



CRIANDO REGRAS PARA O SNORT

Versão 0.2
Maio de 2008

SNOC - SECURITY NETWORK OPERATION CENTER

Glaudson Ocampos
glaudson@intruders.org.br

SUMÁRIO

1– INTRODUÇÃO	03
2 – CAPTURANDO TRÁFEGO NOCIVO.....	04
3 -MODELO DE CRIAÇÃO DE REGRAS.....	08
3.1 – Criação de Regras Básicas no Snort	08
4 – CRIANDO REGRAS PARA O SKYPE.....	13
4.1 – Inserindo Regra no SNOG	14
5 – CONCLUSÃO.....	14
6 – REFERÊNCIAS	14

Gerente do Projeto - Security

Sr. Glaudson Ocampos – Analista de Segurança
Telefone: (61) 3033-7728
glaudson@security.org.br

1 – Introdução

A Equipe do Intruders Tiger Team Security desenvolveu esse documento com o objetivo de fornecer informações sobre como criar regras específicas para o Snort, usado na versão pública do SNOC. Temos grande experiência em criar regras de detecção avançadas para clientes que executam aplicativos específicos.

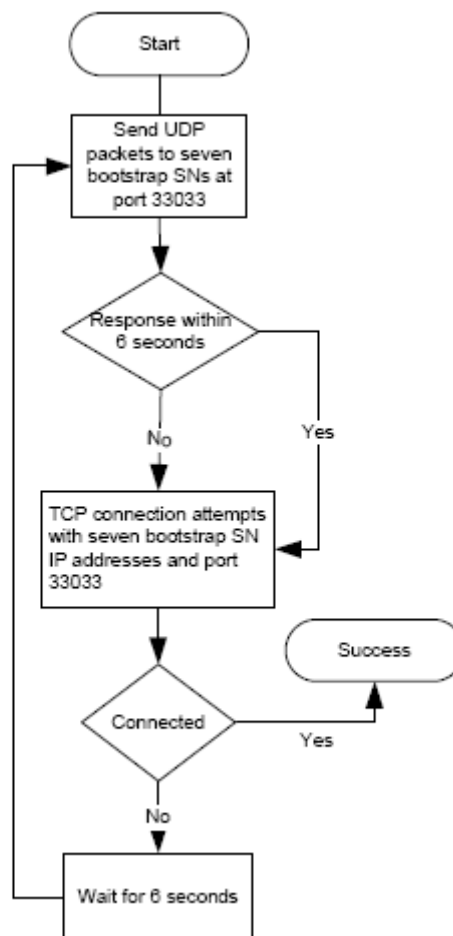
Nesse documento vamos descrever como criar uma regra para detectar o uso do Skype em uma rede Interna. O Skype tem sido usado por algumas pessoas em sistemas onde há dificuldade em barrar o protocolo. Muitas pessoas sabendo disso, utilizam o Skype como ferramenta de bate-papo semelhante a outras em horário e local de trabalho.

2 – Capturando Tráfego Nocivo

Para captura e análise de tráfego, nós utilizamos uma ferramenta de sniffing que possa nos fornecer informações em tempo real e de modo mais transparente sobre os pacotes capturados. Optamos por usar o Wireshark[4] em ambiente Windows para captura dos dados enviados e recebidos pelo Skype, pois se trata de uma ferramenta poderosa de monitoramento de pacotes.

Uma análise detalhada do protocolo usado pelo Skype pode ser vista no documento “**An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol**” - de Salman A. Baset e Henning G. Schulzrinne.

No entanto, o seguinte diagrama retirado do documento citado nos fornece informações úteis na captura do tráfego:

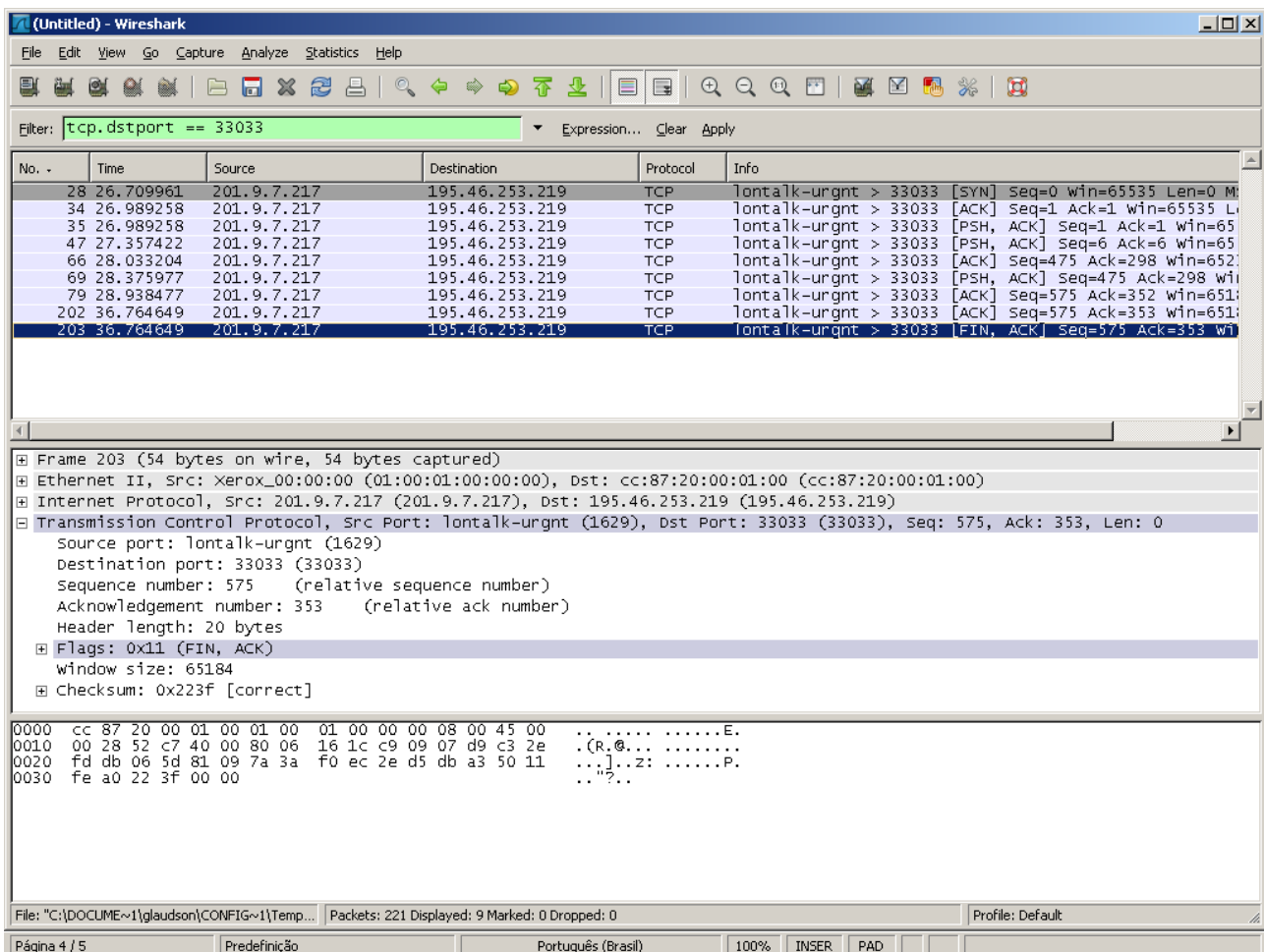


(Figura 1 – Handshake do Skype)

A figura ilustra bem uma espécie de “handshake” do Skype entre o cliente e os SN(Super Nodes em modelos P2P).

Para que essa conexão seja efetivada, o Skype tenta inicialmente enviar pacotes UDP na porta 33033 para 7 bootstrap SN. Capturando múltiplas conexões a essa porta de um único IP local para múltiplos(no caso 7) Ips remotos pode servir como regra de detecção de conexão ao Skype.

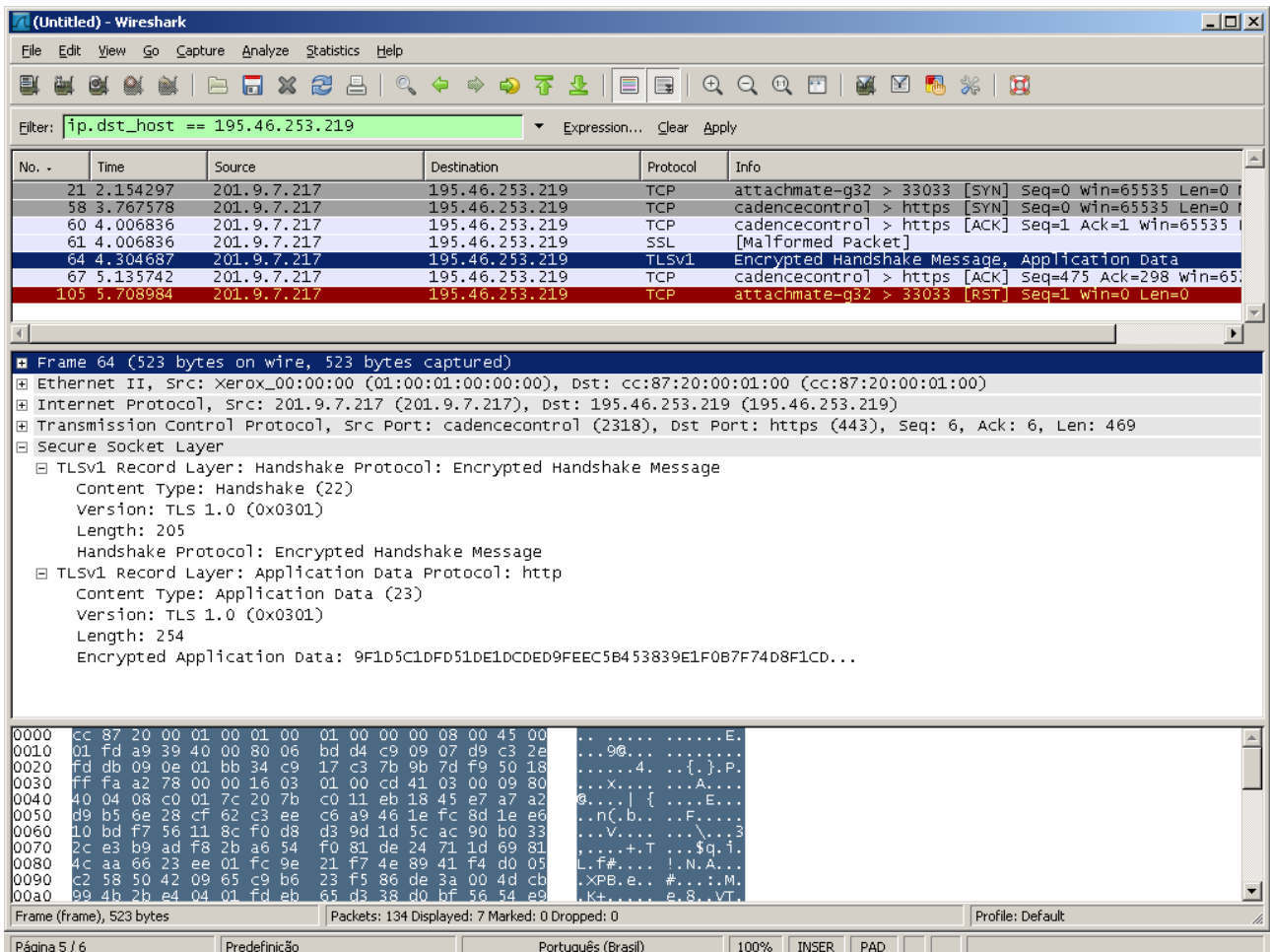
A figura abaixo demonstra a captura de dados enviados para uma porta 33033(TCP) usada por um bootstrap SN do Skype:



(Figura 2 – Wireshark demonstrando conexão na 33033/tcp pelo Skype)

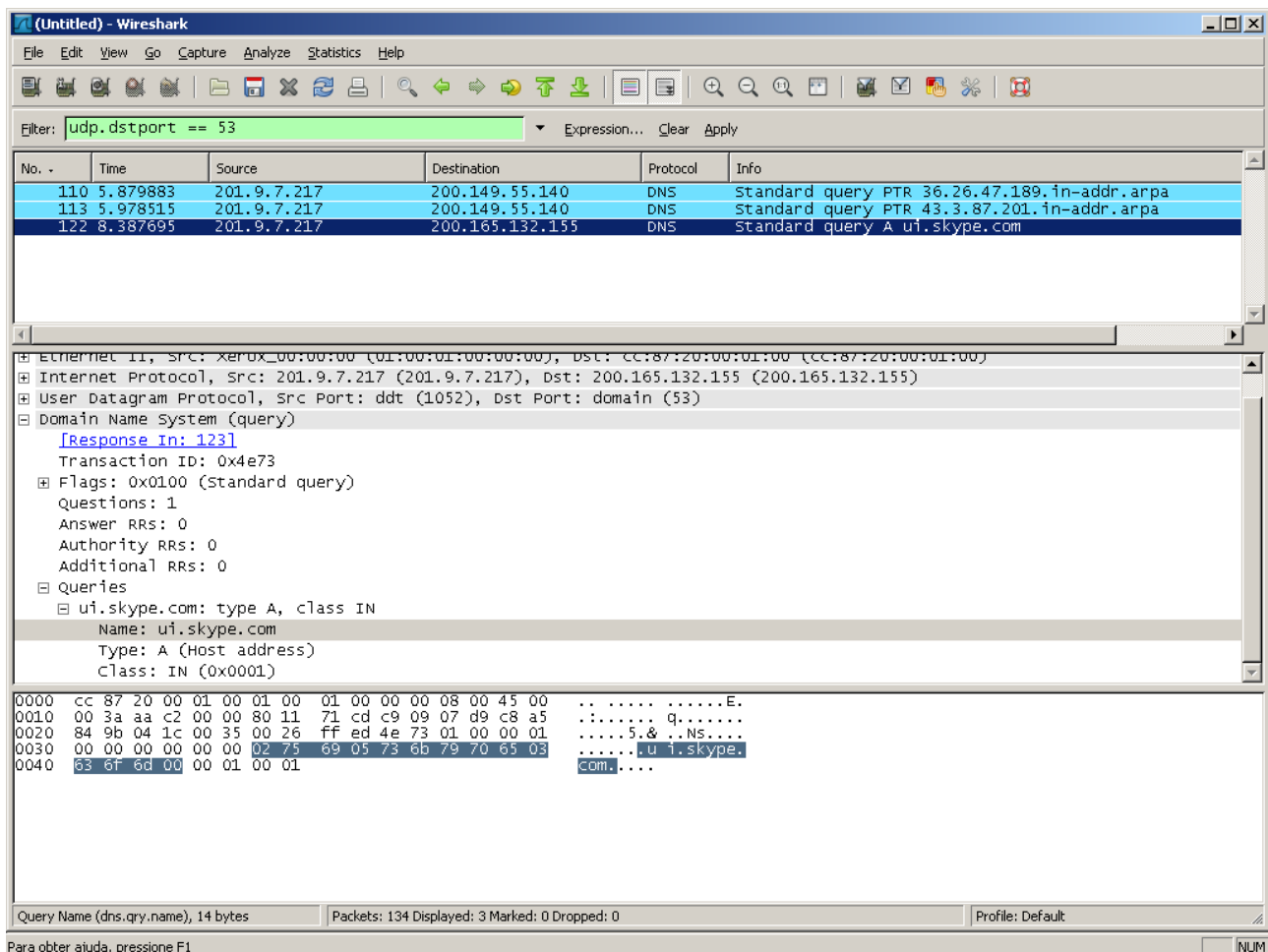
A porta de origem é aleatória, logo não se deve usá-la em parâmetro adicional para evitar falso-positivos. No entanto há pelo menos mais um ponto de captura em que podemos setar corretamente o conteúdo do pacote e detectar com mais exatidão uma conexão ao Skype.

Após a conexão na porta 33033/tcp o cliente do Skype então conecta na porta TCP/443(DNS) para envio das credencias e processo de login. Então, poderíamos analisar o envio desses pacotes para o IP previamente capturado e nos certificar que um processo de login do Skype está em andamento, no entanto isso também não é confiável, mas pode ser usado em conjunto na correlação dos logs. Abaixo tela demonstrando isso:



(Figura 3 – Cliente do Skype se conectando em porta SSL para processo de login)

Essa informação é interessante e pode ser inserida num alerta, mas a análise abaixo é mais interessante. Ela fornece evidência de que uma consulta DNS a um subdomínio do Skype foi realizada. Se fizermos uma correlação de log por analisar o IP origem que está se conectando na porta tcp/33033 e na porta udp/53 executando consulta a um subdomínio do Skype, chegaremos a conclusão de que se trata de uma conexão oriunda de um cliente Skype.



(Figura 4 – Consulta DNS ao domínio .skype.com)

Com base nessas informações, podemos então criar as regras do Snort para detecção.

3 – Modelo de Criação de Regras

Antes de criarmos a regra para a detecção da conexão do Skype e processo de login, precisamos compreender a criação de regras básicas no snort e ver qual o melhor modelo para ser aplicado nesse caso. Se desejar obter maiores informações sobre como escrever regras para o snort, recomendo acessar os sites abaixo:

http://www.snort.org/docs/writing_rules/chap2.html
http://www.ussrback.com/docs/papers/IDS/snort_rules.htm

No entanto repasso informações conclusivas no item abaixo.

3.1 – Criação Básica de Regras do Snort

A Criação de Regras no Snort obedece a um formato ou modelo pré-estabelecido que será enviado para uma espécie de analisador léxico e gerará instruções ao programa em execução.

Na instalação padrão do snort, os programas vêm acompanhados de vários arquivos de regras que se encontram no diretório “rules”. Estes arquivos possuem regras para diversos tipos de ataques, que podem perfeitamente serem adaptadas para a rede a ser protegida. Um pacote de regras também pode ser baixado em:

<http://www.snort.org/pub-bin/downloads.cgi>

O primeiro passo para a criação de regras é setar as variáveis de rede. Geralmente são duas variáveis que são mais usadas por Analistas que manuseiam o Snort:

`$EXTERNAL_NET` -> Refere-se a rede externa, diferente da rede a ser protegida. Isso inclui a Internet ou outras redes quaisquer que não a rede a ser protegida.

`$HOME_NET` -> É a Rede a ser protegida.

Como nós setamos estas variáveis ambientes?

Existem várias formas para fazermos isso, obedecendo a nomenclatura entendida pelo Snort. Podemos definir da seguinte forma:

```
var HOME_NET 192.168.0.1/0
```

```
var EXTERNAL_NET !$HOME_NET
```

Acima, nós definimos o valor em uma classe de IP para `HOME_NET` e em seguida dizemos que tudo que for diferente dessa classe será considerado como `EXTERNAL_NET`. Podemos definir múltiplas redes:

```
var HOME_NET [192.168.40.0/24,192.168.41.0/24,10.14.0.0/16]  
var EXTERNAL_NET !$HOME_NET
```

O modelo para definirmos variáveis é o seguinte:

```
var NOME_DA_VARIAVEL VALOR
```

Onde NOME_DA_VARIAVEL e VALOR são definidos pelo usuário. Abaixo nós podemos ver um exemplo de definição de variável no Snort:

```
var SHELLCODE_PORTS 80
```

Note que atribuímos valor 80 a uma variável de nome SHELLCODE_PORTS. Assim, onde aparecer a string \$SHELLCODE_PORTS, o Snort entenderá como sendo “80”.

Dica: Se você tem poucos serviços ou utiliza máquinas potentes, ao invés de setar uma classe de IP na variável HOME_NET e definir EXTERNAL_NET como sendo IPs diferentes dos pertencentes a classe, recomendando utilizar para ambos o valor “any”, assim seu NIDS vai conseguir detectar ataques oriundos de qualquer máquina, inclusive de máquinas consideradas “confiáveis”. Essa é uma medida de segurança importante, pois muitas vezes as máquinas de confiança podem ter sido comprometidas ou mesmo um funcionário interno pode ocasionar ataques na Rede.

As regras no Snort também obedecem a um modelo. O modelo está descrito abaixo:

```
<tipo_de_alerta> <protocolo> <rede_origem> <porta_origem> ->  
<rede_destino> <porta_destino> (Cabeçalho da Regra; Opções; sid:X;...);
```

Um exemplo bem simples pode ser visto abaixo:

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access";sid:11)
```

Como podemos notar na regra acima, o modelo é obedecido. Temos o tipo do alerta(alert), seguido do protocolo(tcp), a rede origem(any), porta origem(any), o fluído dos pacotes(->), seguido da rede destino(192.168.1.0/24) e da porta destino(111). O cabeçalho da regra tem o conteúdo em binário (“|00 01 86 a5|”), indicando que se este conteúdo passar obedecendo os valores anteriores, a mensagem msg(“mountd access”) será exibida como alerta. Atualmente as regras devem ter um campo de “sid” com a ID utilizada para diferenciar regras, isso é também importante para uso na solução SNOOC.

Em cima deste simples modelo, nós podemos construir regras poderosíssimas para manusear o snort. Por exemplo, vejamos as possibilidades de manuseio dos campos de regras.

- Para tipo_de_alerta, nós temos:

1. **alert** – Gera um alerta usando um método selecionado e então loga o pacote;
2. **log** – loga o pacote;
3. **pass** – ignora o pacote;
4. **activate** – alerta e então ativa outra regra dinâmica;
5. **dynamic** – permanece inativa até ser ativado por uma regra activate, então atua como uma regra de log.

Em cima das opções acima, nós podemos interagir até mesmo na criação de regras dinâmicas. Isso é útil para detectar ataques que envolvem worms ou outras ferramentas de

automação de ataques.

- Para Protocolos, nós temos:

1. **tcp**;
2. **udp**;
3. **icmp**;

- Para Opções de Regras, temos quatro categorias principais:

1. **meta-data** -> Essas opções provêm informação sobre a regra, mas não tem qualquer efeito durante a detecção;

2. **payload** -> Todas essas opções procuram por dados dentro do payload(seção DATA) do pacote e podem ser inter-relacionadas;

3. **non-payload** -> Essas opções procuram por dados fora do payload;

4. **post-detection** -> Essas opções são regras específicas que acontecem após uma regra ter sido detectada.

Em cima de cada categoria, nós temos as opções propriamente ditas que são manuseadas pelo Snort. Veremos a descrição de algumas dessas opções, mas para uma lista mais detalhada, consulte o manual do Snort e os links repassados no item acima.

1 - Opções de Regra de Meta-Data

- **msg**

A opção msg informa a ferramenta de log e alerta qual a mensagem que deve ser imprimida quando o alerta for dado.

Formato:

msg "<mensagem texto aqui>";

- **reference**

A opção reference permite as regras incluírem referências externas ao ataque. Essas referências possuem maiores informações sobre o ataque. Atualmente o Snort suporta apenas alguns sistemas específicos com URLs únicas. Abaixo uma lista deles:

Sistema	PrefixoURL
bugtraq	http://www.securityfocus.com/bid/
cve	http://cve.mitre.org/cgi-bin/cvename.cgi?name=
nessus	http://cgi.nessus.org/plugins/dump.php3?id=
arachnids (inativo)	http://www.whitehats.com/info/IDS
mcafee	http://vil.nai.com/vil/dispVirus.asp?virus k=

Formato:

reference: <id system>,<id>; [reference: <id system>,<id>;]

Vejam os alguns exemplos abaixo:

```
alert tcp any any -> any 7070 (msg:"IDS411/dos-realaudio"; \
flags:AP; content:"|fff4 fffd 06|"; reference:arachnids,IDS411;)
```

```
alert tcp any any -> any 21 (msg:"IDS287/ftp-wuftp260-venglin-linux"; \
flags:AP; content:"|31c031db 31c9b046 cd80 31c031db|"; \
reference:arachnids,IDS287; reference:bugtraq,1387; \
reference:cve,CAN-2000-1574;)
```

- sid

A opção sid é usada unicamente para identificar regras do Snort. Esta informação permite aos plugins que interagem com o snort identificar regras facilmente. Esta opção deve ser usada com a opção “rev”, que veremos mais abaixo.

Formato:

```
sid <snort_rule_id>
```

Deve-se atentar a numeração de id que precisa ser obedecida para se formar um padrão. Ela é a seguinte:

- < 100 – Reservada para uso Futuro;
- 100 – 1000000 – Regras incluídas com a distribuição do Snort;
- > 1000000 – Regras usadas localmente;

Exemplo:

```
alert tcp any any -> any 80 (content:"BOB"; sid:1000983; rev:1;)
```

- rev

Esta opção é usada unicamente para identificar revisões em regras do Snort. Revisões permitem demonstrar uma melhora na escrita das assinaturas de ataques. Esta opção deve ser usada em conjunto com a opção “sid”.

Formato:

```
rev <inteiro da revisão>
```

Exemplo:

```
alert tcp any any -> any 80 (content:"BOB"; sid:1000983; rev:1;)
```

- classtype

Serve para orientar a categoria de ataque que se está sendo detectado. Trabalha em cima de uma tabela com nome, descrição e prioridade. O usuário pode então especificar qual a prioridade que determinado tipo de ataque tem ao ser detectado.

Formato:

classtype: <nome da classe>;

As classificações de regras são definidas num arquivos chamado classification.config. Este arquivo utiliza a seguinte sintaxe:

config classification: <nome da classe>,<descrição>,<prioridade>

Existem várias classes padrões que podem ser utilizadas ou alteradas pelo usuário.

Um exemplo de regra contendo esta opção segue abaixo:

```
alert tcp any any -> any 25 (msg:"SMTP expn root"; flags:A+; content:"expn root"; nocase; classtype:attempted-recon;)
```

Podemos então notar que a classtype descrita acima se chama “attempted-recon”, que possui a seguinte entrada em classification.config:

```
config classification: attempted-recon,Attempted Information Leak,2
```

Então, podemos criar classes de prioridades e interagir com o NIDS para que responda aos ataques de acordo com a prioridade dos mesmos.

- priority

A opção priority emite um nível de alerta para a regra.

Formato:

priority: <inteiro para prioridade>;

Exemplo:

```
alert TCP any any -> any 80 (msg: "WEB-MISC phf attempt"; flags:A+;content: "/cgi-bin/phf"; priority:10;)
```

OBS: Essa opção é importante na definição de alertas críticos no SNOC (Veja manual de operação do Console do SNOC).

Então, essas são as opções básicas de criação de regras do Snort que podem ser usadas de forma poderosa por um analista de segurança capacitado.

4 – Criando regras para o Skype

As informações repassadas nos itens anteriores nos fornece informações precisas na criação de regras para detecção do Skype.

Basicamente, criaremos duas regras capazes de serem visualizadas no Console do SNOOC com prioridade alta de modo que fica fácil descobrir quais as máquinas internas estão utilizando clientes do Skype.

A primeira regra detecta uma conexão na porta tcp/33033:

```
alert tcp any any -> any 33033 (flags: PA; sid:1234512; priority:9;
msg:"Tentativa de Conexao ao Skype");
```

Essa regra detecta qualquer tentativa de conexão a porta 33033 que esteja enviando os flag TCP(PUSH e ACK). Isso evita gerar muitos alertas, mas pode-se setar outro flag TCP como SYN ou FIN no lugar de PA, caso esteja ocorrendo bypass.

A segunda regra segue abaixo:

```
alert udp any any -> any 53 (msg:"Consulta DNS ao Skype"; priority:9; content:
"|05|skype"; depth: 50; sid:1234513; rev:1;)
```

A regra acima detecta o envio de uma consulta DNS a um subdomínio do Skype.

A tela abaixo demonstra a execução de teste prévio usando Snort com apenas essas regras e exibindo alertas no console:



```
==== Initialization Complete ====
o''~)~
''''
-*> Snort! <*-
Version 2.8.1-ODBC-MySQL-FlexRESP-WIN32 (Build 28)
By Martin Roesch & The Snort Team: http://www.snort.org/team.html
(C) Copyright 1998-2008 Sourcefire Inc., et al.
Using PCRE version: 7.4 2007-09-21

Not Using PCAP_FRAMES
05/04-21:18:28.276367  [**] [1:1234512:0] Tentativa de Conexao ao Skype [**] [Pr
iority: 9] <TCP> 201.9.7.217:2853 -> 193.88.6.13:33033
05/04-21:18:28.569336  [**] [1:1234512:0] Tentativa de Conexao ao Skype [**] [Pr
iority: 9] <TCP> 201.9.7.217:2853 -> 193.88.6.13:33033
05/04-21:18:32.870117  [**] [1:1234513:1] Consulta DNS ao Skype [**] [Priority:
9] <UDP> 201.9.7.217:1052 -> 200.165.132.155:53
```

(Figura 5 – Snort Detectando Conexão ao Skype)

Como podemos perceber, a detecção é eficaz e consegue ser bem sucedida através da correlação entre as duas conexões, porta tcp/33033 e porta udp 53 com consulta a skype.com.

4.1 – Inserindo Regra no SNOC

O snoc pode manusear a regra inserida diretamente no sensor após a detecção e envio de alertas baseados na regra específica. Para inserir no SNOC basta inserir no arquivo ativo e correspondente em /etc/snort/rules/, onde o nome do arquivo pode ser específico do cliente ou mesmo no arquivo padrão p2p.rules. Em seguida reinicie o sensor snoc:

```
# /etc/init.d/snoc stop
```

```
# /etc/init.d/snoc start
```

Feito isso, automaticamente a regra irá ser inserida na base de dados do Mysql do servidor/console assim que o primeiro alerta for detectado.

5 – Conclusão

A criação de regras para o snort é bem simples e pode ser usada por um bom analista de segurança para a criação de regras robustas capazes de extrair o máximo de segurança que um NIDS pode oferecer.

Nesse documento abordamos a criação de uma simples regra capaz de detectar conexões de um cliente Skype, comumente usado em sistemas de VOIP e utilizando o conceito de P2P. Esperamos que esse documento sirva como fonte de informações na criação de regras para o SNOC capazes de proporcionar uma segurança avançada.

Glaudson Ocampos
Analista de Segurança
Intruders Tiger Team Security

6 – Referências

- 1 – SNOC – <http://www.security.org.br/>
<http://security-snoc.sourceforge.net/>
- 2 – Snort – <http://www.snort.org/>
- 3 – Break NIDS - Intruders Tiger Team Security – <http://www.intruders.org.br/>
- 4 – Wireshark – <http://www.wireshark.org/>