Ataque, defesa e contra-ataque:

Evasão

Capítulo 15

"Em dia de vitória, ninguém fica cansado"
Provérbio árabe

Não importa se o objetivo do ataque seja causar rebuliço ou obter secretamente algum proveito: qualquer atacante que se preze não deseja ser rastreado e muito menos apanhado. Alguns detalhes devem ser observados para que, depois de uma invasão do tipo "obra de arte", a determinação do culpado seja impossível ou muito difícil.

No capítulo anterior, quando falamos em rootkits, vimos que eles podem esconder um sem-número de detalhes e, assim, mascarar a presença e a ação do hacker no computador atacado. Entretanto, seguro, morreu de velho e, com ou sem rootkits, é altamente desejável que o invasor tome providências para apagar todos os rastros e vestígios de sua passagem pelo sistema. Em várias plataformas há ferramentas que o fazem automaticamente, mas entender os mecanismos de registro é muito importante para conferir a eficácia dessas mesmas ferramentas ou para fazer tudo "na mão", na falta delas.

Outro detalhe para o qual o invasor deve estar atento é a camuflagem dos canais de comuicação entre ele e a máquina invadida. Como vimos no capítulo anterior, há maneiras de tornar invisível o fluxo de dados de um backdoor ou cavalo de tróia, simplesmente escondendo os bytes sendo transmitidos/recebidos em conexões válidas. Neste capítulo, veremos mais algumas maneiras de fazer essas conexões invisíveis.

Antes de falarmos em evasão, lembre-se de que o disfarce anterior ao ataque é importante. Se você pretende invadir alguém, lembre-se de usar algum tipo de spoofing antes, como, por exemplo, cadeias de proxies públicos ou máquinas-laranja rodando redirecionadores Netcat. Usar seu próprio IP para mirar em alguém é burrice.

O básico: apagando os logs

Mesmo kiddies muito ruins sabem que devem apagar seus rastros nos registros do sistema. A primeira coisa que se tem de saber é, então, **onde** estão tais registros. Uma vez com acesso a eles, deve-se ter sempre em mente que **não se pode apagá-los todos**, caso contrário serão levantadas muitas suspeitas sobre o sistema. Apague apenas o que houver sido causado pelas suas andanças em terreno alheio. E cuidado: alguns sistemas possuem IDSs, outros verificadores de integridade de arquivos, e alguns, ainda, possuem logs em locais não-padrão, deixando logs falsos no local padrão para enganar invasores de meia-tigela.

Lembre-se: verifique linha por linha, um arquivo por vez, todos os arquivos de log. Procure por cópias deles em locais não óbvios do sistema e, se notar que está em um *honeypot*[1], fuja!

1. Como vimos em capítulos anteriores, honeypots *são sistemas intencionalmente deixados como "boi de piranha" para que os wannabe hackers ataquem. Normalmente, são sistemas fracamente protegidos, com falhas intencionais e sistemas silenciosos de detecção de intrusos. Muitos IDSs inclusive desviam silenciosamente o intruso para os honeypots sem que eles percebam. Além da marcação cerrada, quando a presa é aparentemente fácil, o invasor torna-se descuidado.*

Encaremos os fatos: mesmo tendo padrões, protocolos e normas em comum, os Unices são diferentes entre si. Cada sabor possui um tipo distinto de sistema de arquivos, hierarquia de diretórios, codificação de caracteres, sintaxe do shell nativo, conjunto de comandos padrão... Há inclusive comandos que, mesmo possuindo o mesmo nome em todas as variedades, fabricantes e versões Unix, possuem sintaxe diferente – ***ps***, ***route*** e, mesmo, o ***ls*** são dois exemplos clássicos. O sistema de registro de eventos não seria, portanto, imune a essas diferenças.

É impossível listar em um livro como este todos os tipos e particularidades de logs nas diversas versões dos sabores. Utilizaremos, então, o sistema Linux como modelo para exemplificar uma sessão de "cirurgia" nos logs. Se algum script kiddie estiver lendo este capítulo, provavelmente pulou este parágrafo e vai tomar a descrição abaixo como receita universal para apagamento de seus rastros. Nossa intenção, entretanto, é outra: mostrar o que você precisa procurar, não onde, e nem como...

A primeira coisa que deve ser verificada é o histórico de comandos do usuário. Não é exatamente um log; antes disso, é uma lista de comandos já emitidos que fica à disposição do usuário para que não precisem ser digitados novamente. Isso evita trabalho repetitivo para o operador – e, como efeito colateral, dá ao administrador uma forma de saber o que o usuário anda fazendo...

Nem todos os shells implementam um histórico de comandos, e cada um guarda o histórico em um arquivo e local diferente. Apenas a título de exemplo, um sistema Linux normalmente usa a versão GNU do Bourne Shell, o Bash. Esse interpretador de comandos guarda o histórico de tudo o que foi digitado (inclusive sem sucesso) no arquivo /home/usuário/.bash_history. É um arquivo de texto puro, portanto, bastaria editá-lo para esconder seus rastros. Uma maneira mais efetiva, entretanto, seria simplesmente desativar a variável de sistema HISTFILE, que indica onde os comandos devem ser armazenados. Para tanto, basta emitir **unset HISTFILE**. Pronto! A partir daí, nada mais será registrado (nem o próprio unset!) Esse deveria ser o primeiro comando a ser feito, mas a maioria dos "invasores" esquece... ou não sabe...

Em Unices que não usem Bash, uma outra maneira é, simplesmente... trocar de shell! É possível (e muito provável) que o shell padrão possua histórico, e os demais não. Portanto, se você conseguiu uma conta e o shell padrão é o Bourne Shell (prompt $) simplesmente mude-o para o C Shell digitando **csh**. Se o shell padrão é justamente o C Shell (prompt %), mude para o Bourne Shell, digitando **sh**. A razão para usar sh e csh é que qualquer um dos outros (bash, zsh, ksh) têm rotinas de histórico de comandos completamente implementadas.

Para ajudar a desviar suspeitas, um atacante mais ousado poderia desviar os comandos (ou copiá-los) do seu próprio histórico para o de outro usuário. Mesmo que o sistema esteja sob suspeita, durante um tempo razoável os santos pagarão pelos pecadores.

Depois de enganar o histórico do shell, temos de apagar os rastros nos logs do sistema. Na maioria dos Unix, o *daemon* responsável pelo registro dos logs de sistema é o ***syslogd***. Em qualquer Unix invadido, portanto, é interessante pesquisar nos arquivos de configuração do daemon (que, **no Linux**, fica em ***/etc/syslog.conf*** – mas isso varia em outros Unices) e verificar **quais** os nomes dos arquivos de registro de eventos e **onde** estão gravados.

Como usuário comum (não root) em um Conectiva Linux 9, o arquivo /etc/syslog.conf mostra:

```
  $ cat /etc/syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none                          /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                                 *

# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

Obviamente, cada um dos arquivos indicados no syslogd possui uma função diferente.Todas as linhas marcadas com um # são consideradas como comentários e ignoradas. O campo da esquerda define uma série de **regras de registro** a serem aplicadas às mensagens. Os campos da direita indicam em quais arquivos os registros têm de ser gravados. Observe que, no syslog.conf acima, o filtro **kern.*** está direcionado para /dev/console – ou seja, as mensagens do kernel seriam ecoadas no terminal, caso a linha estivesse descomentada. Como se pode ver, dispositivos também podem ser usados para *logging*, além dos arquivos comuns.

Pelo que podemos observar do syslog.conf, temos, em /var/log,

- **/var/log/messages**: registra todas as mensagens de nível informativo do sistema.
- **/var/log/secure**: registra acesso a arquivos e processos restritos.
- **/var/log/maillog**: registra mensagens de e-mail enviadas e recebidas.
- **/var/log/spooler**: registra erros em trocas de Mail, UUCP e News.
- **/var/log/boot.log**: registra eventos e erros durante o boot.

Novamente lembrando: **essas localizações são para o Conectiva Linux 9!!!** Procure no próprio Unix invadido quais as localizações reais dos arquivos. É possível que o número e a função deles seja diferente: pode haver um arquivo apenas com todos os logs, ou podem haver, por exemplo, um arquivo separado para UUCP e outro para e-mail. Script kiddies normalmente usam ferramentas e scripts de apagamento de logs de um sistema em outro e, em vez de se esconderem, acabam criando mais mensagens de erro nos logs e alertanto o administrador mais cedo do que o esperado. Olhe antes de agir e faça a coisa certa.

Uma vez descobertos quais os arquivos responsáveis pelo registro de eventos do Unix sob ataque, podemos passar à edição deles. Apesar de existirem ferramentas que aplicam criptografia forte nos arquivos de log, a grande maioria dos sistemas ainda usam o bom e velho texto puro para gravá-los. O que significa que o hacker pode usa seu editor de textos preferido (emacs, vi, joe, pico...) para editá-lo manualmente. Os autores recomendam **sempre** editar manualmente os arquivos, evitando que sujeira, erros ou imprecisões nos scripts que o fazem automaticamente possam colocar tudo a perder.

Ainda no CL9, vamos ao diretório /var/log e vejamos o que há por lá:

```
$ ls
XFree86.0.log      boot.log.4  dmesg           maillog.2  messages.4      netconf.log.3     secure.4   vtund
XFree86.0.log.old  cron        htmlaccess.log  maillog.3  mysql.log       samba             spooler    wtmp
apache             cron.1      iptraf          maillog.4  mysql.log.1.gz  scrollkeeper.log  spooler.1  wtmp.1
boot.log           cron.2      kdm.log         messages   nagios          secure            spooler.2
boot.log.1         cron.3      lastlog         messages.1 netconf.log     secure.1          spooler.3
boot.log.2         cron.4      maillog         messages.2 netconf.log.1   secure.2          spooler.4
boot.log.3         cups        maillog.1       messages.3 netconf.log.2   secure.3          uucp
```

Observe: além dos arquivos do syslogd, o diretório /var/log contém os registros de outros programas e servidores que rodam na máquina, como Apache, Samba, CUPS e o XFree86. Convém verificar os logs de todos eles, uma vez que alguma ação sua (intencional ou não) pode ter deixado marcas em qualquer deles.

Ocupemos-nos primeiro dos cinco arquivos do syslogd. Ainda como usuário comum, abra (com seu editor de textos favorito) o arquivo **boot.log**. Nele estão os serviços sendo executados, os módulos do kernel carregados e os servidores que são iniciados e finalizados pelo inetd. Note que há outros, chamados **boot.log.1**, **boot.log.2**, etc, que guardam logs antigos. Agora tente escrever qualquer coisa e salvar o arquivo (sempre como usuário comum). Permissão negada! Os outros arquivos – **maillog**, **messages**, **secure** e **spooler** – sequer dão permissão de leitura a usuários comuns.

Logue-se como root (ou, em um sistema invadido, consiga acesso a root usando buffer overflow ou outra técnica qualquer) e abra os arquivos. Verifique as sintaxes deles. **Maillog** registra a atividade de SMTP, IMAP e POP locais (sendmail, postfix, qmail, imapd, pop3d...). **Messages** é muito importante: registra as mensagens entre processos e entre eles e o kernel. **Secure**, por outro lado, registra mensagens de acesso privilegiado a arquivos e processos. Por sua vez, **spooler** guarda as mensagens oriundas de programas que usam os serviços de spool (mail, uucp, news, etc). Mesmo que você não tenha mexido com o servidor de SMTP ou POP, é prudente verificar também os arquivos /var/log/maillog e /var/log/spool e ter certeza de que, inadvertidamente, você não disparou algum processo que tenha alterado o log. O mesmo vale para **qualquer** processo ou arquivo, portanto um cuidado geral no log é primordial.

Voltando ao /etc/syslogd.conf, a linha

```
# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                 *
```

indica que quaisquer mensagens de emergência do sistema serão ecoadas a **todos** os logs e a **todos** os usuários. Cuidado com isso: se o sistema detectar coisas estranhas acontecendo, é possível que todos os usuários sejam avisados – portanto, uma boa idéia é parar o serviço syslogd ou reiniciá-lo com essa linha comentada. Consulte as páginas de manual do **syslogd** e do **syslog.conf** no Unix afetado para detalhes.

Mas, além do que é monitorado pelo syslogd, há outros arquivos de real importância. Um deles é o sistema **utmp/wtmp/lastlog**. O utmp é um subsistema que registra quem está conectado no presente momento. Quando o usuário (ou o administrador) emite um comando **who**, o comando vai ler em **/var/run/utmp** as informações sobre todas as contas em uso, em quais terminais elas estão e, se for uma conexão remota, o IP de origem da conexão. Por exemplo, no momento em que estas linhas estão sendo escritas, o comando **who** informa:

```
henrique pts/0     Feb 16 14:42
henrique pts/1     Feb 16 16:07
```

Entretanto, eu posso dizer ao comando **who** em qual arquivo olhar. Se eu emitir o comando **who /var/run/utmp**, o resultado será o mesmo: who sem argumentos sempre lê o arquivo utmp. Mas e se eu pedir para who consultar em **/var/log/wtmp**?

```
henrique     pts/3        Jan  9 05:14 (192.168.1.229)
henrique     pts/3        Jan  9 05:16 (192.168.1.229)
james        pts/0        Fev  2 11:50
root         tty1         Fev  5 22:16
henrique     tty2         Fev  5 22:17
henrique     tty3         Fev  5 22:23
root         tty4         Fev  5 22:40
root         pts/0        Fev  5 22:49
root         pts/1        Fev  5 22:52
```


O arquivo é muito grande, então mostramos apenas algumas linhas. Observe: No dia nove de janeiro, o usuário Henrique fez duas conexões a esta máquina (que, a título de curiosidade, era a 192.168.1.11) a partir de outra, a 192.168.1.229. No dia dois de fevereiro, o usuário James logou-se localmente na máquina, e no dia cinco o usuário Henrique logou-se como root várias vezes.

O comando **last** faz um cruzamento de informações entre os arquivos **/var/log/lastlog** e **/var/log/wtmp** e mostra na tela as informações detalhadas dos últimos dias sobre os tempos de login e logout de cada usuário e mesmo de reboot da máquina. O comando **lastlog**, por outro lado, mostra quando cada um dos usuários do sistema logou-se pela última vez. Um usuário desativado que tenha dado login **ontem** é um evento alarmante.

Para o administrador de sistemas, usar os comandos last, lastlog e who para verificar o /var/log/wtmp é uma medida tanto preventiva como corretiva. Portanto, é imperioso para o cracker que quer apagar seus vestígios exclui todas as menções a ele nesses arquivos. Então nosso intrépido invasor usa o pico (...) para editar, por exemplo, o /var/log/wtmp e tem uma desagradável surpresa: o arquivo não é de texto puro, mas binário! O mesmo ocorre com /var/run/utmp e com /var/log/lastlog. Então, o que fazer? Desespero...

Conforme dissemos antes, há ferramentas automáticas para limpeza do utmp, wtmp e lastlog. Uma delas (entre muitas) é o **Hide** (www.hoobie.net/security/exploits/hacking/hide.c). Este pequeno programa apaga as entradas do usuário que o executou no arquivo utmp, caso esteja liberado para escrita para aquele usuário. Obviamente sistemas modernos não cometem essa tolice e deixam o utmp liberado apenas para root, mas é uma excelente prova de conceito.

O código-fonte do programa, sendo pequeno, é fácil de entender. Obviamente a esta altura o leitor já deve saber que terá de compilar o programinha para que ele funcione. Dependendo do sistema, podem ser necessárias algumas alterações. No Conectiva Linux 9, por exemplo, bastou substituir todas as funções exit( ) por exit(0). Nota-se aí que o candidato a hacker deve **obrigatoriamente** aprender a programar (ou pelo menos, a "se virar") em C e C++. Se você **não quer** aprender a programar, bem, jogue este livro fora: você não quer, definitivamente, elevar seu nível técnico, mas apenas aprender receitinhas de ataque simples. Hackers riem de vocês, enquanto isso.

No mesmo site encontramos outra ferramenta muito famosa, desenvolvida pelo não menos famoso hacker Simple Nomad, chamada (puxa...) **Remove** (www.hoobie.net/security/exploits/hacking/remove.c). Com ela, é possível remover quaisquer usuários de qualquer um dos três arquivos utmp, wtmp e lastlog. Foi desenvolvida para AIX, mas pode ser facilmente compilada (como está ou com modificações simples) em qualquer sabor de Unix, incluindo Linux e {Free,Open,Net}BSD. Além de remover os registros referentes a qualquer usuário (e não apenas ao que executar a ferramenta, como o Hide), o Remove permite que se altere o último usuário que fez login e o local de login (IP, se remoto; tty ou pst, se local).

Compile e teste os dois programas em sua máquina de testes. Faça várias experiências, como aumentar e diminuir o nível de permissões dos seus utmp, wtmp e lastlog e rodar as duas ferramentas. Note que, num ataque real, o invasor normalmente precisa de um shell root para rodar o Remove, embora não precise, necessariamente, de tantos privilégios para atacar o sistema. Depois de brincar com esses, procure por outros tipos de ferramentas. Nomes muito comuns são Cloak, Cloak2 (ou Cloak-2) Zap e LogWEdit. Muitos rootkits possuem ferramentas (embutidas ou não) para edição desses arquivos. Veja, por exemplo, a documentação do Knark e do lrk5. Mas lembre-se: nunca use a peça redonda no buraco quadrado! Pesquise sobre a ferramenta correta para o Unix que está sendo atacado.

Caso nenhuma dessas ferramentas esteja disponível no momento, alguns truques podem ajudá-lo a permanecer escondido por algum tempo. Por exemplo, quando se acessa o sistema por uma conta "hackeada", provavelmente o acesso vai ficar gravado no arquivo lastlog, com o IP de conexão ou o nome de domínio correspondente. Para apagar isso (ao menos no lastlog), uma vez feito o login com sucesso, rode o comando **rlogin** na mesma conta. Pronto! Agora, o último login desta conta terá sido local, e no arquivo lastlog aparecerá "from localhost".

Falta enganar o comando who. Uma vez logado no sistema (e depois de enganar o lastlog...), use o comando **login** e forneça, novamente, o usuário e senha desta conta. Dependendo do Unix (e de quanto ele está atualizado...), isso esconderá a origem da conexão, fazendo o comando who pensar que alguém está conectado localmente.

Para terminar, procure pelos logs específicos dos servidores e aplicativos que rodam no computador invadido. Além dos logs de sistema, procure pelos do Apache ou do servidor de FTP. Cada sistema e sabor Unix possuem daemons de versões e procedências diferentes e com um procedimento diverso de logging. Não vamos discorrer sobre todas as possibilidades aqui, pois tomaria o espaço de vários livros como este; mas é imperativo que o leitor pesquise sobre isso no ambiente invadido. É justamente por isso que a observação é importante, antes de fazer qualquer coisa.

Como tarefa para casa, experimente verificar os logs do Apache e do Squid em sua máquina de testes. Experimente simplesmente conectar-se a ela a partir de outra maquina como um usuário regular (por exemplo, faça uma conexão SSH a ela, use-a como proxy com o Squid ou acesse uma página HTML de testes). Depois, siga todos os passos de invasão vistos nos capítulos anteriores, desde rodar um nmap contra a máquina até realmente conseguir root por algum método e alterar alguma coisa. Observe atentamente os arquivos de log e veja o que acontece.

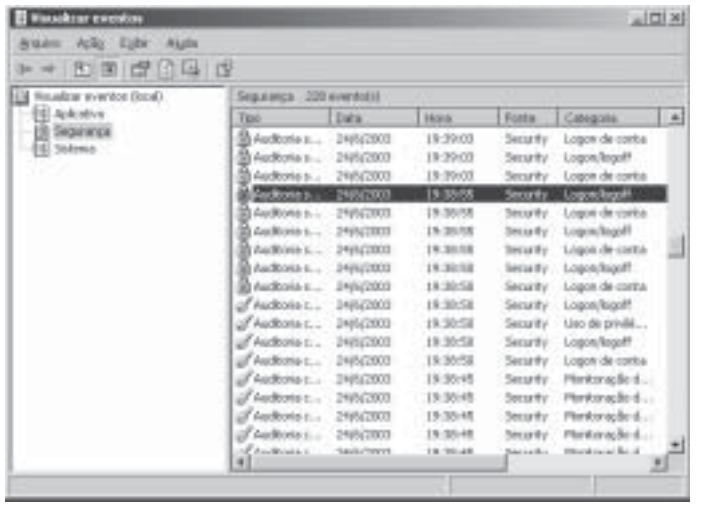
Última dica: www.hoobie.net.

Registro de eventos no Windows NT/2k/XP

Como vimos no capítulo sobre plataformas Windows, todo o funcionamento do Windows é baseado em eventos. Portanto, nada mais lógico que seus logs também registrem os diversos eventos ocorridos no sistema. Um serviço especial chamado EventLog dá conta do recado. E os problemas já começam por ele mesmo: os logs só são criados se as rotinas de **auditoria** do Windows estiverem ativadas. Como nem todos os administradores o fazem, há aí um grande furo de segurança aproveitado por crackers.

Os eventos são registrados em duas etapas. Em primeiro lugar, são armazenados em arquivos temporários chamados APPLICATION.LOG, SYSTEM.LOG e SECURITY.LOG (no Windows XP, esses três arquivos tornaram dezenas deles em C:\WINDOWS ou C:\WINNT). Depois de algum tempo, tais dados são guardados em três arquivos definitivos: SecEvent.Evt, SysEvent.Evt e AppEvent.Evt, todos guardados em C:\WINDOWS\SYSTEM32. São arquivos binários, assim como o utmp do Unix, e portanto necessitam de aplicativos específicos para serem editados. Cada um deles, respectivamente, registra um conjunto diferente de eventos: segurança (tentativas de login fracassadas ou não, acesso a arquivos não autorizados, etc), funcionamento do sistema (inicialização, terminação e falhas em serviços e drivers) e funcionamento de aplicativos (inicialização, terminação e falhas em programas do *user space*).

Da mesma forma como no registro, que pode ser consultado e alterado com o uso do RegEdit, os eventos do sistema podem ser visualizados (mas **não** alterados) com o Event Viewer. Para acessá-lo, clique em Iniciar/Executar e rode o comando **eventvwr**. A tela apresentada será parecida com esta:



Observe: forçamos alguns acessos indesejados com um usuário não-privilegiado e depois analisamos os logs em um Windows XP .

Para alterar os logs, deve-se usar outras ferramentas, uma vez que o Event Viewer é apenas um visualizador. Há algumas disponíveis em sites de hackers, mas é preciso fazer alguma pesquisa para tal. O site NT Security (ntsecurity.nu) oferece uma ferramenta chamada WinZapper, que permite alterar os logs de servidores Windows NT e 2000.

Escondendo suas conexões

Já discutimos anteriormente sobre como tornar suas conexões indetectáveis. Começando com os rootkits, que criam um ambiente artificial e "mentem" sobre as comunicações existentes, passamos por métodos de embutir as conexões em outras e acabamos por usar técnicas que permitem, inclusive, comunicações sem conexão. Vamos apenas relembrar algumas das técnicas vistas, reunidas aqui como uma espécie de "revisão".

Para começar, existem as técnicas de Layer-0 vistas no capítulo anterior e que usam sniffing+spoofing para receber comandos na máquina vítima. Adicionalmente, usam o "estofo" e campos inúteis nos cabeçalhos TCP/IP para enviar dados ao hacker a partir do computador invadido. O hacker tem de se posicionar no meio do fluxo de dados para poder "esnifar" os pacotes sem perder nada, uma vez que não são destinados formalmente a ele.

Há dois capítulos, vimos como usar o Netcat para "empurrar" um shell com uma conexão vinda de dentro para fora. Dessa forma, o administrador, mesmo que verifique a conexão com netstat ou outra ferramenta, verá uma inocente conexão originada de dentro da rede. Há ferramentas que vão mais longe e mascaram essa conexão como se fosse uma requisição HTTP, POP, FTP ou mesmo SMTP vinda de dentro. Na máquina do hacker, há um programa que simula o servidor desejado (HTTP, por exemplo) e manda, travestidos de resposta HTTP, os comandos a serem executados na vítima. Usar HTTP é realmente eficaz e imoral, pois a porta 80 estará sempre aberta para que usuários internos façam conexões à Web.

Há ainda duas maneiras adicionais (entre muitas outras) que deixamos como trabalho para casa. É possível, em máquinas Unix, "ricochetear" um X-terminal para a máquina do hacker, caso o servidor X esteja rodando e sua porta (normalmente a 6000) esteja aberta e não filtrada. E também é possível trocar informações entre a máquina do hacker e a vítima (seja Windows, Unix, Macintosh, Novell...) usando tráfego ICMP. Dica: Loki.

A maneira de esconder tráfego depende de cada caso. Há redes em que a comunicação via ICMP seria impraticável, pois o firewall bloqueia esse tipo de tráfego. Mas é importante que o hacker que queira manter um canal seguro de comunicação com os backdoors de suas presas implemente algum tipo de camuflagem. Hoje em dia, os administradores não são tão despreocupados quanto a tráfego suspeito como há quatro ou cinco anos.

Uma última dica: rode seus backdoors com nomes do sistema, de forma a aparecerem como processos comuns. Por exemplo, no Windows um servidor B02K pode ser renomeado para explorer.exe. No Unix, poderia chamar-se inetd ou lpd.

Defesa e Contra-ataque

Algumas medidas podem ser tomadas para impedir ou, pelo menos, dificultar o trabalho de camuflagem dos hackers maliciosos.

Apagamento de logs

Em primeiro lugar, certifique-se periodicamente de que a auditoria do sistema esteja realmente rodando, e os logs sendo criados. Em máquinas Unix, verifique o syslogd; no Windows, o EventLog. Há hackers que, em vez de editar os logs, desligam a auditoria e deixam registros falsos e que não mudam nunca. Administradores "desligados" e alguns verificadores de integridade de arquivos são enganados com essa técnica simples e grosseira.

Em segundo lugar, verifique se quem tem permissão de escrita ou mesmo leitura dos logs realmente deveria possuí-la. Apesar de óbvio, este cuidado geralmente é deixado de lado mesmo por administradores experientes. Utilize a política da máxima restrição possível. Nesse caso, sempre, menos é mais.

Aplicar criptografia é a terceira providência óbvia. Ao contrário das duas anteriores, normalmente os logs não são criptografados por padrão, com ferramentas do próprio sistema. Em vez disso, são necessários programas de terceiros para a tarefa. Cada sistema operacional possui diversas possibilidades. Contate seu fornecedor para obter mais informações.

Outra maneira muito eficaz (mas não tão óbvia) de proteger os logs é gravá-los em mídias apenas de leitura. CD-Rs são especialmente úteis nessa hora. É possível que haja problemas de desempenho no acesso a disco, mas os drives mais modernos já possuem velocidade aceitável para a tarefa. O fato é que os hackers não conseguirão alterar o log porque isso não é fisicamente possível.

Preparar um servidor especial para armazenar os registros de eventos de todos os outros computadores também é um artifício interessante. Isso cria diversas camadas de dificuldade para o invasor chegar até, ou mesmo saber onde estão, os verdadeiros arquivos de registro. Novamente, contate seu fornecedor para conhecer as opções de servidores de logging disponíveis para seu sistema operacional.

Um último truque, que não bloqueia, mas retarda o invasor mais habilidoso, é colocar os logs em locais fora do padrão e deixar, nos locais-padrão, simulacros. Em se tratando de kiddies, esse expediente simples criará a impressão falsa de que "limparam a barra", quando na verdade continuam sendo monitorados.

Camuflagem de conexões

Para começar, e em consonância com o que vimos nos capítulos anteriores, o invasor não deve conseguir, de forma alguma, acesso privilegiado ao sistema.

Root ou administrador, esse acesso deve ser evitado ao máximo. Portanto, aplique as correções do seu fabricante, mantenha senhas difíceis, desative serviços não usados, blá blá blá... A velha fórmula de sempre, que deve ser aplicada a qualquer situação.

Conheça o que você está rodando. Saiba os processos que podem e os que não deveriam aparecer na lista de tarefas, bem como as conexões que deveriam (ou não) estar estabelecidas. Verifique-os periodicamente, mais de uma vez por dia em casos de suspeita de ataque e mais de uma vez por hora caso o ataque tenha sido detectado.

Uma última dica: o Snort consegue detectar canais de comunicação baseados em ICMP. Snort x Loki... jogo interessante.

Uma despedida?[2]

Bem, chegamos ao final de nossos estudos formais. É emocionante vê-lo formado e diplomado. Mas lembre-se que sua educação sobre hackerismo não termina com o último capítulo: veja o que preparamos para você no CD! Além disso, faça todas as experiências do livro e visite todos os sites indicados. E também teste todas as ferramentas sugeridas. Você verá que são muito poucas, se comparado ao que há disponível. Vá atrás! Aprender a ser "hacker" (ou como quer que você chame) é um processo contínuo.

Despedimo-nos aqui (lágrimas nos olhos!) e esperamos que possamos nos encontrar em breve, em outro curso, talvez.

Mas mantenha contato com seus velhos professores! Mande-nos suas sugestões, críticas e colaborações para este livro. Todos estão convidados a participar.

Escreva mesmo!

Nosso endereço: univh4ck3r@digerati.com.br.

E não esqueça de acessar nosso site em www.digerati.com.br/livro

Um grande abraço, e até mais. **Happy Hacking**!

2 É digno de nota que este livro chegou até aqui sem citar nenhuma vez **A arte da guerra: Sun Tzu** *. Não há publicação moderna que não se aproveite desse venerável exemplo da literatura clássica. Os autores procuraram fugir do lugar-comum, e portanto decidiram (contrariando sugestões disparadas de todos os lados) não incluir nenhum trecho do milenar compêndio. Entretanto, o livro é muito bom e vale a pena ser lido, de graça e on-line, em www.kimsoft.com/polwar.htm.*