Anti-Análise Forense

Autor:

Luís Miguel Silva
Centro de Informática
Instituto Superior Politécnico Gaya, (CIISP)
Rua António Rodrigues da Rocha, 291, 341,
Santo Ovídio, 4400-025 Vila Nova de Gaia
lms@ispgaya.pt

Resumo
A ciência forense permite-nos chegar aos factos através das evidências.
Tal como esta ciência é usada na justiça no campo da medicina legal, com o avanço das tecnologias, ela foi também levada para o campo da informática.
A análise forense informática é o campo que estuda o "como", "quando" e o "porquê" de um ataque a um computador.
Através dela é possível determinar informação vital que poderá ser usada como prova para levar o criminoso à justiça.

1-Introdução

A análise forense é em si um campo de estudo fascinante, qualquer que seja a área em que seja aplicada.
Os especialistas forenses são os detectives dos nossos tempos. Analisam os mais pequenos pormenores e conseguem chegar a conclusões extremamente precisas acerca do "crime".
Tal como o médico legista faz uma autópsia procurando padrões e anomalias num cadáver, o especialista informático forense procura nos pequenos bits do disco informação deixada pelo atacante que possivelmente o acabará por o incriminar.
Este detective é capaz de vasculhar os mais obscuros cantos de um dispositivo físico à procura de provas.
No campo da informática entende-se por "análise forense" o uso da ciência e tecnologia para investigar e estabelecer factos sobre acções criminosas.
A "anti-análise forense" é então a *arte* de esconder dados, informações e pistas após uma penetração num sistema informático.
Este tipo de análise digital forense está a ganhar terreno como resposta a incidentes criminosos. Surgiu uma nova geração de analistas assistidos por ferramentas inovadoras e sofisticadas. Curiosamente, estas ferramentas funcionam todas sobre a mesma base e raramente se fala sobre o campo da anti-análise forense.
Os exemplos mostrados neste artigo serão feitos sobre o tipo de sistema de ficheiros extfs (do Linux) visto este sistema ser o mais amplamente utilizado.
Neste artigo iremos mostrar como é que os analistas geralmente procuram pelo rasto de um atacante e como estes tentam eliminar esse rasto.

Este artigo não pretende de forma alguma incentivar a actos criminosos mas sim mostrar o lado do analista e do atacante tentando encontrar e explicar quais as falhas mais comuns de ambos.

2-Como funciona a Anti-Análise Forense

Um atacante depois de penetrar num sistema tem que ser capaz de garantir o acesso posterior a esse mesmo sistema.
Hoje em dia, é boa prática de administração de sistemas a despistagem de padrões das anomalias detectadas no servidor. Significa, regra geral, que o sistema tem um administrador zeloso por trás.
Ora, há aqui uma contradição. Se existem boas práticas, como justificar a intrusão?
Nenhum administrador pode prevenir ataques contra falhas ainda não conhecidas. Aliás, é dai que vem a famosa afirmação de que "o único computador seguro é aquele que está desligado da corrente, fechado num cofre anti-fogo e anti-bomba, rodeado de guardas e se possível no fundo do oceano". E mesmo assim, tal como o autor Kevin Mitnick dizia "podemos sempre convencer alguém a ligá-lo...".
Ninguém pode proteger um sistema de um ***0'day exploit*** (programa que aproveita uma determinada falha e que apenas é distribuído no "underground"). Por outro lado, mais cedo ou mais tarde, o referido *exploit* será tornado público o que fará com que o administrador analise o seu sistema, detecte as respectivas vulnerabilidades e tente monitorizar se alguém está a tirar partido dela. A arte da anti-análise está em conseguir vaguear livremente por um sistema sem ser detectado.
Para garantir isto o atacante recorre a várias "tecnologias" de que se fala a seguir.

Se um atacante quer manter acesso a um sistema tem que esconder o seu rasto. Procede então à criptografia dos seus ficheiros, instalação de "*rootkits"* (conjuntos de ferramentas completas que garantem o acesso ilegítimo a um sistema por parte de um atacante sem este ser detectado), "*backdoors"*, escondendo os seus ficheiros em *fluxos de dados alternados* (vulgo ADS – Alternate Data Streams existente no sistema de ficheiros NTFS), esconder ficheiros no chamado "*slack space*", re-alocando e mascarando os seus ficheiros como uma biblioteca, processo ou dispositivo que passe despercebido num sistema e recorrendo à "*esteganografia*" (escondendo a sua informação em ficheiros de imagem ou de som).

2.1- A criptografia dos dados

Quando um atacante penetra num sistema geralmente utiliza-o para guardar ficheiros.
Estes ficheiros podem ser o software utilizado (ou que ainda será utilizado) para conseguir o acesso ilegítimo aquele sistema (ou a outros, visto que um computador ligado a uma rede é sempre um novo ponto de acesso para partir para novos ataques).
Se o atacante deixasse estes ficheiros espalhados num directório chamado ***exploits*** ou num simples directório escondido iria levantar muitas suspeitas para o administrador. Surge então a necessidade de os esconder para evitar ser detectado.
Uma das soluções é recorrer à criptografia dos seus ficheiros de forma a que, mesmo que estes sejam detectados, o administrador não seja capaz de decifrar o que são ou para que servem.
O atacante então comprime todos os ficheiros num só e criptografa-os de forma a não ser detectado.

A partir deste momento a única forma que o administrador tem de descobrir os conteúdos desse ficheiro é descobrir a cifra para o descriptografar ou, mais simplesmente ainda, descobrir a localização física do atacante e obter dele a colaboração necessária.

2.2-Rootkits

O primeiro passo a tomar por parte do atacante após penetrar num novo sistema é garantir o seu acesso posterior (sem levantar muitas suspeitas).
Os *rootkits* são bastante populares para fazer isso mesmo.
Um *rootkit* é um conjunto de ferramentas completo que contém todo o software necessário para remover os rastos do atacante dos "*logs*" do sistema, instalar *backdoors* em ficheiros binários, criar contas às quais o sistema não regista informação e que possibilitam a administração remota do sistema.
O interessante destes *rootkits* é que geralmente não é necessário grandes conhecimentos informáticos para os instalar. Isto levanta um problema para o atacante. Se algo corre mal, este perde o acesso ao sistema sem remover convenientemente os seus rastos.
Há algum tempo o autor presenciou um problema enquanto estava de serviço numa conhecida Instituição do Porto. Misteriosamente, um dos servidores utilizado num dos projectos do departamento em causa demonstrou "algumas" anomalias. Apenas o servidor web funcionava e não era possível fazer uma simples autenticação no sistema. Qualquer acção feita por parte do administrador nesse servidor apenas resultava em erros atrás de erros sobre "*bibliotecas*" que não existiam e cujas versões eram mais antigas ou mais recentes do que as necessárias.
O que se passou neste caso foi que o *hacker* entrou naquele servidor e tentou instalar um *rootkit* sem tomar as devidas precauções primeiro. Instalou um rootkit antigo que falhava perante as dependências de ficheiros binários e "*bibliotecas*" existentes no sistema. Isto fez com que o sistema se tornasse completamente inutilizável.

2.3 – Sniffers

A informação flutua livre e anarquicamente por uma rede. É então altura do atacante a interceptar e guardar para uso posterior (novamente, sem ser detectado).
Admita-se que o atacante já penetrou no sistema, possui privilégios de administrador e instalou o seu rootkit para garantir acesso posterior.
Pode agora entrar noutros servidores que estejam acessíveis através da rede física (quer seja cablada, quer seja por tecnologia wireless, como está cada vez mais na moda) simplesmente interceptando a informação circulante.

Então num esquema muito simples de uma *LAN* (Local Area Network), temos o exemplo de 5 computadores ligados a um "*HUB*" (dispositivo físico que permite ligar 2 ou mais computadores em rede).

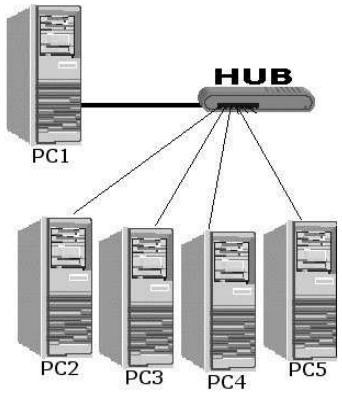

Imagem 1: esquema simples de uma rede

O atacante consegue então comprometer a segurança do "**PC1**". Visto esta rede estar toda interligada por um HUB, se for implementado um "*sniffer*" nesse computador, o atacante vai conseguir interceptar informação que circule entre os 5 pcs.

Então imaginemos que o "**PC2**" contém um serviço de partilha de ficheiros por FTP (File Transfer Protocol), o "**PC1**" tem um "*sniffer*" que analisa o tráfego que circula na rede e o "**PC3**" faz uma ligação por ftp para o "**PC2**".

Podemos ver um exemplo de um "*sniffer*" chamado "ettercap" a funcionar na imagem seguinte.

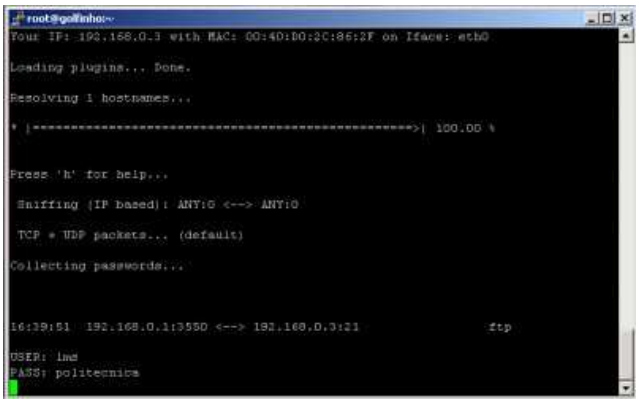

Imagem 2: um sniffer intercepta um login e password enviados por ftp.

Este tipo de software é deixado a correr em segundo plano no servidor e guarda a informação interceptada em ficheiros.
A partir do momento em que o atacante consegue interceptar uma conta válida no sistema que comprometeu vai poder começar a utilizá-la como segunda via de entrada no sistema (no caso de algum dia as suas "*backdoors*" serem descobertas).

Se o administrador não tomar as devidas precauções nunca saberá que o seu sistema está a interceptar passwords de outros computadores na mesma rede (ou do próprio computador).

Uma maneira de descobrir se estamos a interceptar informação é verificar se a placa de rede se encontra em modo "*promíscuo*" (o que, tal como no termo sexual da

palavra significa "manter relações sexuais com grande número de parceiros", na informática significa "capturar a informação de um grande número de parceiros").
O administrador pode confirmar isso através do comando "*ifconfig*", isto claro, se o atacante não tiver modificado esse binário.

```
[root@golfinho root]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:40:D0:2C:86:2F
          inet addr:192.168.0.3  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1611 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1416 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:180440 (176.2 Kb)  TX bytes:166061 (162.1 Kb)
          Interrupt:10 Base address:0x8000

[root@golfinho root]#
```

De qualquer forma o "output" de um programa como o ifconfig nunca é completamente fiável visto que o atacante pode alterar a "*flag*" que mostra que o *interface* se encontra em modo *promíscuo* de forma a despistar o administrador.

2.4 – IDS (Intrusion Detection Systems)

É vital para o administrador analisar periodicamente o seu servidor à procura de anomalias. Se esta tarefa não for levada a sério e o atacante souber o que está a fazer, o administrador nunca saberá que o seu sistema foi comprometido, a não ser que atacante esteja farto de "estar no escuro" e se queira mostrar. Afinal, este é o grande problema do "hacker" (e de todos os seres humanos, em especial os criminosos). Qual é a piada de fazer algo "emocionante" como penetrar num sistema se não pudermos contar a alguém? Geralmente os miúdos mais novos que comprometem a segurança de servidores são quem mais cedo se mostram. Quer pela sua inexperiência, quer pela necessidade de se mostrarem ao mundo.
Este desejo vai-se perdendo à medida que o atacante vai ganhando maturidade.

Existe então a necessidade de instalar um software que detecte a actividade de possíveis intrusos. A esse sistema de análise chamamos ***IDS*** (intrusion detection system).

Um exemplo de um *IDS* é o **tripwire**. Este sistema monitoriza atributos chave de ficheiros que "apenas se modificariam" no caso de uma intrusão. Os atributos analisados são a assinatura binária, tamanho, modificação esperada no tamanho, etc.

O seu funcionamento não é muito complexo. Primeiro analisa e guarda informações sobre todos os ficheiros no sistema (ou ficheiros contidos em directórios "estratégicos" no sistema) para mais tarde os comparar com as assinaturas dos ficheiros actuais.
Como o leitor pode imaginar isto causa uma grande sobrecarga no sistema o que faz com que muitos administradores não utilizem um IDS deste tipo.

Em qualquer dos casos, a sua utilização é fortemente aconselhada. Imagine-se que um atacante consegue comprometer um sistema e modifica binários chave como o "ps" (programa que mostra todos os processos a correr no sistema), o "lsof" (programa que

mostra todos os ficheiros abertos no momento) ou o "netstat" (programa que mostra as ligações TCP/UDP abertas no momento).
Por exemplo, o "lsof" é um comando dos mais úteis no sistema. Num sistema UNIX, tudo se implementa à base de ficheiros. Os directórios e os periféricos surgem retratados no sistema como ficheiros.

2.5 – ADS (Alternate Data Streams) – fluxos de dados alternados

Neste ponto do artigo, fala-se sobre formas que existem para esconder o rasto de um atacante.
No Microsoft Windows NT e seus sucessores (2000 e XP, em todas as suas variantes), existe um sistema de ficheiros chamado **NTFS**. Este sistema de ficheiros é um sistema de ficheiros especialmente concebido para computadores multi-utilizador. Assim sendo, existem algumas diferenças acentuadas entre este sistema de ficheiros e o formato anterior (FAT16 ou FAT32) presente nos sistemas Microsoft Windows 95/98 e Millenium.
Existe uma "característica não documentada" (como a Microsoft prefere chamar-lhe) no NTFS que possibilita guardar dados em ficheiros escondidos que por sua vez estão associados a ficheiros visíveis (não suspeitos).

Os fluxos de informação criados por estas sequências de ficheiros representam um sério risco de segurança porque são completamente invisíveis com as ferramentas normalmente instaladas nos sistemas operativos, tornando-os locais perfeitos para se esconderem ficheiros indesejados tais como vírus e "cavalos de tróia" (vulgo *backdoors*).

Se por um lado esta característica do "SF" pode ser muito facilmente utilizada, por outro lado é apenas detectada com software especializado para o efeito.Um programa como o "explorer" do windows consegue apenas ver os ficheiros normais não mostrando assim que existem fluxos de dados escondidos nem mostrando o espaço realmente ocupado por eles. Visto que esta capacidade do "SF" NTFS é pouco conhecida para os programadores, existe muito pouco software de segurança que detecte ficheiros escondidos através dela.

Interessante também será mencionar que a única maneira de eliminar um fluxo de dados alojado num sistema de ficheiros NTFS e escondido através do ADS é eliminando o ficheiro "hospedeiro".

2.6 – Esconder informação no "slack space"

O chamado "slack space" é o espaço em disco entre o final de um ficheiro e o final de um grupo de blocos ("cluster") no disco físico.
Com os programas certos, o atacante é capaz de utilizar esse espaço para esconder os seus ficheiros. Visto que os ficheiros são guardados em blocos inutilizados (como por exemplo blocos marcados como "corrompidos"), o atacante consegue despistar a maior parte do software existente de análise forense.
A maioria do software de análise forense não procura informação escondida nestes espaços sendo assim, tal como o "ADS" no NTFS um sitio perfeito para o atacante esconder informação maliciosa.

O espaço disponível em cada um destes blocos inutilizados é minúsculo pelo que o atacante necessita de um software especial para unir todos os pequenos bits disponíveis de forma a conseguir um único fluxo de informação suficientemente grande para alojar os seus ficheiros (quer sejam eles um "log" de informação interceptada por um sniffer, ficheiros interessantes encontrados no sistema, *backdoors*, *vírus*, etc).

Mais à frente neste artigo, demonstrar-se-á como fazer isso mesmo utilizando um software chamado "runefs", um conjunto completo de ferramentas para fazer "manutenção" a um nível bastante baixo do disco rígido.

2.7 – Esteganografia

A esteganografia foi catalogada pelos gregos como "escrita secreta".
É uma técnica utilizada há mais de 2500 anos e já foi utilizada com intuitos militares, políticos, pessoais e como forma de resguardar a propriedade intelectual.
Actualmente podemos dizer que é a arte e ciência de qualquer processo que permita esconder uma mensagem num qualquer objecto fazendo com que essa mesma mensagem não seja aparente ao observador.

Um abade alemão, Johannes Trithemius (1462-1516), no seu livro intitulado "Steganographia:hoe est ars per occultam scripturam animi sui voluntatem absentibus aperiendi certa", descreve formas de comunicar com os espiritos. A sua tradução de latim para português significa algo como "Esteganografia: a arte através da qual se escondem escrituras que só são decifráveis através da mente do homem".

Imagem 3: Johannes Trithemius (1462-1516), um abade alemão que escreveu a primeira obra conhecida sobre esteganografia

Os primeiros livros conhecidos e editados sobre criptografia são precisamente os dois primeiros volumes da sua triologia. O terceiro livro versava a astrologia oculta.

Já no nosso tempo, dois investigadores, Dr. Thomas Ernst e Dr. Jim Reeds encontraram mensagens escondidas em tabelas presentes na terceira obra da triologia de Johannes Trithemius. A tradução de latim das mensagens encontradas é algo como "A raposa rápida castanha salta por cima do cão preguiçoso", "O portador desta carta é um ladrão e escumalha. Protege-te contra ele, ele quer fazer-te algo" e a terceira mensagem era o início do vigésimo terceiro salmo.

Podemos dizer que a segunda mensagem escondida é como uma protecção engenhosamente inventada por Johannes Trithemius para proteger a sua propriedade intelectual.

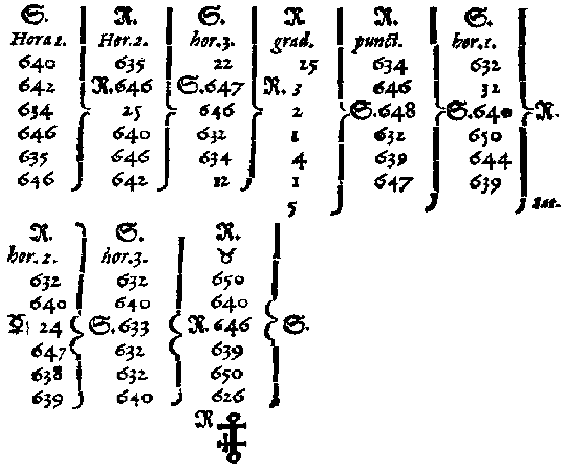

Imagem 4: tabelas criadas por Johannes Trithemius e publicadas no seu terceiro livro sobre estaganografia.

No entanto, a esteganografia é usada há bastante mais tempo.
Segundo documentos existentes, foi utilizada pela primeira vez em 480 aC.
Nessa altura eram utilizadas tábuas de madeira cobertas com cera onde eram gravadas as escrituras.
Um grego chamado Demaratus derreteu então a cera de uma tábua, escreveu uma mensagem na própria madeira e cobriu-a de cera novamente conseguindo assim escondê-la e avisar os espartanos de iminência de uma invasão.

Outro exemplo ilustrativo, este bastante mais recente, é o de uma mensagem, aparentemente inocente e inofensiva, publicada durante a 2ª Guerra Mundial que dizia "Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils". Se retirarmos a segunda letra de cada palavra desta frase conseguimos então encontrar a mensagem escondida, "Pershing sails from NY June I".

Actualmente a esteganografia é frequentemente denominada como "watermarking" (marca de água).
Se o leitor olhar para uma simples nota contra a luz, consegue ver uma imagem escondida, a marca de água. Esta imagem só é visível desta forma, sendo assim um método de esteganografia.

Na informática, a esteganografia é utilizada escondendo informação em ficheiros inofensivos (ficheiros de som, imagem, texto, html, etc) usando para tal os bits inutilizados presentes nesses formatos de ficheiros.
A informação escondida pode ser texto normal, cifrado ou até imagens.

Ao contrário dos dados criptografados, a esteganografia não é facilmente detectada. É portanto, um substituto bem sucedido da criptografia como método de esconder informação.

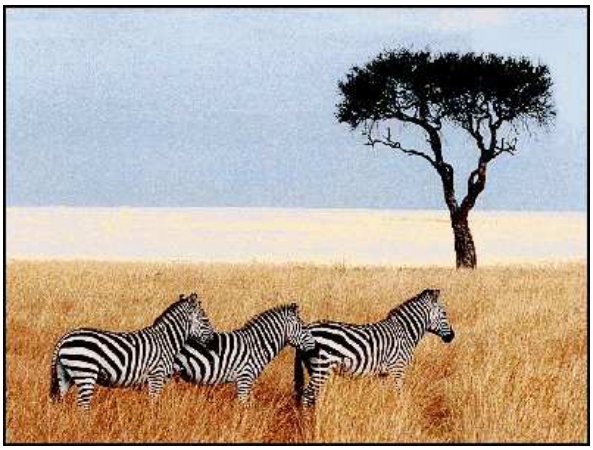

Imagem 5: Imagem sem informação escondida

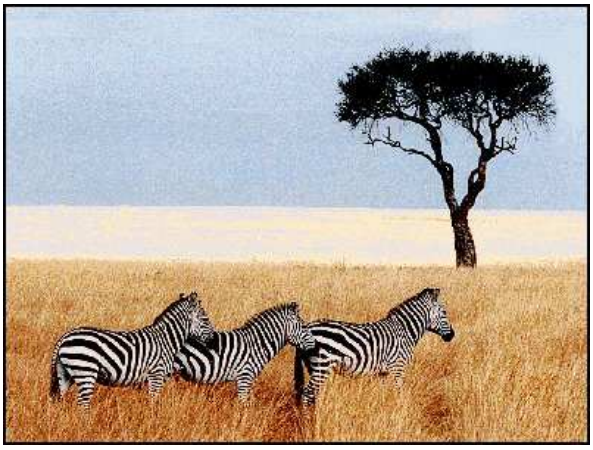

Imagem 6: Imagem com informação escondida.

É necessário software especial para esconder ficheiros através desta técnica. Um destes programas freeware e opensource, é o **steghide**.
O programa **steghide** (freeware e opensource) permite esconder ficheiros dentro de outros, estes últimos habitualmente de tipo inofensivo, sem quebrar a sua funcionalidade. Veja-se o exemplo seguinte.

Imagine-se um ficheiro chamado "politecnica.jpg":

```
[wc@golfinho steg]$ file politecnica.jpg
politecnica.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, 1 x 1
[wc@golfinho steg]$
```

Cria-se agora o ficheiro a esconder...

```
 [wc@golfinho steg]$ cat > ficheiro_privado.txt
isto é um ficheiro privado que se esconde num ficheiro .jpg
^C
[wc@golfinho steg]$
```

E esconde-se o primeiro ficheiro com o **steghide**

```
[wc@golfinho steg]$ steghide embed -cf politecnica.jpg -sf politecnica-steg.jpg -pf
ficheiro_privado.txt
Enter passphrase: inserir_password_aqui
Re-Enter passphrase: inserir_password_aqui
```

```
[wc@golfinho steg]$
```

Como o leitor pode confirmar, existem agora um ficheiro novo chamado **politecnica-steg.jpg**.

```
[wc@golfinho steg]$ ls -l
total 80
-rw-rw-r--    1 wc       wc            64 Feb 24 21:56 ficheiro_privado.txt
-rwxr-xr-x    1 wc       wc         39800 Feb 24 21:26 politecnica.jpg
-rw-rw-r--    1 wc       wc         33190 Feb 24 21:56 politecnica-steg.jpg
 [wc@golfinho steg]$ file politecnica-steg.jpg
politecnica-steg.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, 1 x 1
[wc@golfinho steg]$
```

Este ficheiro contém o primeiro escondido e, no entanto, mantém toda a funcionalidade de uma imagem em formato "jpeg". Será também de notar que o seu tamanho diminuiu.

É então altura de voltar a recuperar o ficheiro que se escondeu no **politecnica-steg.jpg**...

```
 [wc@golfinho steg]$ steghide extract -sf politecnica-steg.jpg
Enter passphrase: inserir_password_aqui
writing plain file to "ficheiro_privado.txt".
[wc@golfinho steg]$ cat ficheiro_privado.txt
isto é um ficheiro privado que vamos esconder num ficheiro .jpg
[wc@golfinho steg]$
```

Esta técnica é bastante eficaz para mover ou guardar informação crucial sem levantar muitas suspeitas. Visto que os ficheiros ficam protegidos com uma palavra passe, um administrador que suspeite de um ficheiro não o conseguirá abrir para investigar.
O algoritmo utilizado pelo **steghide** para mascarar os ficheiros faz com que a única maneira de extrair um ficheiro escondido seja através da palavra passe inserida durante a "metamorfose". Caso contrário, os cabeçalhos do ficheiro hospedeiro não fazem qualquer sentido para o programa que o toma como corrompido, como se pode ver no exemplo seguinte:

```
[wc@golfinho steg]$ steghide extract -sf politecnica-steg.jpg
Enter passphrase: password_errada_inserida_aqui
steghide: the distribution method saved in the stego header is unknown (file corruption ?).
[wc@golfinho steg]$
```

Quando se junta este técnica a outras como a "criptografia", o "ADS" do NTFS ou esconder ficheiros no chamado "slack space", a análise forense torna-se uma tarefa quase impossível e bastante árdua.

3 – Casos práticos...análise forense e anti-análise forense

Nos parágrafos seguintes, demonstrar-se-á o uso de software específico para análise forense e camuflagem do rasto deixado pela intrusão no sistema hospedeiro.
Analise-se então o sistema de ficheiros Unix.

3.1 – Um sistema de ficheiros genérico UNIX.

Os ficheiros num sistema operativo Unix são fluxos contínuos de bytes de tamanho arbitrário e são a principal fonte de "input"/"output" do sistema.

Os dados contidos num disco são geralmente divididos em dois grupos. O da informação sobre os ficheiros e o da informação neles contida.
Chama-se de **meta-dados** à forma como os dados se dispõem no interior dos ficheiros.
Para criar a abstracção do ficheiro, o kernel necessita de transparentemente traduzir e fazer a associação de um ou mais sectores no disco físico, unindo-os e tornando-os num único fluxo de informação. O sistema de ficheiros serve então para fazer a correspondência entre os sectores do disco de forma a ser possível unir esse mesmo fluxo.

O conteúdo de um ficheiro é armazenado em blocos de dados que são guardados nos sectores do disco. Quanto maior for o número de sectores por blocos de dados, maior será a velocidade de transferência do disco melhorando assim a performance do sistema. É por esta razão que a desfragmentação de uma partição melhora o desempenho do computador. A transferência é feita "de uma só vez" em vez de ser necessário ao sistema procurar os bocados de informação espalhados anarquicamente pelo disco rígido.

Os blocos de dados são juntos e organizados em ficheiros pelos *inodes*. Os *inodes* são as estruturas de *meta-dados* que representam os ficheiros visíveis pelo utilizador (um inode por cada ficheiro). Cada *inode* contém um vector de apontadores para os blocos de dados e outras informações adicionais, tais como, o código do utilizador proprietário (UID), o código do seu grupo (GID), as suas permissões de acesso, o seu tamanho, entre outros. Os inodes são guardados num vector especial (a chamada "inode table") e são referidos pelo primeiro elemento dessa tabela (o índice 0). O estado de um *inode* (livre ou ocupado) é então guardado num mapa de bits a que chamamos o "*inode bitmap*".

Os *inodes* estão associados a um nome de um ficheiro através de estruturas de dados especiais chamadas *entradas de directórios* que são guardadas dentro de ficheiros do tipo "directório". A estrutura básica de um ficheiro deste tipo segundo os "standards" é:

```
struct dirent {
   int   inode;
   short rec_size;
   short name_len;
   char  file_name[NAME_LEN];
};
```

O campo "inode" desta estrutura contém um valor que é associado ao nome do ficheiro (que por sua vez fica guardado no vector "file_name"). Para poupar espaço o tamanho real do nome do ficheiro fica guardado no campo 'name_len' da estrutura e o espaço restante do vector "file_name" é utilizado pela próxima estrutura de um directório. O tamanho desta estrutura é geralmente arredondado para a mais próxima potência de dois e este valor é guardado no elemento "rec_size" da estrutura. Quando um ficheiro/inode é removido, o valor do elemento "inode" é alterado para "0" e o valor do campo "rec_size" da estrutura "dirent" anterior é aumentado de forma a

englobar o "*inode*" eliminado. Isto provoca o efeito de guardar o nome de ficheiros eliminados dentro dos ficheiros do tipo directório.
De cada vez que um ficheiro é associado a um outro ficheiro, um contador interno é incrementado. Da mesma forma que quando uma associação é removida, esse mesmo contador é decrementado. Quando esse contador atinge o valor "0" significa que já não existem mais referências para aquele "*inode*" dentro da estrutura do directório e o ficheiro é eliminado. Os ficheiros que tenham sido eliminados podem então libertar os recursos ocupados, blocos de dados e o próprio "*inode*". Isto é possível marcando os *bitmaps* apropriados.
Os próprios ficheiros do tipo directório estão logicamente organizados como uma árvore (começando no directório root "/"). Este directório root utiliza sempre o "*inode*" "2" do disco rígido de forma a que o kernel saiba localiza-lo e fazer o "mount" ao sistema de ficheiros.

Para poder fazer um "mount" ao sistema o kernel necessita de saber o tamanho e a localização das estruturas *meta-dados*. A primeira parte da *meta-dados*, o super bloco, é guardado numa localização conhecida. O super bloco contém informação variada , como por exemplo, o número de *inodes* e blocos, o tamanho de um bloco e outras informações. Baseado na informação lida do super bloco o kernel consegue calcular as localizações e tamanhos das tabelas dos *inodes* e os seus dados associados.

Concluímos, então, a explicação "básica" do funcionamento de um sistema de ficheiros genérico Unix. A partir de agora ,o leitor está apto para melhor perceber o funcionamento da análise forense.

3.2 – Análise-forense

A análise-forense digital a um sistema é geralmente feita por um qualquer motivo.Este motivo geralmente é de índole casual ou legal.
Quando o motivo da análise é legal significa que a segurança de um sistema foi comprometida e o analista tenta então detectar provas que expliquem o "como" do compromisso ou revelem o rasto do atacante.
Se o motivo é casual significa geralmente que a análise partiu de iniciativa própria do administrador ou analista o que dá a ambos uma maior versatilidade durante os testes a efectuar ao sistema.

Independentemente da razão, os seguintes passos devem ser efectuados:

1- Efectuar uma cópia de todo o sistema de ficheiros
2- Toda a informação contida nele é reunida
3- Esta informação é dividida em provas
4- As provas são examinadas;

Após reunir estas provas o analista deverá tentar:

1- Reunir informação sobre o(s) atacante(s)
2- Determinar o que aconteceu
3- Construir um quadro temporal do compromisso do sistema
4- Descobrir quais as ferramentas e exploits foram utilizadas para comprometer o sistema

Tal como expliquei no ponto anterior, de acordo com o funcionamento genérico de um sistema de ficheiros "Unix", quando um ficheiro é eliminado o "link" interno correspondente ao seu "inode" é decrementado até ao seu valor ser "0". Após o "inode" ser apagado, o kernel vai marcar os seus recursos como disponíveis para serem utilizados por outros ficheiros. No entanto, o "inode" continua a conter toda a informação que estava no ficheiro eliminado e os blocos de dados para o qual ele aponta continuam a alojar os conteúdos desse mesmo ficheiro. Isto vai acontecer até os inodes serem realocados e reutilizados de forma a que a data seja reescrita.

Através disto o leitor pode facilmente perceber que simplesmente removendo um ficheiro do sistema não o elimina na realidade.

Isto é o funcionamento normal da maior parte dos sistemas visto que demoraria bastante mais tempo a reescrever os blocos ocupados com dados miscelaneos ou com "zeros". É muito mais simples e rápido o sistema de ficheiros referenciar o espaço como utilizável de forma a ser reescrito.

Um dos programas bastante utilizado na ciência forense digital é o "**The Coroners Toolkit**" (TCT).

O analista forense percorre então o disco a um nível bastante baixo de forma a procurar referências de ficheiros eliminados recuperando-os para análise posterior.
Tal como já disse a maior parte dos atacantes são jovens inconscientes que não possuem grandes conhecimentos.
Quando estes jovens querem eliminar os seus rastos geralmente apenas eliminam os ficheiros com o software existentes no sistema, deixando assim provas e um rasto que poderá levar o analista direito a eles.

3.3 – Anti-análise Forense

A arte e ciência da anti-análise forense parte das falhas da análise forense.
Para o atacante poder camuflar-se completamente do analista forense é necessário conhecer o software especializado geralmente utilizado procurando assim falhas que possam ser usadas em seu proveito.

De acordo com o ponto anterior sobre a análise forense, o analista irá procurar ficheiros eliminados de acordo com o sistema de ficheiros mas ainda referenciados no disco rígido. Ora uma forma bastante simples de corromper a investigação é eliminado completamente os ficheiros a baixo nível.

Isto pode ser feito com software opensource disponível na internet como o **"The Defiler's Toolkit**" (TDT).

O sistema de ficheiros geralmente guarda um registo de toda a actividade "Input"/"Output" a ficheiros. Os analistas forenses tentam ver esse registo de forma a extrairem o máximo de informação deixada para trás pelo atacante. Além do facto de que as suas ferramentas de análise forense podem reportar incorrectamente informação sobre os dados encontrados, elas são completamente inúteis se os dados simplesmente não existirem no disco. Estas metodologias foram implementadas no

"TDT". Iremos então mostrar uma forma de erradicar por completo ficheiros do disco rígido através dele.

Num sistema de ficheiros Unix os seguintes sítios contêm informações sobre a existência de ficheiros:

1- inodes
2- entradas de directórios
3- blocos de dados;

Todos eles representam provas cruciais para o analista.

A maior parte das ferramentas existentes para erradicar por completo um ficheiro do sistema apenas eliminam as provas dos blocos de dados deixando assim os inodes e as entradas de directórios intactos.

O "TDT" contém dois programas utilizados para remover a informação de um disco rígido por completo. O **necrofile** e o **klismafile**.

3.3.1- O necrofile

O necrofile é um programa que permite procurar todos os "inodes" inutilizados ainda contendo informação (devido a uma eliminação deficiente por parte de um programa normal) eliminando-os depois por completo.
Para isso este programa utiliza como argumento a partição que contém os "inodes" a eliminar por completo.

Vamos então analisar o "output" do "TCT" e do "necrofile" como exemplos práticos do que falamos até agora.

```
[root@golfinho root]$ ./ils /dev/hda6
class|host|device|start_time
ils|XXX|/dev/hda6|1026771982
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_mode|\
st_nlink|st_size|st_block0|st_block1
12|f|0|0|1026771841|1026771796|1026771958|1026771958|100644|0|86|545|0
13|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|546|0
14|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|547|0
15|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|548|0
16|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|549|0
17|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|550|0
18|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|551|0
19|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|552|0
20|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|553|0
21|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|554|0
22|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|555|0
23|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|556|0
24|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|557|0
25|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|558|0
26|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|559|0
27|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|560|0
28|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|561|0
29|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|562|0
30|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|563|0
31|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|564|0
```

```
32|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|565|0
33|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|566|0
34|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|567|0
35|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|568|0
36|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|569|0
37|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|570|0
[root@golfinho root]$
```

Então agora utilizamos o "necrofile" para localizar e erradicar por completo todos os "inodes" eliminados.

```
[root@golfinho root]$ ./necrofile -v -v -v -v /dev/hda6
Scrubbing device: /dev/hda6
12 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
13 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
14 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
15 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
16 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
17 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
18 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
19 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
20 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
21 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
22 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
23 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
24 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
25 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
26 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
27 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
28 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
29 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
30 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
31 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
32 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
33 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
34 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
35 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
36 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
37 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
[root@golfinho root]$
```

E voltamos mais uma vez a executar o "TCT" para ver se realmente eliminamos os "inodes" do disco rígido com sucesso.

```
[root@golfinho root]$ ./ils /dev/hda6
class|host|device|start_time
ils|XXX|/dev/hda6|1026772140
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_model|\
st_nlink|st_size|st_block0|st_block1
[root@golfinho root]$
```

Sucesso! Neste momento os "inodes" foram eliminados por completo restando apenas as entradas de directório como possível rasto ou prova incriminatória.
Procedemos então ao uso do "klismafile" para as remover também.

3.3.2- O klismafile

Tal como o “necrofile” elimina os “inodes” com informação mas não referenciados pelo sistema de ficheiros, o “klismafile” está encarregado de apagar a última peça do puzzle que pode levar o analista ao atacante: as entradas de directório.
Sendo assim, o “klismafile” consegue eliminar com segurança as entradas de directório removidas por programas normais. Quando um ficheiro ou ”inode” é eliminado o conteúdo continua descriminado na estrutura do directório. Este programa procura “inodes” nessas circunstâncias e erradica-as por completo do sistema.
Quando o “klismafile” é executado necessita de um parâmetro. Esse parâmetro é o directório no qual deve procurar os inodes que estejam no estado descrito anteriormente.

Vamos então novamente debater ambas ferramentas de análise forense e anti-análise forense para ver como funcionam.

Utilizamos o “fls”, programa incluído no “TCT” para procurar entradas de directório apagadas:

```
[root@golfinho root]$  ./fls -d /dev/hda6 2
? * 0:  a
? * 0:  b
? * 0:  c
? * 0:  d
? * 0:  e
? * 0:  f
? * 0:  g
? * 0:  h
? * 0:  i
? * 0:  j
? * 0:  k
? * 0:  l
? * 0:  m
? * 0:  n
? * 0:  o
? * 0:  p
? * 0:  q
? * 0:  r
? * 0:  s
? * 0:  t
? * 0:  u
? * 0:  v
? * 0:  w
? * 0:  x
? * 0:  y
? * 0:  z
[root@golfinho root]$
```

Como podemos ver o “fls” encontrou entradas de directório que tinham sido eliminadas anteriormente. É então altura de executar o “klismafile” para eliminar por completo estas entradas:

```
[root@golfinho root]$ ./klismafile -v /mnt
Scrubbing device: /dev/hda6
cleansing /
-> a
-> b
-> c
```

```
-> d
-> e
-> f
-> g
-> h
-> i
-> j
-> k
-> l
-> m
-> n
-> o
-> p
-> q
-> r
-> s
-> t
-> u
-> v
-> w
-> x
-> y
-> z
Total files found:      29
Directories checked:    1
Dirents removed :       26
[root@golfinho root]$
```

Conseguimos remover com sucesso por completo 29 ficheiros.
Podemos confirmar isto executando novamente o "fls".

```
[root@golfinho root]$ ./fls -d /dev/hda6 2
[root@golfinho root]$
```

Sem "output" tal como esperavamos. Se isto fosse um sistema comprometido, a partir deste momento, o analista forense nunca iria apanhar o atacante visto que todos os seus rastos tinham sido eliminados.

3.4- Análise Forense HighTech

Embora tenha mostrado aqui formas de encontrar pistas deixadas por um atacante e formas de as eliminar por completo, devemos ter em conta a parte física da análise.
Existe material completamente "irreal" pelas suas capacidades que parece tirado de um filme de ficção científica.

Temos então dois tipos de microscópio capazes de ver o mais pequeno bit num disco rígido ou outro tipo de media qualquer.

O *Magnetic Force Microscopy* (MFM) utilizando o magnetismo e o *Scanning Tunneling Microscopy* (STM) limitando os electrões a estruturas artificiais de tamanhos na ordem dos nano, conseguem analisar informação tão pequena como bits.

Imagem 8: Magnetic Force Microscopy

Para o leitor que **provavelmente** nunca pensou poder ver um bit, deixo-lhe algumas imagens.

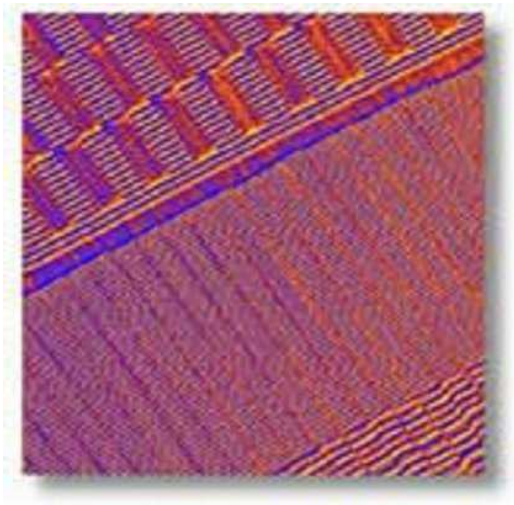

Imagem 9: bits num disco rígido

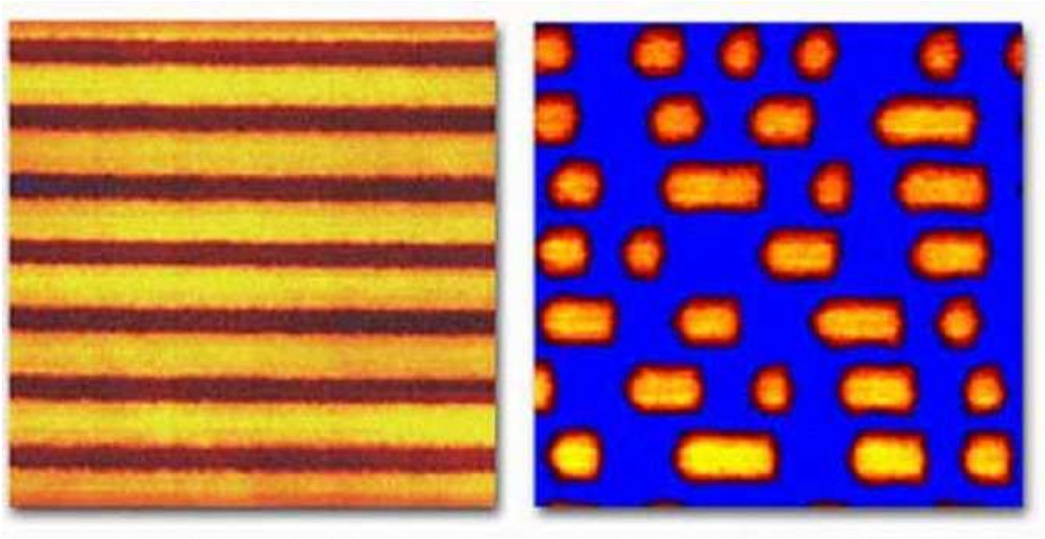

Imagem 10: bits num dvd-rw

É realmente fantástico e parece retirado de um filme de ficção científica.
Existem empresas que se dedicam a recuperar informação em material danificado.
Essas empresas reportam sucessos em caso de discos completamente carbonizados por um incêndio, terramotos, inundações, etc.

O que leva alguém com intuitos maléficos a pensar: será que alguma vez estamos completamente seguros da justiça?
Pois bem, em Portugal estamos. Vivemos num país de brandos costumes e quase sem legislação informática.

A única autoridade no que trata a assuntos de criminalidade informática em Portugal é a Polícia Judiciária.

Esta Polícia não tem um único departamento ou especialista em análise forense o que torna Portugal um paraíso para a criminalidade informática.

Os únicos casos de sucesso reportados são de fraude financeira e pedófilia. Se um "miúdo" penetrar num site e modificar a página principal da empresa ou Instituição, o mais certo é nunca ser julgado.
Costumo dizer que a única coisa organizada em Portugal é o crime. Felizmente não existe vertente de criminalidade informática organizada no nosso país. A maior parte dos ataques que ocorrem são feitos por miúdos com tempo a mais e sem amigos.

4-HoneyPots

Ainda sobre a análise forense convém falar sobre os chamados "*honeypots*".
Um "honeypot" é um sistema cuja segurança é especialmente desenhada para ser testada,atacada e comprometida. Pretende-se dessa forma analisar padrões nos ataques, preve-los e encontrar novas falhas.
O honeypot vem da ideia do "urso que vai atrás do pote de mel". Baseando-se nesta ideia, alguns "whitehats" (os hackers sem intuito malicioso) começaram a desenvolver estes sistemas para apanharem actividades dos "blackhats" (desta vez o

contrário dos “whitehats”) decifrando assim como trabalham e/ou o que os motiva para causarem danos num sistema.

Já agora como curiosidade, a definição de whitehat e blackhat foi baseada na ideia dos filmes antigos de cowboys. O cinema não tinha cor e os bons e os maus também não. Por isso os cowboys “bons” utilizavam chapéus brancos e os “maus” utilizavam a cor preta.

Um honeypot é geralmente apenas um sistema antigo com bastantes serviços, muitas falhas e nenhuma actualização que corre num ambiente controlado de forma a interceptar todos os passos do atacante. Este é um óptimo exemplo de análise forense.

4.1-O vmware e a análise-forense

O vmware é um software que permite executar vários sistemas operativos simultaneamente num único computador. Isto é o ideal para um sistema do tipo “honeypot”.

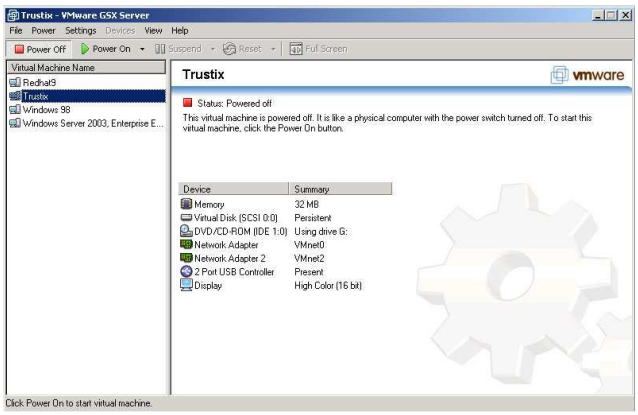

Imagem 7: Vmware GSX Server

Este software permite utilizar uma partição “física” ou criar um espaço virtual. Este espaço virtual é na realidade um único ficheiro embora os sistemas operativos executados o detectem como um disco rígido. Esta aplicação funciona como uma “gateway” entre o sistema operativo “emulado” e o hardware real criando periféricos virtuais para cada um dos sistemas executados.

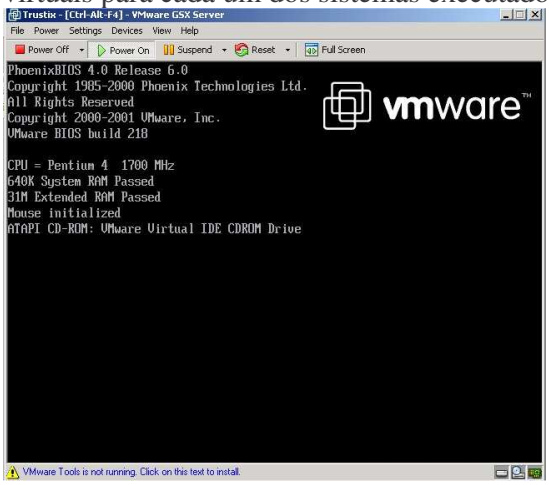

Imagem 8: boot do vmware

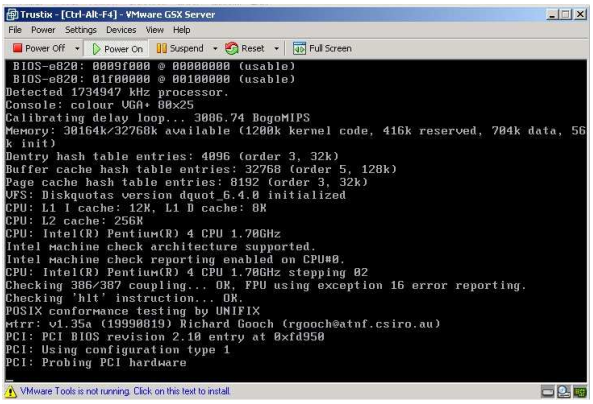

Imagem 9: boot do linux no vmware

Outra característica deveras interessante para o uso do *vmware* nos honeypots é a possibilidade de fazer um "standby" a um sistema em execução, o que o torna ideal para uma análise forense.

Eu utilizo o vmware como firewall em minha casa. O sistema operativo base é um windows 2000 profissional e dentro executo um Linux.
Este Linux está encarregue de fazer o routeamento e filtragem de pacotes entre "a internet e a minha rede local".
Na minha opinião é a firewall perfeita visto que se encontra num sistema completamente controlado e emulado (sem acesso directo ao hardware).

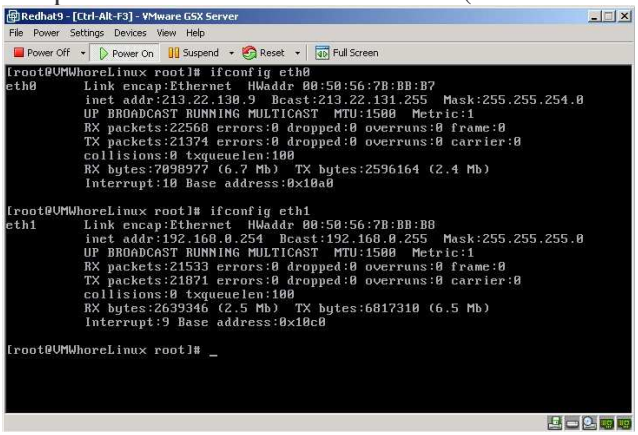

Imagem 10: Interfaces de rede de um sistema operativo linux emulado.

Esta é então uma boa opção para proteger o seu sistema e estudar os atacantes analisando assim as últimas técnicas utilizadas por eles.

4.2- O honeyd

Como desenvolvimento da ideia do projecto "honeypot" foi (e está neste momento) a ser desenvolvido o "honeyd".
O "honeyd" é um serviço cuja finalidade é emular outros serviços de forma a enganar um atacante iludindo sobre o sistema que está a tentar comprometer.

A ideia final deste projecto é criar um sistema tão completo e que pareça tão real que seja capaz de iludir qualquer atacante.
Todas as actividades são interceptadas e registadas para serem levadas a cabo análises forenses.
A grande diferença entre o "honeypot" e o "honeyd" é que este é como se um "super honeypot" completamente virtual sem a possibilidade de o atacante causar danos no servidor que o utiliza.

Este sistema utiliza configurações bastante simples de utilizar e já permite emular alguns sistemas operativos conhecidos como por exemplo o windows, linux, cisco IOS.

Um ficheiro de configuração possível para emular um servidor windows NT seria:

```
•create windows
•set windows personality "Windows NT 4.0 Server SP5-SP6"
•set windows default tcp action reset
•set windows default udp action reset
•add windows tcp port 80 "perl scripts/iis-0.95/iisemul8.pl"
•add windows tcp port 139 open
•add windows tcp port 137 open
•add windows udp port 137 open add windows udp port 135 open
•set windows uptime 3284460
•bind 192.168.1.101 windows
```

Outra configuração para emular um servidor Linux poderia ser:

```
•create linux
•set linux personality "Linux 2.4.16 - 2.4.18"
•set linux default tcp action reset
•set linux default udp action reset
•add linux tcp port 110 "sh scripts/pop3.sh"
•add linux tcp port 25 "sh scripts/smtp.sh"
•add linux tcp port 21 "sh scripts/ftp.sh"
•set linux uptime 3284460
•bind 192.168.1.102 linux
```

E um pseudo router CISCO poderia ser criado através de:

```
•create router
•set router personality "Cisco IOS 11.3 - 12.0(11)"
•set router default tcp action reset
•set router default udp action reset
•add router tcp port 23 "/usr/bin/perl scripts/router-telnet.pl"
•set router uid 32767 gid 32767
•set router uptime 1327650
•bind 192.168.1.104 router
```

Estas configurações dizem ao serviço (honeyd) para abrir determinadas portas, executar alguns scripts de forma a enganar os atacantes, mostrar um determinado "uptime" (tempo total que o sistema se encontra ligado), em suma, moldar a sua "personalidade" ao sistema em questão.

5-Precauções a tomar numa análise forense

Existem alguns passos vitais a seguir caso um sistema tenha sido comprometido e se queira proceder a uma **correcta** análise forense.

Geralmente quando um servidor é penetrado, a grande prioridade é "voltar a garantir o funcionamento mínimo do servidor...o mais rapidamente possível". É claro que isto trás desleixos e, muito provavelmente, novas portas para os atacantes entrarem.

Quando um sistema é penetrado deve-se proceder a uma análise minuciosa detectando **como** o atacante entrou, **quando** e, porque não se possível, o **porquê** de ter entrado nesse sistema.

Então as precauções principais a tomar são:

1-Desligar o sistema da rede
2-Remover os discos e fazer backups com software especializado (que faça cópias integrais dos discos, inclusive do "slack space"
3-Informar as devidas autoridades
4-**Nunca** voltar a usar o sistema comprometido sem antes proceder a uma reinstalação completa do sistema (visto que o mais certo é o atacante ter modificado ficheiros chave do sistema e instalado "rootkits")

6-Conclusões

Cada vez mais vivemos num mundo dominado pelas novas tecnologias. Estas tecnologias estão em constante mutação e evolução.
Devemos ter sempre em conta que com os avanços na **tecnologia** surgem os avanços na **anti-tecnologia**, E, mais importante ainda, na maior parte dos casos o último grito na tecnologia é controlado pelo *equipa adversária* e dependemos das situações aprendidas com eles para nos defendermos. É isso que torna a análise forense uma ciência tão aliciante e importante nas nossas vidas.

7-Agradecimentos

Gostaria de agradecer aos meus pais e à minha namorada pela atenção que não lhes dei enquanto escrevia este artigo e ao leitor por o ter lido.

Bibliografia e links úteis:

[1] Defeating Forensic Analysis on Unix
http://www.phrack.org/phrack/59/p59-0x06.txt
[2] HoneyPot project
http://project.honeynet.org
[3] VMWare
http://www.vmware.com
[4] Webopedia
http://www.webopedia.com
[5] Free Forensics Tools for Unix
http://online.securityfocus.com/infocus/1503
[6] Steganography
http://steghide.sourceforge.net/download.html
[7] ADS – Alteranate Data Streams
http://www.diamondcs.com.au/streams/streams.htm
[8] Linux Data Hiding and Recovery
http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html

[9] TCT – The Coroners Toolkit
http://www.fish.com/tct/
[10] Basic Steps in Forensic Analysis of Unix Systems
http://staff.washington.edu/dittrich/misc/forensics/
[11] HardDisk Recovery
http://www.harddisk-recovery.com/