %CALCULO_SHA256_COMANDOS%
33
34 echo. >> %CALCULO_SHA256_COMANDOS%
35
36 echo Versão: >> %CALCULO_SHA256_COMANDOS%
37
38 ver >> %CALCULO_SHA256_COMANDOS%
39
40 echo. >> %CALCULO_SHA256_COMANDOS%
41
42 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
43
44
45 where cmd > var.txt
46
47 set /P VAR=<var.txt
48
49 del var.txt
50
51
52
53 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
54
55 echo. >> %CALCULO_SHA256_COMANDOS%
56
57 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
58
59 echo. >> %CALCULO_SHA256_COMANDOS%
60
61 echo. >> %CALCULO_SHA256_COMANDOS%
62
63
64
65 ::ifconfig
66
67 echo Comando: ipconfig >> %CALCULO_SHA256_COMANDOS%
68
69 echo. >> %CALCULO_SHA256_COMANDOS%
70
71 echo Versão: >> %CALCULO_SHA256_COMANDOS%
72
73 echo. >> %CALCULO_SHA256_COMANDOS%
74
75 echo. >> %CALCULO_SHA256_COMANDOS%
76
77 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
```

```
78
79
80  where ipconfig > var.txt
81
82  set /P VAR=<var.txt
83
84  del var.txt
85
86
87
88  %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
89
90  echo. >> %CALCULO_SHA256_COMANDOS%
91
92  echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
93
94  echo. >> %CALCULO_SHA256_COMANDOS%
95
96  echo. >> %CALCULO_SHA256_COMANDOS%
97
98
99
100 ::netstat
101
102 echo Comando: netstat >> %CALCULO_SHA256_COMANDOS%
103
104 echo. >> %CALCULO_SHA256_COMANDOS%
105
106 echo Versão: >> %CALCULO_SHA256_COMANDOS%
107
108 echo. >> %CALCULO_SHA256_COMANDOS%
109
110 echo. >> %CALCULO_SHA256_COMANDOS%
111
112 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
113
114
115 where netstat > var.txt
116
117 set /P VAR=<var.txt
118
119 del var.txt
120
121
122 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
123
124 echo. >> %CALCULO_SHA256_COMANDOS%
125
126 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
127
128 echo. >> %CALCULO_SHA256_COMANDOS%
129
130 echo. >> %CALCULO_SHA256_COMANDOS%
131
```

```
132
133
134 ::arp
135
136 echo Comando: arp >> %CALCULO_SHA256_COMANDOS%
137
138 echo. >> %CALCULO_SHA256_COMANDOS%
139
140 echo Versão: >> %CALCULO_SHA256_COMANDOS%
141
142 echo. >> %CALCULO_SHA256_COMANDOS%
143
144 echo. >> %CALCULO_SHA256_COMANDOS%
145
146 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
147
148
149 where arp > var.txt
150
151 set /P VAR=<var.txt
152
153 del var.txt
154
155
156 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
157
158 echo. >> %CALCULO_SHA256_COMANDOS%
159
160 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
161
162 echo. >> %CALCULO_SHA256_COMANDOS%
163
164 echo. >> %CALCULO_SHA256_COMANDOS%
165
166
167
168 ::openedfileview
169
170 echo Comando: openedfileview >> %CALCULO_SHA256_COMANDOS%
171
172 echo. >> %CALCULO_SHA256_COMANDOS%
173
174 echo Versão: >> %CALCULO_SHA256_COMANDOS%
175
176 echo. >> %CALCULO_SHA256_COMANDOS%
177
178 echo. >> %CALCULO_SHA256_COMANDOS%
179
180 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
181
182 %TOOLS%\sha256deep.exe %TOOLS%\OpenedFilesView.exe >> %
      CALCULO_SHA256_COMANDOS%
183
184 echo. >> %CALCULO_SHA256_COMANDOS%
```

```
185
186 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
187
188 echo. >> %CALCULO_SHA256_COMANDOS%
189
190 echo. >> %CALCULO_SHA256_COMANDOS%
191
192
193
194 ::pslist
195
196 echo Comando: pslist >> %CALCULO_SHA256_COMANDOS%
197
198 echo. >> %CALCULO_SHA256_COMANDOS%
199
200 echo Versão: >> %CALCULO_SHA256_COMANDOS%
201
202 echo. >> %CALCULO_SHA256_COMANDOS%
203
204 echo. >> %CALCULO_SHA256_COMANDOS%
205
206 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
207
208 %TOOLS%\sha256deep.exe %TOOLS%\pslist.exe >> %
      CALCULO_SHA256_COMANDOS%
209
210 echo. >> %CALCULO_SHA256_COMANDOS%
211
212 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
213
214 echo. >> %CALCULO_SHA256_COMANDOS%
215
216 echo. >> %CALCULO_SHA256_COMANDOS%
217
218
219
220 ::w32tm
221
222 echo Comando: w32tm >> %CALCULO_SHA256_COMANDOS%
223
224 echo. >> %CALCULO_SHA256_COMANDOS%
225
226 echo Versão: >> %CALCULO_SHA256_COMANDOS%
227
228 echo. >> %CALCULO_SHA256_COMANDOS%
229
230 echo. >> %CALCULO_SHA256_COMANDOS%
231
232 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
233
234
235 where w32tm > var.txt
236
237 set /P VAR=<var.txt
```

```
238
239 del var.txt
240
241
242 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
243
244 echo. >> %CALCULO_SHA256_COMANDOS%
245
246 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
247
248 echo. >> %CALCULO_SHA256_COMANDOS%
249
250 echo. >> %CALCULO_SHA256_COMANDOS%
251
252
253
254 ::wmic
255
256 echo Comando: wmic >> %CALCULO_SHA256_COMANDOS%
257
258 echo. >> %CALCULO_SHA256_COMANDOS%
259
260 echo Versão: >> %CALCULO_SHA256_COMANDOS%
261
262 echo. >> %CALCULO_SHA256_COMANDOS%
263
264 echo. >> %CALCULO_SHA256_COMANDOS%
265
266 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
267
268
269 where wmic > var.txt
270
271 set /P VAR=<var.txt
272
273 del var.txt
274
275
276 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
277
278 echo. >> %CALCULO_SHA256_COMANDOS%
279
280 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
281
282 echo. >> %CALCULO_SHA256_COMANDOS%
283
284 echo. >> %CALCULO_SHA256_COMANDOS%
285
286
287
288 ::schtasks
289
290 echo Comando: schtasks >> %CALCULO_SHA256_COMANDOS%
291
```

```
292 echo. >> %CALCULO_SHA256_COMANDOS%
293
294 echo Versão: >> %CALCULO_SHA256_COMANDOS%
295
296 echo. >> %CALCULO_SHA256_COMANDOS%
297
298 echo. >> %CALCULO_SHA256_COMANDOS%
299
300 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
301
302
303 where schtasks > var.txt
304
305 set /P VAR=<var.txt
306
307 del var.txt
308
309
310 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
311
312 echo. >> %CALCULO_SHA256_COMANDOS%
313
314 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
315
316 echo. >> %CALCULO_SHA256_COMANDOS%
317
318 echo. >> %CALCULO_SHA256_COMANDOS%
319
320
321
322 ::dumpsec
323
324 echo Comando: dumpsec >> %CALCULO_SHA256_COMANDOS%
325
326 echo. >> %CALCULO_SHA256_COMANDOS%
327
328 echo Versão: >> %CALCULO_SHA256_COMANDOS%
329
330 echo. >> %CALCULO_SHA256_COMANDOS%
331
332 echo. >> %CALCULO_SHA256_COMANDOS%
333
334 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
335
336 %TOOLS%\sha256deep.exe %TOOLS%\dumpsec.exe >> %
       CALCULO_SHA256_COMANDOS%
337
338 echo. >> %CALCULO_SHA256_COMANDOS%
339
340 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
341
342 echo. >> %CALCULO_SHA256_COMANDOS%
343
344 echo. >> %CALCULO_SHA256_COMANDOS%
```

```
345
346
347
348 ::logonsessions
349
350 echo Comando: logonsessions >> %CALCULO_SHA256_COMANDOS%
351
352 echo. >> %CALCULO_SHA256_COMANDOS%
353
354 echo Versão: >> %CALCULO_SHA256_COMANDOS%
355
356 echo. >> %CALCULO_SHA256_COMANDOS%
357
358 logonsessions.exe --version >> %CALCULO_SHA256_COMANDOS%
359
360 echo. >> %CALCULO_SHA256_COMANDOS%
361
362 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
363
364 %TOOLS%\sha256deep.exe %TOOLS%\logonsessions.exe >> %
       CALCULO_SHA256_COMANDOS%
365
366 echo. >> %CALCULO_SHA256_COMANDOS%
367
368 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
369
370 echo. >> %CALCULO_SHA256_COMANDOS%
371
372 echo. >> %CALCULO_SHA256_COMANDOS%
373
374
375
376 ::NetUsers
377
378 echo Comando: NetUsers >> %CALCULO_SHA256_COMANDOS%
379
380 echo. >> %CALCULO_SHA256_COMANDOS%
381
382 echo Versão: >> %CALCULO_SHA256_COMANDOS%
383
384 echo. >> %CALCULO_SHA256_COMANDOS%
385
386 echo. >> %CALCULO_SHA256_COMANDOS%
387
388 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
389
390 %TOOLS%\sha256deep.exe %TOOLS%\NetUsers.exe >> %
       CALCULO_SHA256_COMANDOS%
391
392 echo. >> %CALCULO_SHA256_COMANDOS%
393
394 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
395
396 echo. >> %CALCULO_SHA256_COMANDOS%
```

```
397
398 echo. >> %CALCULO_SHA256_COMANDOS%
399
400
401
402 ::di
403
404 echo Comando: di >> %CALCULO_SHA256_COMANDOS%
405
406 echo. >> %CALCULO_SHA256_COMANDOS%
407
408 echo Versão: >> %CALCULO_SHA256_COMANDOS%
409
410 echo. >> %CALCULO_SHA256_COMANDOS%
411
412 echo. >> %CALCULO_SHA256_COMANDOS%
413
414 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
415
416 %TOOLS%\sha256deep.exe %TOOLS%\di.exe >> %
      CALCULO_SHA256_COMANDOS%
417
418 echo. >> %CALCULO_SHA256_COMANDOS%
419
420 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
421
422 echo. >> %CALCULO_SHA256_COMANDOS%
423
424 echo. >> %CALCULO_SHA256_COMANDOS%
425
426
427
428 ::psloglist.exe
429
430 echo Comando: psloglist >> %CALCULO_SHA256_COMANDOS%
431
432 echo. >> %CALCULO_SHA256_COMANDOS%
433
434 echo Versão: >> %CALCULO_SHA256_COMANDOS%
435
436 echo. >> %CALCULO_SHA256_COMANDOS%
437
438 echo. >> %CALCULO_SHA256_COMANDOS%
439
440 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
441
442 %TOOLS%\sha256deep.exe %TOOLS%\psloglist.exe >> %
      CALCULO_SHA256_COMANDOS%
443
444 echo. >> %CALCULO_SHA256_COMANDOS%
445
446 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
447
448 echo. >> %CALCULO_SHA256_COMANDOS%
```

```
449
450 echo. >> %CALCULO_SHA256_COMANDOS%
451
452
453
454 ::RawCopy
455
456 echo Comando: RawCopy >> %CALCULO_SHA256_COMANDOS%
457
458 echo. >> %CALCULO_SHA256_COMANDOS%
459
460 echo Versão: >> %CALCULO_SHA256_COMANDOS%
461
462 echo. >> %CALCULO_SHA256_COMANDOS%
463
464 echo. >> %CALCULO_SHA256_COMANDOS%
465
466 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
467
468 %TOOLS%\sha256deep.exe %TOOLS%\RawCopy.exe >> %
      CALCULO_SHA256_COMANDOS%
469
470 echo. >> %CALCULO_SHA256_COMANDOS%
471
472 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
473
474 echo. >> %CALCULO_SHA256_COMANDOS%
475
476 echo. >> %CALCULO_SHA256_COMANDOS%
477
478
479
480 ::sha256deep
481
482 echo Comando: sha256deep >> %CALCULO_SHA256_COMANDOS%
483
484 echo. >> %CALCULO_SHA256_COMANDOS%
485
486 echo Versão: >> %CALCULO_SHA256_COMANDOS%
487
488 echo. >> %CALCULO_SHA256_COMANDOS%
489
490 sha256deep.exe -v >> %CALCULO_SHA256_COMANDOS%
491
492 echo. >> %CALCULO_SHA256_COMANDOS%
493
494 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
495
496 %TOOLS%\sha256deep.exe %TOOLS%\sha256deep.exe >> %
      CALCULO_SHA256_COMANDOS%
497
498 echo. >> %CALCULO_SHA256_COMANDOS%
499
500 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
```

```
501
502 echo. >> %CALCULO_SHA256_COMANDOS%
503
504 echo. >> %CALCULO_SHA256_COMANDOS%
```

Listagem A.65: *Script* de Integridade Comandos Admin Windows

```
1  @echo off
2
3
4  ::Nome Pastas e caminhos
5  set PCNAME=%1
6  set INTEGRIDADE=%PCNAME%/Integridade
7  set TOOLS=tools\windows
8
9
10 ::Nomes Ficheiros
11 set CALCULO_SHA256_COMANDOS=%INTEGRIDADE%/
       calculo_sha256_comandos.txt
12
13
14 ::::::Integridade
15
16 ::Cálculo SHA256 Comandos
17 echo Cálculo SHA256 Comandos >> %CALCULO_SHA256_COMANDOS%
18 echo. >> %CALCULO_SHA256_COMANDOS%
19
20 ::cmd
21 echo Comando: cmd >> %CALCULO_SHA256_COMANDOS%
22 echo. >> %CALCULO_SHA256_COMANDOS%
23 echo Versão: >> %CALCULO_SHA256_COMANDOS%
24 ver >> %CALCULO_SHA256_COMANDOS%
25 echo. >> %CALCULO_SHA256_COMANDOS%
26 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
27
28 where cmd > var.txt
29 set /P VAR=<var.txt
30 del var.txt
31
32
33 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
34 echo. >> %CALCULO_SHA256_COMANDOS%
35 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
36 echo. >> %CALCULO_SHA256_COMANDOS%
37 echo. >> %CALCULO_SHA256_COMANDOS%
38
39
40 ::ifconfig
41 echo Comando: ipconfig >> %CALCULO_SHA256_COMANDOS%
42 echo. >> %CALCULO_SHA256_COMANDOS%
43 echo Versão: >> %CALCULO_SHA256_COMANDOS%
44 echo. >> %CALCULO_SHA256_COMANDOS%
45 echo. >> %CALCULO_SHA256_COMANDOS%
46 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
```

```
47
48 where ipconfig > var.txt
49 set /P VAR=<var.txt
50 del var.txt
51
52
53 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
54 echo. >> %CALCULO_SHA256_COMANDOS%
55 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
56 echo. >> %CALCULO_SHA256_COMANDOS%
57 echo. >> %CALCULO_SHA256_COMANDOS%
58
59
60 ::netstat
61 echo Comando: netstat >> %CALCULO_SHA256_COMANDOS%
62 echo. >> %CALCULO_SHA256_COMANDOS%
63 echo Versão: >> %CALCULO_SHA256_COMANDOS%
64 echo. >> %CALCULO_SHA256_COMANDOS%
65 echo. >> %CALCULO_SHA256_COMANDOS%
66 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
67
68 where netstat > var.txt
69 set /P VAR=<var.txt
70 del var.txt
71
72 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
73 echo. >> %CALCULO_SHA256_COMANDOS%
74 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
75 echo. >> %CALCULO_SHA256_COMANDOS%
76 echo. >> %CALCULO_SHA256_COMANDOS%
77
78
79 ::arp
80 echo Comando: arp >> %CALCULO_SHA256_COMANDOS%
81 echo. >> %CALCULO_SHA256_COMANDOS%
82 echo Versão: >> %CALCULO_SHA256_COMANDOS%
83 echo. >> %CALCULO_SHA256_COMANDOS%
84 echo. >> %CALCULO_SHA256_COMANDOS%
85 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
86
87 where arp > var.txt
88 set /P VAR=<var.txt
89 del var.txt
90
91 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
92 echo. >> %CALCULO_SHA256_COMANDOS%
93 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
94 echo. >> %CALCULO_SHA256_COMANDOS%
95 echo. >> %CALCULO_SHA256_COMANDOS%
96
97
98 ::openedfileview
99 echo Comando: openedfileview >> %CALCULO_SHA256_COMANDOS%
100 echo. >> %CALCULO_SHA256_COMANDOS%
```

```
101 echo Versão: >> %CALCULO_SHA256_COMANDOS%
102 echo. >> %CALCULO_SHA256_COMANDOS%
103 echo. >> %CALCULO_SHA256_COMANDOS%
104 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
105 %TOOLS%\sha256deep.exe %TOOLS%\OpenedFilesView.exe >> %
        CALCULO_SHA256_COMANDOS%
106 echo. >> %CALCULO_SHA256_COMANDOS%
107 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
108 echo. >> %CALCULO_SHA256_COMANDOS%
109 echo. >> %CALCULO_SHA256_COMANDOS%
110
111
112 ::pslist
113 echo Comando: pslist >> %CALCULO_SHA256_COMANDOS%
114 echo. >> %CALCULO_SHA256_COMANDOS%
115 echo Versão: >> %CALCULO_SHA256_COMANDOS%
116 echo. >> %CALCULO_SHA256_COMANDOS%
117 echo. >> %CALCULO_SHA256_COMANDOS%
118 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
119 %TOOLS%\sha256deep.exe %TOOLS%\pslist.exe >> %
        CALCULO_SHA256_COMANDOS%
120 echo. >> %CALCULO_SHA256_COMANDOS%
121 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
122 echo. >> %CALCULO_SHA256_COMANDOS%
123 echo. >> %CALCULO_SHA256_COMANDOS%
124
125
126 ::w32tm
127 echo Comando: w32tm >> %CALCULO_SHA256_COMANDOS%
128 echo. >> %CALCULO_SHA256_COMANDOS%
129 echo Versão: >> %CALCULO_SHA256_COMANDOS%
130 echo. >> %CALCULO_SHA256_COMANDOS%
131 echo. >> %CALCULO_SHA256_COMANDOS%
132 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
133
134 where w32tm > var.txt
135 set /P VAR=<var.txt
136 del var.txt
137
138 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
139 echo. >> %CALCULO_SHA256_COMANDOS%
140 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
141 echo. >> %CALCULO_SHA256_COMANDOS%
142 echo. >> %CALCULO_SHA256_COMANDOS%
143
144
145 ::wmic
146 echo Comando: wmic >> %CALCULO_SHA256_COMANDOS%
147 echo. >> %CALCULO_SHA256_COMANDOS%
148 echo Versão: >> %CALCULO_SHA256_COMANDOS%
149 echo. >> %CALCULO_SHA256_COMANDOS%
150 echo. >> %CALCULO_SHA256_COMANDOS%
151 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
152
```

```
153 where wmic > var.txt
154 set /P VAR=<var.txt
155 del var.txt
156
157 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
158 echo. >> %CALCULO_SHA256_COMANDOS%
159 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
160 echo. >> %CALCULO_SHA256_COMANDOS%
161 echo. >> %CALCULO_SHA256_COMANDOS%
162
163
164 ::schtasks
165 echo Comando: schtasks >> %CALCULO_SHA256_COMANDOS%
166 echo. >> %CALCULO_SHA256_COMANDOS%
167 echo Versão: >> %CALCULO_SHA256_COMANDOS%
168 echo. >> %CALCULO_SHA256_COMANDOS%
169 echo. >> %CALCULO_SHA256_COMANDOS%
170 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
171
172 where schtasks > var.txt
173 set /P VAR=<var.txt
174 del var.txt
175
176 %TOOLS%\sha256deep.exe %VAR% >> %CALCULO_SHA256_COMANDOS%
177 echo. >> %CALCULO_SHA256_COMANDOS%
178 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
179 echo. >> %CALCULO_SHA256_COMANDOS%
180 echo. >> %CALCULO_SHA256_COMANDOS%
181
182
183 ::dumpsec
184 echo Comando: dumpsec >> %CALCULO_SHA256_COMANDOS%
185 echo. >> %CALCULO_SHA256_COMANDOS%
186 echo Versão: >> %CALCULO_SHA256_COMANDOS%
187 echo. >> %CALCULO_SHA256_COMANDOS%
188 echo. >> %CALCULO_SHA256_COMANDOS%
189 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
190 %TOOLS%\sha256deep.exe %TOOLS%\dumpsec.exe >> %
        CALCULO_SHA256_COMANDOS%
191 echo. >> %CALCULO_SHA256_COMANDOS%
192 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
193 echo. >> %CALCULO_SHA256_COMANDOS%
194 echo. >> %CALCULO_SHA256_COMANDOS%
195
196
197 ::logonsessions
198 echo Comando: logonsessions >> %CALCULO_SHA256_COMANDOS%
199 echo. >> %CALCULO_SHA256_COMANDOS%
200 echo Versão: >> %CALCULO_SHA256_COMANDOS%
201 echo. >> %CALCULO_SHA256_COMANDOS%
202 logonsessions.exe --version >> %CALCULO_SHA256_COMANDOS%
203 echo. >> %CALCULO_SHA256_COMANDOS%
204 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
205 %TOOLS%\sha256deep.exe %TOOLS%\logonsessions.exe >> %
```

```
    CALCULO_SHA256_COMANDOS%
206 echo. >> %CALCULO_SHA256_COMANDOS%
207 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
208 echo. >> %CALCULO_SHA256_COMANDOS%
209 echo. >> %CALCULO_SHA256_COMANDOS%
210
211
212 ::NetUsers
213 echo Comando: NetUsers >> %CALCULO_SHA256_COMANDOS%
214 echo. >> %CALCULO_SHA256_COMANDOS%
215 echo Versão: >> %CALCULO_SHA256_COMANDOS%
216 echo. >> %CALCULO_SHA256_COMANDOS%
217 echo. >> %CALCULO_SHA256_COMANDOS%
218 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
219 %TOOLS%\sha256deep.exe %TOOLS%\NetUsers.exe >> %
    CALCULO_SHA256_COMANDOS%
220 echo. >> %CALCULO_SHA256_COMANDOS%
221 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
222 echo. >> %CALCULO_SHA256_COMANDOS%
223 echo. >> %CALCULO_SHA256_COMANDOS%
224
225
226 ::di
227 echo Comando: di >> %CALCULO_SHA256_COMANDOS%
228 echo. >> %CALCULO_SHA256_COMANDOS%
229 echo Versão: >> %CALCULO_SHA256_COMANDOS%
230 echo. >> %CALCULO_SHA256_COMANDOS%
231 echo. >> %CALCULO_SHA256_COMANDOS%
232 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
233 %TOOLS%\sha256deep.exe %TOOLS%\di.exe >> %
    CALCULO_SHA256_COMANDOS%
234 echo. >> %CALCULO_SHA256_COMANDOS%
235 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
236 echo. >> %CALCULO_SHA256_COMANDOS%
237 echo. >> %CALCULO_SHA256_COMANDOS%
238
239
240 ::psloglist.exe
241 echo Comando: psloglist >> %CALCULO_SHA256_COMANDOS%
242 echo. >> %CALCULO_SHA256_COMANDOS%
243 echo Versão: >> %CALCULO_SHA256_COMANDOS%
244 echo. >> %CALCULO_SHA256_COMANDOS%
245 echo. >> %CALCULO_SHA256_COMANDOS%
246 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
247 %TOOLS%\sha256deep.exe %TOOLS%\psloglist.exe >> %
    CALCULO_SHA256_COMANDOS%
248 echo. >> %CALCULO_SHA256_COMANDOS%
249 echo ---------------------------------- >> %CALCULO_SHA256_COMANDOS%
250 echo. >> %CALCULO_SHA256_COMANDOS%
251 echo. >> %CALCULO_SHA256_COMANDOS%
252
253
254 ::RawCopy
255 echo Comando: RawCopy >> %CALCULO_SHA256_COMANDOS%
```

```
256 echo. >> %CALCULO_SHA256_COMANDOS%
257 echo Versão: >> %CALCULO_SHA256_COMANDOS%
258 echo. >> %CALCULO_SHA256_COMANDOS%
259 echo. >> %CALCULO_SHA256_COMANDOS%
260 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
261 %TOOLS%\sha256deep.exe %TOOLS%\RawCopy.exe >> %
        CALCULO_SHA256_COMANDOS%
262 echo. >> %CALCULO_SHA256_COMANDOS%
263 echo ------------------------------------ >> %CALCULO_SHA256_COMANDOS%
264 echo. >> %CALCULO_SHA256_COMANDOS%
265 echo. >> %CALCULO_SHA256_COMANDOS%
266
267
268 ::sha256deep
269 echo Comando: sha256deep >> %CALCULO_SHA256_COMANDOS%
270 echo. >> %CALCULO_SHA256_COMANDOS%
271 echo Versão: >> %CALCULO_SHA256_COMANDOS%
272 echo. >> %CALCULO_SHA256_COMANDOS%
273 sha256deep.exe -v >> %CALCULO_SHA256_COMANDOS%
274 echo. >> %CALCULO_SHA256_COMANDOS%
275 echo SHA256 Comando: >> %CALCULO_SHA256_COMANDOS%
276 %TOOLS%\sha256deep.exe %TOOLS%\sha256deep.exe >> %
        CALCULO_SHA256_COMANDOS%
277 echo. >> %CALCULO_SHA256_COMANDOS%
278 echo ------------------------------------ >> %CALCULO_SHA256_COMANDOS%
279 echo. >> %CALCULO_SHA256_COMANDOS%
280 echo. >> %CALCULO_SHA256_COMANDOS%
```