DESENVOLVIMENTO E APLICAÇÃO DE EXPLOITS UTILIZANDO O METASPLOIT FRAMEWORK*

DEVELOPMENT AND APPLICATION OF EXPLOITS USING METASPLOIT FRAMEWORK

*Julio Cesar L. Della Flora***

RESUMO:
Este artigo teve como objetivo expor os benefícios da utilização do framework Metasploit e evidenciar como seus recursos aceleram o desenvolvimento de código para exploração de vulnerabilidades. Uma introdução às principais técnicas de exploração, assim como o conhecimento necessário para aplicá-las foram apresentados neste projeto. O prévio conhecimento da linguagem Assembly para arquitetura IA32 se faz necessário à compreensão total da pesquisa. Posteriormente um programa escrito com o intuito de ser vulnerável a buffer overflow serviu de base para o desenvolvimento de um código que se aproveitasse dessa falha, a este código é dado o nome de exploit. O Buffer overflow é uma falha de segurança, ou seja, uma vulnerabilidade na qual se utiliza o estouro do buffer a partir da oferta de dados superior a que a variável é capaz de armazenar. Esse exploit foi codificado utilizando o framework Metasploit, que é uma plataforma de código aberto destinada a acelerar e reforçar o desenvolvimento e a utilização destes códigos. Ao final deste trabalho a interface msfgui presente no framework foi utilizada para que o exploit desenvolvido fosse lançado, obtendo uma sessão não autorizada entre o os computadores envolvidos.

PALAVRAS CHAVE: Metasploit, Exploit, Segurança, Vulnerabilidade.

ABSTRACT:
This paper aimed to explain the benefits of using the Metasploit framework and show how their resources accelerate the development of code to exploit vulnerabilities. Introductions to the main exploit techniques, as well as the minimum knowledge necessary to apply them were presented in this project. Some knowledge of Assembly language to IA32 architecture is necessary to achive the complete understanding of this research. Posteriorly, a program was written with the intention to be buffer overflow vulnerable and served as base for the development of a code that takes advantage of this failure, this code is known as an exploit. The buffer overflow is a security flaw, in other words, a vulnerability in wich you get the buffer burst by providing more data than the variable can store. This exploit has been coded using the Metasploit framework, that is an open source platform designed to accelerate and enhance the development and use of these codes. At the end of this work the msfgui interface, provided by the framework, was used to released the developed exploit, obtaining an unauthorized session among the computers involved.

Keywords: Metasploit, Exploit, Security, Vulnerability

.* Trabalho de conclusão de curso (TCC) apresentado ao curso de Ciência da Computação do Centro Universitário Filadélfia para obtenção do título de Bacharel em Ciência da Computação.

****Graduado em Ciência da Computação pelo Centro Universitário Filadélfia (UNIFIL).**

1. INTRODUÇÃO

Estudos realizados pela Associação Brasileira de Empresas de Software (ABES, 2009) apontam que o mercado brasileiro de software e serviços ocupa a 12ª posição no mercado mundial, tendo movimentado em 2008 aproximadamente 15 bilhões de dólares, equivalente a 0,96% do PIB brasileiro naquele ano. Deste total, foram movimentados cinco bilhões em software, o que representou perto de 1,68% do mercado mundial. Os restantes 10 bilhões foram movimentados em serviços relacionados.

As empresas de desenvolvimento de software vêm crescendo de forma exponencial, em todo mundo a competitividade do mercado obriga os desenvolvedores a acelerar o processo de criação exigindo que programas sejam lançados sem que as verificações relativas à segurança possam ser devidamente aplicadas.

Esses programas em sua maioria apresentam falhas de programação que deveriam ser corrigidas antes de lançados ao mercado, dificultando assim a incidência de vulnerabilidades.

Por se tratar de processos muitas vezes exaustivos, a devida atenção referente á segurança não é empregada, isso torna o software um possível alvo para indivíduos mal intencionados comprometendo dados sigilosos do sistema.

Para que a exaustão no processo de verificação seja minimizada, faz-se necessário uma ferramenta que reduza o tempo empregado no desenvolvimento do teste de segurança efetivo.

O Metasploit é um framework de exploração de código aberto concebido para proporcionar ao usuário um modelo de desenvolvimento de exploits, possibilitando que longos trechos de código sejam reutilizados. Essa funcionalidade diminui o tempo gasto na implementação do código, o qual pode ser reaproveitado em experiências futuras.

Esse framework separa de maneira eficaz o código que explora falhas de software (conhecido como exploit), do código que é executado no sistema objeto com a finalidade de adquirir privilégios do usuário atual (payload), tornando possível a utilização de um payload em vários exploits.

2 O METASPLOIT FRAMEWORK

Aqueles que tiveram a oportunidade de assistir à palestra de H. D. Moore (Black Hat 2004) puderam presenciar algo somente visto em filmes de ficção tornar-se realidade. O título da palestra “Hacking Like in the Movies” referia-se à chegada de sua ferramenta, o Metasploit Framework (MSF) versão 2.2.0.

As atenções se voltaram para dois telões que mostravam respectivamente o console do MSF e um sistema operacional Windows (a ser comprometido). O sistema foi comprometido com apenas alguns comandos, estabelecendo assim uma conexão com privilégios administrativos entre o computador de H. D. Moore e o alvo, mas isso era apenas uma amostra do estava por vir.

A ideia original do Metasploit era criar um jogo que simulasse um ambiente virtual explorável ao qual se assemelhasse com a realidade, o projeto gradualmente se tornou um framework que visava ao funcionamento, configuração e desenvolvimento de exploits para vulnerabilidades já conhecidas. A versão 2.1 do produto foi lançada em junho de 2004, desde então o desenvolvimento do produto e a adição de novos exploits e payloads têm aumentado rapidamente.

Embora inicialmente a estrutura não fornecesse nenhum suporte a colaboradores, com o lançamento da versão 2.2 o framework se tornou muito mais amigável aos desenvolvedores. A versão 2.x originalmente escrita em Perl, Assembly e C, logo concedeu lugar à versão 3.x que

foi completamente reescrita em Ruby, revisando a arquitetura, interface e as API's fornecidas aos usuários.

A popularidade da ferramenta pode ser medida baseada em uma pesquisa desenvolvida por seu criador H. D. Moore e apresentada em Cansecwest 2006 e 2007, o framework foi mencionado em 17 livros, 950 blogs e 190 artigos desde a liberação da versão 3.0 estável em março de 2007, recebendo em menos de dois meses 20.000 requisições de download, neste mesmo período o utilitário msfupdate desenvolvido para atualizar a ferramenta foi usado por mais de 4.000 endereços de IP segundo MAYNOR e MOOKHEY (2007).

2.1 Objetivos do Metasploit

O Metasploit surgiu para fornecer um framework que auxiliasse profissionais de segurança da informação a desenvolver exploits. Segundo MAYNOR e MOOKHEY (2007) o ciclo de vida de uma vulnerabilidade e sua exploração é organizado em seis fases mostradas a seguir:

1. Descoberta: O pesquisador descobre uma falha crítica na segurança de um software.
2. Divulgação: O pesquisador informa o desenvolvedor do software sobre a falha para que as medidas necessárias sejam tomadas.
3. Análise: O pesquisador ou qualquer pessoa interessada começa a análise da vulnerabilidade a fim de determinar a sua explorabilidade. As perguntas mais comuns nessa etapa são: A falha pode ser explorada? Remotamente? A exploração poderia resultar em uma execução remota de código, ou simplesmente causaria um crash no serviço remoto? Que tamanho de código exploratório poderia ser injetado? Esta fase também analisa o aplicativo enquanto o código é inserido.
4. Desenvolvimento do Exploit: Após as principais questões serem respondidas, o desenvolvimento do exploit começa. Este considerado a "arte negra" do processo, exige uma profunda compreensão dos registradores do processador, código Assembly, offsets e payloads.
5. Teste: Esta é a fase em que o codificador verifica o que, de fato, é vulnerável testando varias plataformas, service pack's ou patches e algumas vezes até processadores com arquiteturas diferentes.
6. Lançamento: Depois de ser testado, e os parâmetros necessários para a sua execução serem determinados, o codificador apresenta publicamente o seu projeto. Muitas vezes o código apresenta falhas propositais em sua composição para dissuadir usuários comuns a executá-lo contra sistemas vulneráveis.

Com a chegada do Metasploit, escrever um exploit tornou-se simples mesmo para um programador amador. O framework já vem com mais de 300 exploits prontos para execução. Os desenvolvedores estão avançando rapidamente assim como a popularidade da ferramenta. Essa é em demasia semelhante ao grande numero de plugins que o Nessus possui no momento, porém a fama atribuída ao MSF não se dá somente pelo crescente repositório de código, mas pelo modo que é desenvolvido e arquitetado.

O MSF concorre diretamente com produtos comerciais como o Immunity's Canvas e o Core Security Technology's IMPACT. No entanto, existe uma grande diferença entre os objetivos do MSF comparado aos produtos citados. Os produtos comerciais destinados a testes de intrusão apresentam interfaces amigáveis e extenso repositório de exploits enquanto o MSF explora o desenvolvimento

de exploits, payloads, encoders, geradores de NOP's e ferramentas de reconhecimento. Além disso, é também uma plataforma para projetar utilitários que permitam a investigação e o desenvolvimento de novas técnicas para testes de segurança.

É provável que o Metasploit framework torne-se a primeira ferramenta de segurança (parcialmente open-source, uma vez que agora é distribuída sob a sua própria licença) gratuita a abranger toda uma gama de testes de segurança com módulos para determinar hosts vulneráveis, interface com scanners como o Nmap e Nessus, exploits, payloads e post-exploitation goodies para apropriar-se furtivamente do sistema e, possivelmente, de toda a rede conforme MAYNOR e MOOKHEY (2007).

3 METODOLOGIA

Em um laboratório montado para ilustrar as técnicas de intrusão desse trabalho foram utilizados:

- 1 desktop com processador athlon 64 X2 tendo o Windows XP SP0 como seu sistema operacional.
- 1 notebook com processador athlon 64 X2 tendo o Linux BackTrack 4 como seu sistema operacional.
- 1 roteador D-link dir 500 responsável pela conexão entre os computadores.

O BackTrack 4 foi escolhido para este laboratório por ser uma distribuição Linux voltada a testes de intrusão, tendo o MSF instalado por padrão. Um programa desenvolvido especialmente para ser vulnerável a ataques de bufferoverflow será instalado no computador alvo, esse programa, chamado bof-server (PELAGALLI, 2008) foi desenvolvido para ser utilizado em plataformas Windows.

Técnicas contidas em Writing exploits for Metasploit 3.0 (http://redstack.net) serão utilizadas para conceber um código capaz de efetuar uma conexão não autorizada entre dois computadores.

Utilizando-se de uma vulnerabilidade conhecida como Stack Overflow, o código desenvolvido em Ruby sobrescreve o registrador EIP alterando o fluxo do programa para o endereço de memória onde se encontra o payload, que por sua vez irá estabelecer a conexão entre os computadores.

Como dito anteriormente, o Bof-server foi desenvolvido para ser explorado ao decorrer desse estudo, contendo uma falha de programação proposital, no entanto falhas desse tipo são comumente encontras em programas comerciais.

Para iniciar o programa deve-se utilizar a seguinte sintaxe:

> Bof-server.exe (porta)

O bof-server é um programa extremamente simples com apenas dois comandos, version e quit. Digitando version a versão do programa é exibida, o comando quit fecha o aplicativo.

```
> telnet localhost 4242
> version
bof-server v0.01
> quit
```

O aplicativo permite a execução remota de código devido a um stack overflow introduzido pela função getl(int fd, char * s). Conforme a figura 3.1, quando um conjunto extenso de caracteres é introduzido, o aplicativo para de responder.

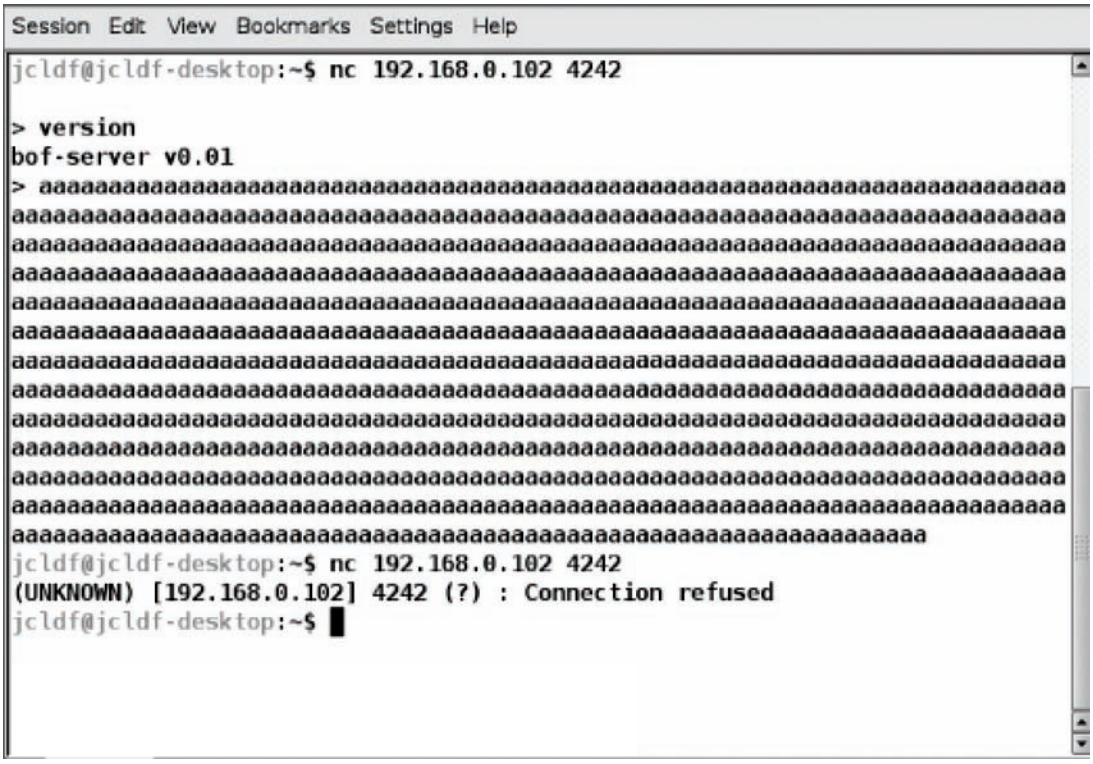

Figura 3.1: Vulnerabilidade no programa bof-server.

O erro gerado pelo programa será essencial para o desenvolvimento do exploit, no entanto sem um depurador de código não se pode determinar certas informações que são essenciais na confecção deste exploit.

Realizando o mesmo procedimento acima citado, já com o depurador de código anexado, conclui-se, por informações contidas no rodapé do debugger, que houve uma violação de acesso quando o bof-server executou o conjunto de caracteres enviados (figura 3.2). Nesse caso o conjunto de números “61” informado pelo depurador diz respeito à representação do caractere “a” derivado do Código Padrão Americano para o Intercâmbio de Informação (ASCII).

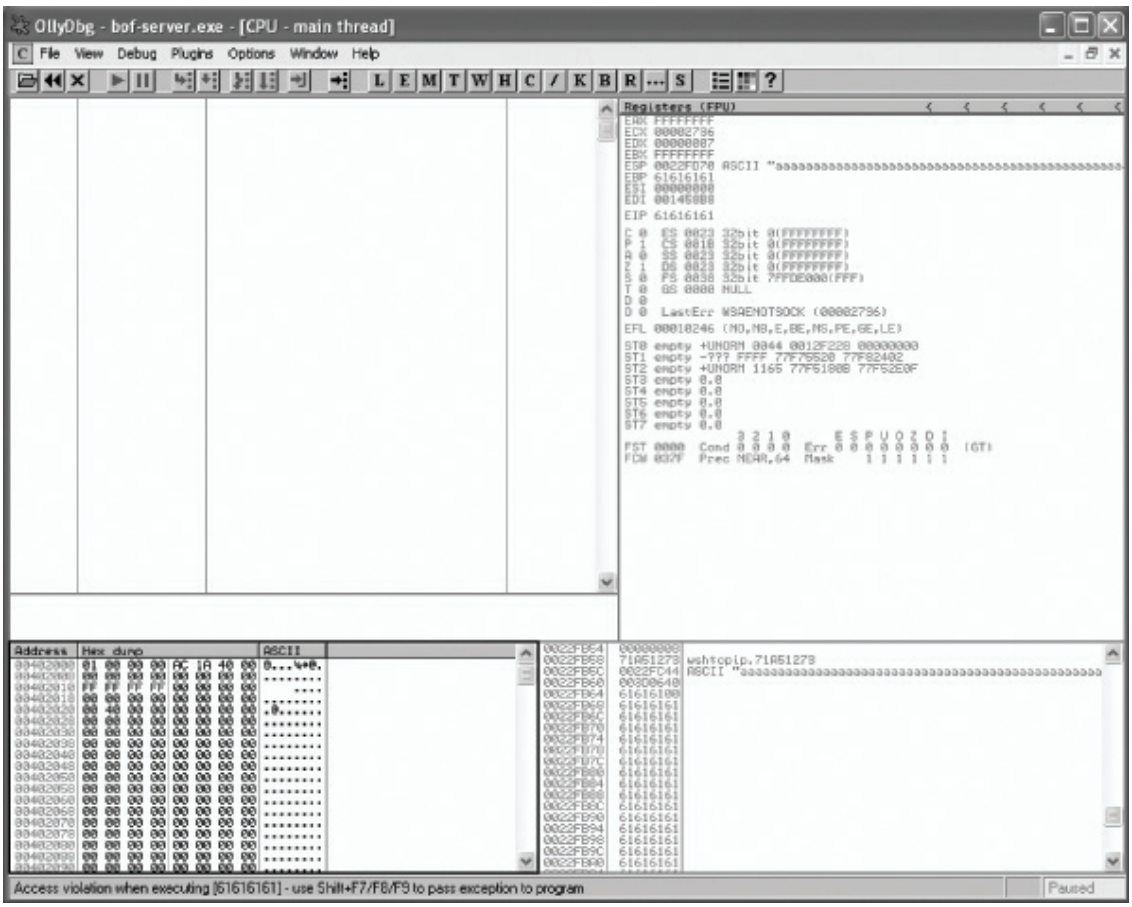

Figura 3.2: Análise do programa utilizando o OllyDbg.

Como descrito anteriormente, o programa deixa de responder quando um grande número de caracteres é enviado ao servidor. Para determinar com maior exatidão a quantidade de bytes necessários para que isso aconteça, o MSF traz uma interessante ferramenta composta por dois scripts: pattern_create.rb e pattern_offset.rb.

A função pattern_create.rb gera uma string única que, quando enviada, retorna o endereço do registrador EIP, por sua vez a função pattern_offset.rb utiliza esta informação para determinar quantos bytes serão necessários para chegar ao endereço do registrador EIP.

Figura 3.3: String única gerada pela função pattern_create.rb.

Observa-se na figura 3.3 que a função pattern_offset.rb retorna o número 520, então serão necessários 520 + 4 bytes para sobrescrever o registrador EIP fazendo com que o programa pare de responder.

Outra informação necessária ao desenvolvimento do exploit é o endereço onde se inicia o overflow na pilha, conforme a figura 3.4 o primeiro endereço totalmente sobrescrito pelo conjunto de caracteres é o “0x22FB68”, posteriormente este endereço será usado como ponto de partida para o payload.

Figura 3.4: Primeiro endereço totalmente sobrescrito.

De posse das informações tratadas anteriormente, pode-se então iniciar a confecção do exploit, o MSF possui um diretório específico situando em “/home/usuário/.msf3” onde os códigos escritos pelo usuário devem ser armazenados, o principal objetivo dessa prática é evitar a perda de dados resultantes das constantes atualizações do framework.

O código disponível no apêndice A diz respeito ao exploit pronto e deverá ser nomeado como “Bof-Server1.rb” e colocado em “/home/usuário/.msf3/modules/exploits/windows/dummy”, os diretórios não existentes deverão ser criados de acordo com o caminho acima. Com o término do desenvolvimento, pode-se enfim iniciar a interface msfgui (figura 3.5).

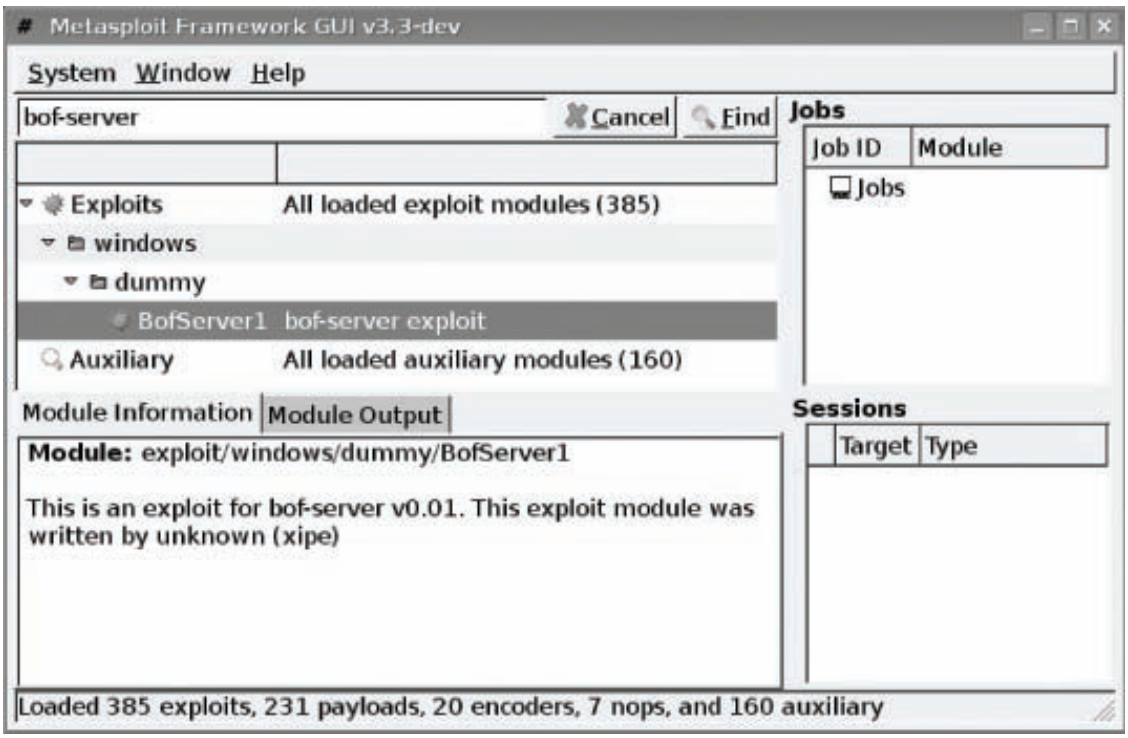

Figura 3.5: Interface msfgui.

Para que o exploit seja lançado, deve-se primeiramente clicar duas vezes com o botão esquerdo em BofServer1 (figura 3.5), uma tela se abrirá pedindo algumas informações como versão do sistema operacional que se deseja atacar, payload que deverá ser usado, endereço IP do alvo, entre outras.O Meterpreter payload será usado nesse exemplo.

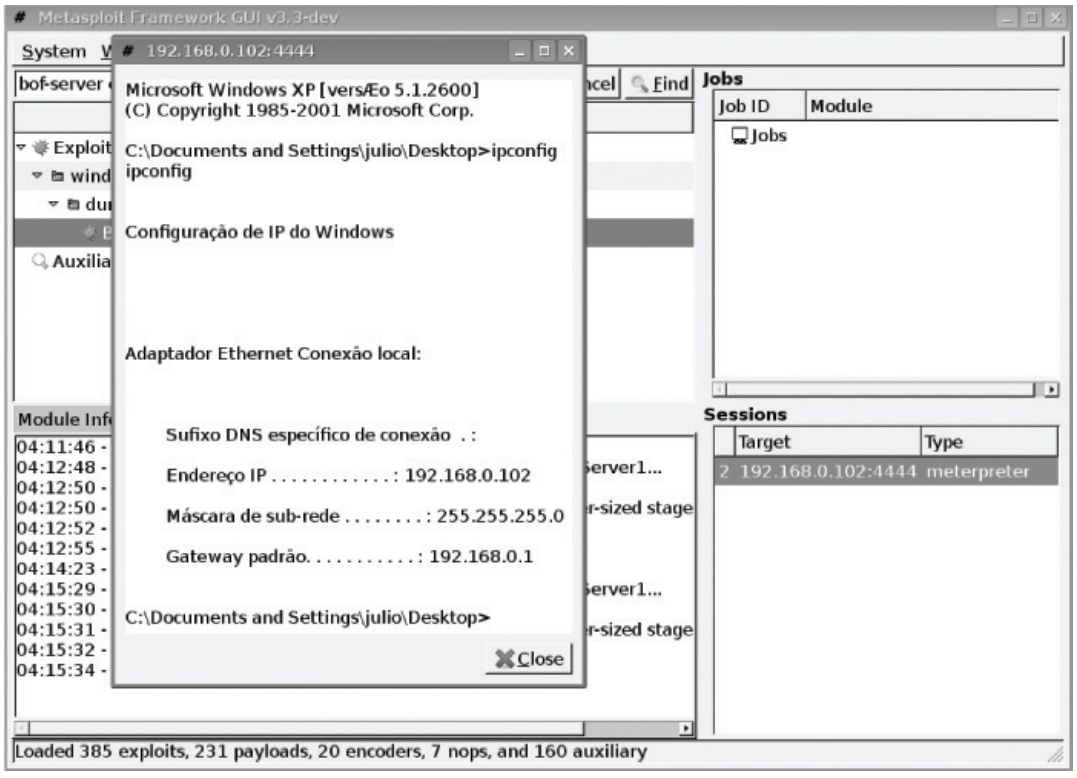

Figura 3.6: Sessão estabelecida.

Após o preenchimento das informações necessárias uma sessão é aberta, indicando que houve êxito na exploração da falha (figura 3.6). Pode-se agora utilizar todas as funções do Meterpreter como copiar e excluir arquivos, obter senhas de usuários, efetuar o upload de executáveis, entre outras, para se aproveitar do sistema invadido, ressaltando que nenhum processo referente ao Metasploit será exibido no gerenciador de tarefas do Windows, tornando o ataque imperceptível ao usuário.

CONCLUSÃO

Com a crescente evolução do mercado brasileiro de software, questões até então ignoradas ou muitas vezes abordadas de maneira errônea, como a segurança de software, devem ser levadas em consideração e discutidas sem discriminação. Tratando-se de um segmento muito competitivo, desenvolvedores correm contra o tempo para atender às necessidades do mercado e muitas vezes concluem produtos de maneira precoce.

Sem que os testes relativos à segurança da informação sejam aplicados, não se pode determinar o comportamento de um produto diante de ataques realizados por indivíduos mal intencionados.

O Metasploit, de fato, agiliza o processo de desenvolvimento de exploits, possibilitando a reutilização de grandes trechos de código e diminuindo o tempo gasto nos testes de segurança, grande parte da sua agilidade é provida pela arquitetura modular ao qual o framework foi desenvolvido e também pela eficiente separação de exploits e payloads.

Conhecer a linguagem Assembly assim como seus registradores e conjuntos de instruções se faz necessário para o desenvolvimento de exploits, sem esse conhecimento prévio os dados retirados do depurador de código não teriam serventia.

Apesar de conhecida e muito grave, a vulnerabilidade de buffer overflow ainda é encontrada em grande parte dos softwares comerciais, por ser uma falha originada exclusivamente pelo codificador, não se pode determinar quanto tempo levará até que ela esteja defasada.

Mesmo contendo uma falha proposital, o programa Bof-server exemplifica de modo análogo as etapas necessárias ao desenvolvimento de códigos que visam à exploração do sistema, portanto, independentemente, aplicativos que contenham essa falha serão explorados de maneira semelhante ao exemplo contido neste projeto.

Os estudos direcionados à exploração de softwares comerciais e à aplicação de novos métodos de exploração exemplificam futuras propostas de trabalho, assim como o desenvolvimento de plugins no framework relacionados à automatização de um sistema de testes podem vir a ser publicações futuras.

Outra proposta seria o desenvolvimento de um scanner de vulnerabilidades que identificasse e atacasse alvos predeterminados, podendo assim utilizar o MSF como suporte a prática.

REFERÊNCIAS

ALMEIDA, A. R. Funcionamento de Exploits, 2003. Disponivel em: <http://www.firewalls.com.br/files/alexisExploit.pdf>. Acesso em: 13 julho 2009.

AMORIM, A. Forúm Metasploit-br, 30 Setembro 2008. Disponivel em: <http://www.metasploit-br.org/>. Acesso em: 25 junho 2009.

ARANHA, D. D. F. Tomando o Controle de Programas Vulneráveis a Buffer Overflow, fevereiro 2003. Disponivel em: <http://www.cic.unb.br/docentes/pedro/sd.php>. Acesso em: 22 junho 2009.

BIRCK, F. A. Utilizando um debugger - OllyDbg, 2008 Maio 22. Disponivel em: <http://www.guiadohardware.net/comunidade/utilizando-debugger/785195/>. Acesso em: 22 junho 2009.

HOGLUND, G.; MCGRAW, G. Como Quebrar Códigos. São Paulo: Pearson, 2006.

MANZANO, J. N. G. Fundamentos em Programação Assembly. São Paulo: Érica, 2007.

MAYNOR, D.; MOOKHEY, K. K. Metasploit Toolkit. Burlington: Syngress, 2007.

METASPLOIT PROJECT. Metasploit 3.0 Developer's Guide, 2006. Disponivel em: <http://www.metasploit.com/documents/developers_guide.pdf>. Acesso em: 23 junho 2009.

METASPLOIT PROJECT. Metasploit Framework User Guide, 2006. Disponivel em: <http://www.metasploit.com/documents/users_guide.pdf>. Acesso em: 14 junho 2009.

METASPLOIT PROJECT. Metasploit's Meterpreter, 2006. Disponivel em: <http://www.metasploit.com/documents/meterpreter.pdf>. Acesso em: 23 junho 2009.

MOOKHEY, K. K.; SINGH, P. Metasploit Framework, 12 jul. 2004. Disponivel em: <http://www.securityfocus.com/infocus/1789>. Acesso em: 13 Maio 2009.

OSBORNE, A. Microprocessadores. São Paulo: McGraw-Hill, 1984.

PELAGALLI, R. Writing exploits for Metasploit 3.0, 24 janeiro 2008. Disponivel em: <http://redstack.net/blog/index.php/2008/01/24/writing-exploits-for-metasploit-30.html>. Acesso em: 23 junho 2009.

SIQUEIRA, D. D. D. Explorando Stack Overflow no Windows, 8 Maio 2009. Disponivel em: <http://www.milw0rm.com/papers/328>. Acesso em: 13 junho 2009.

VIEIRA, L. Metasploit Framework, 11 Novembro 2008. Disponivel em: <http://imasters.uol.com.br/artigo/10645/seguranca/metasploit_framework_-_parte_01/>. Acesso em: 25 Março 2009.