Álvaro Vilobaldo Rios da Silva

Antiforense com uso de rootkits

São Paulo

2013

Álvaro Vilobaldo Rios da Silva

Antiforense com uso de rootkits

Monografia de Conclusão de Curso apresentada à Universidade Presbiteriana Mackenzie de São Paulo como requisito parcial à obtenção do título de Especialista em Computação Forense do Curso Lato Sensu em Computação Forense

Orientador:
Ivete Irene dos Santos

UNIVERSIDADE PRESBITERIANA MACKENZIE

São Paulo

2013

Monografia sob o título Antiforense com uso de rootkits, desenvolvida por Álvaro Vilobaldo Rios da Silva, e aprovada em 20 de abril de 2013, São Paulo capital, pela banca constituída por:

Ivete Irene dos Santos
Orientadora

Ana Cristina Azevedo
Orientadora

Epígrafe

"Se você conhecer o inimigo e a si mesmo, não precisa temer o resultado de uma centena batalhas. Se você se conhecer a si mesmo, mas não o inimigo, para cada vitória você também sofrerá uma derrota. Se você não conhecer nem o inimigo, nem a si mesmo, você sucumbirá em todas as batalhas." (**??**)

Resumo

Esta monografia é um estudo sobre antiforense com utilização de rootkits visando mostrar como funciona um rootkit e como ele pode ser usado na antiforense. Definições de antiforense e rootkits com suas práticas comuns e respectivos históricos junto com algumas técnicas de subversão demonstrarão que o uso combinado de técnicas antiforenses incorporadas a rootkits cria mecanimos de destruíção de provas de forma sistemática e eficiênte.

Palavras-chave: antiforense, rootkit, forense, provas, evidências.

Abstract

Write here the English version of your ’Resumo’...

keywords:

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

A monografia apresenta uma incursão na antiforense focada na utilização de rootkits em ambientes Microsoft principamente Windows, visando mostrar seu funcionamento, as práticas mais comuns e como o rootkit pode ser usado na antiforense.

O rootkit é em geral utilizado por atacantes avançados com fins maliciosos e oferece grande obstáculo para ser detectado, pois seu uso força o perito a ter um conhecimento amplo além de usar técnicas sofisticadas. Entender o seu funcionamento é determinande para uma pericia bem sucedida. Para desenvolver esse tema foi utilizada vasta bibliografia, exemplos práticos e aplicações de rootkits comuns.

Ao longo dos capítulos são desenvolvidos os conhecimentos para o entendimento do tema proposto, onde cada um foi dividido em um assunto específico. No primeiro capítulo será mostrado uma visão ampla do que é a ciência forense e como ela é aplicada na computação. No segundo capítulo são apresentados conceitos de antiforense e os tipos de antiforense. Por fim no terceiro capítulo conduz a uma visão do funcionamento de um rootkit e algumas técnicas para subverter os sistema.

1.1 Tema

Antiforense em Windows 7 com uso de rootkits.

1.2 Hipótese(s)

Técnicas podem ser usadas para ocultar processos ou ações? Rootkits e bombas lógicas são exemplos de como um usuário avançado pode disfarçar, dificultar ou impossíbilidar a ação de um perito forense na obtenção de provas.

1.3 Metodologia

1.4 Justificativa

Por que invadir um país armado arriscando vidas de centenas se você pode destruir centrífugas de enriquecimento de urânio de forma sigilosa do outro lado do planeta? Conforme o mundo se digitalizou se digitalizaram-se também as suas ameaças, onde antes se podia ver mesmo que por instantes misseis ou bombas sendo lançada hoje temos inúmeras ameaças invisíveis que podem causar tanto estrago quanto, contudo pelo príncipio de Locard o problema dessas ameaças invisíveis é que elas podem não ser tão invisíveis assim, levando a procura de metodos antiforense mais eficazes.

O perito deve estar preparado para a ação de um usuário avançado que conheça bem o sistema operacional atacado,comprometido ou usado. Mesmo que não seja algo corriqueiro na rotina da grande maioria dos profissionais, encontrar um atacante de alto nível trará novos desafios e obstáculos tão poucos corriqueiros. Saber como dificultar ou impossibilitar o trabalho do perito é como ele poderá evitar a armadilha de achar que no corpo (corpo de delito) investigado não existe nada.

Sendo assim, esse trabalho poderá ajudar a traçar um processo bem elaborado de trabalho que seja rápida e eficiente sem deixar brechas que permitam ou ajudem ações antiforenses. Com um processo bem definido o perito tende a diminuir o tempo de análise e um melhor aproveitamento das mesmas. Provendo mais qualidade com mais precisão.

Durante a perícia o perito pode se deparar com um ataquante avançado que utiliza técnicas bem sofisticadas para esconder ou destruir evidências imposibilitando ou dificultando a obtenção de provas e comprometendo a qualidade das mesmas quando são obtidas. Para tal feito o ataquante pode se valer de rootkits.

Ao estudar o comportamento dos rootkits e entender como ele pode ser usado na antiforense, permite ao perito lidar corretamente com rootkits, como por exemplo, desenvolvendo rotinas.

O perito deve estar preparado para a ação de um usuário avançado que conheça bem o sistema operacional atacado,comprometido ou usado. Mesmo que não seja algo corriqueiro na rotina da grande maioria dos profissionais, encontrar um atacante de alto nivel trará novos desafios e obstáculos tão poucos corriqueiros. Saber como dificultar ou impossibilitar o trabalho do perito é como ele poderá evitar a armadilha de achar que no corpo (corpo de delito) investigado não existe nada.

Por que invadir um país armado botando vidas de centenas em risco se você pode destruir centrífugas de enriquecimento de urânio de forma sigilosa do outro lado do planeta? Conforme o mundo se digitalizou se digitalizaram-se também as suas ameaças, onde antes se podia ver mesmo que por instantes misseis ou bombas sendo lançada hoje temos inumeras ameaças invísiveis que podem causar tanto estrago quanto, contudo pelo príncipio de Locard o problema dessas ameaças invísiveis é que elas podem não ser tão invisiveis assim, levando a procura de metodos antiforense mais eficazes.

Sun Tzu (544 a.C. - 456 a.C.) há mais ou menos 500 anos de cristo já dizia que se conhecer a si mesmo e ao adversário não temerá o resultado de mil batalhas.

Conhecendo como essas técnicas podem ser usadas o períto pode se precaver e saber agir no momento de detectar o uso dos mesmos. O perito deve tomar alguns cuidados na análise de computadores, pois o mesmo pode ter sido usado por um usuário avançado o que muda razoavelmente a sua abordagem.

1.5 O Problema de Pesquisa

Técnicas podem ser usadas para ocultar processos ou ações. Como um rootkit funciona e como ele pode ser usado para antiforense?

Pode ajudar a traçar um processo bem elaborado de trabalho que seja rápida e eficiente sem deixar brechas que permitam ou ajudem ações antiforenses. Com um processo bem definido o perito tende a diminuir o tempo de análise e um melhor aproveitamento das mesmas. Provendo mais qualidade e mais precisam nas futuras análises.

1.6 Objetivos

Pode esclarecer diversos tópicos obscuros a respeito do que pode ser encontrado em investigações forenses quando o perito se ve de frente com ameaças persistentes e elaboradas, ou seja, ajuda a resolver casos onde existiu antiforense.

1.6.1 Objetivo Geral

Apresentar o funcionamento de rootkits voltados para antiforense.

1.6.2 Objetivo Específico

Apresentar o que é ciência forense, para que serve a ciência forense, o que é computação forense e citar a legislação que garante a existência do perito no Brasil; Mostrar o que é antiforense computacional, categorizar os tipos de antiforense computacional e descrever os tipos de computação forense; e Expor o que é um rootkit, como ele funciona e como ele pode impedir a formação de provas.

1.7 Delimitações do Estudo

Análise de rootkits em ambientes Windows 7 voltados para restringir ou destruir provas.

1.8 descricao dos capitulos

2 *Forense Computacional*

A humanidade sempre teve assuntos divergentes, onde em outrora esses assuntos eram resolvidos com o mais forte ou hapto sobrepujando a vontade doutro, com o advento da civilização se convencionou o uso de um mediador que em teoria deve ser alguem neutro.

Quando trata-se de assuntos de disputa de interesse, espera-se que o mediador busque e chegue o mais próximo possível da verdade antes de tomar uma decisão. Hoje é dado esse poder de mediação ao juiz, contudo em diversos conflitos se faz necessário conhecimentos técnicos especificos. Nesses casos ele é auxiliado por um especialista ou perito no assunto técnico em discussão. Para tal, esse especialista utiliza-se da ciência forense para trazer luz os fatos respaldando a decisão do juiz sobre o assunto.

Segundo **??**) ciência é um "corpo de conhecimentos sistematizados adquiridos via observação, identificação, pesquisa e explicação de determinadas categorias de fonômenos e fatos, e formulados metódica e racionalmente." e forense é "relativo aos tribunais e à justiça". Logo as ciências forenses é a utilização da ciência "à análise de vestígios, no intuito de responder às demandas judiciais" (**??**, p. -3).

De certo a área médica foi a primeira a ser requisitada em tribunais construindo técnicas que levaram ao desenvolvimento ao longo dos anos da medicina legal. No Império Romano médicos eram chamados para lucidar mortes diz **??**), outro exemplo histórico da importância do parecer técnico pode ser visto no *Código Criminal Carolino* feito em 1532 por Carlos Magno que definia a análise médica em determinados crimes.

Com o avanço da tecnologia e o valor da pespectiva especializada, foi natural que outras áreas também fossem utilizadas no foro. A computação forense, apesar da tecnologia e inovação inatas, conceitualmente ainda se propõe a grosso modo a usar a ciência para mostrar fatos demandados judicialmente, como podemos ver em **??**), "[...] computação forense tem como objetivo principal determinar a dinâmica, a materialidade e autoria de ilícitos ligados à área da informática, tendo como questão principal a identificação e o processamento de evidências digitais em provas materiais [...], por métodos técnicos-científicos, conferindo-lhes validade

probatória em juízo.”.

3 *Antiforense Computacional*

??) define antiforense como método para prevenir ou agir contra a ciência usada a favor das leis civis e criminais que são aplicadas por órgãos como a polícia. **??**), amplia essa ideia mostrando que é mais que uma técnica usada, é uma abordagem crimosa. Assim todas as tentativas de afetar negativamente a existência, quantidade e / ou qualidade da evidência de uma cena de crime, ou fazer a análise e exame das provas difíceis ou impossíveis de realizar, para **??**) será considerada antiforense. Dessa forma concluímos resumidamente que a antiforense computacional pode ser definida como qualquer ação praticada para obstruir, dificultar ou destruir evidências ou provas no âmbito computacional.

Dentre as modalidades de antiforense computacional, **??**) as categoriza em cinco grandes grupos, sendo eles: destruição de dados, ocultação de dados, corrupção de dados, fabricação de dados e eliminação da fonte de dados.

4 Rootkits

Malware é a junção das palavras em inglês *malicious* e *software*, termos em inglês que pode ser traduzido livremente como aplicativo malicioso. Dessa forma podemos considerar como malware os vírus, worms, trojans, botnets kit dentro outros, contudo o inverso não é valido, ou seja, todo vírus é um malware, mas nem todo malware é um vírus.

Os vírus e worms são feitos para se espalha a diferença está em como eles se espalham. O vírus precisa ser ativado ou executado pelo usuário e em geral fica atrelado a algum executavel que pode ser ou não um programa legitimo subvertido (**??**), ao contrário do worm que não precisam da ação direta do usuário para se espalhar e permanece apenas na memória (**??**).

Uma *botnet* é uma rede de computadores controlados por cybercriminosos(**??**). Basicamente quando um computador é infectado pelo agente da *botnet* o mesmo se torna parte da rede e é controlado pelo dono da botnet.

No universo *UNIX*[1] ou *UNIX-like*[2] a conta de usuário com menor restrição de segurança é referênciada como conta *root* sendo que em alguns sistemas o nome de usuário é literalmente root, mas isso é apenas uma convenção histórica do que uma imposição (**??**). Enquanto *kit* significa conjunto de peças (**??**).

Rootkit pode ser visto como um "kit" composto de pequenos programas úteism por exemplo, binários, scripts, arquivos de configuração que permitem a um atacante manter o acesso "root". Em outras palavras, um rootkit é um conjunto de programas e de código, que permite a presença permanente ou consistente, não detectável em um computador (**??**).

Essa coleção de ferramentas que permitem aos invasores ocultarem suas atividades em um computador, de modo que eles podem secretamente monitorar e controlar o sistema por um período prolongado, ou seja, existem três serviços que o são inerentes dos rootkits, ocultação, comando e controle (C2) e vigilância (**??**). Ocultação, o rootkit deve passar despercebido, sem que o usuário e/ou antivírus o detecte; vigilância ou monitoramento basicamente consiste em

[1] Um sistema operacional multiusuário amplamente utilizado.
[2] Sistemas Operacionais baseados no Unix, como o GNU/Linux

acompanhar as ações do usuário, o rootkit tem que ser capaz de saber o que o usuário está fazendo; e comando e controle, permite ao dono do rootkit o controle remoto sobre o mesmo, definindo suas ações e direcionando. Desses três serviços o mais importante para o rootkit é a furtividade, pois um rootkit detectável vai durar muito pouco.

Como pode ser visto cada um desses agentes subversivos tem definições e características próprias diferentes e uma em comum todos eles subvertem o sistema de alguma forma, contudo é bom resaltar que nem todo rootkit é um malware, pois existem aplicações legítimas para o mesmo, como de computadores coorporativos e inclusive aplicações investigativas.

Os primeiros rootkits apareceram em de 20 anos atrás no fim dos anos 80 e início dos 90, quando alguns foram percebidos comportamentos anormais em computadores como espaço em disco utilizado sem identificação, conexões de rede não-listadas e uso anormal do CPU (**??**).

Basicamente o processador pode receber dois tipos de exceções, uma gerada por hardware chamada de externa e interrupções geradas por programas(**??**, p. 6-2 Vol. 3A). A grosso modo, quando uma interrupção ou exceção ocorre o processador usa o endereço de memória armazenado no indice correspondente a interrupção lançada que por definição aponta para uma *procedure* que trata a interrupção (**??**), ou seja, esse endereço na memória vai direcionar o processamento a *procedure* especifica para a interrupção ou exceção que ocorreu. O rootkit pode interceptar uma chamada a tabela estrutura de dados com indices dessas tratativas como *Interrupt Vector (IVT)* em modo real ou a *Interrupt Descriptor Table (IDT)* em modo protegido, ambas as estruturas possuem funções similares, apesar da forma como trabalham ser muito distinta.

tipos de rootkit http://www.terena.org/activities/tf-csirt/meeting27/oesterberg-rootkits.pdf

falando assim parece uma tarefa fácil realizar qualquer um desses metodos...

Jamie Butler, the creator of the FU rootkit

O uso de rootkits é limitado a usuários avançados sendo que seu desenvolvimento exige conhecimentos profundos tanto de arquitetura de computadores quanto do funcionamento do Sistema Operacional que o mesmo vai corromper.

Rootkits proporcionam uma grande variedades de opções e pode ser utilizado para quase tudo,

vem sendo utilizado historicamente para subverter sistemas dando permissão

O fato de alguns eventos e rootkits terem ficados famosos, como é o caso do stuxnet(), torna esse assunto muito interessante de ser abordado, porque

A grosso modo rootkits sempre tentarão esconder sua presença e seus rastros, logo a anti-forense desde o inicio já é parte ...

não são novidades e sua aplicação já não é

e durante o avanço do trabalho, poderão ser notadas diversas formas de detecção de rootkits além de indícios de seu uso em análises.

5 *Antiforense com Rootkits*

Quando o perito vai realizar a coleta do material para a análise, ele pode encontrar dois cenários o corpo de delito que na ciência forense computacional quase sempre consiste em computadores, pode estar ligado ou desligado. Quando o mesmo está ligado é possível realizar a análise viva, entretando a análise *live* de um ambiente com rootkit pode gerar dados falsos ou imprecisos.

Análise *live* ou análise viva, consiste em uma análise focada em extrair e examinar dados volatéis (**??**), esses dados volatéis são perdidos ao desligar o computador, como por exemplo, o contéudo de registradores do processador, dados na *cache*, dados na memória etc. Essa abordagem possui diversas vantagens entre elas estão extração de dados voláteis como já citado, triagem de equipamentos, triagem de dados, preservação de dados criptografado e a possibliidade de estabelecer flagrante (**??**).

É importante ressaltar que toda ferramenta ou *hardware* utilizado para a coleta de dados em equipamentos que ainda estão em execução, vai depender de dados fornecidos pelo equipamento periciado, ou seja, em algum momento essas ferramentas forenses vão requisitar dados de um sistema comprometido e passível de alguma interceptação mesmo quando a solução utilizada para coleta é um hardware (**??**). Logo em um ambiente dominado por um rootkit não é um ambiente confiável, pois ele pode interceptar e alterar chamadas das ferramentas de análise, impossibilitando uma coleta real e fiel.

A metodologia empregada na análise de um ambiente supostamente com um rootkit varia muito, mas em termos gerais a análise *Post mortem* é menos arriscada e indicada para casos de rootkits. As medidas antiforense que podem ser tomadas pelo rootkit dependem do ambiente de execução e engloba também a equipe que o administra. No melhor dos cenários para o atacante é quando o sistema é administrado por leigos, não capacitados, sobrecarregados e/ou os sistemas não são atualizados com frequencia tornando mais fácil para o rootkit esconder-se exigindo menos do atacante. No pior dos cenários o administradores são altamente qualificados, realizam rotinas frequentes, esta sempre atualizado e é auditorado (**??**). Entender isso pode

indicar ao perito o quão complexo o rootkit é, pois quanto mais próximo do pior cenário mais avançado tem que ser o rootkit o que pode caracterizar um *Advanced persistent threat (APT)* ou ameaça avançada e persistente, ou seja, o ataque utilizando o rootkit foi orquestrado e planejado apenas para esse alvo específico.

Não existe técnica 100% antiforense, então o objetivo do atacante é tornar a análise cansativa para induzir ao erro podendo adicionar inclusive falsos rootkits ou malwares para despistar, iludir ou levar a conclusões precipitadas. Em (**??**) são mostrados passos para a análise *Post mortem* onde se suspeita de rootkit:

- Clone do *hard disk drive (HDD)*;
- Recuperar arquivos;
- Coletar metadados dos arquivos;
- Retirar arquivos conhecidos;
- Análise estática dos executáveis suspeitos desconhecidos; e
- Análise dinâmica dos executáveis suspeitos.

Esses passos pode ser vistos como filtros, onde cada passo retira cada vez mais ruído, eliminando em cada etapa arquivos superfulos. A seguir serão descritos cada um dos passos e alguns dos problemas que o mesmo pode gerar.

5.1 Cópia forense do HDD

A primeira fase e mais importante é a coleta, que nesse caso é clone do HDD. O clone do HDD, pode ser feito com software, como o dd[1] ou por hardware específico para clonagem de HDDs. Por via de regra essas ferramentas são homologados por órgãos como *National Institute of Standards and Technology (NIST)* que atestam a qualidade e eficácia dessas ferramentas em produzir cópias fidedignas que podem ser usadas em disputas legais como evidências. Basicamente todos essas ferramentas fazem cópias de setor por setor de todo o disco, para que o perito possa trabalhar nessa cópia preservando o HDD original. Essas cópias já foram exaustivamente usadas em tribunais e são muito úteis e indispensáveis para a análise do perito. Hoje sua eficácia não é questionada, pois essa cópia ou clone tem os dados idênticos ao do HDD objeto de análise

[1] http://pubs.opengroup.org/onlinepubs/9699919799/utilities/dd.html

e essa igualdade pode ser atestada com *checksums* como os algoritmo de *hash MD5* ou *SHA1* (**??**). Essa abordagem pode ter alguns empecilhos e o rootkit pode não vir a ser clonado junto com os dados do HDD.

Primeiro e mais óbvio é que o rootkit pode nem estar no disco clonado, como estar somente na memória ou ser alocado em um outro dispositivo. O rootkit poder ser projetado para ficar somente na memória, entretando isso pode impedir que o mesmo persista já que é apagado sempre que o computador desliga (**??**). Contudo o mesmo esteja num parque de máquinas que permita infectar o computador toda vez que este for iniciada, como em grandes *data centers* em geral nunca tem todas as máquinas desligadas o rootkit pode ficar vivo na rede e persistir infectando novamente as máquinas assim que elas iniciarem.

Num *Personal Computer (PC)* durante o *boot* ele inicia o *Basic Input/Output System (BIOS)* fornece suporte básico para os principais periféricos como teclado e vídeo em modo texto e inicia o *Power-On Self-Test (POST)* que além de realizar alguns testes inicia, com base das configurações do SETUP salvas na memória CMOS, todos os circuitos perífericos, vídeo, Sistema Operacional e passa o controle para o sistema operacional (**??**). Durante todo esse processo o computador é executado em *Real Mode*, e como já foi descrito anteriormente, em modo real os programas não são isolados uns dos outros como no *Protected Mode*, logo qualquer programa em execução pode acessar todo conteúdo na memória. O rootkit pode persistir no *firmware*[2] de algum periférico, como a placa de vídeo (**??**) ou placa de rede (**??**), dessa forma que tornariam o clone do HDD inútil além de ter acesso em *Real Mode* na inicialização do computador.

O perito deve ter em mente ainda que o rootkit pode estar em áreas reservadas do disco como *host protected area (HPA)* ou *Device configuration overlay (DCO)*, essas areas são desenvolvidas para não serem modificadas ou mesmo acessadas por usuários, BIOS ou Sistema Operacional (**??**). Dessa forma dependendo da ferramenta utilizada para clone do HDD pode ser que essas áreas não sejam clonadas ((**??**) e (**??**)), inutilizando o clone do HDD.

Com o disco clonado o perito deve se ater a outros fatores, como criptografia, arquivos salvos em estruturas não convencionais etc. O disco clonado pode ter *full disk encryption* o que significa que todos os dados no disco estão criptografados (**??**) e que ao clonar o clone, como é uma cópia fidedigna do original, também tem todos os dados criptografados. Logo o disco deverá ser decriptado antes da análise. É improvável que um rootkit criptografe todo o disco da vítima, pois chamaria muita atenção e iria de encontro a uma de suas premissas que é se ocultar no sistema. Então quando disco está encriptado, ele provavelmente foi feito pela vítima

[2]Todo programa salvo numa *Read-only memory (ROM)* é chamado de firmware

e consequentimente a maior interessada no trabalho do perito facilitando a decriptação do disco. O mais comum nesses casos quando envolve criptografia é do rootkit encriptar somente o seu binário, mas isso vai ser discutido adiante na análise estática.

5.2 Recuperar arquivos

O próximo passo a ser tomado deve ser recuperar todos os arquivos do disco, isso incluí todos os arquivos *Master Table File (MFT)* deletados ou não, seguido por *data carving* com especial preocupação com:

- Assinaturas de executáveis;
- Fragmentos de arquivos;
- Data streams alternativos; e
- Slack space;

Todos os arquivos tem um especificação que define o formato que o arquivo deve ter para ser corretamente interpretado, essas especifição define um padrão de como o *bits* são organizados internamente e é chamado de *file format* (**??**). Esse formato também é usado por ferramentas de *data carving* para encontrar arquivos. Por exemplo a especificação do *Portable Executable (PE) File Format* (**??**) determina que todos os programas comecem com um pequeno executável *MS-DOS*. Esse pequeno executável inicia com o *Magic Number* 0x5A4D ou *MZ* em *ASCII*,as iniciais de Mark Zbikowski um dos arquitetos originais do MS-DOS (**??**), e uma mensagem como pode ser vista na figura **??**. A função desse pequeno programa é apresentar a mensagem "This program cannot be run in DOS mode".

Programas que fazem *data carving* buscam essas estruturas definidas, para encontrar executáveis. Contudo o rootkit pode usar programas como o *Transmogrify* do *Metasploit Anti-Forensics Project (MAFIA)* que é capaz de mascarar um arquivo em qualquer assinatura (**??**), de forma a deixar um executável no formato de um arquivo texto por exemplo.

Outra forma do rootkit esconde-se de um *data carving* é utilizando-se de uma organização de arquivo não convencional, como usar fragmentos de diversos arquivos que só fazem sentido quando organizados de determinada forma, salvar seus arquivos em slack space buscando e indexando-os em um *file system* próprio (**??**), ou ainda usando *features* de um sistema de arquivos como as *streams* do NTFS (**??**). Entretando por mais eficinte que seja o método utilizado,

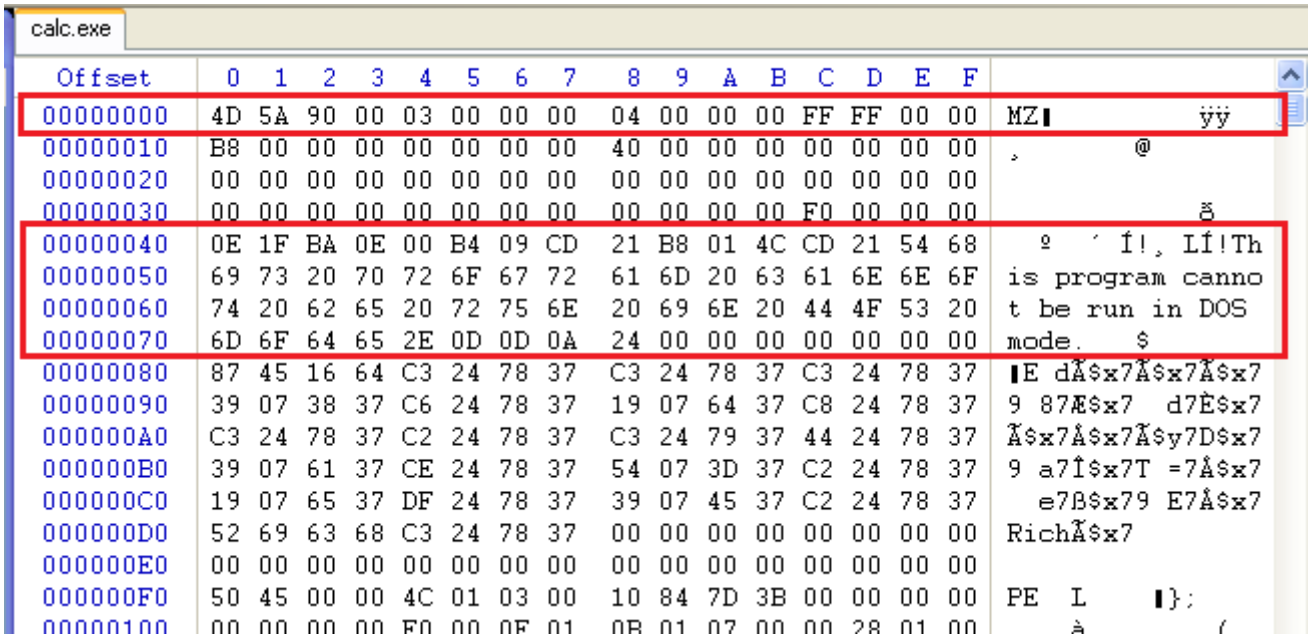

Figura 5.1: Hexadecimal da Microsoft Calculator v. 5.1 visto no WinHex 16.3 SR-2

o rootkit em algum momento tem que executar, logo em algum ponto o rootkit tem que trabalhar com estruturas convencionais para que o sistema operacional o interprete, mesmo que seja apenas um *stub* (ver análise estática).

5.3 Coletar metadados dos arquivos

Nessa etapa é feita a coleta dos metadados de todos os arquivos, segue os principais:

- Hash;
- MAC time;
- Localização completa do arquivo em disco (*Path*); e
- Nome;
- Tamanho;

É importante que se tenha ao menos esses metadados dos arquivos recuperados. Os próximos passos podem depender de um desses elementos. O *hash* é fundamental para o próximo passo, o tamanho vai ser abordado na análise estática, o nome e *path* são obviamente para sua identificação e para localização, e o *MAC (Modify, Access, Create) time* pode vir a ser utilizado para desenvolver uma *timeline*, ferramenta útil e poderosa para favor do perito.

A análise da *timeline* é um importante passo para todo processo tradicional de investigação, com base nela um perito forense computacional pode extrair informações cruciais para o caso (**??**). Criar uma *timeline* de eventos com base nos *MAC time* dos arquivos ajuda a entender o ciclo de contaminação, além de levantar suspeitas sobre varios dados inconsistentes, como por exemplo, porque o papel de parede do Windows tem a data de modificação ou criação recente? Além de ajudar a entender como o rootkit pode ter se mantido no ambiente analisado. Dessa forma é possível recriar a cadeia de eventos gerando ao mesmo tempo contexto.

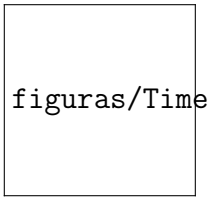

Figura 5.2: Arquivo com MAC time original (**??**)

Para (**??**) um dos principais desafios na análise de *timeline* envolve a fácil manipulação. Para não levantar suspeitas, ou dificultar o processo de análise da *timeline*, o rootkit pode ser programado a alterar o *MAC time* quando necessário. O programa *Timestomp* também do *MAFIA* e hoje integrado ao *Meterpreter* é capaz de alterar esses metadados como é mostrado na figura **??**.

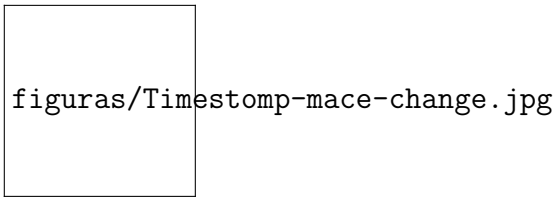

Figura 5.3: Utilização do timestomp (**??**)

5.4 Retirar arquivos conhecidos

No passo anterior foram retirados *hashes* dos arquivos encontrados e eles podem ser comparados com bases de hashes conhecidos, como os fornecidos pelo *National Software Reference Library (NSRL)* [3] fornecidos pelo NIST. Comparando os hashes dos arquivos com essas bases é possível retirar todos os arquivos conhecidos que são seguros, como arquivos do sistema operacional e também arquivos conhecidamente maliciosos diminindo o escopo que o perito deve trabalhar.

[3] http://www.nsrl.nist.gov/Downloads.htm

No passo anterior foram retirados *hashes* dos arquivos encontrados e eles podem ser comparados com bases de hashes ou *hashset* conhecidos, como os fornecidos pelo *National Software Reference Library (NSRL)* [4] fornecidos pelo NIST. Comparando os hashes dos arquivos com *hashset* de arquivos conhecidos é possível retirar todos os arquivos conhecidos maliciosos também conhecidos como *known bad* ou arquivos inequivocamente bons (*known good*) como os do sistema operacional, diminindo o escopo que o perito deve trabalhar. O processo para se fazer esse processo é bem simples, uma vez com o *hashset* dos arquivos do disco investigado, o perito precisa apenas compara-los com os *hashsets* de arquivos conhecidos.

Para evitar ou atrapalhar esse processo o rootikit teria que ter o mesmo hash de um arquivo dito como bom. Seguindo os preceitos do Paradoxo do aniversário que mostra a probabilidade de duas pessoas numa sala terem o mesmo aniversário, (**??**) apresenta a formula: $1 - e^{\frac{-k(k-1)}{2N}}$, onde N é a quantidade de valores distintos que o hash pode apresentar e k é o tamanho do valor sequencial comparado, exemplo na sequencia: 0001_2, 0010_2, 0011_2 e 0100_2, o tamanho seria 0100_2. A tabela **??** mostra probabilidade de uma colisão de hash numa repetição uniforme e sequencial.

32-bit	**64-bit**	**160-bit**	**Probabilidades**
77163	5,06 bilhões	$1,42x10^{24}$	50%
30084	1,97 bilhão	$5,55x10^{23}$	1 em 10
9292	609 milhões	$1,71x10^{23}$	1 em 100 ou um full house no poker
10	607401	$1,71x10^{20}$	1 em 100 milhões ou ser mordido por um tubarão

Tabela 5.1: Probabilidade de uma colisão de hash em números sequenciais (**??**).

Perceba, que o número é sequencial, ou seja, com valores aleatórios, como os encontrados em discos isso é improvável. as chances de um valor arbitrário (como o rootkit) coincidir com o hash de um arquivo bom são improváveis, contudo não impossíveis, como pode ser visto no trabalho dos chineses Xiaoyun Wang e Hongbo Yu da Universidade de Shandong em 2005 (**??**), no trabalho eles desenvolvem um algoritmo que pode ser usado para criar arquivos arbitrários com o mesmo hash MD5, mesmo assim essa técnica é muito complexa, pois mesmo em MD5 tornar um rootkit igual a algum arquivo do bom é bem dificil e além disso não se tem noticias de nenhum algoritmo que faça com que o hash de um arquivo seja o mesmo que de outro em dois algoritmos diferentes, exemplo SHA1 e MD5.

Outra forma que o rootkit pode se valer para anular essa técnica, de forma menos sofisticada é alterar partes não significantes de arquivos do computador, como na mensagem mostrada do cabeçalho da *PE*, pois mudar qualquer uma das letras da mensagem altera também o hash do mesmo, contudo isso fugiria de uma regra básica dos rootkits que é ficar indetectável. No fim

[4] http://www.nsrl.nist.gov/Downloads.htm

das contas esse método é recomendado por sua grande utilidade e dificilmente um rootkit tentará ou consiguirá perverte-lo, entretanto vale as ressalvas apresentadas.

5.5 Análise estática dos executáveis suspeitos desconhecidos

Análise estática é a analise do código de determinado malware para se ter uma melhor compreensão sobre suas funções, características e objetivos (**??**). Depois de filtrar os arquivos conhecidos o perito deverá iniciar a análise estática de alguns arquivos suspeitos, certamente analisar todos os arquivos desconhecidos levaria muito tempo, inviabilizando na maioria dos casos a pericia. O ideal nesse momento é filtrar dentre os arquivos desconhecidos os que são potenciamente maliciosos, para isso o especialista deverá ficar atendo principalmente a binários. Arquivos suspeitos são encontrados em todas as fases apresentadas, principalmente na recuperação de arquivos e na coleta de metadados, essas fases podem ajudar e muito a levantas suspeitas de arquivos, por exemplo, porque um arquivo de imagem com assinatura de jpg tem 2GB?

A fase onde são retirados os arquivos conhecidos, pode revelar programas conhecidamente maliciosos, esses arquivos podem ser investigados nessa fase também, para responder a pergunta primordial que é: esse código malicioso poderia subverter o(s) sistema(s) analisado? A resposta vai depender muito do tipo de ambiente analisado, pois se o ambiente for muito seguro, dificilmente uma ameaça qualquer conseguiria persistir, além do fato de que as chances de um rootkit desenvolvido por um *APT* estar em qualquer base de arquivos conhecidos é mínima.

De qualquer forma apesar da resposta óbvia ser em sua grande maioria verdadeira, o perito não pode aceita-la resoluto se apegando a qualquer evidência solta ou mal compreendida, porque existe a chance, mesmo que mínima de que o artefato malicioso encontrado tenha sido plantado. Esse texto percore sobre rootkit e sempre tenderá ao pior dos cenários, contudo cautela e bom senso é vital.

A antiforense de processos de análise estática, consiste em impedir que o(s) código(s) do rootkit sejam analisados, para isso é comum que se use compreensão e criptografia nos arquivos do rootkit. Essa compreensão e/ou criptografia utilizada em qualquer código, sendo ele malicioso ou não, mexe com a estrutura do arquivo e o torna irreconhecível, de modo que não pode ser executado normalmente. Técnicas como essa são muito utilizadas por (**??**) para não serem detectados por antivirus, contudo sua aplicação também implica que sem uma estrutura padrão ele não pode ser interpretado pelo ambiente em que será executado e como já foi mostrado o rootkit em algum momento deverá ter uma estrutura padrão para que o mesmo possa ser

executado. Para resolver esse problema existe o *stub*.

Um *packer* é um programa que comprime um executável dentro de uma estrutura anormal e menor, de forma análoga a um programa de compactação de arquivos para o vulgamente chamado *zip* só que com executáveis. Com (**??**) e (**??**) é possível entender como funciona um packer, mas especificamente o *Ultimate Packer for eXecutables (UPX)* [5] é um *packer* que suporta diversos tipos de executáveis de diversos ambientes como mostra a tabela **??**.

Nome completo	**Descrição**
amd64-linux.elf	Linux ELF
amd64-linux.kernel.vmlinux	Linux kernel
arm-wince.pe	Windows CE executable or DLL
fat-darwin.macho	Mac OS X executable
i086-dos16.com	DOS 16-bit .com file
i086-dos16.exe	DOS 16-bit executable
i086-dos16.sys	DOS 16-bit .sys file
i386-dos32.djgpp2.coff	DOS 32-bit COFF
i386-dos32.tmt.adam	DOS 32-bit executable
i386-dos32.watcom.le	DOS 32-bit linear executable
i386-win32.pe	Windows 32-bit executable or DLL

Tabela 5.2: Amostra de executáveis suportados pelo UPX (**??**).

Ao lidar com executáveis Windows principalmente *PE*, será encontrado uma estrutura com seções como .text, .data, .idata e .fill, como mostra a figura **??**.

figuras/pe-sections.png

Figura 5.4: Hello World - PIMAGE SECTION HEADER (**??**)

Depois de compactado o *UPX*, deixa as seções do executável como o da figura **??**. O *UPX* combina todas as seções .text, .data, .idata etc em uma única seções chamada upx1, onde tem uma *stub* que descompacta o binário original.

O que o *UPX* faz é colocar o binário numa forma comprimida que o deixa com uma estrutura incomum ao ambiente de execução, logo as ferramentas de engenharia reversa como *disassemblers* não conseguem interpretar e consequentimente o sistema operacional também não, contudo o *UPX* também adiciona um executável capaz de descomprimir e executar o executável original em tempo de execução chamado de *stub*. O *stub* descompacta o *payload* que

[5] http://upx.sourceforge.net/

nesse caso é o executável original e muda o controle do executável para o *entry point* do binário original.

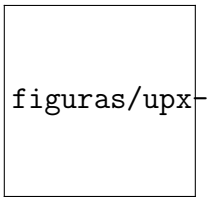

Figura 5.5: Packed Hello World - PIMAGE SECTION HEADER (**??**)

A seção upx0 é basicamente espaço vazio que ocuparão na memória, o upx1 é onde fica o *stub* e o executável original compactado. O *stub* irá descompactar o executável original o inserindo no espaço vazio utilizado pela seção upx0 que ficam na *low memory* e executado-o, dessa forma o *stub* que fica logo acima na parte da memória também é sobrescrito durante o processo. Como pode ser percebido pela pecularidade o executável original não pode ser análisado enquanto estiver compactado.

Outra forma de impossibilitar a análise estática, é compilar o rootkit em algum *bytecode*, que são interpretados por alguma *virtual machine* de forma análoga com código feito em Java e/ou .Net, ou seja, o código escrito em geral em uma linguagem de alto nível é compilado para algum tipo de *bytecode*, conhecidos como o *Java bytecode* ou o *.Net bytecode*, ou até mesmo algum *bytecode* privado. Em qualquer uma dessas possibilidades, o ambiente que o mesmo ira ser executado deverá ter a *virtual machine*, porque será ela que irá interpretar o *bytecode* e traduzir para instruções que o processador compreende. As complicações para o uso dessa abordagem envolve que necessariamente o criador do rootkit terá que utilizar uma *virtual machine* para interpretar o *bytecode* em tempo de execução, caso seja feito em algum *bytecode* conhecido, o rootkit vai ser facilmente decompilado, mas se usar um *bytecode* privado o mesmo irá ter que criar a *virtual machine* privada também o que demanda muito tempo, mas em compensação dificulta muito o trabalho do perito, que antes de analisar o código terá que entender e decompilar a *virtual machine*. Da mesma forma que o código compactado, o código em *bytecode* não possui instruções conhecidas, logo ferramentas ou até mesmo o processador não o interpreta.

Esses foram dois metódos que podem ser utilizados para evitar ou complicar o processo de análise estática. Técnicas como essa podem ser usadas por programas legitimos também como é o caso do skype (**??**). O importante é entender que não importa o que o rootkit pode usar, mas ele sempre vai ter que ter uma estrutura padrão para sua execução e que quanto mais baixo o nível de execução menos o rootkit pode fazer para se esconder, menos ele precisa efetivamente para se esconder e mais controle ele consegue do sobre o sistema infectado, por exemplo, se

um rootkit roda desde a inicialização da *BIOS*, quando o sistema operacional for executado o rootkit já está sendo executado.

5.6 Análise dinâmica dos executáveis suspeitos

Essas técnicas podem ser combinadas dificultando ainda mais a análise, contudo o stub tem a chave para decodificar o binário, mas isso pode ser mais dificil quando *crypters* e *packer* privado, pode dividir o código em diversos setores e cada setor ter uma chave criptogradada diferente SALT + HWID + TIME + PCI_ROM para criar a chave

antidebugger FS flag determina o debug na arquitetura IA32, explicar que assim e processador faz uma instrução por vez

Windows SDK duas rotinas, WINAPI isDEBUggerPresent(void) user-mode ou KdRefreshDebuggerNotPresent() Kernel-mode (being debugged PEB http://www.symantec.com/connect/articles/wind anti-debug-reference

antivirtualização, para testar qualquer código malicioso deve-se

http://www.chmag.in/article/sep2011/rootkits-are-back-boot-infection

5.7 outro

nao existe tecnica antiforense 100% nem metodologia para analise infalivel, nem 100% segura e confiavel