GIAC CERTIFICATIONS

Documento de Certificação Global de Garantia de Informação

Interessado em saber mais?

Consulte a lista de próximas ofertas de eventos "Security Essentials" (Princípios Básicos de Segurança): Network, Endpoint, and Cloud (Security 401)" em http://www.giac.org/registration/gsec

Kevin Mitnick, Hacking

Ninguém transformou Kevin Mitnick num hacker; essa foi a sua própria escolha e responsabilidade. Criado no Vale de San Fernando perto de Los Angeles pela sua mãe, Mitnick tem estado dentro e fora de problemas com a lei desde 1981. Foi então, quando tinha 17 anos, que foi colocado em liberdade condicional por roubar manuais de computador de um centro de comutação telefónica da Pacific Bell em Los Angeles. Aqueles que conhecem Mitnick pintam um quadro de um homem obcecado com o poder inerente ao controlo das redes informáticas e telefónicas da nação. Quem é este demónio anti-social dos computadores? Quem poderá exorcizar isto demónio? Que tal um jovem físico, nascido no Japão, especialista em segurança informática, um feiticeiro como Tsutomu Shimomura?

No dia de Natal de 1994, um hacker lançou um sofisticado ataque de IP Spoofing contra os computadores de Tsutomu Shimomura em San Diego. O ataque foi lançado do toad.com em San Francisco, o computador Toad Hall propriedade de John Gilmore, um empregado fundador da Sun Microsystems. Por uma estranha coincidência, Shimomura passou o dia no Toad Hall com a sua amiga, Julia Menapace. A perseguição de Shimomura ao hacker levou a computadores no condado de Marin, onde os ficheiros roubados de Shimomura foram encontrados em The Well, Denver, San Jose e finalmente a Kevin Mitnick, o hacker fugitivo, em Raleigh, Carolina do Norte. Foram utilizados dois mecanismos de ataque diferentes. A falsificação do endereço IP e a previsão do número de sequência TCP foram utilizadas para obter acesso inicial a uma estação de trabalho sem disco, sendo utilizadas principalmente como um terminal X. Após ter sido obtido o acesso de raiz, uma ligação existente a outro sistema foi desviada através de um módulo STREAMS de kernel carregável.
Estas são algumas das máquinas que figuravam no hacking de Mitnick. Para aqueles que não sabem, Telnet é um processo UNIX que permite o login remoto a um computador em rede, e é uma das principais ferramentas que um hacker utiliza na Rede: http é o protocolo para servir páginas da World Wide Web.

O ataque de spoofing IP começou por volta das 14:09:32 PST em 25/12/94. As primeiras sondas foram de toad.com (esta informação derivou de registos de pacotes):

```
14:09:32 toad.com# dedo -l @target 14:10:21
toad.com# dedo -l @server 14:10:50
toad.com# dedo -l root@server 14:11:07
toad.com# dedo -l @x-terminal 14:11:38
toad.com# showmount -e x-terminal 14:11:49
toad.com# rpcinfo -p x-terminal 14:12:05
toad.com# dedo -l root@x-terminal
```

O objectivo aparente destas sondas era determinar se poderia haver algum tipo de relação de confiança entre estes sistemas que pudesse ser explorada com um ataque de IP spoofing. Os números das portas de origem para o showmount e rpcinfo indicam que o atacante está enraizado no toad.com. Cerca de seis minutos depois, vemos um fluxo de

TCP SYNs (pedidos iniciais de ligação) de 130.92.6.97 para a porta 513 (login) no servidor. O objectivo destes

SYNs é preencher a fila de ligação para a porta 513 no servidor com ligações "semi-abertas" para que não responda a quaisquer novos pedidos de ligação. Em particular, não irá gerar TCP RSTs em resposta a SYN-ACKs inesperados.

Como a porta 513 é também uma porta "privilegiada" (< IPPORT_RESERVED), server.login pode agora ser utilizado com segurança como fonte putativa para um ataque de falsificação de endereços no UNIX "r-services" (rsh, rlogin). 130.92.6.97 parece ser um endereço aleatório (forjado) não utilizado (um que não irá gerar qualquer resposta aos pacotes que lhe são enviados):

14:18:22.516699 130.92.6.97.600 > server.login: S 1382726960:1382726960(0) vencer 4096
14:18:22.566069 130.92.6.97.601 > server.login: S 1382726961:1382726961(0) vencer 4096
14:18:22.744477 130.92.6.97.602 > server.login: S 1382726962:1382726962(0) vencer 4096
14:18:22.830111 130.92.6.97.603 > server.login: S 1382726963:1382726963(0) vencer 4096
14:18:22.886128 130.92.6.97.604 > server.login: S 1382726964:1382726964(0) vencer 4096
14:18:22.943514 130.92.6.97.605 > server.login: S 1382726965:1382726965(0) vencer 4096
14:18:23.002715 130.92.6.97.606 > server.login: S 1382726966:1382726966(0) vencer 4096
14:18:23.103275 130.92.6.97.607 > server.login: S 1382726967:1382726967(0) vencer 4096

O servidor gerou SYN-ACKs para os primeiros oito pedidos SYN antes da fila de ligação ser preenchida. O servidor retransmitirá periodicamente estes SYN-ACKs uma vez que não há nada para ACK-los. Abaixo, vemos 20 tentativas de ligação, apenas as primeiras nove são mostradas, desde apollo.it.luc.edu até x-terminal.shell. O objectivo destas tentativas é determinar o comportamento do gerador de números de sequência TCP do x-terminal. Note-se que a sequência inicial
números incrementais de um para cada ligação, indicando que os pacotes SYN estão *não* a ser gerados pela implementação TCP do sistema. Isto resulta na geração conveniente de RSTs em resposta a cada SYN-ACK inesperado, pelo que a fila de ligação em
x-terminal não se enche:

14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S 1382726990:1382726990(0) win 4096
14:18:26.094731 x-terminal.shell > apollo.it.luc.edu.1000: S 2021824000:2021824000(0) ack 1382726991 vitória 4096
14:18:26.172394 apollo.it.luc.edu.1000 > x-terminal.shell: R 382726991:1382726991(0) vencer 0
14:18:26.507560 apollo.it.luc.edu.999 > x-terminal.shell: S 1382726991:1382726991(0) ganhar 4096
14:18:26.694691 x-terminal.shell > apollo.it.luc.edu.999: S 2021952000:2021952000(0) ack 1382726992 vencer 4096
14:18:26.775037 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)

ganhar 0
14:18:26.775395 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)
ganhar 0
14:18:27.014050 apollo.it.luc.edu.998 > x-terminal.shell: S 1382726992:1382726992(0)
ganhar 4096
14:18:27.174846 x-terminal.shell > apollo.it.luc.edu.998: S 2022080000:2022080000(0)
ack 1382726993 vitória 4096

Note-se que cada pacote SYN-ACK enviado por x-terminal tem um número de sequência inicial que é 128.000 maior do que o anterior. Abaixo, vemos um SYN (pedido de ligação) forjado, alegadamente de server.login para x-terminal.shell. A suposição é que o terminal x provavelmente confia no servidor, pelo que o x-terminal fará tudo o que o servidor (ou qualquer coisa disfarçada de servidor) pedir. O x-terminal responde então a servir0r 6 4 um SYN-ACK, que tem de ser ACK'd para que a ligação possa ser aberta. Como servidor está a ignorar os pacotes
enviado para server.login, o ACK também deve ser forjado. Normalmente, o número de sequência do SYN-ACK é necessário para gerar um ACK válido. No entanto, o atacante é capaz de prever o número de sequência contido no SYN-ACK com base no comportamento conhecido do gerador de números de sequência TCP do x-terminal, e é assim capaz de ACK o SYN-ACK sem o ver:
14:18:36.245045 server.login > x-terminal.shell: S 1382727010:1382727010(0) win 4096
14:18:36.755522 server.login > x-terminal.shell: . ack 2024384001 win 4096

A máquina de spoofing tem agora uma ligação unidireccional ao x-terminal.shell, que parece ser do server.login. Pode manter a ligação e enviar dados desde que possa ACK devidamente quaisquer dados enviados por x-terminal. Envia o seguinte:
14:18:37.265404 server.login > x-terminal.shell: P 0:2(2) ack 1 win 4096
14:18:37.775872 server.login > x-terminal.shell: P 2:7(5) ack 1 win 4096
14:18:38:38.287404 server.login > x-terminal.shell: P 7:32(25) ack 1 win
4096 que corresponde a:14:18:37 server# rsh x-terminal "echo + +
>>/.rhosts"
Vemos agora RSTs para repor as ligações "semi-abertas" e esvaziar a fila de ligações para server.login:
14:18:52.298431 130.92.6.97.600 > server.login: R 1382726960:1382726960(0) vencer 4096
14:18:52.363877 130.92.6.97.601 > server.login: R 1382726961:1382726961(0) vencer 4096
Tempo total decorrido desde o primeiro pacote falsificado: < 16 segundos.
A ligação falsificada é desligada. Server.login pode novamente aceitar ligações. Após o acesso root ter sido obtido através de spoofing de endereço IP, um módulo do kernel chamado "tap-2.01" foi compilado e instalado no x-terminal:
x-terminal% modstat
Tipo Id Tamanho da carga B-major C-major Sysnum Mod
Nome 1 Pdrv ff050000 1000 59. tap/tap-2.01 alfa
x-terminal% ls -l /dev/tap
crwxrwxrwx 1 root 37, 59 Dez 25 14:40 /dev/tap
Este parece ser um módulo STREAMS de núcleo que pode ser empurrado para uma pilha STREAMS existente e utilizado para assumir o controlo de um dispositivo tty. Era utilizado para assumir o controlo de um

sessão de login já autenticada para o alvo.

Portanto, como foi fácil para Kevin Mitnick utilizar esta técnica, vejamos. O IP-spoofing é um método técnico complexo de ataque. É composto por vários componentes que incluem a exploração da relação de confiança. Os Jogadores requeridos são:
A: Anfitrião
alvo B:Anfitrião
de confiança
X:Hospedeiro
inalcançável Z:
Hospedeiro
atacante
(1)2: Anfitrião 1 disfarçado de anfitrião 2

Os números são:
assinale a anfitrião a
controlo b
1 A ---SYN---> B

carrapato: Um tique de tempo. Não há distinção quanto a *como* muito tempo passa entre carraças, apenas que o tempo passa. Geralmente não é grande coisa. **Hospedar a**: Uma máquina que participa numa conversa baseada em TCP.
controlo: Este campo mostra quaisquer bits de controlo relevantes definidos no cabeçalho TCP e a direcção em que os dados estão a fluir. **Host b**: Uma máquina que participa numa conversa baseada em TCP. Neste caso, no primeiro ponto de referência no tempo o anfitrião está a enviar um segmento TCP para o anfitrião b com o bit SYN ligado. A menos que indicado, geralmente não estamos preocupados com a parte de dados do segmento TCP.
Relacionamentos de Confiança: No mundo Unix, a confiança pode ser dada com demasiada facilidade. Digamos que tem uma conta na máquina A, e na máquina B. Para facilitar a transição entre as duas com um mínimo de incómodo, pretende estabelecer uma relação de confiança full-duplex
entre eles. No seu directório pessoal em A cria um ficheiro .rhosts: `echo "B nome de utilizador > ~/.rhosts` no seu directório pessoal em B cria um ficheiro .rhosts: `echo "Um nome de utilizador > ~/.rhosts` (Alternativamente, a raiz pode configurar regras semelhantes em /etc/hosts.equiv, com a diferença de que as regras são de acolhimento amplo, em vez de apenas numa base individual). Agora, pode usar qualquer um dos comandos r* sem aquele incómodo de autenticação de senha.
Estes comandos permitirão uma autenticação baseada no endereço, que concederá ou negará o acesso com base no endereço IP do requisitante do serviço.
Rlogin: O Rlogin é um protocolo simples baseado no cliente-servidor que utiliza TCP como transporte. Rlogin permite que um utilizador faça o login remotamente de um host para outro, e, se a máquina alvo confiar no outro, o rlogin permitirá a conveniência de não solicitar uma palavra-passe. Em vez disso, terá autenticado o cliente através do endereço IP de origem. Assim, a partir do nosso exemplo acima, podemos utilizar o rlogin para iniciar sessão remotamente em A a partir de B (ou vice-versa) e não ser solicitados a pedir uma palavra-passe.
Protocolo Internet: IP é o protocolo de rede sem ligação e não fiável no

Conjunto TCP/IP. Tem dois campos de cabeçalho de 32 bits para guardar informação de endereço. IP é também o mais ocupado de todos os protocolos TCP/IP, uma vez que quase todo o tráfego TCP/IP está encapsulado em datagramas IP. A função do IP é encaminhar pacotes em torno da rede. Não fornece nenhum mecanismo de fiabilidade ou responsabilidade, para isso, depende das camadas superiores. O IP envia simplesmente datagramas e espera que estes o tornem intacto. Se não o fizerem, o IP pode tentar enviar uma mensagem de erro ICMP de volta à fonte, contudo este pacote também se pode perder. (ICMP é Internet

Control Message Protocol e é utilizado para retransmitir condições de rede e diferentes erros para IP e os outros níveis). O IP não tem meios para garantir a entrega. Uma vez que o IP é sem ligação, não mantém qualquer informação sobre o estado da ligação. Cada datagrama IP é enviado sem ter em conta o último ou o próximo. Isto, juntamente com o facto de que
é trivial modificar a pilha IP para permitir um endereço IP escolhido arbitrariamente nos campos de origem (e destino), tornando o IP facilmente subvertido.

Protocolo de Controlo de Transmissão : TCP é o protocolo de transporte fiável e orientado para a ligação no conjunto TCP/IP. Orientado para a ligação significa simplesmente que os dois anfitriões que participam numa discussão devem primeiro estabelecer uma ligação antes que os dados possam mudar de mãos. A fiabilidade é fornecida de várias maneiras, mas as duas únicas que nos preocupam são a sequenciação e o reconhecimento dos dados. O TCP atribui números de sequência a cada segmento e reconhece todo e qualquer segmento de dados recebido do outro extremo.
(Os ACK consomem um número de sequência, mas não são eles próprios 'd.) Esta fiabilidade torna o TCP mais difícil de enganar do que o IP.

Números de Sequência, Agradecimentos e outras bandeiras: Uma vez que o TCP é fiável, deve ser capaz de recuperar de dados perdidos, duplicados, ou fora de ordem. Atribuindo um número de sequência a cada byte transferido, e exigindo um reconhecimento de
o outro extremo no momento da recepção, o TCP pode garantir uma entrega fiável. A extremidade receptora utiliza os números sequenciais para assegurar a encomenda adequada dos dados e para eliminar bytes de dados duplicados. Os números sequenciais TCP podem ser pensados simplesmente como contadores de 32 bits. Eles variam de 0 a 4.294.967.295. Cada byte de dados trocado através de uma ligação TCP (juntamente com certas bandeiras) é sequenciado. O campo do número de sequência no cabeçalho TCP conterá o número de sequência do byte *primeiro* de dados no segmento TCP. O campo do número de reconhecimento no cabeçalho TCP contém o valor do próximo número sequencial *esperado*, e também reconhece *todos* os dados através deste número ACK menos um. O TCP utiliza o conceito de publicidade em janela para controlo de fluxo. Utiliza uma janela deslizante para dizer à outra extremidade a quantidade de dados que pode guardar. Uma vez que o tamanho da janela é de 16 bits, um TCP receptor pode anunciar até um máximo de 65535 bytes. A publicidade de janela pode ser pensada de um TCP para o outro de quão altos podem ser os números de sequência aceitáveis. Outras bandeiras de cabeçalho TCP de nota são RST (reset), PSH (push) e FIN (finish). Se for recebido um RST, a ligação é imediatamente destruída. Os RSTs são normalmente enviados quando uma extremidade recebe um segmento que simplesmente não se conecta com a ligação actual (encontraremos um exemplo abaixo). A bandeira PSH diz ao receptor para passar todos os dados está em fila de espera para a aplicação, o mais rapidamente possível. A bandeira FIN é a forma como uma aplicação inicia um encerramento gracioso de uma ligação (o encerramento da ligação é um processo de 4 vias). Quando uma extremidade recebe uma FIN, ela ACEITA-a, e não espera receber mais dados (o envio ainda é possível, no entanto).

Estabelecimento da ligação TCP: A fim de trocar dados utilizando TCP, os anfitriões devem estabelecer uma ligação. O TCP estabelece uma ligação num processo de 3 passos chamado
o aperto de mão de 3 vias. Se a máquina A estiver a executar um cliente rlogin e desejar ligar-se a um daemon rlogin na máquina B, o processo é o seguinte:

1 A ---SYN---> B

2 A<---SYN/ACK--- B
3 A ---VOLTAR--->B

Em (1) o cliente está a dizer ao servidor que quer uma ligação. Este é o único propósito da bandeira SYN. O cliente está a dizer ao servidor que o campo do número de sequência é válido, e deve ser verificado. O cliente irá definir o campo do número de sequência no cabeçalho TCP para o seu ISN (número de sequência inicial). O servidor, ao receber este segmento (2) responderá com a sua própria ISN (portanto a bandeira SYN está ligada) e um ACKnowledgement do primeiro segmento do cliente (que é a ISN+1 do cliente). O cliente depois ACK é a ISN do servidor (3). Agora, a transferência de dados pode ter lugar.

O ISN e o Número de Sequência: É importante compreender como os números de sequência são inicialmente escolhidos, e como mudam em relação ao tempo. O número sequencial inicial quando um anfitrião é inicializado é inicializado em 1 (TCP na realidade chama a esta variável 'tcp_iss', pois é o número sequencial inicial *send*. A outra variável do número de sequência, 'tcp_irs' é o número de sequência inicial *receber* e é aprendida durante os 3 passos
estabelecimento de ligação. Não nos vamos preocupar com a acção0e6dEi4). Esta prática está errada, e é reconhecida como tal num comentário a função tcp_init() onde ela aparece. A ISN é incrementada em 128.000 a cada segundo, o que provoca a função tcp_init() de 32 bits
Contador ISN para embrulhar a cada 9,32 horas se não ocorrerem ligações. No entanto, cada vez que é emitida uma ligação(), o contador é incrementado em 64.000. Uma razão importante por detrás desta previsibilidade é minimizar a hipótese de que dados de uma encarnação mais antiga (isto é, do mesmo 4-tuplo dos endereços IP locais e remotos das portas TCP) da ligação actual possam chegar e sujar as coisas. O conceito do tempo de espera 2MSL aplica-se aqui, mas está para além do âmbito deste documento. Se os números sequenciais fossem escolhidos aleatoriamente quando uma ligação chegasse, não poderiam ser dadas garantias de que os números sequenciais seriam diferentes de uma encarnação anterior. Se alguns dados que estavam presos num laço de encaminhamento algures finalmente se libertassem e vagueassem para a nova encarnação da sua antiga ligação, poderiam realmente sujar as coisas.

Portos: Para conceder acesso simultâneo ao módulo TCP, o TCP fornece uma interface de utilizador chamada porta. As portas são utilizadas pelo núcleo para identificar processos de rede, são entidades estritamente da camada de transporte (ou seja, que o IP se poderia importar menos com elas).
Juntamente com um endereço IP, uma porta TCP fornece um ponto final para as comunicações em rede. De facto, em qualquer momento *todas* as ligações à Internet podem ser descritas por 4 números: o endereço IP de origem e a porta de origem e o endereço IP de destino e a porta de destino. Os servidores estão ligados a portas "bem conhecidas", de modo a poderem ser localizados numa porta padrão em diferentes sistemas. Por exemplo, o daemon rlogin situa-se na porta TCP 513. IP-spoofing consiste em várias etapas, que aqui descreverei brevemente, depois explicarei em pormenor. Primeiro, é escolhido o anfitrião alvo. Em seguida, é descoberto um padrão de confiança, juntamente com um hospedeiro de confiança. O hospedeiro de confiança é então desactivado, e os números de sequência TCP do alvo são amostrados. O hospedeiro de confiança é imitado, os números de sequência são adivinhados, e é feita uma tentativa de ligação a um serviço que requer apenas autenticação baseada em endereços. Se bem sucedido, o atacante executa um simples comando para deixar uma porta traseira.

Artigos Necessários: Há um par de itens que são necessários para realizar este ataque: hospedeiro alvo, hospedeiro de confiança, hospedeiro atacante (com acesso à raiz), e software IP-spoofing. Geralmente o ataque é feito a partir da conta de raiz no anfitrião atacante contra a conta de raiz no alvo. Um factor frequentemente negligenciado, mas

crítico no IP-spoofing é o facto de o ataque ser cego. O atacante vai assumir a identidade de um hospedeiro de confiança, a fim de subverter a segurança do hospedeiro alvo. O hospedeiro de confiança é desactivado usando

o método descrito abaixo. Tanto quanto o alvo sabe, está a levar a cabo uma conversa com um amigo de confiança. Na realidade, o atacante está sentado em algum canto escuro da Internet, forjando pacotes a partir deste anfitrião de confiança enquanto está preso numa batalha de negação de serviço.

Os datagramas IP enviados com o endereço IP forjado atingem a multa alvo (recorde-se que IP é um protocolo orientado para a ligação - cada datagrama é enviado sem consideração pela outra extremidade) mas os datagramas que o alvo envia (destinados ao hospedeiro de confiança) acabam no balde de bits. O atacante nunca os vê. Os routers intervenientes sabem para onde os datagramas devem ir. É suposto irem para o hospedeiro de confiança. No que diz respeito à camada da rede, é de onde eles vieram originalmente, e é para aqui que as respostas devem ir. Claro que uma vez que os datagramas são encaminhados para lá, e a informação é desmultiplexada na pilha de protocolos, e chega ao TCP, é descartada (o TCP do hospedeiro de confiança não pode responder. Portanto, o atacante tem de ser inteligente e *saber* o que foi enviado, e *saber* qual a resposta que o servidor procura. O attac0k6eEr4cannot vê o que o hospedeiro alvo envia, mas pode *predizer* o que vai enviar; isso, juntamente com o conhecimento do que *virá* enviar, permite ao atacante trabalhar em torno desta cegueira.

Depois de um alvo ser escolhido, o atacante deve determinar os padrões de confiança (por uma questão de argumentação, vamos assumir que o anfitrião alvo *confia de facto em alguém. Se não confiasse, o ataque terminaria aqui). Descobrir em quem um hospedeiro confia pode ou não ser fácil. Um 'showmount -e' pode mostrar onde os sistemas de ficheiros são exportados, e o rpcinfo também pode dar informações valiosas. Se se souber informação suficiente sobre o anfitrião, não deve ser muito difícil. Se tudo o resto falhar, tentar endereços IP vizinhos num esforço de força bruta pode ser uma opção viável. Uma vez encontrado o hospedeiro de confiança, este deve ser desactivado. Uma vez que a atacante vai fazer-se passar por ela, ela deve certificar-se de que este anfitrião não pode receber qualquer tráfego de rede e coisas sujas. Há muitas maneiras de o fazer, a que vou discutir é a inundação do TCP SYN. Uma ligação TCP é iniciada com um cliente que emite um pedido a um servidor com a bandeira SYN no cabeçalho TCP.

Normalmente o servidor irá emitir um SYN/ACK de volta para o cliente identificado pelo endereço de origem de 32 bits no cabeçalho IP. O cliente enviará então um ACK para o servidor (como vimos na figura 1 acima) e a transferência de dados pode começar. No entanto, existe um limite superior de quantos pedidos SYN simultâneos TCP podem ser processados para um determinado socket. Este limite chama-se backlog, e é o comprimento da fila onde as ligações de entrada (ainda incompletas) são mantidas. Este limite da fila aplica-se tanto ao número de ligações incompletas (o aperto de mão de 3 vias não está completo) como ao número de ligações completadas que não foram puxadas da fila pela aplicação através de
a chamada ao sistema de aceitação(). Se este limite de atraso for atingido, o TCP descartará silenciosamente todos os pedidos SYN recebidos até que as ligações pendentes possam ser tratadas. Aí reside o ataque. O anfitrião atacante envia vários pedidos SYN para a porta TCP que deseja desactivar. O anfitrião atacante deve também certificar-se de que o endereço IP de origem é falsificado para ser o de outro anfitrião, actualmente inatingível (o TCP alvo enviará a sua resposta para este endereço. (O IP pode informar o TCP de que o anfitrião não é contactável, mas o TCP considera estes erros transitórios e deixa a resolução dos mesmos para o IP (redireccionar os pacotes, etc.) ignorando-os efectivamente). O endereço IP deve ser inalcançável porque o atacante não quer que nenhum anfitrião receba os SYN/ACKs que virão do TCP alvo (isto

resultaria no envio de um RST para o TCP alvo, o que iria impedir o nosso ataque). O processo é o seguinte:

```
1     Z(x)   ---SYN--->      B
      Z(x)   ---SYN--->      B
      Z(x)   ---SYN--->      B
      Z(x)   ---SYN--->      B
      Z(x)   ---SYN--->      B
2     X<---SYN/ACK---         B
      X   <--SYN/ACK---       B
3     X<---RST---             B
```

Em (1) o hospedeiro atacante envia uma multidão de pedidos SYN para o alvo (lembre-se que o alvo nesta fase do ataque é o hospedeiro de confiança) para preencher a sua fila de espera com ligações pendentes. (2) O alvo responde com SYN/ACKs ao que acredita ser o
fonte dos SYNs recebidos. Durante este tempo todos os outros r e q 0 u 6 e E s t 4 s a esta porta TCP serão ignorados. Diferentes implementações de TCP têm diferentes tamanhos de atraso. O BSD tem geralmente um atraso de 5 (o Linux tem um atraso de 6). Há também uma margem de 'graça' de 3/2. Que
é, o TCP permitirá até ligações em atraso*3/2+1. Agora o atacante precisa de ter uma ideia de onde está o TCP do alvo no espaço do número de sequência de 32 bits. O atacante liga-se a uma porta TCP no alvo (SMTP é uma boa escolha) imediatamente antes de lançar o ataque e completa o aperto de mão de três vias. O processo é exactamente igual ao da fig(1), excepto que o atacante irá guardar o valor da ISN enviada pelo hospedeiro alvo.
Muitas vezes, este processo é repetido várias vezes e a ISN final enviada é armazenada. O atacante precisa de ter uma ideia do que o RTT (round-trip time) do alvo para o anfitrião. O processo pode ser repetido várias vezes, e é calculada uma média dos RTT's). O RTT é necessário para ser capaz de prever com precisão a próxima ISN. O atacante tem a linha de base (a última ISN enviada) e sabe como os números da sequência são incrementados (128.000/segundo e 64.000 por ligação) e tem agora uma boa ideia de quanto tempo levará um datagrama IP a percorrer a Internet para atingir o alvo (aproximadamente metade do RTT, uma vez que a maioria das vezes as rotas são simétricas). Depois de o atacante ter esta informação, passa imediatamente à fase seguinte do ataque (se outra ligação TCP chegasse a qualquer porta do alvo antes de o atacante poder continuar o ataque, a ISN prevista pelo atacante estaria desligada por 64.000 do que foi previsto). Quando o segmento falsificado chega ao alvo, podem acontecer várias coisas diferentes, dependendo da precisão da previsão do atacante:
Se o número de sequência estiver exactamente onde o TCP receptor espera que esteja, os dados recebidos serão colocados na próxima posição disponível no buffer de recepção. Se o número sequencial for MENOR do que o valor esperado, o byte de dados é considerado uma retransmissão, e é descartado. Se o número sequencial for MAIOR do que o valor esperado mas ainda dentro dos limites da janela de recepção, o byte de dados é considerado um byte futuro, e é mantido pelo TCP, enquanto se aguarda a chegada dos outros bytes em falta. Se um segmento chegar com um número de sequência MAIS GRANDE do que o valor esperado e NÃO dentro dos limites da janela de recepção, o segmento é abandonado, e o TCP enviará um segmento de volta com o número de sequência *esperado*.

```
1     Z(b)   ---SYN--->      A
2     B   <--SYN/ACK---      A
```

3 Z(b) ---VOLTAR--->A
4 Z(b) ---PSH---> A

O anfitrião atacante falsificou o seu endereço IP para ser o do anfitrião de confiança (que ainda deve estar nas gargantas da morte do ataque D.O.S.) e envia o seu pedido de ligação à porta 513 no alvo (1). Em (2), o alvo responde ao pedido de ligação falsificada com um SYN/ACK, o que fará o seu caminho para o hospedeiro de confiança (que, se *poderia* processar o segmento TCP de entrada, considerá-lo-ia um erro, e enviaria imediatamente um RST para o alvo). Se tudo correr como planeado, o SYN/ACK será abandonado pelo hospedeiro de confiança amordaçado. Depois (1), o atacante deve recuar um pouco para dar ao alvo tempo suficiente para enviar o SYN/ACK (o atacante não pode ver este segmento). Depois, ao (3) o atacante envia um ACK ao alvo com o número de sequência previsto (mais um,
porque estamos a ACREDITÁ-LO). Se o atacante estiver correcto no seu p r e d 0 i c 6 t E i o 4 n , o alvo aceitará o ACK. O alvo está comprometido e a transferência de dados pode começar (4). Geralmente, após o compromisso, o atacante irá inserir uma porta traseira no sistema que irá
permitir uma forma mais simples de intrusão. (Muitas vezes é feito um `cat + + >> ~/.rhosts'. Esta é uma boa ideia por várias razões: é rápida, permite a simples reentrada, e não é interactiva.
Lembre-se que o atacante não pode ver nenhum tráfego vindo do alvo, pelo que quaisquer respostas são enviadas para o esquecimento). O IP-Spoofing funciona porque os serviços de confiança dependem apenas da autenticação baseada no endereço da rede. Uma vez que o IP é facilmente enganado, a falsificação de endereços não é difícil. A parte mais difícil do ataque está na previsão do número de sequência, porque é aí que entra em jogo o trabalho de adivinhação. Mesmo uma máquina que envolve todas as suas ligações TCP de entrada com invólucros TCP, ainda é vulnerável ao ataque. Os TCP wrappers dependem de um hostname ou de um endereço IP para autenticação. Reduzir ao mínimo as incógnitas e adivinhações, e o ataque tem mais hipóteses de sucesso.

CONCLUSÃO

Embora este método de ataque pareça impecável, Kevin Mitnick, o conhecido hacker, foi detido a 15 de Fevereiro de 1995. A sua onda de crimes incluiu o roubo de milhares de ficheiros de dados e pelo menos 20.000 números de cartões de crédito de sistemas informáticos de todo o país.
Foi apanhado pela sua própria obsessão e provou que o crime e a punição online é muito mais difícil de pontuar do que Dungeons-and-Dragons.

REFERÊNCIAS

Revista Phrack, www.fc.net/phrack/files/p48/p48-14.html

Mitnick's Malice, Shimomura's Chivalry, www.salon.com

Evidência, www.takedown.com

O caso Kevin Mitnick/Tsutomu Shimomura, www.gulker.com/ra/hack

A Intrusão de Sistemas de Janeiro de 1995, www.well.com/intrusion