Marcos Aurélio Domingues

Comparação de Ferramentas de Verificação de Integridade de Arquivos

Monografia apresentada ao Curso de Administração em Redes Linux (ARL) da Universidade Federal de Lavras como parte das exigências da disciplina Monografia para obtenção do título de Especialista em Administração em Redes Linux.

Orientador
Prof. Msc. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2003

Marcos Aurélio Domingues

Comparação de Ferramentas de Verificação de Integridade de Arquivos

Monografia apresentada ao Curso de Administração em Redes Linux (ARL) da Universidade Federal de Lavras como parte das exigências da disciplina Monografia para obtenção do título de Especialista em Administração em Redes Linux.

Aprovada em *16 de Novembro de 2003*

Prof. Msc. Bruno de Oliveira Schneider

Prof. Msc. Fernando Cortez Sica

Prof. Msc. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil

Sumário

Lista de Figuras

Lista de Tabelas

À minha família.
Com carinho especial, a meus
pais Isabete e José Pedro e a meus
irmãos Marcelo e Márcio.

Agradecimentos

Agradeço a Deus por ter me ajudado a vencer mais uma etapa da minha vida.
Agradeço ao Prof. MSc. Joaquim Quinteiro Uchôa, pela dedicação e incentivo durante a orientação deste trabalho.
Agradeço aos meus familiares, pelo apoio e incentivo.

Resumo

Uma das primeiras ações de um invasor costuma ser a substituição de arquivos e programas do sistema com o intuito de mascarar sua visita atual e, principalmente, facilitar as visitas futuras. Portanto, se houver a possibilidade de verificar a integridade de arquivos do sistema, haverá uma grande possibilidade de detectar uma invasão. E o melhor é que esse recurso permite que se saiba quais arquivos foram modificados, possibilitando que o administrador decida entre reinstalar o sistema ou apenas substituir os arquivos alterados. Neste trabalho tem-se como objetivo analisar e comparar as ferramentas de controle de integridade de arquivos *AIDE*, *Integrit*, *Osiris* e *Tripwire*.

Capítulo 1

Introdução

É indiscutível a importância da segurança da informação no cenário atual. O aumento no número de aplicações, a distribuição dessas aplicações através do uso maciço de redes de computadores e o número crescente de ataques a essas aplicações retrata a preocupação e justifica os esforços em trabalhos voltados a essa área. Novos mecanismos, técnicas mais eficientes e normas internacionais para a gestão da segurança da informação são constantemente desenvolvidos, incentivados por organismos governamentais e empresas preocupadas com o atual estágio de fragilidade da maioria dos sistemas computacionais.

Segundo [Russel & Gangemi (1991)], a segurança de sistemas computacionais compreende os quatro seguintes atributos:

1. **Autenticidade:** entidades, como usuários e processos, devem ter sua identidade devidamente certificada a fim de possibilitar o emprego de controles de acesso eficientes aos recursos de um sistema computacional, bem como permitir a realização de auditorias;

2. **Confidencialidade:** um sistema computacional seguro deve evitar que qualquer informação seja revelada para entidades que não possuam autorização para acessá-la;

3. **Integridade:** o sistema deve impedir que as informações nele contidas sofram modificações não autorizadas, sejam estas acidentais ou intencionais;

4. **Disponibilidade:** o sistema deve manter as informações disponíveis para os seus usuários legítimos.

A disponibilização de novas técnicas em segurança computacional que englobam os atributos apresentados, no entanto, não tem representado um aumento equivalente no nível de segurança das organizações, retrato, principalmente, da falta de uma abordagem correta na seleção e implementação dessas técnicas. A

maioria dos profissionais responsáveis pela gerência de segurança, oriundos principalmente da área de computação, possui um formação técnica que os levam a selecionar mecanismos sem considerar as reais necessidades da organização. Essa prática causa distorções que vão desde o gasto desnecessário com mecanismos não apropriados para o problema, enfrentado até a falta de proteção às informações realmente importantes.

Diante desse contexto, esta monografia tem por objetivo estudar e comparar ferramentas de verificação de integridade de arquivos. Essas ferramentas são importantes pois permitem manter a integridade das informações armazenadas nos sistemas de uma organização.

A monografia está organizada como apresentado a seguir: no Capítulo 2 são apresentadas as principais atividades relacionadas ao controle de integridade de arquivos. No Capítulo 3 são descritas as ferramentas de verificação de integridade de arquivos *AIDE, Integrit, Osiris* e *Tripwire*, que são objetos de estudo deste trabalho. É apresentada uma descrição das ferramentas, instalação, configuração e uso das mesmas. Já no Capítulo 4 são apresentados os resultados da comparação das ferramentas descritas no Capítulo 3. No Capítulo 5 são apresentadas as conclusões sobre este trabalho. Por fim são apresentadas as Referências Bibliográficas utilizadas.

Capítulo 2

Integridade de Arquivos

2.1 Considerações Iniciais

Dentre os subsistemas contidos em um sistema operacional, pode-se mencionar que o sistema de arquivos ou *file system* (FS) é um dos mais importante. Nele são mantidos arquivos de usuários, arquivos de configuração dos mais diversos serviços e utilitários, arquivos executáveis, além de conter o sistema operacional em si.

A partir da modificação de alguns dos arquivos mantidos pelo sistema é possível alterar, de forma significativa, o seu comportamento podendo até provocar o comprometimento de toda a sua segurança de funcionamento. Dentre alguns resultados da alteração de arquivos pode-se ter: subversão de mecanismos de autenticação, interrupção ou deturpação de serviços, subversão de mecanismos de controle de acesso, paralisação total do sistema, entre outros. Devido a esses fatos, os FSs são um dos principais alvos de ataque de invasores de sistemas.

Nesse contexto, o controle de integridade de arquivos passa a ser um recurso de extrema importância já que, sem ele, a inserção de código malicioso no sistema e as alterações não autorizadas de arquivos dificilmente seriam detectadas. O objetivo deste capítulo, baseado nos estudos de [Serafim (2002)], é apresentar as principais atividades relacionadas a manutenção da integridade de arquivos.

2.2 Controle de Integridade de Arquivos

O processo chave para o controle de integridade de arquivos é a reconciliação de objetos, ou simplesmente comparação de objetos. Para tanto, é necessário que o objeto, cuja integridade deverá ser futuramente verificada, seja copiado (duplicado). Em um momento seguinte, quando é necessária a verificação da integridade do objeto em questão, compara-se a versão original desse com a cópia armazenada. Cabe aqui colocar que a cópia dos objetos para fins de controle de integridade é

comumente chamada de instantâneo, ou *snapshot*.

Esse processo pode ser facilmente aplicado aos arquivos de um FS, para isso, basta a realização de uma cópia dos arquivos, e posterior comparação dos originais com a cópia previamente realizada. Assim, do processo de comparação pode-se obter os seguintes resultados:

Nenhuma alteração detectada: confirma que no momento em que a verificação foi realizada, os arquivos monitorados encontravam-se íntegros e autênticos;

Alterações não autorizadas detectadas: denunciam a presença de arquivos indevidamente alterados, esses arquivos podem ser facilmente substituídos a partir do *snapshot* ou das mídias originais;

Alterações legítimas detectadas: indicam a necessidade de se atualizar o *snapshot* para refletir um novo estado íntegro dos arquivos.

Caso fosse realizada a aplicação desse processo tal com descrito, certamente haveria problemas de espaço de armazenamento no sistema, visto que ocorreria a duplicação do espaço ocupado por cada arquivo monitorado. Além disso, o processo de comparação de dois arquivos *byte* a *byte* ocasionaria um desperdício de tempo considerável.

A forma atualmente mais apropriada para a solução desses problemas é a utilização de funções matemáticas que, quando aplicadas ao conteúdo de um arquivo, geram um resumo matemático desse arquivo. Esse resumo matemático geralmente possui um tamanho pequeno e fixo de bits, que depende da função utilizada e normalmente varia entre 2 e 32 *bytes*. Tem-se assim, uma forma de identificação de diversos conjuntos de *bytes* (arquivos), diferentes tanto em conteúdo quanto em tamanho, através da sua representação por uma seqüência singular e de tamanho fixo de bits. A essa seqüência dá-se o nome de digital, ou *fingerprint*.

Através do uso dessas funções matemáticas, o *snapshot* não precisa ser constituído de uma cópia fiel de cada arquivo, cujo tamanho é variável, mas apenas das digitais de cada um deles, cujo tamanho é sempre constante e pequeno (cerca de 2 a 32 *bytes*), proporcionando uma economia substancial de espaço de armazenamento do *snapshot*. Apesar desse ganho de espaço, a maior vantagem é a velocidade nos processos de geração do *snapshot* e da verificação da integridade dos arquivos, o que torna possível a criação de mecanismos de controle de integridade eficazes e realmente aplicáveis, conforme será visto ao longo deste trabalho. Entre as principais funções matemáticas para a geração do *snapshot* podemos citar as funções de *hash*, apresentadas na seção a seguir.

2.3 Funções de *Hash* Unidirecionais

As funções de *hash* unidirecionais, também conhecidas como funções criptográficas de *hash* ou simplesmente funções de *hash*, desempenham um importante papel

na criptografia moderna, fazendo parte dos mais diversos protocolos criptográficos existentes. Essas funções mapeiam uma massa de dados de tamanho variável para uma seqüência de tamanho fixo de bits.

O resultado do uso de uma função de *hash*, chamado de digital (*fingerprint*), *message digest* (MD) ou simplesmente *hash*, pode ser comparado a um *checksum*, ou seja, ambos permitem a detecção de alterações dos dados utilizados para gerá-los, mas não fornecem qualquer tipo de recurso para a correção das mesmas [Schneier (1996)].

De modo mais abrangente, uma função de *hash* deve ter no mínimo as seguintes características [Menezes (1996), Schneier (1996)]:

- **compressão:** uma entrada x de tamanho arbitrário é mapeada para uma saída $h(x)$ de tamanho fixo;
- **facilidade de cálculo:** dada uma função h e uma entrada x, deve ser fácil calcular $h(x)$;
- **unidirecionalidade:** dado um *hash* de uma entrada não conhecida, $h(x)$, tal que seja muito difícil obter a entrada x em parte ou na íntegra, essa dificuldade deve manter-se mesmo que parte de x seja conhecida;
- **resistência a colisões fracas:** dado uma determinada entrada x, deve ser muito difícil encontrar outra entrada x', tal que $h(x') = h(x)$;
- **resistência a colisões fortes:** deve ser muito difícil encontrar duas entradas quaisquer, x e x', tal que $h(x') = h(x)$; a principal diferença dessa em relação as duas anteriores é que o invasor tem liberdade total para escolher tanto x quanto x';
- **resistência a colisões próximas:** deve ser difícil encontrar duas entradas x e x' tal que $h(x)$ e $h(x')$ difiram em apenas um número pequeno de bits;
- **inexistência de correlação:** os bits da entrada x e da saída $h(x)$ não devem ser correlacionados, a idéia aqui é que cada bit da entrada afete todos os bits da saída, isto é conhecido como efeito avalanche.

Entre as funções de *hash* mais usuais podemos citar: SHA-1 [Nist (1995)], Tiger [Anderson & Biham (1996)], RIPMED [Dobbertin et al (1996)], HAVAL [Zheng et al (1992)] e MD5 [Touch (1995)].

Independente da forma como o *snapshot* é gerado, o processo de controle de integridade de arquivos possui uma série de pontos sensíveis e potencialmente vulneráveis sob o ponto de vista da segurança. O tratamento inadequado desses pontos pode proporcionar a um invasor a chance de intervir no processo, invalidando completamente todo o controle de integridade desejado.

Dessa forma, é de extrema importância que esses pontos sejam destacados e devidamente analisados. Esses pontos são apresentados nas seções seguintes.

2.4 Estado Inicial dos Arquivos

A geração do *snapshot* é crucial para garantir o sucesso do controle de integridade pois daí resultará o parâmetro de comparação para a identificação de arquivos violados. Caso o *snapshot* seja gerado a partir de um sistema já comprometido, todo o processo é invalidado já que as possíveis alterações feitas pelo invasor não poderão ser detectadas.

O momento mais adequado para a geração do *snapshot* de um sistema é logo após sua instalação, pois se realizada em ambiente adequado, nenhum invasor terá a possibilidade de comprometer qualquer parte do sistema antes ou durante a geração do *snapshot*. Como ambiente adequado entende-se que:

- há controle do acesso físico ao equipamento onde a instalação está sendo realizada;
- não há nenhuma possibilidade de acesso remoto ao sistema, seja por usuários autorizados ou não;
- todos os programas a serem instalados têm sua origem e integridade verificadas;
- após a instalação, o sistema é varrido por programas de detecção de códigos maliciosos, tais como um antivírus.

Nos casos, muitas vezes freqüentes, em que o sistema já está em funcionamento, a dificuldade de obter-se um alto nível de confiança na integridade dele aumenta consideravelmente, pois deve-se então levar em conta os níveis de segurança e de monitoramento implementados no sistema desde a sua instalação, além do histórico de incidentes ocorridos e a forma como foram tratados. Tais procedimentos de avaliação não estão compreendidos no escopo do presente trabalho e por isso não serão discutidos com maior profundidade, sem que isso diminua sua importância no processo.

Levando-se a efeito os procedimentos aqui comentados, tem-se um nível razoável de confiança na integridade do sistema, diminuindo-se consideravelmente os riscos de geração de um *snapshot* já comprometido.

2.5 Armazenamento do *Snapshot*

O *snapshot* não é somente vulnerável no momento da primeira geração, mas também durante o período em que é armazenado. Armazenar o *snapshot* de forma adequada é um procedimento básico para possibilitar o controle de integridade de arquivos.

Em caso de perda do *snapshot*, sua disponibilidade é comprometida tornando impossível a verificação da integridade dos arquivos monitorados. Ainda pior do

que o comprometimento da disponibilidade do *snapshot*, é a alteração desse realizada por um invasor, a fim de refletir as modificações por ele realizadas nos arquivos do sistema, evitando assim qualquer detecção por parte do mecanismo de controle de integridade. É preferível a perda ou invalidação total do *snapshot*, pois, no segundo caso, a realização de uma verificação de integridade irá assegurar ao administrador que o sistema encontra-se íntegro, quando na realidade não está.

Sendo assim, o armazenamento adequado do *snapshot*, visando garantir a sua disponibilidade e integridade, é um fator crítico em qualquer método de controle de integridade de arquivos, tornando necessário o emprego de mecanismos para garantir a proteção do *snapshot* armazenado. Alguns mecanismos que podem ser empregados para a proteção do *snapshot* são apresentados nas subseções a seguir.

2.5.1 CD-*Recordable*

A gravação do *snapshot* em uma mídia como o CD não regravável garante a sua validade, dificultando consideravelmente a ação de qualquer agente malicioso. Ataques que contam apenas com o acesso remoto ao sistema, mesmo com os privilégios do administrador, são impossíveis de serem realizados já que a própria mídia tem restrições físicas inalteráveis que impedem qualquer modificação do seu conteúdo.

Mesmo com acesso físico à mídia, os ataques possíveis são limitados. O ataque mais fácil de ser realizado é a destruição da mídia – comprometimento de sua disponibilidade – impedindo a realização da verificação de integridade caso não haja uma outra mídia reserva. Outro ataque é a substituição da mídia original por uma outra já modificada, para isso não só é preciso o acesso físico mas também lógico ao sistema. Esse ataque pode ser facilmente impedido com o uso do CD não regravável em combinação com um mecanismo lógico, como as assinaturas digitais, que serão apresentadas na subseção 2.5.4.

Em resumo, esse mecanismo físico fornece um nível bastante alto de proteção ao *snapshot*, praticamente evitando que sejam utilizados *snapshots* inválidos em processos de reconciliação de objetos.

2.5.2 CD-RW em unidades de CD-ROM

Esse mecanismo é apenas uma variação do anterior. Nesse, o *snapshot* é gravado em um CD regravável, o que possibilita a sua atualização, mas quando em uso é mantido somente em unidades de CD-ROM, pois essas não possuem condições físicas para alterar a mídia nelas inserida. Essa é a proteção fornecida.

Os ataques possíveis para esse mecanismo são os mesmos do mecanismo anterior. Como vantagens tem-se a economia de mídias e a redução de tempo, já que não é necessário gerar novamente todo o *snapshot* para realizar qualquer atualização. Ainda, assim como no mecanismo anterior, é recomendado o uso em conjunto

com um mecanismo lógico.

Apesar de simples, tanto esse quanto o mecanismo da seção anterior, impõem sérias restrições às possíveis investidas de um invasor contra o *snapshot* armazenado, resultando na obtenção de um nível de segurança relativamente alto.

2.5.3 Montagem do FS em modo *read-only*

O *snapshot* pode ser mantido em um FS que é montado em modo *read-only*, evitando-se assim que qualquer modificação seja realizada no FS em questão. A idéia aqui empregada é bastante semelhante à de manter um CD-RW em uma unidade de CD-ROM. A diferença crucial é que esse mecanismo é lógico e não físico, resultando daí o seu ponto fraco: caso o atacante obtenha acesso (remoto ou não) como administrador do sistema, ele pode desmontar o FS e montá-lo novamente de forma que modificações possam ser realizadas, ou seja, a proteção só funciona contra usuários comuns do sistema.

Mesmo sofrendo da deficiência acima descrita, quando utilizado em conjunto com outros mecanismos lógicos, essa abordagem pode constituir um barreira significativa para as ações de um invasor.

2.5.4 Assinaturas Digitais

O uso de assinaturas digitais pode garantir a confidencialidade e a autenticidade do *snapshot*. As assinaturas digitais são realizadas com o uso de algoritmos de chaves assimétricas.

O fato das chaves serem assimétricas significa, em outras palavras, que chaves diferentes são utilizadas no processo de cifragem e decifragem. O par de chaves é constituído de uma chave privada e outra pública, e a relação entre elas é a seguinte: tudo o que for cifrado com a chave pública somente pode ser decifrado com o uso da chave privada correspondente; e tudo que for cifrado com a chave privada só pode ser decifrado com a chave pública correspondente [Menezes (1996)].

Sendo assim, o administrador do sistema teria o seu par de chaves e com sua chave privada cifra (assina) o *snapshot*, ou apenas o seu resumo matemático [Menezes (1996)]. Dessa forma, somente a chave pública precisa estar armazenada no sistema. Nesse caso, mesmo que o atacante obtenha acesso à ela, ele não é capaz de modificar o *snapshot*. Aqui, somente a autenticidade do *snapshot* é garantida.

Opcionalmente, o próprio sistema poderia ter um par de chaves, o que possibilitaria autenticidade e também confidencialidade. O administrador primeiro assina o *snapshot* e, em seguida, cifra-o com a chave pública do sistema.

O ataque mais prático aqui é a substituição de chaves, ou seja, o atacante gera um novo par de chaves, substitui a chave pública do administrador armazenada no sistema pela sua e, em seguida altera o *snapshot* e assina-o. Dependendo da forma

como a chave pública e o *snapshot* são armazenados, esse ataque pode se tornar muito difícil ou praticamente impossível de ser realizado principalmente quando usado em conjunto com um mecanismo físico de proteção, tal como o CD-R.

2.6 Atualização do *Snapshot*

Ao longo do uso de um sistema, diversas tarefas administrativas acabam por causar a execução de alterações intencionais e não maliciosas, portanto válidas, no sistema de arquivos. Entre essas tarefas tem-se:

- aplicação de correções em programas já existentes (*patches*);
- atualização de programas já existentes (*updates*);
- instalação de novos programas;
- customização de configurações executadas pelo próprio administrador.

Independentemente do tipo de tarefa que deu origem às modificações válidas no sistema de arquivos, o *snapshot* deve ser atualizado a fim de refletir esse novo estado íntegro do sistema. Esse processo de atualização é tão sensível quanto o processo de criação do *snapshot* inicial do sistema, apresentado na seção 2.4. A falta de conhecimento e de controle do que realmente foi alterado ou adicionado ao sistema de arquivos, pode levar à consolidação de modificações inválidas, potencialmente intencionais e maliciosas no *snapshot*.

Nenhum mecanismo de proteção do *snapshot* é capaz de evitar a ocorrência desse tipo de incidente, já que esse é resultante da imperícia do próprio administrador. Felizmente é possível reduzir consideravelmente o risco de comprometimento do *snapshot*, através da utilização do próprio mecanismo de controle de integridade de arquivos e do emprego de procedimentos adequados por parte do administrador. Abaixo são citados e comentados alguns passos genéricos que podem ser adotados como procedimentos para a referida atualização:

Verificação da integridade dos arquivos do sistema: esse passo deve ocorrer antes da alteração dos arquivos, a fim de obter com certeza o estado íntegro atual dos arquivos do sistema; quaisquer violações detectadas devem ser imediatamente investigadas e solucionadas pelo administrador;

Execução das alterações: executando o primeiro passo, pode-se então realizar as alterações necessárias;

Nova verificação da integridade dos arquivos do sistema: essa segunda verificação tem por objetivo levantar todas as modificações realizadas nos arquivos do sistema; em seguida, todas as modificações identificadas devem ser

investigadas e relacionadas às tarefas executadas no segundo passo; modificações não relacionadas não devem ser mantidas no sistema e devem ser tratadas como violações potencialmente intencionais e maliciosas, sendo imediatamente corrigidas;

Atualização do *snapshot*: finalmente o *snapshot* pode ser atualizado com segurança.

Ainda, visando um maior nível de segurança, o procedimento aqui descrito deve ser realizado em um ambiente similar ao apresentado na seção 2.4.

Como foi visto, apesar de não ser possível resolver o problema descrito apenas com o uso de mecanismos de controle de integridade, a adoção dos procedimentos apropriados por parte do administrador pode reduzir substancialmente as chances de comprometimento do *snapshot*.

2.7 Restauração dos Arquivos Violados

A detecção de uma violação no sistema de arquivos, o que pode sinalizar a execução de ações maliciosas por exemplo, não basta para fornecer segurança para um sistema. Necessita-se ainda de algum meio para corrigir a violação detectada, fazendo com que o sistema volte ao seu estado íntegro original.

Infelizmente, mecanismos de controle de integridade baseados em resumos matemáticos não são capazes de corrigir violações, uma vez que não é possível obter-se o arquivo original apenas partindo de seu resumo. Sendo assim, o uso de um mecanismo de controle de integridade deve ser combinado com uma prática já bem documentada e universalmente recomendada: a realização de cópias de segurança, ou *backups*.

As cópias de segurança são indicadas para arquivos customizados e particulares de cada sistema já que arquivos de programas e também o sistema operacional podem ser facilmente obtidos a partir da mídia original de instalação. A inexistência de cópias de segurança ou das mídias originais pode levar o sistema a um estado sabidamente inválido e sem a possibilidade de correção das violações detectadas. Nesses casos, que devem ser considerados inadmissíveis tanto do ponto de vista da segurança quanto administrativo, pode-se ter que recorrer à paralisação e à reinstalação de partes ou de todo o sistema envolvido.

Muito embora a cópia de segurança de arquivos esteja, de certa forma, intimamente relacionada ao controle de integridade, não faz parte do objetivo deste trabalho o detalhamento das técnicas e procedimentos envolvidos na sua execução já que é extremamente farta e difundida a documentação existente sobre o assunto.

2.8 Identificação de Ações Maliciosas

Os mecanismos de controle de integridade somente detectam violações nos arquivos cuja integridade é por eles monitorada, ou seja, ele não indica quando uma violação foi intencional ou não e muito menos se foi maliciosa.

Para tais classificações, deve ser realizada uma investigação da violação detectada, utilizando-se para isso ferramentas específicas e o próprio conhecimento do administrador do sistema. Entre as diversas ferramentas disponíveis e conhecidas que podem ser empregadas tem-se: programas que varrem o sistema a procura de códigos maliciosos com vírus, *backdoors* e *rootkits*, o próprio registro das operações do sistema (arquivos de *log*) e as ferramentas utilizadas para sua análise.

Mesmo com o emprego dessas e de outras ferramentas, o conhecimento e experiência do administrador são muitas vezes determinantes para o sucesso de uma investigação de violações ocorridas em arquivos do sistema.

2.9 Integridade do Mecanismo de Controle

Exatamente por ser um dos responsáveis pela segurança do sistema, o mecanismo de controle de integridade pode virar alvo da ação de invasores pois, caso ele consiga comprometer o funcionamento do mecanismo, uma importante barreira para o sucesso de seus propósitos será eliminada. E, ainda, dependendo de como o mecanismo é comprometido, ele pode ser convertido em ferramenta de apoio ao invasor fazendo com que o administrador e usuários tenham uma falsa sensação de segurança ao pensar que o mecanismo está funcionando corretamente e não investiguem qualquer comportamento estranho do sistema.

Chega-se então ao ponto em que a integridade do próprio mecanismo de controle deve ser regularmente verificada, tanto por processo automatizado quanto por processo manual. Esse procedimento é necessário pois o comprometimento do mecanismo pode indicar que uma grave vulnerabilidade foi previamente explorada pelo atacante, expondo o sistema ao risco de perda total ou parcial dos arquivos nele contidos, bem como de vazamento e deturpação de informações.

2.10 Intervalo entre Verificações

O intervalo de tempo entre duas verificações de integridade é o tempo que um invasor tem para agir sem que seja detectado, podendo assim alterar o conteúdo dos arquivos conforme seus propósitos. Como conseqüência, a relação entre intervalo de tempo e segurança é inversamente proporcional, ou seja, quanto maior esse intervalo, menor será a segurança fornecida e vice-versa.

Dessa forma, quando o objetivo é atingir um nível alto de segurança, tende-se a reduzir ao mínimo possível o intervalo de tempo. Infelizmente essa prática

coloca em risco o desempenho do sistema como um todo, podendo prejudicar o fornecimento dos serviços que estão sendo oferecidos aos usuários. Desempenho é, portanto, um importante fator a ser levado em conta na modelagem e implementação de qualquer mecanismo desse tipo.

À medida que o intervalo de verificação aumenta, a perda de desempenho torna-se mais aceitável, ou seja, com intervalos grandes de tempo entre duas verificações, qualquer perda de desempenho será momentânea e não deverá causar prejuízo às operações de rotina do sistema. No entanto, perde-se em relação à segurança.

A perda de desempenho e a segurança fornecida não são somente determinadas pelo intervalo de tempo utilizado entre duas verificações, mas também pela forma como o mecanismo é implementado, os momentos em que o mesmo realiza o controle de integridade e a quantidade de arquivos monitorados.

2.11 Momentos para a Verificação

Os momentos escolhidos para a realização da verificação da integridade de arquivos influenciam diretamente o nível de segurança fornecido pelo mecanismo empregado e o mínimo intervalo de tempo possível entre duas verificações. De modo geral, pode-se dizer que existem dois momentos para a realização da verificação de integridade: durante o isolamento e durante a sua operação normal.

O primeiro momento é encontrado em sistemas que pode ter suas atividades temporariamente suspensas em períodos regulares de tempo. Nesses casos a principal vantagem é que praticamente qualquer perda de desempenho é aceitável, pois não há o risco de prejuízo para as operações de rotina do sistema, possibilitando a monitoração de uma grande quantidade de arquivos. Ainda, evita-se nesses momentos até mesmo uma possível monitoração realizada por um invasor, caso ele já tenha obtido algum nível de acesso ao sistema, diminuindo suas chances de interferir no processo de verificação de integridade.

Embora pareça ser um momento ideal para a verificação, principalmente considerando a despreocupação com o desempenho, normalmente o isolamento do sistema não ocorre muito freqüentemente, prejudicando a periodicidade da verificação, ou seja, o intervalo de tempo entre verificações tende a ser longo, fazendo com que as possíveis violações demorem a ser detectadas, comprometendo o nível de segurança fornecido.

Intervalos de tempo menores são obtidos de forma mais fácil quando a verificação é realizada durante a operação normal do sistema, uma vez que esse último não precisa ter suas atividades paralisadas. Porém, possíveis perdas de desempenho são aqui fator limitante, pois quanto menor o intervalo, maiores serão os gastos de recursos, principalmente tempo de processador. A princípio, fica limitado também o número de arquivos a serem monitorados, já que quanto maior for

esse número, maiores serão os custos de uma verificação.

O modo como a verificação é executada durante a operação normal do sistema influencia fortemente o impacto resultante no desempenho desse último. Existem basicamente dois modos de execução:

Verificação disparada em um dado instante: a verificação é executada em um dado instante escolhido pelo administrador, podendo ser disparada de forma interativa ou automatizada; nesse modo, a integridade de todos os arquivos é verificada de uma só vez, concentrando a perda de desempenho em um curto espaço de tempo e tornando-a sensível para os usuários do sistema;

Verificação sob demanda: a verificação é realizada cada vez que for requisitado acesso (leitura, escrita ou execução) a um arquivo monitorado; a perda de desempenho é aqui diluída ao longo do tempo já que somente os arquivos utilizados são verificados, o que não acontece a todo momento; além dessa vantagem, esse tipo de mecanismo evita que um arquivo violado seja utilizado sem que ocorra a detecção e ainda é possível que ele bloqueie o acesso ao arquivo violado, evitando o seu uso no sistema.

Em determinadas situações, a verificação sob demanda apresenta vantagens com relação a verificação disparada em um dado instante, pois além de reduzir o impacto no desempenho do sistema, fornece um maior nível de segurança para o mesmo.

2.12 Considerações Finais

Neste capítulo foram apresentados alguns conceitos relacionados às funções de *hash* utilizadas na geração do *snapshot* dos arquivos de um sistema. Também foram descritas as principais atividades relacionadas ao controle de integridade de arquivos. No próximo capítulo são apresentadas algumas ferramentas utilizadas para controlar a integridade de arquivos.

Capítulo 3

Ferramentas para Controle de Integridade de Arquivos

3.1 Considerações Iniciais

Vários projetos têm sido desenvolvidos buscando suprir a falta de mecanismos de controle de integridade de arquivos nos sistemas operacionais. Infelizmente, com raras exceções, esses projetos são muito pouco documentados, ficando limitados à sua implementação e simples manuais de uso. Com uma breve pesquisa realizada na **WWW** e com informações obtidas em [Anonymous (1999)] foi possível a obtenção da seguinte lista de ferramentas de controle de integridade de arquivos:

- *Tripwire* (http://www.tripwire.org/);
- *Aide* (http://www.cs.tut.fi/ rammer/aide.html/);
- *L6* (http://www.pgci.ca/common/p_l6.htm/);
- *Claymore* (http://linux.rice.edu/magic/claymore/);
- *Integrit* (http://integrit.sourceforge.net/);
- *Rocksoft Veracity* (http://www.rocksoft.com/veracity/);
- *Tara* (http://www-arc.com/tara/);
- *Tiger* (http://www.gnu.org/directory/security/system/tiger.html/);
- *Osiris* (http://osiris.shmoo.com/).

Neste capítulo são descritas as ferramentas *AIDE*, *Integrit*, *Osiris* e *Tripwire*, que são o foco de estudo deste trabalho. Será apresentada uma descrição das ferramentas, bem como uma breve explicação sobre a instalação, a configuração e o uso das mesmas.

3.2 A Ferramenta *AIDE*

O *AIDE* (*Advanced Intrusion Detection Environment*) é um *software* que tem a finalidade de verificar a integridade dos arquivos de um sistema [Lehti (2003)]. Esse *software* constrói uma base de dados com várias informações dos arquivos especificados em seu arquivo de configuração. Essa base de dados pode conter várias informações dos arquivos como: permissões, número do *inode*, dono, grupo, tamanho, data e hora de criação, último acesso e última modificação.

Além disso, o *AIDE* também pode gerar e armazenar nessa base de dados o *checksum* criptográfico dos arquivos, utilizando um, ou uma combinação dos seguintes algoritmos: MD5 [Touch (1995)], Tiger [Anderson & Biham (1996)], RIPMED-160 [Dobbertin et al (1996)] e SHA-1 [Nist (1995)].

O procedimento recomendado é que o usuário crie essa base de dados em um sistema recém-instalado antes de conectá-lo a uma rede. Essa base de dados será a fotografia do sistema em seu estado normal e o parâmetro a ser utilizado para medir alterações no sistema de arquivos. Obviamente, sempre que o usuário modificar o seu sistema, como por exemplo através da instalação, atualização ou remoção de programas, uma nova base de dados deverá ser gerada. Essa nova base de dados é que deverá ser utilizada como parâmetro para medir alterações nos arquivos. A base de dados deve conter informações sobre binários, bibliotecas e arquivos de cabeçalhos importantes do sistema já que esses não costumam ser alterados durante o uso normal do sistema. Informações sobre arquivos de *log*, filas de correio eletrônico e de impressão, diretórios temporários e de usuários não devem ser armazenados na base de dados já que esses arquivos e diretórios são freqüentemente alterados.

3.2.1 Instalação e Configuração

Para realizar a instalação é necessário que o sistema operacional possua as bibliotecas libgcrypt[1] e libmhash[2] instaladas.

O pacote de instalação, .tar.gz, da ferramenta *AIDE* pode ser obtido no endereço http://www.cs.tut.fi/~rammer/aide.html. Essa ferramenta também se encontra disponível em pacotes .rpm (disponível em http://rpmfind.net) e .deb (disponível em http://www.debian.org). Cada pacote segue a sua forma tradicional de instalação.

A configuração do *AIDE* reside no arquivo /etc/aide.conf. Esse arquivo tem quatro tipos de linhas:

Linhas de comentários: iniciadas pelo símbolo "#".

[1] Disponível em http://ftp.linux.hr/pub/gcrypt/alpha/libgcrypt/
[2] Disponível em http://mhash.sourceforge.net/

Linhas de seleção: utilizadas para indicar quais arquivos terão suas informações adicionadas à base de dados.

Linhas de configuração: utilizadas para definir parâmetros de configuração do *AIDE*.

Linhas de macro: utilizadas para definir variáveis no arquivo de configuração.

Apenas as linhas de seleção são essenciais ao funcionamento do *AIDE*. As linhas de seleção são constituídas de um nome de arquivo (ou nome de diretório) e grupos de informações para a verificação. Existem, por sua vez, três tipos de linhas de seleção. Essas linhas são interpretadas como expressões regulares. Linhas que começam com uma barra "/" indicam que os arquivos que casarem com o padrão terão suas informações adicionadas a base de dados. Se a linha iniciar com um ponto de exclamação "!", ocorre o contrário: os arquivos que casam com o padrão são desconsiderados. Linhas iniciadas por um sinal de igualdade "=" informam ao *AIDE* que somente arquivos que sejam exatamente iguais ao padrão devem ser considerados. Para referir-se a um único arquivo deve-se colocar um "$" no final da expressão regular. Com isso, o padrão casará apenas com o nome exato do arquivo, desconsiderando arquivos que tenham o início do nome similar.

As linhas de configuração definem alguns parâmetros de funcionamento do *AIDE*. Essas linhas têm o formato parâmetro=valor. Os parâmetros de configuração estão descritos a seguir:

database O endereço do arquivo de onde as informações são lidas. Deve haver somente uma linha dessa. Se houver mais de uma, apenas a primeira será considerada.

database_out O endereço do arquivo de onde são escritas as informações. Assim como "*database*", deve haver apenas uma linha dessa. No caso de haver várias, somente a primeira ocorrência será considerada.

report_url O endereço onde a saída do *software* é escrita. Se existirem várias instâncias desse parâmetro, a saída será escrita em todos os endereços. Se o usuário não definir esse parâmetro, a saída será enviada para a saída padrão (*stdout*).

verbose Define o nível de mensagens que é enviado à saída. Pode assumir valores entre 0 e 255.

gzip_dbout Informa se a base de dados deve ser compactada ou não. Valores válidos para esse parâmetro são *yes*, *true*, *no* e *false*.

Definições de grupos Se o parâmetro não for nenhum dos anteriores, então ele é considerado uma definição de grupo. Embora existam alguns grupos predefinidos que informam ao *AIDE* quais as informações dos arquivos que

devem ser armazenadas na base de dados, o usuário pode criar suas próprias definições. O usuário pode, por exemplo, definir um grupo para verificar apenas o dono e o grupo dos arquivos da seguinte maneira: trivial=u+g. A Tabela 3.1 mostra os grupos predefinidos.

Tabela 3.1: Grupos predefinidos da ferramenta *AIDE*

Grupo	**Descrição**
p	permissões do arquivo
i	número do *inode*
n	número de *links*
u	uid do arquivo
g	gid do arquivo
s	tamanho do arquivo em *bytes*
m	data e hora da última modificação
a	data e hora do último acesso
c	data e hora da criação do arquivo
S	verifica o aumento do tamanho do arquivo
md5	*checksum* MD5 do arquivo
sha1	*checksum* SHA-1 do arquivo
rmd160	*checksum* RIPMED-160 do arquivo
tiger	*checksum* Tiger do arquivo
R	p+i+n+u+g+s+m+c+md5
L	p+i+n+u+g
E	grupo vazio
>	arquivo de *log* (aumenta o tamanho) - p+u+g+i+n+S

Por fim, as linhas de macro podem ser utilizadas para definir variáveis e tomar decisões baseadas no valor destas. Informações detalhadas podem ser encontradas na página de manual do arquivo de configuração (man aide.conf).

Um exemplo de configuração do arquivo /etc/aide.conf é apresentado na Figura 3.1.

```
# Localizacao da base de dados
database=file:/var/aide/aide.db
# Local onde e' criada uma nova base de dados
database_out=file:/var/aide/aide.db.new
# Arquivo onde sera salva a saida do programa
report_url=file:/var/aide/report.aide

# Grupo para verificacao dos arquivos
minhaRegra=md5+u+g+p+s

# Analisa os arquivos contidos nos diretórios /bin,
# /sbin, /etc, /usr/bin e /usr/sbin

/bin minhaRegra
/sbin md5+u+g+p+s
/etc minhaRegra
/usr/bin minhaRegra
/usr/sbin minhaRegra
```

Figura 3.1: Exemplo de um arquivo /etc/aide.conf

3.2.2 Utilização do *AIDE*

Após a edição do arquivo de configuração /etc/aide.conf, o usuário pode gerar o *snapshot* dos arquivos do sistema usando a seguinte seqüência de comandos:

```
# /usr/bin/aide -i
# mv aide.db.new aide.db
```

Para verificar a integridade dos arquivos do sistema, basta executar o comando:

```
# /usr/bin/aide -C
```

Os arquivos que sofreram qualquer mudança, seja no tamanho, conteúdo, permissões, etc, serão enviados para o arquivo report.aide. Um exemplo de relatório gerado pelo *AIDE* é apresentado no Apêndice A. Outras informações sobre a ferramenta *AIDE* podem ser obtidas em [Lehti (2003)].

3.3 A Ferramenta *Integrit*

O *Integrit* é um *software* para controle de integridade de arquivos. O *software* ajuda o administrador a determinar se um invasor modificou ou não os arquivos de um computador. As principais vantagens do *Integrit* são sua simplicidade, pequeno consumo de memória, atualização de algoritmos criptográficos, conjunto de regras em cascata, entre outras vantagens.

O *Integrit* funciona através da criação de uma base de dados como *snapshot* (que deve ser mantida em local seguro) das partes mais essenciais do sistema sendo utilizado para comparações do *software* e, em caso de incidente, o administrador saberá exatamente o que foi modificado, adicionado ou removido no sistema [Cashin (2003)].

3.3.1 Instalação e Configuração

O pacote de instalação, .tar.gz, da ferramenta *Integrit* pode ser obtido no endereço http://integrit.sourceforge.net. Pacotes .rpm e .deb podem ser obtidos nos endereços http://rpmfind.net e http://www.debian.org. Cada pacote segue a sua forma tradicional de instalação.

A configuração do *Integrit* reside em um arquivo como o integrit.conf (Figura 3.2). Esse arquivo possui vários elementos de configuração que são descritos a seguir:

Comentários O seguintes tipos de linhas são ignorados no arquivo de configuração: linhas em branco, linhas consistindo apenas de espaços em branco e linhas iniciadas pelo símbolo "#".

Base de dados *known* O local onde a base de dados *known* (contendo informações sobre o estado anterior dos arquivos) será armazenada é especificado pela linha: known=/var/integrit/integrit_known.cdb.

Base de dados *current* O local onde a base de dados *current* (contendo informações sobre o estado atual dos arquivos) será armazenada é especificado pela linha: current=/var/integrit/integrit_current.cdb.

Raiz para verificação da integridade Define o local onde será iniciado a verificação dos arquivos. Um exemplo configurando o inicio da verificação pelo diretório "/" é representado pela linha: root=/.

Regras As regras do arquivo de configuração dizem ao *Integrit* como realizar a verificação dos diretórios e arquivos. O usuário pode dizer ao *Integrit*, por exemplo, para não gerar o *checksum* dos arquivos de *log*.

Cada regra possui um prefixo (opcional), um nome de arquivo (ou nome de diretório) e o conjunto de verificações que o *Integrit* irá realizar. Regras que tem como prefixo uma barra "/" indicam que os arquivos que casarem com o padrão terão suas informações adicionadas a base de dados. Se a regra tem como prefixo um ponto de exclamação "!", ocorre o contrário: os arquivos que casam com o padrão são desconsiderados. Regras iniciadas por um sinal de igualdade "=" informam ao *Integrit* que somente arquivos que sejam exatamente iguais ao padrão devem ser considerados. Por fim, regras

que tem como prefixo um sinal de dólar "$" indicam que a regra é *non-cascading*, ou seja, a regra não é herdada pelos subdiretórios e arquivos.

O conjunto de verificações diz ao *Integrit* quais informações devem ou não ser armazenadas na base de dados. Um conjunto de verificações é apresentado na Tabela 3.2.

Tabela 3.2: Conjunto de verificações do *Integrit*

Opção	Descrição
s	*checksum* MD5
i	número do *inode*
p	permissões do arquivo
l	número de *links*
u	uid do arquivo
g	gid do arquivo
z	tamanho do arquivo em *bytes*
m	data e hora da última modificação
a	data e hora do último acesso
c	verifica se as informações do arquivo de tempo UN*X foram alteradas
r	reinicia o tempo de acesso

Em uma regra, se a opção de verificação for definida com letra maiúscula, a informação será ignorada. Já se a opção de verificação for definida com letra minúscula, a informação será adicionada à base de dados. Um exemplo de configuração do arquivo integrit.conf é apresentado na Figura 3.2.

3.3.2 Utilização do *Integrit*

Após a configuração do arquivo integrit.conf, o usuário pode gerar o *snapshot* dos arquivos do sistema usando a seguinte seqüência de comandos:

```
# /usr/local/sbin/integrit -C integrit.conf -u
# mv integrit_current.cdb integrit_known.cdb
```

Para verificar a integridade dos arquivos do sistema, basta executar o comando:

```
# /usr/local/sbin/integrit -C integrit.conf -c
```

Os arquivos que sofreram qualquer mudança, seja no tamanho, conteúdo, permissões, etc, serão listados na tela. Um exemplo de relatório gerado pelo *Integrit* é apresentado no Apêndice A. Outras informações sobre a ferramenta *Integrit* podem ser obtidas em [Cashin (2003)].

```
# Localizacao da base de dados conhecida
known=/var/integrit/integrit_known.cdb
# Local onde e' criada uma base de dados nova
current=/var/integrit/integrit_current.cdb
# Local na arvore de arquivos onde sera iniciado a verificação
root=/

# Diretórios com as respectivas informações que irão ou não ser
# analizados

!/dev
!/home
!/initrd
!/lib
!/lost+found
!/misc
!/mnt
!/opt
!/proc
!/root
!/tmp
!/usr
!/var
!/boot
/bin sugpz
/sbin sugpz
/etc sugpz
/usr/bin sugpz
/usr/sbin sugpz
```

Figura 3.2: Exemplo de um arquivo integrit.conf

3.4 A Ferramenta *Osiris*

Osiris é um sistema de controle de integridade de arquivos que pode ser utilizado para monitorar mudanças nos arquivos do sistema [The Shmoo Group (2001)].

O *Osiris* consiste de um par de aplicativos: *osiris* e *scale*. O primeiro, *osiris*, é utilizado para coletar dados específicos do sistema de arquivos local e armazená-los em uma base de dados. O segundo aplicativo, *scale*, é utilizado para analisar e/ou comparar as diferenças entre duas bases de dados. Juntos esses dois aplicativos fornecem ao administrador facilidades para especificar atributos e tipos de arquivos para monitoração.

3.4.1 Instalação e Configuração

O *Osiris* pode ser obtido no endereço http://osiris.shmoo.com. Essa ferramenta se encontra disponível apenas em pacotes .tar.gz, sendo que, esse pacote segue a sua forma tradicional de instalação.

A configuração do *Osiris* reside em um arquivo como o osiris.conf (Figura 3.3).

Esse arquivo é muito similar ao arquivo de configuração usado pelo Apache[3]. O usuário especifica um arquivo de configuração para o aplicativo *osiris* usando o argumento "-f". Embora o aplicativo *osiris* possa ser utilizado sem um arquivo de configuração, não é recomendado fazê-lo devido a limitação de suas funcionalidades.

Um arquivo de configuração consiste de uma seqüência de blocos. Um bloco especifica um nome de diretório, uma lista de opções para verificação e uma lista de ações a serem realizadas. Os blocos devem ser especificados pelas marcações: <Directory> </Directory>. O arquivo de configuração também permite especificar uma seção de configuração global. Caso não seja especificada nenhuma opção a um bloco, as configurações da seção global serão assumidas pelo bloco. Se um bloco é um subdiretório de um outro bloco, o bloco do subdiretório terá precedência.

Lista de Opções para Verificação

Opções ditam a natureza da verificação dos arquivos e diretórios. Usualmente as opções aparecem primeiro nos blocos de configuração, mas sua posição dentro do bloco é irrelevante. As opções que não estão dentro de um bloco são chamadas de globais. A lista de opções do *Osiris* para a verificação da integridade de arquivos é apresentada a seguir.

Recursive <*bool*> Permite a verificação de diretórios recursivamente.

FollowLinks <*bool*> Permite seguir qualquer *link* simbólico. Se um *link* simbólico é um *link* para um diretório, esse *link* somente será seguido se a opção *Recursive* for configurada.

Verbose <*bool*> Exibe na saída padrão as informações sobre a verificação.

Prompt <*bool*> Se a base de dados para os blocos não puder ser aberta, ou se a base já existir, o usuário terá através de um *prompt*, a opção de parar a verificação ou sobrescrever a base de dados.

ShowErrors <*bool*> Enquanto verifica os diretórios, exibe na saída padrão qualquer erro encontrado. Essa opção é independente da opção *Verbose*.

Hash <*hash*> Especifica o algoritmo de hash que será utilizado nos arquivos ou diretórios. Essa opção pode assumir os valores *md5*, *haval*, *sha* e/ou *ripemd*, que são referentes aos algoritmos de geração de *checksum* ("*hash*"): MD5 [Touch (1995)], HAVAL [Zheng et al (1992)], SHA-1 [Nist (1995)] e/ou RIPMED [Dobbertin et al (1996)].

[3]Para maiores informações sobre esse arquivo, consulte o site http://www.apache.org.

Database <*string*> Especifica a base de dados onde será armazenado as informações referentes aos arquivos verificados. A sintaxe para "*string*" dependerá do tipo da base de dados utilizada (*Osiris* permite o uso de base de dados GDBM ou MySQL). Para base de dados GDBM (default), a "*string*" é o caminho do arquivo onde será armazenado as informações.

Lista de Ações

Uma ação permite incluir ou excluir arquivos, ou especificamente ou por propriedades. As ações listadas primeiro possuem precedência. A lista de ações e seus respectivos formatos são apresentados a seguir.

IncludeAll <*attr*> Inclui todos os arquivos com os atributos especificados em "*attr*".

ExcludeAll Exclui todos os arquivos. Qualquer ação após essa não terá efeito.

Include <*filter*> **[***attr***]** Inclui todos os arquivos, com os atributos especificados em "*attr*", que passam por um filtro. Devida a não necessidade do uso de filtros neste trabalho, o mesmo não irá abordar tal assunto. Para maiores informações sobre filtros, consulte [The Shmoo Group (2001)].

Exclude <*filter*> **[***attr***]** Exclui todos os arquivos que passam pelo filtro.

NoEntry <*directory*> Não entra no diretório especificado.

Os atributos da lista de atributos ("*attr*") são separados por vírgula (","). A lista de atributos da ferramenta *Osiris* é apresentada na Tabela 3.3.

Tabela 3.3: Lista de atributos do *Osiris*

Atributo	Descrição
inode	número do *inode*
perm	permissões do arquivo
links	número de *hard links*
uid	uid do arquivo
gid	gid do arquivo
bytes	tamanho do arquivo em *bytes*
blocks	tamanho do arquivo em blocos
mtime	data e hora da última modificação
atime	data e hora do último acesso
flags	qualquer marcação de arquivo oculto
all	todos os atributos acima

Um exemplo de configuração do arquivo osiris.conf é apresentado na Figura 3.3.

```
# ------------------------------------------------
#
# GLOBAL SECTION
#
# this sets up the global values for the directory blocks
# specified below. The following keywords are supported:
#
# runtime configuration, can be overriden by any
# command line arguments specified.

Database    /var/osiris/current.osi
Verbose     no
Prompt      yes
ShowErrors  yes
Recursive   yes
FollowLinks no


# default attributes to get for files. The default is to
# get every attribute, except "atime", about files.
IncludeAll perm,uid,gid,bytes


# default hash algorithm to use is sha.
Hash md5


# RULE SECTION - specify all the files or directories to
# be scanned including any custom attribute
# or options. a block with no rules inherits
# the default rule, in this case including all files.


# monitor all /etc files.
<Directory /etc>
    IncludeAll perm,uid,gid,bytes
</Directory>

# monitor all /bin files.
<Directory /bin>
</Directory>

# monitor all /sbin files.
<Directory /sbin>
    IncludeAll perm,uid,gid,bytes
</Directory>

# monitor all /usr/bin files.
<Directory /usr/bin>
</Directory>

# monitor all /usr/bin files.
<Directory /usr/sbin>
</Directory>

# EOF
```

Figura 3.3: Exemplo de um arquivo osiris.conf

3.4.2 Utilização do *Osiris*

Após a configuração do arquivo osiris.conf, o usuário pode gerar o *snapshot* dos arquivos do sistema, usando a seguinte seqüência de comandos:

```
# osiris -f osiris.conf
# mv current.osi known.osi
```

No *Osiris* é necessário ter duas bases de dados para realizar a verificação da integridade de arquivos e diretórios. Para verificar a integridade dos arquivos e diretórios do sistema, basta executar o comando:

```
# osiris -f osiris.conf
# scale -l known.osi -r current.osi -c osiris.conf -o relatorio.txt
```

Os arquivos que sofreram qualquer mudança, seja no tamanho, conteúdo, permissões, etc, serão enviados para o arquivo relatorio.txt. Um exemplo de relatório gerado pelo *Osiris* é apresentado no Apêndice A. Outras informações sobre a ferramenta *Osiris* podem ser obtidas em [The Shmoo Group (2001)].

3.5 A Ferramenta *Tripwire*

O *Tripwire* é um controlador de integridade de arquivos e diretórios que compara um conjunto de arquivos e diretórios com informações de uma base de dados gerada anteriormente. Qualquer diferença é apontada, incluindo entradas adicionadas ou removidas. Com isso é possível tomar medidas imediatamente para evitar danos quando alguma mudança for apontada em arquivos críticos do sistema. Com o *Tripwire*, os administradores do sistema podem concluir, com um alto grau de certeza, que um dado conjunto de arquivos permanecem livres de modificações, se o *Tripwire* não acusar mudanças [Tripwire, Inc (2000)].

3.5.1 Instalação e Configuração

O *Tripwire* pode ser obtido no endereço http://www.tripwire.org. Essa ferramenta se encontra disponível em pacotes .tar.gz e .rpm. Ambos os pacotes seguem sua forma tradicional de instalação com a exceção da necessidade de executar o *script* install.sh após a instalação.

Durante a execução do *script* (install.sh), o *Tripwire* irá pedir duas senhas. A primeira é a senha de site, utilizada para assinar vários arquivos como as bases de dados, o arquivo de configuração do *Tripwire* (twcfg.txt) e o arquivo de regras (twpol.txt). Após a instalação, essa senha é exigida pelo aplicativo twadmin. A segunda é a senha de chave local exigida pelo aplicativo tripwire durante a criação

da base de dados e verificação da integridade dos arquivos. Durante a instalação, também será pedido a confirmação para cada um das senhas. Ao final será exibida uma mensagem informando que a instalação foi bem sucedida.

Antes de utilizar o *Tripwire* é preciso personalizar dois arquivos [Vilela (2001)]:

1. O arquivo de configuração do *Tripwire*;
2. O arquivo de regras do *Tripwire*.

O arquivo de configuração é o twcfg.txt, esse arquivo armazena informações específicas do sistema e encontra-se no diretório /etc/tripwire/. O arquivo de regras é o twpol.txt. Esse arquivo armazena a especificação de quais objetos (arquivos e diretórios) o *Tripwire* deve monitorar bem como suas localizações (também encontra-se no diretório /etc/tripwire/).

Devido a simplicidade e também pelo arquivo twcfg.txt contido no pacote de instalação fornecer as configurações necessárias ao funcionamento do *Tripwire*, o mesmo não será descrito neste trabalho. Informações detalhadas podem ser encontradas na documentação do *Tripwire*.

Por outro lado, o arquivo de regras (twpol.txt) fornece recursos para realizar desde uma configuração simples até uma configuração avançada. Entre esses recursos, tem-se:

- definição de regras para verificação;
- uso da marcação "#" para inserir comentários no arquivo de regras;
- envio de email, para um determinado endereço, durante a verificação dos arquivos;
- especificação de nomes para as regras, facilitando a identificação das regras no relatório gerado pelo *Tripwire*;
- uso de variáveis, possibilitando a alteração dos parâmetros de várias regras de uma só vez;
- regras como procedimentos, permitindo o usuário aplicar uma regra a um grupo de arquivos ou diretórios;
- entre outros.

Embora o *Tripwire* apresente uma grande quantidade de recursos, neste trabalho apenas os recursos de configuração mais simples serão apresentados. Informações detalhadas sobre os recursos avançados podem ser encontradas na documentação do *Tripwire*.

Uma regra simples no arquivo de regras é constituída de um nome de arquivo (ou nome de diretório) , das marcações "->" e "+", do grupo de informações que

devem ser verificadas e de um ponto e vírgula (“;”) colocado no final da regra. Caso o nome do arquivo (ou diretório) seja precedido do símbolo “!”, o *Tripwire* não irá realizar a verificação desse arquivo (ou diretório). As informações que podem ser verificadas pelo *Tripwire* são apresentadas na Tabela 3.4.

Tabela 3.4: Informações de verificação do *Tripwire*

Informação	**Descrição**
a	data e hora do último acesso
b	tamanho do arquivo em blocos
c	data e hora de criação ou modificação do *inode*
d	número do dispositivo de disco que armazena o *inode*
u	uid do arquivo
g	gid do arquivo
i	número do *inode*
m	data e hora da última modificação
n	quantidade de referências ao *inode*
p	permissão e bits de modo do arquivo
s	tamanho do arquivo em *bytes*
t	tipo de arquivo
l	verifica o aumento do tamanho do arquivo
C	*checksum* CRC-32 do arquivo
M	*checksum* MD5 do arquivo
S	*checksum* SHA-1 do arquivo
H	checksum *Haval* do arquivo

Um exemplo de configuração do arquivo twpol.txt é apresentado na Figura 3.4.

```
# Informações para verificação da integridade de arquivos e
# diretorios

/bin -> +Mugps (rulename=diretorio_bin);
/sbin -> +Mugps (rulename=diretorio_sbin);
/etc -> +Mugps (rulename=diretorio_etc);
/usr/bin -> +Mugps (rulename=diretorio_usr_bin);
/usr/sbin -> +Mugps (rulename=diretorio_usr_sbin);
```

Figura 3.4: Exemplo de um arquivo twpol.txt

Para finalizar a configuração do *Tripwire*, o usuário dever ir ao diretório /etc/tripwire/ e digitar (em um terminal) o comando:

/usr/sbin/twadmin --create-cfgfile --site-keyfile site.key twcfg.txt

Em resposta o *Tripwire* vai pedir ao usuário a sua senha de site. Em seguida é necessário atualizar o arquivo de regras. Para isso digite (em um terminal) o seguinte comando:

```
# /usr/sbin/twadmin --create-polfile twpol.txt
```

Novamente será pedido ao usuário a senha de site. Agora o *Tripwire* está pronto para ser utilizado.

3.5.2 Utilização do *Tripwire*

Após a configuração do *Tripwire*, o usuário pode gerar o *snapshot* dos arquivos do sistema, usando o seguinte comando:

```
# /usr/sbin/tripwire --init
```

Nesse momento, será pedido a senha de chave local. O que acontece a seguir depende da configuração do sistema do usuário. Se o usuário não editar adequadamente o arquivo de regras, poderá ocorrer vários erros que deverão ser corrigidos.

Para verificar a integridade dos arquivos do sistema, basta executar o comando:

```
# /usr/sbin/tripwire --check
```

Os arquivos que sofreram qualquer mudança, seja no tamanho, conteúdo, permissões, etc, serão listados. Um exemplo de relatório gerado pelo *Tripwire* é apresentado no Apêndice A. Outras informações sobre a ferramenta *Tripwire* podem ser obtidas em [Tripwire, Inc (2000)].

3.6 Considerações Finais

Durante o uso de ferramentas de controle de integridade de arquivos, normalmente, o ideal é ignorar diretórios que são modificados com muita freqüência, a não ser que o administrador do sistema goste de *logs* gigantescos. É um procedimento recomendado excluir diretórios temporários, filas de impressão, diretórios de *logs* e quaisquer outras áreas freqüentemente modificadas. Por outro lado, é recomendado que sejam incluídos todos os binários, bibliotecas e arquivos de cabeçalhos do sistema. Muitas vezes é interessante incluir diretórios que o administrador não costuma observar, como o /dev/ e o /usr/man. Esses diretórios, muitas vezes, recebem pouca atenção dos administradores de sistemas, o que os tornam úteis para serem utilizados em possíveis tentativas de invasão.

As ferramentas descritas neste capítulo possuem várias características que tornam interessante, o uso de cada uma, nas mais diversas situações. No Capítulo 4 é apresentado um estudo de caso e uma análise comparativa das principais características das quatro ferramentas aqui descritas.

Capítulo 4

Análise e Avaliação de Ferramentas para Controle de Integridade de Arquivos

4.1 Considerações Iniciais

Atualmente existem disponíveis várias ferramentas para controle de integridade de arquivos. Além disso, cada ferramenta possui algumas características que tornam o seu uso interessante em determinadas situações.

Essa grande quantidade de ferramentas levam os administradores de sistemas a terem dificuldades na escolha de uma determinada ferramenta de controle de integridade de arquivos, uma vez, que não existem informações comparativas entre as várias ferramentas disponíveis. Nesse contexto, o objetivo deste capítulo é apresentar uma análise comparativa e um estudo de caso sobre as quatro ferramentas descritas no Capítulo 3.

4.2 Análise das Ferramentas *AIDE*, *Integrit*, *Osiris* e *Tripwire*

Nesta seção é descrita uma análise comparativas das ferramentas *AIDE* (versão 0.8), *Integrit* (versão 3.02.00), *Osiris* (versão 1.5.2) e *Tripwire* (versão 2.3-47) tendo-se critérios de comparação como: tipos de pacotes de instalação disponíveis, licença de uso das ferramentas, informações apresentadas pelos relatórios gerados pelas ferramentas. Os critérios de comparação são divididos em subseções e analisados para cada uma das quatro ferramentas em estudo.

4.2.1 Uso Avançado do Arquivo de Configuração de Regras

Em relação às quatro ferramentas em estudo, apenas o *Osiris* e o *Tripwire* permitem o uso de recursos avançados em seus arquivos de configuração de regras. As ferramentas *AIDE* e *Integrit* permitem apenas a realização de configurações simples, como apresentado nas subseções 3.2.1 e 3.3.1.

A ferramenta *Osiris*, além dos recurso de configuração apresentados na subseção 3.4.1, permite a configuração de filtros [The Shmoo Group (2001)]. Já a ferramenta *Tripwire* possui vários recursos avançados, como o envio de email, uso de variáveis, especificação de diretivas, regras como procedimentos, entre outros. Informações sobre os recursos avançados de configuração de regras do *Tripwire* podem ser obtidas na documentação da ferramenta.

4.2.2 Licença de Uso das Ferramentas

Com relação às licenças de uso, as ferramentas *AIDE* e *Integrit* são licenciadas pela *GNU General Public License*[1]. A ferramenta *Osiris* possui licença própria [The Shmoo Group (2001)] com algumas condições semelhantes a *GPL*. Já o *Tripwire* possui uma versão gratuita licenciada pela *GPL*[2] e uma versão comercial com licença própria[3].

4.2.3 Pacotes Disponíveis para Instalação

A seguir são descritos os tipos de pacotes disponíveis para cada uma das quatro ferramentas em estudo:

AIDE: Pacote .tar.gz disponível em http://www.cs.tut.fi/~rammer/aide.html, pacote .rpm disponível em http://rpmfind.net e pacote .deb disponível em http://www.debian.org.

Integrit: Pacote .tar.gz disponível em http://integrit.sourceforge.net, pacote .rpm disponível em http://rpmfind.net e pacote .deb disponível em http://www.debian.org.

Osiris: Pacote .tar.gz disponível em http://osiris.shmoo.com.

Tripwire: Pacotes .tar.gz e .rpm ambos disponíveis em http://www.tripwire.org.

[1] Para maiores informações sobre a licença, consulte http://www.fsf.org.

[2] Disponível em http://www.tripwire.org.

[3] Maiores informações sobre a licença comercial do *Tripwire* podem ser obtidas em http://www.tripwire.com.

4.2.4 Documentação Disponível

As quatro ferramentas possuem uma boa documentação em seu sites, como apresentado no Capítulo 3. Todas as ferramentas possuem a sua documentação na língua inglesa. A ferramenta *AIDE* possui, também, uma tradução de sua documentação para a língua portuguesa do Brasil. Essa tradução foi realizada pela Empresa *Conectiva* e se encontra disponível em http://www.conectiva.com.br.

4.2.5 Tipo de Base de Dados

Todas as ferramentas armazenam as informações, obtidas durante a verificação, em arquivos. A ferramenta *Osiris* permite, também, que as informações sejam armazenadas em uma base de dados MySQL. Já a ferramenta *AIDE* permite que as informações obtidas seja armazenadas em um base de dados PostgreSQL.

4.2.6 Formato do Relatório de Verificação

As ferramentas *Integrit* e *Tripwire* exibem o relatório de verificação na saída padrão (*stdout*), entretanto, a saída dessas ferramentas podem ser redirecionadas para um arquivo ASCII usando o operador de redirecionamento de saída ">>"[4]. O *Integrit* também possui a opção de gerar o relatório no formato de um arquivo XML[5]. Entre as várias vantagens de se usar o formato XML, pode-se citar a facilidade de visualização do relatório em uma grande variedade de aplicativos que adotam o XML como formato interno de seus documentos.

Já as ferramentas *AIDE* e *Osiris* (na verdade a ferramenta *scale*) enviam o relatório de verificação para um arquivo ASCII.

4.2.7 Senhas para Autenticação

De todas as ferramentas analisadas, apenas o *Tripwire* exige que o usuário entre com uma senha para poder utilizar a ferramenta e acessar as bases de dados.

4.2.8 Modo de Funcionamento das Ferramentas

As ferramentas *AIDE*, *Osiris* e *Tripwire* apenas verificam a integridade dos arquivos (ou diretórios) especificados nas regras dos arquivos de configuração de regras.

Já a ferramenta *Integrit* inicia a verificação da integridade dos arquivos (ou diretórios) pelo diretório definido na variável de configuração "root=". Assim, o *Integrit* realiza a verificação de integridade de todos os arquivos e diretórios abaixo do diretório definido em "root=" e, não apenas, dos arquivos e diretório especificados nas regras do arquivo de configuração de regras.

[4]Maiores informações sobre redirecionamento de saídas podem ser obtidas em http://focalinux.cipsga.org.br/guia/iniciante/ch-redir.htm#s-redir-maior2.

[5]Consulte http://www.w3.org/XML/, para maiores informações sobre arquivos XML.

4.2.9 Informações para Verificação de Integridade

Na Tabela 4.1 é apresentada uma lista de informações que podem ser geradas, durante a análise dos arquivos (ou diretórios), com o uso das quatro ferramentas de verificação de integridade.

Tabela 4.1: Informações para verificação de integridade

Informação	**Ferramentas**			
	AIDE	*Integrit*	*Osiris*	*Tripwire*
permissões do arquivo	Sim	Sim	Sim	Sim
número do *inode*	Sim	Sim	Sim	Sim
número de *links*	Sim	Sim	Sim	Sim
uid do arquivo	Sim	Sim	Sim	Sim
gid do arquivo	Sim	Sim	Sim	Sim
tamanho do arquivo em *bytes*	Sim	Sim	Sim	Sim
tamanho do arquivo em blocos	Não	Não	Sim	Sim
data e hora da última modificação	Sim	Sim	Sim	Sim
data e hora do último acesso	Sim	Sim	Sim	Sim
data e hora da criação do arquivo	Sim	Não	Não	Não
data e hora de criação ou modificação do *inode*	Não	Não	Não	Sim
verifica o aumento do tamanho do arquivo	Sim	Não	Não	Sim
checksum MD5 do arquivo	Sim	Sim	Sim	Sim
checksum SHA-1 do arquivo	Sim	Não	Sim	Sim
checksum RIPEMD-160 do arquivo	Sim	Não	Não	Não
checksum RIPEMD do arquivo	Não	Não	Sim	Não
checksum Tiger do arquivo	Sim	Não	Não	Não
checksum CRC-32 do arquivo	Não	Não	Não	Sim
checksum HAVAL do arquivo	Não	Não	Sim	Sim
grupo vazio	Sim	Não	Não	Não
alterações no arquivo de tempo UN*X	Não	Sim	Não	Não
reinicia o tempo de acesso	Não	Sim	Não	Não
qualquer marcação de arquivo oculto	Não	Não	Sim	Não
número do dispositivo de disco do *inode*	Não	Não	Não	Sim
tipo de arquivo	Não	Não	Não	Sim

4.2.10 Ferramentas Auxiliares

Das quatro ferramentas analisadas, apenas o *Integrit* possui ferramentas auxiliares incluídas em seu pacote de instalação – o *i-viewdb* e o *i-ls*. A ferramenta auxiliar *i-viewdb* permite o usuário ver as informações armazenadas na base de dados ge-

rada pelo *Integrit*. Já o *i-ls* é uma ferramenta *standalone* que permite o usuário ver informações sobre arquivos e diretórios (as mesmas informações que são geradas pela ferramenta *Integrit*). Maiores informações sobre as ferramentas auxiliares *i-viewdb* e *i-ls* podem ser obtidas em [Cashin (2003)].

4.3 Estudo de Caso

O objetivo deste estudo de caso é verificar a eficiência das quatro ferramentas de controle de integridade de arquivo perante alterações sofridas por um sistema, bem como obter o tempo que cada ferramenta leva para gerar a base de dados e também para verificar a integridade dos arquivos, buscando, desse modo, identificar a ferramenta mais lenta e a mais rápida.

O estudo de caso foi realizado em um micro computador PC Athlon XP 1.7 GHz, memória DDR de 512 MB e sistema operacional Linux - *RedHat 8*. Por questões de simplicidade foram analisados apenas os arquivos contidos nos diretórios /sbin, /bin, /etc, /usr/sbin e /usr/bin. Para cada arquivo analisado foram geradas as seguintes informações: *checksum* MD5, uid, gid, permissões e tamanho do arquivo.

Os arquivos de configuração de regras para cada uma das quatro ferramentas de controle de integridade de arquivos, os quais descrevem o contexto apresentado anteriormente, podem ser visualizados na Figura 3.1 (*AIDE*), Figura 3.2 (*Integrit*), Figura 3.3 (*Osiris*) e Figura 3.4 (*Tripwire*).

Para verificar a eficiência das ferramentas foi simulado uma situação de invasão do sistema, em que, foram realizadas as seguintes alterações:

- inserção de um arquivo ssftp (desenvolvido pelo invasor) no diretório /usr/bin;
- inserção de um arquivo ssftp.conf (desenvolvido pelo invasor) no diretório /etc;
- substituição do arquivo useradd localizado no diretório /usr/sbin por um arquivo useradd desenvolvido pelo invasor;
- remoção do arquivo ps localizado no diretório /bin;
- inserção de um usuário comum;
- inserção de um usuário com permissões de root.

As quatro ferramentas foram executadas antes e depois da simulação de invasão, sendo que, os tempos[6] obtidos para gerar a base de dados e para realizar a

[6] Os tempos foram obtidos em segundos (s).

verificação da integridade dos arquivos são descritos na Tabela 4.2. Cabe ressaltar que na Tabela 4.2, os valores 3 e 5 segundos obtidos pela ferramenta *Osiris*, durante a geração da base de dados, são referentes a geração da base de dados antes (3 s) e depois da simulação de invasão do sistema (5 s), uma vez que, a ferramenta *Osiris* necessita de duas bases de dados para realizar a verificação da integridade dos arquivos e diretórios. Já aos tempos 19 e 18 segundos, obtidos pelo *Tripwire*, estão acrescidos um tempo de aproximadamente 3 segundos referente a digitação da senha necessária para a utilização da ferramenta *Tripwire*. O relatório de resposta gerado por cada uma das quatro ferramentas são apresentados na Figura A.1 (*AIDE*), Figura A.2 (*Integrit*), Figura A.3 (*Osiris*) e Figura A.4 (*Tripwire*) do Apêndice A.

Tabela 4.2: Análise de tempo das ferramentas *AIDE, Integrit, Osiris* e *Tripwire*

	Ferramentas			
	AIDE	*Integrit*	*Osiris*	*Tripwire*
Geração da base de dados	3 s	1 s	3 e 5 s	19 s
Verificação dos arquivos	3 s	1 s	1 s	18 s

Analisando os valores descritos na Tabela 4.2 e, também as consideração apresentadas sobre essa tabela, é possível verificar que a ferramenta mais lenta foi o *Tripwire* e a mais rápida o *Integrit*. Já uma análise dos relatórios de resposta apresentados no Apêndice A mostra que as quatro ferramentas identificaram os dois usuário inseridos durante a simulação da invasão do sistema. Isso pode ser verificado através da alteração dos arquivos /etc/group, /etc/group-, /etc/passwd, /etc/passwd-, /etc/shadow, /etc/shadow- e /etc/gshadow-, apresentada nos relatórios de resposta. Cabe ressaltar que a ferramenta *Integrit* também identificou alterações no arquivo /etc/gshadow.

Com relação aos arquivos adicionados, removidos e alterados, durante a simulação da invasão do sistema, as ferramentas *AIDE, Osiris* e *Tripwire* identificaram todos esses arquivos. A ferramenta *Integrit* identificou apenas os arquivos /etc/ssftp.conf (adicionado) e /bin/ps (removido). O *Integrit* não foi capaz de identificar o arquivo /usr/bin/ssftp (adicionado) e o arquivo /usr/sbin/useradd (alterado). Tal fato aponta que a ferramenta *Integrit* pode possuir problemas de implementação.

4.4 Avaliação das Ferramentas *AIDE, Integrit, Osiris* e *Tripwire*

As informações apresentadas nesta monografia permitem verificar que dentre as ferramentas analisadas, o *Tripwire*, embora apresente maior dificuldade na configuração de seus arquivos, é a ferramenta mais completa e a que apresenta um

maior número de recursos de segurança. Assim, o seu uso no controle de integridade de arquivos é altamente recomendado em situações que exijam um alto nível de segurança uma vez que a grande quantidade de recursos que a ferramenta possui permitem que a mesma empregue grande segurança nos arquivos de um sistema. O uso da ferramenta *Tripwire* não é recomendado apenas em situações que exigem que o controle de integridade dos arquivos seja realizado com rapidez já que entre as ferramentas analisadas, o *Tripwire* foi a que apresentou tempo de execução mais lento.

As ferramentas *AIDE* e *Osiris* diferem em dois aspectos principais: a sua forma de configuração e a possibilidade do uso de filtros pela ferramenta *Osiris*. Por outro lado, os recursos de segurança oferecidos por essas ferramentas são muito semelhantes. Por apresentarem uma quantidade de recursos e configurações inferior ao *Tripwire*, o uso do *AIDE* e *Osiris* é recomendado para situações que não necessitem de um nível de segurança tão alto no controle de integridade dos arquivos. Além disso, ambas as ferramentas são altamente recomendadas para situações em que se necessite que o controle de integridade de arquivos seja realizado com rapidez. Isso porque as informações apresentadas na Seção 4.3 demonstraram que ambas as ferramentas são extremamente rápidas quando comparadas à ferramenta *Tripwire*.

A ferramenta *Integrit*, por sua vez, apresenta uma quantidade de recursos semelhantes às ferramentas *AIDE* e *Osiris*. Segundo as informações apresentadas na Seção 4.3, o *Integrit* é considerada a ferramenta mais rápida entre as ferramentas analisadas, mas não foi capaz de identificar todas as alterações realizadas durante a simulação de invasão. Dessa forma, o uso da ferramenta *Integrit* não é recomendado, pois a mesma provavelmente possui problemas de implementação.

4.5 Considerações Finais

Este capítulo teve como objetivo a análise e a comparação das ferramentas de controle de integridade de arquivos *AIDE, Integrit, Osiris* e *Tripwire*, procurando fornecer ao administrador de sistemas informações que o possibilite escolher a ferramenta de controle de integridade de arquivos que seja mais adequada a uma determinada situação.

Capítulo 5

Conclusão

Uma das primeiras ações de um invasor costuma ser substituir arquivos e programas do sistema com o intuito de mascarar sua visita atual e, principalmente, facilitar as visitas futuras.

Uma boa prática para identificar alterações em arquivos e programas do sistema e, conseqüentemente, identificar possíveis invasões de sistemas, consiste no uso de ferramentas de controle de integridade de arquivos.

Ferramentas de controle de integridade de arquivos como o *AIDE, Integrit, Osiris* e *Tripwire*, que focaram os estudos deste trabalho, constroem uma base de dados refletindo o estado atual do sistema e utilizam essa base de dados atual para futuramente verificar a integridade dos arquivos do sistema.

Este trabalho procurou analisar e comparar as ferramentas de controle de integridade de arquivos *AIDE, Integrit, Osiris* e *Tripwire*, para fornecer ao administrador de sistemas informações que o possibilite escolher, entre as quatro ferramentas apresentadas, a ferramenta de controle de integridade de arquivos que seja mais adequada a uma determinada situação.

Buscando atingir os objetivos definidos para este trabalho, o Capítulo 2 apresentou várias atividades que dever ser realizadas durante o controle de integridade de arquivos. A definição de cada uma das quatro ferramentas, bem como, as informações referentes a instalação, configuração e uso, apresentadas no Capítulo 3, demonstraram a facilidade de uso das ferramentas *AIDE, Integrit, Osiris* e *Tripwire* no controle de integridade de arquivos. Já no Capítulo 4 foram apresentadas várias informações comparativas sobre as quatro ferramentas em estudo, procurando fornecer ao administrador de sistema recursos que o possibilite escolher a ferramenta mais adequada para uma determinada situação. Além disso, no Capítulo 4 foi apresentado um estudo de caso que demonstrou que a ferramenta de controle de integridade de arquivos mais rápida é a ferramenta *Integrit* e que a ferramenta *Tripwire* é a mais lenta. Ao final, a análise do relatório de resposta da ferramenta *Integrit* apontou que a mesma pode possuir problemas de implementação.

Considerando as informações apresentadas nesta monografia, pode-se concluir

que o uso da ferramenta *Tripwire* é altamente recomendado em situações que requerem um alto nível de segurança dos arquivos de um sistema. Não sendo recomendado o seu uso em situações em que o controle de integridade de arquivos deva ser realizado com rapidez. As ferramentas *AIDE* e *Osiris* podem ser utilizadas em diversas situações, sendo o seu uso recomendado para situações que não requerem um nível de segurança tão alto. Além disso, ambas as ferramentas (*AIDE* e *Osiris*) são muito rápidas comparadas a ferramenta *Tripwire*. Já o uso da ferramenta *Integrit* não é recomendado, pois segundo o estudo apresentado na seção 4.3, essa ferramenta provavelmente possui problemas de implementação.

Como trabalhos futuros, pretende-se aumentar o número de ferramentas de controle de integridade de arquivos analisadas uma vez que esse tipo de análise fornece informações importantes ao administrador de sistema durante a escolha de uma ferramenta para uma determinada situação. Além disso, pretende-se aumentar o número de critérios adotados na analise das ferramentas, como por exemplo, analisar o nível de dificuldade de edição do arquivo de configuração das ferramentas e o nível de dificuldade de interpretação dos relatórios (gerados pelas ferramentas) com relação ao tamanho dos mesmos.

Referências Bibliográficas

[Anderson & Biham (1996)] Anderson, R. & Biham, E. *Tiger: A Fast New Hash Function*. Proceedings of Fast Software Encryption 3, Cambridge, 1996.

[Anonymous (1999)] Ed. Scott D. Meyers Anonymous. *Maximum Linux Security*. Indianapolis, Indiana, 1999.

[Cashin (2003)] Cashin, Ed L. *Integrit File Verification System Manual*. URL: http://integrit.sourceforge.net/texinfo/integrit.html. Último acesso: 12 de setembro de 2003.

[Dobbertin et al (1996)] Dobbertin, H. et al. *RIPEMD-160, a strengthened version of RIPEMD*. Fast Software Encryption, LNCS 1039, Ed. D. Gollmann, Springer-Verlag, 1996, pp. 71-82. URL: http://www.esat.kuleuven.ac.be/~cosicart/pdf/AB-9601/AB-9601.pdf. Último acesso: 25 de outubro de 2003.

[Lehti (2003)] Lehti, Rami. *The Aide manual*. URL: http://www.cs.tut.fi/~rammer/aide/manual.html. Último acesso: 12 de setembro de 2003.

[Menezes (1996)] Menezes, A. J. et al. *Handbook of Applied Cryptography*. Florida: CRC Press, 1996.

[Nist (1995)] Nist. *Secure Hash Standard*. FIPS PUB 180-1. Washington D. C., 1995. URL: http://www.itl.nist.gov/fipspubs/fip180-1.htm. Último acesso: 25 de outubro de 2003.

[Russel & Gangemi (1991)] Russel, D. & Gangemi, G. T. *Computer Security Basics*. Sebastopol: O'Reilly & Associates, 1991.

[Schneier (1996)] Schneier, B. *Applied Cryptography*. Second Edition. New York: John Wiley & Sons, 1996.

[Serafim (2002)] Serafim, V. S. *Um Verificador Seguro de Integridade de Arquivos*. Porto Alegre, UFRGS, 2002. 97p. (Dissertação de Mestrado).

[The Shmoo Group (2001)] The Shmoo Group. *Osiris 1.5.1 User Documentation*. URL: http://osiris.shmoo.com/. Último acesso: 12 de setembro de 2003.

[Touch (1995)] Touch, J. D. *Performance Analysis of MD5*. URL: http://citeseer.nj.nec.com/touch95performance.html. Último acesso: 25 de outubro de 2003.

[Tripwire, Inc (2000)] Tripwire, Inc. *Home Tripwire.org*. URL: http://www.tripwire.org/. Último acesso: 12 de setembro de 2003.

[Vilela (2001)] Vilela, A. V. *Estudos de técnicas de detecção e prevenção de intrusos*. Monografia de Graduação. URL: http://www.comp.ufla.br/~joukim/extensao/intruso.pdf. Último acesso: 12 de setembro de 2003.

[Zheng et al (1992)] Zheng et al. *HAVAL – A One-way Hashing Algorithm with Variable Length of Output*. AUSCRYPT: Advances in Cryptology – AUSCRYPT '90, International Conference on Cryptology. LNCS, Spring-Verlag, 1992. URL: http://citeseer.nj.nec.com/zheng93haval.html. Último acesso: 25 de outubro de 2003.

Apêndice A

Relatórios de Resposta das Ferramentas de Controle de Integridade de Arquivos

A.1 Relatório de Resposta da Ferramenta *AIDE*

O relatório de resposta da ferramenta *AIDE* é composto por 2 partes:

1. Sumário (*Summary*): apresenta informações gerais, como, por exemplo, se um arquivo foi adicionado (*added*), removido (*removed*) ou alterado (changed);

2. Informação detalhada sobre mudanças (*Detailed information about changes*): descreve o nome do arquivo alterado e as informações que foram alteradas nesse arquivo (apresenta os valores anterior e posterior a verificação).

Um exemplo do relatório gerado pela ferramenta *AIDE* é apresentado na Figura A.1.

A.2 Relatório de Resposta da Ferramenta *Integrit*

O relatório de resposta da ferramenta *Integrit* é composto por um cabeçalho e por informações referentes a cada arquivo analisado. As linhas após o cabeçalho apresentam o estado do arquivo (*changed* = alterado, *new* = novo e *missing* = removido), o nome do arquivo (por exemplo, /etc/group) e as opções usadas na verificação, sendo que, para cada opção é apresentado, quando cabível, os valores obtidos antes e depois da verificação. Ao final do relatório é descrito o valor do *checksum* MD5 da nova base de informações gerada pela ferramenta *Integrit*.

Um exemplo do relatório gerado pela ferramenta *Integrit* é apresentado na Figura A.2.

```
AIDE found differences between database and filesystem!!
Start timestamp: 2003-09-08 10:19:05
Summary:
Total number of files=4566,added files=2,removed files=1,changed files=8

Added files:
added:/etc/ssftp.conf
added:/usr/bin/ssftp
Removed files:
removed:/bin/ps
Changed files:
changed:/etc/group
changed:/etc/passwd
changed:/etc/group-
changed:/etc/passwd-
changed:/etc/shadow-
changed:/etc/gshadow-
changed:/etc/shadow
changed:/usr/sbin/useradd
Detailed information about changes:

File: /etc/group
  Size     : 616                              , 617
  MD5      : vfYsbwAbHkvouAojtpOnpg==         , RtFH2MDU6Br2eZw7srKUSw==

File: /etc/passwd
  Size     : 1578                             , 1609
  MD5      : DqyIX2j9XVg/mQWWfQJrdQ==         , Izgn9Ipiwl1F4SUuuqVfgA==

File: /etc/group-
  Size     : 628                              , 616
  MD5      : QlvG4+Yy3kDtxxM9yM3NOQ==         , vfYsbwAbHkvouAojtpOnpg==

File: /etc/passwd-
  Size     : 1642                             , 1609
  MD5      : cuBndNuh0CrUSYbhsn1eeA==         , Izgn9Ipiwl1F4SUuuqVfgA==

File: /etc/shadow-
  Size     : 1065                             , 1019
  MD5      : piKyGAsietk29bUphXIs8g==         , F/wjSVEn8h7FKui93ZFmsw==

File: /etc/gshadow-
  Size     : 528                              , 526
  MD5      : e2xtMK9egnji4q3rtzlGJw==         , 8E66p+OKgjR9I5lAPhEQTQ==

File: /etc/shadow
  Size     : 995                              , 1030
  MD5      : tyZmpnLTfkfxAYS8LmA6ZA==         , mrIu5xl2JGU93UgRjB7aXg==

File: /usr/sbin/useradd
  Size     : 51992                            , 5
  Permissions: -rwxr-xr-x                       , -rw-r--r--
  MD5      : xLvkk+FNwxp95oZH3Qiz9Q==         , KwAEL3SBx7BWxLQQ0o8zzw==
```

Figura A.1: Exemplo de um relatório de resposta da ferramenta *AIDE*

```
integrit: ---- integrit, version 3.02 -----------------
integrit:                      output : human-readable
integrit:                   conf file : integrit.conf
integrit:                    known db : /var/integrit/integrit_known.cdb
integrit:                  current db : /var/integrit/integrit_current.cdb
integrit:                        root : /
integrit:                    do check : yes
integrit:                   do update : yes
changed: /etc   m(20030908-095342:20030908-101547) c(20030908-095342:20030908-101547)
changed: /etc/sysconfig/redhat-config-users   m(20030908-095521:20030908-101814) c(20030908-095521:20030908-101814)
changed: /etc/group   s(a8d6e6adc80a6b434ded20a860cd3bbde1681ee6:7b2cdff5f47fd1ccb357ca0593846c3b8ca41e1a)
changed: /etc/group   z(616:617) m(20030908-095654:20030908-101749) c(20030908-095654:20030908-101749)
changed: /etc/passwd   s(59eefeb0079db64f9fa1c7ef9f1aeca4868b37b8:015560812e10d487c212302d386a1ebd87884c7f)
changed: /etc/passwd   z(1578:1609) m(20030908-095623:20030908-101749) c(20030908-095623:20030908-101749)
changed: /etc/group-   s(76f7e021784bd9a2b319958c5dd636372468cf29:a8d6e6adc80a6b434ded20a860cd3bbde1681ee6)
changed: /etc/group-   z(628:616) m(20030908-095440:20030908-101749) c(20030908-095440:20030908-101749)
changed: /etc/passwd-   s(af172dc371fd3761d40d4a6f38ef6b8fdc3c3af3:015560812e10d487c212302d386a1ebd87884c7f)
changed: /etc/passwd-   z(1642:1609) m(20030908-095440:20030908-101749) c(20030908-095440:20030908-101749)
new:     /etc/ssftp.conf   p(644) u(0) g(0) z(4) m(20030908-101547)
changed: /etc/shadow-   s(03dfc482ec9408f9ff12bfb722df60839128faf7:bc3231d4899b90da4efd89a21e47ec9881021481)
changed: /etc/shadow-   z(1065:1019) m(20030908-095440:20030908-101749) c(20030908-095440:20030908-101749)
changed: /etc/gshadow-   s(780d54f82990e273e9bbcf317c6ee0491ef94e36:c4bf73e174b99e89e003f9948c612bed4e727da8)
changed: /etc/gshadow-   z(528:526) m(20030908-095440:20030908-101749) c(20030908-095440:20030908-101749)
changed: /etc/shadow   s(8003fe7100d84c179d5bb86946355895b43547aa:5b393e35104f6ab886cb05b68f687f3aa3b3684b)
changed: /etc/shadow   z(995:1030) m(20030908-095720:20030908-101749) c(20030908-095720:20030908-101749)
changed: /etc/gshadow   m(20030908-095440:20030908-101749) c(20030908-095440:20030908-101749)
changed: /bin   m(20030908-095407:20030908-101629) c(20030908-095407:20030908-101629)
integrit: checking for missing files --------------
missing: /bin/ps   p(555) u(0) g(0) z(64652) m(20020812-072027)
missing: /bin/ps   s(2dca12dbce5bc21506ee268805fe63abf486c4ff)
integrit: current-state db md5sum --------------
integrit: 92adde46f7bd2cf2f387de19bd6f4b09  /var/integrit/integrit_current.cdb
```

Figura A.2: Exemplo de um relatório de resposta da ferramenta *Integrit*

A.3 Relatório de Resposta da Ferramenta *Osiris*

O relatório de resposta da ferramenta *Osiris* é composto por um cabeçalho que descreve as principais características das duas bases de dados utilizadas pela ferramenta para realizar a verificação dos arquivos, e por três seções descritas a seguir:

file difference Descreve o nome dos arquivos alterados e as informações que foram alteradas nesses arquivos (apresenta os valores anterior e posterior a verificação);

new files Descreve os arquivos que foram adicionados ao sistema;

missing files Descreve os arquivos que foram removidos do sistema.

Ao final do relatório é apresentado, também, um resumo informando o número de arquivos verificados, alterados, adicionados e removidos.

Um exemplo do relatório gerado pela ferramenta *Osiris* é apresentado na Figura A.3.

A.4 Relatório de Resposta da Ferramenta *Tripwire*

O relatório de resposta da ferramenta *Tripwire* é composto por três partes. Na primeira parte tem-se um cabeçalho contendo informações gerais da ferramenta e do sistema no qual a ferramenta foi utilizada. A segunda parte apresenta um sumário das regras utilizadas na análise, sendo que, para cada regra é descrita o seu nível de segurança (*Severity Level*) e se algum arquivo (ou diretório) foi adicionado (*Added*), removido (*Removed*) ou modificado (*Modified*). Na terceira parte do relatório, para cada regra utilizada na análise, são descritos os nomes dos arquivos (ou diretórios) que foram adicionados (*Added*), removidos (*Removed*) ou modificados (*Modified*). Caso ocorram erros durante o uso da ferramenta, os mesmos são descritos ao final do relatório.

Um exemplo do relatório gerado pela ferramenta *Tripwire* é apresentado na Figura A.4.

```
osiris database comparison
Mon Sep  8 10:23:25 2003


[ database: known.osi ]

  records:       4566
  source:        config file

  created on:    Mon Sep  8 10:13:01 2003
  created by:    root
  created with:  osiris 1.5.2

[ database: current.osi ]

  records:       4567
  source:        config file

  created on:    Mon Sep  8 10:22:36 2003
  created by:    root
  created with:  osiris 1.5.2

[ file differences ]

  /etc/shadow
    md5               b72666a672d37e47f10184bc2e603a64 ==> 9ab22ee7197624653ddd48118c1eda5e
    bytes                                         995 ==> 1030

  /etc/shadow-
    md5               a622b2180b227ad936f5b52985722cf2 ==> 17fc23495127f21ec52ae8bddd9166b3
    bytes                                        1065 ==> 1019

  /etc/group
    md5               bdf62c6f001b1e4be8b80a23b693a7a6 ==> 46d147d8c0d4e81af6799c3bb2b2944b
    bytes                                         616 ==> 617

  /etc/gshadow-
    md5               7b6c6d30af5e8278e2e2adebb7394627 ==> f04ebaa7e38a82347d2399403e11104d
    bytes                                         528 ==> 526

  /usr/sbin/useradd
    md5               c4bbe493e14dc31a7de68647dd08b3f5 ==> 2b00042f7481c7b056c4b410d28f33cf
    permission                           -rwxr-xr-x ==> -rw-r--r--
    bytes                                     51992 ==> 5

  /etc/passwd
    md5               0eac885f68fd5d583f9905967d026b75 ==> 233827f48a62c25d45e1252ebaa55f80
    bytes                                        1578 ==> 1609

  /etc/passwd-
    md5               72e06774dba1d02ad44986e1b27d5e78 ==> 233827f48a62c25d45e1252ebaa55f80
    bytes                                        1642 ==> 1609

  /etc/group-
    md5               425bc6e3e632de40edc7133dc8cdcd39 ==> bdf62c6f001b1e4be8b80a23b693a7a6
    bytes                                         628 ==> 616


[ new files (2) ]

  770:344351  /usr/bin/ssftp
  770:229472  /etc/ssftp.conf

[ missing files (1) ]

  /bin/ps
records compared:    4565
records that differ: 8
new records:         2
missing records:     1

config file used for comparison: osiris.conf
```

Figura A.3: Exemplo de um relatório de resposta da ferramenta *Osiris*

```
Parsing policy file: /etc/tripwire/tw.pol
*** Processing Unix File System ***
Performing integrity check...
Wrote report file: /var/lib/tripwire/report/frajola-20030908-102424.twr


Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Mon Sep  8 10:24:24 2003
Database last updated on:     Never

===============================================================================
Report Summary:
===============================================================================

Host name:                    frajola
Host IP address:              142.106.231.156
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/frajola.twd
Command line used:            /usr/sbin/tripwire --check

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
* diretorio_bin                 0                 0        1        0
  (/bin)
  diretorio_sbin                0                 0        0        0
  (/sbin)
* diretorio_etc                 0                 1        0        7
  (/etc)
* diretorio_usr_bin             0                 1        0        0
  (/usr/bin)
* diretorio_usr_sbin            0                 0        0        1
  (/usr/sbin)

Total objects scanned:  4567
Total violations found:  11

===============================================================================
Object Summary:
===============================================================================

-------------------------------------------------------------------------------
# Section: Unix File System
-------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
Rule Name: diretorio_bin (/bin)
Severity Level: 0
--------------------------------------------------------------------------------

Removed:
"/bin/ps"

--------------------------------------------------------------------------------
Rule Name: diretorio_etc (/etc)
Severity Level: 0
--------------------------------------------------------------------------------

Added:
"/etc/ssftp.conf"

Modified:
"/etc/group"
"/etc/group-"
"/etc/gshadow-"
"/etc/passwd"
"/etc/passwd-"
"/etc/shadow"
"/etc/shadow-"

--------------------------------------------------------------------------------
Rule Name: diretorio_usr_bin (/usr/bin)
Severity Level: 0
--------------------------------------------------------------------------------

Added:
"/usr/bin/ssftp"

--------------------------------------------------------------------------------
Rule Name: diretorio_usr_sbin (/usr/sbin)
Severity Level: 0
--------------------------------------------------------------------------------

Modified:
"/usr/sbin/useradd"

================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
Integrity check complete.
```

Figura A.4: Exemplo de um relatório de resposta da ferramenta *Tripwire*